ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

"Η ΕΞΕΛΙΞΗ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ ΒΑΣΙΣΜΕΝΩΝ ΣΤΟΝ 8051"

Πτυχιακή Εργασία

Σπουδαστής: Γεώργιος Αντιπάτης

Υπεύθυνος καθηγητής: Χρήστος Καραΐσκος

Περιεχόμενα

σελ.

ΚΕΦΑΛΑΙΟ 4: Ο μικροελεγκτής XC800

ΕΙΣΑΓΩΓΗ

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι να παρουσιάσει ορισμένα βασικά χαρακτηριστικά μικροελεγκτών.

Ξεκινάει από τον 8051 της Intel, ο οποίος ήταν ένας εμβληματικός μικροελεγκτής για τη δεκαετία του '80 και έθεσε τις βάσεις για να βρουν έδαφος και να αναπτυχθούν δεκάδες άλλοι μικροελεγκτές διευρύνοντας τα μοτίβα του 8051. Στη συνέχεια παρουσιάζονται ο μικροελεγκτής AT85C51SND3 της εταιρείας ATMEL, ο TSC80251 της εταιρείας TEMIC και τέλος ο XC800 της εταιρείας Infineon.

Οι μικροελεγκτές χρησιμοποιούνται σε ευρεία κλίμακα σε τεράστια πληθώρα εφαρμογών όπως υπολογιστικά συστήματα, τηλεπικοινωνίες, αυτοκινητοβιομηχανίες και αποτελούν αναπόσπαστο κομμάτι της τεχνολογικής εξέλιξης της σύγχρονης εποχής.

ΚΕΦΑΛΑΙΟ 1: Ο μικροελεγκτής 8051

1.1 Γενικά

Ο 8051 είναι ένας στοιχειώδης μικροϋπολογιστής αναπτυγμένος σε ένα chip (one-chip microcomputer) των 40 ακροδεκτών. Επειδή είναι σχεδιασμένος για γρήγορο και εκτεταμένο χειρισμό σημάτων από και προς ελεγχόμενες από αυτόν εξωτερικές συσκευές, ονομάζεται μικροελεγκτής.

Αναπτύχθηκε από την Intel το 1980 για χρήση σε ενσωματωμένα συστήματα και παρέμεινε δημοφιλής την δεκαετία του 1980 μέχρι και τις αρχές του 1990. Είχε αρχικά υλοποιηθεί σε NMOS τεχνολογία. Σήμερα ο 8051 διατίθεται και σε Intellectual property core μορφή από εταιρείες όπως π.χ. η Aldec. Στη μορφή αυτή όλα τα λειτουργικά χαρακτηριστικά του 8051 περιγράφονται από γλώσσα περιγραφής υλικού (HDL) όπως η VHDL ή η Verilog HDL. Στον HDL κώδικα η εταιρεία που αγοράζει το IP core προσθέτει νέα τμήματα που περιγράφουν τα επιπρόσθετα εξειδικευμένα χαρακτηριστικά για την εφαρμογή της, μεταφράζει τον κώδικα σε κύκλωμα (synthesis) και τον μεταφέρει σε FPGA ή το διαθέτει σε μορφή νέου εμπορικού ολοκληρωμένου κυκλώματος.

Ένα πλήθος μεγάλων εταιρειών (Philips, Atmel, Maxim, TDK, Analog Devices κτλ) διαθέτουν σήμερα τον 8051 με διάφορες παραλλαγές ως προς τα περιφερειακά που έχουν ενσωματωμένα καθώς και το μέγεθος της μνήμης RAM και ROM. Υλοποιείται κυρίως σε CMOS τεχνολογία και για το λόγο αυτό συχνά προστίθεται στο όνομά του το γράμμα C

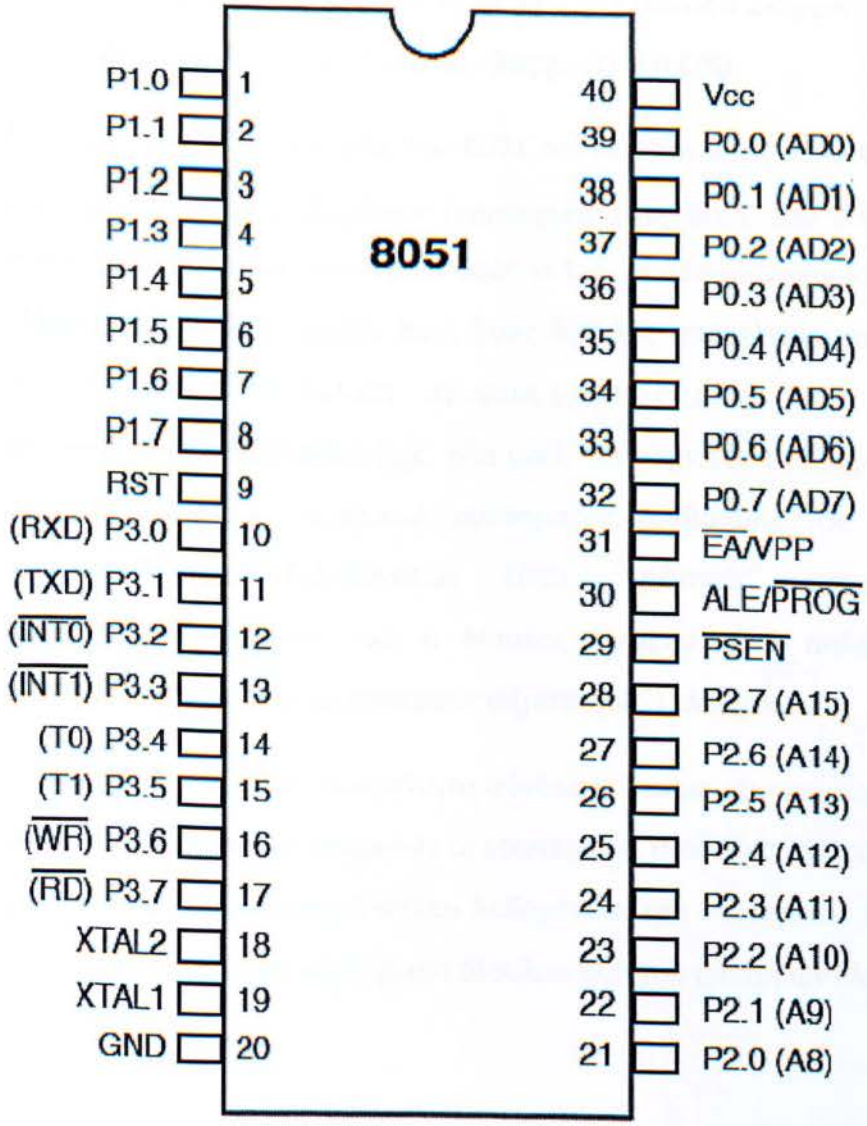
(80C51). Όλες οι εταιρείες αυτές διαθέτουν σήμερα τον 8051 με ενσωματωμένη μνήμη Flash διαφόρων μεγεθών η οποία προγραμματίζεται εύκολα ακόμα και πάνω στην πλακέτα της εφαρμογής στην οποία χρησιμοποιείται το ολοκληρωμένο.

Τα κύρια στοιχεία που έχει πάνω του ο 8051 είναι η μονάδα επεξεργασίας δεδομένων, η μονάδα ελέγχου (στα συστήματα μικροϋπολογιστών αυτές οι δύο μονάδες μαζί αποτελούν τον μικροεπεξεργαστή), η εσωτερική μνήμη RAM για προσωρινή αποθήκευση δεδομένων και η εσωτερική μνήμη ROM ή EPROM για μόνιμη αποθήκευση προγραμμάτων και δεδομένων. Προς τον έξω κόσμο παρέχει τέσσερις παράλληλες πόρτες των 8 bits για σύνδεση με εξωτερικές συσκευές.

Ένα ιδιαίτερα χρήσιμο χαρακτηριστικό του πυρήνα του 8051 είναι η συμπερίληψη μιας μηχανής επεξεργασίας η οποία επιτρέπει τη διεξαγωγή λειτουργιών σε επίπεδο bit, βασιζόμενες σε άλγεβρα Bool, απ'ευθείας και αποτελεσματικά σε συγκεκριμένους εσωτερικούς καταχωρητές και θέσεις μνήμης RAM. Αυτό το πλεονεκτικό χαρακτηριστικό βοήθησε στο να καθιερωθεί ο 8051 σε εφαρμογές βιομηχανικού ελέγχου επειδή μείωνε το μέγεθος του κώδικα μέχρι και 30%.

Ένα άλλο πολύτιμο χαρακτηριστικό είναι η συμπερίληψη τεσσάρων σετ καταχωρητών με δυνατότητα επιλογής τράπεζας τα οποία μειώνουν σε μεγάλο βαθμό την ποσότητα του χρόνου που απαιτείται για να ολοκληρωθεί μια ρουτίνα εξυπηρέτησης διακοπών. Με μία και μόνο εντολή ο 8051 έχει τη δυνατότητα να εναλλάσει τράπεζες καταχωρητών σε αντίθεση με τον χρονοβόρα διαδικασία της μετακίνησης των

κρίσιμων καταχωρητών στο σωρό ή σε αναδεδειγμένες θέσεις μνήμης RAM. Επίσης αυτοί οι καταχωρητές επιτρέπουν στον 8051 να εκτελέσει μια γρήγορη εναλλαγή πλαισίου.



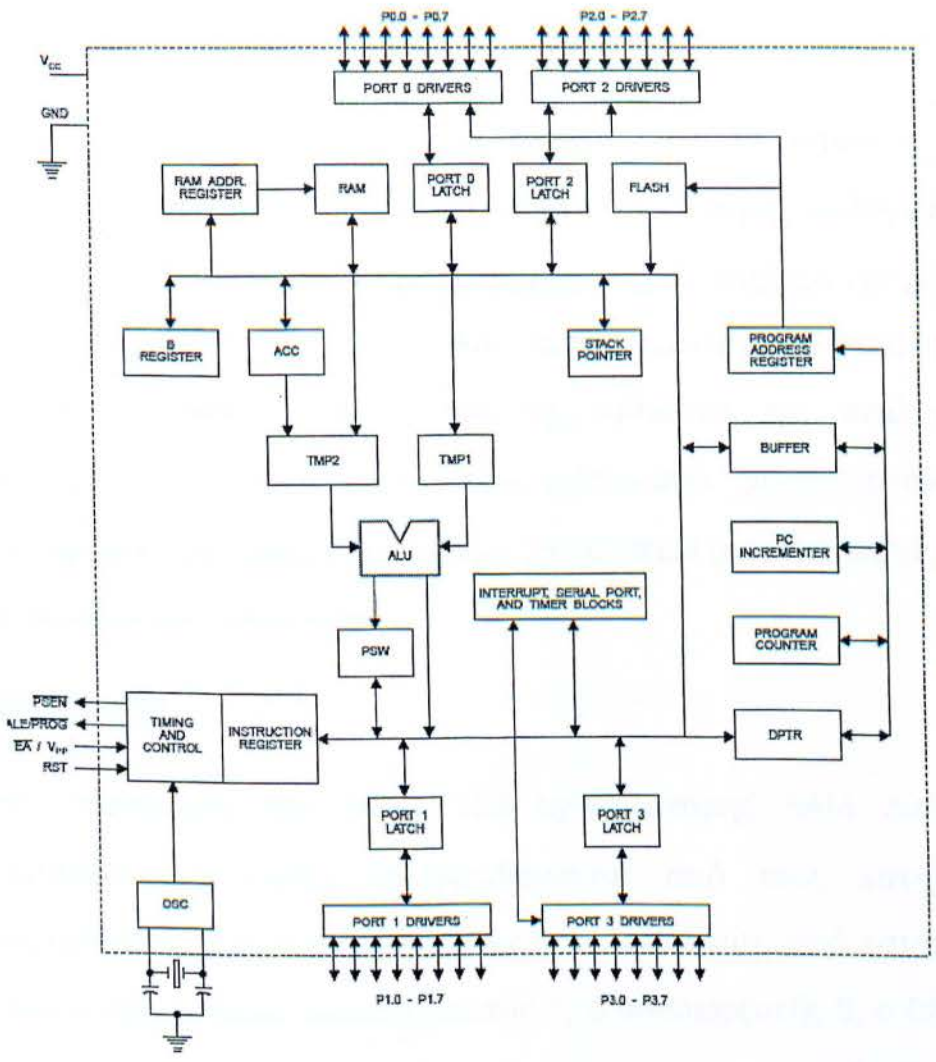
Οι ακροδέκτες του 8051

1.2 Η δομή του 8051

Η Κεντρική Μονάδα Επεξεργασίας (CPU) που είναι η καρδιά του συστήματος περιλαμβάνει την αριθμητική και λογική μονάδα (ALU), το σετ των καταχωρητών, την μονάδα προσκόμισης και αποκωδικοποίησης εντολών (instruction fetch and decoding), την μονάδα ελέγχου (control unit) καθώς και οτιδήποτε άλλο περιλαμβάνει μια CPU.

Αυτά τα εσωτερικά στοιχεία του 8051 συνδέονται μεταξύ τους με τον εσωτερικό δίαυλο δεδομένων (εσωτερικό data bus), τον εσωτερικό δίαυλο διαυθύνσεων (εσωτερικό address bus) και τον εσωτερικό δίαυλο ελέγχου (εσωτερικό control bus). Ένας δίαυλος αποτελείται από έναν αριθμό από παράλληλους αγωγούς-γραμμές μέσω των οποίων μεταφέρονται παράλληλα, δηλ. όλα μαζί, ταυτόχρονα, τα ψηφία (bits) ενός αριθμού του δυαδικού συστήματος αρίθμησης. Οι δίαυλοι δεδομένων και διευθύνσεων είναι "εμφανείς" σχηματισμοί παράλληλων γραμμών, ενώ ο δίαυλος ελέγχου είναι απλώς ένας αριθμός γραμμών που μεταφέρουν σήματα (bits) ελέγχου.

Ο 8051 έχει επίσης τη δυνατότητα σύνδεσης και με εξωτερικές μνήμες ROM και RAM, αν δεν επαρκούν οι εσωτερικές. Η σύνδεση γίνεται με τη βοήθεια του εξωτερικού δίαυλου δεδομένων, του εξωτερικού δίαυλου διαυθύνσεων και του εξωτερικού δίαυλου ελέγχου (γραμμών ελέγχου).



Η δομή του 8051

Οι περισσότεροι ακροδέκτες του 8051 έχουν εκτός από κάποια βασική λειτουργία πολυπλεγμένη και κάποια δεύτερη, η οποία για να χρησιμοποιηθεί θα πρέπει να γίνει η κατάλληλη διαμόρφωση (configuration) σε σχετικό εσωτερικό καταχωρητή ελέγχου. Οι πολυπλεγμένες λειτουργίες μπορεί να διαφέρουν ελαφρά ανάμεσα σε σύγχρονες εκδόσεις του 8051 λόγω κάποιων επιπλέον περιφερειακών που είναι ενσωματωμένα.

1.3 Οργάνωση μνήμης

Ο 8051 ακολουθεί το μοντέλο Harvard κατά το οποίο η μνήμη προγράμματος είναι ξεχωριστή από την μνήμη δεδομένων. Ο διαχωρισμός διευθύνσεων προγράμματος και δεδομένων επιτρέπει την προσπέλαση θέσεων μνήμης και προγράμματος με διαφορετικού μήκους διευθύνσεις και μπορεί να ρυθμιστεί και ανάλογα την εφαρμογή. Έτσι στις περισσότερες εφαρμογές μικροϋπολογιστικών συστημάτων, μια μνήμη δεδομένων 256 θέσεων (προσπελάσιμη από 8-bit διευθύνσεις) είναι αρκετή.

