ΤΕΙ ΗΠΕΙΡΟΥ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Προγραμματισμός οθόνης αφής

Σερσέμη Χιονία-Χρυσοβαλάντου ΑΜ: Ι101

Επιβλέπων καθηγητής: Χαριλόγης Βασίλειος

ΆΡΤΑ 2017

ΔΗΛΩΣΗ ΠΕΡΙ ΛΟΓΟΚΛΟΠΗΣ

Δηλώνω πως είμαι η συγγραφέας της παρούσας πτυχιακής εργασίας η οποία παραδόθηκε τον μήνα Σεπτέμβρη του 2017.Η αναφερόμενη πτυχιακή εργασία δεν αποτελεί προϊόν αντιγραφής ή ανάθεσης σε τρίτους και διεξάχθηκε στο εργαστήριο του τμήματος «Εργαστήριο Υπολογιστικών και Τηλεπικοινωνιακών Συστημάτων.Όλες οι πηγές που χρησιμοποιήθηκαν αναφέρονται στο παρών κείμενο καθώς και στην βιβλιογραφία.Τα υπόλοιπα είναι επινόηση της γραφώντος η οποία φέρει την καθολική ευθύνη για το κείμενο αυτό. Δηλώνω υπεύθυνα ότι δεν υπάρχει λογοκλοπή για την εργασία αυτή.

ΣΕΡΣΕΜΗ ΧΙΟΝΙΑ-ΧΡΥΣΟΒΑΛΑΝΤΟΥ

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Χαριλόγη , για την πολύτιμη καθοδήγηση του καθ' όλη τη διάρκεια της εκπόνησης της πτυχιακής εργασίας.

ΠΡΟΛΟΓΟΣ

Στην παρούσα πτυχιακή εργασία αναπτύσσεται η διαδικασία προγραμματισμού μιας οθόνης αφής η οποια είναι συνδεδεμένη με μικροελεγκτή Arduino. Με την χρήση της συγκεκριμένης οθόνης δίνεται η δυνατότητα σε χρήστες να ελέγχουν τη θερμοκρασία, την υγρασία του εδάφους καθώς και να θέτουν όρια (εύρος θερμοκρασιών) που επιθυμούν οι ίδιοι.Αναλυτικότερα,ο προγραμματισμός και ο σχεδιασμός των επιπέδων πραγματοποιήθηκαν με τη βοήθεια του λογισμικού που παρέχει η κατασκευάστρια εταιρία (4DSystems)της οθόνης.

Ο προγραμματισμός της οθόνης ώστε να αναπαριστά γραφικά την ροή ενός σεναρίου για την λήψη και απεικόνιση των τιμών για έναν μικροελεγκτή, στοχεύει στην ευκολότερη αλληλεπίδραση του χρηστή με τον μικροελεγκτή και την λειτουργικότερη απεικόνιση των αποτελεσμάτων των μετρήσεων. Η δημιουργία ενός τέτοιου περιβάλλοντος αλληλεπίδρασης του χρήστη με τον μικροελεγκτή δημιουργεί ένα φιλικό και σύγχρονο τρόπο χρήσης των δυνατοτήτων του μικροελεγκτή ακόμη και από χρήστες οι οποίοι δεν έχουν εξειδικευμένες γνώσεις. Ο προγραμματισμός της οθόνης για την δημιουργία των γραφικών απεικονίσεων και ερωτήσεων προς τον χρήστη καθώς και η πλοήγηση του σε διαφορετικά επίπεδα σχεδιασμού αποτελεί το πρώτο μέρος του θέματος της παρούσας εργασίας. Επιπρόσθετα, ο προγραμματισμός του μικροελεγκτή και η ανάδραση του με την οθόνη αποτελεί το δεύτερο μέρος της παρούσας πτυχιακής δίνοντας έμφαση στον κώδικα προγραμματισμού του μικροελεγκτή.Αναλυτικότερα, τα πρώτα κεφάλαια είναι η εισαγωγή στο θέμα και περιγράφουν τον σκοπό της προσπάθειας αυτής αλλά και οι απαραίτητες αναφορές.

Στο δεύτερο κεφάλαιο υπάρχει η γενική περιγραφή της διαδικασίας που οδήγησε στην εκπόνηση της εργασίας. Αναλυτικά, θα γίνει εκτενής αναφορά στην εταιρία της οποίας χρησιμοποιήθηκε το προϊόν καθώς και η λεπτομερείς περιγραφή της συγκεκριμένης οθόνης, του επεξεργαστή αλλά και του μικροελεγκτή Arduino.

Στο επόμενο κεφάλαιο γίνεται η μελέτη του προγράμματος που χρησιμοποιήθηκε για την ολοκλήρωση της οθόνης για την ομαλή λειτουργίας της αλλά και για την εύχρηστη λειτουργία της. Συγκεκριμένα, αναπτύσσονται τα επίπεδα του προγράμματος, ο κώδικας αλλά και το σχεδιαστικό μέρος.

Έπειτα από την ανάλυση των παραπάνω βρίσκεται η γλώσσα γραφής, ο κώδικας με ερμηνεία της κάθε γραμμής αλλά και σχολιασμός. Με την ολοκλήρωση του κεφαλαίου τούτου υπάρχουν αναλυτικά οι παραπομπές και η αντίστοιχη βιβλιογραφία.

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

Με την πάροδο των χρόνων εμφανίζεται όλο και μεγαλύτερη βελτίωση των τεχνολογικών επιτευγμάτων σε όλους τους τεχνολογικούς τομείς.Ο όρος τεχνολογία ποτέ δεν ήταν σαφής.Αναλυτικότερα, αφορά την χρήση και την εφαρμογή όλων των μορφών γνώσεων για να επιτύχει κάποιο πρακτικό απότελεσμα.Στην παρούσα πτυχιακή εργασία διεξάχθηκε μία προσπάθεια για την εξυπηρέτηση πελατών σε πρακτικό επίπεδο με την χρήση οθόνης αφής. Η συγκεκριμένη οθόνη δίνει την δυνατότητα στους χρήστες να υλοποιήσουν ένα μεγάλο εύρος εφαρμογών ανάλογα με τις ανάγκες τους.Στη συνέχεια θα αναπτυχθούν οι δυνατότητες της οθόνης και ειδικότερα ο τρόπος που χρησιμοποιήθηκε για την πτυχιακή εργασία. .

1.1 Σκοπός

Η συγκεκριμένη οθόνη έχει ως στόχο την διευκόλυνση των ατόμων που θα την χρησιμοποιήσουν με ποικίλους τρόπους. Ο τρόπος λειτουργίας της την καθιστά ως μία εύχρηστη συσκευή η οποία δείχνει σαφή και γρήγορα αποτελέσματα, αξιόπιστα, κατανοητά για όλους τους χρήστες ,επαγγελματίες και μη.Ο ευκολος τρόπος προγραμματισμου της οθόνης σε συνδυασμό με την ποικιλία των εφαρμογών της εξυπηρετεί μεγάλο εύρος αναγκών και πλέον την καθυστά ένα καθημερινό εργαλείο.

1.2 Λέξεις κλειδιά

Λέξεις κλειδια που θα διευκολύνουν τους αναγνώστες στην αναζήτηση επιπλέον πληροφοριών και ειδικότερης έρευνας σε τμήματα του software και του hardware είναι οι ακόλουθες:4DSystems, μLCD-32PTU, Workshop 4D,Touchscreen.Picaso processor ,Lcd touch screens, Arduino processor.

2. Γενική περιγραφή

Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν προϊόντα μίας συγκεκριμένης εταιρίας η οποία σαν οδηγός συνέβαλε στην ολοκλήρωση του

συγκεκριμένου έργου με τις πολύτιμες πληροφορίες που παρέχει σε όλους τους αναγνώστες της ιστοσελίδας της.

Στην αγορά υπάρχουν διαθέσιμοι μικροελεγκτές και οθόνες αφής. Η επιλογή εξαρτάται αποκλειστικά από την χρήστη. Ανάλογα με τις δυνατότητες και τα χαρακτηριστικά που προσφέρει το κάθε προϊόν ξεχωριστά. Κάθε εταιρία διαθέτει ποικίλες επιλογές στον αγοραστή.Στην προκειμένη περίπτωση η εταιρία με την οποία επιλέχθηκε να γίνει το έργο αφορά κατά κύριο λόγο τις πιο εξελιγμένες εφαρμογές καθώς παρέχει μεγάλη ποικιλία προϊόντων σε οθόνες αφής , arduino processors και άλλων τεχνολογικών εργαλείων.Η πρώτη εμφάνιση οθόνης που να χρησιμοποιεί τεχνολογία αφής, τοποθετείται στο 1965. Εμπνευστής της είναι ο E.A. Johnson του Royal Radar Establishment στο Malvern του Ηνωμένου Βασιλείου. Η τεχνολογία που χρησιμοποιούσε η συγκεκριμένη οθόνη είναι η capacitive touch που χρησιμοποιείται ακόμα σε smartphones.

Τη δεκαετία του 1970, ο Dr. G. Samuel Hurst και η ομάδα του δημιούργησαν μία νέα τεχνολογία αφής για οθόνες, γνωστές ως resistive touchscreens. Αυτή τεχνική, χρησιμοποιούσε ένα αγώγιμο φύλλο καλύμματος, που όταν κάποιος του ασκούσε πίεση, αυτό ερχόταν σε επαφή με ένα φύλλο που περιείχε ένα σύστημα συντεταγμένων .

Οι resistive touchscreens ήταν οικονομικά προσιτές. Ήταν αρκετά ανθεκτικές για απαιτητικές εφαρμογές όπως σε εργοστάσια και νοσοκομεία.Έχουν χρησιμοποιηθεί από τους κατασκευαστές smartphone συσκευών.

Η δεκαετία του 1980, ονομάστηκε «δεκαετία αφής»,πρώτη φορά κάνουν την εμφάνισή τους οθόνες πολλαπλών επαφών (multitouch screens). Η πρώτη συσκευή με αυτή την τεχνολογία, έγινε το 1982 στο πανεπιστήμιο του Τορόντο και δημιουργήθηκε από το Nimish Mehta.

Το 1983, ο Myron Krueger, παρουσίασε το Video Place ένα οπτικό σύστημα που μπορούσε να παρακολουθήσει τις κινήσεις του χεριού, με κάμερες και projectors.

Μετά το 2000 η χρήση τους σε φορητές συσκευές γνώρισε τεράστια άνοδο.

Συγκεκριμένα, η οθόνη αφής ανηκει στην κατηγορία των LCD (LiquidCrystalDisplay οθονών δηλαδή είναι οθόνη υγρών κρυστάλλων. Είναι η πιο διαδεδομένη τεχνολογία των τελευταίων χρόνων.

Η αρχή λειτουρίας της οθόνης βασίζεται στο γεγονός ότι το φως παράγεται από λαμπτήρες φθορισμού οι οποίοι βρίσκονται πίσω από την οθόνη με κατεύθυνση προς τους υγρούς κρυστάλλους.

Εικόνα 1 : φωτογραφία υγρών κρυστάλλων από μικροσκόπιο

Οι LCD οθόνες χρησιμοποιούν φωτοεκπέμπουσες διόδους (LED) [1](1) αντί των λαμπτήρων φθορισμού καθώς παρέχουν περισσότερα πλεονεκτήματα όπως:

- μεγαλύτερη διάρκεια ζωής
- αυξημένο χρωματικό εύρος
- βαθύτερο μαύρο.

Οι υγροί κρύσταλλοι είναι αυτοί που καθορίζουν το ποσό του φωτός που θα περάσει από την οθόνη στον θεατή-χρήστη. Λειτουργούν σαν «φωτοφράχτες» και κάποια χαρακτηριστικά είναι η μεγαλύτερη φωτεινότητα, η εξαιρετικά λεπτή κατασκευή καθώς και η εκπομπή ελάχιστης ακτινοβολίας.Βέβαια υπάρχουν διάφορα είδη οθόνης όπως για παράδειγμα οι OLED.

2.1 4D systems

Η εταιρία 4Dsystems ιδρύθηκε το 1990 με σκοπό να γίνει παγκόσμιος ηγέτης στην ανάπτυξη , την έρευνα , την υλοποίηση ευφυών γραφικών και όχι μόνο

[1] LED(lightemittingdiode): ημιαγωγός ο οποίος εκπέμπει φωτεινή ακτινοβολία στενού φάσματος όταν του παρέχεται ηλεκτρική τάση κατά τη φορά ορθής πόλωσης.

ιδεών. Θέλει να προσφέρει στους πελάτες καινοτόμες και οικονομικές επιλογές.

Αναλυτικότερα ,η συγκεκριμένη επιχείρηση σχεδιάζει, αναπτύσσει και κατασκευάζει γραφικές λύσεις με την χρήση τελευταίας τεχνολογίας LCD(οθόνη υγρών κρυστάλλων) αλλά και OLED (organic light-emitting diode / οργανική δίοδος εκπομπής φωτός). Επιπλέον , παρέχει την επιλογή πλέον τεχνολογικά ανεπτυγμένων επεξεργαστών καθώς και ένα πρόγραμμα που λειτουργεί ως προσομοιωτής. Τέλος, τα προϊόντα που κατασκευάζουν θέλουν να είναι εύχρηστα και αποδοτικά για τον κάθε χρήστη για αυτό και οι οθόνες είναι allinone δηλαδή θέλουν να προσφέρουν μια απλή διεπαφή με δύο καλώδια σε κάθε κεντρικό επεξεργαστή.

Είναι μία εταιρία η οποία δημιουργήθηκε στην Αυστραλία και είναι ίσως ο παγκόσμιος ηγέτης τεχνολογικών επιτευγμάτων. Ειδικεύεται στην μικροηλεκτρονική προσφέροντας οικονομικές και ποιοτικές λύσεις σε ενσωματωμένες πλατφόρμες

2.1.1 μLCD-32PTU

Εικόνα 2: http://www.4dsystems.com.au/product/uLCD_32PTU/

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε το προϊόν μLCD-32PTU το οποίο είναι μία οθόνη υγρών κρυστάλλων. Ιστορικά ,η ανακάλυψη του υγρού κρυστάλλου έγινε το 1888 από τον βοτανολόγο Friedrich Reinitzer αλλά επί της ουσίας η πρώτη πειραματική χρήση τους σε ηλεκτρονική συσκευή απεικόνισης έγινε το 1968 από την εταιρία RCA.

