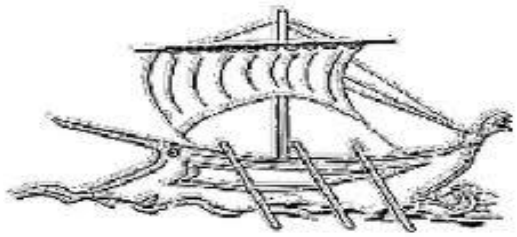
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ



Τ.Ε.Ι ΠΕΙΡΑΙΑ

"ΑΠΟΜΑΚΡΥΣΜΕΝΟΣ ΕΛΕΓΧΟΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΡΗΣΗΣ ΜΕΣΩ ΤΟΥ ΠΡΩΤΟΚΟΛΛΟΥ TCP/IP"

ΓΕΩΡΓΙΟΥ-ΜΕΪΜΑΡΗΣ ΝΙΚΟΣ

ΠΥΡΟΜΑΛΗΣ ΔΗΜΗΤΡΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΣΕΠΤΕΜΒΡΙΟΣ 2013

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ
Π. Ραλλη & Θηβών 250, 12244 Αιγάλαιω , Αθήνα – Ελλάδα Τηλ.
210-5381488

ΠΕΡΙΕΧΟΜΕΝΑ

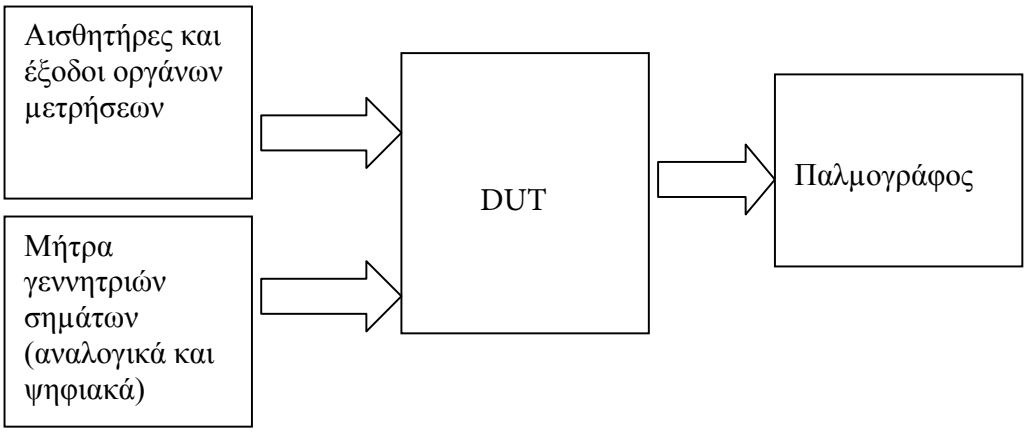
Κεφάλαιο 1

Εισαγωγή

Παγκοσμίως τα εικονικά εργαστήρια αποτελούν ένα βασικό εργαλείο στο χώρο των ηλεκτρονικών μετρήσεων και της επεξεργασίας δεδομένων. Εφαρμογές τους μπορούμε να συναντήσουμε τόσο στο χώρο της βιομηχανίας όσο και στο χώρο της εκπαίδευσης. Στη βιομηχανία χρησιμοποιούνται για την συλλογή και ανάλυση πληροφοριών από απομακρυσμένα μέρη με σημαντικά οφέλη στην εξοικονόμηση χρημάτων, ανθρωπίνου δυναμικού και χρόνου. Στην εκπαίδευση το εικονικό εργαστήριο μπορεί να αποτελέσει μια οριστική λύση στο όλο και αυξανόμενο πρόβλημα του συνωστισμού στα φυσικά εργαστήρια μιας και η δημιουργία μεγάλων και συγχρόνων εργαστηρίων που να καλύπτουν πλήρως τις ανάγκες είναι εξαιρετικά δαπανηρή.

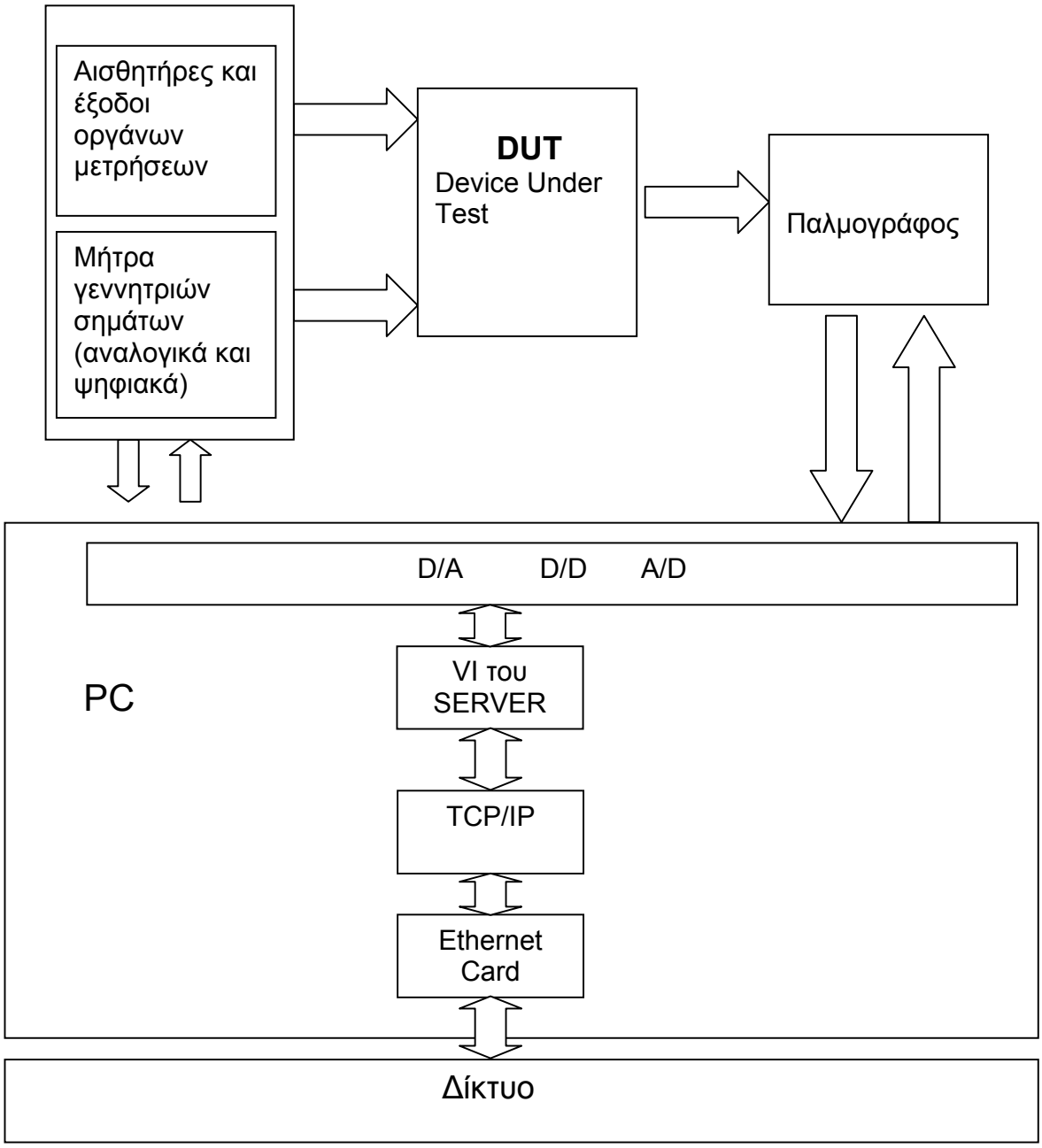
LabVIEW είναι το εμπορικό όνομα και η συντομογραφία του όρου **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. Πρόκειται για μια πλατφόρμα η οποία δίνει τη δυνατότητα στο χρήστη να δημιουργήσει προγράμματα όπως και σε πλατφόρμες της γλώσσας C ή της BASIC. Η μεγάλη διαφορά όμως είναι ότι ενώ άλλες πλατφόρμες χρησιμοποιούν γλώσσες προγραμματισμού που αναπτύσσουν τα προγράμματα τους με κείμενα, το LabVIEW χρησιμοποιεί μια γραφική γλώσσα προγραμματισμού η οποία ονομάζεται G. Η ορολογία , τα εικονίδια και η γενικότερη φιλοσοφία που χρησιμοποιεί το LabVIEW και η γλώσσα G είναι οικεία σε επιστήμονες και μηχανικούς. Με αυτό τον τρόπο ο χρήστης δημιουργεί προγράμματα με μεγαλύτερη ευκολία χωρίς να παγιδεύεται σε μια πληθώρα συντακτικών λεπτομερειών. Έτσι είναι πιο φιλικό προς τον χρήστη και μπορεί ακόμα και κάποιος χωρίς καμία εμπειρία στον προγραμματισμό να μάθει εύκολα να το χρησιμοποιεί. Το LabVIEW περιέχει επίσης βιβλιοθήκες με πληθώρα έτοιμων εφαρμογών οι οποίες μπορούν να βοηθήσουν το χρήστη σε οποιαδήποτε προγραμματιστική εφαρμογή.

Για να γίνει κατανοητή η λειτουργία ενός εικονικού εργαστηρίου δίδεται παρακάτω μια σχηματική παράσταση του. Έστω ένα πείραμα παλμογράφου το οποίο θα αποτιμήσει τη λειτουργία της συσκευής υπό δοκιμή(DUT). Στο DUT συνδέονται αισθητήρες και έξοδοι οργάνων μετρήσεων καθώς και μια γεννήτρια σημάτων. Οι τελικές μετρήσεις λαμβάνονται από έναν παλμογράφο.

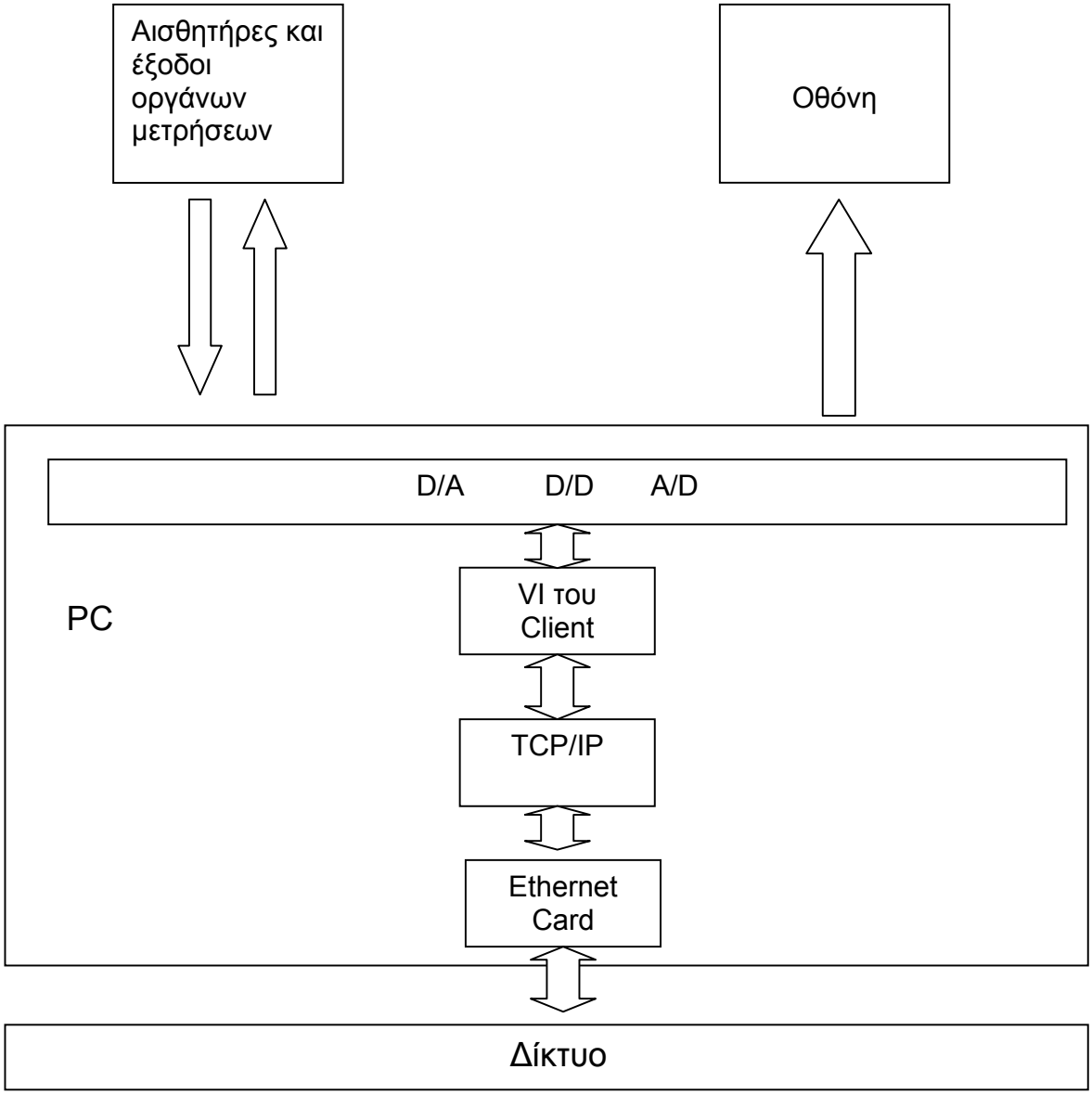


Εικόνα 1: Σχεδιάγραμμα Device Under Test

Το ίδιο πείραμα οργανώνεται κάτω από την εποπτεία ενός εικονικού εργαστηρίου. Το φυσικό εργαστήριο επικοινωνεί με το διακομιστή(server) μέσω ειδικών προγραμμάτων (drivers) επιτυγχάνοντας έτσι την ανταλλαγή πληροφοριών μεταξύ τους. Οι πληροφορίες μέσω του δικτύου φτάνουν στον πελάτη (client). Αυτός με τη σειρά του και μέσω του server έχει τη δυνατότητα να ελέγξει και να μεταβάλλει διάφορες παραμέτρους του συστήματος. Η αμφίδρομη επικοινωνία φυσικού εργαστηρίου-server και server-client υλοποιεί το εικονικό εργαστήριο. Παρακάτω δίδεται μια σχηματική παράσταση των προαναφερθέντων.

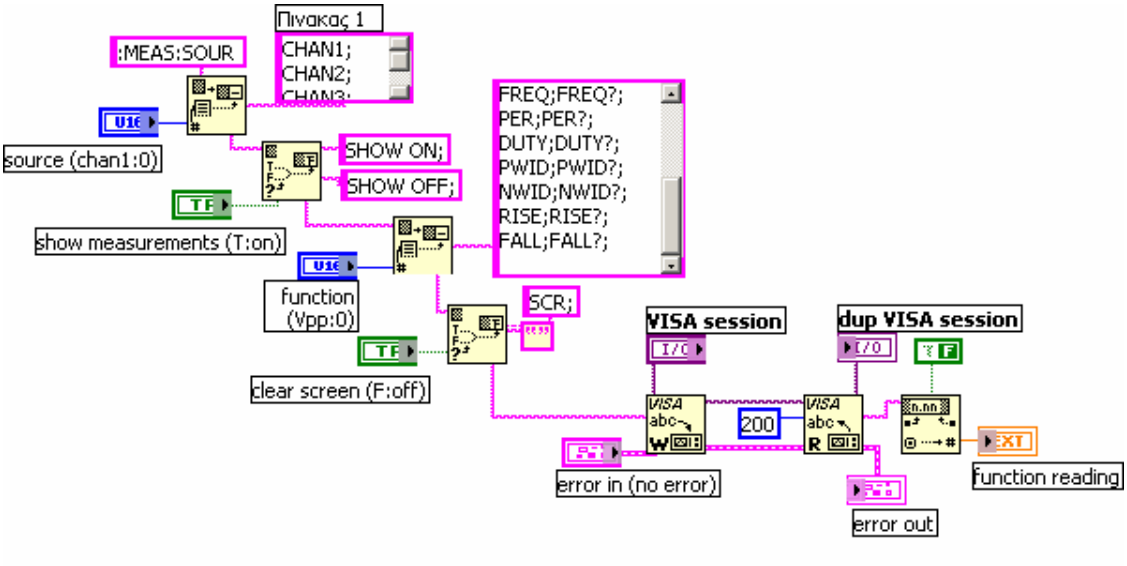


Εικόνα 2: Σχεδιάγραμμα επικοινωνίας εργαστηρίου-Server



Εικόνα 3: Σχεδιάγραμμα επικοινωνίας server-client

Ο παλμογράφος καταγράφει τις ζητούμενες μετρήσεις συνδέοντας στους ακροδέκτες του στο DUT. Το πρώτο πρόβλημα στην σχεδίαση του συστήματος, του εικονικού εργα-στηρίου, είναι η επικοινωνία του φυσικού οργάνου με το υπολογιστικό σύστημα. Το λογισμικό του LabVIEW επιλύει το πρόβλημα αυτό με την χρήση του διαδρόμου GBIP. Η Hewlett Packard ανέπτυξε τον διάδρομο GBIP (General Purpose Interface Bus) στα τέλη του 1960 για να επιτύχει επικοινωνία μεταξύ υπολογιστών και οργάνων. Ο διάδρομος είναι το μέσο για ανταλλαγή δεδομένων και πληροφοριών, που δημιουργήθηκε για τον έλεγχο και τις μετρήσεις σε όργανα. Ο διάδρομος GBIP είναι ένας 24 pins παράλληλος διάδρομος και έχει οκτώ γραμμές δεδομένων, πέντε γραμμές χειρισμού του διαδρόμου, τρεις γραμμές χειραψίας και οκτώ γραμμές γείωσης. Έτσι ολόκληρα bytes μεταφέρονται σειριακά με την μέθοδο χειραψίας. Επειδή τα δεδομένα στέλνονται κατά bytes, τα μηνύματα κωδικοποιούνται σε μορφή strings ASCII χαρακτήρων. Αυτό μπορεί να γίνει μόνο αν διαθέτουμε κάρτα GPIB και τους αντίστοιχους drivers. Υπάρχει ένας ελεγκτής, συνήθως ο υπολογιστής μας, που ελέγχει τον διάδρομο. Το όργανο μεταφοράς στέλνει τα δεδομένα στον διάδρομο, ενώ ο ελεγκτής καθορίζει ‘ομιλητή’ και ’ακροατήριο’ (talker και listeners). Οι γραμματοσειρές των δεδομένων μεταφέρονται έπειτα από τον ομιλητή στους ακροατές του διαδρόμου. Τα VIs του LabVIEW χειρίζονται αυτόματα τέτοιες διαδικασίες(όπως το να ορίζουν διευθύνσεις). Παρακάτω δίδεται ένα παράδειγμα της χρήσης του GBIP. Έστω ότι υπάρχει ένας παλμογράφος που συνδέεται μέσω μιας κάρτας GBIP με τον υπολογιστή μας. Η απεικόνιση των μετρήσεων ,στον υπολογιστή, που πραγματοποιεί γίνεται μέσω ενός driver. Το πρόγραμμα στο LabVIEW που υλοποιεί τον συγκεκριμένο driver απεικονίζεται παρακάτω.



Εικόνα 4: Απεικόνιση VI του driver του GBIP

Το συγκεκριμένο SubVI απεικονίζει στο Front Panel του εικονικού παλμογράφου μία τιμή που αντιστοιχεί στις επιλογές VPP, VAV, VRMS, VMAX, VMIN, VTOP, VBAS, FREQ. Οι τιμές αυτές μετρούνται από το φυσικό όργανο και με χρήση του driver μεταφέρονται στον υπολογιστή μας. Η δομή του προγράμματος είναι απλή και μπορεί να χρησιμοποιηθεί για όλες τις λειτουργίες του παλμογράφου (Time Base Config, Channel Config, Trigger, Config Acquisition, Digitize, Waveform) λαμβάνοντας υπόψη τις επιμέρους παραμέτρους. Στο συγκεκριμένο παράδειγμα αρχικά γίνεται επιλογή του καναλιού του οπίου θα διαβαστεί η μέτρηση. Τα κανάλια τα αντιστοιχούμε σε ακέραιους αριθμούς (το 0 για το κανάλι 1,το 1 για το κανάλι 2 κτλ) και εισάγουμε το επιθυμητό κανάλι στην συνάρτηση Pick Line. Η συνάρτηση αυτή προσθέτει στο string :MEAS:SOUR το string χαρακτήρων από τον πίνακα 1 ανάλογα της τιμής του (το επιθυμητό κανάλι). Εφαρμόζοντας την ίδια φιλοσοφία και στις επόμενες επιλογές, δημιουργείται ένα string που περιέχει της μεταβλητές source, show measurements, function και clear screen. Το επόμενο βήμα είναι να αποσταλεί το string στο φυσικό όργανο. Η συνάρτηση VISA Write μεταφέρει τα δεδομένα στην συσκευή που ορίζουμε μέσω της μεταβλητής VISA session . Όταν γίνει σύνδεση του παλμογράφου με τον υπολογιστεί, μέσω του LabVIEW, γίνεται η αναγνώριση της συσκευής όπου και ορίζουμε την διεύθυνση της στον διάδρομο GBIP (από 1 έως 30). Η μεταβλητή VISA session αναφέρεται σε αυτήν την διεύθυνση. Σε αυτό το σημείο έχουν αποσταλεί όλα τα αναγκαία δεδομένα και αναμένεται η αποστολή της μέτρησης, που επιλέχθηκε, από το φυσικό όργανο. Η ανάγνωση γίνεται με την συνάρτηση VISA Read οπότε γίνεται ανάκτηση της μέτρησης και απεικόνισης της στην οθόνη του υπολογιστή. Με αυτόν τον τρόπο επιλύεται το πρόβλημα της επικοινωνίας του παλμογράφου με τον υπολογιστή.

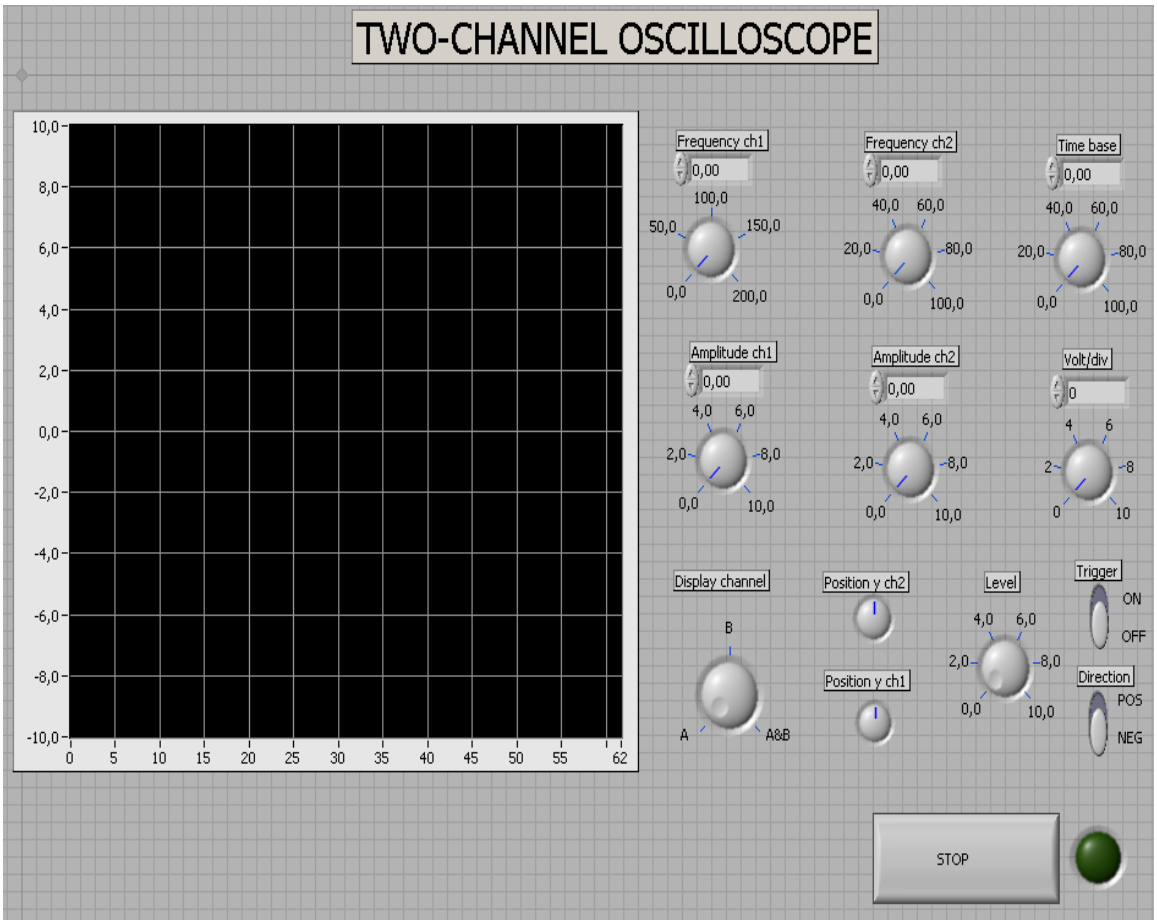
Στην περίπτωση της δικής μας εργασίας λόγω διαφόρων ανασταλτικών παραγόντων, τους οποίους θα αναφέρουμε αναλυτικά παρακάτω, η δημιουργία ενός εικονικού εργαστηρίου με πραγματικό παλμογράφο ήταν ανέφικτη. Ως εκ τούτου αναγκαστήκαμε να δημιουργήσουμε ένα εικονικό όργανο και να προχωρήσουμε κατόπιν στην ανάπτυξη του εικονικού εργαστηρίου. Στόχος ήταν η «κατασκευή» ενός παλμογράφου στο περιβάλλον του LabVIEW που να προσομοιώνει αρκετές από τις λειτουργίες ενός αντίστοιχου πραγματικού, έτσι ώστε να γίνει αντιληπτή μετέπειτα η χρησιμότητα του εικονικού εργαστηρίου.

Το αποτέλεσμα που προέκυψε ήταν ένα VI το οποίο ονομάσαμε oscilloscope.VI και διαθέτει τα ακόλουθα χαρακτηριστικά και λειτουργίες:

- Δύο κανάλια εισόδου, A και B
- Δυνατότητα επιλογής εμφάνισης καναλιού στην οθόνη (Display channel)
- Δυνατότητα μετακίνησης του προσλαμβάνοντος σήματος και των δύο καναλιών ως προς τον άξονα y (Position y ch1-Position y ch2)
- Βαθμονόμηση του άξονα y (Volt/div)
- Βαθμονόμηση του άξονα x (Time base)
- Σκανδαλισμό του προσλαμβάνοντος σήματος (Triggering)
- Επιλογή της τιμής του πλάτους του προσλαμβάνοντος σήματος από την οποία αρχίζει ο σκανδαλισμός (Level)
- Επιλογή της κατεύθυνσης(θετική-αρνητική) κατά την οποία αρχίζει ο σκανδαλισμός του σήματος (Direction)
- Stop button.

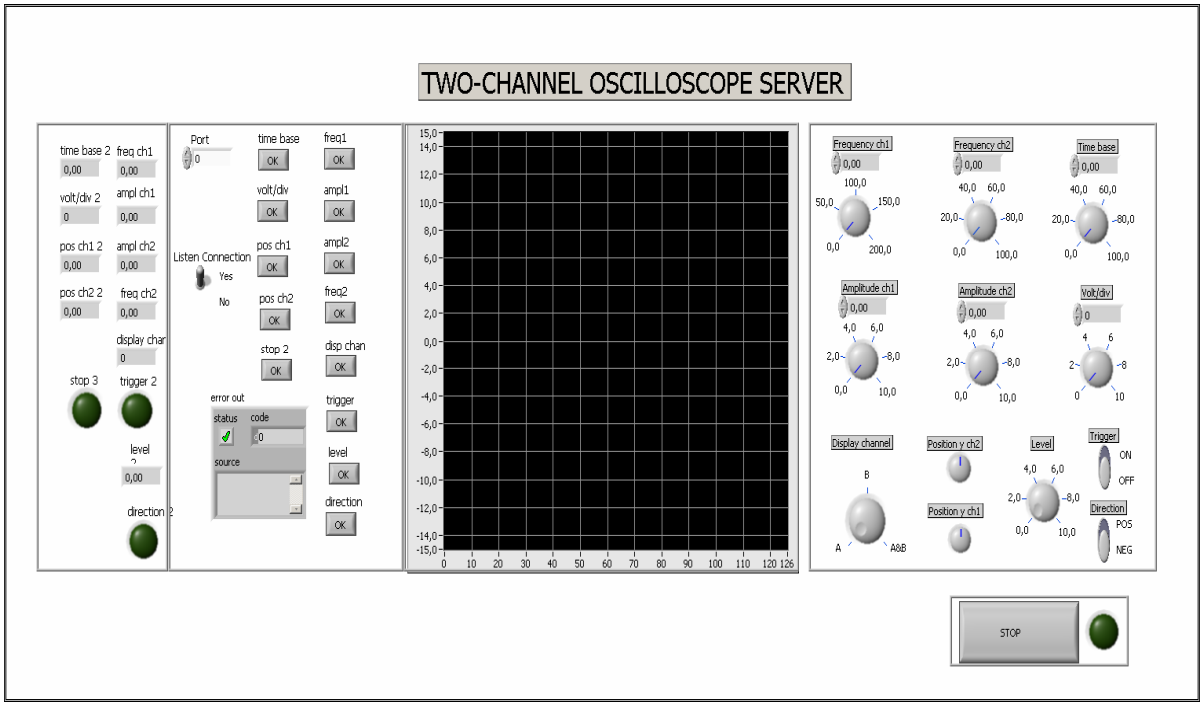
Λόγω έλλειψης εξωτερικής πηγής σήματος χρησιμοποιήσαμε γι' αυτό το σκοπό έτοιμες συναρτήσεις που μας παρέχει το LabVIEW. Συνδέσαμε λοιπόν στο κανάλι A του παλμογράφου μια γεννήτρια ημιτονικού σήματος και στο κανάλι B μια γεννήτρια παλμού. Τα controls του πλάτους και της συχνότητας τα συμπεριλάβαμε στο front panel του παλμογράφου έτσι ώστε ο χρήστης να μπορεί να πειραματιστεί με διαφορετικά σήματα. Ένα μειονέκτημα της χρήσης των έτοιμων γεννητριών σημάτων ήταν η αναντιστοιχία που παρατηρήθηκε στην συχνότητα του τεχνητού σήματος και στην απεικόνιση του στην οθόνη του παλμογράφου. Να σημειώσουμε ότι η συχνότητα αυτών των σημάτων δίνεται κανονικοποιημένη σε 7.8125E-3 cycles/sample και παρά τις προσπάθειές μας δε βρέθηκε.

