ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ | ΤΜΗΜΑ ΗΜ&ΜΥ

# ΣΧΕΔΙΑΣΜΟΣ ΕΝΣΩΜΑΤΩΜΕΝΩΝ ΣΥΣΤΗΜΑΤΩΝ

Τελική εργασία – 2ο μέρος – 19/12/2025


## Στοιχεία Φοιτητών 9ου Εξαμήνου – Ομάδα 19

Βαϊντερλής Άγγελος, 58647

Κοκορότσικου Αγνή Ιωάννα, 58767

Ξανθόπουλος Ηλίας, 58545


## GitHub Page

AgniKoko/embedded-systems-design-FIR-MSP

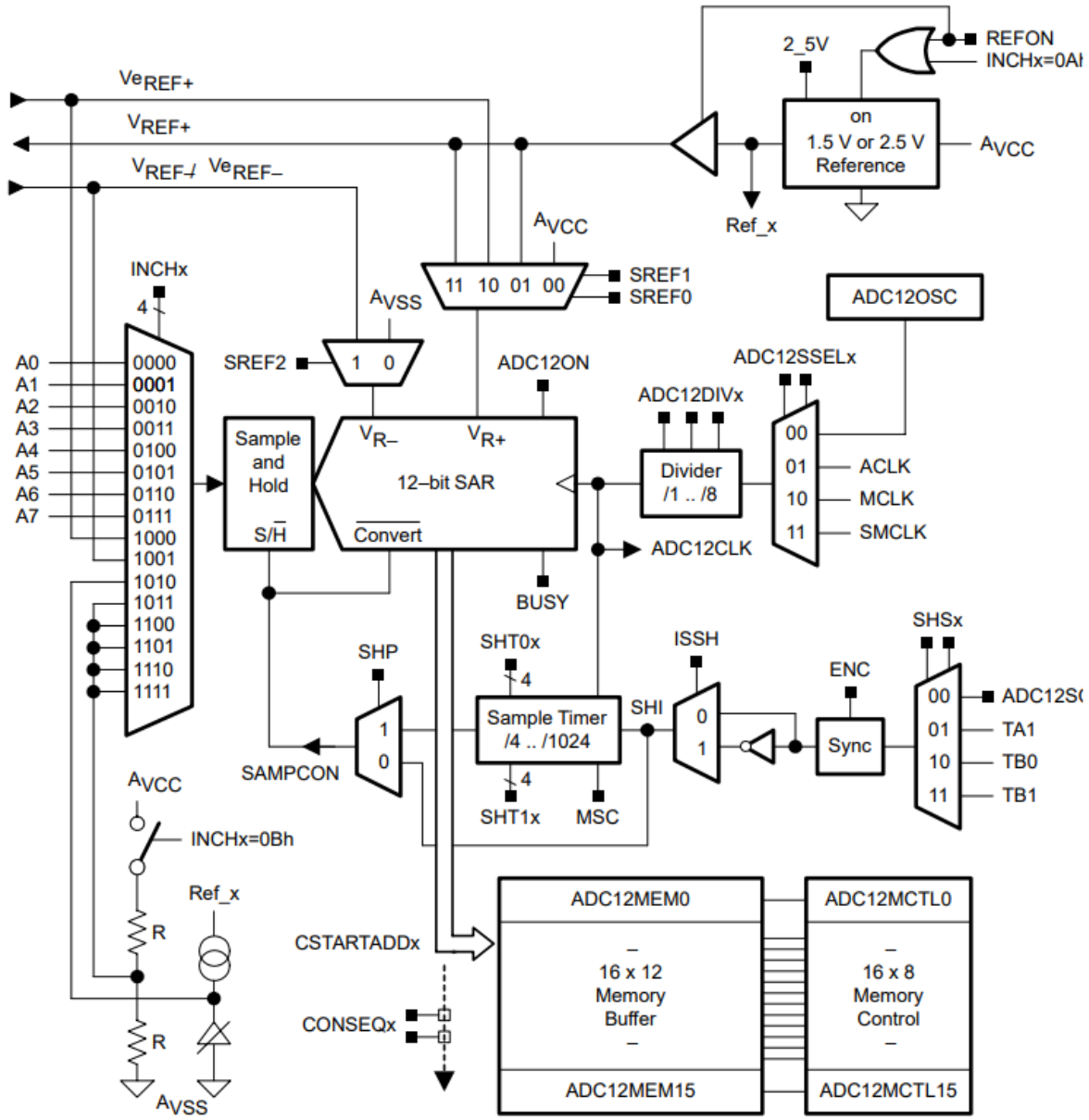

## Επιβλέπων Καθηγητής

Δρ. Ιωάννης Βούρκας

# Περιεχόμενα

# 1. Εισαγωγή

Στην παρούσα εργασία χρησιμοποιείται ο μικροελεγκτής MSP430 της οικογένειας x1xx, ο οποίος ενσωματώνει 12-bit αναλογικό σε ψηφιακό μετατροπέα (ADC12) και χρονιστή Timer_A. Ο ADC12 χρησιμοποιείται για την περιοδική δειγματοληψία ενός αναλογικού σήματος στην είσοδο P6.1 (κανάλι A1), ενώ ο Timer_A αναλαμβάνει τον ρόλο μίας γεννήτριας ρολογιού δειγματοληψίας, καθορίζοντας κάθε πότε θα ξεκινάει μία νέα μετατροπή. Τα δείγματα που παράγονται τροφοδοτούν έναν διακριτό FIR φίλτρο (γραμμική συνέλιξη), το οποίο υλοποιείται στον πυρήνα του MSP430.




*Εικόνα 1 – (MSP430x1xx User's Guide, p. 17.3)*

# 2. Θεωρητικό υπόβαθρο

### 2.1. Λειτουργία και ρυθμίσεις του ADC12

Ο ADC12 είναι μετατροπέας 12-bit τύπου successive approximation (SAR). Για κάθε μετατροπή, το σήμα δειγματοληπτείται πρώτα σε έναν πυκνωτή sample-and-hold και στη συνέχεια συγκρίνεται διαδοχικά με επιλεγμένα επίπεδα αναφοράς, ώστε να παραχθεί ένας 12-bit ψηφιακός κωδικός. (MSP430x1xx User's Guide, p. 17.2)