Προγραμματισμός οθόνης αφής

Η συγκεκριμένη οθόνη αφής είναι μία έξυπνη οθόνη με πολλά χαρακτηριστικά για να δημιουργηθεί το γραφικό περιβάλλον που θέλει ο κάθε χρήστης αλλά και ελεγκτής διασύνδεσης διαφόρων εφαρμογών.

Στο επίκεντρο του σχεδιασμού βρίσκεται ο picaso processor, ο οποίος καθοδηγείται από μια ιδιαίτερα βελτιστοποιημένη εικονική μηχανή η οποία ονομάζεται EVE (Extensible Virtual Engine). Στον σχεδιασμό έχει ενσωματωθεί ένα ευρύ φάσμα περιφερειακών συσκευών hardwareαλλά και λογισμικού. Σκοπός των δυνατοτήτων είναι ο χρήστης να έχει όσο το δυνατόν περισσότερες επιλογές. Να δώσει την ελευθερία να προσαρμόσει το προϊόν σε οποιαδήποτε εφαρμογή θελήσει.

Θα ακολουθήσει μία πλήρης ανάλυση των χαρακτηριστικών τις οθόνης καθώς και γενικές πληροφορίες που αφορούν την οθόνη αφής.

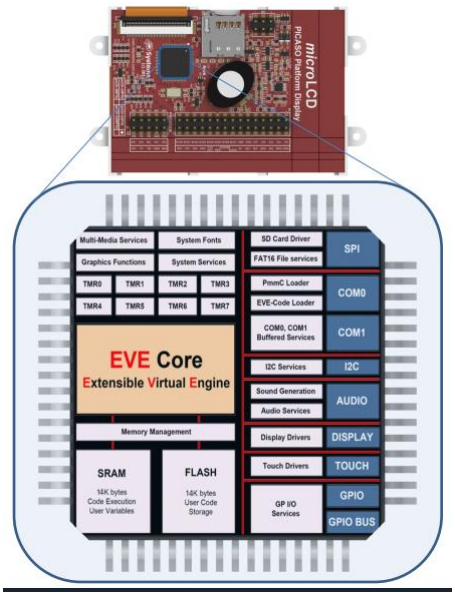
- Οθόνη αφής:3,2’’ TFTLCD
- Διαστάσεις:56,3mmx80,5mmx15,9mm
- Βάρος:50 gr
- Χρωματική ανάλυση: 240x320 RGB ,65Kχρώματα
- Μπαταρία λιθίου
- Συμβατότητα με αρχεία DOS, RoHS ,CE
- Εκτεταμένα γραφικά και βιβλιοθήκες
- Επεξεργαστής picaso
- Εύκολη διασύνδεση σε συσκευή υποδοχής VCC, TX, RX, GND, RESET
- Micro-SDκάρτα μνήμης
- 2x5 pins για σειριακή διασύνδεση
- 2x15 pinsheader για επεκτασιμότητα
- 2x3 pinsheader σύνδεση μπαταρίας
- Υποστήριξη εικόνων, κινούμενων σχεδίων,βίντεο
- Είναι συμβατή με αρχεία DO

Αρχικά, είναι οθόνη αφής TFTLCDστις 3,2’’ . Η ανάλυση βρίσκεται στα 240x320 RGB με 65K χρωματικές επιλογές.

Χρησιμοποιεί επεξεργαστή γραφικώνPicaso οποίος είναι ενσωματωμένος και έχει σχεδιαστεί για διασύνδεση με συσκευές OLEDκαι LCD.Διαθέτει ισχυρά γραφικά, προσφέρει τη δυνατότητα χρήσης εικόνας, βίντεο, κινούμενων σχεδίων και αμέτρητα άλλα χαρακτηριστικά τα οποία είναι «χτισμένα» στο εσωτερικό του τσιπ.

Προγραμματισμός οθόνης αφής

Αποτελεί μια πλήρη σουίτα πρώτης κατηγορίας για οποιονδήποτε χρήστη θέλει να χρησιμοποιήσει χρώμα και όχι μόνο. Ένα πολύ σημαντικό χαρακτηριστικό του συγκεκριμένου επεξεργαστή είναι η επικοινωνία του με άλλες εξωτερικές συσκευές.



Εικόνα 3:http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_productbrief_R_1_2.pdf

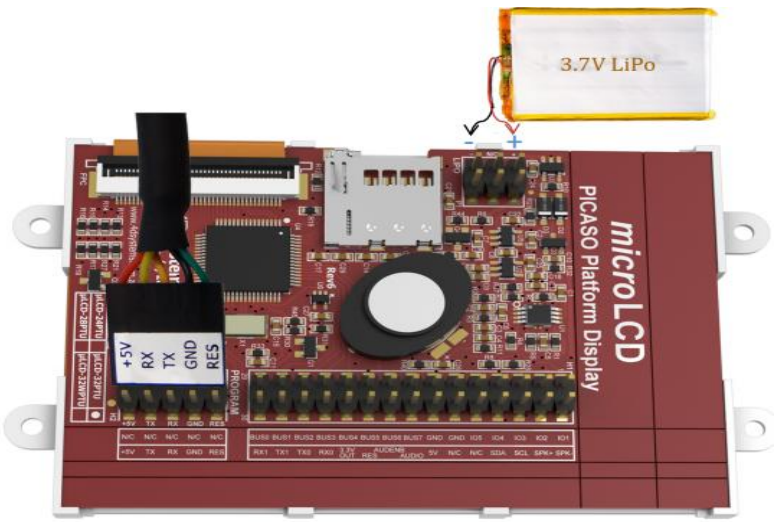
Ο επεξεργαστής διαθέτει υποστήριξη ήχου και η οθόνη αναπαράγει αρχεία τύπου WAV τα οποία αποθηκεύονται στην μνήμη micro-SD. Ο ενισχυτής ήχου λειτουργεί στα 1,2W με ένα μικρό ηχείο στα 8Ω.

Εικόνα 4: http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_productbrief_R_1_2.pdf

Επιπλέον υπάρχει η επιλογή επεκτασιμότητας της μνήμης. Έτσι ο χρήστης μπορεί να ανακτήσει όλων των τύπων αρχεία αλλά και να καταγράψει δεδομένα της εφαρμογής.Υποστηρίζει μέχρι 2Bmicro-SD αλλά και micro-SDHC η οποία ξεκινά από 4GB.

Υπάρχουν δύο σειριακές πόρτες οι RXO/TXOκαι RX1/TX1. Δίαυλος επικοινωνίας SPI για αποθήκευση σε uSD, Επίσης υποστηρίζει γρήγορη παράλληλη μεταφορά δεδομένων μέσω 8 bits, διαθέτει δεκατρία γενικού σκοπού I/O pins. Ένα ακόμη χαρακτηριστικό είναι πως έχει 8x16 timersμε ανάλυση 1ms. Υποστηρίζει όλες τις γραμματοσειρές των windows.Υπάρχουν τέσσερις πλάκες γωνιών με 2,7 χιλιοστά μέγεθος τρυπών για μηχανική στερέωση.Σε γενικές γραμμές είναι όπως προαναφέρθηκε μία έγχρωμη οθόνη αφής η οποία υποστηρίζει εικόνες ,κινούμενα σχέδια και βίντεο.

Η φόρτιση της μLCD-32PTUείναι απλή, Συνδέοντάς την σε ένα υπολογιστή με ένα 4D programming cable όπως επίσης και από πηγή 5VDC. Εναλλακτικά, υπάρχει επαναφορτιζόμενη LiPo μπαταρία στην πλακέτα με σύνδεση 2x3 pin. Μία μπαταρία LiPo στα 3,7V είναι η ιδανική λύση για φόρτιση δια χειρός.



Εικόνα 5: http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_productbrief_R_1_2.pdf

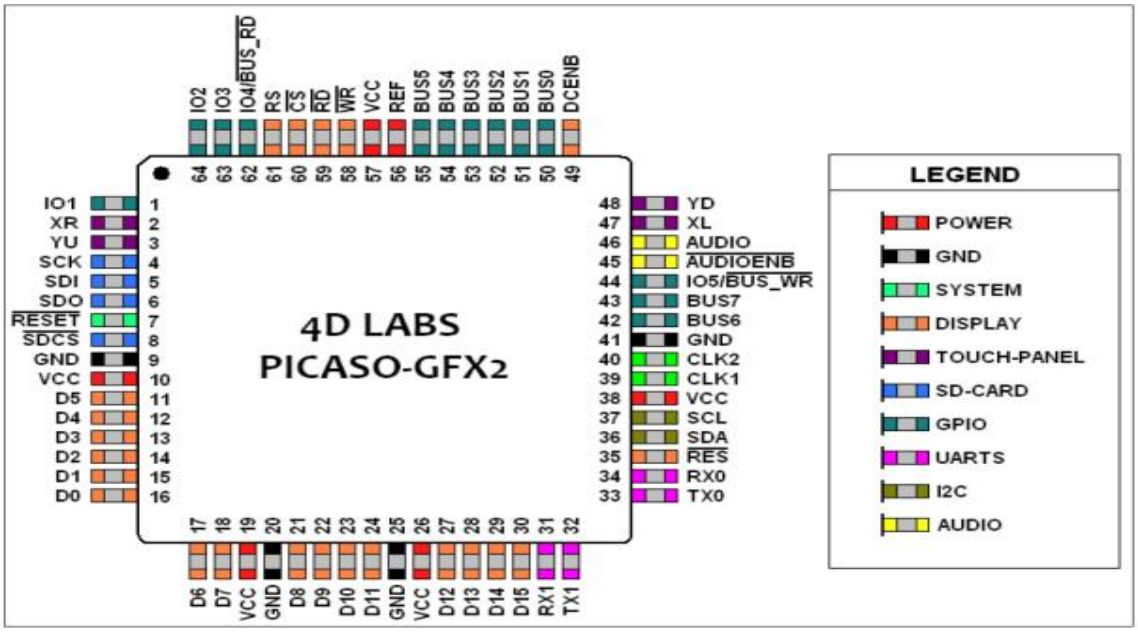
2.1.2 Picaso

Ο picaso είναι ένας επεξεργαστής ενσωματωμένος ο οποίος έχει εύκολη διασύνδεση με οθόνες LCDκαι OLED. Διαθέτει ισχυρά γραφικά , κείμενα , εικόνες ,κινούμενα σχέδια και πολλά άλλα χαρακτηριστικά. Προσφέρει interface plug-n-playσε 16 bits σε 80 series colour σε LCD και OLED.Έχει σχεδιαστεί για να δουλεύει με ελάχιστα δεδομένα καθώς τα παρέχει όλα το συγκεκριμένο τσιπ και αυτό είναι αρκετά εύχρηστο για τον κάθε χρήστη. Δίνει την δυνατότητα της γρήγορης ανάπτυξης και σχεδιασμού, εξοικονόμησης κόστους και δεν υπάρχει πλέον το βάρος ενός χαμηλού επιπέδου σχεδιασμού.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ:

- -Γραφικά χαμηλού κόστους
- -Ιδανικό ως ενσωματωμένος επεξεργαστής
- -Ιδανική διεπαφή ως επεξεργαστή γραφικών σε οποιοδήποτε ελεγκτή Η/Υ
- -Συνδέεται σε οποιαδήποτε έγχρωμη οθόνη που υποστηρίζει 80 series 16 bitsστη CPU
- -Ενσωματωμένος πυρήνας εικονικού επεξεργαστή υψηλής απόδοσης
- -Πλήρης σειρά ενσωματωμένων γραφικών και υπηρεσίες πολυμέσων
- -Εμφάνιση πλήρων έγχρωμων εικόνων, κινούμενων σχεδίων, βίντεο
- -14 KB για αποθήκευση κώδικα χρηστών
- -14KB για τις μεταβλητές χρηστών
- -13 ψηφιακά pinsεισόδου-εξόδου
- -Διεπαφή I2C (Master)
- -Υπηρεσίες αρχείων FAT16
- -2 σειριακές θύρες
- -Υποστήριξη διεπαφής SPI
- -Διεπαφή οθόνης αφής 4 καλωδίων
- -Υποστήριξη ήχου για αρχεία κυμάτων και σύνθετο ήχο
- -Χρονοδιακόπτες 8x16bitμε ανάλυση ενός χιλιοστού του δευτερολέπτου
- -Ενιαία τροφοδοσία στα 3,3 volt
- -Συμβατότητα με RoHS
- -Διαθέτει 64 pinsTQFPμε διαστάσεις 10mmx 10mm

Προγραμματισμός οθόνης αφής



Εικόνα 6:
http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf

Αναλυτικά:

Pins	Σύμβολα	I/O	Περιγραφή
1	IO1	I/O	Μέχρι 5.0 V
2	XR	A	Σύνδεση με XRή X
3	YU	A	Σύνδεση με YUή Y
4	SCK	O	Έξοδος σειριακού ρολογιού. Χρήση μόνο SDκάρτας μνήμης Σήμα σειριακού (SCK)
5	SD1	I	SPIσειριακής εισόδου δεδομένων.Χρήση μόνο SDκάρτας μνήμης. Σύνδεσημόνο με SPIserialdata out (SDO).
6	SD0	O	SPIσειριακή εξόδου δεδομένων. .Χρήση μόνο SDκάρτας μνήμης. Σύνδεση μόνο με SPIserialdata in (SDI).
7	RESET	I	Κύριοreset.Σύνδεση αντίστασης στα 4.7K του pin με το VCC.

