



Τμήμα Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών

**ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ**

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ Ι

Νικόλαος Σπ. Βώρος
Καθηγητής

Χρήστος Αντωνόπουλος
Αναπληρωτής Καθηγητής

Θεόδωρος Σκανδάμης
Μηχανικός Τηλεπικοινωνιακών Συστημάτων & Δικτύων

Γιώργος Αλεξάνδρου
Ηλεκτρολόγος Μηχανικός & Μηχανικός Υπολογιστών

Έκδοση 3^η

Μάιος 2025

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ	2
ΠΙΝΑΚΑΣ ΕΡΩΤΗΜΑΤΩΝ	6
ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ	9
ΕΙΣΑΓΩΓΗ	11
1 ΕΡΓΑΣΤΗΡΙΟ 1.....	14
Προϋποθέσεις	14
Εισαγωγή.....	15
1.1 Το Υλικό του Αναπτυξιακού (Hardware).....	16
1.1.1 Διαβάζοντας το σχηματικό	17
1.1.2 STR912FAW44XB	18
1.1.3 Χάρτης Μνήμης του STR912FAW44XB (Memory Map).....	20
1.1.4 LCD, 2x16 αλφαριθμητικό.....	23
1.1.5 Ethernet θύρα	23
1.1.6 CAN 2.0B θύρα.....	24
1.1.7 2x RS-232 σειριακές θύρες.....	24
1.1.8 USB 2.0 θύρα.....	25
1.1.9 Μετατροπέας αναλογικής τάσης (Ποτενσιόμετρο)	26
1.1.10 Πλήκτρα και LEDs.....	26
1.1.11 JTAG θύρα.....	27
2 ΕΡΓΑΣΤΗΡΙΟ 2.....	28
Προϋποθέσεις	28
Εισαγωγή.....	29
2.1 Το παράδειγμα Blinky	30
2.2 Εκτελώντας το παράδειγμα Blinky στο αναπτυξιακό	32
2.3 Το παράδειγμα Blinky. Απλή χρήση του LCD.	34
3 ΕΡΓΑΣΤΗΡΙΟ 3.....	37
Προϋποθέσεις	37
Εισαγωγή.....	38

3.1	Ρυθμίσεις uVision για χρήση του αποσφαλματωτή	40
3.2	Παρακολούθηση κατάστασης καταχωρητών	43
3.3	Χρησιμοποίηση των breakpoints	45
3.4	Χρησιμοποίηση των παραθύρων Disassembly και Registers	47
4	ΕΡΓΑΣΤΗΡΙΟ 4.....	55
	Προϋποθέσεις	55
	Εισαγωγή.....	56
4.1	Τύποι δεδομένων.....	57
4.2	Καταχωρητές προγραμματισμού	57
4.3	Καταστάσεις λειτουργίας.....	58
4.4	Εξαιρέσεις (Exceptions).....	60
4.5	Παρατηρώντας την εκτέλεση ενός προγράμματος	61
5	ΕΡΓΑΣΤΗΡΙΟ 5.....	72
	Προϋποθέσεις	72
	Εισαγωγή.....	72
5.1	Διαδικασία προγραμματισμού GPIO	73
6	ΕΡΓΑΣΤΗΡΙΟ 6.....	77
	Προϋποθέσεις	77
	Εισαγωγή.....	78
6.1	Ανάλυση του κώδικα.....	79
6.2	Εκτέλεση του κώδικα	89
6.3	Δημιουργία κώδικα για το πλήκτρο INT6	90
7	ΕΡΓΑΣΤΗΡΙΟ 7.....	91
	Προϋποθέσεις	91
	Εισαγωγή.....	91
7.1	Ανάλυση του κώδικα.....	93
8	ΕΡΓΑΣΤΗΡΙΟ 8.....	102
	Προϋποθέσεις	102
	Εισαγωγή.....	103

8.1	Εκτέλεση του κώδικα	104
8.2	Έλεγχος μέσο του αποσφαλματωτή	104
9	ΕΡΓΑΣΤΗΡΙΟ 9.....	109
	Προϋποθέσεις	109
	Εισαγωγή.....	110
9.1	Ένδειξη της τάσης του ποτενσιόμετρου στην οθόνη LCD	111
10	ΕΡΓΑΣΤΗΡΙΟ 10	116
	Προϋποθέσεις	116
	Εισαγωγή.....	117
10.1	Εκτέλεση του κώδικα	119
10.2	Ανάλυση του κώδικα.....	119
10.2.1	Αποτύπωση της ώρας	120
10.2.2	Periodic RTC Interrupt	123
10.2.3	Alarm RTC Interrupt.....	124
10.3	Δοκιμή στις αλλαγές ώρας και Alarm.....	125
11	ΕΡΓΑΣΤΗΡΙΟ 11	127
	Προϋποθέσεις	127
	Εισαγωγή.....	128
11.1	Προγραμματισμός Alarm	129
11.2	Απεικόνιση του Alarm στο LCD	131
12	ΕΡΓΑΣΤΗΡΙΟ 12	134
	Προϋποθέσεις	134
	Εισαγωγή.....	135
12.1	Προγραμματισμός και απεικόνιση ημερομηνίας	136
13	ΕΡΓΑΣΤΗΡΙΟ 13	138
	Εισαγωγή.....	138
	Εγκατάσταση e^2 Studio.....	141
	Περιήγηση εφαρμογής e^2 Studio	144
14	ΕΡΓΑΣΤΗΡΙΟ 14	149

15	ΕΡΓΑΣΤΗΡΙΟ 15	160
16	ΕΡΓΑΣΤΗΡΙΟ 16	168
	ΠΑΡΑΡΤΗΜΑ	176

ΠΙΝΑΚΑΣ ΕΡΩΤΗΜΑΤΩΝ

Ερώτημα 1	17
Ερώτημα 2	20
Ερώτημα 3	20
Ερώτημα 4	20
Ερώτημα 5	20
Ερώτημα 6	22
Ερώτημα 7	22
Ερώτημα 8	22
Ερώτημα 9	23
Ερώτημα 10	24
Ερώτημα 11	24
Ερώτημα 12	24
Ερώτημα 13	25
Ερώτημα 14	25
Ερώτημα 15	25
Ερώτημα 16	26
Ερώτημα 17	26
Ερώτημα 18	27
Ερώτημα 19	33
Ερώτημα 20	34
Ερώτημα 21	36
Ερώτημα 22	43
Ερώτημα 23	45
Ερώτημα 24	46
Ερώτημα 25	48
Ερώτημα 26	49
Ερώτημα 27	51
Ερώτημα 28	51
Ερώτημα 29	51
Ερώτημα 30	52
Ερώτημα 31	53
Ερώτημα 32	53
Ερώτημα 33	54
Ερώτημα 34	66

Ερώτημα 35	67
Ερώτημα 36	68
Ερώτημα 37	70
Ερώτημα 38	70
Ερώτημα 39	74
Ερώτημα 40	74
Ερώτημα 41	74
Ερώτημα 42	74
Ερώτημα 43	74
Ερώτημα 44	75
Ερώτημα 45	80
Ερώτημα 46	81
Ερώτημα 47	82
Ερώτημα 48	85
Ερώτημα 49	85
Ερώτημα 50	86
Ερώτημα 51	90
Ερώτημα 52	98
Ερώτημα 53	98
Ερώτημα 54	99
Ερώτημα 55	99
Ερώτημα 56	99
Ερώτημα 57	99
Ερώτημα 58	100
Ερώτημα 59	100
Ερώτημα 60	100
Ερώτημα 61	100
Ερώτημα 62	101
Ερώτημα 63	101
Ερώτημα 64	106
Ερώτημα 65	107
Ερώτημα 66	108
Ερώτημα 67	108
Ερώτημα 68	111
Ερώτημα 69	112
Ερώτημα 70	113
Ερώτημα 71	113

Ερώτημα 72	115
Ερώτημα 73	118
Ερώτημα 74	125
Ερώτημα 75	125
Ερώτημα 76	129
Ερώτημα 77	131
Ερώτημα 78	137
Ερώτημα 79	141
Ερώτημα 80	141
Ερώτημα 81	151
Ερώτημα 82	151
Ερώτημα 83	151
Ερώτημα 84	152
Ερώτημα 85	154
Ερώτημα 86	157
Ερώτημα 87	157
Ερώτημα 88	158
Ερώτημα 89	158
Ερώτημα 90	159
Ερώτημα 91	164
Ερώτημα 92	165
Ερώτημα 93	165
Ερώτημα 94	165
Ερώτημα 95	166
Ερώτημα 96	166
Ερώτημα 97	167
Ερώτημα 98	172
Ερώτημα 99	172
Ερώτημα 100	174
Ερώτημα 101	174
Ερώτημα 102	174
Ερώτημα 103	175

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

Σχήμα 1: MCBSTR9 Keil Development Board	15
Σχήμα 2: Σχηματικό διάγραμμα της MCBSTR9 πλακέτας.....	16
Σχήμα 3: Διάγραμμα MCU της οικογένειας STR9xFAxxx.....	19
Σχήμα 4: Χάρτης Μνήμης του STR912FAW44XB	21
Σχήμα 5: Το περιβάλλον του uVision4.....	29
Σχήμα 6: Επιλογή Project στο uVision3	30
Σχήμα 7: Παράθυρο Project Workspace	31
Σχήμα 8: Build Toolbar	31
Σχήμα 9: Παράθυρο Build Output μετά από επιτυχές build	32
Σχήμα 10: Παράθυρο Build Output μετά από επιτυχές binary κατέβασμα	33
Σχήμα 11: Το όνομα μας στο LCD	34
Σχήμα 12: Αρχική ένδειξη του LCD	34
Σχήμα 13: Παράθυρο Project	35
Σχήμα 14: ULINK2 USB-JTAG μετατροπέας.....	38
Σχήμα 15: Επιλογή εξομοίωσης στο uVision4.....	39
Σχήμα 16: Επιλογές debugger	40
Σχήμα 17: Επιλογές debugger	41
Σχήμα 18: Επιλογή Εκκίνησης Debugger.....	41
Σχήμα 19: Παράθυρο Debugger μVision4.....	42
Σχήμα 20: Επιλογή παρακολούθηση κατάστασης GPIO	43
Σχήμα 21: Παράθυρο καταχωρητών για το GPIO3	43
Σχήμα 22: Επιλογή παρακολούθησης καταχωρητή.....	44
Σχήμα 23: Εκτέλεση κώδικα μέσω του Debugger.....	45
Σχήμα 24: Dissassembly και Registers.....	47
Σχήμα 25: Καταχωρητής στο παράθυρο Watch 1	48
Σχήμα 26: Παράθυρο Watch 1 και αρχική τιμή του SCU_PRR1.....	48
Σχήμα 27: Εκτέλεση Step	51
Σχήμα 28: Παράθυρα στην επιλογή Debug	52
Σχήμα 29: Πίνακας καταχωρητών ARM	59
Σχήμα 30: Πίνακας καταστάσεων ARM.....	60
Σχήμα 31: Πίνακας Εξαιρέσεων ARM.....	60
Σχήμα 32: Παράθυρα στη διαδικασία αποσφαλμάτωσης.....	63
Σχήμα 33: Διάγραμμα ελέγχου διακοπών	92
Σχήμα 34: Επιλογή Wake Up Interrupt Controller	105

Σχήμα 35: Δομή παραθύρων Debugger για έλεγχο του Εξωτερικού Interrupt	105
Σχήμα 36: Breakpoint στην εξυπηρέτηση του Interrupt.....	106
Σχήμα 37: Ένδειξη τάσης ποτενσιόμετρου στο LCD	110
Σχήμα 38: Ένδειξη LCD πριν τις αλλαγές μας	110
Σχήμα 39: Οθόνη Ψηφιακού Ρολογιού	117
Σχήμα 40: Καταχωρητής RTC_TR	120
Σχήμα 41: Πίνακας τιμών καταχωρητή RTC_TR.....	120
Σχήμα 42: Ροή προγράμματος Ψηφιακού Ρολογιού	121
Σχήμα 43: Διαδικασία RTC Interrupt	124
Σχήμα 44: Διάβασμα των RTC καταχωρητών στην Debug λειτουργία	126
Σχήμα 45: Απεικόνιση Alarm ενώ το πλήκτρο INT5 είναι πατημένο	131
Σχήμα 46: Απεικόνιση της ώρας και της ημερομηνίας στο LCD.....	136
Σχήμα 47: RENESAS CK-RA6M5	138
Σχήμα 48: Σχηματικό διάγραμμα της πλακέτας	139
Σχήμα 49: Χάρτης Μνήμης του CK-RA6M5.....	140
Σχήμα 50: Παράθυρο Expressions	156
Σχήμα 51: Παράθυρο Memory.....	156
Σχήμα 52: Παράθυρο Memory Usage.....	158
Σχήμα 53: Αισθητήρας HS3xxx.....	160
Σχήμα 54: Τοποθεσία του HS3xxx στην πλακέτα	161
Σχήμα 55: Τιμές Συνάρτησης HS300x	164
Σχήμα 56: Αισθητήρας ZMOD4410	169
Σχήμα 57: Τοποθεσία του ZMOD4410	169
Σχήμα 58: Τιμές Συνάρτησης IAQ.....	172
Σχήμα 59: Πίνακας IAQ – TVOC	173
Σχήμα 60: Πίνακας CO ₂	173

ΕΙΣΑΓΩΓΗ

- ♦ Τι είναι οι μικροελεγκτές και ποιος ο σκοπός τους
- ♦ Τι είναι τα αναπτυξιακά και ποιος ο σκοπός τους

Οι μικροελεγκτές, microcontrollers ή MCUs, αποτελούνται από έναν πυρήνα-μικροεπεξεργαστή, (ARM, C51 κτλ) ο οποίος συνδέεται με διάφορα περιφερειακά μέσω διαύλων επικοινωνίας. Ένας μικροελεγκτής είναι ουσιαστικά ένας μικρός υπολογιστής αποτελούμενος από επεξεργαστή, μνήμη και περιφερειακά, κατασκευασμένος όμως για συγκεκριμένες εφαρμογές. Επειδή οι μικροελεγκτές είναι κυρίως κατασκευασμένοι για συγκεκριμένες εφαρμογές, το κόστος μπορεί να παραμείνει χαμηλό όπως επίσης η κατανάλωση τους, γιατί μόνο τα πλήρως απαραίτητα στοιχεία για την συγκεκριμένη εφαρμογή εμπεριέχονται.

Ποιός πυρήνας και τι περιφερειακά επιλέγονται για να αποτελέσουν έναν μικροελεγκτή, εξαρτάται από τον σκοπό για τον οποίο θα κατασκευαστεί ο μικροελεγκτής αυτός. Έτσι για παράδειγμα εάν ο σκοπός του κατασκευαστή είναι να δημιουργήσει έναν μικροελεγκτή για ένα κινητό τηλέφωνο, ανάλογα με τα χαρακτηριστικά και τις δυνατότητες που θα θέλει να έχει το τηλέφωνο, θα επιλέξει εκείνο τον πυρήνα και εκείνα τα περιφερειακά που θα δώσουν στο κινητό τηλέφωνο τα χαρακτηριστικά που χρειάζονται με το χαμηλότερο κόστος. Εάν θέλει, για παράδειγμα, ένα απλό τηλέφωνο χωρίς κάμερες, ραδιόφωνο κτλ, θα επιλέξει να κατασκευάσει ένα μικροελεγκτή με μικρό και φθηνό πυρήνα. Τα περιφερειακά του προφανώς θα είναι ένας LCD controller ένας SIM card controller και κάποια ADCs. Αρκετές φορές, οι εταιρίες κατασκευάζουν έναν μικροελεγκτή ο οποίος μπορεί να χρησιμοποιηθεί σε μεγαλύτερη

γκάμα σειράς προϊόντων. Δηλαδή στο παράδειγμα με το κινητό μπορεί η εταιρία να επιλέξει να κατασκευάσει μικροελεγκτή που θα έχει παραπάνω δυνατότητες από αυτές που χρειάζονται σε ένα απλό κινητό. Σκοπός θα είναι να μπορεί να χρησιμοποιηθεί ο ίδιος μικροελεγκτής μετέπειτα σε ένα πιο high-end κινητό. Έτσι η εταιρία θα θυσιάσει το υψηλό κόστος κατασκευής για ένα απλό κινητό αλλά θα γλιτώσει την παραπάνω έρευνα για κατασκευή ενός δεύτερου μικροελεγκτή που θα καλύψει μελλοντικές ανάγκες αναβάθμισης.

Υπάρχουν βέβαια και εταιρίες που κατασκευάζουν μικροελεγκτές γενικού σκοπού. Οι εταιρίες αυτές, όπως η STMicroelectronics η Atmel κ.α., δεν κατασκευάζουν μικροελεγκτές με σκοπό συγκεκριμένα προϊόντα αλλά έχουν σκοπό στην πώληση σε μια ευρύτερη αγορά. Έτσι μια εταιρία που για παράδειγμα κατασκευάζει πλυντήρια, ο μικροελεγκτής που θα χρησιμοποιήσει δεν χρειάζεται να περιέχει κάποια εξειδικευμένα περιφερειακά. Μπορεί έτσι να επιλέξει εκείνον τον μικροελεγκτή, γενικού σκοπού, που πληρή τις προδιαγραφές με το χαμηλότερο κόστος. Σαφώς το κόστος δεν αποτελεί πάντα πρωταρχικό ρόλο, καθώς μια εταιρία μπορεί να επιλέξει ένα μικροελεγκτή γενικού σκοπού από έναν κατασκευαστή που έχει καλύτερη τεχνική υποστήριξη και μικρότερο χρόνο παράδοσης.

Οι αναπτυξιακές πλακέτες χρησιμοποιούνται κυρίως για τους εξής λόγους:

- Αποσφαλμάτωση των MCU. Οι εταιρίες, κατά τη διάρκεια σχεδίασης των MCU ή και γενικώς κατασκευής οποιαδήποτε άλλου IC, κατασκευάζουν παράλληλα και αναπτυξιακές πλακέτες για να διαπιστώσουν την εύρυθμη λειτουργία τους. Η πρώτη παρτίδα των IC που θα βγει από το εργοστάσιο (pilot run tape-out) δοκιμάζεται σε τέτοιου είδους πλακέτες.

- Επίδειξη της σωστής και εύρυθμης λειτουργίας του μικροελεγκτή και των περιφερειακών του (demonstration). Συνήθως οι κατασκευαστές των IC, κατασκευάζουν αναπτυξιακές πλακέτες για να δείξουν στους πελάτες τους ότι το IC τους δουλεύει βάση των αναφερόμενων προδιαγραφών.
- Ανάπτυξη εφαρμογών λογισμικού για συσκευές/προϊόντα που χρησιμοποιούν τον συγκεκριμένο ή παραπλήσιο μικροελεγκτή.
- Επίδειξη της εύρυθμης λειτουργίας εφαρμογών λογισμικού (demonstration) για μελλοντική χρήση τους σε συσκευές/προϊόντα που χρησιμοποιούν τον συγκεκριμένο ή παραπλήσιο μικροελεγκτή.

1 ΕΡΓΑΣΤΗΡΙΟ 1

♦ Μαθαίνοντας το hardware του αναπτυξιακού

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

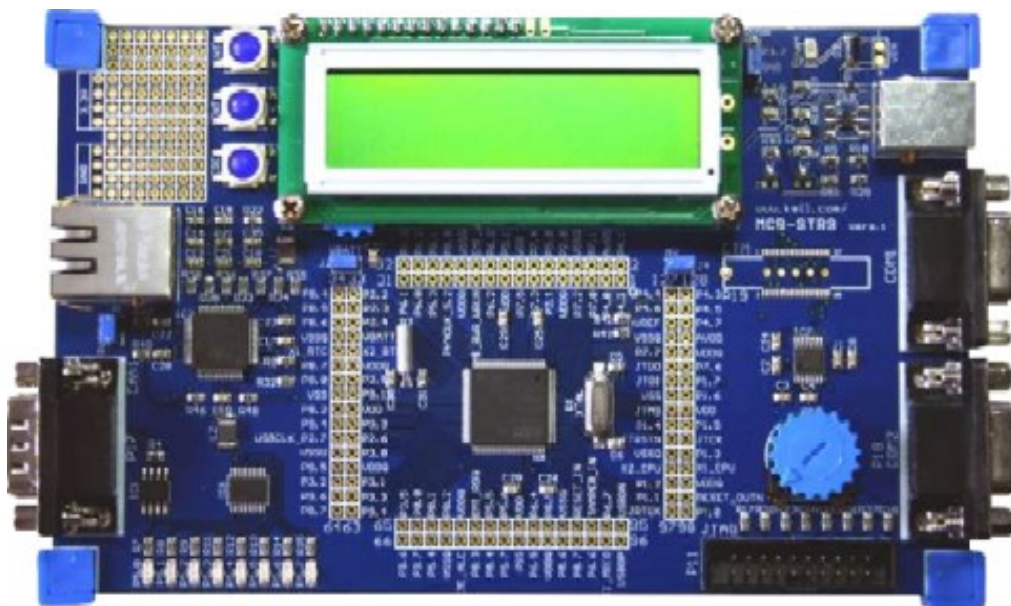
- a. **Αρχείο mcbstr9.chm HTML**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Είναι ένας πλήρης οδηγός του αναπτυξιακού που ο σπουδαστής πρέπει να διαβάσει πριν προχωρήσει στην υλοποίηση των εργαστηρίων.
- b. **Αρχείο MCBSTR9_schematic.pdf**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Είναι το σχηματικό της αναπτυξιακής πλακέτας MCBSTR9.
- c. **Αρχείο STR91xFAxxx.pdf**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Σελίδες 14, 56, 57 και από σελίδα 15 – 39 μια ανάγνωση για τα κύρια χαρακτηριστικά της κάθε μονάδας του μικροελεγκτή.
- d. **Αρχείο STR91xFA_Reference_Manual.pdf**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Το χρησιμοποιούμε ως αναφορά για τους καταχωρητές και περιφερειακά του μικροελεγκτή.
- e. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 2, παράγραφος 2.2. Ο Επεξεργαστής ARM.

Εισαγωγή

Στο εργαστήριο αυτό θα προσπαθήσουμε να αποκτήσουμε μια σφαιρική εικόνα για το αναπτυξιακό πακέτο MCBSTR9 της Keil και κυρίως για το **υλικό** που χρησιμοποιείται (**hardware**).

Πρώτα θα ασχοληθούμε με τα **περιφερειακά** της πλακέτας και μετά με τον **μικροελεγκτή** (MCU) του συστήματος, τον STR912FAW44 της STMicroelectronic's.

Η αναπτυξιακή πλακέτα MCBSTR9 της Keil που φαίνεται στο παρακάτω σχήμα, αποτελεί μια ολοκληρωμένη λύση για ανάπτυξη προγραμμάτων πάνω στους μικροελεγκτές (MCUs) **STR912FAW44** της **STMicroelectronic's**, βασισμένοι σε πυρήνες της ARM. Έχει όλα τα απαραίτητα στοιχεία, όπως USB, UART, CAN και ETHERNET διεπαφές αλλά και περιοχή για πρωτότυπα κυκλώματα, ώστε ο χρήστης να μπορέσει να ξεδιπλώσει όλες τις δυνατότητες του μικροελεγκτή.



Σχήμα 1: MCBSTR9 Keil Development Board

1.1 Το Υλικό του Αναπτυξιακού (Hardware)

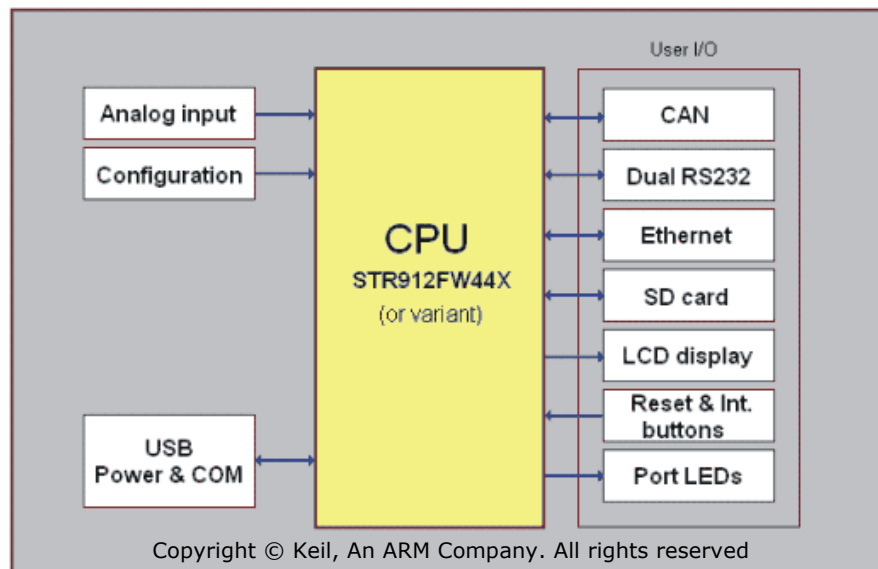
Η πλακέτα του αναπτυξιακού αποτελείται από έναν μικροελεγκτή και τα περιφερειακά του όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 2). Τα περιφερειακά της αναπτυξιακής πλακέτας, όπως για παράδειγμα η USB, θύρα, προσφέρουν ουσιαστικά μια πρόσβαση στους αντίστοιχους περιφερειακούς ελεγκτές που βρίσκονται εσωτερικά στον μικροελεγκτή.

Μικροελεγκτής:

- STR912FAW4XB της STMicroelectronics.

Περιφερειακά:

- LCD, 2x16 αλφαριθμητικό
- USB 2.0 θύρα
- Ethernet θύρα
- 3 Πλήκτρα
- 7 LEDs
- CAN 2.0B θύρα
- 2x RS-232 σειριακές θύρες
- 1x Μετατροπέας αναλογικής τάσης
- JTAG θύρα
- SD μετατροπέας



Σχήμα 2: Σχηματικό διάγραμμα της MCBSTR9 πλακέτας

1.1.1 Διαβάζοντας το σχηματικό

Το σχηματικό, που δίνεται στο MCBSTR9_schematic.pdf, αποτελεί μια από τις βασικότερες πληροφορίες και χρησιμοποιείται για να βρούμε πως τα διάφορα στοιχεία πάνω στην πλακέτα συνδέονται μεταξύ τους. Το σχηματικό πρέπει να αποτελεί πάντα την βάση της οποιαδήποτε υλοποίηση μας.

Στο σχηματικό, κάθε στοιχείο έχει ένα όνομα. Για παράδειγμα, στην πρώτη σελίδα, του MCBSTR9_schematic.pdf, κάτω αριστερά, μπορούμε να δούμε κάποιους πυκνωτές. Τα ονόματά τους είναι C30 και C31. Αντίστοιχα, ονόματα έχουν τα ICs οι θύρες και γενικά οτιδήποτε υπάρχει πάνω σε μια πλακέτα, όπως μία τρύπα για βίδα ή μια θύρα για USB κτλ.

Παρατηρώντας το σχηματικό απαντήστε τα παρακάτω ερωτήματα.

Ερώτημα 1

Βρείτε ποια ονόματα έχουν δοθεί στο σχηματικό για τα παρακάτω στοιχεία.

USB θύρα	
7x LED	
LCD Οθόνη	
3x Πλήκτρα	
JTAG θύρα	
Ποτενσιόμετρο	

Παρακάτω θα δώσουμε μια σύντομη περιγραφή του μικροελεγκτή και των περιφερειακών του.

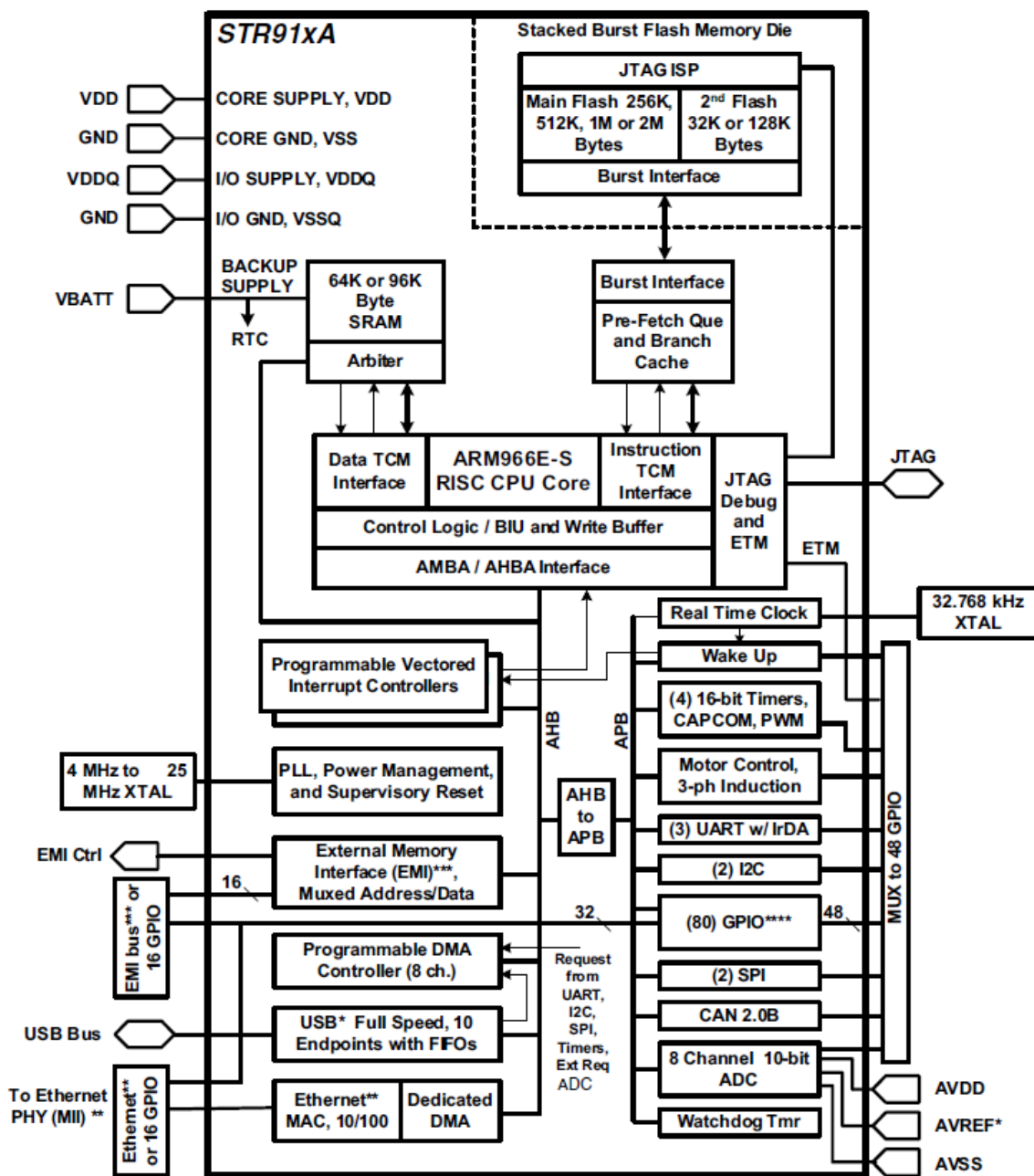
1.1.2 STR912FAW44XB

Στα συνημμένα έγγραφα του εργαστηρίου θα βρείτε δύο κείμενα STR91xFAxxx.pdf και STR91xFA_Reference_Manual.pdf. Τα κείμενα αυτά περιέχουν την πλήρη περιγραφή του μικροελεγκτή. Εδώ θα περιγράψουμε τα βασικότερα στοιχεία του.

Πυρήνας (CPU)	16/32-bit 96 MHz ARM966E-S RISC
2 εσωτερικές μνήμες Flash	• 512Kb πρωτεύουσα • 32Kb δευτερεύουσα
Εσωτερική Static RAM	64Kb
Εσωτερικός ταλαντωτής ρολογιού	Λειτουργεί με εξωτερικό κρύσταλλο 4-25MHz. Ο ταλαντωτής τροφοδοτεί το εσωτερικό PLL, το οποίο μπορεί να δώσει ταχύτητες έως 96MHz
Εξωτερική διεπαφή μνήμης	External Memory Interface (EMI)
80 I/O Γενικού σκοπού	General Purpose I/O (GPIO)
RTC (Real Time Clock)	Ρολόι πραγματικού χρόνου
10-bit ADC 8-καναλιών	Analogue to Digital Converter.
10 Communication Interfaces	Διασυνδέσεις επικοινωνίας • 10/100 Ethernet MAC • USB 2.0 slave • CAN 2.0B Active • 3x UART • 2x Fast I²C 400KHz
4x 16-bit Timers	Χρονιστές
3-Phase Induction Motor Controller	Ελεγκτής επαγωγικού κινητήρα 3-φάσεων
JTAG	Διεπαφή ελέγχου και προγραμματισμού

Πίνακας 1: Βασικά χαρακτηριστικά του STR912FAW44XB MCU

Το επόμενο σχήμα δείχνει την βασική δομή του STR912F44XB. Εδώ παρουσιάζονται τα βασικά στοιχεία του MCU σε blocks και παράλληλα δίδεται σχηματικά πως αυτά επικοινωνούν μεταξύ τους.



Σχήμα 3: Διάγραμμα MCU της οικογένειας STR9xFxxxx

Παρατηρώντας το σχήμα 3, προσπαθήστε να απαντήσετε στα επόμενα ερωτήματα.

Ερώτημα 2**Τι αρχιτεκτονικής είναι ο ARM966E-S; Δικαιολογήστε το.**

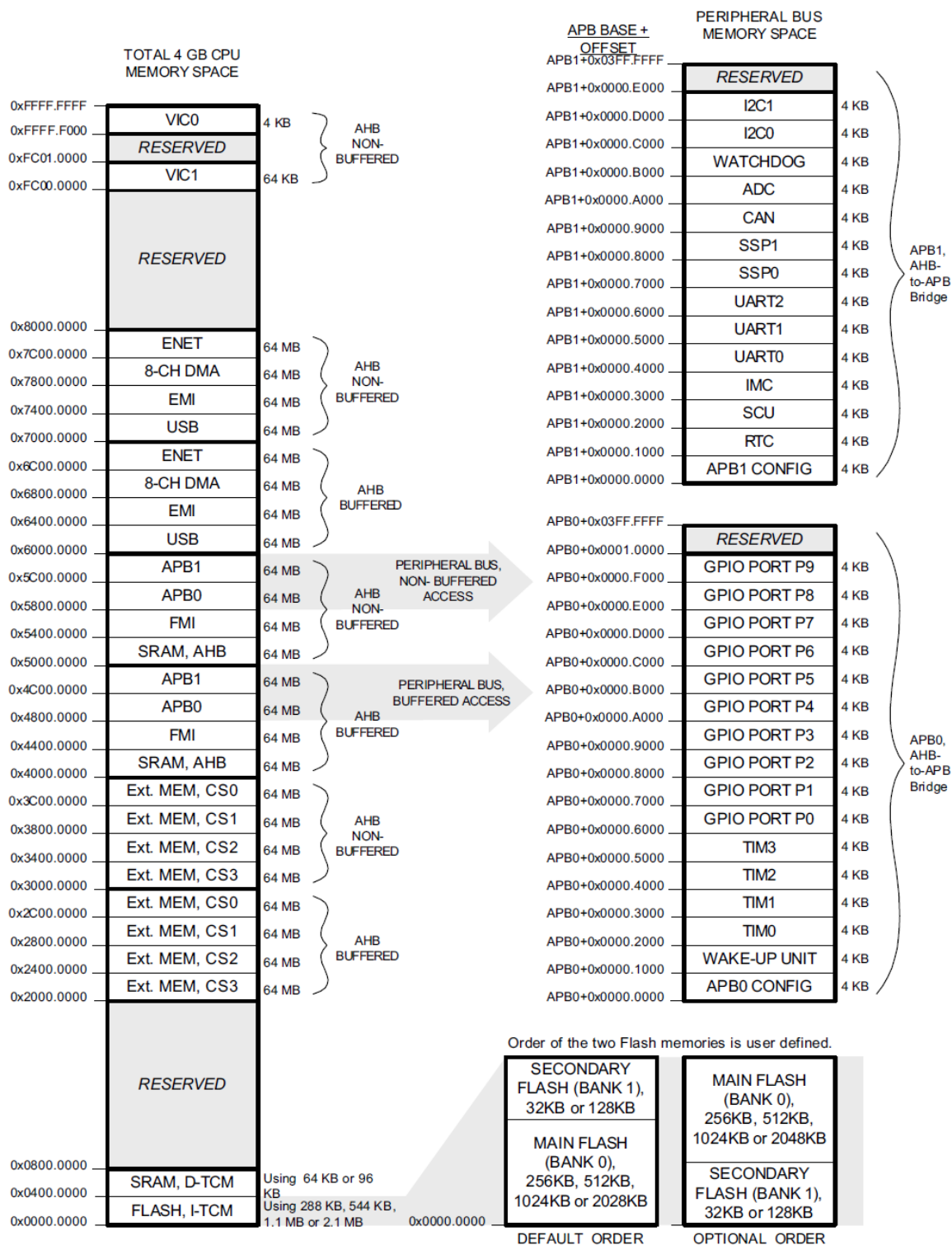
Ερώτημα 3**Πως συνδέετε ο ελεγκτής CAN με τον πυρήνα ARM966E-S;**

Ερώτημα 4**Πόσοι και ποιοι τύποι μνήμης υπάρχουν εσωτερικά στον μικροελεγκτή;**

Ερώτημα 5**Πόσα I/O γενικού σκοπού υπάρχουν (GPIO);**

1.1.3 Χάρτης Μνήμης του STR912FAW44XB (Memory Map)

Κάθε στοιχείο του μικροελεγκτή είναι ορισμένο σε 32-bit καταχωρητές, ξεκινώντας από την θέση 0 έως την θέση 0xFFFFFFFF. Οτιδήποτε έχει να κάνει με τα περιφερειακά, τα interrupts και τις εσωτερικές ή εξωτερικές μνήμες ρυθμίζεται ή απεικονίζεται μέσω των καταχωρητών αυτών.



Σχήμα 4: Χάρτης Μνήμης του STR912FAW44XB

Έτσι για παράδειγμα, η θύρα I/O 3 γενικού σκοπού (General Purpose Input/Output ή GPIO Port 3) έχει καταχωρητές από την θέση μνήμης $APB0 + 0x00009000$ έως $APB0 + 0x000093FC$. Δηλαδή στις θέσεις $0x48009000-0x480093FC$. Στην θέση μνήμης $0x4800A000$ ξεκινάνε οι καταχωρητές για το GPIO Port 4.

Το αρχείο STR91xFA_Reference_Manual.pdf περιέχει την πλήρη περιγραφή όλων των καταχωρητών του STR912FAW44XB. Ανοίγοντας το στην σελίδα 116 μπορούμε να διαβάσουμε τη λειτουργία των καταχωρητών για τα GPIO Ports.

Ανατρέχοντας στο STR91xFA_Reference_Manual.pdf, απαντήστε στις παρακάτω ερωτήσεις.

Ερώτημα 6

Ποια η διεύθυνση των καταχωρητών για το RTC (Real Time Clock);
Πόσοι 32-bit καταχωρητές υπάρχουν για το RTC;

Ερώτημα 7

Ποια η αρχική διεύθυνση των καταχωρητών για την UART1;
Πόσοι 32-bit καταχωρητές υπάρχουν για την UART1;

Ερώτημα 8

Ανατρέχοντας στην σελίδα 83, περιγράψτε των σκοπό του SCU (System Control Unit).

1.1.4 LCD, 2x16 αλφαριθμητικό

Ένα από τα περιφερειακά της πλακέτας ανάπτυξης MCBSTR9 είναι το LCD. Τα LCD αυτού του τύπου αποτυπώνουν αλφαριθμητικούς χαρακτήρες οι οποίοι βρίσκονται σε εσωτερική μνήμη.

Ερώτημα 9

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες συνδέετε το LCD με τον μικροελεγκτή.

1.1.5 Ethernet θύρα

Το τμήμα Ethernet του μικροελεγκτή υλοποιεί το υποεπίπεδο Ελέγχου Προσπέλασης Μέσου (Medium Access Control ή MAC) που ανήκει στο επίπεδο Σύνδεσης Δεδομένων. Το Φυσικό επίπεδο δεν υλοποιείται από τον μικροελεγκτή και έτσι απαιτείται εξωτερική λύση. Για να συνδεθεί το Φυσικό επίπεδο με το επίπεδο Σύνδεσης Δεδομένων, το STE100P chip της STMicroelectronics χρησιμοποιείται. Το STE100P είναι εάν IC υψηλών επιδόσεων που υλοποιεί την διεπαφή μεταξύ του Φυσικού και του επιπέδου Σύνδεσης Δεδομένων για 10-100Base-T Ethernet εφαρμογές.

Ερώτημα 10

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες συνδέετε η θύρα Ethernet με το STE100P.

Ερώτημα 11

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες του μικροελεγκτή συνδέετε ο STE100P.

1.1.6 CAN 2.0B θύρα

Ένας κλασικός DB9 connector χρησιμοποιείται για την σύνδεση συσκευών CAN στο αναπτυξιακό. Όπως στην περίπτωση του Ethernet έτσι και στο CAN ένα εξωτερικό chip χρειάζεται για να μετατρέψει το διαφορικό σήμα του CAN (δύο συμπληρωματικά σε τάση, ξεχωριστά σήματα) (differential) σε ένα μόνο σήμα (single-ended). Το chip αυτό είναι το SN65HVD230 της Texas Instruments.

Ερώτημα 12

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες του μικροελεγκτή συνδέετε το SN65HVD230.

1.1.7 2x RS-232 σειριακές θύρες

Αντίστοιχα με το CAN και το Ethernet, για να συνδεθούν συσκευές με το σειριακό πρωτόκολλο RS-232, χρειάζεται μετατροπέας. Η κύρια λειτουργία

του μετατροπέα είναι να μετατρέψει τα επίπεδα τάσης του RS-232 (μέγιστο $\pm 25V$) σε TTL/CMOS επίπεδα (3.3-5V). Το ST3232 είναι το chip που χρησιμοποιείται στην αναπτυξιακή πλακέτα για να μετατρέψει τα επίπεδα της τάσης και ουσιαστικά να συνδέσει το Φυσικό επίπεδο με το επίπεδο Σύνδεσης Δεδομένων.

Ερώτημα 13

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες του μικροελεγκτή συνδέετε το ST3232.

1.1.8 USB 2.0 θύρα

Η MCBSTR9 πλακέτα έχει μια USB 2.0 θύρα. Η θύρα αυτή συνδέεται με τον μικροελεγκτή μέσω ενός IC, του USBLC6-2. Το εσωτερικό USB τμήμα του μικροελεγκτή προσφέρει έναν USB ελεγκτή ο οποίος εφαρμόζει το Φυσικό και το επίπεδο Σύνδεσης Δεδομένων των OSI επιπέδων.

Ερώτημα 14

Βρείτε από το διαδίκτυο ποια η λειτουργία του IC USBLC6-2 και δικαιολογήστε εάν το IC αυτό αποτελεί κομμάτι του Φυσικού επιπέδου για το USB 2.0 πρωτόκολλο.

Ερώτημα 15

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες του μικροελεγκτή συνδέετε η USB θύρα. Από το STR91xFAXxx.pdf βρείτε τη λειτουργία των ακροδεκτών αυτών.

1.1.9 Μετατροπείας αναλογικής τάσης (Ποτενσιόμετρο)

Ένας μετατροπείας αναλογικής τάσης, που βρίσκεται πάνω στην αναπτυξιακή πλακέτα μπορεί να χρησιμοποιηθεί για να δοκιμάσουμε τους ADCs του μικροελεγκτή. Ο μετατροπείας αυτός μπορεί να μεταβάλει την τάση εισόδου από 0 έως 3,3V.

Ερώτημα 16

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιο ακροδέκτη του μικροελεγκτή συνδέετε το Ποτενσιόμετρο. Από το STR91xFAXxx.pdf βρείτε τη λειτουργία του ακροδέκτη αυτού.

1.1.10 Πλήκτρα και LEDs

Τρία πλήκτρα και επτά LED υπάρχουν πάνω στην αναπτυξιακή πλακέτα. Ένα πλήκτρο είναι συνδεδεμένο στο Reset pin του MCU και τα υπόλοιπα δύο είναι συνδεδεμένα σε δύο ακροδέκτες γενικού σκοπού εισόδου/εξόδου (GPIO) του μικροελεγκτή τα οποία μπορούν να προγραμματιστούν και ως πηγές εξωτερικών διακοπών (External Interrupts).

Ερώτημα 17

Παρατηρώντας το σχηματικό MCBSTR9_schematic.pdf, βρείτε σε ποιούς ακροδέκτες του μικροελεγκτή συνδέονται τα πλήκτρα. Από το STR91xFAXxx.pdf βρείτε τη λειτουργία των ακροδεκτών αυτών.

--

1.1.11 JTAG θύρα

Η θύρα JTAG προσφέρει μια κύρια μονάδα αποσφαλμάτωσης και προσομοίωση του συστήματος σε πραγματικό χρόνο. Επίσης χρησιμοποιείται για προγραμματισμό της εσωτερικής flash. Εμείς θα χρησιμοποιήσουμε το JTAG για να κατεβάσουμε κώδικα στον μικροελεγκτή.

Ερώτημα 18

Βάση των βασικών χαρακτηριστικών του μικροελεγκτή αναφέρεται οκτώ προϊόντα στα οποία θα μπορούσε να χρησιμοποιηθεί και γιατί;

1.

2.

3.

4.

5.

6.

7.

8.

2 ΕΡΓΑΣΤΗΡΙΟ 2

- ♦ Μαθαίνοντας το λογισμικό του αναπτυξιακού
- ♦ Απλή χρήση του LCD

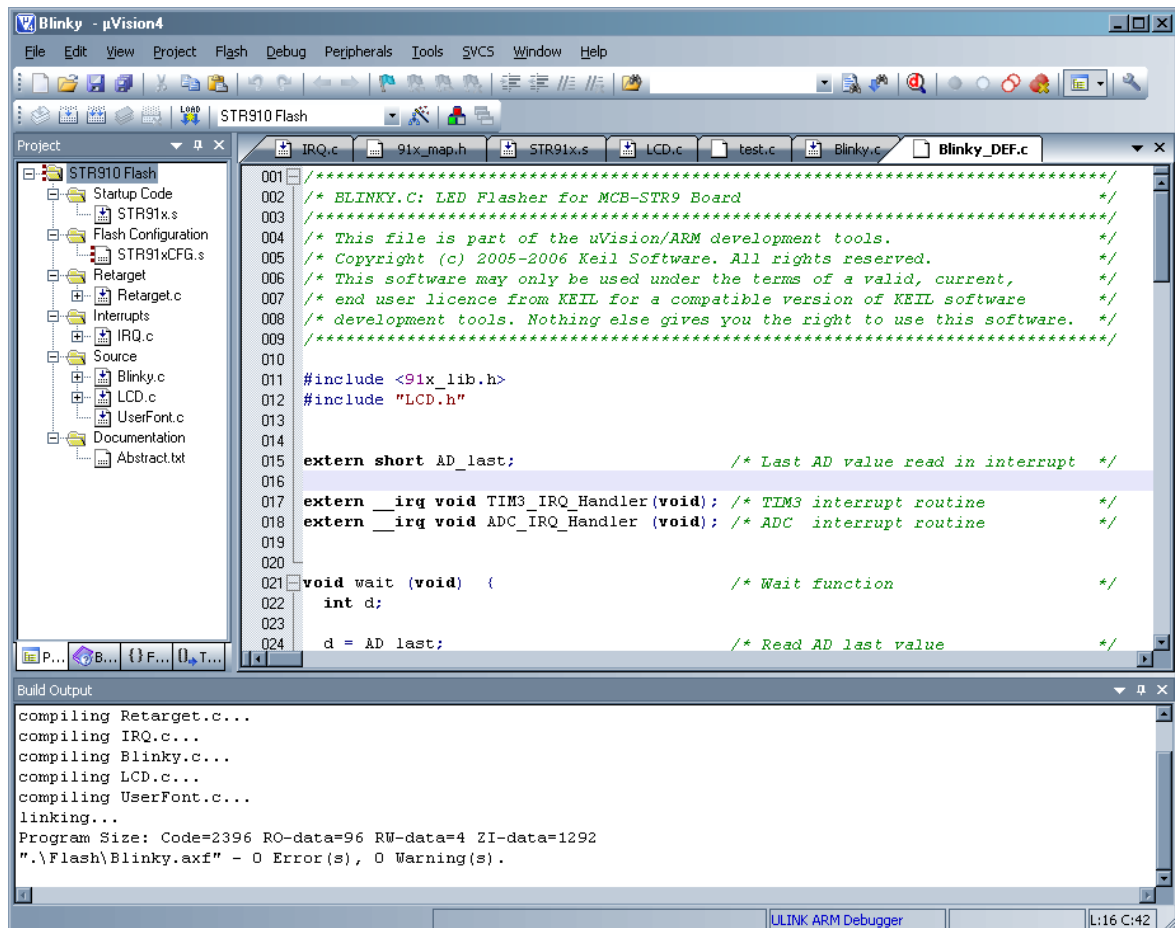
Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

- f. **Αρχείο mcbstr9.chm HTML**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Είναι ένας πλήρης οδηγός του αναπτυξιακού που ο σπουδαστής πρέπει να διαβάσει πριν προχωρήσει στην υλοποίηση των εργαστηρίων. Κεφάλαιο: “Writing Programs”.
- g. **Αρχεία Blinky.c και LCD.c**, μέσα από τον φάκελο **Lab2_Blinky.rar**, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου.
- h. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 4, παράγραφος 4. Συσκευές Εισόδου-Εξόδου, Οθόνες.

Εισαγωγή

Το λογισμικό που περιλαμβάνεται στο πακέτο του αναπτυξιακού και που θα χρησιμοποιήσουμε στο εργαστήριο, είναι το μ Vision4 της Keil. Το λογισμικό αυτό παρέχει ένα πλήρη περιβάλλον για δημιουργία και επεξεργασία κώδικα, απασφαλμάτωση προγραμμάτων με προσομοίωση πραγματικού χρόνου ή εξομοίωση και προγραμματισμό flash.

