

Multidisciplinary Engineering Design

from theory to practice

2013 Edition

M. Reza Emami

**University of Toronto
Institute for Aerospace Studies
Division of Engineering Science**

**McGraw-Hill Ryerson Limited
New York, NY**

Multidisciplinary Engineering Design: from theory to practice, 2013 edition, by M.R. Emami.

Product Development Manager, Learning Solutions: Jason Giles
Learning Solutions Custom Print Specialist: Lauren Ligterink
Learning Solutions Manager: Jeff Snook

Cover Design: M. R. Enami

CHAPTERS

Chapter 1 : Course Organization

Chapter 2 : Equipment for Engineering Design

Chapter 3 : Machine Shop

Chapter 4 : Design

Chapter 5 : Circuits

Chapter 6 : Electromechanical Systems

Chapter 7 : PIC Microcontrollers

Chapter 8 : Sensors

Chapter 9 : Suppliers

Chapter 1

Course Organization

1.1	Outline	1
1.2	Schedule	2
1.3	Procedure	4
1.4	Marking Scheme	7
1.4.1	Project Cost	7
1.4.2	Project Proposal (10%)	7
1.4.3	Individual Performance Evaluations (22.5% total)	8
1.4.4	Design Notebook (15% total)	9
1.4.5	Team Evaluations (32.5% total)	10
1.4.6	Public Project Demonstrations	11
1.4.7	Project Review	12
1.4.8	Final Report (20%)	12
1.4.9	Allocation of Team Marks	13
1.5	General Notes	13
1.5.1	Safety First!	13
1.5.2	Attendance	15
1.5.3	Petitions and Extensions	15
1.5.4	Laboratory Hours	16
1.5.5	E-Mail and Lecture Announcements	16
1.5.6	Machine Shop	16
1.5.7	Laboratory Cleanliness	16
1.5.8	Food	16
1.5.9	Equipment Loans	17
1.5.10	Storage of Project Items and Equipment Kits	17
1.5.11	Rewards of Excellence	17
1.5.12	Design Store	17
1.6	Request for Proposals	19

1.1 Outline

INSTRUCTORS:	M. Reza Emami, Ph.D., P.Eng.	Room: SF4003 Phone: 416-946-3357 Email: emami@utias.utoronto.ca		
WEBSITE:	aer201.aerospace.utoronto.ca and www.pml4all.org			
TEACHING ASSISTANTS:				
Vincent Caux-Brisebois vincent.caux.brisebois@mail.utoronto.ca	Mark Feero m.feero@mail.utoronto.ca	Justin Girard justin.girard@mail.utoronto.ca		
Gilbert Jiang chuan.jiang@mail.utoronto.ca	Jason Kereluk jason.kereluk@mail.utoronto.ca	Mina Mitry mina.mitry@mail.utoronto.ca		
Jeffrey Osborne josborne@utias.utoronto.ca	Victor Ragusila victor.ragusila@mail.utoronto.ca	Rene Rail-Ip rene.rail.ip@mail.utoronto.ca		
Maria Sakovsky Maria.sakovsky@mail.utoronto.ca	Valentin Stolbunov valentin.stolbunov@mail.utoronto.ca	Peter Szabo peter.szabo@mail.utoronto.ca		
LECTURE:	Tuesday Wednesday Friday Monday Monday	Week 1 Week 1 Week 1 and 3 Week 2, 3, 4 Week 5, 6, 8-13	15-18 16-18 14-18 9-12 11-12	HS610 (Health Sciences Bldg.) MP103 (McLennan Physical Labs) ES1050 (Earth Sciences Centre) HS610 WB116 (Wallberg Bldg.)
SEMINAR:	Friday Friday Friday	Week 1 Week 3 Week 4	16-17 16-17 14-15	ES1050 ES1050 ES1050
WORKSHOP:	Mon, Tue, Wed Friday	Week 2 Week 4	14-18 15-17	Machine Shop Safety, SF4003 ES1050
PROJECT:	BARL: STKR: PIPE:	The Barrel Inspector Machine The Sticker Inspector Machine The Pipe Inspector Machine		
LABORATORY:	BARL: STKR: PIPE:	Monday Tuesday Wednesday	Lab. SF4003, 4102, 4103 Lab. SF4003, 4102, 4103 Lab. SF4003, 4102, 4103	13-18 13-18 13-18
Note 1: Safety rules and guidelines are the first and foremost consideration in all course activities.				
Note 2: Attendance in the designated laboratory sessions is mandatory.				
Note 3: An additional weekly Friday session (14-18) is optional for all sections in Week 4-6, 8-11, 13.				
Note 4: Late submission is not accepted for any milestone in this course.				
MARKING:	Project Proposal (Week 4, group, marked by Instructor and TA's) Week-5 Performance Evaluation (individual, marked by Instructor and TA's) Intermediate Design Notebook (individual, marked by TA's) Week-8 Performance Evaluation (individual, marked by Instructor and TA's) Week-10 Progress Evaluation (group, marked by Instructor and TA's) Week-12 Project Evaluation (group, marked by Instructor and TA's) Project Demonstration (Week 14, group, marked by Instructor and TA's) or Project Review (week 13, individual, marked by Instructor and TA's) Final Design Notebook (individual, marked by TA's) Final Report (group, marked by Instructor)	10% 10% 7.5% 12.5% 7.5% 12.5% 12.5% 7.5% 20%		

1.2 Schedule

	MONDAY L: 11-12, HS610 - WB116 P: 13-18, SF4003, 4102-3	TUESDAY P: 13-18, SF4003, 4102-3	WEDNESDAY P: 13-18, SF4003, 4102-3	FRIDAY (L/S/P: 14-18)
Week 1 (Jan. 07-11)		L: Course Introduction and Safety Guidelines (15-16, HS610) L: Course Learning Strategy (16-17, HS610) L: Teamwork (17-18, HS610) <u>Obtain the Course Notes from Bookstore</u>	L: Information & Communication (16-17, MP103) L: Project Definition (17-18, MP103) <u>Submission of Teams on Design Portal starts at 18:00</u>	L: PIC Microcontroller (14-16, ES1050) Seminar: AER201 Past Experiences (16-17, ES1050) L: PIC Microcontroller (17-18, ES1050) <u>Submission of Teams on Design Portal ends at 21:00</u>
Week 2 (Jan. 14-18)	L: PIC Microcontroller (9-11, HS610) L: Circuits (11-12, HS610) P: Machine Shop Safety for Electromechanical Members (14-18, SF4003) P: Meet with TAs, Obtain Design Kit, Obtain Project Kit, Open Design Store Account, Discuss the Project	P: Machine Shop Safety for Electromechanical Members (14-18, SF4003) P: Meet with TAs, Obtain Design Kit, Obtain Project Kit, Open Design Store Account, Discuss the Project	P: Machine Shop Safety for Electromechanical Members (14-18, SF4003) P: Meet with TAs, Obtain Design Kit, Obtain Project Kit, Open Design Store Account, Discuss the Project	
Week 3 (Jan. 21-25)	L: Circuits (9-11, HS610) L: Circuits (11-12, HS610) P: Subsystem Fabrication	P: Subsystem Fabrication	P: Subsystem Fabrication	L: Sensors (14-15, ES1050) Seminar: Real World Design (16-17, ES1050) L: Electromechanical Systems (16-18, ES1050)
Week 4 (Jan. 28 - Feb. 01)	L: Electromechanical Systems (9-11, HS610) L: Electromechanical Systems (11-12, HS610) P: Subsystem Fabrication/Integration	P: Subsystem Fabrication/Integration	P: Subsystem Fabrication/Integration <u>Project Proposal, All Teams, Submission to Instructor in SF4003 by 14:00</u>	Seminar: PML Development Boards (14-15, ES1050) P: Circuit Soldering Training for All Sections (15-17, ES1050) Optional Session for All Sections
Week 5 (Feb. 04-08)	L: Project Planning (WB116) <u>Individual Evaluation Subsystem Integration</u> P: Subsystem Fabrication/Integration	<u>Individual Evaluation Subsystem Integration</u> P: Subsystem Fabrication/Integration	<u>Individual Evaluation Subsystem Integration</u> P: Subsystem Fabrication/Integration	Optional Session for All Sections
Week 6 (Feb. 11-15)	L: Project Planning (WB116) P: Subsystem Functionality	P: Subsystem Functionality	P: Subsystem Functionality <u>Interim Notebook Evaluation All Students, Submission to Instructor in SF4003 by 18:00</u>	Optional Session for All Sections
Week 7 (Feb. 18-22)	Family Day	READING WEEK		

Week 8 (Feb. 25- Mar. 01)	L: Decision Making Process (WB116) <u>Individual Evaluation</u> <u>Subsystem Functionality</u> P: Subsystem Functionality, System Integration	<u>Individual Evaluation</u> <u>Subsystem Functionality</u> P: Subsystem Functionality, System Integration	<u>Individual Evaluation</u> <u>Subsystem Functionality</u> P: Subsystem Functionality, System Integration	Optional Session for All Sections
Week 9 (Mar. 04-08)	L: Decision Making Process (WB116) P: System Integration	P: System Integration	P: System Integration	Optional Session for All Sections
Week 10 (Mar. 11-15)	L: Reliability and Risk Assessment (WB116) <u>Team Evaluation</u> <u>System Integration</u> P: System Integration/Functionality	<u>Team Evaluation</u> <u>System Integration</u> P: System Integration/Functionality	<u>Team Evaluation</u> <u>System Integration</u> P: System Integration/Functionality	Optional Session for All Sections
Week 11 (Mar. 18-22)	L: Reliability and Risk Assessment (WB116) P: System Functionality	P: System Functionality	P: System Functionality	Optional Session for All Sections
Week 12 (Mar. 25-29)	L: Professionalism in Design (WB116) <u>Team Evaluation</u> <u>System Functionality</u> P: System Functionality/Debugging	<u>Team Evaluation</u> <u>System Functionality</u> P: System Functionality/Debugging	<u>Team Evaluation</u> <u>System Functionality</u> P: System Functionality/Debugging	<u>Good Friday</u>
Week 13 (Apr. 01-05)	L: Ethics in Design (WB116) <u>Project Review</u> P: System Debugging	<u>Project Review</u> P: System Debugging	<u>Project Review</u> P: System Debugging	Optional Session for All Sections
Week 14 (Apr. 08-12)	<u>Public Demonstration</u>	<u>Public Demonstration</u>	<u>Public Demonstration</u>	<u>Return Lab. Kits,</u> <u>Close Design Store Account</u> <u>All Teams</u> <u>(14-17, SF4003)</u>
Week 15 (Apr. 15-10)	<u>Final Report Submission</u> <u>All Teams by 17:00</u> <u>Notebook Submission</u> <u>All Students by 18:00</u>			

1.3 Procedure

The Engineering Design course, by nature, is a sophisticated process, due to its hands-on characteristics, large number of students, and wide variety of requirements that students must fulfill to pass the course successfully. The course proceeds through the following steps. It is imperative that students fully understand these steps and meet their requirements. The lack of appreciation of each step may result in severe penalties for individuals and teams.

Course Introduction: Instructor describes the course outline, schedule, marking scheme, regulations, expectations, etc. in the first introductory lecture of Week 1.

Course Note Distribution: Course notes can be purchased from the University Bookstore. Students are advised to acquire the course notes as early as possible, preferably on the first day of semester after the introductory lecture.

Project Definition: Instructor discusses the design projects and related issues for the entire class in Week 1. This-year projects are posted on the course web site. Students will discuss their projects with the TAs during the introductory lab. session in Week 2.

Design Portal Login: The course is facilitated by a central gateway that provides various utilities for effective team communication, collaboration, and other design activities. Students are advised to log onto the portal (Username: aerospace\<student number>, and Password: <student number>), and set their passwords after the Portal seminar during the first week. In addition, students will share their experience of using the development tools in the course at www.pml4all.org.

Team Assignment: Design projects are introduced on the first day of semester. Students then study their project, discuss it with their colleagues, acquire some information from higher-year students as well as other experts, and select their team members. Note that in this year students have already been divided in three sections, and given that there is one design project for each section, they do not select their project. There is a designated period for submitting the elected team to Instructor. This period starts from Wednesday January 9th at 18:00 and ends at 21:00 Friday January 11th. Students can take the opportunity during the lectures and seminars in Week 1 to look for teammates. Those who do not submit a team will be assigned to a team by Instructor in the second week of semester.

For team submission, one member of each team must go to www.aerospace.utoronto.ca/DesignTeamSubmission13 and fill out a form and submit it. If a student shows up in more than one team, no submission will be accepted from any of those teams and all students in those teams will be assigned to teams by Instructor. Appreciation of the fact that there will be three major subsystems, namely Electromechanical, Circuits, and Microcontroller, could help students team up with the right people. On the team-assignment form it is also specified which subsystem is carried out by each member.

First Meeting with TA's, Lab. and Project Kit Distribution, and Design Store Accounts: In Week 2, teams show up in their designated lab. sessions to meet and discuss their project with their TA's, receive and check their Design and Project kits, and open Design Store accounts. For the Design Kit, each team leaves \$100 deposit to return the kit at the end of the course in its best shape. Charges will be made against the deposit in case of damage or lost, as explained in Chapter 2. Each team also purchases one Project Kit that contains most of the devices, parts, and materials that are needed for the project. The cost of the Project Kit is \$350, and the payment must be in money order (can be obtained from the UofT Bookstore) payable to "University of Toronto." In addition to the Design and Project kits, some additional items can also be purchased from the Design Store. These items are listed in Chapter 2. Those teams that foresee to purchase some items from the Design Store must open an account with the store, and deposit \$100 into their account during the first lab. session. All purchases from the Design Store will be charged against the account, and the remainder will be returned to the teams at the end of the course. Since no item in the Design Store can be purchased by cash, all teams are advised to open an account with the Design Store. During the introductory lab. session the electromechanical members will also attend the machine shop safety training as explained in a subsequent section.

Design Lectures: A series of Design lectures is provided during the first week and Week 5-13 on topics such as teamwork, information collection and communication, project planning, decision making, risk assessment, and professionalism and ethics. All students are strongly advised to attend these lectures.

Design Seminars: Four seminars will be held during on Friday of Weeks 1, 3 and 4 between the lectures. The first seminar is conducted by some senior students discussing their experience of passing AER201 successfully. The second seminar will be about the challenges that designers face dealing with real-world problems. In the third seminar, the Utility and Driver boards that are included in the Project Kit will be introduced.

Technical Lectures: During the first four weeks of semester, Instructor delivers technical lectures on the PIC Microcontroller, Circuits, Electromechanical Systems, and Sensors. These lectures provide the minimum knowledge that students need to conduct their design projects. All students are advised to attend technical lectures regardless of their favourable subsystem. Past experience has shown that teams with members who tried to grasp all three subsystems together had a higher success rate during the integration and system-debugging phases.

Machine Shop Safety Training: This four-hour workshop, held in Week 2, is for Electromechanical members only. The Electromechanical member of each team will attend the workshop in her/his designated laboratory session. The location is the Aero-Design Lab Machine Shop in SF4003. **Attendance is mandatory for Electromechanical members.**

Design Laboratory Sessions: Each team must attend one designated laboratory session per week. **Note that attendance of all students in their designated laboratory sessions is mandatory, and closely observed by Instructor and TA's.** Laboratory sessions start from Week 2. It is important that students begin and finish their session on time (packing up their stuff 15 minutes before the end of session), and perform efficiently and behave professionally during the session. Their performance is constantly observed by Instructor and TA's, and will be part of their evaluation criteria. In addition to the designated sessions, Friday sessions (14:00-18:00) are also available to all sections optionally.

Submission of the Proposals: Teams complete their preparations and conceptual design phase by Week 3-4, and submit a hard copy of their proposals to Instructor in SF4003 by Wednesday, January 30th at 14:00. Late submission is not accepted for any milestone in this course. This year, there will be 1.5% bonus for on-line submission (in addition to the hard copy). Each team (only one member) can submit an electronic copy of the “complete” proposal (all pages, appendices, figures, tables, etc.) as one PDF through the Design Portal. Note that incomplete documents or any other formats than PDF will not be accepted. The deadline for submitting the Electronic version is also 14:00 on Wednesday January 30th.

Reading Week: The Tuesday and Wednesday laboratory sessions will be open to all teams, but attendance is not mandatory. (University is closed on Monday, and there will be no Friday lab. session in the Reading Week.)

Week-5 Individual Performance Evaluation: In Week 5, the progress of individual subsystems will be evaluated. At this stage, each team is expected to have a reasonably complete analysis/simulation of their system containing all three subsystems. All the necessary hardware for each subsystem must have been acquired, and some sections of each subsystem must be functional. At this stage, the machine overall frame and structure must be complete. Also, the Electromechanical member must show the full functionality of at least one actuation mechanism in the system, and that building some other mechanisms has been started. The Circuit member must also have a complete circuit for the actuators and sensors on the breadboard, and provide adequate investigation of the possible effects of signal interference when all the drivers and sensors will be working together. Power specifications must also be complete, and power supplies and cables must have been acquired. For the Microcontroller member, the code for running the keypad and LCD along with the first version of the machine interface must be complete and functional. The pseudo-code of the final program must also be ready. Further details of grading will come in the sequel.

Notebook Grading: Each student records all the relevant activities performed during the design project in a notebook. This notebook will be evaluated twice, once in Week 6 and the other at the end of the semester. Please note the notebook submission deadlines in the course schedule. Late submission is not accepted for any milestone in this course. Details of how to generate an engineering notebook are addressed in Chapter 4, and they will be discussed at the Design lecture in Week 1.

Week-8 Individual Performance Evaluation: The process is similar to Week-5 evaluation, but at this stage, each subsystem is expected to be entirely complete and functional. Note that this evaluation will be followed by a team evaluation right in the consequent Week 10. Therefore, it is crucial for the team to understand that each subsystem must indeed be complete by Week 8 so that the group will have time to start the integration right after to be able to receive a satisfactory team evaluation in Week 10.

Week-10 Team Progress Evaluation: Integration of subsystems is expected to be complete by Week 10, when the status of the project will be evaluated for each team. At this stage, the system could still suffer from some defects or deficiencies that must be debugged during the rest of the project.

Week-12 Team Project Evaluation: Teams are expected to demonstrate the entire functionality of their project. Successful projects will be sent to the public demonstrations. Some incomplete projects may also be eligible for public demonstration if Instructor recognizes that it can be completed with reasonable effort within two more weeks.

Project Reviews and Public Demonstrations: Complete projects will attend the demonstration, and those that are not complete shall go through the review process, as will be explained in the sequel.

Lab. Kit Return and Closing Design Store Accounts: Laboratory kits must be returned **only** within the designated time slot, i.e., 14:00-17:00 on Friday April 12th. No kits are accepted before or after this period. Please note that those who do not return the kit cannot submit the final report. During the same period, Design Store accounts must be cleared and closed.

Final Report Submission: This is the final stage of the course activities. The deadline is Monday April 15th by 17:00. Similar to the proposal, there will be 1.5% bonus for on-line submission (in addition to the hard copy). Late submission is not accepted for any milestone in this course. Each team (only one member) can submit an electronic copy of the “complete” report (all pages, appendices, figures, tables, etc.) as one PDF through the Design Portal. Note that incomplete documents or any other formats than PDF will not be accepted. The deadline for submitting the Electronic version is also 17:00 on Monday April 15th.

1.4 Marking Scheme

1.4.1 Project Cost

Students will be required to state and break down the cost of the project in the final report, supported by the receipts in their laboratory notebooks.

The spirit of the following guidelines is to evaluate the cost of the project on terms common to all students.

- The cost of unsuccessful components or mistakes will not be included, nor will the cost of labour by students in the group.
- The cost of genuine, scrapped material, in the garbage, heading for landfill and salvaged will be assigned the value that was paid for it.
- Other disused material “donated” or sold to the project shall be assigned a fair value unless everyone in the class who needs one/some is supplied with the material. If an ample supply can be procured then the value assigned is what was paid.
- Manufacturer's samples will be expensed at full price unless everyone in the class who needs one for their project can get one for less.
- The cost of specialty processes, performed by others must be expensed at the market value for that labour, even if it is donated by a friend or acquaintance.

1.4.2 Project Proposal (10%)

(3%)	Title Page
(7%)	Executive Summary
(10%)	Problem Formulation
(10%)	Survey
(15%)	Conceptualization – Alternative Solutions and Decision Making Criteria
(15%)	Specification – Description and Analysis of the Selected Solution
(15%)	Project Management – Task Assignment, Statement of Work, and Gantt Charts
(5%)	Budgeting
(5%)	Conclusion
(15%)	Figures, Tables, and Supporting Documentation
(1.5%)	Bonus for On-line Submission (in addition to the hard copy)

In this course, in addition to proposing a selected solution, proposals also report the phase of conceptual design. Though neither neatness nor English are graded here, legibility is appreciated and English errors will degrade the overall appearance of the material.

Each group shall submit a project proposal where the design conceptualization process is reported and the concept of the selected design is laid out. Proposals shall contain a reasonably complete survey and a detailed schedule for each group member's subsystem. The proposal for this course is the collected back-of-the-envelope figures and calculations that are always done to prove a design is possible. Alternative ideas should be presented clearly; drawings should depict the system in the most descriptive views. Material must be legible. CAD drawings are neither necessary nor desired at this point. CAD drawings may restrain the creativity and the freedom afforded in hand drawings to add appropriate notes, annotations and accompanying calculations - all on one page. Calculations, photocopies of supporting documents and tables from manufacturers or data books are not only encouraged but expected. This supporting data is especially important for the critical points of the design, i.e., the one, uncommon chip on which a critical part of the design hinges. Documentation is not expected on common components such as nuts, bolts, rivets, screws, wire or solder.

The project Proposal should acknowledge issues of cost and complexity. A proposal should reject the use of a Pentium processor to turn some switches on and off when not even a PIC16F877 would be required to control an office building full of elevators, automatic doors and lights. It must also convince the reader that the design budget, considering all the uncertainties, will remain within the assigned limit.

The Project Proposal shall consider issues of reliability and Factors of Safety (FS), where necessary.

Concepts for more than one design alternative should be presented where appropriate. A definite protocol, schedule and criteria for making the selection between the alternatives shall be presented.

No limit is set on figures (a good proposal may contain 10-25 pages of informative sketches.) However, students should be realistic and present a clear, complete CONCEPT of their system. All figures should have a title and appropriate annotations to indicate their place in the proposal as a whole - do not leave the reader guessing. A very terse text is expected of no more than 30 single-spaced typed pages.

Task assignments to the team members must be explained clearly and in as much detail as possible. The schedule, in Gantt charts, must include at least weekly milestones for each subsystem (team member) including all the completion milestones mentioned in the course schedule. This schedule, tempered by the marker's judgment, will be used as a yardstick to measure the student's weekly performance. Review mechanisms, such as team meetings, meetings with Instructor and/or TA, external visits, etc., must be indicated.

Each copy of the proposal shall be bound separately in a gray or blue laboratory report cover (available from the Engineering Stores) with a staple affixing the pages to the inside of the back cover. Proposals not in this format will not be accepted.

This year, there will be 2.5% bonus for on-line submission (in addition to the hard copy). Each team (only one member) can submit by the proposal submission deadline an electronic copy of the "complete" proposal (all pages, appendices, figures, tables, etc.) as one PDF through the Design Portal. Note that incomplete documents or any other formats than PDF will not be accepted.

Instructor and TAs mark the proposal, and the grade is for all members of the group. This grade shares 10% of the overall course grade. Further, Instructor and TAs will review the proposals and use them in the weekly performance appraisals. Further details about writing an engineering proposal are given in Chapter 4.

1.4.3 Individual Performance Evaluations (22.5% total)

There will be two evaluation sessions for each student in Weeks 5, and 8 of the course. Instructor and teaching assistants will appraise each student's performance against her/his schedules. Evaluation grades will be balanced and uniformed statistically to make the evaluations independent of TA's characteristics, subsystem differences, and other biasing effects.