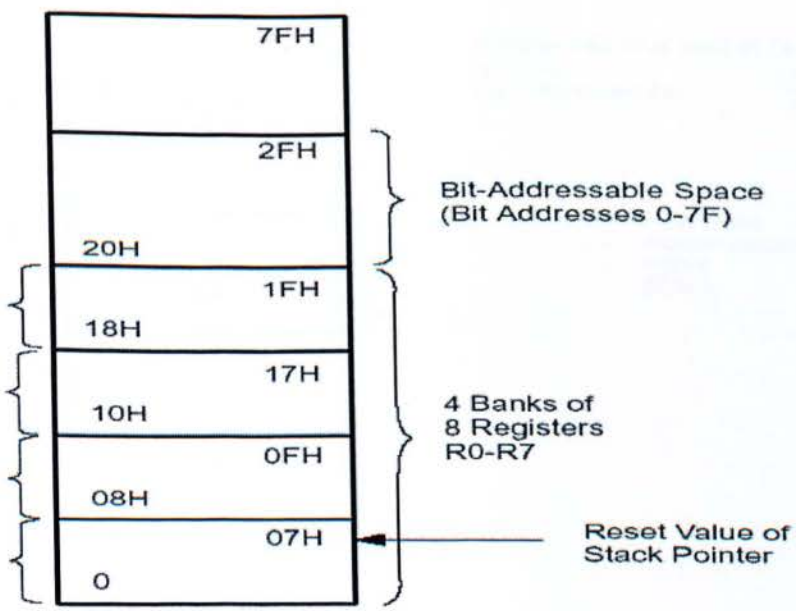
Μνήμη RAM

Στην περίπτωση του 8051, 128 bytes μνήμης RAM διατίθενται εσωτερικά, τα οποία καταλαμβάνονται από τους καταχωρητές διαμόρφωσης των ενσωματωμένων περιφερειακών, από καταχωρητές ειδικού τύπου όπως ο συσσωρευτής A, ο συσσωρευτής B, ο DPTR κλπ, ενώ οι υπόλοιπες θέσεις από γενικής χρήσης καταχωρητές. Το πλήθος των εσωτερικών καταχωρητών μπορεί να φτάσει τους 384. Παρ'όλα αυτά στο 8051 είναι δυνατή διευθυνσιοδότηση εξωτερικής μνήμης δεδομένων 16-bit για την κάλυψη εφαρμογών που έχουν ανάγκη από περισσότερη μνήμη RAM. Μια τέτοια δυνατότητα δίνεται με τη χρήση του 16-μπιτου καταχωρητή DPTR.

Όσον αφορά την μνήμη προγράμματος αυτή μπορεί να φτάσει τα 64K, ενώ στις πιο κλασσικές εκδόσεις του 8051 υπάρχει μια εσωτερική μνήμη προγράμματος 4K η οποία επεκτείνεται εξωτερικά.

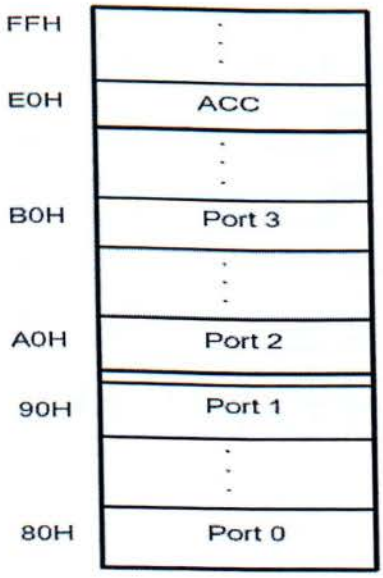
Οι διευθύνσεις της εσωτερικής μνήμης RAM, αφού είναι 128, χρειάζονται 7 bits για να γραφούν. Έτσι οι διευθύνσεις των θέσεων της

εσωτερικής μνήμης RAM παίρνουν τιμές από 0000000B = 0 μέχρι 1111111B = 127. Μέσα στον 8051 υπάρχουν κι άλλες θέσεις, σε αριθμό μέχρι 128, στις οποίες γίνεται γραφή και ανάγνωση bytes. Αυτές οι θέσεις ονομάζονται καταχωρητές ειδικών λειτουργιών (Special Function Registers - SFR). Χρειάζεται και αυτές να έχουν τις δικές τους διευθύνσεις RAM. Για αυτό το λόγο, ο 8051 χρησιμοποιεί 8-bit διευθύνσεις για εσωτερικές θέσεις γραφής και ανάγνωσης. Έτσι με 8-bit διευθύνσεις η εσωτερική μνήμη RAM του 8051 καταλαμβάνει τις θέσεις με διευθύνσεις 0000000B = 00H = 0 μέχρι 1111111B = 7FH = 127 και οι SFR καταλαμβάνουν μερικές διευθύνσεις στην περιοχή διευθύνσεων από 10000000 = 80H μέχρι 11111111B = FFH. Οι υπόλοιπες διευθύνσεις στο μπλοκ 80H - FFH μένουν αχρησιμοποίητες (δεν αντιστοιχούν σε θέσεις αποθήκευσης bytes).



Τα χαμηλότερα 128 bytes της εσωτερικής μνήμης RAM

Οι θέσεις της εσωτερικής μνήμης RAM που έχουν διευθύνσεις 00H - 07H αναφέρονται στη γλώσσα του 8051 και ως 8 καταχωρητές με ονόματα R0 - R7, αντίστοιχα. Συνιστούν την ομάδα καταχωρητών 0 (Register Bank 0). Οι θέσεις της εσωτερικής μνήμης RAM που έχουν διευθύνσεις 08H - 0FH αναφέρονται και αυτές ως καταχωρητές R0 - R7, αλλά της ομάδας καταχωρητών 1 (Register Bank 1). Ομοίως οι θέσεις με διευθύνσεις 10H - 17H αναφέρονται και ως καταχωρητές R0 - R7 της ομάδας καταχωρητών 2 (Register Bank 2) και οι θέσεις με διευθύνσεις 18H - 1FH αναφέρονται και ως καταχωρητές R0 - R7 της ομάδας καταχωρητών 3 (Register Bank 3). Δηλ. ο 8051 διαθέτει 4 ομάδες καταχωρητών, των 8 καταχωρητών η κάθε ομάδα.



Register-Mapped Ports

Addresses that end in 0H or 8H are also bit-addressable.

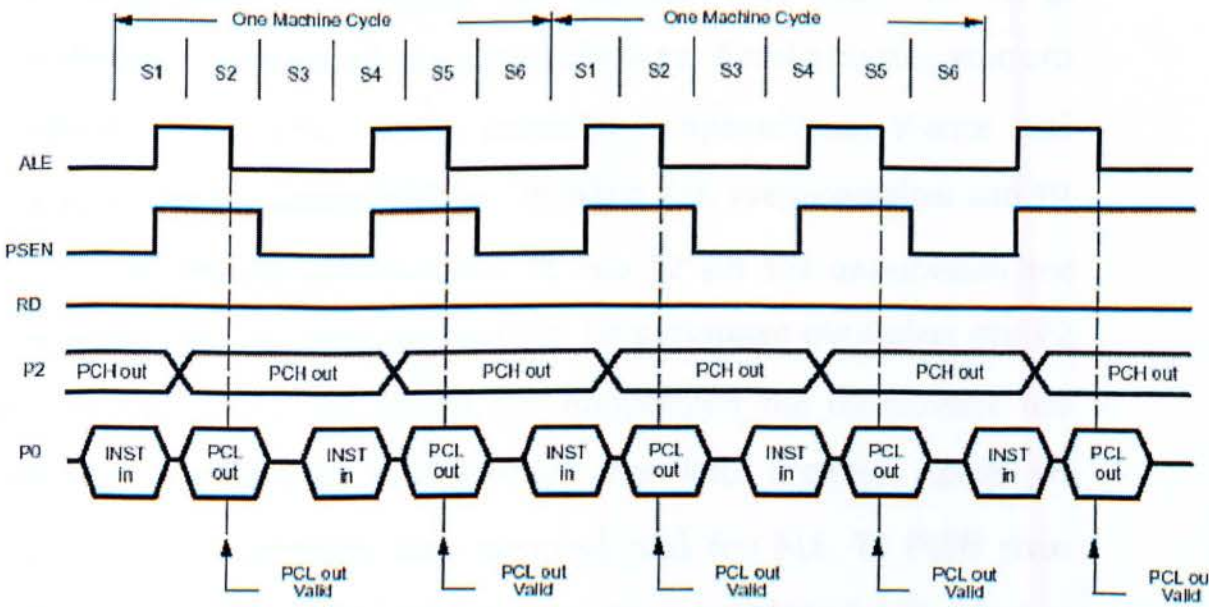
- Port Pins
- Accumulator
- PSW
(Etc.)

Οι καταχωρητές ειδικών λειτουργιών (SFR)

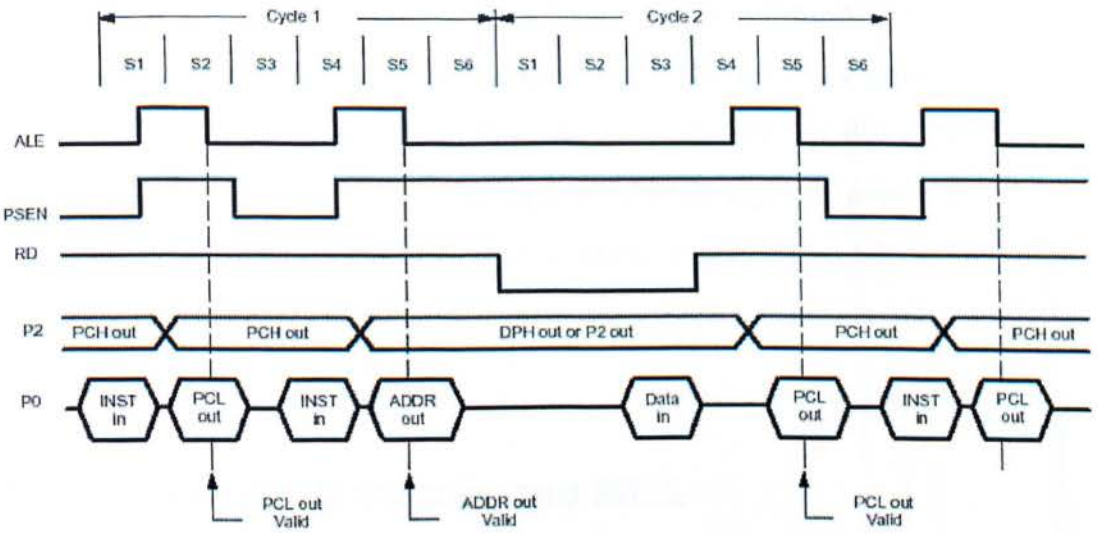
1.4 Οι κύκλοι μηχανής του 8051

Οι εντολές του 8051 αποτελούνται από 1 ή 2 κύκλους μηχανής. Κάθε κύκλος μηχανής αποτελείται από 6 στάδια (states) που ονομάζονται S1 εώς S6. Κάθε στάδιο αποτελείται από 2 παλμούς ρολογιού, την φάση 1 (P1) και την φάση 2 (P2). Συνολικά ο χρόνος εκτέλεσης ενός κύκλου μηχανής διαρκεί 12 παλμούς ρολογιού.

Σε κάθε κύκλο μηχανής προσκομίζονται 2 bytes από την μνήμη προγράμματος στα στάδια S1 και S4. Αν η εντολή είναι του ενός byte τότε το δεύτερο αγνοείται και ο μετρητής προγράμματος PC δεν αυξάνεται.



Χωρίς εντολή MOV



Με εντολή MOV

Στο σχήμα φαίνεται η μορφή των σημάτων ALE, PSEN, RD και οι ακροδέκτες των P0 και P2 που σχηματίζουν τον δίαυλο συστήματος στα διάφορα στάδια ενός κύκλου μηχανής. Η προσπέλαση γίνεται από εξωτερική μνήμη προγράμματος. Το σήμα ALE ενεργοποιείται στο P2 του S1 και απενεργοποιείται στο P1 του S2 για την απομόνωση της διεύθυνσης του op code της εντολής. Το αντίστοιχο συμβαίνει στο P2 του S4 και στο P1 του S5 για την απομόνωση της διεύθυνσης του ορίσματος της εντολής. Το ALE οδηγεί έναν latch ο οποίος πρέπει να κλειδώνει την διεύθυνση στην αρνητική τιμή του ALE. Το PSEN είναι active low σήμα το οποίο ενεργοποιείται αφού η πλήρης διεύθυνση που πρόκειται να διαβαστεί έχει σταθεροποιηθεί στην θύρα P2 και στην έξοδο του latch. Στην θετική ακμή του PSEN η τιμή που θα διαβαστεί έχει ήδη σταθεροποιηθείστους ακροδέκτες της θύρας P0.

Η εντολή MOV που χρησιμοποιείται για έμμεση διευθυνσιοδότηση μέσω των R0, R1έχει διαφορετική μορφή. Απαιτεί 2 κύκλους μηχανής, δεδομένου ότι πρέπει εσωτερικά να μεταφέρει την τιμή των R0 ή R1 στον δίαυλο διευθύνσεων στο S5 του πρώτου κύκλου μηχανής ώστε να πραγματοποιηθεί η προσπέλαση της μνήμης στους S1 - S3 του δεύτερου κύκλου μηχανής.

1.5 Το σύστημα διακοπών του 8051

Απο τους 40 ακροδέκτες του 8051, ο ακροδέκτης RESET (RST) προκαλεί διακοπή της λειτουργίας του 8051 και επανεκκίνησή του. Διακοπές ενός προγράμματος που τρέχει ο 8051 μπορούν επίσης να προκληθούν:

α) Αν κάποιον από τους ακροδέκτες P3.2 ή P3.3 τον οδηγήσουμε σε λογικό 0 ή αν του δώσουμε αρνητικό μέτωπο τάσης

β) αν κάποιος από τους απαριθμητές T0 και T1, που έχουμε βάλει να μετράει, υπερχειλίσει.

Υπερχείλιση ενός απαριθμητή έχουμε όταν η ένδειξη του μεταβαίνει από τη μέγιστη τιμή της (όλο 1) στην ελάχιστη τιμή της (συνήθως όλο 0). Τέλος, η διακοπή του προγράμματος που τρέχει ο 8051 μπορεί να προκληθεί και από τη λειτουργία της σειριακής πόρτας του 8051.

Σε κάθε περίπτωση, όταν προκληθεί μια διακοπή της εκτέλεσης ενός προγράμματος που τρέχει ο 8051, πραγματοποιείται μια κλήση ρουτίνας, η οποία ονομάζεται ρουτίνα εξυπηρέτησης της διακοπής αυτής, από συγκεκριμένη αντίστοιχη διεύθυνση της μνήμης

προγράμματος (μνήμης ROM ή EPROM). Αυτή η διεύθυνση ονομάζεται άνυσμα ή διάνυσμα διακοπής της συγκεκριμένης διακοπής.

Άνυσμα διακοπής της διακοπής RESET είναι το 0000H, της εξωτερικής διακοπής P3.2 είναι το 0003H, της διακοπής από υπερχείλιση του απαριθμητή T0 είναι το000BH, της εξωτερικής διακοπής P3.3 είναι το 0013H, της διακοπής από υπερχείλιση του απαριθμητή T1 είναι το 001BH και της διακοπής από τη λειτουργία της σειριακής πόρτας είναι το 0023H.

Προφανώς επειδή στη συντριπτική πλειονότητα των περιπτώσεων μια ρουτίνα εξυπηρέτησης διακοπής δεν χωράει να γραφεί μεταξύ δύο ανυσμάτων διακοπής (μεταξύ αυτών υπάρχουν μόνο 3 ή 8 θέσεις μνήμης προγράμματος), στη θέση κάθε ανύσματος διακοπής γράφουμε απλώς μια εντολή άλματος προς κάποια άλλη θέση της μνήμης προγράμματος, όπου έχουμε γράψει με την άνεσή μας τη ρουτίνα εξυπηρέτησης της διακοπής, ή προς το κυρίως πρόγραμμα (όταν πρόκειται για RESET).

