Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2023-2024

ΑΘΗΝΑ, 3 Νοεμβρίου 2023

5η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Χρήση εξωτερικών Θυρών Επέκτασης στον AVR

Εξέταση – Επίδειξη: Παρασκευή 10/11/2022
Προθεσμία για παράδοση Έκθεσης: Τρίτη 14/11/2022 (23:59)

Διεπαφή TWI (Two wire Serial Interface)

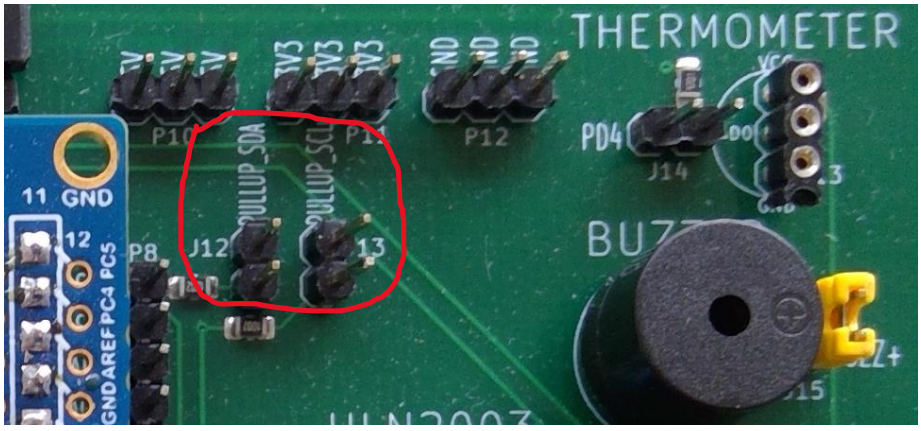
Το πρωτόκολλο TWI επιτρέπει τη διασύνδεση έως και 127 διαφορετικές συσκευών χρησιμοποιώντας μόνο δύο αμφίδρομες γραμμές διαύλου, μία για ρολόι (SCL) και μία για δεδομένα (SDA). Τα μόνα εξωτερικά υλικά που χρειάζονται για την υλοποίηση του διαύλου είναι δύο αντιστάσεις πρόσδεσης (pull-up resistors) για τις γραμμές του διαύλου TWI.

- Όλες οι συσκευές που είναι συνδεδεμένες στο δίαυλο TWI έχουν ατομικές διευθύνσεις.
- Το πρωτόκολλο TWI ενσωματώνει μηχανισμό αποφυγής σύγκρουσης δεδομένων στο δίαυλο.
- Το πρωτόκολλο TWI είναι συμβατό με το πρωτόκολλο I^2C της Philips.

Διεπαφές TWI στο ntuAboard_G1

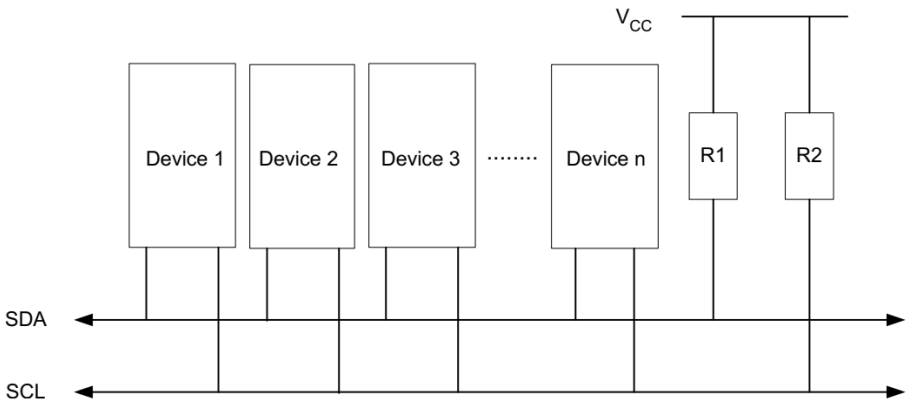
Στο ntuAboard_G1_υπάρχουν δύο διεπαφές TWI, η TWI0 και η TWI1. Οι πληροφορίες που παρουσιάζονται στη συνέχεια αναφέρονται στη διεπαφή TWI0. Ακριβώς τα ίδια ισχύουν και για διεπαφή TWI1.

Οι δυο γραμμές του TWI0 διαύλου συνδέονται στους ακροδέκτες PC4 (SDA) και PC5 (SCL) του ATmega328PB. Οι αντιστάσεις πρόσδεσης (pull-up resistors) έχουν τιμή 10 Khom και συνδέονται στο δίαυλο με την τοποθέτηση βραχυκυκλωτήρων στους κονέκτορες J12 και J13.



Σχήμα 5.1: Κονέκτορες J12 και J13 για σύνδεση των TWI pull-up resistors

Όταν ο δίαυλος TWI είναι ενεργός τότε οι ακροδέκτες PC4 (SDA) και PC5 (SCL) δεν μπορούν να χρησιμοποιηθούν για άλλο σκοπό. Στο παρακάτω σχήμα φαίνεται η συνδεσμολογία του διαύλου TWI



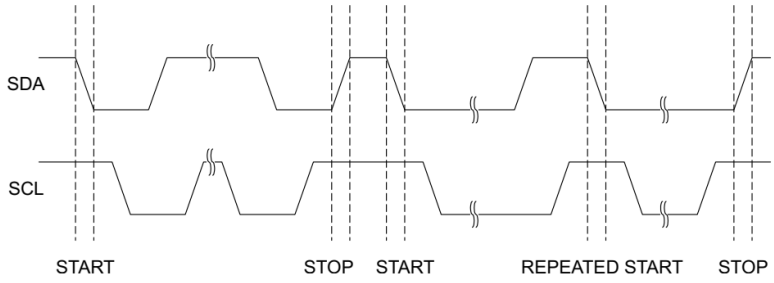
Σχήμα 5.2: Συνδεσμολογία TWI διαύλου.

Ορολογία διαύλου TWI:

- **Master**: Η συσκευή που αρχίζει μια μεταφορά, παράγει το σήμα του ρολογιού (SCL) και τερματίζει τη μεταφορά.
- **Slave**: Η συσκευή που καλείται κάθε φορά από τον Master κάνοντας χρήση της ατομικής διεύθυνσης της.
- **Transmitter**: Η συσκευή που στέλνει δεδομένα στο δίαυλο.
- **Receiver**: Η συσκευή που λαμβάνει δεδομένα από το δίαυλο.
- **Multi-master**: Η δυνατότητα ύπαρξης πολλών Master στο δίαυλο, χωρίς απώλεια δεδομένων λόγω σύγκρουσης. Ωστόσο, κάθε χρονική στιγμή, υπάρχει ενεργός μόνο ένας Master στο δίαυλο.
- **Arbitration**: Η διαδικασία που εγκρίνει κάθε φορά, μόνο έναν Master, να πάρει τον έλεγχο του διαύλου.
- **Synchronization**: Η διαδικασία που συγχρονίζει τα σήματα ρολογιών που παρέχονται από δύο ή περισσότερους Master.
- **SDA**: Η γραμμή του σήματος δεδομένων.
- **SCL**: Η γραμμή του σήματος ρολογιού

Συνθήκες START και STOP

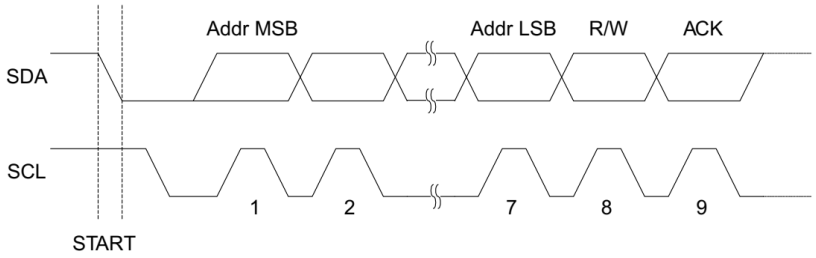
Η μετάδοση ξεκινά όταν ο Master εκδίδει μια συνθήκη START στο δίαυλο και τερματίζεται όταν ο Master εκδίδει μια συνθήκη STOP. Μεταξύ μιας συνθήκης START και STOP, ο δίαυλος θεωρείται απασχολημένος και κανένας άλλος Master δεν πρέπει να προσπαθήσει να πάρει τον έλεγχο του διαύλου. Μια ειδική περίπτωση παρουσιάζεται όταν εκδίδεται μια νέα συνθήκη START μεταξύ μιας συνθήκης START και STOP. Αυτό αναφέρεται ως συνθήκη REPEATED START και χρησιμοποιείται όταν ο Master επιθυμεί να ξεκινήσει μια νέα μεταφορά χωρίς να παραιτηθεί από τον έλεγχο του διαύλου.