Προγραμματισμός οθόνης αφής

8	SDCS	O	Επιλογή κάρτας μνήμης. Χρήση μόνο SD.Σύνδεση του pinμε το chip enable (CS) της κάρτας μνήμης.
9,20,25,41	GND	P	Γείωση συσκευής
10,19,26,38,57	VCC	P	Θετική παροχή συσκευής
11	D5	I/O	Δίαυλος δεδομένων οθόνης 5 bit
12	D4	I/O	Δίαυλος δεδομένων οθόνης 4 bit
13	D3	I/O	Δίαυλος δεδομένων οθόνης 3 bit
14	D2	I/O	Δίαυλος δεδομένων οθόνης 2 bit
15	D1	I/O	Δίαυλος δεδομένων οθόνης 1 bit
16	D0	I/O	Δίαυλος δεδομένων οθόνης 0 bit
17	D6	I/O	Δίαυλος δεδομένων οθόνης 6 bit
18	D7	I/O	Δίαυλος δεδομένων οθόνης 7 bit
21	D8	I/O	Δίαυλος δεδομένων οθόνης 8 bit
22	D9	I/O	Δίαυλος δεδομένων οθόνης 9 bit
23	D10	I/O	Δίαυλος δεδομένων οθόνης 10 bit
24	D11	I/O	Δίαυλος δεδομένων οθόνης 11 bit
27	D12	I/O	Δίαυλος δεδομένων οθόνης 12 bit
28	D13	I/O	Δίαυλος δεδομένων οθόνης 13 bit
29	D14	I/O	Δίαυλος δεδομένων οθόνης 14 bit
30	D15	I/O	Δίαυλος δεδομένων οθόνης 15 bit
31	RX1	I	Ασύγχρονη σειριακή πόρτα COM1, pinPX1. Εκπομπή σήματος (Tx).Αντοχή στα 5,0V
32	TX1	O	Ασύγχρονη σειριακή πόρτα COM1, pinTX1. Εκπομπή σήματος (Rx).Αντοχή στα 5,0V
33	TX0	O	Ασύγχρονη σειριακή πόρτα COM1, pinTX. Εκπομπή σήματος

			(Rx).Αντοχή στα 5,0V.Λαμβάνει δεδομένα από τον Picaso.
34	RX0	I	Ασύγχρονη σειριακή πόρτα COM1, pinTX. Εκπομπή σήματος (Rx).Αντοχή στα 5,0V.Στέλνει δεδομένα στον Picaso.
35	RES	O	Εμφάνισηreset.O Picaso αρχικοποιεί τη οθόνη πατώντας αυτό το pin.. Απαιτείται σύνδεση του pinμε το σήμα reset της οθόνης.
36	SDA	I/O	12C Δεδομένα εισόδου-εξόδου
37	SCL	O	12C Ρολόι εξόδου
39	CLK1	I	Εισαγωγή ρολογιού 1 έως 12MHz κρυστάλλου.
40	CLK2	O	Εισαγωγή ρολογιού 2 έως 12MHz κρυστάλλου.
42	BUS6	I/O	Γενικού σκοπού παράλληλο I/O BUS(0.7),6 bit.Αντοχή μέχρι 5V.
43	BUS7	I/O	Γενικού σκοπού παράλληλο I/O BUS(0.7),7 bit.Αντοχή μέχρι 5V.
44	IO5/BUS_WR	I/O	Γενικού σκοπού pin IO5.Χρησιμοποιείται επίσης για σήμα BUS_WRγια εγγραφή δεδομένων στο παράλληλο GPIOBUS (0.7).
45	AUDENB	O	Ενεργοποίηση ήχου. LOW: ενεργοποίηση εξωτερικού ενισχυτή ήχου. HIGH: Απενεργοποίηση εξωτερικού ενισχυτή ήχου.
46	AUDIO	O	Λειτουργία εξόδου ήχου PWM (PulseWidthModulated)
47	XL	O	Σύνδεση του pinμε XRή X σήμα στον πίνακα αφής

48	YD	O	Σύνδεση του pinμε YDή Y σήμα στον πίνακα αφής
49	DCENB	O	Σήμα ενεργοποίησης υψηλής τάσης DC-DC. Υψηλή: Ενεργοποίηση DC-DC. Χαμηλή:Απενεργοποίηση DC-DC.
50	BUS0	I/O	Γενικού σκοπού παράλληλο I/O BUS(0.7),0bit.Αντοχή μέχρι 5V.
51	BUS1	I/O	Γενικού σκοπού παράλληλο I/O BUS(0.7),1bit.Αντοχή μέχρι 5V.

I: Είσοδος O: Έξοδος P: Ενέργεια A: Αναλογικό

(Πηγή:http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf)

Η χρήσεις του συγκεκριμένου επεξεργαστή είναι αρκετές. Μερικές από αυτές είναι ο βιομηχανικός έλεγχος και η ρομποτική, για εξοπλισμό παιχνιδιών, ιατρικά εργαλεία και εφαρμογές, σε συστήματα αεροπορίας. Επιπλέον , χρησιμοποιείται για συστήματα ελέγχου ανελκυστήρων, σε τερματικά σημείων πώλησης, σε οικιακές συσκευές, συστήματα ελέγχου ασφάλειας και πρόσβασης καθώς επίσης και σε συστήματα πλοήγησης GPS.

CS	RS	RD	WR	Operation
0	0	0	1	Read Display Status Register
0	0	1	0	Write Display Index Register
0	1	0	1	Read Display GRAM Data
0	1	1	0	Write Register or GRAM Data
1	X	X	X	No Operation

Εικόνα 7: http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf

Προγραμματισμός οθόνης αφής

Συγκεκριμένα:

- DO-D15 pins (Δίσκος απεικόνισης δεδομένων)
  Ο δίαυλος απεικόνισης δεδομένων είναι στα 16 bitείναι αμφίδρομη θύρα και όλα τα δεδομένα γράφουν και διαβάζουν πάνω σε αυτό. Άλλα σήματα ελέγχου είναι τα RW, RD, CSκαι RS τα οποία συγχρονίζουν τα δεδομένα για την μεταφορά από και προς την οθόνη.
- CS pin (Επιλογή εμφάνισης)
  Η πρόσβαση στην οθόνη είναι δυνατή μόνο όταν η εμφάνιση του chip display είναι Low.Έπειτα απαραίτητη είναι η σύνδεση του pinμε το CSτης οθόνης.
- RS pin(επιλογή εγγραφής)
  Το συγκεκριμένο σήμα καθορίζει εάν αποστέλλονται τα δεδομένα ή εντολές στην οθόνη. Με το LOW όταν είναι επιλεγμένο εμφανίζεται δείκτης ή καταχωρητής κατάστασης. Αντίστοιχα με το HIGH εμφανίζεται το GRAMή τα δεδομένα καταχωρήσεων.
- RES pin (Επαναφορά)
  Ο Picaso αρχικοποιεί τη οθόνη πατώντας αυτό το pin.. Απαιτείται σύνδεση του pin με το σήμα reset της οθόνης.
- DCENB pin(Εξωτερική ενεργοποίηση DC-DC)
  Επί της ουσίας γίνεται ενεργοποίηση σήματος υψηλής τάσης. Συγκεκριμένα Υψηλή:Ενεργοποίηση DC-DC.Χαμηλή:Απενεργοποίηση DC-DC.
- WR pin (Επιλογή γραφής)
  Δίνεται το σήμα εγγραφής. Ο επεξεργαστής picaso επιβεβαιώνει το σήμα Low κατά την εγγραφή δεδομένων σε συνδυασμό με τον δίαυλο απεικόνισης (DO-D15). Έπειτα απαραίτητη είναι η σύνδεση με την οθόνη.
- RD pin(Επιλογή γραφής)
  Αυτό είναι το σήμα ανάγνωσης της οθόνης. Ο επεξεργαστής picaso επιβεβαιώνει το σήμα Low όταν διαβάζει τα δεδομένα από την οθόνη και σε συνδυασμό με DO-D15.

Ο επεξεργαστής picaso όπως προαναφέρθηκε υποστηρίζει microSD και MMC κάρτες μνήμης, μέσω διεπαφής SPI. Η κάρτα μνήμης χρησιμοποιείται για όλα τα αρχεία πολυμέσων όπως η ανάκτηση εικόνων , κινούμενων σχεδίων και βίντεο διότι η διεπαφή έχει τη συγκεκριμένη λειτουργία ως μοναδικό σκοπό. Η κάρτα μνήμης μπορεί επίσης να χρησιμοποιηθεί για αποθήκευση γενικού σκοπού για καταγραφή δεδομένων εφαρμογής.

- SDI pin (SPIείσοδος σειριακών δεδομένων)
  Γίνεται χρήση αποκλειστικά της κάρτας μνήμης SD. Επόμενο βήμα είναι η σύνδεση του pinμε SPI έξοδος δεδομένων (SDO) της κάρτας μνήμης.

Προγραμματισμός οθόνης αφής

- SDO pin (SPIέξοδος σειριακών δεδομένων)
  Γίνεται χρήση αποκλειστικά της κάρτας μνήμης SD. Επόμενο βήμα είναι η σύνδεση του pinμε SPI είσοδος δεδομένων (SDI) της κάρτα μνήμης.
- SCK pin(SPIσειριακό ρολόι)
  Είναι η έξοδος σειριακού ρολογιού SPI. Γίνεται χρήση μόνο της κάρτας μνήμης SD. Επιπλέον , πρέπει να γίνει η σύνδεση του pinστο σειριακό ρολόι της κάρτας μνήμης.
- SDCS pin(Επιλογή κάρτας μνήμης)
  Επιλογή κάρτας μνήμης SD (SDCS). Γίνεται χρήση μόνο κάρτας μνήμης SD. Απαραίτητη είναι η σύνδεση του pinμε το Chip Enable (CS) της κάρτας.

Ο επεξεργαστής picaso διαθέτει δύο ασύγχρονες σειριακές θύρες οι οποίες επικοινωνούν με εξωτερικές σειριακές συσκευές. Οι πόρτες αυτές ονομάζονται COM0 και COM1. Η θύρα COM0 είναι η κύρια διεπαφή για τον χρήστη 4DGL για λήψεις προγραμμάτων. Μόλις ολοκληρωθεί το πρόγραμμα εφαρμογής δηλαδή έχει γίνει η λήψη και εκτελεστεί, η σειριακή θύρα είναι και πάλι διαθέσιμη. Θα ακολουθήσουν pinsγια τις δύο πόρτες.

- TX0 (Σειριακή μετάδοση COM0)
  Η ασύγχρονη σειριακή θύραCOM0 μεταδίδει με το pin TX0.Απαραίτητη είναι η σύνδεση του συγκεκριμένου pinμε εξωτερική συσκευή για λήψη (RX). Υπάρχει ανεκτικότητα μέχρι τα 5,0 Volt.
- RX0 (Σειριακή λήψη COM0)
  Η ασύγχρονη σειριακή θύρα COM0 λαμβάνει με το pin TX0.Απαραίτητη είναι η σύνδεση του συγκεκριμένου pinμε εξωτερική συσκευή για μετάδοση (TX). Υπάρχει ανεκτικότητα μέχρι τα 5,0 Volt.
- TX1(Σειριακή μετάδοση COM1)
  Η ασύγχρονη σειριακή θύρα COM1 μεταδίδει με το pin TX0.Απαραίτητη είναι η σύνδεση του συγκεκριμένου pinμε εξωτερική συσκευή για λήψη (RX). Υπάρχει ανεκτικότητα μέχρι τα 5,0 Volt.
- RX1 (Σειριακή λήψη COM1)
  Η ασύγχρονη σειριακή θύρα COM1λαμβάνει με το pin TX0.Απαραίτητη είναι η σύνδεση του συγκεκριμένου pinμε εξωτερική συσκευή για μετάδοση (TX). Υπάρχει ανεκτικότητα μέχρι τα 5,0 Volt.

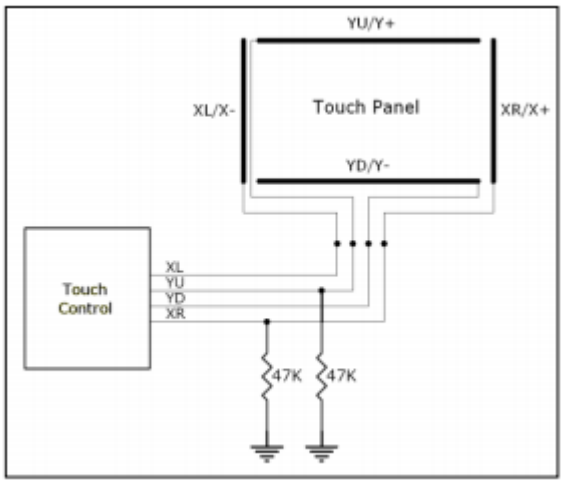
Η αποκλειστική υποστήριξη ήχου που παρέχει ο συγκεκριμένος επεξεργαστής τον καθιστά καλύτερο με άλλους του είδους του. Το PWM εξασφαλίζει καλύτερη ποιότητα ήχου. Επιπλέον μπορεί να εκτελεστούν αρχεία ήχου παράλληλα με άλλες εντολές. Αναλυτικά ακολουθούν κάποια από τα pins.

- AUDIO pin (Έξοδος ήχουPWM)

Εξωτερικός ενισχυτής ήχου παρέχει το συγκεκριμένοpin. Επιπλέον, παρέχει έξοδο ήχου 16 bit DAC/PWMγια χρήση με έναν εξωτερικό ενισχυτή ήχου. Επίσης παρέχει μία χαμηλού κόστους υλοποίηση.

- AUDENB pin (Ενεργοποίηση εξόδου ήχου)
  Είναι η εξωτερική ενεργοποίηση του pin. Παρέχει την επιλογή on/off. Όπως έχει προαναφερθεί Low είναι για ενεργοποίηση εξωτερικού ενισχυτή και High για απενεργοποίηση.

Ο picaso υποστηρίζει 4 wire touch panels.Ακολουθεί ένα μικρό διάγραμμα μεταξύ της διεπαφής και της οθόνης αφής.