Στη συνέχεια παρατίθεται το front panel του εικονικού παλμογράφου που δημιουργήσαμε.



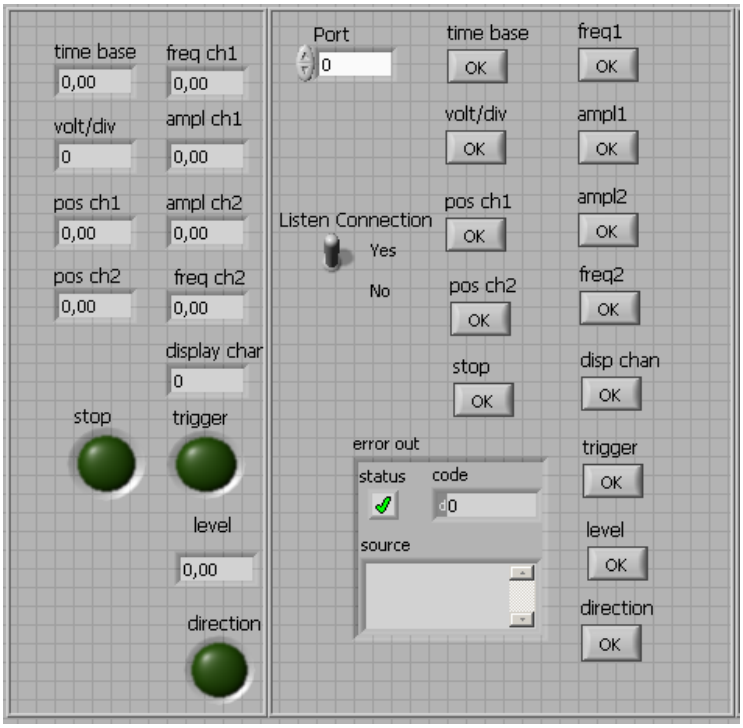
Εικόνα 5: Front Panel εικονικού παλμογράφου

Στο επόμενο βήμα της σχεδίασης του συστήματος υλοποιείται το πρόγραμμα του διακομιστή. Βασικός σκοπός του συγκεκριμένου προγράμματος είναι η δημιουργία σύνδεσης με έναν πελάτη μέσω δικτύου, με χρήση του πρωτοκόλλου TCP/IP. Το Front Panel του δίδεται παρακάτω.



Εικόνα 6: Front panel διακομιστή

Το Front Panel αποτελείται από τον παλμογράφο που αναπτύχθηκε παραπάνω και από ένα πλαίσιο με ενδείξεις και ελεγκτές που δίνουν την δυνατότητα, στον χρήστη του server, να ορίζει τα δικαιώματα προσπέλασης του πελάτη στο πρόγραμμα.



Εικόνα 7: Ενδείξεις και ρυθμίσεις του διακομιστή

Ε'ίδε ΰΌε ΌΌΰΰΰή γcεgΌΌΰgε ´ə˘Όΰιə˘Όγΰΰιə ιΰΰΌ3˘Ό g´Ύjcΰə ΰə˘Όχjəΰ Όήgΰ˘ΌΎΌΰιΌ ΰΰΌgΰ ΰə˘ΰΰΌΰə˘ΰgΰ. ¢ε ´ə˘Όΰιəε ΌΌΰΰgΰ ΰə˘ΰΰΌΰə˘ΰgΰ ΌΌΰ´ə˘djεΰjΌιəΎdΌ cjε3əəε ΰΰΌεΌε ΰΰΌ ΰΰιεΌγ3Ό´εg ΰəιΎ>ΌəεcΌjdΰΰεgΰΌεg ´ΰə˘ΎΰΌ. ¢ε ´ə˘Όΰιəε ΌΌΰΌεgg ΰə˘ΰə˘ΰΰγg ΰœεΰə ΰΌ3 ΰg3Ό-ΰήΰΌΰΰΰ ΰΰΌε3 cjεΰΰΌ Όεgΰ ΰəΌεcΌəΰΰΌε 3Ό ΰœεΰə ΰəεΌα ΌΌ ´jεΰ´γə˘ΌΰdgΰΰΌgΰ ΰəΎΰεjΌgΰ ΰə˘ΰΌεgjədΰgΰΌεg ´Όə˘ΌεcjΎΰεg. ΰəΌ3ΌΌ Ό´εjΌιΌ3ΌεΌΰjəΰΌΰι ΌεgΰΌ ə˘ΰΌεgjədΰgΰΌεg ´Όə˘ΌεcjΎΰεg ε cjεΰΰΌΰgΰΌεg ´ΰə˘ΎΰΌ, dΌ´jε´ΰə 3Ό γcΰə ΰə˘Ύjε>´εΰΰΰə ΌεgΌΌΰΰΰΌ ΌεgΌΌΰΰΰΌ ε cjεΰΰΌΰgΰΌεg Server, ´Όΰΰι3ΌΌgΌΌΰ ΌΌΌ Όə˘ΌΰΰΰΌΰΰΌ əεgΌ´ΎY (time base, volt/div, posch1, posch2, stop, freq1, freq2, ampl1, ampl2, dispchan, trigger, level, direction). Α´ΰΰdgΰ g´Ύjcΰə əΌə yΌΌgΰ ΰə˘ΰə˘əΌεgΰ ΌΌ εΌεΌΌΌdΌ port ΰΰΌε ε´εœε ΰœεΌΌΌə dιjΌΌΌγΰχ ΌΌgε´εdΌgΰdΌ ɔœΰə dΰιjΌΌΌdq ΌΌΰΌε3 ´ΰə˘ΎΰΌ. ¢ dιjΌΌdΌ ´jγ´ΰə 3Ό ΰœεΰə dq œΎəΌ əΌə ɔəΌ ΌεgΌ Server ήγgΰ əΌə ɔəΌ ΌεgΌ Client.z γΰχ ΌεgΌ ΰəΌ-əή´ΌΌq Listen Connection Ůəœγ>ΌΌΌə dq əΌΌΎΰΌΌΰdq ΌεgΌ server ΌΌΌΎΌ ήΌ ΌəΌΌ εΌgcdΌγ3dq ΰιjΌ-ΰΌΰdq ΌΌəΎεəε3 Client. ȳΌΌ3 ΰjΌΰəΌΌΌə ΰΌΌq dγΰdq Yes ο ΰəΌə´ΌΌgε Server ΰg3ΰΰœΰə3Ό´ΰjεΌγ3ΰə3γΌΰιjΌΰdq ΌΌClient ΰ3ι ΰΌΌq dγΰdq No ΌΌjΌΌΌœΰə ΌΌ3 ə˘ΰΌεgjədΌ ΌεgΌ. Β cjεΰΰΌΰgΰΌεg server γcΰə εgΰəΌΰΰΌε˘Y 3Ό ΰ9ΌΌəγΰΰə ΌγΰΰΰΌjεgΰ ΰ3γjəΌΰgΰ:

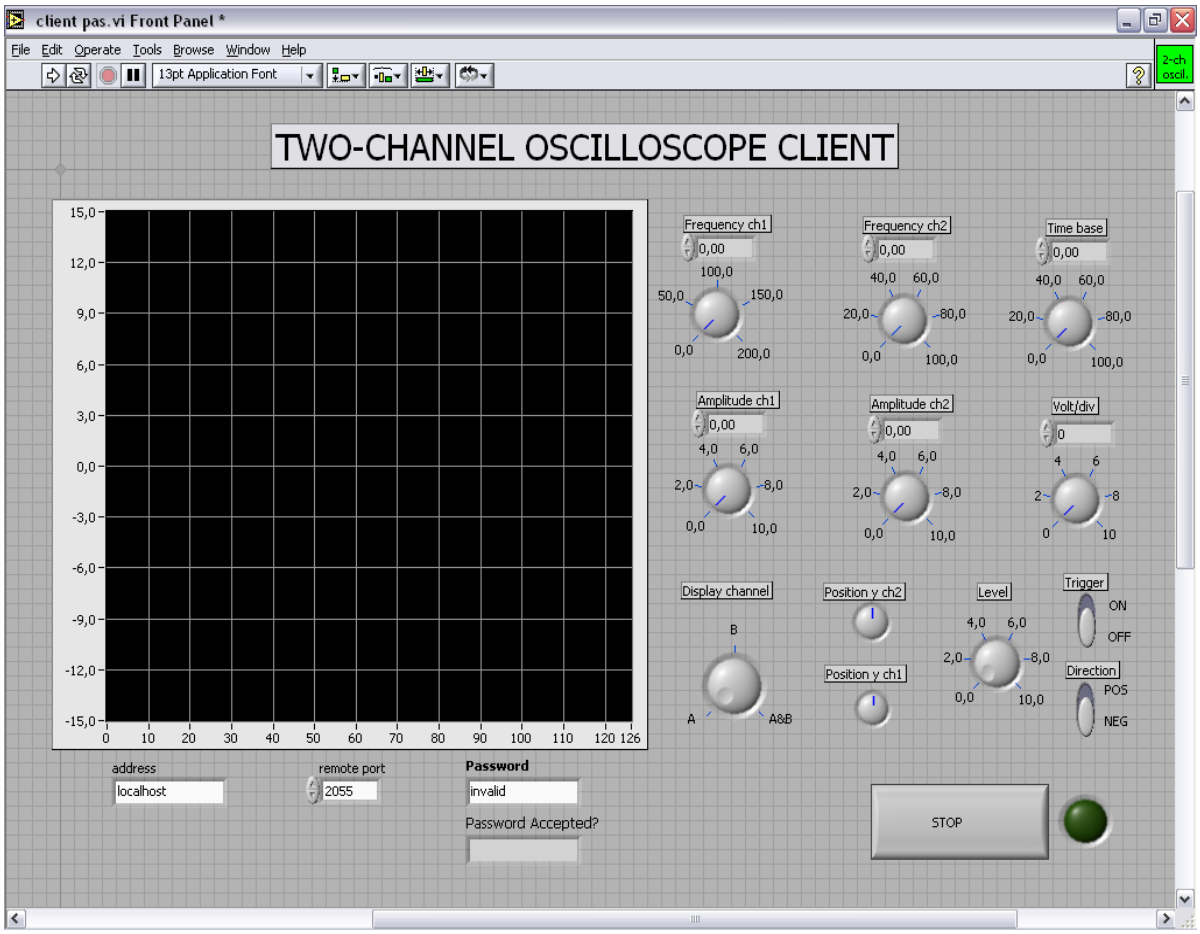
Α. Το πάτημα του κουμπιού run

Β. Την εισαγωγή του αριθμού της θύρας μέσω της οποίας θα γίνει η επικοινωνία

Γ. Επιλογή των λειτουργιών που θα έχει προσπέλαση ο χρήστης του client.

Δ. Επιλογή του Listen Connection.

Μετά τη δημιουργία του διακομιστή ακολούθησε η σχεδίαση του πελάτη. Το συγκεκριμένο πρόγραμμα έχει σκοπό την σύνδεση του με το διακομιστή μέσω του δικτύου και του πρωτοκόλλου TCP/IP και την ανταλλαγή δεδομένων μεταξύ τους. Για να συμβεί αυτό ο χρήστης του client πρέπει πρώτα να εισάγει τη IP address του server, τη θύρα επικοινωνίας και ένα αποδεκτό από τον διακομιστή password. Με την προσθήκη του password εξασφαλίζεται η ότι πρόσβαση στο server θα έχουν μόνο εξουσιοδοτημένοι χρήστες. Ακολουθεί η εικόνα του front panel του client.

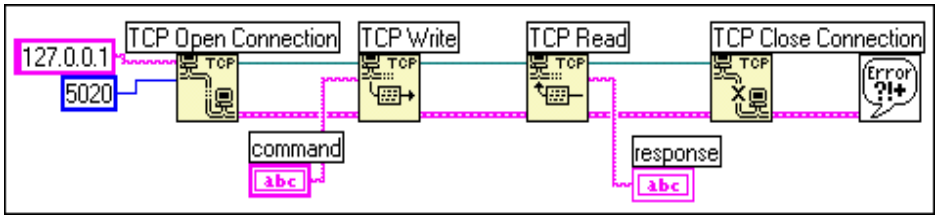


Εικόνα 8: Front panel χρήστη (client)

Όπως γίνεται εύκολα κατανοητό από την παρατήρηση της παραπάνω εικόνας το front panel του client περιλαμβάνει τους ίδιους ελεγκτές με τον εικονικό παλμογράφο. Ο χρήστης του client μέσω των ελεγκτών επιλέγει τις ρυθμίσεις του παλμογράφου οι οποίες μέσω δικτύου μεταβιβάζονται στον server, ο οποίος με τη σειρά του τις επεξεργάζεται και στέλνει πίσω μέσω της οθόνης του client την επιθυμητή κυματομορφή. Στο front panel υπάρχουν οι ελεγκτές και οι ενδείξεις που καθορίζουν το password, τη θύρα επικοινωνίας και την IP address.

Όπως προαναφέρθηκε η αμφίδρομη επικοινωνία του διακομιστή με τον πελάτη επιτυγχάνεται με τη χρήση του πρωτόκολλου TCP/IP που είναι το βασικό εργαλείο για δικτυακή επικοινωνία.

Με το TCP/IP επιτυγχάνεται επικοινωνία τόσο σε τοπικά δίκτυα όσο και μέσω του Internet. Το πλεονέκτημα του συγκεκριμένου πρωτόκολλου είναι ότι προσφέρει ένα απλό user interface και διασφαλίζει αξιόπιστη δικτυακή επικοινωνία. Με τη σύνδεση μέσω TCP/IP ο υπολογιστής μπορεί να λειτουργήσει είτε ως διακομιστής είτε ως πελάτης. Το παρακάτω block diagram απεικονίζει μια εφαρμογή πελάτη ο οποίος αρχίζει μια σύνδεση με έναν απομακρυσμένο διακομιστή με τη συνάρτηση TCP Open Connection. Ο διακομιστής περιμένει αιτήσεις επικοινωνίας και ανταποκρίνεται καταλλήλως.



Εικόνα 9: Block diagram σύνδεσης πελάτης στο διακομιστή

Στην εργασία περιγράφεται το περιβάλλον του LabVIEW (κεφ.2). Στη συνέχεια γίνεται η ανάλυση του εικονικού παλμογράφου που σχεδιάστηκε(κεφ.3). Ακολουθούν τα κεφάλαια 4 και 5 όπου παρουσιάζονται τα προγράμματα (VIs) που υλοποιούν το ζεύγος server-client. Τέλος στο προσάρτημα δίνονται οι έτοιμες συναρτήσεις του LabVIEW που χρησιμοποιήθηκαν στη δημιουργία του διακομιστή και του πελάτη.

Κεφάλαιο 2

Εισαγωγή στο περιβάλλον LABVIEW

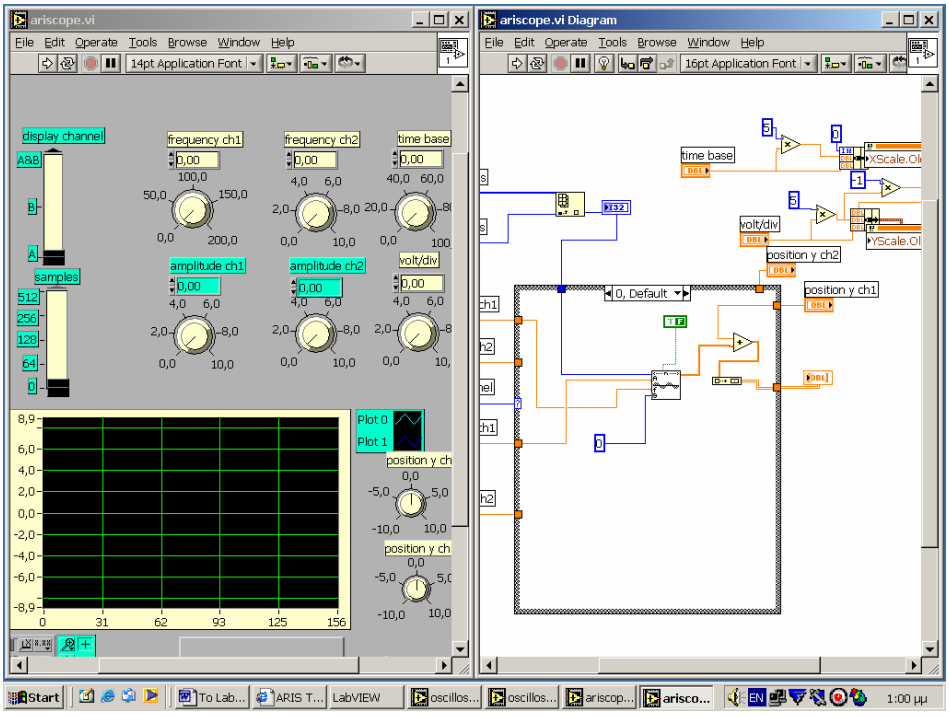
LabVIEW είναι το εμπορικό όνομα και η συντομογραφία του όρου **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. Πρόκειται για μια πλατφόρμα η οποία δίνει τη δυνατότητα στο χρήστη να δημιουργήσει προγράμματα όπως και σε πλατφόρμες της γλώσσας C ή της BASIC. Η μεγάλη διαφορά όμως είναι ότι ενώ άλλες πλατφόρμες χρησιμοποιούν γλώσσες προγραμματισμού που αναπτύσσουν τα προγράμματα τους με κείμενα, το LabVIEW χρησιμοποιεί μια γραφική γλώσσα προγραμματισμού η οποία ονομάζεται G. Η ορολογία , τα εικονίδια και η γενικότερη φιλοσοφία που χρησιμοποιεί το LabVIEW και η γλώσσα G είναι οικεία σε επιστήμονες και μηχανικούς. Με αυτό τον τρόπο ο χρήστης δημιουργεί προγράμματα με μεγαλύτερη ευκολία χωρίς να παγιδεύεται σε μια πληθώρα συντακτικών λεπτομερειών. Έτσι είναι πιο φιλικό προς τον χρήστη και μπορεί ακόμα και κάποιος χωρίς καμία εμπειρία στον προγραμματισμό να μάθει εύκολα να το χρησιμοποιεί. Το LabVIEW περιέχει επίσης βιβλιοθήκες με πληθώρα έτοιμων εφαρμογών οι οποίες μπορούν να βοηθήσουν το χρήστη σε οποιαδήποτε προγραμματιστική εφαρμογή. Τα προγράμματα του LabVIEW ονομάζονται virtual instruments (για συντομία VIs) και αυτό γιατί η εμφάνισή τους και η λειτουργία τους μιμούνται πραγματικά όργανα. Ένα VI αποτελείται από τρία κύρια μέρη:

- Το front panel που είναι ουσιαστικά το user interface. Μπορεί να περιέχει διακόπτες, οθόνες που να παρουσιάζουν γραφικές παραστάσεις, διάφορα όργανα επιλογής εισόδων και ένδειξης εξόδων. Είναι ουσιαστικά ένας συνδυασμός από controls και indicators. Ο χρήστης μπορεί να εισάγει δεδομένα χρησιμοποιώντας το πληκτρολόγιο ή το ποντίκι μέσω των controls και στη συνέχεια να δει τα αποτελέσματα που παράγονται από το πρόγραμμα στην οθόνη μέσω των indicators. Η εισαγωγή των controls και των indicators γίνεται μέσα από το Control Palette.

Μόλις τοποθετήσουμε ένα αντικείμενο στο front panel τότε μπορούμε να προσαρμόσουμε το μέγεθος, το σχήμα και τη θέση του. Είναι by default χρωματισμένο γκρι αλλά μας δίνεται η δυνατότητα να επιλέξουμε ένα χρώμα της αρεσκείας μας μέσω του tools palette όπως θα δούμε στη συνέχεια.

- Το block diagram είναι ουσιαστικά ο πηγαίος κώδικας του VI, και το εκτελέσιμο πρόγραμμα. Τα στοιχεία που το αποτελούν είναι εικονίδια τα οποία αντιπροσωπεύουν χαμηλότερης στάθμης VIs, έτοιμες συναρτήσεις του LabVIEW και δομές ελέγχου του προγράμματος. Ο χρήστης πρέπει να συνδέσει καλώδια για να ενώσει τα εικονίδια υποδεικνύοντας έτσι τη ροή των δεδομένων. Είναι by default λευκό.

- Το εικονίδιο και οι σύνδεσμοι του VI που του επιτρέπουν να ανταλλάσσει δεδομένα με άλλα VIs. Το εικονίδιο είναι τοποθετημένο στην πάνω-δεξιά γωνία του front panel και του block diagram. Το εικονίδιο εμφανίζεται επίσης στο block diagram όταν χρησιμοποιούμε το συγκεκριμένο VI ως subVI. Με διπλό click πάνω στο εικονίδιο του ενεργού VI (όχι ενός subVI), εμφανίζεται Icon Editor ο οποίος δίνει τη δυνατότητα στο χρήστη να διαμορφώσει το εικονίδιο όπως αυτός επιθυμεί.



Εικόνα 2.1. Το front panel καταλαμβάνει το αριστερό μισό της οθόνης και είναι χρωματισμένο γκρι ενώ το block diagram καταλαμβάνει το δεξί μισό της οθόνης και είναι λευκό.

2.1 Το περιβάλλον του LabVIEW

2.1.1 Pull-Down Menus

Το LabVIEW έχει δύο τύπους μενού, pull-down και pop-up, τα οποία ο χρήστης θα πρέπει να χρησιμοποιήσει εκτενώς στην ανάπτυξη των προγραμμάτων του. Στη συνέχεια θα παρουσιάσουμε αναλυτικά τα pull-down menus τα οποία και βρίσκονται στο menu bar του front panel και του block diagram. Το ίδιο δεν μπορούμε να κάνουμε για τα pop-up menus μιας και σχεδόν κάθε αντικείμενο έχει το δικό του μενού αποτελούμενο από διαφορετικές εντολές και επιλογές, και ως εκ τούτου η καταγραφή και η παρουσίαση τους γίνεται αδύνατη.

File Menu

Σε αυτό το μενού βρίσκουμε συνηθισμένες εντολές διαχείρισης αρχείων. Οι εντολές που περιλαμβάνονται είναι οι παρακάτω:

- **New VI** δημιουργεί ένα νέο VI.
- **New** εμφανίζει ένα dialog box το οποίο μπορεί να χρησιμοποιηθεί για τη δημιουργία διαφόρων στοιχείων του LabVIEW που βοηθούν στο χτίσιμο μιας εφαρμογής.
- **Open** εμφανίζει ένα dialog box από το οποίο μπορούμε να επιλέξουμε το αρχείο που θέλουμε να ανοίξουμε.
- **Close** κλείνει το τρέχον αρχείο αφού πρώτα ρωτήσει το χρήστη αν θέλει να αποθηκεύσει τυχούσες αλλαγές που έχουν γίνει.
- **Close All** κλείνει όλα τα ανοιχτά αρχεία.
- **Save** αποθηκεύει το τρέχον αρχείο. Αν το αρχείο αποθηκεύεται για πρώτη φορά τότε εμφανίζεται ένα dialog box το οποίο ζητάει από τον χρήστη να ονομάσει το αρχείο και να καθορίσει την θέση στην οποία επιθυμεί να το τοποθετήσει.
- **Save As** επιτρέπει στον χρήστη να αποθηκεύσει το τρέχον αρχείο με ένα διαφορετικό όνομα, προέκταση αρχείου ή σε διαφορετική θέση.
- **Save All** αποθηκεύει όλα τα ανοιχτά αρχεία.

- **Save with Options** εμφανίζει ένα dialog box το οποίο μπορεί να χρησιμοποιηθεί για να αποθηκευθεί ένα VI χρησιμοποιώντας διάφορες επιλογές.
- **Revert** επαναφέρει το αρχείο στην κατάσταση που βρισκόταν όταν είχε αποθηκευθεί για τελευταία φορά.
- **Page Setup** εμφανίζει ένα dialog box το οποίο μπορεί να χρησιμοποιηθεί είτε για να αλλάξουμε τις ρυθμίσεις του εκτυπωτή είτε για να διαμορφώσουμε την εικόνα του αρχείου.
- **Print** εμφανίζει ένα dialog box το οποίο δίνει στο χρήστη διάφορες επιλογές που αφορούν την εκτύπωση ενός αρχείου.
- **Print Window** εμφανίζει ένα dialog box το οποίο μπορεί να χρησιμοποιηθεί για την εκτύπωση του τρέχοντος front panel ή block diagram window.
- **VI Properties** εμφανίζει ένα dialog box που μπορεί να χρησιμοποιηθεί για να σεταριστούν κάποιες γενικές επιλογές και για σεταριστούν ή απλώς να ειδωθούν κάποιες επιλογές που αφορούν memory usage, documentation, revision history, security, window appearance, window size, execution και printing.
- **Recently Opened Files** επιτρέπει στο χρήστη να ανοίξει αρχεία τα οποία έχουν προσπελαστεί πρόσφατα.
- **Exit** κλείνει το LabVIEW αφού πρώτα ερωτηθεί ο χρήστης αν επιθυμεί να αποθηκεύσει αλλαγές που έχουν γίνει στα αρχεία που έχει ανοίξει.

Edit Menu

Περιέχει εντολές οι οποίες μας επιτρέπουν να ψάξουμε για LabVIEW αρχεία και να τροποποιήσουμε το περιεχόμενο τους. Οι εντολές που περιλαμβάνονται είναι οι παρακάτω:

- **Undo** ακυρώνει την τελευταία ενέργεια του χρήστη.
- **Redo** ακυρώνει την τελευταία Undo ενέργεια.
- **Cut** απομακρύνει το επιλεγμένο αντικείμενο και το αποθηκεύει στο clipboard.
- **Copy** αντιγράφει το επιλεγμένο αντικείμενο και το αποθηκεύει στο clipboard.
- **Paste** τοποθετεί τα αντικείμενα του clipboard στο ενεργό παράθυρο.
- **Clear** απομακρύνει το επιλεγμένο αντικείμενο χωρίς να το αποθηκεύει στο clipboard.

- **Find** εμφανίζει το Find dialog box το οποίο μπορεί να χρησιμοποιηθεί για την εύρεση VIs, functions, type definitions, text, front panel objects και terminals στο block diagram.
- **Show Search Results**
- **Customize Control** επιτρέπει στον χρήστη να τροποποιήσει το επιλεγμένο front panel control αντικείμενο και να αποθηκεύσει το αρχείο που προκύπτει με την .ctl επέκταση.
- **Scale Object With Panel** κάνει το επιλεγμένο αντικείμενο του front panel να αναπροσαρμόζει το μέγεθος του ανάλογα με το μέγεθος του front panel, κάθε φορά που αλλάζει το μέγεθος του front panel.
- **Set Tabbing Order** επιτρέπει στον χρήστη να επιλέξει την σειρά αρίθμησης των αντικειμένων του front panel.
- **Import Picture from File** εισάγει μια εικόνα στο VI του χρήστη.
- **Remove Broken Wires** διαγράφει όλα τα καλώδια που εμφανίζονται με μια διακεκομμένη γραμμή και συνεπώς είναι κακώς τοποθετημένα, από το ενεργό VI.
- **Create SubVI** δημιουργεί ένα νέο subVI από τα επιλεγμένα αντικείμενα.
- **Run-Time Menu** εμφανίζει το Menu Editor dialog box το οποίο μπορεί να αξιοποιήσει ο χρήστης για να δημιουργήσει και να επεξεργαστεί run-time menu (RTM) αρχεία και να τα συσχετίσει με ένα VI.

Operate Menu

Περιέχει εντολές οι οποίες χρησιμεύουν στον έλεγχο της λειτουργίας των VIs. Οι εντολές που περιλαμβάνονται είναι οι παρακάτω:

- **Run** εκτελεί το VI. Μπορεί ακόμα να χρησιμοποιηθεί το κουμπί **Run** που βρίσκεται στο toolbar.
- **Stop** σταματάει την εκτέλεση του VI πριν αυτή ολοκληρωθεί. Πρέπει να αποφεύγεται η χρήση του **Stop** για την έξοδο από ένα VI γιατί απορεί να αφήσει το σύστημα σε μια ασταθή κατάσταση. Πρέπει να χρησιμοποιείται ένας Boolean διακόπτης για αν σταματάει ένα VI που εκτελείται συνεχώς.
- **Suspend when Called** έχει σαν συνέπεια να ανασταλεί η εκτέλεση του VI όταν αυτό καλείται ως subVI.