Σχήμα 5.3 Συνθήκες START, REPEATED START, και STOP

Διευθυνσιοδοτηση TWI διαύλου

Όλα τα πακέτα διευθύνσεων που μεταδίδονται στον δίαυλο TWI έχουν μήκος εννέα bit, όπως φαίνεται στο παρακάτω σχήμα:

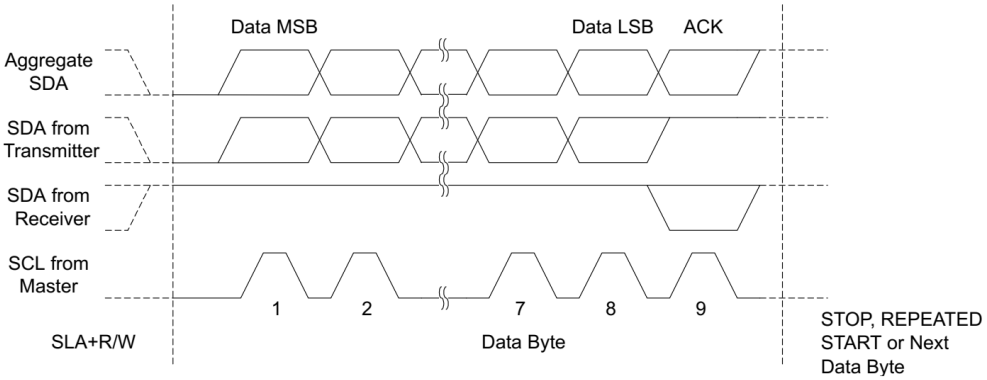


Σχήμα 5.4 Μορφή πακέτου διευθύνσεων

Αποτελούνται από επτά bit διεύθυνσης, ένα bit ελέγχου READ/WRITE και ένα bit επιβεβαίωσης (ACK). Εάν το bit READ/WRITE είναι λογικό 1 πρέπει να εκτελεστεί μια λειτουργία ανάγνωσης, διαφορετικά θα πρέπει να εκτελεστεί μια λειτουργία εγγραφής. Το MSBit του byte διεύθυνσης μεταδίδεται πρώτο. Ένα πακέτο διευθύνσεων που αποτελείται από μια διεύθυνση slave και ένα bit READ ή WRITE ονομάζεται SLA+R ή SLA+W αντίστοιχα
Όταν ένας Slave αναγνωρίσει ότι έχει διευθυνσιοδοτηθεί, τότε πρέπει να στείλει σήμα αναγνώρισης (ACK) κρατώντας το SDA χαμηλά στον ένατο κύκλο ρολογιού (κύκλος ACK). Εάν ο Slave είναι απασχολημένος ή για κάποιο άλλο λόγο δεν μπορεί να εξυπηρετήσει το αίτημα του Master, η γραμμή SDA θα πρέπει να παραμείνει ψηλά στον κύκλο ρολογιού ACK. Το Master μπορεί στη συνέχεια να μεταδώσει μια συνθήκη STOP ή μια συνθήκη REPEATED START για να ξεκινήσει μια νέα μετάδοση. Οι διευθύνσεις των συσκευών του διαύλου μπορούν να εκχωρηθούν ελεύθερα, αλλά η διεύθυνση '0000 000' προορίζεται για μια γενική κλήση. Όταν εκδίδεται μια γενική κλήση, όλοι οι Slave θα πρέπει να ανταποκρίνονται τραβώντας τη γραμμή SDA χαμηλά στον κύκλο ACK. Μια γενική κλήση χρησιμοποιείται όταν ένας Master επιθυμεί να μεταδώσει το ίδιο μήνυμα σε πολλούς Slave στο σύστημα. Όταν η διεύθυνση γενικής κλήσης ακολουθούμενη από ένα bit εγγραφής μεταδίδεται στο δίαυλο, όλοι οι Slave που έχουν ρυθμιστεί για να επιβεβαιώνουν τη γενική κλήση θα τραβήξουν τη γραμμή SDA χαμηλά στον κύκλο ACK. Τα ακόλουθα πακέτα δεδομένων θα ληφθούν από όλους τους Slave που αναγνώρισαν τη γενική κλήση. Όλες οι διευθύνσεις της μορφής «1111xxxx» θα πρέπει να δεσμευτούν για μελλοντικούς σκοπούς.

Πακέτα δεδομένων

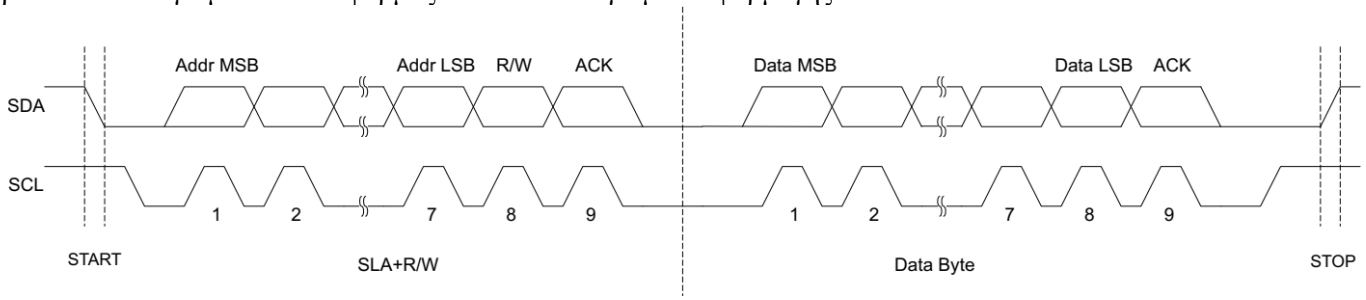
Όλα τα πακέτα δεδομένων που μεταδίδονται στο δίαυλο TWI έχουν μήκος εννέα bit, 8 bit δεδομένων και ένα bit επιβεβαίωσης. Το MSBit του byte δεδομένων μεταδίδεται πρώτο. Κατά τη μεταφορά δεδομένων, ο Master δημιουργεί το ρολόι, το START και το STOP, ενώ ο δέκτης είναι υπεύθυνος για την επιβεβαίωση της παραλαβής (ACK). Εάν ο δέκτης παραλείψει το ACK τότε αυτό ισοδυναμεί με μη επιβεβαίωση (Not Acknowledge, NACK). Όταν ο δέκτης λάβει το τελευταίο byte ή για κάποιο λόγο δεν μπορεί να λάβει άλλα byte, θα πρέπει να ενημερώσει τον πομπό στέλνοντας ένα NACK μετά το τελευταίο byte.