In Week-5 evaluation, the analysis and/or software simulation of each subsystem must be complete; hardware identification and acquisition for each subsystem must be done; and the following progress must be visible for each subsystem:

- **Electromechanical:** The detailed geometry, kinematics, and weight and moment of inertia estimation are complete. All the hardware (actuators, mechanisms, materials, etc.) has been acquired. The machine overall frame and structure must be complete. Also, the Electromechanical member must show the full functionality of at least one actuation mechanism in the system, and that building some other mechanisms has been started.
- **Circuit:** The detailed analysis of currents and voltages is ready. Power requirement and specifications are finalized, and power supplies and cables have been acquired. All circuits for the actuators and sensors are made on the breadboard. Some investigations on the possible effects of signal interference when all the drivers and sensors work together must be presented.
- **Microcontroller:** The code for running the keypad and LCD along with the first version of the machine interface must be complete and functional. The pseudo-code of the final machine program must also be ready.

In Week-8 evaluation, each student should be able to demonstrate the complete functionality of her/his subsystem. That is, the electromechanical member must have completed the entire machine platform and mechanisms; the Circuit member must have assembled all circuits on the solder board and debugged them; and the Microcontroller member must have completed the final machine code and downloaded on the PIC by demonstrating its functionality. It is very important for the team to understand that each subsystem must indeed be complete by Week 8, so that the group will have time to start the integration right after to be able to receive a satisfactory team evaluation in Week 10.

Instructor and TA's mark the evaluations, and the grades shares 10% and 12.5% of the overall mark, respectively.

Evaluating the students' performance is an ongoing process starting from the second week when they show up in the lab. The following factors are taken into account in grading the student's performance:

(50%) Effort / Outcome

Following the assigned schedule, delivering the milestones, arriving on time at the lab. in the designated sessions, terminating the activities and packing up in time at the end of each session, concentration on the problem at hand, working efficiently.

(10%) Preparation

Background research, gathering material together to prepare the work, arriving at the laboratory ready and able to work immediately.

(5%) New Skills

Does the student recognize the need to new skills for the project? Does the student improve her/his skills during the course?

(10%) Ability to Recognize and Solve Problems

Does the student regularly require the instructor or TA to criticize her/his work? Does the student follow feedbacks positively?

(5%) Innovation

How does the student improve or expand the original concept or find novel solutions to problems within the concept?

(5%) Contribution to Group

Does the student pull her/his weight? Is the student instrumental in causing or resolving group disharmony? Does the student put in extra work to benefit the project? Does the student communicate well with teammates?

(10%) Safe Work Habits

Does the student work safely? Is the student aware and vigilant of dangers to humans and equipment? Does the student follow safety instructions in the laboratory and machine shop?

(5%) Care in Organization

Does the student customarily comment code and label wires and connectors to reduce risk of bugs and mistakes? Does the student care about the safety, reliability, and robustness of what s/he fabricates?

1.4.4 Design Notebook (15% total)

Each student shall keep an engineering notebook for the design project. The notebook shall contain a **complete** record of all matters pertaining to the project from shopping lists, to notes from design and technical lectures, to the evolving sketches of the project. These notebooks are marked twice by TA's during the course, and each mark forms 7.5% of the overall course grade.

The design notebook is the most important record of student work because often students will rely on it to write the final report and prepare for other evaluations. A black or blue physics-type hard cover notebook is required for each student. All entries shall be made in ink because pencil will smear leaving the first pages unintelligible after a term of riding in a book bag. All entries shall be dated. All entries that are not purely the result of a student's imagination should be referenced in some way. The source or inspiration for all circuits should be noted. Any casual reference to sources should be given in full format somewhere in the notebook, i.e. perhaps on one bibliography page near the back. All entries that show a circuit, a design, a part or anything purchased from, copied from, adapted from or inspired by a source other than a student's uninfluenced imagination should contain a notation to indicate the source of the entry. The student will always be able to determine where something was purchased, why this subroutine was included, where the interface format was originally specified, etc. A clear organization of the notebook would end up very helpful, i.e., making a section for lecture notes, another one for meetings, etc. Some features of a lab notebook:

- At the front:
 - your name and information sufficient for return if it is lost;
 - a disk or CD envelop with a disk or CD in;
 - a table of contents listing the page numbers of all entries up to the current status.

- A chronological record of all work and notes concerning the project dated on number pages
- Near the back:
 - a page of references to people (tape in their business cards);
 - a page of references for companies and suppliers;
 - a page of all full references for books cited in the notebook;
 - a page of all important and relevant web sites visited during the project;
 - a page where all library call numbers are scratched down during computer searches, even if the book is not accessed;
 - an envelope glued into the back cover for all project receipts.

Students shall obliterate credit card numbers from receipts to protect their credit information should your book be lost, picked up by another individual or while it is being marked or returned.

More information about maintaining the engineering notebooks is given in Chapter 4. The following factors are taken into account for marking the notebooks:

(10%) Basic Organization

Table of contents, page numbering, dates, directories of important material, use of volumes, different sections all clear?

(45%) Technical Content

Ideas, drawings, notes, sketches, flowcharts, sample calculations, pseudo-codes. Do they appear when appropriate? Is the content complete? Are all details included? Lecture notes are also part of the technical content. For the interim design book marking, the evaluator checks how clearly and completely student has taken notes from the lectures.

(5%) Receipts

Are all purchase receipts and invoices included in the notebook?

(10%) References

Phone numbers, addresses, page references in the data books.

(20%) Schedules and Task Planning

Is it presents? Is it followed? Do multiple schedules appear to indicate revisions of the plans? Are notes from the meetings present? Are project review discussions and thoughts present?

(10%) Description of Experiments and Results

Is it clear and detailed? Are logical conclusions made after each experiment?

1.4.5 Team Evaluations (32.5% total)

The performance of each team will be evaluated at the system integration, debugging, and presentation stages in Weeks 10, 12, and 14, respectively.

In Week 10, the entire system is expected to be integrated and “more or less” functional, i.e., some indications of functionality must exist in the system. In Week 12, the debugging activities shall result in a complete and functional system. Week 12 evaluation is crucial, because based on this evaluation Instructor will decide whether or not to send the project to the public demonstration event. Those projects that do not illustrate reasonable functionality cannot go to the public demonstration. However, under special circumstances, some of the incomplete projects may continue their development within the remaining two weeks before demonstration, if Instructor recognizes that there is a fair chance of completion. For other incomplete projects, students will be advised to stop working and concentrate on preparing the project review and finishing their final report. Students who continue to work on their projects after this point without Instructor’s permission will not earn a demonstration time slot and are risking the quality of their final report - worth 20% of the grade.

Instructor and TA’s will evaluate the teams, and the grades for Week 10, 12, and 14 are 7.5%, 12.5%, and 12.5% of the overall course grade, respectively. Similar to the individual evaluations, group evaluations take place throughout the course, and it does not occur within one session only. The following factors are taken into account in grading the team’s performance.

One important factor in marking the Team Work portion of all team evaluations is the team conduct out of laboratory sessions. In the previous years, the Engineering Science Office and/or Residence Offices had serious complaints about poor conduct by some design teams in their residence or in the common rooms for making noise, disturbing other people's convenience, creating mess, and not cleaning after them. This year, if Instructor receives a complaint about a team, after investigation and verification, the 20% Team Work will be deducted from the evaluation mark for that team.

(50%) Effort / Outcome

Following the assigned schedule, reaching the milestones, i.e, completion of the system and functionality

(5%) Preparation

Arriving at the laboratory well prepared with a specific agenda

(20%) Team Work

Harmony in performing tasks, close interrelation between members of the team, ability to resolve any dispute or disagreement quickly and independently. Any complaints about the team conduct out of lab. sessions?

(10%) Load Distribution

Is the team able to distribute the load equally among the members? How well does the team divide the subtasks between members?

(10%) Communication

Does the team interact efficiently with the TA? IS the team aware of the activity of the other teams working on the same project?

(5%) Innovation

Is the team able to further expand and/or improve the original proposed idea?

1.4.6 Public Project Demonstrations

Projects judged functional by Instructor at Week-12 Progress Evaluation will be examined at the public Project Demonstration. All group members shall attend and will make available design notebooks and supporting calculations, and shall answer questions about the functionality of their design. A functioning project is its own best representative at the demonstrations thus rewarding the hard work and good design of the students over the term. Demonstrations are not without their tension due to potential for failure in front of the public. In a company, even a delay of a review may be less embarrassing than inviting the customer to a demonstration and having the product fail right in front of them when moments earlier the designers looked confident. All students are encouraged to be present on the day of demonstrations and observe and ask questions from the students making a demonstration.

Conduct during the public Project Demonstration is very important. Students are presenting their own work but also representing Engineering Science and the Faculty to the relatives, high school students and other members of the public who attend. Having a poster or display for describing the machine in a graphical manner would be a proper option. Teams will be marked on how well they present their machine as well as their interaction with visitors, particularly on their answers to all questions, including those of their classmates. Where the project demonstration takes the form of a competition, the team organization, sportsmanship, and other conduct that contributes to the competition and audience interest will be weighted into their interaction mark, along with their answers to questions. Teams who bury their heads in the project trying to correct a small flaw (like a loose wire to one of 8 sensors) leaving none of their group attentive to visitors are missing the point of the demonstrations. In such a case, at least one group member should be standing (not sitting) facing visitors answering all questions and explaining to visitors the operation currently underway. This is especially true of competition projects. When not in a competition, they should be in a publicly accessible area talking to visitors.

The following factors are taken into account for marking this stage:

(50%) Full Basic Functionality (must be at least 30 out of 50)

(15%) Extra Design Functionality and System Reliability

(25%) Presentation and Demonstration

(10%) Answers to Questions

1.4.7 Project Review

Projects judged non-functional at the Week-12 Progress Evaluation will be presented and fully examined at a Project Review session. All members of a group presenting at the Project Review shall make available their lab notebooks and supporting calculations. Each group member will make a short presentation outlining the design of their subsystem and how it fits into the final project. Students shall not accuse other team members of failing to meet their group obligations, but may accept responsibility for failing to meet their own group obligations. Following each presentation the markers may ask clarifying questions. After all members of a group have presented, the floor will be open to questions from the markers.

Such reviews are common in companies when projects are delayed, or have failed to meet specifications or design objectives. The review is a surgical, but in general, a constructive process. Students should familiarize the audience with their project, subsystem, its problems and potential solutions. Design reviews are stressful. However, since your audience members are designers and, particularly in a company, have made similar mistakes and probably even been in the hot seat at one time or another themselves, they can identify with a presenter who is open and accurate in their analysis of the problems and suggests solutions. Presenters who are less than open or seem to have failed to recognize a major flaw will find the questions more pointed. In companies, staff changes often follow reviews. Be thankful it is just a few marks at this point. Individual marks are assigned to the group members once the panel is satisfied. Individual marks are assigned because invariably an unequal contribution has been made by the group members of an incomplete project.

Both the design demonstration and review are very important experiences for the class that will help students better understand key aspects of design presentation in both cases of success and failure. Note that some projects that were sent to public Project Demonstration event but obtained less than 30 (out of 50) for the Full Basic Functionality may also be called back for a Project Review session.

The following factors are taken into account for marking the project reviews:

- (50%) **Fraction of Functionality**
- (30%) **Met Team Obligations**
- (20%) **Presentation and Answers to Questions**

1.4.8 Final Report (20%)

Each team will submit a final report covering their work in detail, which will become part of the permanent Aerospace Laboratory resources. Reports will not be accepted unless they are cerlox bound with a proper cover page presenting the given format, and include a colour photo of the group and project. Students who wish to retain a copy should photocopy their report **before** handing it in. Under no circumstances at any time in the future, even for PEY interviews, will the Instructor either loan the report or unbind the report and attempt to feed it through an auto-feed photocopier for students who neglected to do it beforehand.

The final report has 20% share of the overall grade. Instructor will mark the report. Marks are allotted as indicated below, and some guidelines for the report format are given in Chapter 4.

COMMON

- (-) Colour Photo of the team and prototype (required for submission)
- (2) Front Cover
- (3) Title Page
- (5) Acknowledgement
- (45) Abstract
- (5) Table of Contents
- (5) Symbols and Abbreviations
- (20) Introduction
- (100) Perspective: Theory, History, Survey, etc.
- (50) Objectives, Constraints, Acceptance Criteria in Decision Making, and AHP Analysis
- (60) Budget
- (60) Division of the Problem

SUBSYSTEMS

- (100) Assessment of the Problem
- (150) Solution
- (100) Supporting Calculations, Computer Programs, Simulation Results
- (50) Suggestions for Improvement of the Subsystem
- (50) Tables
- (200) Figures

COMMON

- (100) Integration
- (100) System Improvement Suggestions
- (100) Accomplished Schedule
- (100) Conclusions
- (100) Description of Overall Machine (Figures, Tables)
- (100) Standard Operating Procedure
- (15) References and Bibliography

GENERAL

- (5) Page numbering
- (15) Proof Reading, English and Style
- (10) Logical Organization

TOTAL 1000 (Common and General) +650 × (subsystems)

Additional 1.5% bonus for on-line submission (in addition to the hard copy).

Reports that actually include all parts for all subsystems, and present the material in an organized manner receive a good mark. This means for instance Microcontroller requires both figures and tables to properly report it and if they are left out, the report loses 250 marks. Similar to the proposal, there will be 2.5% bonus for on-line submission (in addition to the hard copy). Each team (only one member) can submit an electronic copy of the “complete” report (all pages, appendices, figures, tables, etc.) as one PDF through the Design Portal. Note that incomplete documents or any other formats than PDF will not be accepted. The deadline for submitting the Electronic version is the same as that for hard copy submission. Late submission is not accepted for any milestone in this course.

1.4.9 Allocation of Team Marks

The default assumption is equal distribution of team marks among the members. However, if a member finds reasons that this distribution is not fair, s/he shall raise the issue to Instructor by sending an Email as early as possible but no later than Monday April 15th at 18:00. In this case, Instructor will call all members for a meeting in which a percentage of the team mark will be assigned to each member. The percentages will add up to 300% for a team of 3 students. All team members will then sign the mark distribution in front of Instructor, and indicate verbally that they agree to this mark distribution.

1.5 General Notes

1.5.1 Safety First!

Safe working procedures must be followed at all times. All students must read the safety notices in the workshop and on the bulletin boards. At all times, students must be aware of physical hazards around themselves in the form of stored energy, sharp edges and ends, and toxins.

Hazards due to stored energy can be divided into kinetic and potential energy. Kinetic energy, in the form of moving machinery can pose many hazards. Machinery can throw shavings or parts when broken and consequently should always be treated with respect. Though not proof against all injuries caused by moving machinery, eye protection is mandatory at all times in the workshop and whenever working with power tools or tools that strike metal on metal.

Tangle hazards are serious problem around moving machinery. Loose clothing, ties, scarfs, jewelry and long hair pose a significant tangle hazard around machinery such as fans, blenders, mixers, dryers, drill presses, lathes, gear trains, chain-driven sprockets (as on bicycles) or vehicles. Once entangled, the user is drawn closer to the moving parts. Even if the machinery is turned off, its angular momentum may still draw the person in and cause severe injury. In some cases gloves are a liability, especially if they are a type that tighten on the hand when pulled at a fingertip (normally this type of glove would be removed by pulling at the wrist). In industrial situations, material being drawn into hoppers and wire, cable or rope being drawn into machinery poses the hazard of entangling people and drawing them in as well.

Related tangling hazards not commonly found in the lab but common causes of death and injury in the community are caused by long hair in swimming pools, hood drawstrings on children's clothing, and hammocks. When a person swims too close to certain water intake covers their hair is easily drawn in through the grill and tangles on the other side due to turbulence. The person becomes tied by her/his hair underwater and only quick thinking and a pair of scissors will save them. Similarly the drawstring of the hood of a child's jacket can become caught on the top of a slide or jungle gym choking the child until someone lifts them off. When dismounting from, or playing in, certain types or badly installed hammocks, typically small children, but occasionally teenagers as old as 17, have become entangled around the neck and hanged.

Hazardous potential energy can take a number of forms but most are so common in everyday life we do not recognize the hazard. Stored chemical potential energy poses a severe fire hazard. We think of explosives as being very energetic and dangerous, but common materials and fuels like paper and gasoline contain far more energy per unit mass than dynamite. Fires in a small room with ample combustible material can generate a power output comparable to that of a small generating station. Fire and explosion will be a significant portion of the safety lectures associated with this course however, all students must make themselves familiar with the practical aspects of fire in the laboratory before beginning work. There are three fire extinguishers (suitable for all fire types), in the laboratory, all students must familiarize themselves with their location and read the instructions regarding their use.

Stored mechanical energy in the form of pressure in gas cylinders or tightly wound springs is another common hazard. In all cases, students must recognize the potential hazard in such items and treat them with respect until known to be discharged or saved.

Potential energy due to elevated mass is another commonly overlooked hazard. Most would see a 40-60kg mass suspended precariously above people as quite hazardous yet most would not think twice about climbing up on a rickety chair or reaching beyond a safe distance on a ladder. Falls are a common cause of death and injury. Consider even a short fall in a cluttered space. The chances of becoming impaled on something usually considered harmless are significant - like pencils standing up in a cup on your desk. Students are prohibited from climbing on anything but the step-stools provided in the laboratory and then, only with caution.

An obvious but invisible hazard is stored electrical energy and electrical sources. The chapter on electric circuits deals with specific hazards. However, in general, students should exercise extreme caution with any equipment that plugs into the wall, even after it is unplugged, and any circuit containing inductors.

Soldering irons and other hot items have a great deal of stored heat energy, which has the potential for harm. Care must be taken with soldering irons not to burn yourself or others. What is not so obvious, is to take care where you set it down, it can burn through live electrical cords and cause fires. All equipment kits come with a soldering stand to attempt to minimize hazards. Please make sure it is put on a solid level surface and electrical cords, including the soldering iron's own cord, are not allowed to lay across the iron through the wires in the holder. Lastly, when at home and in the lab, students should be sure to wear appropriate clothing when soldering. Double-knit polyester and other synthetic fibers will melt onto your skin when contacted by hot solder and shorts and open tops are a poor idea since solder splashes outward when it falls onto a flat surface. For this reason, and because connections sometimes sputter when heated, safety glasses are required when soldering.

Sharp items pose a hazard to people when improperly used or stored. When not in use, blades, knives, scribes, pins, and even ICs should be in their proper holders. Many students have 16 little punctures in the bottom of a foot, in two perfectly spaced lines of 8, from a 16 pin DIP chip left on the floor in their residence room after working late the night before - an infected foot is not funny. When in the lab students must wear shoes and clothing suitable for the small physical hazards in the environment. When using cutting edges, always face the blade away when applying force in case the blade slips. Never cut against your person, such as your thumb or leg (the movie The Edge), always use a piece of scrap plywood or plastic to cut against. (Please don't cut against the lab tables, at least your thumb will probably heal by itself in time if left alone, the tables will not.)

Many toxins could pose a significant health hazard in the laboratory. Because of the communal nature of the air circulation system, the use of paint and exterior caulking materials are not permitted in the laboratory or workshop. Solvents such as alcohol, acetone etc. must be used with discretion. The vapour levels of these solvents considered safe in the workplace is often only slightly greater than the concentration at which they can be detected by smell. No industrial chemical may be brought into the laboratory or used without the knowledge and approval of the instructor. All consumer products that carry warnings of a toxic or volatile nature must be approved by the instructor before use. Any chemical, but particularly solvents and adhesives must be thoroughly cleaned up after use.

For safety reasons, food and drinks are strictly forbidden on or near any table where solvents, adhesives or electrical equipment is present.

Remember, Epoxy is deadly and Coke is an excellent conductor.

1.5.2 Attendance

On the course schedule, there are several events where the entire class or students from a particular subsystem are expected to attend. **All students must arrive at their designated lab. sessions on time, stay and work in the lab. for the entire session, and terminate the activities, pack and clean up in time at the end of each session.** TA's will also be present at all sessions, each helping 4-6 teams.

The distribution of projects across the lab days and TA's is designed to provide support for students in all laboratory sessions. TAs from the other teams can be a valuable source of suggestions and can help students form a better idea of the state of their project. Students are encouraged not to overuse these other TA's; their first responsibility is to their own student groups. Of course, Instructor will be available in all laboratory sessions to provide assistance.

Attendance is mandatory for all AER201 students during their designated laboratory periods. During each scheduled lab period every student must contact their TA and provide an activity update. Students are expected to inform their TA where they intend to work during the laboratory period should the TA or instructor need to find them. Students who were not found when needed are considered absent. Unless specific approval is granted to work elsewhere, all students are expected to remain and work in the laboratory or other scheduled design classrooms for the full laboratory period. If students are required to eat (or leave the laboratory for an extended period for some other activity) for reasons of religious faith or a medical condition, they should inform their TA before they leave. At the end of each session, all students must clean up their working station and leave in time. **Laboratory session will be closed at 18:00 sharp.**

All planning, literature searching (studying in the library), parts shopping etc. shall be done outside of student's regularly scheduled laboratory period.

Although attendance is not checked during Design and Technical lectures, all students are strongly advised to be present in all lectures. Some critical information will be presented during these lectures. Further, this would be a unique opportunity for all members to know about all subsystems. **Past experience has shown that teams with members who tried to understand all three subsystems together were far more successful during the integration and system debugging phases.**

1.5.3 Petitions and Extensions

This is just a restatement of what is in the Faculty Regulations. The Faculty policy regarding Petitions for Consideration for Term Work requires students to present petitions directly to the course instructor. Owing to the large number of students in AER 201, the administrative task of collating marks requires all requests for consideration to be made by petition. Documentation is then present some weeks or a month later when marks are collated assuring all cases are properly reflected in the marks. All petitions must be accompanied by documentation. Where medical reasons lead to the petition, students are required to include a note from the doctor. The note should bear the doctor's name, address and phone number (their note pad or letterhead) stating the degree to which and time frame during which the illness impacted the student's work. For cases where a family emergency, or legal proceeding mandates the student's absence from class, a neatly, handwritten letter from the student, signed and dated by the student will be accepted if no other documentation (such as a photocopy of a court appearance order) is available. Always include your specific suggestion for how the marks may be fairly evaluated for your case. Please present any petition and documentation in the first laboratory period directly following the absence or illness.

1.5.4 Laboratory Hours

The laboratory will not be open on any additional days and hours for work on AER201 projects other than the designated sessions. Please do not expect to extend the lab. sessions beyond 6p.m. Any additional hours, or days, particularly during Reading Week, will be posted. The best times to ask Instructor for advice are when the population in the lab is at a minimum.

1.5.5 E-Mail and Lecture Announcements

Course announcements will be made in the beginning of Design and Technical lectures, including scheduling and procedural changes. Most of the announcements will also be sent out via e-mail and/or posted in the Design Portal. All students are advised to check the Design Portal regularly.

1.5.6 Machine Shop

Only Electromechanical members will be permitted to enter the machine shop. There will be a training session on machine shop operation and safety, after which the shop will be open to student access. Electromechanical students will be required to attend a specific in-shop safety and procedures workshop before using any power tools or machines. All Electromechanical students are required to bring their safety glasses in every laboratory period. Any other student having work requiring one of the machines is to give it to the Electromechanical group member. This policy will be enforced in an effort to reduce crowding in the machine shop (the largest hazard), reduce the chance of damage to the machines by misuse and eliminate the chance of injury to students not trained in the used of the machines.

1.5.7 Laboratory Cleanliness

Each student is required to clean up their station and the floor underneath directly after completing work there. In the workshop this means students must clean before exiting the workshop area (even for a moment) or before leaving a machine to another student. Safe and tidy work habits are a factor in the mark evaluation. During and after each laboratory session, notes will be taken by the TA's and Instructor as to which students have not cleaned up after themselves. When there is doubt, all students who worked in a given area will be so noted. Students are encouraged to remind each other to clean up.

1.5.8 Food

Be polite, all students are being evaluated during scheduled lab periods, they do not need the distraction. Please, do not bring food to the lab during scheduled lab periods. The cafeteria is for food, the lab is for work.