Στην εφαρμογή χρησιμοποιείται ως θετική και αρνητική αναφορά η τροφοδοσία του μικροελεγκτή, κάτι που υλοποιείται με τη ρύθμιση SREF_0 στο register ADC12MCTL1. Έτσι, για τροφοδοσία $V_{dd} \cong 3.3V$, η είσοδος κβαντίζεται στο διάστημα [0, 3.3 V].

Η επιλογή καναλιού εισόδου γίνεται επίσης μέσω του ADC12MCTL1, όπου τίθεται INCH_1, δηλώνοντας ότι η μετατροπή αφορά το κανάλι A1 (P6.1). Παράλληλα, με τη ρύθμιση CSTARTADD_1 στο ADC12CTL1 ορίζεται ότι τα αποτελέσματα των μετατροπών θα αποθηκεύονται στο ADC12MEM1. (MSP430x1xx User's Guide, pp. 17.20-17.23)

Η παραμετροποίηση του ADC12 γίνεται στους καταχωρητές ADC12CTL0 και ADC12CTL1:

1. **Register: ADC12CTL0:**
   - ADC12ON: ενεργοποίηση της μονάδας ADC.
   - SHT0_4: επιλογή χρόνου δειγματοληψίας για τα κανάλια ADC12MEM0–7. Στη συγκεκριμένη ρύθμιση αντιστοιχούν 64 κύκλοι του ρολογιού ADC12CLK.
2. **Register: ADC12CTL1:**
   - ADC12SSEL_2: επιλογή ρολογιού μετατροπής από το MCLK.
   - ADC12DIV_3: διαίρεση του MCLK δια 4 πριν τροφοδοτηθεί ο ADC (ADC12CLK = MCLK/4).
   - CONSEQ_0: λειτουργία single-channel, single-conversion, δηλαδή κάθε φορά εκτελείται μία μόνο μετατροπή σε ένα κανάλι.
   - SHS_0: η μετατροπή ξεκινά με software, μέσω του bit ADC12SC.
   - SHP: ενεργοποίηση του εσωτερικού sample timer, ώστε ο ADC να φροντίζει αυτόματα για τον χρόνο δειγματοληψίας.