Οι διακοπές ενεργοποιούνται στα bits του καταχωρητή IE (Interrupt Enable) που έχει διεύθυνση A8H. Τα bits του καταχωρητή IE με τα συμβολικά τους ονόματα είναι τα εξής:

	EA	X	X	ES	ET1	EX1	ET0	EX0
bits	7	6	5	4	3	2	1	0
address	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H

Μια διακοπή μπορεί να προγραμματιστεί να είναι υψηλής ή χαμηλής προτεραιότητας, Αυτό το κάνουν τα bits του καταχωρητή IP (Interrupt Priority) που έχει διεύθυνση B8H. Τα bits του καταχωρητή IP με τα συμβολικά τους ονόματα είναι τα εξής:

	-	X	X	PS	PT1	PX1	PT0	PX0
bits	7	6	5	4	3	2	1	0
address	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H

1.6 Οι εντολές του 8051

8051 Instruction Set Summary

Rn Register R7-R0 of the currently selected Register Bank.
Data 8-bit internal data location's address. This could be an internal Data RAM location (0-127) or a SFR [i.e. I/O port, control register, status register, etc. (128-255)].
@Ri 8-bit internal Data RAM location (0-255) addressed indirectly through register R1 or R0.
#data 8-bit constant included in instruction.
#data16 16-bit constant included in instruction.
addr16 16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64k byte Program Memory address space.
addr11 11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k byte page of Program Memory as the first byte of the following instruction.
rel Signed (two's component) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit Direct Addressed bit in Internal Data RAM or Special Function Register.

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Note that operations on SFR byte address 206 or bit addresses 209-215 (i.e. the PSW or bits in the PSW) will also affect flag settings.

Mnemonic	Description	Byte	Cycle
Arithmetic operations			
ADD A,Rn	Add register to accumulator	1	1
ADD A,direct	Add direct byte to accumulator	2	1
ADD A,@Ri	Add indirect RAM to accumulator	1	1
ADD A,#data	Add immediate data to accumulator	2	1
ADDC A,Rn	Add register to accumulator with carry flag	1	1
ADDC A,direct	Add direct byte to A with carry flag	2	1
ADDC A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC A,#data	Add immediate data to A with carry flag	2	1
SUBB A,Rn	Subtract register to accumulator with borrow	1	1
SUBB A,direct	Subtract direct byte to A with carry borrow	2	1
SUBB A,@Ri	Subtract indirect RAM to A with carry borrow	1	1
SUBB A,#data	Subtract immediate data to A with carry borrow	2	1
INC A	Increment accumulator	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	1
INC @Ri	Increment indirect RAM	1	1
DEC A	Decrement accumulator	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	1
DEC @Ri	Decrement indirect RAM	1	1
INC DPTR	Increment data pointer	1	2
MUL AB	Multiply A and B -> [B hi]:[A lo]	1	4
DIV AB	Divide A by B -> A=result, B=remainder	1	4
DA A	Decimal adjust accumulator	1	1
CLR A	Clear accumulator	1	1

Mnemonic		Description	Byte	Cycle
CPL	A	Complement accumulator	1	1
RL	A	Rotate accumulator left	1	1
RLC	A	Rotate accumulator left through carry	1	1
RR	A	Rotate accumulator right	1	1
RRC	A	Rotate accumulator right through carry	1	1
SWAP	A	Swap nibbles within the accumulator	1	1
Logic operations				
ANL	A,Rn	AND register to accumulator	1	1
ANL	A,direct	AND direct byte to accumulator	2	1
ANL	A,@Ri	AND indirect RAM to accumulator	1	1
ANL	A,#data	AND immediate data to accumulator	2	1
ANL	direct,A	AND accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to accumulator	1	1
ORL	A,direct	OR direct byte to accumulator	2	1
ORL	A,@Ri	OR indirect RAM to accumulator	1	1
ORL	A,#data	OR immediate data to accumulator	2	1
ORL	direct,A	OR accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive OR register to accumulator	1	1
XRL	A,direct	Exclusive OR direct byte to accumulator	2	1
XRL	A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL	A,#data	Exclusive OR immediate data to accumulator	2	1
XRL	direct,A	Exclusive OR accumulator to direct byte	2	1
XRL	direct,#data	Exclusive OR immediate data to direct byte	3	2
Boolean variable manipulation				
CLR	C	Clear carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set carry flag	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to carry flag	2	2
ANL	C,/bit	AND complement of direct bit to carry	2	2
ORL	C,bit	OR direct bit to carry flag	2	2
ORL	C,/bit	OR complement of direct bit to carry	2	2
MOV	C,bit	Move direct bit to carry flag	2	1
MOV	bit,C	Move carry flag to direct bit	2	2
Program and machine control				
ACALL	addr11	Absolute subroutine call	2	2
LCALL	addr16	Long subroutine call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr11	Absolute jump	2	2
LJMP	addr16	Long jump	3	2
SJMP	rel	Short jump (relative address)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if accumulator is zero	2	2
JNZ	rel	Jump if accumulator is not zero	2	2
JC	rel	Jump if carry flag is set	2	2
JNC	rel	Jump if carry flag is not set	2	2
JB	bit,rel	Jump if bit is set	3	2
JNB	bit,rel	Jump if bit is not set	3	2
JBC	bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE	A,direct,rel	Compare direct byte to A and jump if not equal	3	2

Mnemonic	Description	Byte	Cycle
CJNE A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE Rn,#data,rel	Compare immed. to reg. and jump if not equal	3	2
CJNE @Rn,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ Rn,rel	Decrement register and jump in not zero	2	2
DJNZ direct,rel	Decrement direct byte and jump in not zero	3	2
NOP	No operation	1	1
Data transfer			
MOV A,Rn	Move register to accumulator	1	1
MOV A,direct*)	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri,#data	Move immediate data to indirect RAM	2	1
MOV DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A+DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A+PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A,Rn	Exchange register to accumulator	1	1
XCH A,direct	Exchange direct byte to accumulator	2	1
XCH A,@Ri	Exchange indirect RAM to accumulator	1	1
XCHD A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

*) MOV A,ACC is not a valid instruction

jne A,#data,@ *(jump if A ! = data)*	cjne	A,#data,@				
je A, #data,@ *(jump if A == data)*	add jz	A,#low(-data) @	*or*	ne:	cjne jmp ...	A,#(data),ne @
ja, jnbe A,#data,@ *(jump if A > data)*	add jc	A,#low(-data-1) @	*or*	ne:	cjne jnc	A,#(data+1),ne @
jae, jnb A,#data,@ *(jump if A >= data)*	add jc	A,#low(-data) @	*or*	ne:	cjne jnc	A,#(data),ne @
jb, jnae A,#data,@ *(jump if A < data)*	add jnc	A,#low,(-data) @	*or*	ne:	cjne jc	A,#(data),ne @
jbe, jna A,#data,@ *(jump if A <= data)*	add jnc	A,#low(-data-1) @	*or*	ne:	cjne jc	A,#(data+1),ne @
switch A <,==,> #data *(no A modification)*	cjne ... ne: jc jnc	A,#data,ne is_below is_above			; execute code if A==data ; jump if A<data ; jump if A>data or exec. code	

1.7 Οι καταχωρητές ειδικών λειτουργιών του 8051

8051 special function registers (SFRs)

Symbol	Name	Address	Contents after reset
ACC*	Accumulator	0E0H	00000000
B*	B register	0F0H	00000000
PSW*	Program Status Word	0D0H	00000000
SP	Stack Pointer	81H	00000111
DPTR	Data Pointer 2 bytes		
DPL	Data Pointer low byte	82H	00000000
DPH	Data Pointer high byte	83H	00000000
P0*	Port 0	80H	11111111
P1*	Port 1	90H	11111111
P2*	Port 2	0A0H	11111111
P3*	Port 3	0B0H	11111111
IP*	Interrupt Priority Control	0B8H	XXX00000
IE*	Interrupt Enable Control	0A8H	0XX00000
TMOD	Timer/Counter Mode Control	89H	00000000
TCON*	Timer/Counter Control	88H	00000000
TH0	Timer/Counter 0 high byte	8CH	00000000
TL0	Timer/Counter 0 low byte	8AH	00000000
TH1	Timer/Counter 1 high byte	8DH	00000000
TL1	Timer/Counter 1 low byte	8BH	00000000
SCON*	Serial Control	98H	00000000
SBUF	Serial Data Buffer	99H	Indeterminate
PCON	Power Control	87H	0XXXXXXX

(= bit addressable)*

ΚΕΦΑΛΑΙΟ 2: Ο μικροελεγκτής AT85C51SND3

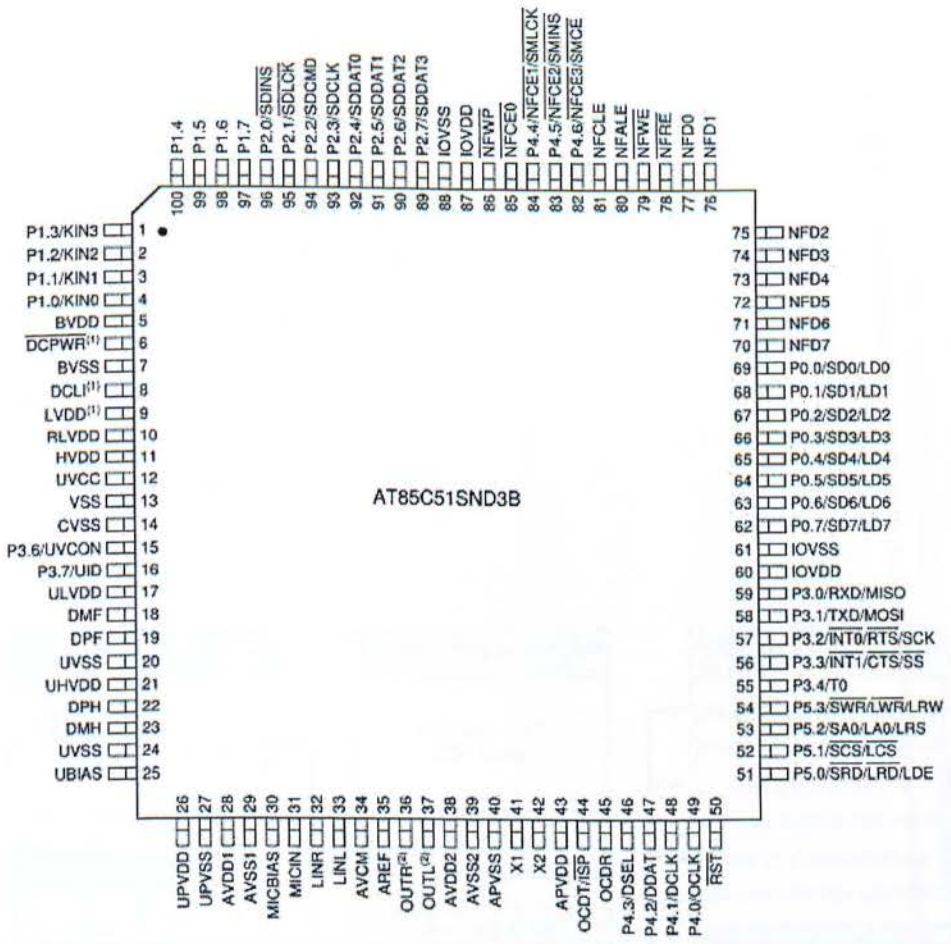
2.1 Γενικά

Συσκευές αναπαραγωγής ψηφιακής μουσικής, κινητά τηλέφωνα χρειάζονται έτοιμες προς χρήση λύσεις χαμηλού κόστους με όσο το δυνατόν μικρότερο χρόνο κατασκευής. Ο μικροελεγκτής AT85C51SND3 της εταιρείας ATMEL με το σχετικό λογισμικό, ενσωματώνει σε ένα και μόνο τσιπ, όλα τα χαρακτηριστικά, υλικό και λογισμικό, για συσκευές αναπαραγωγής ψηφιακών μουσικών αρχείων, κινητά τηλέφωνα και ηχοσυστήματα αυτοκινήτων. Δηλαδή αποκωδικοποιητές αρχείων τύπου MP3, WMA, διεπαφές σειριακού και παράλληλου τύπου, USB υψηλής ταχύτητας.

Κοντά σε λύσεις τύπου "plug and play" για τις περισσότερες εφαρμογές, ο AT85C51SND3 μειώνει δραστικά τη διαδικασία ανάπτυξης για τον καλύτερο χρόνο κατασκευής. Ο AT85C51SND3 κάνει πλήρη διαχείριση συστήματος με κάρτες τύπου Flash. Ο AT85C51SND3 χρησιμοποιείται είτε ως Master ελεγκτής ή ως Slave ελεγκτής αλληλεπιδρώντας εύκολα με τους περισσότερους επεξεργαστές της αγοράς.

Πέραν των αρχείων τύπου MP3, WMA, το λογισμικό του AT85C51SND3 θα υποστηρίζει αργότερα και αρχεία τύπου OGG, βασικά χαρακτηριστικά MIDI για κινητά τηλέφωνα χαμηλού κόστους καθώς και αποκωδικοποίηση εικόνων JPEG.

Για να διευκολύνει τη δημιουργία προσαρμοσμένων εφαρμογών σε συγκεκριμένα κριτήρια, είναι διαθέσιμο για τον AT85C51SND3 ένα κιτ ανάπτυξης με βάση δεδομένων υλικού και λογισμικού.



Οι ακροδέκτες του AT85C51SND3

Κύρια χαρακτηριστικά:

Λογισμικό υποστήριξης αρχείων:

- MP3
- WMA
- ADPCM/WAV
- OGG, MIDI, JPEG

Codec ήχου:

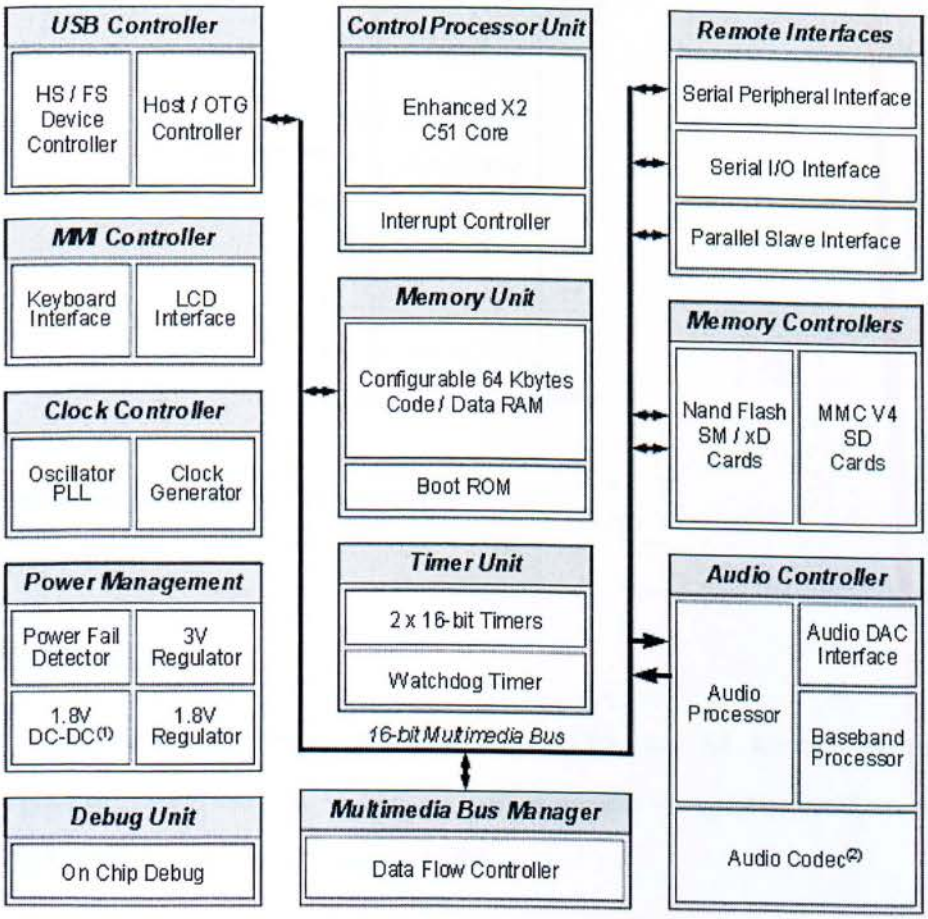
- Εσωτερικό DAC
- Σήμα εισόδου FM

Μνήμη:

- Μέχρι και 4x Nand-Flash
- Κάρτες SD/MMC

USB:

- High Speed
- OTG

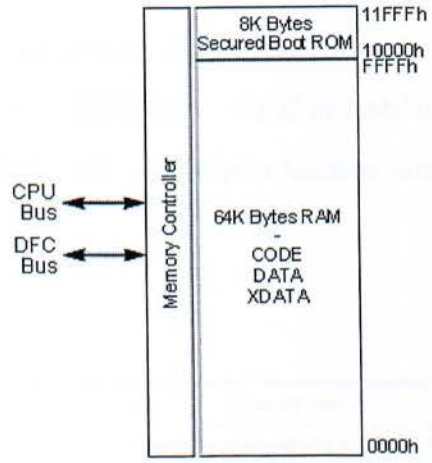


Το μπλοκ διάγραμμα του AT85C51SND3

2.2 Χώρος μνήμης του AT85C51SND3

Ο AT85C51SND3 παρέχει 64 Kbytes μνήμης RAM χωρισμένα στους 3 πρότυπους τομείς μνήμης του C51:

- CODE
- DATA
- XDATA



Οργάνωση μνήμης του AT85C51SND3

Τομέας CODE

Ο AT85C51SND3 μπορεί να εκτελέσει μέχρι και 64 Kbytes μνήμης προγράμματος/κώδικα. Ο AT85C51SND3 μπορεί να χρησιμοποιήσει και επιπροσθέτως 4 Kbytes μνήμης ROM.