Εικόνα 8:
http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf

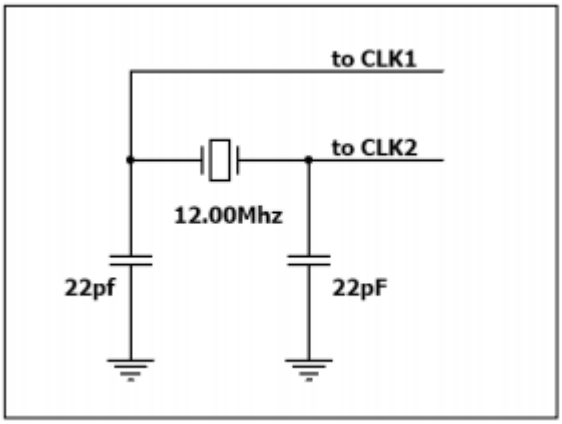
- XR pin (ΕίσοδοςX-Read)
  Γίνεται ανάγνωση αναλογικού σήματος. Απαραίτητα χρειάζεται η σύνδεση του pin με XRή X+και του σήματος της οθόνης αφής.
- XL pin (ΈξοδοςX-Drive)
  Γίνεται η έξοδος σήματος οδήγησης. Απαραίτητα χρειάζεται η σύνδεση του pin με XLή X- και του σήματος της οθόνης αφής.
- YU pin (ΕίσοδοςY-Read)
  Γίνεται ανάγνωση αναλογικού σήματος. Απαραίτητα χρειάζεται η σύνδεση του pin με YUή Y+και του σήματος της οθόνης αφής.
- YD pin (ΈξοδοςY-Drive)
  Γίνεται η έξοδος σήματος οδήγησης. Απαραίτητα χρειάζεται η σύνδεση του pin με YDήY- και του σήματος της οθόνης αφής.

Προγραμματισμός οθόνης αφής

Γενικότερα υπάρχουν διαθέσιμες στον χρήστη δεκατρείς είσοδοι και έξοδοι γενικού σκοπού. Ομαδοποιούνται με συγκεκριμένο τρόπο δηλαδή IO1 μέχρι IO5και από BUS0 έως BUS7. Τα πρώτα πέντε pins παρέχουν ευελιξία στις επιμέρους λειτουργίες ενώ τα υπόλοιπα οχτώ εξυπηρετούν συλλογικά.

- IO1-IO3 pins (pins GPIO)
  Είναι όπως προαναφέρθηκε pins γενικού σκοπού. Κάθε pinρυθμίζεται ξεχωριστά ένα θα είναι δηλαδή inputή output.Όταν μιλάμε για power up και resetείναι input.
- IO4 /BUS_RD pin
  Το pin IO4 είναι γενικού σκοπού. Επίσης χρησιμοποιείται για BUS_RD σήμα δηλαδή για ανάγνωση και τοποθέτηση των δεδομένων στο παράλληλο GPIOBUS0 έως BUS7.
- IO5/ BUS_WR pin
  Το pin IO5 είναι γενικού σκοπού. Χρησιμοποιείται για BUS_WR σήμα δηλαδή για εγγραφή και τοποθέτηση των δεδομένων στο παράλληλο GPIOBUS0 έωςBUS7.
- BUS0- BUS7 pins (GPIO 8-bit BUS)
  Είναι 8bitπαράλληλος δίαυλος γενικού σκοπού I/O.
  Τέλος, θα γίνει αναφορά στα pinstου συστήματος.
- VCC pins (Τάση τροφοδοσίας συσκευής)
  Τα συγκεκριμένα pinsπρέπει να είναι ρυθμισμένα σε μία τάση μεταξύ των 3,0V και των 3.6V. Ιδανική είναι η τιμή της τάσης στα 3,3V.
- GND pins (Γείωση συσκευής)
  Τα pinsπρέπει να είναι συνδεδεμένα με το «έδαφος» του συστήματος.
- Reset pin (Αρχικοποίηση)
  Το pin δίνει την δυνατότητα μέσα σε δύο microsecondsνα αρχικοποιήσει την συσκευή.
- CLK1, CLK2
  Είναι είσοδοι ταλαντωτή συσκευής. Η παρακάτω εικόνα δείχνει αναλυτικά:

Προγραμματισμός οθόνης αφής



Εικόνα 9:
http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf

2.2 Πρόγραμμα Workshop 4D

Τα ευρήματα της έρευνας που προηγήθηκε ,προκειμένου να γίνει η σωστή επιλογή του προϊόντος, ήταν καθοριστικά για την επιτυχία του σταδίου της υλοποίησης που ακολούθησε Για να ολοκληρωθεί το έργο σωστά έπρεπε να γίνουν κάποια βήματα. Η οθόνη έπρεπε να περάσει από διάφορα επίπεδα για να είναι διαθέσιμη στον χρήστη.

Απαραίτητη, ήταν η χρήση του λογισμικού workshop 4D το οποίο είναι ελεύθερα διαθέσιμο. Το συγκεκριμένο πρόγραμμα είναι μια ολοκληρωμένη σουίτα IDE λογισμικού για τα Microsoft Windows η οποία παρέχει μία πλατφόρμα ανάπτυξης λογισμικού processors αλλά και modules.Το IDE συνδυάζει editor, compiler, linkerκαι downloaderγια να αναπτυχθεί πλήρης 4DGL κώδικας.

Το workshop 4D περιλαμβάνει τέσσερα διαφορετικά επίπεδα για να επιλέξει ο χρήστης με βάση των απαιτήσεων της εφαρμογής ή ανάλογα και με τις δεξιότητες του χρήστη.

Εικόνα 10: http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_1_2.pdf

Ακολουθεί η ανάπτυξη όλων των δυνατοτήτων που παρέχει το συγκεκριμένο πρόγραμμα. Αναλυτικά, το πρώτο περιβάλλον που παρέχει είναι το Designerμε το οποίο ο χρήστης μπορεί να γράψει τον δικό του κώδικα σε 4DGLγλώσσα και να δημιουργήσει από το μηδέν όποιο έργο θέλει. Ένα ακόμη περιβάλλον είναι το VisiGenie εξαιρετικά προηγμένο καθώς ο χρήστης δεν χρειάζεται να προγραμματίσει καθόλου. Συγκεκριμένα, πρέπει απλά να συνδέσει το προϊόν με το πρόγραμμα και ο κώδικας θα δημιουργηθεί αυτόματα καθώς παρέχει την τελευταία ανάπτυξη που παρέχει η 4DSystems. Στην συνέχεια υπάρχει το Visi το οποίο παρέχει στον χρήστη όλα τα γραφικά που χρειάζεται αλλά του δίνει και την δυνατότητα του προγραμματισμού. Τέλος, υπάρχει και το περιβάλλον Serial.Η κύρια δυνατότητα είναι η φόρτωση σε οποιαδήποτε σειριακή εφαρμογή και επιπλέον μπορεί να μετασχηματιστεί οποιοδήποτε προϊόν επιτρέποντας στο χρήστη να το ελέγχει από όποιον κεντρικό υπολογιστή θέλει.

2.2.1 Designer

Αρχικά, ο χρήστης όταν ανοίξει το πρόγραμμα πρέπει να επιλέξει το περιβάλλον που θέλει να χρησιμοποιήσει. Ακολουθεί μία συνοπτική ανάλυση της δυνατότητας αυτής.

Βημα 1

Προγραμματισμός οθόνης αφής

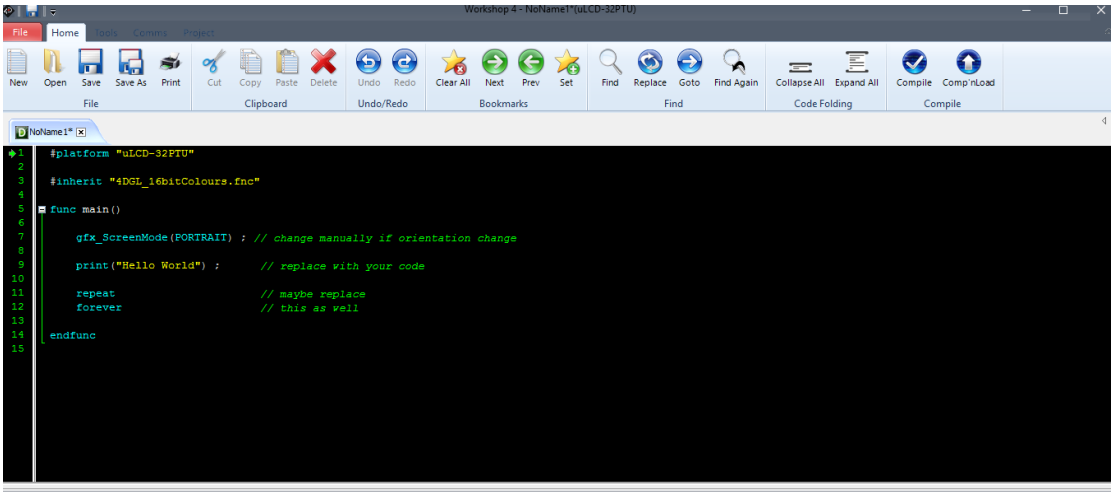
Ο χρήστης πρέπει να επιλέξει με ποιον τρόπο θα δημιουργήσει το πρόγραμμα που θέλει. Στη συνέχεια, θα διαλέξει ποιο προϊόν το οποίο διατίθεται .

Βήμα 2

Ο χρήστης θα κληθεί να επιλέξει το περιβάλλον που θεωρεί πως διαθέτει τις προϋποθέσεις που χρειάζεται και στην παρούσα φάση θα επιλέξουμε το Designer .

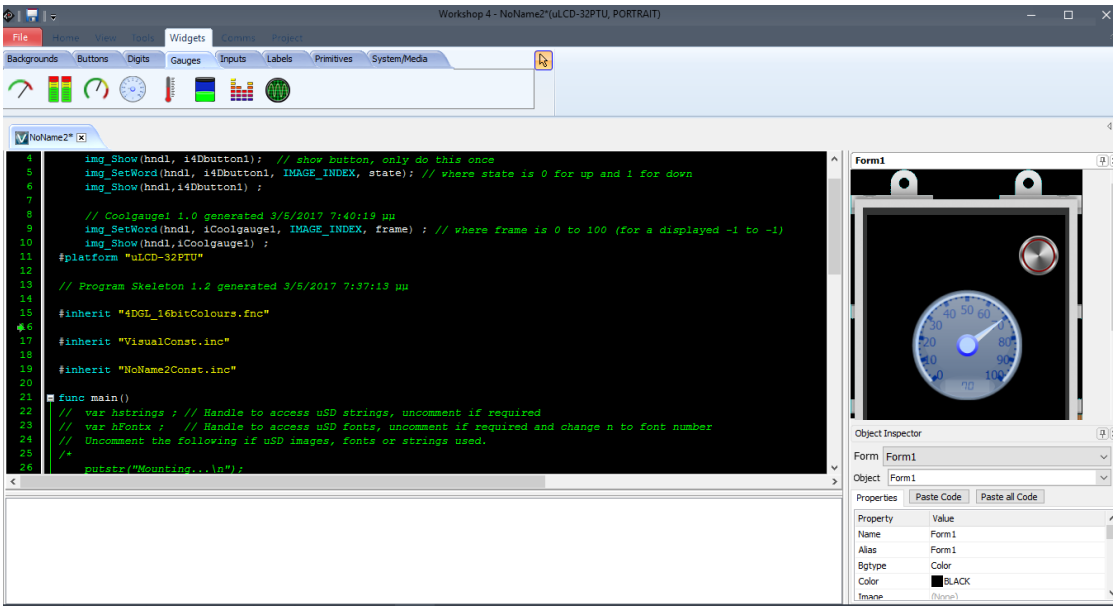
Προγραμματισμός οθόνης αφής

Τέλος, το περιβάλλον είναι έτοιμο για ανάπτυξη.



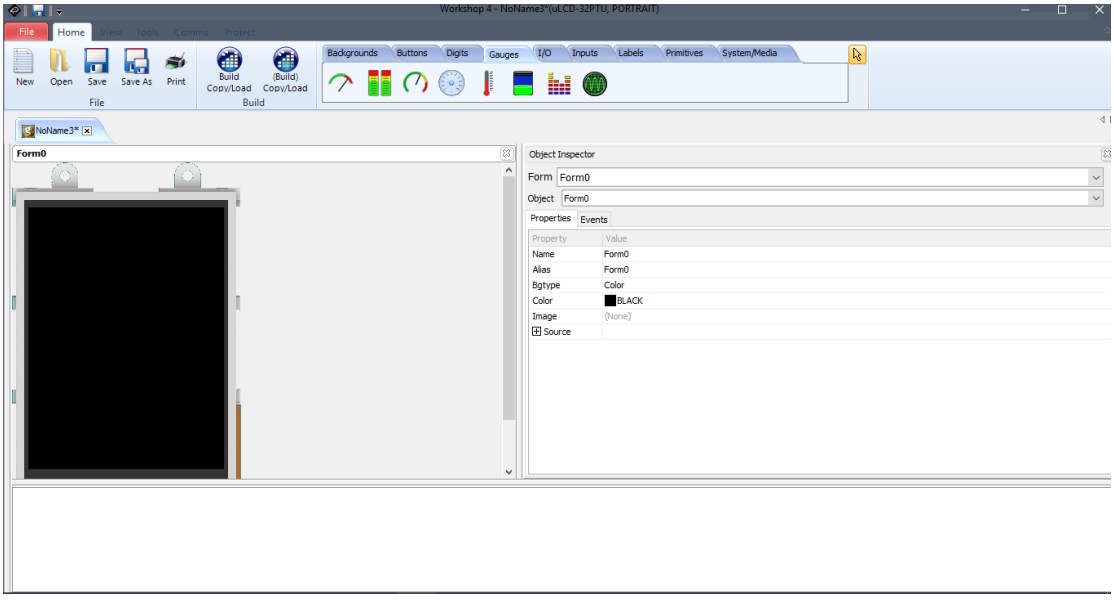
2.2.2 Visi

Η διαδικασία επιλογής του συγκεκριμένου περιβάλλοντος είναι η ίδια. Η μόνη αλλαγή που υπάρχει είναι στις δυνατότητες που παρέχει. Στο περιβάλλον αυτό θα αναπτυχθεί και το έργο. Στα δεξιά βρίσκεται η οθόνη αφής στην οποία έχουν προστεθεί κάποια γραφικά για μεγαλύτερη ευκολία στον αναγνώστη. Στα αριστερά ο χρήστης μπορεί να γράψει τον κώδικα που χρειάζεται.