- **Print at Completion** εκτυπώνει την εικόνα του front panel όταν ολοκληρωθεί η εκτέλεση του VI.
- **Log at Completion** εκτελεί data logging όταν ολοκληρώνεται η εκτέλεση του VI.
- **Data Logging** επιτρέπει στον χρήστη να αποκτήσει πρόσβαση στις παρακάτω data logging λειτουργίες **Log**, **Retrieve**, **Purge Data**, **Change Log File Binding** και **Clear Log File Binding**.
- **Make Current Values Default** αποθηκεύει τις τρέχουσες τιμές των controls και των σταθερών σαν τις default τιμές τους.
- **Reinitialize All to Default** επιστρέφει σε όλα τα controls και τις σταθερές στις default τιμές τους.
- **Change to Run Mode** θέτει το VI σε Run mode. Όταν αυτό βρίσκεται σε Run mode, η εντολή αλλάζει σε **Change to Edit Mode**.
- **Connect to Remote Panel** επιτρέπει στον χρήστη να συνδεθεί και να ελέγξει ένα front panel που βρίσκεται σε ένα απομακρυσμένο υπολογιστή.

Window Menu

Περιέχει εντολές οι οποίες επιτρέπουν στο χρήστη να διαμορφώσει την εμφάνιση των παραθύρων και των palettes. Ο χρήστης μπορεί επίσης να προσπελάσει το Error List window και να δει τα περιεχόμενα του clipboard. Οι εντολές που περιλαμβάνονται είναι οι παρακάτω:

- **Show Diagram/Show Panel** ενεργοποιεί το παράθυρο του block diagram και του front panel του VI, αντίστοιχα.
- **Show Controls Palette** εμφανίζει την **Controls** palette. Στο block diagram αυτή η εντολή αντικαθίσταται από την **Show Functions Palette** η οποία και εμφανίζει την **Functions** palette.
- **Show Tools Palette** εμφανίζει την **Tools** palette στην οθόνη.
- **Show Clipboard** εμφανίζει τα περιεχόμενα του clipboard.
- **Show Error List** δίνει στο χρήστη τη δυνατότητα να προσπελάσει το **Error List window**, στο οποίο υπάρχει η λίστα λαθών του τρέχοντος VI.
- **Tile Left and Right** χωρίζει την οθόνη καθέτως και ισομερώς τοποθετώντας το front panel στην αριστερή πλευρά της οθόνης και το block diagram στη δεξιά.

- **Tile Up and Down** χωρίζει την οθόνη οριζοντίως και ισομερώς τοποθετώντας το front panel στην πάνω πλευρά της οθόνης και το block diagram στο κάτω.

- **Full Size** το ενεργό παράθυρο (front panel ή block diagram) εξαπλώνεται έτσι ώστε να καταλάβει όλη την οθόνη του υπολογιστή.

Μια λίστα όλων των ανοιχτών παραθύρων εμφανίζεται στο τέλος του Window Menu. Επιλέγοντας ένα από τα παράθυρα που βρίσκονται στη λίστα αυτό ενεργοποιείται και εμφανίζεται στην οθόνη.

Tools Menu

Περιέχει εντολές οι οποίες χρησιμοποιούνται για τη διαμόρφωση του LabVIEW, των projects του χρήστη και των VIs. Οι εντολές που περιλαμβάνονται είναι οι παρακάτω:

- Windows **Measurement & Automation Explorer** δίνει πρόσβαση στον Measurement & Automation Explorer, τον οποίο μπορεί να χρησιμοποιήσει ο χρήστης για να διαμορφώσει το instruments and data acquisition hardware που είναι συνδεδεμένο στο σύστημα του.
- **Instrumentation** δίνει πρόσβαση στις **Instrument Driver Network**, **Import CVI Instrument Driver, Update VXI plug & play Drivers** εντολές.
- **Data Acquisition** δίνει πρόσβαση στο **DAQ Channel Viewer** και στο **DAQ Solution Wizard**.
- **Datalogging and Supervisory Control** δίνει πρόσβαση σε εργαλεία από το LabVIEW Datalogging and Supervisory Control module σε συστήματα που έχουν εγκατεστημένο τα συγκεκριμένα module.
- **Real-Time** δίνει πρόσβαση σε εργαλεία από το LabVIEW RT module σε συστήματα που έχουν εγκατεστημένο το συγκεκριμένο module.
- **IMAQ Vision** δίνει πρόσβαση σε εργαλεία από το IMAQ Vision module σε συστήματα που έχουν εγκατεστημένο το συγκεκριμένο module on.
- **Compare** δίνει πρόσβαση στις λειτουργίες Compare VIs, Show Differences, Compare VI Hierarchies, and Compare Files. Είναι διαθέσιμη μόνο στους χρήστες που διαθέτουν το Professional Development System του LabVIEW.

Source Code Control δίνει πρόσβαση στις source code control functions. Είναι διαθέσιμη μόνο στους χρήστες που διαθέτουν το Professional Development System του LabVIEW.

- **VI Revision History** εμφανίζει ένα παράθυρο μέσω του οποίου μπορεί ο χρήστης να καταγράφει τις αλλαγές κάνει στο τρέχον VI.
- **User Name** εμφανίζει ένα dialog box μέσω του οποίου ο χρήστης μπορεί να αλλάξει το LabVIEW user name του.
- **VI Library Manager** εμφανίζει ένα dialog box μέσω του οποίου ο χρήστης μπορεί να αντιγράψει, να αλλάξει το όνομα και να διαγράψει αρχεία που βρίσκονται στις VI libraries.
- **Edit VI Library** επιτρέπει στο χρήστη να διαμορφώσει τα περιεχόμενα της VI library ή ακόμα και να δημιουργήσει μια νέα.
- **Remote Panel Connection Manager** επιτρέπει στο χρήστη να παρακολουθεί όλο το client traffic προς τον server.
- **Web Publishing Tool** δίνει πρόσβαση στο Web Publishing Tool VI.
- **Advanced** δίνει πρόσβαση σε **Mass Compile**, VI Metrics, Profile VIs, **Export Strings**, **Import Strings**, Import ActiveX Controls, and **ActiveX Property Browser**. Use **Export Strings** and **Import Strings** to localize a VI.
- **Options** εμφανίζει ένα dialog box το οποίο μπορεί να χρησιμοποιήσει ο χρήστης για να customize την εμφάνιση και τη συμπεριφορά των εφαρμογών του LabVIEW.

Browse Menu

Περιέχει τις παρακάτω εντολές:

- **Show VI Hierarchy** εμφανίζει ένα Hierarchy window το οποίο χρησιμοποιείται για να ειδωθούν τα subVIs και άλλοι κόμβοι οι οποίοι αποτελούν το ενεργό VI.

- **This VI's Callers** επιτρέπει στον χρήστη να αποκτήσει πρόσβαση σε μια λίστα η οποία περιλαμβάνει όλα τα VIs τα οποία καλούν το τρέχον VI σαν subVI.

- **This VI's SubVIs** επιτρέπει στον χρήστη να αποκτήσει πρόσβαση σε μια λίστα η οποία περιλαμβάνει όλα τα subVIs τα οποία υπάρχουν στο τρέχον VI.

- **Unopened SubVIs** επιτρέπει στον χρήστη να αποκτήσει πρόσβαση σε μια λίστα η οποία περιλαμβάνει όλα τα unopened subVIs του τρέχοντος VI.
- **Unopened Type Defs** επιτρέπει στον χρήστη να αποκτήσει πρόσβαση σε μια λίστα η οποία περιλαμβάνει όλα τα unopened type definitions του τρέχοντος VI.
- **Breakpoints** ψάχνει τα breakpoints του τρέχοντος VI και τα εμφανίζει στο Search Results window.

Help Menu

Περιέχει τις παρακάτω εντολές:

- **Show Context Help** εμφανίζει το **Context Help** window, το οποίο παρέχει βασικές πληροφορίες για οποιοδήποτε VI, function ή control. Για να εμφανιστεί στο παράθυρο η βοήθεια ενός συγκεκριμένου αντικειμένου το πρέπει να μετακινήσουμε τον κέρσορα πάνω σε αυτό.
- **Lock Context Help** κλειδώνει το τρέχον περιεχόμενο του **Context Help** window. Όταν αυτή η εντολή είναι ενεργοποιημένη τότε η μετακίνηση του κέρσορα πάνω σε ένα άλλο αντικείμενο είτε του block diagram είτε του front panel, δεν αλλάζει το περιεχόμενο του **Context Help** window.
- **VI, Function, & How-To Help** εμφανίζει το πλήρες περιεχόμενο της βοήθειας που διαθέτει το LabVIEW σε ηλεκτρονική μορφή. Ο χρήστης μπορεί να πάρει πληροφορίες και βοήθεια για οποιοδήποτε αντικείμενο που υπάρχει στο LabVIEW συμπεριλαμβανομένου και των παρακάτω: palettes, menus, tools, VIs, και functions. Επίσης περιέχει βήμα προς βήμα οδηγίες για την χρήση του LabVIEW.
- **Search the LabVIEW Bookshelf** εμφανίζει την PDF (Portable Documentation Format) version του εγχειριδίου του LabVIEW.
- **Help for This VI** δίνει την πλήρη βοήθεια για το συγκεκριμένο VI η οποία και προέρχεται από το *LabVIEW Help*.
- **Find Examples** επιτρέπει στο χρήστη να ψάξει και να επιλέξει εκατοντάδες παραδείγματα από VIs. Ο χρήστης μπορεί να τροποποιήσει ένα παράδειγμα έτσι ώστε να μπορεί να το χρησιμοποιήσει στις εφαρμογές του.

- **Web Resources** δίνει στο χρήστη πρόσβαση σε Internet links τα οποία και παραπέμπουν σε National Instruments Technical Support, την LabVIEW KnowledgeBase, NI Developer Zone, και άλλες online National Instruments πηγές.

- **Explain Error** δίνει όλες τις πληροφορίες που αφορούν τα λάθη του τρέχοντος VI.

- **About LabVIEW** δίνει πρόσβαση σε γενικές πληροφορίες που αφορούν την εγκατεστημένη έκδοση του LabVIEW, συμπεριλαμβανομένου version number και serial number.

2.1.2 Palettes

Το LabVIEW περιέχει τρεις παλέτες την control palette, την functions palette και την tools palette τις οποίες και παρουσιάζουμε στη συνέχεια. Από τις δύο πρώτες αντλούμε τα στοιχεία για να σχεδιάσουμε το front panel και το block diagram ενώ η tools palette περιέχει κάποια εργαλεία μορφοποίησης.

Control Palette

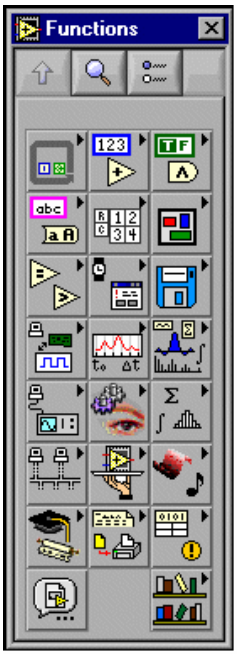
Για να ενεργοποιήσουμε την control palette πατάμε δεξί click στο χώρο του front panel, όταν αυτό είναι ενεργοποιημένο ή επιλέγουμε την εντολή **Show Control Palette** από το **Windows Menu** του front panel. Κατόπιν εμφανίζεται η παλέτα η οποία είναι πολύ εύχρηστη και βοηθάει τον προγραμματιστή να επιλέξει controls και indicators, όλων των μορφών, για το πρόγραμμα του. Οι κατηγορίες των controls που περιλαμβάνονται στην παλέτα είναι οι παρακάτω: numeric, Boolean, string & path, array & cluster, list & tables, graph, ring & enum, I/O, refnum, dialog control, classic controls, activex, decorations, select a control και user controls.

Εικόνα 2.2 Control Palette του Front panel

Functions Palette

Για να ενεργοποιήσουμε την functions palette πατάμε δεξί click στο χώρο του block diagram, όταν αυτό είναι ενεργοποιημένο ή επιλέγουμε την εντολή **Show Functions Palette** από το **Windows Menu** του block diagram. Από αυτήν την παλέτα μπορούμε να επιλέξουμε

μια build-in function του LabVIEW ή ένα VI και να χτίσουμε το block diagram του δικού μας VI. Οι κατηγορίες των functions που περιλαμβάνονται στην παλέτα είναι οι παρακάτω: structures, numeric, Boolean, string, array, cluster, comparison, time & dialog, file I/O, data acquisition, analyze, waveform, instrument I/O, motion & vision, mathematics, communication, application control, graphics & sound, tutorial, report generation, advanced, select a VI και user libraries.



Εικόνα 2.3 Η Functions Palette του block diagram

Tools Palette

Για να ενεργοποιήσουμε την tools palette επιλέγουμε την εντολή **Show Tools Palette** από το **Windows Menu** είτε του front panel είτε του block diagram.

Εικόνα 2.4 Η Tools Palette του LabVIEW

Τα σημαντικότερα εργαλεία αυτή της παλέτας είναι τα παρακάτω:

-  **Automatic Tool Selection**. Αν είναι ενεργοποιημένο αυτό το εργαλείο τότε όταν ο χρήστης μετακινώντας τον κέρσορα πάνω από ένα αντικείμενο είτε του front panel είτε του block diagram, το LabVIEW επιλέγει αυτόματα το κατάλληλο tool το οποίο πρέπει να χρησιμοποιηθεί.
-  Το **Operating** tool που μας επιτρέπει να αλλάζουμε τις τιμές των controls που βρίσκονται στο front panel αλλά και να χειριζόμαστε όλα τα όργανα που θα χρειαζόταν και στον αληθινό κόσμο την παρέμβαση του ανθρώπινου χεριού για τη λειτουργία τους όπως για παράδειγμα ένας διακόπτης. Είναι το μόνο tool που είναι διαθέσιμο στο run mode.
-  Το **Positioning** tool επιλέγει, μετακινεί και αλλάζει τις διαστάσεις των αντικειμένων
-  Το **Labeling** tool μας επιτρέπει να δημιουργήσουμε και να επεξεργαστούμε ετικέτες με κείμενο.
-  Το **Wiring** tool μας επιτρέπει να συνδέουμε τα διάφορα αντικείμενα που είναι τοποθετημένα στο block diagram μεταξύ τους.
-  Το **Object Shortcut Menu** tool εμφανίζει το μενού ενός αντικειμένου χωρίς να χρειαστεί να κάνουμε right click σε αυτό.
-  Το **Scrolling** tool μας επιτρέπει να κάνουμε scroll στην οθόνη χωρίς να χρησιμοποιούμε τα scrolling bars.
-  Το **Breakpoint** tool μας επιτρέπει να θέτουμε breakpoints σε VIs, κόμβους, καλώδια με σκοπό να σταματάμε την εκτέλεση του προγράμματος σε εκείνο το σημείο.
-  Το **Probe** tool μας επιτρέπει να δημιουργήσουμε ένα probe πάνω σε ένα καλώδιο με σκοπό να ελέγχουμε τις ενδιάμεσες τιμές σε ένα VI σε περίπτωση που αυτό παράγει μη αναμενόμενες τιμές.
-  Το **Color Copying** tool μας επιτρέπει να αντιγράψουμε ένα χρώμα και στη συνέχει να το χρησιμοποιήσουμε με το coloring tool
-  Το **Coloring** tool μας επιτρέπει να αλλάζουμε το χρώμα ενός αντικείμενου ή του φόντου.

2.2 Τα κυριότερα στοιχεία του LabVIEW

2.2.1 Structures

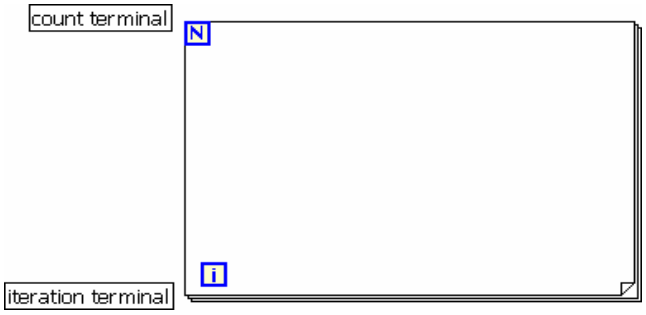
While Loop

Το while loop είναι μια δομή που επαναλαμβάνει ένα μέρος του κώδικα του προγράμματος μέχρι να εκπληρωθεί μια συνθήκη. Στο LabVIEW ένα while loop παριστάνεται με ένα ορθογώνιο σχήμα του οποίου τις διαστάσεις μπορεί να αλλάξει ο χρήστης. Εντός του loop υπάρχει το iteration terminal που περιέχει ανά πάσα στιγμή τον αριθμό των επαναλήψεων του βρόγχου. Πρέπει να ληφθεί υπ΄όψιν ότι επειδή η αρχική του τιμή είναι πάντα μηδενική αν ο βρόγχος εκτελεστεί μια φορά το iteration terminal περιέχει την τιμή 0. Εντός του βρόγχου επίσης υπάρχει και το conditional terminal το οποίο λειτουργεί σαν είσοδος και αποτελεί ουσιαστικά τη συνθήκη ελέγχου του βρόγχου. Πρόκειται για μια Boolean μεταβλητή εισόδου η οποία όταν πάρει την τιμή FALSE τερματίζεται η λειτουργία του whileloop. Το επιλέγουμε από το functions palette<structures<while loop

Εικόνα 2.5. While Loop

For Loop

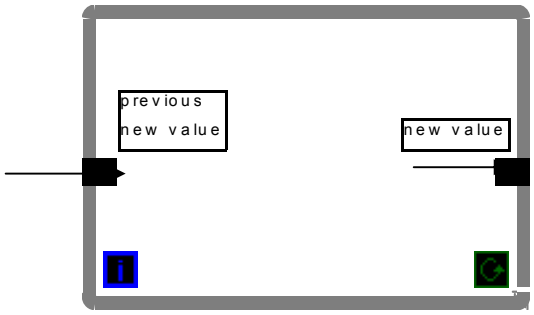
Το for loop είναι μια δομή η οποία εκτελεί ένα μέρος του προγράμματος για έναν καθορισμένο αριθμό επαναλήψεων. Όπως το while loop παριστάνεται με ένα ορθογώνιο σχήμα του οποίου οι διαστάσεις μπορούν να αλλαχθούν. Περιλαμβάνει δύο terminals, το count terminal το οποίο λειτουργεί σαν είσοδος και καθορίζει τον αριθμό των επαναλήψεων και το iteration terminal που περιέχει τον αριθμό των επαναλήψεων που έχουν εκτελεστεί. Το επιλέγουμε από το functions palette<structures<for loop .



Εικόνα 2.6.For Loop

Shift Registers

Οι shift registers χρησιμοποιούνται στο while και στο for loop και μεταφέρουν τιμές από μία επανάληψη του βρόγχου στην επόμενη. Για να δημιουργήσουμε ένα νέο shift register κάνουμε left click στην δεξιά ή αριστερή πλευρά των loops και από το μενού που εμφανίζεται επιλέγουμε Add Shift Register. Ένας shift register αποτελείται από δύο terminals τα οποία είναι τοποθετημένα το ένα στη δεξιά και το άλλο στην αριστερή πλευρά του βρόγχου. Το δεξί terminal αποθηκεύει την τιμή μιας μεταβλητής κατά την ολοκλήρωση μιας επανάληψης του βρόγχου. Η τιμή αυτή εμφανίζεται στη συνέχεια στο αριστερό terminal κατά την έναρξη της επόμενης επανάληψης του βρόγχου. Μπορεί να αποθηκεύσει όλων των ειδών τα δεδομένα – αριθμητικά, Boolean, string, array κ.α.

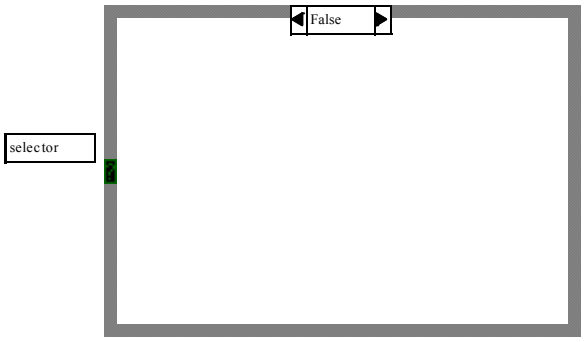


Εικόνα 2.7. Shift Registers

Case Structure

Η δομή Case εμπεριέχει δύο ή περισσότερα subdiagrams ή αλλιώς cases από τα οποία εκτελείται μόνο το ένα κάθε φορά που εκτελείται η δομή. Στην αριστερή πλευρά του συμβόλου της δομής case υπάρχει ένας επιλογέας που ανάλογα με την τιμή που παίρνει σαν είσοδο διαλέγει και εκτελεί το ανάλογο subdiagram. Η τιμή της εισόδου μπορεί να είναι ακέραια, Boolean ή string. Για να προσθέσουμε cases πατάμε δεξί click στις πλευρές της case structure και επιλέγουμε **Add Case**.

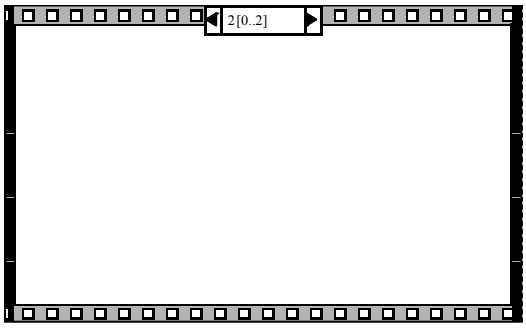
Το επιλέγουμε από το functions palette<structures<case



Εικόνα 2.8. Case Structure

Sequence Structure

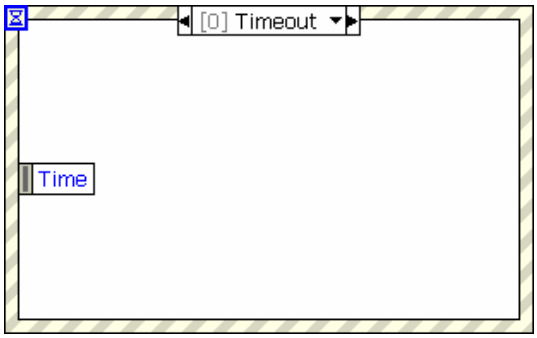
Η sequence structure εκτελεί ακολουθιακά έναν αριθμό block diagrams. Μοιάζει με τη δομή case διότι περιέχει πολλά στιγμιότυπα. Σε αυτή την περίπτωση όμως δεν εκτελείται μόνο το diagram που περιέχεται στο στιγμιότυπο που επιλέχθηκε αλλά όλα τα diagrams με την σειρά που τα έχουμε τοποθετήσει. Έτσι πρώτα θα εκτελεστεί το diagram που περιέχεται στο στιγμιότυπο 0, ακολούθως αυτό που περιέχεται στο στιγμιότυπο 1 κ.ο.κ. Η δομή αυτή είναι χρήσιμη στον έλεγχο της ροής των δεδομένων αποφεύγοντας έτσι πιθανά λάθη από την εκτέλεση κάποιου κόμβου πριν την επιθυμητή σειρά εκτέλεσης του. Για να προσθέσουμε ένα frame πατάμε δεξί click στις πλευρές του sequence structure και στη συνέχεια επιλέγουμε **Add Frame Before** ή **Add Frame After** ανάλογα με την θέση που θέλουμε το τοποθετήσουμε.Το επιλέγουμε από το functions palette<structures<sequence .



Εικόνα 2.9. Sequence structure

Event Structure

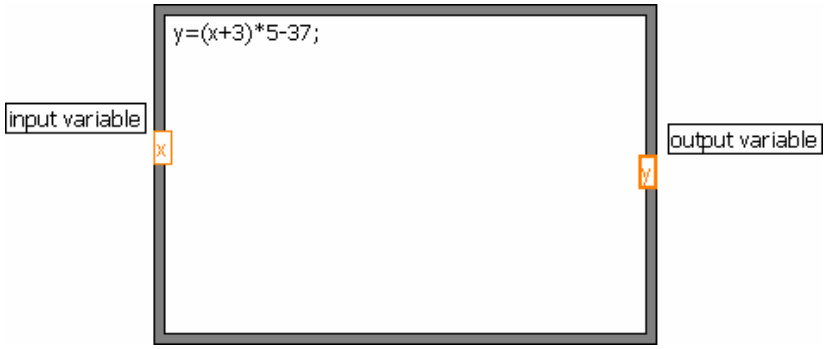
Έχει ένα ή περισσότερα subdiagrams, ή αλλιώς event cases, από τις οποίες εκτελείται μία μόνο όταν εκτελείται η δομή. Το Event structure περιμένει μέχρι να συμβεί ένα συγκεκριμένο γεγονός στο front panel, και στη συνέχεια εκτελεί την κατάλληλη. Πατώντας δεξί click στο σύνορο της δομής προστίθενται νέα event cases καθώς επίσης και με ποια γεγονότα θα συσχετιστεί η δομή. Το επιλέγουμε από το functions palette<structures<event structure .



Εικόνα 2.10. Event structure

Formula Node

Πρόκειται για μια δομή μέσα στην οποία μπορούμε να εισάγουμε απ΄ ευθείας αλγεβρικές φόρμουλες. Αυτό είναι πολύ χρήσιμο γιατί όταν έχουμε μια πολύπλοκη έκφραση η υλοποίηση της με τα αριθμητικά functions που περιέχει το LabVIEW καταλαμβάνει μεγάλη έκταση στο χώρο και γενικά κάνει το block diagram περίπλοκο. Επίσης υποστηρίζει και εκφράσεις παρόμοιες με αυτές της C. Οι built-in συναρτήσεις που υποστηρίζει είναι οι παρακάτω: abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, rand, rem, sec, sign, sin, sinc, sinh, sqrt, tan, tanh. Για να δημιουργήσουμε τα input και output terminals αρχικά πατάμε δεξί click στις πλευρές του Formula Node και στη συνέχεια επιλέγουμε **Add Input** ή **Add Output.** Κατόπιν βάζουμε τις ονόματα των μεταβλητών στα input και output boxes. Ο μέγιστος αριθμός των χαρακτήρων από τους οποίους μπορεί να αποτελείται το όνομα μιας μεταβλητής είναι δύο. Ακολούθως συνδέουμε το control το οποίο επιθυμούμε στο input box καθώςκαι ένα indicator στο output box για να λαμβάνουμε τα αποτελέσματα. Το επιλέγουμε από το functions palette<structures<formula node .

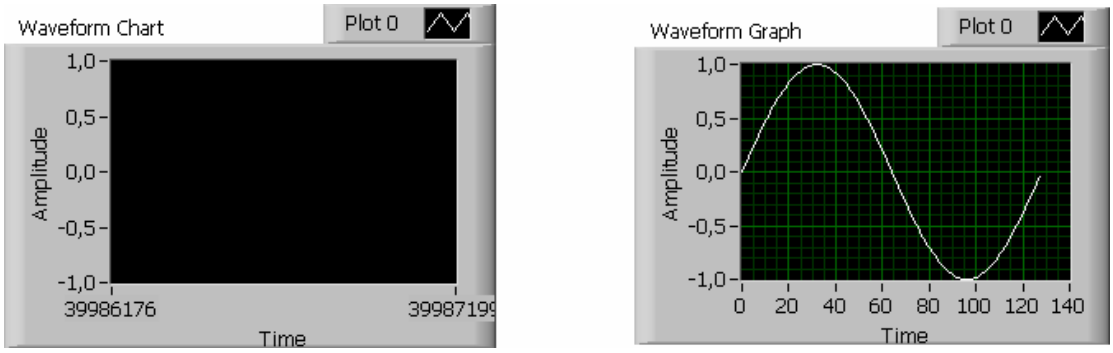


Εικόνα 2.11. Formula Node

2.2.2 Τα κυριότερα στοιχεία του Front Panel

Charts and Graphs

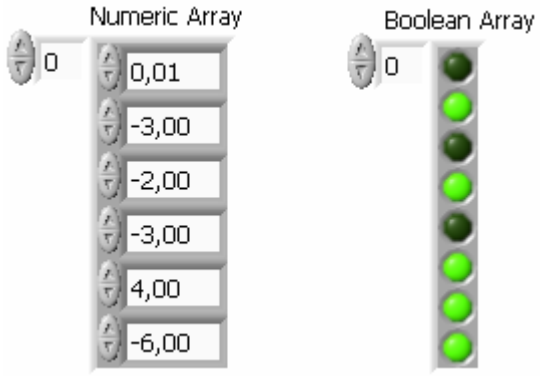
Το LabVIEW μας δίνει τη δυνατότητα να παραστήσουμε γραφικά μια σειρά από δεδομένα με εργαλεία που ονομάζονται charts και graphs. Η βασική διαφορά ανάμεσα σε charts και graphs έγκειται στον τρόπο με τον οποίο συλλέγουν και αποτυπώνουν τα δεδομένα. Έτσι VIs που διαθέτουν graphs συνήθως συλλέγουν τα δεδομένα σε ένα πίνακα και στη συνέχεια τα αποτυπώνουν στην οθόνη του graph μονομιάς. Αντίθετα τα charts προσθέτουν στην ήδη υπάρχουσα απεικόνιση τα νέα δεδομένα που έρχονται. Πρέπει επίσης να τονιστεί ότι υπάρχουν δύο τύποι graphs, το waveform graph και το XY graph. Η διαφορά ανάμεσα στα δύο είναι ότι το μεν πρώτο αποτυπώνει μονοσήμαντες συναρτήσεις, ενώ το δεύτερο είναι ένα γενικότερης χρήσης εργαλείο το οποίο μπορεί να απεικονίσει ακόμα και πολυσήμαντες συναρτήσεις.