Τομέας DATA

Ο τομέας DATA χωρίζεται σε:

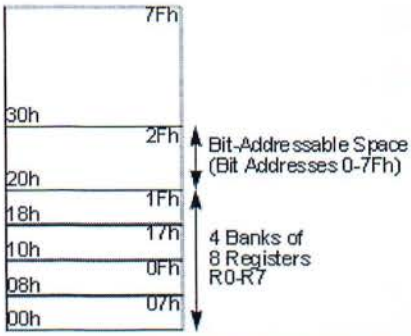
- Τα χαμηλότερα 128 Bytes μνήμης RAM
- Τα υψηλότερα 128 Bytes μνήμης RAM

Τα χαμηλότερα 128 Bytes μνήμης RAM είναι προσπελάσιμα στις διευθύνσεις από 00H μέχρι 7FH χρησιμοποιώντας έμμεσους ή άμεσους τρόπους διευθυνσιοδότησης. Τα χαμηλότερα 32 bytes χωρίζονται σε 4 ομάδες των 8 καταχωρητών (R0 μέχρι R7). Τα bits RS0 και RS1 στον καταχωρητή PSW επιλέγουν ποια ομάδα θα χρησιμοποιείται. Αυτό επιτρέπει πιο αποτελεσματική χρήση του χώρου κώδικα, μιας και οι εντολές καταχωρητών είναι μικρότερες από τις εντολές που χρησιμοποιούν απευθείας διευθυνσιοδότηση και μπορούν να χρησιμοποιηθούν για εναλλαγή πλαισίου στις ρουτίνες εξυπυρέτησης διακοπών.

RS1	RS0	Description
0	0	Register bank 0 from 00h to 07h
0	1	Register bank 1 from 08h to 0Fh
1	0	Register bank 2 from 10h to 17h
1	1	Register bank 3 from 18h to 1Fh

Επιλογή τράπεζας καταχωρητών

Τα επόμενα 16 bytes πάνω από την ομάδα καταχωρητών σχηματίζουν ένα μπλοκ bit-addressable χώρου μνήμης. Το σετ εντολών του C51 περιλαμβάνει ευρεία επιλογή εντολών του ενός bit και τα 128 bits αυτής της περιοχής μπορούν να δειυθυνσιοδοτηθούν άμεσα μέσω αυτών των εντολών. Η διευθύνσεις των bit σε αυτήν την περιοχή είναι από 00H μέχρι 7FH.



Οργάνωση εσωτερικής RAM στα κάτω 128 bytes

Τα υψηλότερα 128 bytes της μνήμης RAM είναι προσπελάσιμα στις διευθύνσεις από 80h μέχρι FFh χρησιμοποιώντας μόνο έμμεσο τρόπο διευθυνσιοδότησης. Χρησιμοποιώντας άμεση διευθυνσιοδότηση σε αυτή την περιοχή επιλέγονται οι καταχωρητές ειδικής λειτουργίας (Special Function Registers - SFR's).

F8									FF
F0	**B**								F7
E8									EF
E0	**ACC**								E7
D8									DF
D0	**PSW**					SPCR			D7
C8	**T2CON**	T2MOD	RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	**IP**	SADEN							BF
B0	**P3**							IPH	B7
A8	**IE**	SADDR	SPSR						AF
A0	**P2**						WDTRST	WDTCON	A7
98	**SCON**	SBUF							9F
90	**P1**						EECON		97
88	**TCON**	TMOD	TL0	TL1	TH0	TH1	AUXR	CLKREG	8F
80	**P0**	SP	DP0L	DP0H	DP1L	DP1H	SPDR	PCON	87

Οι καταχωρητές ειδκών λειτουργιών του AT85C51SND3

Τομέας XDATA

Η επεκταμένη μνήμη RAM του τσιπ (XRAM), είναι προσπελάσιμη χρησιμοποιώντας έμμεση διευθυνσιοδότηση με τις εντολές τύπου MOV.

2.3 Το σύστημα διακοπών του AT85C51SND3

Ο AT85C51SND3 χρησιμοποιεί μια μέθοδο διακοπών προγράμματος, όπως κάθε επεξεργαστής προσανατολιζόμενος σε έλεγχο. Αυτή η λειτουργία διακλαδώνεται σε μια υπορουτίνα και εκτελεί μια υπηρεσία σε ανταπόκριση της διακοπής. Όταν η υπορουτίνα ολοκληρωθεί, η εκτέλεση συνεχίζεται από το σημείο όπου συνέβη η διακοπή. Διακοπές γίνεται να προκύψουν σαν αποτέλεσμα εσωτερικής δραστηριότητας του AT85C51SND3 (π.χ. υπερχείλιση χρονιστών) ή στην έναρξη ηλεκτρικών σημάτων εκτός του μικροελεγκτή (π.χ. πληκτρολόγιο). Σε κάθε περίπτωση, η λειτουργία της διακοπής προγραμματίζεται από τον σχεδιαστή ο οποίος προσδιορίζει την προτεραιότητα της υπηρεσίας διακοπής σχετικά με την κανονική εκτέλεση του κώδικα και άλλων ρουτινών εξυπυρέτησης διακοπών.

Μια τυπική αλληλουχία περίπτωσης διακοπών είναι ως εξής:

- Μια εσωτερική ή εξωτερική συσκευή στέλνει ένα σήμα αίτησης διακοπής. Ο AT85C51SND3 βάζει αυτό το αίτημα σε ένα flag buffer.

- Η προτεραιότητα του flag συγκρίνεται με την προτεραιότητα άλλων διακοπών από τον διαχειριστή διακοπών. Υψηλή προτεραιότητα κάνει τον διαχειριστή να θέσει ένα flag διακοπής.
- Αυτό δίνει σήμα στην μονάδα εκτέλεσης εντολών να εκτελέσει μια άλλαγή πλαισίου. Αυτή η αλλαγή πλαισίου σπάει την τρέχουσα ροή ακολουθίας εντολών. Η μονάδα εκτέλεσης οληκληρώνει την τρέχουσα εντολή
- Η ρουτίνα εξυπηρέτησης λογισμικου εκτελεί τις εργασίες που της έχουν ανατεθεί και σαν τελευταία δραστηριότητα εκτελεί μια εντολή RETI (Return from Interrupt). Αυτή η εντολή σηματοδοτεί την ολοκλήρωση της διακοπής, επανεκκινεί την προτεραιότητα διακοπών και επαναφορτώνει τον μετρητή προγράμματος. Έπειτα η λειτουργία του προγράμματος συνεχίζεται από το σημείο διακοπής.

Έξι καταχωρητές διακοπών χρησιμοποιούνται για έλεγχο του συστήματος διακοπών:

- Δύο 8-μπιτοι καταχωρητές για να ενεργοποιήσουν ξεχωριστά τις πηγές διακοπών. Οι IEN0 και IEN1.
- Τέσσερις 8-μπιτοι καταχωρητές για να ρυθμίσουν το επίπεδο προτεραιότητας των διαφορετικών πηγών: Οι IPH0, IPL0, IPH1 και IPL1.

Κάθε πηγή διακοπής του AT85C51SND3 μπορεί ξεχωριστά να τεθεί σε ένα από 4 διαφορετικά επίπεδα προτεραιότητας. Αυτό πραγματοποιείται με ένα μπιτ στους καταχωρητές IPH0 και IPH1 (Interrupt Priority High) και ένα μπιτ στους καταχωρητές IPL0 και IPL1

(Interrupt Priority Low). Αυτό παρέχει σε κάθε πηγή διακοπής 4 δυνατά επίπεδα προτεραιότητας.

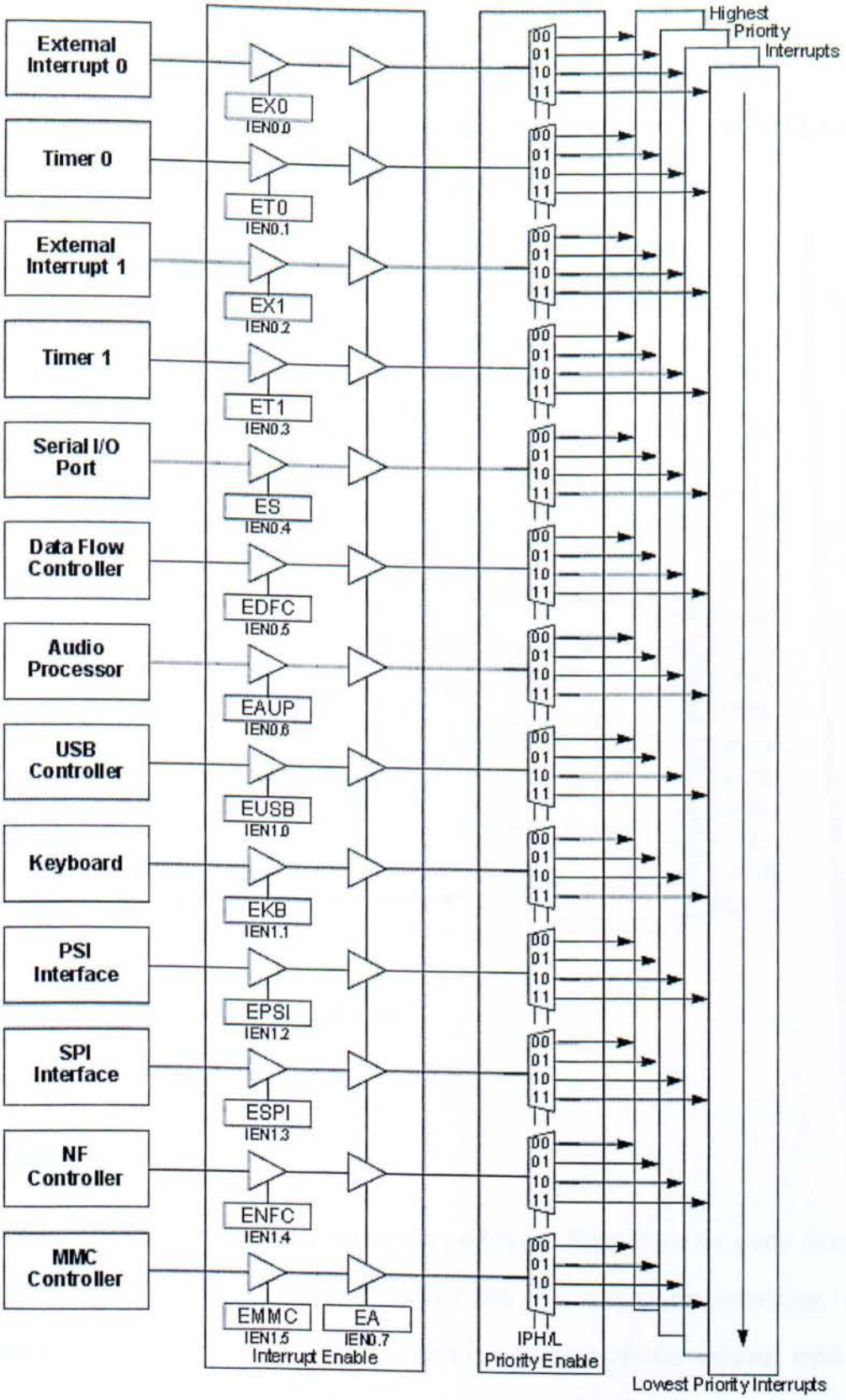
IPHxx	IPLxx	Priority Level
0	0	0 Lowest
0	1	1
1	0	2
1	1	3 Highest

Επίπεδο προτεραιότητας

Μια διακοπή χαμηλής προτεραιότητας διακόπτεται πάντα από μια διακοπή υψηλής προτεραιότητας αλλά όχι από μια διακοπή ίσης ή χαμηλότερης προτεραιότητας. Οι υψηλού επιπέδου διακοπές εξυπηρετούνται πριν από τις χαμηλού επιπέδου διακοπές. Η απόκριση σε ταυτόχρονη σημτοδότηση διακοπών ίσης προτεραιότητας καθορίζεται από μια εσωτερική σταθμοσκόπηση. Έτσι, έχουμε μέσα σε κάθε επίπεδο προτεραιότητας, ένα δεύτερο τύπο προτεραιοτήτων.

Interrupt Name	Priority Number	Interrupt Address Vectors	Interrupt Request Flag Cleared by Hardware (H) or by Software (S)
INT0	0 (Highest Priority)	C:0003h	H if edge, S if level
Timer 0	1	C:000Bh	H
INT1	2	C:0013h	H if edge, S if level
Timer 1	3	C:001Bh	H
Serial I/O Port	4	C:0023h	S
Data Flow Controller	5	C:002Bh	S
Audio Processor	6	C:0033h	S
USB Controller	7	C:003Bh	S
Keyboard	8	C:0043h	S
Parallel Slave Interface	9	C:004Bh	S
Serial Peripheral Interface	10	C:0053h	S
Nand Flash Controller	11	C:005Bh	S
MMC Controller	12	C:0063h	S
Reserved	13	C:006Bh	-
Reserved	14 (Lowest Priority)	C:0073h	-

Προτεραιότητες εντός ίδιου επιπέδου

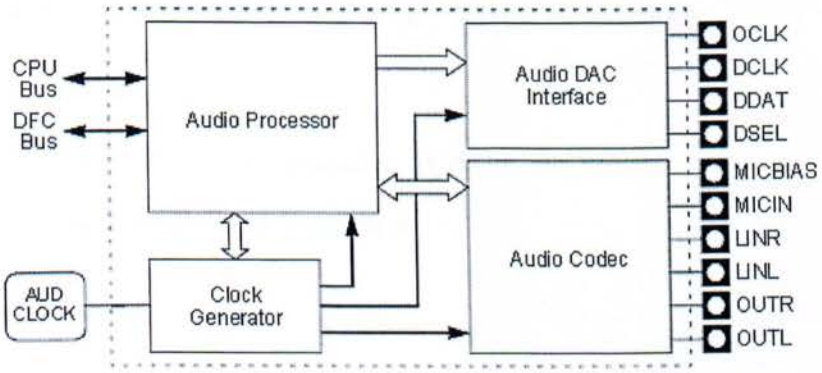


Το σύστημα ελέγχου διακοπών

2.4 Ελεγκτής ήχου

Ο ελεγκτής ήχου που είναι ενσωματωμένος στον AT85C51SND3 βασίζεται σε τέσσερα μπλοκ λειτουργίας:

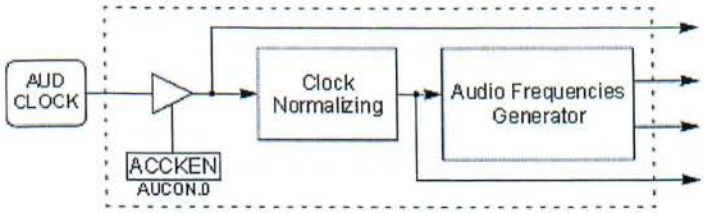
- Στον χρονιστή
- Στον επεξεργαστή ήχου
- Στο Codec ήχου
- Στην διεπαφή ήχου DAC



Το μπλοκ διάγραμμα του ελεγκτή ήχου

Χρονιστής

Ο χρονιστής παράγει τους παλμούς ρολογιού βασιζόμενος στον ελεγκτή ρολογιού. Περιλαμβάνει μια γεννήτρια ηχητικών συχνοτήτων, που παράγει τις συχνότητες δειγματοληψίας που τροφοδοτούνται από ένα κανονικοποιημένο ρολόι. Αυτή η γεννήτρια βασίζεται σε ένα PLL και είναι ολοκληρωτικά ελεγχόμενη από τον επεξεργαστή ήχου.