2.2.3 VisiGenie

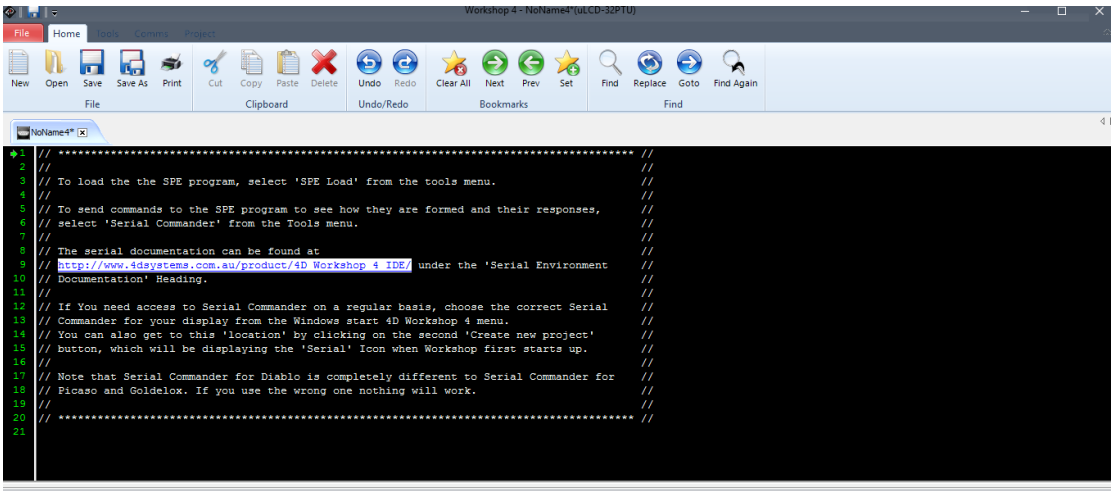
Είναι το τρίτο περιβάλλον που παρέχει το Workshop και το πιο προηγμένο σύμφωνα με την εταιρία. Η κάθε επιλογή όμως εξαρτάται από τον κάθε χρήστη.



2.2.4 Serial

Προγραμματισμός οθόνης αφής

Το τελευταίο περιβάλλον είναι και το συγκεκριμένο. Όπως προαναφέρθηκε μπορούν να συνδεθούν σειριακές συσκευές.



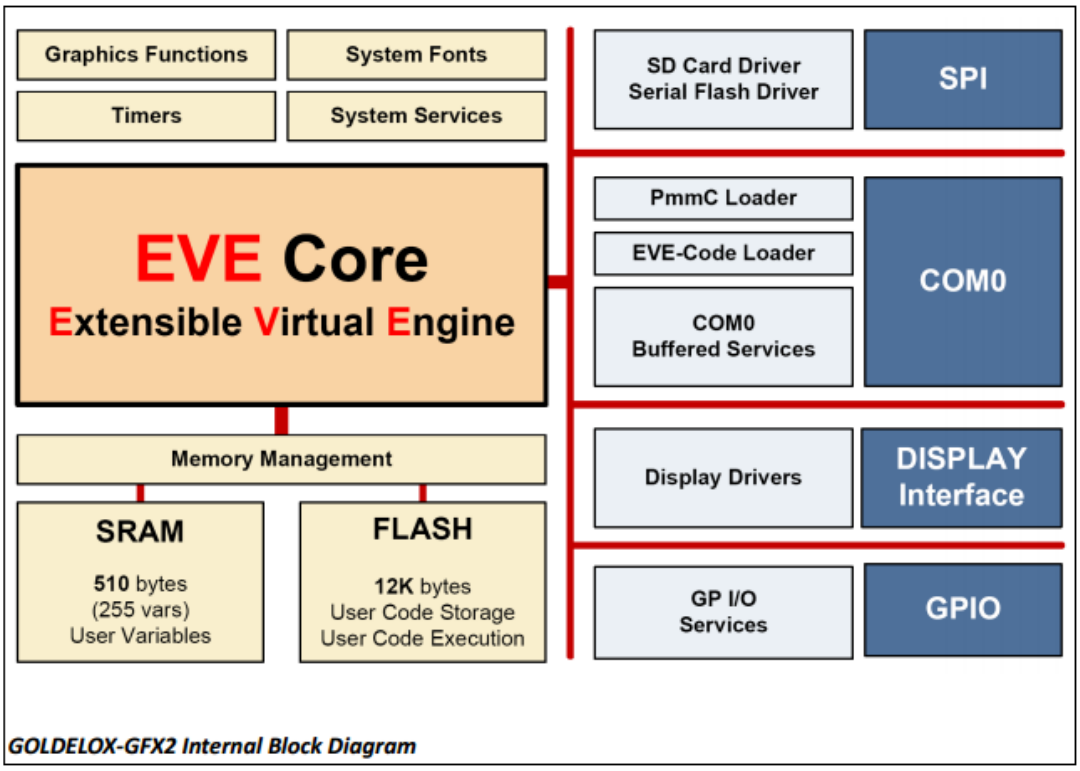
3. Κώδικας

Η 4DSystems παρέχει στους χρήστες της μια καινούργια γλώσσα.
Διαθέτει ποικιλία στους επεξεργαστές όπως GOLDELOX-GFX, PICASO-GFX και DIABLO-GFX.

Το EVE είναι ένας εικονικός επεξεργαστής υψηλής απόδοσης με ένα σύνολο εκτεταμένων συνόλων εντολών για την βέλτιστή εκτέλεση συντάξεων στην 4DGL (γλώσσα γραφικών). Είναι μία γλώσσα υψηλού επιπέδου η οποία είναι εύκολη στη χρήση και στην κατανόηση για την δημιουργία εφαρμογών.

Η 4DGLείναι μία γραφική γλώσσα η οποία επιτρέπει την γρήγορη ανάπτυξη εφαρμογών. Μία εκτεταμένη βιβλιοθήκη γραφικών, κειμένων και πολλών άλλων για την ευκολία χρήσης της γλώσσας . Η σύνταξη του κώδικα θυμίζει την γλώσσα Cμε κάποιες σημαντικές αλλαγές. Έτσι, οι χρήστες που γνωρίζουν τις γλώσσες C, C++, Pascal και Basicθα είναι εξοικειωμένοι και με την γλώσσα της συγκεκριμένης εταιρίας. Περιλαμβάνει πολλές γνωστές εντολές όπως οι ακόλουθες: if...else...endif, while…wend, repeat…untilκαθώς και functionsπου περιλαμβάνουν: serin…serout, gfx_line, gfx_circleκαι πολλές άλλες.

Η παρακάτω εικόνα παρουσιάζει τον τρόπο λειτουργίας.



3.1 Ανάπτυξη κώδικα

Για την ανάπτυξη του κώδικα χρησιμοποιήθηκε όπως προαναφέρθηκε το περιβάλλον Visi έτσι και η οθόνη αφής θα έχει τα ακόλουθα γραφικά. Σκοπός , είναι ο χρήστης να θέτει τα όρια της θερμοκρασίας και της υγρασίας να στέλνει την πληροφορία στον μικροελεγκτή. Έτσι όταν λάβει τις μετρήσεις να κάνει έλεγχο των τιμών και να εμφανίσει το αποτέλεσμα. Ο χρήστης θα μπορεί να επαναλάβει τις μετρήσεις όσες φορές θέλει. Τέλος, η χρήση του θα σταματά με το κουμπί στο off. Θα ακολουθήσει μία πρώτη εικόνα με τα γραφικά.

Τα γραφικά είναι ήδη διαμορφωμένα με τέτοιο τρόπο που διευκολύνει την ανάπτυξη του κώδικα κατά τον οποίο χρειάζεται να δημιουργηθεί για την ολοκλήρωση του συγκεκριμένου έργου. Ο κώδικας των γραφικών θα αναπτυχθεί παρακάτω. Αναλυτικά,

Προγραμματισμός οθόνης αφής

//**** VISI BUTTONS AND UI FUNCTIONALITY ****// Ο παρακάτω κώδικας αφορά την δημιουργία των γραφικών (κουμπιά, κείμενο, πλαίσια κτλ)
// και την λειτουργικότητας τους.

```
    // ***** ΑΡΧΙΚΗ ΟΘΟΝΗ

    // Strings0 1.1
    txt_FontID(hFont1) ; // Font index correct at time of code generation
    txt_FGcolour(WHITE) ;
    txt_BGcolour(BLACK) ;
    gfx_MoveTo(0 , 48) ;
    PrintDiskUnicode(hstrings, Strings0StartH, Strings0StartL, Strings0Size, i)
; // where i is Message 0 - Strings0Count-1

    // Κώδικας κουμπιού "Έναρξη"
    img_ClearAttributes(hndl, iWinbutton1, I_TOUCH_DISABLE); // set to
enable touch, only need to do this once
    img_Show(hndl, iWinbutton1);  // show button, only do this once
    img_SetWord(hndl, iWinbutton1, IMAGE_INDEX, state); // where state is 0
for up and 1 for down
    img_Show(hndl,iWinbutton1) ;


   // **** ΟΘΟΝΗ ΕΙΣΑΓΩΓΗΣ ΕΥΡΟΥΣ ΘΕΡΜΟΚΡΑΣΙΑΣ
   // Statictext6 1.0
   img_Show(hndl,iStatictext6) ;   // Πλαίσιο κειμένου

   // Statictext7 MIN TEMP LABEL   // Κείμενο MIN
   img_Show(hndl,iStatictext7) ;


   // Statictext8 1.0 MAX TEMP LABEL  // Κείμενο MAX
   img_Show(hndl,iStatictext8) ;


   // Border2
   // Borders are 'embedded' in the background of each form, this means even
a form without a background image will have one created if it has a border.
   // Borders, Gradients and Scales are all done this way, so you only need to
Paste the main form, or one of them, if several are included on a form.
   img_Show(hndl,iForm2) ;

   // Strings1 1.1 MIN TEMP INPUT  // Πλαίσιο εισαγωγής χαμηλού ορίου
θερμοκρασίας
   txt_FontID(FONT3) ;
   txt_FGcolour(WHITE) ;
   txt_BGcolour(BLACK) ;
   gfx_MoveTo(24 , 100) ;
```

Προγραμματισμός οθόνης αφής

```
// Border3 1.1
// Borders are 'embedded' in the background of each form, this means even a form without a background image will have one created if it has a border.
// Borders, Gradients and Scales are all done this way, so you only need to Paste the main form, or one of them, if several are included on a form.
img_Show(hndl,iForm2);


// Strings2 1.1 MAX TEMP INPUT  // Πλαίσιο εισαγωγής υψηλού ορίου θερμοκρασίας
txt_FontID(FONT3) ;
txt_FGcolour(WHITE) ;
txt_BGcolour(BLACK) ;

// ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΕΙΣΑΓΩΓΗ ΤΟΥ ΠΛΗΚΤΡΟΛΟΓΙΟΥ

img_Show(hndl,iKeyboard0) ;
for (i := iKeyboard0+1; i <= iKeyboard0+oKeyboard0[KbButtons]; i++)
   img_ClearAttributes(hndl, i, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
next
// keydown event
if ((n >= iKeyboard0) && (n <= iKeyboard0+oKeyboard0[KbButtons]))
   kbDown(iKeyboard0, oKeyboard0, iKeyboard0keystrokes, n-iKeyboard0, KbHandler) ;
endif
// keyup event
if (oKeyboard0[KbDown] != -1) kbUp(iKeyboard0, oKeyboard0) ;
gfx_MoveTo(148 , 100) ;


// ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΕΙΣΑΓΩΓΗ ΤΟΥ ΚΟΥΜΠΙΟΥ "Επόμενο"

img_ClearAttributes(hndl, iWinbutton2, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
img_Show(hndl, iWinbutton2);  // show button, only do this once
img_SetWord(hndl, iWinbutton2, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
img_Show(hndl,iWinbutton2) ;

// **** ΟΘΟΝΗ ΥΓΡΑΣΙΑΣ

// Statictext9 1.0    // Πλάισιο κειμένου
img_Show(hndl,iStatictext9) ;

// Statictext10 1.0 // Πλαίσιο εισαγωγής χαμηλού ορίου υγρασίας
img_Show(hndl,iStatictext10) ;

// Statictext11 1.0 // Πλαίσιο εισαγωγής υψηλού ορίου υγρασίας
img_Show(hndl,iStatictext11) ;
```

```
   // Border0
   // Borders are 'embedded' in the background of each form, this means even a form without a background image will have one created if it has a border.
   // Borders, Gradients and Scales are all done this way, so you only need to Paste the main form, or one of them, if several are included on a form.
   img_Show(hndl,iForm3) ;

   // Border1 1.1
   // Borders are 'embedded' in the background of each form, this means even a form without a background image will have one created if it has a border.
   // Borders, Gradients and Scales are all done this way, so you only need to Paste the main form, or one of them, if several are included on a form.
   img_Show(hndl,iForm3) ;



   // Strings3 1.1 HUMIDITY MIN INPUT // Πλαίσιο εισαγωγής χαμηλού ορίου υγρασίας
   txt_FontID(FONT3) ;
   txt_FGcolour(WHITE) ;
   txt_BGcolour(BLACK) ;
   gfx_MoveTo(28 , 96) ;



   // Strings4 1.1 HUMIDITY MAX INPUT  // Πλαίσιο εισαγωγής υψηλόύ ορίου υγρασίας
   txt_FontID(FONT3) ;
   txt_FGcolour(WHITE) ;
   txt_BGcolour(BLACK) ;
   gfx_MoveTo(136 , 96) ;

   //  ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΕΙΣΑΓΩΓΗ ΤΟΥ ΠΛΗΚΤΡΟΛΟΓΙΟΥ

   img_Show(hndl,iKeyboard1) ; // show initial keyboard
   for (i := iKeyboard1+1; i <= iKeyboard1+oKeyboard1[KbButtons]; i++)
      img_ClearAttributes(hndl, i, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
   next
   // keydown event
   if ((n >= iKeyboard1) && (n <= iKeyboard1+oKeyboard1[KbButtons]))
      kbDown(iKeyboard1, oKeyboard1, iKeyboard1keystrokes, n-iKeyboard1, KbHandler) ;
   endif
   // keyup event
   if (oKeyboard1[KbDown] != -1) kbUp(iKeyboard1, oKeyboard1) ;

   //  ΚΩΔΙΚΑΣ ΓΙΑ ΤΗΝ ΕΙΣΑΓΩΓΗ ΤΟΥ ΚΟΥΜΠΙΟΥ "Επόμενο"
```

```
   img_ClearAttributes(hndl, iWinbutton3, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
   img_Show(hndl, iWinbutton3);  // show button, only do this once
   img_SetWord(hndl, iWinbutton3, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
   img_Show(hndl,iWinbutton3) ;



   // **** ΟΘΟΝΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΝΤΟΣ ΟΡΙΩΝ

   // Statictext12 1.0            // Οι παρακάτω "Statictext" μεταβλήτες αφορούν την εισαγωγή των πλαισίων κειμένου στην τελική οθόνη.
   img_Show(hndl,iStatictext12) ;

   // Statictext16 1.0
   img_Show(hndl,iStatictext16) ;

   // Statictext13 1.0
   img_Show(hndl,iStatictext13) ;

   // Statictext17 1.0
   img_Show(hndl,iStatictext17) ;

   // Statictext14 1.0
   img_Show(hndl,iStatictext14) ;

   // ΚΩΔΙΚΑΣ ΕΙΣΑΓΩΓΗΣ ΕΙΚΟΝΙΔΙΟΥ ΘΕΡΜΟΜΕΤΡΟΥ
   // Thermometer1 1.0
   img_SetWord(hndl, iThermometer1, IMAGE_INDEX, frame) ; // where frame is 0 to 100 (for a displayed -1 to -1)
   img_Show(hndl,iThermometer1) ;

   // Statictext15 1.0
   img_Show(hndl,iStatictext15) ;

   // ΚΩΔΙΚΑΣ ΕΙΣΑΓΩΓΗΣ ΕΙΚΟΝΙΔΙΟΥ ΜΕΤΡΗΣΗΣ ΥΓΡΑΣΙΑΣ
   // Coolgauge1 1.0
   img_SetWord(hndl, iCoolgauge1, IMAGE_INDEX, frame) ; // where frame is 0 to 100 (for a displayed -1 to -1)
   img_Show(hndl,iCoolgauge1) ;



   // ***** ΟΘΟΝΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΕΚΤΟΣ ΟΡΙΩΝ
   // Form0

   // ΚΩΔΙΚΑΣ ΕΙΣΑΓΩΓΗΣ ΕΙΚΟΝΙΔΙΟΥ ΘΕΡΜΟΜΕΤΡΟΥ

   // Thermometer0 1.0
```

```
   img_SetWord(hndl, iThermometer0, IMAGE_INDEX, frame) ; // where frame is 0 to 100 (for a displayed -1 to -1)
   img_Show(hndl,iThermometer0) ;

   // ΚΩΔΙΚΑΣ ΕΙΣΑΓΩΓΗΣ ΕΙΚΟΝΙΔΙΟΥ ΜΕΤΡΗΣΗΣ ΥΓΡΑΣΙΑΣ

   // Coolgauge0 1.0
   img_SetWord(hndl, iCoolgauge0, IMAGE_INDEX, frame) ; // where frame is 0 to 100 (for a displayed -1 to -1)
   img_Show(hndl,iCoolgauge0) ;

   // Label1 1.0        // Ετικέτα
   txt_FGcolour(WHITE) ;
   txt_BGcolour(BLACK) ;
   gfx_MoveTo(21, 126) ;
   putstr("θερμοκρασια") ;

   // Statictext0 1.0              // Οι παρακάτω "Statictext" μεταβλήτες αφορούν την εισαγωγή των πλαισίων κειμένου στην τελική οθόνη.
   img_Show(hndl,iStatictext0) ;

   // Statictext1 1.0
   img_Show(hndl,iStatictext1) ;

   // Statictext2 1.0
   img_Show(hndl,iStatictext2) ;

   // Statictext3 1.0
   img_Show(hndl,iStatictext3) ;

   // Statictext4 1.0
   img_Show(hndl,iStatictext4) ;

   // Statictext5 1.0
   img_Show(hndl,iStatictext5) ;

   // ΚΩΔΙΚΑΣ ΕΙΣΑΓΩΓΗΣ ΚΟΥΜΠΙΟΥ "Αρχική"

   img_ClearAttributes(hndl, iWinbutton0, I_TOUCH_DISABLE); // set to enable touch, only need to do this once
   img_Show(hndl, iWinbutton0);  // show button, only do this once
   img_SetWord(hndl, iWinbutton0, IMAGE_INDEX, state); // where state is 0 for up and 1 for down
   img_Show(hndl,iWinbutton0) ;

#platform "µLCD-43"


// Ο παρακάτω κώδικας αφορά την εκτέλεση των γεγονότων της οθόνης σύμφωνα με τις εντολές που δίνει ο χρήστης.
```