### 2.2. Λειτουργία και ρυθμίσεις του Timer A

Ο Timer_A είναι ένας 16-bit μετρητής που μπορεί να λειτουργήσει σε διάφορα modes. Στο συγκεκριμένο project χρησιμοποιείται σε up mode, με πηγή ρολογιού τον ACLK:

- TASSEL_1: επιλογή ρολογιού ACLK.
- MC_1: λειτουργία up mode, όπου ο μετρητής αυξάνει από 0 έως TACCR0 και στη συνέχεια επανεκκινεί.

Με κατάλληλη τιμή στο TACCR0 καθορίζεται η περίοδος δειγματοληψίας. Για παράδειγμα, με $ACLK \cong 32.768Hz$ και TACCR0 = 500, η συχνότητα των Timer_A interrupts είναι περίπου 65 Hz. Σε κάθε interrupt του Timer_A εκτελείται η αντίστοιχη ISR, στην οποία τίθεται το bit ADC12SC στο ADC12CTL0. (MSP430x1xx User's Guide, pp. 11.2-11.3)

# 3. Interrupt Service Routines (ISR)

Ο μικροελεγκτής MSP430 υποστηρίζει μηχανισμό διακοπών (interrupts), μέσω του οποίου περιφερειακά όπως οι timers, μονάδες ADC ή γραμμές εισόδου/εξόδου μπορούν να διακόψουν την «κανονική» ροή εκτέλεσης της CPU και να καλέσουν ειδικές ρουτίνες εξυπηρέτησης διακοπών (ISR). Για την συγκεκριμένη άσκηση χρησιμοποιηθήκαν 3 διαφορετικές ρουτίνες, οι οποίες αναλύονται παρακάτω. (MSP430x1xx User's Guide, pp. 17.18-17.19)

### 3.1. Port 1 ISR: Enable/Disable του συστήματος

Το Pin 0 του Port 1 χρησιμοποιείται ως είσοδος "Enable", με ενεργοποίηση διακοπής από το Low σε High. Η αντίστοιχη ISR καλείται κάθε φορά που ανιχνεύεται ένα τέτοιο γεγονός (π.χ. πάτημα κουμπιού). Στο εσωτερικό της:

- Γίνεται toggle της μεταβλητής system_enabled.
- Ανάλογα με τη νέα κατάσταση, ενημερώνεται και η έξοδος LED στην P1.7 (ON όταν το σύστημα είναι Enabled, OFF όταν είναι Disabled), παρέχοντας οπτική ένδειξη στον χρήστη.
- Τέλος, το flag P1IFG καθαρίζεται, ώστε να μη δημιουργείται συνεχώς η ίδια διακοπή.

### 3.2. ADC12 ISR: Αποθήκευση του δείγματος

Η μονάδα ADC12 διαθέτει πολλαπλές πηγές διακοπών, οι οποίες ομαδοποιούνται σε έναν ενιαίο interrupt vector ADC12_VECTOR. Στην εργασία αυτή, χρησιμοποιείται μόνο η σημαία ADC12IFG1, η οποία τίθεται όταν ολοκληρωθεί η μετατροπή και γραφτεί αποτέλεσμα στο ADC12MEM1. Με την ενεργοποίηση του ADC12IE για το αντίστοιχο κανάλι, κάθε ολοκληρωμένη μετατροπή προκαλεί κλήση της ISR ADC12_ISR.