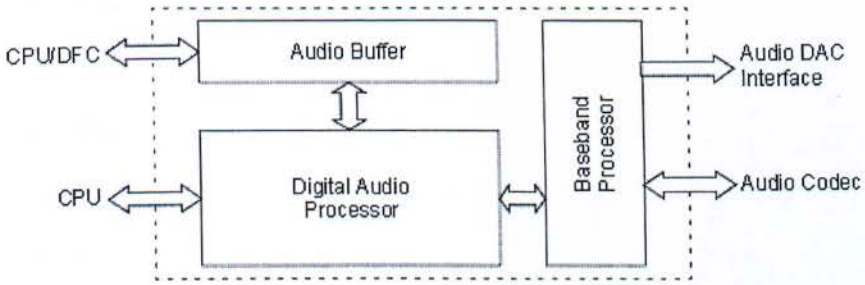


Ο χρονιστής του ελεγκτή ήχου

Επεξεργαστής ήχου

Ο επεξεργαστής ήχου βασίζεται σε 3 μπλοκ λειτουργίας:

- Το Audio Buffer
- Τον ψηφιακό επεξεργαστή ήχου
- Τον επεξεργαστή Baseband



Το μπλοκ διάγραμμα του επεξεργαστή ήχου

ΚΕΦΑΛΑΙΟ 3: Ο μικροελεγκτής TSC80251

3.1 Γενικά

Η TSC80251 της εταιρείας Temic είναι μια οικογένεια μικροελεγκτών των 8 ή 16 bit και αποτελούν μια αναβάθμιση των ευρέως χρησιμοποιούμενων μικροελεγκτών 80C51. Κάνει επέκταση των χαρακτηριστικών και των επιδόσεων τους, διατηρώντας παράλληλα την συμβατότητα με δυαδικό κώδικα, έτσι ώστε το αντίκτυπο στο ήδη υπάρχον υλικό και λογισμικό να είναι το ελάχιστο δυνατό. Μερικές εφαρμογές στις οποίες χρησιμοποιούνται οι συγκεκριμένοι μικροελεγκτές είναι:

Βιομηχανία αυτοκινήτου:

- Αερόσακος
- Κιβώτιο ταχυτήτων
- ABS
- Κλιματισμός
- Ράδιο
- GPS

Επικοινωνίες:

- Ασύρματα τηλέφωνα
- Κινητά τηλέφωνα
- Modem υψηλής ταχύτητας
- Τηλέφωνα ISDN

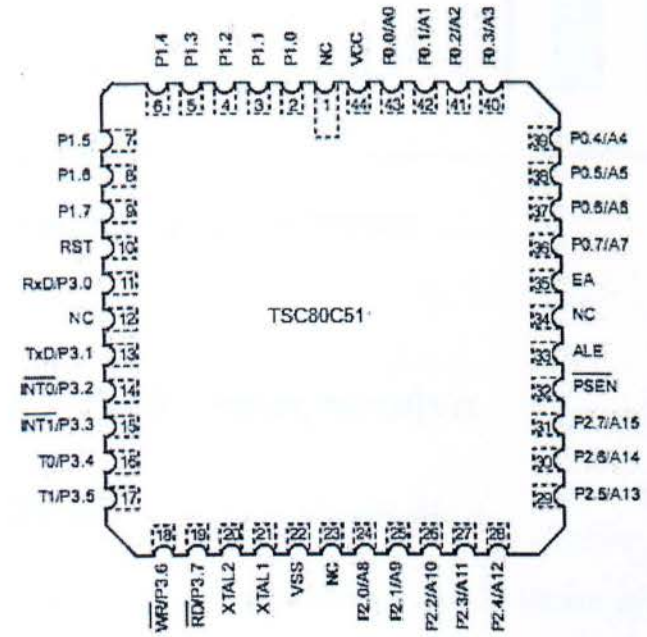
- Τερματικά δικτύων

Υπολογιστικά Συστήματα:

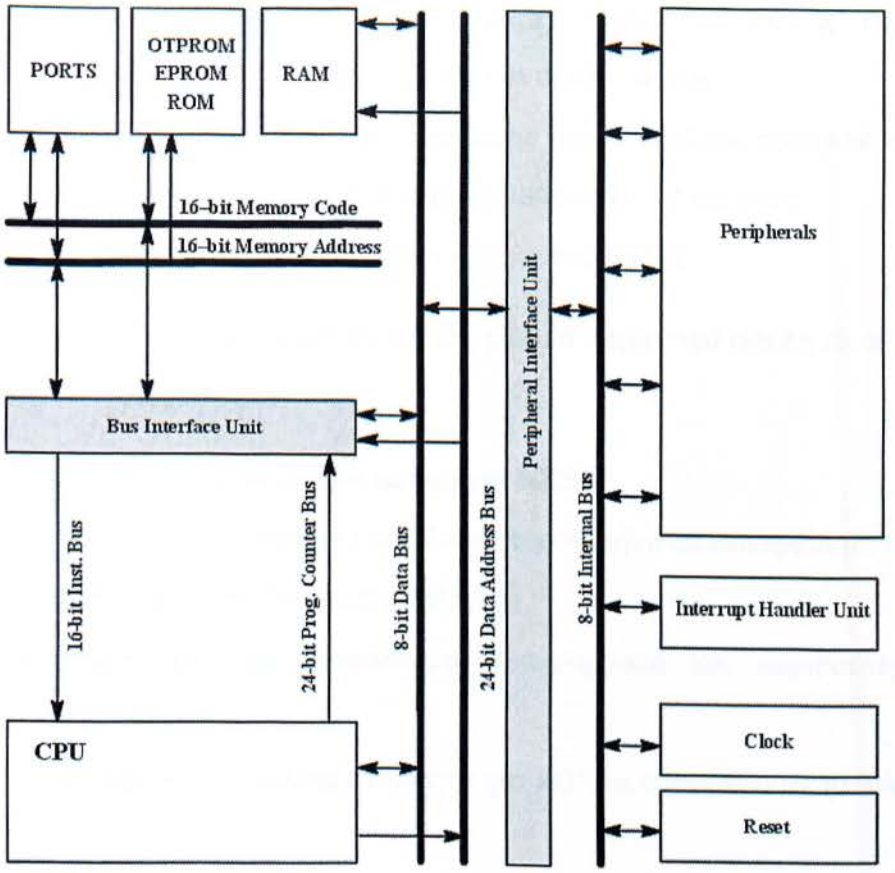
- Οθόνες
- DVD-Rom
- Σαρωτές barcode
- Τηλεφωνία μέσω υπολογιστών

Εργοστάσια:

- Έλεγχος επιτήρησης διεργασιών
- Συστήματα κλιματισμού
- Αυτοματισμοί



Οι ακροδέκτες του TSC80C51



Το μπλοκ διάγραμμα του TSC80251

3.2 Σχεδιασμός πυρήνα

Ο πυρήνας του C251 περιλαμβάνει:

- Γραμμικές διευθύνσεις των 24 bit και μνήμη μέχρι 16 Mbytes
- Εμπλουστευμένο σετ εντολών, περιλαμβανομένων αριθμητικών και λογικών εντολών
- Σωρό χωρητικότητας 64 Kbytes

- CPU βασισμένη σε αρχείο καταχωρητών, οι οποίοι είναι προσπελάσιμοι ως bytes, words και double words.
- Ελάχιστο χρόνο εντολής - εκτέλεσης τους 2 κύκλους ρολογιού (σε αντίθεση με τον 80C51 που χρησιμοποιούσε 12 κύκλους)
- Συμβατότητα με τον δυαδικό κώδικα του 80C51

Εξαιτίας αυτών των χαρακτηριστικών, μερικά σημαντικά οφέλη είναι τα εξής:

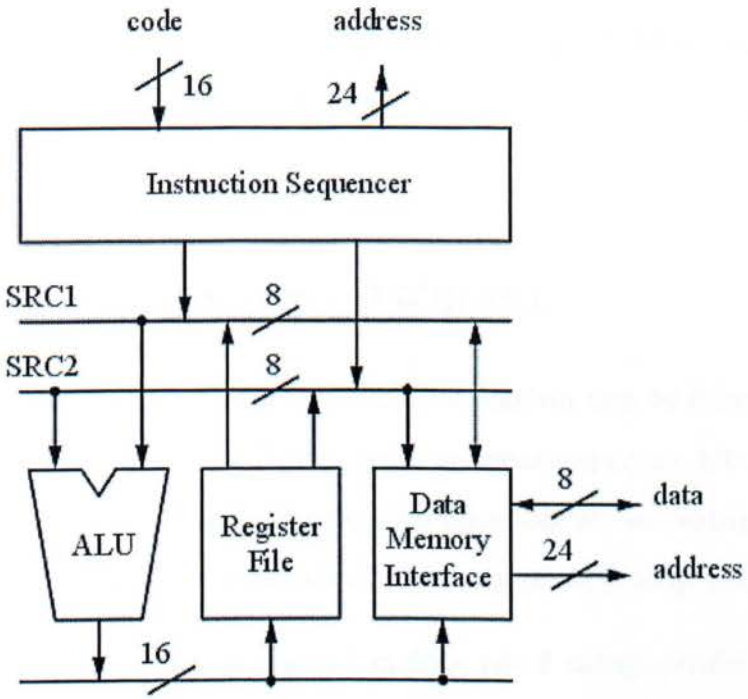
- Διατήρηση κώδικα γραμμένου για 80C51
- Αύξηση στην ταχύτητα εκτέλεσης του πυρήνα σε σύγκριση με τον 80C51 για τον ίδιο δείκτη ρολογιού
- Υποστήριξη για μεγαλύτερα προγράμματα και περισσότερα δεδομένα
- Αυξημένη αποτελεσματικότητα για κώδικα γραμμένο σε γλώσσα C

Ο πυρήνας του μικροελεγκτή

Ο πυρήνας του TSC80251 περιλαμβάνει την CPU, την μονάδα επανεκκίνησης και ρολογιού, τον χειρισμό των διακοπών, την διεπαφή δίαυλων και την διεπαφή περιφερειακών.

CPU

Ο TSC80251 διαβάζει τις εντολές από την μνήμη κώδικα που βρίσκεται στο chip με ρυθμό 2 bytes τη φορά ή από εξωτερική μνήμη με ρυθμό 1 byte τη φορά. Οι εντολές αποστέλλονται από τον 16-μπιτο δίαυλο εντολών στην κεντρική μονάδα επεξεργασίας.



Το μπλοκ διάγραμμα της CPU του TSC80251

Το αρχείο καταχωρητών του TSC80251 διαθέτει 40 καταχωρητές οι οποίοι μπορούν να προσπελαστούν ως bytes (8-bit δεδομένα), words (16-bit δεδομένα) και double words (32-bit δεδομένα). Όπως στην δομή του 80C51, οι καταχωρητές 0-7 αποτελούνται από τέσσερις ομάδες των 8 καταχωρητών η καθεμιά, όπου για γρήγορη εναλλαγή πλαισίου, η ενεργός ομάδα επιλέγεται από το PSW (Program Status Word).

Η CPU του TSC80251 είναι μια μηχανή συνεχούς διοχέτευσης. Δηλ. αποτελείται από συνδεδεμένες μεταξύ τους βαθμίδες, κάθε μία από τις οποίες κάνει μια συγκεκριμένη εργασία σε κάποιο δεδομένο και το

περνάει στην επόμενη για να κάνει και εκείνη τη δική της. Δηλ. είναι σαν μια γραμμή παραγωγής αυτοκινήτων. Γεμάτη διοχέτευση έχουμε όταν όλες οι βαθμίδες λειτουργούν ταυτόχρονα/παράλληλα και κάθε μία επεξεργάζεται το δικό της δεδομένο.

3.3 Οι καταχωρητές του TSC80251

Το αρχείο καταχωρητών του TSC80251 αποτελείται από 40 θέσεις byte: 0-31 και 56-63. Αυτές οι θέσεις είναι προσπελάσιμες ως bits, bytes, words, και dwords. Αρκετές θέσεις είναι αφιερωμένες για καταχωρητές ειδικών λειτουργιών. Οι άλλες είναι για καταχωρητές γενικής χρήσης.

Οι θέσεις 0-7 αποτελούνται από 4 ομάδες των 8 καταχωρητών η κάθε μία. Οι 4 ομάδες χρησιμοποιούνται σαν τα πρώτα 32 byte της μνήμης RAM του chip και είναι πάντα προσπελάσιμες στις θέσεις μνήμης διευθύνσεων 00:0000H - 00:001FH. Σε μια συγκεκριμένη χρονική στιγμή, μόνο μία από τις 4 ομάδες είναι προσπελάσιμη μέσω του αρχείου καταχωρητών. Η ενεργός ομάδα επιλέγεται από τα bits RS0 και RS1 του PSW. Αυτή η επιλογή ομάδας μπορεί να χρησιμοποιηθεί για γρήγορες εναλλαγές πλαισίου.

Bank	Address Range	PSW Selection Bits	
		RS1	RS0
Bank 0	00h-07h	0	0
Bank 1	08h-0Fh	0	1
Bank 2	10h-17h	1	0
Bank 3	18h-1Fh	1	1

Οι ομάδες καταχωρητών του TSC80251

Οι θέσεις καταχωρητών αρχείου 8-31 και 56-63 είναι πάντα προσπελάσιμοι. Αυτές οι θέσεις εφαρμόζονται σαν καταχωρητές στην CPU. Οι θέσεις καταχωρητών αρχείου 32-55 κρατούνται για συγκεκριμένο σκοπό και δεν μπορούν να προσπελαστούν.