Be safe, do not bring food or drinks of any kind to any table with toxic substances or electronics. Since the Aerospace Design Laboratory room has electronics on every table, no food or drink is permitted in any part of the room (SF 4003, 4102, and 4103) at any time.

Always be clean. Since the Aero-Design Laboratories (SF 4003, 4102, and 4103) do not have regular janitorial service or garbage pickup, food residue in either room will linger. Foods that are particulate (rice, nuts, sunflower seeds, popcorn etc.) or fragmentable (chips, nachos etc.) are particularly discouraged at all times because they will negatively impact lab cleanliness. In all cases, take every crumb of waste and any spills, whatever the food consumed, to the garbage containers in the hall.

Work smart. If someone in a group needs to eat, probably the rest of the group does too. If one person brings some food the other classmates are likely to become more hungry (or repulsed), particularly if it exudes a heavy odor, distracting them from work.

If students must bring food to the design area the hallway is the best place.

1.5.9 Equipment Loans

Each team may borrow an equipment kit for the duration of the term. Students may take this equipment home but must bring it to the lab. for every laboratory session. No additional equipment may be borrowed by students who fail to bring their kit. A security deposit of \$100 is required to ensure the supplies are returned in good condition. Do not loan or mix kit parts with other groups. Lost parts from a kit are charged against the deposit regardless of who lost the parts.

1.5.10 Storage of Project Items and Equipment Kits

If any items are to be left in the lab. they must be stored in the lockers in the Aerospace Design Lab (SF4102 and 4103) when students are not present working on them. Exceptions may be made in some circumstances. However, students should note that other courses also use the Aero-Design Laboratories. There are not enough lockers to assign additional lockers to groups so keep within the storage capacity afforded by your one locker.

1.5.11 Rewards of Excellence

The project demonstration days (all students of AER201 should attend) will be attended by invited guests, members of the University and any interested outsiders. Students are welcome to invite friends and relatives. Representatives from at least some of the other options will attend in addition to those from Aerospace.

Selected projects may be required for special demonstrations for camera or film crews or for special events, particularly University programs to promote Engineering and Engineering Science in high schools. It is a widely-held belief that inadequate understanding of Engineering dissuades many students from choosing a mathematical and scientific program in high school and Engineering at the University of Toronto.

Selected projects may be asked to demonstrate at various the University events. As always, the Engineering Design projects are a big favorite and spark a great deal of interest in Engineering and Engineering Science.

All of these events provide students the opportunity to show off their good design work and gain recognition. Nothing is more memorable than “the students who did the x project last year.” Pictures of AER201 projects have appeared disproportionately in all the faculty publicity publications.

Letters of recommendation will be awarded to the best students. Students will find these useful for employment applications both in departments around the University as well as in industry.

Some outstanding students of the course may be hired as summer employment to further develop the course for the future years. Some of them may also be hired as teaching assistants for the course when they become senior undergraduate (or graduate) students.

Lastly, a good mark on a student's transcript beside “AER201 - Engineering Design” can say a lot in certain circumstances - particularly for students headed into industry for PEY or after the 4th year. Interviewers often ask applicants about their design projects and theses. Applicants can often be engaged in a good, lively discussion rather than a question and answer session. Through this discussion, the interviewer will form an opinion of the applicant's abilities, particularly in communication, technical recall and application, teamwork and innovation.

1.5.12 Design Store

The Engineering Design Laboratory provides the students with a Project Kit containing a number of parts and components that are crucial for their projects for the original purchase prices, including the microcontroller development board, driver and utility boards, motor, solenoid, etc. Additionally, some extra items that could be useful for the projects can be purchased from the Design Store. No profit is made from the Project Kits or Design Store. The list of items in the Project Kit and those that can be purchased from the Design Store is mentioned in Chapter 2. Teams that plan to purchase some of the listed items must open an account with the Store and deposit \$100 in the beginning of the course. All purchases will be charged against the deposit, and the remainder will be returned to the teams at the end of the course. **No item in the Design Store can be purchased by cash.** Hence, all teams are advised to open an account with the Design Store. Design Store is open ONLY during the FIRST hour of each laboratory session.

Request for Proposal #1**The Ball-packer Machine****Need**

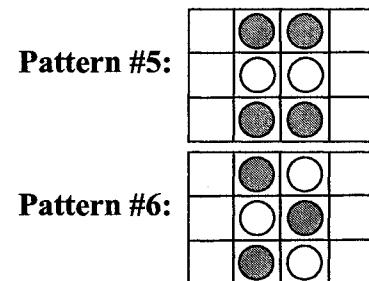
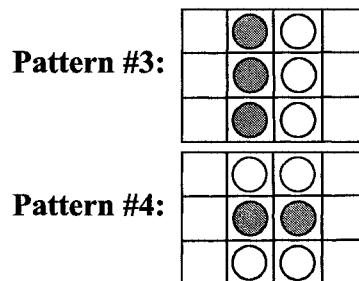
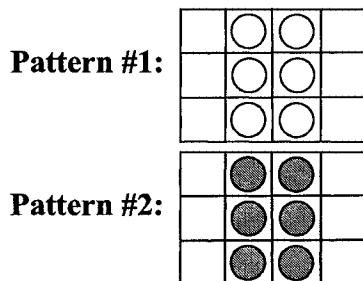
A sports company needs to package two colours of Ping-Pong balls in organizer boxes in various mixed combinations.

Goal

Design and manufacture the proof-of-concept prototype of a machine that can package white and orange Ping-Pong balls in organizer boxes in a variety of combinations reliably and quickly, according to the operator's keyed-in requests.

Specifications

The machine is expected to package 2 boxes, in each 6 Ping-Pong balls, in no longer than 2 minutes. Each ball is a standard Ping-Pong ball with $40^{\pm 0.5}$ mm in diameter and a weight of $2.70^{\pm 0.07}$ g. The balls are to be supplied in no particular format into two separate reservoirs in the machine, one for white and one for orange balls. A maximum number of 15 balls may be supplied to each reservoir, but their exact number is not known *a priori*. The boxes are made of frosted plastic with a hinged cover with two latches that snap when it is closed. The overall dimensions of each box are $182^{\pm 2}$ mm (length) by $140^{\pm 2}$ mm (width) by $42^{\pm 2}$ mm (height). The weight of each empty box is $145^{\pm 5}$ g. There are 12 compartments in the box, each with the dimensions of $45^{\pm 2}$ mm by $45^{\pm 2}$ mm by $37^{\pm 2}$ mm, but only 6 of the two middle columns are to be filled as shown in the following patterns. Samples of boxes and balls are available from the client for a closer examination. A minimum number of 3 boxes should be supplied to the machine for each operation. Methods of loading and delivering balls and boxes are up to the design, but must be convenient for the operator (e.g., easily accessible and no need for disassembling any parts.) In each box, a combination of white and orange balls must be placed in the two middle columns of each box in a pattern specified by the operator through a keypad in the beginning of the operation. Possible patterns are shown below, and there will be two different patterns for the two boxes to be packaged in each operation.



After placing all the balls in each box, it must be placed in the pickup location/bin with its cover completely closed (snapped). After each operation, the machine is expected to return to the standby mode, display a completion or termination message on the LCD, and be ready to communicate with the operator the operation information. Also, all the remaining balls must be returned in their reservoir after each operation. The information to be retrieved from the machine after each operation shall include: operation time, number of balls remaining in each reservoir, and the packaging pattern for each box. The machine shall accept operator's instructions for packaging options through a keypad. The menus displayed on the LCD must be self-explanatory and provide easy navigation for operators of various skill levels. The client requires that the

machine be portable with no need for installations, and as such there are constraints on weight and dimensions. Also for safety purposes the machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately. The machine can be plugged into the AC outlet.

Operation

The machine is normally in the standby mode. After loading the balls and boxes, the operator enters on a keypad the instructions about the packaging patterns, and then starts the operation by pressing a *<start>* button on the keypad. The entire packaging process must be done autonomously, and must take no longer than 2 minutes. Upon completion of the operation, all the remaining balls must be returned in their reservoir, and packed boxes be placed in the pickup area/bin. Then, the machine returns to the standby mode by displaying a completion or termination message on the display. The operator can then communicate with the machine through the keypad and display to retrieve the operation information.

Machine performance will be evaluated depending on the operation time, the quality of packaging (balls placed properly, covers closed), and the accuracy of the retrieved information, as detailed in the sequel.

Performance Evaluation

The prototype will run two separate but consecutive operations, and the total time, quality, and accuracy of these operations are measured. Reward and Penalty points will be given to the prototype performance according to the following scheme. Each operation is qualified for scoring if the machine delivers at least one packed and closed box (with minimum 5 balls in it, even if not placed or sorted correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.

➤ Each qualified operation	+1000
➤ Each delivered box “packaged correctly”	+500 / box
➤ Each delivered box “closed completely”	+500 / box
➤ Missing ball in the boxes	-300 / ball
➤ “Damaged” ball	-500 / ball
➤ Ball placed in the correct compartment	+200 / ball
➤ Box not delivered to the pickup location/bin	-300 / box
➤ Remaining balls not returned to their reservoir at the end of operation	-500 / reservoir
➤ The recorded pattern for the box shown on the display is incorrect	-200 / box
➤ The displayed number of remaining balls in each reservoir shown on the display is incorrect	-500 / reservoir
➤ The operation time recorded on the display is “incorrect”	-500
➤ Time penalty:	- 10 per second of operation
➤ Each disqualified operation	0

Bonus Points for Extra Features:

➤ Robustness and Durability	0 to +300
➤ Operability and Sustainability	0 to +300

➤ Elegance and Safety	0 to +300
➤ Extendibility	0 to +300
➤ Dexterity	0 to +600
➤ Compactness and Portability	+500
➤ Real-time Date/Time Display	+400
➤ Permanent Logs	+400
➤ PC Interface	+300

Constraints

- a. The entire prototype, excluding the packed boxes, pickup bin, and extended cable to the AC plug, shall completely fit within a $75 \times 75 \times 75 \text{ cm}^3$ envelope at all operation times.
- b. The weight of the machine (without the balls and boxes) shall not exceed 6 kg.
- c. The total prototype costs shall not exceed \$230CDN.
- d. The machine can be plugged in the AC, 110V-60Hz, 3-pin outlet or use its own on-board power supply during the operation.
- e. The machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately.
- f. The machine must be fully autonomous, and no interaction with an external PC or remote control is permitted during the operation. The operation must start by pressing a *<start>* button on the keypad.
- g. No installation or instrumentation is allowed in addition to what is devised within the machine.
- h. The locations for supplying balls and boxes and also the pickup location/bin must be specified in the machine clearly.
- i. Dispensing the packed boxes and retrieving the returning balls must be easy with no need for disassembling any parts of the machine.
- j. At the end of each run, the machine display must be on prompt to show the following information per operator's request: operation time, number of balls remaining in each reservoir, and the packaging pattern for each box.
- k. The machine user interface for both operation and information retrieval shall be self-explanatory, and provide easy navigation for operators of various skill levels.
- l. Each box is "packaged correctly" if all balls are placed according to the operator's selected pattern for the particular box.
- m. Each box is "closed completely" only if the cover is completely closed and at least one latch is snapped.
- n. Each box or ball (in the box, reservoir or machine) is considered as "damaged" if there are clear defects as a result of the operation, to the referee's discretion.
- o. The operation time is the duration between when the *<start>* button on the keypad is pressed and when the termination/completion message is shown on the machine LCD. The operation time shall not exceed 2 minutes. Further, the time required for entering the operator's instructions on the keypad before the operation shall not exceed 1 minute.
- p. The recorded operation time is considered "correct" if it is equal to the time measured by the referee $\pm 5\%$. Otherwise, it is assumed "incorrect."
- q. The machine's response to the operator's inquiry about the packaging statistics is considered "correct" if it is according to the outcome of the performed operation, otherwise it is considered as "incorrect."
- r. Each operation is "qualified" for scoring if the machine delivers at least one packed and closed box (with minimum 5 balls in each, even if not placed or sorted correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.

- s. Each operation is “disqualified” if the machine structurally collapses, falls over, or hangs or jams unpredictably (for more than 2 minutes) with no termination display, or damages a box, or terminates the operation before delivering at least one packed and closed box (with minimum 5 balls in each, even if not placed or sorted correctly) to the pickup location/bin, or does not display termination/completion message on the LCD at the end of operation, or the team declares the termination. If any of the above happens to the first run, the team will have 3 minutes to fix the system and run for the second time, should they wish.
- t. Each team will have a period of maximum 2 minutes to set up the machine before each run. If the preparation time exceeds 2 minutes, the run will be “disqualified.”
- u. There will be no control on the conditions of the contest environment.
- v. The machine must pose no hazard to the operator, and shall not be perceived as hazardous (e.g., too much vibration or noise or frequent sparks during the operation is perceived as dangerous.)

Extra Design Features

The following features would enhance the machine performance, and increase the Bonus Points:

- **Robustness and Durability:** Machine is durably constructed and functions consistently with a small failure frequency and under different conditions of the operation environment.
- **Operability and Sustainability:** Little time/effort is needed to set up and calibrate the machine, and the machine is modular so that parts can be replaced or repaired easily.
- **Elegance and Safety:** Machine looks elegant, and operates quietly and smoothly with little or no sensible noise or vibration.
- **Extendibility:** Machine can accept balls and boxes of different sizes and can package more than 3 boxes in each operation with little or no need for modifications.
- **Dexterity:** Machine can perform extra functions, such as labeling, box stacking, etc..
- **Compactness and Portability:** The entire prototype weighs not more than half of the maximum permitted weight and fits within a cubic envelope whose side is 75% of that of the maximum allowed envelope.
- **Real-time Date/Time Display:** Date and time of each inspection are displayed on the LCD in standby mode.
- **Permanent Logs:** Machine stores log information in permanent (EEPROM) memory.
- **PC Interface:** The operation information can be readily downloaded on a PC.

Expected Outcomes

Design and Construction Process: The team must follow a logical and systematic process in accomplishing their tasks of design, analysis, and construction. Conceptual design and system analysis are important steps of this project where the team has to compromise speed, accuracy, reliability, and cost. The detailed process must be reflected in the final report submitted by the team.

Proposal: Each team must work together to generate a proposal documentation on the design. The design proposal should reflect the conceptual design phase, team and project management with the scheduling, the steps to be taken for the detailed design and prototype fabrication, and the methods of manufacturing, integration and debugging to be followed in building the prototype.

Final Report: The final report details the entire process of detail-design, analysis, fabrication, and evaluation.

Final Prototype: The final prototype developed by the team should reflect the work presented in the proposal. Any major or significant change in the design of the prototype after submitting the proposal must be agreed upon by the client and justified in the final report. The quality of the prototype may vary widely depending on the background of the team, the difficulty of the concept, and other limitations. Many of the deficiencies of these prototypes can be resolved later in the students' academic career.

Team Dynamics: The team must propose a solution and the plan in the proposal, and remain *loyal* to the proposal during the entire process. Hence, a close interaction between members of the team is required initially to be able to "*plan ahead*." Early team dynamics may be strained, but interaction increases as the construction and integration of the machine proceed. Maximum team interaction occurs during the system integration, test and demonstration. The instructor will enhance the team dynamics by spending some time with the teams examining the progress. In many cases students remember this team experience (including their teammates) when they are seniors, or even when they are returning alumni. Professional and humane characteristics are expected in all team activities.

Grade evaluation will be heavily weighted to the generated design concepts, proposal, final report, and the way each individual/team has interacted and performed the tasks. Nevertheless, the final product and performance evaluation (competition) will remain as crucial portions in the overall grade.

Statement of Work

Each team is composed of three students. Conceptual design, system analysis, project planning, and system integration and debugging must be performed through a close interaction of all members of the team. However, for the sake of implementation, tasks can be broken into the following categories:

Processing and Control (Microcontroller)

One student shall program all the software for the system. In addition to combinatorial and sequential logic required for the operation algorithm, the keypad and display interface with the microcontroller is also part of this assignment. Some extra coding may also be needed for system debugging. Further utilization of the microcontroller may be needed if the team plans to accomplish some of the Extra Design Features, such as Real-time Date/Time Display, Permanent Logs, and PC Interface. For a low-power, high-end microcontroller, the assembly language would be the most efficient option for programming. Some cross-assemblers can translate C and/or Basic into machine code resulting in a less efficient and tractable code, to a degree that it may deteriorate system functionality. For the processing hardware, the use of the microcontroller board in the Project Kit is permitted if budget allows. Otherwise, the microcontroller student has the responsibility of assembling the microcontroller board. It is required that the microcontroller be functional for basic design features and programmable by the Reading Week, so that system integration and testing may begin right after the Reading Week. Often integration requires additional adjustments to the processor hardware and software. In addition, after the Reading week, the person responsible for the Microcontroller subsystem shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Mechanism and Actuation (Electromechanical)

One student shall be primarily responsible for constructing the structure and frames and incorporating whatever actuators and mechanisms are required in the machine. Nonetheless, after the Reading Week, the Microcontroller and Circuit members will join the Electromechanical member for completing the tasks, according to the plan specified in the proposal. Major subsystems of the Electromechanical category can include: frame and structure, ball positioning and dispensing, and box closing mechanisms. In addition to design and analysis of these subsystems, their fabrication and/or assembly as well as assigning the locations of the sensors and boards are also parts of the Electromechanical category. Some off-the-shelf mechanisms or platforms can also be used for the above-mentioned subsystems, but this must be clearly addressed in the proposal and authorized by the instructor. Although integration of the entire system might seem as a "mechanical" task by nature, all members of the team should equally and effectively take part in the integration process.

Instrumentation and Interfacing (Circuit)

One student shall construct all the digital and analog interfacing electronics to connect the sensors and actuators to the microcontroller board. This includes motor/solenoid driver circuits. All sensors and

input/output signal calibration/protection are also part of this category. In those situations where the primary calibration for a transducer is positional in nature, such as a stop switch, the task is still part of the Circuit subsystem, but consultation with the Electromechanical member is advised. For the actuator drivers, the use of driver board in the Project Kit or driver IC's is permitted if the budget allows, but the Circuit member must design and build at least one "open" circuit for a motor (DC or Stepper) in the system and prove their functionality. Ball detection and position sensing (and possible shaft encoding) could be the major sensory tasks of this subsystem, in addition to the driver circuits and cabling. The Circuit member shall also acquire suitable power supplies for the actuators, circuits, sensors, and the microcontroller. Further, after the Reading Week, the person responsible for circuits shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Discussion

In this design, speed, accuracy, reliability, and cost are competing factors. Designers should first analyze the performance criteria to specify the level of acceptable compromise in each of the above factors. A variety of solutions can be proposed for transporting the balls from the reservoir to the box, for counting the balls, and for closing the covers. Hence, a careful analysis of the force and power required for each function is important.

Students might encounter problems with manufacturing the product. With limited experience in shop practices, final prototypes may not always work as anticipated. This can be frustrating to the students. As with any life experience, product fabrication will improve as the students gain maturity, not only in shop activities, but also in the engineering science background. The contest session provides a proof of the paper design. It also demonstrates to students that in real life the result does not always follow the prediction of theory. This is a good time to remind the students that *"equations, tables and curves are only a mortal's representation of reality."*

Request for Proposal #2

The Chip-packer Machine

Need

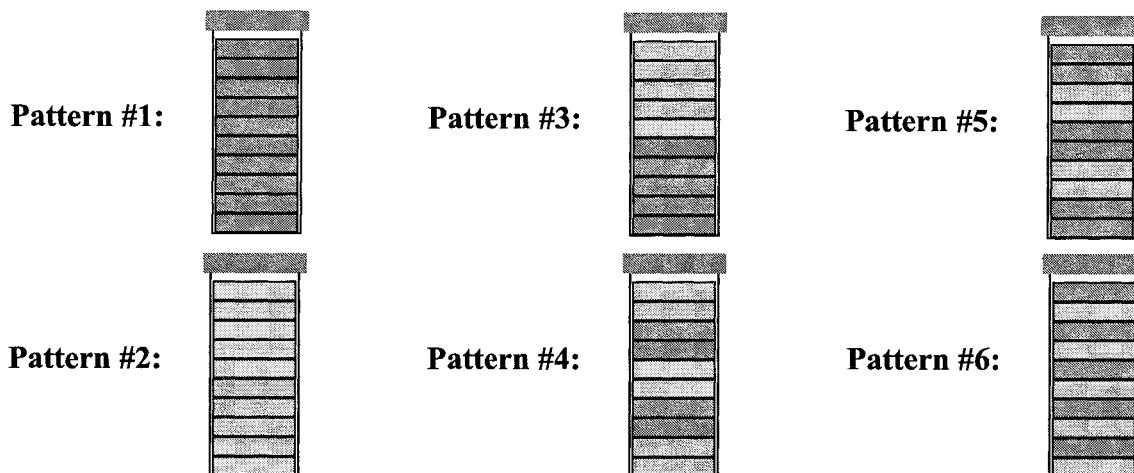
A games company needs to package two colours of Backgammon chips in cylindrical plastic containers in various mixed combinations.

Goal

Design and manufacture the proof-of-concept prototype of a machine that can package Backgammon chips in two colours, amber and camel, in cylindrical plastic containers in a variety of combinations reliably and quickly, according to the operator's keyed-in requests.

Specifications

The machine is expected to package 2 containers, in each 10 chips, in no longer than 2 minutes. Each chip is a disc made of polyester-resin plastic with $45^{\pm 1}$ mm in diameter and $10^{\pm 1}$ mm in thickness and a weight of $20^{\pm 1}$ g. The chips are to be supplied in no particular format into two separate reservoirs in the machine, one for amber and one for camel chips. A maximum number of 15 chips may be supplied to each reservoir, but their exact number is not known *a priori*. The containers are made of clear plastic with a hinged cover with a latch that snaps when it is closed. Each container is cylindrical with a height of $107^{\pm 2}$ mm (excluding the cover) and inner diameter of $52^{\pm 1}$ mm at the top and $48^{\pm 1}$ mm at the bottom. The cover is made of soft plastic, and it is roughly of a disc shape with a diameter of $63^{\pm 1}$ mm and thickness of $12^{\pm 1}$ mm. The weight of each empty container is $53^{\pm 1}$ g. Samples of containers and chips are available from the client for a closer examination. A minimum number of 3 containers should be supplied to the machine for each operation. Methods of loading and delivering chips and containers are up to the design, but must be convenient for the operator (e.g., easily accessible and no need for disassembling any parts.) In each container, a combination of amber and camel chips must be piled up horizontally in a pattern specified by the operator through a keypad in the beginning of the operation. Possible patterns are shown below, and there will be two different patterns for the two containers to be packaged in each operation.



After placing all the chips in each container, it must be placed in the pickup location/bin with its cover completely closed (snapped). After each operation, the machine is expected to return to the standby mode, display a completion or termination message on the LCD, and be ready to communicate with the operator the

operation information. Also, all the remaining chips must be returned in their reservoir after each operation. The information to be retrieved from the machine after each operation shall include: operation time, number of chips remaining in each reservoir, and the packaging pattern for each container. The machine shall accept operator's instructions for packaging options through a keypad. The menus displayed on the LCD must be self-explanatory and provide easy navigation for operators of various skill levels. The client requires that the machine be portable with no need for installations, and as such there are constraints on weight and dimensions. Also for safety purposes the machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately. The machine can be plugged into the AC outlet.

Operation

The machine is normally in the standby mode. After loading the chips and containers, the operator enters on a keypad the instructions about the packaging patterns, and then starts the operation by pressing a *<start>* button on the keypad. The entire packaging process must be done autonomously, and must take no longer than 2 minutes. Upon completion of the operation, all the remaining chips must be returned in their reservoir, and packed containers be placed in the pickup area/bin. Then, the machine returns to the standby mode by displaying a completion or termination message on the display. The operator can then communicate with the machine through the keypad and display to retrieve the operation information.