Σχήμα 5.5 Μορφή πακέτου δεδομένων

Τυπική μετάδοση δεδομένων

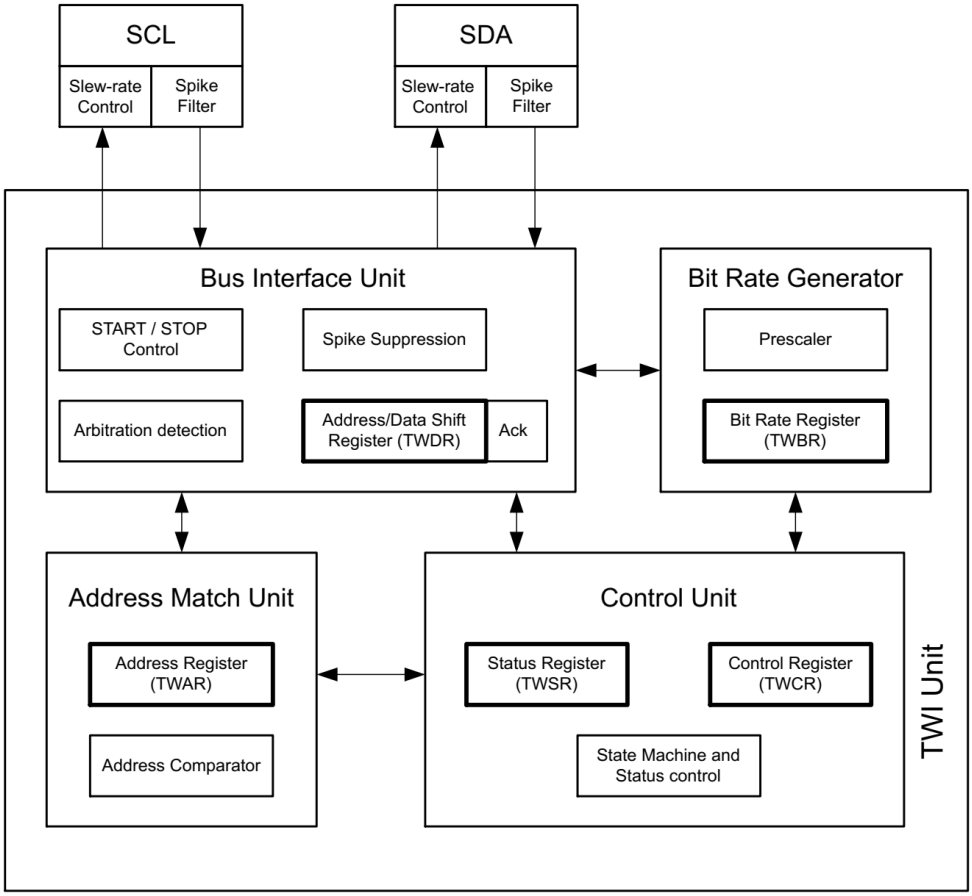
Μια μετάδοση αποτελείται από μια συνθήκη START, ένα πακέτο διεύθυνσης (Slave address(SLA) +R/W), ένα ή περισσότερα πακέτα δεδομένων και μια συνθήκη STOP. Ένα κενό μήνυμα, που αποτελείται από ένα START ακολουθούμενο από ένα STOP, είναι μη έγκυρο. Ο Slave μπορεί να παρατείνει τη χαμηλή περίοδο του ρολογιού κρατώντας τη γραμμή SCL χαμηλά. Αυτό είναι χρήσιμο εάν η ταχύτητα ρολογιού που έχει ρυθμιστεί από τον Master είναι πολύ γρήγορη για το Slave ή εάν το Slave χρειάζεται επιπλέον χρόνο για επεξεργασία μεταξύ των μεταδόσεων δεδομένων. Το παρακάτω σχήμα απεικονίζει μια τυπική μετάδοση δεδομένων. Σημειώστε ότι πολλά byte δεδομένων μπορούν να μεταδοθούν μεταξύ του SLA+R/W και της συνθήκης STOP, ανάλογα με το πρωτόκολλο λογισμικού που εφαρμόζεται από το λογισμικό εφαρμογής.



Σχήμα 5.6 Τυπική μετάδοση δεδομένων

Διεπαφή TWI στον ATmega328PB

Η Διεπαφή TWI στον ATmega328PB αποτελείται από πολλά υποσυστήματα, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 5.7 Διάγραμμα διεπαφής TWI στον ATmega328PB

Στη συνέχεια παρουσιάζονται οι καταχωρητές του ATmega328PB που σχετίζονται με τη λειτουργία του TWI.

Register	Offset	Reset Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TWBR0	0xB8	0b00000000	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn
TWSR0	0xB9	0b11111X00	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
TWAR0	0xBA	0b00000010	TWA[6:0]							TWGCE
TWDR0	0xBB	0b00000001	TWD[7:0]							
TWCR0	0xBC	0b000000X0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
TWAMR0	0xBD	0b0000000X	TWAM[6:0]							

Πίνακας 5.1 Οι καταχωρητές του ATmega328PB που σχετίζονται με τη λειτουργία του TWI

TWBR0 (Offset 0xB8), TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0
	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn	TWBRn
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Σχήμα 5.8 TWBR0, TWI Bit Rate Register

TWBR0[7:0] Ρυθμίζουν τη συχνότητα του σήματος SCL σύμφωνα με την ακόλουθη εξίσωση:

$$\text{Συχνότητα SCL} = \frac{\text{Συχνότητα ρολογιού CPU}}{16 + 2 \cdot (\text{TWBR0}) \cdot \text{PrescalerValue}}$$

Η τιμή του Prescaler καθορίζεται από τα bits TWPS[1:0] του καταχωρητή TWSR0.

TWSR0(Offset 0xB9), TWI Status Register

Bit	7	6	5	4	3	2	1	0
	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
Access	R	R	R	R	R		R/W	R/W
Reset	1	1	1	1	1		0	0

Σχήμα 5.9 TWSR0, TWI Status Register

TWPS[1:0] Η τιμή του Prescaler για τη ρύθμιση της συχνότητας του σήματος SCL καθορίζεται από τα bits TWPS[1:0] σύμφωνα με τον παρακάτω πίνακα:

TWPS[1:0]	Prescaler Value
00	1
01	4
10	16
11	64

Πίνακας 5.2 Bit rate Prescaler Values

TWS[7:3] αντικατοπτρίζουν την κατάσταση λειτουργίας του σειριακού διαύλου TWI για καθένα από τους τέσσερις κύριους τρόπους λειτουργίας:

1. Status Codes for Master Transmitter Mode

TWSR	Status of the Bus and Interface Hardware
0x08	A START has been transmitted
0x10	A repeated START has been transmitted
0x18	SLA+W has been transmitted; ACK has been received
0x20	SLA+W has been transmitted; NOT ACK has been received
0x28	Data byte has been transmitted; ACK has been received
0x30	Data byte has been transmitted; NOT ACK has been received
0x38	Arbitration lost in SLA+W or data bytes

2. Status codes for Master Receiver Mode

TWSR	Status of the Bus and Interface Hardware
0x08	A START has been transmitted
0x10	A repeated START has been transmitted
0x38	Arbitration lost in SLA+R or NOT ACK bit
0x40	SLA+R has been transmitted; ACK has been received
0x48	SLA+R has been transmitted; NOT ACK has been received
0x50	Data byte has been received; ACK has been returned
0x58	Data byte has been received; NOT ACK has been returned

3. Status Codes for Slave Transmitter Mode

TWSR	Status of the Bus and Interface Hardware
0xA8	Own SLA+R has been received; ACK has been returned
0xB0	Arbitration lost in SLA+R/W; Own SLA+R has been received; ACK has been returned;
0xB8	Data byte has been transmitted; ACK has been received;
0xC0	Data byte has been transmitted; NOT ACK has been received;
0xC8	Last data byte has been transmitted; ACK has been received;

4. Status codes for Slave Receiver Mode

TWSR	Status of the Bus and Interface Hardware
0x60	Own SLA+W has been received; ACK has been returned;
0x68	Arbitration lost in SLA+R/W; Own SLA+W has been received; ACK has been returned;
0x70	General call address has been received; ACK has been returned;
0x78	Arbitration lost in SLA+R/W; General call address has been received; ACK has been returned;
0x80	Data has been received; ACK has been returned;
0x88	Data has been received; NOT ACK has been returned;
0x90	General call data has been received; ACK has been returned;
0x98	General call data has data has been received; NOT ACK has been returned;
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave;

TWAR0 (Offset 0xBA), TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0
	TWA[6:0]							TWGCE
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	0

Σχήμα 5.10 TWAR0**,** TWI (Slave) Address

TWA[6:0]: είναι η 7-bit διεύθυνση του slave

TWGCE: Χρησιμοποιείται για να επιτρέψει την αναγνώριση της γενικής διεύθυνσης κλήσης (0x00).

TWDR0 (Offset 0xBB), TWI Data Register

Bit	7	6	5	4	3	2	1	0
	TWD[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	1

Σχήμα 5.11 TWDR0, TWI Data Register

TWD[7:0] Byte δεδομένων που θα μεταδοθεί ή το τελευταίο byte δεδομένων που ελήφθη.

TWCR0 (Offset 0xBC), TWI Control Register

Bit	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
Access	R/W	R/W	R/W	R/W	R	R/W		R/W
Reset	0	0	0	0	0	0		0

Σχήμα 5.12 TWCR0, TWI Control Register

TWINT: Σημαία διακοπής TWI. Τίθεται όταν το TWI ολοκληρώσει την τρέχουσα εργασία του και αναμένει την απόκριση του λογισμικού εφαρμογής. Αν TWINT=1 τότε η χαμηλή περίοδος του SCL επιμηκύνεται. Η σημαία TWINT μηδενίζεται μόνο από το λογισμικό γράφοντας λογική 1 σε αυτήν. Αν το TWINT μηδενιστεί τότε ξεκινά τη λειτουργία του TWI. Όλες οι προσβάσεις στα TWAR0, TWSR0 και TWDR0 πρέπει να έχουν ολοκληρωθεί πριν από την εκκαθάριση αυτής της σημαίας.