Core SFRs

Mnemonic	Name	Address
A *	Accumulator	S:E0h
B *	B register	S:F0h
PW	Program Status Word	S:D0h
PSW1	Program Status Word 1	S:D1h
SP	Stack Pointer - LSB of SPX	S:81h
SPH *	Stack Pointer high - MSB of SPX	S:BEh
DPTR *	Data Pointer (2 bytes)	–
DPL *	Low Byte of DPTR	S:82h
DPH *	high Byte of DPTR	S:83h
DPXL *	Data Pointer, Extended Low	S:84h
IE0	Interrupt Enable Control 0	S:A8h
IE1	Interrupt Enable Control 1	S:B1h
IPL0	Interrupt Priority Control Low 0	S:B8h
IPL1	Interrupt Priority Control Low 1	S:B3h
IPH0	Interrupt Priority Control High 0	S:B7h
IPH1	Interrupt Priority Control High 1	S:B2h

Οι καταχωρητές ειδικών λειτουργιών του TSC80251

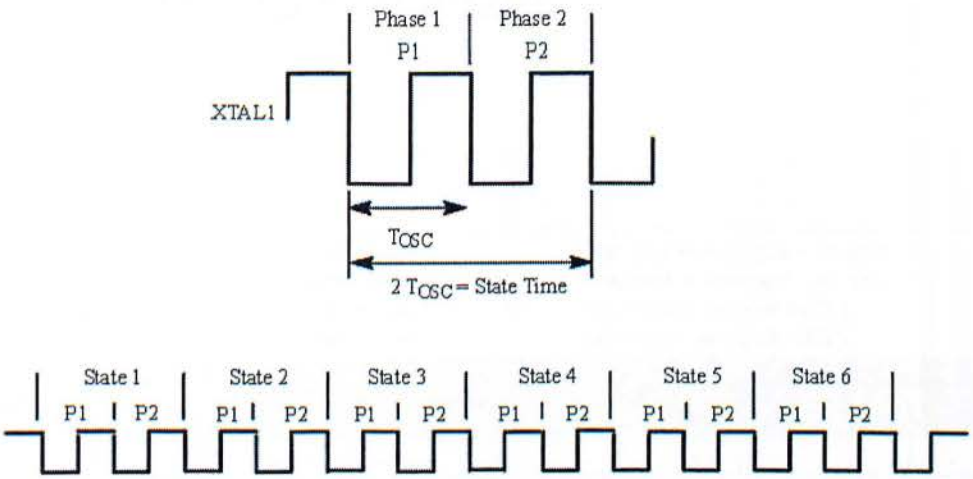
3.4 Η μονάδα χρονισμού του TSC80251

Η πηγή χρονισμού του TSC80251 μπορεί να είναι ένας εξωτερικός ταλαντωτής ή ένας εσωτερικός ταλαντωτής με εξωτερικό αντηχείο. Η βασική μονάδα χρόνου στον TSC80251 είναι ο κύκλος μηχανής που αντιστοιχεί σε 2 περιόδους του ταλαντωτή. Ο κύκλος μηχανής χωρίζεται

στην φάση P1 και στην φάση P2. Τα περιφερειακά του TSC80251 λειτουργούν σε περιφερειακό κύκλο, ο οποίος αντιστοιχεί σε 6 κύκλους μηχανής (αυτός ο περιφερειακός κύκλος δεν αποτελεί χαρακτηριστικό της αρχιτεκτονικής του C251). Το διάστημα ενός ρολογιού στον περιφερειακό κύκλο υποδηλώνεται από την κατάσταση και τη φάση του.

3.5 Το σύστημα διακοπών του TSC80251

Το σύστημα διακοπών του TSC80251 μπορεί να λάβει αιτήματα διακοπών από πολλές πηγές: εσωτερικές, εξωτερικές και την εντολή TRAP. Όταν το σύστημα διακοπών παρέχει ένα αίτημα διακοπής, η CPU διακόπτει την κανονική ροή των εντολών και διακλαδώνεται σε μια ρουτίνα που εξυπηρετεί την πηγή που έστειλε το αίτημα διακοπής. Ο χρήστης δύναται να ενεργοποιήσει ή να απενεργοποιήσει τις διακοπές ξεχωριστά (πλην των εντολών TRAP και NMI που δεν γίνεται να απενεργοποιηθούν) και να επιλέξει ανάμεσα απο 4 επίπεδα προτεραιότητας για κάθε διακοπή.



Οι φάσεις ρολογιού του TSC80251

3.6 Οι εντολές του TSC80251

Add	ADD <dest>, <src>	dest opnd ← dest opnd + src opnd
Subtract	SUB <dest>, <src>	dest opnd ← dest opnd – src opnd
Add with Carry	ADDC <dest>, <src>	(A) ← (A) + src opnd + (CY)
Subtract with Borrow	SUBB <dest>, <src>	(A) ← (A) – src opnd – (CY)

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
ADD	A, Rn	Register to ACC	1	1	2	2
	A, dir8	Direct address to ACC	2	1(2)	2	1(2)
	A, @Ri	Indirect address to ACC	1	2	2	3
	A, #data	Immediate data to ACC	2	1	2	1
ADD / SUB	Rmd, Rms	Byte register to/from byte register	3	2	2	1
	WRjd, WRjs	Word register to/from word register	3	3	2	2
	DRkd, DRks	Dword register to/from dword register	3	5	2	4
	Rm, #data	Immediate 8-bit data to/from byte register	4	3	3	2
	WRj, #data16	Immediate 16-bit data to/from word register	5	4	4	3
	DRk, #0data16	16-bit unsigned immediate data to/from dword register	5	6	4	5
	Rm, dir8	Direct address (on–chip RAM or SFR) to/from byte register	4	3(2)	3	2(2)
	WRj, dir8	Direct address (on–chip RAM or SFR) to/from word register	4	4	3	3
	Rm, dir16	Direct address (64K) to/from byte register	5	3(3)	4	2(3)
	WRj, dir16	Direct address (64K) to/from word register	5	4(4)	4	3(4)
	Rm, @WRj	Indirect address (64K) to/from byte register	4	3(3)	3	2(3)
	Rm, @DRk	Indirect address (16M) to/from byte register	4	4(3)	3	3(3)
ADDC / SUBB	A, Rn	Register to/from ACC with carry	1	1	2	2
	A, dir8	Direct address (on–chip RAM or SFR) to/from ACC with carry	2	1(2)	2	1(2)
	A, @Ri	Indirect address to/from ACC with carry	1	2	2	3
	A, #data	Immediate data to/from ACC with carry	2	1	2	1

Summary of Increment and Decrement Instructions

Increment	INC <dest>	dest opnd ← dest opnd + 1
Increment	INC <dest>, <src>	dest opnd ← dest opnd + src opnd
Decrement	DEC <dest>	dest opnd ← dest opnd – 1
Decrement	DEC <dest>, <src>	dest opnd ← dest opnd – src opnd

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
INC DEC	A	ACC by 1	1	1	1	1
	Rn	Register by 1	1	1	2	2
	dir8	Direct address (on-chip RAM or SFR) by 1	2	2(2)	2	2(2)
	@Ri	Indirect address by 1	1	3	2	4
INC DEC	Rm, #short	Byte register by 1, 2, or 4	3	2	2	1
	WRj, #short	Word register by 1, 2, or 4	3	2	2	1
INC	DRk, #short	Double word register by 1, 2, or 4	3	4	2	3
DEC	DRk, #short	Double word register by 1, 2, or 4	3	5	2	4
INC	DPTR	Data pointer by 1	1	1	1	1

Notes:
1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *If this instruction addresses an I/O Port (Px, x= 0-3), add 2 to the number of states. Add 3 if it addresses a Peripheral SFR.*

Summary of Compare Instructions

Compare	CMP <dest>, <src>	dest opnd – src opnd

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
CMP	Rmd, Rms	Register with register	3	2	2	1
	WRjd, WRjs	Word register with word register	3	3	2	2
	DRkd, DRks	Dword register with dword register	3	5	2	4
	Rm, #data	Register with immediate data	4	3	3	2
	WRj, #data16	Word register with immediate 16-bit data	5	4	4	3
	DRk, #0data16	Dword register with zero-extended 16-bit immediate data	5	6	4	5
	DRk, #1data16	Dword register with one-extended 16-bit immediate data	5	6	4	5
	Rm, dir8	Direct address (on-chip RAM or SFR) with byte register	4	3(1)	3	2(1)
	WRj, dir8	Direct address (on-chip RAM or SFR) with word register	4	4	3	3
	Rm, dir16	Direct address (64K) with byte register	5	3(2)	4	2(2)
	WRj, dir16	Direct address (64K) with word register	5	4(3)	4	3(3)
	Rm, @WRj	Indirect address (64K) with byte register	4	3(2)	3	2(2)
	Rm, @DRk	Indirect address (16M) with byte register	4	4(2)	3	3(2)

Summary of Logical Instructions (1/2)

Logical AND[1]	ANL <dest>, <src>	dest opnd ← dest opnd ∧ src opnd
Logical OR[1]	ORL <dest>, <src>	dest opnd ← dest opnd ∨ src opnd
Logical Exclusive OR[1]	XRL <dest>, <src>	dest opnd ← dest opnd ∀ src opnd
Clear[1]	CLR A	(A) ← 0
Complement[1]	CPL A	(A) ← Ø(A)
Rotate Left	RL A	$(A)_{n+1} \leftarrow (A)_n$, n=0..6 $(A)_0 \leftarrow (A)_7$
Rotate Left Carry	RLC A	$(A)_{n+1} \leftarrow (A)_n$, n=0..6 $(CY) \leftarrow (A)_7$ $(A)_0 \leftarrow (CY)$
Rotate Right	RR A	$(A)_{n-1} \leftarrow (A)_n$, n=7..1 $(A)_7 \leftarrow (A)_0$
Rotate Right Carry	RRC A	$(A)_{n-1} \leftarrow (A)_n$, n=7..1 $(CY) \leftarrow (A)_0$ $(A)_7 \leftarrow (CY)$

Mnemonic	<dest>, <src>[2]	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
ANL ORL XRL	A, Rn	register to ACC	1	1	2	2
	A, dir8	Direct address (on-chip RAM or SFR) to ACC	2	1[3]	2	1[3]
	A, @Ri	Indirect address to ACC	1	2	2	3
	A, #data	Immediate data to ACC	2	1	2	1
	dir8, A	ACC to direct address	2	2[4]	2	2[4]
	dir8, #data	Immediate 8-bit data to direct address	3	3[4]	3	3[4]
	Rmd, Rms	Byte register to byte register	3	2	2	1
	WRjd, WRjs	Word register to word register	3	3	2	2
	Rm, #data	Immediate 8-bit data to byte register	4	3	3	2
	WRj, #data16	Immediate 16-bit data to word register	5	4	4	3
	Rm, dir8	Direct address to byte register	4	3[3]	3	2[3]
	WRj, dir8	Direct address to word register	4	4	3	3
	Rm, dir16	Direct address (64K) to byte register	5	3[5]	4	2[5]
	WRj, dir16	Direct address (64K) to word register	5	4[6]	4	3[6]
	Rm, @WRj	Indirect address (64K) to byte register	4	3[5]	3	2[5]
	Rm, @DRk	Indirect address (16M) to byte register	4	4[5]	3	3[5]
CLR	A	Clear ACC	1	1	1	1
CPL	A	Complement ACC	1	1	1	1
RL	A	Rotate ACC left	1	1	1	1
RLC	A	Rotate ACC left through CY	1	1	1	1
RR	A	Rotate ACC right	1	1	1	1
RRC	A	Rotate ACC right through CY	1	1	1	1

Summary of Logical Instructions (2/2)

Shift Left Logical	SLL <dest>	$<dest>_0 \leftarrow 0$ $<dest>_{n+1} \leftarrow <dest>_n$, n= 0..msb–1 (CY) ← $<dest>_{msb}$
Shift Right Arithmetic	SRA <dest>	$<dest>_{msb} \leftarrow <dest>_{msb}$ $<dest>_{n-1} \leftarrow <dest>_n$, n= msb..1 (CY) ← $<dest>_0$
Shift Right Logical	SRL <dest>	$<dest>_{msb} \leftarrow 0$ $<dest>_{n-1} \leftarrow <dest>_n$, n= msb..1 (CY) ← $<dest>_0$
Swap	SWAP A	$A_{3:0} \leftrightarrow A_{7:4}$

Mnemonic	<dest>, <src>[1]	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
SLL	Rm	Shift byte register left through the MSB	3	2	2	1
	WRj	Shift word register left through the MSB	3	2	2	1
SRA	Rm	Shift byte register right	3	2	2	1
	WRj	Shift word register right	3	2	2	1
SRL	Rm	Shift byte register left	3	2	2	1
	WRj	Shift word register left	3	2	2	1
SWAP	A	Swap nibbles within ACC	1	2	1	2

Note:
1. A shaded cell denotes an instruction in the C51 Architecture.

Summary of Multiply, Divide and Decimal-adjust Instructions

Multiply	MUL AB MUL <dest>, <src>	(B:A) ← (A)×(B) extended dest opnd ← dest opnd × src opnd
Divide	DIV AB	(A) ← Quotient ((A)/(B)) (B) ← Remainder ((A)/(B))
Divide src opnd) opnd/src opnd)	DIV <dest>, <src>	ext. dest opnd high ← Quotient (dest opnd/ ext. dest opnd low ← Remainder (dest
Decimal-adjust ACC for Addition (BCD)	DA A	IF [[$(A)_{3:0} > 9$] ∨ [(AC)= 1]] THEN $(A)_{3:0} \leftarrow (A)_{3:0} + 6$!affects CY; IF [[$(A)_{7:4} > 9$] ∨ [(CY)= 1]] THEN $(A)_{7:4} \leftarrow (A)_{7:4} + 6$

Mnemonic	<dest>, <src>[1]	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
MUL	AB	Multiply A and B	1	5	1	5
	Rmd, Rms	Multiply byte register and byte register	3	6	2	5
	WRjd, WRjs	Multiply word register and word register	3	12	2	11
DIV	AB	Divide A and B	1	10	1	10
	Rmd, Rms	Divide byte register and byte register	3	11	2	10
	WRjd, WRjs	Divide word register and word register	3	21	2	20
DA	A	Decimal adjust ACC	1	1	1	1

Summary of Move Instructions (1/3)

Move to High word	MOVH <dest>, <src>	dest opnd$_{31:16}$ ← src opnd
Move with Sign extension	MOVS <dest>, <src>	dest opnd ← src opnd with sign extend
Move with Zero extension	MOVZ <dest>, <src>	dest opnd ← src opnd with zero extend
Move Code	MOVC A, <src>	(A) ← src opnd
Move eXtended	MOVX <dest>, <src>	dest opnd ← src opnd

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
MOVH	DRk, #data16	16-bit immediate data into upper word of dword register	5	3	4	2
MOVS	WRj, Rm	Byte register to word register with sign extension	3	2	2	1
MOVZ	WRj, Rm	Byte register to word register with zeros extension	3	2	2	1
MOVC	A, @A +DPTR	Code byte relative to DPTR to ACC	1	6(3)	1	6(3)
	A, @A +PC	Code byte relative to PC to ACC	1	6(3)	1	6(3)
MOVX	A, @Ri	Extended memory (8-bit address) to ACC(2)	1	4	1	5
	A, @DPTR	Extended memory (16-bit address) to ACC(2)	1	3(4)	1	3(4)
	@Ri, A	ACC to extended memory (8-bit address)(2)	1	4	1	4
	@DPTR, A	ACC to extended memory (16-bit address)(2)	1	4(3)	1	4(3)

Notes:

1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *Extended memory addressed is in the region specified by DPXL (reset value = 01h).*
3. *If this instruction addresses external memory location, add N+1 to the number of states (N: number of wait states).*
4. *If this instruction addresses external memory location, add N+2 to the number of states (N: number of wait states).*

Summary of Move Instructions (2/3)

Move(1)		MOV <dest>, <src>	dest opnd ← src opnd			
Mnemonic	<dest>, <src>(2)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
MOV	A, Rn	Register to ACC	1	1	2	2
	A, dir8	Direct address (on-chip RAM or SFR) to ACC	2	1(3)	2	1(3)
	A, @Ri	Indirect address to ACC	1	2	2	3
	A, #data	Immediate data to ACC	2	1	2	1
	Rn, A	ACC to register	1	1	2	2
	Rn, dir8	Direct address (on-chip RAM or SFR) to register	2	1(3)	3	2(3)
	Rn, #data	Immediate data to register	2	1	3	2
	dir8, A	ACC to direct address	2	2(3)	2	2(3)
	dir8, Rn	Register to direct address	2	2(3)	3	3(3)
	dir8, dir8	Direct address to direct address	3	3(4)	3	3(4)
	dir8, @Ri	Indirect address to direct address	2	3(3)	3	4(3)
	dir8, #data	Immediate data to direct address	3	3(3)	3	3(3)
	@Ri, A	ACC to indirect address	1	3	2	4
	@Ri, dir8	Direct address to indirect address	2	3(3)	3	4(3)
	@Ri, #data	Immediate data to indirect address	2	3	3	4
	DPTR, #data16	Load Data Pointer with a 16-bit constant	3	2	3	2

Summary of Move Instructions (3/3)