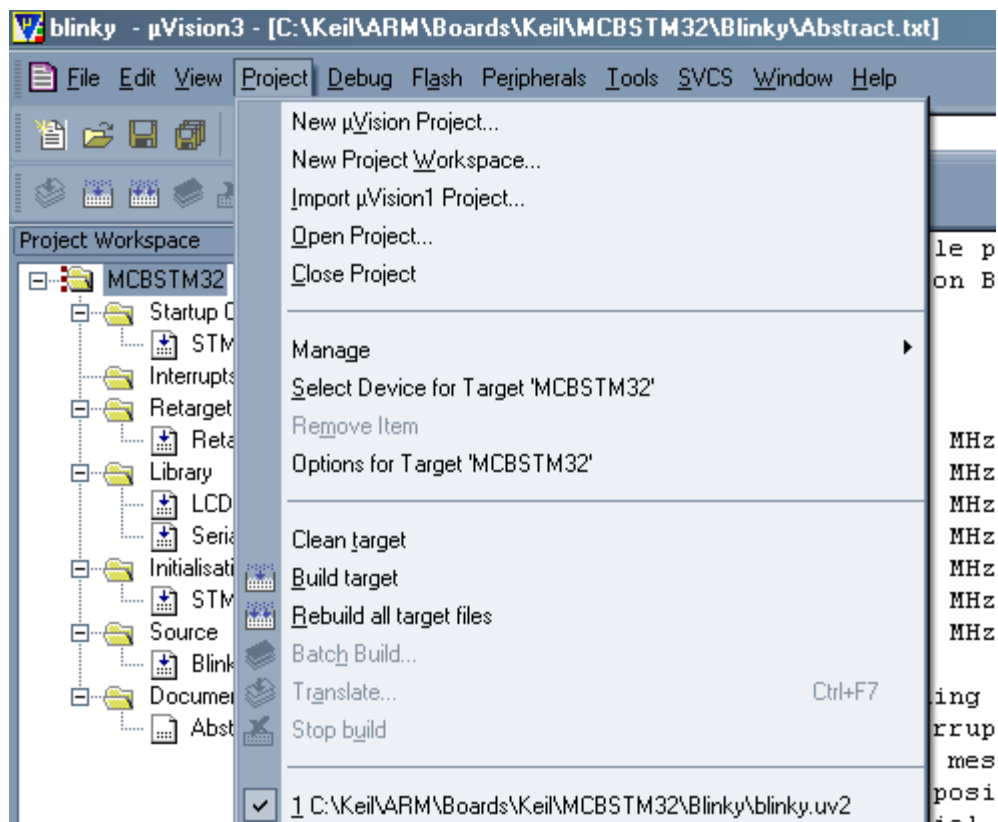


Σχήμα 5: Το περιβάλλον του μ Vision4

2.1 Το παράδειγμα Blinky

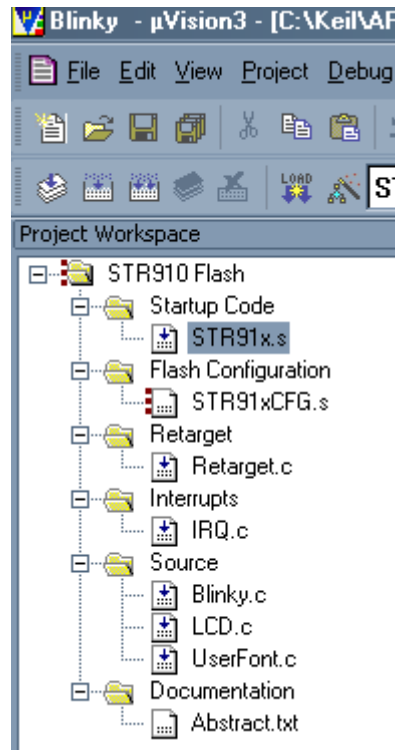
Παρακάτω θα αναλύσουμε τα βήματα που χρειάζονται για να εκτελέσουμε το παράδειγμα Blinky που περιέχεται στο πακέτο λογισμικού της Keil.

- a. Πριν τρέξουμε το μ Vision4 αντιγράφουμε τον φάκελο: **Lab2_Blinky.rar** στην επιφάνεια εργασίας και τον αποσυμπιέζουμε για να μπορέσουμε να τον επεξεργαστούμε.
- b. Ξεκινάμε το μ Vision4 από το **Start->All Programs->Keil uVision4** είτε από την συντόμευση στην επιφάνεια εργασίας.
- c. Από την επιλογή **Project** επιλέγουμε **Open Project** όπως φαίνεται στο παρακάτω σχήμα. Αναζητούμε το σημείο που προηγουμένως αντιγράψαμε τον φάκελο Lab2_Blinky (επιφάνεια εργασίας) και ανοίγουμε το **Blinky.uvproj**.



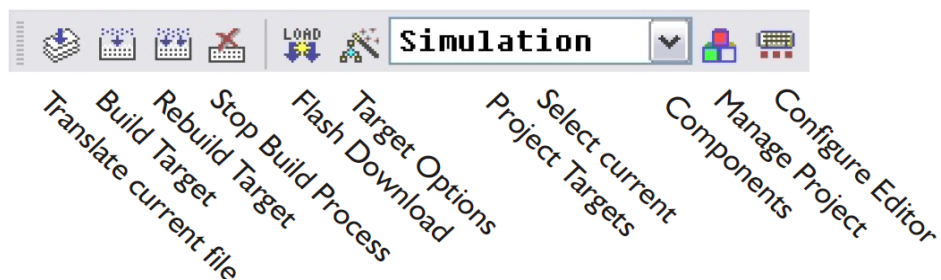
Σχήμα 6: Επιλογή Project στο uVision3

- d. Στο **Project Workspace** παρατηρούμε τα διάφορα αρχεία του project (Σχήμα 7). Πατώντας διπλό κλικ στα αρχεία αυτά μπορούμε να τα ανοίξουμε, να δούμε το περιεχόμενό τους και να τα επεξεργαστούμε.



Σχήμα 7: Παράθυρο Project Workspace

- e. Χτίζουμε τον κώδικα πηγαίνοντας στο **Build Toolbar**. Βεβαιωθείτε ότι στην επιλογή **Select Target** είναι επιλεγμένο **STR 910 Flash**. Πατάμε το εικονίδιο του **Build Target**.



Σχήμα 8: Build Toolbar

Εάν το χτίσιμο του κώδικα έχει ολοκληρωθεί χωρίς λάθη, στο παράθυρο Build Output θα πρέπει να έχουμε τα αποτελέσματα του παρακάτω σχήματος (Σχήμα 9).

Build Output E

```
Build target 'STR910 Flash'
assembling STR91x.s...
compiling Retarget.c...
compiling IRQ.c...
compiling Blinky.c...
compiling LCD.c...
compiling UserFont.c...
linking...
Program Size: Code=2260 RO-data=96 RW-data=4 ZI-data=1292
".\Flash\Blinky.axf" - 0 Error(s), 0 Warning(s).
```

Σχήμα 9: Παράθυρο Build Output μετά από επιτυχές build

2.2 Εκτελώντας το παράδειγμα Blinky στο αναπτυξιακό

Έχοντας επιτυχώς τελειώσει το παραπάνω παράδειγμα, μπορούμε να κατεβάσουμε το **binary Blinky.axf** αρχείο που δημιουργήθηκε, στην πλακέτα. Για να το κάνουμε αυτό πρώτα βεβαιωνόμαστε για τα εξής πράγματα:

- Δεν υπάρχουν πάνω ή κοντά στην πλακέτα άλλα μεταλλικά αντικείμενα ή υγρά.
- Δεν ακουμπάμε με τα χέρια μας την πλακέτα, παρά μόνο τα πλήκτρα και τις θύρες (από πλαστικό) που βρίσκονται σε αυτή.

Αφού βεβαιωθούμε για τα παραπάνω τροφοδοτούμε και συνδέουμε το JTAG στο αναπτυξιακό, κάνοντας τα εξής:

- Συνδέουμε το USB καλώδιο στην USB θύρα.
- Το LED Power, χρώματος κόκκινο, ανάβει.
- Συνδέουμε την συσκευή ULINK2 της Keil, στην JTAG θύρα.

Τώρα είμαστε έτοιμοι να κατεβάσουμε το binary αρχείο στην αναπτυξιακή πλακέτα. Η διαδικασία που ακολουθείται από τα εργαλεία της Keil για να κατεβάσει το Binary στην flash είναι η εξής:

- Erase: Σβήνει τα προηγούμενα περιεχόμενα της flash
- Program: Προγραμματίζει την flash με το νέο binary
- Verify: Πιστοποιεί ότι ο προγραμματισμός της Flash έγινε σωστά.

Τα στάδια αυτά μπορούμε να τα δούμε στο Build Output παράθυρο. Όταν το κατέβασμα τελειώσει επιτυχώς, το Build Output παράθυρο θα μας ενημερώσει αναλόγως, όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 10).

Πατάμε το πλήκτρο **Flash Download** στο **Build Toolbar** για να κατεβάσουμε το binary στη flash.

Σχήμα 10: Παράθυρο Build Output μετά από επιτυχές binary κατέβασμα

```
Build Output
Build target 'STR910 Flash'
linking...
Program Size: Code=2260 RO-data=96 RW-data=4 ZI-data=1292
".\Flash\Blinky.axf" - 0 Error(s), 0 Warning(s).
Load "C:\Keil\ARM\Boards\Keil\MCBSTR9\Blinky\Flash\Blinky.AXF"
Erase Done.
Programming Done.
Verify OK.
Application running ...
```

Στο παράδειγμα που εκτελούμε, ο ρυθμός ανακύλισης των LEDs και η ένδειξη του LCD αυξομειώνεται ανάλογα με την τάση εισόδου του ADC0, δηλαδή ανάλογα με την ρύθμιση του ποτενσιόμετρου POT1.

Ερώτημα 19

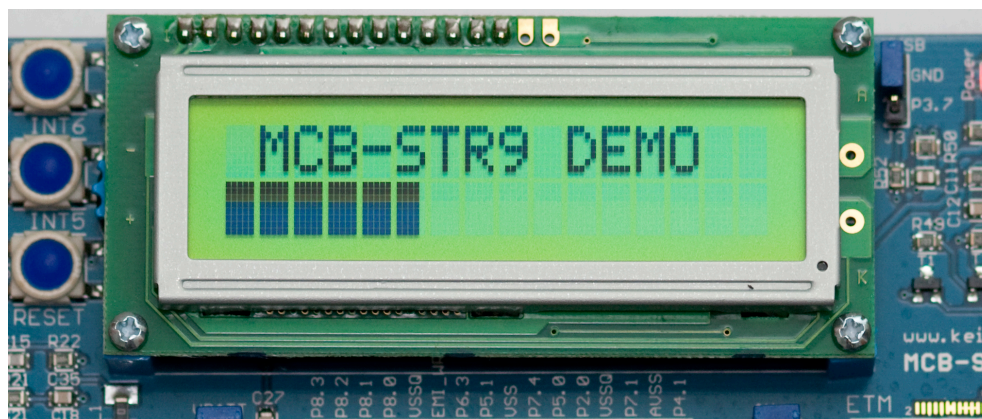
Χρησιμοποιώντας ένα πολύμετρο βρείτε την σχέση της ανακύλισης των LEDs και της τάσης εξόδου του ποτενσιόμετρου. Όταν αυξάνεται η τάση αυξάνεται ο ρυθμός ανακύλισης ή το αντίθετο συμβαίνει; Ανατρέξτε στο σχηματικό, MCBSTR9_schematic.pdf, για να επιλέξετε σημείο μέτρησης.

2.3 Το παράδειγμα Blinky. Απλή χρήση του LCD.

Θα κάνουμε αλλαγές στον κώδικα του παραδείγματος Blinky ώστε στην πρώτη γραμμή της οθόνης του LCD να τυπωθεί το μικρό μας όνομα και στην δεύτερη το επώνυμο μας, όπως φαίνεται παρακάτω.



Σχήμα 11: Το όνομα μας στο LCD



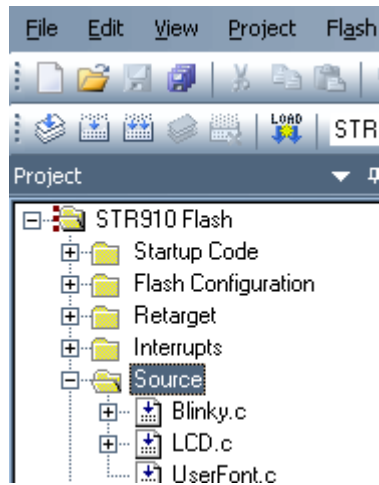
Σχήμα 12: Αρχική ένδειξη του LCD

Για να γράψουμε στο LCD πρέπει να χρησιμοποιήσουμε την συνάρτηση `lcd_print`.

Ερώτημα 20

Ανοίγοντας το αρχείο `LCD.c` βρείτε και γράψτε τι παραμέτρους δέχεται η `lcd_print` και τι επιστρέφει.

Ανοίγουμε το Blinky.c το οποίο βρίσκουμε στο Project, στο αριστερό παράθυρο του uVision4 (Σχήμα 13).



Σχήμα 13: Παράθυρο Project

Στην main συνάρτηση του Blinky.c αρχείου, μπορούμε να βρούμε που αλλού αλλά και πώς χρησιμοποιείται η lcd_print και να το χρησιμοποιήσουμε ως παράδειγμα.

Για να γράψουμε το όνομα μας στην LCD οθόνη ακολουθήσουμε τα παρακάτω βήματα:

- Σβήνουμε τον ατέρμονο βρόχο (while(1)) και ότι αυτός περιέχει. Δεν θέλουμε να υπάρχει πλέον η ανακύλιση στα LED αλλά ούτε να γεμίζει με μαύρα κουτιά η δεύτερη γραμμή του LCD. Σβήνοντας τον ατέρμονο βρόχο δεν σταματάμε τη λειτουργία του ADC. Απλώς η ένδειξη για την τάση του ποτενσιόμετρου μέσω του Analoge to Digital Converter (ADC) δεν θα απεικονίζεται ούτε στο LCD αλλά ούτε μέσω των LED. Ο ADC θα λειτουργεί κανονικά αλλά εμείς δεν θα διαβάζουμε την τιμή του.
- Προσθέτουμε την lcd_print συνάρτηση με το όνομα μας. Συμπληρωματικές συναρτήσεις, που υπάρχουν στο αρχείο LCD.c, μπορεί να χρειαστούν.

- Χτίζουμε και κατεβάζουμε τον κώδικα όπως κάναμε στην 2.1 και 2.2 παράγραφο.
- Εάν δεν λειτουργεί όπως θέλουμε, τρέχουμε πάλι τα βήματα b και c.

Ερώτημα 21

Εδώ συμπληρώστε τον τελικό κώδικα σας.

3 ΕΡΓΑΣΤΗΡΙΟ 3

♦ Μέθοδοι απασφαλμάτωσης

Προϋποθέσεις

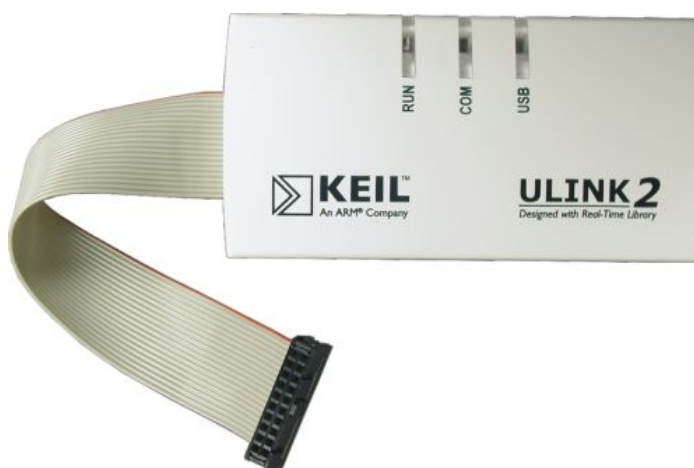
Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

- i. **Αρχείο mcbstr9.chm HTML**, Κεφάλαιο “Writing Programs->Debugging Programs”
- j. **Αρχείο debug_test.c**, μέσα από τον φάκελο **Lab3_Debug.rar**.
- k. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για καταχωρητές.
- l. **Βιβλίο Θεωρίας Wayne Wolf**, “Οι Υπολογιστές ως Συστατικά Στοιχεία”. Κεφάλαιο 4, παράγραφος 7. Ανάπτυξη και Αποσφαλμάτωση.

Εισαγωγή

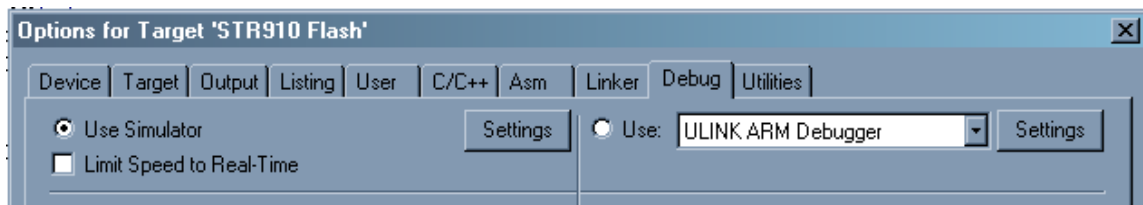
Στο εργαστήριο αυτό θα ακολουθήσουμε διάφορες μεθόδους απασφαλμάτωσης, χρησιμοποιώντας τον απασφαλματωτή (debugger) του μVision4 σε συνδυασμό με την συσκευή ULINK2. Η συσκευή αυτή είναι ουσιαστικά ένας USB-JTAG μετατροπέας και επιτρέπει στο λογισμικό μVision4 να επικοινωνεί με την JTAG διεπαφή του μικροελεγκτή. Χρησιμοποιεί **JTAG (Joint Test Action Group) εντολές** διεπαφής για να πραγματοποιήσει τα παρακάτω:

- Φόρτωση προγραμμάτων λογισμικού στη πλακέτα
- Απασφαλμάτωσης του τρέχοντος προγράμματος με διάφορες μεθόδους.
- Προγραμματισμό της εσωτερικής μνήμης FLASH του μικροελεγκτή.
- Προγραμματισμό εξωτερικής μνήμης FLASH στην πλακέτα.



Σχήμα 14: ULINK2 USB-JTAG μετατροπέας

Παράλληλα το λογισμικό μVision4 μπορεί να χρησιμοποιηθεί κάποια πλακέτα και μικροελεγκτή, επιλέγοντας το “Use Simulator” στο παράθυρο “Options for Target “STR910 Flash”, όπως φαίνεται στο επόμενο σχήμα. Με την επιλογή αυτή, ο χρήστης μπορεί να δοκιμάσει τον κώδικα του καθώς το μVision4 έχει την δυνατότητα να εξομοιώσει το μεγαλύτερο κομμάτι του μικροελεγκτή.



Σχήμα 15: Επιλογή εξομοίωσης στο uVision4

Παρακάτω, θα χρησιμοποιήσουμε τρεις διαφορετικές μεθόδους απασφαλμάτωσης.

- Άμεση παρακολούθηση των καταχωρητών του μικροελεγκτή.
- Breakpoints.
- Παράθυρα Dissassembly και Register.

3.1 Ρυθμίσεις uVision για χρήση του αποσφαλματωτή

Ανοίγουμε το **Project debug_test.uvproj** αποσυμπιέζοντας το **Lab3_Debug.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε το αρχείο **debug_test.c**

```
int main (void)
{
    int i, dummy_loop;

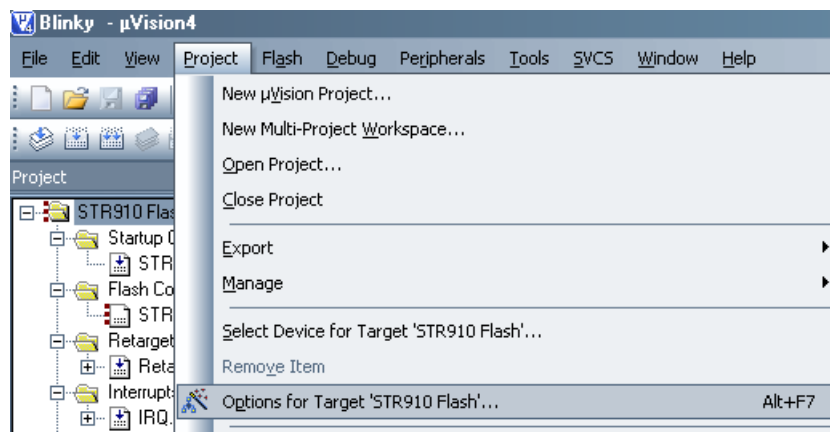
    for (i=0; i<100; i++)
        dummy_loop = i;

    /* System Control Unit Set-up */
    SCU->PRR1      = 0x00EE2803;
    SCU->PCGR1      = 0x00EE2803;

    /* Buttons Setup */
    SCU->GPIOIN[3]   = 0x00;      /* P3.5 input - mode 1 */
    SCU->GPIOOUT[3]  &= 0xF3FF; /* P3.5 input - mode 0 */
    GPIO3->DDR       &= 0xDF;    /* P3.5 direction - input */
}
```

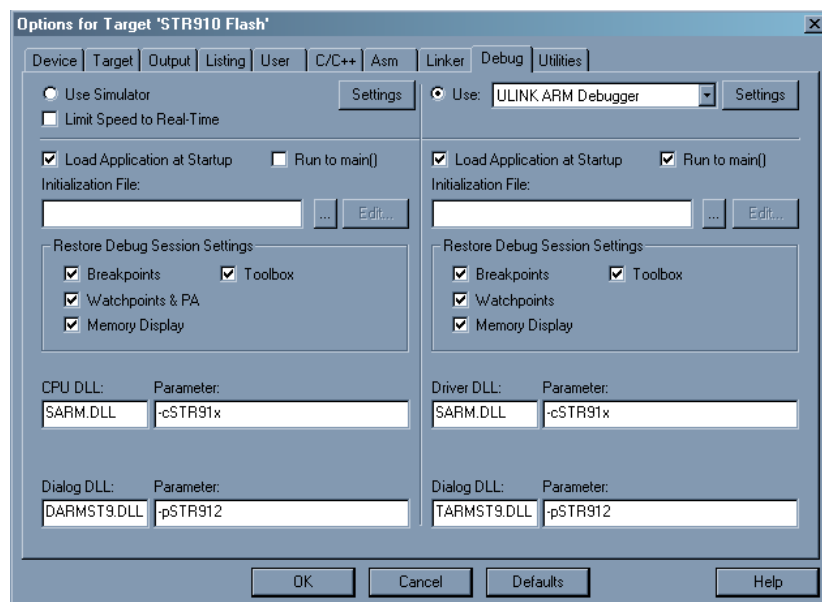
Χτίζουμε και κατεβάζουμε τον κώδικα στο αναπτυξιακό ακολουθώντας τα βήματα των Κεφαλαίων 2.1 και 2.2.

Επιλέγουμε το Options for Target 'STR910 Flash' όπως φαίνεται στο επόμενο σχήμα.



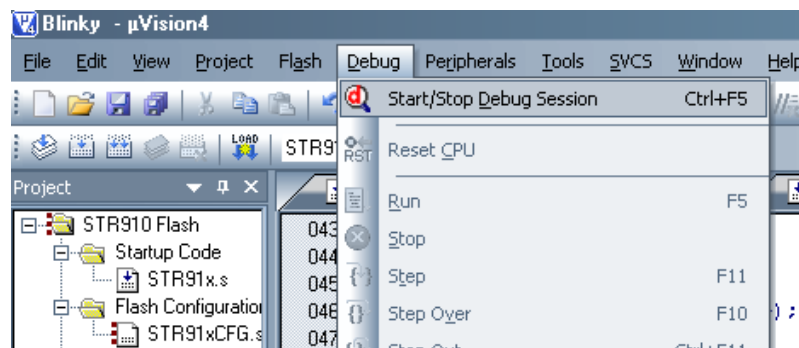
Σχήμα 16: Επιλογές debugger

Από το παράθυρο που αναδύεται επιλέγουμε το tab **Debug** και το ρυθμίζουμε όπως φαίνεται στο επόμενο σχήμα.



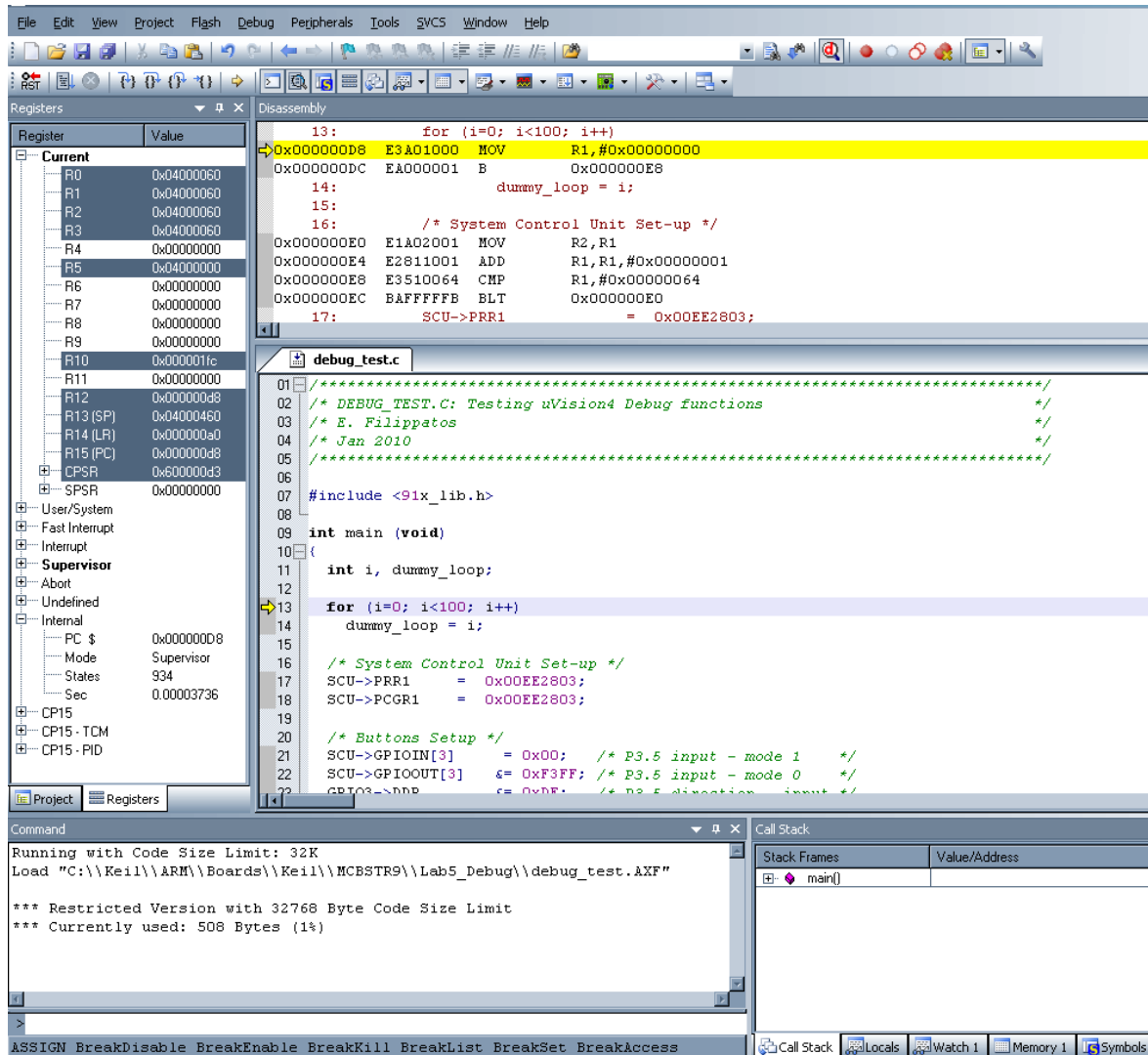
Σχήμα 17: Επιλογές debugger

Κλείνουμε το παράθυρο και επιλέγουμε το **Start/Stop Debug Session** όπως φαίνεται στην παρακάτω εικόνα.



Σχήμα 18: Επιλογή Εκκίνησης Debugger

Πατάμε OK στο μήνυμα: *EVALUATION MODE, Running with Code Size Limit : 32K*. Ο Debugger ξεκινάει και αφού κάνει ένα reset στον μικροελεγκτή, περιμένει στην πρώτη εντολή της main του προγράμματος μας. Αυτό φαίνεται παρακάτω.



Σχήμα 19: Παράθυρο Debugger μVision4

3.2 Παρακολούθηση κατάστασης καταχωρητών

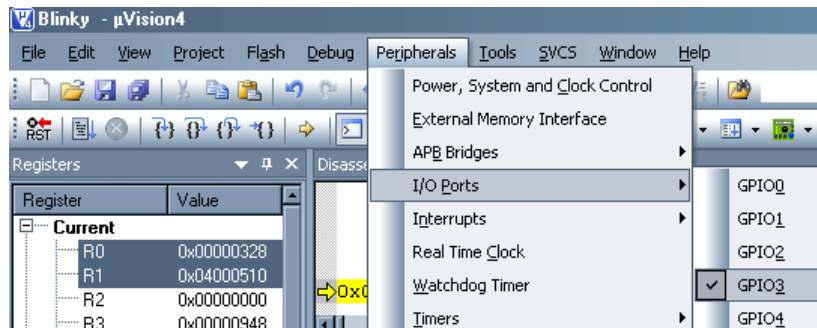
Μέσω της χρήσης του debugger θα προσπαθήσουμε να ελέγξουμε την κατάσταση του GPIO 3.5 στο οποίο είναι συνδεδεμένο το πλήκτρο INT5. Για να μπορέσουμε να παρακολουθούμε την κατάσταση του GPIO 3.5, πρέπει να παρακολουθούμε τον αντίστοιχο καταχωρητή που κρατάει τις καταστάσεις ή αλλιώς τα δεδομένα του GPIO Port 3.

Ερώτημα 22

Σε ποιόν καταχωρητή αποθηκεύεται η κατάσταση του GPIO 3.5;

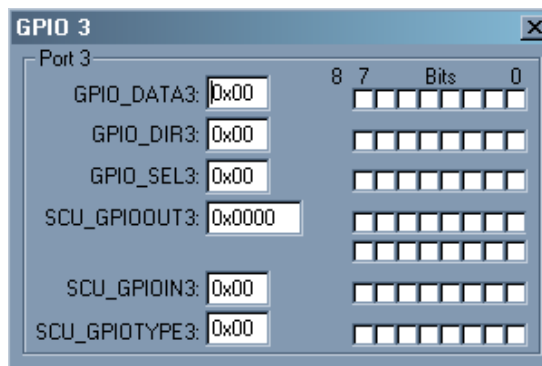
Ο έλεγχος του καταχωρητή μπορεί να γίνει με δύο διαφορετικούς τρόπους.

- Επιλέγουμε τα παράθυρα των περιφερειακών όπως φαίνεται παρακάτω.



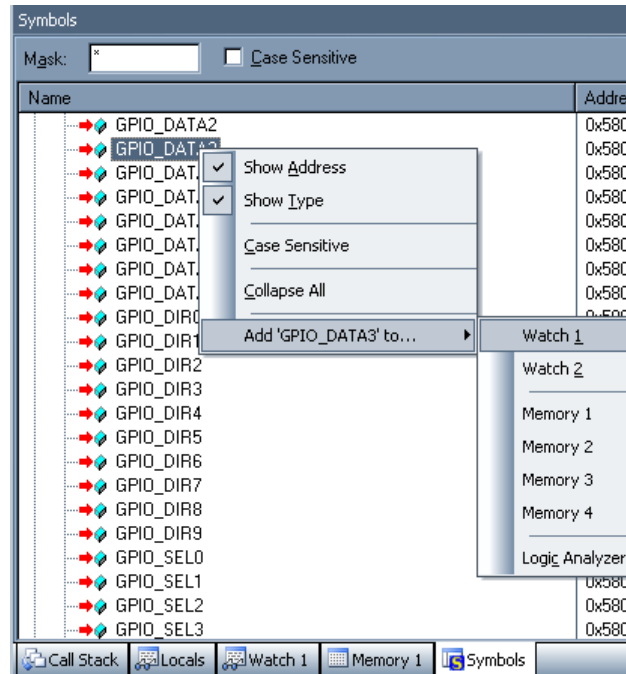
Σχήμα 20: Επιλογή παρακολούθηση κατάστασης GPIO

Το επόμενο παράθυρο θα πρέπει να εμφανιστεί.



Σχήμα 21: Παράθυρο καταχωρητών για το GPIO3

- Επιλέγουμε να παρακολουθούμε τον καταχωρητή GPIO_DATA3 μέσα από την επιλογή Symbols -> Peripheral Registers όπως φαίνεται στο παρακάτω σχήμα.

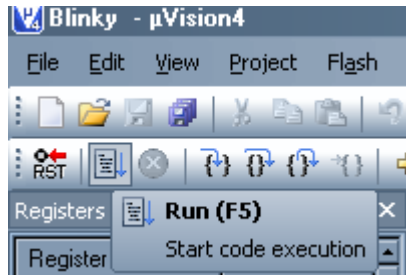


Σχήμα 22: Επιλογή παρακολούθησης καταχωρητή

Στο παράθυρο Watch 1 θα έχουμε τώρα τον καταχωρητή GPIO_DATA3. Έχοντας και τα δυο παράθυρα παρακολούθησης του καταχωρητή GPIO_DATA3 μπορούμε να παρατηρούμε την τιμή του GPIO3.5, αλλά και όλων των GPIO, από 0 έως 7, στο Port 3. Λογικά, **αρχικά το GPIO 3.5 πρέπει να έχει τιμή 1 και κρατώντας πατημένο το πλήκτρο να πάρει την τιμή 0.**

- Ο μικροελεγκτής, με το που ξεκινάμε τη διαδικασία του Debug γίνεται reset. Πατάμε το **Run** ή το **F5** για να ξεκινήσει να εκτελείται ο κώδικας.

Η ανακύλιση των LED ξεκινάει και θα πρέπει να αυξομειώνεται καθώς μεταβάλλουμε το ποτενσιόμετρο. Δοκιμάστε το.



Σχήμα 23: Εκτέλεση κώδικα μέσω του Debugger

- b. Πατήστε το **Stop**, που βρίσκεται δίπλα στο Run.
- c. Παρακολουθήστε την τιμή του GPIO_DATA3. Θα πρέπει να είναι 0x7D.
- d. Πατήστε πάλι το Run ή το F5.
- e. Κρατήστε πατημένο το πλήκτρο INT5 στο αναπτυσσικό και παράλληλα σταματήστε την εκτέλεση του κώδικα πατώντας το Stop στο uVision4.

Ερώτημα 23

Ποιά η τιμή του GPIO3.5 στον καταχωρητή GPIO_DATA3 καθώς το πλήκτρο INT5 είναι πατημένο;

3.3 Χρησιμοποίηση των breakpoints

Τα breakpoints μπορούν να οριστούν σε οποιαδήποτε μέρος του κώδικα μας. Η λειτουργία των breakpoint είναι αρκετά απλή. Όταν η εκτέλεση του κώδικα μας φθάσει στο σημείο στο οποίο έχουμε ορίσει το breakpoint και πριν εκτελεστεί ο κώδικας στο σημείο αυτό, τότε η εκτέλεση σταματάει ή αλλιώς κάνει παύση. Μπορούν να οριστούν και breakpoints υπό-συνθήκη (conditional breakpoints ή αλλιώς watchpoints), τα οποία μπορούν να

ελέγχουν εναλλαγή συγκεκριμένων τιμών σε καταχωρητές, σε μεταβλητές του κώδικα μας και άλλα πολλά.

Παρακάτω θα ορίσουμε ένα απλό breakpoint στο σημείο του κώδικα που προγραμματίζουμε τον Peripheral Reset Register 1, δηλαδή στην εντολή:
SCU->PRR1 = 0x00EE2803;

Ερώτημα 24

Ποιος ο σκοπός της SCU->PRR1 = 0x00EE2803 εντολής

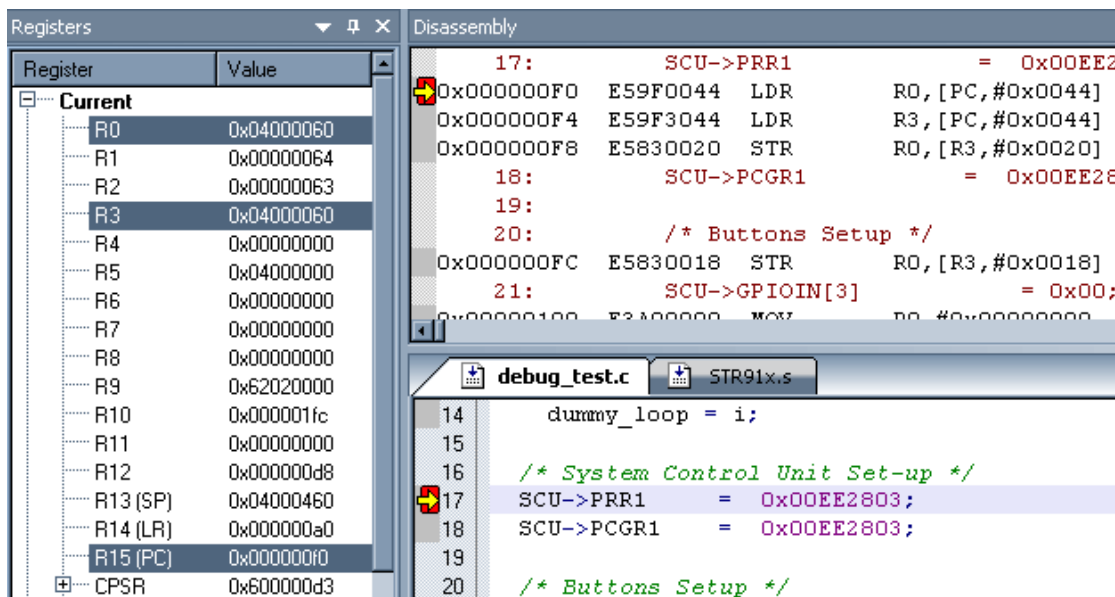
Για να ορίσουμε και να ελέγξουμε το breakpoint ακολουθούμε τα παρακάτω βήματα:

- a. Σταματάμε την εκτέλεση του κώδικα μας πατώντας το **Stop**.
- b. Κάνουμε δεξί κλικ στην γραμμή του κώδικα που θέλουμε να ορίσουμε το Breakpoint, στην γραμμή SCU->PRR1 = 0x00EE2803 και επιλέγουμε το **Insert/Remove Breakpoint** ή πατάμε το F9.
- c. Πατάμε το **Run** για να ξεκινήσει πάλι η εκτέλεση.
- d. Παρατηρούμε εάν και σε ποιο σημείο έχει σταματήσει η εκτέλεση του κώδικα μας.
- e. Πατάμε το πλήκτρο **Rst** και μετά το **Run** για επανεκκίνηση του κώδικα και επανέλεγχο της λειτουργία του breakpoint.

3.4 Χρησιμοποίηση των παραθύρων Disassembly και Registers

Το παράθυρο Disassembly μας δείχνει τον C κώδικα μας και την αντίστοιχη μετάφραση του σε assembly ή μόνο την assembly εάν εκτελούμε απλή assembly και όχι C. Το παράθυρο Registers μας δείχνει τους καταχωρητές του ARM CPU που χρησιμοποιούνται για την επεξεργασία των δεδομένων και για την εκτέλεση του κώδικα μας. Για αυτούς θα μιλήσουμε αργότερα. Τώρα θα προσπαθήσουμε να παρατηρήσουμε το παράθυρο Disassembly και παράλληλα τη λειτουργία του παραθύρου Registers.

Εκτελούμε τον κώδικα μας με τις ρυθμίσεις της προηγούμενης παραγράφου. Η εκτέλεση, λόγω του breakpoint, σταματάει πριν εκτελεστεί η εντολή: SCU->PRR1 = 0x00EE2803.



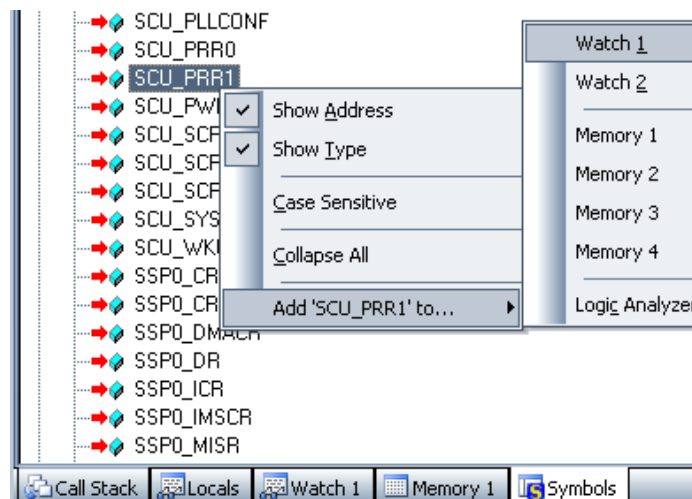
Σχήμα 24: Disassembly και Registers

Στο προηγούμενο σχήμα παρατηρούμε δύο κίτρινα βελάκια. Τα βελάκια αυτά υποδεικνύουν ποιά είναι η επόμενη εντολή που ο CPU θα εκτελέσει. Επίσης παρατηρούμε ότι βρίσκονται πάνω στα κόκκινα κουτιά του breakpoint γιατί η εκτέλεση έχει σταματήσει στο breakpoint που έχουμε ορίσει, δηλαδή στην εντολή SCU->PRR1 = 0x00EE2803.

Ερώτημα 25**Σε ποια θέση μνήμης του MCU βρίσκεται ο καταχωρητής SCU_PRR1;**

Η παραπάνω εντολή της C, στην οποία ο CPU θα καταχωρήσει στην θέση μνήμης του SCU->PRR1 την τιμή 0x00EE2803, έχει μεταφραστεί σε μια σειρά εντολών assembly. Για να δούμε ποιες εντολές είναι αυτές πρέπει να παρακολουθήσουμε τα παράθυρα Disassembly και Registers. Επίσης πρέπει να ορίσουμε τον SCU_PRR1 καταχωρητή στο Watch 1 παράθυρο.

Για να θέσουμε τον καταχωρητή SCU_PRR1 στο Watch 1 παράθυρο ακολουθούμε τη διαδικασία της παραγράφου 5.2. Πηγαίνουμε στο παράθυρο Symbols και από την επιλογή Peripheral Registers θέτουμε τον καταχωρητή SCU_PRR1 στο παράθυρο Watch 1 όπως φαίνεται παρακάτω:

**Σχήμα 25:** Καταχωρητής στο παράθυρο Watch 1

Η αρχική τιμή του SCU_PRR1, είναι αυτή που φαίνεται στο Watch 1.

Watch 1	
Name	Value
`SCU_PRR1	0x00EC0803
<double-click or F2 to add>	

Σχήμα 26: Παράθυρο Watch 1 και αρχική τιμή του SCU_PRR1

Λόγω του breakpoint, η εκτέλεση του κώδικα μας έχει σταματήσει ακριβώς πριν εκτελεστεί η εντολή: SCU->PRR1 = 0x00EE2803. Η μετάφραση της εντολής αυτής από C σε assembly αποτελείται από τις παρακάτω τρεις εντολές, που διακρίνονται στο Disassembly παράθυρο.

```
0x000000F0 E59F0044 LDR    R0,[PC,#0x0044]
0x000000F4 E59F3044 LDR    R3,[PC,#0x0044]
0x000000F8 E5830020 STR    R0,[R3,#0x0020]
```

Η πρώτη στήλη περιέχει την θέση του Μετρητή Προγράμματος – Program Counter (PC). Ο Program Counter αυξάνεται κατά 4bytes (32bit) για κάθε μια εντολή που εκτελείται εκτός και εάν εκτελεστεί κάποιο παρακλάδι (Branch). Την τρέχουσα θέση του Program Counter (PC) μπορούμε να την δούμε από τον καταχωρητή R15 (PC) στο παράθυρο των καταχωρητών (Registers).

Ερώτημα 26

Σε ποια θέση βρίσκεται ο PC όταν η εκτέλεση σταματήσει λόγω του breakpoint που έχουμε θέσει;

Μπορούμε επίσης να βρούμε την τρέχουσα τιμή του PC χρησιμοποιώντας το Disassembly παράθυρο και κοιτώντας την πρώτη στήλη στην οποία το κίτρινο βέλος δείχνει. Παρατηρώντας το Σχήμα 24 μπορούμε να δούμε ότι ο PC, το κίτρινο βέλος, έχει σταματήσει στο breakpoint, στο κόκκινο κουτί του Disassembly παραθύρου. Η διεύθυνση του PC την στιγμή αυτή είναι η 0xF0. Η ίδια διεύθυνση υπάρχει και στον καταχωρητή R15 (PC) στο παράθυρο Registers.

Η assembly εντολή που θα εκτελεστεί στην διεύθυνση 0x214 είναι η:

```
LDR    R0,[PC,#0x0044]
```

Η οποία σημαίνει:

Φόρτωσε τον καταχωρητή R0 με ένα word (4bytes ή 32bits) από την διεύθυνση $PC + 0x0044 + 8$.

Πηγαίνοντας στην διεύθυνση $PC + 0x0048 + 8 = 0xF0\ 0x0044 + 8 = 0x13C$ θα βρούμε την εξής assembly εντολή:

```
0x0000013C 00EE2803 DD      0x00EE2803
```

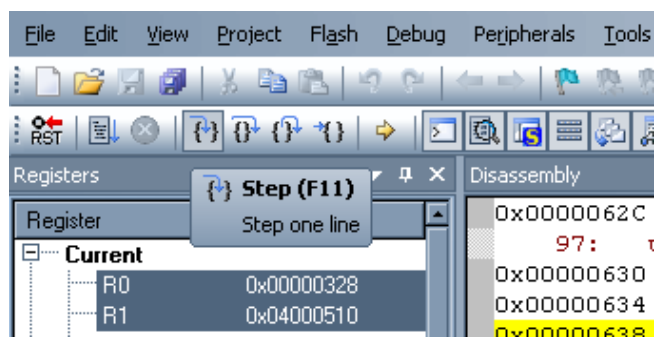
Οι ARM assembly εντολές DD ορίζουν ένα literal pool. Το literal pool είναι μια περιοχή στην μνήμη που χρησιμοποιείται για να αποθηκευτούν σταθερές τιμές, όπως είναι η τιμή 0x00EE2803 που θα καταχωρηθεί στον SCU_PRR1.

Οπότε η εντολή:

```
LDR      R0,[PC,#0x00044]
```

πηγαίνει στην περιοχή μνήμης εκείνη, που προκύπτει από την πρόσθεση $PC + 0x0044 + 8$, στην οποία έχει αποθηκευτεί η τιμή 0x00EE2803. Την τιμή αυτή την φορτώνει στον καταχωρητή R0.

Παρακολουθώντας τα δεδομένα του καταχωρητή R0 πατήστε το πλήκτρο **Step (F11)**, όπως φαίνεται στο επόμενο σχήμα. Ο CPU θα εκτελέσει ένα βήμα, δηλαδή θα εκτελέσει μόνο την τρέχουσα εντολή του PC και θα κάνει παύση.



Σχήμα 27: Εκτέλεση Step

Ερώτημα 27

Ποια εντολή θα εκτελεστεί όταν πατηθεί το Step (F11); Σε ποια θέση βρισκόταν ο PC και σε ποια θέση βρίσκεται τώρα;

Ερώτημα 28

Τι τιμή έχει τώρα ο R0; Ήταν αυτό αναμενόμενο; Δικαιολογήστε το.

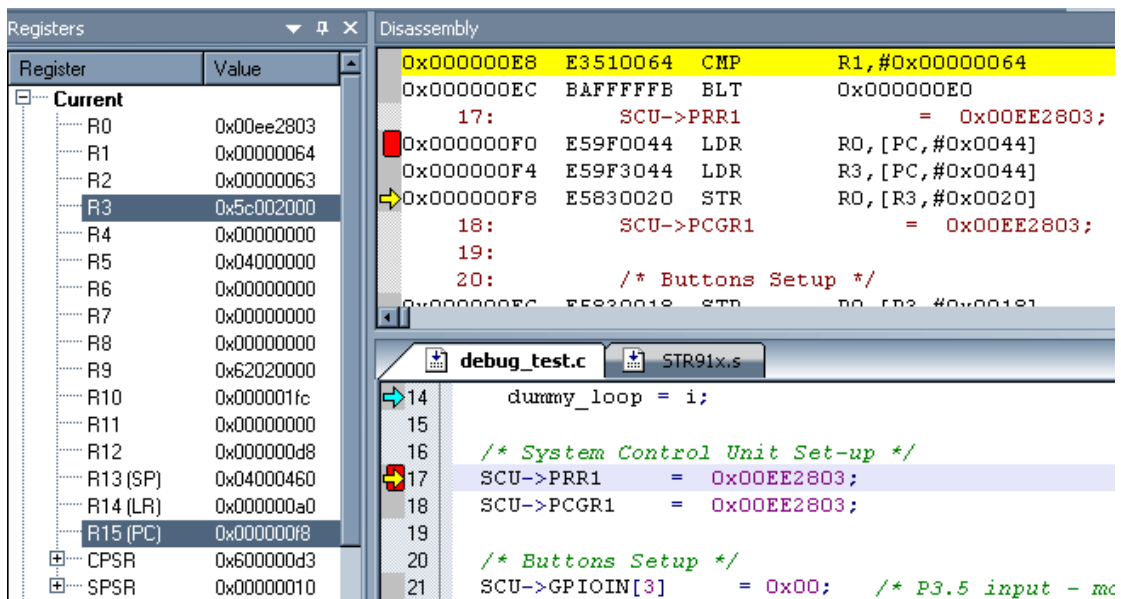
Ερώτημα 29

Ποια η εντολή που υπάρχει στην νέα θέση του PC;
Ποια η λειτουργία της;

Ξαναπατάμε το πλήκτρο Step ή το F11. Η τρέχουσα εντολή στην διεύθυνση του PC εκτελείται.

Ερώτηση 30

**Ποια είναι τα αποτελέσματα της νέας εντολής που εκτελέστηκε;
Που βλέπουμε τα αποτελέσματα αυτά;**



Σχήμα 28: Παράθυρα στην επιλογή Debug

Τώρα ο Program Counter πρέπει να βρίσκεται στην διεύθυνση 0xf8. Τα debug παράθυρα πρέπει να είναι όπως φαίνεται παρακάτω.