TWEA: Αυτό το bit ελέγχει τη δημιουργία του παλμού επιβεβαίωσης. Εάν TWEA=1 το ACK δημιουργείται στον δίαυλο εάν έχει ληφθεί η ατομική διεύθυνση του slave ή έχει ληφθεί μια γενική κλήση και ταυτόχρονα TWAR0.TWGCE =1 ή έχει ληφθεί ένα byte δεδομένων σε λειτουργία Master Receiver ή Slave Receiver.

Γράφοντας το bit TWEA στο μηδέν, η συσκευή μπορεί ουσιαστικά να αποσυνδεθεί από το δίαυλο προσωρινά. Στη συνέχεια, η αναγνώριση διεύθυνσης μπορεί να ενεργοποιηθεί ξανά, θέτοντας ξανά το bit TWEA.

TWSTA: Συνθήκη START. Η εφαρμογή θέτει το TWSTA όταν επιθυμεί να γίνει Master. Το υλικό TWI ελέγχει εάν ο δίαυλος είναι διαθέσιμος και δημιουργεί μια συνθήκη START στο δίαυλο εάν είναι ελεύθερος. Ωστόσο, εάν ο δίαυλος δεν είναι ελεύθερος, το TWI περιμένει μέχρι να εντοπιστεί μια συνθήκη STOP και στη συνέχεια δημιουργεί μια νέα συνθήκη START για να δώσει το δικαίωμα του Master. Το TWSTA πρέπει να μηδενιστεί από το λογισμικό όταν έχει γίνει εκπομπή του START.

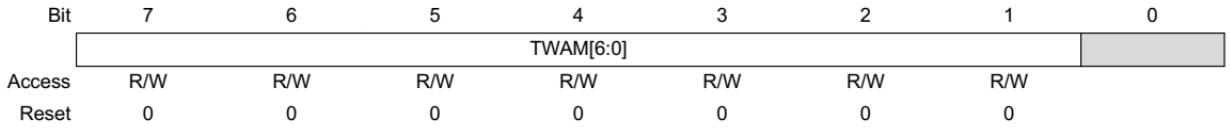
TWSTO: Συνθήκη STOP. Σε λειτουργία Master θα δημιουργήσει μια συνθήκη STOP στον δίαυλο. Όταν η συνθήκη STOP εκτελεστεί στο δίαυλο, το TWSTO μηδενίζεται αυτόματα. Στη λειτουργία Slave το TWSTO τίθεται για την ανάκτηση από μια συνθήκη σφάλματος. Αυτό δεν θα δημιουργήσει μια συνθήκη STOP, αλλά το TWI επιστρέφει σε μια προκαθορισμένη λειτουργία μη διευθυνσιοδοτούμενου Slave και απελευθερώνει τις γραμμές SCL και SDA.

TWWC: Σημαία σύγκρουσης εκπομπής στο δίαυλο. Το bit TWWC τίθεται όταν γίνεται προσπάθεια εγγραφής του TWDR0 ενώ TWCR0.TWINT =0. Αυτή η σημαία μηδενίζεται γράφοντας στον καταχωρητή TWDR0 ενώ TWINT=1.

TWEN: Ενεργοποίηση του TWI. Όταν TWEN=1 το TWI n αναλαμβάνει τον έλεγχο των ακροδεκτών που είναι συνδεδεμένες με τα σήματα SCL και SDA. Επίσης ενεργοποιεί τους περιοριστές του ρυθμού μεταβολής της τάσης και τα φίλτρα θορύβου σε αυτούς τους ακροδέκτες. Εάν αυτό το TWEN μηδενιστεί το TWI απενεργοποιείται και όλες οι μεταδόσεις του τερματίζονται.

TWIE: Ενεργοποίηση της διακοπής του TWI. Όταν TWIE =1 και SREG.I=1 το αίτημα διακοπής του TWI θα ενεργοποιηθεί για όσο διάστημα η σημαία TWCR0.TWINT είναι υψηλή.

TWAMR0 (Offset 0xBD), TWI Slave Address Mask Register



Σχήμα 5.13 TWAMR0 , TWI Slave Address Mask Register

TWAM[6:0]: Το TWAMR0 μπορεί να φορτωθεί με μια μάσκα για τη διεύθυνση του Slave (7 bit). Κάθε ένα από τα bit TWAMR[6:0] μπορεί να κρύψει(απενεργοποιήσει) τα αντίστοιχα bit διεύθυνσης στον καταχωρητή διευθύνσεων TWAR0. Εάν το bit μάσκας έχει οριστεί σε λογικό 1, τότε η διάταξη αντιστοίχισης διεύθυνσης αγνοεί τη σύγκριση μεταξύ του εισερχόμενου bit διεύθυνσης και του αντίστοιχου bit στο TWAR0.

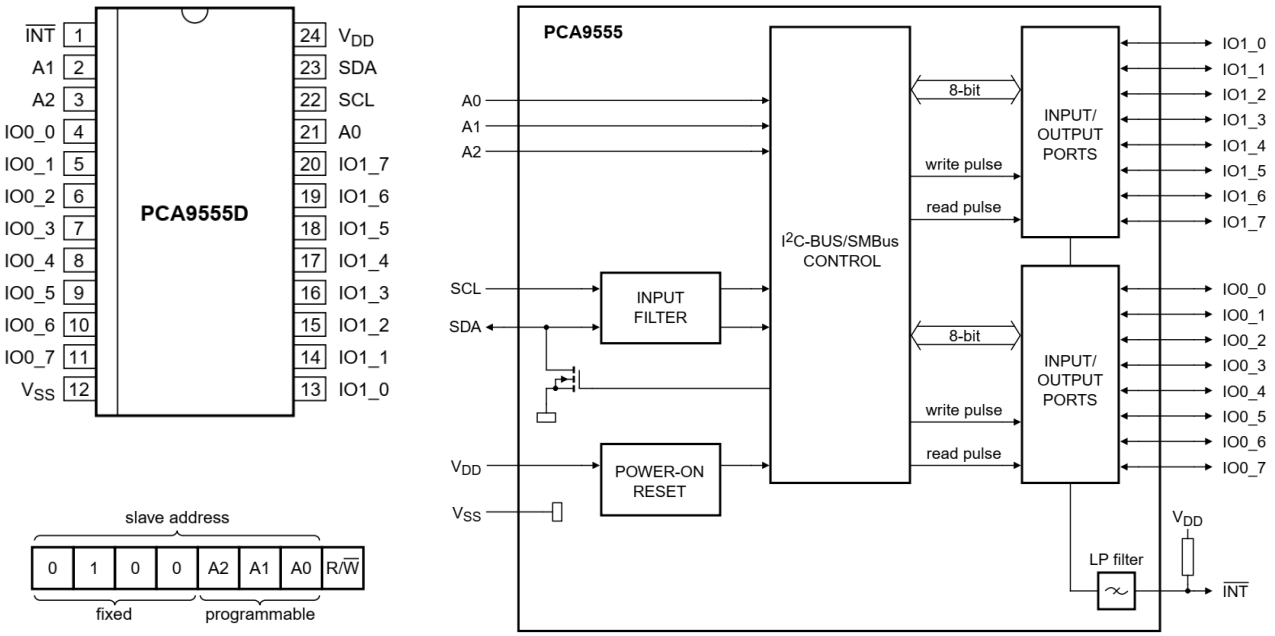
Παράδειγμα μετάδοσης ενός byte δεδομένων σε ένα slave.

Το TWI βασίζεται σε διακοπές. Όταν τεθεί το TWINT, τότε το TWI έχει ολοκληρώσει μια λειτουργία και ο καταχωρητής κατάστασης TWSR0 περιέχει μια τιμή που υποδεικνύει την τρέχουσα κατάσταση του TWI διαύλου. Το λογισμικό εφαρμογής ρυθμίζει τους καταχωρητές TWCR0 και TWDR0 για τον επόμενο κύκλο του διαύλου TWI .