Εικόνα 2.12. Chart και Graph

Arrays

Οι πίνακες (arrays) είναι ένα σύνολο στοιχείων του ίδιου τύπου. Μπορεί να έχουν μια ή περισσότερες διαστάσεις και μέχρι 2^{31} στοιχεία ανά διάσταση. Τα στοιχεία του πίνακα μπορούν να είναι οποιουδήποτε τύπου εκτός από array, chart και graph. Για να δημιουργήσουμε ένα array επιλέγουμε το Array &Graph Menu από το Control Palette. Στη συνέχεια εμφανίζεται στο front panel το array shell ενώ για να καθορίσουμε τον τύπο των στοιχείων του πίνακα, επιλέγουμε αρχικά από το Control Menu ένα data object και στη συνέχεια το σέρνουμε μέσα στο array shell. Ένας εναλλακτικός τρόπος επιλογής του τύπου των δεδομένων είναι μέσω του pop-up menu του πίνακα. Αν θέλουμε να προσθέσουμε μια νέα διάσταση στον πίνακα τότε επιλέγουμε από το pop-up menu **Add Dimension** και έτσι εμφανίζεται ένα επιπλέον index display που αναφέρεται στην καινούρια διάσταση. Το LabVIEW περιλαμβάνει πολλές λειτουργίες που σχετίζονται με τους πίνακες και ονομάζονται array functions. Οι λειτουργίες αυτές διευκολύνουν την και αξιοποιούν τη χρήση των πινάκων και βρίσκονται στη **Functions Palette** και συγκεκριμένα στο μενού **Array**.

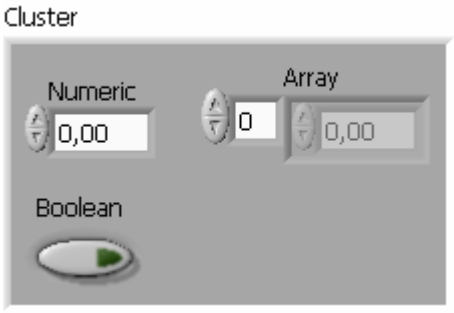


Εικόνα 2.13. Δύο μονοδιάστατα arrays, ένα με αριθμητικές και ένα με Boolean τιμές

Clusters

Το cluster είναι ένας τύπος δεδομένου ο οποίος περιέχει πολλαπλά στοιχεία δεδομένων τα οποία μπορούν να είναι διαφορετικού τύπου. Αυτό είναι και το στοιχείο που διαφοροποιεί ένα cluster από ένα array μιας και το δεύτερο ενώ περιλαμβάνει πολλαπλά στοιχεία αυτά θα πρέπει να είναι του ίδιου τύπου. Ο μόνος περιορισμός που υπάρχει ως προς τα στοιχεία του cluster είναι ότι θα πρέπει να είναι όλα ή controls ή indicators.

Μπορούμε να δημιουργήσουμε ένα cluster επιλέγοντας το από το Array&Graph Menu του Control Palette. Τα στοιχεία που περιέχονται σε ένα cluster έχουν μια αρίθμηση που έχει σχέση με τη σειρά που εισήχθησαν σε αυτό. Έτσι το πρώτο στοιχείο έχει τον αριθμό 0, το δεύτερο τον αριθμό 1 κ.ο.κ. Αν κάποιο από τα στοιχεία διαγραφεί τότε η σειρά αναπροσαρμόζεται αυτόματα. Ο χρήστης μπορεί να αλλάξει την αρίθμηση επιλέγοντας **Cluster Order** από το pop-up menu του cluster. Η αρίθμηση μέσα σε ένα cluster είναι πολύ σημαντική μιας και η προσπέλαση ενός στοιχείου γίνεται με βάση τον αριθμό του και όχι με βάση το όνομά του.



Εικόνα 2.14. Cluster αποτελούμενο από ένα numeric control, ένα Boolean control και ένα array

2.2.3 Τύποι Δεδομένων

Το LabVIEW υποστηρίζει ένα μεγάλο αριθμό από τύπους δεδομένων. Την πλήρη λίστα των τύπων δεδομένων την παραθέτουμε στον πίνακα 1. Μεγάλο ενδιαφέρον έχει ο τρόπος που αναπαρίστανται όλοι αυτοί οι διαφορετικοί τύποι δεδομένων στο γραφικό περιβάλλον προγραμματισμού του LabVIEW. Προς μεγάλη διευκόλυνση του χρήστη κάθε κατηγορία τύπων δεδομένων έχει το δικό της διαφορετικό χρώμα, ενώ μέσα στη γραφική αναπαράσταση του κάθε δεδομένου αναφέρεται με χαρακτηριστική συντομογραφία ο τύπος του. Να σημειώσουμε ότι με πορτοκαλί χρώμα παριστάνονται οι floating point αριθμοί, με μπλε οι ακέραιοι, με ροζ τα string, και με πράσινο οι Boolean. Για τα arrays ισχύει ότι το χρώμα τους εξαρτάται από τον τύπο των δεδομένων τους. Έτσι αν έχουμε ένα array που αποτελείται από ακέραιους αυτό θα είναι χρώματος μπλέ, αν αποτελείται από floating point αριθμούς θα είναι πράσινο κ.ο.κ.

Symbol	Data Type
	Floating point-Extended-precision **(EXT)**
	Floating point-Double-precision **(DBL)**
	Floating point-Single-precision **(SGL)**
	Integer-Long **(I32)**
	Integer-Word **(I16)**
	Integer-Byte **(I8)**
	Unsigned integer-Long **(U32)**
	Unsigned integer-Word **(U16)**
	Unsigned integer-Byte **(U8)**
	Boolean
	String
	Refnum
	Array αποτελούμενο από στοιχεία οποιοδήποτε από τους προηγούμενους τύπους δεδομένων
	Cluster

Πίνακας 1. Συνοπτική παράθεση των τύπων δεδομένων που υποστηρίζει το LabVIEW και το σύμβολο με το οποίο εμφανίζονται στο block diagram.

Ο ίδιος κανόνας ισχύει και στη χρωματική αναπαράσταση των καλωδίων τα οποία με τη σειρά τους παίρνουν ανάλογο χρώμα με αυτό των δεδομένων που μεταφέρουν. Εκτός όμως από το χρώμα ο τύπος των δεδομένων που μεταφέρουν τα καλώδια επηρεάζει και το πάχος τους. Έτσι αν μεταφέρουν απλούς αριθμούς εμφανίζονται λεπτότερα από ότι όταν μεταφέρουν arrays. Διαφορετική είναι επίσης η αναπαράσταση ενός καλωδίου όταν μεταφέρει πολυδιάστατα arrays καθώς και διαφορετική όταν μεταφέρει clusters.

Πρέπει να αναφέρουμε επίσης και τη διαφορά που υπάρχει στην αναπαράσταση των controls και των indicators. Η διαφορά αυτή είναι απλή και ταυτόχρονα πολύ χαρακτηριστική μιας και το μόνο που αλλάζει σε ένα control και ένα indicator που υποστηρίζουν ίδιο τύπο δεδομένων είναι το περίγραμμά τους το οποίο σε ένα control είναι χρωματισμένο ενώ σε ένα indicator όχι. Παράλληλα ένα μικρό βελάκι υποδεικνύει ότι το control μεταδίδει δεδομένα ενώ ένα indicator δέχεται δεδομένα.

Εικόνα 2.15. Το αριστερό ζευγάρι δείχνει ένα control και ένα indicator που υποστηρίζουν floating point-double precision αριθμούς, ενώ το δεξί ζευγάρι δείχνει ένα control και ένα indicator που υποστηρίζουν arrays αποτελούμενα από integer-long(I32) αριθμούς.

Τέλος αν θέλουμε να αλλάξουμε τον τύπο δεδομένου ενός control ή ενός indicator επιλέγουμε από το pop-up menu του την επιλογή **representation** και κατόπιν τον τύπο που επιθυμούμε.

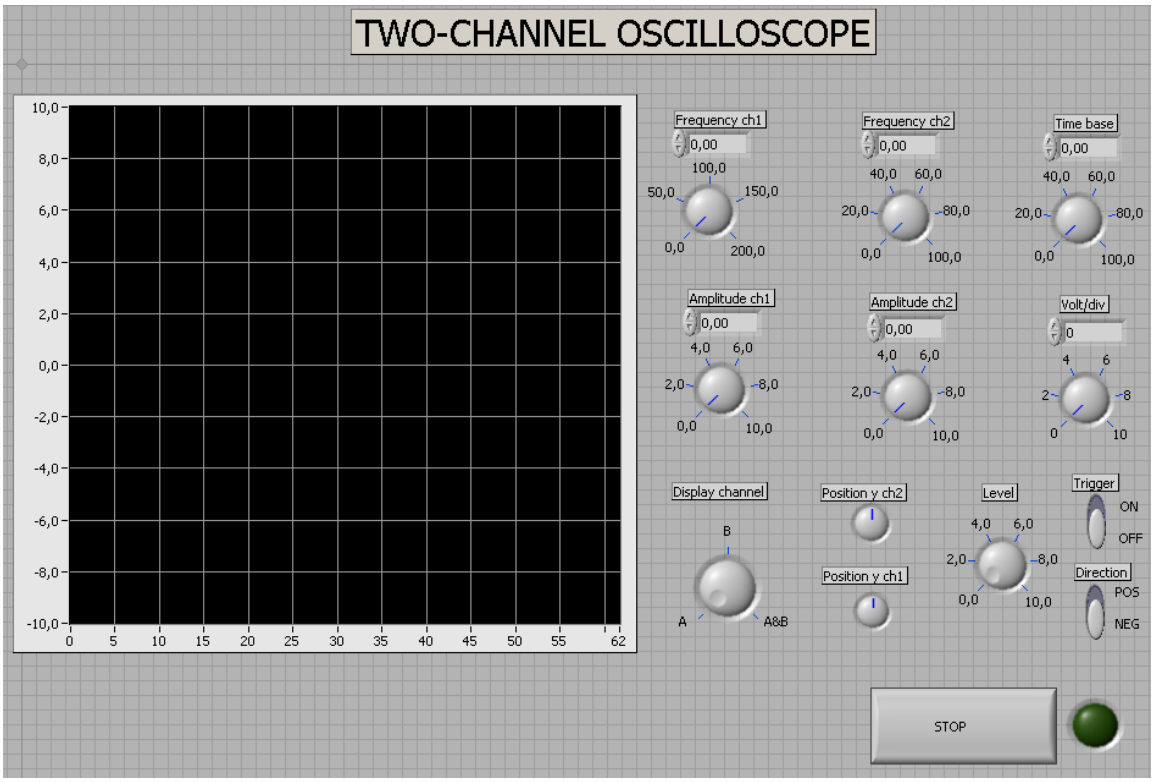
Κεφάλαιο 3

Το όργανο VI του εικονικού παλμογράφου

Σε αυτό το κεφάλαιο παρουσιάζουμε αναλυτικά το oscilloscope VI. Το VI αυτό είναι ένα εικονικός παλμογράφος δύο καναλιών. Στη συνέχεια θα αναλύσουμε το front panel και το block diagram σε δυο ξεχωριστά μέρη. Να υπενθυμίσουμε ότι το front panel αποτελεί το user interface του VI και το block diagram τον κώδικα του προγράμματος στην γλώσσα G.

3.1 Front Panel

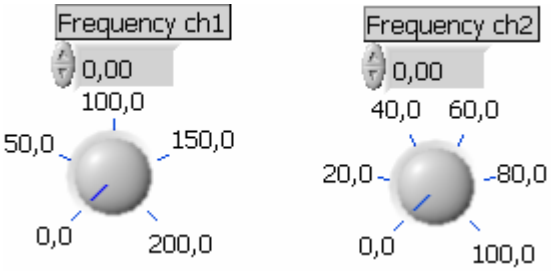
Στο πρώτο μέρος του κεφαλαίου θα παρουσιάσουμε αναλυτικά το front panel του παλμογράφου που δημιουργήσαμε. Η εικόνα 1 είναι αυτή που παρουσιάζεται στην οθόνη του υπολογιστή μόλις ‘ανοίξουμε’ το αρχείο **oscilloscope.VI** και αντιστοιχεί στο front panel του συγκεκριμένου VI. Στη συνέχεια θα αναφέρουμε ένα προς ένα τα controls που είναι τοποθετημένα στο front panel.



Εικόνα 3.1. Το front panel του παλμογράφου που δημιουργήσαμε

Frequency ch1 - Frequency ch2

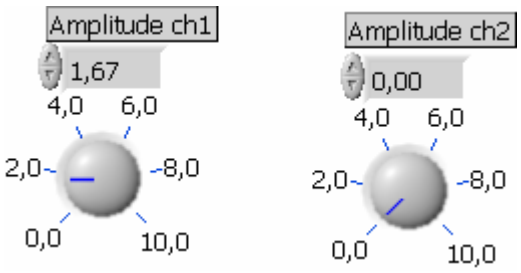
Με τα δύο αυτά controls ελέγχουμε από το front panel τη συχνότητα των σημάτων που εισάγονται στο κανάλι1 και το κανάλι 2 αντίστοιχα. Να σημειώσουμε ότι η συχνότητα των εισερχομένων σημάτων είναι κανονικοποιημένη και μετριέται ως προς κύκλους/δείγμα. Έτσι το πρώτο με το Frequency ch1 ρυθμίζουμε την συχνότητα του ημιτόνου που έχουμε συνδέσει στο κανάλι 1 και με το Frequency ch1 ρυθμίζουμε τη συχνότητα του τετραγωνικού παλμού που έχουμε συνδέσει στο κανάλι 2.Τα δύο controls που επιλέξαμε είναι τα **dial** και τα επιλέγουμε ως εξής: **controls palette<numeric<dial**



Εικόνα 3.2. Τα δύο dials όπως εμφανίζονται στο front panel

Amplitude ch1- Amplitude ch2

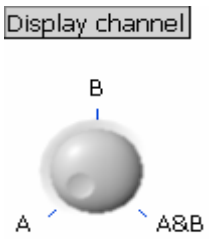
Με τα δύο αυτά controls ελέγχουμε από το front panel το πλάτος των σημάτων που εισάγονται στο κανάλι1 και το κανάλι 2 αντίστοιχα. Με το Amplitude ch1 ρυθμίζουμε το πλάτος του ημιτόνου που έχουμε συνδέσει στο κανάλι 1 και με το Amplitude ch1 ρυθμίζουμε το πλάτος του τετραγωνικού παλμού που έχουμε συνδέσει στο κανάλι 2. Να σημειώσουμε ότι το πλάτος δεν αναφέρεται σε peak-to-peak τιμές. Τα δύο controls που επιλέξαμε είναι τα **dial** και τα επιλέγουμε ως εξής: **controls palette<numeric<dial**



Εικόνα 3.3. Τα δύο dials όπως εμφανίζονται στο front panel

Display channel

Με αυτό το control επιλέγουμε το κανάλι του οποίου η προσλαμβάνουσα κυματομορφή επιθυμούμε να εμφανιστεί στην οθόνη του παλμογράφου. Υπάρχουν τρεις επιλογές **A,B, A&B**. Οι επιλογές αυτές εμφανίζουν αντίστοιχα την κυματομορφή του καναλιού Α, την κυματομορφή του καναλιού Β και τέλος και τις δύο κυματομορφές ταυτόχρονα. Το control που επιλέξαμε είναι το knob και το επιλέγουμε ως εξής: **controls palette<numeric<knob**

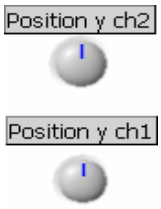


Εικόνα 3.4. Το knob του display channel όπως εμφανίζεται στο front panel

Position y ch1 - Position y ch2

Με αυτά τα δύο controls ελέγχουμε την θέση του προσλαμβάνοντος σήματος ως προς τον άξονα y. Έτσι μπορούμε να μετακινήσουμε την κυματομορφή είτε πάνω είτε κάτω από των άξονα των x.

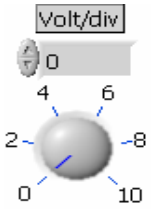
Το control που επιλέξαμε είναι το dial και το επιλέγουμε ως εξής: **controls palette<numeric<dial .**



Εικόνα 3.5. Τα δύο dials όπως εμφανίζονται στο front panel

Volt/div

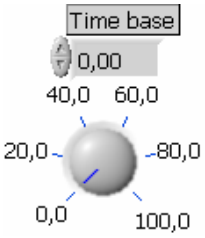
Με αυτό το control ελέγχουμε τη βαθμονόμηση του άξονα y. Έτσι η τιμή που επιλέγουμε αντιστοιχεί σε κάθε 'σκαλί' του άξονα y. Το control που επιλέξαμε είναι το dial και το επιλέ-γουμε ως εξής: **controls palette<numeric<dial**



Εικόνα 3.6. Το dial του Volt/div όπως εμφανίζεται στο front panel

Timebase

Με αυτο το control ελέγχουμε τη βαθμονόμηση του άξονα x. Έτσι η τιμή που επιλέγουμε αντιστοιχεί σε κάθε 'σκαλί' του άξονα x. Το control που επιλέξαμε είναι το dial και το επιλέγουμε ως εξής: **controls palette<numeric<dial**



Εικόνα 3.7. Το dial του Time base όπως εμφανίζεται στο front panel

Trigger

Με αυτό το control ελέγχουμε την ενεργοποίηση του trigger. Το trigger μας δίνει στην οθόνη του παλμογράφου μια σταθερή κυματομορφή, κάνοντας έτσι την παρατήρηση της

ευκολότερη. Το control που επιλέξαμε είναι το vertical slide switch και το επιλέγουμε ως εξής: **controls palette<Boolean<vertical slide switch**

Εικόνα 3.8. Το vertical slide switch του Trigger όπως εμφανίζεται στο front panel

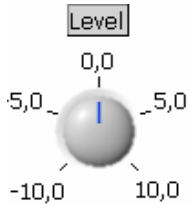
Direction

Με αυτό το control ελέγχουμε αν το triggering θα γίνει κατά τη στιγμή που το πλάτος του σήματος αυξάνεται ή κατά τη στιγμή που το πλάτος του σήματος μειώνεται. Έτσι αν θέσουμε το διακόπτη στο **POS** το triggering θα γίνει στην ημιπερίοδο κατά την οποία το πλάτος αυξάνεται ενώ αν θέσουμε το διακόπτη στο **NEG** το triggering θα γίνει στην ημιπερίοδο κατά την οποία το πλάτος μειώνεται. Το control που επιλέξαμε είναι το vertical slide switch και το επιλέγουμε ως εξής: **controls palette<Boolean<vertical slide switch**

Εικόνα 3.9. Το vertical slide switch του Slope όπως εμφανίζεται στο front panel

Level

Με το control αυτό ελέγχουμε το επίπεδο του πλάτους του προσλαμβανομένου σήματος από το οποίο ξεκινάει το triggering. Καθορίζει συνεπώς την τιμή της κυματομορφής στην αρχή των αξόνων. Το control που επιλέξαμε είναι το dial και το επιλέγουμε ως εξής: **controls palette<numeric<dial**



Εικόνα 3.10. Το dial του Level όπως εμφανίζεται στο front panel

Stop

Με το control αυτό σταματάμε τη λειτουργία του παλμογράφου. Να σημειώσουμε ότι το συγκεκριμένο control αφορά αποκλειστικά το σταμάτημα της λειτουργίας του παλμογράφου και όχι την έναρξή της. Το control που επιλέξαμε είναι το stop button και το επιλέ-γουμε ως εξής: **controls palette<Boolean<stop button**

Εικόνα 3.11. Το stop button όπως εμφανίζεται στο front panel

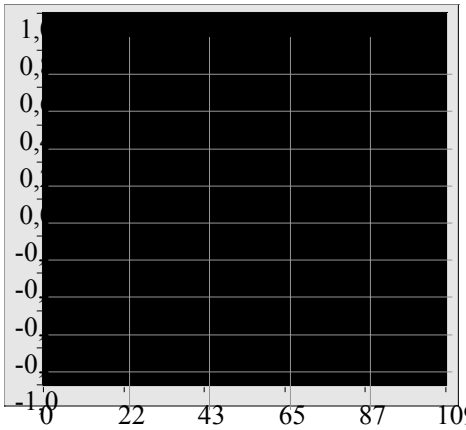
Led

Στο front panel έχουμε βάλει και ένα indicator εκτός από controls. Αυτό είναι ένα Led το οποίο αλλάζει χρώμα ανάλογα αν ο παλμογράφος είναι σε λειτουργία ή όχι. Έτσι όταν ο παλμογράφος λειτουργεί το Led έχει ανοιχτό πράσινο χρώμα ενώ όταν είναι εκτός λει-τουργίας είναι σκούρο πράσινο. Το επιλέγουμε ως εξής: **controls palette <Boolean< <Round Led**

Εικόνα 3.12. Το round Led και οι δύο καταστάσεις του OFF-ON όπως εμφανίζονται στο front panel

Graph

Το graph που υπάρχει στο front panel είναι ουσιαστικά η οθόνη του παλμογράφου μας και συνεπώς πρόκειται για ένα indicator και όχι για ένα control. Το επιλέγουμε ως εξής: **controls palette<Graph<Waveform Graph**



Εικόνα 3.13. Το Waveform graph που χρησιμεύει ως οθόνη του παλμογράφου μας

3.2 Block Diagram

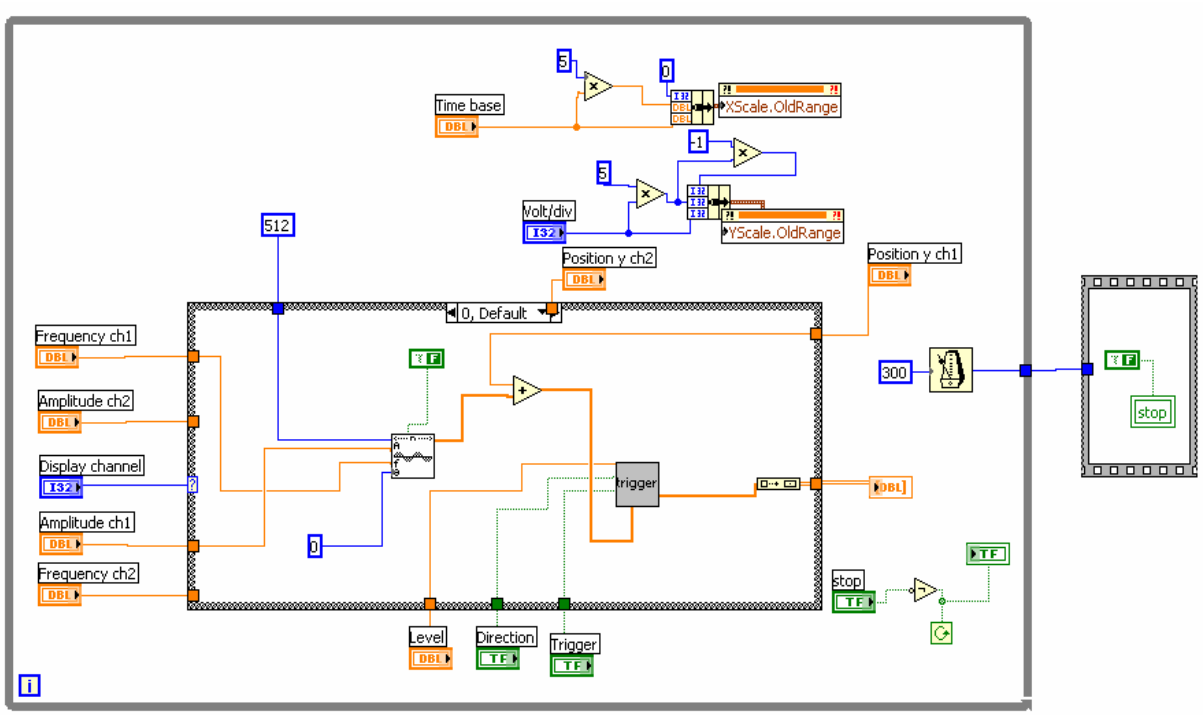
Το δεύτερο μέρος του κεφαλαίου χωρίζεται και αυτό σε δύο μέρη. Στο πρώτο θα παρουσιάσουμε με αναλυτικό τρόπο το block diagram και τον τρόπο λειτουργίας του έτσι ώστε να την κάνουμε κατανοητή στον αναγνώστη. Στο δεύτερο μέρος θα παραθέσουμε όλα τα έτοιμα functions τα οποία και χρησιμοποιήθηκαν για τη δημιουργία του block diagram.

3.2.1 Τρόπος λειτουργίας του Block Diagram

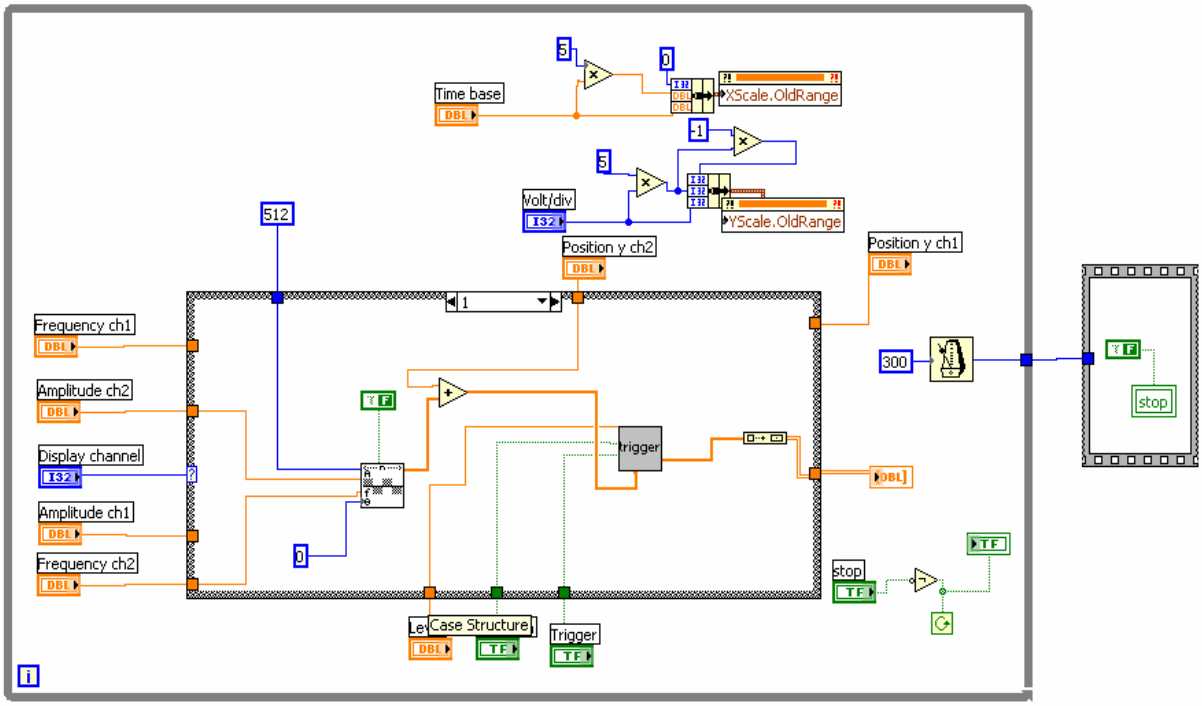
Για να γίνει πιο εύκολα κατανοητός ο τρόπος λειτουργίας του προγράμματος έχουμε χωρίσει το block diagram σε μικρότερα κομμάτια που ωστόσο εκτελούν μια διαφορετική λειτουργία. Έτσι παρουσιάζουμε ξεχωριστά τα εξής μέρη του block diagram: επιλογή καναλιού, εκκίνηση και τερματισμός λειτουργίας, volt/div & time base, position ch1 ch2, trigger.VI και slope.VI.

Επιλογή καναλιού

Για να δημιουργήσουμε το μηχανισμό επιλογής καναλιών χρησιμοποιήσαμε ένα case structure και έναν επιλογέα τριών καταστάσεων. Η λογική είναι ότι ανάλογα με την θέση του επιλογέα εκτελείται και μια διαφορετική case η οποία περιέχει και ένα διαφορετικό subdiagram. Όταν ο επιλογέας (display channel) τίθεται στη θέση A εκτελείται η case 0, στη θέση B εκτελείται η case 1 και στη θέση A&B εκτελείται η case 2. Στην case 0 υπάρχει το subdiagram το οποίο εμφανίζει στην οθόνη του παλμογράφου το ημιτονικό σήμα αφού αυτό περάσει μέσα από το VI του trigger. Στην case 1 υπάρχει το subdiagram το οποίο εμφανίζει στην οθόνη του παλμογράφου τον τετραγωνικό παλμό με την ίδια διαδικασία. Τέλος στην case 2 υπάρχει το subdiagram το οποίο ταυτόχρονα εμφανίζει και τα δύο σήματα στην οθόνη αφού πρώτα περάσουν από το trigger.