Παρατηρούμε ότι στο παράθυρο με τον C κώδικα το κίτρινο βέλος έχει παραμείνει στην εντολή SCU->PRR1 = 0x00EE2803, παρόλο που εκτελέσαμε 2 εντολές assembly. Αυτό συμβαίνει γιατί η εντολή αυτή χρειάζεται 3 assembly εντολές να πραγματοποιηθεί και εμείς έχουμε

εκτελέσει τις 2 από αυτές. Όταν λογικά εκτελέσουμε και την επόμενη και ο PC φτάσει στην διεύθυνση 0xFC τότε η εντολή θα έχει πλήρως εκτελεστεί.

Στο παράθυρο Registers παρατηρούμε ότι ο R0 έχει την τιμή 0x00EE2803. Την απέκτησε όταν εκτελέστηκε η εντολή LDR R0,[PC,#0x0044] στην διεύθυνση 0xF0. Αντίστοιχα ο R1 έχει την τιμή 0x5C002000 και την απέκτησε όταν εκτελέστηκε η εντολή LDR R1,[PC,#0x0044].

Η assembly εντολή στην διεύθυνση 0xF8, στην διεύθυνση στην οποία πρέπει να βρίσκεται τώρα ο PC, είναι η εξής:

```
STR    R0,[R1,#0x0020]
```

Η λειτουργία της είναι να αποθηκεύσει (STR = store) τα δεδομένα του καταχωρητή R0 στην διεύθυνση που υπάρχει στα περιεχόμενα του R1 συν 0x20.

Ερώτημα 31

Ποια διεύθυνση τελικά προκύπτει από το [R1, #0x0020].

Ποια διεύθυνση καταχωρητή του MCU είναι αυτή;

Ερώτημα 32

Τι αποτέλεσμα θα περιμέναμε με την εκτέλεση της STR R0,[R1,#0x0020] εντολής;

Πατάμε για τελευταία φορά το F11 ώστε να εκτελεστεί η παραπάνω εντολή;

Ερώτημα 33

Πώς μπορούμε να δούμε εάν ο καταχωρητής SCU_PRR1 πήρε τελικά την τιμή που περιμέναμε;

4 ΕΡΓΑΣΤΗΡΙΟ 4

- ♦ Μοντέλο προγραμματισμού του ARM
- ♦ Εισαγωγή στην assembly του ARM

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

- m. Αρχείο **mcbstr9.chm** HTML, Κεφάλαιο “Writing Programs”.
- n. Αρχείο **ARM.pdf**, κεφάλαιο A2, Programmers’ Model
- o. Αρχείο **STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για καταχωρητές.
- p. Βιβλίο Θεωρίας **Wayne Wolf**, “Οι Υπολογιστές ως Συστατικά Στοιχεία”. Κεφάλαιο 3, παράγραφος 2.4. Διακοπές.

Εισαγωγή

Στο εργαστήριο αυτό θα αναλύσουμε τον μοντέλο προγραμματισμού του ARM. Θα μας είναι χρήσιμο για την περαιτέρω κατανόηση και εφαρμογή των εργαστηρίων. Θα αναφέρουμε τα βασικά για τους καταχωρητές γενικού και ειδικού σκοπού. Οι καταχωρητές αυτοί χρησιμοποιούνται από το CPU για τη διαδικασία εκτέλεσης ενός προγράμματος.

Στη συνέχεια θα εκτελέσουμε ένα απλό πρόγραμμα και με τη βοήθεια του debugger θα προσπαθήσουμε να κατανοήσουμε την ροή εκτέλεσης του προγράμματος και την χρήση των καταχωρητών γενικού σκοπού.

4.1 Τύποι δεδομένων

Ο ARM υποστηρίζει τους εξής τύπους δεδομένων:

Byte:	8 Bits
Halfword	16 bits
Word	32 Bits

4.2 Καταχωρητές προγραμματισμού

Ο ARM έχει συνολικά 31 καταχωρητές γενικού σκοπού. Σε οποιαδήποτε στιγμή 17 από αυτούς είναι προσπελάσιμοι. Οι υπόλοιποι χρησιμοποιούνται για να αυξήσουν την ταχύτητα εκτέλεσης των **Εξαιρέσεων (exceptions)**. Από το Σχήμα 29 μπορείτε να δείτε ποιες καταστάσεις θεωρούνται Εξαιρέσεις (exceptions). Έτσι ο προγραμματιστής μπορεί να χρησιμοποιήσει οποιοδήποτε από τους 17 προσπελάσιμους καταχωρητές. Τρεις από τους προσπελάσιμους καταχωρητές, ο R13, R14 και R15, έχουν ειδικό ρόλο.

Stack Pointer (R13): Ο καταχωρητής αυτός καταχωρεί τοπικές μεταβλητές συναρτήσεων με την μέθοδο LIFO (Last In First Out – τα τελευταία δεδομένα που μπαίνουν βγαίνουν πρώτα). Χρησιμοποιείται από τις εντολές PUSH, για να εισχωρηθεί στην κορυφή της στοίβας (Stack) ένα νέο στοιχείο και POP για να μετακινήσει ένα στοιχείο από την κορυφή.

Link Register (R14): Ο καταχωρητής αυτός κρατάει την διεύθυνση της επόμενης εντολής μετά από μια Branch and Link (BI ή BLX) εντολή, η οποία είναι ή εντολή που χρησιμοποιείται για να καλεστεί μια υπορουτίνα. Με άλλα λόγια κρατάει την διεύθυνση επιστροφής από την υπορουτίνα. Όταν η υπορουτίνα πρέπει να επιστρέψει, η τιμή του LR αντιγράφεται στον Program Counter. Επίσης χρησιμοποιείται για να κρατάει την διεύθυνση επιστροφής σε περίπτωση που το σύστημα μπει σε κάποια κατάσταση εξαίρεσης. Τις υπόλοιπες φορές μπορεί να χρησιμοποιηθεί σαν καταχωρητής γενικού σκοπού.

Program counter (R15): Κρατάει την διεύθυνση της επόμενης εντολής που θα εκτελεστεί.

CPSR (Current Program Status Register): Κρατάει την τρέχουσα κατάσταση λειτουργίας του CPU όπως αυτές αναφέρονται στην επόμενη παράγραφο.

SPSR (Saved Program Status Register): Χρησιμοποιείται στις εξαιρέσεις και κρατάει την τιμή του CPSR πριν το σύστημα εξυπηρετήσει μια εξαίρεση. Ουσιαστικά χρησιμοποιείται για να σώζει ποια ήταν η προηγούμενη κατάσταση λειτουργίας της CPU ώστε να επιστρέψει σε αυτήν μετά την εξυπηρέτηση της εξαίρεσης.

4.3 Καταστάσεις λειτουργίας

Ο ARM έχει επτά καταστάσεις λειτουργίας συν μια ειδική λειτουργία λογισμικού που ονομάζεται THUMB που μπορεί να χρησιμοποιηθεί παράλληλα με οποιαδήποτε άλλη από τις επτά καταστάσεις. Οι διαφορετικοί τρόποι λειτουργίας παρέχουν έξτρα χαρακτηριστικά για ομαλή λειτουργία του συστήματος και εκτέλεση διαφορετικών λειτουργιών του CPU ανάλογα με την κατάσταση.

ΠΡΟΝΟΜΙΟΥΧΕΣ ΚΑΤΑΣΤΑΣΕΙΣ						
		ΕΞΑΙΡΕΣΕΙΣ				
User	System	FIQ	IRQ	SVC	Undef	Abort
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5

R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8_fiq	R8	R8	R8	R8
R9	R9	R9_fiq	R9	R9	R9	R9
R10	R10	R10_fiq	R10	R10	R10	R10
R11	R11	R11_fiq	R11	R11	R11	R11
R12	R12	R12_fiq	R12	R12	R12	R12
R13/SP	R13/SP	R13_fiq	R13_irq	R13_svc	R13_undef	R13_abort
R14/LR	R14/LR	R14_fiq	R14_irq	R14_svc	R14_undef	R14_abort
R15/PC	R15/PC	R15/PC	R15/PC	R15/PC	R15/PC	R15/PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
-	-	SPSR_fiq	SPSR_irq	SPSR_svc	SPSR_undef	SPSR_abort

Σχήμα 29: Πίνακας καταχωρητών ARM

Υπάρχουν 17 καταχωρητές σε κανονική λειτουργία (User ή System). Στις υπόλοιπες καταστάσεις λειτουργίας υπάρχει ένας ακόμα, ο SPSR. Σε privileged κατάσταση, οι Exception καταστάσεις σύν την System (FIQ, ISQ, Supervisor, Undefined και Abort), οι καταχωρητές R8-R14 για το FIQ και R13-R14 για τις υπόλοιπες καταστάσεις, είναι διαφορετικοί από τους αντίστοιχους σε κανονική λειτουργία. Όταν ο CPU μπει σε κάποια privileged κατάσταση λειτουργίας τότε οι διαφορετικοί καταχωρητές (οι σκιασμένοι στο Σχήμα 29), εναλλάσσονται με τους αντιστοίχους χωρίς να χάσουν τα δεδομένα τους οι πρώτοι. Έτσι για παράδειγμα εάν το σύστημα είναι σε κανονική λειτουργία (user mode) και εκτελέσει την εντολή MOV R13, #0x01, ο καταχωρητής R13 θα πάρει την τιμή 0x01. Εάν αμέσως μετά το σύστημα μπει σε κατάσταση IRQ και εκτελεστεί η εντολή MOV R13, #0x02, τότε η τιμή 0x02 θα γραφτεί στον R13_irq και όχι στον R13. Όταν το σύστημα επανέλθει σε κανονική λειτουργία ο R13 θα έχει ακόμα την αρχική τιμή 0x01.

Κατάσταση	Χρήση	CPSR
User	Κανονική εκτέλεση προγράμματος	0x10
FIQ	Fast Interrupt. Εξυπηρέτηση γρήγορων διακοπών	0x11

IRQ	Εξυπηρέτηση απλών διακοπών	0x12
Supervisor	Reset, Εκκίνηση και εξυπηρέτηση διακοπών λογισμικού (software interrupts)	0x13
Abort	Διαδικασίες λαθών μνήμης	0x17
Undefined	Εξυπηρέτηση μη ορισμένων εντολών	0x1B
System	Προνομιούχες διαδικασίες συστήματος	0x1F

Σχήμα 30: Πίνακας καταστάσεων ARM

4.4 Εξαιρέσεις (Exceptions)

Εξαιρέσεις μπορούν να δημιουργηθούν από εσωτερικές ή και εξωτερικές πηγές, όπως μια διακοπή μέσω του USB ή μέσω της UART. Η προηγούμενη κατάσταση πριν την εξυπηρέτηση της εξαίρεσης σώζεται ώστε το σύστημα να μπορεί να επανέλθει σε κανονική λειτουργία. Επειδή παραπάνω από μια εξαίρεση μπορεί να δημιουργηθεί την ίδια στιγμή, υπάρχουν προτεραιότητες. Στον παρακάτω πίνακα μπορούμε να δούμε τις εξαιρέσεις του ARM και τις αντίστοιχες διευθύνσεις τους.

Εξαίρεση	Κατάσταση λειτουργίας	Vector Interrupt Enable	Κανονική διεύθυνση
Reset	Supervisor		0x00000000
Undefined	Undefined		0x00000004
Software Interrupt	Supervisor		0x00000008
Instruction Fetch Abort	Abort		0x0000000C
Data Access Memory Abort	Abort		0x00000010
IRQ	IRQ	0	0x00000018
		1	Χρήστης
FIQ	FIQ	0	0x0000001C
		1	Χρήστης

Σχήμα 31: Πίνακας Εξαιρέσεων ARM

Οι εξαιρέσεις IRQ προγραμματίζονται σαν **Vectored Interrupts**, θέτοντας το bit VE = 1. Το bit αυτό μπορούμε να το θέσουμε, προγραμματίζοντας τον Control καταχωρητή (Register 1) του **CP15 coprocessor**. Ο CP15 είναι

υπεύθυνος για το MMU (memory management unit), δηλαδή για την μονάδα ελέγχου της μνήμης, την cache και άλλα. Περιέχει 16 καταχωρητές και μπορεί να προγραμματιστεί με συγκεκριμένες εντολές όπως η MCR για εγγραφή ή η MRC για διάβασμα.

4.5 Παρατηρώντας την εκτέλεση ενός προγράμματος

Ανοίγουμε το **Project Lab4_Program_Model.uvproj** αποσυμπιέζοντας το **Lab4_Program_Model.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε το αρχείο **program_model.c**.

```
static int func1(int x, int y);
static int func2(int x, int y);

int main (void)
{
    int result = 0;
    int z1 = 0;
    int z2 = 0;
    int x = 10;
    int y = 5;

    z1 = func1(x, y);
    z2 = func2(x, y);

    result = z1 - z2;

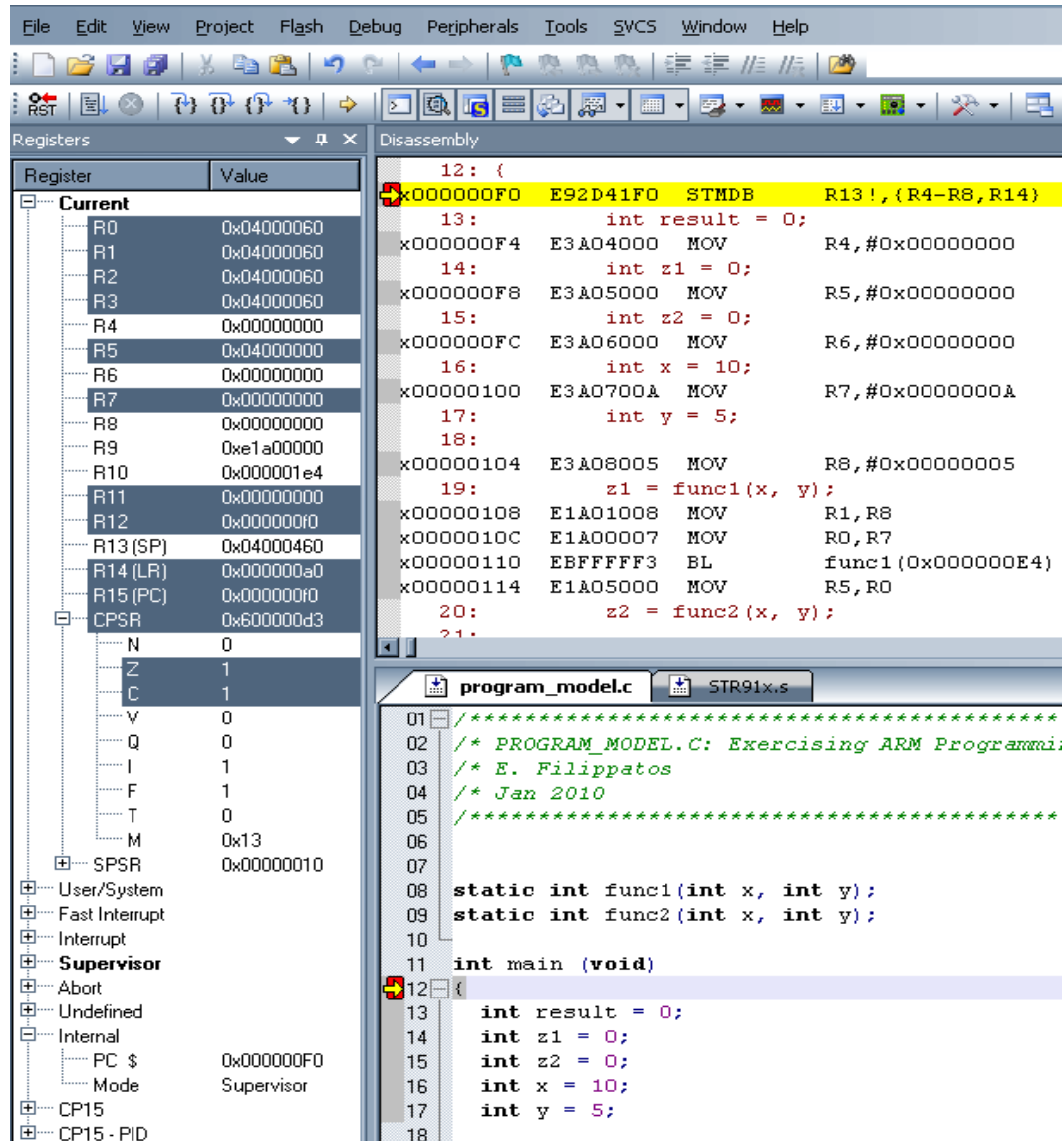
    return result;
}

static int func1(int x, int y)
{
    int result;
    result = x + y;
    return result;
}

static int func2(int x, int y)
{
    int result;
    result = x - y;
    return result;
}
```

- Χτίζουμε και κατεβάζουμε τον κώδικα στο αναπτυξιακό ακολουθώντας τα βήματα των Κεφαλαίων 2.1 και 2.2.
- Στα Options for Target 'Target 1' ελέγχουμε ότι το στο tab Debug είναι επιλεγμένο το Use: ULINK ARM Debugger και όχι το Use Simulation.

- Ξεκινάμε το debugger και τον ρυθμίζουμε ώστε να φαίνονται τα παράθυρα Disassembly, Registers και ο κώδικας program_model.c, όπως φαίνεται παρακάτω.



Σχήμα 32: Παράθυρα στη διαδικασία αποσφαλμάτωσης

- Προσθέτουμε ένα breakpoint στην αρχή της `main` όπως φαίνεται στο προηγούμενο σχήμα.
- Πατάμε το Run. Η εκτέλεση θα σταματήσει. Οι εντολές που θα εκτελεστούν πρώτα φαίνονται στο παράθυρο disassembly.

- Ο κώδικας την στιγμή αυτή βρίσκεται στην πρώτη αγκύλη της main. Παρ' όλ' αυτά ο CPU θα εκτελέσει την πρώτη του εντολή.

0x000000F0 E92D41F0 STMDB R13!,{R4-R8,R14}

Η εντολή αυτή αποθηκεύει τα περιεχόμενα πολλαπλών καταχωρητών στον καταχωρητή SP (Stack Pointer), δηλαδή στην στοίβα (Stack). Είναι ουσιαστικά μια εντολή PUSH.

- Πατάμε το Step ή το F11 για να εκτελείται ο κώδικας μας με βήματα, μια assembly εντολή σε κάθε βήμα. Για κάθε πάτημα του F11 διαβάζουμε τα παρακάτω και παρατηρούμε τους Registers. Πρέπει να γνωρίζουμε ότι ορισμένες εντολές στην C μπορεί να χρειάζονται παραπάνω από μια assembly εντολή για να ολοκληρωθούν.

0x000000F4 E3A04000 MOV R4,#0x00000000

Μετακινεί στον R4 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R4.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R4.

-

0x000000F8 E3A05000 MOV R5,#0x00000000

Μετακινεί στον R5 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R5.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R5.

-

0x000000FC E3A06000 MOV R6,#0x00000000

Μετακινεί στον R6 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R6.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R6

•

0x00000100 E3A0700A MOV R7,#0x0000000A

Μετακινεί στον R7 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R7.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R7.

•

0x00000104 E3A08005 MOV R8,#0x00000005

Μετακινεί στον R8 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R8.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R8.

•

Ερώτημα 34

**Στο σημείο αυτό, ποιο το περιεχόμενο των καταχωρητών R4-R8;
Τί ουσιαστικά έχει αποθηκευτεί στους καταχωρητές αυτούς;**

ΤΙΜΗ	ΕΡΜΗΝΕΙΑ
R4:	
R5:	
R6:	
R7:	
R8:	

•

0x00000104 E3A08005 MOV R8,#0x00000005

Μετακινεί στον R8 την τιμή 0x00000000.

Παρατηρούμε τα δεδομένα του R8.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R8.

0x00000108 E1A01008 MOV R1,R8

Μετακινεί στον R1 τα περιεχόμενα του R8.

Παρατηρούμε τα δεδομένα του R1.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R1.

•

0x0000010C E1A00007 MOV R0,R7

Μετακινεί στον R0 τα περιεχόμενα του R7.

Παρατηρούμε τα δεδομένα του R0.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R0.

•

Ερώτημα 35

Γιατί γίνονται οι δύο τελευταίες μετακινήσεις;

0x00000110 EBFFFFFF BL func1(0x000000E4)

Παρατηρούμε τα δεδομένα των LR, PC.

Branch with Link: Αποθηκεύει στον καταχωρητή R14 (LR - Link Register) την διεύθυνση της εντολής που υπάρχει μετά το Branch και μεταπηδάει στην διεύθυνση 0x000000E4 για να εκτελέσει τον κώδικα της func1.

Μετά την ολοκλήρωση των εντολών που υπάρχουν στο Branch μια μετακίνηση της τιμής του LR στο Programm Counter (PC ή R15) θα επαναφέρει την εκτέλεση στην εντολή αμέσως μετά το Branch.

•

Ερώτημα 36

Η προηγούμενη εντολή βρίσκεται στην διεύθυνση 0x00000110. Μετά την εκτέλεση της, τι τιμή θα περιμένουμε να έχει ο LR (R14) και γιατί;


```
0x000000E4  E1A02000  MOV      R2,R0
0x000000E8  E0820001  ADD      R0,R2,R1
```

Παρατηρούμε τα δεδομένα των R0, R2.

Μετακινεί στον R2 τα περιεχόμενα του R0.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R2.

Προσθέτει τα περιεχόμενα των R1, R2 και τα αποθηκεύει στον R0.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του R0.

•

0x000000EC E12FFF1E BX R14

Επιστρέφει από ένα κάλεσμα συνάρτησης στην διεύθυνση που ορίζεται από τον R14 (Link Register). Ουσιαστικά τα περιεχόμενα του R14 μετακινούνται στο PC.

Παρατηρούμε τα δεδομένα των PC, LR.

Πατάμε το F11 και ελέγχουμε την νέα τιμή του PC. Θα πρέπει να είναι ίδια με τα δεδομένα του LR.

•

Ερώτημα 37

Η συνάρτηση func1 έχει εκτελεστεί. Ποιο είναι το αποτέλεσμα της και σε ποιόν καταχωρητή έχει αποθηκευθεί;

```
0x00000114 E1A05000 MOV    R5,R0
```

Παρατηρούμε τα δεδομένα του R5.

Μετακινεί το περιεχόμενο του R0 στον R5.

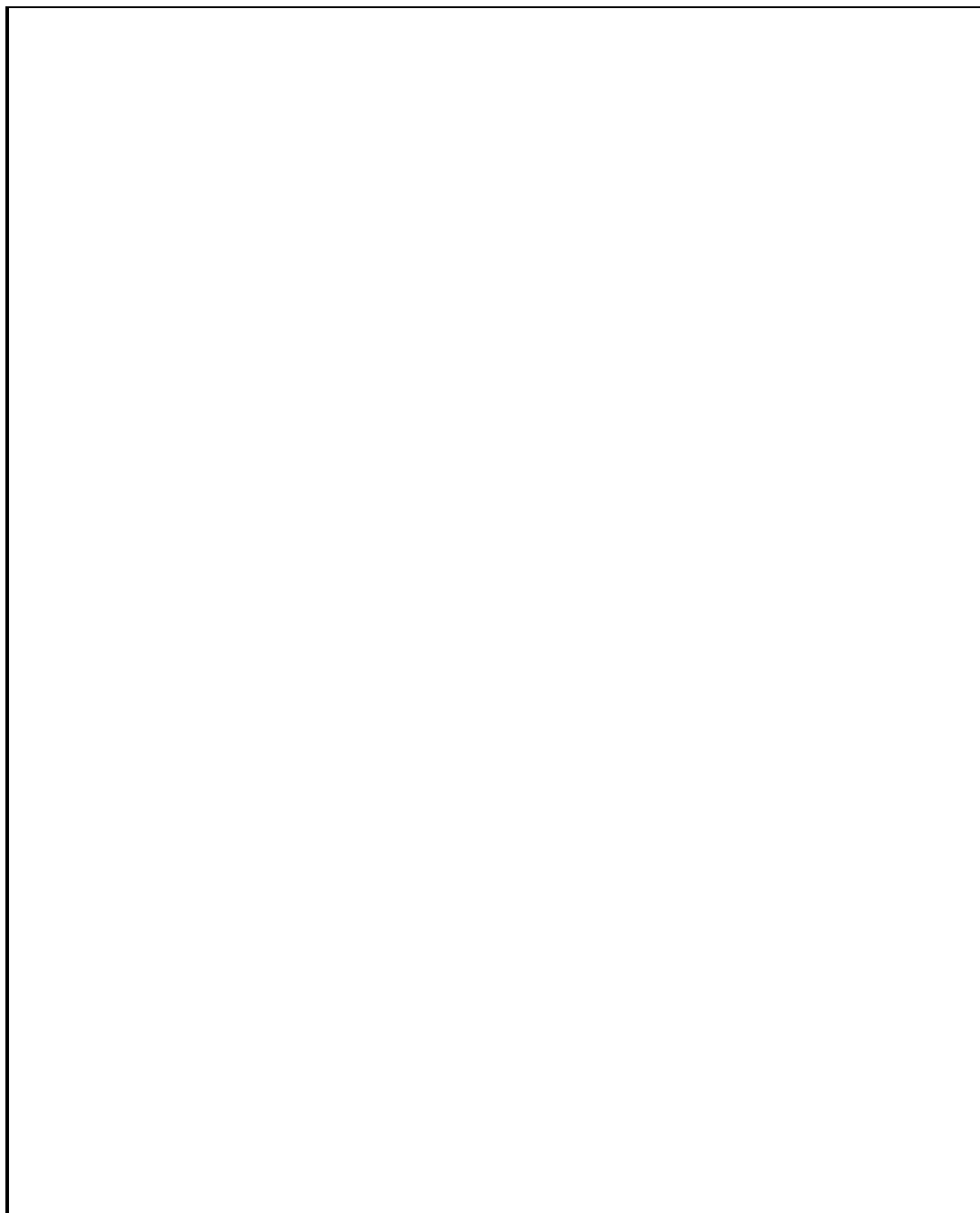
Πατάμε το F11 και ελέγχουμε την νέα τιμή του R5.

•

Στο σημείο αυτό ξεκινάει η εκτέλεση της δεύτερης συνάρτησης.

Ερώτημα 38

Προσθέστε παρακάτω τις εντολές που εκτελούνται και αντίστοιχα σχόλια ώστε να δείξετε ότι έχετε καταλάβει τις εντολές assembly του ARM που χρησιμοποιούνται.



5 ΕΡΓΑΣΤΗΡΙΟ 5

- ♦ Χρησιμοποιώντας τα πλήκτρα
- ♦ Είσοδοι/Εξοδοι Γενικού Σκοπού (GPIO) Μέρος 1ο

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

- q. **Αρχείο mcbstr9.chm HTML**, Κεφάλαια “Writing Programs” και “Theory of Operation->Push Buttons”.
- r. **Αρχείο Blinky.c**, μέσα από τον φάκελο **Lab2_Blinky.rar**.
- s. **Αρχείο MCBSTR9_schematic.pdf**. Το σχηματικό της αναπτυξιακής πλακέτας MCBSTR9 για να δούμε τις συνδέσεις των πλήκτρων.
- t. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για τους GPIO καταχωρητές.
- u. **Βιβλίο Θεωρίας Wayne Wolf**, “Οι Υπολογιστές ως Συστατικά Στοιχεία”. Κεφάλαιο 4, παράγραφος 4. Συσκευές Εισόδου/Εξόδου. Πληκτρολόγια.

Εισαγωγή

Το αναπτυξιακό MCBSTR9 παρέχει τρία πλήκτρα. Το όνομα των πλήκτρων τα έχουμε βρει στο Ερώτημα 1 από το σχηματικό MCBSTR9_schematic.pdf.

Επίσης στο Ερώτημα 17 έχουμε βρει σε ποια pins (GPIO) του μικροελεγκτή είναι τα πλήκτρα αυτά συνδεδεμένα. Στο κομμάτι αυτό του εργαστηρίου θα χρησιμοποιήσουμε ένα από τα πλήκτρα αυτά για να σταματάμε και να ξεκινάμε την διεργασία του ADC στο παράδειγμα Blinky.

Το πλήκτρο INT5 (S2 στο σχηματικό) και οπότε το αντίστοιχο GPIO, τροφοδοτείται με τάση 3,3V μέσω μίας weak pull-up αντίστασης (R43). Δείτε τα στο MCBSTR9_schematic.pdf σχηματικό. Πατώντας το INT5 πλήκτρο το αντίστοιχο GPIO από 1 γίνεται 0. Η ρουτίνα μας πρέπει να μπορεί να ελέγχει την κατάσταση του GPIO. Εάν το πλήκτρο κρατείται πατημένο, δηλαδή είναι σε κατάσταση 0, τότε θα πρέπει να τυπώσουμε στο LCD “Paused” και τα LED να σταματούν την ανάκληση, ενώ όταν το αφήνουμε, δηλαδή είναι στην κατάσταση 1, θα πρέπει να συνεχίζεται η προηγούμενη διαδικασία.

5.1 Διαδικασία προγραμματισμού GPIO

Για να μπορέσουμε να διαβάσουμε μέσα στον κώδικα μας την κατάσταση του GPIO, με το οποίο συνδέεται το πλήκτρο INT5, θα πρέπει πρώτα να το ενεργοποιήσουμε και να το ρυθμίσουμε καταλλήλως. Αυτό θα γίνει προγραμματίζοντας τους κατάλληλους καταχωρητές. Στον παρακάτω πίνακα θα συμπληρώσουμε του καταχωρητές που χρησιμοποιούνται για τον προγραμματισμό των GPIO.

Ερώτημα 39**Συγκεντρωτικός πίνακας καταχωρητών για τα GPIO**

Καταχωρητής	Περιγραφή

Παρακάτω θα συμπληρώσουμε τις τιμές που πρέπει να πάρουν οι παραπάνω καταχωρητές ώστε το GPIO 3.5 να λειτουργεί ως απλή είσοδος.

Ερώτημα 40**Βγάζουμε το GPIO Port 3 από κατάσταση reset:**

--

Ερώτημα 41**Ενεργοποιούμε το ρολόι του GPIO Port 3:**

--

Ερώτημα 42**Απενεργοποιούμε εναλλακτική έξοδο του GPIO: (SCU GPIOOUT)**

--

Ερώτημα 43**Ρυθμίζουμε το GPIO ως είσοδο:**

--

Αντίστοιχες ρυθμίσεις, που μπορούμε να πάρουμε ως παράδειγμα, υπάρχουν για το GPIO 0 του Port 4 (GPIO 4.0) που χρησιμοποιείται ως είσοδος για τον ADC (Analogue to Digital Converter).

Έχοντας καταλλήλως ρυθμίσει τους καταχωρητές του GPIO, μπορούμε τώρα να ελέγχουμε την κατάσταση του. Αυτό γίνεται ελέγχοντας την τιμή στον καταχωρητή:

GPIO3->DR[0x80]

Ο λόγος που βάζουμε 0x80 εξηγείται στις σελίδες 116-117 του STR91xFA_Reference_Manual.pdf.

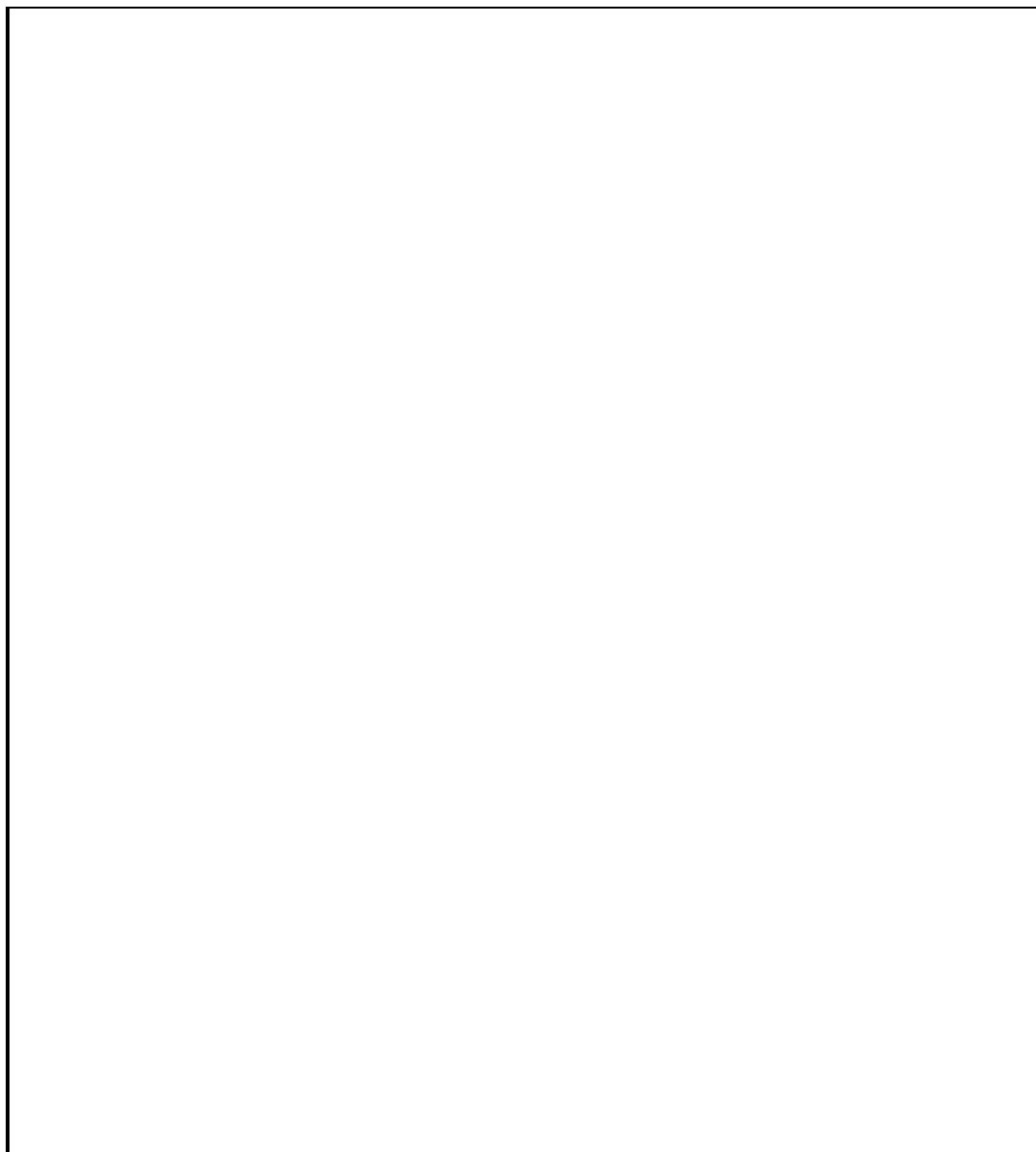
Επειδή το GPIO δεν το έχουμε συνδέσει με κάποιο interrupt, πρέπει ο κώδικας μας να ελέγχει τον καταχωρητή αυτόν συνεχώς (polling). **Συνεπώς πρέπει ο έλεγχος να μπει μέσα στο ατέρμονο βρόγχο του Blinky παραδείγματος.** Σε κάθε κύκλο του βρόγχου η κατάσταση του GPIO πρέπει να ελέγχεται χρησιμοποιώντας μια από τις δύο παρακάτω εντολές και πράττοντας αναλόγως:

```
if (GPIO3->DR[0x80] == 0x20) /* Εάν η κατάσταση του GPIO 3.5 είναι 1 */  
if (GPIO3->DR[0x80] == 0x00) /* Εάν η κατάσταση του GPIO 3.5 είναι 0 */
```

Ανοίγουμε το **Project Blinky** (με τη διαδικασία που αναφέρεται στην παράγραφο 2.1) και στο αρχείο **Blinky.c**, υλοποιούμε τον κώδικα μας.

Ερώτημα 44

Εδώ γράψτε τον κώδικα για το έλεγχο του πλήκτρου. Προσθέστε μόνο τις επιπλέον εντολές που γράψατε για την αρχικοποίηση και έλεγχο του GPIO.



6 ΕΡΓΑΣΤΗΡΙΟ 6

- ♦ **ARM9 Assembly Μέρος 1^ο**
- ♦ **LEDs**
- ♦ **Είσοδοι/Εξοδοι Γενικού Σκοπού (GPIO) Μέρος 2^ο**

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

- v. **Αρχείο mcbstr9.chm HTML**, Κεφάλαιο “Writing Programs”.
- w. **Αρχείο Lab6_GPIOs_assembly.s**, μέσα από τον φάκελο **Lab6_GPIOs_assembly.rar**.
- x. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για καταχωρητές.
- y. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 4, παράγραφος 4. Συσκευές Εισόδου / Εξόδου – LEDs.

Εισαγωγή

Στο εργαστήριο αυτό θα χρησιμοποιήσουμε ARM assembly γλώσσα προγραμματισμού για να ρυθμίσουμε το πλήκτρο INT5 ως είσοδο. Το πλήκτρο INT5 συνδέεται με το GPIO 3.5, δηλαδή με τον 5^ο ακροδέκτη του Port 3. Ο κώδικας μας θα παρατηρεί την κατάσταση του GPIO 3.5. Εάν το πλήκτρο INT5 κρατείται πατημένο πρέπει να γνωρίζουμε ότι το GPIO 3.5 γίνεται 0 ενώ στην αντίθετη περίπτωση που το πλήκτρο δεν είναι πατημένο, το GPIO 3.5 έχει την κατάσταση 1.

Το πρόγραμμα που θα εκτελέσουμε παρατηρεί την κατάσταση του GPIO 3.5 και αναλόγως την κατάσταση του πλήκτρου ανάβει ή σβήνει το LED 0. Εάν το πλήκτρο είναι πατημένο (GPIO 3.5 = 0), τότε το LED 0 (P 7.0 στο σχηματικό) ανάβει, ενώ στην αντίθετη περίπτωση (GPIO 3.5 = 1) το LED 0 σβήνει.

Το συγκεκριμένο κομμάτι κώδικα είναι ήδη υλοποιημένο και δίδεται με τα συνημμένα αρχεία του εργαστηρίου. Παρ' όλ' αυτά ο φοιτητής, αφού πρώτα διαβάσει, κατανοήσει και εκτελέσει την έτοιμη υλοποίηση για το GPIO 3.5 και LED 0, πρέπει να **προσθέσει** δικό του κώδικα σε assembly ώστε επιπλέον το πλήκτρο INT6 να ανάβει το LED 1 (P 7.1 στο σχηματικό).

6.1 Ανάλυση του κώδικα

Ανοίγουμε το Project **Lab6_GPIOs_assembly.uvproj** αποσυμπιέζοντας το **Lab6_GPIOs_assembly.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε το αρχείο **Lab6_GPIOs_assembly.s**. Διαβάζουμε τον κώδικα και τα σχόλια του. Παρακάτω θα δώσουμε μια σύντομη περιγραφή του κώδικα.

```
; APB0, 1 Bridge definition (Peripherals)
```

```
APB0_NBUF_BASE EQU 0x58000000 ; APB Bridge 0 Non-buffered Base Address
```

```
APB1_NBUF_BASE EQU 0x5C000000 ; APB Bridge 1 Non-buffered Base Address
```

Δηλώνουμε την αρχική διεύθυνση των γεφυρών APB0, 1 (non-buffered). Η διεύθυνση θα λειτουργεί ως η βάση στην οποία οι υπόλοιπες διευθύνσεις θα χτίζονται.

; System Control Unit (SCU) definitions

SCU_BASE EQU APB1_NBUF_BASE + 0x2000 ; SCU Base Address

SCU_PCGR1_OFS EQU 0x18 ; Peripheral Clock Gating Register 1 Offset

SCU_PRR1_OFS EQU 0x20 ; Peripheral Reset Register 1 Offset

SCU_GPIOUT7_OFS EQU 0x60 ; GPIO Out Register 7

Δηλώνουμε την αρχική διεύθυνση της Μονάδας Ελέγχου του Συστήματος (System Control Unit - SCU), η οποία ξεκινάει 0x2000 bytes από την αρχική διεύθυνση του APB1. Επίσης δηλώνουμε τις μετατοπίσεις των υπολοίπων καταχωρητών που θα χρησιμοποιήσουμε.

Από το Ερώτημα 39 μπορούμε να καταλάβουμε γιατί μας χρειάζονται οι παραπάνω καταχωρητές.

Ερώτημα 45

Γιατί στις παραπάνω δηλώσεις δεν έχουμε δηλώσει την απόκλιση του καταχωρητή SCU_GPIOUT3, μιας και το GPIO Port 3 το χρειαζόμαστε για το πλήκτρο INT5;

; Values for the SCU equivalent registers

SCU_PCGR1_Val EQU 0x00220000

SCU_PRR1_Val EQU 0x00220000

Δηλώνουμε τις τιμές που πρέπει να προγραμματίσουμε στους καταχωρητές SCU_PCGR1 και SCU_PRR1.

Ερώτημα 46

Γιατί πρέπει να προγραμματίσουμε τους καταχωρητές SCU_PCGR1 και SCU_PRR1 με τις τιμές 0x00220000;

; GPIO Port 7 Addresses

GPIO7_DATA_ADDR EQU APB0_NBUF_BASE + 0x0000D000

GPIO7_DIR_OFS EQU 0x00000400

Δηλώνουμε τις διευθύνσεις για τους καταχωρητές του GPIO Port 7. Ο συνολικός αριθμός των καταχωρητών για κάθε GPIO είναι 8. Εμείς εδώ δηλώνουμε 2 σύν 3 στο SCU παραπάνω. Αυτούς τους 5 καταχωρητές θα χρειαστούμε μόνο διότι οι υπόλοιποι 3 έχουν προκαθορισμένες τιμές 0x00000000 και δεν χρειάζεται να τους αλλάξουμε.

; GPIO Port 7 Program values

SCU_GPIOUT7_Val EQU 0x00005555

GPIO7_DIR_Val EQU 0xFF

Δηλώνουμε τις τιμές που πρέπει να προγραμματίσουμε στους καταχωρητές του GPIO 7.

; GPIO Port 3 Data Address

GPIO3_DATA_ADDR EQU APB0_NBUF_BASE + 0x00009000

Δηλώνουμε την διεύθυνση του καταχωρητή των δεδομένων του GPIO P3.

Ερώτημα 47

Πόσους καταχωρητές για το GPIO Port 3 έχουμε δηλώσει; Γιατί δεν τους έχουμε δηλώσει όλους;

; Area Definition and Entry Point

; Startup Code must be linked first at Address at which it expects to run.

AREA Reset, CODE, READONLY

Η εντολή AREA σημαδεύει την αρχή ενός ξεχωριστού νέου κομματιού κώδικα ή δεδομένων. Πληροφορεί τον συμβολομεταφραστή (assembler) να μεταφράσει ένα νέο κομμάτι κώδικα. Η εντολή AREA έχει την μορφή

`AREA sectionname{,attr}{,attr}...`

sectionname: το όνομα που θα δώσουμε στο νέο κομμάτι κώδικα ή δεδομένων.

attr: Ιδιότητες του νέου κομματιού. Π.χ. εάν είναι κώδικας ή δεδομένα.

; Area Definition and Entry Point

; Startup Code must be linked first at Address at which it expects to run.

ENTRY

NOP ; Wait for OSC stabilization

NOP

Η εντολή ENTRY σημαδεύει την πρώτη εντολή που θα εκτελεστεί. Στον κώδικα αυτό η πρώτη εντολή θα είναι μια NOP. Η NOP σημαίνει No Operation, καμία λειτουργία και διαρκεί 2 κύκλους ρολογιού. Οπότε χρησιμοποιείται όπου χρειάζονται καθυστερήσεις. Εδώ έχουν μπει 10 NOP και θα είναι οι πρώτες εντολές που θα εκτελεστούν αμέσως μόλις η πλακέτα τροφοδοτηθεί με ρεύμα. Στις περιπτώσεις αυτές, επειδή υπάρχουν μεταβατικά φαινόμενα τάσεων και επειδή οι κρύσταλοι θέλουν κάποιο χρόνο για να σταθεροποιηθούν, NOP εντολές χρησιμοποιούνται για δημιουργία μαις αρχικής καθυστέρησης μέχρι το σύστημα να σταθεροποιηθεί.

Ερώτημα 48

Εάν το σύστημα τρέχει στα 25MHz πόσο χρόνο θα διαρκέσουν οι 10 NOP εντολές;


```
; Load SCU Base address to R0
; Then we can move by using offsets from that address
LDR    R0, =SCU_BASE
; Enable GPIO Port 3 and Port 7 clocks
LDR    R1, =SCU_PCGR1_Val
STR    R1, [R0, #SCU_PCGR1_OFS]
```

Φορτώνεται στον R0 η τιμή της σταθεράς SCU_BASE.

Φορτώνεται στον R1 η τιμή της σταθεράς SCU_PCGR1_Val.

Αποθηκεύεται στον R0 + SCU_PCGR1_OFS η τιμή του R1.

Ερώτημα 49

Ποια θα περιμέναμε να είναι τα αποτελέσματα των τριών παραπάνω εντολών;


```
; Set GPIO Port 3 and Port 7 out of reset
LDR    R1, =SCU_PRR1_Val
STR    R1, [R0, #SCU_PRR1_OFS]
```

Φορτώνεται στον R0 η τιμή της σταθεράς SCU_BASE.

Φορτώνεται στον R1 η τιμή της σταθεράς SCU_PRR1_Val.

Αποθηκεύεται στον R0 + SCU_PRR1_OFS η τιμή του R1.

Ερώτημα 50

Ποιά θα περιμέναμε να είναι τα αποτελέσματα των δύο παραπάνω εντολών; Τα δεδομένα του R0 δεν έχουν μεταβληθεί από την πρώτη εντολή:

LDR R0, =SCU_BASE

; Set SCU GPIO Port7 Out at Mode 1: Normal Output

LDR R1, =SCU_GPIOOUT7_Val
STR R1, [R0, #SCU_GPIOOUT7_OFS]

; Set GPIO Port7 Output

LDR R0, =GPIO7_DATA_ADDR
LDR R1, =GPIO7_DIR_Val
STR R1, [R0, #GPIO7_DIR_OFS]

Αντίστοιχα, όπως δείξαμε προηγουμένως, δουλεύουν και οι παραπάνω εντολές.

Check_Button

```
LDR      R0, =GPIO3_DATA_ADDR
LDRB     R0, [R0, #0x080]
CMP      R0, #0x00
BNE      Led0_OFF
B         Led0_ON
```

Το Check_Button είναι μια ετικέτα η οποία χρησιμοποιείται για να ονομάσει το συγκεκριμένο σημείο του κώδικα. Οι ετικέτες χρησιμοποιούνται κυρίως στα branches (διακλάδωση) ώστε αντί να καλούνται branches με αριθμό διεύθυνσης να καλούνται με το ονομά τους. Για παράδειγμα:

B Led0_ON αντί για B 0x00000092

Το παραπάνω κομμάτι του κώδικα είναι πολύ σημαντικό.

Διαβάζει τον καταχωρητή των δεδομένων του GPIO Port 3

```
LDR      R0, =GPIO3_DATA_ADDR
```

και κοιτάει εάν το GPIO 3.5 είναι 0.

```
CMP      R0, #0x00000000
```

Δηλαδή ελέγχει εάν το κουμπί είναι πατημένο. Εάν είναι 0 τότε κάνει ένα branch στην διεύθυνση Led0_OFF διαφορετικά στην διεύθυνση Led0_ON.

```
; Set GPIO Port7 GPIO1 High  
Led0_ON  
    LDR    R0, =GPIO7_DATA_ADDR  
    LDR    R1, =0x01  
    STR    R1, [R0, #0x004]  
    B      Check_Button
```

```
Led0_OFF  
    LDR    R0, =GPIO7_DATA_ADDR  
    LDR    R1, =0x00  
    STR    R1, [R0, #0x004]  
    B      Check_Button
```

Τα δύο αυτά κομμάτια κώδικα ανάβουν ή σβήνουν το LED 0 (P 7.0 στο σχηματικό).

```
LDR    R0, =GPIO7_DATA_ADDR
```

Φορτώνεται, στον R0, η διεύθυνση του καταχωρητή των δεδομένων του GPIO Port 7.

```
LDR    R1, =0x01
```

ή

```
LDR    R1, =0x00
```

Φορτώνεται, στον R1, η τιμή η οποία θέλουμε να περάσουμε στον R0 καταχωρητή. Η τιμή είναι 1 ή 0.

```
STR    R1, [R0, #0x004]
```

Αποθηκεύονται τα περιεχόμενα του R1, στην διεύθυνση του καταχωρητή που προκύπτει από τα δεδομένα του R0 + 0x004.

```
B      Check_Button
```

Αφού τελιώσουμε με το LED επιστρέφουμε στον έλεγχο του κουμπιού.

6.2 Εκτέλεση του κώδικα

Έχοντας διαβάσει και κατανοήσει την δομή του κώδικα, μπορούμε να τον εκτελέσουμε. Χρησιμοποιώντας τον debugger μπορούμε να δούμε την χρησιμοποίηση των καταχωρητών προγραμματισμού R0 και R1 καθώς επίσης και τον Program Counter στα καλέσματα των branches.

- **Βάλτε ένα breakpoint στην διεύθυνση Led0_ON.**
- Πατήστε το πλήκτρο της πλακέτας.
- Θα πρέπει η εκτέλεση του κώδικα να σταματήσει με το πάτημα του πλήκτρου.
- Εκτελέστε τον κώδικά από το σημείο αυτό βηματικά, πατώντας το F11.
- Παρατηρήστε τους καταχωρητές R0, R1.
- Παρατηρήστε τους καταχωρητές περιφερειακών GPIO, όπως φαίνεται στο Σχήμα 23.
- Δείτε ποιά στιγμή το LED ανάβει.

6.3 Δημιουργία κώδικα για το πλήκτρο INT6

Ερώτημα 51

Συνεχίστε τον Lab6_GPIOs_assembly.s κώδικα ώστε το πάτημα του πλήκτρου INT6 να ανάβει το LED 1 (P 7.1 στο σχηματικό).

7 ΕΡΓΑΣΤΗΡΙΟ 7

- ♦ **ARM9 Assembly Μέρος 2^ο**
- ♦ **Εξωτερικές Διακοπές – External Interrupts Μέρος 1^ο**

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

z. Αρχείο mcbstr9.chm HTML, Κεφάλαιο “Writing Programs”.

aa. Αρχείο Lab7_Ext_Int.s, μέσα από τον φάκελο **Lab7_Ext_Int.rar**.

bb. Αρχείο STR9_Programming_Manual.pdf

cc. Αρχείο STR91xFA_Reference_Manual.pdf. Θα το χρησιμοποιούμε ως αναφορά για καταχωρητές.