Προγραμματισμός οθόνης αφής

// Ο παρακάτω κώδικας παράχθηκε στο περιβάλλον της εφαρμογής Visi Genie.

```
#MODE RUNFLASH

#inherit "4DGL_16bitColours.fnc"

#inherit "VisualConst.inc"


#inherit "Temp_Hum_GUIConst.inc"

#inherit "CLPrintStrings.inc"

#constant IPDatasize 22

#CONST
   CMDLenMAX   80
   seroutX     $serout
   serinX      $serin
#END

#CONST
   ColorBGimage     0x0020
   ACK         0x06
   NAK         0x15
   ReadCmd     0x80
   WriteCmd    0x00
//  IPD_TYPE    0 // offsets are doubled as FLASH is byte addressable
   Ofs_IPD_P1      2
   Ofs_IPD_P2      4
   Ofs_IPD_P3      6
   Ofs_IPD_P4      8
   Ofs_IPD_P5      10
   Ofs_IPD_P6      12
   Ofs_IPD_P7      14
   Ofs_IPD_DOWN    16
   Ofs_IPD_RELEASE 18
   Ofs_IPD_OBJVIDX 20
// object indexes
   tDipSwitch      0
   tKnob           1
   tRockerSwitch   2
   tRotarySwitch   3
   tGSlider        4
   tTrackbar       5
   tWinButton      6
   tAngularmeter   7
   tCoolgauge      8
   tCustomdigits   9
```

```
    tForm          10
    tGauge         11
    tImage         12
    tKeyboard      13
    tLed           14
    tLeddigits     15
    tMeter         16
    tStrings       17
//    tStringUNI      0x3f | 0x40
//    tStringANSII    0x3f
    tThermometer   18
    tUserled       19
    tVideo         20
    tStaticText    21
// Remove, check for non visual objects instead
//    MaxVisObjects   21
    tSounds        22
    tTimer         23
    tSpectrum      24
    tScope         25
    tTank          26
    tUserImages    27
    tPinOutput     28
    tPinInput      29
    t4Dbutton      30
    tAniButton     31
    tColorPicker   32
    tUserButton    33
    tMagicObject   34
    MaxTotObjects  33
//    OT_DISPLAY     22
    OT_REPORT      100
    OT_SETCONST    101
    OT_SETANOTHER  102
    OT_ACTIVATE    103
    OT_NEXTFRAME   104
    OT_PREVFRAME   105
    OT_NEXTSTRING  106
    OT_PREVSTRING  107
    OT_MAGIC       108
// other OT_s Form activate,
// Indexes into LedDigits and CustomDigits arrays
    Ofs_Digits_Left          0
    Ofs_Digits_Digits        2
    Ofs_Digits_MinDigits     4
    Ofs_Digits_Widthdigit    6
    Ofs_Digits_LeadingBlanks 8
// indexes to Strings arrays
    Ofs_String_StartH        0
    Ofs_String_StartL        2
```

```
    Ofs_String_Size         4
    Ofs_String_x1           6
    Ofs_String_y1           8
    Ofs_String_x2           10
    Ofs_String_y2           12
    Ofs_String_FGColor      14
    Ofs_String_BGColor      16
    Ofs_String_FontAttribs  18
    Ofs_String_Transparent  20 // bit transparent should 'refresh' background,
otherwise rectangle out
    Ofs_String_Ansi         22 // bit defines write/draw routine
    Ofs_String_Form         24 // form this string can be seen in
// Command codes
    READ_OBJ        0
    WRITE_OBJ       1
    WRITE_STR       2
    WRITE_STRU      3
    WRITE_CONTRAST  4
    REPORT_OBJ      5
    REPORT_EVENT    7
    WRITE_MAGIC_BYTES 8
    WRITE_MAGIC_DBYTES  9
    REPORT_MAGIC_EVENT_BYTES 10
    REPORT_MAGIC_EVENT_DBYTES 11
// End P1.inc
    nObjects        64
    nInputs         8
    nAniTimers      0
    nKeyboards      2
    nStrings        5
#END

#CONST
    Ofs_kb_Down
    Ofs_kb_Mvt
    Ofs_kb_State
    Ofs_kb_Ign
    Ofs_kb_Caps
    Ofs_kb_Shift1
    Ofs_kb_Shift2
    Ofs_kb_Ctrl1
    Ofs_kb_Ctrl2
    Ofs_kb_Lock
    Ofs_kb_Buttons
    Ofs_kb_ShiftCaps
#END

#constant KbShiftBit        01
#constant KbCapsBit         02
#constant KbShiftCapsBits   03
```