	ASSEMBLY CODE	C CODE	ΣΧΟΛΙΑ
1	Ldi r16,1<<TWINT) | (1<<TWSTA) | (1<<TWEN) out TWCR0, r16	TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)	Αποστολή START
2	Call wait1 Wait1: in r16, TWCR0 sbrs r16, TWINT rjmp wait1 ret	while (!(TWCR0 & (1<<TWINT)));	Αναμονή έως TWINT=1 που σημαίνει ότι το START μεταδόθηκε.
3a	in r16, TWSR0 andi r16, 0xF8 cpi r16, START brne ERROR	if ((TWSR0 & 0xF8) != START) ERROR();	Αν TWSR0.TWS[7:3] δείχνει κωδικό διαφορετικό από START τότε υπάρχει σφάλμα.
3b	ldi r16, SLA_W out TWDR0, r16 ldi r16, (1<<TWINT) | (1<<TWEN) out TWCR0, r16	TWDR0 = SLA_W; TWCR0 = (1<<TWINT) | (1<<TWEN);	Φορτώστε το SLA_W στον καταχωρητή TWDR0 . Μηδένισε το TWCR.WINT για να ξεκινήσει η μετάδοση.
4	Call wait1	while (!(TWCR0 & (1<<TWINT)));	
5a	in r16, TWSR0 andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR	if ((TWSR0 & 0xF8) != MT_SLA_ACK) ERROR();	Αν TWSR0.TWS[7:3] δείχνει κωδικό διαφορετικό από SLA_ACK τότε υπάρχει σφάλμα.
5b	ldi r16, DATA out TWDR0, r16 ldi r16, (1<<TWINT) | (1<<TWEN) out TWCR, r16	TWDR0 = DATA; TWCR0 = (1<<TWINT) | (1<<TWEN);	Φορτώστε τα DATA στον καταχωρητή TWDR0 . Μηδένισε το TWCR.WINT για να ξεκινήσει η μετάδοση.
6	Call wait1	while (!(TWCR0 & (1<<TWINT)));	
7a	in r16, TWSR0 andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR	if ((TWSR0 & 0xF8) != MT_DATA_ACK) ERROR();	Αν TWSR0.TWS[7:3] δείχνει κωδικό διαφορετικό από DATA_ACK τότε υπάρχει σφάλμα.
7b	ldi r16, (1<<TWINT) | (1<<TWEN) | (1<<TWSTO) out TWCR0, r16	TWCR0 = (1<<TWINT)| (1<<TWEN) | (1<<TWSTO);	Αποστολή STOP

Ολοκληρωμένο PCA9555

Το PCA9555 είναι ένα ολοκληρωμένο που παρέχει δύο πόρτες εισόδου/εξόδου γενικής χρήσης, IO0 και IO1, των 8-bit η κάθε μια, για εφαρμογές I2C-bus. Διαθέτει μία έξοδο διακοπής (open-drain) που ενεργοποιείται όταν η κατάσταση εισόδου διαφέρει από την αντίστοιχη κατάσταση του καταχωρητή θύρας εισόδου και χρησιμοποιείται για να υποδείξει ότι μια κατάσταση εισόδου έχει μεταβληθεί. Κατά την εφαρμογή της τάσεις τροφοδοσίας γίνεται επαναφορά της συσκευής και οι καταχωρητές τίθενται στις προεπιλεγμένες τιμές τους. Η I^2C διεύθυνση του PCA9555 είναι 7-bit εκ των οποίων τα 4bit είναι σταθερά και τα υπόλοιπα τρία ρυθμίζονται από τους ακροδέκτες A0, A1 και A2. Στο ntuAboard_G1 οι ακροδέκτες A0, A1 και A2 είναι συνδεδεμένοι στη γη (0 Volt) οπότε η διεύθυνση του ενσωματωμένου PCA9555 είναι 0b0100000. Το διάγραμμα, οι ακροδέκτες και η I^2C διεύθυνση του PCA9555 φαίνονται στο παρακάτω σχήμα:



Σχήμα 5.14 Διάγραμμα, ακροδέκτες και I2C διεύθυνση του PCA9555

Καταχωρητές του PCA9555

Κατά τη διάρκεια μιας μετάδοσης εγγραφής, το πρώτο byte που ακολουθεί το byte της διεύθυνσης(command byte) χρησιμοποιείται ως δείκτης για να προσδιορίσει ποιος από τους παρακάτω καταχωρητές θα γραφεί ή θα διαβαστεί σύμφωνα με τον παρακάτω πίνακα:

Command	Register
0	Input port 0
1	Input port 1
2	Output port 0
3	Output port 1
4	Polarity Inversion port 0
5	Polarity Inversion port 1
6	Configuration port 0
7	Configuration port 1

Πίνακας 5.3 Command byte

Οι καταχωρητές Input port 0 και Input port 1 είναι θύρες μόνο εισόδου. Αντικατοπτρίζουν τα εισερχόμενα λογικά επίπεδα των ακροδεκτών, ανεξάρτητα από το εάν ένας ακροδέκτης ορίζεται ως είσοδος ή έξοδος. Οι εγγραφές σε αυτούς τους καταχωρητές δεν έχουν κανένα αποτέλεσμα. Η προεπιλεγμένη τιμή «X» καθορίζεται από το εξωτερικά εφαρμοζόμενο λογικό επίπεδο.

Input port 0 Register

Bit	7	6	5	4	3	2	1	0
Symbol	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Default	X	X	X	X	X	X	X	X

Input port 1 Register

Bit	7	6	5	4	3	2	1	0
Symbol	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Default	X	X	X	X	X	X	X	X

Οι καταχωρητές Output port 0 και Output port 1 είναι θύρες μόνο εξόδου. Αντικατοπτρίζουν τα εξερχόμενα λογικά επίπεδα των ακροδεκτών, που ορίζονται ως έξοδοι. Οι αναγνώσεις από αυτούς τους καταχωρητές αντικατοπτρίζουν την τιμή που βρίσκεται στα flip-flop που ελέγχουν την επιλογή εξόδου, όχι την πραγματική τιμή των ακροδεκτών.

Output port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	O0.7	O0.6	O0.5	O0.4	O0.3	O0.2	O0.1	O0.0
Default	1	1	1	1	1	1	1	1

Output port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0
Default	1	1	1	1	1	1	1	1

Οι καταχωρητές Polarity Inversion port 0 και Polarity Inversion port 1 επιτρέπουν την αντιστροφή της πολικότητας των δεδομένων των καταχωρητών της θύρας εισόδου. Εάν έχει τεθεί κάποιο bit σε έναν από αυτούς τους καταχωρητές, η πολικότητα δεδομένων της θύρας εισόδου αντιστρέφεται.

Polarity Inversion port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	N0.7	N0.6	N0.5	N0.4	N0.3	N0.2	N0.1	N0.0
Default	0	0	0	0	0	0	0	0

Polarity Inversion port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	N1.7	N1.6	N1.5	N1.4	N1.3	N1.2	N1.1	N1.0
Default	0	0	0	0	0	0	0	0

Οι καταχωρητές Configuration port 0 και Configuration port 1 διαμορφώνουν τις κατευθύνσεις των ακροδεκτών I/O. Εάν έχει τεθεί ένα bit σε έναν από αυτούς τους καταχωρητές, ο αντίστοιχος ακροδέκτης της θύρας ενεργοποιείται

ως είσοδος. Υπάρχει μια αντίσταση υψηλής τιμής(pull-up resistor) για πρόσδεσης κάθε ακροδέκτη στο VDD. Κατά την επαναφορά, οι θύρες της συσκευής είναι είσοδοι με αντιστάσεις πρόσδεσης στο VDD.