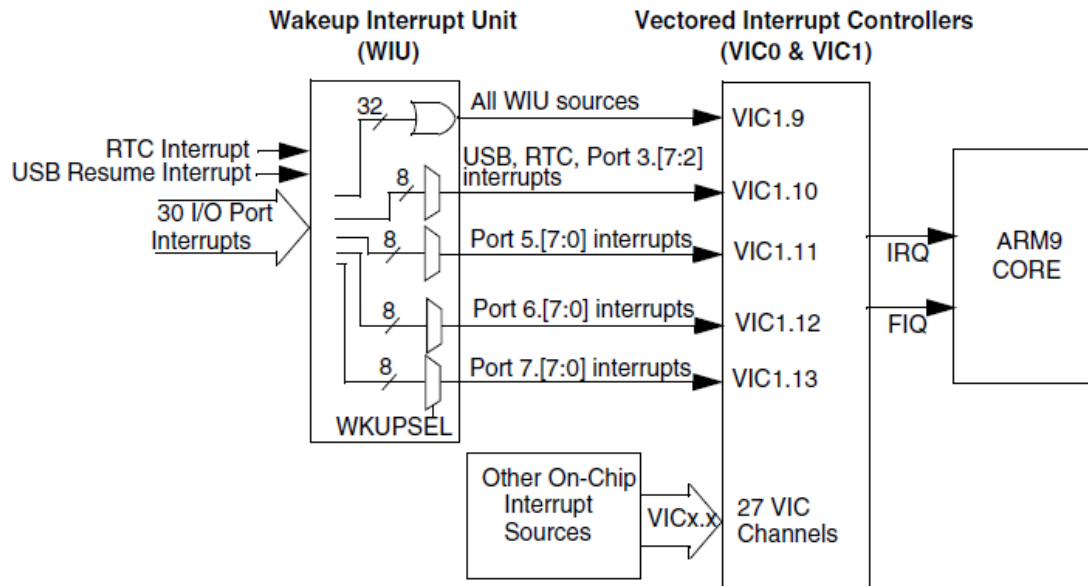
dd. Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”. Κεφάλαιο 3, παράγραφος 2.4. Διακοπές.

Εισαγωγή

Οι διακοπές είναι μια σηματοδότηση προς τον CPU. Ονομάζονται έτσι διότι διακόπτουν την εκτέλεση του τρέχοντος προγράμματος και εξαναγκάζουν τον

CPU να εκτελέσει τον κώδικα που είναι γραμμένος ειδικά για την εξυπηρέτηση της συγκεκριμένης διακοπής. Όταν η εξυπηρέτηση ολοκληρωθεί, δηλαδή όταν το κομμάτι κώδικα γραμμένο ειδικά για την συγκεκριμένη διακοπή εκτελεστεί, η ροή του προγράμματος επιστρέφει στην θέση που ήταν πριν την διακοπή αυτή.

Ανάλογα με τον μικροελεγκτή και τον CPU, υπάρχει διαφορετικός αριθμός αλλά και είδος διακοπών που μπορούν να υποστηριχθούν. Ο STR921FAW44 υποστηρίζει συνολικά 32 εισόδους διακοπών, από τις οποίες οι 5 μπορεί να προέρχονται από εξωτερικές πηγές όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 33: Διάγραμμα ελέγχου διακοπών

Τα εσωτερικά περιφερειακά, όπως οι μετρητές, ο ADC, ή UART μπορούν να διακόψουν τον CPU.

Ο ARM CPU άμεσα μπορεί να δεχθεί ένα FIQ ή ένα IRQ Interrupt. Τα υπόλοιπα IRQs μπαίνουν σε προτεραιότητα ανάλογα με το αριθμό του Vector (το μικρότερο έχει μεγαλύτερη προτεραιότητα) ή ανάλογα με τον προγραμματισμό που υπάρχει για κάθε vector. Vector Interrupt σημαίνει ότι η ρουτίνα εξυπηρέτησης μπορεί να οριστεί έτσι ώστε να βρίσκεται σε οποιαδήποτε θέση μνήμης. Αυτό όμως δημιουργεί κάποια καθυστέρηση,

γιατί η διεύθυνση της ρουτίνας εξυπηρέτησης του Interrupt πρέπει να προσδιοριστεί. Απο την άλλη το FIQ Interrupt έχει πάντα προτεραιότητα απο όλα τα υπόλοιπα Interrupts. Δεν μπαίνει σε κάποια σειρά προτεραιότητας και επίσης η διεύθυνση της ρουτίνας εξυπηρέτησης του FIQ βρίσκεται πάντα σε προκαθορισμένο σημείο. Όλα αυτά έχουν ως αποτέλεσμα το FIQ να είναι ένα πολύ γρήγορο Interrupt. Παρ' όλ' αυτά μόνο μία πηγή μπορεί να το χρησιμοποιεί.

Εμείς θα ασχοληθούμε με τα εξωτερικά Interrupts, τα οποία συνδέονται στα Vectored IRQs. Συγκεκριμένα θα ορίσουμε ένα εξωτερικό interrupt στο πλήκτρο INT5 (GPIO 3.5). Αντί να κάνουμε polling συνεχώς για να ελέγχουμε την κατάσταση του πλήκτρου θα δημιουργήσουμε μια ρουτίνα διακοπής (interrupt routine) η οποία θα καλείται μόνο όταν πατηθεί το πλήκτρο.

Το πρόγραμμα μας θα εκτελεί έναν κύριο βρόχο. Σε κάθε κύλιση του βρόχου τα δεδομένα του R2 ελέγχονται και ανάλογα με την τιμή του τα LED στην οθόνη ανάβουν ή σβήνουν. Η τιμή του R2 αλλάζει κάθε φορά που πατιέται το πλήκτρο INT5, μέσα στην εξυπηρέτηση του Interrupt.

7.1 Ανάλυση του κώδικα

Ανοίγουμε το Project **Lab7_Ext_Int.uvproj** αποσυμπιέζοντας το **Lab7_Ext_Int.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε το αρχείο **Lab7_Ext_Int.s**. Διαβάζουμε τον κώδικα και τα σχόλια του. Παρακάτω θα δώσουμε μια σύντομη περιγραφή των ποιό βασικών σημείων του κώδικα.

Mode_USR	EQU	0x10
Mode_FIQ	EQU	0x11
Mode_IRQ	EQU	0x12
Mode_SVC	EQU	0x13
Mode_ABT	EQU	0x17
Mode_UND	EQU	0x1B
Mode_SYS	EQU	0x1F
I_Bit	EQU	0x80
F_Bit	EQU	0x40

Δηλώσεις σταθερών τιμών που ορίζουν την τιμή του CPSR ή των αντίστοιχων SPSR για τις διάφορες καταστάσεις του ARM.

Τα I_Bit και F_Bit, χρησιμοποιούνται για να απενεργοποιήσουν τα αντίστοιχα Interrupts.

```

UND_Stack_Size    EQU    0x00000000
SVC_Stack_Size    EQU    0x00000008
ABT_Stack_Size    EQU    0x00000000
FIQ_Stack_Size    EQU    0x00000000
IRQ_Stack_Size    EQU    0x00000100
USR_Stack_Size    EQU    0x00000400

```

```

ISR_Stack_Size    EQU    (UND_Stack_Size + SVC_Stack_Size +
ABT_Stack_Size + FIQ_Stack_Size + IRQ_Stack_Size)

```

```

AREA    STACK, NOINIT, READWRITE, ALIGN=3

```

```

Stack_Mem    SPACE    USR_Stack_Size
__initial_sp    SPACE    ISR_Stack_Size

```

```

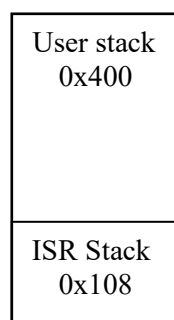
Stack_Top

```

Το κομμάτι αυτό ορίζει το μέγεθος της στοίβας (Stack) για την κατάσταση χρήστη και για οποιαδήποτε κατάσταση εξαίρεσης. Η στοίβα είναι ένας προκαθορισμένος χώρος μνήμης που δεσμεύεται στην αρχή εκκίνησης του συστήματος. Ο χώρος αυτός χρησιμοποιείται από τις διάφορες συναρτήσεις για να αποθηκεύουν τρέχουσες μεταβλητές που άμεσα χρησιμοποιούν, αλλά παράλληλα δεν χρειάζονται μετά το πέρας της εκτέλεσής τους. Όταν μια συνάρτηση επιστρέψει, η μεταβλητή αυτή και οποιαδήποτε άλλη μεταβλητή αποδεσμεύονται από την στοίβα. Ανάλογα με την κατάσταση λειτουργίας του MCU υπάρχει και η ανάλογη στοίβα. Εδώ για παράδειγμα ορίζεται ένα Stack για την κατάσταση χρήστη και ένα για τις εξαιρέσεις. Στη στοίβα αποθηκεύονται δεδομένα από την μεγαλύτερη προς τη μικρότερη διεύθυνση και έτσι το 0x400 και 0x108 είναι τα μεγίστα όρια για τις δύο στοίβες. Για παράδειγμα ένα PUSH μιας 32-bit μεταβλητής στο User Stack θα αποθηκευθεί στην διεύθυνση 0x39C-0x400.

Η επόμενη στην 0x390-39C κτλ.

Η εντολή SPACE δεσμεύει χώρο για την Στοίβα της κατάστασης χρήστη και για την στοίβα των Εξαιρέσεων.



IRQHandler

```
STMDB R13!,{R0-R1}
```

```
; Clear VIC interrupt
```

```
LDR R0,=VIC1_BASE
LDR R1,=0x00000000
STR R1,[R0,#VIC1_VAR_OFS]
```

```
; Clear WIU Interrupt
```

```
LDR R0,=WIU_CTRL_BASE
LDR R1,=WIU_PR_Val
STR R1,[R0,#WIU_PR_OFS]
```

```
MVN R2,R2
LDMIA R13!,{R0-R1}
SUBS PC,R14,#0x4
```

Εδώ ορίζεται η εξυπηρέτηση του IRQ Interrupt. Όταν το κουμπί INT5 πατηθεί, ένα Interrupt θα δημιουργηθεί στο Vector Interrupt 1.10 και ο παραπάνω κώδικας θα εκτελεστεί. Στο τέλος, η εκτέλεση του προγράμματος θα επιστρέψει στην θέση στην οποία ήταν πριν την εξυπηρέτηση του Interrupt αυτού. Η εντολή MVN R2, R2 αντιστρέφει τα δεδομένα του R2 κάθε φορά που πατιέται το κουμπί ώστε ο κύριος βρόχος να γνωρίζει εάν τα LED είναι αναμμένα ή όχι. Έτσι ο R2 μπορεί να είναι 0xFFFFFFFF είναι 0x00000000.

Main_Loop

```
CMP R2,#0xFFFFFFFF
BNE LED_Off
B LED_On
```

```
LED_On
```

```
LDR R0,=GPIO7_DATA_ADDR
LDR R1,=0xFF
STR R1,[R0,#0x3FC]
```

```
B Main_Loop
```

```
LED_Off
```

```
LDR R0,=GPIO7_DATA_ADDR
LDR R1,=0x00
STR R1,[R0,#0x3FC]
B Main_Loop
```

Το σύστημα ξεκινάει με τα LED σβηστά και τον R2=0. Με το πρώτο πάτημα του κουμπιού INT5 ένα Interrupt θα διακόψει τον CPU και η εξυπηρέτησή του θα εντιστρέψει τον R2 από 0x0 σε 0xFFFFFFFF.

Μετά η εκτέλεση του κώδικα θα επιστρέψει στο σημείο του Main_Loop που βρισκόταν πριν το Interrupt.

Το Main_Loop συγκρίνει την κατάσταση του R2 εάν είναι 0xFFFFFFFF. Εάν είναι 0xFFFFFFFF, σημαίνει ότι το κουμπί έχει πατηθεί και οπότε ανάβει τα LED. Διαφορετικά τα κρατάει σβηστά.

Πριν τον ορισμό του Main_Loop γίνονται αρκετές αρχικοποιήσεις σε καταχωρητές. Τον σκοπό των αρχικοποιήσεων αυτών θα τον συζητήσουμε και θα τον συμπληρώσουμε παρακάτω.

Ερώτημα 52

SCU_PCGR0

Τιμή	Εξήγηση

Ερώτημα 53

SCU_PRR0

Τιμή	Εξήγηση

Ερώτημα 54**SCU_PRR1**

Τιμή	Εξήγηση

Ερώτημα 55**SCU_WKUPSEL**

Τιμή	Εξήγηση

Ερώτημα 56**VIC_INTER**

Τιμή	Εξήγηση

Ερώτημα 57**VIC1_VA10R**

Τιμή	Εξήγηση

Ερώτημα 58

VIC1_VC10R

Τιμή	Εξήγηση

Ερώτημα 59

SCU_GPIOOUT7

Τιμή	Εξήγηση

Ερώτημα 60

GPIO7_DIR

Τιμή	Εξήγηση

Ερώτημα 61

WIU_CTRL

Τιμή	Εξήγηση

Ερώτημα 62**WIU_MASK (WIU_MR)**

Τιμή	Εξήγηση

Ερώτημα 63**WIU_TRIG (WIU_TR)**

Τιμή	Εξήγηση

8 ΕΡΓΑΣΤΗΡΙΟ 8

- ♦ **ARM9 Assembly Μέρος 3^ο**
- ♦ **Εξωτερικές Διακοπές – External Interrupts Μέρος 2^ο**

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

ee. **Αρχείο mcbstr9.chm HTML**, Κεφάλαιο “Writing Programs”.

ff. **Αρχείο Lab7_Ext_Int.s**, μέσα από τον φάκελο **Lab7_Ext_Int.rar**.

gg. **Αρχείο STR9_Programming_Manual.pdf**

hh. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για καταχωρητές.

ii. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 3, παράγραφος 2.4. Διακοπές.

Εισαγωγή

Το εργαστήριο αυτό είναι συνέχεια του προηγούμενου εργαστηρίου.

Στο προηγούμενο εργαστήριο αναλύσαμε τον κώδικα, γραμμή-γραμμή για να κατανοήσουμε τις ρυθμίσεις και τον προγραμματισμό που απαιτείται για να ρυθμίσουμε τις εξωτερικές διακοπές στο STR912FAW44 μικροελεγκτή.

Σε αυτό το εργαστήριο θα εκτελέσουμε τον κώδικα μας και θα χρησιμοποιήσουμε τον Debugger για να παρακολουθήσουμε τους καταχωρητές των Interrupt και τη διαδικασία που ακολουθείται. Θα χρησιμοποιήσουμε Breakpoints και θα παρακολουθούμε τους Registers γενικού σκοπού.

Τέλος θα προσπαθήσουμε να μετατρέψουμε τον κώδικα μας ώστε πλέον το Interrupt να έρχεται από το πλήκτρο INT6 και όχι από το πλήκτρο 5.

8.1 Εκτέλεση του κώδικα

Ανοίγουμε το Project **Lab7_Ext_Int.uvproj** αποσυμπιέζοντας το **Lab7_Ext_Int.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε το αρχείο **Lab7_Ext_Int.s**.

Εκτελούμε τον κώδικα σύμφωνα με τη διαδικασία που αναφέρεται στην παράγραφο 2.1 και 2.2.

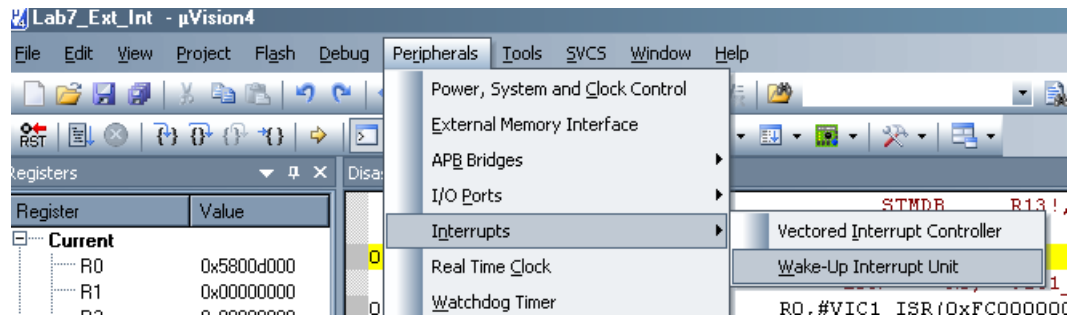
Πατάμε το πλήκτρο INT5 και παρατηρούμε τα LED να ανάβουν. Ξαναπατώντας το πλήκτρο INT5 τα πλήκτρα θα πρέπει να σβήνουν.

8.2 Έλεγχος μέσο του αποσφαλματωτή

Ανοίγουμε τον απασφαλματωτή αφού σιγουρευτούμε ότι έχουμε επιλέξει το *Use ULINK ARM Debugger* κάτω από το Debug tab στο παράθυρο *Options for Target 'Target 1'*. Το *Run to main* πρέπει να είναι επίσης επιλεγμένο.

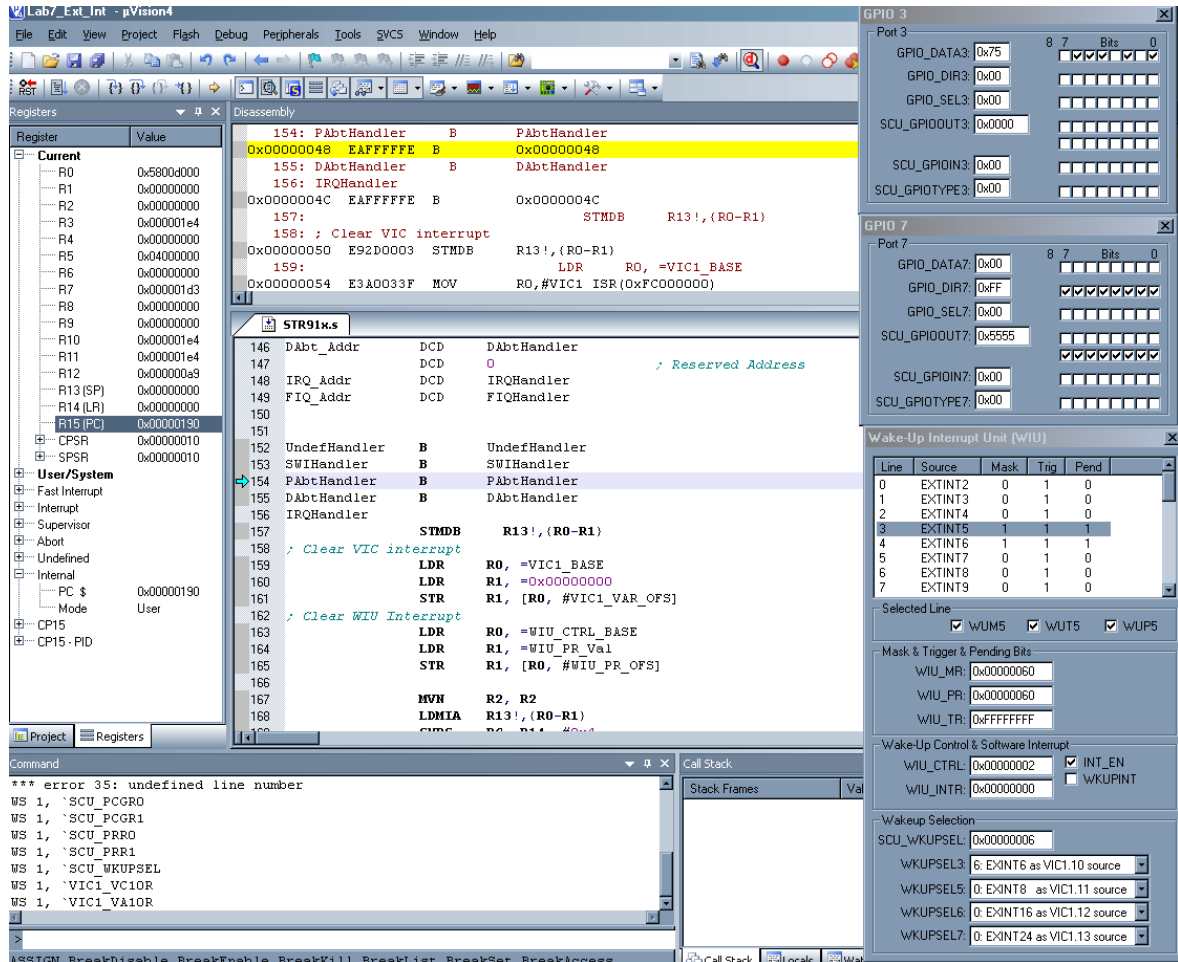
Από την επιλογή Peripherals ανοίγουμε τα εξής παράθυρα:

Wake-Up Interrupt Controller, GPIO3 και GPIO7



Σχήμα 34: Επιλογή Wake Up Interrupt Controller

Η τελική μορφή του Debugger πρέπει να έχει την δομή που φαίνεται στο παρακάτω σχήμα.



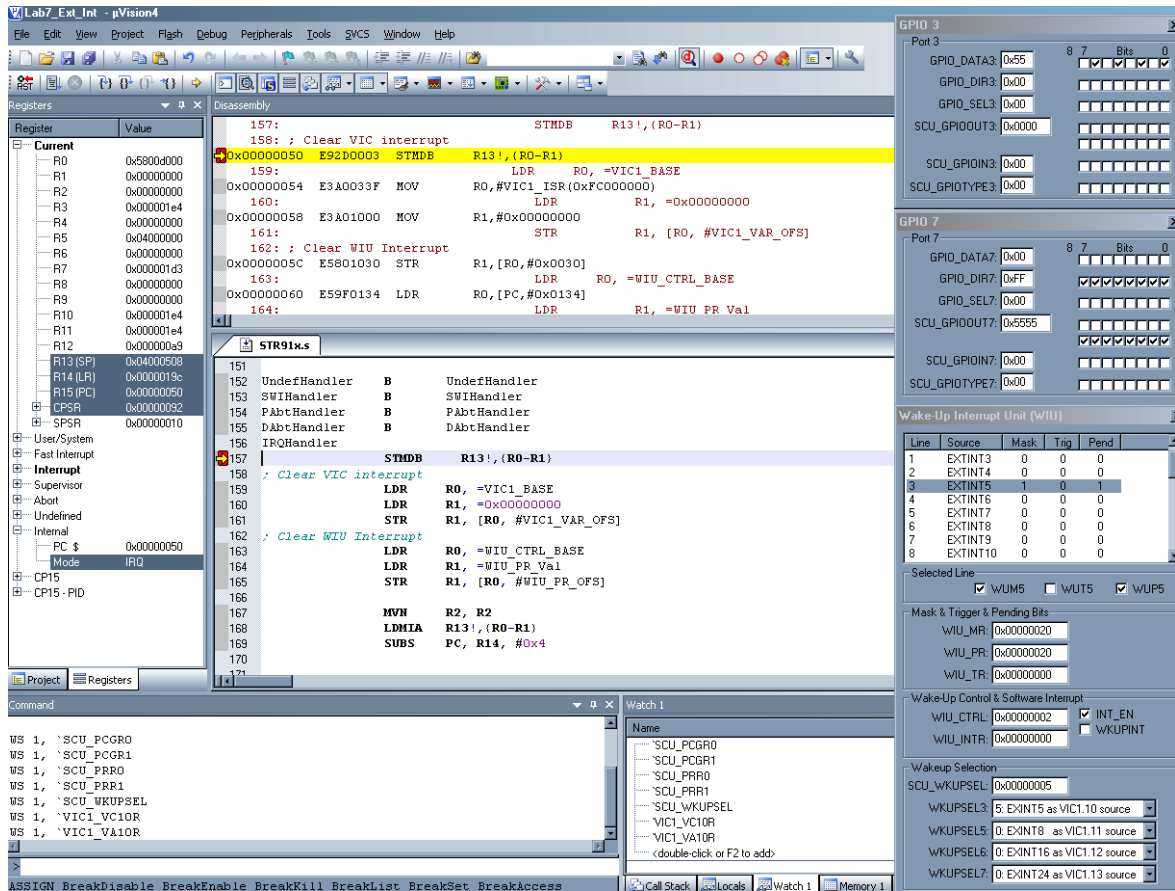
Σχήμα 35: Δομή παραθύρων Debugger για έλεγχο του Εξωτερικού Interrupt

Από το GPIO3 παράθυρο παρατηρούμε ότι το GPIO 3.5 είναι 1. Με το πάτημα γίνεται 0. Αυτό είναι αναμενόμενο και για τον λόγο αυτό έχουμε ορίσει στον WIU Interrupt Trigger Register να ενεργεί το Interrupt με την πτίπουσα ακμή του GPIO.

Στο παράθυρο **Wake-UP Interrupt Unit** επιλέγουμε το **EXTINT5**. Παρατηρούμε την μάσκα (Mask) ότι είναι 1, και το INT_EN bit επίσης.

Θέτουμε ένα breakpoint στην αρχή του IRQHandler και εκτελούμε τον κώδικα πατώντας το Run ή το F5.

Πατάμε το πλήκτρο και ο κώδικας σταματάει στο breakpoint λόγω του Interrupt.



Σχήμα 36: Breakpoint στην εξυπηρέτηση του Interrupt

Στην παραπάνω εικόνα έχουμε τα στοιχεία του debugger την στιγμή που πατήθηκε το πλήκτρο. Στο παράθυρο GPIO3, το GPIO3.5 έχει γίνει 0. Επίσης στο παράθυρο Wake-UP Interrupt Unit το bit WUIP5 έχει γίνει 1 (Pending = 1).

Ερώτημα 64

Ποια η ερμηνεία του bit WUIP5;

Από το σημείο που ο κώδικας μας έχει σταματήσει λόγω του Interrupt και του breakpoint, τον εκτελούμε βηματικά πατώντας το F11. Παρατηρούμε τον καθαρισμό του WIU EXTINT5 Pending bit μετά την εκτέλεση των παρακάτω εντολών:

```
LDR      R0, =WIU_CTRL_BASE
LDR      R1, =WIU_PR_Val
STR      R1, [R0, #WIU_PR_OFS]
```

Ερώτημα 65

Τι ακριβώς κάνουν οι παραπάνω εντολές ώστε να καθαρίσει το Pending bit στον WIU EXTINT5 καταχωρητή.

Όταν ο PC βρίσκεται στην εντολή:

```
MVN      R2, R2
```

παρατηρήστε τον R12 πριν και μετά την εκτέλεση της.

Συνεχίστε την βηματική εκτέλεση του κώδικα έως ότου επιστρέψουμε στην Main_Loop. Όταν φτάσετε την εντολή:

```
CMP      R2, #0xFFFFFFFF
```

παρατηρήστε τις δύο επόμενες εντολές

```
BNE      LED_Off
B         LED_On
```

και τα περιεχόμενα του R2 και απαντήστε:

Ερώτημα 66

Ποια είναι τα περιεχόμενα του R2, και σε ποιο από τα δύο Branches θα περιμένατε ο κώδικας να μεταπηδήσει και γιατί;

Μπορείτε να πειραματιστείτε αλλάζοντας τον αριθμό των LED που ανάβουμε και σβήνουμε.

Ερώτημα 67

Ποιες αλλαγές πρέπει να γίνουν στον κώδικα μας ώστε το External Interrupt να έρχεται πλέον από το INT6 και όχι από το INT5; Σημειώστε παρακάτω τις αλλαγές/αντικαταστήσεις.

9 ΕΡΓΑΣΤΗΡΙΟ 9

♦ Εισαγωγή στους ADCs

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

jj. Αρχείο mcbstr9.chm HTML, Κεφάλαια “Writing Programs” και “Theory of Operation->Potentiometer”.

kk. Αρχεία Blinky.c και LCD.c, μέσα από τον φάκελο **Lab2_Blinky.rar**.

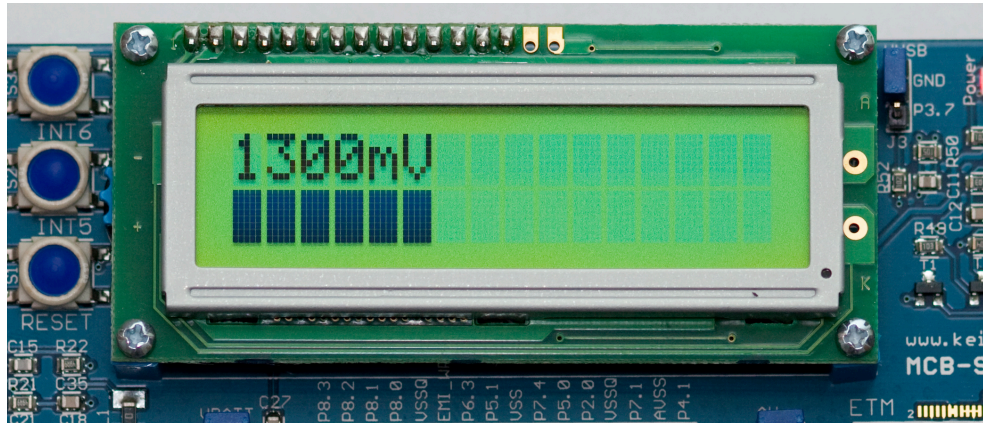
ll. Αρχείο MCBSTR9_schematic.pdf. Το σχηματικό της αναπτυξιακής πλακέτας MCBSTR9 για να δούμε τις συνδέσεις του ποτενσιόμετρου.

mm. Αρχείο STR91xFA_Reference_Manual.pdf. Θα το χρησιμοποιούμε ως αναφορά για τους ADC καταχωρητές.

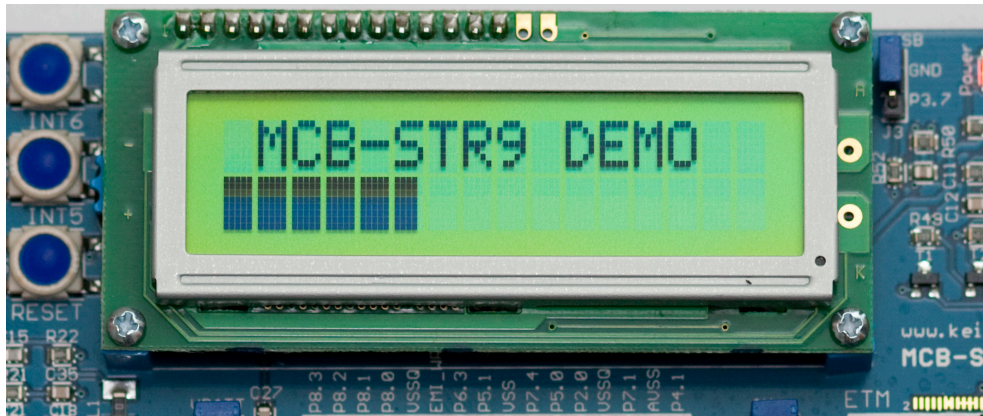
nn. Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”. Κεφάλαιο 4, παράγραφος 4. Συσκευές Εισόδου-Εξόδου, Μετατροπείς A/D και D/A.

Εισαγωγή

Στο εργαστήριο αυτό θα χρησιμοποιήσουμε το παράδειγμα Blinky της Keil. Θα κάνουμε αλλαγές στον κώδικα ώστε στην πρώτη γραμμή της LCD οθόνης, να τυπώνετε η πραγματική τιμή της τάσης του ποτενσιόμετρου POT1 σε mV, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 37: Ένδειξη τάσης ποτενσιόμετρου στο LCD



Σχήμα 38: Ένδειξη LCD πριν τις αλλαγές μας

Στο αρχείο STR91xFA_Reference_Manual.pdf, σελίδα 463, υπάρχουν πληροφορίες για τους ADCs του μικροελεγκτή. Σε γενικές γραμμές η λειτουργία των ADC έχει ως εξής.

- Ο STR912FAW44 έχει 8 κανάλια ADC.
- Κάθε ADC έχει διακριτότητα 10-bit.
- Λόγω ότι η διακριτότητα των ADC είναι 10-bit, η μέγιστη τιμή που δίνουν ως αποτέλεσμα μετατροπής είναι η $2^{10} = 1023$.

- Η μέγιστη και ελάχιστη τιμή που μπορεί ο ADC να μετατρέψει εξαρτάται από την ειδική παροχή τροφοδοσίας στον μικροελεγκτή, VREF. Έτσι για παράδειγμα εάν το VREF είναι 1023mV, τότε η μέγιστη μετατρεπόμενη τιμή του ADC, το 1023, θα αντιστοιχεί σε 1023mV και εάν το VREF είναι 3300mV τότε η μέγιστη τιμή του ADC, το 1023, θα αντιστοιχεί σε 3300mV. Στην περίπτωση αυτή όμως επειδή ο ADC μπορεί να δώσει 1023 τιμές, η αύξηση των τιμών του ADC θα γίνεται κάθε $3300\text{mV}/1023 = 3,2\text{mV}$. Ή διαφορετικά για να υπολογίσουμε την τάση σε mV πολλαπλασιάζουμε την δεκαδική τιμή του ADC με 3,2. Αυτό σημαίνει ότι έχουμε 3,2mV διαφορά ανά βήμα του ADC.

Ερώτημα 68

Έστω ότι ο ADC ήταν 12-bit και το VREF στον μικροελεγκτή είναι 10V. Υπολογίστε το βήμα του ADC ανά mV αλλαγής στην είσοδο του.

Για να μπορέσουμε να δούμε την τιμή της τάσης του ποτενσιόμετρου στην οθόνη του LCD πρέπει πρώτα να μετατρέψουμε την τιμή του ADC σε πραγματικά mV. Είδαμε πριν ότι με VREF 3,2V για κάθε 3,2mV αύξηση της τάσης μετρήσεως του ADC, έχουμε ένα βήμα αύξηση. Δηλαδή την τιμή που μας δίνει ο ADC ως αποτέλεσμα εάν την πολλαπλασιάσουμε με 3,2 θα πάρουμε την πραγματική τιμή της τάσης εισόδου στον ADC.

Ανοίγουμε το **Project Blinky** (με τη διαδικασία που αναφέρεται στην παράγραφο 2.1) και το αρχείο **Blinky.c**.

9.1 Ένδειξη της τάσης του ποτενσιόμετρου στην οθόνη LCD

Στο αρχείο **Blinky.c** πρέπει να προσθέσουμε κώδικα ώστε να τυπώσουμε την τάση του ποτενσιόμετρου στο LCD.

Η μεταβλητή **AD_last** κρατάει την τελευταία μέτρηση του ADC. Η μεταβλητή αυτή είναι εξωτερική μεταβλητή και ανανεώνεται μέσα στο αρχείο **IRQ.c**, στην συνάρτηση **ADC_IRQ_Handler**. Όταν μια μετατροπή έχει τελειώσει από τον ADC ένα ειδικό **interrupt** χτυπάει και η εξυπηρέτηση του interrupt αυτού γίνεται στην συνάρτηση αυτή.

Το πρόγραμμα Blinky τρέχει ουσιαστικά έναν κλειστό ατέρμονο βρόχο. Όταν ξεκινήσει το πρόγραμμα γίνονται πρώτα κάποιες αρχικοποιήσεις στο σύστημα, προγραμματίζοντας κάποιους καταχωρητές. Οι καταχωρητές ή αλλιώς ο χάρτης μνήμης (**Memory Map**) του μικροελεγκτή, υλοποιείται στο αρχείο **C:\Keil\ARM\INC\ST\91x\91x_map.h**. Χρησιμοποιώντας τις δομές μέσα από το αρχείο αυτό, μπορούμε να έχουμε πρόσβαση σε όλους τους καταχωρητές του συστήματος.

Έτσι για παράδειγμα η εντολή:

SCU->GPIOIN[4] |= 0x01;

πηγαίνει στον καταχωρητή GPIOIN4 του SCU (SCU_GPIOIN4 σελίδα 115 στο STR91xFA_Reference_Manual.pdf) και θέτει το πρώτο bit, το bit 0, με την τιμή 1. Αφήνει τα υπόλοιπα bit ως είχαν.

Ερώτημα 69

Ποιες αρχικοποιήσεις γίνονται στην αρχή του **Blink.c** και γιατί;

Μετά εκτελείτε ο κώδικας μέσα στον βρόχο, όπως φαίνεται στον επόμενο πίνακα.

```
while (1) {                                /* Loop forever          */
    for (n = 0x01; n <= 0xFF; n <= 1) {
        GPIO7->DR[0x3FC] = n;             /* Turn on LED              */
    }
```

```

wait();                /* Delay                */
AD_value = AD_last;    /* Read AD_last value        */
if (AD_value != AD_last) /* Make sure that AD interrupt did */
    AD_value = AD_last; /* not interfere with value reading */
AD_value /= 13;        /* Scale to AD_Value to 0 - 78 */
if (AD_old != AD_value) { /* If AD value has changed
    set_cursor (0, 1);
    AD_old = AD_value;
    for (i = 0; i < 16; i++) { /* Disp bargraph according to AD */
        if (AD_value > 5) {
            lcd_putchar (0x05);
            AD_value -= 5;
        } else {
            lcd_putchar (AD_value);
            AD_value = 0;
        }
    }
}

```

Ερώτημα 70

Περιγράψτε τα βήματα που ακολουθούνται μέσα στον ατέρμονο βρόχο.

Ερώτημα 71

Εφόσον το πρόγραμμα μας εκτελεί τον κώδικα μέσα στον ατέρμονο βρόχο, πως η τιμή του ADC (AD_last) αλλάζει;

--

Για να γράψουμε στο LCD θα χρησιμοποιήσουμε την συνάρτηση **lcd_print**. Με την συνάρτηση **set_cursor** επιλέγουμε σε ποια γραμμή και θέση θα βρίσκεται ο cursor. Ουσιαστικά επιλέγουμε που θα πάει να γράψει η lcd_print. Το LCD του αναπτυξιακού έχει 2 γραμμές με 16 θέσεις για χαρακτήρες σε κάθε γραμμή. Έτσι για παράδειγμα εάν θέλουμε να γράψουμε στην θέση 5 της πρώτης γραμμής, επιλέγουμε πρώτα την θέση με την συνάρτηση **set_cursor(4, 0)** (4 και όχι 5 γιατί η πρώτη θέση είναι το 0 και όχι το 1) και μετά γράφουμε με την **lcd_print**. Σε κάθε γράψιμο ο cursor αυξάνεται αυτομάτως.

Η συνάρτηση **lcd_print** παίρνει ως όρισμα **string**. Η μεταβλητή **AD_value** είναι **δεκαδικός** και οπότε πρέπει να μετατραπεί η μεταβλητή **AD_value** σε **string**.

Ερώτημα 72

Εδώ γράψτε τον κώδικα, ώστε να βλέπουμε στην LCD οθόνη την τάση του ποτενσιόμετρου σε mV.

10 ΕΡΓΑΣΤΗΡΙΟ 10

- ♦ **Real Time Clock Μέρος 1^ο**
- ♦ **Alarm και Periodic Interrupts Μέρος 1^ο**

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

οο. **Αρχείο mcbstr9.chm HTML**, Κεφάλαια “Writing Programs”.

pp. **Αρχεία RTC.c και IRQ.c**, μέσα από τον φάκελο **Lab10_RTC.rar**.

qq. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για τους καταχωρητές του MCU αλλά και για την μονάδα RTC.

rr. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 4, παράγραφος 9. Ρολόι Ξυπνητήρι.

Εισαγωγή

Στο εργαστήριο αυτό θα χρησιμοποιήσουμε το Real Time Clock του μικροελεγκτή και θα υλοποιήσουμε ένα ψηφιακό ρολό. Το ψηφιακό ρολόι έχει την δυνατότητα προγραμματισμού ενός Alarm κάποια συγκεκριμένη χρονική στιγμή. Το Alarm αυτό είναι συνδεδεμένο με ένα Interrupt το οποίο θα ανάβει τα LED για 8 δευτερόλεπτα. Η ώρα εμφανίζεται στο LCD όπως φαίνεται στην παρακάτω φωτογραφία.



Σχήμα 39: Οθόνη Ψηφιακού Ρολογιού

Το Real Time Clock του μικροελεγκτή είναι ένα ενσωματωμένο κομμάτι υλικού και το μόνο που χρειάζεται για να λειτουργήσει εκτός από την ανάλογη τάση, που μπορεί να δοθεί και ξεχωριστά από το υπόλοιπο IC, είναι ένας απλός κρύσταλλος 32,768KHz.

Τα κύρια χαρακτηριστικά του RTC είναι:

- Ημερολόγιο 9999 ετών σε BCD (Binary Coded Decimal).
- Φορμάτ 24 ωρών σε BCD (Binary Coded Decimal).
- Ξεχωριστή είσοδος ρολογιού και τάσεως ώστε να μπορεί να λειτουργεί ολοκληρωτικά αυτόνομα ακόμα και από μία μπαταρία χωρίς να το επηρεάζει εάν το υπόλοιπο κομμάτι του μικροελεγκτή είναι σε τάση.
- Υποστήριξη δίσεκτου έτους.
- Προγραμματιζόμενο Alarm Interrupt μέγιστης διάρκειας ενός μήνα.
- Ευκρίνεια σε millisecond.

- Τα περιεχόμενα των καταχωρητών της ώρας και της ημερομηνίας είναι τα μόνα στον μικροελεγκτή STR912FAW44 που δεν επιστρέφουν στις προκαθορισμένες (default) τιμές τους μετά απο Reset.
- Όταν ο μικροελεγκτής είναι σε Sleep Mode οι ταχύτητες των εσωτερικών ρολογιών μειώνονται αρκετά για εξοικονόμηση ενέργειας. Η λειτουργία του RTC όμως δεν επηρεάζεται μιας και έχει δικό του εσωτερικό δίκτυο ρολογιού που δημιουργείται από τον εξωτερικό κρύσταλλο.
- Εκτός από το Alarm Interrupt μπορεί να δημιουργήσει και περιοδικά Interrupts ανά 1024Hz, 128Hz, 16Hz και 2Hz.

Ερώτημα 73

Βρείτε από το STR91xFAxxx.pdf σε ποια pins του μικροελεγκτή πρέπει να συνδεθεί ο 32,768KHz κρύσταλλος. Στη συνέχεια βρείτε από το σχηματικό MCBSTR9_schematic.pdf εάν στα αντίστοιχα pins είναι συνδεδεμένος κάποιος κρύσταλλος και σημειώστε το όνομα που έχει στο σχηματικό.

Στο εργαστήριο αυτό θα εκτελέσουμε το πρόγραμμα και θα δούμε την εναλλαγή της ώρας στο LCD. Το Alarm είναι προγραμματισμένο να δώσει ένα Interrupt στα 30sec από την αρχική ώρα εκκίνησης εκτέλεσης του προγράμματος. Θα παρατηρήσουμε ότι μετά από αυτά τα 30sec όλα τα LED ανάβουν. Παραμένουν αναμμένα για 8sec και μετά σβήνουν.

Στη συνέχεια θα κάνουμε αλλαγές στον κώδικά μας για να ξεκινάμε το ρολόι από διαφορετική ώρα αλλά και για να πειραματιστούμε με το Alarm Interrupt.

10.1 Εκτέλεση του κώδικα

Ανοίγουμε το Project **Lab10_RTC.uvproj** αποσυμπιέζοντας το **Lab10_RTC.rar** αρχείο, που δίδεται με τα υπόλοιπα αρχεία του εργαστηρίου. Ανοίγουμε τα αρχεία **RTC.c** και **IRQ.c**.

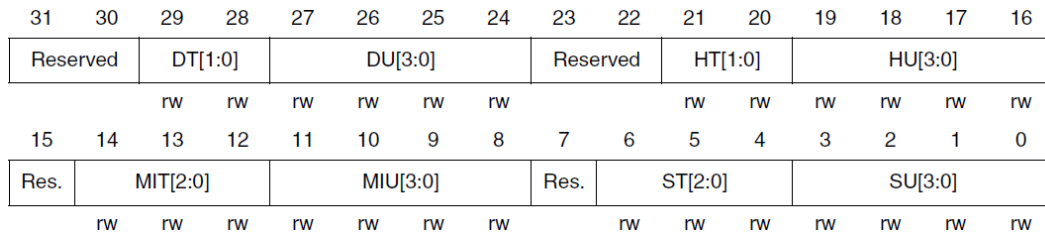
Εκτελούμε τον κώδικα σύμφωνα με τη διαδικασία που αναφέρεται στην παράγραφο 2.1 και 2.2.

Παρατηρούμε την εναλλαγή της ώρας. Περιμένουμε 30sec για να χτυπήσει το Alarm Interrupt και ανάψουν τα LED. Πατάμε το Reset για επανεκκίνηση.

10.2 Ανάλυση του κώδικα

Παρακάτω θα αναλύσουμε τη διαδικασία προγραμματισμού του μικροελεγκτή για το παράδειγμα του Ψηφιακού Ρολογιού. Το σύστημα βασίζεται στη λειτουργία του Real Time Clock (RTC) και στις δυνατότητες που έχει για προγραμματισμό Alarm και παραγωγή Interrupt.

Δύο ειδών Interrupt έχουν προγραμματιστεί στο RTC. Το Alarm Interrupt και το Periodic Interrupt. Το Periodic Interrupt είναι υπεύθυνο για την ανανέωση της ώρας στην οθόνη. Είναι προγραμματισμένο να χτυπάει κάθε 2Hz (0.5sec) και οπότε κάθε 2Hz η τιμή του καταχωρητή RTC_TR που κρατάει την ώρα, διαβάζετε και τυπώνετε στην οθόνη. Το διάβασμα του καταχωρητή RTC_TR γίνεται με έναν ιδιαίτερο τρόπο μιας και ο καταχωρητής κρατάει την ώρα σε μορφή BCD με τα χαμηλά bit να αντιστοιχούν στις μονάδες και τα υψηλά στις δεκάδες. **Διαβάζουμε το STR91xFA_Reference_Manual.pdf σελίδα 147.** Στο επόμενο σχήμα φαίνεται η δομή του καταχωρητή RTC_TR.



Σχήμα 40: Καταχωρητής RTC_TR

Τ

Bytes	Περιγραφή	Τιμές
0-3	Seconds Μονάδες	0-9
4-6	Seconds Δεκάδες	0-5
8-11	Minutes Μονάδες	0-9
12-14	Minutes Δεκάδες	0-5
16-19	Hour Μονάδες	0-9
20-21	Hour Δεκάδες	0-2
24-27	Date Μονάδες	1-9
28-29	Date Δεκάδες	0-3

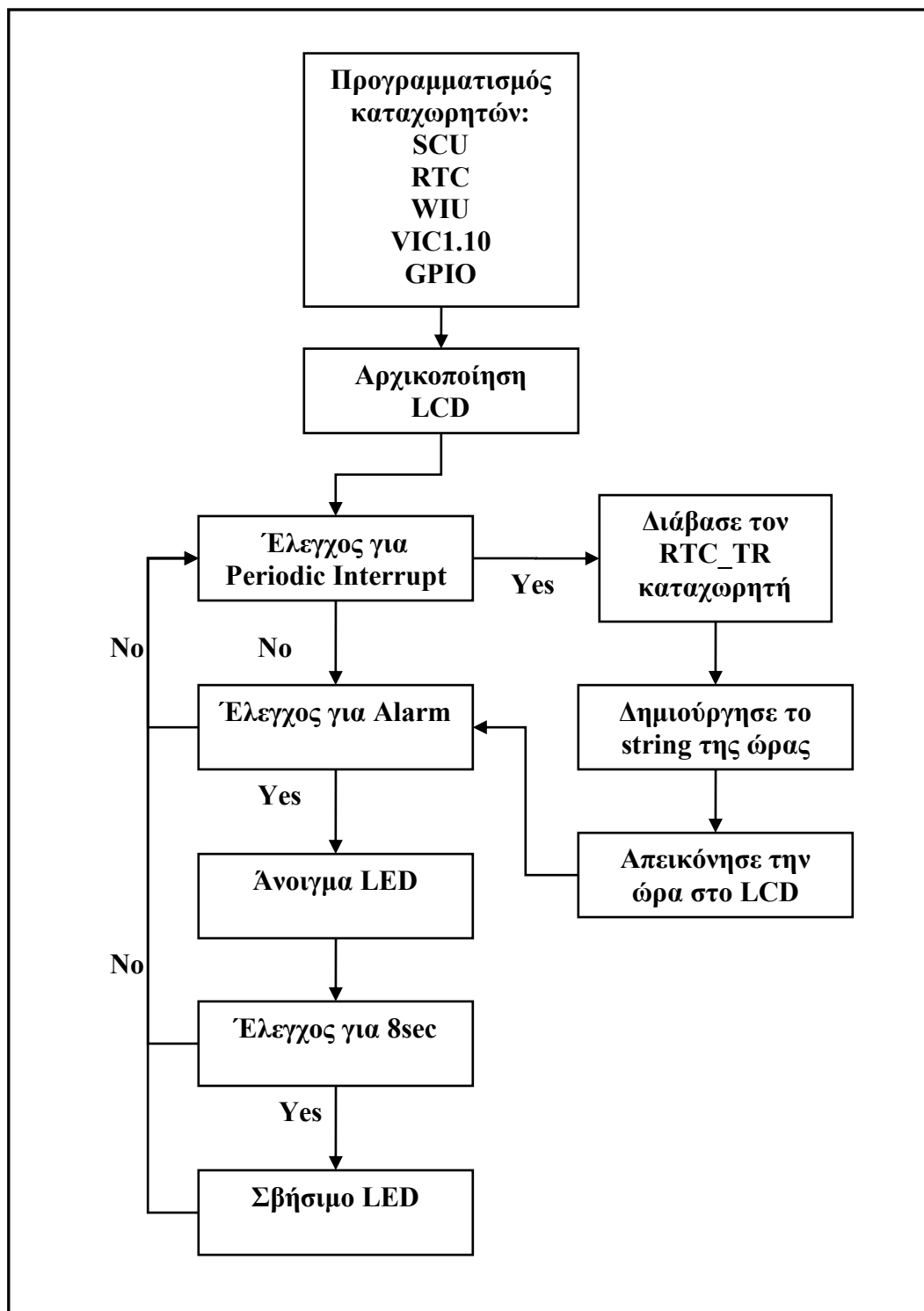
Σχήμα 41: Πίνακας τιμών καταχωρητή RTC_TR

10.2.1 Αποτύπωση της ώρας

Η ώρα αποτυπώνεται στην οθόνη με την συνάρτηση **lcd_print** μετά από την παρακάτω διαδικασία:

```
if (new_time_value){
    get_time_str(time_str);
    set_cursor(3, 1);
    lcd_print(time_str);
    new_time_value = 0;
}
```

Το παρακάτω σχηματικό μας δείχνει την ροή του προγράμματος μας.