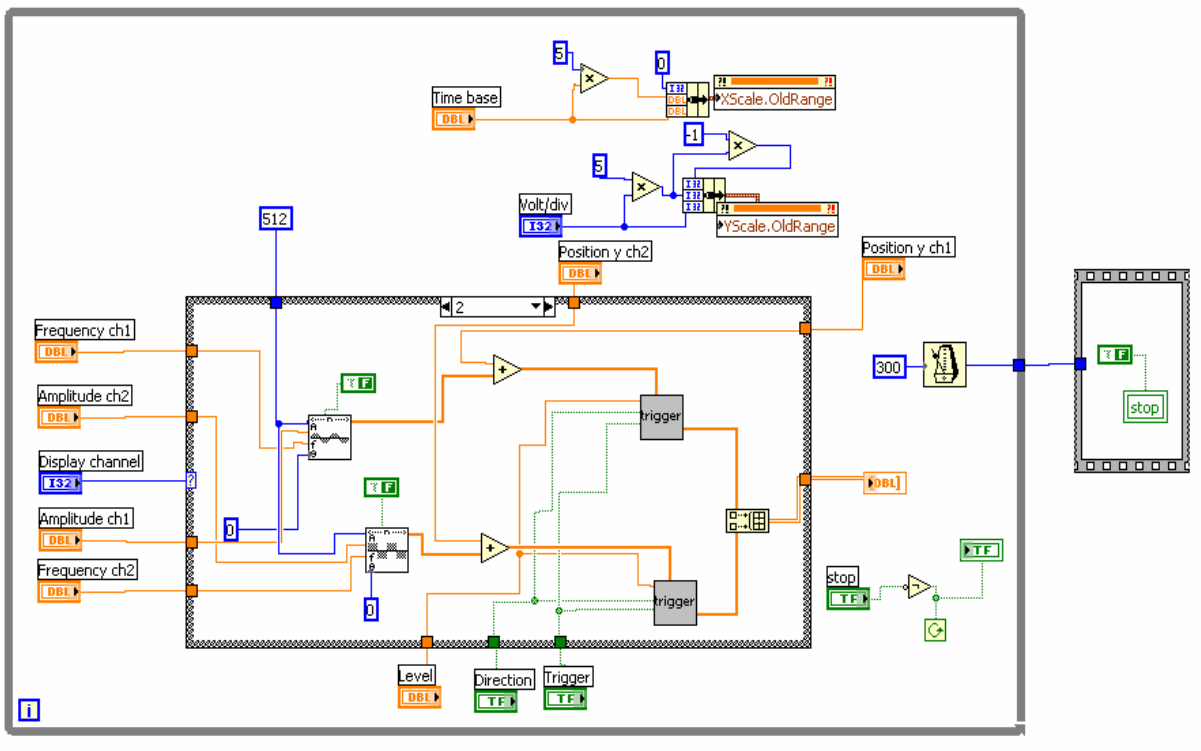


Εικόνα 3.14. Το block diagram όπως εμφανίζεται μετά την επιλογή του καναλιού Α



Εικόνα 3.15. Το block diagram όπως εμφανίζεται μετά την επιλογή του καναλιού Β

44



Εικόνα 3.16. Το block diagram όπως εμφανίζεται μετά την επιλογή των καναλιών A&B

Εκκίνηση και τερματισμός λειτουργίας

Παρατηρώντας το block diagram είναι φανερό ότι περιβάλλεται από ένα πλαίσιο το οποίο δεν είναι τίποτα άλλο από ένα while loop. Μέσα στο σώμα του while loop βρίσκεται ουσιαστικά ολόκληρο το πρόγραμμα. Αυτό συμβαίνει γιατί θέλουμε να εκμεταλλευτούμε την ιδιότητα του while loop που είναι η συνεχής λειτουργία, με έλεγχο της διακοπής από τον χρήστη. Με αυτόν τον τρόπο εξασφαλίζουμε συνεχή ροή του προγράμματος με το πάτημα του Run αλλά και τερματισμό του όποτε εμείς το επιθυμούμε. Η όλη λογική του μηχανισμού τερματισμού της λειτουργίας του while loop και κατά συνέπεια του μηχανισμού τερματισμού της λειτουργίας του παλμογράφου στηρίζεται στο ότι θέλουμε να σταματάει ο παλμογράφος με το πάτημα ενός κουμπιού -stop button- το οποίο όμως μετά το σταμάτημα θα επανέρχεται στην προηγούμενη του κατάσταση αυτόματα, χωρίς να χρειάζεται να το επαναφέρει ο χρήστης. Έτσι για να γίνει επανεκκίνηση του παλμογράφου θα χρειαστεί μόνο το πάτημα εκ νέου του Run, αντί να χρειάζεται να επαναφέρουμε πρώτα το stop button και μετά να εκκινήσουμε τον παλμογράφο. Παράλληλα θέλουμε κατά τη διάρκεια λειτουργίας του παλμογράφου να υπάρχει μια φωτεινή ένδειξη η οποία θα σβήνει όταν ο παλμογράφος τίθεται εκτός λειτουργίας. Ο μηχανισμός τερματισμού της λειτουργίας του παλμογράφου αποτελείται από ένα Boolean control, ένα Boolean indicator, μια not function, ένα sequence structure, μια Boolean σταθερά και μια local variable. Το conditional terminal του while loop τερματίζει την

λειτουργία όταν του δοθεί η τιμή FALSE. Αυτός είναι και ο λόγος που δε συνδέουμε το stop button απευθείας στο conditional terminal αλλά μέσω μιας πύλης not. Έτσι όταν πατάμε το stop button και το θέτουμε ουσιαστικά στην κατάσταση TRUE στο conditional terminal φτάνει η τιμή FALSE και τερματίζεται η λειτουργία του παλμογράφου. Για τον ίδιο λόγο και το Led (Boolean indicator) συνδέεται με το stop button (Boolean control) μέσω της πύλης not, ούτως ώστε να εμφανίζεται η φωτεινή ένδειξη όταν αυτό βρίσκεται στην κατάσταση not και συνεπώς ο παλμογράφος είναι σε λειτουργία.

Για να επαναφέρουμε το stop button στην αρχική του FALSE κατάσταση, μετά το σταμάτημα της λειτουργίας του παλμογράφου χρησιμοποιούμε ένα sequence structure. Εκεί έχουμε τοποθετήσει μια local variable του stop button την οποία και έχουμε μόνιμα συνδεδεμένη με μια Boolean σταθερά η οποία έχει την τιμή TRUE. Έτσι μετά το σταμάτημα της λειτουργίας του while loop το πρόγραμμα εκτελεί το sequence structure επαναφέροντας έτσι το stop button στην τιμή FALSE.

Volt/div & Time base

Για να δημιουργήσουμε το μηχανισμό αλλαγής της κλίμακας μέτρησης των αξόνων x,y χρησιμοποιήσαμε το property node, τα functions bundle και multiply καθώς και κάποιες σταθερές. Η λογική είναι ότι μέσω του bundle δίνουμε στα property nodes της οθόνης του παλμογράφου το άνω και κάτω όριο των αξόνων καθώς και την τιμή που θα έχει το κάθε division του άξονα. Επειδή έχουμε σταθερό τον αριθμό των divisions του κάθε άξονα, είναι λογικό ότι τα όρια που θέτουμε στους άξονες είναι σε σχέση με την τιμή του division που επιλέγει ο χρήστης. Έτσι για τον άξονα x επιλέγουμε ως κάτω όριο το –5*Volt/div και ως άνω όριο το 5*Volt/div. Για τον άξονα y επιλέγουμε ως κάτω όριο το 0 και ως άνω όριο το 5*Time base. Οι τιμές Volt/div & Time base επιλέγονται από τον χρήστη όπως περιγράψαμε στο κεφάλαιο του control panel.

Position ch1,ch2

Για να επιτύχουμε την μετακίνηση της κυματομορφής κατά τον άξονα των y εφαρμόσαμε έναν πολύ απλό μηχανισμό αποτελούμενο από ένα add function και μια floating point μεταβλητή. Στην είσοδο που λαμβάνει ο παλμογράφος προσθέτουμε την floating point τιμή - την οποία καθορίζει ο χρήστης με ένα dial – μετακινώντας ουσιαστικά την κυμα-τομορφή είτε προς τα πάνω, αν η τιμή είναι θετική, είτε προς τα κάτω, αν η τιμή είναι αρνητική.

Trigger.VI

Το trigger.VI έχει τέσσερις εισόδους και μια έξοδο. Οι τέσσερις είσοδοι είναι οι εξής: α)μια Boolean μεταβλητή που ενεργοποιεί ή απενεργοποιεί το triggering **(trigger)**, β)μια Boolean μεταβλητή που καθορίζει το αρνητικό ή θετικό μέτωπο της κυματομορφής **(slope)**, γ)μια ακέραια μεταβλητή που καθορίζει το επίπεδο από το οποίο ξεκινάει το triggering **(level)** και δ)η κυματομορφή πάνω στην οποία θα εφαρμοστεί το triggering. Η έξοδος του VI είναι η προκύπτουσα κυματομορφή.

Η Boolean μεταβλητή α) είναι αυτή που συνδέεται στο selector του case structure που υπάρχει στο block diagram του VI. Έτσι όταν η μεταβλητή που ελέγχεται από τον αντίστοιχο διακόπτη του front panel παίρνει την τιμή TRUE, τότε το VI εκτελεί το αντίστοιχο subdiagram του case structure ενώ το ίδιο συμβαίνει όταν παίρνει την τιμή FALSE.

Το subdiagram FALSE ουσιαστικά είναι κενό και το μόνο που κάνει είναι να συνδέει απευθείας την κυματομορφή της εισόδου με την έξοδο. Αυτό είναι κάτι το αναμενόμενο μας και αφού ο χρήστης δεν έχει ενεργοποιήσει το triggering η κυματομορφή που θα παίρνουμε στην έξοδο θα πρέπει να είναι ίδια με αυτήν που προσλαμβάνει ο παλμογράφος.

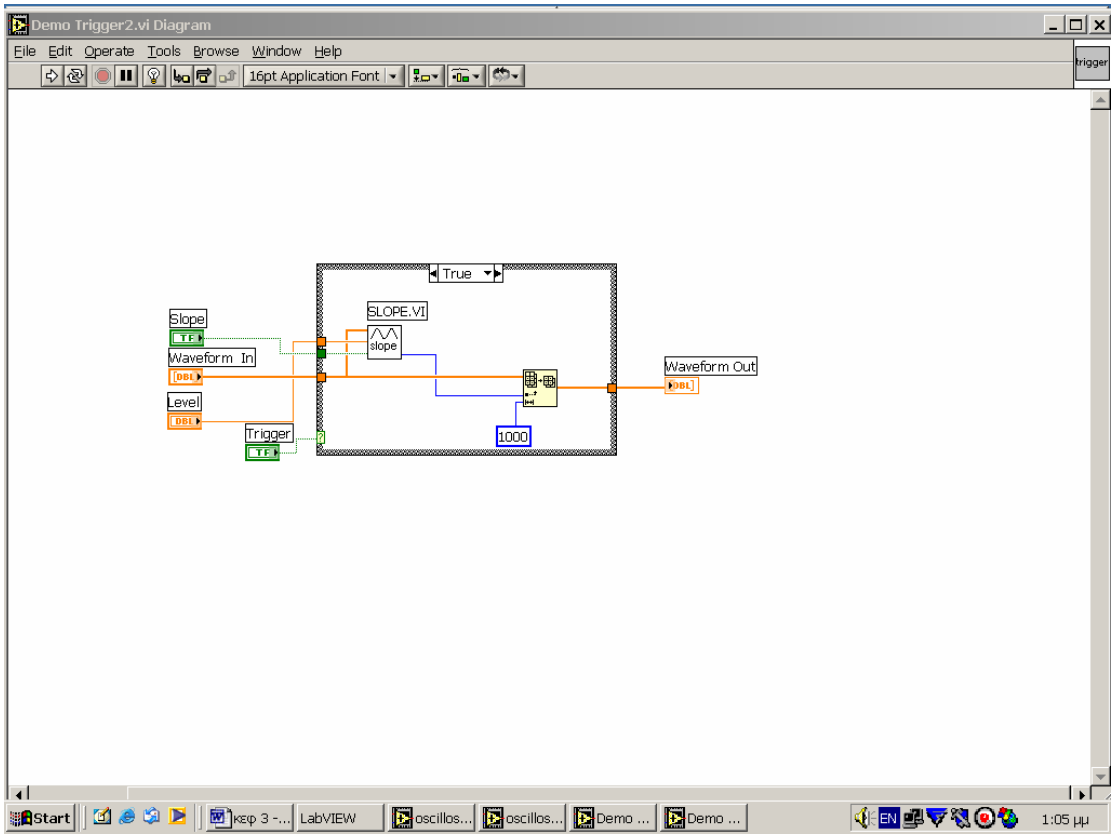
Το subdiagram TRUE είναι αυτό που περιέχει το πρόγραμμα με το οποίο γίνεται το triggering. Σε αυτό περιλαμβάνεται ένα VI το οποίο θα αναλύουμε (**slope.VI**) ξεχωριστά και ένα function (**Array Subset**). Η λογική είναι πολύ απλή και βασίζεται στο γεγονός ότι οι κυματομορφές είναι σε μορφή πινάκων. Στο array subset φτάνουν τρεις είσοδοι οι οποίες είναι:

a) μια σταθερά την οποία έχουμε καθορίσει στην τιμή 1000 και συνδέεται στην είσοδο length

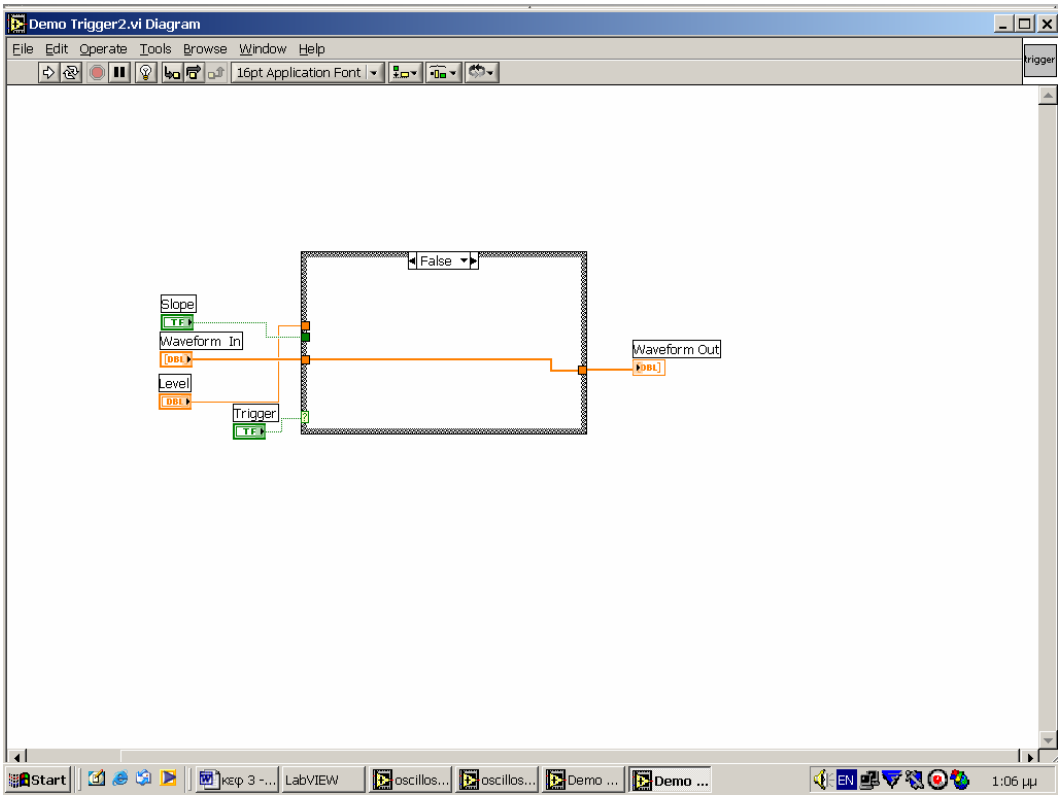
b) η κυματομορφή της εισόδου που συνδέεται στην είσοδο array και

c) μια ακέραια τιμή η οποία προκύπτει από το slope.VI και συνδέεται στην είσοδο index.

Στην έξοδο του function προκύπτει ένα καινούριο array του οποίου το πρώτο στοιχείο είναι το "index" στοιχείο του πίνακα της κυματομορφής εισόδου και το τελευταίο το "index + 999" στοιχείο του πίνακα της κυματομορφής εισόδου. Αυτός ο πίνακας που μόλις περιγράψαμε είναι και η κυματομορφή εξόδου.



Εικόνα 3.17. Το block diagram του trigger.VI όταν ο διακόπτης trigger του front panel είναι στη θέση on.



Εικόνα 3.18. Το block diagram του trigger VI όταν ο διακόπτης trigger του front panel είναι στη θέση off.

Slope.VI

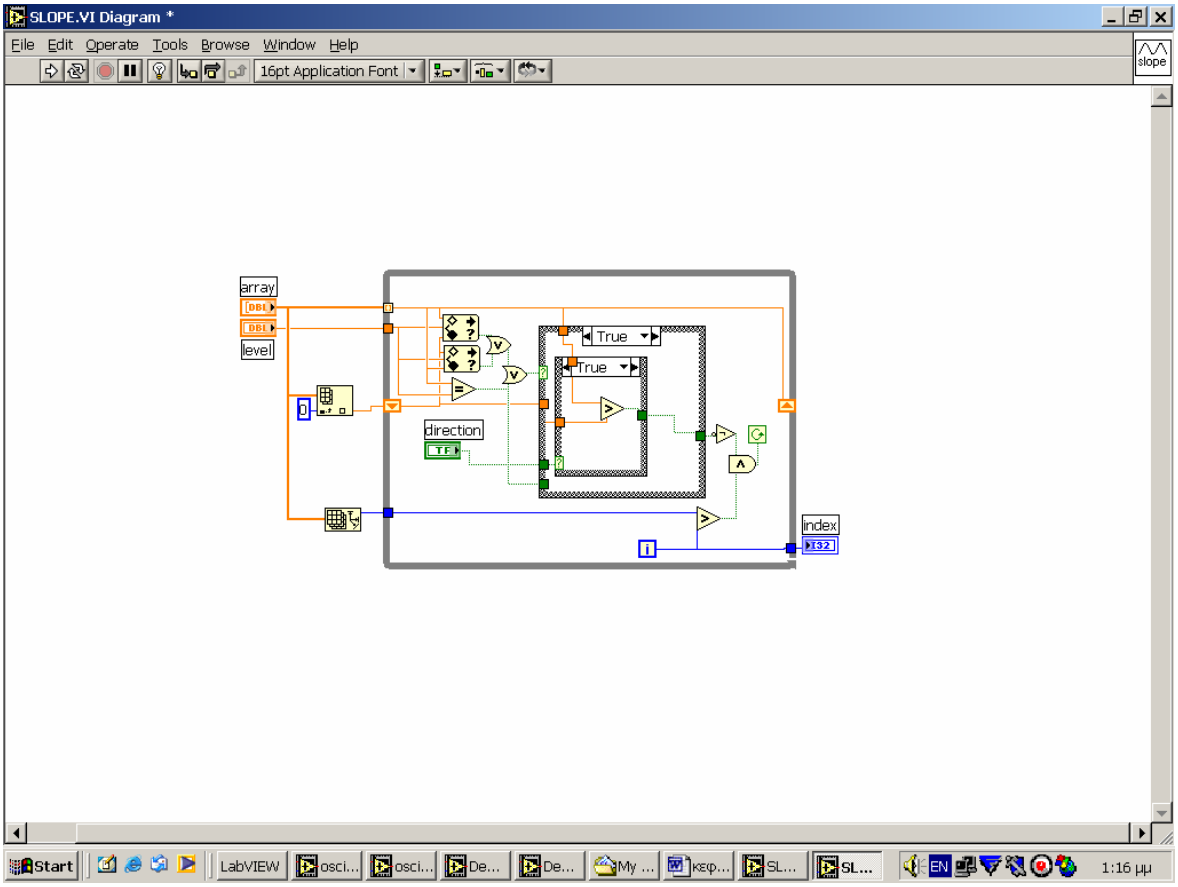
Όπως προαναφέραμε στην ανάλυση του trigger.VI, το slope.VI έχει σαν σκοπό τον καθορισμό της τιμής του δείκτη-index. Ο δείκτης αυτός με τη σειρά του καθορίζει το πρώτο στοιχείο του νέου πίνακα που προκύπτει μετά τη διαδικασία του triggering. Ουσιαστικά δηλαδή, καθορίζει το σημείο από το οποίο θα ξεκινάει η κυματομορφή η οποία έχει υποστεί το triggering.

Το block diagram του VI φαίνεται αρκετά περίπλοκο, χωρίς όμως αυτό να είναι πραγματικότητα. Πρόκειται ουσιαστικά για μια σειρά συγκρίσεων οι οποίες και καταλήγουν στην εύρεση της τιμής του index. Αρχικά να αναφερθούμε στους shift registers οι οποίοι έχουν σαν ρόλο την μεταβίβαση της προηγούμενης τιμής του πίνακα της κυματομορφής εισόδου, στην νέα επανάληψη της εκτέλεσης του while loop. Έτσι ουσιαστικά έχουμε σε κάθε επανάληψη τρεις τιμές οι οποίες υπόκεινται σε μεταξύ τους συγκρίσεις. Οι τρεις αυτές τιμές είναι: δύο διαδοχικές τιμές του πίνακα της κυματομορφής εισόδου και η τιμή του level.

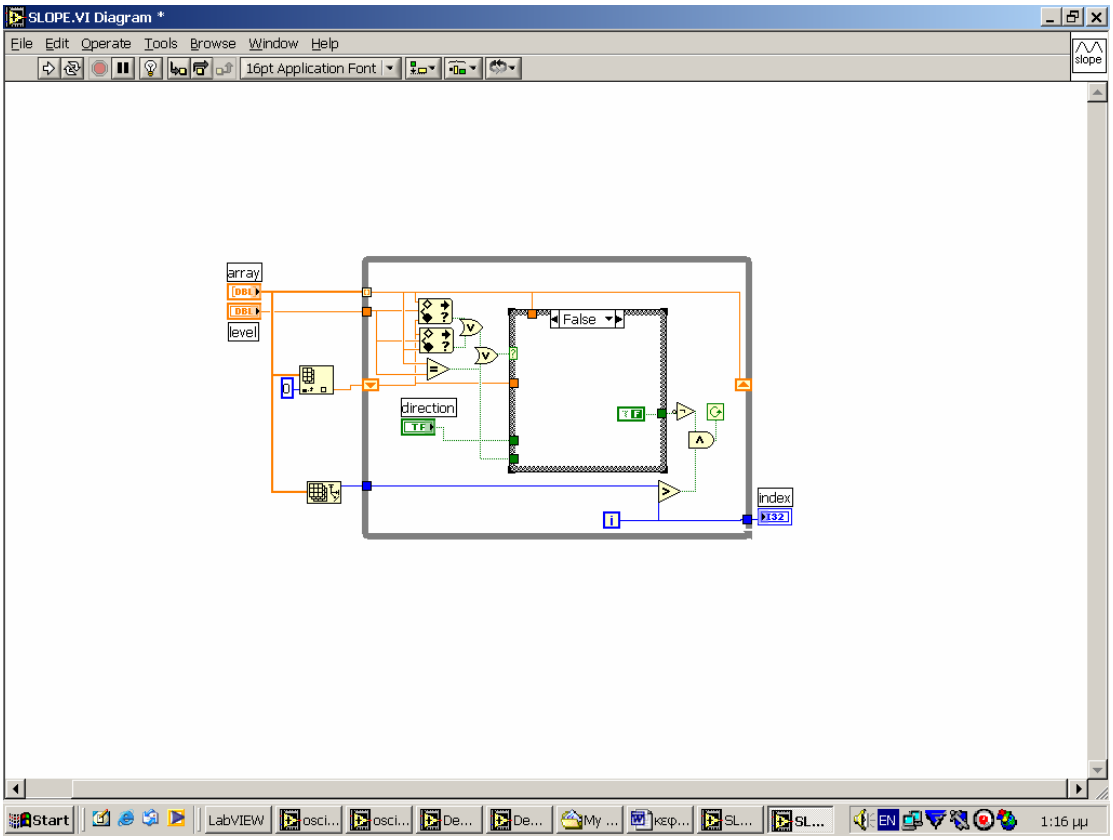
Ο μηχανισμός των αρχικών συγκρίσεων περιλαμβάνει τρεις συγκριτές και δύο πύλες OR. Η έξοδος όλου αυτού του μηχανισμού καταλήγει στο selector του εξωτερικού case structure. Για να είναι η τιμή αυτή TRUE αρκεί να συμβαίνει ένα από τα παρακάτω ενδεχόμενα $\alpha_{i-1}<level<\alpha_i$, $\alpha_i<level<\alpha_{i-1}$, $\alpha_i=level$. Αν δεν ισχύει κανένα από τα προηγούμενα ενδεχόμενα τότε η τιμή που πηγαίνει στο selector είναι η FALSE. Μέσα στο block diagram διακρίνουμε δύο case structures. Το εξωτερικό έχει συνδεδεμένο στο selector του μια Boolean τιμή η οποία και προκύπτει από τις διαδοχικές συγκρίσεις των τριών τιμών που προαναφέραμε. Αν στο selector φτάσει η τιμή FALSE και συνεπώς επιλεγεί το αντίστοιχο case για εκτέλεση γίνεται εύκολα κατανοητό ότι δεν βρέθηκε η κατάλληλη τιμή του Index και ότι θα ακολουθήσει και νέα επανάληψη του while loop. Αυτό το επιτυγχάνουμε με μια απλή Boolean σταθερά η οποία υπάρχει στην FALSE case και συνδέεται στο μηχανισμό τερματισμού λειτουργίας του while loop. Η σταθερά αυτή είναι στην τιμή FALSE και μέσω του μηχανισμού που προαναφέραμε διατηρεί στο conditional terminal την τιμή TRUE. Στην περίπτωση που στο selector φτάσει η τιμή TRUE τότε εκτελείται και η αντίστοιχη TRUE case η οποία και περιλαμβάνει ένα νέο case structure. Το εσωτερικό αυτό structure έχει συνδεδεμένο στο selector του τον διακόπτη καθορισμού της διεύθυνσης του triggering. έτσι γίνεται κατανοητό ότι ακόμα και αν έχουν πετύχει η αρχικές συγκρίσεις για να τερματιστεί η λειτουργία του loop και κατά συνέπεια θα πρέπει να πετύχει και η σύγκριση του εσωτερικού case.

Το index array που βρίσκεται εκτός από το while loop έχει σαν σκοπό την αρχικοποίηση του shift register δίνοντας του ως πρώτη τιμή το πρώτο στοιχείο του πίνακα της κυματομορφής εισόδου. Το array size μεταφέρει στο μηχανισμό του τερματισμού της λειτουργίας του while loop τον αριθμό των στοιχείων που περιλαμβάνονται στον πίνακα εισόδου.

Ο μηχανισμός τερματισμού είναι απλός και περιλαμβάνει δύο πύλες και ένα comparison function. Στο conditional terminal του loop συνδέεται η έξοδος της πύλης AND. Οι είσοδοι της πύλης AND είναι η έξοδος της πύλης NOT καθώς και η έξοδος του function Greater?. Η πύλη NOT με τη σειρά της αντιστρέφει την τιμή η οποία προκύπτει από τα case structures. Όπως γίνεται κατανοητό αν η τιμή που εισάγεται στον αντιστροφέα είναι η TRUE τότε είναι σίγουρος ο τερματισμός ατου while loop. Ο συγκριτής με τη σειρά του έχει σαν εισόδους το iteration terminal και την τιμή που προκύπτει από το array size. Για να δώσει στην έξοδό του ο συγκριτής την τιμή FALSE και κατά συνέπεια να τερματιστεί η λειτουργία του loop, θα πρέπει το VI να έχει εξετάσει όλα τα στοιχεία του πίνακα εισόδου και να μην έχει βρει το κατάλληλο σημείο από το οποίο πρέπει να ξεκινήσει το triggering σύμφωνα πάντα με τα δεδομένα που του έχουμε δώσει(level-direction).