Machine performance will be evaluated depending on the operation time, the quality of packaging (chips placed properly, covers closed), and the accuracy of the retrieved information, as detailed in the sequel.

Performance Evaluation

The prototype will run two separate but consecutive operations, and the total time, quality, and accuracy of these operations are measured. Reward and Penalty points will be given to the prototype performance according to the following scheme. Each operation is qualified for scoring if the machine delivers at least one packed and closed container (with minimum 7 chips in it, even if not placed correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.

➤ Each qualified operation	+1000
➤ Each delivered container "packaged correctly"	+500 / container
➤ Each delivered container "closed completely"	+500 / container
➤ Missing chip in the containers	-300 / chip
➤ "Damaged" chip	-500 / chip
➤ Chip placed in the correct order (only if no chip is missing in the container)	+200 / chip
➤ Container not delivered to the pickup location/bin	-300 / container
➤ Remaining chips not returned to their reservoir at the end of operation	-500 / reservoir
➤ The recorded pattern for the container shown on the display is incorrect	-200 / container
➤ The displayed number of remaining chips in each reservoir shown on the display is incorrect	-500 / reservoir
➤ The operation time recorded on the display is "incorrect"	-500
➤ Time penalty:	- 10 per second of operation
➤ Each disqualified operation	0

Bonus Points for Extra Features:

➤ Robustness and Durability	0 to +300
➤ Operability and Sustainability	0 to +300
➤ Elegance and Safety	0 to +300
➤ Extendibility	0 to +300
➤ Dexterity	0 to +600
➤ Compactness and Portability	+500
➤ Real-time Date/Time Display	+400
➤ Permanent Logs	+400
➤ PC Interface	+300

Constraints

- a. The entire prototype, excluding the packed containers, pickup bin, and extended cable to the AC plug, shall completely fit within a $50 \times 50 \times 50 \text{ cm}^3$ envelope at all operation times.
- b. The weight of the machine (without the chips and containers) shall not exceed 6 kg.
- c. The total prototype costs shall not exceed \$230CDN.
- d. The machine can be plugged in the AC, 110V-60Hz, 3-pin outlet or use its own on-board power supply during the operation.
- e. The machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately.
- f. The machine must be fully autonomous, and no interaction with an external PC or remote control is permitted during the operation. The operation must start by pressing a *<start>* button on the keypad.
- g. No installation or instrumentation is allowed in addition to what is devised within the machine.
- h. The locations for supplying chips and containers and also the pickup location/bin must be specified in the machine clearly.
- i. Dispensing the packed containers and retrieving the returning chips must be easy with no need for disassembling any parts of the machine.
- j. At the end of each run, the machine display must be on prompt to show the following information per operator's request: operation time, number of chips remaining in each reservoir, and the packaging pattern for each container.
- k. The machine user interface for both operation and information retrieval shall be self-explanatory, and provide easy navigation for operators of various skill levels.
- l. Each container is "packaged correctly" if all chips are placed according to the operator's selected pattern for the particular container.
- m. Each container is "closed completely" only if the cover is completely closed and snapped.
- n. Each container or chip (in the container, reservoir or machine) is considered as "damaged" if there are clear defects as a result of the operation, to the referee's discretion.
- o. The operation time is the duration between when the *<start>* button on the keypad is pressed and when the termination/completion message is shown on the machine LCD. The operation time shall not exceed 2 minutes. Further, the time required for entering the operator's instructions on the keypad before the operation shall not exceed 1 minute.
- p. The recorded operation time is considered "correct" if it is equal to the time measured by the referee \pm 5%. Otherwise, it is assumed "incorrect."
- q. The machine's response to the operator's inquiry about the packaging statistics is considered "correct" if it is according to the outcome of the performed operation, otherwise it is considered as "incorrect."

- r. Each operation is “qualified” for scoring if the machine delivers at least one packed and closed container (with minimum 7 chips in each, even if not placed correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.
- s. Each operation is “disqualified” if the machine structurally collapses, falls over, or hangs or jams unpredictably (for more than 2 minutes) with no termination display, or damages a container, terminates the operation before delivering at least one packed and closed container (with minimum 7 chips in each, even if not placed correctly) to the pickup location/bin, or does not display termination/completion message on the LCD at the end of operation, or the team declares the termination. If any of the above happens to the first run, the team will have 3 minutes to fix the system and run for the second time, should they wish.
- t. Each team will have a period of maximum 2 minutes to set up the machine before each run. If the preparation time exceeds 2 minutes, the run will be “disqualified.”
- u. There will be no control on the conditions of the contest environment.
- v. The machine must pose no hazard to the operator, and shall not be perceived as hazardous (e.g., too much vibration or noise or frequent sparks during the operation is perceived as dangerous.)

Extra Design Features

The following features would enhance the machine performance, and increase the Bonus Points:

- **Robustness and Durability:** Machine is durably constructed and functions consistently with a small failure frequency and under different conditions of the operation environment.
- **Operability and Sustainability:** Little time/effort is needed to set up and calibrate the machine, and the machine is modular so that parts can be replaced or repaired easily.
- **Elegance and Safety:** Machine looks elegant, and operates quietly and smoothly with little or no sensible noise or vibration.
- **Extendibility:** Machine can accept chips and containers of different sizes and can package more than 3 containers in each operation with little or no need for modifications.
- **Dexterity:** Machine can perform extra functions, such as labeling, container stacking, etc.
- **Compactness and Portability:** The entire prototype weighs not more than half of the maximum permitted weight and fits within a cubic envelope whose side is 75% of that of the maximum allowed envelope.
- **Real-time Date/Time Display:** Date and time of each inspection are displayed on the LCD in standby mode.
- **Permanent Logs:** Machine stores log information in permanent (EEPROM) memory.
- **PC Interface:** The operation information can be readily downloaded on a PC.

Expected Outcomes

Design and Construction Process: The team must follow a logical and systematic process in accomplishing their tasks of design, analysis, and construction. Conceptual design and system analysis are important steps of this project where the team has to compromise speed, accuracy, reliability, and cost. The detailed process must be reflected in the final report submitted by the team.

Proposal: Each team must work together to generate a proposal documentation on the design. The design proposal should reflect the conceptual design phase, team and project management with the scheduling, the steps to be taken for the detailed design and prototype fabrication, and the methods of manufacturing, integration and debugging to be followed in building the prototype.

Final Report: The final report details the entire process of detail-design, analysis, fabrication, and evaluation.

Final Prototype: The final prototype developed by the team should reflect the work presented in the proposal. Any major or significant change in the design of the prototype after submitting the proposal must be agreed upon by the client and justified in the final report. The quality of the prototype may vary widely depending on the background of the team, the difficulty of the concept, and other limitations. Many of the deficiencies of these prototypes can be resolved later in the students' academic career.

Team Dynamics: The team must propose a solution and the plan in the proposal, and remain *loyal* to the proposal during the entire process. Hence, a close interaction between members of the team is required initially to be able to "*plan ahead*." Early team dynamics may be strained, but interaction increases as the construction and integration of the machine proceed. Maximum team interaction occurs during the system integration, test and demonstration. The instructor will enhance the team dynamics by spending some time with the teams examining the progress. In many cases students remember this team experience (including their teammates) when they are seniors, or even when they are returning alumni. Professional and humane characteristics are expected in all team activities.

Grade evaluation will be heavily weighted to the generated design concepts, proposal, final report, and the way each individual/team has interacted and performed the tasks. Nevertheless, the final product and performance evaluation (competition) will remain as crucial portions in the overall grade.

Statement of Work

Each team is composed of three students. Conceptual design, system analysis, project planning, and system integration and debugging must be performed through a close interaction of all members of the team. However, for the sake of implementation, tasks can be broken into the following categories:

Processing and Control (Microcontroller)

One student shall program all the software for the system. In addition to combinatorial and sequential logic required for the operation algorithm, the keypad and display interface with the microcontroller is also part of this assignment. Some extra coding may also be needed for system debugging. Further utilization of the microcontroller may be needed if the team plans to accomplish some of the Extra Design Features, such as Real-time Date/Time Display, Permanent Logs, and PC Interface. For a low-power, high-end microcontroller, the assembly language would be the most efficient option for programming. Some cross-assemblers can translate C and/or Basic into machine code resulting in a less efficient and tractable code, to a degree that it may deteriorate system functionality. For the processing hardware, the use of the microcontroller board in the Project Kit is permitted if budget allows. Otherwise, the microcontroller student has the responsibility of assembling the microcontroller board. It is required that the microcontroller be functional for basic design features and programmable by the Reading Week, so that system integration and testing may begin right after the Reading Week. Often integration requires additional adjustments to the processor hardware and software. In addition, after the Reading week, the person responsible for the Microcontroller subsystem shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Mechanism and Actuation (Electromechanical)

One student shall be primarily responsible for constructing the structure and frames and incorporating whatever actuators and mechanisms are required in the machine. Nonetheless, after the Reading Week, the Microcontroller and Circuit members will join the Electromechanical member for completing the tasks, according to the plan specified in the proposal. Major subsystems of the Electromechanical category can include: frame and structure, chip positioning and dispensing, and container closing mechanisms. In addition to design and analysis of these subsystems, their fabrication and/or assembly as well as assigning the locations of the sensors and boards are also parts of the Electromechanical category. Some off-the-shelf mechanisms or platforms can also be used for the above-mentioned subsystems, but this must be clearly addressed in the proposal and authorized by the instructor. Although integration of the entire system might seem as a

“mechanical” task by nature, all members of the team should equally and effectively take part in the integration process.

Instrumentation and Interfacing (Circuit)

One student shall construct all the digital and analog interfacing electronics to connect the sensors and actuators to the microcontroller board. This includes motor/solenoid driver circuits. All sensors and input/output signal calibration/protection are also part of this category. In those situations where the primary calibration for a transducer is positional in nature, such as a stop switch, the task is still part of the Circuit subsystem, but consultation with the Electromechanical member is advised. For the actuator drivers, the use of driver board in the Project Kit or driver IC's is permitted if the budget allows, but the Circuit member must design and build at least one “open” circuit for a motor (DC or Stepper) in the system and prove their functionality. Chip detection and position sensing (and possible shaft encoding) could be the major sensory tasks of this subsystem, in addition to the driver circuits and cabling. The Circuit member shall also acquire suitable power supplies for the actuators, circuits, sensors, and the microcontroller. Further, after the Reading Week, the person responsible for circuits shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Discussion

In this design, speed, accuracy, reliability, and cost are competing factors. Designers should first analyze the performance criteria to specify the level of acceptable compromise in each of the above factors. A variety of solutions can be proposed for transporting the chips from the reservoir to the container, for counting the chips, and for closing the covers. Hence, a careful analysis of the force and power required for each function is important.

Students might encounter problems with manufacturing the product. With limited experience in shop practices, final prototypes may not always work as anticipated. This can be frustrating to the students. As with any life experience, product fabrication will improve as the students gain maturity, not only in shop activities, but also in the engineering science background. The contest session provides a proof of the paper design. It also demonstrates to students that in real life the result does not always follow the prediction of theory. This is a good time to remind the students that *“an ounce of application is worth a ton of abstraction.”*

Request for Proposal #3

The Dowel-packer Machine

Need

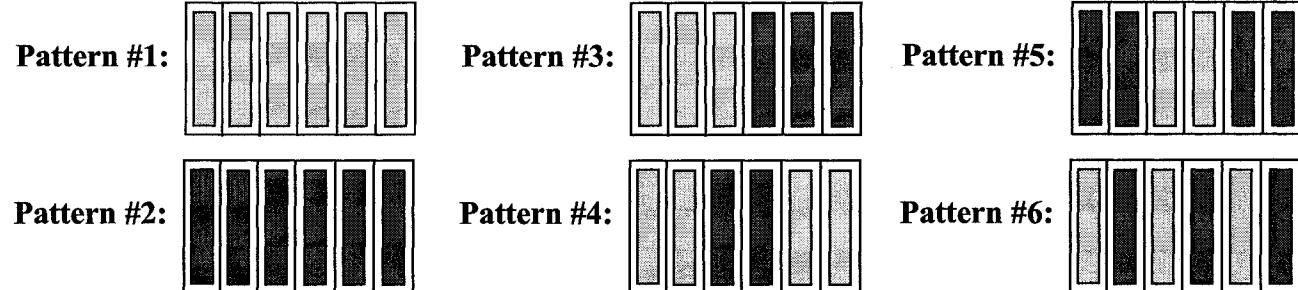
A utility company needs to package two colours of wood dowels in organizer boxes in various mixed combinations.

Goal

Design and manufacture the proof-of-concept prototype of a machine that can package light and dark wood dowels in organizer boxes in a variety of combinations reliably and quickly, according to the operator's keyed-in requests.

Specifications

The machine is expected to package 2 boxes, in each 6 wood dowels, in no longer than 2 minutes. Each dowel is cylindrical with a diameter of $25.5^{\pm 0.5}$ mm, a height of $105^{\pm 2}$ mm and a weight of $25^{\pm 2}$ g. The dowels are to be supplied in no particular format into two separate reservoirs in the machine, one for light and one for dark dowels. A maximum number of 12 dowels may be supplied to each reservoir, but their exact number is not known *a priori*. The boxes are made of frosted plastic with a hinged cover with a latch in the middle that snaps when it is closed. The overall dimensions of each box are $209^{\pm 2}$ mm (length) by $121^{\pm 2}$ mm (width) by $35^{\pm 2}$ mm (height). There are 6 compartments in the box, each with the dimensions of $115^{\pm 2}$ mm by $32^{\pm 2}$ mm by $30^{\pm 2}$ mm. The weight of each empty box is $115^{\pm 5}$ g. Samples of boxes and dowels are available from the client for a closer examination. A minimum number of 3 boxes should be supplied to the machine for each operation. Methods of loading and delivering dowels and boxes are up to the design, but must be convenient for the operator (e.g., easily accessible and no need for disassembling any parts.) In each box, a combination of light and dark dowels must be placed in the compartments in a pattern specified by the operator through a keypad in the beginning of the operation. Possible patterns are shown below, and there will be two different patterns for the two boxes to be packaged in each operation.



After placing all the dowels in each box, it must be placed in the pickup location/bin with its cover completely closed (snapped). After each operation, the machine is expected to return to the standby mode, display a completion or termination message on the LCD, and be ready to communicate with the operator the operation information. Also, all the remaining dowels must be returned in their reservoir after each operation. The information to be retrieved from the machine after each operation shall include: operation time, number of dowels remaining in each reservoir, and the packaging pattern for each box. The machine shall accept operator's instructions for packaging options through a keypad. The menus displayed on the LCD must be self-explanatory and provide easy navigation for operators of various skill levels. The client requires that the

machine be portable with no need for installations, and as such there are constraints on weight and dimensions. Also for safety purposes the machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately. The machine can be plugged into the AC outlet.

Operation

The machine is normally in the standby mode. After loading the dowels and boxes, the operator enters on a keypad the instructions about the packaging patterns, and then starts the operation by pressing a *<start>* button on the keypad. The entire packaging process must be done autonomously, and must take no longer than 2 minutes. Upon completion of the operation, all the remaining dowels must be returned in their reservoir, and packed boxes be placed in the pickup area/bin. Then, the machine returns to the standby mode by displaying a completion or termination message on the display. The operator can then communicate with the machine through the keypad and display to retrieve the operation information.

Machine performance will be evaluated depending on the operation time, the quality of packaging (dowels placed properly, covers closed), and the accuracy of the retrieved information, as detailed in the sequel.

Performance Evaluation

The prototype will run two separate but consecutive operations, and the total time, quality, and accuracy of these operations are measured. Reward and Penalty points will be given to the prototype performance according to the following scheme. Each operation is qualified for scoring if the machine delivers at least one packed and closed box (with minimum 5 dowels in it, even if not placed or sorted correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.

- | | |
|--|------------------------------|
| ➤ Each qualified operation | +1000 |
| ➤ Each delivered box “packaged correctly” | +500 / box |
| ➤ Each delivered box “closed completely” | +500 / box |
| ➤ Missing dowel in the boxes | -300 / dowel |
| ➤ “Damaged” dowel | -500 / dowel |
| ➤ Dowel placed in the correct compartment | +200 / dowel |
| ➤ Box not delivered to the pickup location/bin | -300 / box |
| ➤ Remaining dowels not returned to their reservoir at the end of operation | -500 / reservoir |
| ➤ The recorded pattern for the box shown on the display is incorrect | -200 / box |
| ➤ The displayed number of remaining dowels in each reservoir shown on the display is incorrect | -500 / reservoir |
| ➤ The operation time recorded on the display is “incorrect” | -500 |
| ➤ Time penalty: | - 10 per second of operation |
| ➤ Each disqualified operation | 0 |

Bonus Points for Extra Features:

- | | |
|----------------------------------|-----------|
| ➤ Robustness and Durability | 0 to +300 |
| ➤ Operability and Sustainability | 0 to +300 |

➤ Elegance and Safety	0 to +300
➤ Extendibility	0 to +300
➤ Dexterity	0 to +600
✗ ➤ Compactness and Portability	+500
✗ ➤ Real-time Date/Time Display	+400
✗ ➤ Permanent Logs	+400
✗ ➤ PC Interface	+300

Constraints

- a. The entire prototype, excluding the packed boxes, pickup bin, and extended cable to the AC plug, shall completely fit within a $75 \times 75 \times 75$ cm³ envelope at all operation times.
- b. The weight of the machine (without the dowels and boxes) shall not exceed 6 kg.
- c. The total prototype costs shall not exceed \$230CDN.
- d. The machine can be plugged in the AC, 110V-60Hz, 3-pin outlet or use its own on-board power supply during the operation.
- e. The machine must have an easily-accessible emergency STOP switch that stops all the mechanical moving parts immediately.
- f. The machine must be fully autonomous, and no interaction with an external PC or remote control is permitted during the operation. The operation must start by pressing a *<start>* button on the keypad.
- g. No installation or instrumentation is allowed in addition to what is devised within the machine.
- h. The locations for supplying dowels and boxes and also the pickup location/bin must be specified in the machine clearly.
- i. Dispensing the packed boxes and retrieving the returning dowels must be easy with no need for disassembling any parts of the machine.
- j. At the end of each run, the machine display must be on prompt to show the following information per operator's request: operation time, number of dowels remaining in each reservoir, and the packaging pattern for each box.
- k. The machine user interface for both operation and information retrieval shall be self-explanatory, and provide easy navigation for operators of various skill levels.
- l. Each box is "packaged correctly" if all dowels are placed according to the operator's selected pattern for the particular box.
- m. Each box is "closed completely" only if the cover is completely closed and the latch is snapped.
- n. Each box or dowel (in the box, reservoir or machine) is considered as "damaged" if there are clear defects as a result of the operation, to the referee's discretion.
- o. The operation time is the duration between when the *<start>* button on the keypad is pressed and when the termination/completion message is shown on the machine LCD. The operation time shall not exceed 2 minutes. Further, the time required for entering the operator's instructions on the keypad before the operation shall not exceed 1 minute.
- p. The recorded operation time is considered "correct" if it is equal to the time measured by the referee \pm 5%. Otherwise, it is assumed "incorrect."
- q. The machine's response to the operator's inquiry about the packaging statistics is considered "correct" if it is according to the outcome of the performed operation, otherwise it is considered as "incorrect."
- r. Each operation is "qualified" for scoring if the machine delivers at least one packed and closed box (with minimum 5 dowels in each, even if not placed or sorted correctly) to the pickup location/bin within 2 minutes, returns to the standby mode by displaying the completion or termination message at the end of its operation, and prompts for the packaging information.

- s. Each operation is “disqualified” if the machine structurally collapses, falls over, or hangs or jams unpredictably (for more than 2 minutes) with no termination display, or damages a box, or terminates the operation before delivering at least one packed and closed box (with minimum 5 dowels in each, even if not placed or sorted correctly) to the pickup location/bin, or does not display termination/completion message on the LCD at the end of operation, or the team declares the termination. If any of the above happens to the first run, the team will have 3 minutes to fix the system and run for the second time, should they wish.
- t. Each team will have a period of maximum 2 minutes to set up the machine before each run. If the preparation time exceeds 2 minutes, the run will be “disqualified.”
- u. There will be no control on the conditions of the contest environment.
- v. The machine must pose no hazard to the operator, and shall not be perceived as hazardous (e.g., too much vibration or noise or frequent sparks during the operation is perceived as dangerous.)

Extra Design Features

The following features would enhance the machine performance, and increase the Bonus Points:

- **Robustness and Durability:** Machine is durably constructed and functions consistently with a small failure frequency and under different conditions of the operation environment.
- **Operability and Sustainability:** Little time/effort is needed to set up and calibrate the machine, and the machine is modular so that parts can be replaced or repaired easily.
- **Elegance and Safety:** Machine looks elegant, and operates quietly and smoothly with little or no sensible noise or vibration.
- **Extendibility:** Machine can accept dowels and boxes of different sizes and can package more than 3 boxes in each operation with little or no need for modifications.
- **Dexterity:** Machine can perform extra functions, such as labeling, box stacking, etc.
- **Compactness and Portability:** The entire prototype weighs not more than half of the maximum permitted weight and fits within a cubic envelope whose side is 75% of that of the maximum allowed envelope.
- **Real-time Date/Time Display:** Date and time of each inspection are displayed on the LCD in standby mode.
- **Permanent Logs:** Machine stores log information in permanent (EEPROM) memory.
- **PC Interface:** The operation information can be readily downloaded on a PC.

Expected Outcomes

Design and Construction Process: The team must follow a logical and systematic process in accomplishing their tasks of design, analysis, and construction. Conceptual design and system analysis are important steps of this project where the team has to compromise speed, accuracy, reliability, and cost. The detailed process must be reflected in the final report submitted by the team.

Proposal: Each team must work together to generate a proposal documentation on the design. The design proposal should reflect the conceptual design phase, team and project management with the scheduling, the steps to be taken for the detailed design and prototype fabrication, and the methods of manufacturing, integration and debugging to be followed in building the prototype.

Final Report: The final report details the entire process of detail-design, analysis, fabrication, and evaluation.

Final Prototype: The final prototype developed by the team should reflect the work presented in the proposal. Any major or significant change in the design of the prototype after submitting the proposal must be agreed upon by the client and justified in the final report. The quality of the prototype may vary widely depending on the background of the team, the difficulty of the concept, and other limitations. Many of the deficiencies of these prototypes can be resolved later in the students' academic career.

Team Dynamics: The team must propose a solution and the plan in the proposal, and remain *loyal* to the proposal during the entire process. Hence, a close interaction between members of the team is required initially to be able to "*plan ahead*." Early team dynamics may be strained, but interaction increases as the construction and integration of the machine proceed. Maximum team interaction occurs during the system integration, test and demonstration. The instructor will enhance the team dynamics by spending some time with the teams examining the progress. In many cases students remember this team experience (including their teammates) when they are seniors, or even when they are returning alumni. Professional and humane characteristics are expected in all team activities.

Grade evaluation will be heavily weighted to the generated design concepts, proposal, final report, and the way each individual/team has interacted and performed the tasks. Nevertheless, the final product and performance evaluation (competition) will remain as crucial portions in the overall grade.