Σχήμα 42: Ροή προγράμματος Ψηφιακού Ρολογιού

Η `lcd_print` συνάρτηση, κοιτώντας μέσα στο αρχείο `LCD.c` ορίζεται ως

void lcd_print (unsigned char const *string)

και οπότε δέχεται ως όρισμα string. Άρα τα περιεχόμενα του RTC_TR καταχωρητή πρέπει να μετατραπούν στην αντίστοιχη μορφή. Για να μπορέσουν να τυπωθούν.

Η συνάρτηση που καλούμε για να μας διαβάσει τις τιμές του καταχωρητή RTC_TR και να τις μετατρέψει σε μορφή string είναι η εξής:

```
int get_time_str(char *time)
{
    sprintf(time,"%02d:%02d:%02d",get_hour(),get_minutes(),get_seconds());
    return 0;
}
```

Η get_time_str με την σειρά της καλεί τρεις επιπλέον συναρτήσεις για να δημιουργήσει το string με την χρήση της **sprintf**. Οι τρεις αυτές συναρτήσεις είναι οι παρακάτω:

```
int get_seconds(void)
{
    int seconds = 0;
    seconds = bcd2dec(RTC->TR & 0x7F);
    return seconds;
}

int get_minutes(void)
{
    int minutes = 0;
    minutes = bcd2dec((RTC->TR & 0x7F00) >> 8);
    return minutes;
}

int get_hour(void)
{
    int hour = 0;
    hour = bcd2dec((RTC->TR & 0x3F0000) >> 16);
    return hour;
}
```

Κάθε μία από τις παραπάνω συναρτήσεις διαβάζει το αντίστοιχο μέρος του RTC_TR καταχωρητή που χρειάζεται. Έτσι για παράδειγμα, η συνάρτηση **get_seconds** θα διαβάσει τα πρώτα 7bits του καταχωρητή για να πάρει μόνο την τιμή των seconds. Η τιμή αυτή είναι σε μορφή BCD και πρέπει να

μετατραπεί σε δεκαδικό (decimal). Η παρακάτω συνάρτηση, μετατρέπει μια τιμή της μορφής BCD σε δεκαδική και μας την επιστρέφει.

```
int bcd2dec(int bcd)
{
    bcd = bcd & 0xFF;
    return ((bcd>>4)*10) + bcd%16;
}
```

Εφόσον η συνάρτηση `get_time_str` δημιουργήσει το string της ώρας με την παραπάνω διαδικασία, το επιστρέφει έμμεσα μέσα από την ίδια της την παράμετρο.

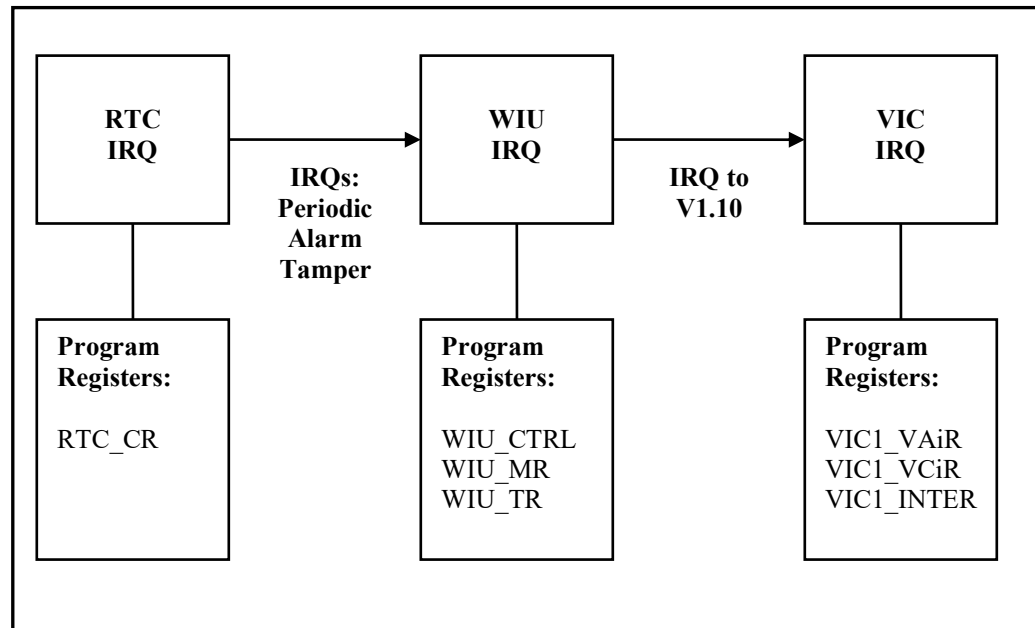
Για να τυπωθεί η ώρα στην οθόνη πρέπει η συνθήκη,

if (new_time_value)

να είναι αληθής. Η τιμή της μεταβλητής `new_time_value` γίνεται 1 όταν ένα Periodic Interrupt εξυπηρετηθεί.

10.2.2 Periodic RTC Interrupt

Όπως ήδη έχουμε αναφέρει το RTC μπορεί να δημιουργεί περιοδικές διακοπές σε χρόνους 1024Hz, 128Hz, 16Hz και 2Hz. Για να συμβεί αυτό ο ανάλογος προγραμματισμός των απαραίτητων καταχωρητών πρέπει να προηγηθεί. Τα RTC Interrupts συνδέονται στα Vector Interrupts μέσω της μονάδας Wake-UP Interrupt όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 43: Διαδικασία RTC Interrupt

Την περιοδική διακοπή την προγραμματίσαμε για τον λόγο να την χρησιμοποιούμε στην ανανέωση της οθόνης. Μας παρέχει έναν σταθερό χρόνο καθυστέρησης χωρίς την χρήση κάποιου for ή while loop που θα μας μπλόκαρε το σύστημα. Ο σταθερός χρόνος καθυστέρησης, δηλαδή της περιοδικής διακοπής είναι 0,5sec. Οπότε κάθε 0,5sec διαβάζουμε την ώρα και την γράφουμε στο LCD. Θα μπορούσαμε βέβαια μέσα σε έναν ατέρμονο βρόχο να διαβάζουμε συνεχώς τον καταχωρητή των δεδομένων της ώρας και κάθε φορά να γράφουμε στο LCD είτε έχει αλλάξει η ώρα είτε όχι, αλλά αυτό θα δημιουργούσε μεγάλο φόρτο.

10.2.3 Alarm RTC Interrupt

Το Alarm Interrupt του RTC ακολουθεί την ίδια λογική με αυτή των περιοδικών διακοπών όπως φαίνεται στο Σχήμα 42. Δηλαδή περνάει μέσα από την μονάδα WIU για να φθάσει στα Vector Interrupts. Η κύρια διαφορά ανάμεσα στα Alarm και στα Periodic Interrupts είναι ότι ο Alarm Interrupt πρέπει να ξαναπρογραμματιστεί εφόσον χτυπήσει, εάν πρέπει να τον ξαναχρησιμοποιήσουμε μιας και ο χρόνος που είχε οριστεί για να χτυπήσει έχει πλέον περάσει.

10.3 Δοκιμή στις αλλαγές ώρας και Alarm

Παρακάτω θα προσπαθήσουμε να κάνουμε αλλαγές στον κώδικα ώστε να δώσουμε διαφορετική αρχική τιμή στην ώρα. Επίσης θα πρέπει να διαβάσουμε το Κεφάλαιο 5 RTC από το κείμενο STR91xFA_Reference_Manual.pdf για να απαντήσουμε μερικές από τις παρακάτω ερωτήσεις.

- Βρείτε στον κώδικα, στο αρχείο RTC.c, που ορίζεται η αρχικοποίηση της ώρας και αλλάξτε την.

Ερώτημα 74

Ποιες είναι οι μεταβλητές που αρχικοποιούν την ώρα;

Ερώτημα 75

Ποια είναι η διαδικασία που πρέπει να ακολουθείτε για να προγραμματίζετε η ώρα, η ημερομηνία και ο συναγερμός στο RTC;

- Κατεβάστε στην πλακέτα τον κώδικα με τις αλλαγές σας και παρατηρήστε εάν οι αλλαγές σας προγραμματίστηκαν κανονικά.
- Πειραματιστείτε με την συχνότητα του Periodic Interrupt δοκιμάζοντας και τις 4 διαφορετικές συχνότητες.
- από το παράθυρο Real Time Clock (RTC) στην Debug λειτουργία του uVision4 μπορούμε να παρακολουθούμε τους καταχωρητές της μονάδας RTC κατά τη διάρκεια εκτέλεσης του προγράμματος μας.

The screenshot shows a window titled "Real Time Clock (RTC)" with a close button. It contains four main sections: Control, Status, Time & Date, and Alarm Time.

Control Section:

- RTC_CR: 0x00000000
- PISEL: Reserved (dropdown menu)
- Checkboxes: AE, PIE, TE, PWR, C, TIE, TIS, TM, W, AIE.

Status Section:

- RTC_SR: 0x00000000
- Checkboxes: TSF, AF, PISF.

Time & Date Section:

- RTC_DTR: 0x00000000
- RTC_TR: 0x00000000
- CENT YEAR MONTH DATE DAY HOUR MIN SEC: 00 00 - 00 - 00 0 00 : 00 : 00
- RTC_MILR: 0x00000000
- MILLISEC: 000

Alarm Time Section:

- RTC_ATR: 0x00000000
- DATE HOUR MIN SEC: 00 00 00 00

Σχήμα 44: Διάβασμα των RTC καταχωρητών στην Debug λειτουργία

11 ΕΡΓΑΣΤΗΡΙΟ 11

- ♦ **Real Time Clock Μέρος 2^ο**
- ♦ **Alarm και Periodic Interrupts Μέρος 2^ο**

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

ss. **Αρχείο mcbstr9.chm HTML**, Κεφάλαια “Writing Programs”.

tt. **Αρχεία RTC.c, LCD.c και IRQ.c**, μέσα από τον φάκελο **Lab10_RTC.rar**.

uu.**Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για τους καταχωρητές του MCU αλλά και για την μονάδα RTC.

vv. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 4, παράγραφος 9. Ρολόι Ξυπνητήρι.

Εισαγωγή

Στο εργαστήριο αυτό θα αναπτύξουμε το προηγούμενο παράδειγμα του ψηφιακού ρολογιού φτιάχνοντας νέες συναρτήσεις για την ρύθμιση του Alarm αλλά και της ημερομηνίας.

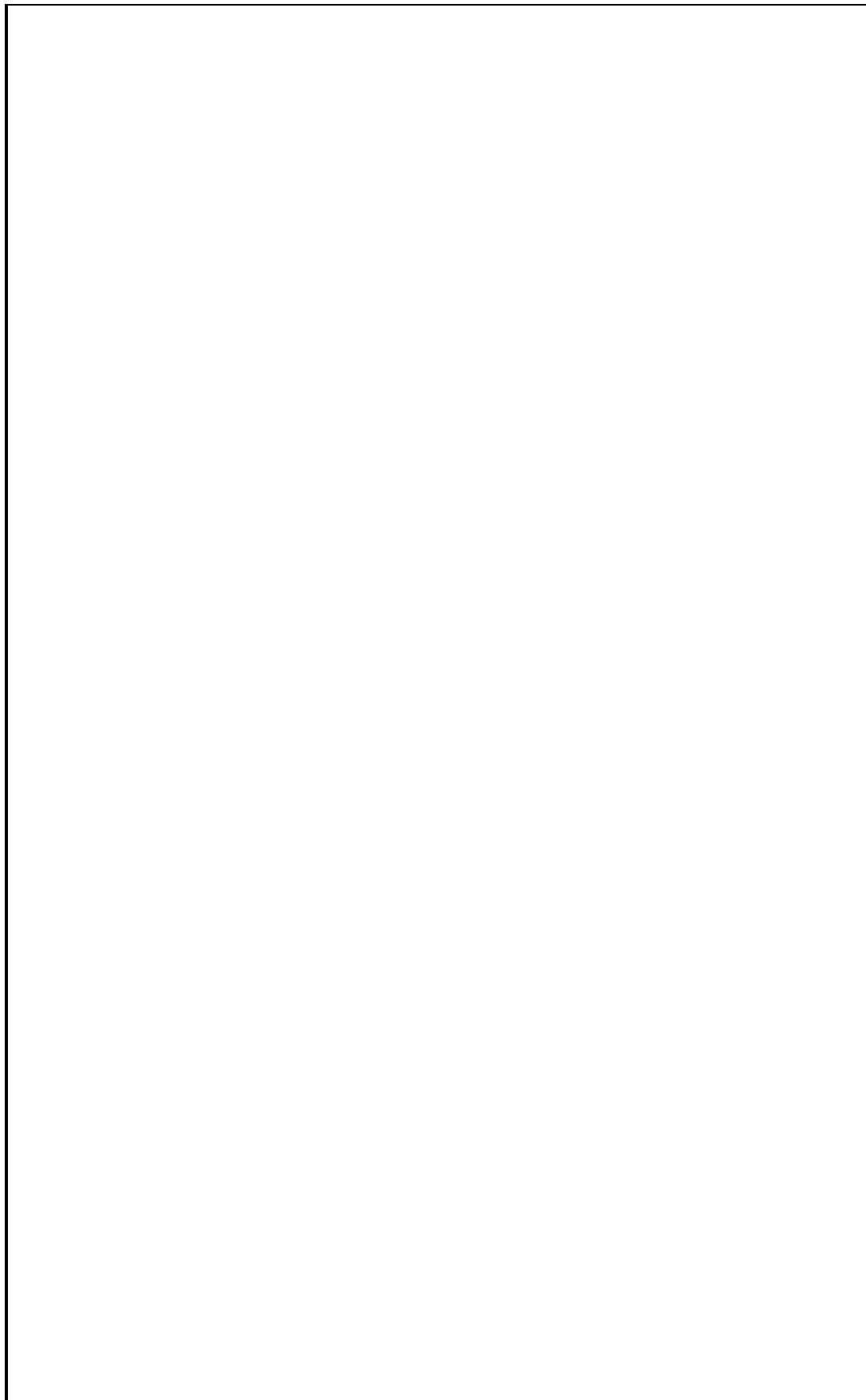
11.1 Προγραμματισμός Alarm

Χρησιμοποιώντας τις παρακάτω σταθερές, όπως αντιστοίχως και με την ώρα, και δημιουργώντας νέες συναρτήσεις προγραμματίστε την ώρα και την ημερομηνία του Alarm.

```
#define ALARM_SECONDS  
#define ALARM_MINUTES  
#define ALARM_HOUR  
#define ALARM_DAY
```

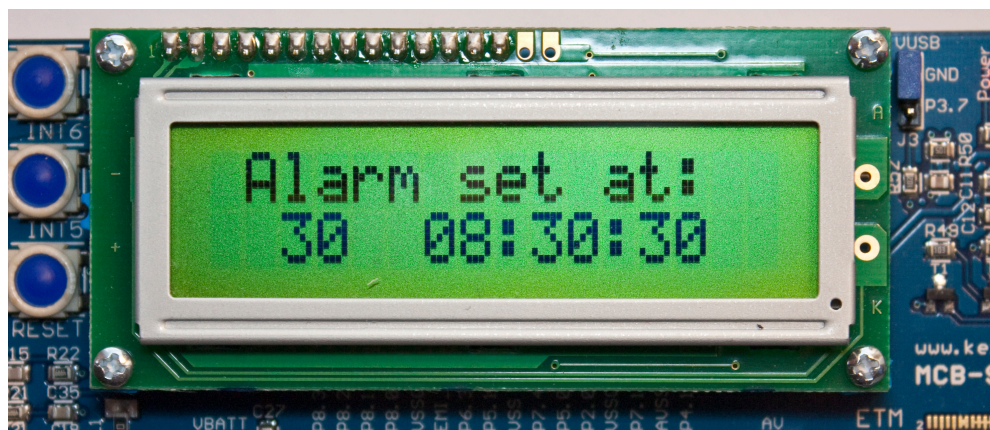
Ερώτημα 76

Εδώ συμπληρώστε τον επιπλέον κώδικα που χρειάζεται για την υλοποίηση.



11.2 Απεικόνιση του Alarm στο LCD

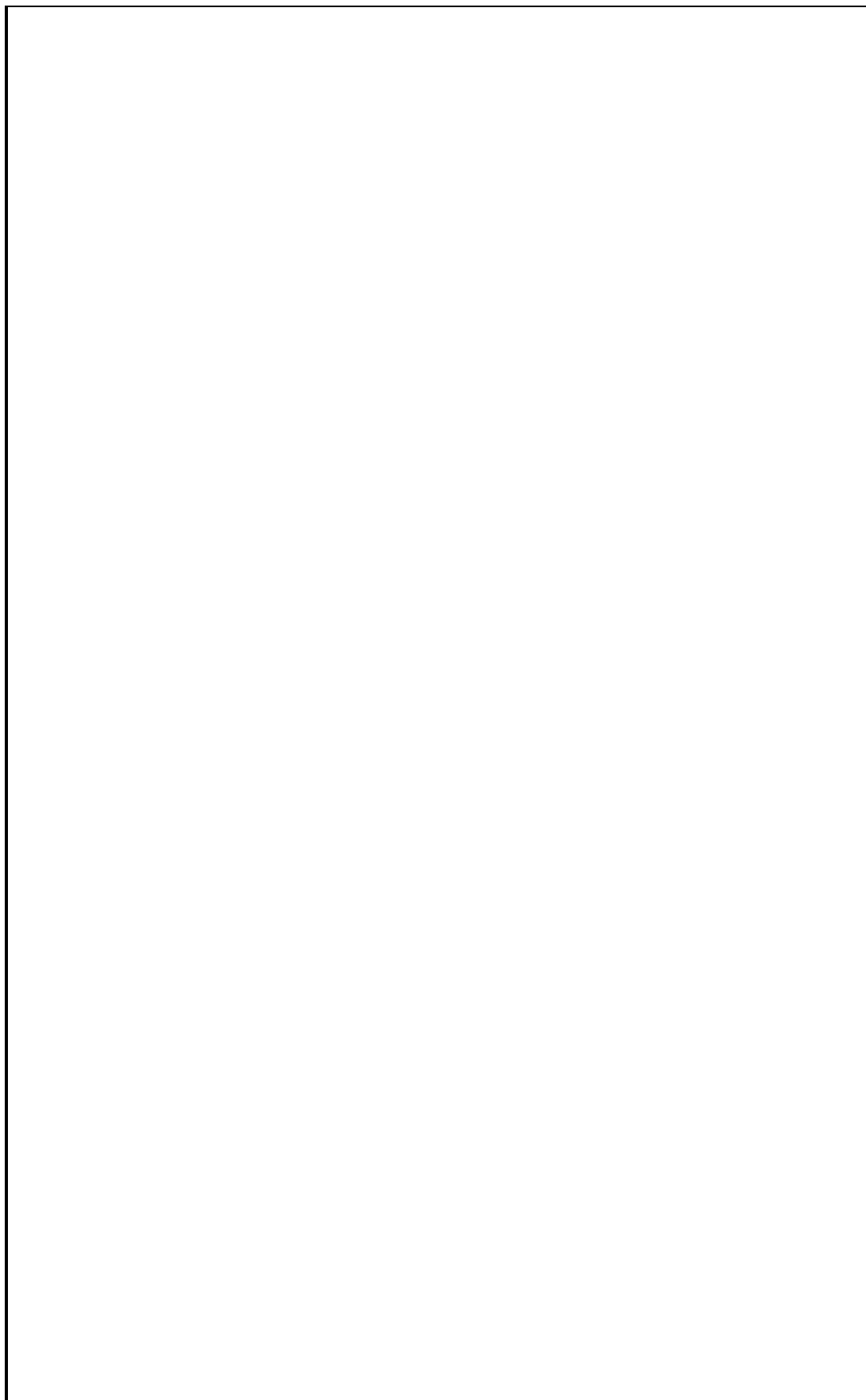
Επιπλέον προσθέστε λογική η οποία θα χρησιμοποιεί το πλήκτρο INT5 ως απλό GPIO (όχι ως WIU Interrupt). Πατώντας το και για όσο χρόνο το κρατάτε πατημένο απεικονίστε στο LCD της ρύθμισης του Alarm όπως φαίνεται στην παρακάτω φωτογραφία. Οι ρυθμίσεις του Alarm πρέπει να διαβάζονται από τον αντίστοιχο καταχωρητή στην μονάδα RTC.

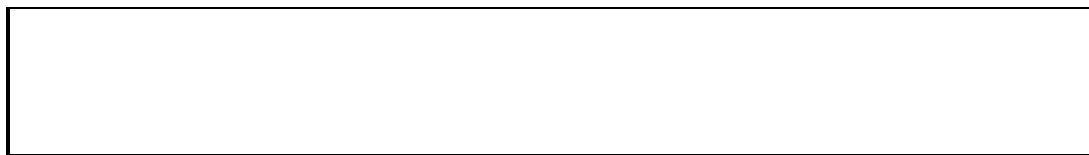


Σχήμα 45: Απεικόνιση Alarm ενώ το πλήκτρο INT5 είναι πατημένο

Ερώτημα 77

Εδώ συμπληρώστε τον επιπλέον κώδικα που χρειάζεται για την υλοποίηση του πατήματος του πλήκτρου και ένδειξη του Alarm.





12 ΕΡΓΑΣΤΗΡΙΟ 12

- ♦ Real Time Clock Μέρος 3^ο
- ♦ Alarm και Periodic Interrupts Μέρος 3^ο

Προϋποθέσεις

Το εργαστήριο αυτό προϋποθέτει το διάβασμα και χρήση των εξής:

ww. **Αρχείο mcbstr9.chm HTML**, Κεφάλαια “Writing Programs”.

xx. **Αρχεία RTC.c και IRQ.c**, μέσα από τον φάκελο **Lab10_RTC.rar**.

yy. **Αρχείο STR91xFA_Reference_Manual.pdf**. Θα το χρησιμοποιούμε ως αναφορά για τους καταχωρητές του MCU αλλά και για την μονάδα RTC.

zz. **Βιβλίο Θεωρίας Wayne Wolf, “Οι Υπολογιστές ως Συστατικά Στοιχεία”**. Κεφάλαιο 4, παράγραφος 9. Ρολόι Ξυπνητήρι.

Εισαγωγή

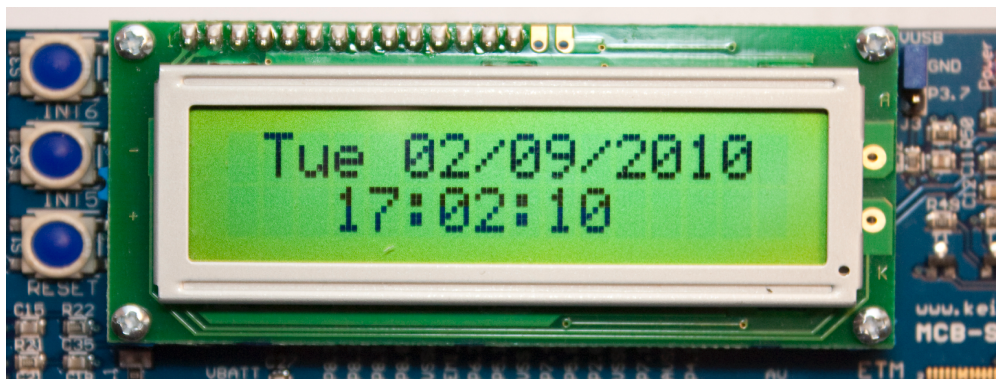
Στο εργαστήριο αυτό θα αναπτύξουμε το προηγούμενο παράδειγμα του ψηφιακού ρολογιού φτιάχνοντας νέες συναρτήσεις και λογική για την ρύθμιση και απεικόνιση της ημερομηνίας στο LCD.

12.1 Προγραμματισμός και απεικόνιση ημερομηνίας

Χρησιμοποιώντας τις παρακάτω σταθερές, όπως και με το Alarm, δημιουργήστε νέες συναρτήσεις ώστε να προγραμματίσετε την ημερομηνία στο RTC.

```
#define DATE  
#define MONTH  
#define YEAR  
#define CENTURY  
#define WEEKDAY
```

Παράλληλα απεικονίστε την πληροφορία της ημερομηνίας στην πρώτη γραμμή του LCD όπως φαίνεται στην παρακάτω φωτογραφία. Στην δεύτερη γραμμή του LCD παραμένει η απεικόνιση της ώρας.



Σχήμα 46: Απεικόνιση της ώρας και της ημερομηνίας στο LCD

Ερώτημα 78

Εδώ συμπληρώστε τον επιπλέον κώδικα που χρειάζεται για την υλοποίηση του προγραμματισμού και απεικόνιση της ημερομηνίας.

13 ΕΡΓΑΣΤΗΡΙΟ 13

◆ RENESAS CK-RA6M5

Εισαγωγή

Σε αυτό το μάθημα θα έρθουμε σε επαφή με την αναπτυξιακή πλακέτα CK-RA6M5 της RENESAS, η οποία φαίνεται στο παρακάτω σχήμα (Σχήμα 47). Η συγκεκριμένη πλακέτα επιτρέπει στους χρήστες να συνδέονται με ασφάλεια στο cloud και να εξερευνούν τις δυνατότητες της ομάδας MCU (Micro Controller Unit) και υπηρεσιών cloud RA6M5, που βασίζεται στο Cortex®-M33.



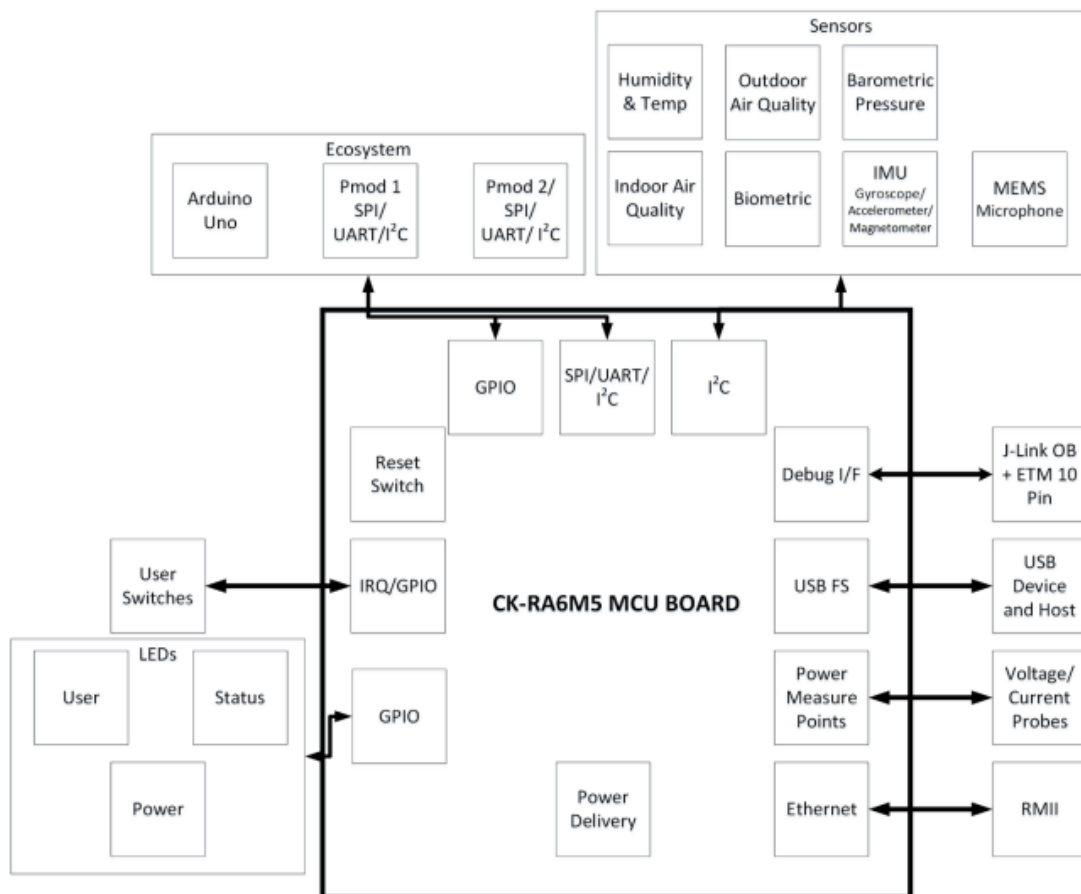
Σχήμα 47: RENESAS CK-RA6M5

Η πλακέτα του αναπτυξιακού αποτελείται από έναν μικροελεγκτή και τα περιφερειακά του όπως φαίνεται στο παρακάτω σχήμα (Σχήμα 48).

Η πλακέτα είναι εξοπλισμένη ως εξής:

- Επεξεργαστή Cortex®-M33 200MHZ
- 2 MB Code Flash, 8 kB data flash, 512 kB SRAM
- 2 Πλήκτρα (User, Reset)
- 6 LED (Red, Blue, Green, RGB, Debug, Power)

- 2 ενώσεις Pmod type 6A and 2A
- 2 ενώσεις Arduino Uno
- Θύρα Ethernet
- Αισθητήρα Θερμοκρασίας / Υγρασίας
- Αισθητήρα ποιότητας αέρος Εσωτερικού / Εξωτερικού χώρου
- Βιομετρικό αισθητήρας
- Βαρομετρικό αισθητήρας πίεσης
- Γυροσκόπιο, Μαγνητόμετρο, Επιταχυνσιόμετρο
- Μικρόφωνο



Σχήμα 48: Σχηματικό διάγραμμα της πλακέτας

Το MCU υποστηρίζει έναν γραμμικό χώρο διευθύνσεων 4 GB που κυμαίνεται από την διεύθυνση 0x0000_0000 έως 0xFFFF_FFFF που μπορεί να περιέχει τόσο το πρόγραμμα και τα δεδομένα όπως φαίνεται στο Σχήμα 49.

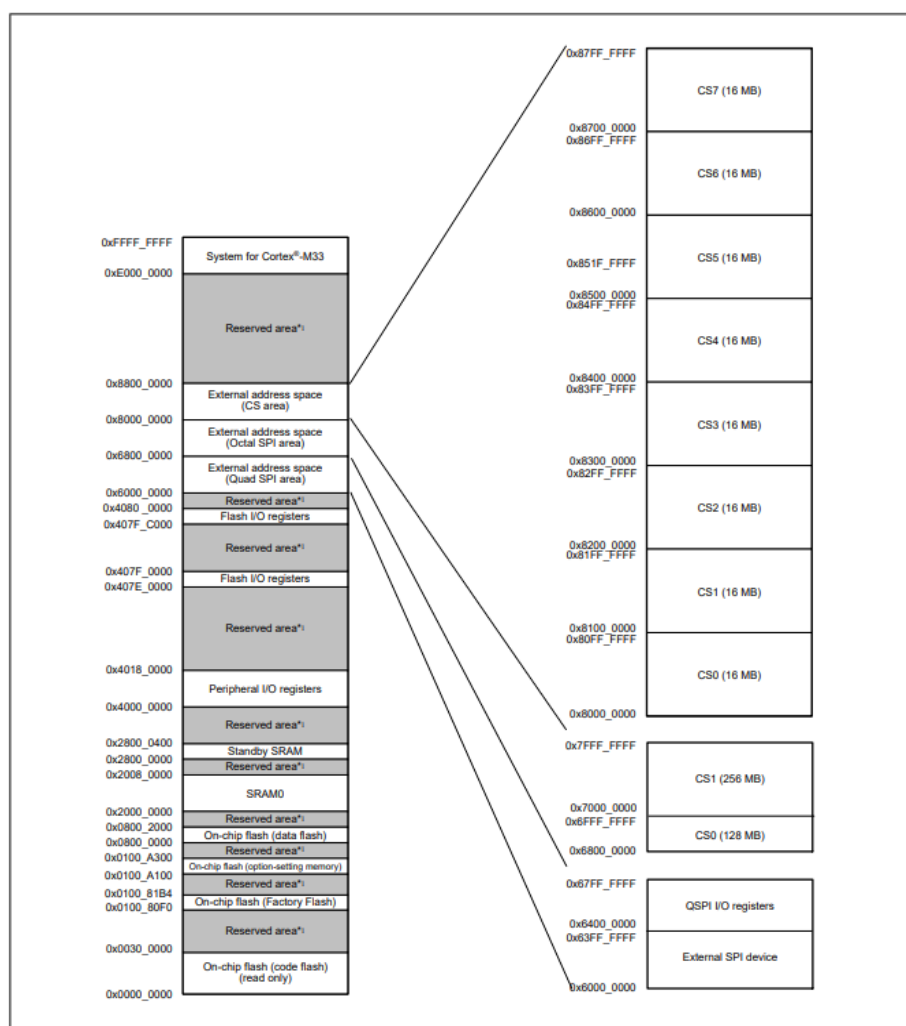
Ο χώρος εξωτερικών διευθύνσεων χωρίζεται σε περιοχές:

- CS (Chip Select, CS0 έως CS7),

- σε οχτώ περιοχές SPI (Serial Peripheral Interface, CS0 και CS1) και
- τέσσερις περιοχές SPI.

Οι περιοχές CS (CS0 έως CS7) αντιστοιχούν η καθεμία στο σήμα CSn που εξάγεται από έναν ακροδέκτη CSn ($n = 0$ έως 7). Οι δύο οκταπλές περιοχές SPI (CS0 και CS1) αντιστοιχούν η καθεμία στην έξοδο σήματος OM_CSn από έναν ακροδέκτη OM_CSn ($n = 0, 1$). Η τετραγωνική περιοχή SPI διαιρείται σε δύο περιοχές, τους καταχωρητές εισόδου/εξόδου QSPI και τον χώρο εξωτερικής συσκευής SPI.

Το Σχήμα 49 δείχνει τις περιοχές διευθύνσεων που σχετίζονται με τις επιμέρους περιοχές CS (CS0 έως CS7), την περιοχή Octal SPI (CS0 και CS1) και την περιοχή τετραπλού SPI.



Σχήμα 49: Χάρτης Μνήμης του CK-RA6M5

Ερώτημα 79

Με βάση το Σχήμα 49 να αναφαίρετε σε ποιες διευθύνσεις αποθηκεύεται ο κώδικας της πλακέτας

Ερώτημα 80

Με βάση το User's Manual βρείτε σε ποια port συνδέονται τα παρακάτω:

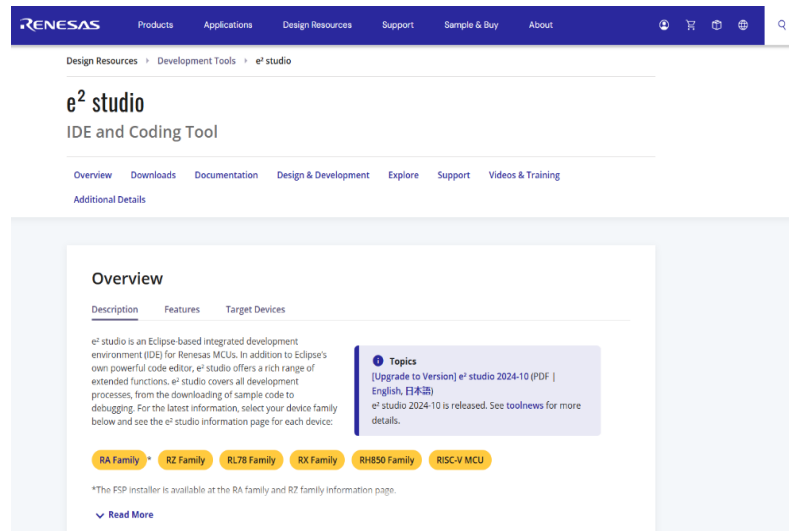
LED2 RGB	
LED5 Yellow	
Reset Switch	
LED6 White	

Για επιπλέον πληροφορίες ανατρέξτε στην [ιστοσελίδα](#) του συγκεκριμένου αισθητήρα.

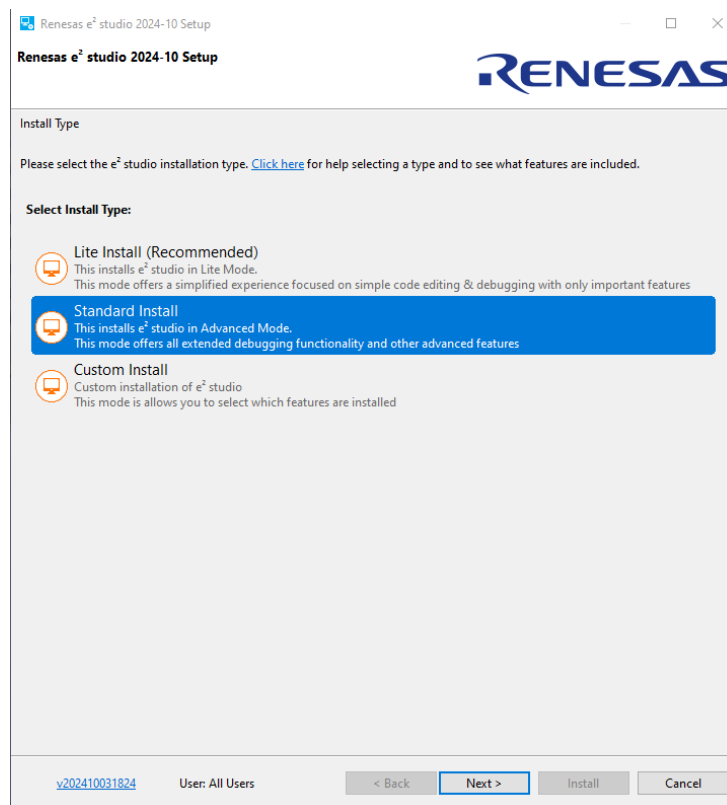
Εγκατάσταση e² Studio

Για να μπορέσουμε να προγραμματίσουμε την πλακέτα μας, χρειάζεται να εγκαταστήσουμε την εφαρμογή της RENESAS e² Studio. Ακολουθήστε τα παρακάτω βήματα για την εγκατάσταση της εφαρμογής.

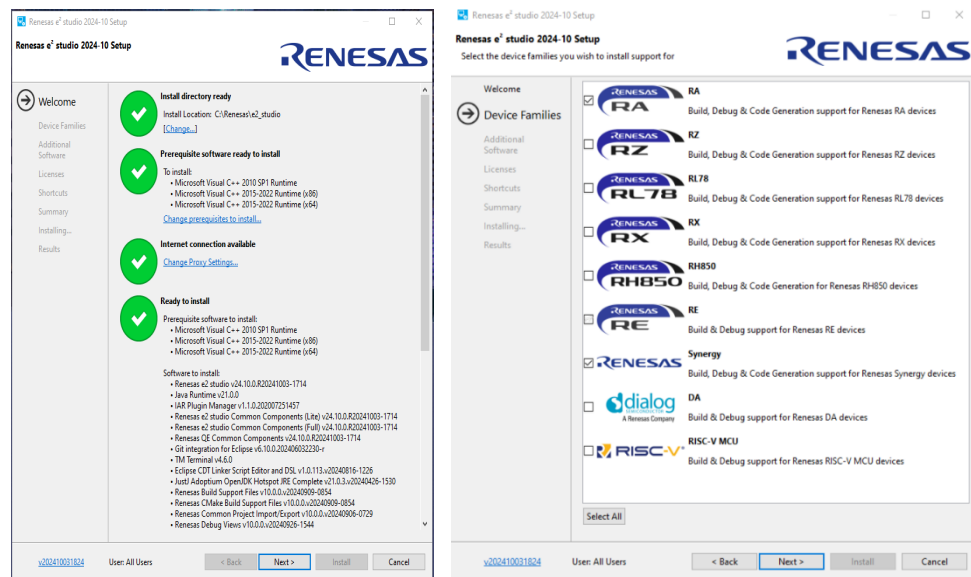
- Πηγαίνετε στην ιστοσελίδα της [RENESAS](#) όπου μπορείτε να κατεβάσετε την εφαρμογή.



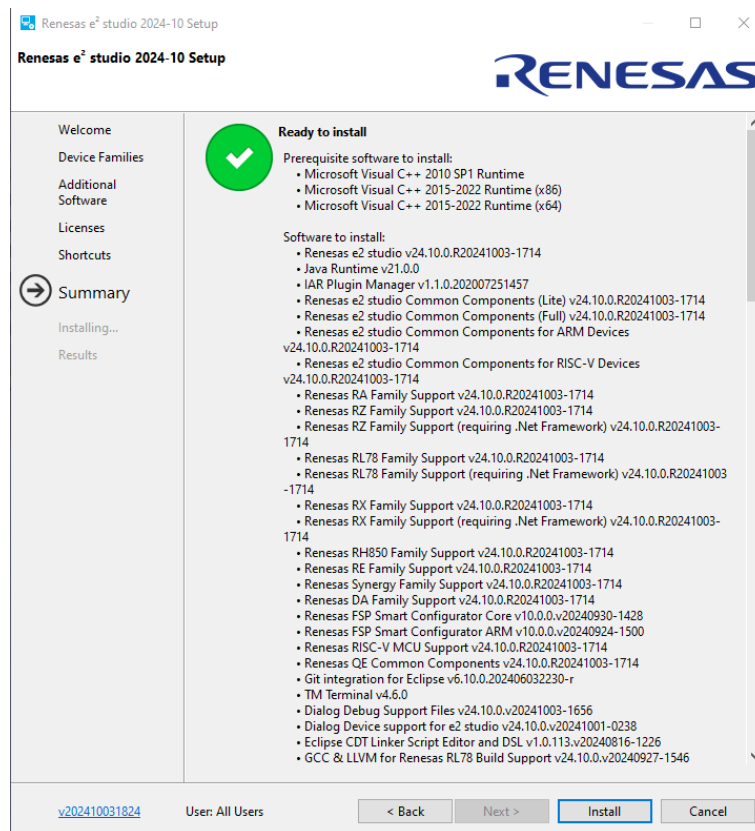
- Αφού γίνει η λήψη της εφαρμογής συνεχίζετε στην εγκατάσταση της.



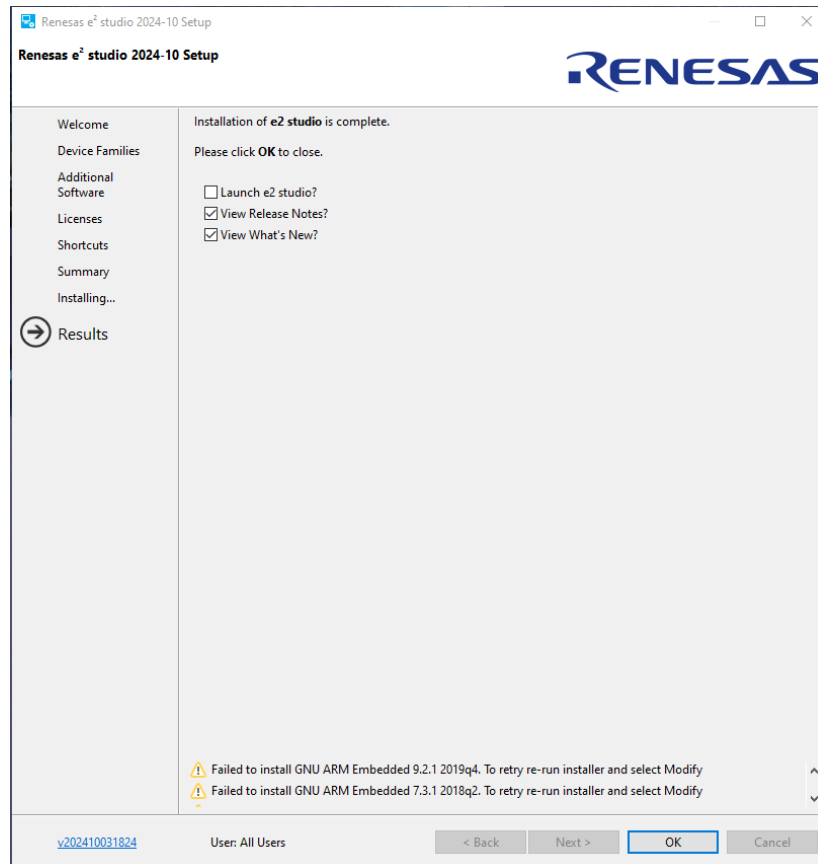
- Αφού γίνει έλεγχος από το σύστημα συνεχίζετε στην εγκατάσταση των πακέτων που θα χρησιμοποιήσετε. Αυτά είναι τα **RA** και **SYNERGY**.



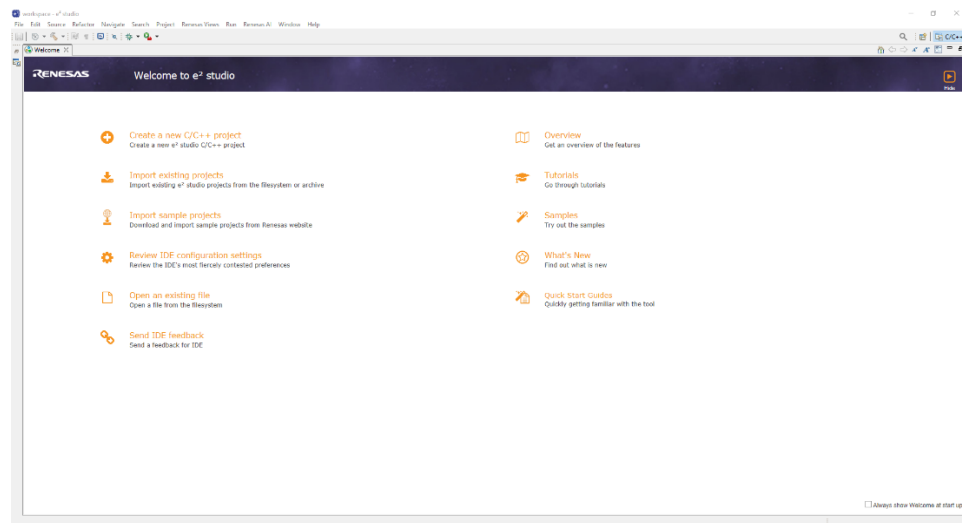
- Όταν το πρόγραμμα κατεβάσει τα πακέτα που χρειάζεται, πατάτε **INSTALL** και προχωράτε στην εγκατάσταση της εφαρμογής.



- Όταν τελειώσει η εγκατάσταση της εφαρμογής, θα εμφανιστεί το συγκεκριμένο παράθυρο.



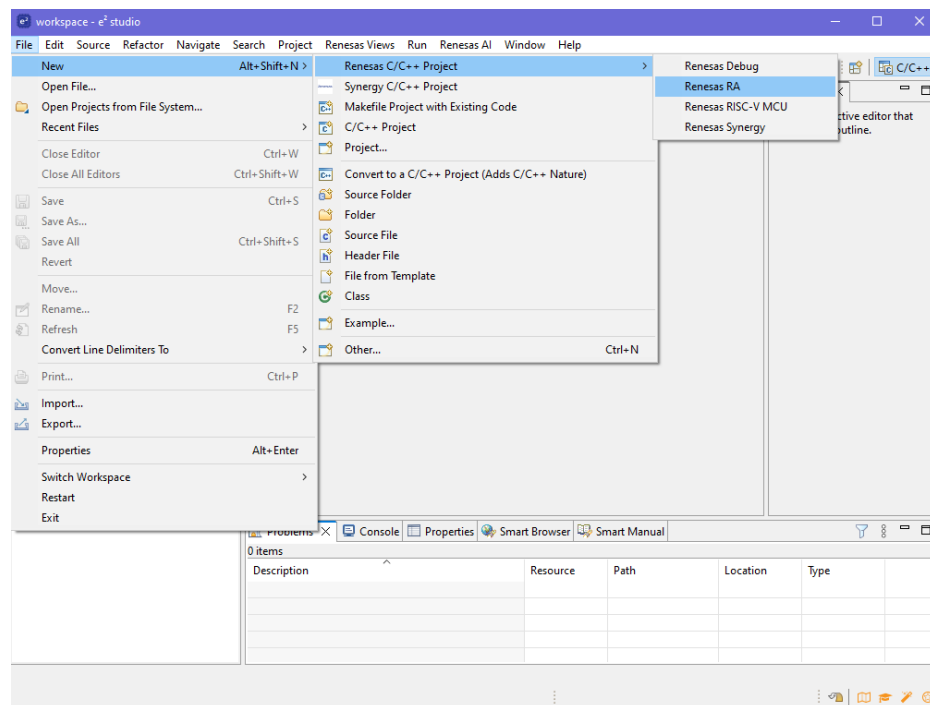
- Ανοίγουμε την εφαρμογή e² Studio για να ξεκινήσουμε το project μας.