Μέσα στην ISR:

- Διαβάζεται το αποτέλεσμα από το ADC12MEM1.
- Αν το σύστημα είναι ενεργό (system_enabled == true), η τιμή μετατρέπεται σε float και αποθηκεύεται στη θέση signal_buffer[write_index] του κυκλικού buffer.
- Αυξάνονται οι μεταβλητές write_index και num_samples, με wrap-around όταν ο buffer γεμίσει, ώστε να διατηρείται πάντα το τελευταίο παράθυρο δειγμάτων σταθερού μήκους.
- Τίθεται η σημαία adc_new_sample = true, ώστε το main loop να γνωρίζει ότι υπάρχει νέο δείγμα προς επεξεργασία.
- Τέλος, καλείται η intrinsic __low_power_mode_off_on_exit(), η οποία φροντίζει ώστε, κατά την έξοδο από το interrupt, η CPU να «ξυπνήσει» από τη LPM0 και να συνεχίσει την εκτέλεση στο main (εκεί όπου θα πραγματοποιηθεί η συνέλιξη).

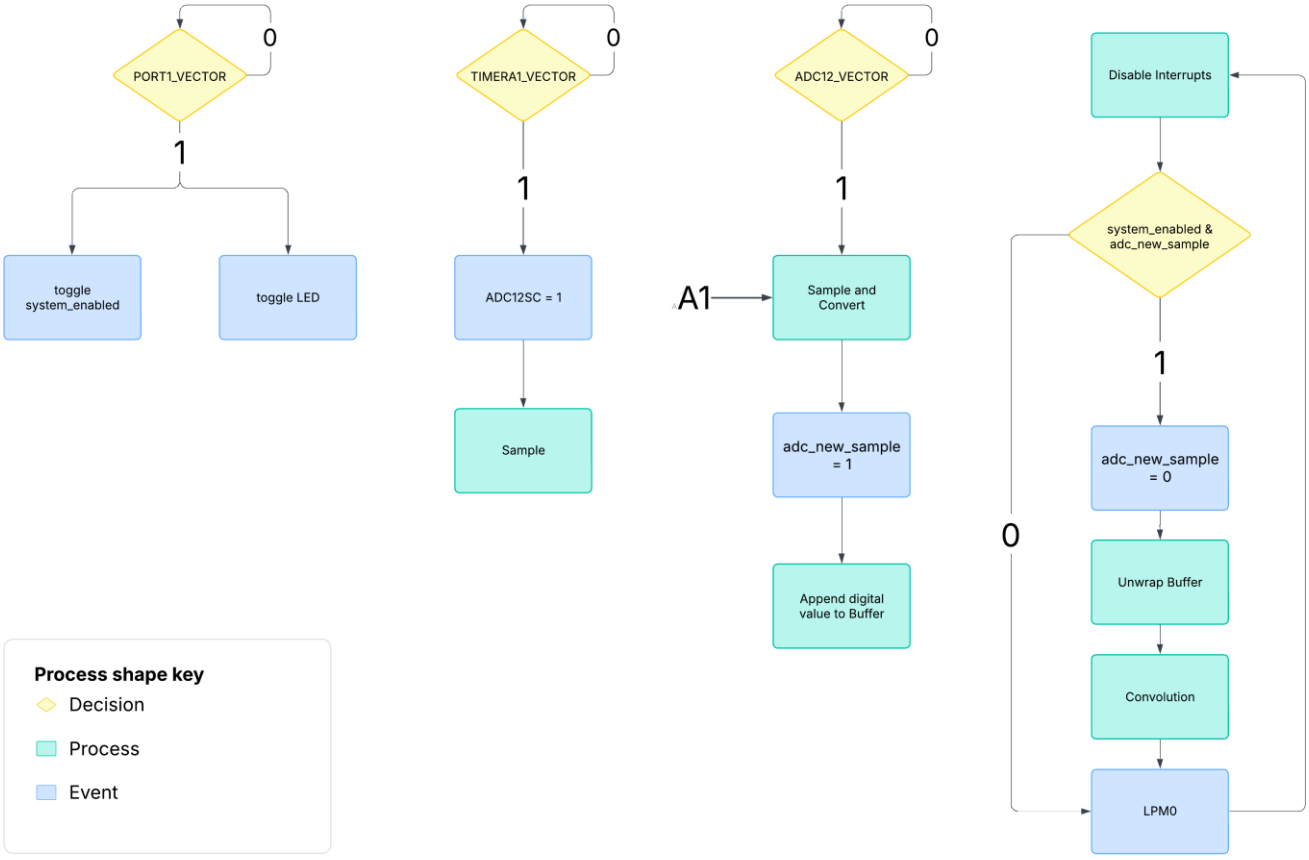
### 3.3. Timer_A ISR: Έναρξη νέας μετατροπής ADC