Statement of Work

Each team is composed of three students. Conceptual design, system analysis, project planning, and system integration and debugging must be performed through a close interaction of all members of the team. However, for the sake of implementation, tasks can be broken into the following categories:

Processing and Control (Microcontroller)

One student shall program all the software for the system. In addition to combinatorial and sequential logic required for the operation algorithm, the keypad and display interface with the microcontroller is also part of this assignment. Some extra coding may also be needed for system debugging. Further utilization of the microcontroller may be needed if the team plans to accomplish some of the Extra Design Features, such as Real-time Date/Time Display, Permanent Logs, and PC Interface. For a low-power, high-end microcontroller, the assembly language would be the most efficient option for programming. Some cross-assemblers can translate C and/or Basic into machine code resulting in a less efficient and tractable code, to a degree that it may deteriorate system functionality. For the processing hardware, the use of the microcontroller board in the Project Kit is permitted if budget allows. Otherwise, the microcontroller student has the responsibility of assembling the microcontroller board. It is required that the microcontroller be functional for basic design features and programmable by the Reading Week, so that system integration and testing may begin right after the Reading Week. Often integration requires additional adjustments to the processor hardware and software. In addition, after the Reading week, the person responsible for the Microcontroller subsystem shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Mechanism and Actuation (Electromechanical)

One student shall be primarily responsible for constructing the structure and frames and incorporating whatever actuators and mechanisms are required in the machine. Nonetheless, after the Reading Week, the Microcontroller and Circuit members will join the Electromechanical member for completing the tasks, according to the plan specified in the proposal. Major subsystems of the Electromechanical category can include: frame and structure, dowel positioning and dispensing, and box closing mechanisms. In addition to design and analysis of these subsystems, their fabrication and/or assembly as well as assigning the locations of the sensors and boards are also parts of the Electromechanical category. Some off-the-shelf mechanisms or platforms can also be used for the above-mentioned subsystems, but this must be clearly addressed in the proposal and authorized by the instructor. Although integration of the entire system might seem as a "mechanical" task by nature, all members of the team should equally and effectively take part in the integration process.

Instrumentation and Interfacing (Circuit)

One student shall construct all the digital and analog interfacing electronics to connect the sensors and actuators to the microcontroller board. This includes motor/solenoid driver circuits. All sensors and input/output signal calibration/protection are also part of this category. In those situations where the primary calibration for a transducer is positional in nature, such as a stop switch, the task is still part of the Circuit subsystem, but consultation with the Electromechanical member is advised. For the actuator drivers, the use of driver board in the Project Kit or driver IC's is permitted if the budget allows, but the Circuit member must design and build at least one "open" circuit for a motor (DC or Stepper) in the system and prove their functionality. Dowel detection and position sensing (and possible shaft encoding) could be the major sensory tasks of this subsystem, in addition to the driver circuits and cabling. The Circuit member shall also acquire suitable power supplies for the actuators, circuits, sensors, and the microcontroller. Further, after the Reading Week, the person responsible for circuits shall effectively assist the Electromechanical subsystem with duplication or fabrication of components and subassemblies. **The division of Electromechanical tasks among all members must be specified clearly in the project proposal.**

Discussion

In this design, speed, accuracy, reliability, and cost are competing factors. Designers should first analyze the performance criteria to specify the level of acceptable compromise in each of the above factors. A variety of solutions can be proposed for transporting the dowels from the reservoir to the box, for counting the dowels, and for closing the covers. Hence, a careful analysis of the force and power required for each function is important.

Students might encounter problems with manufacturing the product. With limited experience in shop practices, final prototypes may not always work as anticipated. This can be frustrating to the students. As with any life experience, product fabrication will improve as the students gain maturity, not only in shop activities, but also in the engineering science background. The contest session provides a proof of the paper design. It also demonstrates to students that in real life the result does not always follow the prediction of theory. This is a good time to remind the students that "***what we have to learn to do, we learn by doing.***"

Chapter 2

Equipment for Engineering Design

2.1	Design Kit Parts List	1
2.2	Initial Checkout	2
2.3	Returning the Equipment Package	2
2.4	Using the Equipment Package.....	3
2.4.1	Power Supply.....	3
2.4.2	Solderless Protoboard.....	4
2.4.2.1	“Proper Way” to Use a Protoboard	4
2.4.2.2	The Best Way to Use a Protoboard.....	5
2.4.2.3	Physical Limitations.....	6
2.4.3	Anti-Static Wrist Strap	6
2.4.3.1	Safety	6
2.4.3.2	Choosing a Ground	7
2.4.4	Prototyping Tools	7
2.4.5	Digital Volt Meter - Multimeter	8
2.4.5.1	Measuring Voltage.....	8
2.4.5.2	Measuring Resistance	9
2.4.5.3	Measuring Current	9
2.4.6	Logic Probe	10
2.4.7	Soldering Equipment	10
2.4.7.1	Soldering Iron and Solder	10
2.4.7.2	Soldering Iron Controller	12
2.4.7.3	Soldering Stand and Sponge	13
2.4.7.4	Desoldering Pump.....	14
2.4.7.5	Other Soldering Tools.....	14
2.4.8	Electronics Tools	14
2.4.9	Hand Tools	15
2.4.10	Storage Tote and Bag.....	16
2.5	Returning the Kit.....	16
2.6	Project Kit	17
2.7	Design Store	18
2.8	Laboratory Oscilloscope.....	20
2.8.1	What is an oscilloscope?.....	20
2.8.2	Setting the Main Controls.....	20
2.8.3	Other Controls	22
2.9	Laboratory Function Generator	23

2.1 Design Kit Parts List

The following is a list of the items in the Design Kit, and their replacement cost if items are lost or damaged. These costs will be charged against the kit deposit.

Qty	Item	Cost	Comments
1	Power Supply	\$70	Lost (\$30 for defective)
1	Power Cord for (Some) Power Supplies (if applicable)	\$10	Lost or Damaged
1	Anti-Static Wrist Strap	\$15	Lost or Damaged
1	Digital Volt Meter (DVM, Multimeter)	\$50	Lost or Damaged
2	Probe Leads (red and black)	\$10	Lost or Damaged
2	Mini-Clip DVM Leads (red and black)	\$3	Each Lead
1	Logic Probe	\$50	Lost or Damaged
1	Iron Controller	\$30	Lost (\$15 for defective)
1	Soldering Iron	\$15	Lost or Damaged
1	Soldering Stand	\$10	Lost or Damaged
1	Desoldering Pump	\$15	Lost or Damaged
1	Side Cutters	\$6	Lost or Severely Damaged
1	Pliers	\$6	Lost or Damaged
1	Wire Strippers	\$6	Lost or Damaged
1	6-piece Precision Screwdriver Set	\$6	lost, damaged, or items missing
1	Black or White Toolbox	\$5	Lost or Damaged
1	Precision Tweezers	\$6	Lost or Damaged
1	Chip Puller	\$6	Lost or Damaged
1	Power Bar	\$10	Lost or Damaged
1	Kit Storage Tote, and Lid	\$30	Lost, Damaged, or Cracked
2	Squares of Guard Plywood	\$2 each	Lost or Damaged
1	Mastercraft 3/8" Drill	\$60	Lost or Damaged
1	Drill Chuck Key (if applicable)	\$5	Excludes keyless drill design kits
1	Third-hand Tool with Magnifying Glass	\$15	Lost or Damaged
1	Metal Hack Saw (without blade)	\$25	Lost or Damaged
1	Mini Hacksaw (with one blade)	\$5	Lost or Damaged
1	Allen Key Set	\$5	Lost or Damaged
1	Screwdriver Set Bits with Handle	\$15	Lost or Damaged or items missing
1	Measuring Tape	\$5	Lost or Damaged
1	Level	\$5	Lost or Damaged
1	Hammer	\$5	Lost or Damaged
1	Utility Knife	\$5	Lost or Damaged
1	Scissors	\$5	Lost or Damaged
2	5" C-Clamp	\$5 each	Lost or Damaged
2	2" C-Clamp	\$5 each	Lost or Damaged
1	Adjustable Wrench	\$6	Lost or Damaged
1	Caliper	\$5	Lost or Damaged
1	Metal File	\$10	Lost or Damaged
1	Belt Clamp	\$10	Lost or Damaged
1	12" Handsaw	\$15	Lost or Damaged
1	Rotary Tool Kit	\$50	Lost or Damaged
1	Rotary Tool Kit Wrench	\$5	Lost or Damaged

2.2 Initial Checkout

After receiving the package, check that all items are in the bag and the numbering of each item corresponds to the bag number.

AS SOON AS YOU RECEIVE THE LAB. KIT, DO THE FOLLOWING CHECKS

- a) Assemble the soldering iron stand, then plug the iron into the soldering iron controller.
- b) Connect the controller to a 110V outlet and set the dial to the middle position, this setting should always be used for the iron - do not operate the iron at a higher voltage.
- c) Wet the sponge (in your Project Kit) with water (always use a wet sponge for wiping the tip before soldering).
- d) Cut a 10cm length of a piece of wire or jumper. Strip 1cm from one end of each piece using the wire strippers with the screw set to the correct wire gauge. Only cut copper wire < 0.75mm with the side cutters or the wire strippers - NEVER cut steel!!
- e) Tin each stripped length of wire and solder (Design Store) the two pieces together. If students have problems, they should ask for advice. Do not leave the lab until you have done this satisfactorily - the iron tip may be faulty.
- f) Plug in the power supply and check the output voltages with the multimeter.
- g) Make sure the multimeter measures the voltage correctly by connecting it (in the Voltage mode) to the leads of the power supply.
- h) Switch the multimeter to the Ω (resistance) setting, and confirm the meter reads zero when the probe tips are shorted together.
- i) The multimeter requires special fuses; always use the correct rating otherwise the instrument will be permanently damaged. NEVER measure **CURRENT** using this meter unless you are positive you are doing it correctly. Measuring current improperly will blow the meter's fuse, may destroy the meter, or may cause a fire or explosion. Nearly all devices used in the lab are voltage devices or are used in circuits where the current should be inferred by Ohm's law.

Students must bring the equipment package with them each time they come to the laboratory, no equipment may be borrowed if they forget their kit.

Resistors are \$0.02 each at Active Surplus. Students will be charged ten times this rate to make shopping trips for resistors worthwhile for the instructor, if not for the students.

To discourage people from 'losing' their power supply, a basic power supply (+5V@4A, +12V@1A, -12V@0.4A) will become costly (\$70) if lost or damaged. An AC-DC adapter with 3 voltages is also available in the Design Store with a reasonable price for using in the final prototype.

2.3 Returning the Equipment Package

Lay out all items so they can be checked for damage. Plug in the power supply and demonstrate that it is in good working order with the kit multimeter. All equipment must be inspected before deposit money is returned. Replacement multimeters not manufactured by Elenco, Kelvin, Mastech, or RSR will not be accepted, due to calibration issues.

2.4 Using the Equipment Package

A brief description is provided of how each component of the kit is used. The photograph below shows a complete kit.



2.4.1 Power Supply

The Engineering Design power supplies come in two types, the galvanized tissue-box sided ones with no fan or a recovered PC computer power supplies that includes a fan and a standard PC power cord. Each is capable of producing at a minimum: (Note: Some of the non-PC power supplies cannot provide the -5 V output.)

+5 V	+12 V	-12 V
3.5 A	1.4 A	0.3 A

The purpose of the power supply is to power circuits under test on the solderless protoboard for AER 201. If its output capabilities are sufficient to power other components of the AER 201 project, then it may be used for those components; however, damage will be charged against the kit deposit. It is quite feasible the kit power supply may not be able to drive

some motors in a project. The +12V output is not made to charge batteries or run motors - it is made to run a few chips.

When using the power supply for noise susceptible circuitry such as analog amplifiers, please note they are switching power supplies, and hence carry a high frequency signal on their outputs, typically of 40 kHz. Always have sufficient capacitance on power supply rails near the circuits in question to attenuate this noise; a 47 μ F tantalum capacitor will work well for low current circuits. Noise coupled into IC power supplies often appears on the circuit outputs.

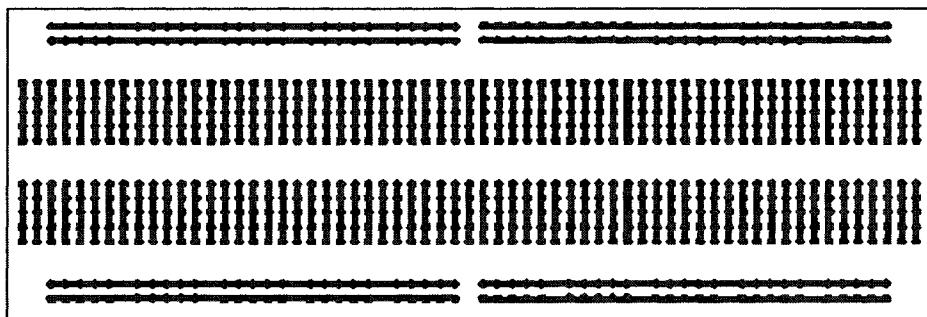
When shorted, the non-PC supplies will make a high frequency whine and generally the LED will go out or blink. If this sound is heard, please turn off the supply immediately and check the circuitry for the short. Please do not use the sound the supply makes as a diagnostic tool to find shorts - it hurts the supply and a multimeter is the correct diagnostic tool.

The behavior of the PC supplies when shorted is, in general, not known and could vary greatly from one supply to the next. Some supplies will make no noise; the fan simply will not run when it is turned on. Also, please keep the fan clear of bits of wire, plastic bags and other debris.

2.4.2 Solderless Protoboard

Each Project kit contains solderless protoboard for prototyping IC circuitry for the AER 201 design project. Figure below gives the connection pattern of a solderless protoboard. Note that the power rails along the top and bottom of the board are frequently broken along the mid-line.

This standard protoboard is designed for creating circuits for non-surface mount discrete components and integrated circuits (ICs) in Dual In-line Packages (DIP) with a 0.3 or 0.6 inch pin-row separation. The protoboard is designed so that ICs straddle the mid-line and each pin connects to a column 5 connection points. Resistors, capacitors, jumper wires and other components bridge from one column to another or to the power rails. Where a component is too large to connect at a connection point, a wire should be soldered to the component and to a small U that will hold in two adjacent connection points at once (two connection points of the same potential). Where an IC cannot be found in DIP packaging, a DIP-shaped component carrier can be used, with the component (carefully) soldered to the pins on the top of the carrier.

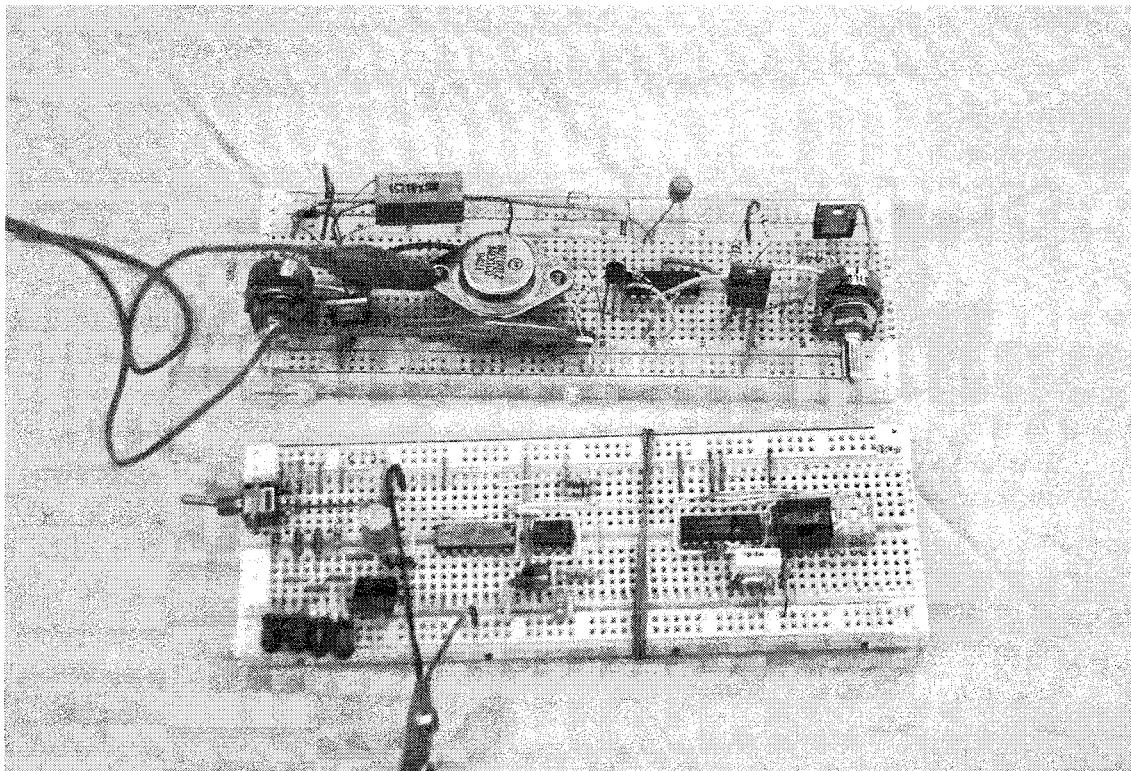


2.4.2.1 "Proper Way" to Use a Protoboard

Examples of the proper way to create a reliable protoboard circuit, and the typical way students create protoboard circuits are shown in the photograph below. In the circuit board demonstrating the proper way, wires and components lay close to the protoboard surface, their lengths trimmed before inserting. The connections are made with proper pre-tinned protoboard wire which is of a larger gauge than typical telephone wire. No stranded wire is used in the protoboard in the example of correct protoboard usage. With this method of construction, a gold-contact protoboard, in zero-vibration environment, can be almost as reliable as a soldered circuit. The time required to create such circuit boards and the expense of proper jumpers and a high quality protoboard is usually beyond the resources of a 2nd year student in a 13 week course. The "proper" way of prototyping would precede first production runs of an item that will one day be mass produced.

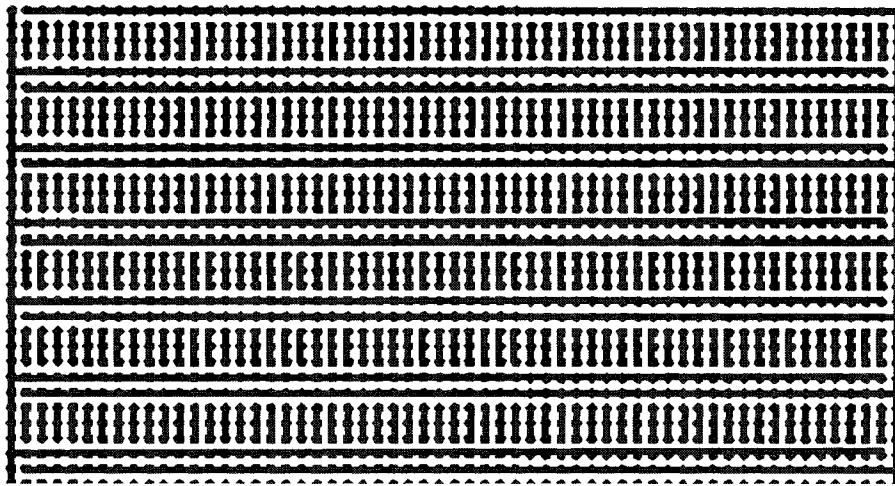
2.4.2.2 The Best Way to Use a Protoboard

Second year students typically use un-tinned telephone wire or wire-wrap wire due to its availability. Students neither pre-measure their wires, nor trim the leads of their components to make them lay against the surface of the board. This allows students to get one working prototype on a lab bench very rapidly. Students should expect noise to be a greater factor because longer leads allow more noise coupling. Mechanically, the circuit is not very robust at all. Since the board is used only for prototyping, with some care in transportation, even a not-so-neat protoboard will provide service for the week or so until the circuit design is finalized.



Students have attempted to use protoboard as final board in projects. There is a critical mass of circuitry, on a sloppily constructed protoboard, after which more problems are created with every problem diagnosed. Typically, these problems result from disconnecting something, accidentally shorting two component leads, or changing the geometry (and therefore noise coupling) in the 3d maze that is the cloud of jumpers, wires and components hovering above the board. While these circuits may be made to work once or twice, once handed over to partners and integrated into other hardware, they will surely suffer constant breakdowns.

To complete the rapid development, the circuits are copied and duplicated on a solder-protoboard (connection pattern shown below) to make them final. In this scenario, students buy a number of solder-protobards sufficient to build their project's circuitry instead of high quality solderless protobards. The cost of a solder-protoboard is about \$4 where high quality solderless-protoboard strips are \$15 for an equivalent area. Leads and wires are sized on the solder board by pulling them through the other side and cutting them off. The student method is both faster and cheaper but care must be taken in testing the prototype circuit and in duplicating the circuit on solder board. Provided it is not physically crushed, a soldered board is quite resistant to problems with handling and vibration common in student book-bags and robot projects. The protoboard provided with the Project kit is provided as a prototyping area, **NOT** for final circuits.



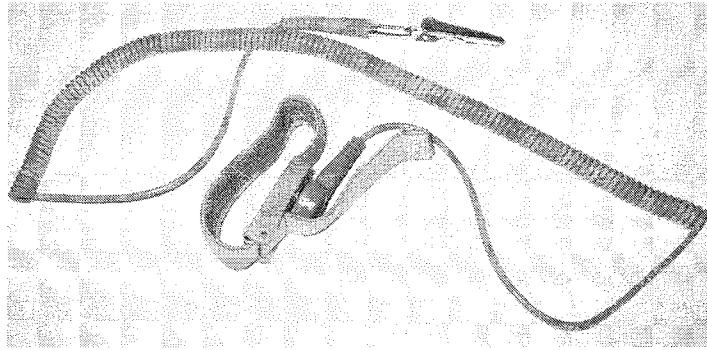
2.4.2.3 Physical Limitations

The connection points in the protoboard can typically take 0.03 inch diameter leads as a maximum. In the cross-board direction, each connection point will accept wider leads. For packages with wide flat leads, such as the TO-220, typical for power transistors and regulators, each lead may be twisted 90 degrees to fit more easily into the protoboard.

Protoboard connecting columns will conduct between 0.4 to 1 Amp. Care should be taken at these current drains, particularly along the supply rails. Voltage drops along the length of the connectors will occur and this can be quite significant for analog circuitry and noise coupling.

2.4.3 Anti-Static Wrist Strap

The anti-static wrist strap is used to control static when working at a workbench and protect integrated circuits and transistor devices, particularly MOSFET and CMOS devices.

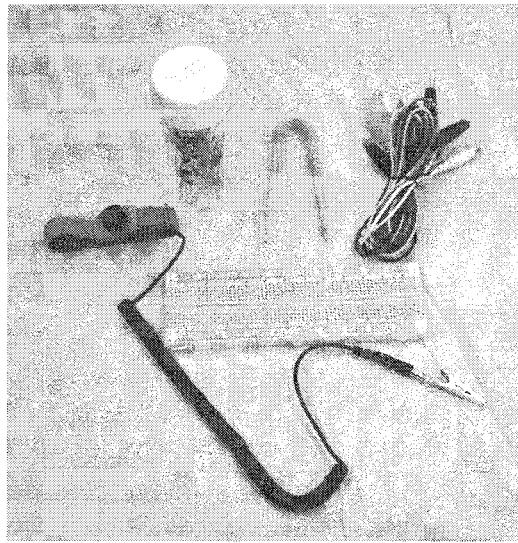


2.4.3.1 Safety

Do not wear the wrist strap in the presence of rotating machinery or appliances such as fans, blenders, mixers, dryers, drill presses, lathes, gear trains, chain-driven sprockets (as on bicycles) or vehicles. As with any loose, dangling clothing or long hair, the cord may become entrained in the equipment and pull a person into the machinery. Even large robot-motors may have sufficient torque to pose a danger - keep the off-switch very accessible. Do not wear the wrist strap in the presence of electrical equipment with exposed heating coils such as toasters - the clip end could fall into the slot for the bread. Since the strap is connected firmly, both physically and electrically, to a person's arm the electrocution hazard is extreme.

2.4.3.2 Choosing a Ground

The ground clip should always be attached to whatever circuitry is being worked on and, if possible, to a good ground. A good ground can be found on the metal case of anything that has a 3-prong plug (except on some shoe-box sized kit power supplies), on metal electrical face plates, and metal water fixtures like taps and pipes. If students plan to do electronic work in their residence, they should give some thought to grounding before choosing a work place. In the Aerospace Design Lab, all of the wire-frame shelves holding the equipment are ground, as well as the ground connection on the front of the oscilloscope. For anti-static procedures at times away from the work bench, please see Chapter 5 on circuits.