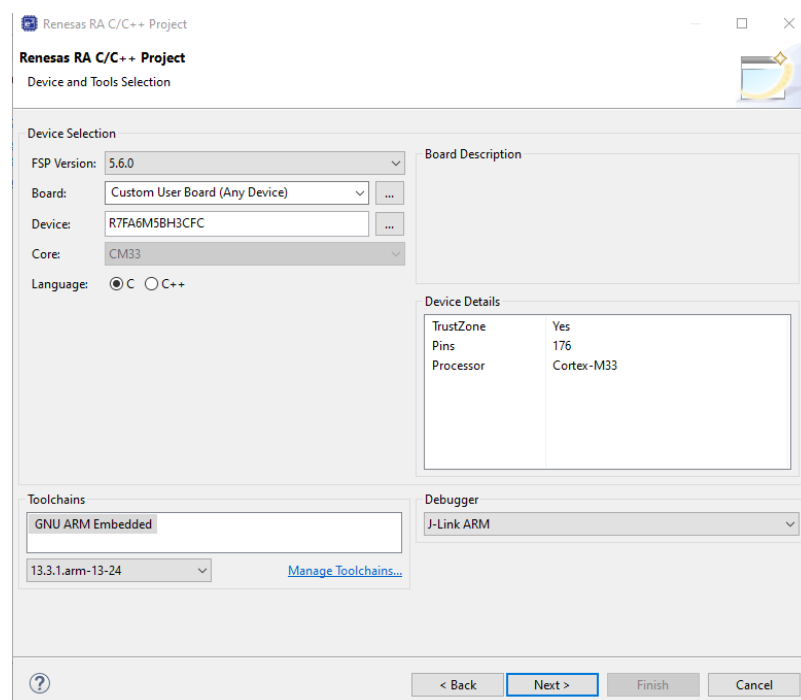
Περιήγηση εφαρμογής e² Studio

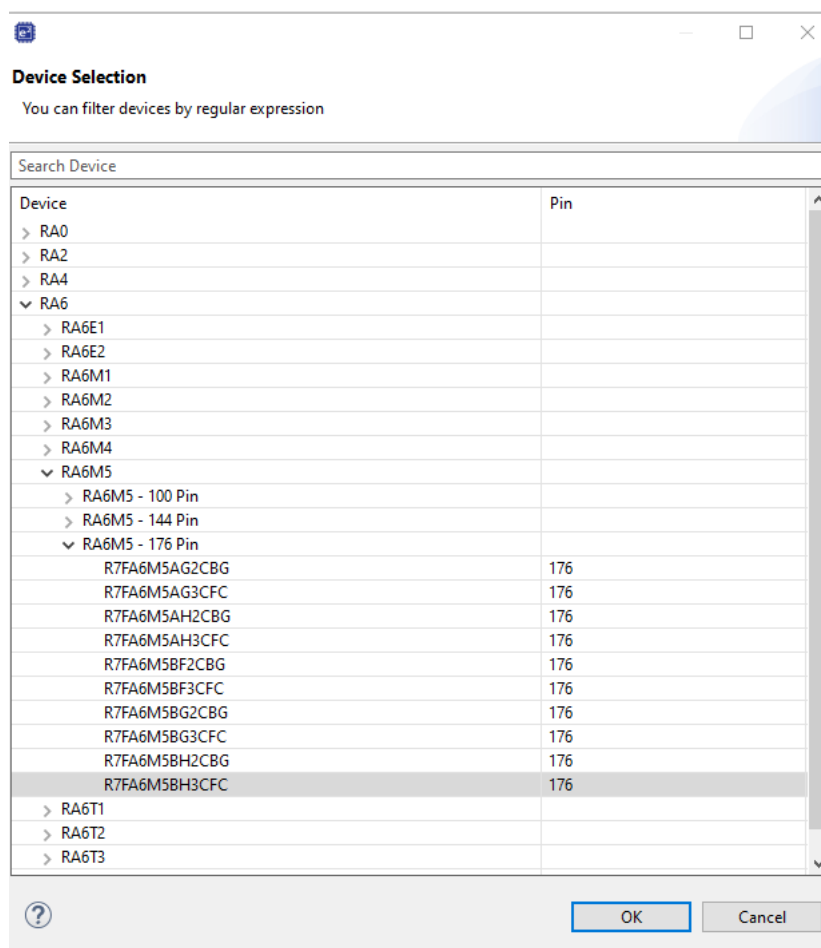
Αφού έγινε η εγκατάσταση της εφαρμογής ας συνεχίσουμε με τη δημιουργία του πρώτου μας project.

- Αρχικά πηγαίνετε στην επιλογή: **File** → **New** → **Renesas C/C++ Project** → **Renesas RA**

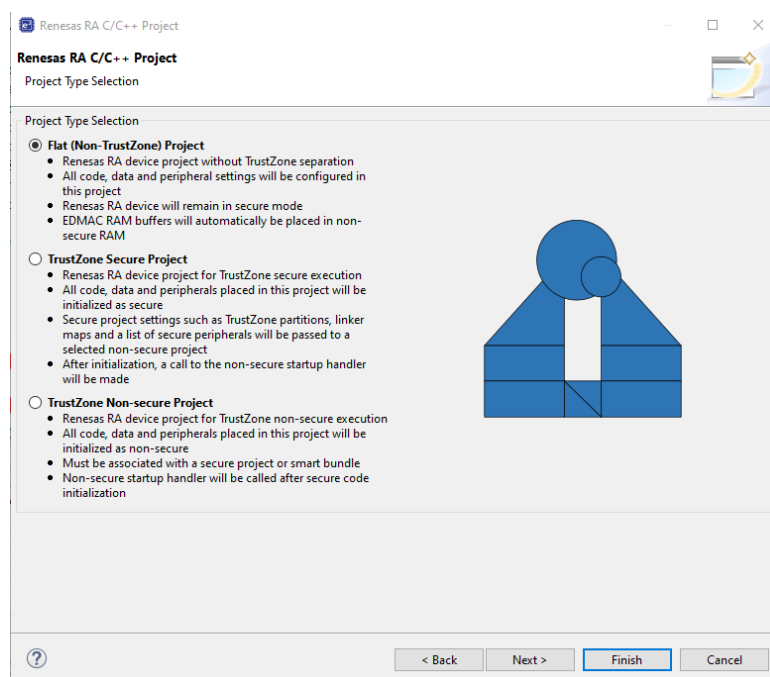


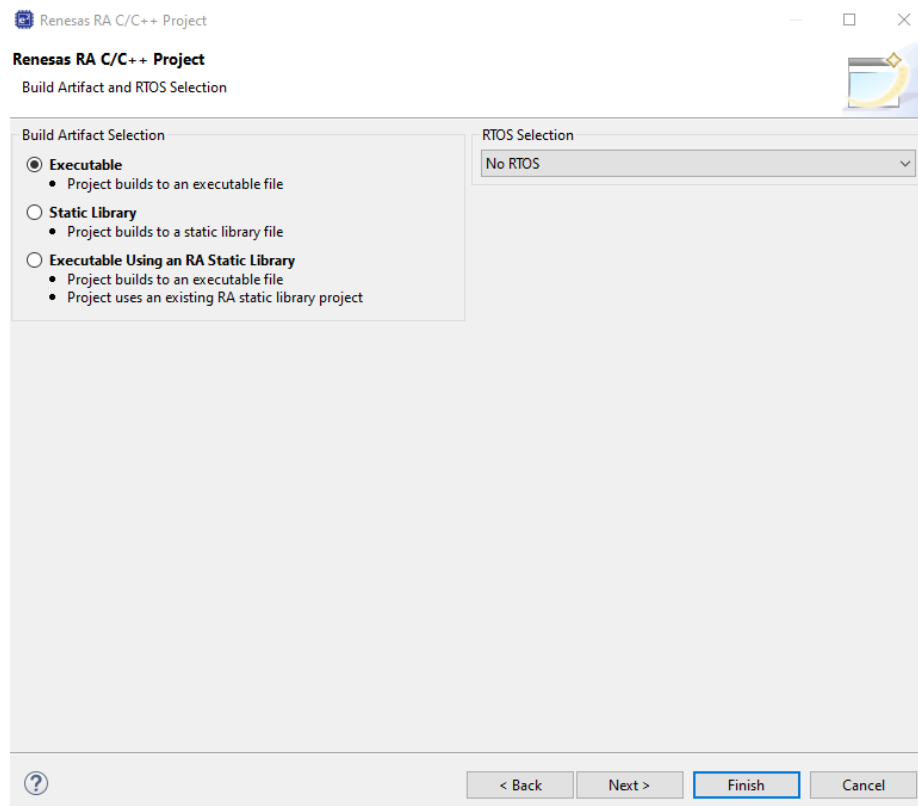
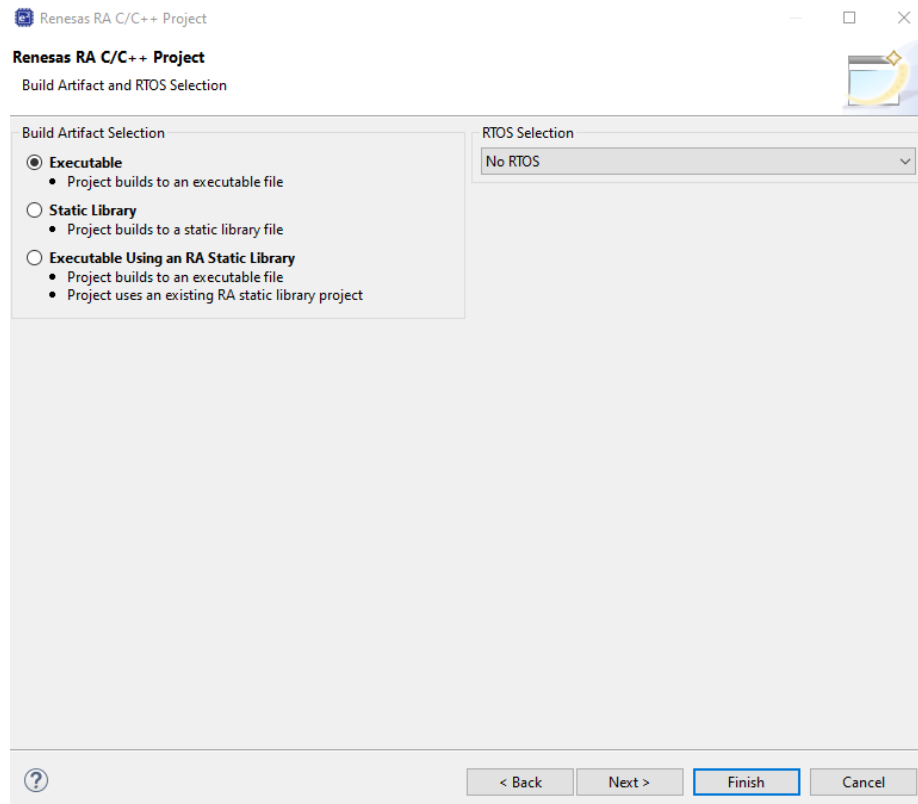
- Στη συνέχεια αφού δώσετε ένα όνομα στο Project σας συνεχίζετε στην επιλογή της πλακέτας σας. Πιο συγκεκριμένα η πλακέτα η οποία θα χρησιμοποιήσετε αποτελεί το μοντέλο **R7FA6M5BH3CFC**.



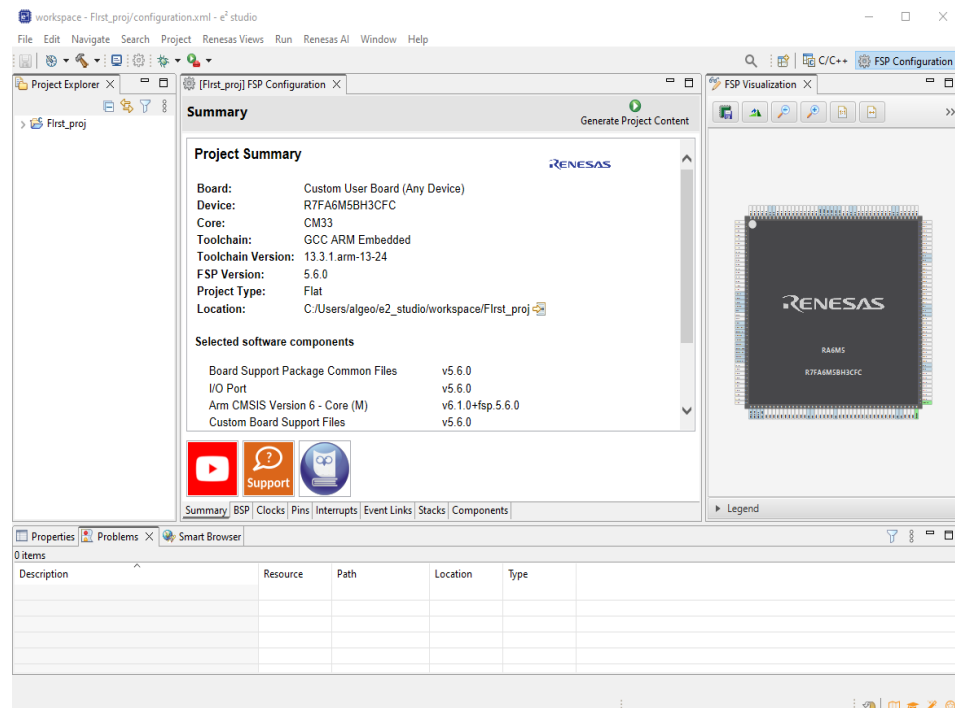


- Αφού αφήσετε τις επόμενες ρυθμίσεις ίδιες, προχωράτε στη δημιουργία του project σας.





- Μετά τη δημιουργία του project θα εμφανιστεί το συγκεκριμένο παράθυρο.



- Για να μπορέσετε να γράψετε τον δικό σας κώδικα επιλέγετε:
Project Explorer → (Όνομα Project) → src → hal_entry.c

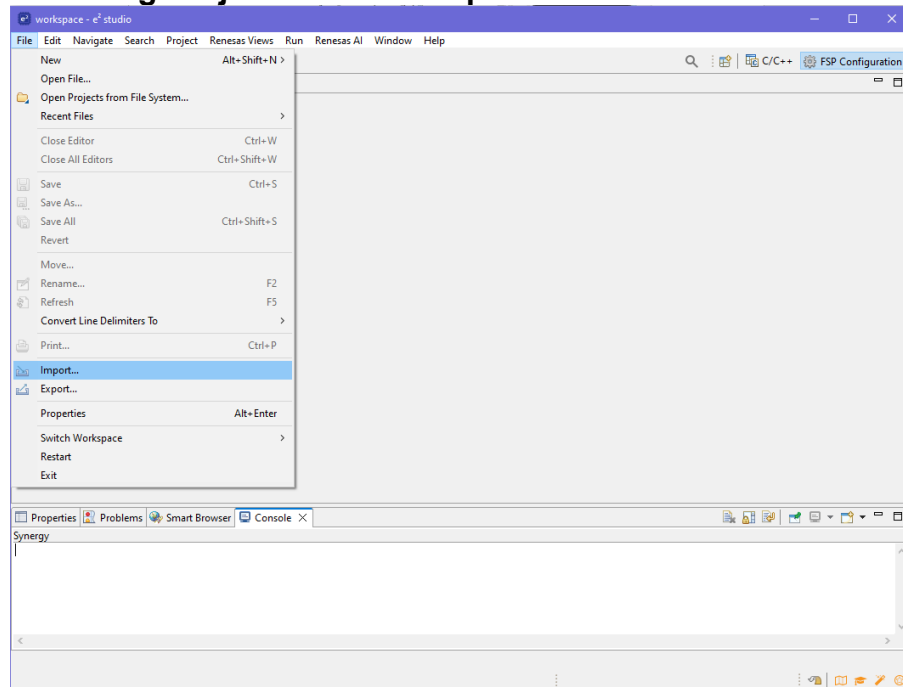
14 ΕΡΓΑΣΤΗΡΙΟ 14

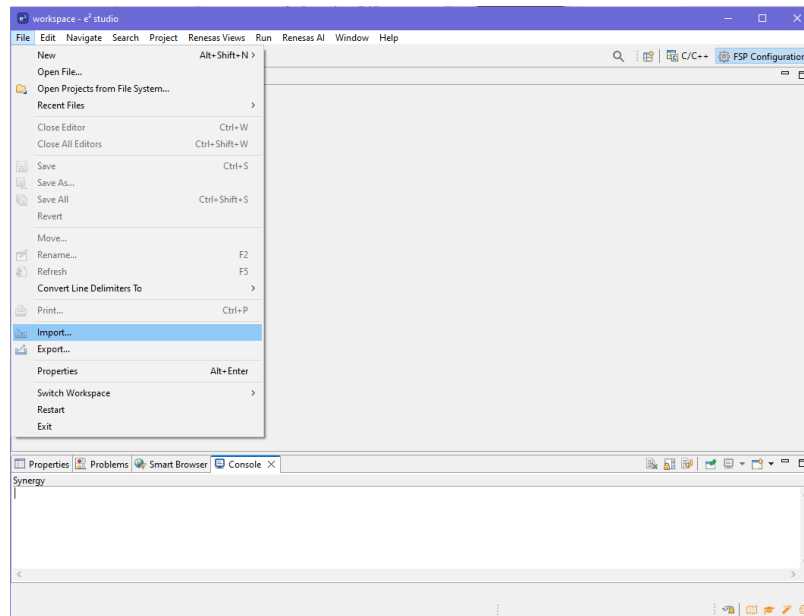
♦ Παράδειγμα Blinky

Σε αυτό το εργαστήριο θα ασχοληθούμε με το παράδειγμα Blinky. Πως μπορούμε να αξιοποιήσουμε τα leds της πλακέτας και να πειραματιστούμε με τα υπόλοιπα περιφερικά της πλακέτας.

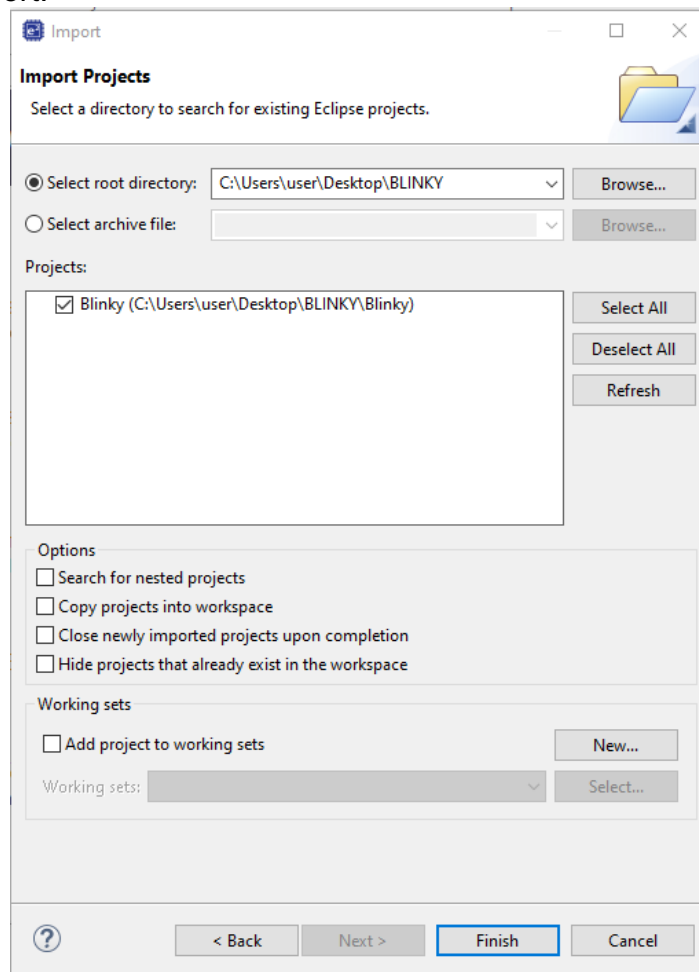
Παρακάτω θα αναλύσουμε τα βήματα που χρειάζονται για να εισάγουμε ένα έτοιμο project, στην προκείμενη περίπτωση, το Blinky. Πώς γίνεται η διαδικασία του Build – Debug και πολλά ακόμα.

- Για να εισάγουμε το Blinky πηγαίνουμε στο: **File → Import → General → Existing Projects into Workspace**





- Στην επόμενη καρτέλα πατάμε Browse και προσπαθούμε να βρούμε την διεύθυνση του αρχείου του project.
- Έπειτα επιλέγουμε το project από το παρακάτω menu και προχωράμε στο import.



- Αφού η εφαρμογή έχει αναγνωρίσει και φορτώσει το Project μας, πατάμε Finish και περιμένουμε να σεταριστεί.

Αφού εντοπίσετε τον κώδικα, ο οποίος βρίσκεται στο αρχείο `hal_data.h` τοποθετημένο στον φάκελο `src`, μπορείτε να προχωρήσετε στην ανάλυση και στην κατανόηση του κώδικα.

Αφού τελειώσετε, απαντήστε στις παρακάτω ερωτήσεις:

Ερώτημα 81

Βρείτε σε ποιο σημείο του κώδικα ανάβουν και σβήνουν τα `leds` και να αναφέρετε πως γίνεται αυτό

Ερώτημα 82

Βρείτε το όνομα και τις διευθύνσεις των `leds`

Ερώτημα 83

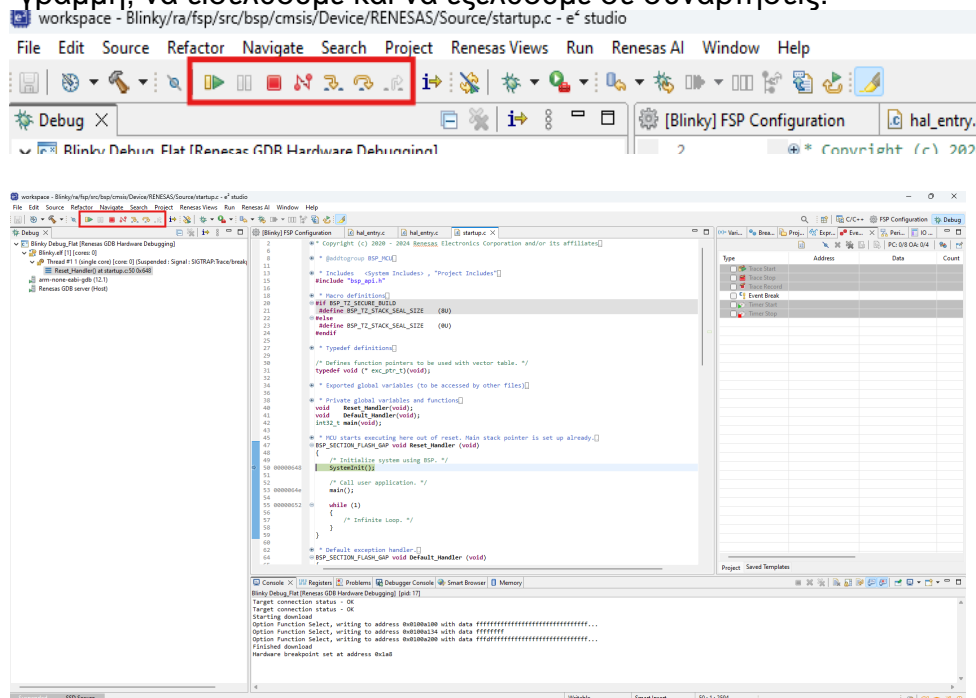
Να αναφέρεται με ποια σειρά θα ανάψουν τα `led` και γιατί

Ερώτημα 84

Με ποιος τρόπο μπορείτε να αλλάξετε την ταχύτητα με την οποία αναβοσβήνουν τα leds

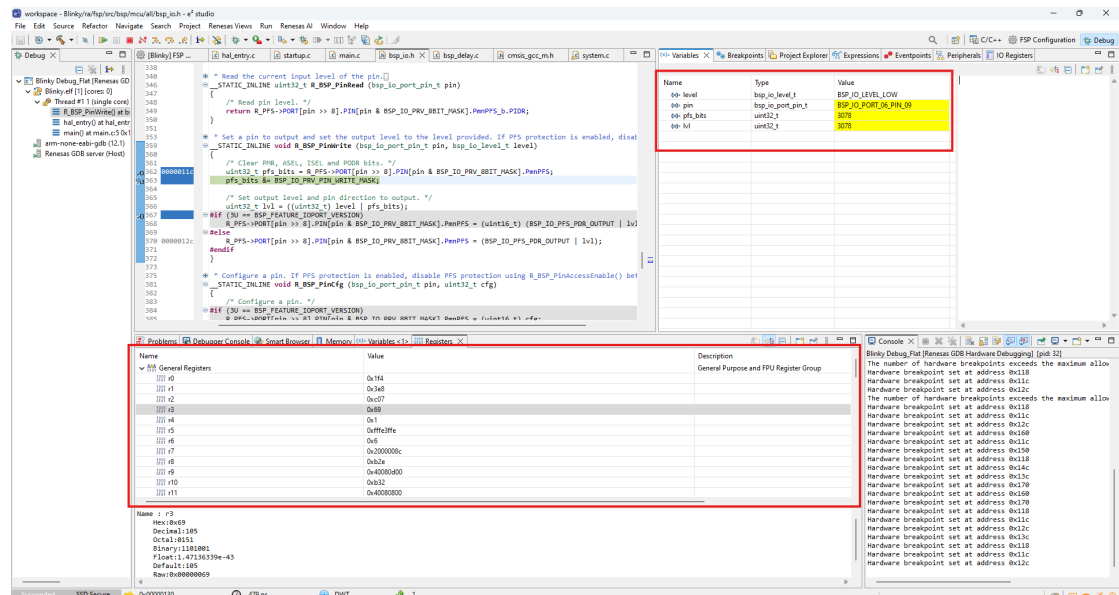
Στη συνέχεια θα προχωρήσουμε στην εκτέλεση του κώδικα πάνω στην πλακέτα:

- Αρχικά κάνουμε Build.
- Το πρόγραμμα μας θα πρέπει να μην έχει κανένα απολύτως error.
- Συνεχίζουμε στη διαδικασία του Debug. **Debug → Resume (F8)**. Για να περάσουμε το πρόγραμμά μας στην πλακέτα, και να εκτελεστεί.
- Χρησιμοποιώντας τα κουμπιά που βρίσκονται δίπλα στο Resume (F8) και πιο συγκεκριμένα το Step Into (F5), το Step Over (F6) και Step Return (F7) μπορούμε να καθοδηγηθούμε μέσα στον κώδικα γραμμή-γραμμή, να εισέλθουμε και να εξέλθουμε σε συναρτήσεις.

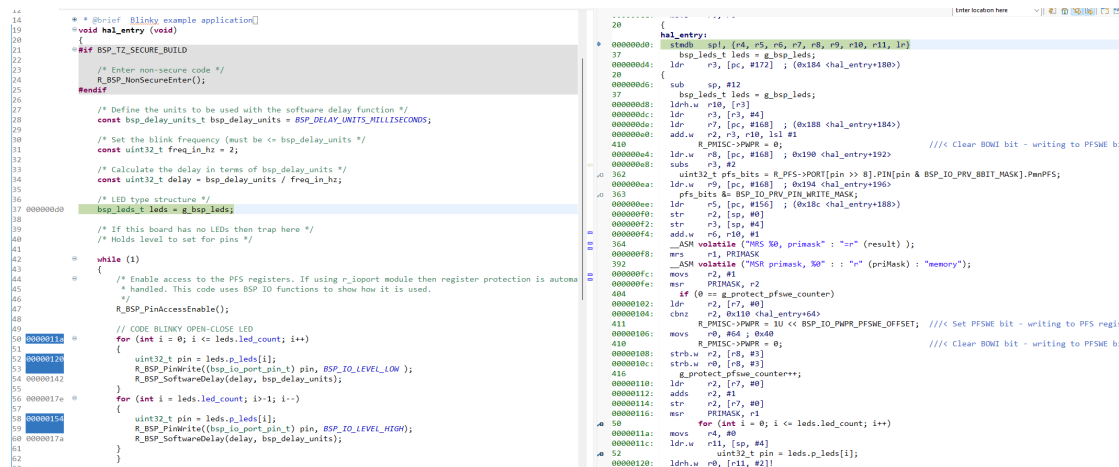


Στη συνέχεια επιστρέφουμε στην καρτέλα hal_entry.c και τοποθετούμε breakpoint στις παρακάτω γραμμές του κώδικα : γραμμή 52, 53, 58, 59. Στη συνέχεια πηγαίνουμε στην καρτέλα bsp_io.h και τοποθετούμε breakpoint στις

παρακάτω γραμμές του κώδικα : γραμμή 362, 363, 367. Με τη βοήθεια των παράθυρων Variables, Registers μπορούμε να παρατηρήσουμε τις αλλαγές στις παραμέτρους του κώδικα.



Το Debug mode μπορεί να μας παρέχει και άλλες χρήσιμες πληροφορίες. Πατώντας την επιλογή **Window → Show view → Disassembly**, εμφανίζεται ένα παράθυρο το οποίο μας δείχνει την μετατροπή του κώδικα σε γλώσσα υπολογιστή Assembly καθώς και ποιοι καταχωρητές χρησιμοποιούνται για την εκτέλεση των εντολών.



Πιο συγκεκριμένα στο παράθυρο αυτό, μπορούμε να δούμε τις εντολές του προγράμματος και τις αντίστοιχες εντολές που χρειάζεται για να μετατραπεί σε γλώσσα Assembly όπως και τον program counter που ακολουθεί το

σύστημα και στις δυο γλώσσες εντολών. Μερικά παραδείγματα κώδικα σε Assembly μπορούμε να δούμε παρακάτω.

Μπορούμε να παρατηρήσουμε ότι ο assembler χωρίζει την εντολή:

For (uint32_t i = 0; i <= leds.led_count; i++)

σε 3 επιμέρους κομμάτια. Πρώτο κομμάτι είναι το: **movs r4, #0**, όπου στον καταχωρητή **r4** αποθηκεύεται η τιμή του **i** σε όλη τη διάρκεια του προγράμματος. Στη συνέχεια, στο δεύτερο κομμάτι πραγματοποιείται η σύγκριση του **i** με **led_count** όπως φαίνεται στην εντολή **cmp r4, r6**. Ο καταχωρητής **r6** αποθηκεύει την τιμή του **leds.led_count** σε όλη τη διάρκεια του προγράμματος. Τέλος στο τρίτο κομμάτι πραγματοποιείται η αύξηση του **i** μέσω της εντολής **adds r4, #1**.

```

50      for (uint32_t i = 0; i <= leds.led_count; i++)
00000142:  cmp     r4, r6

50      for (uint32_t i = 0; i <= leds.led_count; i++)
0000013c:  adds    r4, #1

```

Αντίστοιχα στην δεύτερη for μπορούμε να δούμε την ίδια διαδικασία, με τις ίδιες εντολές, με την διαφορά στο τέλος αντί για αύξηση της μεταβλητής **i** γίνεται μείωση αυτής.

```

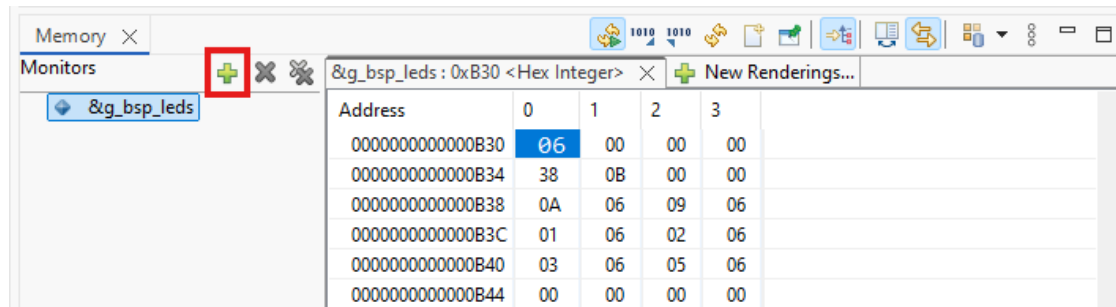
56      for (int i = leds.led_count; i>-1; i--)
00000174:  subs    r4, #1

```

Ερώτημα 85

Με το ίδιο σκεπτικό προσπαθήστε να εξηγήσετε πως μεταφράζεται σε γλώσσα Assembly η εντολή SoftwareDelay (περιγραφικά)

Άλλο ένα χρήσιμο παράθυρο που μπορούμε να συναντήσουμε στο Debug Mode είναι το Memory. Πατώντας την επιλογή **Window → Show View → Memory** εμφανίζεται το παράθυρο με τις διευθύνσεις μνήμης.



Πατώντας το πράσινο κουμπί μας δίνεται η ευκαιρία να προσθέσουμε συναρτήσεις ώστε να μπορούμε να βλέπουμε πιο εύκολα πότε ανανεώνονται οι τιμές της συγκεκριμένης διεύθυνσης.

Για να μπορέσουμε να καταλάβουμε καλύτερα την μνήμη και πως λειτουργεί, μπορούμε να τρέξουμε το συγκεκριμένο παράδειγμα:

```
static int func1(int data1, int data2);
```

```
static int func2(int data1, int data2);
```

```
static int func1(int data1, int data2)
```

```
{
```

```
    int result;
```

```
    result = data1 + data2;
```

```
    return result; }
```

```
static int func2(int data1, int data2)
```

```
{
```

```
    int result;
```

```
    result = data1 - data2;
```

```
    return result;}
```

```
int result = 0;
```

```
int z1 = 0;
```

```
int z2 = 0;
```

```

int data1 = 0x656e6552; //κρυμμένη πληροφορία

int data2 = 0x5f736173; // κρυμμένη πληροφορία

void hal_entry(void)
{
    z1 = func1(data1, data2);

    z2 = func2(data1, data2);

    result = z1 - z2;
}

```

Αφού αντιγράψουμε τον κώδικα στην εφαρμογή, προχωράμε στη διαδικασία του Debugging. Στη συνέχεια ανοίγουμε τα παράθυρα τα οποία θα χρειαστούμε για το παράδειγμα που πρόκειται να δούμε. Πηγαίνοντας στην επιλογή **Window → Show View → Memory** και **Window → Show View → Expressions** θα βρούμε τα εργαλεία που χρειαζόμαστε. Αφού προσθέσουμε τις μεταβλητές μας στα συγκεκριμένα παράθυρα προχωράμε στην εκτέλεση του προγράμματος (για τη διαμόρφωση του παραθύρου Memory, επιλέγουμε Format πατώντας δεξί κλικ).

Expression	Type	Value	Address
data1	int	-2056362670	0x20000004
data2	int	-1888263821	0x20000000
z1	int	-168098849	0x20000050
z2	int	1213025625	0x2000004c
result	int	350340805	0x20000054
Add new expression			

Σχήμα 50: Παράθυρο Expressions

Memory					
Monitor &z2 : 0x2000004C <Hex Integer> New Renderings...					
&z1					
&z2					
&result					
Address	0	1	2	3	
000000002000004C	59	4D	4D	48	
0000000020000050	DF	03	FB	F5	
0000000020000054	C5	C6	E1	14	
0000000020000058	E6	C2	E6	1E	
000000002000005C	54	52	4F	50	
0000000020000060	00	00	00	00	
0000000020000064	00	00	00	00	
0000000020000068	00	00	00	00	
000000002000006C	00	00	00	00	

Σχήμα 51: Παράθυρο Memory

Αρχικά παρατηρούμε ότι οι τιμές που μπορεί να λάβει μια μεταβλητή αποτελούνται από 8 bit, δηλαδή από την τιμή 0x00000000 έως 0xFFFFFFFF, αναλόγως με την χρήση προσημασμένων ή μη αριθμών αυτό σημαίνει ότι μπορούμε να λάβουμε τιμές μεταξύ -2147483647 έως 4294967295.

Κάτι άλλο που μπορούμε να παρατηρήσουμε είναι ότι σε περίπτωση overflow το πρόγραμμα διαγράφει το 9^ο bit και κρατάει τα υπόλοιπα bit στην διάθεση του.

Ερώτημα 86

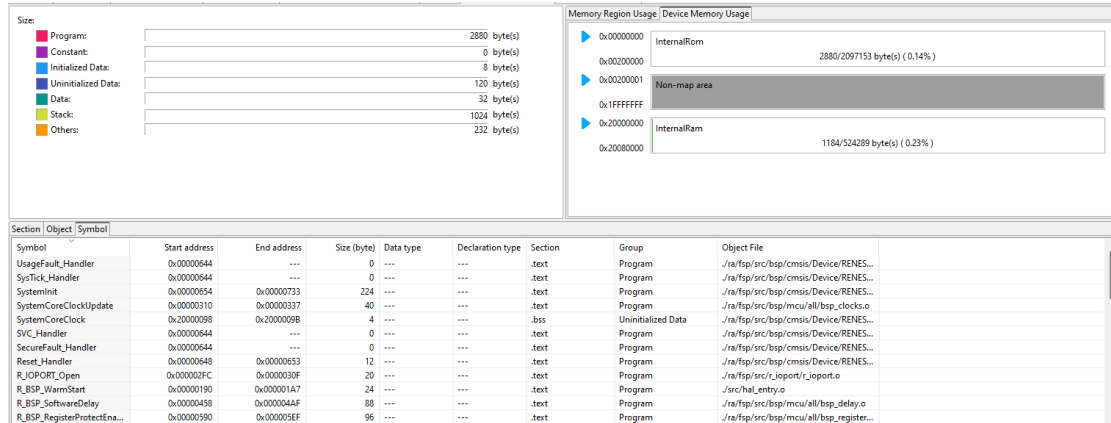
Αφού μελετήσετε και τροποποιήσετε τον κώδικα, να εμφανίσετε τα αποτελέσματα (z1,z2, result) των παρακάτω πράξεων: $\text{data1} * \text{data2}$, $\text{data1}/\text{data2}$, $\text{data1} * 2$, $\text{data2} * 3$, $\text{data1}/4$, $\text{data2}/2$ (να εξηγήσετε το αποτέλεσμα που εμφανίζεται)

Πρέπει να επισημάνουμε ότι μια μεταβλητή μπορεί να κουβαλάει πληροφορίες διαφορετικών ειδών, από αριθμητικές τιμές, σε κείμενο ακόμα και εικόνα. Με βάση αυτή την πληροφορία να απαντήσετε την παρακάτω ερώτηση.

Ερώτημα 87

Να βρείτε τις παρακάτω κρυμμένες πληροφορίες: $\text{data3} = 0x484d4d59$, $\text{data4} = 0x45534441$, $\text{data5} = 37383032$, $\text{data6} = 416c4765$, $\text{data7} = 0x4475636b$

Άλλο ένα χρήσιμο παράθυρο στο οποίο μπορούμε να δούμε την μνήμη του προγράμματος είναι το Memory Usage. Πατώντας την επιλογή **Reenas Views → C/C++ → Memory Usage**, μας εμφανίζεται ένα παράθυρο στο οποίο μπορούμε να δούμε αναλυτικά την μνήμη που καταλαμβάνει κάθε εντολή. Εκτός αυτού μας δίνεται η επιλογή να δούμε τον χάρτη μνήμης του μικροεπεξεργαστή, σε συνδυασμό με το διάγραμμα μνήμης του Σχήμα 52 μπορούμε να πάρουμε μια ιδέα της κατάταξης μνήμης.



Σχήμα 52: Παράθυρο Memory Usage

Ερώτημα 88

Να βρείτε πόσο χώρο καταλαμβάνει το πρόγραμμα που φτιάξατε

Ερώτημα 89

Να βρείτε σε ποια διεύθυνση αποθηκεύεται και πόσο χώρο καταλαμβάνουν οι συγκεκριμένες εντολές: *Main*, *board led*, *register*, *delay*, *clock*

Αφού είδατε τη λειτουργία των leds, τη διαδικασία που χρειάζεται για να τρέξετε τον κώδικα και πολλά αλλά, ήρθε η ώρα να συμπληρώσετε τον δικό σας κώδικα.

Ερώτημα 90

Συμπληρώστε τον τελικό σας κώδικα

15 ΕΡΓΑΣΤΗΡΙΟ 15

♦ Αισθητήρας Θερμοκρασίας

Σε αυτό το κεφάλαιο θα ασχοληθούμε με έναν άλλον αισθητήρα που συμπεριλαμβάνεται στην πλακέτα μας, τον Αισθητήρα Θερμοκρασίας.

Όπως αναφέραμε και στην εισαγωγή, η πλακέτα μας είναι εξοπλισμένη με πολλούς χρήσιμους αισθητήρες, ένας από αυτούς είναι ο αισθητήρας θερμοκρασίας HS3001.

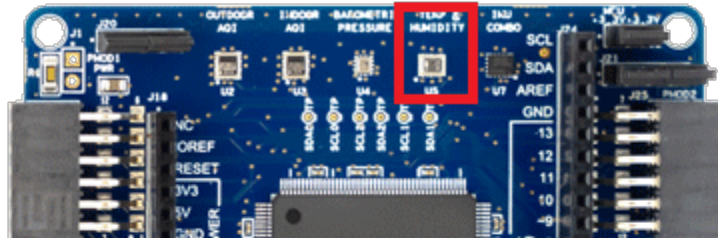
Η σειρά HS3xxx (HS300x και HS310x) είναι ένα εξαιρετικά υψηλής ακρίβειας, πλήρως βαθμονομημένη σχετική υγρασία και θερμοκρασίας. Ενώ οι συσκευές HS300x έρχονται σε τυποποιημένη συσκευασία με είσοδο ανοικτού αέρα, οι συσκευές HS310x διαθέτουν υδρόφοβη μεμβράνη, που τις καθιστά ανθεκτικές στη σκόνη και αδιάβροχες με IP67 αξιολόγηση. Και οι δύο εκδόσεις αισθητήρων, αδιάβροχες και μη αδιάβροχες, διατίθενται σε τρεις διαφορετικές ακρίβειες κατηγορίες: HS3x01, HS3x02 και HS3x03.

Πάρα το μικρό του μέγεθος, το οποίο επεκτείνεται στα 3 x 2.41 x 0.8 mm, αυτό δεν το εμποδίζει από το να κάνει την δουλειά του με υψηλή ακρίβεια μέτρησης, πιο συγκεκριμένα τιμές μεταξύ -40° C και +125° C, γρήγορο χρόνο απόκρισης που φτάνει στο 1 δευτερόλεπτο και χαμηλή κατανάλωση που κυμαίνεται από 1 μ A έως 24,4 μ A.

Πρακτικά οι μηχανικοί συνιστούν την χρήση του αισθητήρα σε θερμοκρασίες πιο προσιτές όπως 10° C με 50° C και 20-80% ποσοστά υγρασίας. Σε άλλες περιπτώσεις θα υπάρχει μια μικρή μετατόπιση στην ακρίβεια των μετρήσεων της κλίμακας $\pm 1 - 3$ βαθμούς Κέλσιου ανάλογως και με τη θερμοκρασία που βρισκόμαστε.



Σχήμα 53: Αισθητήρας HS3xxx

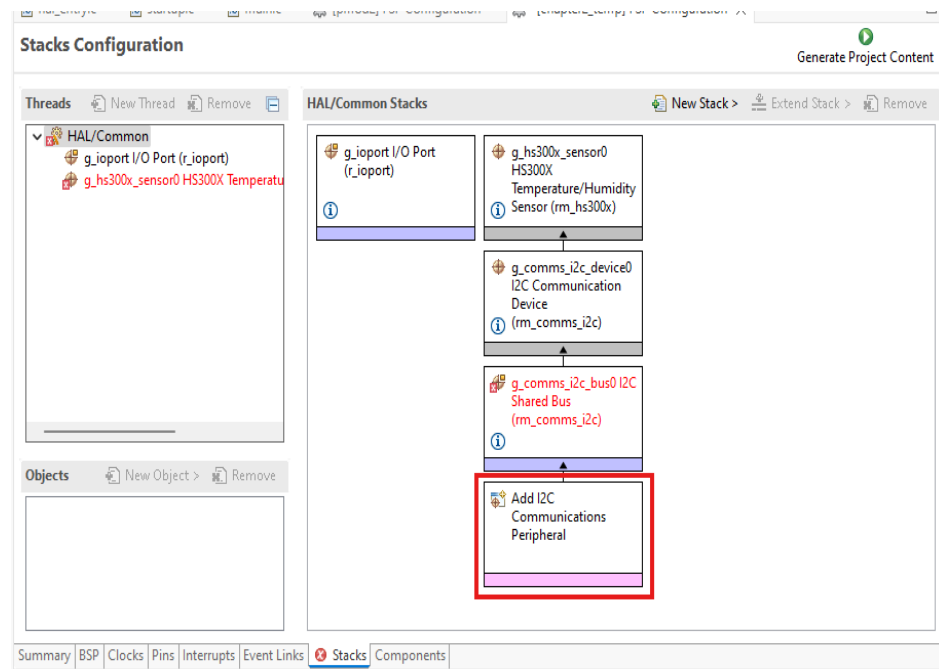


Σχήμα 54: Τοποθεσία του HS3xxx στην πλακέτα

Για επιπλέον πληροφορίες ανατρέξτε στην ιστοσελίδα του [συγκεκριμένου αισθητήρα](#).

Για το πρακτικό κομμάτι της εργασίας θα χρησιμοποιήσουμε ένα βοηθητικό εργαλείο του e² Studio, το **Developer Assistance**. Η συγκεκριμένη επιλογή μας παρέχει έτοιμες συναρτήσεις με βάση τα περιεχόμενα όλων των πλακετών που μπορούμε να συναντήσουμε σε όλη την οικογένεια της Renesas Ra. Σκοπός της επιλογής αυτής είναι να βοηθήσει τους χρήστες να κατανοήσουν πιο εύκολα τον κώδικα, τη λειτουργία κάθε περιφερειακού και κάνει τη διαδικασία του προγραμματισμού πιο εύκολη για όλους.

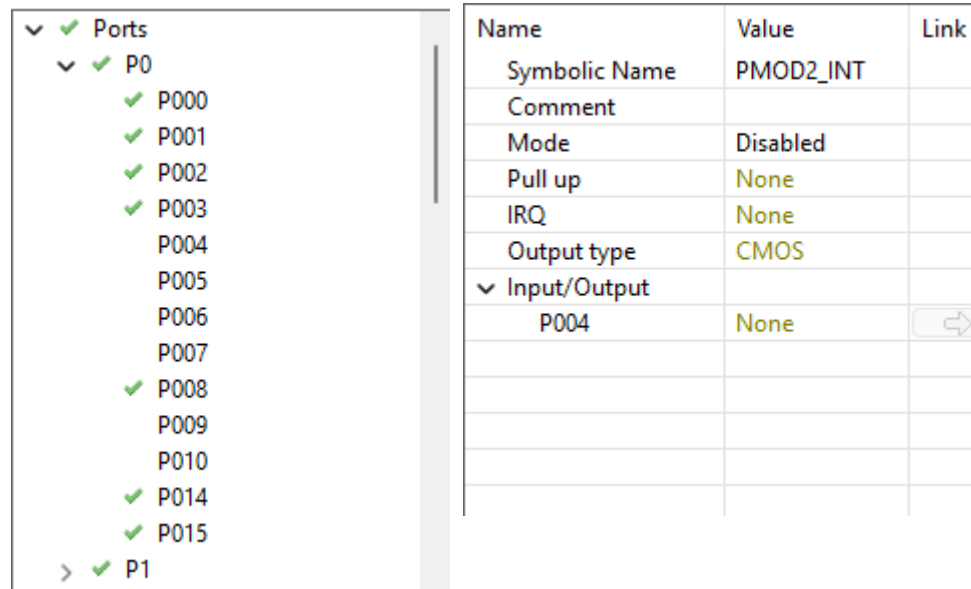
- Αφού ακολουθήσουμε τα βήματα της εισαγωγής και δημιουργήσουμε το project, βλέπουμε με την αρχική σελίδα. Το πρώτο βήμα που ακολουθούμε είναι να εισάγουμε τον αισθητήρα θερμοκρασίας. Πηγαίνουμε στο παράθυρο **Stack** → **New Stack** → **Sensor** → **HS300X Temperature/Humidity Sensor (rm_hs300x)**



- Στη συνέχεια επιλέγουμε το **Add I2C Communications Peripheral → New → I2C Master (r_iic_master)** με σκοπό να προσθέσουμε τον δίαυλο δεδομένων που θα χρησιμοποιήσει ο αισθητήρας μας.

Σε αυτό το σημείο πρέπει να επισημάνουμε ότι η εφαρμογή e² Studio, όσον αφορά τα GPIO, δίνει προτεραιότητα στις ενώσεις που σχετίζονται με τα Pmod και το Arduino Uno, καθώς οι περισσότερες πλακέτες της Renesas δεν έχουν ενσωματωμένους τους αισθητήρες που χρησιμοποιεί η πλακέτα μας. Έτσι στο επόμενο βήμα χρειάζεται να απενεργοποιήσουμε τις συγκεκριμένες εισόδους, δίνοντας στην εφαρμογή να καταλάβει ότι πρέπει να λάβει τις τιμές κατευθείαν από τον αισθητήρα και όχι από κάποιο GPIO που αναφέραμε προηγουμένως.

- Πηγαίνουμε στο παράθυρο **Pins**, ψάχνουμε να απενεργοποιήσουμε τα pins P004, P005, P006 και P111 που αντιστοιχούν στις τιμές των εισόδων των GPIO.

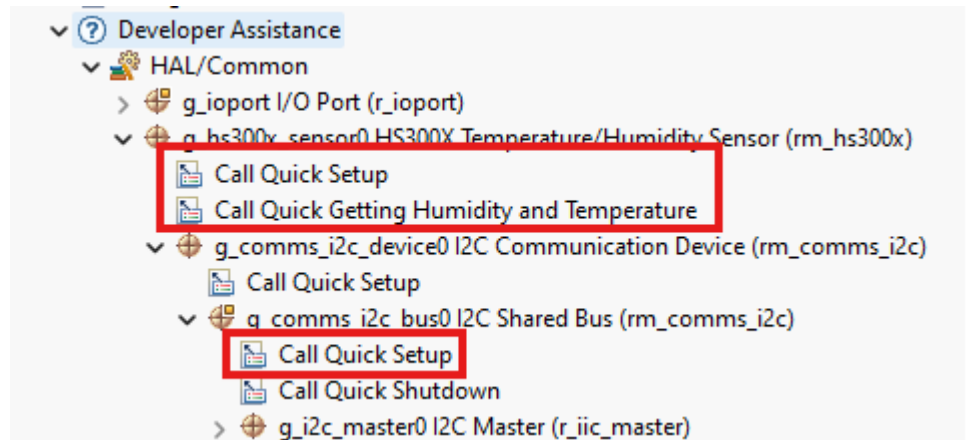


Name	Value	Link
Symbolic Name	PMOD2_INT	
Comment		
Mode	Disabled	
Pull up	None	
IRQ	None	
Output type	CMOS	
Input/Output		
P004	None	

- Τέλος πατάμε την επιλογή **Generate Project Content** με σκοπό να δημιουργηθεί το περιεχόμενο του project.

Αφού τελειώσαμε με το κομμάτι του περιεχομένου, επιστρέφουμε στο παράθυρο `hal_entry.c` με σκοπό να δομήσουμε τον κώδικα.

- Ανοίγουμε την επιλογή **Developer Assistance** και σύρουμε τις παρακάτω συναρτήσεις στο παράθυρο του κώδικα.