Εικόνα 3.19. Το block diagram του slope.VI όταν εκτελείται το case TRUE του εξωτερικού case structure



Εικόνα 3.20. Το block diagram του slope.VI όταν εκτελείται το case FALSE του εξωτερικού case structure

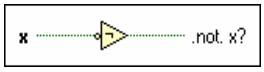
3.3 Functions

Σε αυτό το μέρος του κεφαλαίου παρουσιάζουμε όλα τα Functions τα οποία χρησιμοποιήσαμε στο «χτίσιμο» του block diagram του παλμογράφου μας. Αναφέρεται περιληπτικά η συνάρτηση που υλοποιούν, η χρήση τους, ενώ αναλυτικά αναφέρουμε τον τρόπο επιλογής τους.

a. Boolean Functions

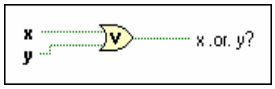
Not

Υπολογίζει το λογικό συμπλήρωμα της εισόδου. Αν το x είναι FALSE, η συνάρτηση επιστρέφει την τιμή TRUE. Αν το **x** είναι TRUE, η συνάρτηση επιστρέφει την τιμή FALSE. Το επιλέγουμε ως εξής: **Functions palette<Boolean<Not**



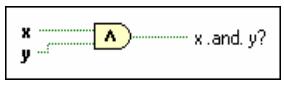
Or

Υπολογίζει το λογικό OR μεταξύ των εισόδων. Και οι δύο είσοδοι θα πρέπει να είναι Booleans ή numerics. Αν και οι δύο είσοδοι είναι FALSE, τότε επιστρέφεται η τιμή FALSE. Σε κάθε άλλη περίπτωση επιστρέφεται η τιμή TRUE. Το επιλέγουμε ως εξής: **Functions palette<Boolean<Or**



And

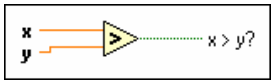
Υπολογίζει το λογικό AND μεταξύ των εισόδων. Και οι δύο είσοδοι θα πρέπει να είναι Booleans ή numerics. Αν και οι δύο είσοδοι είναι TRUE, τότε ε[πιστρέφεται η τιμή TRUE. Σε κάθε άλλη περίπτωση επιστρέφεται η τιμή FALSE. Το επιλέγουμε ως εξής: **Functions palette<Boolean<And**



b. Comparison Functions

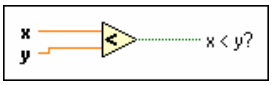
Greater

Επιστρέφει την τιμή TRUE αν το **x** είναι μεγαλύτερο από το **y**. Σε αντίθετη περίπτωση επιστρέφει την τιμή FALSE. Το επιλέγουμε ως εξής: **Functions palette<Comparison<Greater**



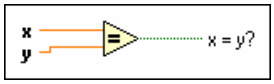
Less

Επιστρέφει την τιμή TRUE αν το **x** είναι μικρότερο από το **y**. Σε διαφορετική περίπτωση επιστρέφει την τιμή FALSE. Το επιλέγουμε ως εξής: **Functions palette<Comparison<Less**



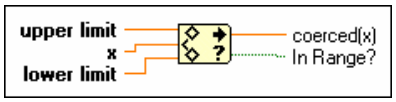
Equal

Επιστρέφει την τιμή TRUE αν το **x** είναι ίσο με το y. Σε αντίθετη περίπτωση επιστρέφει την τιμή FALSE. Το επιλέγουμε ως εξής: **Functions palette<Comparison<Equal?**



In Range and Coerce

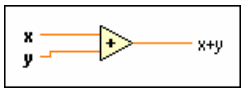
Σε αυτό το function θέτουμε στην είσοδο δύο όρια, ένα ανώτερο (**upper limit**) και ένα κατώτερο (**lower limit**) καθώς και την τιμή που θέλουμε να περάσουμε από το συγκεκριμένο function. Στην έξοδο παίρνουμε δύο τιμές, μια Boolean (**In Range?)** και μια αριθμητική (**coerced(x)**). Η Boolean τιμή είναι TRUE όταν η τιμή **x** είναι εντός των ορίων και FALSE όταν η τιμή **x** είναι εκτός αυτών των ορίων. Η αριθμητική τιμή της εξόδου είναι η ίδια η **x** αν είναι εντός των ορίων αλλά αν είναι εκτός αυτών των ορίων η τιμή **coerced(x)** παίρνει την τιμή του άνω ορίου αν **x>upper limit** και την τιμή του κάτω ορίου αν x<lower limit. Το επιλέγουμε ως εξής: Functions palette<Comparison<In Range and Coerse .



c. Numeric Functions

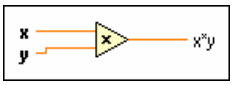
Add

Υπολογίζει το άθροισμα των εισόδων που δέχεται. Τα x και y μπορούν να είναι είτε αριθμοί είτε πίνακες αριθμών είτε ακόμα clusters αποτελούμενα από αριθμούς. Το x+y είναι το άθροισμά τους. Το επιλέγουμε ως εξής: **Functions palette<Numeric<Add**



Multiply

Υπολογίζει το γινόμενο των εισόδων που δέχεται. Τα x και y μπορούν να είναι είτε αριθμοί είτε πίνακες αριθμών είτε ακόμα clusters αποτελούμενα από αριθμούς. Το x*y είναι το γινόμενο τους. Το επιλέγουμε ως εξής: **Functions palette<Numeric<Multiply**

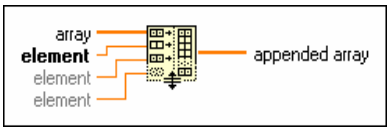


d. Array Functions

Build Array

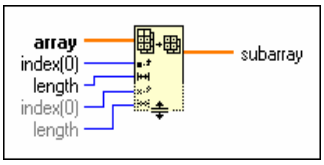
Ενοποιεί διαφορετικούς πίνακες ή εισάγει στοιχεία σε ν-διάστατους πίνακες. Με την επιλογή **Concatenate Inputs** η οποία μας δίνεται στο pop-up μενού του Build Array διαφοροποιείται η χρήση του. Έτσι έχοντας ενεργοποιημένη αυτή την επιλογή το αποτέλεσμα που μας δίνεται είναι ένας πίνακας ίδιων διαστάσεων με τους πίνακες εισόδου, ο οποίος όμως θα περιέχει όλα τα στοιχεία των πινάκων που θέσαμε σαν εισόδους. Για παράδειγμα αν σαν εισόδους χρησιμοποιούσαμε δύο μονοδιάστατους πίνακες με τα παρακάτω στοιχεία {1, 0, 8} και {3, 5, 6, 9}, τότε στην έξοδο θα παίρναμε τον πίνακα {1, 0, 8, 3, 5, 6, 9}. Αν η επιλογή **Concatenate Inputs** είναι απενεργοποιημένη, τότε στην έξοδο προκύπτει ένας πίνακας ο οποίος είναι κατά μία διάσταση μεγαλύτερος από τους πίνακες εισόδου. Για τους πίνακες εισόδου που χρησιμοποιήσαμε και στο προηγούμενο παράδειγμα το αποτέλεσμα θα ήταν ο δυδιάστατος πίνακας {{1, 0, 8, 0}, {3, 5, 6, 9}}.

Η επιλογή αυτή μας δίνεται μόνο όταν έχουμε σαν εισόδους πίνακες των ίδιων διαστάσεων γιατί σε διαφορετική περίπτωση η επιλογή ενεργοποιείται αυτόματα χωρίς να υπάρχει η δυνατότητα απενεργοποίησης της. Οι πίνακες εισόδου θα πρέπει να περιέχουν στοιχεία του ίδιου τύπου. Το επιλέγουμε ως εξής: **Functions palette<Array<Build Array**



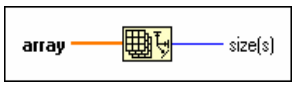
Array Subset

Επιστρέφει ένα μέρος ενός array ξεκινώντας από το στοιχείο που υποδεικνύει το **index** και περιέχοντας αριθμό στοιχείων που υποδεικνύεται από το **length.** Όταν συνδέουμε ένα array στο συγκεκριμένο function, τότε αυτό προσαρμόζεται αυτόματα έτσι ώστε να περιέχει τόσες **index** και **length** εισόδους όσες και οι διαστάσεις του array. Το επιλέγουμε ως εξής: **Functions palette<Array<Array Subset**



ArraySize

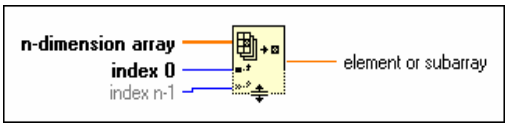
Επιστρέφει τον αριθμό τον στοιχείων που υπάρχουν σε κάθε διάσταση ενός **array**. Αν έχουμε πολυδιάστατο array τότε η έξοδος του function είναι ένας μονοδιάστατος πίνακας με τόσα στοιχεία όσες και οι διαστάσεις του array. Το επιλέγουμε ως εξής: **Functions palette<Array<Array Size**



Index Array

Επιστρέφει την τιμή ενός στοιχείου ή ενός υποπίνακα. Όταν συνδέουμε ένα πίνακα στο function τότε αυτόματα εμφανίζει τόσα index όσα και οι διαστάσεις του πίνακα. Για να μας επιστρέψει το function ένα συγκεκριμένο στοιχείο θα πρέπει να συνδέσουμε όλα τα Index και στη συνέχεια να επιλέξουμε το κατάλληλο στοιχείο. Αν θέλουμε ένα υποπίνακα τότε πρέπει να αφήσουμε χωρίς σύνδεση τα κατάλληλα index παίρνοντας έτσι όλα τα στοιχεία

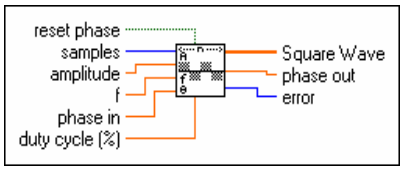
της συγκεκριμένης διάστασης. Το επιλέγουμε ως εξής: **Functions palette<Array<Index Array**



e. Analyze Functions

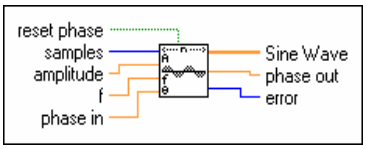
Square Wave

Δημιουργεί ένα array το οποίο περιέχει ένα τετραγωνικό παλμό. Μας δίνεται η δυνατότητα να επιλέξουμε της συχνότητα το πλάτος και άλλες παραμέτρους. Το επιλέγουμε ως εξής: **Functions palette<Analyze<Signal Processing<Signal Generation<Square Wave.VI**



Sine Wave

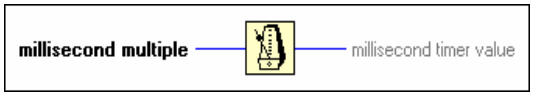
Δημιουργεί ένα array το οποίο περιέχει ένα ημιτονικό σήμα. Μας δίνεται η δυνατότητα να επιλέξουμε της συχνότητα το πλάτος και άλλες παραμέτρους. Το επιλέγουμε ως εξής: **Functions palette<Analyze<Signal Processing<Signal Generation<Sine Wave.VI**



f. Διάφορα Functions

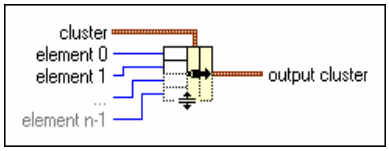
WaitUntilNextmsMultiple

Εισάγει καθυστέρηση (σε millisecond) ίση με την τιμή που επιλέγουμε ως **millisecond multiple**. Χρησιμοποιείται για τον συγχρονισμό διαφόρων εργασιών. Μπορούμε να το χρησιμοποιήσουμε για να ελέγξουμε το ρυθμό εκτέλεσης ενός βρόγχου. Το επιλέγουμε ως εξής: **Functions palette<Time & Dialog<Wait Until Next ms Multiple**



Bundle

Δημιουργεί ένα cluster από μεμονωμένα στοιχεία. Δίνεται επίσης η δυνατότητα να αλλαχθεί η τιμή κάποιων συγκεκριμένων στοιχείων ενός cluster χωρίς να χρειαστεί να οριστούν νέες τιμές για όλα τα στοιχεία. Αυτό επιτυγχάνεται συνδέοντας το cluster το οποίο θέλουμε να διαφοροποιήσουμε στο μεσαίο terminal. Το επιλέγουμε ως εξής: **Functions palette<Cluster<Bundle**



Κεφάλαιο 4

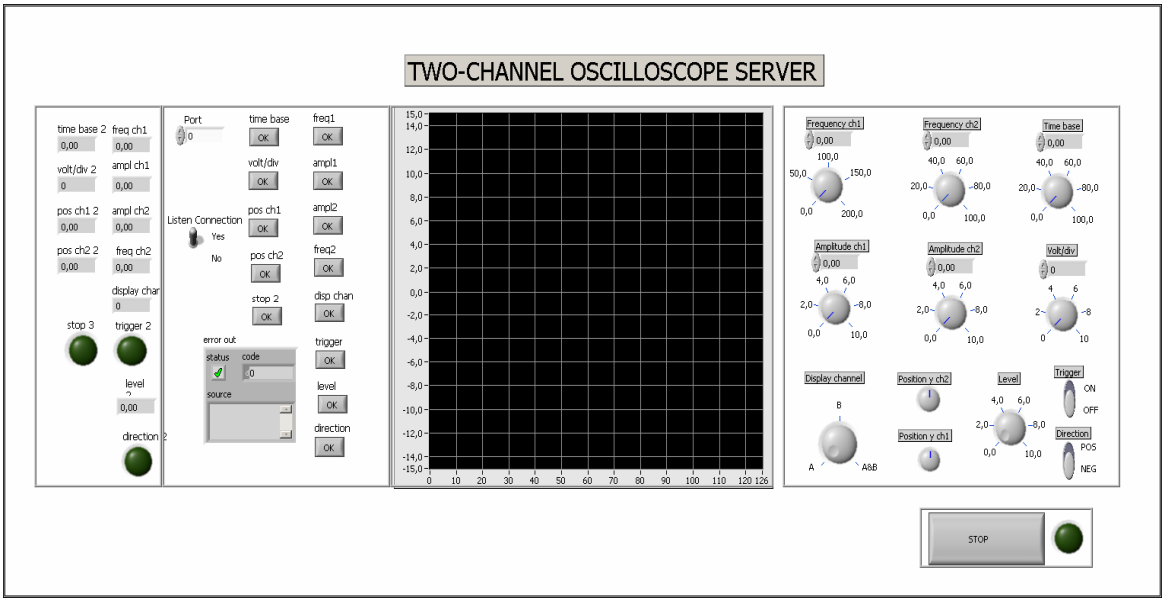
Σχεδίαση του διακομιστή (server)

Στο κεφάλαιο που ακολουθεί γίνεται η ανάλυση, του προγράμματος που αναπτύχθηκε, για να υλοποιηθεί ο διακομιστής (server).

Αρχικά γίνεται η παρουσίαση και ο τρόπος χρήσης του front panel του server. Έπειτα γίνεται αναφορά στο block diagram του server και αναλύονται τα επιμέρους VI που σχεδιάσθηκαν για την υλοποίηση του συγκεκριμένου προγράμματος.

4.1 Front Panel

Το front panel του Server παλμογράφου αποτελείται από τον παλμογράφο που αναπτύχθηκε στο προηγούμενο κεφάλαιο και από ένα σύνολο ελεγκτών και ενδείξεων που δίνουν την δυνατότητα στον χρήστη του Server να ελέγχει την προσπέλαση τον επιμέρους λειτουργιών του παλμογράφου.

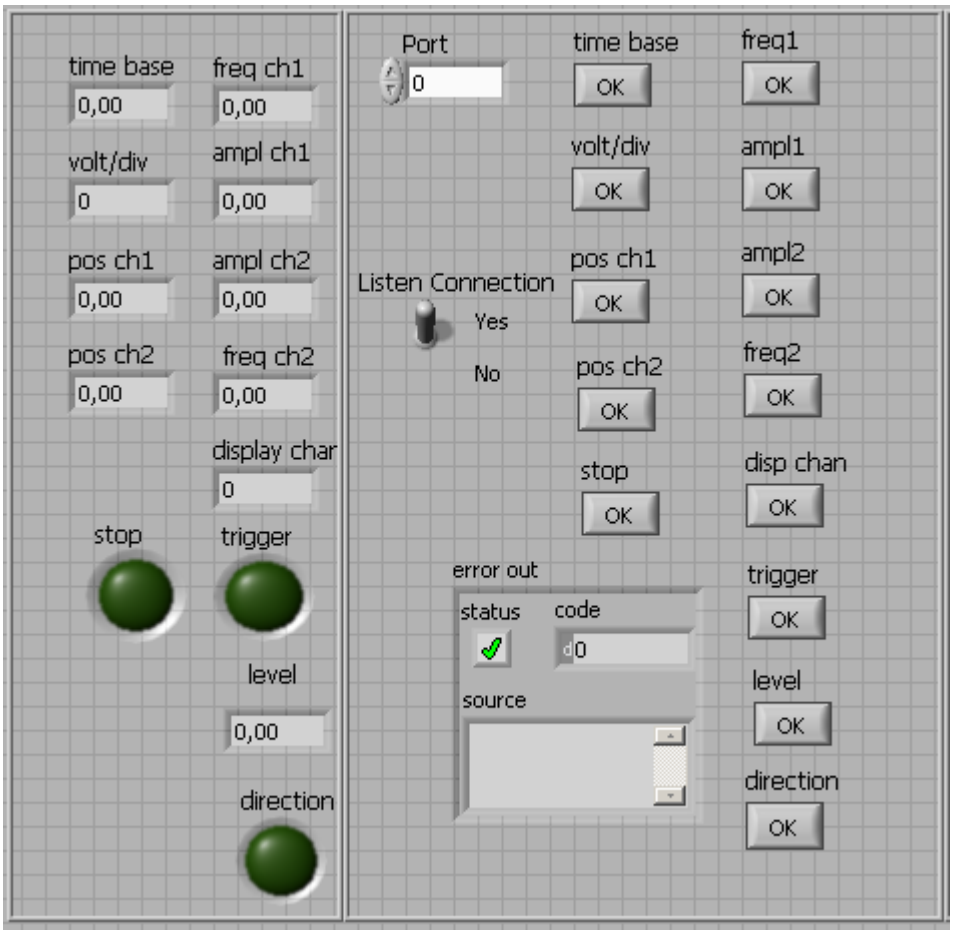


Εικόνα 4.1. Server

Τα controls που χρησιμοποιήθηκαν είναι αριθμητικά controls, ok buttons και toggle switches. Για indicators επιλέχθηκαν αριθμητικά indicators και led's. Τα αριθμητικά controls τα επιλέγουμε ως εξής: **controls palette<numeric controls<Num Ctrl.**Τα ok buttons από: **controls palette<Buttons& Switches<Ok Button** και τα toggle switches από: **controls palette< Buttons& Switches<Vertical toggle switch.**Τα LED's επιλέχ-θηκαν από: **controls palette<LED'S<round led**. Τα αριθμητικά indicators προκύπτουν από τα controls αν με δεξί κλικ πάνω στο control επιλεχθεί change to indicator. Στο front panel υπάρχει και ένα VI το οποίο εμφανίζει, σε περίπτωση σφάλματος, μηνύματα τα οποία περιγράφουν την φύση του προβλήματος που προέκυψε και ονομάζεται General Error Handler.VI.

Εικόνα 4.2. controls και indicators που χρησιμοποιήθηκαν

Στην επόμενη εικόνα παρουσιάζεται το τμήμα που αναπτύχθηκε ώστε ο παλμογράφος να λειτουργεί ως server.



Εικόνα 4.3.Τμήμα ελέγχου του server

Στο τμήμα αυτό έχουμε δυο πλαίσια έτσι ώστε να υπάρχει διαχωρισμός ανάμεσα στις ενδείξεις και τα controls. Το πλαίσιο με τις ενδείξεις μας πληροφορεί κάθε χρονική στιγμή τα δεδομένα που εισάγει ο χρηστής του client. Το πλαίσιο με τα controls δίνει την δυνατότητα στον χρήστη του Server να δίνει δικαίωμα προσπέλασης στις διάφορες λειτουργίες του παλμογράφου. Για να μπορεί να χειριστεί τις λειτουργίες του παλμογράφου ο χρήστης του Client θα πρέπει να έχει ενεργοποιήσει τις αντίστοιχες λειτουργίες ο χρήστης του Server,πατώντας τα αντίστοιχα κουμπιά (time base, volt/div, posch1, posch2, stop, freq1, freq2, ampl1, ampl2, dispchan, trigger, level, direction). Επίσης υπάρχει και ένα control με ονομασία port στο οποίο δίνεται η θύρα μέσω της οποίας θα γίνει η σύνδεση με τον client.Η θύρα θα πρέπει να είναι η ίδια και για τον Server όπως και για τον Client.Μέσω του διακόπτη Listen Connection επιλέγεται η κατάσταση του server μετά από μια αποτυχημένη σύνδεση με κάποιον Client,όταν βρίσκεται στην θέση Yes ο διακόπτης, ο Server

συνεχίζει να **π**ερι**μ**ένει νέα σύνδεση **μ**ε **Client** ενώ στην θέση **No** τερ**μ**ατίζει την λειτουργία του**.** Ο χρήστης του **server** έχει ουσιαστικά να ε**π**ιτελέσει τέσσερις ενέργειες**.**

Α**.**Το **π**άτη**μ**α του κου**μπ**ιού **run**

Β**.** Την εισαγωγή του αριθ**μ**ού της θύρας **μ**έσω της ο**π**οίας θα γίνει η ε**π**ικοινωνία**.**

Γ**.** Ε**π**ιλογή τον λειτουργιών **π**ου θα έχει **π**ροσ**π**έλαση ο χρήστης του **client.**

Δ**.** Ε**π**ιλογή του **Listen Connection**

4.2 Block Diagram

Η υλοποίηση του server στο Labview, με χρήση της γλώσσας G, παρουσιάζεται στην εικόνα 4. Για την ανάπτυξη του συγκεκριμένου κώδικα προγράμματος χρησιμοποιήθηκαν έτοιμες συναρτήσεις του Labview που θα αναλυθούν στο παράρτημα Α. Η ανάλυση του block diagram του VI θα γίνει αρχικά γενικά και έπειτα θα γίνει η ανάλυση των επιμέρους υποπρογραμμάτων (subVI's) που αναπτύχθηκαν για την υλοποίηση του Server.

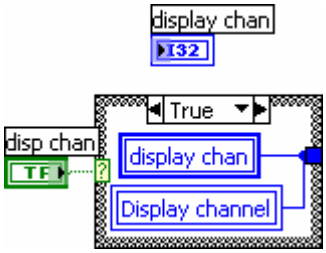
Η μεταβλητή Port είναι η πρώτη μεταβλητή η οποία χρησιμοποιείται και που πρέπει ο χρήστης του Server να εισάγει μέσω του Front Panel. Η συγκεκριμένη μεταβλητή εισέρχεται σε ένα While Loop του οποίου η συνθήκη αλήθειας ελέγχεται από την μεταβλητή Listen continue. Εάν αυτή είναι true, ο server συνεχίζει την λειτουργία του με στόχο την νέα σύνδεση με client εφόσον υπάρξει αποτυχία στην υπάρχων σύνδεση (εμφάνιση σφάλματος κατά την επικοινωνία Server-Client, εισαγωγή λάθους κωδικού από τον χρήστη του client, τερματισμός της επικοινωνία από τον client, δεν έχει εμφανιστεί αίτηση σύνδεσης σε κάποιο προσδιορισμένο χρόνο). Σε περίπτωση που η μεταβλητή Listen continue έχει την τιμή false ο Server τερματίζει την λειτουργία του στην πρώτη εμφάνιση σφάλματος.

Στη συνέχεια η μεταβλητή Port εισέρχεται σε νέο While Loop όπου χρησιμοποιείται ως είσοδος του Connection VI. Εφόσον υπάρξει επιτυχείς σύνδεση με τον Client η έξοδοι του Con.VI μεταβιβάζονται στο επόμενο While Loop που ελέγχεται από την τιμή του stop button του παλμογράφου. Εάν αυτή είναι false (εισέρχεται σε μια Not συνάρτηση) ο έλεγχος παραμένει στο συγκεκριμένο Loop.Όπως φαίνεται από την εικόνα 4 οι έξοδοι του Con.VI είναι είσοδοι στο Communication.VI, που αναπτύχθηκε για την μεταφορά των δεδομένων. Σε κάθε επανάληψη του βρόγχου αποστέλλονται αρχικά οι τιμές του γράφου του server και έπειτα γίνεται μεταφορά των στοιχείων που αποστέλλει ο client στον παλμογράφο( ) που αναπτύχθηκε στο κεφάλαιο 2.

Οι έξοδοι του Com.vi αποθηκεύονται στης μεταβλητές time base, volt/div, pos ch1, pos ch2, stop, freq ch1, freq ch2, ampl ch1, ampl ch2, display chan, trigger, level και direction

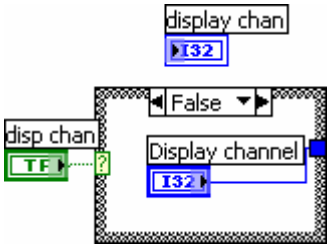
που εμφανίζονται στο αριστερό άκρο του Front Panel του Server. Για κάθε μια από αυτές τις μεταβλητές έχουν δημιουργηθεί αντίστοιχες τοπικές μεταβλητές που έχουν τοποθετηθεί ξεχωριστά μέσα σε δομές case.

Παρακάτω θα γίνει η ανάλυση της χρήσης μίας τοπικής μεταβλητής αφού οι υπόλοιπες λειτουργούν με παρόμοιο τρόπο. Στην εικόνα 4.5 παρουσιάζεται η χρήση της τοπικής μεταβλητής display chan.



Εικόνα 4.5.Χρήση τοπικής μεταβλητής display chan.

Όπως προαναφέρθηκε η τοπική μεταβλητή display chan, που διαβάζει δεδομένα, τοποθετείται μέσα σε μία case δομή της οποίας η συνθήκη ελέγχεται από την λογική μεταβλητή disp chan. Η μεταβλητή αυτή αντιστοιχεί στο κουμπί του Front Panel disp chan και έχει αρχική τιμή false. Όταν ο χρήστης του Server δώσει δικαίωμα πρόσβασης η μεταβλητή disp chan έχει τιμή true και η δομή case εκτελεί την λειτουργία της εικόνας 4.5. Εσωτερικά της δομής έχει τοποθετηθεί η προαναφερθέν τοπική μεταβλητή όπως και η τοπική μεταβλητή Display channel που γράφει δεδομένα. Η τοπική μεταβλητή Display channel(είναι στο δεξί μέρος του Front Panel του Server τοποθετημένη) αναφέρεται στην αντίστοιχη μεταβλητή του παλμογράφου με αποτέλεσμα να μεταβιβάζει την τιμή που έχει αποστείλει ο client στον παλμογράφο. Οι έξοδοι όλων των δομών case συνδέονται με τους αντίστοιχους ακροδέκτες του παλμογράφου. Με αυτόν τον τρόπο με την έξοδο από όλες τις δομές case ο παλμογράφος έχει πλέον ως εισόδους τις τιμές που έχει στείλει ο Client.Ταυτόχρονα τα δεδομένα αυτά απεικονίζονται στο Front Panel του Server. Σε περίπτωση που η τιμή της λογικής μεταβλητής disp chan είναι false τότε έχουμε το block diagram της εικόνας 4.6.



Εικόνα 4.6.Η λογική μεταβλητή disp chan είναι false

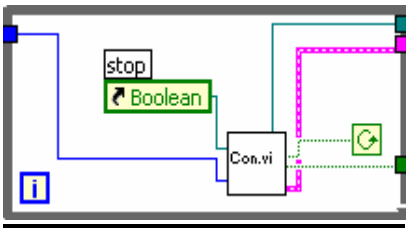
Όπως φαίνεται από την εικόνα 6 σε αυτήν την περίπτωση μεταβιβάζεται στον παλμογράφο η τιμή που έχει η μεταβλητή Display channel και είναι η τιμή που εισάγει ο χρήστης του Server μέσω του Front Panel.

Ο παλμογράφος αφού εκτελεστεί επιστρέφει στον γράφο την ζητούμενη κυματομορφή. Τα δεδομένα του γράφου εισάγονται στην συνάρτηση Flatten To String( ) και το string που δημιουργείται χρησιμοποιείται ως είσοδος του Com.VI με αποτέλεσμα να αποστέλλονται τα δεδομένα του γράφου στον Client.

Όταν γίνει έξοδος από το Loop ο έλεγχος του προγράμματος επανέρχεται στο εξωτερικό Loop.Η συνάρτηση TCP Close Connection έχει ως εισόδους τον αριθμό αναφοράς της δικτυακής σύνδεσης και το error string που της μεταβιβάζει το Com.VI.Σε αυτό το σημείο γίνεται η διακοπή της σύνδεσης με τον Client και το error string μεταφέρεται στο General Error Handler.VI που θα εμφανίσει στο Front Panel το είδος του σφάλματος που προέκυψε. Παρακάτω γίνεται η ανάλυση των υποπρογραμμάτων που σχεδιάσθηκαν .

4.2.1 Connection.VI

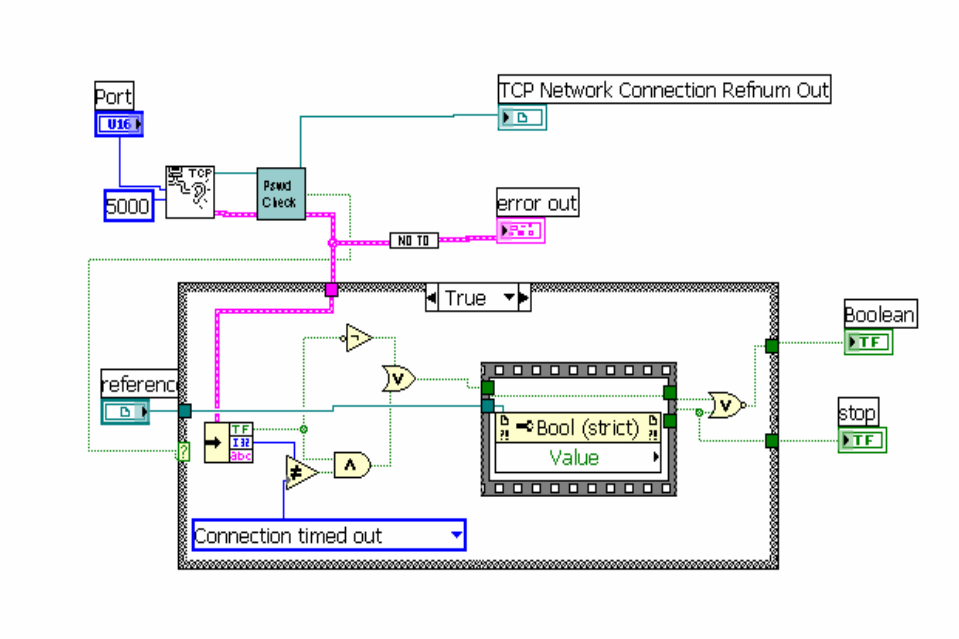
Το συγκεκριμένο VI έχει ως βασική λειτουργία την δημιουργία της σύνδεσης με τον client όταν αυτή ζητείται και ταυτόχρονα εκπληρώνονται κάποιες συγκεκριμένες προϋπόθεσης .Η προϋπόθεσης αυτές συσχετίζονται με την φύση της επικοινωνίας που ζητά ο Client όπως και με την επαλήθευση κάποιου κωδικού που εισάγεται από τον χρήστη του Client έτσι ώστε να είναι εφικτή η σύνδεση του με τον Server.