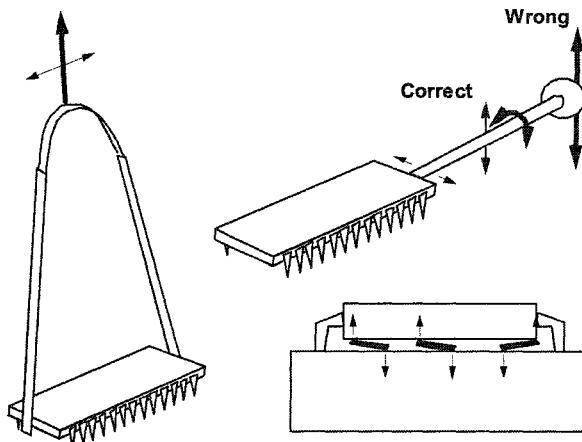


2.4.4 Prototyping Tools

The photograph above shows the prototyping equipment contained in the kit: the protoboard, IC extractor, anti-static wrist strap, alligator leads..

Each kit contains a set of 10 alligator leads. These leads are not intended to be consumed in the project however is it not important whether the original alligator leads are returned, just the proper number and colors.

Below is a diagram showing how to remove chips using either an IC puller or a small slot screwdriver. **Note:** the heavy arrows indicate the dominant motion. For the chip puller, the IC is pulled upward while gently rocking fore and aft. For the screwdriver, the chip is not levered out by a pushing up or down on the end of the screwdriver - this risks flipping the chip out suddenly, bending the leads on the last pins to leave the socket. The correct method is to pry the chip up by twisting the flat blade of the screwdriver under the chip at various locations. This will remove the entire chip most of the way from the socket.



2.4.5 Digital Volt Meter - Multimeter



The multimeters (DVM) supplied with the equipment kit will be the most important piece of diagnostic equipment for the design group. The ranges available on a typical kit multimeter are visible in the photograph above.

The multimeters in the kits vary from one to the next but they all have the ability to measure: DC Volts from milliVolts to 200 Volts, DC Amps from μ Amps to 10 Amps, AC Volts to 200 Volts RMS, Resistance from 1 to 1M Ohm and a continuity checker. Some will have a diode checker, frequency meter, frequency generator, capacitor checker, or transistor gain checker. When faced with the need to measure something and calculate other things to get complete data on a situation, students should favor measurements in the order: Voltage, Resistance, and Current. **Only measure current as a last resort.**

Each kit comes with two sets of leads, probe leads for probing a number of connections by hand, and mini-clip leads for clipping on one connection and monitoring one measurement, hands free. The mini-clip leads are far more useful in AER 201Y.

2.4.5.1 Measuring Voltage

Important Safety Tip: Never attempt to measure high voltage (> 200 Volts) with any hand held meter in AER 201Y. Not only is it likely to destroy the meter, it is also likely to provide a nasty zap in the hand if the voltage is a little higher than anticipated and more than the meter can take.

Place the red lead in the Volts (V) terminal and the black lead in the common (COM) terminal of the meter. If the group has been measuring current (on some meters this is a separate plug), *please* make sure the next user checks that the red lead is in the correct plug or damage to the meter or circuit may result. Place the black probe or clip on the ground rail of the circuit. Use the red probe or clip to measure voltages in the circuit. Use the dial to select the appropriate range. Voltage can be measured without powering off or disassembling the circuit. Voltage measurement is the easiest to do and should be used as the primary measurement over any other measurements. It is for this reason the multimeters are more frequently called DVMs or digital volt meters.

When using a multimeter to measure an AC voltage, note the meter is guaranteed to give a correct answer only for sinusoidal, steady signals within certain frequency ranges. In all other cases of time varying signals, a voltmeter on AC will give some reading that may or may not properly describe the RMS voltage of the signal.

One of the two most common errors in using a multimeter is using a multimeter exclusively in circuit design and diagnosis. When measuring voltages, the meter will always give a stable DC reading even when probing high frequency, oscillating signals. Frequently a student will build a circuit with some logic, regulators or transistors in it and get very unusual DC readings because the circuit is in fact oscillating. Usually, switching to the AC range will give a non-zero

reading - the first clue a multimeter is not the right tool. When a multimeter indicates an otherwise simple circuit is not working and giving strange readings, use an oscilloscope.

2.4.5.2 Measuring Resistance

Important Safety Tip: Never, never measure resistance on an element that is part of a powered circuit. Not only is it impossible to get any valid reading (no, Ohm meters do not give differential resistance about operating points), it is almost certain to permanently damage the meter. Never attempt to directly measure the "output resistance" of a battery, function generator, op-amp or any other source of current or signal.

Power off the circuit and remove the element from the circuit by disconnecting one of its two leads. Place the red lead in the Ohms (Ω) terminal and the black lead in the common (COM) terminal of the meter. Place one probe on one lead and the other probe on the other. Use the dial to select the appropriate range.

Be particularly cautious not to touch the leads of an inductor while testing it. The Ohm meter is supplying current and voltage to determine resistance. An inductor will attempt to continue driving that same current after the meter is detached. Meters will supply a maximum of 0.2 to 2.6V, depending on the range and the component under test. If the resistance of the coil is 30Ω and the skin resistance of a person is $30-50\text{ k}\Omega$, a person touching the leads as contact is broken will experience a shock of more than 1000 times (use Ohm's Law!) the voltage output by the meter - in a short pulse. This shock will frequently be on the order of hundreds of volts and can be sufficient to destabilize the heart. The safe way to test inductors for resistance is to short them with an alligator lead while the meter is attached and detached. Since the resistance of an alligator lead is very low, the voltages created by the inductor trying to drive the current will be very low. Please use caution with all inductors.

2.4.5.3 Measuring Current

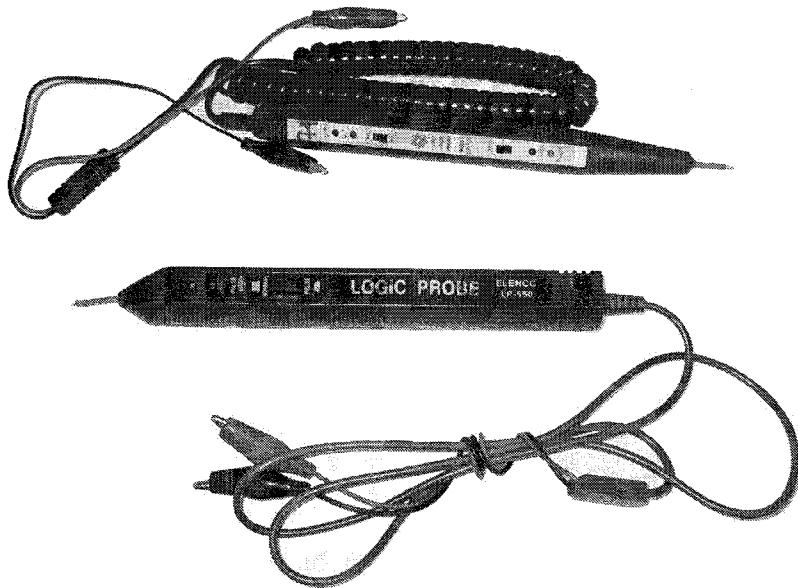
Important Safety Tip: Never measure current without thinking carefully about what is being attempted! Of 38 multimeters that went out last year in perfect working order, 5 were *destroyed* (at great cost to the student) and another 30 had blown fuses, all from incorrectly measuring current.

- Ammeters measure currents in a *wire in a circuit*.
- There must be a circuit or current cannot be measured.

Be sure the current to be measured is less than the maximum the meter can measure, typically 10 Amps. If the group is using batteries, especially Gel-Cells or NiCads, they can supply far in excess of 10 Amps if the load is of suitably low resistance. Power off the circuit and disconnect the wire in which the current is to be measured. Place the red lead in the 10 Amps (10A) terminal and the black lead in the common (COM) terminal of the meter. Select the 10 Amp scale. Place one probe where the wire came from and the other where it went to. Power on the circuit and read the current. If the current is sufficiently small ($< 200\text{mA}$), switch the lead to the Amps (A) terminal and select the appropriate range. Use the dial to select the appropriate range.

The other common error in using a multimeter is to attempt to measure the current of a battery. Batteries have fixed currents only as a result of putting them in circuits. Ammeters have nearly zero resistance. If a piece of wire were connected across a battery, it would melt its plastic quickly and possibly catch fire. If an ammeter were connected across the terminals of a battery, in some lame attempt to measure current, it is equivalent to shorting the battery with a wire. In the moment that the battery current jumps to 20 or more Amps, even for a D cell, the fuse of the multimeter is vaporized, depositing a silver-gray coating on the inside of the fuse glass. In such high current situations, it is common for semiconductor circuits to be damaged before the wire of the fuse has time to heat and break the contact. In short, the fuse may not protect the multimeter. On the 10 Amp setting, there is no fuse to protect the meter, so it will certainly be destroyed.

2.4.6 Logic Probe



A typical kit logic probe is shown in the above photograph along with a multimeter. The logic probe is at the same time both the most useful and the most useless, troublesome item in the kit. It is essentially a multimeter without the capability to give a quantitative answer. All it can do is say a logic signal is high or low, or if it changes state. It is invaluable for quick answers on computer I/O pins during integration, when double checking interface circuits or software algorithms. It is hand held and has an audible output.

A logic probe costs \$50 and it is easily lost or damaged. In cases where the circuitry is really not up to spec and is not outputting digital signals as it should, a logic probe can mislead students. Its capability to detect state changes is fine but tells the user little about the pattern of the changes, usually an important factor. In analog circuits, where most of the real challenges are, logic probes do not aid development or diagnosis.

2.4.7 Soldering Equipment

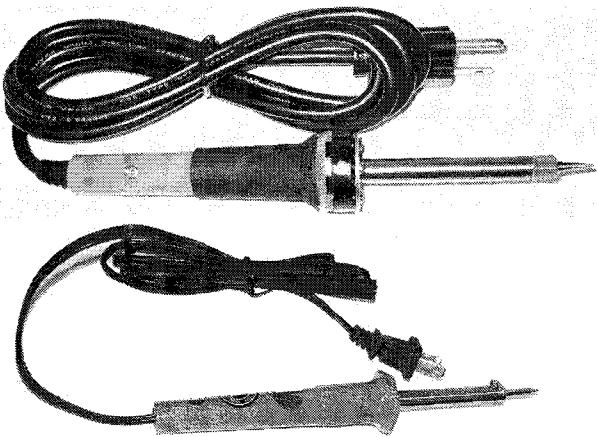
Each kit has a soldering iron, stand, sponge, controller, and a solder sucker. Some kits also include 4 soldering tools - knives, picks, a brush and a heat sink. The soldering equipment is shown in the photograph below:

2.4.7.1 Soldering Iron and Solder

Important Safety Tip: Make sure the soldering iron is cool before putting it in the plastic storage tote. (The plastic of the storage tote probably smells really bad when it gets hot again.) Please do not immerse the iron in water or snow to cool it - it risks damaging the heating element.

- The soldering iron is used as the heat source for soldering.
- Note the soldering iron is not the source of solder.
- Never apply solder to the iron to carry the solder to a connection.

Solder is an alloy of lead and tin. Soldering is a process that has developed over many years. It is essentially an adhesive process, and like any adhesive, solder requires surface preparation. Fluxes of various sorts are used to lift small amounts of surface oxidation from the materials to be soldered. In the case of heavy oxidation, the surface may require preparation

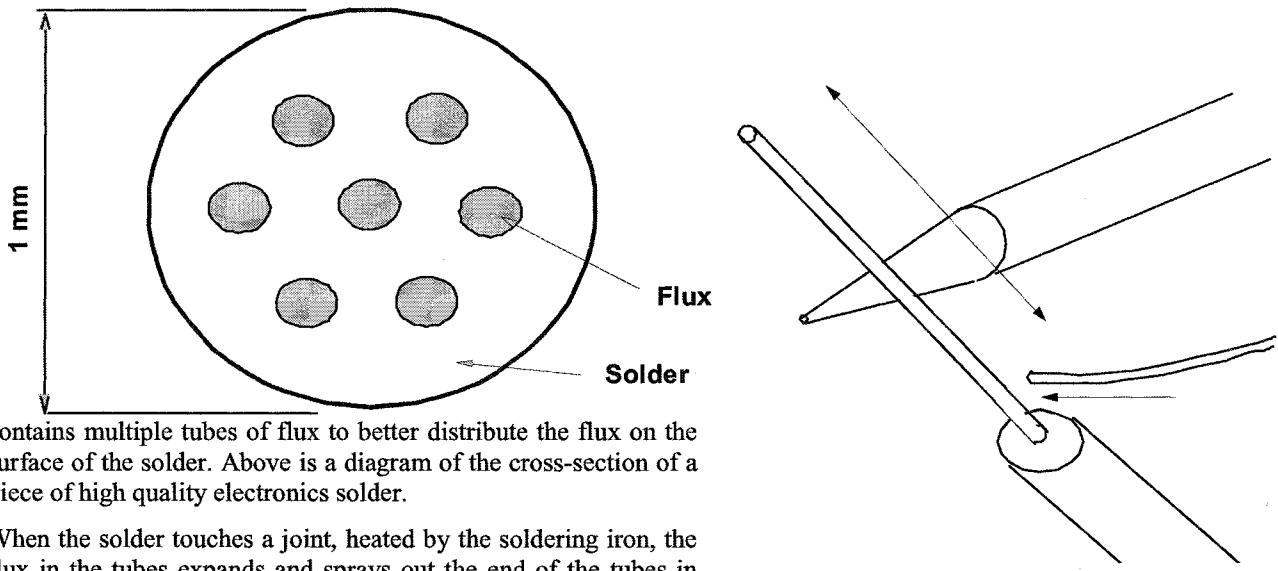


with a wire brush or sand paper. The surface is heated and flux and solder are applied. The flux cleans the surface and the solder spreads to cover it. The freshly soldered surfaces can then be joined by bringing them together and heating them.

If a good job is done, the bond is very strong both electrically and mechanically. The process of soldering is used in plumbing, stained glass construction, and electrical and electronic work. For convenience, most types of solder contain flux in voids or tubes in the solder. The following table gives some idea of the types of solder and their uses:

Type	% Tin	% Lead	Flux Core	Format	Use
50/50	50	50	none	Bar or Wire	Plumbing, Hobby
50/50	50	50	Acid	2.5mm Wire	Plumbing
50/50	50	50	Resin	2.5mm Wire	Plumbing, Hobby
60/40	60	40	Single Resin	1mm Wire	Electronics
63/37	63	37	Multiple Resin	1mm Wire	Electronics

Each type of solder has its application. Only the last two are acceptable for electrical work. Acid flux core solder should be particularly avoided because the flux residue left on boards damages the components. High quality electronics solder



contains multiple tubes of flux to better distribute the flux on the surface of the solder. Above is a diagram of the cross-section of a piece of high quality electronics solder.

When the solder touches a joint, heated by the soldering iron, the flux in the tubes expands and sprays out the end of the tubes in advance of the solder melting. The flux runs and covers the part to be soldered, just in advance of the solder.

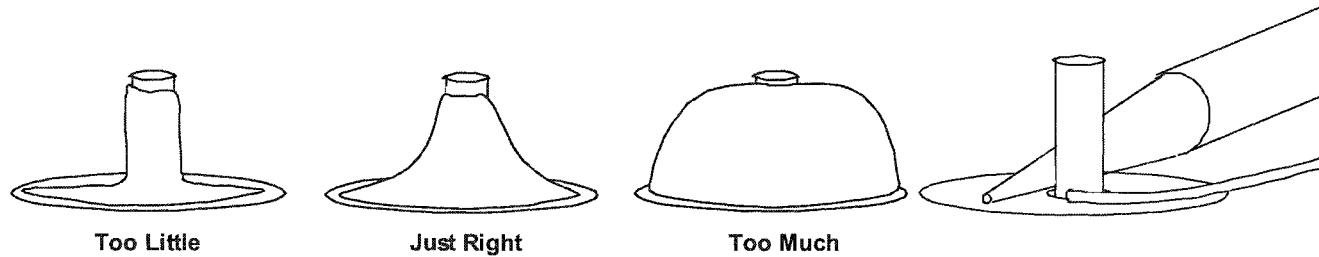
Applying solder is a two step process: heat and apply solder. The process of making a connection is a two-step process. Solder is applied to each part individually. This is the process called tinning. Once tinned, the two parts are brought

together and heat is re-applied to fuse them. Some additional solder may be required, especially for thick solid wires and connectors. The purpose of tinning is to make sure solder covers all surfaces that will later be connected. The proper geometry for tinning a wire for use in electronics is shown in figure right.

Some parts, such as gold plated connectors and sockets do not require tinning because there is no oxide on the surface and solder will flow freely.

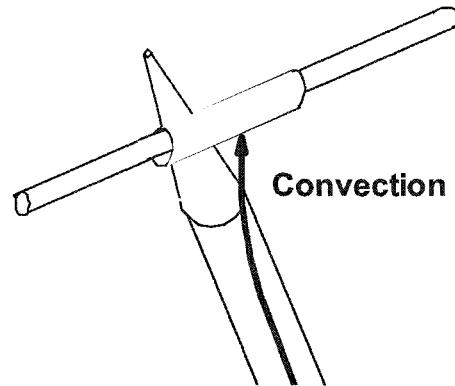
The following figure shows the proper geometry for making a circuit board connection and the correct appearance of a finished connection. Note the iron touches both the pin and the pad on the circuit board and the solder is applied in at the juncture of the two, on the side opposite the iron.

In soldering, it is useful to have a tiny drop of solder near the tip of the soldering iron. This small puddle of solder changes shape when the iron is held against a contact, increasing the contact surface area and rate of heat conduction. If

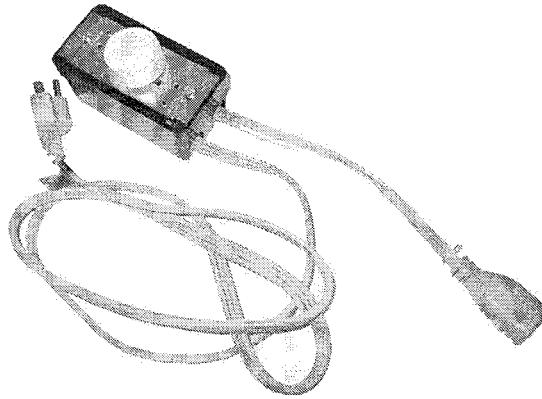


the right amount of solder is added to the connection, the quantity of solder on the tip will not change drastically but may need to be added to or wiped off after many connections if there is not enough or too much.

A soldering iron may also be used to shrink heat shrink wrap. Note in the figure right, the iron does not touch the heat shrink. The process relies on holding the iron upwards so that hot air is convected along to the tip and rises to heat the shrink wrap.



2.4.7.2 Soldering Iron Controller



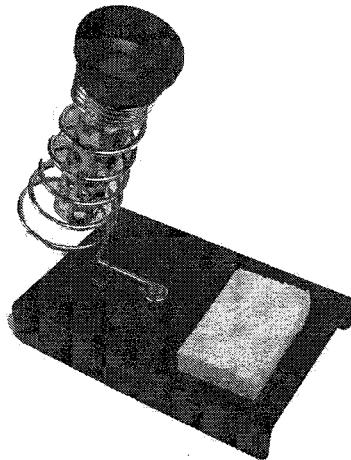
This essentially takes the cheap, kit soldering irons and turns them into very useful tools for fine soldering work. By adjusting the voltage to the iron, the temperature of the iron is controlled. Normally, cheap hobby-store irons run very hot. For those who are skilled at soldering, working on jobs where high temperature is not a problem, one can compensate in soldering style for a soldering iron that is a bit too hot. A slightly hotter soldering iron is better for larger wire and component leads that sink heat away from the connection faster. If a component has to be removed, requiring prolonged heating of several connections, or if the operator of the iron is having trouble making a particular connection, a hot soldering iron can be a serious problem. The added heat can melt insulation, damage components, lift traces from circuit boards, and oxidize the solder. A hot soldering iron can permanently damage the circuit board. The worst problem with a

hot soldering iron is its tendency to oxidize the tip of the iron, burning away the coating on the tip. It is the surface coating on the tip that makes the solder melt. Coating the tip with solder protects the underlying copper heat conductor.

If the soldering iron is too cold, it will have to be held against the joint longer to raise its temperature to the point where good soldering can be done. Below a certain point, the iron will have to be held for a very long time. All the while, heat is conducting along the component leads and into elements. A faster, hotter soldering contact leaves less total heat in the connection than a slow, simmering iron.

Obviously, the secret is the correct balance and the soldering controller provides the ability to adjust. When soldering heavy connectors, a high temperature may be selected. When soldering tiny wires to the prongs of a surface mount chip, a cooler iron may be used. Each soldering controller is marked at 60V, a value proven over a number of years to be the best for the type of soldering irons in the Engineering Design Lab. A higher setting may be selected to heat the iron up quickly (1 minute) and dialed down to 60V to stabilize at the proper temperature. To protect the tips, please do not use the Design Lab. soldering irons at any settings above 60V. If a student wishes to use their personal soldering iron at a higher voltage, as the instructor does, they are free to do so.

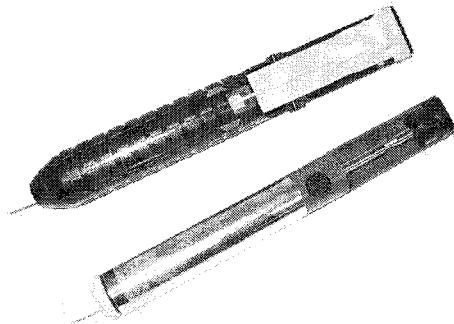
2.4.7.3 Soldering Stand and Sponge



Important Safety Tip: Please always use the soldering stand. Soldering irons are round and roll nicely. They invariably roll into some plastic that melts to the hot barrel and stinks for weeks after. Occasionally soldering irons roll into something combustible, or onto a delicate circuit, or they fall into a partner's lap or burn the lab tables.

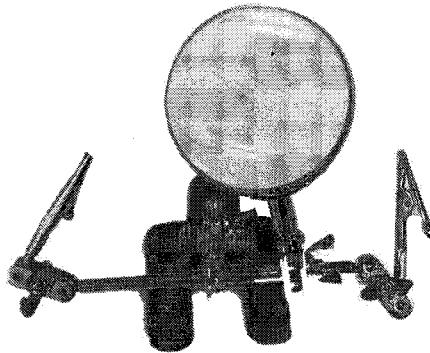
The solder sponge is used to clean the tip of the soldering iron. The sponge should be soaked and then the dripping excess water should be gently shaken from it. It should not drip but should be very wet. When the solder on the tip of the soldering iron becomes oxidized, wipe it on the sponge and re-apply a small amount of solder. If the iron needs to be "whited" on the sponge too often, it is a sign the iron is too hot. Please do not use the sponge to cool the iron; adjust the soldering controller down instead.

2.4.7.4 Desoldering Pump



The desoldering pump is a kit item students may never need to use. However it is the best method of removing solder from connections that must be separated. To use the pump properly, prime the pump by depressing the plunger. With the soldering iron, heat the connection from one side. When the connection is hot and the solder is liquid, place the tip of the pump on the side opposite the iron and remove the iron. Press the trigger on the pump. If the switch process is done quickly, the solder will be sucked away leaving the connection tinned but otherwise dry. On some connections, the process may have to be repeated. When de-soldering a pin on a circuit board, the pump may have to be placed over the connection to suck solder from inside the hole in the board. Once the solder is removed, the contacts may have to be pried slightly to break the last, small solder bond areas.