- Πηγαίνουμε στον κώδικα και τροποποιούμε τα εξής:

```
/* TODO: Enable if you want to open HS300X */
#define G_HS300X_SENSOR0_NON_BLOCKING (1)

#if G_HS300X_SENSOR0_NON_BLOCKING
volatile bool g_hs300x_completed = false;
#endif

#if RM_HS300X_CFG_PROGRAMMING_MODE
uint32_t g_hs300x_sensor_id;
#endif

/* TODO: Enable if you want to use a callback */
#define G_HS300X_SENSOR0_CALLBACK_ENABLE (1)
#if G_HS300X_SENSOR0_CALLBACK_ENABLE
void hs300x_callback(rm_hs300x_callback_args_t * p_args)
{
    if G_HS300X_SENSOR0_NON_BLOCKING
    {
        if (RM_HS300X_EVENT_SUCCESS == p_args->event)
        {
            g_hs300x_completed = true;
        }
    }
    else
    {
        FSP_PARAMETER_NOT_USED(p_args);
    }
}
#endif
```

- Τέλος αντικαθιστούμε το παρακάτω κομμάτι κώδικα με την hal_entry που βρίσκεται μέσα.

```
volatile rm_hs300x_data_t hs300x_data;
```

```
void hal_entry(void)
```

```
{
```

```
    g_comms_i2c_bus0_quick_setup();
```

```
    g_hs300x_sensor0_quick_setup();
```

```
while(1)
```

```
{
```

```
    g_hs300x_sensor0_quick_getting_humidity_and_temperature
    ((rm_hs300x_data_t*)& hs300x_data);
```

```
}
```

```
#if BSP_TZ_SECURE_BUILD
```

```
    /* Enter non-secure code */
```

```
    R_BSP_NonSecureEnter();
```

```
#endif
```

```
}
```

Για περαιτέρω πληροφορίες μπορείτε να δείτε ένα παρόμοιο παράδειγμα στο ακόλουθο [video](#).

Αφού ακολουθήσαμε σωστά τα βήματα, συνεχίζουμε στο Build και Debugging του κώδικα. Για να μπορέσουμε να μετρήσουμε τη θερμοκρασία θα χρειαστεί να ανοίξουμε το παράθυρο Expressions στο οποίο θα χρησιμοποιήσουμε την συνάρτηση `hs300x_data` με σκοπό να πάρουμε μετρήσεις θερμοκρασίας.

Expression	Type	Value
▼ <code>hs300x_data</code>	volatile r...	{...}
▼ <code>humidity</code>	rm_hs300...	{...}
(x)= <code>integer_part</code>	int16_t	42
(x)= <code>decimal_part</code>	int16_t	50
▼ <code>temperature</code>	rm_hs300...	{...}
(x)= <code>integer_part</code>	int16_t	24
(x)= <code>decimal_part</code>	int16_t	90
<code>Add new expression</code>		

Σχήμα 55: Τιμές Συνάρτησης HS300x

Ερώτημα 91

Να βρείτε ποιες διευθύνσεις χρησιμοποιεί ο αισθητήρας HS3001 και τι αποθηκεύει η κάθε μια

Το αντίστοιχο project αποτελεί καλό παράδειγμα για κατανόηση του memory καθώς μπορούμε μέσω της επιλογής Real-Time Refresh να δούμε τις τιμές του αισθητήρα που ανανεώνονται κατά τη διάρκεια της εκτέλεσης του κώδικα. Για να το κάνουμε αυτό θα ανοίξουμε το παράθυρο **Memory** και θα πάμε στην διεύθυνση που βρήκαμε προηγουμένως και ταυτόχρονα θα εκτελέσουμε το πρόγραμμα με σκοπό να δούμε τις τιμές να ανανεώνονται ζωντανά.

Ένα ενδιαφέρον παράθυρο που μπορούμε να συναντήσουμε στο e² Studio είναι το Visual Expression (**Renesas Views → Debug → Visual Expressions**). Το συγκεκριμένο παράθυρο μας παρέχει διαφορά διαδραστικά widgets, τα οποία μπορούμε να προγραμματίσουμε.

Ερώτημα 92

Να προγραμματίσετε μερικά από τα widget και να εμφανίσετε τα αποτελέσματά σας

Ένα χρήσιμο παράθυρο που μας παρέχει το e² Studio είναι το Real-Time Chart (**Renesas Views → Debug → Real-Time Chart**). Χρησιμοποιώντας την συγκεκριμένη επιλογή μας δίνεται η δυνατότητα να βάλλουμε τις τιμές που λαμβάνουμε από τον αισθητήρα σε γραφική ανάρτηση. Αφού εισάγετε τις κατάλληλες συναρτήσεις και πραγματοποιήσετε το πειραματικό σας κομμάτι, να απαντήσετε στην παρακάτω ερώτηση.

Ερώτημα 93

Εμφανίστε τις γραφικές της Θερμοκρασίας. Τι παρατηρείτε;

Ερώτημα 94

Χρησιμοποιώντας ένα διαφορετικό θερμόμετρο, να αναφέρετε αν οι τιμές των δυο θερμομέτρων είναι ίδιες, ειδάλλως να αναφέρετε τον λόγο που διαφέρουν οι μετρήσεις μεταξύ τους

Πηγαίνετε στο παράθυρο **Memory Usage** και απαντήστε στις παρακάτω ερωτήσεις.

Ερώτημα 95

Να βρείτε πόσο χώρο καταλαμβάνει το πρόγραμμα που φτιάξατε

Ερώτημα 96

Να βρείτε σε ποια διεύθυνση αποθηκεύεται και πόσο χώρο καταλαμβάνουν οι συγκεκριμένες εντολές: `main`, `clock init`, `hal_entry`, `leds`, `delay`, `system init`

Αφού είδατε τον αισθητήρα Θερμοκρασίας, τη διαδικασία που χρειάζεται για να τρέξετε τον κώδικα και πολλά αλλά, στη συνέχεια ήρθε η ώρα να συμπληρώσετε τον δικό σας κώδικα.

Ερώτημα 97

Συμπληρώστε τον τελικό σας κώδικα

16 ΕΡΓΑΣΤΗΡΙΟ 16

♦ Αισθητήρας Οξυγόνου

Σ' αυτό το κεφάλαιο θα ασχοληθούμε με έναν άλλον αισθητήρα που συμπεριλαμβάνεται στην πλακέτα μας, τον Αισθητήρα Μέτρησης Ποιότητας Αέρος.

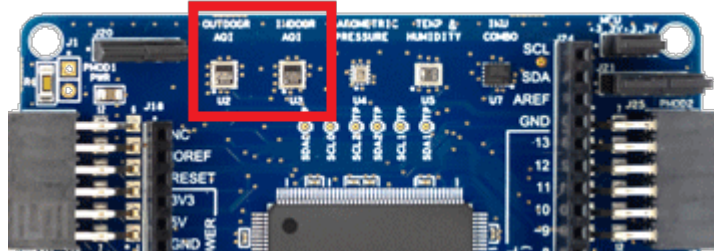
Όπως αναφέραμε και στην εισαγωγή η πλακέτα μας είναι εξοπλισμένη με πολλούς χρήσιμους αισθητήρες, ένας από αυτούς είναι ο αισθητήρας μέτρησης ποιότητας αέρος ZMOD4410. Ο αισθητήρας αυτός έχει σχεδιαστεί για την ανίχνευση των ουσιών VOC, οι οποίες αναφέρονται στις πτητικές οργανικές ουσίες, την εκτίμηση του διοξειδίου του άνθρακα CO₂ και την παρακολούθηση της ποιότητας του αέρα σε εσωτερικούς χώρους. Ο αισθητήρας είναι εξοπλισμένος με ένα θερμαντικό στοιχείο σε δομή MEMS που βασίζεται σε πυρίτιο και μια χημειαντίσταση οξειδίου μετάλλου (MOx). Ο επεξεργαστής σήματος ελέγχει τις θερμοκρασίες του αισθητήρα και μετρά τη MOx αγωγιμότητα. Χρησιμοποιώντας παραδοσιακούς αλγορίθμους και αλγορίθμους μηχανικής μάθησης με ενσωματωμένη τεχνητή νοημοσύνη μπορεί να προσδιοριστεί η συγκέντρωση TVOC, μια IAQ τιμή και μια η εκτιμώμενη ποσότητα διοξειδίου του άνθρακα (eCO₂) στο χώρο.

Πάρα το μικρό του μέγεθος, το οποίο επεκτείνεται στα 3 x 3 x 0.7 mm, αυτό δεν τον εμποδίζει να λειτουργεί σε συνθήκες όπου έρχεται σε επαφή με υγρασία, σκόνη και λόγω της IP67 αξιολόγησης, μπορεί να έρθει σε επαφή με νερό ή ακόμα και να βυθιστεί σε αυτό. Επίσης χάρη στη χρήση των συνθέτων αλγορίθμων μπορούμε να υπολογίσουμε επίσης την ποσότητα μόλυβδου στον αέρα, όπως και οσμές με βάση το θείο, όπως είναι η κακή ανάσα, το γκρέιπφρουτ και το κρεμμύδι.

Πρέπει να επισημάνουμε ότι στην πλακέτα υπάρχει ένας δεύτερος αισθητήρας μέτρησης αέρος, ο ZMOD4510. Σκοπός του είναι η ανίχνευση των ουσιών VOC σε εξωτερικούς χώρους αλλά μπορεί να μας μετρήσει επιπλέον την τιμή του διοξειδίου του αζώτου και του όζοντος αέρα.



Σχήμα 56: Αισθητήρας ZMOD4410

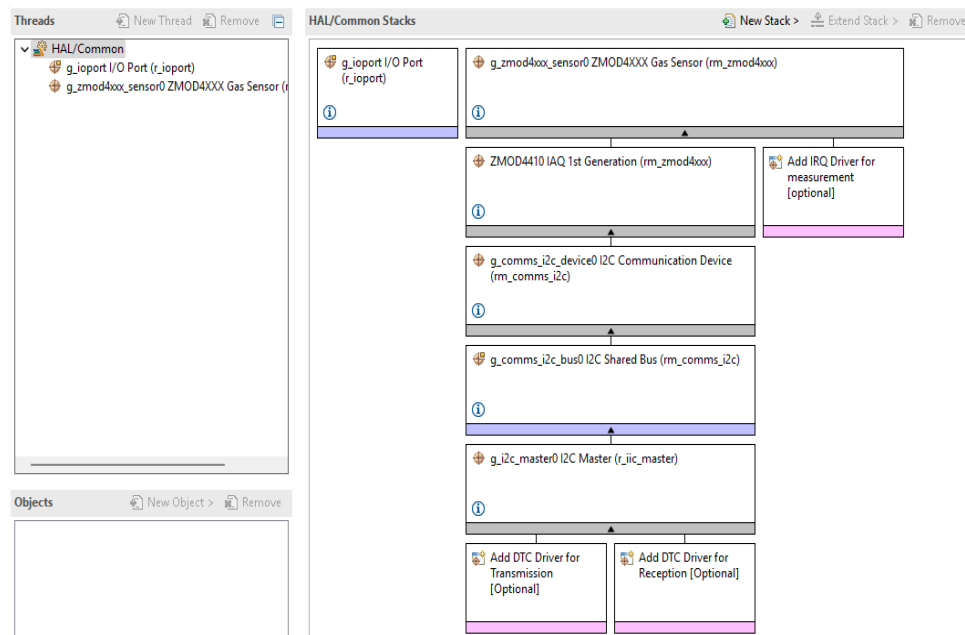


Σχήμα 57: Τοποθεσία του ZMOD4410

Για επιπλέον πληροφορίες ανατρέξτε στην ιστοσελίδα του [συγκεκριμένου αισθητήρα](#).

Για το πρακτικό κομμάτι της εργασίας θα χρησιμοποιήσουμε ένα βοηθητικό εργαλείο του e² Studio, το **Developer Assistance**.

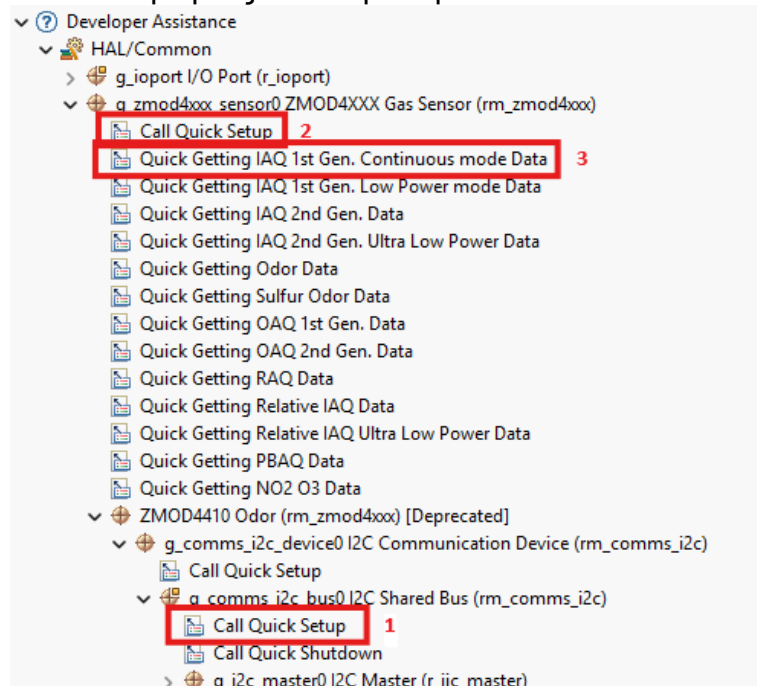
- Αφού ακολουθήσουμε τα βήματα της εισαγωγής και δημιουργήσουμε το project, ερχόμαστε στην αρχική σελίδα. Το πρώτο βήμα που ακολουθούμε είναι να εισάγουμε τον αισθητήρα που θα χρειαστούμε. Πηγαίνουμε στο παράθυρο **Stack** → **New Stack** → **Sensor** → **ZMOD4XXX Gas Sensor (rm_zmod4xxx)**.
- Διαλέγουμε την επιλογή **Add Requires ZMOD Library** → **New** → **ZMOD4410 IAQ 1st Generation** με σκοπό να προσθέσουμε την συνάρτηση που θα χρειαστούμε ώστε να πάρουμε τις μετρήσεις. Σε περίπτωση που θέλουμε να δοκιμάσουμε κάποια άλλη συνάρτηση αλλάζουμε αυτή την επιλογή.
- Στη συνέχεια επιλέγουμε το **Add I2C Communications Peripheral** → **New** → **I2C Master (r_iic_master)** με σκοπό να προσθέσουμε τον δίαυλο δεδομένων που θα χρησιμοποιήσει ο αισθητήρας μας.



- Πηγαίνουμε στο παράθυρο **Pins**, ψάχνουμε να απενεργοποιήσουμε τα pins P004, P005, P006 και P111 που αντιστοιχούν στις τιμές των εισόδων των Pmod.
- Τέλος πατάμε την επιλογή **Generate Project Content** με σκοπό να δημιουργηθεί το περιεχόμενο του project.

Αφού τελειώσαμε με το κομμάτι του περιεχομένου, επιστρέφουμε στο παράθυρο `hal_entry.c` με σκοπό να δομήσουμε τον κώδικα.

- Ανοίγουμε την επιλογή **Developer Assistance** και σύρουμε τις παρακάτω συναρτήσεις στο παράθυρο του κώδικα.



- Πηγαίνουμε στον κώδικα και τροποποιούμε τα εξής:

```

/* TODO: Enable if you want to open ZMOD4XXX */
#define G_ZMOD4XXX_SENSOR0_NON_BLOCKING (1)
#define G_ZMOD4XXX_SENSOR0_IRQ_ENABLE (0)

⊕ #if G_ZMOD4XXX_SENSOR0_NON_BLOCKING
⊖ #if G_ZMOD4XXX_SENSOR0_IRQ_ENABLE
    volatile bool g_zmod4xxx_irq_completed = false;
#endif

/* TODO: Enable if you want to use a I2C callback */
#define G_ZMOD4XXX_SENSOR0_I2C_CALLBACK_ENABLE (1)
⊕ #if G_ZMOD4XXX_SENSOR0_I2C_CALLBACK_ENABLE

/* TODO: Enable if you want to use a IRQ callback */
#define G_ZMOD4XXX_SENSOR0_IRQ_CALLBACK_ENABLE (0)
⊖ #if G_ZMOD4XXX_SENSOR0_IRQ_CALLBACK_ENABLE
⊕ void zmod4xxx_irq_callback(rm_zmod4xxx_callback_args_t * p_args)
#endif

```

- Τέλος αντικαθιστούμε το παρακάτω κομμάτι κώδικα με την `hal_entry` που βρίσκεται μέσα.

```
volatile rm_zmod4xxx_iaq_1st_data_t iaq_1st_gen_data;
```

```
void hal_entry(void)
```

```

{
    g_comms_i2c_bus0_quick_setup();
    g_zmod4xxx_sensor0_quick_setup();

    while(1)
    {

        g_zmod4xxx_sensor0_quick_getting_iaq_1st_gen_continuous_mod
        e_data((rm_zmod4xxx_iaq_1st_data_t*)& iaq_1st_gen_data);
    }

    #if BSP_TZ_SECURE_BUILD
        /* Enter non-secure code */
        R_BSP_NonSecureEnter();
    #endif
}

```

Αφού ακολουθήσαμε σωστά τα βήματα, συνεχίζουμε στο Build και Debugging του κώδικα. Για να μπορέσουμε να μετρήσουμε την ποιότητα του IAQ θα χρειαστεί να ανοίξουμε το παράθυρο Expressions στο οποίο θα χρησιμοποιήσουμε την συνάρτηση `iaq_1st_gen_data` με σκοπό να πάρουμε μετρήσεις.

Expression	Type	Value
▼  iaq_1st_gen_data	volatile rm_zm...	{...}
(x)= rmox	float	27483.6562
(x)= rcda	float	27483.6562
(x)= iaq	float	0.999999881
(x)= tvoc	float	0.0158526525
(x)= etoh	float	0.00843226351
(x)= eco2	float	404.523804

Σχήμα 58: Τιμές Συνάρτησης IAQ

Ερώτημα 98

Να βρείτε ποιες διευθύνσεις χρησιμοποιεί ο αισθητήρας ZMOD4410 και τι αποθηκεύει η κάθε μια

Μεταβείτε στο παράθυρο **Real-Time Chart** και αφού εισάγεται τις κατάλληλες συναρτήσεις και πραγματοποιήσετε το πειραματικό σας κομμάτι, να απαντήσετε στην παρακάτω ερώτηση.

Ερώτημα 99

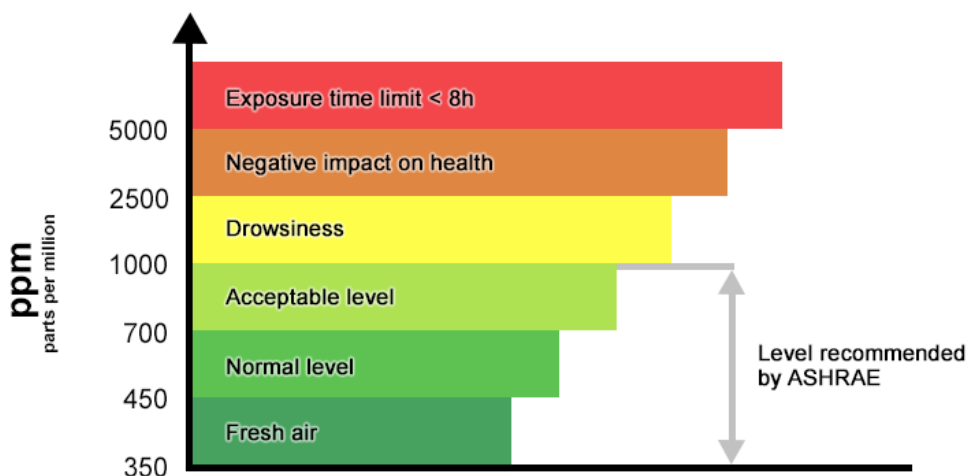
Να εμφανίσετε τη γραφική του ZMOD4410. Τι παρατηρείτε;

Παρακάτω μπορείτε να δείτε το διάγραμμα συγκεντρώσεων διοξειδίου του άνθρακα (CO₂) σε ppm (μέρη ανά εκατομμύριο), τις επιπτώσεις τους στην ανθρώπινη υγεία αλλά και το πίνακα αξιολόγησης της ποιότητας του αέρα (IAQ - Indoor Air Quality), βασισμένο στη συγκέντρωση πτητικών οργανικών ενώσεων (TVOC) σε mg/m³. Όπως μπορούμε να δούμε, για μια καλή διαβίωση και εργασία η ποιότητα του αέρα πρέπει να διατηρείται στα

Επίπεδα 1 ή 2 δηλαδή IAQ μικρότερο από την τιμή 3 τα επίπεδα του CO₂ πρέπει να μην ξεπερνούν τα 1000 ppm.

IDT IAQ Rating	Reference Level*	Air Information	TVOC (mg/m ³)	Air Quality
≤ 1.99	Level 1	Clean Hygienic Air (Target Value)	< 0.3	Very Good
2.00 - 2.99	Level 2	Good Air Quality (if no threshold is exceeded)	0.3 - 1.0	Good
3.00 - 3.99	Level 3	Noticeable Comfort Concerns (Not recommended for exposure > 12 months)	1.0 - 3.0	Medium
4.00 - 4.99	Level 4	Significant Comfort Issues (Not recommended for exposure > 1 months)	3.0 - 10.0	Poor
≥ 5.00	Level 5	Unacceptable conditions (Not recommended)	> 10.0	Bad

Σχήμα 59: Πίνακας IAQ – TVOC



Σχήμα 60: Πίνακας CO₂

Σε γενικές γραμμές η Ελλάδα βρίσκεται στη θέση 92 στον κόσμο ως η πιο μολυσμένη χώρα με επίπεδο 3, με την πιο μολυσμένη πόλη να είναι τα Ιωάννινα και την πιο καθαρή να είναι η Νέα Ραιδεστός. Στο άλλο άκρο της γης μπορούμε να συναντήσουμε χώρες όπως το Μπαγκλαντές, το Πακιστάν

και η Ινδία με επίπεδο 7 όπου οι τιμές του TVOC φτάνουν έως 76.9 $\mu\text{g}/\text{m}^3$ με βάση τη [δημοσκόπηση](#) της εταιρία IQAir.

Ερώτημα 100

Με βάση τις τιμές που λάβατε από τον αισθητήρα, να αναφέρετε τι ποιότητα αέρος έχετε και πως αυτή επηρεάζεται σε γενικές γραμμές

Μεταβείτε στο παράθυρο **Memory Usage** και απαντήστε στις παρακάτω ερωτήσεις.

Ερώτημα 101

Να βρείτε πόσο χώρο καταλαμβάνει το πρόγραμμα που φτιάξατε

Ερώτημα 102

Να βρείτε σε ποια διεύθυνση αποθηκεύεται και πόσο χώρο καταλαμβάνουν οι συγκεκριμένες εντολές: `main`, `zmod4410_iaq_1st_gen`, `1st_gen_continuous_mode_data`, `sensor0_quick_setup`, `calc_iaq_1st_gen`

Αφού είδατε τον αισθητήρα Zmod4XXX, πως μπορεί να χρησιμοποιηθεί, τη διαδικασία που χρειάζεται για να τρέξετε τον κώδικα και πολλά αλλά, στη συνέχεια ήρθε η ώρα να συμπληρώσετε το δικό σας κώδικα.

Ερώτημα 103**Συμπληρώστε τον τελικό σας κώδικα**

ΠΑΡΑΡΤΗΜΑ

Cloud Kit for RA6M5 Microcontroller Group

CK-RA6M5 v1 with RYZ014A Pmod

User's Manual

RA6M5 Group

Cloud Kit for RA6M5 Microcontroller Group
CK-RA6M5 v1 with RYZ014A Pmod
User's Manual

Renesas RA Family
RA6 Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.renesas.com>).

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Renesas CK-RA6M5 Disclaimer

By using this CK-RA6M5, the User accepts the following terms, which are in addition to, and control in the event of disagreement, with Renesas' General Terms and Conditions available at [renesas.com/legal-notices](https://www.renesas.com/legal-notices).

The CK-RA6M5 is not guaranteed to be error free, and the entire risk as to the results and performance of the CK-RA6M5 is assumed by the User. The CK-RA6M5 is provided by Renesas on an "as is" basis without warranty of any kind whether express or implied, including but not limited to the implied warranties of good workmanship, fitness for a particular purpose, title, merchantability, and non-infringement of intellectual property rights. Renesas expressly disclaims any implied warranty.

Renesas does not consider the CK-RA6M5 to be a finished product and therefore the CK-RA6M5 may not comply with some requirements applicable to finished products, including, but not limited to recycling, restricted substances and electromagnetic compatibility regulations. Refer to Certifications section, for information about certifications and compliance information for the CK-RA6M5. It is the kit User's responsibility to make sure the kit meets any local requirements applicable to their region.

Renesas or its affiliates shall in no event be liable for any loss of profit, loss of data, loss of contract, loss of business, damage to reputation or goodwill, any economic loss, any reprogramming or recall costs (whether the foregoing losses are direct or indirect) nor shall Renesas or its affiliates be liable for any other direct or indirect special, incidental or consequential damages arising out of or in relation to the use of this CK-RA6M5, even if Renesas or its affiliates have been advised of the possibility of such damages.

Renesas has used reasonable care in preparing the information included in this document, but Renesas does not warrant that such information is error free nor does Renesas guarantee an exact match for every application or parameter to part numbers designated by other vendors listed herein. The information provided in this document is intended solely to enable the use of Renesas products. No express or implied license to any intellectual property right is granted by this document or in connection with the sale of Renesas products. Renesas reserves the right to make changes to specifications and product descriptions at any time without notice. Renesas assumes no liability for any damages incurred by you resulting from errors in or omissions from the information included herein. Renesas cannot verify, and assumes no liability for, the accuracy of information available on another company's website.

Precautions

This Evaluation Kit is only intended for use in a laboratory environment under ambient temperature and humidity conditions. A safe separation distance should be used between this and any sensitive equipment. Its use outside the laboratory, classroom, study area, or similar such area invalidates conformity with the protection requirements of the Electromagnetic Compatibility Directive and could lead to prosecution.

The product generates, uses, and can radiate radio frequency energy and may cause harmful interference to radio communications. There is no guarantee that interference will not occur in a particular installation. If this equipment causes harmful interference to radio or television reception, which can be determined by turning the equipment off or on, you are encouraged to try to correct the interference by one or more of the following measures:

- Ensure attached cables do not lie across the equipment.
- Reorient the receiving antenna.
- Increase the distance between the equipment and the receiver.
- Connect the equipment into an outlet on a circuit different from that which the receiver is connected.
- Power down the equipment when not in use.
- Consult the dealer or an experienced radio/TV technician for help.

Note: It is recommended that wherever possible shielded interface cables are used.

The product is potentially susceptible to certain EMC phenomena. To mitigate against them it is recommended that the following measures be undertaken:

- The user is advised that mobile phones should not be used within 10 m of the product when in use.
- The user is advised to take ESD precautions when handling the equipment.

The Evaluation Kit does not represent an ideal reference design for an end product and does not fulfill the regulatory standards for an end product.

Renesas RA Family

CK-RA6M5 v1 with RYZ014A Pmod
Contents

1. Kit Overview	5
1.1 Assumptions and Advisory Notes	7
2. Kit Contents	8
3. Ordering Information	8
4. Hardware Architecture and Default Configuration	9
4.1 Kit Architecture	9
4.2 System Block Diagram	10
4.3 Jumper Settings	10
4.3.1 Copper Jumpers	10
4.3.2 Traditional Pin Header Jumpers	11
4.3.3 Default Jumper Configuration	11
5. System Control and Ecosystem Access	12
5.1 Power	12
5.1.1 Power Supply Options	12
5.1.2 Power Supply Considerations	13
5.1.3 Power-up Behavior	13
5.2 Debug and Trace	13
5.2.1 Debug On-Board	13
5.2.2 Debug In	14
5.3 Ecosystem	15
5.3.1 Digilent Pmod™ Connectors	15
5.3.2 Arduino™ Connector	17
5.4 Connectivity	18
5.4.1 USB Full Speed	18
5.5 Miscellaneous	19
5.5.1 User and Power LEDs	19
5.5.2 User and Reset Switches	19
5.5.3 MCU Boot Mode	20
5.6 Ethernet	20
5.7 Onboard Sensors	21
5.7.1 HS3001 Temperature and Humidity Sensor	21
5.7.2 OB1203 Biometric Sensor	22
5.7.3 ZMOD4510 Outdoor Air Quality Sensor	22

5.7.4	ZMOD4410 Indoor Air Quality Sensor	23
5.7.5	ICM-20948 9-AXIS MEMS Motion Tracking Sensor	23
5.7.6	ICP-10101 Barometric Pressure Sensor	24
5.7.7	SPH0645 MEMS Microphones	24
6.	MCU Native Access Area	24
6.1	MCU and Pmod Current Measurement	24
7.	Recommended Parts	25
8.	Certifications	26
8.1	EMC/EMI Standards	26
8.2	Material Selection, Waste, Recycling and Disposal Standards	26
8.3	Safety Standards	26
9.	Design and Manufacturing Information	27
10.	Website and Support	27
	Revision History	28

Figures

Figure 1.	CK-RA6M5 Board Top Side	6
Figure 2.	CK-RA6M5 Board Bottom Side	7
Figure 3.	CK-RA6M5 Kit Contents	8
Figure 4.	CK-RA6M5 Board Block Diagram	10
Figure 5.	Copper Jumpers	11
Figure 6.	Power Supply Options	12
Figure 7.	CK-RA6M5 Debug Interface	14
Figure 8.	Pmod 1 Connector	16
Figure 9.	Pmod 2 Connector	16
Figure 10.	Arduino Uno Connectors	18
Figure 11.	USB Full Speed Connector	18
Figure 12.	User LEDs	19
Figure 13.	Power LED	19
Figure 14.	Reset and User Switches	20
Figure 15.	Boot Mode	20
Figure 16.	Ethernet Connector	21
Figure 17.	On Board Sensors	21
Figure 18.	RA Pmod Current Measurement Circuit	24
Figure 19.	RA +3.3 V Current Measurement Circuit	25
Figure 20.	RA MCU Current Measurement	25

Tables

Table 1.	List of Abbreviations and Acronyms	4
Table 2.	Default Jumper Settings	11
Table 3.	Debug Modes	13
Table 4.	Jumper Connection Summary for Different Debug Modes	13
Table 5.	Debug USB Connector	13
Table 6.	Debug On-Board Jumper Configuration	14
Table 7.	Debug In Mode Jumper Configuration	14
Table 8.	JTAG/SWD Connector	14
Table 9.	Pmod 1 Connector	15
Table 10.	Pmod 2 Connector	16
Table 11.	Arduino Uno Connections	17
Table 12.	USB Full Speed Connector	18
Table 13.	CK-RA6M5 Board LED Functions	19
Table 14.	CK-RA6M5 Board Switches	19
Table 15.	Ethernet Port Assignments	20
Table 16.	Ethernet Components	20
Table 17.	HS3001 Port Assignment	21
Table 18.	OB1203 Port Assignment	22
Table 19.	ZOMD4510 Port Assignment	22
Table 20.	ZMOD4410 Port Assignment	23
Table 21.	ICM-20948 Port Assignment	23
Table 22.	ICP-10101 Port Assignment	24
Table 23.	SPH0645 Port Assignment	24
Table 24.	Part Numbers	25
Table 25.	CK-RA6M5 Board Design Package Contents	27

Glossary**Table 1. List of Abbreviations and Acronyms**

BoM	Bill of Materials
CK	Cloud Kit
ETM	Embedded Trace Module
FSP	Flexible Software Package
GPIO	General Purpose Input Output
I ² C (or IIC)	Inter-Integrated Circuit
IDE	Integrated Development Environment
I/O	Input/Output
IRQ	Interrupt Request
LDO	Low Dropout
LED	Light Emitting Diode
LQFP	Low-profile Quad Flat Pack
MCU	Micro Controller Unit
MEMS	Micro-Electromechanical Systems
MISO	Master In Slave Out
MOSI	Master Out Slave In
NC	Not Connected
PMOD™	Peripheral Module
PWM	Pulse Width Modulation
RXD	Receive Data
SCI	Serial Communications Interface
SCL	Serial Clock Line
SDA	Serial Data Line
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SWD	Serial Wire Debug
TXD	Transmit Data
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus

1. Kit Overview

The CK-RA6M5, Cloud Kit for RA6M5 MCU Group, enables users to experience the cloud connectivity options available from Renesas and Renesas Partners. A broad array of sensors on the CK-RA6M5 provide multiple options for observing user interaction with the Cloud Kit. By selecting from a choice of add-on devices, multiple cloud connectivity options are available.

MCU Features

- R7FA6M5BH3CFC MCU (referred to as RA MCU)
- 200 MHz, Arm® Cortex®-M33 core
- 2 MB Code Flash, 8 kB data flash, 512 kB SRAM
- 176 pins, LQFP package
- MCU current measurement points for precision current consumption measurement
- Multiple clock sources - RA MCU oscillator and sub-clock oscillator crystals, providing precision 24.000 MHz and 32,768 Hz reference clock. Additional low-precision clocks are available internal to the RA MCU

Ecosystem Access and Connectivity

- USB Full Speed Host and Device (micro AB connector)
- USB High Speed Device (micro AB connector)
- Three 5 V input sources
 - USB (Debug, Full Speed)
 - External power supply (using traditional pin jumpers)
- Debug modes
 - Debug on-board (SWD)
 - Debug in (ETM, SWD, and JTAG)
- User LEDs and buttons
 - Four User LEDs (red, blue, green, RGB)
 - Power LED (white) indicating availability of regulated power
 - Debug LED (yellow) indicating the debug connection
 - One User button
 - One Reset button
- Two of the most popular ecosystems expansions
 - 2 Digilent Pmod™ (Type 2A and Type 6A) connectors
 - Arduino™ (Uno R3) connector
- MCU boot configuration jumper
- Ethernet (RJ45 RMII interface)

Sensors

- Renesas HS3001 Humidity and Temperature sensor
- Renesas ZMOD4410 Indoor Air Quality Index sensor
- Renesas ZMOD4510 Outdoor Air Quality Index sensor
- Renesas OB1203 Biometric sensor
- TDK ICP-10101 Barometric Pressure sensor
- TDK ICM-20948 IMU (Gyroscope, Magnetometer, Accelerometer)
- Knowles SPH0645LM4H-B MEMS Microphones in stereo configuration

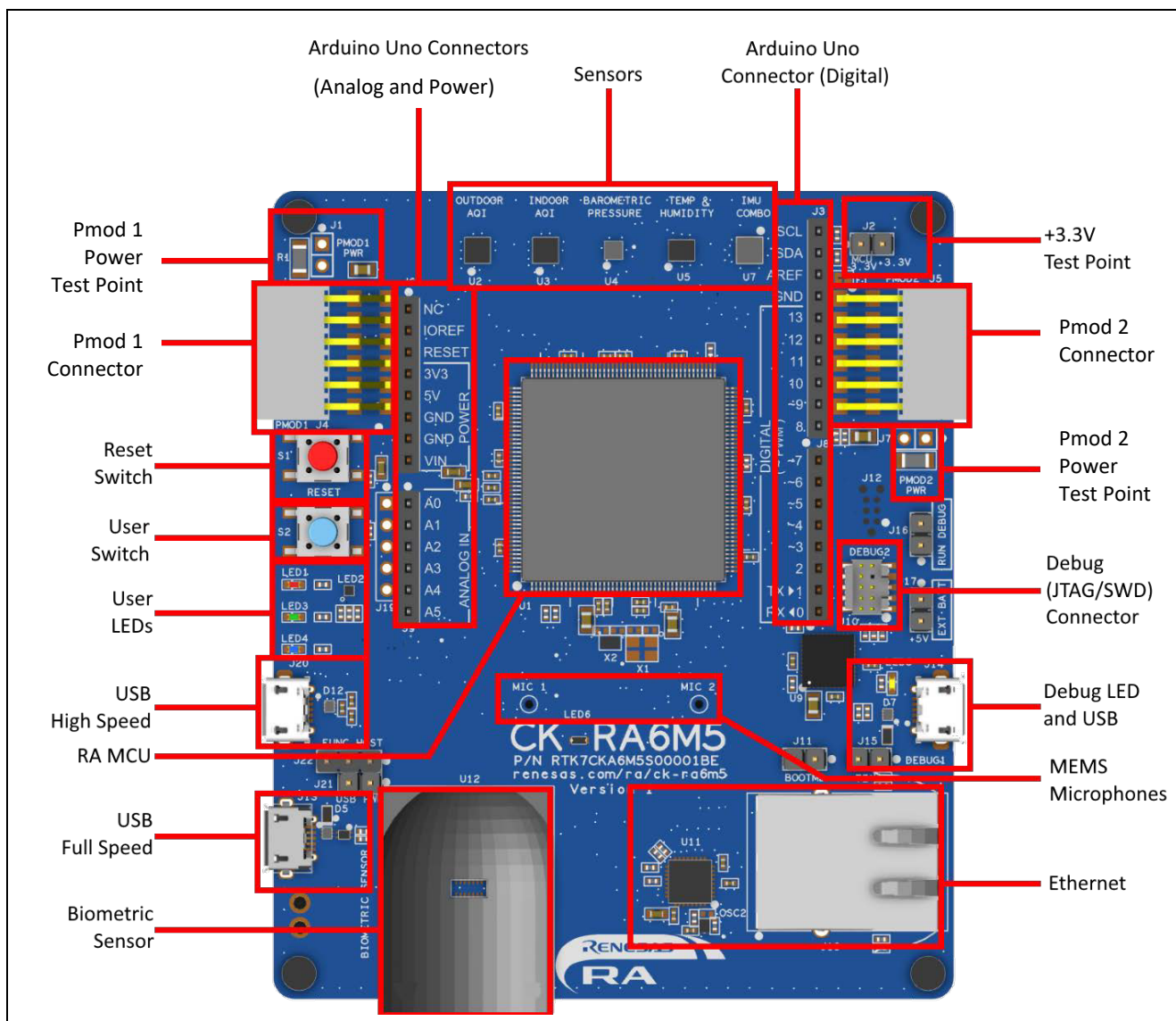


Figure 1. CK-RA6M5 Board Top Side

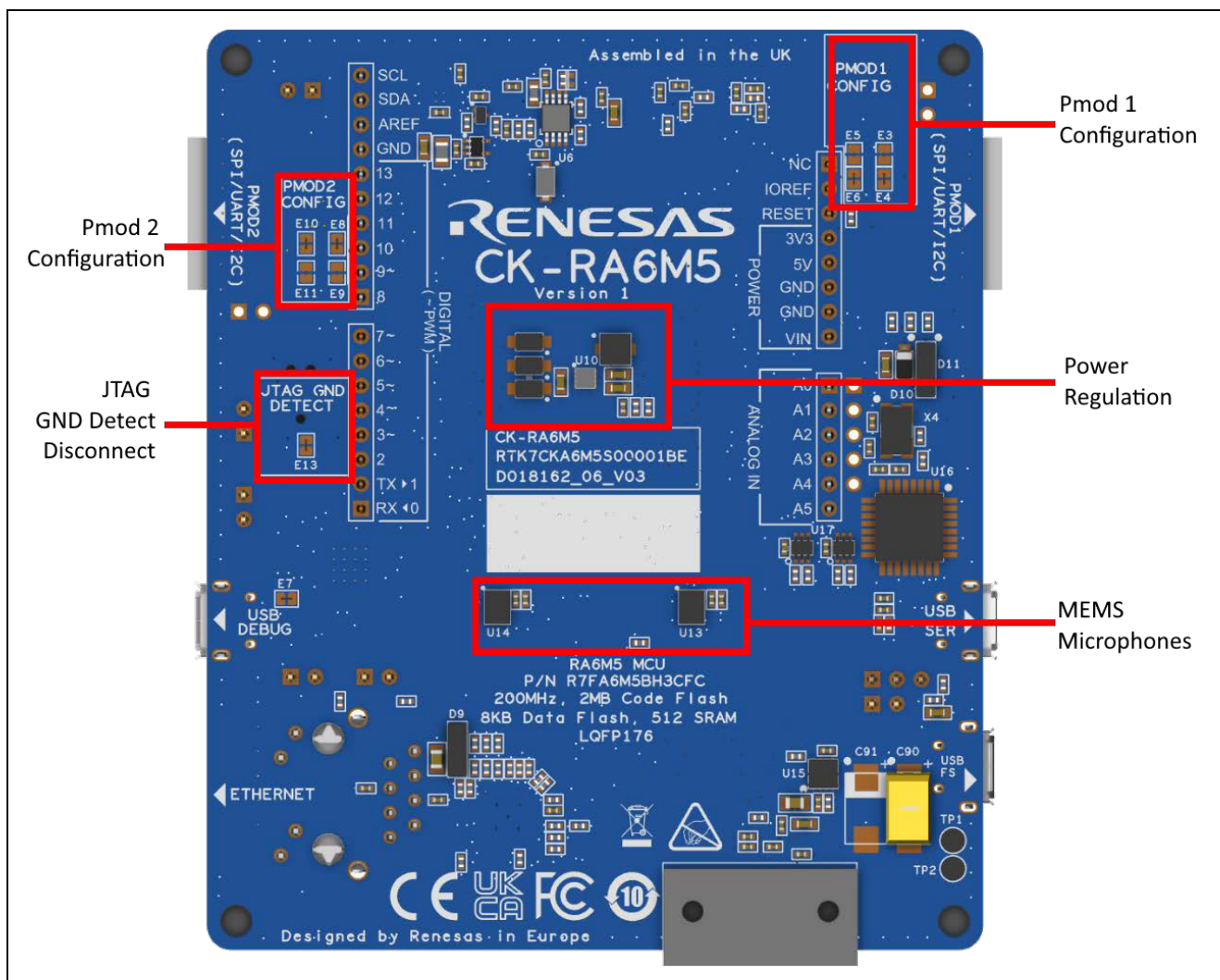


Figure 2. CK-RA6M5 Board Bottom Side

1.1 Assumptions and Advisory Notes

1. It is assumed that the user has a basic understanding of microcontrollers and embedded systems hardware.
2. It is recommended that the user refers to the *CK-RA6M5 Quick Start Guide* to get acquainted with the kit and the Quick Start example project that the CK-RA6M5 board comes pre-programmed with.
3. Flexible Software Package (FSP) and an Integrated Development Environment (IDE) such as e² studio are required to develop embedded applications on the CK-RA6M5 kit.
4. Instructions to download and install software, import example projects, build them and program the CK-RA6M5 board are provided in the quick start guide.
5. The MCU fitted to the CK board may not contain the latest version of the on-chip boot firmware.

2. Kit Contents

The following components are included in the kit:

1. CK-RA6M5 v1 board
2. RYZ014A PMOD (RA CAT-M1 Cloud Kit)
3. Micro USB to A cable
4. Micro USB A/B to A adapter cable
5. Sim Card (RA CAT-M1 Cloud Kit)
6. Antenna (RA CAT-M1 Cloud Kit)

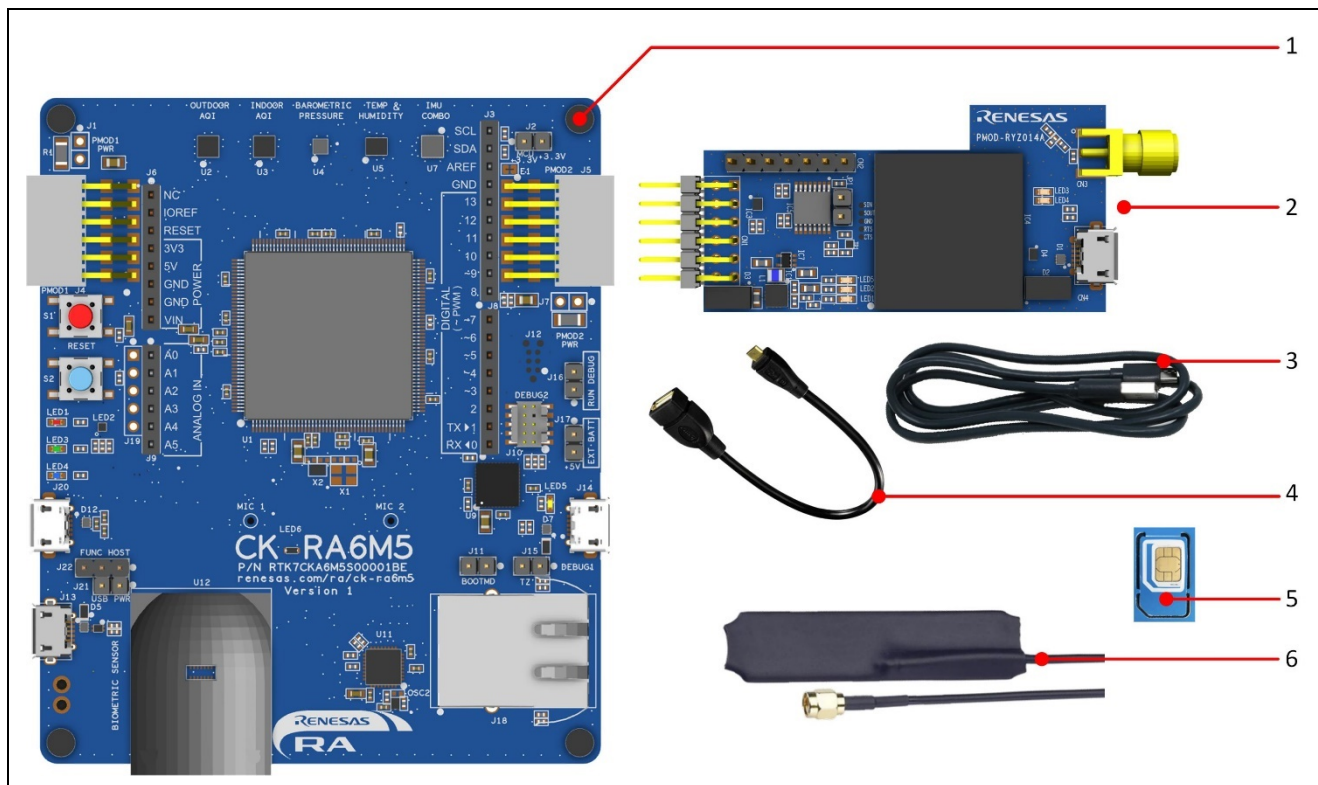


Figure 3. CK-RA6M5 Kit Contents

3. Ordering Information

- CK-RA6M5 v1 kit orderable part number: RTK7CKA6M5S04001BE

Note: The underlined character in the orderable part number represents the kit version.

- CK-RA6M5 board dimensions: 80 mm (width) x 95 mm (length)

4. Hardware Architecture and Default Configuration

4.1 Kit Architecture

The CK-RA6M5 board is designed to demonstrate many options available to connect the RA6M5 MCU to various cloud services. The board includes the RA6M5 MCU, multiple environmental sensors and connectivity options such as USB, Ethernet, and Pmod connections for use with add-on wireless connectivity devices, as well as support for Arduino Uno R3 shields. The board also includes an on-board debugger and all required hardware needed to power, program, and debug software running on the MCU. There are LED indicators and a push-button available for user interaction with the board. To allow portability the board includes a 2-pin header for a battery connection which can run with 3 x AA batteries.

Most of the on-board sensors are connected to the MCU using I²C. The on-board microphones use the SSIE peripheral on the RA6M5 MCU.

Board Functionality	Area Features	Function present on all similar kits	Functionality is:
MCU Native Access	RA MCU, power / current measurement	Yes	MCU dependent
System Control and Ecosystem Access	Power, Debug MCU, User LED and buttons, reset, ecosystem connectors, USB Full Speed, Boot configuration, Ethernet	Yes	Same or similar across similar kits

4.2 System Block Diagram

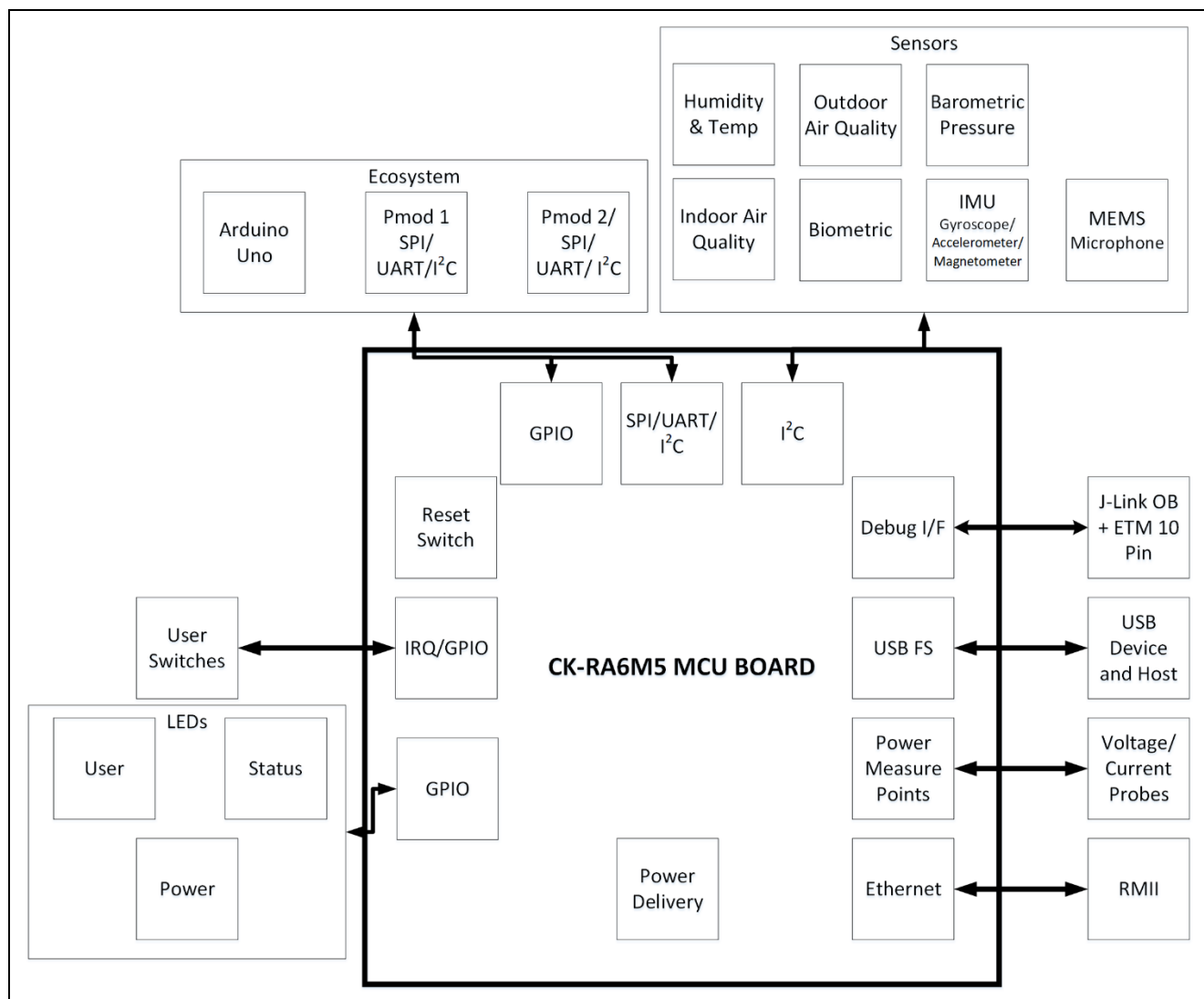


Figure 4. CK-RA6M5 Board Block Diagram

4.3 Jumper Settings

Two types of jumpers are provided on the CK-RA6M5 board.