Configuration port 0 register

Bit	7	6	5	4	3	2	1	0
Symbol	C0.7	C0.6	C0.5	C0.4	C0.3	C0.2	C0.1	C0.0
Default	1	1	1	1	1	1	1	1

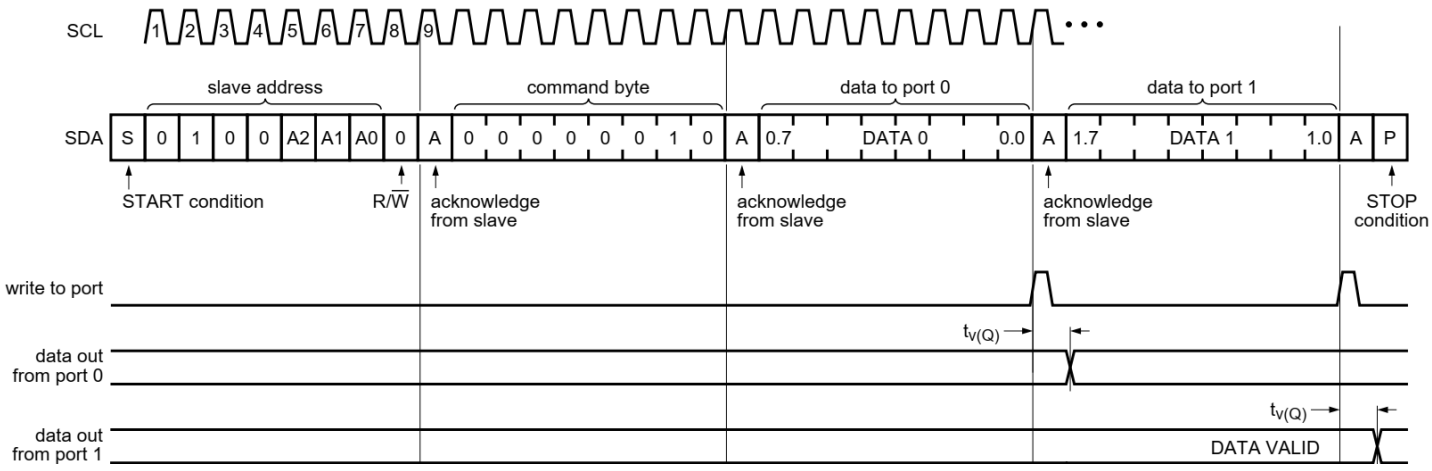
Configuration port 1 register

Bit	7	6	5	4	3	2	1	0
Symbol	C1.7	C1.6	C1.5	C1.4	C1.3	C1.2	C1.1	C1.0
Default	1	1	1	1	1	1	1	1

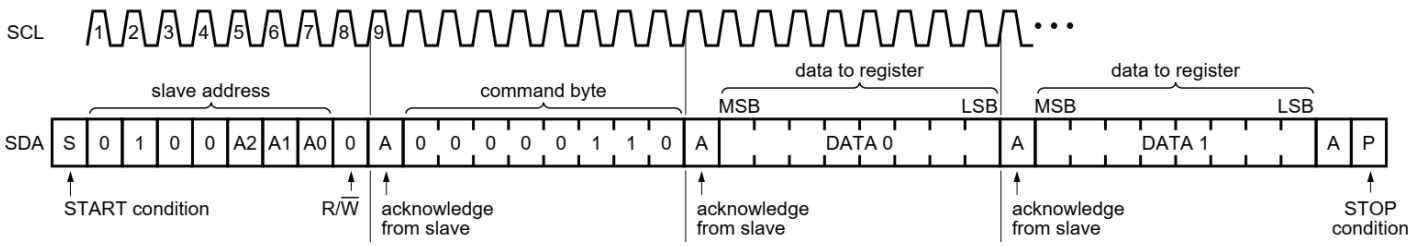
Τα δεδομένα μεταδίδονται στο PCA9555 στέλνοντας τη διεύθυνση της συσκευής και μηδενίζοντάς το λιγότερο σημαντικό bit. Το byte εντολής αποστέλλεται μετά τη διεύθυνση και καθορίζει ποιος καταχωρητής θα λάβει τα δεδομένα που ακολουθούν. Οι οκτώ καταχωρητές τουPCA9555 έχουν ρυθμιστεί να λειτουργούν ως τέσσερα ζεύγη καταχωρητών. Τα τέσσερα ζεύγη είναι θύρες εισόδου, θύρες εξόδου, θύρες αναστροφής πολικότητας και θύρες διαμόρφωσης.

Εγγραφή στους καταχωρητές θυρών

Μετά την αποστολή δεδομένων σε έναν καταχωρητή, το επόμενο byte δεδομένων θα σταλεί στον άλλο καταχωρητή του ζεύγους. Για παράδειγμα, εάν το πρώτο byte αποσταλεί στην Output Port 1 (register 3) τότε το επόμενο byte θα αποθηκευτεί στη Output Port 0 (register 2). Δεν υπάρχει περιορισμός στον αριθμό των byte δεδομένων που αποστέλλονται σε μία μετάδοση εγγραφής. Με αυτόν τον τρόπο, κάθε καταχωρητής 8-bit μπορεί να ενημερώνεται ανεξάρτητα από τους άλλους καταχωρητές. Στα επόμενα δύο σχήματα απεικονίζεται η διαδικασία εγγραφής στα Output port registers και στα Configuration registers.



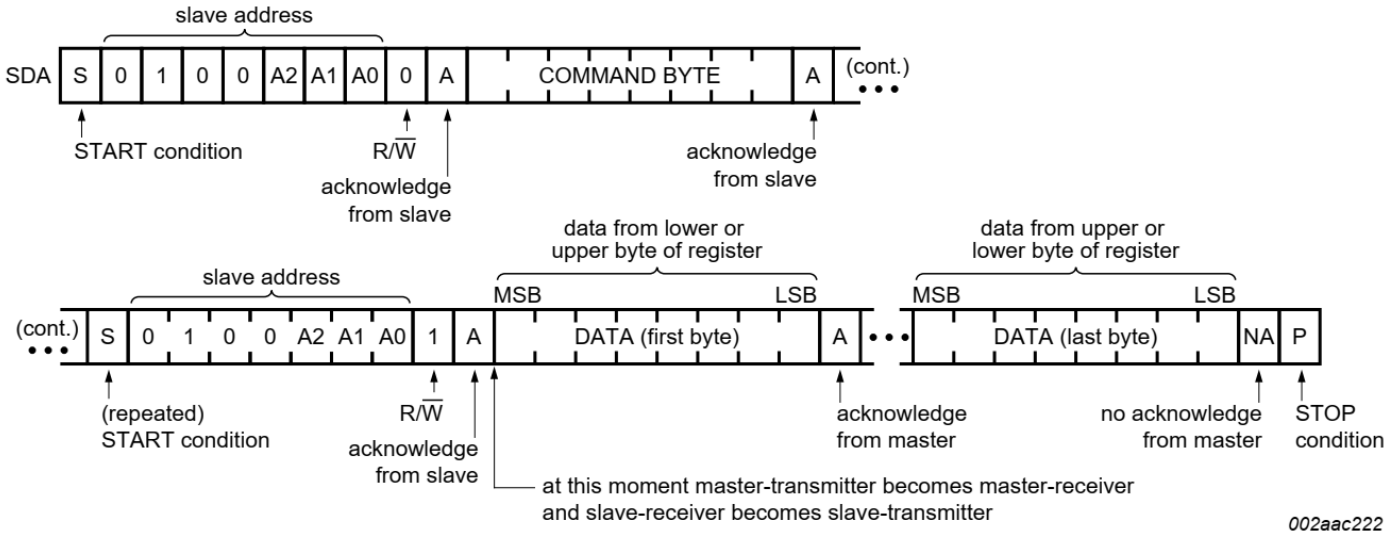
Σχήμα 5.15 Write to Output port registers