Εικόνα 4.7.While loop του Con.VI

Η εικόνα 7 απεικονίζει τον βρόχο ελέγχου του Connection.VI.Το συγκεκριμένο VI έχει ως εισόδους την θύρα μέσω της οποίας θα γίνει η σύνδεση με τον Client και μια αναφορά της κατάστασης του stop button. Το Con.VI ελέγχει την συνθήκη αλήθειας του βρόχου

με αποτέλεσμα να ελέγχει την επικοινωνία του Server με τον client. Όταν η έξοδος του Con.VI είναι false ο έλεγχος μεταφέρεται στο επόμενο While loop, ενώ στην κατάσταση true έλεγχος παραμένει στο συγκεκριμένο loop. Σε κάθε επανάληψη του βρόχου έχουμε ως εξόδους των αριθμό αναφοράς της δικτυακής σύνδεσης και το error string.Ακολουθεί το block diagram του Connection.VI.



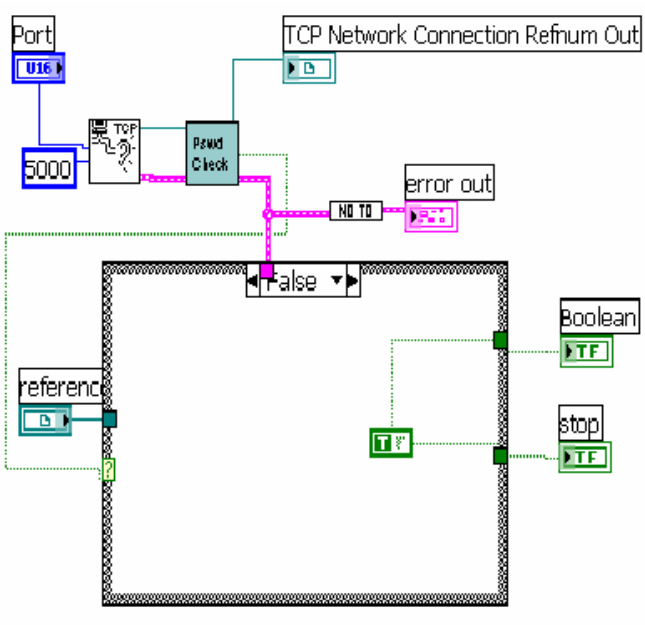
Εικόνα 4.8.Block diagram του Con.VI

Αρχικά εισάγεται στο TCP Listen VI , ο ακέραιος αριθμός της θύρας και ο μέγιστος χρόνος αναμονής του VI για αίτηση σύνδεσης με τον server. Με το πέρας αυτού του χρονικού διαστήματος εμφανίζεται μήνυμα λάθους και τερματίζεται η λειτουργία του server (το Listen connection είναι στο No,αλλιώς επανέρχεται στο συγκεκριμένο Loop και αναμένει νέα σύνδεση client). Σε περίπτωση αίτησης σύνδεσης μέσα στο προαναφερθέν χρονικό διάστημα ,έχουμε σύνδεση με τον Client και ο έλεγχος του προγράμματος μεταφέρεται στο Password Check.VI . Στο συγκεκριμένο πρόγραμμα συνδέονται οι ακροδέκτες του αριθμού αναφοράς της δικτυακής σύνδεσης και το error string. Εφόσον γίνει η εκτέλεση του προγράμματος επιστρέφεται μια λογική τιμή, το error string και τον αριθμό αναφοράς της δικτυακής σύνδεσης.Η λογική τιμή που επιστρέφεται είναι true αν ο χρήστης του client έχει εισάγει σωστό κωδικό και false σε κάθε άλλη περίπτωση. Στη συνέχεια το error string εισέρχεται στο πρόγραμμα No Time OuT Error VI το οποίο ελέγχει εάν έχει εμφανιστεί ένα connection timed out error και το κάνει reset,αφού ο έλεγχος του συγκεκριμένου σφάλματος θα γίνει παρακάτω. Έπειτα επιστρέφεται το error string και ο έλεγχος μεταφέρεται στην case δομή της οποίας η συνθήκη ελέγχεται από την λογική τιμή της εξόδου του Password Check VI.

Σε περίπτωση που η λογική τιμή είναι true έχουμε το block diagram της εικόνας 8.
Το error string το εισάγεται στην συνάρτηση Unbundle που διαχωρίζει το στοιχεία από τα οποία είναι δομημένο.

Εφόσον έχει διαχωριστεί το error string η τιμή του code (IR) συγκρίνεται με την τιμή της ring σταθερά Connection timed out και εάν δεν είναι ίση επιστρέφεται η τιμή true ,αλλιώς επιστρέφεται η τιμή false σε περίπτωση ισότητας. Η Boolean τιμή που συμπεριλαμβάνεται στο error string είναι true σε περίπτωση που έχει εμφανιστεί σφάλμα κατά την διάρκεια της σύνδεσης ,διαφορετικά είναι false.Αυτή η τιμή μαζί με το αποτέλεσμα της προηγούμενης σύγκρισης εισάγονται σε μια συνάρτηση And.Ταυτόχρονα η λογική τιμή εισάγεται σε μια συνάρτηση Not .Έπειτα τα αποτελέσματα των δυο συγκρίσεων εισάγονται σε μια συνάρτηση OR . Το αποτέλεσμα αυτόν των διαδοχικών συγκρίσεων είναι να υπάρχει στην έξοδο η τιμή false μόνο στην περίπτωση που θα εμφανιστεί το σφάλμα connection timed out και σε κάθε άλλο γεγονός το αποτέλεσμα να έχει την τιμή true. Ακολουθεί μία δομή Stacked Sequence στην οποία εισέρχονται το αποτέλεσμα των συγκρίσεων και η αναφορά της κατάστασης του stop button. Η αναφορά αυτή εισέρχεται σε ένα Property Node όπου γίνεται ανάγνωση της τιμής του stop button. Με την έξοδο από την δομή εισέρχονται οι τιμές του stop button και των συγκρίσεων που προηγήθηκαν σε μια συνάρτηση Not Or .Η έξοδος της συνάρτησης μεταβιβάζεται στο βρόγχο ελέγχου του Con.VI (εικόνα 4.7) και με συνέπεια η τιμή αυτή να ελέγχει την επικοινωνία μεταξύ Server και Client.Όταν δεν εμφανίζεται το σφάλμα connection timed out και το stop button έχει τιμή false(δεν είναι πατημένο), έχει ως έξοδο το Con.VI την τιμή false με αποτέλεσμα τον τερματισμό του While Loop και την μεταφορά του ελέγχου του προγράμματος στο επόμενο βρόγχο. Όταν εμφανιστεί το συγκεκριμένο σφάλμα ο έλεγχος του προγράμματος παραμένει στο συγκεκριμένο Loop με σκοπό νέας σύνδεσης με τον Client. Οι συγκρίσεις έγιναν με σκοπό τον εντοπισμό το σφάλματος connection timed out όπως και για τερματισμό το βρόγχου του Con,VI όταν πατηθεί το stop button.

Έγινε η ανάλυση της ροής του προγράμματος όταν η λογική τιμή της εξόδου του Password Check.VI είναι true. Σε περίπτωση όμως που είναι false έχουμε το block diagram της εικόνας 4.9.

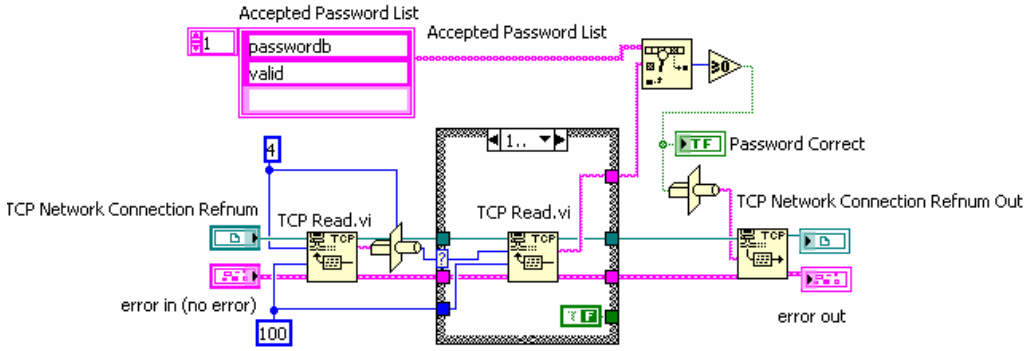


Εικόνα 4.9.Η έξοδος του Password Check είναι false

Σε αυτήν την περίπτωση τίθενται και οι δυο λογικές τιμές στην κατάσταση true με συνέπεια να έχουμε ως έξοδο του Con.VI την τιμή true,οπότε να παραμένει ο έλεγχος του προγράμματος στο συγκεκριμένο While Loop αναμένοντας νέα σύνδεση με τον Client.

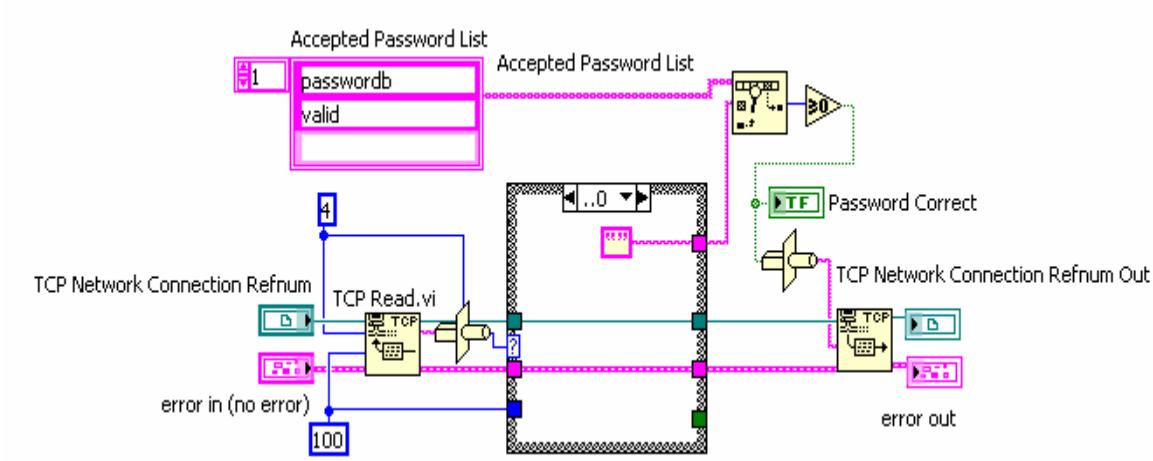
4.2.1.1 Password Check.VI

Το συγκεκριμένο υποπρόγραμμα αναπτύχθηκε για την ασφάλεια του συστήματος .Έτσι για να γίνει εφικτή η σύνδεση με τον Server θα πρέπει ο χρήστης του Client να κατέχει κάποιον κωδικό ασφάλειας, ο οποίος εισάγεται στον Client, με αποτέλεσμα την ταυτοποίηση του στο Password Check.VI με τους υπάρχον αποδεκτούς κωδικούς στο υποπρόγραμμα.Το block diagram απεικονίζεται στην εικόνα 4.10.



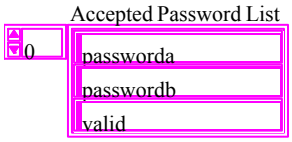
Εικόνα 4.10. Block diagram του Password Check

Οι είσοδοι του συγκεκριμένου VI είναι δυο ο αριθμός αναφοράς της δικτυακής σύνδεσης και το error string. Αρχικά γίνεται ανάγνωση του μήκους του κωδικού που έχει εισάγει ο χρήστης του Client. Αυτό επιτυγχάνεται με την χρήση της συνάρτησης TCP Read. Στους ακροδέκτες της συγκεκριμένης συνάρτησης συνδέονται ο αριθμός αναφοράς της δικτυακής σύνδεσης, το error string, ο αριθμός των byte που θα διαβαστούν και ο μέγιστος χρόνος αναμονής ανάγνωσης δεδομένων (^{100} σε ms ,έπειτα εμφανίζεται μήνυμα σφάλματος). Η έξοδος συνδέεται με την συνάρτηση Type Cast ,όπου στον ακροδέκτη type συνδέεται ένας ακέραιος ^{4}) που δίνει το μήκος του κωδικού ως ακέραια τιμή. Ακολουθεί μία case δομή που ελέγχεται από την τιμή του μήκους του κωδικού που έχει εισαχθεί. Έτσι για μήκος κωδικού ένα και μεγαλύτερο έχουμε το διάγραμμα που απεικονίζεται στην εικόνα 4.10 ,όπου γίνεται ανάγνωση του κωδικού που έχει εισάγει ο χρήστης του client μέσω της συνάρτησης TCP Read με την σύνδεση του μήκους του κωδικού με τον ακροδέκτη Bytes to read της συνάρτησης. Οι έξοδοι της δομής case είναι ο αριθμός αναφοράς της δικτυακής σύνδεσης, το error string ο κωδικός που έχει εισάγει ο χρήστης του Client.Στην περίπτωση που δεν έχει εισάγει ο χρήστης του Client κωδικό επιλέγεται από την δομή case το διάγραμμα ..0.



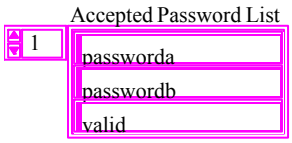
Εικόνα 4.11.Case δομή ..0

Όπως φαίνεται από την εικόνα 11 σε αυτήν την περίπτωση έχουμε ως εξόδους της δομής case τον αριθμό αναφοράς της δικτυακής σύνδεσης, το error string και μια μηδενική σταθερά τύπου string. Η σταθερά αυτή χρησιμοποιείται γιατί η χρήση της συνάρτησης TCP Read με 0 συνδεδεμένο στον ακροδέκτη bytes to read δεν έχει νόημα εφόσον η έξοδος δεν θα περιέχει τίποτα. Το επόμενα βήμα είναι η χρησιμοποίηση της συνάρτησης Search 1D array όπου εισάγονται ο κωδικός και ένα string που περιέχει τους αποδεκτούς κώδικες. Αυτό το string δημιουργείται από τον μονοδιάστατο πίνακα της εικόνας 4.12.



Εικόνα 4.12.Μονοδιάστος πίνακας με στοιχεία τους αποδεκτούς κώδικες

Ο μονοδιάστατος πίνακας έχει στοιχεία τους αποδεκτούς κώδικες που εισάγει ο χρήστης του Server με απλό αριστερό πάτημα του ποντικιού και γράφοντας τον επιθυμη-τό κωδικό. Ο πίνακας δημιουργείται με χρήση του array constant και εισάγοντας εντός αυτού του πίνακα μια σταθερά string . Για να εισαχθούν περισσότερα του ενός στοιχειά επιλέγεται ένα άκρο του πίνακα και ανάλογα το επιθυμητό μήκος, με αρι-στερό πάτημα του ποντικιού αυξάνεται η μειώνεται ο αριθμός των στοιχείων. Επίσης μπορούν να δημιουργηθούν περισσότεροι τέτοιοι πίνακες, με τον ίδιο αριθμό στοιχεί-ων πάντα ,αυξάνοντας τον δείκτη που βρίσκεται στο πάνω αριστερό άκρο του πίνακα έχοντας έτσι την επιλογή διαφόρων set κωδικών(εικόνα 4.13).



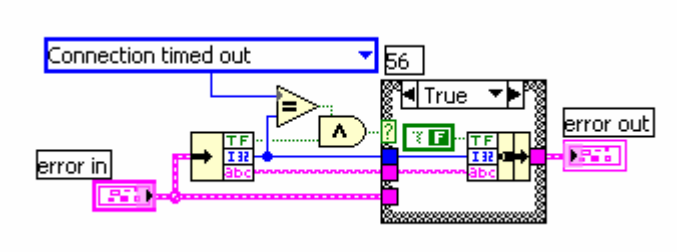
Εικόνα 4.13.Παράδειγμα χρήσης set κωδικών

Με την χρήση της συνάρτησης Search 1D array γίνεται γραμμική αναζήτηση του κωδικού, που έχει εισάγει ο χρήστης του client, μέσα στο string με τους αποδεκτούς κωδικούς. Το αποτέλεσμα της συνάρτησης είναι ο αριθμός της θέσης του προς αναζήτησης στοιχειού μέσα στον πίνακα. Αν δεν υπάρχει μέσα στον πίνακα επιστρέφεται η τιμή -1. Το αποτέλεσμα της αναζήτησης εισάγεται στην συνάρτηση Greater Or Equal To 0 με έξοδο true αν είναι μεγαλύτερο ή ίσο από 0 και false σε περίπτωση που είναι αρνητικό.

Στην συνέχεια η τιμή αυτή δίνεται στην έξοδο του VI και ταυτόχρονα μετατρέπεται σε string,μέσω της συνάρτησης Type Cast, με σκοπό την αποστολή της στον client. Αυτό επιτυγχάνεται με χρήση του TCP Write με αποτέλεσμα την ενημέρωση του χρήστη του Client για αποδοχή ή απόρριψη του κωδικού που έχει εισάγει ο ίδιος.

4.2.1.2 No Time Out Error.VI

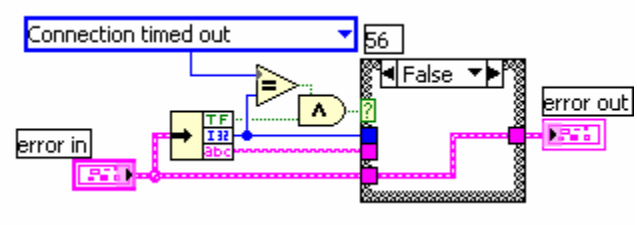
Η λειτουργία του VI είναι ο εντοπισμός του σφάλματος Connection timed out error. Σε περίπτωση που εντοπιστεί το συγκεκριμένο σφάλμα τροποποιεί το error string και το θέτει σε no time out error. Ακολουθεί το block diagram του VI.



Εικόνα 4.14.Block diagram του No Time Out Error. VI.

Το VI δέχεται ως είσοδος το error string και το εισάγει σε μία συνάρτηση Unbundle . Από το error string επιλέγεται ο ακέραιος αριθμός IR, που χαρακτηρίζει το είδος του σφάλματος(με τον αριθμό 56) και μεταφέρεται σε μία συνάρτηση Equal και συγκρίνεται με το connection timeout και επιστρέφεται η τιμή true σε περίπτωση ισότητας.
Στη συνέχεια το αποτέλεσμα της σύγκρισης μεταβιβάζεται σε μία συνάρτηση AND όπου συνδέεται και ο ακροδέκτης TF. Το αποτέλεσμα της συνάρτησης είναι true μόνο σε περίπτωση που εμφανιστεί το σφάλμα connection timed out με συνέπεια η δομή case να γίνεται true (εικόνα 4.14). Μέσα στην δομή ενώνονται τα στοιχεία που πριν διαχωρίστηκαν μέσω της συνάρτησης Bundle με διαφορά την τοποθέτηση της λογικής μεταβλητής TF στην κατάσταση false.Η έξοδος του VI είναι το error string χωρίς το σφάλμα Connection timed out.

Αν οι συγκρίσεις έχουν αποτέλεσμα false (δεν έχει εμφανιστεί το σφάλμα Connection timed out) η δομή case παίρνει την μορφή της εικόνας 4.15.

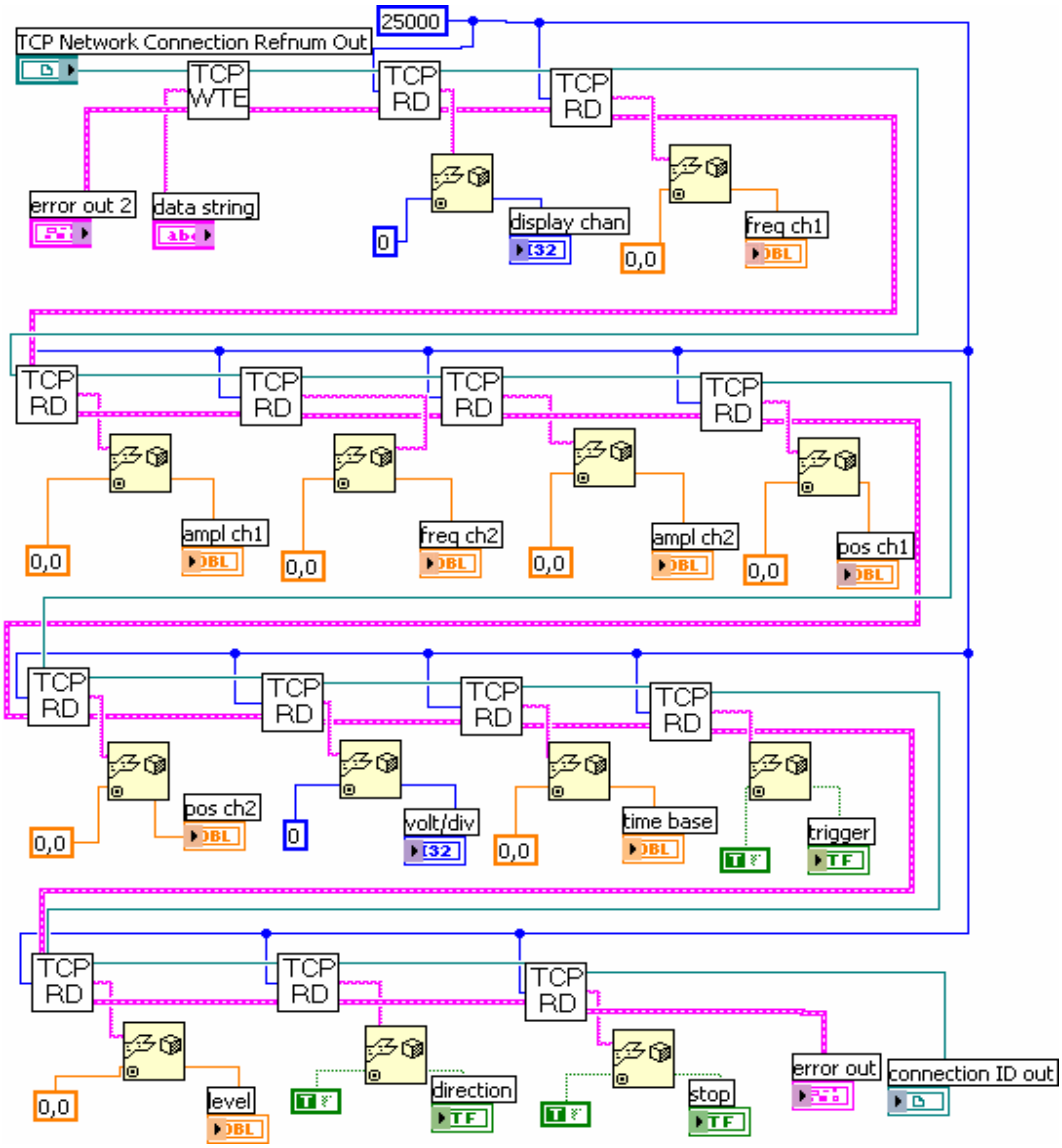


Εικόνα 4.15.Δομή case στην περίπτωση false

Σε αυτήν την περίπτωση στην έξοδο μεταβιβάζεται χωρίς επεξεργασία η είσοδος error in.

4.2.2 Communication.VI

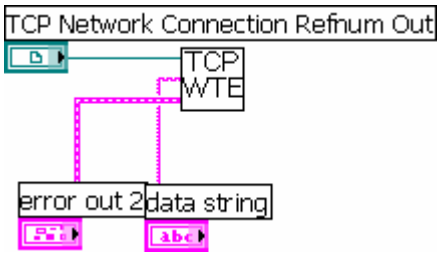
Το συγκεκριμένο VI επιτυγχάνει την αμφίδρομη επικοινωνία μεταξύ του Server και του Client. Η δομή του είναι απλή και έχει ως βασικές δομές του block διαγράμματος του δυο subVI's το TCP_IP Read και το TCP_IP Write που θα αναλυθούν παρακάτω. Το Com VI έχει τρεις εισόδους (τον αριθμό αναφοράς της δικτυακής σύνδεσης, το error string και το string δεδομένων από τον γράφο) και δεκαπέντε εξόδους που αντιστοιχούν στα δεκατρία κουμπιά του Client που ο χρήστης του μπορεί να μεταβάλλει, τον αριθμό αναφοράς της δικτυακής σύνδεσης και το error string. Στην εικόνα 4.16 έχουμε το συνολικό block diagram Com.VI.



Εικόνα 4.16.Το block diagram του com.VI

Από την εικόνα 4.16 γίνεται εύκολα κατανοητό η λειτουργία του συγκεκριμένου υποπρογράμματος. Η επικοινωνία επιτυγχάνεται με τα δυο υποπρογράμματα TCP_IP Write(για να αποστέλλονται δεδομένα) και TCP_IP Read (για λήψη δεδομένων) και με την κατάλληλη επεξεργασία των δεδομένων. Όταν εκτελείται το Com.VI αποστέλλονται αρχικά τα δεδομένα που περιέχει ο γράφος του Server στον γράφο του client. Έπειτα γίνεται ανάγνωση των δεδομένων που αποστέλλει ο client. Τα δεδομένα που λαμβάνονται έχουν ακριβώς την ίδια διαδοχή με τα δεδομένα που αποστέλλονται από τον client έτσι ώστε να είναι δυνατή η αντιστοίχηση τους με τα αντίστοιχα indicators του server. Στο τέλος της εκτέλεσης του Com.VI έχουν ενημερωθεί όλα τα αναγκαία δεδομένα που απαιτείται για την σωστή επικοινωνία του Server με τον Client. Παρακάτω γίνεται η ανάλυση της αποστολής και λήψης των δεδομένων.

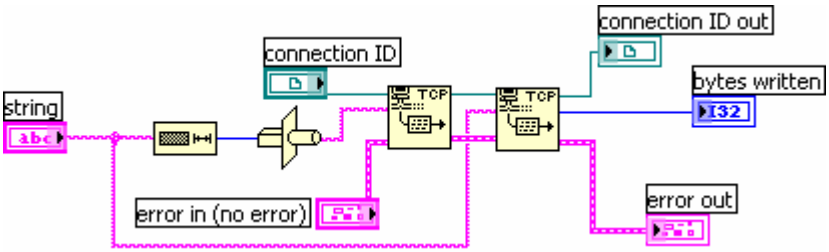
4.2.2.1 Αποστολή δεδομένων



Εικόνα 4.17.Αποστολη δεδομένων

Στην εικόνα 17 απεικονίζεται η αποστολή των δεδομένων του γράφου στον client .Η χρήση του TCP_IP Write απλοποιεί την διαδικασία αφού η δομή του είναι απλή. Έτσι για να γίνει η αποστολή των δεδομένων αρκεί να είναι γνωστή ο αριθμός αναφοράς της δικτυακής σύνδεσης μέσω της οποίας θα αποσταλούν τα δεδομένα, το error string και τα δεδομένα του γράφου να είναι διαθέσιμα. Συνδέοντας τα συγκεκριμένα στοιχειά στους αντίστοιχους ακροδέκτες του TCP_IP Write γίνεται εφικτή η αποστολή των δεδομένων .Στη συνέχεια γίνεται η ανάλυση του TCP_IP Write.

TCP_IP Write.VI

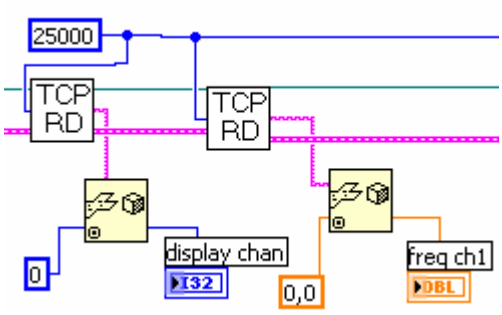


Εεικόνα 4.18.Block diagram του TCP_IP Write

Η εικόνα 4.18 απεικονίζει το block diagram του TCP_IP Write. Όταν γίνει η εκτέλεση του TCP_IP Write το string που περιέχει τα δεδομένα του γράφου εισάγεται στην αρχή του προγράμματος και εισέρχεται στο String length που επιστρέφει τον αριθμό των χαρακτήρων ,από τους οποίους αποτελείται το string. Έπειτα ο αριθμός μετατρέπεται σε string, έτσι ώστε να είναι εφικτή η αποστολή του μέσω του TCP Write. Αυτό επιτυγχάνεται μέσω του Type Cast που μετατρέπει τον τύπο δεδομένων της εισόδου του σε όποιο τύπο

δεδομένων επιλέξουμε (default μετατρέπει την είσοδο σε string). Το επόμενο βήμα είναι να αποσταλεί αυτό το string μέσω του TCP Write. Το συγκεκριμένο πρόγραμμα απαιτεί τον αριθμό αναφοράς της δικτυακής σύνδεσης μέσω της οποίας θα σταλεί το string όπως και το error string,τα οποία είναι διαθέσιμα αφού αποτελούν είσοδοι του TCP_IP Write. Σε αυτό το σημείο έχει ήδη αποσταλεί το μήκος του string στον client οπότε ακολουθεί στη συνέχεια και η αποστολή του string με τα δεδομένα του γράφου όπως φαίνεται από την εικόνα 4.18. Τέλος ως εξόδους του TCP Write έχουμε τον αριθμό αναφοράς της δικτυακής σύνδεσης, το error string και τον αριθμό των bytes που απεστάλησαν.