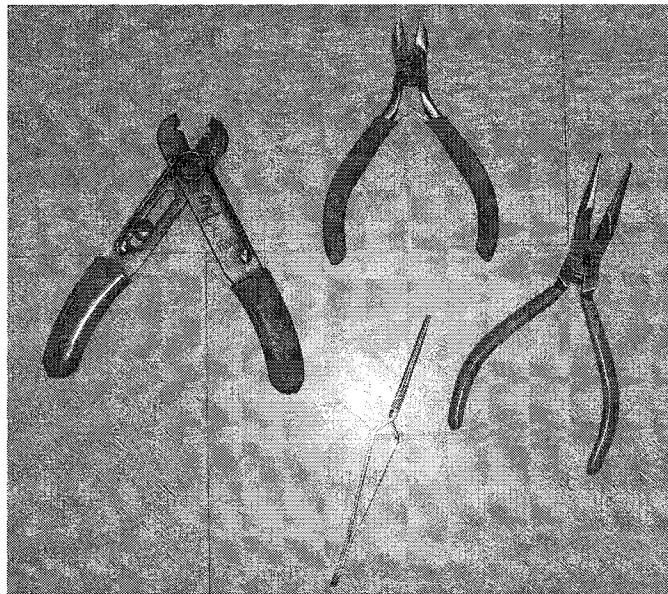
2.4.7.5 Other Soldering Tools



These tools are unavailable in the kits, but students might find some commercially and make use of them. Sharp picks, blades and awls are used to clean between connections to make sure there are no solder bridges. The smallest of the screwdrivers from the set in the kit is also ideal for this job. A small wire brush and an aluminum heat sink might also prove useful. The heat sink is like a small pair of pliers that grabs a lead between the component and the solder joint and absorbs the heat traveling up the lead wire. It is especially useful when soldering metal can op-amp leads to the tops of sockets. The needle-nosed pliers from the kit will do, with an elastic band around the handles to make them self-closing. Hemostats and medical clamps, available at Active Surplus, are also quite handy for this and other fine work in electronics. Solder wicks provide another method of removing solder from connections. The wick consists of a braided copper wire, which is placed over the connection to be desoldered. The soldering iron is pressed down onto the wick, and the solder flows up into it, where it solidifies, leaving the connection merely tinned.

2.4.8 Electronics Tools

Each kit contains side cutters, pliers, wire strippers, tweezers and a set of fine screwdrivers. These are the tools most used in electronics work. Care must be taken with the side cutters. They are designed to cut soft copper wire and that is all. Cutting anything else will only damage the blades. The tips of the screwdrivers are very fine and, like all screwdrivers, made of hard, brittle metal. It is quite possible to break the corners off of the tips if they are misused on heavier jobs. The kit tools are shown in the above photograph.



2.4.9 Hand Tools

The hand tools provided in the kit are shown above, and include a metal and mini hacksaw, wood saw, caliper, belt tightener, file, screwdriver set, measuring tape, level, and a set of Allen keys (also known as hex keys). The metal hacksaw can be used to cut short and thin materials. Generally, the blade of the hacksaw is designed for cutting when it is pushed. Hence, more force should be used when pushing the hacksaw forward, than on the return stroke. Make sure the piece is clamped securely in a vice, or with hand-clamps. The mini-hacksaw can be used to cut similar to the metal hacksaw. It is ideal for cutting circuit boards. Allen keys are often used in machine, or socket screws. There are two



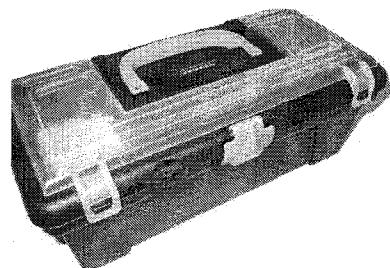
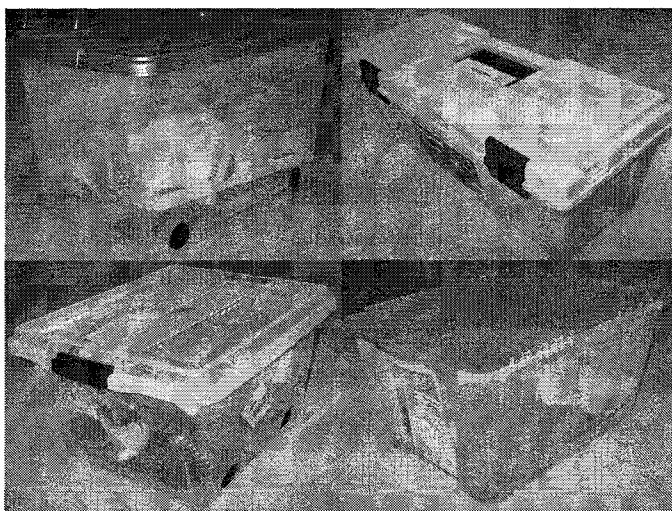
types of Allen key systems – metric and imperial. The set provided is standard imperial, measured in inches, and should not be used on metric screwheads. This is because although the fit will seem close, the small difference will quickly strip the head of the screw, or destroy the Allen key. The wood saw is good for cutting through wood quickly. Start by pulling back a few times so that a groove has been made before providing a forward and back motion. The belt tightener can be used to quickly tighten bolts. Simply place the bolt between the belt and the plastic arm and then tighten and twist. The file is used to smooth rough edges of pieces of metal or wood.

2.4.10 Storage Tote and Bag

Important Safety Tip: The storage tote is made from a plastic that will likely become very brittle in cold weather. Please do not drop, kick or throw the tote at any time and if it is cold, expect it to smash into a litter of small pieces.

All kits contain a cloth bag. Students may use this as they see fit during the term but must return it intact. The bag would be useful if a student wants to take only some of the kit parts home for the night. Be aware that the bag does not protect the items in the kit from damage to the same extent that the totes do. Be especially careful of the power switches on the tissue-box sized Design Lab. power supplies. If the kit is dropped, they can be easily broken and will short when the supply is next plugged in. Be weary of the straps on the cloth bags – they tend to break if too much is carried.

These kits also contain a small clear plastic parts box subdivided into cells. If the group's kit does not have one or another is desired for convenience, a fishing tackle organizer is recommended. Such items are available at large suppliers of sporting goods such as Canadian Tire and LeBaron's. (The instructor's favorite is LeBaron's "Model 3598 Phantom Tackle Mate" for \$5.72. It has 15 "worm-proof" compartments in $6.5 \times 4.25 \times 2.25$ inches! "StowAway" sells a slightly more compact case for \$1.85 Code: 3449-7)



2.5 Returning the Kit

If a team's kit is returned complete they will receive their entire kit deposit back. Students should buy replacements for lost or damaged items before returning the kit because they will get back much more kit money than they spend on the replacements. The rates charged for damaged items reflect the non-sale cost, plus tax, times an inconvenience factor that makes it worthwhile for students to shop for replacements. If 70 kit holders spend 1 hour shopping to complete their kit at the end of term, this will save a week's work for the instructor. For kits with so many damaged or missing parts the charges exceed the deposit, the kit holder will be required to return what is left of the kit and pay the shortfall. It is up to the kit holder to pressure partners and classmates who lost or damaged the parts of the kit to reimburse the kit holder. It is a good idea never to loan out parts from the kit to classmates - they have their own kit. Please treat the kit with respect and it will provide good service for the term.

2.6 Project Kit

Project kit contains most of the parts and components that teams need for building their prototype, including various actuators, mechanisms, parts, peripherals and electronic components. Three important boards are also included in the kit, i.e., Microcontroller, Driver, and Utility Board. However, students are not limited to the items in the kit and can use other items that they may need.

Actuators	QTY
Shenzhen DC Gearhead Motor (Straight)	2
Shenzhen DC Gearhead Motor (Angle)	2
Universal Stepper Motor	1
Solenoid with Spring/Frame	1
Zheng DC Gearhead Motor	1

Mechanical Peripherals	QTY
Rotary Tool Head Pack	1
Safety Glasses	1
Drill bit Set	1
Hack Saw cutting blade	1

Electronic Peripherals	QTY
74HC14 Hex Schmitt Inverter	3
74HC86 Quad XOR	3
74HC00 Quad NAND	3
74HC08 Quad AND	3
74HC32 Quad OR	3
74HC02 Quad NOR	3
74HC451 3-State Non-inverting Buffer	1
LM358 Operational Amplifiers	3
LM555/NE555N Timer	2
74HC154 - Demultiplexer	1
LM7805 - 5V Regulator	2
LM7806 - 6V Regulator	1
LM7812 - 12V Regulator	1
ULN2001 Transistor Array	2
SN754410 Motor Driver	2
8 pin Socket	4
14 pin Socket	10
16 pin Socket	5
20 pin Socket	2
100 ohm, 5% 1/4W Resistor	10
330 ohm, 5% 1/4W Resistor	10
470 ohm, 5% 1/4W Resistor	10
1k ohm, 5% 1/4W Resistor	10
3.3k ohm, 5% 1/4W Resistor	10
10k ohm, 5% 1/4W Resistor	10
22k ohm, 5% 1/4W Resistor	10
47k ohm, 5% 1/4W Resistor	10
100k ohm, 5% 1/4W Resistor	10
470k ohm, 5% 1/4W Resistor	10
1M ohm, 5% 1/4W Resistor	10

10M ohm, 5% 1/4W Resistor	10
20k 15-turn Potentiometer	2
0.1nF Ceramic Capacitor	5
1nF Ceramic Capacitor	5
0.01uF Ceramic Capacitor	5
0.1uF Ceramic Capacitor	5
1uF Electrolytic Bipolar Capacitor	5
Red LED	4
Green LED	4
1N4001 Power Diode	8
1N4148 Schottky Diode	2
1N5230B Zener Diode	1
2N2222A Signal Transistor	2
5V Coil Relay	2
S/DPDT Toggle Switch	2
ON/OFF Pushbutton Switch	2
Microswitch with long/Curved Slider	1
Jumper Set	1
Solderless Protoboard	1
Alligator Clip (set of 10)	1
Sponge Piece	1

Development Boards	QTY
Microcontroller Board	1
Driver Board	1
Utility Board	1

Board Accessories	QTY
PIC 16F877 Chip	1
PIC 18 F4550 Chip	1
PIC 18F2550 Chip	2
AC-DC Power Adaptor	1
4x4 Keypad	1
16x2 LCD with Backlight	1
USB Cable	1
40-pin I/O Cable Bus	1

Containers	QTY
Electronic Toolbox	1
Tote	1

2.7 Design Store

The following components can be purchased from the Design Store. Teams that plan to purchase some of the following items must open an account with the Store and deposit \$100 in the beginning of the semester. All purchases will be charged against the account. The remainder will be returned to the teams at the end of the course. Design Store will be open ONLY during the first two hours of each laboratory session.

MCU Board Chips & Accessories	Unit Price	Team Limit
PIC16F877 Microcontroller	\$6	2
PIC18F4620 Microcontroller	\$8	1
PIC18F4550 Microcontroller	\$8	1
PIC18F2550 Microcontroller	\$5	1
PIC18F2455 Microcontroller	\$5	1
PIC Driver/Programmer PCB	\$15	1
MM74C922N Keypad Encoder	\$8	1
74LS04 Hex Inverter	\$1	2
4.096MHz Resonator	\$1	5
DS1307 Real-time Chip	\$5	2
Energizer Lithium Coin 3V 24.5MM Battery	\$3	2
AC-DC High-Power Voltage Adapter (+12, +5, -12)	\$15	1
40-pin I/O Cable Bus	\$7	1
16x2 Backlight LCD (pin)	\$10	1
16x2 LCD with Cable	\$8	1
4x4 Keypad	\$7	1
2x8 Keypad	\$4	1
USB Cable	\$5	1

Actuators	Unit Price	Team Limit
Zheng DC Gearhead Motor	\$9	2
Shenzhen DC Gearhead (Straight)	\$5	2
Shenzhen DC Gearhead (Angle)	\$5	2
COPAL Gearhead Motor	\$4	1
Small Cylindrical DC Motor with Worm Gear	\$2	3
Small Pancake DC Motor	\$2	3
Vibration DC Motor	\$3	3
Miniature Pager Motor Pack (5 Motors)	\$3	1

Universal Stepper Motor	\$8	1
NMB Unipolar Stepper Motor	\$5	1
Servo Motor SM-S4306R (Continuous Rotation)	\$10	2
Servo Motor S3006 (60° limit)	\$8	2
Solenoid with Spring/Frame	\$5	2
Large Solenoid	\$7	1
2-Wire Car Door Actuator	\$7	1

Sensors and Switches	Unit Price	Team Limit
SonaSwitch Ultrasonic Trnasducer	\$10	2
Matsushita 0D24K2 24KHz Ultrasonic piezoelectric sensor	\$5	2
Sharp GP2D12 IR Analog Ranger	\$20	1
Sharp GP2D02 IR Digital Ranger	\$25	1
Sharp GP1UD28 IR 40KHz Receiver	\$5	1
Sharp GP1UD287 IR 56.8KHz Receiver	\$5	1
Motorola MIM-5383H4 38.0 KHz Receiver	\$5	1
Vishay Photoreflector	\$5	5
OP805SL IR Phototransistor	\$2	5
OP505A IR Phototransistor	\$2	5
1N6264 IR LED	\$1.5	5
K-2246 IR LED	\$1.5	5
TAOS Colour Sensor - Blue	\$3.5	3
TAOS Colour Sensor - Red	\$3.5	3
Photoresistor Variety Pack (more than 10 sensors)	\$8	1
Piezoelectric Disk Strain-Force Sensor	\$1	10
Allegro 3503 Hall-effect Sensor	\$3	1
Tilt Sensor	\$3	1
Velleman Metal Detector Kit	\$15	1
FairChild H21A1 Break-Beam Optosensor	\$3	2

Break-Beam Optosensor Variety Pack (4 Sensors)	\$8	1
Reed Switch with Frame	\$3	2
Small DPDT 5V Relay Switch	\$2	2
Microswitch (Long Lever)	\$2	3
Microswitch (Short Lever)	\$1.5	3
DPDT Sliding Switch	\$2	3
Toggle Switch	\$2	2
Small Push Button Switch	\$1	10

Electronic Peripherals	Unit Price	Team Limit
LM555 Timer	\$2	2
LM7805 5V Regulator	\$3	2
LM7806 6V Regulator	\$3	2
74LS194 Shift Register	\$3	2
74HC451 3-State Non-inverting Buffer	\$2	2
74LS04 Hex Inverter	\$1.5	2
HCF4050 Buffer	\$3	2
LMF100 Filter IC	\$5	1
L298 H-Bridge IC	\$8	2
LMD18200 H-Bridge IC	\$15	1
ULN2001A Stepper Driver IC	\$2	3
TIP 112 NPN Transistor TO-220	\$2	4
TIP 147 PNP Transistor TO-220	\$2	4
TIP 142 NPN Transistor TO-218	\$2.5	4
TIP 147 PNP Transistor TO-218	\$2.5	4
TIPL 760A NPN Transistor TO-220r	\$1	4
TIPL762A NPN Transistor TO-218	\$0.5	12
Variable Resistor (Trimmer) 200Ω, 500Ω, 5KΩ	\$2	1
1N4001 Power Diode	\$0.33	18
1N4148 Schottky Diode	\$0.33	6
9V Batter Connector	\$0.5	3
Soldering Iron (60W)	\$10	1
Solder	\$2	1

Mechanical Peripherals	Unit Price	Team Limit
Large Gear Set Variety Pack	\$6	1

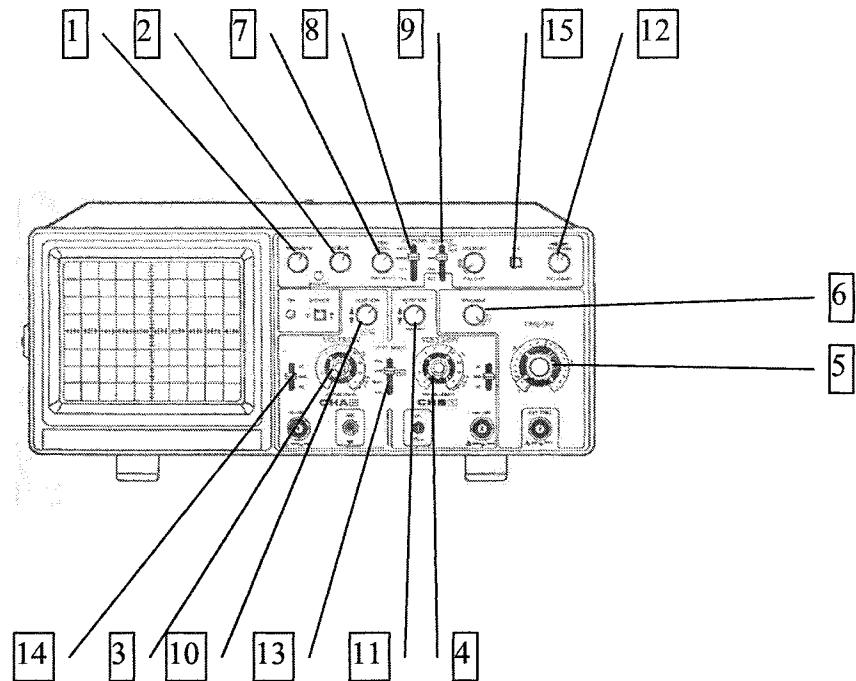
Small Gear Set Variety Pack	\$2	1
Pulley Gear Variety Pack	\$2	1
2in1 Gearhead Motor Kit	\$8	1
TAMIYA Gear Set 72003	\$15	1
Sliding Crank with Break-Beam Optosensor	\$5	1
4.5-6.5mm Flexible Metal Coupling	\$4	2
6.5-9.5mm Flexible Metal Coupling	\$4	2
Small thrust bearing	\$3	2
Large trust bearing	\$5	1
Ball Caster	\$5	1
Cylindrical Double-roller Wheel	\$3	2
Plastic Wheel	\$4	2
Mastercraft Complete 52-piece Rotary Kit	\$35	1
Rotary Tool Head pack	\$15	1
Drill Bit Set	\$8	1
Sand Paper (5 Pieces)	\$1.5	1
Hot Glue Gun with Sticks	\$2	1
10"×10"×3/8" Acrylic Plate	\$7	2
Metal Bracket	\$2	4
Large Bolt and Nut	\$2	1
Steel Wire Pack (3 Rolls)	\$2	1
Clamp Set	\$1	1
Cable Tie Set	\$1	1
Safety Glasses	\$3	1
Swiss Army Knife	\$2	1
Gripper Arm	\$2	1
Coin Sorter Machine	\$35	1
Electric Scooter	\$40	1

Project Items	Unit Price	Team Limit
Box with Square Compartments	\$1	4
Box with Rectangular Compartments	\$2	4
Cylindrical Container	\$1	4
Backgammon Chips 30-piece Pack	\$17	1
Dowels 24-piece Pack	\$15	1

2.8 Laboratory Oscilloscope

2.8.1 What is an oscilloscope?

The oscilloscope is an instrument that is used to observe and measure electrical signals. It provides us with a graph of either a static or a time-varying signal, with the voltage represented in the vertical axis and the time represented in the horizontal axis. This simple graph of the signal is very useful since it allows us to determine the frequency, amplitude and the shape of the signal. The oscilloscope is also useful when measuring DC quantities since it allows us to determine if there exists any AC component in the signal, which cannot be picked up by other instruments such as a DVM. The purpose of this section is to familiarize you with some of the basic controls of the oscilloscope. Refer to the following figure for the location of the controls called out in the following section.



2.8.2 Setting the Main Controls

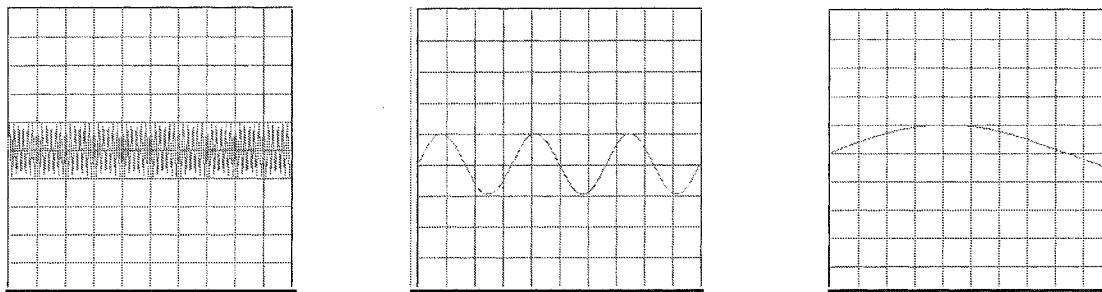
In order to make useful and accurate measurements of a signal it is important to obtain a suitable trace of the signal on the display screen of the oscilloscope. This is accomplished by adjusting the following main controls:

- The intensity and focus
- The attenuation or amplification using the volts/div knob
- The time base using the sec/div knob
- The triggering of the oscilloscope

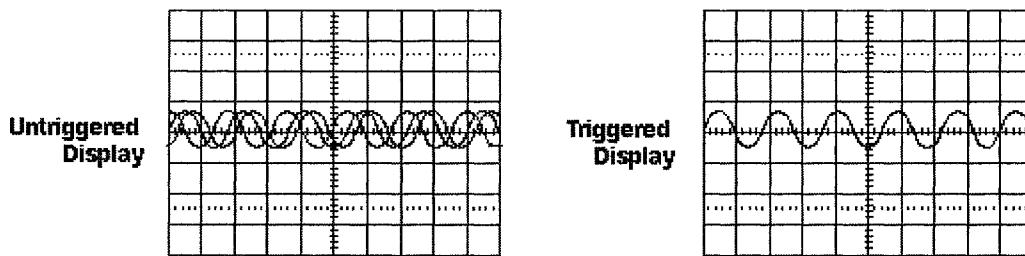
The intensity knob is located to the top right hand side of the display, (1) and it is used to adjust the brightness of the trace. Rotating the knob clockwise increases the brightness and counterclockwise rotation decreases it. The focus control (2) is located next to the intensity control and is used in a similar manner to adjust the sharpness of the trace. These two controls are set correctly when the signal appears bright but not too intense and as sharp as possible.

The volts/div control allows you to scale the trace of the signal vertically. Adjust this setting (3) for channel A and (4) for channel B, so that the trace is as large as possible without going out of the display screen. Before you make any amplitude measurements, make sure that the VAR PULL 5X MAG control (part of the volts/div control) is set to its calibrated position (fully clockwise and pushed in). Once you have found the suitable position for the volts/div control, for example 2V/div, then each one of the eight vertical squares represents 2 Volts.

The signal to be viewed may have a frequency of a couple of Hertz or a few KHz, and it is usually not known beforehand. In order to properly view the signal it is important to properly adjust the sec/div control (5) located at the right side of the oscilloscope. This control allows you to adjust the trace horizontally. Rotate this knob until two or three periods of the signal appear on the display. Before making any frequency measurements make sure that the VAR control (6) is located at its calibrated position (fully clockwise). Below are some examples of the same signal viewed at different positions of the sec/div control (20ms/div, 1ms/div and 0.2 ms/div).



In order to properly view a rapidly time-varying signal, it is important to correctly trigger the oscilloscope. This will result in a stable trace of the signal on the display screen and will in turn enable you to take correct and accurate measurements of the signal's characteristics. In order to understand how triggering works it is important to know how the oscilloscope displays signals. The oscilloscope is constantly drawing the voltage vs time graph of the signal on the display. This makes the trace of the signal on the display appear continuous. In order to make the trace appear stable as well, it is important that every time the oscilloscope displays the signal, it starts plotting at the same point of the display area. This is referred to as triggering the oscilloscope. When the amplitude of the signal to be viewed reaches a pre-selected value (called the triggering level) the oscilloscope starts displaying the signal on the left hand side of the display screen. It then "sweeps" across the screen and draws the graph of the signal. When the trace of the signal reaches the right hand side of the screen the oscilloscope returns to the left side of the screen and waits for the signal to reach the triggering level. When this occurs again the oscilloscope "sweeps" across the screen once more and draws another graph of the signal. If the signal to be viewed is periodic then each graph will look the same as the previous one and the trace will appear to stand still on the screen. Below are some examples of un-triggered and triggered waveforms.