Σχήμα 5.16 Write to Configuration registers

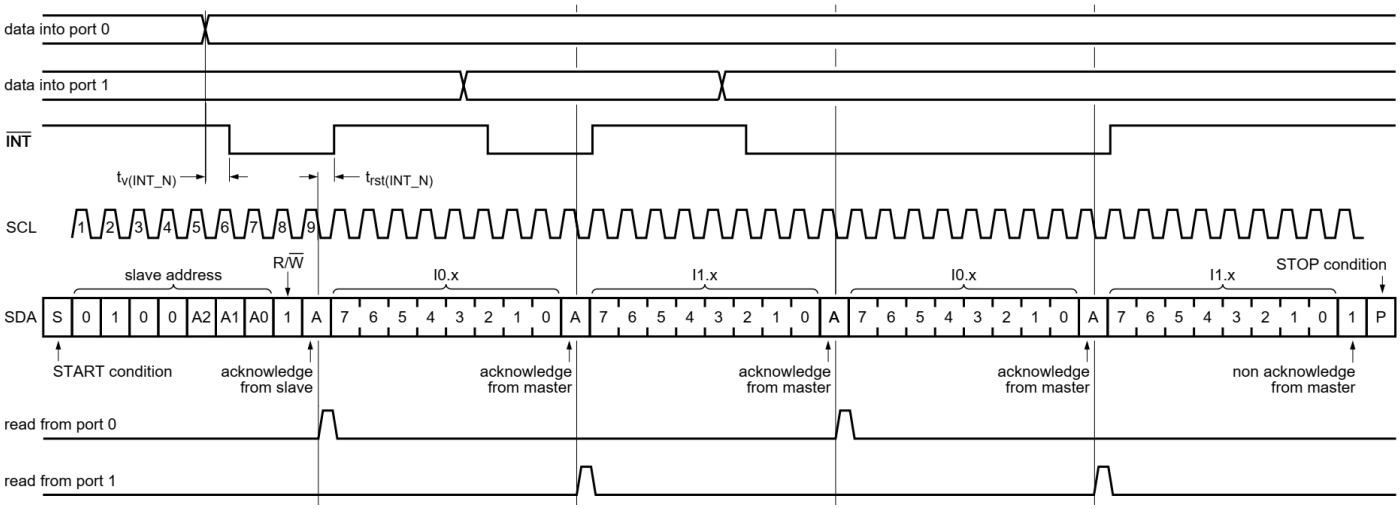
Ανάγνωση των καταχωρητών των θυρών

Για να διαβαστούν τα δεδομένα από το PCA9555, ο master του I2C διαύλου πρέπει πρώτα να στείλει τη διεύθυνση του PCA9555 με το λιγότερο σημαντικό bit να έχει οριστεί σε λογικό 0. Μετά τη διεύθυνση αποστέλλεται το command byte και καθορίζει σε ποιον καταχωρητή θα γίνει προσπέλαση. Μετά από επανεκκίνηση, η διεύθυνση της συσκευής αποστέλλεται ξανά, αλλά αυτή τη φορά το λιγότερο σημαντικό bit ορίζεται σε ένα λογικό 1. Τα δεδομένα από τον καταχωρητή που ορίζεται από command byte θα σταλούν στη συνέχεια από το PCA9555. Μετά την ανάγνωση του πρώτου byte, μπορούν να διαβαστούν επιπλέον byte, αλλά τα δεδομένα θα αντικατοπτρίζουν τώρα τις πληροφορίες στον άλλο καταχωρητή του ζεύγους. Δεν υπάρχει περιορισμός στον αριθμό των byte δεδομένων που λαμβάνονται σε μία μετάδοση ανάγνωσης, αλλά στο τελικό byte που λαμβάνεται, ο master toy διαύλου δεν πρέπει να στείλει ACK . Στα επόμενα σχήματα απεικονίζεται η διαδικασία ανάγνωσης από τους διάφορους καταχωρητές.



Σχήμα 5.17 Read from register

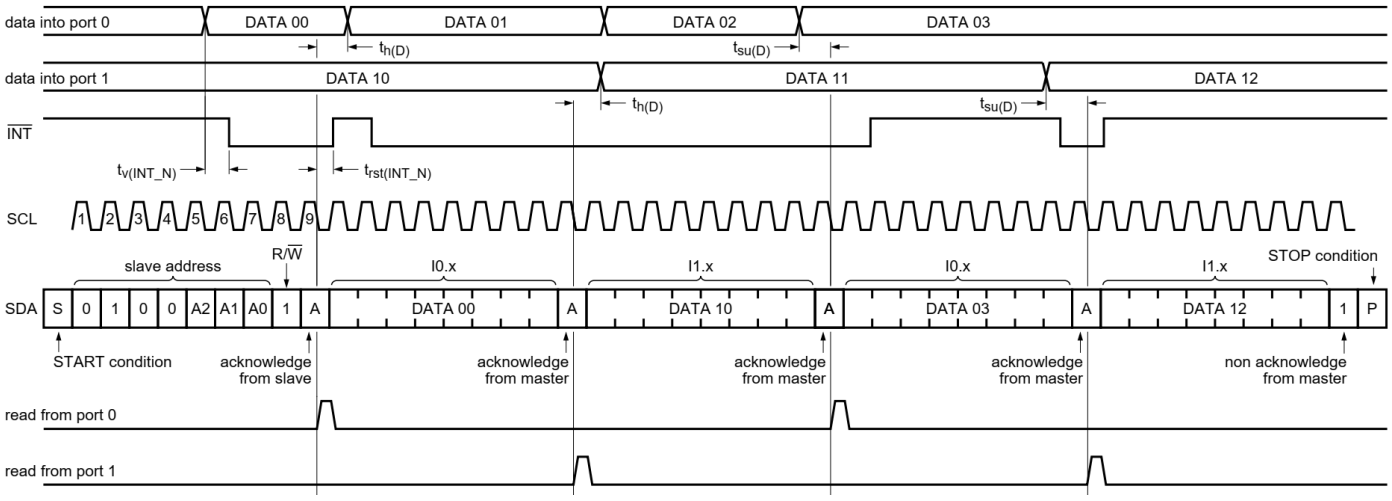
Η μεταφορά μπορεί να διακοπεί ανά πάσα στιγμή με αποστολή STOP.



Σχήμα 5.18 Read Input port register, scenario 1

Η μεταφορά δεδομένων μπορεί να διακοπεί ανά πάσα στιγμή με μια συνθήκη STOP. Όταν συμβεί αυτό, τα δεδομένα που υπάρχουν στην τελευταία φάση επιβεβαίωσης είναι έγκυρα.

Υποτίθεται ότι το command byte έχει προηγουμένως ρυθμιστεί σε «00» (Διάβασμα Input Port register).



Σχήμα 5.19 Read Input port register, scenario 2

Η μεταφορά δεδομένων μπορεί να διακοπεί ανά πάσα στιγμή με μια συνθήκη STOP. Όταν συμβεί αυτό, τα δεδομένα που υπάρχουν στην τελευταία φάση επιβεβαίωσης είναι έγκυρα.

Υποτίθεται ότι το command byte έχει προηγουμένως ρυθμιστεί σε «00» (Διάβασμα Input Port register).

Παράδειγμα 5.1

Στο παρακάτω παράδειγμα ρυθμίζεται η θύρα επέκτασης 0 του ολοκληρωμένου PCA9555 έτσι ώστε η λογική ή κατάσταση των ακροδεκτών να εναλλάσσεται μεταξύ λογικού 0 και λογικού 1. Ο χρόνος παραμονής σε καθεμία από τις δύο αυτές λογικές καταστάσεις είναι ένα δευτερόλεπτο. Το ολοκληρωμένο PCA9555 επικοινωνεί με τον μικροελεγκτή ATmega328PB μέσω της σειριακής Θύρας TWI.

```
#define F_CPU 16000000UL

#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>


#define PCA9555_0_ADDRESS 0x40          //A0=A1=A2=0 by hardware
#define TWI_READ    1                   // reading from twi device
#define TWI_WRITE   0                   // writing to twi device
#define SCL_CLOCK  100000L              // twi clock in Hz

//Fscl=Fcpu/(16+2*TWBR0_VALUE*PRESCALER_VALUE)
#define TWBR0_VALUE ((F_CPU/SCL_CLOCK)-16)/2

// PCA9555 REGISTERS
typedef enum {
  REG_INPUT_0              = 0,
  REG_INPUT_1              = 1,
  REG_OUTPUT_0             = 2,
  REG_OUTPUT_1             = 3,
  REG_POLARITY_INV_0       = 4,
  REG_POLARITY_INV_1       = 5,
  REG_CONFIGURATION_0      = 6,
  REG_CONFIGURATION_1      = 7
} PCA9555_REGISTERS;

//----------- Master Transmitter/Receiver --------------------
#define TW_START                    0x08
#define TW_REP_START                0x10

//---------------- Master Transmitter ----------------------
#define TW_MT_SLA_ACK               0x18
#define TW_MT_SLA_NACK              0x20
#define TW_MT_DATA_ACK              0x28

//---------------- Master Receiver ----------------
#define TW_MR_SLA_ACK               0x40
#define TW_MR_SLA_NACK              0x48
#define TW_MR_DATA_NACK             0x58
```

```
#define TW_STATUS_MASK  0b11111000
#define TW_STATUS   (TWSR0 & TW_STATUS_MASK)

//initialize TWI clock
void twi_init(void)
{
  TWSR0 = 0;                    // PRESCALER_VALUE=1
 TWBR0 = TWBR0_VALUE;           // SCL_CLOCK  100KHz
}

// Read one byte from the twi device (request more data from device)
unsigned char twi_readAck(void)
{
        TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
        while(!(TWCR0 & (1<<TWINT)));
        return TWDR0;
}

//Read one byte from the twi device, read is followed by a stop condition
unsigned char twi_readNak(void)
{
        TWCR0 = (1<<TWINT) | (1<<TWEN);
        while(!(TWCR0 & (1<<TWINT)));
        return TWDR0;
}

// Issues a start condition and sends address and transfer direction.
 // return 0 = device accessible, 1= failed to access device
unsigned char twi_start(unsigned char address)
{
   uint8_t   twi_status;

        // send START condition
        TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) return 1;

        // send device address
        TWDR0 = address;
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wail until transmission completed and ACK/NACK has been received
        while(!(TWCR0 & (1<<TWINT)));
```

```
        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_MT_SLA_ACK) && (twi_status != TW_MR_SLA_ACK) )
   {
      return 1;
   }

        return 0;
}

// Send start condition, address, transfer direction.
// Use ack polling to wait until device is ready
void twi_start_wait(unsigned char address)
{
   uint8_t   twi_status;

   while ( 1 )
   {
        // send START condition
        TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) continue;

        // send device address
        TWDR0 = address;
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wail until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status == TW_MT_SLA_NACK )||(twi_status ==TW_MR_DATA_NACK) )
        {
           /* device busy, send stop condition to terminate write operation */
              TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

              // wait until stop condition is executed and bus released
              while(TWCR0 & (1<<TWSTO));

           continue;
        }
        break;
    }
}
```

```
//  Send one byte to twi device,  Return 0 if write successful or 1 if write failed
unsigned char twi_write( unsigned char data )
{
        // send data to the previously addressed device
        TWDR0 = data;
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));
        if( (TW_STATUS & 0xF8) != TW_MT_DATA_ACK) return 1;
        return 0;
}

// Send repeated start condition, address, transfer direction
//Return:  0 device accessible
//         1 failed to access device
unsigned char twi_rep_start(unsigned char address)
{
   return twi_start( address );
}


// Terminates the data transfer and releases the twi bus
void twi_stop(void)
{
   // send stop condition
        TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

        // wait until stop condition is executed and bus released
        while(TWCR0 & (1<<TWSTO));
}


void PCA9555_0_write(PCA9555_REGISTERS reg, uint8_t value)
{
    twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
    twi_write(reg);
    twi_write(value);
    twi_stop();
}

uint8_t PCA9555_0_read(PCA9555_REGISTERS reg)
{
uint8_t ret_val;

    twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
    twi_write(reg);
    twi_rep_start(PCA9555_0_ADDRESS + TWI_READ);
    ret_val = twi_readNak();
    twi_stop();
```

```
    return ret_val;
}


int main(void) {

    twi_init();
   PCA9555_0_write(REG_CONFIGURATION_0, 0x00); //Set EXT_PORT0 as output

   while(1)
   {
      PCA9555_0_write(REG_OUTPUT_0, 0x00);
      _delay_ms(1000);

      PCA9555_0_write(REG_OUTPUT_0, 0xFF);
      _delay_ms(1000);
   }
}
```