Προγραμματισμός οθόνης αφής

```
#constant KbCtrlBit        04

func refreshstate(var iKB, var oKB)
   var shifted ;
   shifted := oKB[Ofs_kb_State] & KbShiftCapsBits ;
   if (!shifted || (shifted == KbShiftCapsBits))
      shifted := 0 ;
      oKB[Ofs_kb_Caps] := 0 ;
   else
      shifted := 1 ;
      oKB[Ofs_kb_Caps] := 2 ;
   endif
   setkeystate(iKB,shifted) ;
   if (oKB[Ofs_kb_State] & KbCapsBit)
      setkeystate(iKB + oKB[Ofs_kb_Lock],1) ;
   endif
   if ((oKB[Ofs_kb_State] & KbShiftBit) && (shifted))
      setkeystate(iKB + oKB[Ofs_kb_Shift1],1) ;
      setkeystate(iKB + oKB[Ofs_kb_Shift2],1) ;
   endif
   if (oKB[Ofs_kb_State] & KbCtrlBit)
      setkeystate(iKB + oKB[Ofs_kb_Ctrl1],1) ;
      setkeystate(iKB + oKB[Ofs_kb_Ctrl2],1) ;
   endif
endfunc

func kbDown(var iKB, var oKB, var KBKeys, var key)
   var keyval, rtn ;
   oKB[Ofs_kb_Mvt] := 1 ;
   oKB[Ofs_kb_Ign]  := 0 ;
   if ((key == oKB[Ofs_kb_Shift1]) || (key == oKB[Ofs_kb_Shift2]))
      if (oKB[Ofs_kb_State] & KbShiftBit)
         oKB[Ofs_kb_State] &= ~KbShiftBit ;
         oKB[Ofs_kb_Mvt] := 0 ;
      else
         oKB[Ofs_kb_State] |= KbShiftBit ;
      endif
      refreshstate(iKB, oKB) ;
      oKB[Ofs_kb_Ign] := 1 ;
   else if ((key == oKB[Ofs_kb_Ctrl1]) || (key == oKB[Ofs_kb_Ctrl2]))
      if (oKB[Ofs_kb_State] & KbCtrlBit)
         oKB[Ofs_kb_State] &= ~KbCtrlBit ;
         oKB[Ofs_kb_Mvt] := 0 ;
      else
         oKB[Ofs_kb_State] |= KbCtrlBit ;
      endif
      setkeystate(iKB+oKB[Ofs_kb_Ctrl1],oKB[Ofs_kb_Mvt]) ;
      key := oKB[Ofs_kb_Ctrl2] ;
      oKB[Ofs_kb_Ign] := 1 ;
   else if (key == oKB[Ofs_kb_Lock])
```

```
        if (oKB[Ofs_kb_State] & KbCapsBit)
            oKB[Ofs_kb_State] &= ~KbCapsBit ;
            oKB[Ofs_kb_Mvt] := 0 ;
        else
            oKB[Ofs_kb_State] |= KbCapsBit ;
        endif
        refreshstate(iKB, oKB) ;
        oKB[Ofs_kb_Ign] := 1 ;
    endif

    if (!oKB[Ofs_kb_Ign])
        if (oKB[Ofs_kb_ShiftCaps])
            keyval := (oKB[Ofs_kb_State] & KbShiftCapsBits) *
oKB[Ofs_kb_Buttons] - 1 ;
        else if (((oKB[Ofs_kb_State] & KbShiftCapsBits) == 0) ||
((oKB[Ofs_kb_State] & KbShiftCapsBits) == KbShiftCapsBits))
                keyval := - 1 ;
            else
                keyval := oKB[Ofs_kb_Buttons] - 1 ;
        endif
        keyval := KBKeys[key+keyval] ;
        if (oKB[Ofs_kb_State] & KbCtrlBit) keyval &= 0x9F ;
//        SendReport(REPORT_EVENT, tKeyboard, actKB, keyval & 0xff) ;
// sendReport, or KBEvent(actKB, keyval & 0xff)
    rtn := rKeyboardRoutines[ActiveKeyboard] ;
        rtn(REPORT_EVENT, tKeyboard, ActiveKeyboard, keyval & 0xff) ;
        setkeystate(iKB+key,oKB[Ofs_kb_Mvt]+oKB[Ofs_kb_Caps]) ;
    endif
    oKB[Ofs_kb_Down] := key ;
endfunc

func setkeystate(var key, var idx)
    img_SetWord(hndl, key,IMAGE_INDEX, idx);
    img_Show(hndl,key) ;
endfunc

func kbUp(var iKB, var oKB)
    if (!oKB[Ofs_kb_Ign])
        setkeystate(iKB + oKB[Ofs_kb_Down],oKB[Ofs_kb_Caps]) ;
        if (oKB[Ofs_kb_State] & KbShiftBit)
            oKB[Ofs_kb_State] &= ~KbShiftBit ;
            refreshstate(iKB, oKB) ;
        endif
        if (oKB[Ofs_kb_State] & KbCtrlBit)
            oKB[Ofs_kb_State] &= ~KbCtrlBit ;
            setkeystate(iKB + oKB[Ofs_kb_Ctrl1],0) ;
            setkeystate(iKB + oKB[Ofs_kb_Ctrl2],0) ;
        endif
        oKB[Ofs_kb_Down] := -1 ;
    endif
```

```
endfunc

#DATA
   word FormStartIndex 0, 9, 10, 33, 56
   word FormEndIndex 9, 10, 33, 56, 64
   word InputControls -1, -1, -1, -1, -1, -1, -1, -1, 0, 44, -1, 88, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 88, -1, -1, -1, -1, -1, -1, -1, 132, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 132
   word InputData tWinButton, 1, -1, -1, -1, -1, -1, -1, 0, 22, 2,
            OT_ACTIVATE, 10, 1, 0, 18, 0, 0, 0, 0, 0, 0,
            tWinButton, 1, -1, -1, -1, -1, -1, -1, 0, 66, 4,
            OT_ACTIVATE, 10, 2, 0, 18, 0, 0, 0, 0, 0, 0,
            tWinButton, 1, -1, -1, -1, -1, -1, -1, 0, 110, 6,
            OT_ACTIVATE, 10, 3, 0, 18, 0, 0, 0, 0, 0, 0,
            tWinButton, 1, -1, -1, -1, -1, -1, -1, 0, 154, 8,
            OT_ACTIVATE, 10, 0, 0, 18, 0, 0, 0, 0, 0, 0
   word iStrings0  Strings0StartH, Strings0StartL, Strings0Size, 0, 48, 239,
136, WHITE, BLACK, 0, 1, 0, 1
   word iStrings1  Strings1StartH, Strings1StartL, Strings1Size, 24, 100, 95,
135, WHITE, BLACK, 0, 1, 1, 2
   word iStrings2  Strings2StartH, Strings2StartL, Strings2Size, 148, 100, 219,
135, WHITE, BLACK, 0, 1, 1, 2
   word iStrings3  Strings3StartH, Strings3StartL, Strings3Size, 28, 96, 100,
132, WHITE, BLACK, 0, 1, 1, 3
   word iStrings4  Strings4StartH, Strings4StartL, Strings4Size, 136, 96, 206,
134, WHITE, BLACK, 0, 1, 1, 3
   word oDipSwitchs 0
   word oKnobs 0
   word oRockerSwitchs 0
   word oRotarySwitchs 0
   word oGSliders 0
   word oTrackbars 0
   word oWinButtons 4, iWinbutton0, iWinbutton1, iWinbutton2, iWinbutton3
   word oAngularmeters 0
   word oCoolgauges 2, iCoolgauge0, iCoolgauge1
   word oCustomdigitss 0
   word oForms 5, -1, -1, iForm2, iForm3, -1
   word oGauges 0
   word oImages 0
   word oKeyboards 2, iKeyboard0, iKeyboard1
   word oLeds 0
   word oLeddigitss 0
   word oMeters 0
   word oStringss 5, iStrings0, iStrings1, iStrings2, iStrings3, iStrings4
   word oThermometers 2, iThermometer0, iThermometer1
   word oUserleds 0
   word oVideos 0
   word oStaticTexts 18, iStatictext0, iStatictext1, iStatictext2, iStatictext3,
iStatictext4, iStatictext5, iStatictext6, iStatictext7, iStatictext8, iStatictext9,
```

```
iStatictext10, iStatictext11, iStatictext12, iStatictext13, iStatictext14,
iStatictext15, iStatictext16, iStatictext17
   word oSpectrums 0
   word oScopes 0
   word oTanks 0
   word oUserImagess 0
   word oPinInputs 0
   word o4Dbuttons 0
   word oAniButtons 0
   word oColorPickers 0
   word oUserButtons 0
   word oMagicObjects 0
   word oSmartGauges 0
   word oSmartSliders 0
   word oSmartKnobs 0
   word oTimers 0
   word oSoundss 0
   word oPinOutputs 0
   word FormBGcolors 0x0000, 0x0000, ColorBGimage, ColorBGimage,
0x0000
   word kKeyboardKeystrokes iKeyboard0keystrokes, iKeyboard1keystrokes
   word rKeyboardRoutines SendReport, SendReport
#END

var hFonts[5] ;
var stringsCV[5] := [0, 0, 0, 0, 0], hstrings ;
var dKeyboard[2], ActiveKeyboard ;
// Start P2.inc
var oObjects[MaxTotObjects+1] ;
var CurrentForm ;
var TouchXpos, TouchYpos ;
var InputType, TouchState, CurInputData, pInputIndex ;
var comRX[40], cmd[CMDLenMAX] ;

var InputCS, OutputCS ;



func seroutCS(var op)
   serout(op) ;
   OutputCS ^= op ;
endfunc

func nak0()
   serout(NAK) ;
   InputCS := 0 ;
endfunc

func seroutOcs()
   serout(OutputCS) ;
```

```
    OutputCS := 0 ;
endfunc

func SendReport(var id, var objt, var objn, var val)
    seroutCS(id) ;
    seroutCS(objt) ;
    seroutCS(objn) ;
    seroutCS(val >> 8) ; // first 8 bits
    seroutCS(val) ;
    seroutOcs() ;
endfunc


func ReadObject(var ObjectType, var ObjectIdx)
    var j, k, Objects ;
    Objects := *(oObjects+ObjectType) ;

    j := 2 + ObjectIdx * 2 + Objects ;
     if (ObjectType == tForm)
        k := CurrentForm ;
    else if (ObjectType == tStrings)
        k := stringsCV[ObjectIdx];
        else
        k := img_GetWord(hndl, *j, IMAGE_INDEX);
        if (((ObjectType == tWinButton) || (ObjectType == tAniButton) ||
(ObjectType == tUserButton) || (ObjectType == t4Dbutton)) && (k)) k := 1 ; //
this is not correct for blocked buttons and cannot be fixed as we cannot

// determine if button is momentary or not which is needed for correct answer.
    endif
    SendReport(REPORT_OBJ, ObjectType, ObjectIdx, k) ;
endfunc


func WriteObject(var ObjectType, var ObjectIdx, var NewVal)
    var i, j, k, Objects ;
    ObjectType &= 0x3f ;
    if (ObjectType == tForm)
        ActivateForm(ObjectIdx) ;
    else
        Objects := *(oObjects+ObjectType)+ObjectIdx*2+2 ;
        i := *(Objects) ;
        switch (ObjectType)
            case tWinButton :
                j := InputControls[oWinButtons[ObjectIdx+1]] ;
                break ;
            default : j := -1 ;
        endswitch
        if (j != -1)
            k := img_GetWord(hndl, i , IMAGE_INDEX) ;
```

```
            NewVal := NewVal << 1 ;
            if (OVF())      // button group index change
               if (*(j+InputData+Ofs_IPD_P1))
                  k &= 1 ;    // mask off old group index for momentary
               else
                  k &= 3 ;    // mask off old group index for toggle
               endif
            else           // normal set
               if (*(j+InputData+Ofs_IPD_P2) != -1)
TurnOffButtons(*(j+InputData+Ofs_IPD_P2)) ;
               k &= 0xfffc ;    // retain only group index for state set
            endif
            NewVal |= k ;
         endif
          if (ObjectType == tStrings)
            PrintStrings(ObjectIdx, NewVal, 0);
         else
            img_SetWord(hndl, i , IMAGE_INDEX, NewVal);
            img_Show(hndl, i) ; // will only display if form is current
         endif
      endif
endfunc

func TurnOffButtons(var group)
   var j, k, l;
   for (j := 0; j < nInputs; j++)
      k := j*IPDatasize ;
      if (*(InputData+k+Ofs_IPD_P2) == group)
         l := -1 ;
          if (*(InputData+k) == tWinButton)
            l := oWinButtons[*(InputData+k+Ofs_IPD_OBJVIDX)/2] ;
            img_SetWord(hndl, l, IMAGE_INDEX, 0);
            img_Show(hndl, l);      // only shows on current form
         endif
      endif
   next
endfunc




func ActivateForm(var newform)
   var i, j, *p ;

   if (CurrentForm != -1) // deactivate old form, by disabling all inputs
      for (i := FormStartIndex[CurrentForm]; i <= FormEndIndex[CurrentForm];
i++)
         if (img_GetWord(hndl, i, IMAGE_TAG))
            img_Disable(hndl,i) ;
         endif
```

```
        next
    endif
    CurrentForm := newform ;
    // display newform image or clear to image color
    if (FormBGcolors[CurrentForm] != ColorBGimage)
        gfx_Set(BACKGROUND_COLOUR,FormBGcolors[CurrentForm]);
        gfx_Cls() ;
        DoGFXObjects() ;                        // display GFX 'widgets'
    endif

    // enable inputs
    for (i := FormStartIndex[CurrentForm]; i < FormEndIndex[CurrentForm];
i++)
        j := img_GetWord(hndl, i, IMAGE_TAG) ;
        if (j)
            j-- ;
            img_SetAttributes(hndl, i, I_STAYONTOP+I_ENABLED);
            //if (j != tKeyboard)
            if ((j <= tWinButton) || (j >= t4Dbutton) )
                img_ClearAttributes(hndl, i, I_TOUCH_DISABLE);
            endif
            img_Show(hndl,i) ; // show initialy, if required
            if (j == tForm)
                DoGFXObjects() ;
            endif
        endif
    next
    for (i := 0; i < nStrings; i++)
        if (stringsCV[i] != -1)
            WriteObject(tStrings, i, stringsCV[i]) ;
        endif
    next

endfunc

func UpdateObjects(var newval)
    var IPidx, otherOBJ ;
    if ( ( img_GetWord(hndl, *(pInputIndex), IMAGE_INDEX) != newval) ||
(TouchState == Ofs_IPD_RELEASE) )
        img_SetWord(hndl, *(pInputIndex), IMAGE_INDEX, newval);
            img_Show(hndl, *(pInputIndex));      // only shows on current form
        if ((InputType == t4Dbutton) || (InputType == tUserButton) || (InputType
== tWinButton))
            if (*(CurInputData+Ofs_IPD_P1))
                newval &= 1;
            else
                newval &= 3;
            endif
            if (newval > 1) newval := 1;
        endif
```