Move[1]	MOV <dest>, <src>	dest opnd ← src opnd				
Mnemonic	**<dest>, <src>[2]**	**Comments**	**Binary Mode**		**Source Mode**	
			Bytes	**States**	**Bytes**	**States**
	Rmd, Rms	Byte register to byte register	3	2	2	1
	WRjd, WRjs	Word register to word register	3	2	2	1
	DRkd, DRks	Dword register to dword register	3	3	2	2
	Rm, #data	Immediate 8-bit data to byte register	4	3	3	2
	WRj, #data16	Immediate 16-bit data to word register	5	3	4	2
	DRk, #0data16	zero-ext 16bit immediate data to dword register	5	5	4	4
	DRk, #1data16	one-ext 16bit immediate data to dword register	5	5	4	4
	Rm, dir8	Direct address to byte register	4	3[3]	3	2[3]
	WRj, dir8	Direct address to word register	4	4	3	3

Mnemonic	<dest>, <src>[2]	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
MOV	DRk, dir8	Direct address to dword register	4	6	3	5
	Rm, dir16	Direct address (64K) to byte register	5	3[4]	4	2[4]
	WRj, dir16	Direct address (64K) to word register	5	4[5]	4	3[5]
	DRk, dir16	Direct address (64K) to dword register	5	6[6]	4	5[6]
	Rm, @WRj	Indirect address (64K) to byte register	4	3[4]	3	2[4]
	Rm, @DRk	Indirect address (16M) to byte register	4	4[4]	3	3[4]
	WRjd, @WRjs	Indirect address (64K) to word register	4	4[5]	3	3[5]
	WRj, @DRk	Indirect address (16M) to word register	4	5[5]	3	4[5]
	dir8, Rm	Byte register to direct address	4	4[3]	3	3[3]
	dir8, WRj	Word register to direct address	4	5	3	4
	dir8, DRk	Dword register to direct address	4	7	3	6
	dir16, Rm	Byte register to direct address (64K)	5	4[4]	4	3[4]
	dir16, WRj	Word register to direct address (64K)	5	5[5]	4	4[5]
	dir16, DRk	Dword register to direct address (64K)	5	7[6]	4	6[6]
	@WRj, Rm	Byte register to indirect address (64K)	4	4[4]	3	3[4]
	@DRk, Rm	Byte register to indirect address (16M)	4	5[4]	3	4[4]
	@WRjd, WRjs	Word register to indirect address (64K)	4	5[5]	3	4[5]
	@DRk, WRj	Word register to indirect address (16M)	4	6[5]	3	5[5]
	Rm, @WRj +dis16	Indirect with 16-bit dis (64K) to byte register	5	6[4]	4	5[4]
	WRj, @WRj +dis16	Indirect with 16-bit dis (64K) to word register	5	7[5]	4	6[5]
	Rm, @DRk +dis24	Indirect with 16-bit dis (16M) to byte register	5	7[4]	4	6[4]
	WRj, @WRj +dis24	Indirect with 16-bit dis (16M) to word register	5	8[5]	4	7[5]
	@WRj +dis16, Rm	Byte register to indirect with 16-bit dis (64K)	5	6[4]	4	5[4]
	@WRj +dis16, WRj	Word register to indirect with 16-bit dis (64K)	5	7[5]	4	6[5]
	@DRk +dis24, Rm	Byte register to indirect with 16-bit dis (16M)	5	7[4]	4	6[4]
	@DRk +dis24, WRj	Word register to indirect with 16-bit dis (16M)	5	8[5]	4	7[5]

Notes:

1. *Instructions that move bits are in Table 5. 16. .*
2. *Move instructions unique to the C251 Architecture.*
3. *If this instruction addresses an IO Port (Px, x= 0-3), add 1 to the number of states. Add 2 if it addresses a Peripheral SFR.*
4. *If this instruction addresses external memory location, add N+2 to the number of states (N: number of wait states).*
5. *If this instruction addresses external memory location, add 2(N+1) to the number of states (N: number of wait states).*

Summary of Bit Instructions

Clear Bit	CLR <dest>	dest opnd ← 0
Set Bit	SETB <dest>	dest opnd ← 1
Complement Bit	CPL <dest>	dest opnd ← Ø bit
AND Carry with Bit	ANL CY, <src>	(CY) ← (CY) ∧ src opnd
AND Carry with Complement of Bit (CY) ← (CY) ∧ Ø src opnd		ANL CY, /<src>
OR Carry with Bit	ORL CY, <src>	(CY) ← (CY) ∨ src opnd
OR Carry with Complement of Bit (CY) ← (CY) ∨ Ø src opnd		ORL CY, /<src>
Move Bit to Carry	MOV CY, <src>	(CY) ← src opnd
Move Bit from Carry	MOV <dest>, CY	dest opnd ← (CY)

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
CLR	CY	Clear carry	1	1	1	1
	bit51	Clear direct bit	2	2(3)	2	2(3)
	bit	Clear direct bit	4	4(3)	3	3(3)
SETB	CY	Set carry	1	1	1	1
	bit51	Set direct bit	2	2(3)	2	2(3)
	bit	Set direct bit	4	4(3)	3	3(3)
CPL	CY	Complement carry	1	1	1	1
	bit51	Complement direct bit	2	2(3)	2	2(3)
	bit	Complement direct bit	4	4(3)	3	3(3)
ANL	CY, bit51	And direct bit to carry	2	1(2)	2	1(2)
	CY, bit	And direct bit to carry	4	3(2)	3	2(2)
	CY, /bit51	And complemented direct bit to carry	2	1(2)	2	1(2)
	CY, /bit	And complemented direct bit to carry	4	3(2)	3	2(2)
ORL	CY, bit51	Or direct bit to carry	2	1(2)	2	1(2)
	CY, bit	Or direct bit to carry	4	3(2)	3	2(2)
	CY, /bit51	Or complemented direct bit to carry	2	1(2)	2	1(2)
	CY, /bit	Or complemented direct bit to carry	4	3(2)	3	2(2)
MOV	CY, bit51	Move direct bit to carry	2	1(2)	2	1(2)
	CY, bit	Move direct bit to carry	4	3(2)	3	2(2)
	bit51, CY	Move carry to direct bit	2	2(3)	2	2(3)
	bit, CY	Move carry to direct bit	4	4(3)	3	3(3)

Notes:

1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *If this instruction addresses an I/O Port (Px, x= 0-3), add 1 to the number of states. Add 2 if it addresses a Peripheral SFR.*
3. *If this instruction addresses an I/O Port (Px, x= 0-3), add 2 to the number of states. Add 3 if it addresses a Peripheral SFR.*

Summary of Exchange, Push and Pop Instructions

Exchange bytes	XCH A, <src>	(A) ↔ src opnd
Exchange Digit	XCHD A, <src>	$(A)_{3:0}$ ↔ src $opnd_{3:0}$
Push	PUSH <src>	(SP) ← (SP) +1; ((SP)) ← src opnd; (SP) ← (SP) +size (src opnd) – 1
Pop	POP <dest>	(SP) ← (SP) – size (dest opnd) + 1; dest opnd ← ((SP)); (SP) ← (SP) –1

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
XCH	A, Rn	ACC and register	1	3	2	4
	A, dir8	ACC and direct address (on-chip RAM or SFR)	2	3(3)	2	3(3)
	A, @Ri	ACC and indirect address	1	4	2	5
XCHD	A, @Ri	ACC low nibble and indirect address (256 bytes)	1	4	2	5
PUSH	dir8	Push direct address onto stack	2	2(2)	2	2(2)
	#data	Push immediate data onto stack	4	4	3	3
	#data16	Push 16-bit immediate data onto stack	5	5	4	5
	Rm	Push byte register onto stack	3	4	2	3
	WRj	Push word register onto stack	3	5	2	4
	DRk	Push double word register onto stack	3	9	2	8
POP	dir8	Pop direct address (on-chip RAM or SFR) from stack	2	3(2)	2	3(2)
	Rm	Pop byte register from stack	3	3	2	2
	WRj	Pop word register from stack	3	5	2	4
	DRk	Pop double word register from stack	3	9	2	8

Notes:
1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *If this instruction addresses an I/O Port (Px, x= 0-3), add 1 to the number of states. Add 2 if it addresses a Peripheral SFR.*
3. *If this instruction addresses an I/O Port (Px, x= 0-3), add 2 to the number of states. Add 3 if it addresses a Peripheral SFR.*

Summary of Conditional Jump Instructions (1/2)

Jump conditional on status	Jcc rel	(PC) ← (PC) +size (instr); IF [cc] THEN (PC) ← (PC) + rel

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode(2)		Source Mode(2)	
			Bytes	States	Bytes	States
JC	rel	Jump if carry	2	1/4(3)	2	1/4(3)
JNC	rel	Jump if not carry	2	1/4(3)	2	1/4(3)
JE	rel	Jump if equal	3	2/5(3)	2	1/4(3)
JNE	rel	Jump if not equal	3	2/5(3)	2	1/4(3)
JG	rel	Jump if greater than	3	2/5(3)	2	1/4(3)
JLE	rel	Jump if less than, or equal	3	2/5(3)	2	1/4(3)
JSL	rel	Jump if less than (signed)	3	2/5(3)	2	1/4(3)
JSLE	rel	Jump if less than, or equal (signed)	3	2/5(3)	2	1/4(3)
JSG	rel	Jump if greater than (signed)	3	2/5(3)	2	1/4(3)
JSGE	rel	Jump if greater than or equal (signed)	3	2/5(3)	2	1/4(3)

Notes:
1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *States are given as jump not-taken/taken.*
3. *In internal execution only, add 1 to the number of states of the 'jump taken' if the destination address is internal and odd.*

Summary of unconditional Jump Instructions

Absolute jump	AJMP <src>	(PC) ← (PC) +2; $(PC)_{10:0}$ ← src opnd
Extended jump opnd	EJMP <src>	(PC) ← (PC) + size (instr); $(PC)_{23:0}$ ← src
Long jump opnd	LJMP <src>	(PC) ← (PC) + size (instr); $(PC)_{15:0}$ ← src
Short jump	SJMP rel	(PC) ← (PC) +2; (PC) ← (PC) +rel
Jump indirect (DPTR)	JMP @A +DPTR	$(PC)_{23:16}$ ← FFh; $(PC)_{15:0}$ ← (A) +
No operation	NOP	(PC) ← (PC) +1

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
AJMP	addr11	Absolute jump	2	3(2)(3)	2	3(2)(3)
EJMP	addr24	Extended jump	5	6(2)(4)	4	5(2)(4)
	@DRk	Extended jump (indirect)	3	7(2)(4)	2	6(2)(4)
LJMP	@WRj	Long jump (indirect)	3	6(2)(4)	2	5(2)(4)
	addr16	Long jump (direct address)	3	5(2)(4)	3	5(2)(4)
SJMP	rel	Short jump (relative address)	2	4(2)(4)	2	4(2)(4)
JMP	@A +DPTR	Jump indirect relative to the DPTR	1	5(2)(4)	1	5(2)(4)
NOP		No operation (Jump never)	1	1	1	1

Notes:
1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *In internal execution only, add 1 to the number of states if the destination address is internal and odd.*
3. *Add 2 to the number of states if the destination address is external.*
4. *Add 3 to the number of states if the destination address is external.*

Summary of Call and Return Instructions

Absolute call	ACALL <src>	(PC) ← (PC) +2; push $(PC)_{15:0}$; $(PC)_{10:0}$ ← src opnd
Extended call	ECALL <src>	(PC) ← (PC) + size (instr); push $(PC)_{23:0}$; $(PC)_{23:0}$ ← src opnd
Long call	LCALL <src>	(PC) ← (PC) + size (instr); push $(PC)_{15:0}$; $(PC)_{15:0}$ ← src opnd
Return from subroutine	RET	pop $(PC)_{15:0}$
Extended return from subroutine	ERET	pop $(PC)_{23:0}$
Return from interrupt	RETI	IF [INTR= 0] THEN pop $(PC)_{15:0}$ IF [INTR= 1] THEN pop $(PC)_{23:0}$; pop (PSW1)
Trap interrupt	TRAP	(PC) ← (PC) + size (instr); IF [INTR= 0] THEN push $(PC)_{15:0}$ IF [INTR= 1] THEN push (PSW1); push $(PC)_{23:0}$

Mnemonic	<dest>, <src>(1)	Comments	Binary Mode		Source Mode	
			Bytes	States	Bytes	States
ACALL	addr11	Absolute subroutine call	2	9(2)(3)	2	9(2)(3)
ECALL	@DRk	Extended subroutine call (indirect)	3	14(2)(3)	2	13(2)(3)
	addr24	Extended subroutine call	5	14(2)(3)	4	13(2)(3)
LCALL	@WRj	Long subroutine call (indirect)	3	10(2)(3)	2	9(2)(3)
	addr16	Long subroutine call	3	9(2)(3)	3	9(2)(3)
RET		Return from subroutine	1	7(2)	1	7(2)
ERET		Extended subroutine return	3	9(2)	2	8(2)
RETI		Return from interrupt	1	7(2)(4)	1	7(2)(4)
TRAP		Jump to the trap interrupt vector	2	12(4)	1	11(4)

Notes:
1. *A shaded cell denotes an instruction in the C51 Architecture.*
2. *In internal execution only, add 1 to the number of states if the destination/return address is internal and odd.*
3. *Add 2 to the number of states if the destination address is external.*
4. *Add 5 to the number of states if INTR= 1.*

New Instructions for the C251 Architecture

Bin	A5x8*	A5x9*	A5xA*	A5xB*
Src	x8*	x9*	xA*	xB*
0	JSLE rel	MOV Rm, @WRj +dis16	MOVZ WRj, Rm	INC R, #short (1) MOV reg, ind
1	JSG rel	MOV @WRj +dis16, Rm	MOVS WRj, Rm	DEC R, #short (1) MOV ind, reg
2	JLE rel	MOV Rm, @DRk +dis24		
3	JG rel	MOV @DRk +dis24, Rm		
4	JSL rel	MOV WRj, @WRj +dis16		
5	JSGE rel	MOV @WRj +dis16, WRj		
6	JE rel	MOV WRj, @DRk +dis24		
7	JNE rel	MOV @DRk +dis16, WRj	MOVZ op1, reg (2)	
8		JMP @WRj EJMP @DRk	EJMP addr24	
9		LCALL @WRj ECALL @DRk	ECALL addr24	
A		*Escape* Bit Instructions (3)	ERET	
B		TRAP		
C			PUSH op1 (4) MOV DRk, PC	
D			POP op1 (4)	
0				
1				
2	ADD Rmd, Rms	ADD WRjd, WRjs	ADD reg, op2 (2)	ADD DRkd, DRks
3			SLL reg	
4	ORL Rmd, Rms	ORL WRjd, WRjs	ORL reg, op2 (2)	
5	ANL Rmd, Rms	ANL WRjd, WRjs	ANL reg, op2 (2)	
6	XRL Rmd, Rms	XRL WRjd, WRjs	XRL reg, op2 (2)	
7	MOV Rmd, Rms	MOV WRjd, WRjs	MOV reg, op2 (2)	MOV DRkd, DRks
8	DIV Rmd, Rms	DIV WRjd, WRjs		
9	SUB Rmd, Rms	SUB WRjd, WRjs	SUB reg, op2 (2)	SUB DRkd, DRkd
A	MUL Rmd, Rms	MUL WRjd, WRjs		
B	CMP Rmd, Rms	CMP WRjd, WRjs	CMP reg, op2 (2)	CMP DRkd, DRks

Notes :

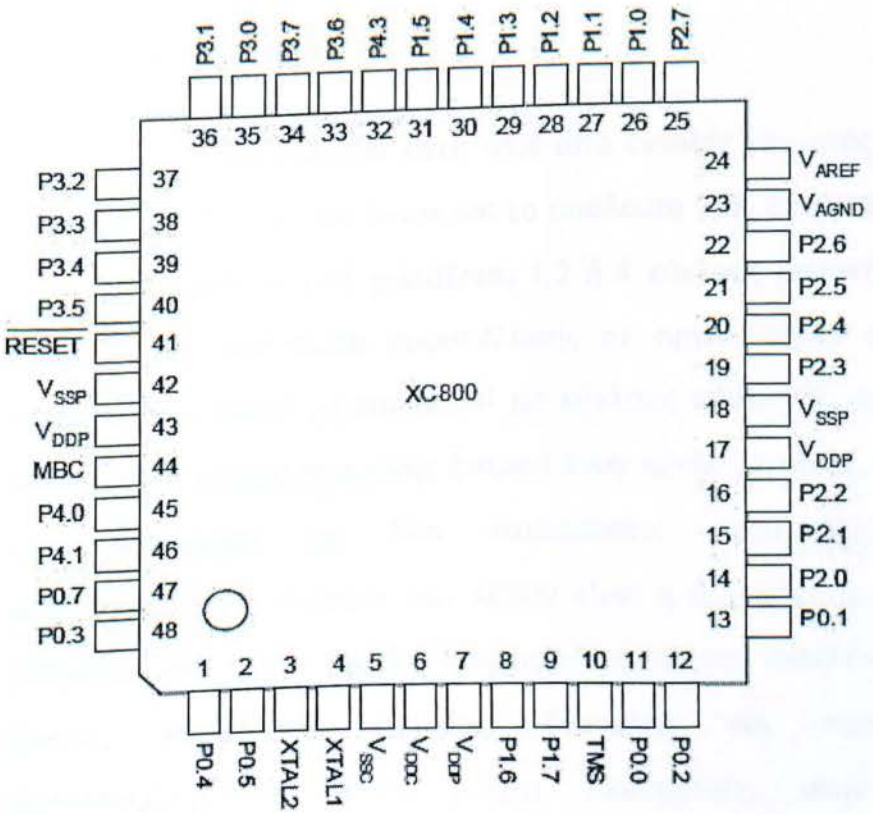
1. *R = Rm/WRj/DRk*
2. *op1, op2 are defined in Table 5.24.*
3. *See Table 5.26. and Table 5.27.*
4. *See Table 5.28.*

* x takes the values found in Bin and Src column.