Τα ζητούμενα της 5ης εργαστηριακής άσκησης

Ζήτημα 5.1

Να υλοποιηθούν για το μικροελεγκτή ATmega328PB, σε γλώσσα C, οι λογικές συναρτήσεις:

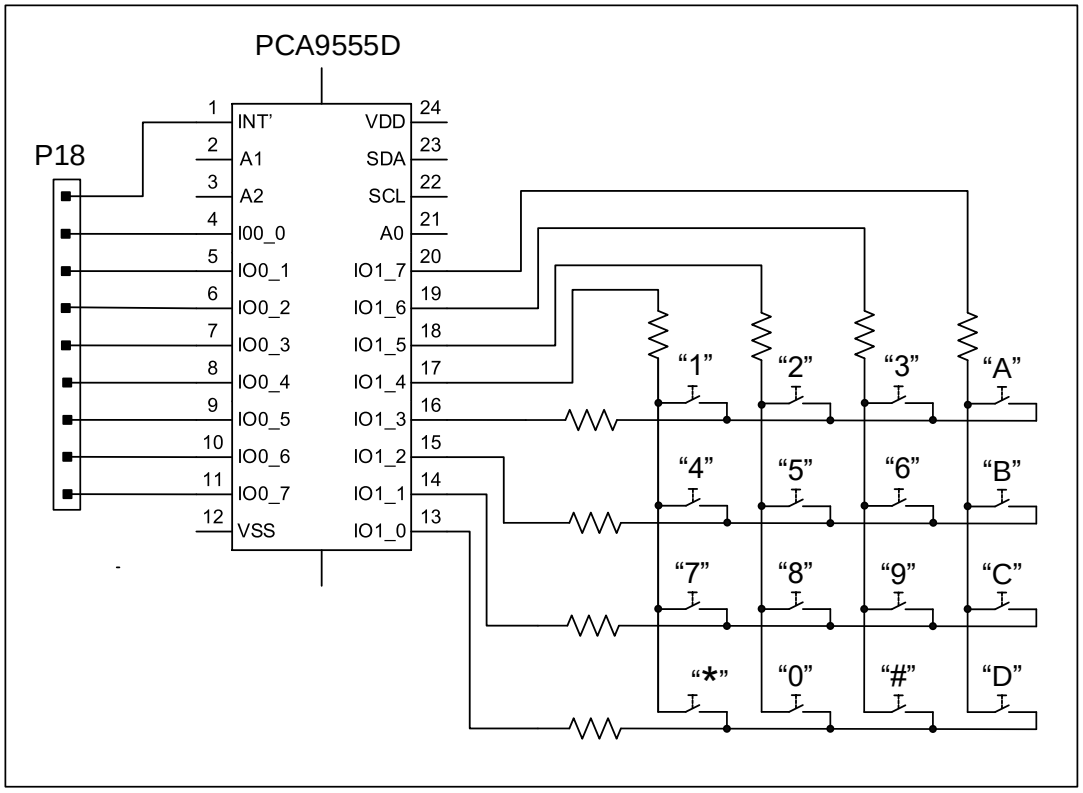
$F0 = (A'B + B'CD)'$

$F1 = (A \cdot C) \cdot (B + D)$

Οι μεταβλητές εισόδου δίνονται στα bit **PORTB [3:0]** (*A= PORTB.0, B= PORTB.1, C= PORTB.2, D= PORTB.3)* του ntuAboard_G1. Οι τιμές των *F*0- *F*1 να εμφανίζονται αντίστοιχα στους ακροδέκτες IO0_0 και IO0_1 του ολοκληρωμένου επέκτασης θυρών PCA9555, στο ntuAboard_G1. Οι ακροδέκτες IO0_0 και IO0_1 του κονέκτορα P18 να συνδεθούν, μέσω καλωδίων, με τους ακροδέκτες LED_PD0 και LED_PD1 του κονέκτορα J18 αντίστοιχα, έτσι ώστε οι τιμές των συναρτήσεων F0 και F1 να εμφανίζονται στα led PD0 PD1.

Ζήτημα 5.2

Στο παρακάτω σχήμα εμφανίζεται η διασύνδεση του ολοκληρωμένου PCA9555D στο ntuAboard_G1.



Προκειμένου να μπορέσει να υπάρξει οπτική απεικόνιση της λογικής κατάστασης των ακροδεκτών IO0_0 έως IO0_3, της θύρας επέκτασης 0, να ρυθμιστούν ως έξοδοι και να συνδεθούν, μέσω καλωδίων, με τους ακροδέκτες LED_PD0 έως LED_PD3 του κονέκτορα J18 αντίστοιχα.

Ο ακροδέκτης IO1_0 της θύρας επέκτασης 1 να ρυθμιστεί ως έξοδος ενώ οι ακροδέκτες IO1_4 έως IO0_7 να ρυθμιστούν ως είσοδοι.

Να υλοποιηθεί κώδικας για το μικροελεγκτή ATmega328PB, σε γλώσσα C, ο οποίος όταν πιέζεται το πλήκτρο "*" να ανάβει το led PD0, όταν πιέζεται το πλήκτρο "0" να ανάβει το led PD 1, όταν πιέζεται το πλήκτρο "#" να ανάβει το led PD2 και όταν πιέζεται το πλήκτρο "D" να ανάβει το led PD3. Αν δεν πιέζεται κανένα πλήκτρο τότε όλα τα led να παραμένουν σβηστά.

Ζήτημα 5.3

Το ζήτημα 5.3 είναι προαιρετικό και η μη υλοποίηση του δεν θα έχει καμία επίπτωση στην βαθμολογία της εργαστηριακής άσκησης.

Να συνδεθεί η LCD οθόνη 2x16 χαρακτήρων, δια μέσω του κονέκτορα J19, στην θύρα επέκτασης 1 του ολοκληρωμένου PCA9555 και να υλοποιηθεί κώδικας για το μικροελεγκτή ATmega328PB, σε γλώσσα C, ο οποίος θα απεικονίζει στην οθόνη το όνομα και το επίθετο σας.