4.2.2.2 Λήψη δεδομένων

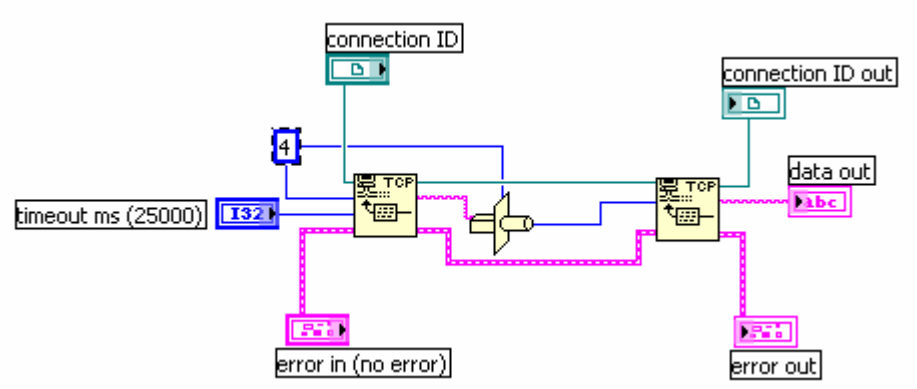


Εικόνα 4.19.Λήψη δεδομένων από client

Στην εικόνα 4.19 απεικονίζεται η λήψη των δεδομένων από τον client. Η δομή αυτή εξασφαλίζει ασφαλή επικοινωνία με τον client και ακριβής λήψη των δεδομένων που θα απεικονιστούν στις ενδείξεις του Server (τα indicators είναι τοποθετημένα στο αριστερό άκρο του Server). Αρχικά εισάγεται στο υποπρόγραμμα TCP_IP Read ένας σταθερός ακέραιος(η χρησιμότητα του θα επισημανθεί παρακάτω) και έπειτα γίνεται η ανάγνωση των δεδομένων που αποστέλλει ο client από το υποπρόγραμμα TCP_IP Read. Η έξοδος του TCP_IP Read είναι ένα string το οποίο εισάγουμε στην συνάρτηση Unflatten From String η οποία μετατρέπει το string στον τύπο δεδομένων που έχουμε συνδέσει στον ακροδέκτη type(κάτω αριστερά της συνάρτησης Unflatten From String π.χ. ^{0} , δηλαδή σε ακέραιο για το πρώτο TCP_IP Read,π.χ. ^{0,0} ,δεκαδικός για το δεύτερο TCP_IP Read).Με αυτόν τον τρόπο γίνεται η μεταφορά των δεδομένων που απεστάλησαν από τον Client στον Server (στην εικόνα 4.19 μετά το πρώτο TCP_IP Read έχει ενημερωθεί η ένδειξη display chan2). Η διαδικασία αυτή εφαρμόζεται και για τις υπόλοιπες ενδείξεις όπως φαίνεται στην εικόνα 4.16 ,συνδέοντας τους κατάλληλους ακροδέκτες μεταξύ τους(αριθμός αναφοράς της δικτυακής σύνδεσης, error string,connection timeout).

TCP_IP Read.VI

Σε αυτό το κεφάλαιο θα γίνει η ανάλυση του TCP_IP Read που εξασφαλίζει την λήψη των δεδομένων που εστάλησαν από τον client.Στην εικόνα 20 δίδεται το block diagram του subVI.



Εικόνα 4.20.Block diagram του TCP_IP READ

Στην αρχή της εκτέλεσης του TCP_IP Read παρατηρούμε την κλήση της συνάρτησης TCP Read που διαβάζει τα δεδομένα που αποστέλλει ο client. Αυτή η συνάρτηση έχει ως ορίσματα, τον αριθμό αναφοράς της δικτυακής σύνδεσης, έναν ακέραιο που δηλώνει πόσα bytes θα διαβάσει , το χρονικό διάστημα σε ms στο οποίο θα λαμβάνει δεδομένα (μετά το πέρας αυτού το διαστήματος μεταδίδει μήνυμα σφάλματος μέσω του error string) και το error string.Το πρώτο TCP Read μας δίνει ως έξοδος το string, που περιέχει το μήκος του string δεδομένων (βλέπε λειτουργία του TCP_IP Write),που εστάλη από τον client. Έπειτα γίνεται η ανάγνωση του string δεδομένων ,εφόσον η έξοδος του πρώτου TCP Read έχει μετατραπεί σε ακέραιο μέσω του Type Cast , με ορίσματα τα ίδια με πριν. Στην έξοδο του TCP_IP Read έχουμε τα δεδομένα που εστάλησαν, τον αριθμό αναφοράς της δικτυακής σύνδεσης και το error string.

Κεφάλαιο 5

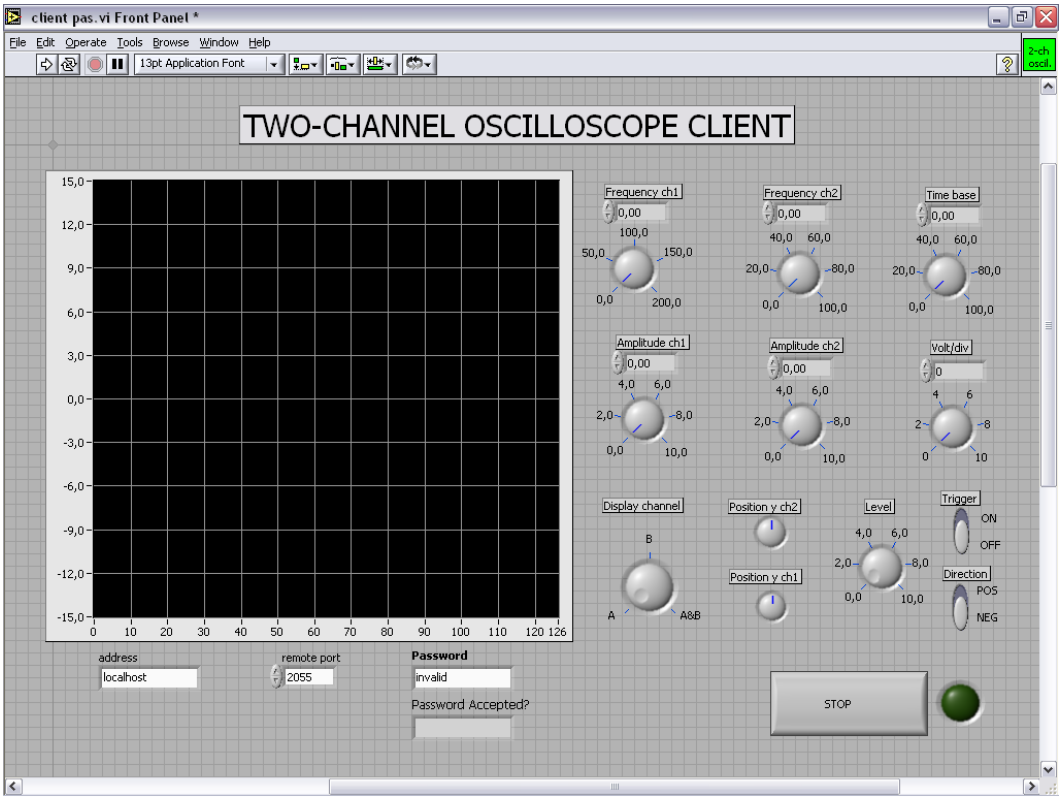
Σχεδίαση πελάτη (client)

Στο κεφάλαιο που ακολουθεί γίνεται η ανάλυση, του προγράμματος που αναπτύχθηκε, για να υλοποιηθεί ο πελάτης του εικονικού εργαστηρίου. Αρχικά γίνεται η παρουσίαση και ο τρόπος χρήσης του front panel του client. Έπειτα γίνεται αναφορά στο block diagram του client και αναλύονται τα επιμέρους VI που σχεδιάσθηκαν για την υλοποίηση του συγκεκριμένου προγράμματος.

5.1 FRONT PANEL

Το front panel του client δεν έχει ουσιαστικές διαφορές με αυτό του απλού παλμογράφου που παρουσιάσαμε σε προηγούμενο κεφάλαιο όσο και με αυτό του server. Ωστόσο θα λέγαμε ότι εκεί περιορίζονται και οι μόνες ομοιότητες που έχει τόσο με τον παλμογράφο όσο και με το server. Αυτό συμβαίνει γιατί ουσιαστικά αυτό που θέλουμε από τον client είναι να έχει την ίδια «βιτρίνα» με το server και το VI που θέλουμε να χρησιμοποιήσουμε - γι' αυτό και τα όμοια front panels αλλά τελείως διαφορετικό περιεχόμενο γι' αυτό και τα τελείως διαφορετικά block diagrams.

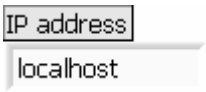
Με απλή παρατήρηση αρκεί για να διαπιστώσει κανείς ότι στο front panel του client έχουν προστεθεί μόνο 4 στοιχεία. Αυτά είναι δύο string controls, ένα digital control και ένα string indicator.



Εικόνα 5.1. το front panel του client

IP address

Αυτό το string control χρησιμεύει ούτως ώστε ο απομακρυσμένος χρήστης-client να δηλώνει την IP διεύθυνση του υπολογιστή στον οποίο είναι εγκατεστημένο το VI του server. Όπως γίνεται εύκολα αντιληπτό μόνο αν δηλωθεί η σωστή IP διεύθυνση από τον client γίνεται εφικτή η σύνδεση. Για να επιλέξουμε το string control ακολουθούμε τη διαδρομή: **controls palette<string & path<string control**



Εικόνα 5.2. το string control του IP address όπως εμφανίζεται στο front panel

Password

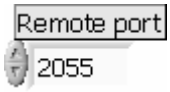
Αυτό το string control χρησιμεύει ούτως ώστε ο απομακρυσμένος χρήστης-client να δηλώνει το προσωπικό του password. Το password ελέγχεται από το server και ανάλογα με

το αν αυτό είναι έγκυρο ολοκληρώνει τη σύνδεση ή όχι. Με τη χρήση του password εξασφαλίζουμε ότι ακόμα και αν κάποιος έχει πρόσβαση στο VI του client αν δεν είναι εξουσιοδοτημένος δε θα μπορέσει να συνδεθεί με το server. Για να επιλέξουμε το string control ακολουθούμε τη διαδρομή: **controls palette<string & path<string control**

Εικόνα 5.3. το string control του password όπως εμφανίζεται στο front panel

Remote port

Αυτό το digital control χρησιμοποιείται για τη δήλωση της θύρας του υπολογιστή μέσω της οποίας θα γίνει η επικοινωνία και η TCP σύνδεση. Να σημειωθεί ότι δεν είναι δυνατόν να γίνουν δύο διαφορετικές συνδέσεις σε μία θύρα. . Για να επιλέξουμε το digital control ακολουθούμε τη διαδρομή: **controls palette<numeric<digital control**



Εικόνα 5.4. το digital control του remote port όπως εμφανίζεται στο front panel

PasswordAccepted

Αυτό το string indicator ενημερώνει τον client αν το password που δήλωσε έγινε δεκτό από το server. Μετά την πληκτρολόγηση του password και ανάλογα με το αν αυτό έγινε δεκτό ή όχι, εμφανίζονται στο indicator οι λέξεις **yes** ή **no** αντίστοιχα.

Εικόνα 5.5. το string indicator του password accepted όπως εμφανίζεται στο front panel

5.2 BLOCK DIAGRAM

Το δεύτερο μέρος του κεφαλαίου χωρίζεται και αυτό σε δύο μέρη. Στο πρώτο θα παρουσιάσουμε με αναλυτικό τρόπο το block diagram και τον τρόπο λειτουργίας του έτσι ώστε να την κάνουμε κατανοητή στον αναγνώστη. Στο δεύτερο μέρος θα παραθέσουμε όλα τα έτοιμα functions τα οποία και χρησιμοποιήθηκαν για τη δημιουργία του block diagram.

5.2.1 Τρόπος λειτουργίας του Block Diagram

Το block diagram του client είναι απλά δομημένο και συνεπώς γίνεται εύκολα κατανοητός ο τρόπος λειτουργίας του. Αυτό συμβαίνει γιατί το VI του client δεν υπολογίζει ουσιαστικά τίποτε, παρά μόνο λαμβάνει και στέλνει κάποια δεδομένα.
Αρχικά όπως παρατηρούμε το block diagram φροντίζει να ανοίγει μια TCP σύνδεση μέσω μιας έτοιμης function του LabVIEW της **TCP Open Connection**. Για να γίνει αυτό θα πρέπει φυσικά πρώτα ο client να έχει δηλώσει μέσω του front panel την κατάλληλη IP address και θύρα επικοινωνίας. Στη συνέχεια ο έλεγχος του block diagram περνάει στο subVI με την ονομασία **Send Password** το οποίο και θα αναλύσουμε παρακάτω. Η γενική λειτουργία του συγκεκριμένου subVI είναι να στέλνει το password το οποίο έχει πληκτρολογήσει ο χρήστης στον server και να λαμβάνει σαν απάντηση μια Boolean τιμη. Η τιμή αυτή είναι TRUE αν το password είναι έγκυρο και FALSE αν είναι άκυρο. Η τιμή αυτή αποτελεί και την έξοδο του subVI η οποία στη συνέχεια συνδέεται με τον επιλογέα μιας case structure η οποία περιλαμβάνει δύο cases(TRUE-FALSE). Έτσι όταν έχει γίνει επιτυχως η σύνδεση και ο απομακρυσμένος χρήστης έχει πληκτρολογήσει ένα έγκυρο password επιλέγεται και εκτελείται το διάγραμμα που βρίσκεται στην case TRUE ενώ όταν πληκτρολογείται λάθος password επιλέγεται και εκτελείται το διάγραμμα της case FALSE.
Το διάγραμμα της case FALSE είναι ουσιαστικά κενό. Αυτό συμβαίνει γιατί αφού το password ελέγχθηκε και βρέθηκε άκυρο το μόνο που μένει είναι να κλείσει η σύνδεση. Έτσι το μόνο που διακρίνουμε μέσα στο σώμα της case FALSE είναι τα καλώδια που μεταφέρουν το connection ID και το error. Ωστόσο υπάρχει ακόμα και το terminal του string indicator -Password Accepted- που συνδέεται με μια string constant που περιέχει την τιμή **No**. Με αυτόν τον τρόπο ο χρήστης αντιλαμβάνεται -μέσω του front panel- ότι έχει πληκτρολογήσει ένα λανθασμένο password. Σε αντίθεση με το ουσιαστικά κενό διάγραμμα της case FALSE, το διάγραμμα της case TRUE περιλαμβάνει πολλά στοιχεία μιας και εκεί μεταφέρεται ο έλεγχος του block diagram όταν έχει ξεκινήσει μια επιτυχής επικοινωνία.

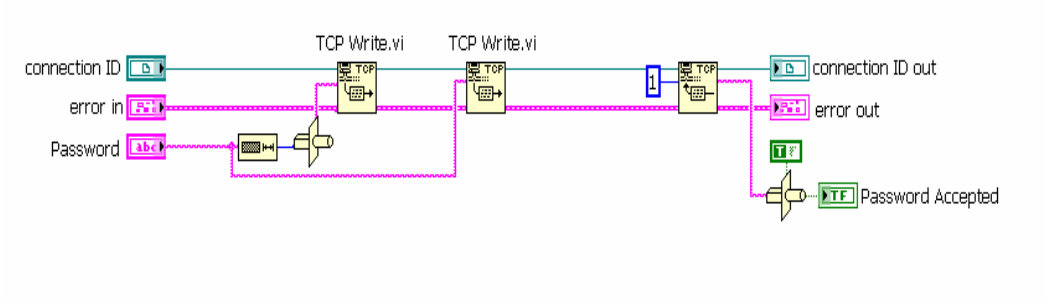
Παρατηρούμε ότι όλα βρίσκονται μέσα σε ένα while loop το οποίο ελέγχεται από το stop button του front panel. Αν και με την πρώτη ματιά το διάγραμμα φαίνεται πολύπλοκο μια δεύτερη προσεκτικότερη ματιά αρκεί για να παρατηρήσει κάποιος ότι ουσιαστικά έχουμε μια αλυσίδα από λήψεις και αποστολές δεδομένων και πέραν τούτου ουδέν. Η διαδικασία μέσω της οποίας λαμβάνεται και στέλνεται κάποιο δεδομένο έχει αναλυθεί στο κεφάλαιο του server οπότε περιττεύει η επανάληψη της. Πρέπει να αναφέρουμε όμως τη σειρά με την οποία γίνεται η λήψη και αποστολή των δεδομένων. Η σειρά αυτή είναι ίδια με την σειρά την οποία λαμβάνονται και στέλνονται τα δεδομένα από τον server μόνο που έχουμε την αντίστροφη λειτουργία. Για παράδειγμα το πρώτο στοιχείο της αλυσίδας είναι τα δεδομένα του waveform graph τόσο στην αλυσίδα του server όσο και σε αυτή του client. Η διαφορά είναι ότι ο server στέλνει τα δεδομένα(write) ενώ ο client τα διαβάζει(read). Η συνέχεια της αλυσίδας του client συνεχίζεται με την αποστολή των δεδομένων που έχει επιλέξει μέσω των αντίστοιχων controls του panel. Η σειρά αυτή όπως προαναφέραμε είναι ίδια με τη σειρά με την οποία ο server λαμβάνει τα αντίστοιχα δεδομένα.

Μέσα στο διάγραμμα της case TRUE υπάρχει και η local variable του string indicator -Password Accepted- που συνδέεται με μια string constant που περιέχει την τιμή **Yes**. Με αυτόν τον τρόπο ο χρήστης πληροφορείται -μέσω του front panel- ότι έχει πληκτρολογήσει ένα αποδεκτό password.

Κάποιες άλλες λειτουργίες που δεν επηρεάζουν την κυρίως λειτουργία του VI ως client έχουν αναλυθεί στο block diagram του oscilloscope.VI.

Τέλος να αναφέρουμε ότι μετά το case structure ακολουθεί το function TCP Close Connection το οποίο και χρησιμεύει στο κλείσιμο της σύνδεσης. Ακολουθεί το General Error Handler το οποίο αν έχει προκύψει κάποιο σφάλμα κατά τη λειτουργία, επιστρέφει μια περιγραφή του και ενίοτε εμφανίζει ένα dialog box.

5.2.2 Simple Provide Password.VI



Εικόνα 5.7. το block diagram του Simple Provide Password

Το VI αυτό χρησιμεύει στην απόστολή του password που πληκτρολογεί ο χρήστης – client στον server και λαμβάνει μια Boolean τιμή σαν επιστροφή. Ο χρήστης εισάγει το password στο front panel. Κατόπιν το string του password περνάει μέσα από τη function string length η οποία επιστρέφει τον αριθμό των bytes του string. Ο αριθμός αυτός περνάει μέσα από τη function type cast μετατρέπεται σε string και οδηγείται σε ένα TCP Write όπως και το string του password. Αφού σταλούν αυτά τα δύο δεδομένα στον server τότε το VI διαβάζει από το αντίστοιχο VI που βρίσκεται στο server μια Boolean τιμή η οποία είναι και η απάντηση αν το password του client έγινε δεκτό. Η τιμή αυτή φυσικά έχει σταλεί σε μορφή string γι' αυτό και χρειάζεται να περάσει μέσα από ακόμα ένα type cast στου οποίου την είσοδο type έχουμε συνδέσει μια Boolean constant. Με αυτόν τον τρόπο επαναφέρουμε την πληροφορία στον τύπο δεδομένων που στάλθηκε.

Π.Α. Ανάλυση συναρτήσεων που δομούν το ζεύγος client-server

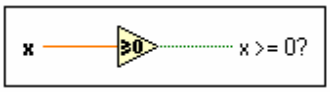
Σε αυτό το κεφάλαιο θα γίνει η ανάλυση των συναρτήσεων του LabVIEW που χρησιμοποιήθηκαν για την υλοποίηση του Server και του Client.Η συναρτήσεις θα διαχωριστούν σε λογικές συναρτήσεις ,συναρτήσεις σύγκρισης, συναρτήσεις πινάκων, συναρτήσεις TCP IP και διάφορες συναρτήσεις.

a. Λογικές συναρτήσεις

Η ανάλυση των συναρτήσεων Not, And, Equal και Or έγιναν στο κεφάλαιο 2.2.2.

Greater or Equal To 0

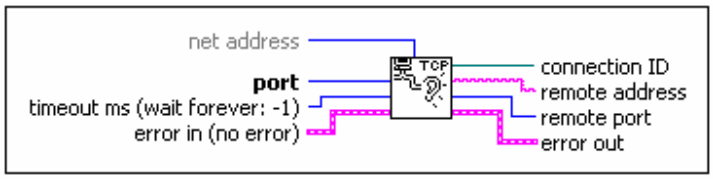
Επιστρέφει την τιμή true αν ο x είναι μεγαλύτερος η ίσος με το 0,αλλιώς επιστρέφει την τιμή false.Το επιλέγουμε ως εξής:**All Functions palette<Comparison<Greater Or Equal To 0**



b. TCP IP συναρτήσεις

TCP Listen.VI.

Δημιουργεί μία κατάσταση αναμονής για μία αποδεκτή δικτυακή TCP επικοινωνία.Το επιλέγουμε ως εξής:**All Functions palette<Communication < TCP<TCP Listen.VI.**



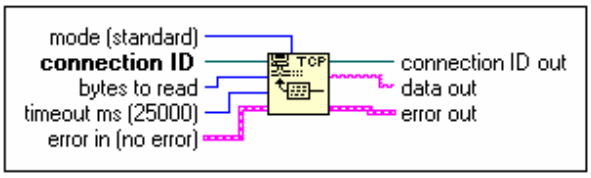
Στα προγράμματα χρησιμοποιήθηκαν οι ακροδέκτες εισόδου port και timeout σε ms και οι ακροδέκτες εξόδου connection ID και error out. Η είσοδος port είναι ο αριθμός της θύρας στην οποία θα αναμένει την επικοινωνία το VI. Ο ακροδέκτης timeout είναι το χρονικό διάστημα μέσα στο οποίο το VI θα αναμένει επικοινωνία.

Μετά το πέρας αυτού του διαστήματος επιστρέφει μήνυμα σφάλματος το VI. Η έξοδος connection ID είναι ο αριθμός αναφοράς της δικτυακής επικοινωνίας που χαρακτηρίζει μονοσήμαντα την συγκεκριμένη TCP επικοινωνία και χρησιμοποιείται ο αριθμός, αυτός ως αναφορά στη συγκεκριμένη επικοινωνία, σε επόμενες κλήσεις άλλων VI. Τέλος χρησιμοποιείται ο ακροδέκτης error out που επιστρέφει ένα string που περιέχει πληροφορίες για τυχόν σφάλματα που προκύπτουν κατά την εκτέλεση .του συγκεκριμένου VI. Το string αποτελείται από μία λογική τιμή, έναν ακέραιο αριθμό και από ένα σύνολο χαρακτήρων. Η λογική τιμή είναι true σε περίπτωση εμφάνιση σφάλματος, αλλιώς είναι false. Ο αριθμός προσδιορίζει το σφάλμα που προέκυψε και είναι μη μηδενικός, ενώ σε περίπτωση μη εμφανίσεις σφάλματος είναι μηδέν. Οι χαρακτήρες προσδιορίζουν την θέση όπου εμφανίστηκε το σφάλμα.

TCP Read.VI

Το συγκεκριμένο VI διαβάζει συγκεκριμένο πλήθος bytes από την TCP δικτυακή επικοινωνία και επιστρέφει το αποτέλεσμα στον ακροδέκτη data out.

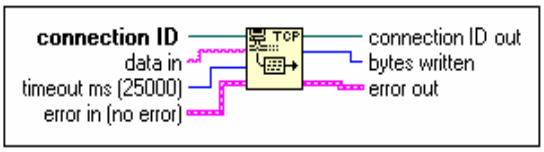
Το επιλέγουμε ως εξής:**All Functions palette<Communication<TCP<TCP Read.VI.**



Χρησιμοποιήθηκαν οι είσοδοι connection ID, bytes to read, timeout ms error in και όλες οι έξοδοι του. Οι ακροδέκτες connection ID, timeout ms και error out έχουν ίδιες λειτουργίες με τους αντίστοιχους ακροδέκτες του TCP Listen.VI και η ανάλυση τους έγινε παραπάνω. Η είσοδος bytes to read δέχεται έναν ακέραιο αριθμό που προσδιορίζει τα bytes που θα διαβάσει το συγκεκριμένο VI. Η είσοδος error in είναι ένα string όμοιο με το string του TCP Listen.VI. Περιέχει το σφάλμα που μπορεί να έχει προκύψει πριν την εκτέλεση του TCP Read.VI και το μεταβιβάζει αμέσως στο error out χωρίς να εκτελέσει την λειτουργία του. Αν το error in δεν περιέχει σφάλμα εκτελεί κανονικά το πρόγραμμα του και μεταβιβάζει στο error out την τιμή του no error. Σε περίπτωση που εμφανιστεί σφάλμα κατά την εκτελέσει του TCP Read.VI μεταβιβάζει στο error out το σφάλμα του. Ό ακροδέκτης data out περιέχει τα δεδομένα που έχει διαβάσει το VI.

TCP Write.VI

Γράφει δεδομένα σε μία TCP δικτυακή επικοινωνία. Το επιλέγουμε ως εξής: **All Functions palette<Communication<TCP< TCP Write.VI.**

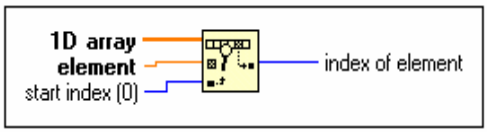


Χρησιμοποιήθηκαν οι είσοδοι connection ID, bytes to read, timeout ms error in και όλες οι έξοδοι του. Ο μόνοι ακροδέκτες που διαφέρει είναι η έξοδος bytes written και η είσοδος data in. Ο ακροδέκτης data in περιέχει τα δεδομένα που θα γραφούν και ο ακροδέκτης bytes written τον αριθμό τον bytes που γράφτηκαν.

c. Συναρτήσεις πινάκων

Search 1D Array

Με χρήση της συνάρτησης αυτής γίνεται αναζήτηση στον μονοδιάστατο πίνακα 1D Array του στοιχείου element. Το επιλέγουμε ως εξής: **All Functions palette<Array<Search 1D Array.**

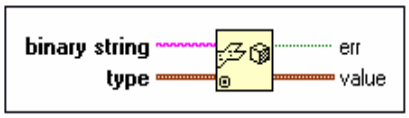


Ο ακροδέκτης start index προσδιορίζει από ποια θέση του πίνακα θα ξεκινήσει η αναζητήσει. Η έξοδος index of element περιέχει την θέση του πίνακα όπου βρέθηκε το στοιχείο, ενώ σε περίπτωση αποτυχίας έχει την τιμή -1.

d. Διάφορες συναρτήσεις

Unflatten From String

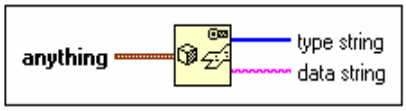
Μετατρέπει ένα δυαδικό string στον τύπο δεδομένων που έχει εισαχθεί στον ακροδέκτη type. Το επιλέγουμε ως εξής:**All Functions palette<Advanced<Data Manipulation<Unflatten From String.**



Το err είναι True σε περίπτωση που εμφανιστεί σφάλμα κατά την μετατροπή.

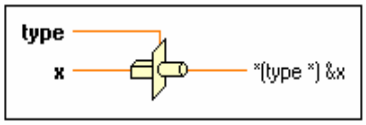
Flatten To String

Μετατρέπει την είσοδο σε ένα δυαδικό string. Το type string είναι μία αναφορά στον τύπο δεδομένων της εισόδου. Το επιλέγουμε ως εξής:**All Functions palette<Advanced<Data Manipulation<Flatten To String.**



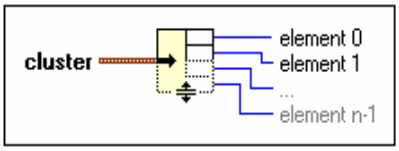
Type Cast

Μετατρέπει την είσοδο στον τύπο δεδομένων που προσδιορίζει ο ακροδέκτης type. Το επιλέγουμε ως εξής:**All Functions palette<Advanced<Data Manipulation<Type cast.**



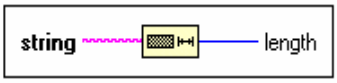
Unbundl

Διαχωρίζει τα στοιχεία που αποτελούν ένα cluster.Με την σύνδεση του ακροδέκτη cluster αυτόματα η συνάρτηση εμφανίζει τα στοιχεία του cluster στις θέσης element 0..n-1. Το επιλέγουμε ως εξής:**All Functions palette<Cluster<Unbundle.**



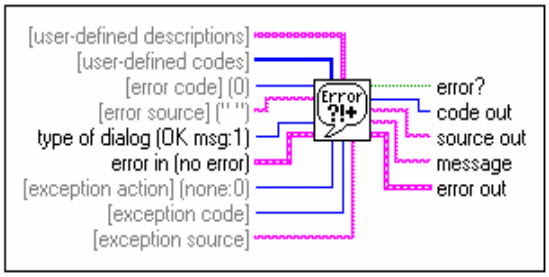
String Length

Επιστρέφει τον αριθμό των χαρακτήρων του string που είναι στην είσοδο της συνάρτησης



General Error Handler.VI

Εάν ένα λάθος εμφανιστεί, αυτό το VI επιστρέφει μια περιγραφή του λάθους και εμφανίζει προαιρετικά ένα πλαίσιο διαλόγου.



ΒΙΒΛΙΟΓΡΑΦΙΑ

- BridgeVIEW and LabVIEW-G Programming Reference Manual

- The LabVIEW Student Edition Users Guide

- LabVIEW Basics Course Manual

- www.ni.com

- http://zone.ni.com/dzhp/app/main (developer zone)

- http://forums.ni.com/ (forum zone)