Ο Timer_A είναι ρυθμισμένος σε up mode με πηγή ρολογιού τον ACLK, ώστε να παράγει περιοδικά interrupts με περίοδο που ορίζεται από την τιμή του TACCR0. Η ISR του Timer εκτελείται κάθε φορά που συμπληρώνεται η περίοδος του Timer. Μέσα σε αυτή τη ρουτίνα, εφόσον η μεταβλητή κατάστασης system_enabled είναι true, τίθεται το bit ADC12SC στο ADC12CTL0. Με τον τρόπο αυτό ξεκινά μία νέα μετατροπή στον ADC12. (MSP430x1xx User's Guide, p. 17.10)

Άρα, η ISR του Timer_A λειτουργεί ως ένα ρολόι δειγματοληψίας του συστήματος: καθορίζει κάθε πότε θα ζητηθεί νέα μέτρηση από τον ADC, χωρίς η κύρια ροή του προγράμματος (main loop) να χρειάζεται να ασχολείται με χρονομέτρηση.

# 4. Υπολογισμός Συνέλιξης

Στο κύριο βρόχο υλοποιείται η ακολουθία “νέο δείγμα → επεξεργασία → αναμονή”. Αρχικά, απενεργοποιούνται τα interrupts, ώστε κατά τη διάρκεια της αριθμητικής επεξεργασίας να μην προκύψει νέα διακοπή από τον ADC και να μην αλλοιωθεί ο κυκλικός buffer. Στη συνέχεια ελέγχεται η συνθήκη system_enabled && adc_new_sample. Δηλαδή, το σύστημα πρέπει να είναι ενεργό και να έχει φτάσει νέο δείγμα. Αν η συνθήκη ισχύει, το flag adc_new_sample μηδενίζεται, ο κυκλικός buffer ξετυλίγεται σε γραμμικό πίνακα signal_linear[] με σωστή χρονολογική σειρά και καλείται η συνάρτηση convolve(), η οποία υπολογίζει τη γραμμική συνέλιξη signal–kernel και αποθηκεύει το αποτέλεσμα στον πίνακα result[]. Αφού ολοκληρωθεί η επεξεργασία, τα interrupts ενεργοποιούνται ξανά, ώστε να μπορούν να εξυπηρετηθούν οι επόμενες μετατροπές, και η CPU επιστρέφει σε χαμηλή κατανάλωση (LPM0), περιμένοντας την επόμενη διακοπή από τον ADC για να ξεκινήσει ο ίδιος κύκλος από την αρχή.




*Εικόνα 2 – Flowchart της λογικής εκτέλεσης*

Σημειώνεται ότι για να μην χαθούν interrupt requests, η περίοδος δειγματοληψίας πρέπει να είναι μεγαλύτερη από τον χρόνο όπου τα interrupts είναι απενεργοποιημένα μέσα στη while, δηλαδή πρέπει $T_{sampling} > T_{disable}$. Για την μέτρηση του $T_{disable}$ χρησιμοποιήθηκε ο register CCTIMER1 σύμφωνα με τον οποίο, για SIGNAL_LEN = 16, χρειάστηκαν 26533 steps. Άρα για συχνότητα ρολογιού 800kHz έχουμε $T_{disabled} = 33.2ms$. Άρα πρέπει:

$$f_{sampling} < f_{disable} = \frac{1}{33.2ms} = 30.15Hz$$

Watch 1

| Expression | Value | Location | Type |
|---|---|---|---|
| signal_b... | <array> | Memory:0x1100 | float[16] |
| [0] | 1.0 | Memory:0x1100 | float |
| [1] | 2.0 | Memory:0x1104 | float |
| [2] | 3.0 | Memory:0x1108 | float |
| [3] | 4.0 | Memory:0x110C | float |
| [4] | 5.0 | Memory:0x1110 | float |
| [5] | 6.0 | Memory:0x1114 | float |
| [6] | 7.0 | Memory:0x1118 | float |
| [7] | 8.0 | Memory:0x111C | float |
| [8] | 9.0 | Memory:0x1120 | float |
| [9] | 16.0 | Memory:0x1124 | float |
| [10] | 1.0 | Memory:0x1128 | float |
| [11] | 2.0 | Memory:0x112C | float |
| [12] | 4.0 | Memory:0x1130 | float |
| [13] | 3.0 | Memory:0x1134 | float |
| [14] | 6.0 | Memory:0x1138 | float |
| [15] | 7.0 | Memory:0x113C | float |
| signal_li... | <array> | Memory:0x1142 | float[16] |
| [0] | 1.0 | Memory:0x1142 | float |
| [1] | 2.0 | Memory:0x1146 | float |
| [2] | 3.0 | Memory:0x114A | float |
| [3] | 4.0 | Memory:0x114E | float |
| [4] | 5.0 | Memory:0x1152 | float |
| [5] | 6.0 | Memory:0x1156 | float |
| [6] | 7.0 | Memory:0x115A | float |
| [7] | 8.0 | Memory:0x115E | float |
| [8] | 9.0 | Memory:0x1162 | float |
| [9] | 16.0 | Memory:0x1166 | float |
| [10] | 1.0 | Memory:0x116A | float |
| [11] | 2.0 | Memory:0x116E | float |
| [12] | 4.0 | Memory:0x1172 | float |
| [13] | 3.0 | Memory:0x1176 | float |
| [14] | 6.0 | Memory:0x117A | float |
| [15] | 7.0 | Memory:0x117E | float |
| result | <array> | Memory:0x1182 | float[20] |
| [0] | 2.000000... | Memory:0x1182 | float |
| [1] | 6.000000... | Memory:0x1186 | float |
| [2] | 1.20000004 | Memory:0x118A | float |
| [3] | 2.0 | Memory:0x118E | float |
| [4] | 3.0 | Memory:0x1192 | float |
| [5] | 4.0 | Memory:0x1196 | float |
| [6] | 5.0 | Memory:0x119A | float |
| [7] | 6.0 | Memory:0x119E | float |
| [8] | 7.0 | Memory:0x11A2 | float |
| [9] | 9.19999981 | Memory:0x11A6 | float |
| [10] | 8.19999981 | Memory:0x11AA | float |
| [11] | 7.20000028 | Memory:0x11AE | float |
| [12] | 6.40000009 | Memory:0x11B2 | float |
| [13] | 5.20000028 | Memory:0x11B6 | float |
| [14] | 3.20000004 | Memory:0x11BA | float |
| [15] | 4.40000009 | Memory:0x11BE | float |
| [16] | 4.0 | Memory:0x11C2 | float |
| [17] | 3.20000004 | Memory:0x11C6 | float |
| [18] | 2.5999999 | Memory:0x11CA | float |
| [19] | 1.39999997 | Memory:0x11CE | float |
| system_... | 1 | Memory:0x11D2 | bool |
| adc_new... | 0 | Memory:0x11D3 | bool |
| write_ind... | 0 | Memory:0x1140 | unsigned int |
| TACTL | 274 | Memory:0x160 | unsigned short |

*Εικόνα 3 – Στιγμιότυπο των αποτελεσμάτων εκτέλεσης της συνέλιξης*

# 5. Βιβλιογραφία

[1] Instruments, T. (2003). *MSP430x1xx User's Guide*. Dallas, TX: Instruments, Texas.