1. Copper jumpers (trace-cut type and solder bridge type)
2. Traditional pin header jumpers

The following sections describe each type and their default configuration.

4.3.1 Copper Jumpers

Copper jumpers are of two types, designated **trace-cut** and **solder-bridge**.

A **trace-cut jumper** is provided with a narrow copper trace connecting its pads. The silk screen overlay printing around a trace-cut jumper is a solid box. To isolate the pads, cut the trace between pads adjacent to each pad, then remove the connecting copper foil either mechanically or with the assistance of heat. Once the etched copper trace is removed, the trace-cut jumper is turned into a solder-bridge jumper for any later changes.

A **solder-bridge** jumper is provided with two isolated pads that may be joined together. The silkscreen overlay printed around a solder-bridge jumper has gaps in the surround. Three methods can be used for joining:

- Solder may be applied to both pads to develop a bulge on each and the bulges joined by touching a soldering iron across the two pads.
- A small wire may be placed across the two pads and soldered in place.
- A SMT resistor, size 0805, 0603, or 0402, may be placed across the two pads and soldered in place. A zero-ohm resistor shorts the pads together.

For any copper jumper, the connection is considered **closed** if there is an electrical connection between the pads (default for trace-cut jumpers.) The connection is considered **open** if there is no electrical connection between the pads (default for the solder-bridge jumpers.)

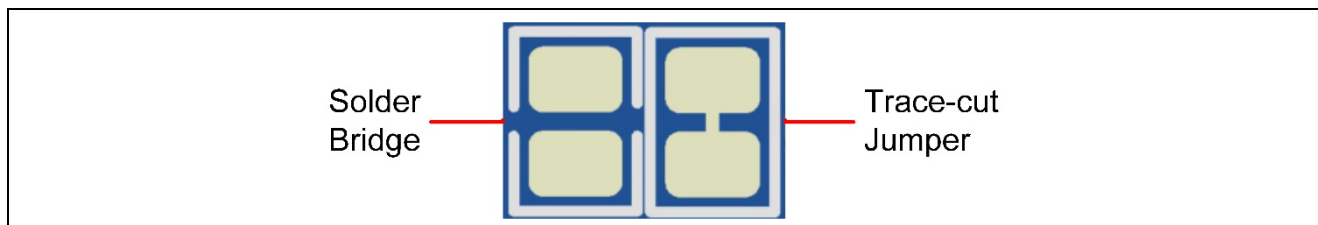


Figure 5. Copper Jumpers

4.3.2 Traditional Pin Header Jumpers

These jumpers are traditional small pitch jumpers that require an external shunt to open/close them. The traditional pin jumpers on the CK-RA6M5 board are 2.54 mm pitch headers and require compatible 2.54 mm shunt jumpers.

4.3.3 Default Jumper Configuration

The following table describes the default settings for each jumper on the CK-RA6M5 board. This includes copper jumpers (Ex designation) and traditional pin jumpers (Jx designation).

The Circuit Group for each jumper is the designation found in the board schematic (available in the Design Package). Functional details for many of the listed jumpers may be found in sections associated with each functional area of the kit.

Table 2. Default Jumper Settings

Location	Circuit Group	Default Open/Closed	Function
J1	Pmod 1	Open	Current Measurement point for Pmod 1
J2	MCU Power	Closed	Current Measurement point for MCU
J7	Pmod 2	Open	Current Measurement point for Pmod 2
J11	MCU Boot Mode	Open	Configures the MCU for normal boot mode
J15	JLOB	Open	Enable TrustZone support
J16	JLOB	Open	Enable use of external JTAG emulator
J17	HMI Power	Open	Connect external power source
J21	USBFS	Closed	Enable USB bus-powered operation
J22	USBFS	Link 2-3	USB Device Mode
E1	AREF	Closed	Links AREF to +3.3V
E3	Pmod 1	Open	Connects P512 (SCL1) to Pmod 1
E4	Pmod 1	Closed	Connects P202 (MISOA/RXD9) to Pmod 1
E5	Pmod 1	Open	Connects P511 (SDA1) to Pmod 1
E6	Pmod 1	Closed	Connects P204 (RSPCKA/SCK9) to Pmod 1
E7	Power	Closed	Connects JLOB to +3.3 V
E8	Pmod 2	Closed	Connects P412 (RSPCKB/SCK0) to Pmod 2
E9	Pmod 2	Open	Connects P613 (SDA7) to Pmod 2
E10	Pmod 2	Closed	Connects P410 (MISOB/RXD0) to Pmod 2
E11	Pmod 2	Open	Connects P614 (SCL7) to Pmod 2
E13	JTAG	Closed	Connects JTAG GND detect pin on J10 to GND

5. System Control and Ecosystem Access

The CK-RA6M5 provides a power supply regulator, an on-board debugger, simple I/O (switches and LEDs), and popular I/O ecosystem connectors. These are all described in detail below.

5.1 Power

The CK-RA6M5 kit is designed for +5 V operation. An on-board switching voltage regulator (ISL8002IRZ) is used to convert the 5 V supply to a 3.3 V supply. Also available is a low dropout regulator (ISL9008AIECZ-T) to convert the 3.3 V supply to a 1.8 V supply. The 3.3 V supply is used to power the RA MCU and other peripheral features as well as some of the sensors. The 1.8 V supply is used to power some of the motion and pressure sensors.

5.1.1 Power Supply Options

This section describes the different ways in which CK-RA6M5 kit can be powered.

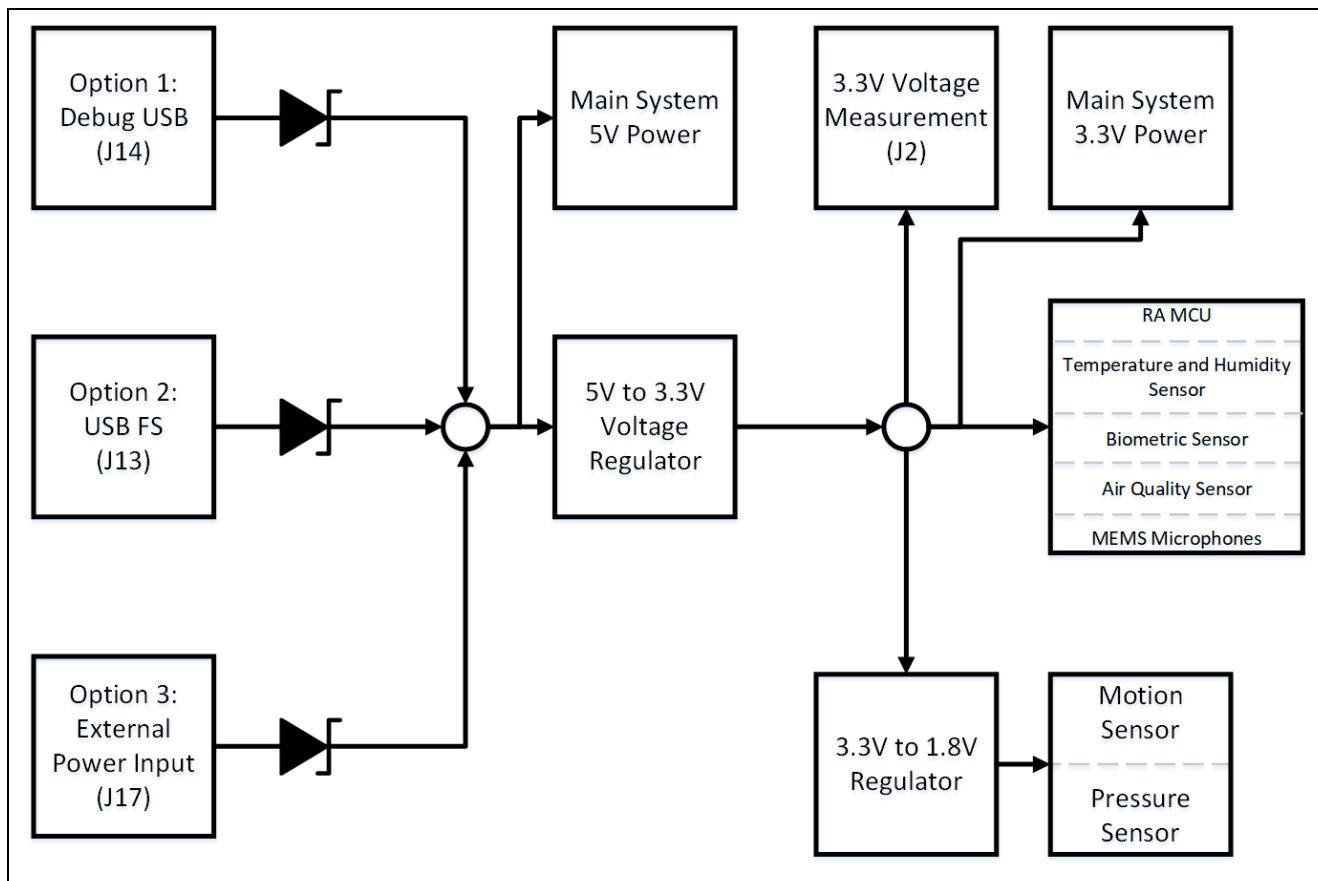


Figure 6. Power Supply Options

5.1.1.1 Option 1: Debug USB

5 V may be supplied from an external USB host to the USB Debug connector (J14) labelled DEBUG1 on the board. Power from this source is connected to the Main System 5 V Power. Reverse current protection is provided between this connector and the Main System 5 V Power.

5.1.1.2 Option 2: USB Full Speed

5 V may be supplied from an external USB host to the USB Full Speed connector (J13) labelled USB FS on the board. Power from this source is connected to the Main System 5 V Power. Reverse current protection is provided between this connector and the Main System 5 V Power.

5.1.1.3 Option 3: External Power Input

5 V may be supplied from an external power supply or battery to 2 pin header J17. Polarity is J17-2 (5 V) and J17-1 (GND) Pin headers, Power from this source is connected to the Main System 5 V Power. Reverse current protection is provided between the 5 V test points and the Main System 5 V Power.

5.1.2 Power Supply Considerations

The on-board regulator which supplies +3.3 V has a built-in current limit of 2.0 A and the regulator which supplies +1.8 V has a built-in current limit of 150 mA. Make sure that the total current required by the RA MCU, any active on-board features, and any connected peripheral device does not exceed these limits.

Note: The total current available from a typical USB host is 500 mA maximum. Depending on the configuration of the kit, multiple power sources may be required.

5.1.3 Power-up Behavior

When powered, the white LED near the center of the board (the “dash” in the CK-RA6M5 name) will light up. For more details on initial power up behavior, see the *CK-RA6M5 Quick Start Guide*.

5.2 Debug and Trace

The CK-RA6M5 board supports the following two debug modes.

Table 3. Debug Modes

Debug Modes	Debug MCU (one that connects to the IDE on PC)	Target MCU (one that is being debugged)	Debugging Interface/Protocol	Connector Used
Debug on-board	S124 (on-board)	RA6M5 (on-board)	SWD	Micro USB (J14)
Debug in	External debugging tools	RA6M5 (on-board)	SWD, ETM, JTAG	10-pin connector (J10)

Notes:

- See Table 5 for the Debug USB connector pin definition.
- See Table 8 for the 10-pin JTAG connector pin definition.

The following table summarizes the jumper setting for each of the debug modes.

Table 4. Jumper Connection Summary for Different Debug Modes

Debug Modes	J15	J16
Debug on-board	Closed	Open
Debug in	Closed	Closed

5.2.1 Debug On-Board

The on-board debug functionality is provided using Renesas S124 Debug MCU and SEGGER J-Link® firmware. Debug USB Micro-B connector (J14) connects the S124 Debug MCU to an external USB Full Speed Host, allowing re-programming and debugging of the target RA MCU firmware. This connection is the default debug mode for the CK-RA6M5 board.

The S124 Debug MCU connects to the target RA MCU using the SWD interface.

Table 5. Debug USB Connector

Debug USB Connector		CK-RA6M5
Pin	Description	Signal/Bus
J14-1	+5VDC	+5V_USB_DBG
J14-2	Data-	USB_DM (U9-12)
J14-3	Data+	USB_DP (U9-13)
J14-4	USB ID, jack internal switch, cable inserted	NC
J14-5	Ground	GND

A yellow indicator, LED5, shows the visual status of the debug interface. When the CK-RA6M5 board is powered on, and LED5 is blinking, it indicates that the S124 Debug MCU is not connected to a programming host. When LED5 is on solid, it indicates that the S124 Debug MCU is connected to a programming interface. When LED5 is flickering it indicates that there is active communication between the debugger and the host PC.

To configure the CK-RA6M5 board to use the Debug On-Board mode, configure the jumpers using the following table.

Table 6. Debug On-Board Jumper Configuration

Location	Default Open/Closed	Function
J15	Closed	Target RA MCU MD connected to debug
J16	Open	S124 Debug MCU in normal operation mode

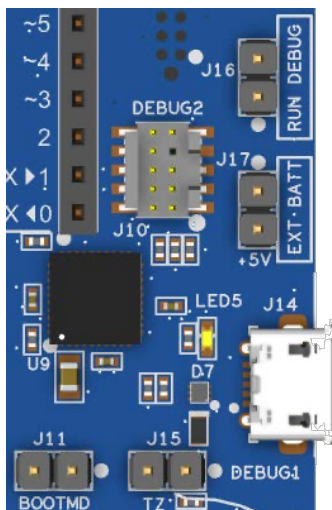


Figure 7. CK-RA6M5 Debug Interface

5.2.2 Debug In

One 10-pin Cortex® Debug Connector at J10 supports JTAG and SWD. This connector may be used for external debug of the target RA MCU.

To configure the CK-RA6M5 board to use the Debug In mode, configure the jumpers using the following table.

Table 7. Debug In Mode Jumper Configuration

Location	Default Open/Closed	Function
J15	Closed	Target RA MCU MD connected to debug
J16	Closed	S124 Debug MCU is held in RESET

Table 8. JTAG/SWD Connector

JTAG Connector			CK-RA6M5
Pin	JTAG Pin Name	SWD Pin Name	Signal/Bus
J10-1	Vtref	Vtref	+3V3
J10-2	TMS	SWDIO	P108/SWDIO
J10-3	GND	GND	GND
J10-4	TCK	SWCLK	P300/SWCLK
J10-5	GND	GND	GND
J10-6	TDO	SWO	P109/TDO
J10-7	Key	Key	NC
J10-8	TDI	NC/EXTb	P110/TDI
J10-9	GNDDetect	GNDDetect	GND (cut E13 to open)
J10-10	nSRST	nSRST	RESET#

Note: The Cortex® Debug Connector is fully described in the Arm® CoreSight™ Architecture Specification.

5.3 Ecosystem

The Ecosystem connectors provide users the option to simultaneously connect several third-party add-on modules compatible with two popular ecosystems using the following connectors:

1. Two Digilent Pmod™ (SPI and UART) connectors
2. Arduino™ (Uno R3) connectors

5.3.1 Digilent Pmod™ Connectors

Two 12-pin connectors are provided to support Pmod modules where the RA MCU acts as the master, and the connected module acts as a slave device.

These interfaces may be configured in firmware to support several Pmod types such as Type-2A (expanded SPI) and Type-3A (expanded UART).

The CK-RA6M5 board also provides jumpers so the 12-pin connector may alternatively be used for 2 independent 6-pin Pmod devices where the top row supports Pmod Type-2 (SPI) and Type-3 (UART) while the bottom row supports Pmod Type-6 (I²C).

The default 12-pin Pmod interface supports +3.3 V devices. Please ensure that any Pmod device installed is compatible with a +3.3 V supply.

The 6-pin Pmod interface option for Type-6 (I²C) may also support +5.0 V devices. Please ensure that only 6-pin modules are used when this option is selected.

5.3.1.1 Pmod 1 Connector

A 12-pin Pmod connector is provided at J4, Pmod 1.

Table 9. Pmod 1 Connector

Pmod 1 Connector Default			CK-RA6M5	Pmod 1 Configuration	
Pin	Description	Option Type 6A	Signal/Bus	Short	Open
J4-1	SSLA / CTS_RTS	NC	P205 (SSLA0/CTSRTS9)		
J4-2	MOSI / TXD	NC	P203 (MOSIA/TXD9)		
J4-3	MISO / RXD		P202 (MISOA/RXD9)	E4	E3
		SCL	P512 (SCL1)	E3	E4
J4-4	SCK/RTS		P204 (RSPCKA/SCK9)	E6	E5
		SDA	P511 (SDA1)	E5	E6
J4-5	GND		GND		
J4-6	VCC		+3.3 V	E1	E2
J4-7	GPIO / INT (slave to master)		P005 (IRQ10)		
J4-8	GPIO / RESET (master to slave)		P311		
J4-9	GPIO / CS1		P206 (SSLA1)		
J4-10	GPIO / CS2		P207 (SSLA2)		
J4-11	GND		GND		
J4-12	VCC		+3.3 V	E1	E2

Note: Exercise caution while modifying power source trace jumpers, E1 and E2. Permanent damage to the CK-RA6M5 board and/or connected modules may result.

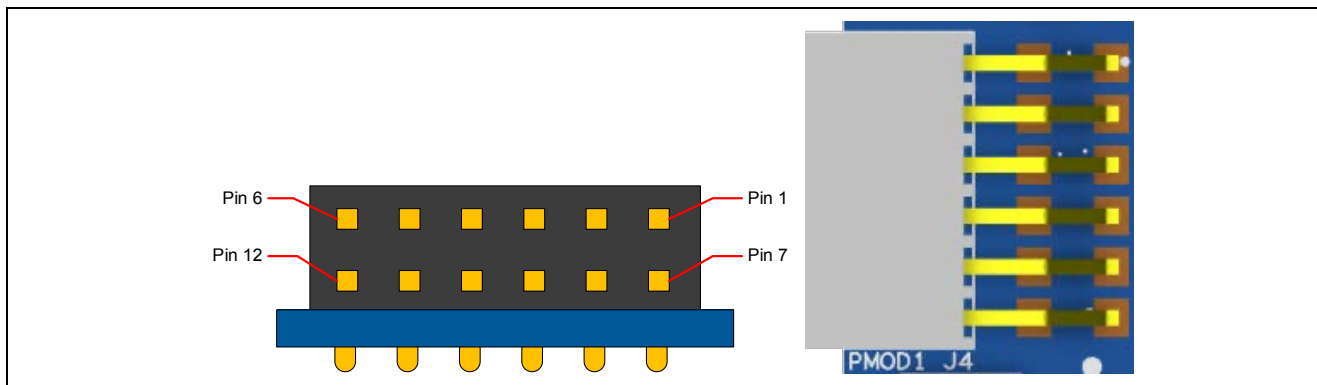


Figure 8. Pmod 1 Connector

5.3.1.2 Pmod 2 Connector

A 12-pin Pmod connector is provided at J5, Pmod 2.

Table 10. Pmod 2 Connector

Pmod 2 Connector Default			CK-RA6M5	Pmod 2 Configuration	
Pin	Description	Option Type 6A	Signal/Bus	Short	Open
J5-1	SS / CTS	NC	P413 (SSLB0/CTS0)		
J5-2	MOSI / TXD	NC	P411 (MOSIB/TXD0)		
J5-3	MISO / RXD		P410 (MISOB/RXD0)	E10	E11
		SCL	P614 (SCL7)	E11	E10
J5-4	SCK/RTS		P412 (RSPCKB/SCK0)	E8	E9
		SDA	P613 (SDA7)	E9	E8
J5-5	GND		GND		
J5-6	VCC		+3.3 V	E14	E12
J5-7	GPIO / INT (slave to master)		P004 (IRQ9)		
J5-8	GPIO / RESET (master to slave)		P409		
J5-9	GPIO / CS1		P414 (SSLB1)		
J5-10	GPIO / CS3		P708 (SSLB3)		
J5-11	GND		GND		
J5-12	VCC		+3.3 V	E14	E12

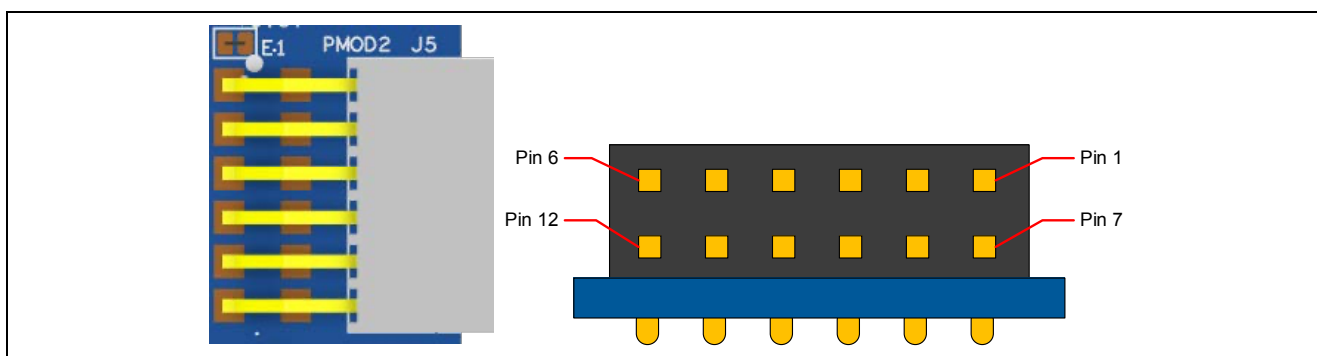


Figure 9. Pmod 2 Connector

5.3.2 Arduino™ Connector

Near the center of the System Control and Ecosystem Access area is an Arduino Uno R3 compatible connector interface.

Table 11. Arduino Uno Connections

	Arduino Compatible Connector	CK-RA6M5
Pin	Description	Signal/Bus
J6-1	NC	NC
J6-2	IOREF	+3.3 V
J6-3	RESET	P303
J6-4	3.3 V	+3.3 V
J6-5	5 V	+5 V
J6-6	GND	GND
J6-7	GND	GND
J6-8	VIN	NC
J9-1	A0	P000 (AN000)
J9-2	A1	P001 (AN001)
J9-3	A2	P002 (AN002)
J9-4	A3	P003 (AN003)
J9-5	A4	P014 (AN012)
J9-6	A5	P015 (AN013)
J8-1	D0 / RXD	P706 (RXD3)
J8-2	D1 / TXD	P707 (TXD3)
J8-3	D2 / INT0	P006 (IRQ11)
J8-4	D3 / INT1 / PWM	P111 (IRQ4/GTIOC3A)
J8-5	D4 / PWM	P415 (IRQ8/GTIOC0A)
J8-6	D5 / PWM	P105 (GTIOC1A)
J8-7	D6 / PWM	P104 (GTIOC1B)
J8-8	D7 / PWM	P115 (GTIOC4A)
J3-1	D8 / CLKOUT	P611
J3-2	D9 / PWM	P608 (GTIOC4B)
J3-3	D10 / SPI_SS / PWM	P503 (SSL6)
J3-4	D11 / SPI_MOSI / PWM	P506 (MOSI6)
J3-5	D12 / SPI_MISO	P505 (MISO6)
J3-6	D13 / SPI_SCK	P504 (SCK6)
J3-7	GND	GND
J3-8	AREF	VREFH / VREFH0 (linked to +3.3 V)
J3-9	I2C_SDA	P511 (SDA1)
J3-10	I2C_SCL	P512 (SCL1)

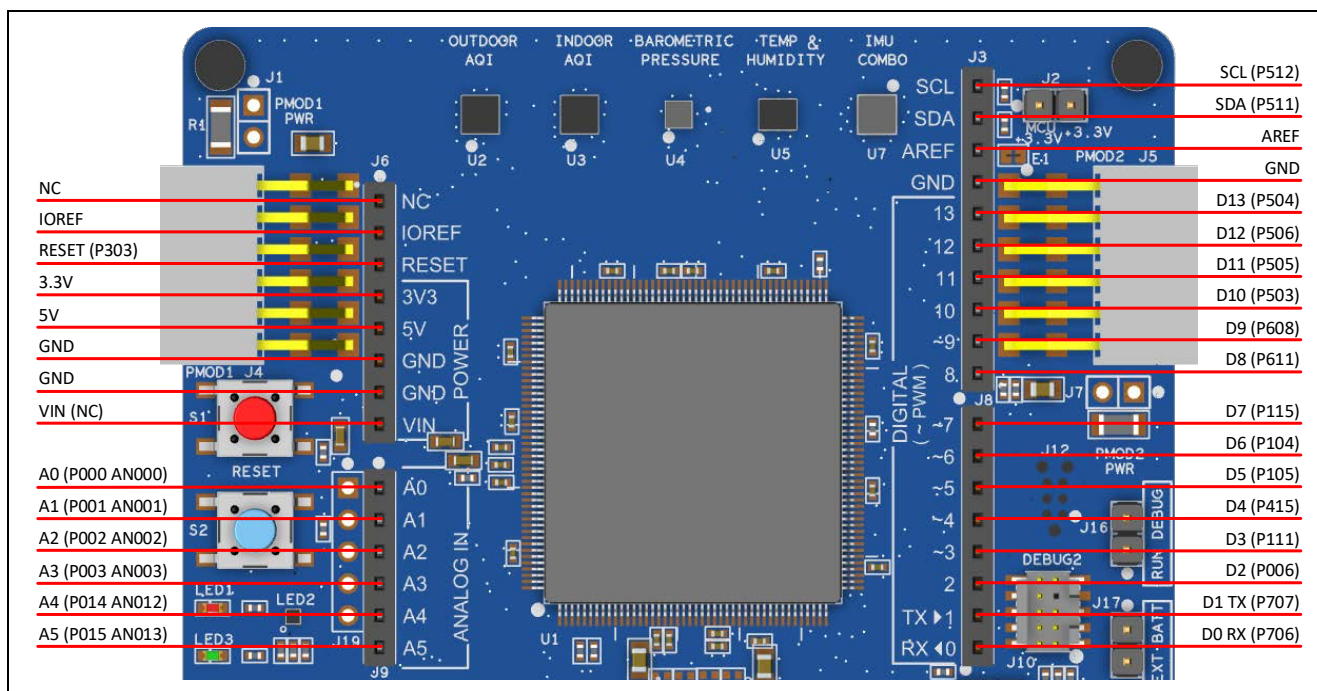


Figure 10. Arduino Uno Connectors

5.4 Connectivity

5.4.1 USB Full Speed

The USB Micro-AB connection jack (J13) connects the RA MCU USB Full Speed interface to an external USB interface, allowing communications for testing and use of the RA MCU firmware.

Table 12. USB Full Speed Connector

USB Full Speed Connector		CK-RA6M5
Pin	Description	Signal/Bus
J13-1	+5 VDC	+5V_USBFS, P407/USBFS_VBUS
J13-2	Data-	USB_DM
J13-3	Data+	USB_DP
J13-4	USB ID, jack internal switch, cable inserted	NC
J13-5	Ground	GND

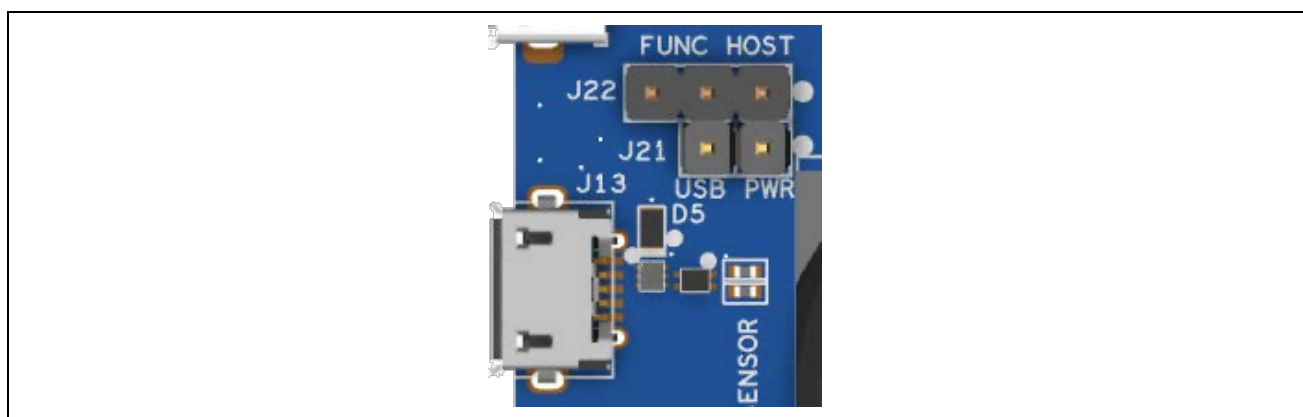


Figure 11. USB Full Speed Connector

5.5 Miscellaneous

5.5.1 User and Power LEDs

6 LEDs are provided on the CK-RA6M5 board. In addition, the Ethernet connector has built-in link status and link speed LEDs.

Behavior of the LEDs on the CK-RA6M5 board is described in the following table.

Table 13. CK-RA6M5 Board LED Functions

Designator	Color	Function	MCU Control Port
LED1	Red	User LED	P610
LED2	RGB	User LED	R:P602, G:P603, B:P605
LED3	Green	User LED	P609
LED4	Blue	User LED	P601
LED5	Yellow	Debug LED	J-Link OB MCU
LED6	White	Power on indicator	+3.3 V

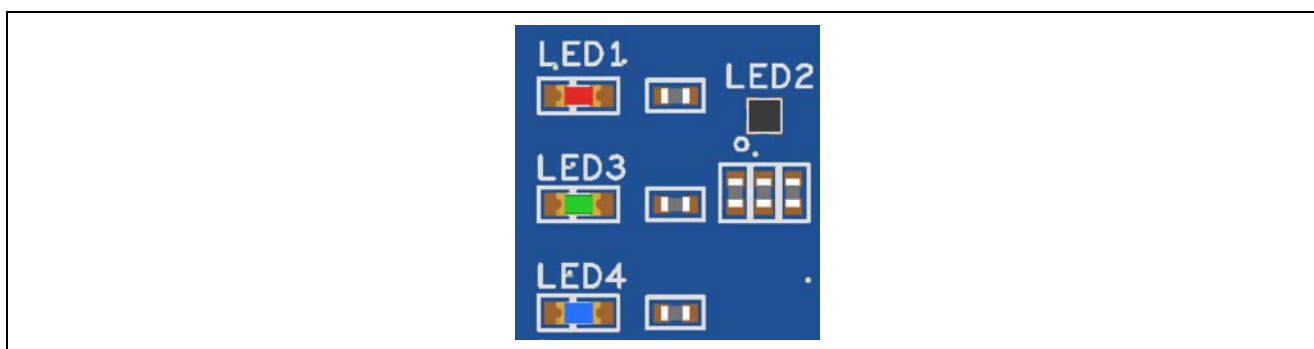


Figure 12. User LEDs



Figure 13. Power LED

5.5.2 User and Reset Switches

Two miniature, momentary, mechanical push-button type SMT switches are mounted on the CK-RA6M5 board.

Pressing the Reset switch (S1) generates a reset signal to restart the RA MCU.

Table 14. CK-RA6M5 Board Switches

Designator	Function	MCU Control Port	Button Color
S1	MCU Reset Switch	RESET#	Red
S2	User Switch	P804 (IRQ01)	Blue

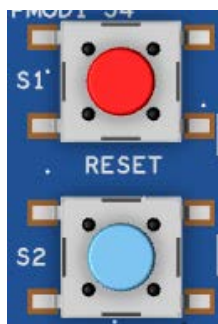


Figure 14. Reset and User Switches

5.5.3 MCU Boot Mode

A two-pin header (J11) is provided to select the Boot mode (P201) of the RA MCU. For normal operation, or Single-Chip mode, leave J11 open. To enter SCI Boot mode or USB Boot mode, place a jumper on J11.



Figure 15. Boot Mode

5.6 Ethernet

The Ethernet interface uses an RMII Ethernet Physical Layer Transceiver (PHY) (U11), connected to an RJ45 standard Ethernet connector (J18) with integrated magnetics and status indicators. The Ethernet clock is sourced from a precision 25 MHz clock crystal connected directly to the Ethernet PHY.

Table 15. Ethernet Port Assignments

Ethernet Signal Description	RA6M5 MCU Port
IRQ	P803 (IRQ02)
MDC	P214
MDIO	P211
TXD1	P406
TXD0	P700
TXD_EN	P405
RXD1	P703
RXD0	P702
CRS_DV	P705
RX_ER	P704
REF-50-CK	P701
RST#	P210

Table 16. Ethernet Components

Component	Manufacturer	Manufacturer Part Number
Ethernet PHY	Microchip	ICS1894-32
RJ45 Connector	Pulse	J0011D21BNL
25 MHz Oscillator	EuroQuartz	25.000 x22/30/30-40+85/12pF

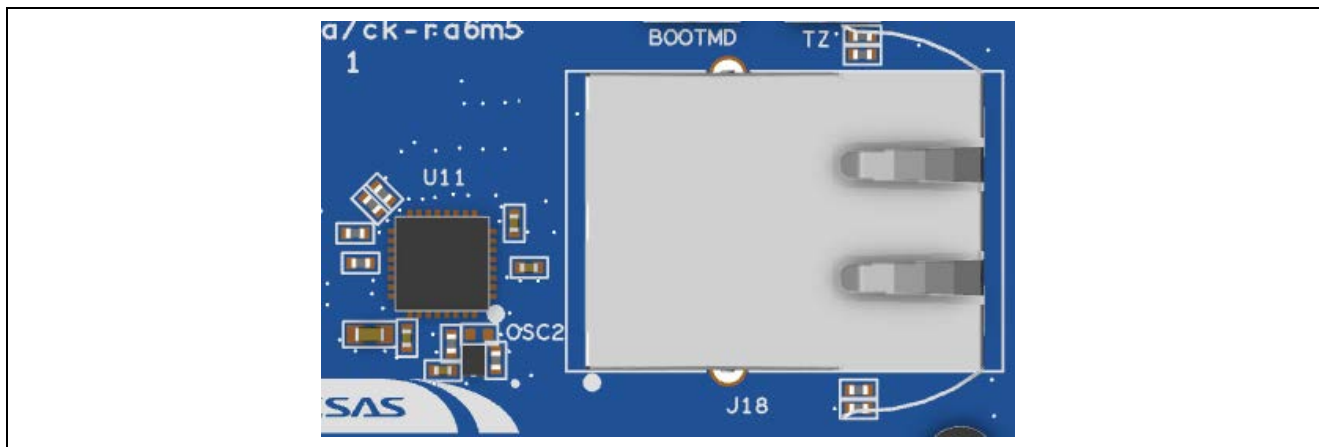


Figure 16. Ethernet Connector

5.7 Onboard Sensors

The CK-RA6M5 is fitted with a number of onboard sensors, including a barometric pressure sensor, 9-axis MEMS motion tracking sensor, indoor air quality sensor, outdoor air quality sensor, biometric sensor, temperature and humidity sensor.

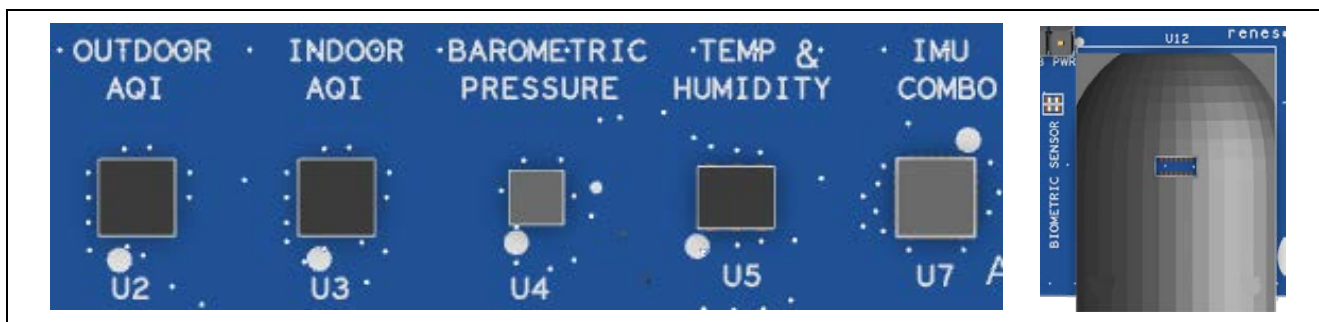


Figure 17. On Board Sensors

5.7.1 HS3001 Temperature and Humidity Sensor

An integrated calibration and temperature compensation logic provides fully corrected Relative Humidity and Temperature values via a standard I²C output. The measured data is internally corrected and compensated for accurate operation over a wide range of temperature and humidity levels.

Table 17. HS3001 Port Assignment

Pin Number	HS3001 Signal Description	RA6M5 MCU Port
1	SCL	P400
2	SDA	P401
3	VC	Connect a 0.1 μ F decoupling capacitor from VC to ground
4	VDD	3.3 V
5	NC	NC
6	VSS	GND

5.7.2 OB1203 Biometric Sensor

The OB1203 is a fully integrated optical sensing module for Pulse Oximetry (PPG - photoplethysmography), multi-channel Color Sensing (CS), Ambient Light Sensing (LS) and Proximity Sensing (PS). It can determine human heart rate, blood oxygen saturation (SpO2), respiration rate and heart rate variability (a measure of stress). The OB1203 integrates all light sources, drivers, and the sensor elements, in a single optically optimized package.

Table 18. OB1203 Port Assignment

Pin Number	OB1203 Signal Description	RA6M5 MCU Port
1	NC	NC
2	SCL	P400
3	SDA	P401
4	LGND	GND
5	LED1	TP1
6	LED2	TP2
7	LVDD	3.3 V
8	LVDD	3.3 V
9	LVDD	3.3 V
10	LVDD	3.3 V
11	VDD	3.3 V
12	AGND	GND
13	INT	P403 (IRQ14)
14	NC	NC

5.7.3 ZMOD4510 Outdoor Air Quality Sensor

The ZMOD4510 gas sensor platform detects air quality in a variety of indoor and outdoor applications. The signal conditioner controls the sensor temperature and measures the MOx resistance, which is a function of the gas concentration. The device is ideal for monitoring and reporting Outdoor Air Quality (OAQ).

Table 19. ZOMD4510 Port Assignment

Pin Number	ZMOD4510 Signal Description	RA6M5 MCU Port
1	SCL	P400
2	SDA	P401
3	INT	P404 (IRQ 15)
4	NC	NC
5	VDD	3.3 V
6	VSS	GND
7	VSS	GND
8	NC	NC
9	VSS	GND
10	VDDH	3.3 V
11	RES_N	P306
12	VDDIO	3.3 V

5.7.4 ZMOD4410 Indoor Air Quality Sensor

The ZMOD4410 Gas Sensor Module is designed for detecting total volatile organic compounds (TVOC) and monitoring indoor air quality (IAQ). The module's sense element consists of a heater element on a Si-based MEMS structure and a metal oxide (MOx) chemiresistor. The signal conditioner controls the sensor temperature and measures the MOx conductivity, which is a function of the gas concentration.

Table 20. ZMOD4410 Port Assignment

Pin Number	ZMOD4410 Signal Description	RA6M5 MCU Port
1	SCL	P400
2	SDA	P401
3	INT	P402 (IRQ4)
4	NC	NC
5	VDD	3.3 V
6	VSS	GND
7	VSS	GND
8	NC	NC
9	VSS	GND
10	VDDH	3.3 V
11	RES_N	P307
12	VDDIO	3.3 V

5.7.5 ICM-20948 9-AXIS MEMS Motion Tracking Sensor

The ICM-20948 is a low power 9-axis motion tracking sensor with 3-axis gyroscope, 3-axis accelerometer, and 3-axis compass, as well as Digital Motion Processor; offloading computation from the RA6M5 MCU improving system performance.

Table 21. ICM-20948 Port Assignment

Pin number	ICM-20948 Signal Description	RA6M5 MCU Port
1	NC	NC
2	NC	NC
3	NC	NC
4	NC	NC
5	NC	NC
6	NC	NC
7	AUX_CL	NC
8	VDDIO	GND via 100nF Capacitor
9	ADO/SD0	Pulldown to GND
10	REGOUT	GND via 100nF Capacitor
11	FSYNC	GND
12	INT1	P008 (IRQ12)
13	VDD	1.8 V
14	NC	NC
15	NC	NC
16	NC	NC
17	NC	NC
18	GND	GND
19	RESV_0	NC
20	RESV_1	GND
21	AUX_DA	NC
22	CS#	1.8 V
23	SCL	P400
24	SDA	P401

5.7.6 ICP-10101 Barometric Pressure Sensor

The ICP-10101 is based on MEMS capacitive technology which provides ultra-low noise at the lowest power, providing a high level of relative accuracy, sensor throughput, and temperature stability. The pressure sensor can measure pressure differences with an accuracy of ± 1 Pa, enabling altitude measurement differentials as small as 8.5 cm.

Table 22. ICP-10101 Port Assignment

Pin Number	ICP-10101 Signal Description	RA6M5 MCU Port
1	RESV	NC
2	SCL	P400
3	RESV	NC
4	SDA	P401
5	RESV	NC
6	RESV	NC
7	RESV	NC
8	GND	GND
9	GND	GND
10	VDD	1.8 V

5.7.7 SPH0645 MEMS Microphones

The SPH0645LM4H-B is a miniature, low power, bottom port microphone with an I²S digital output. The solution consists of a high-performance acoustic sensor, a serial Analog to Digital convertor, and an interface to condition the signal into an industry standard 24-bit I²S format.

Table 23. SPH0645 Port Assignment

Pin Number	SPH0645 Signal Description	RA6M5 MCU Port
1	WS	P113 (SSIE0_WS)
2	SELECT	3.3 V
3	GND	GND
4	BCLK	P112 (SSIE0_SCK)
5	VDD	3.3 V
6	DATA-OUT	P114 (SSIE0_RXD)

6. MCU Native Access Area

6.1 MCU and Pmod Current Measurement

Included in the Native Pin Access area are current measurement resistors and test points to measure the MCU USB controller current and the MCU core power current.

The CK-RA6M5 board has pin header test points for current measurement of the main 3.3 V MCU power and for Pmod power. To measure Pmod current remove 0 Ohm resistor and install pin header. Alternatively use a resistor of known value and measure the voltage drop across the resistor and use Ohm's Law to calculate the current. This process is not necessary for MCU power, because there is no resistor to be removed. J2 can be used to measure the MCU power, J1 can be used to measure Pmod 1 power, and J7 can be used to measure Pmod 2 power.

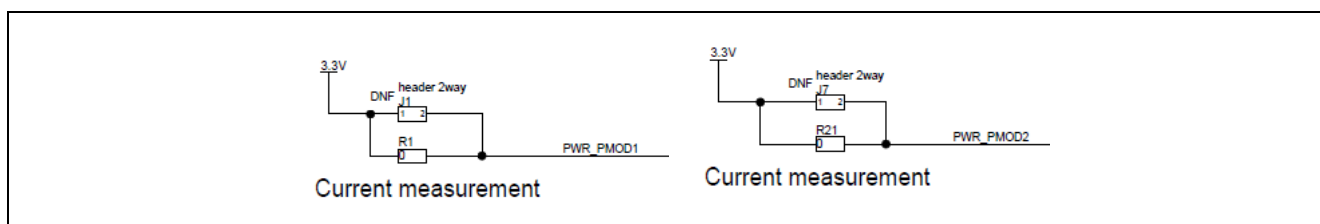


Figure 18. RA Pmod Current Measurement Circuit

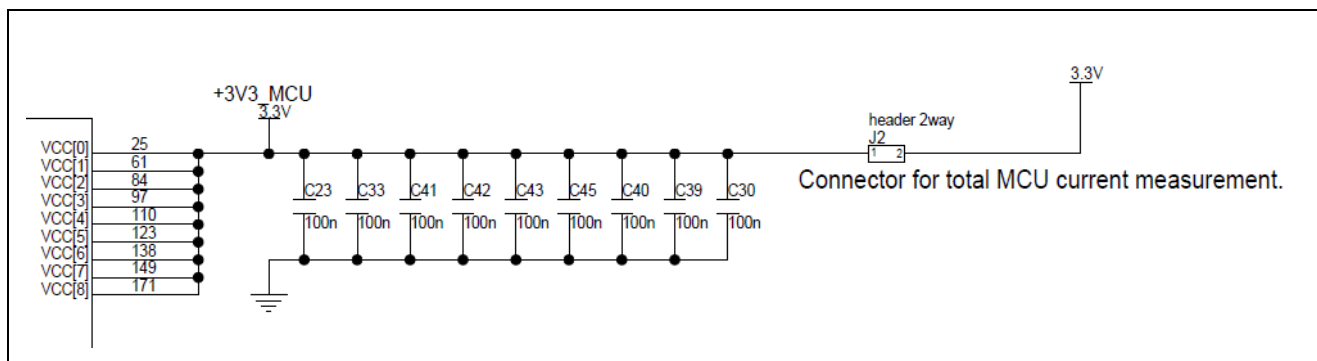


Figure 19. RA +3.3 V Current Measurement Circuit

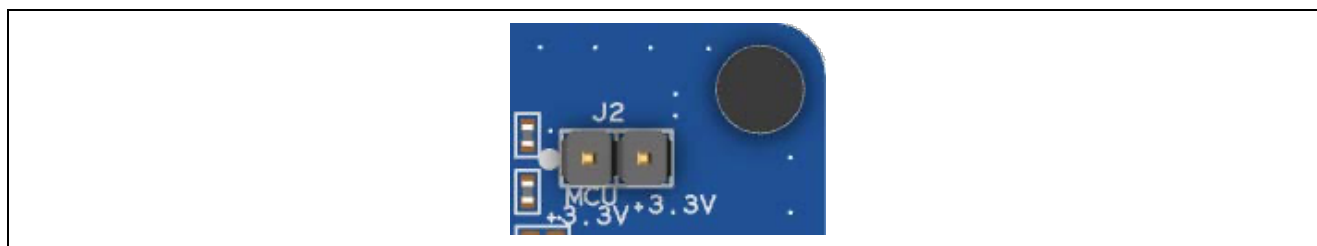


Figure 20. RA MCU Current Measurement

7. Recommended Parts

The below table lists recommended part numbers for optional components that can be fitted as required.

Table 24. Part Numbers

Designator(s)	Description	Manufacturer	Part Number
C18, C19	0402 Capacitor	YAGEO	CC0402JRNPO9BN150
C20, C21	0402 Capacitor	YAGEO	CC0402JRNPO9BN270
J1, J7	1 Row 2 Way header	Wuerth Elektronik	613 002 111 21
X1	24MHz Crystal	Diodes Inc.	FL2400022

8. Certifications

The CK-RA6M5 v1 kit meets the following certifications/standards. See page 3 of this user's manual for the disclaimer and precautions.

8.1 EMC/EMI Standards

- FCC Notice (Class A)



This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

NOTE- This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/television technician for help.

- Innovation, Science and Economic Development Canada ICES-003 Compliance:

CAN ICES-3 (A)/NMB-3(A)

- CE Class A (EMC)



This product is herewith confirmed to comply with the requirements set out in the Council Directives on the Approximation of the laws of the Member States relating to Electromagnetic Compatibility Directive 2014/30/EU.

Warning – This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures to correct this interference.

- UKCA Class A (EMC)



This product is in conformity with the following relevant UK Statutory Instrument(s) (and its amendments): 2016 No. 1091 Electromagnetic Compatibility Regulations 2016.

Warning – This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures to correct this interference.

- Taiwan: Chinese National Standard 13438, C6357 compliance, Class A limits
- Australia/New Zealand AS/NZS CISPR 32:2015, Class A

8.2 Material Selection, Waste, Recycling and Disposal Standards

- EU RoHS
- China SJ/T 113642014, 10-year environmental protection use period.

8.3 Safety Standards

- UL 94V-0

9. Design and Manufacturing Information

The design and manufacturing information for the CK-RA6M5 v1 kit is available in the “CK-RA6M5v1 Design Package” available on renesas.com/ra/ck-ra6m5.

- Design package file name: ck-ra6M5-v1-designpackage.zip
- Design package contents

Table 25. CK-RA6M5 Board Design Package Contents

File Type	Content	File/Folder Name
File (PDF)	Schematics	ck-ra6m5-v1-schematics
File (PDF)	Mechanical Drawing	ck-ra6m5-v1-mechdwg
File (PDF)	3D Drawing	ck-ra6m5-v1-3d
File (PDF)	BOM	ck-ra6m5-v1-bom
Folder	Manufacturing Files	Manufacturing Files
Folder	Design Files	Design Files-Cadence Allegro

10. Website and Support

Visit the following URLs to learn about the kit and the RA family of microcontrollers, download tools and documentation, and get support.

CK-RA6M5 Resources	renesas.com/ra/ck-ra6m5
RA Kit Information	renesas.com/ra/kits
RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Videos	renesas.com/ra/videos
RA Kit Feedback and Feature Request	renesas.com/ra/kitfeedback
Renesas Support	renesas.com/support

Provide Feedback/Request a Feature

Renesas aims to provide the best microcontroller kit experience to help our customers jumpstart innovation and take products to market faster with the RA family of microcontrollers. The Renesas RA microcontroller kits have been designed with a lot of attention to detail and customer-centered thinking in every aspect of the design. Renesas aims to exceed customer expectation.

Renesas looks forward to hearing your feedback and learning how we can enhance your experience. Please share your feedback at renesas.com/ra/kitfeedback.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00		—	Initial release
1.10	Jan.20.22	—	Updated figure 4, figure 6, and corrected various typos
2.00	Jan.20.22	—	Updated for WS2 release
3.00	May.03.22	—	Updated for MP release
3.10	Jan.16.23	20	Corrected ethernet phy part number

CK-RA6M5 v1 with RYZ014A Pmod – User's Manual

Publication Date: Jan.16.23

Published by: Renesas Electronics Corporation

CK-RA6M5 v1 with RYZ014A Pmod – User's Manual