ΚΕΦΑΛΑΙΟ 4: Ο μικροελεγκτής XC800

4.1 Γενικά

Η XC800 της εταιρείας Infineon είναι μια οικογένεια 8-μπιτων μικροελεγκτών, που πρωτοεισήχθη το 2005, με πυρήνα έναν βέλτιστοποιημένο 8051 διπλού κύκλου.



Οι ακροδέκτες του XC800

Χωρίζονται σε δυο κατηγορίες: την οικογένεια A και την οικογένεια I. Η οικογένεια I περιλαμβάνει μνήμες Flash από 2 KB έως 64 KB, καθώς και

ολοκληρωμένα κυκλώματα 16 έως 64 ακροδεκτών. Οι συγκεκριμένοι μικροελεγκτές εμφανίζονται σε ηλεκτρικούς κινητήρες, ηλεκτρικά ποδήλατα ή σε κλιματιστικά με πλήκτρα ελέγχου αφής. Μικροελεγκτές της οικογένειας A εμφανίζονται στην βιομηχανία αυτοκινήτων/μοτοσυκλετών και τηρούν όλες τις προδιαγραφές του AEC Q100 (Automotive Electronics Council) για τα ηλεκτρονικά αυτοκινούμενων οχημάτων.

4.2 Πυρήνας

Το set εντολών αποτελείται κατα 45% από εντολές του ενός byte, το 41% από εντολές των δυο bytes και το υπόλοιπο 14% από εντολές των τριών bytes. Κάθε εντολή χρειάζεται 1,2 ή 4 κύκλους μηχανής για να εκτελεστεί. Σε περίπτωση προσπέλασης σε αργή μνήμη, ο χρόνος προσπέλασης μπορεί να επεκταθεί με κύκλους αναμονής, κατά τους οποίους κάθε κύκλος αναμονής διαρκεί έναν κύκλο μηχανής, ο οποίος είναι ισοδύναμος με δυο καταστάσεις αναμονής. Βασικό χαρακτηριστικό του πυρήνα του XC800 είναι η δυνατότητα επίλυσης σφαλμάτων όπως στο βασικό ξεκίνημα/σταμάτημα, εκτέλεση μονού βήματος, υποστήριξη σημείου διακοπής και προσπέλαση ανάγνωσης/εγγραφής στην μνήμη δεδομένων, στην μνήμη προγράμματος και στους καταχωρητές ειδικού τύπου. Ένας 16-μπιτος συνεπεξεργαστής παρέχει επιπλέον υπολογιστική απόδοση για την επεξεργασία εφαρμογών πολλαπλασιασμού/διαίρεσης και για την εκτέλση ειδικών αλγόριθμων σε τριγωνομετρικές εφαρμογές.

4.3 Οργάνωση μνήμης

Οι 8-μπιτοι μικροελεγκτές έχουν μια ενσωματωμένη, προγραμματιζόμενη από τον χρήστη, μνήμη Flash η οποία επιτρέπει την γρήγορη και αξιόπιστη αποθήκευση κώδικα και δεδομένων. Λειτουργεί με μια παροχή των 2.5V από τον ενσωματωμένο ρυθμιστή τάσης (EVR) και δεν απαιτεί επιπλέον προγραμματισμό ή διαγραφή τάσης. Η τμηματοποίηση της μνήμης Flash επιτρέπει σε κάθε τομέα να διαγράφεται ανεξάρτητα. Η διόρθωση σφαλμάτων Flash (ECC) μπορεί να ανιχνεύσει σφάλματα διπλών bit και να διορθώσει σφάλματα μονού bit καθώς και να προστατέψει την εκτέλεση μη έγκυρου κώδικα.

4.4 ADC

Το στοιχείο μετατροπής αναλογικού σήματος σε ψηφιακό (ADC) χρησιμοποιεί την μέθοδο διαδοχικής προσέγγισης για να μετατρέψει τις τιμές των αναλογικών σημάτων εισόδου σε διακριτές ψηφιακές τιμές με ανάλυση 10 bit. Ένας πυρήνας ADC λειτουργεί σε έναν αριθμό από κανάλια εισόδου που επιλέγονται από τον χρήστη.

4.5 Επικοινωνία

Ο XC800 περιλαμβάνει ένα σετ από διεπαφές για σειριακή επικοινωνία με συσκευές όπως UART, SPI, I2C και CAN. Ο CAN (Controller Area Network) είναι ένας σειριακός δίαυλος σχεδιασμένος για board-to-board επικοινωνία σε θορυβώδη περιβάλλοντα όπως αυτοκινητιστικά ή βιομηχανικά συστήματα ελέγχου. Το MultiCAN που αναπτύχθηκε από την Infineon, βασίζεται σε βελτιωμένες προηγούμενες CAN εφαρμογές, προσθέτοντας χαρακτηριστικά όπως επιπλέον κόμβοι CAN και λίστες διαχείρησης για αντικείμενα μηνυμάτων.

4.6 Εργαλεία Ανάπτυξης

Υπάρχουν διάφορα kits αξιολόγησης διαθέσιμα για όλες τις συσκευές XC800. Αυτά τα kits είναι προσανατολισμένα για συγκεκριμένες εφαρμογές π.χ. για σχέδια ελέγχου κινητήρα. Περιέχουν παραδείγματα κώδικα για διάφορα συστήματα ελέγχου και συνοδεύονται από το κατάλληλο υλικό.

Επίσης είναι διαθέσιμα αρκετά δωρεάν εργαλεία ανάπτηυξης όπως το DAVE (Digital Application Virtual Engineer) το οποίο χρησιμεύει στη διαμόρφωση οδηγών χαμηλού επιπέδου και αυτόματα παράγει πηγαίο κώδικα.

4.7 Οι εντολές του XC800

Instruction Table

Mnemonic	Description	Hex Code	Bytes	Cycles
INC DPTR	Increment data pointer	A3	1	2
MUL AB	Multiply A by B	A4	1	4
DIV AB	Divide A by B	84	1	4
DA A	Decimal Adjust A	D4	1	1
	LOGICAL			
ANL A,Rn	AND register to A	58-5F	1	1
ANL A,direct	AND direct byte to A	55	2	1
ANL A,@Ri	AND indirect memory to A	56-57	1	1
ANL A,#data	AND immediate to A	54	2	1
ANL direct,A	AND A to direct byte	52	2	1
ANL direct,#data	AND immediate to direct byte	53	3	2
ORL A,Rn	OR register to A	48-4F	1	1
ORL A,direct	OR direct byte to A	45	2	1
ORL A,@Ri	OR indirect memory to A	46-47	1	1
ORL A,#data	OR immediate to A	44	2	1
ORL direct,A	OR A to direct byte	42	2	1
ORL direct,#data	OR immediate to direct byte	43	3	2
XRL A,Rn	Exclusive-OR register to A	68-6F	1	1
XRL A,direct	Exclusive-OR direct byte to A	65	2	1
XRL A,@Ri	Exclusive-OR indirect memory to A	66-67	1	1
XRL A,#data	Exclusive-OR immediate to A	64	2	1
XRL direct,A	Exclusive-OR A to direct byte	62	2	1
XRL direct,#data	Exclusive-OR immediate to direct byte	63	3	2
CLR A	Clear A	E4	1	1
CPL A	Complement A	F4	1	1
SWAP A	Swap Nibbles of A	C4	1	1
RL A	Rotate A left	23	1	1
RLC A	Rotate A left through carry	33	1	1
RR A	Rotate A right	03	1	1

Mnemonic	Description	Hex Code	Bytes	Cycles
RRC A	Rotate A right through carry	13	1	1
	DATA TRANSFER			
MOV A,Rn	Move register to A	E8-EF	1	1
MOV A,direct	Move direct byte to A	E5	2	1
MOV A,@Ri	Move indirect memory to A	E6-E7	1	1
MOV A,#data	Move immediate to A	74	2	1
MOV Rn,A	Move A to register	F8-FF	1	1
MOV Rn,direct	Move direct byte to register	A8-AF	2	2
MOV Rn,#data	Move immediate to register	78-7F	2	1
MOV direct,A	Move A to direct byte	F5	2	1
MOV direct,Rn	Move register to direct byte	88-8F	2	2
MOV direct,direct	Move direct byte to direct byte	85	3	2
MOV direct,@Ri	Move indirect memory to direct byte	86-87	2	2
MOV direct,#data	Move immediate to direct byte	75	3	2
MOV @Ri,A	Move A to indirect memory	F6-F7	1	1
MOV @Ri,direct	Move direct byte to indirect memory	A6-A7	2	2
MOV @Ri,#data	Move immediate to indirect memory	76-77	2	1
MOV DPTR,#data16	Move immediate to data pointer	90	3	2
MOVC A,@A+DPTR	Move code byte relative DPTR to A	93	1	2
MOVC A,@A+PC	Move code byte relative PC to A	83	1	2
MOVX A,@Ri	Move external data (A8) to A	E2-E3	1	2
MOVX A,@DPTR	Move external data (A16) to A	E0	1	2
MOVX @Ri,A	Move A to external data (A8)	F2-F3	1	2
MOVX @DPTR,A	Move A to external data (A16)	F0	1	2
PUSH direct	Push direct byte onto stack	C0	2	2
POP direct	Pop direct byte from stack	D0	2	2
XCH A,Rn	Exchange A and register	C8-CF	1	1

Mnemonic	Description	Hex Code	Bytes	Cycles
XCH A,direct	Exchange A and direct byte	C5	2	1
XCH A,@Ri	Exchange A and indirect memory	C6-C7	1	1
XCHD A,@Ri	Exchange A and indirect memory nibble	D6-D7	1	1
	BOOLEAN			
CLR C	Clear carry	C3	1	1
CLR bit	Clear direct bit	C2	2	1
SETB C	Set carry	D3	1	1
SETB bit	Set direct bit	D2	2	1
CPL C	Complement carry	B3	1	1
CPL bit	Complement direct bit	B2	2	1
ANL C,bit	AND direct bit to carry	82	2	2
ANL C,/bit	AND direct bit inverse to carry	B0	2	2
ORL C,bit	OR direct bit to carry	72	2	2
ORL C,/bit	OR direct bit inverse to carry	A0	2	2
MOV C,bit	Move direct bit to carry	A2	2	1
MOV bit,C	Move carry to direct bit	92	2	2
	BRANCHING			
ACALL addr11	Absolute call within current 2 K	11->F1	2	2
LCALL addr16	Long call to addr16	12	3	2
RET	Return from subroutine	22	1	2
RETI	Return from interrupt routine	32	1	2
AJMP addr11	Absolute jump within current 2 K	01->E1	2	2
LJMP addr16	Long jump unconditional	02	3	2
SJMP rel	Short jump to relative address	80	2	2
JC rel	Jump relative on carry = 1	40	2	2
JNC rel	Jump relative on carry = 0	50	2	2
JB bit,rel	Jump relative on direct bit = 1	20	3	2
JNB bit,rel	Jump relative on direct bit = 0	30	3	2
JBC bit,rel	Jump relative and clear on direct bit = 1	10	3	2

Mnemonic	Description	Hex Code	Bytes	Cycles
JMP @A+DPTR	Jump indirect relative DPTR	73	1	2
JZ rel	Jump relative on accumulator = 0	60	2	2
JNZ rel	Jump relative on accumulator = 1	70	2	2
CJNE A,direct,rel	Compare direct memory to accumulator, jump relative if not equal	B5	3	2
CJNE A,#data,rel	Compare immediate to accumulator, jump relative if not equal	B4	3	2
CJNE Rn,#data,rel	Compare immediate to register, jump relative if not equal	B8-BF	3	2
CJNE @Ri,#data,rel	Compare immediate to indirect memory, jump relative if not equal	B6-B7	3	2
DJNZ Rn,rel	Decrement register and jump relative if not zero	D8-DF	2	2
DJNZ direct,rel	Decrement direct memory and jump relative if not zero	D5	3	2
MISCELLANEOUS				
NOP	No operation	00	1	1
ADDITIONAL INSTRUCTIONS (selected through EO[7:4])				
MOVC @(DPTR++),A	XC800-specific instruction for software download into program memory: Copy from accumulator, then increment DPTR	A5	1	2
TRAP	XC800-specific software break command	A5	1	1

4.8 Οι καταχωρητές ειδικής λειτουργίας του XC800

Οι καταχωρητές ειδικής λειτουργίας του XC800 βρίσκονται μεταξύ των διευθύνσεων 80H και FFH. Είναι προσπελάσιμοι μέσω απ'ευθείας διευθυνσιοδότησης. Οι SFR που βρίσκονται σε θέσεις με διεύθυνση bit 0-2 ίσο με 0 (80H, 88H, 90H,...,F8H) έχουν δική τους διεύθυνση bit. Κάθε ένας από αυτούς τους έχει διεύθυνση bit σύμφωνα με τη διεύθυνση byte του SFR καθώς και της θέσης του εντός του byte. Για παράδειγμα το bit 7 του SFR της διεύθυνσης 80H έχει διεύθυνση bit 87H.

ΕΠΙΛΟΓΟΣ

Όπως είδαμε τα περιθωρία εξέλιξης και προσαρμογής είναι ατελέιωτα και καθώς οι ανάγκες προχωράνε δεν μπορεί παρά να υπάρξει και η ανάλογη εξέλιξη και στο χώρο των μικροελεγκτών με συνεχώς νέες καινοτομίες και τροποποιήσεις.

Όπως όλα δείχνουν οι εταιρείες σταδιακά θα αφήσουν πίσω τους μικροελεγκτές των 8 ή 16 bit και θα βασίζονται πλέον αποκλειστικά στα 32 bit.

Πηγές:

Χρήστος Καραΐσκος, *Ο μικροελεγκτής 8051*, Σύγχρονη Εκδοτική, 2010

http://www.ceid.upatras.gr/courses/micro/8051SHMEIOSEIS.PDF

http://en.wikipedia.org/wiki/Intel_MCS-51

http://www.keil.com/dd/docs/datashts/atmel/at85c51snd3.pdf

http://www.alldatasheet.com/datasheet-pdf/pdf/242466/TEMIC/TSC80251.html

http://en.wikipedia.org/wiki/XC800

http://www.keil.com/dd/docs/datashts/infineon/xc800_is.pdf