Προγραμματισμός οθόνης αφής

```
      IPidx := *(CurInputData+TouchState) ;
      while(IPidx != 0)
         otherOBJ := IPidx + InputData;
         if (*(otherOBJ) == OT_REPORT)
      SendReport(REPORT_EVENT, InputType,
*(otherOBJ+Ofs_IPD_OBJVIDX), newval) ;
         else if (*(otherOBJ) == OT_MAGIC)
            IPidx := *(otherOBJ+Ofs_IPD_P5) ;
            IPidx(newval) ;
         else if (TouchState == *(otherOBJ+Ofs_IPD_P4))
            if (*(otherOBJ) == OT_ACTIVATE)
               ActivateForm(*(otherOBJ+Ofs_IPD_P2) ) ;
               InputType := tForm ;
            else if (*(otherOBJ) == OT_SETCONST)
               newval := *(otherOBJ+Ofs_IPD_P3) ;
               WriteObject(*(otherOBJ+Ofs_IPD_P1),
*(otherOBJ+Ofs_IPD_P2), newval) ;
            else if (*(otherOBJ) == OT_SETANOTHER)
               WriteObject(*(otherOBJ+Ofs_IPD_P1),
*(otherOBJ+Ofs_IPD_P2), newval) ;
            else if (*(otherOBJ) == OT_PREVFRAME)
               if (img_GetWord(hndl, *(otherOBJ+Ofs_IPD_P6),
IMAGE_INDEX))

WriteObject(*(otherOBJ+Ofs_IPD_P5),*(otherOBJ+Ofs_IPD_P2),img_GetWor
d(hndl, *(otherOBJ+Ofs_IPD_P6), IMAGE_INDEX)-1) ;
               endif
               newval := img_GetWord(hndl, *(otherOBJ+Ofs_IPD_P6),
IMAGE_INDEX) ;
            else if (*(otherOBJ) == OT_NEXTFRAME)
               if (img_GetWord(hndl, *(otherOBJ+Ofs_IPD_P6),
IMAGE_INDEX) < *(otherOBJ+Ofs_IPD_P3))

WriteObject(*(otherOBJ+Ofs_IPD_P5),*(otherOBJ+Ofs_IPD_P2),img_GetWor
d(hndl, *(otherOBJ+Ofs_IPD_P6), IMAGE_INDEX)+1) ;
               endif
               newval := img_GetWord(hndl, *(otherOBJ+Ofs_IPD_P6),
IMAGE_INDEX) ;
            else if (*(otherOBJ) == OT_PREVSTRING)
               if (stringsCV[*(otherOBJ+Ofs_IPD_P2)])

WriteObject(tStrings,*(otherOBJ+Ofs_IPD_P2),stringsCV[*(otherOBJ+Ofs_IP
D_P2)]-1) ;
               endif
            else if (*(otherOBJ) == OT_NEXTSTRING)
               if (stringsCV[*(otherOBJ+Ofs_IPD_P2)] <
*(otherOBJ+Ofs_IPD_P3)) // fix IPD_P2 not filled in yet

WriteObject(tStrings,*(otherOBJ+Ofs_IPD_P2),stringsCV[*(otherOBJ+Ofs_IP
D_P2)]+1) ;
```

```
                endif
            endif
        endif
        IPidx := *(otherOBJ+TouchState) ;
    wend
  endif
endfunc

// End P2.inc
func DoGFXObjects()
  switch (CurrentForm)
    case 0:
      txt_FontID(FONT1);
      txt_FGcolour(WHITE) ;
      txt_BGcolour(BLACK) ;
      gfx_MoveTo(21, 126) ;
      putstr("θερμοκρασια") ;
    break ;
  endswitch
endfunc

// Start P3.inc
func main()
  var comTX[50], cmdi, i, j, ImageTouched, TouchStatus, oldn ;

  ActiveKeyboard := -1 ;

  gfx_ScreenMode(PORTRAIT) ;

  putstr("Mounting...\n");
  if (!(file_Mount()))
    while(!(file_Mount()))
      putstr("Drive not mounted...");
      pause(200);
      gfx_Cls();
      pause(200);
    wend
  endif
//   gfx_MoveTo(0, 0);
//   print(mem_Heap()," ") ;
//   gfx_TransparentColour(0x0020);
//   gfx_Transparency(ON);

  // open image control
  hndl := file_LoadImageControl("TEMP_H~1.dat", "TEMP_H~1.gci", 1);

  // init 'constants'
// End P3.inc

  oObjects[tDipSwitch] := oDipSwitchs ;
```

```
    oObjects[tKnob] := oKnobs ;
    oObjects[tRockerSwitch] := oRockerSwitchs ;
    oObjects[tRotarySwitch] := oRotarySwitchs ;
    oObjects[tGSlider] := oGSliders ;
    oObjects[tTrackbar] := oTrackbars ;
    oObjects[tWinButton] := oWinButtons ;
    oObjects[tAngularmeter] := oAngularmeters ;
    oObjects[tCoolgauge] := oCoolgauges ;
    oObjects[tCustomdigits] := oCustomdigitss ;
    oObjects[tForm] := oForms ;
    oObjects[tGauge] := oGauges ;
    oObjects[tImage] := oImages ;
    oObjects[tKeyboard] := oKeyboards ;
    oObjects[tLed] := oLeds ;
    oObjects[tLeddigits] := oLeddigitss ;
    oObjects[tMeter] := oMeters ;
    oObjects[tStrings] := oStringss ;
    oObjects[tThermometer] := oThermometers ;
    oObjects[tUserled] := oUserleds ;
    oObjects[tVideo] := oVideos ;
    oObjects[tStaticText] := oStaticTexts ;
    oObjects[tSounds] := oSoundss ;
    oObjects[tTimer] := oTimers ;
    oObjects[tSpectrum] := oSpectrums ;
    oObjects[tTank] := oTanks ;
    oObjects[tUserImages] := oUserImagess ;
    oObjects[tPinOutput] := oPinOutputs ;
    oObjects[tPinInput] := oPinInputs ;
    oObjects[t4Dbutton] := o4Dbuttons ;
    oObjects[tAniButton] := oAniButtons ;
    oObjects[tColorPicker] := oColorPickers ;
    oObjects[tUserButton] := oUserButtons ;
    hFonts[0] := file_LoadImageControl("TEMP_H~1.d01", "TEMP_H~1.g01",
1)  ;
    hFonts[1] := FONT3 ;
    hFonts[2] := FONT3 ;
    hFonts[3] := FONT3 ;
    hFonts[4] := FONT3 ;
    dKeyboard[0] := oKeyboard0 ;
    dKeyboard[1] := oKeyboard1 ;
// Start P4.inc
    hstrings := file_Open("TEMP_H~1.txf", 'r') ; // Open handle to access uSD
strings, uncomment if required
    // init comms
    com_Init(comRX,CMDLenMAX,0);
    com_SetBaud(COM0,960);
    com_TXbuffer(comTX, 100, 0);
    // tag 'real' objects
    for (i := 0; i <= MaxTotObjects; i++)
        if (   (i != tSounds)
```

```
          && (i != tTimer)
          && (i != tPinOutput)
          && (i != tPinInput) )
          TouchXpos := oObjects[i] ;
          TouchYpos := *(TouchXpos) ;
          for (ImageTouched := 1; ImageTouched <= TouchYpos;
ImageTouched++)
             oldn := *(TouchXpos+ImageTouched*2) ;
             img_SetAttributes(hndl, oldn, I_TOUCH_DISABLE);  // ensure touch
is enabled
             if (oldn != -1)
                img_SetWord(hndl, oldn, IMAGE_TAG, i+1);
                img_Disable(hndl, oldn) ;
             endif
          next
       endif
    next

    for (i := 0; i < nKeyboards; i++) // for each kb key, set tag to -1
       for(ImageTouched := oKeyboards[i+1]+1; ImageTouched <=
oKeyboards[i+1]+*(dKeyboard[i] + Ofs_kb_Buttons); ImageTouched++)
          img_SetWord(hndl, ImageTouched, IMAGE_TAG, -1);
       next
    next
    // display initial form
    CurrentForm := -1 ;
// End P4.inc
// Start P5.inc
    ActivateForm(0) ; // need to change this according to first actual form

// End P5.inc
// Start P6.inc
    touch_Set(TOUCH_ENABLE);                    // enable the touch screen
    oldn := -1 ;
    repeat

       // check comms for command, how to NAK invalid command
       if (com_Count() != 0)
          i := serin() ;
          InputCS ^= i ;              // update checksum
          if (   (cmdi > 2)
             && (cmd[0] == WRITE_STRU) )
             j := (cmdi-1) >> 1 + 2 ;
             if (j == CMDLenMAX)    // max length exceeded
                nak0() ;
                cmdi := -1 ;
             else if (cmdi & 1)
                cmd[j] := i ;
                if (cmd[2] == 0)    // if string complete
                   if (InputCS)
```

```
                    nak0() ;
                 else
                    if (cmd[0] == WRITE_STRU)
                    cmd[j] := 0 ;                    // terminate it
                    PrintStrings(cmd[1], &cmd[3], 1) ;
                    serout(ACK) ;
                    else
                    endif
                 endif
                 cmdi := -1 ;
              endif
           else
              cmd[j] := cmd[j] << 8 + i ;
              cmd[2]-- ;          // dec length
           endif
           cmdi++ ;
        else // not unicode string
           cmd[cmdi++] := i ;
            if (cmd[0] == WRITE_STR)                  // Ansi String
              if (cmdi == CMDLenMAX)      // max length exceeded
                 nak0() ;
                 cmdi := 0 ;
              else if (cmdi > 2)
                 if (cmd[2] == -1)
                    if (InputCS)
                       nak0() ;
                    else
                       if (cmd[0] == WRITE_STR)
                       cmd[cmdi-1] := 0 ;                    // terminate it
                       PrintStrings(cmd[1], &cmd[3], 1) ;
                       serout(ACK) ;
                       else
                       endif
                    endif
                    cmdi := 0 ;
                 else
                    cmd[2]-- ;          // dec length
                 endif
              endif
           else if (   (cmd[0] == READ_OBJ)
                    && (cmdi == 4)         )
              if (InputCS)
                 nak0() ;
              else
                 ReadObject(cmd[1], cmd[2]) ;
              endif
              cmdi := 0 ;
           else if (   (cmd[0] == WRITE_OBJ)    // 6 byte write command (gen
option)
                    && (cmdi == 6)          )
```

```
                if (InputCS)
                   nak0() ;
                else
                   WriteObject(cmd[1], cmd[2], cmd[3] << 8 + cmd[4]) ;
                   serout(ACK) ;
                endif
                cmdi := 0 ;
             else if (   (cmd[0] == WRITE_CONTRAST)
                    && (cmdi == 3)        )
                if (InputCS)
                   nak0() ;
                else
                   gfx_Contrast(cmd[1]) ;
                   serout(ACK) ;
                endif
                cmdi := 0 ;
             else if (cmdi == 6)
                nak0() ;
                cmdi := 0 ;
             endif
          endif   // not unicode string
       endif   // a character is available


   // touch code processing

       TouchStatus := touch_Get(TOUCH_STATUS);              // get
touchscreen status
       ImageTouched := img_Touched(hndl,-1) ;
       if ((TouchStatus == TOUCH_PRESSED) || (TouchStatus ==
TOUCH_RELEASED) || (TouchStatus == TOUCH_MOVING))
          if ((TouchStatus != TOUCH_RELEASED) && (ImageTouched != oldn)
&& (oldn != -1))
             TouchStatus := TOUCH_RELEASED ;       // simulate release if we
move off object
          endif
          if (TouchStatus != TOUCH_RELEASED)       // if not released
             if (oldn != -1)
                ImageTouched := oldn ;
             else
                if (oldn != ImageTouched)
             oldn := ImageTouched ;
                   TouchStatus := TOUCH_PRESSED ;
                endif
             endif
             TouchXpos  := touch_Get(TOUCH_GETX);
             TouchYpos  := touch_Get(TOUCH_GETY);
             TouchState := Ofs_IPD_DOWN ;
          else
```

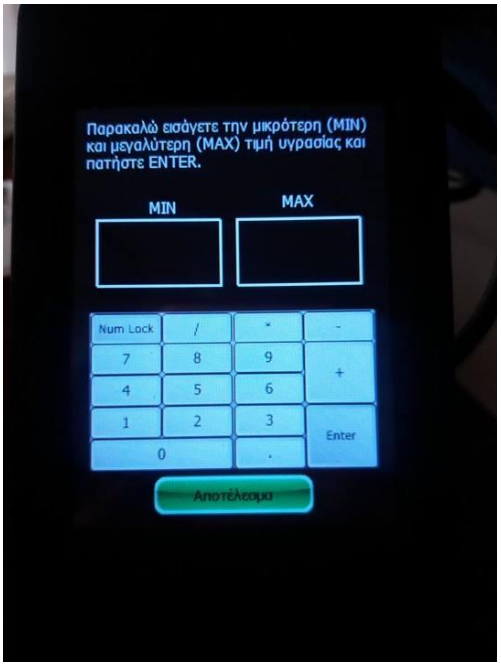
```
            ImageTouched := oldn ;                  // simulate release of what we
touched
            oldn := -1 ;                // prevent double release
            TouchState := Ofs_IPD_RELEASE ;
         endif
         if (ImageTouched != -1)
            // if touch released then find a keyboard down, if one then release it
            if ((TouchStatus == TOUCH_RELEASED) && (ActiveKeyboard != -
1))
               kbUp(oKeyboards[ActiveKeyboard+1],
dKeyboard[ActiveKeyboard]) ;
               ActiveKeyboard := -1 ;
            else
               i := 0 ;
               while ((i < nKeyboards) && ((ImageTouched <= oKeyboards[i+1])
|| (ImageTouched > oKeyboards[i+1] + *(dKeyboard[i]+Ofs_kb_Buttons))))
                  i++ ;
               wend
               if (i < nKeyboards)
                  if (TouchStatus == TOUCH_PRESSED)
                     ActiveKeyboard := i ;
                     kbDown(oKeyboards[ActiveKeyboard+1],
dKeyboard[ActiveKeyboard], kKeyboardKeystrokes[ActiveKeyboard],
ImageTouched-oKeyboards[ActiveKeyboard+1]) ;
                  endif
               else
                  CurInputData := InputControls[ImageTouched] + InputData;
                  InputType := *(CurInputData) ;
                  i := InputType ;
                  if (InputType >= t4Dbutton) i -= 23 ; // adjust to ensure next in
gosub
                  gosub (i), (cDipswitch, cKnob, cRockerswitch, cRotaryswitch,
cSlider, cTrackbar, cWinbutton, c4DButton, cAniButton, cColorPicker,
cUserButton) ;
               endif
            endif
         endif
      endif
//       if ((n != -1) && (oldn == -1)) oldn := n ;    // save what we touched in
case we move off it

      sys_EventsResume() ;
   forever

cDipswitch:

cKnob:

cRockerswitch:
```

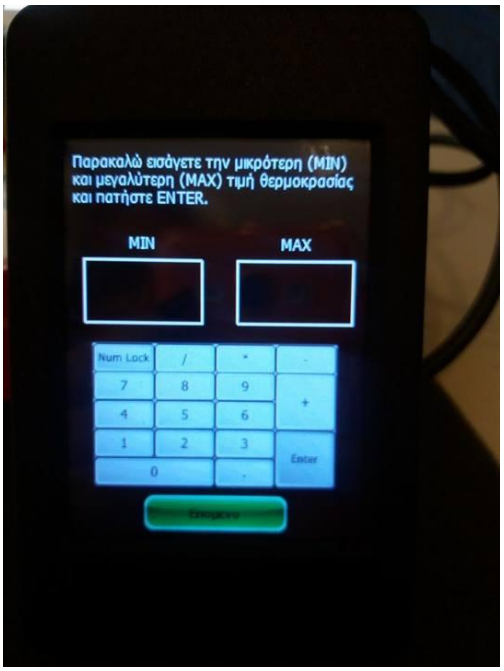
Προγραμματισμός οθόνης αφής

```
cRotaryswitch:

cSlider:
cTrackbar:

c4DButton:
cUserButton:
cWinbutton:
   pInputIndex := oWinButtons + *(CurInputData+Ofs_IPD_OBJVIDX) ;
gbutton:
   i := img_GetWord(hndl, *(pInputIndex), IMAGE_INDEX) ; // current state
   if (*(CurInputData+Ofs_IPD_P1))
      if (TouchStatus == TOUCH_RELEASED)
         i &= 0xfffe ;
         TouchState == Ofs_IPD_DOWN ;
      else
         i |= 1 ;
         TouchState == Ofs_IPD_RELEASE ;
      endif
   else if (*(CurInputData+Ofs_IPD_P2) == -1)
      if (TouchStatus == TOUCH_RELEASED)
         if ((i & 3) == 3)
            i &= 0xfffc ;
         else
            i++ ;
         endif
      else
         i |= 1 ;
      endif
   else                    // group action, up all other buttons on touch press,
reports 0 for button down
      if (TouchStatus == TOUCH_PRESSED)
         TurnOffButtons(*(CurInputData+Ofs_IPD_P2)) ;
      endif
      i := (i & 0xfffc) | 2 ;
   endif

   UpdateObjects(i) ;
endsub ;

cAniButton:

cColorPicker:

endfunc
// End P6.inc
```

Προγραμματισμός οθόνης αφής

Μερικές φωτογραφίες του πρακτικού μέρους του προγραμματισμού της οθόνης





ΒΙΒΛΙΟΓΡΑΦΙΑ

Οι πηγές που αναζητήθηκαν και χρησιμοποιήκαν είναι:

http://www.4dsystems.com.au/product/uLCD_32PTU/
http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_datasheet_R_2_1.pdf
http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_productbrief_R_1_2.pdf
http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_datasheet_R_2_0.pdf
http://www.4dsystems.com.au/productpages/PICASO/downloads/PICASO_productbrief_R_1_0.pdf
http://www.4dsystems.com.au/productpages/uLCD-32PTU/downloads/uLCD-32PTU_productbrief_R_1_2.pdf
https://el.wikipedia.org/wiki/%CE%A4%CE%B7%CE%BB%CE%B5%CF%8C%CF%81%CE%B1%CF%83%CE%B7_%CF%85%CE%B3%CF%81%CF%8E%CE%BD_%CE%BA%CF%81%CF%85%CF%83%CF%84%CE%AC%CE%BB%CE%BB%CF%89%CE%BD
http://www.ceti.gr/digitech2/index.php?option=com_content&task=view&id=94&Itemid=2
Πτυχιακή Στεργίου,Τριαντάφυλλος

Το πρόγραμμα στο οποίο πραγματοποιήθηκε η πρακτική υλοποίηση της πτυχιακής εργασίας είναι:

http://www.4dsystems.com.au/product/4D_Workshop_4_IDE/downloads