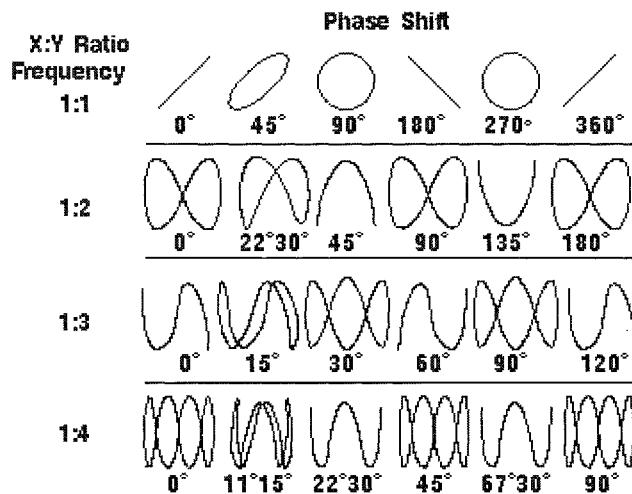
The two main controls for triggering are the TRIGGERING LEVEL (7) and the PULL SLP (-), which is part of the triggering level control (7). The first one allows you to choose the voltage level at which the oscilloscope will start displaying the signal. Note that you cannot trigger the oscilloscope correctly if the triggering level is greater than the

magnitude of the signal. Since the triggering level does not appear on the screen, you just have to keep rotating it until the trace appears to stand still instead of “running” across the screen. The second setting allows you to select the trigger slope, positive or negative. The SOURCE (8) control allows you to choose the source of the triggering signal. As a rule of thumb choose CHA when viewing a signal on that channel A and CHB when viewing a signal on channel B. The COUPLING control (9) is typically left at the auto position.

In conclusion, it is important to note that by adjusting these main controls of the oscilloscope we are not changing any signal characteristics themselves. We are only changing the way the signal is displayed on the oscilloscope so that it is easier to observe it and make useful measurements.

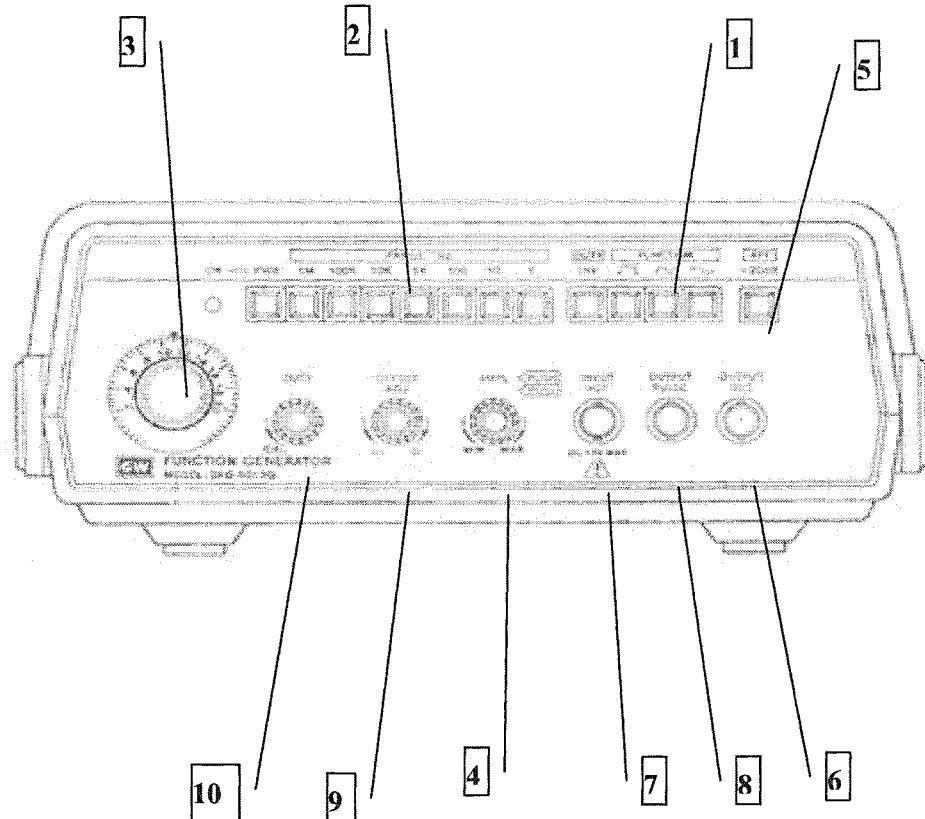
2.8.3 Other Controls

The controls mentioned above are only the basic ones that need to be adjusted in order to get a suitable trace of the signal. As seen on the figure, there are many other controls available to you, which allow you to further manipulate the way the signal appears on the oscilloscope screen. Controls (10) and (11) allow you to move the CHA and CHB trace respectively, vertically on the screen. Control (12) allows you to move the trace horizontally on the screen. These controls are used when trying to make accurate measurements of amplitude or frequency and it is easier to align the trace with the screen gridlines. Also control (13) allows you to choose between viewing channel A only (CHA), channel B only (CHB), both of them at the same time (DUAL), or their algebraic sum (ADD). Common problems relate to these simple settings so make sure that you have the right channel selected. Also make sure that switch (14) is set at the appropriate position. Choosing AC blocks any DC component and allows you to observe only the AC component of the signal. Usually the switch is left at DC, which allows both DC and AC components of the input signal to be applied to the vertical amplifier input. In order to avoid seeing an artificial DC shift, make sure that the “ground” is properly set to zero on the screen. In order to do this, set switch (14) to GND. You should be able to see a straight line across the screen. This is the reference ground level for the displayed signals. You may rotate control (10) or (11), for channel A or B respectively, in order to position that line on the zero of the screen. Control (15) allows you to get the oscilloscope into X-Y mode. In this mode, the oscilloscope no longer plots a voltage vs time graph, but displays channel A on the X axis and channel B on the Y axis. This is a very specific control and depending on the phase shift and frequency ratio of the input signals one may get very interesting figures as seen below. If you are planning on using the oscilloscope in the normal mode, make sure that this switch is not accidentally pushed in.



2.9 Laboratory Function Generator

The function (or waveform) generator is a device, which is used to generate signals of known frequency, amplitude and shape. The purpose of this section is to familiarize you with the basic controls of the function generator. Refer to the following figure for the location of the various controls described below. Even though there exists a few different models of function generators in the design lab, you should be able to find these basic controls on all of them.



Basic operation of the function generator mainly involves the following four tasks:

- Choosing the shape of the desired signal
- Setting the desired frequency
- Adjusting the amplitude of the signal
- Connecting the output to the circuit

The FUNCTION switch (1) allows you to control the shape of the output signal of the function generator. Simply press the switch that corresponds to the desired type of waveform; square, triangle, and sine are provided, satisfying most applications.

Setting the frequency involves two controls: the RANGE switches (2) and the MULTIPLIER (3). By pressing one of the seven RANGE switches you are selecting the output frequency range. You can further adjust the frequency by rotating the MULTIPLIER knob. This allows you to reach 0.2 or 2 times the frequency selected by the RANGE switch. Some function generators in the lab may have two knobs for frequency setting (a COARSE and a FINE control), instead of just the MULTIPLIER. By carefully adjusting these similar controls you may set the frequency of the output to the desired value.

Before connecting the signal provided by the function generator to a circuit it is important to make sure that it has the correct amplitude. You may adjust the amplitude of the output signal with the AMPL control (4). The ATT switch (5) also allows you to change the amplitude of the output signal. When this switch is engaged, the signal provided at the output is attenuated by 20 dB, which means that the signal is roughly 1000 times attenuated. Unless you want to achieve signals in the few mV range make sure that this switch is not pushed in.

The output waveform, as selected by the various controls, is available at the OUTPUT (6) jack, with a BNC connector. The VCG (7) jack allows you to externally control the frequency of the output signal by applying a DC voltage input. The OUTPUT PULSE (8) jack provides a TTL output pulse suitable for driving TTL logic. For the purposes of the design project you will most likely only need to connect to the main OUTPUT (6) jack.

When using the function generator it is important to be familiar with some other secondary controls that also affect the characteristics of the output signal. The OFFSET control (9) adds a DC offset to any signal chosen by the FUNCTION switches. If you want your output signal to be centered around zero volts, make sure that this switch is not accidentally set to a different value. Moreover, if you want the output signal to be perfectly time symmetric make sure that the DUTY control (10) is rotated as far as possible, counterclockwise, in order to ensure correct calibration.

Chapter 3

Machine Shop

3.1	Safety First!	1
3.2	Measurement	5
3.2.1	Calipers	5
3.2.2	Micrometer	6
3.3	Part Layout	8
3.3.1	Scribing Lines	8
3.3.2	Using A Center Punch	8
3.3.3	Using a Compass	8
3.4	Band Saw Machine	9
3.4.1	Selecting and Installing A Blade	9
3.4.2	Operating a Band Saw	9
3.4.3	Lubricating the Blade	10
3.4.4	Cutting Round Stock	10
3.5	Belt Sander Machine	10
3.6	Drill Press	11
3.6.1	Using A Centre Finder	11
3.6.2	Drilling A Hole	12
3.6.3	Deburring A Hole	12
3.6.4	Reaming a Hole	12
3.6.5	Thread Standards	12
3.6.6	Tapping a Hole	13
3.7	Lathe	14
3.7.1	Choosing a Cutting Tool	15
3.7.2	Installing a Cutting Tool	15
3.7.3	Positioning the Tool	16
3.7.4	Feed, Speed, and Depth of Cut	16
3.7.5	Turning	16
3.7.6	Facing	17
3.7.7	Parting	17
3.7.8	Drilling	17
3.8	Milling Machine	18
3.8.1	Squaring the Vise	19
3.8.2	Tilting the Head	19
3.8.3	Types of Milling Cutters	20
3.8.4	Installing Milling Cutters	20
3.8.5	Climb vs. Conventional Milling	21
3.8.6	Calculating Speeds and Feeds	21
3.8.7	Squaring Stock	21
3.8.8	Face Milling	21
3.8.9	Milling Slots	22
3.8.10	Work Holding	22
3.9	Grinding and Buffing	23
3.10	Bending Brake	24



3.1 Safety First!

The purpose of this chapter is to provide basic instructions on the use of standard workshop machines and equipment. But, before any thing, safety issues must be addressed and strictly followed whilst using the machine shop facilities throughout the course. There are some general safety rules, and some more instructions specific to each device, both of which help eliminate or reduce chances of serious injury. Some important notes of safety are listed below in different categories.

GENERAL

- **Shop Capacity**

AT NO TIME SHOULD THERE BE MORE THAN 5 STUDENTS IN THE FRONT SHOP AND MORE THAN 4 STUDENTS IN THE REAR SHOP.

- **Safety Glasses**

EVERYONE MUST WEAR SAFETY GLASSES IN THE SHOP.

Even when you are not working on a machine, you must wear safety glasses. A chip from a machine someone else is working on could fly into your eye.

- **Clothes and Hair**

Check your clothes and hair before you walk into the shop. In particular:

- **IF YOU HAVE LONG HAIR OR LONG BEARD, TIE IT UP.**

If your hair is caught in spinning machinery, it will be pulled out if you are lucky. If you're unlucky, you will be pulled into the machine.

- **NO LOOSE CLOTHING.**

Ties, scarves, loose sleeves, etc. are prohibited

- **NO GLOVES.**

- **REMOVE JEWELRY.**

- **WEAR APPROPRIATE SHOES.**

No open-toed sandals. Wear shoes that give a sure footing. If you are working with heavy objects, steel toes are recommended. REMEMBER! Shop floor can be very slippery.

- **Safe Conduct in the Shop**

➢ A minimum of 2 persons shall be present in the shop at any time the facility is used.

➢ Be aware of what is going on around you. For example, be careful not to bump into people while they are cutting with the band saw (they could lose a finger!).

➢ Concentrate on what you are doing. If you get tired, leave!

➢ Don't hurry. If you catch yourself rushing, slow down.

➢ Don't rush speeds and feeds. You will end up damaging your part, the tools, and maybe the machine itself.

➢ Listen to the machine. If something doesn't sound right, turn the machine off.

➢ Don't let someone else talk you into doing something dangerous.

➢ Don't attempt to measure a part that is moving.

- **Operation**

➢ **IF YOU DON'T KNOW HOW TO DO SOMETHING, ASK!**

➢ **BEFORE YOU START THE MACHINE:**

- Study the machine. Know which parts move, which are stationary, and which are sharp.

- Double check that your work piece is securely held.

- Remove chuck keys and wrenches.

- **DO NOT LEAVE MACHINES RUNNING UNATTENDED.**
- **CLEAN UP MACHINES, BENCHES AND FLOOR AFTER YOU USE THEM.**
 - A dirty machine is unsafe and uncomfortable to work on.
 - Do not use compressed air to blow machines clean. This endangers people's eyes and can force dirt into machine bearings.
 - Dirty floor can be very slippery, WATCH OUT!

The following checklists are given below as a means of quick reference only. Before beginning work on a new machine, read the section that deals in depth with the working of that machine thoroughly. It is a good idea to quickly go through these checklists each time you use a machine.

Belt Sander Machine

- 1) Check the belt or disk to make sure it is in good condition and not torn. The shop supervisor will replace worn belts or disks.
- 2) Keep fingers and hands clear off the moving or rotating surface.
- 3) Hold the workpiece securely and use only moderate pressure.
- 4) Sand only on the downward motion side of the disk sander.
- 5) Move the workpiece side to side on the sanding surface to prevent rapid wear of the belt or disc.

Drill Press

- 1) Check the drill press head and table for security and condition before starting.
- 2) A centre punch will help locate the hole to be drilled in the correct place.
- 3) Select the correct speed for the material and size drill being used.
- 4) All workpieces must be held securely for drilling by using either a drill vise or C-clamps. A workpiece that moves when being drilled can break the drill, and injure the operator and destroy the workpiece. Large workpieces must be set firmly against the drill press column so that if the drill "grabs" the workpiece can not spin and cause injury to the operator or others. If the drill grabs the workpiece and it is yanked loose of the clamps and begins to spin, maintain downward pressure with the press and turn off the power. Do not retract the drill as this would allow the workpiece to be thrown from the press and may cause serious injury.
- 5) Hands are to be kept clear of the revolving spindle, drill and chips.
- 6) Always ease up on the feed or drill pressure as the drill begins to break through the workpiece. Heavy feed pressure will cause the drill to dig in, and could damage the material being drilled, break the drill, or cause the workpiece to spin.
- 7) When drilling large holes, drill a pilot hole with a small drill such as 1/8 and then step up in size to prevent drill chatter.
- 8) Be sure the drill press is stopped before removing the work piece, chips or cuttings.

Lathe Machine

- 1) Roll up loose sleeves, and do not wear loose clothes such as sweaters or neckties while operating the lathe.
- 2) Be certain the workpiece is set up securely and tightly when using chucks and collets.

- 3) REMOVE THE CHUCK KEY IMMEDIATELY AFTER EACH USE. If the lathe where accidentally activated while the chuck key was still in the chuck, the key would become a very fast moving projectile and possibly cause serious injury.
- 4) Keep hands on the controls or at your side while the lathe is running.
- 5) Keep hands away from chips as they are very sharp and possibly hot.
- 6) Complete cuts that are close to the chuck or against a shoulder by hand feeding to prevent machinery or workpiece damage.
- 7) Remove the tool holder and tool post before filing or polishing.
- 8) Never push the reverse switch while a chuck is moving forward as this could cause the chuck to unscrew itself and fall off and cause serious injury.
- 9) Regulate the depth of cut according to the size and type of material.
- 10) Use tools that are properly ground for the particular job.
- 11) You may never check measurements or surface finishes of the workpiece while it is spinning.
- 12) After you have chucked up your workpiece and completed your tool setup, you must spin the chuck by hand to ensure that the jaws of the chuck and the workpiece will not hit the carriage of the lathe or the tool.
- 13) Counterbalance workpiece on the faceplate if it is irregular in shape.
- 14) Stand to one side of the revolving faceplate to avoid being hit by flying objects.

Metal Band Saw

- 1) All workpieces must be secured in the machine's clamp.
- 2) The moveable jaw of the machine's clamp pivots about its center. Thus, if your workpiece extends less than half way through the jaws of the clamp, you must use a spacer on the other side of the pivot in order to prevent slipping.
- 3) Do not allow the machine to drop rapidly causing the blade to impact the workpiece. Slowly lower the saw and let it gently engage the workpiece.
- 4) A minimum of three saw teeth must be engaged in the workpiece at all times. If less teeth are engaged then the force per tooth is so great that the teeth will tear off the blade.
- 5) Control the decent of the blade through the entire cut, do not allow it cut through the material as fast as it can possibly go.

Milling Machine

- 1) The milling machine is a precision piece of equipment, so it is important to not damage the table. The table is not a work bench or a place to put tools.
- 2) Be sure you know how to stop the milling machine quickly before operating the machine.
- 3) Be sure the power feed controls are in their "Neutral" position before turning on the machine.
- 4) Handle cutters carefully. They are sharp. If they can cut metal, they can cut you.
- 5) Use a soft hammer or mallet to seat the workpiece against the parallel bars or bottom of the vice.
- 6) Secure the workpiece firmly in the vice or with appropriate clamps.
- 7) Check the workpiece, milling machine table, and holding device for clearance of the quill during the cutting.
- 8) Set the machine for the proper depth of cut.

- 9) Check the correct spindle speed for the type of material and the cutter being used. Ask your supervisor if you need to change the speed.
- 10) Select the proper direction of rotation for the cutter.
- 11) Feed the workpiece against or opposite the direction of rotation of the cutter.
- 12) Keep hands on the controls while the machine is running.
- 13) Never try to feel the finished surface while the cut is being taken.

Shear Brake

- 1) Follow the manufacturer's specifications as to gauge of sheet metal that can be safely cut. This sheer machine can cut up to .025 inch steel sheet or .050 inch aluminum sheet.
- 2) Keep fingers and measuring scales out of the way of the blade.
- 3) Do not cut round stock or anything except sheet metal in the sheer.
- 4) Place the sheet against the guide and then clamp it in position with the clamp on the machine.
- 5) The treadle is operated with one foot, and the other foot must be kept clear as the treadle comes down.
- 6) Return the treadle to the up position slowly with foot pressure. Do not let it make a rapid return.
- 7) Pick up the scrap pieces when you have completed cutting.
- 8) Bend only sheet stock in the brake. No round stock.
- 9) Adjust the clamping bar correctly to suit gauge of metal being formed, and stand clear of the moving part of the brake.

Wood Band Saw

- 1) This machine is NOT for cutting hard materials such as metal.
- 2) Stand to one side while doing power-on testing of blade tracking. Should the blade come off the wheels or break it could cause serious injury.
- 3) Adjust the blade guides and rollers properly, and adjust the speed. The upper saw guide should be 1/4 inch or less above the workpiece.
- 4) Check the workpiece to be sure it is free of defects (i.e. broken off tool bits, etc.)
- 5) Plan the cut so as to prevent backing out of a cut, as this will pull the blade off the wheels. Make relief cuts as needed.
- 6) Holding the workpiece firmly, feed the workpiece at a moderate rate.
- 7) Use a push stick when sawing small pieces.
- 8) When feeding a workpiece into the band saw blade, your fingers should not be in line with the blade in case the workpiece cuts faster than you expected.
- 9) A minimum of three teeth must be engaged in the workpiece at all times or the teeth will be torn off the blade.

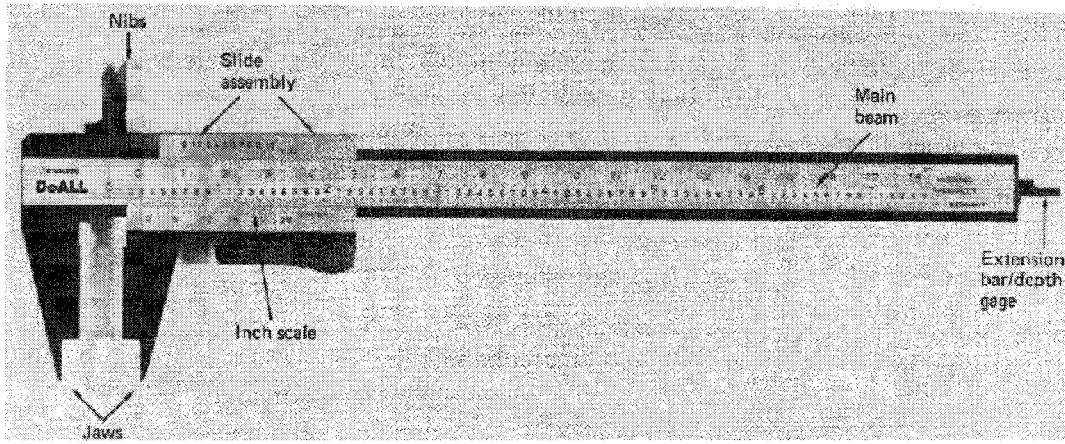


Figure 3.1: A Vernier Caliper

3.2 Measurement

Measurement is the comparison of an unknown dimension to a known standard. Good measuring instruments are key to high volume production. Without them, parts could not be built accurately enough to be interchangeable. Each assembly had to be hand fitted together. Today, measuring tools are essential for most machining operations from initial part layout to final inspection.

3.2.1 Calipers

Figure 3.1 depicts a caliper. It can measure lengths from 0 to 7.5 inches to a precision of one thousandth of an inch. One can measure the outside of a part with the jaws, the inside of a hole or slot with the nibs, or the depth of a hole or shoulder with the extension bar. The particular caliper shown in the figure has a vernier scale. It takes a little practice to read it properly. Calipers often have a dial or digital readouts instead. To read a vernier caliper:

- Read the large number division first.
- Read the small number division.

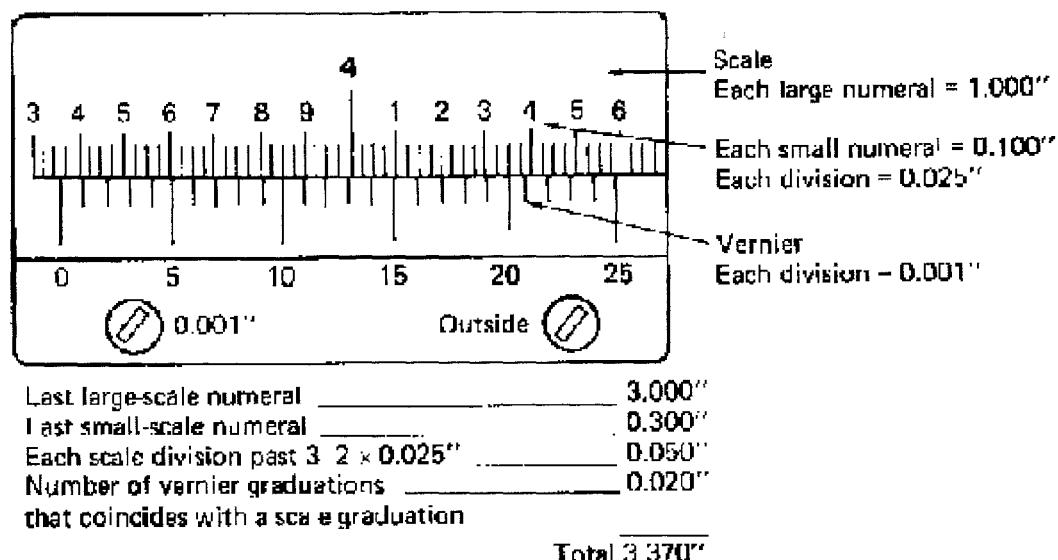


Figure 3.2: Reading a Vernier Scale - An Example

- Read the number of smaller subdivisions. Each represents 0.025 inches to be added to the measurement.
- Read which line on the vernier lines up with a line on the main beam. For each line a thousandth must be added to the measurement.

An example is shown in Figure 3.2.

3.2.2 Micrometer

A micrometer generally provides greater precision than a caliper, but can measure a smaller range of lengths. A micrometer is depicted in Figure 3.3. To use a micrometer, place the part in the opening. Next, turn the thimble until the spindle contacts the work. To apply a consistent pressure to the part, use the ratchet stop. Use the clamp ring to hold the thimble in place while you read the micrometer.

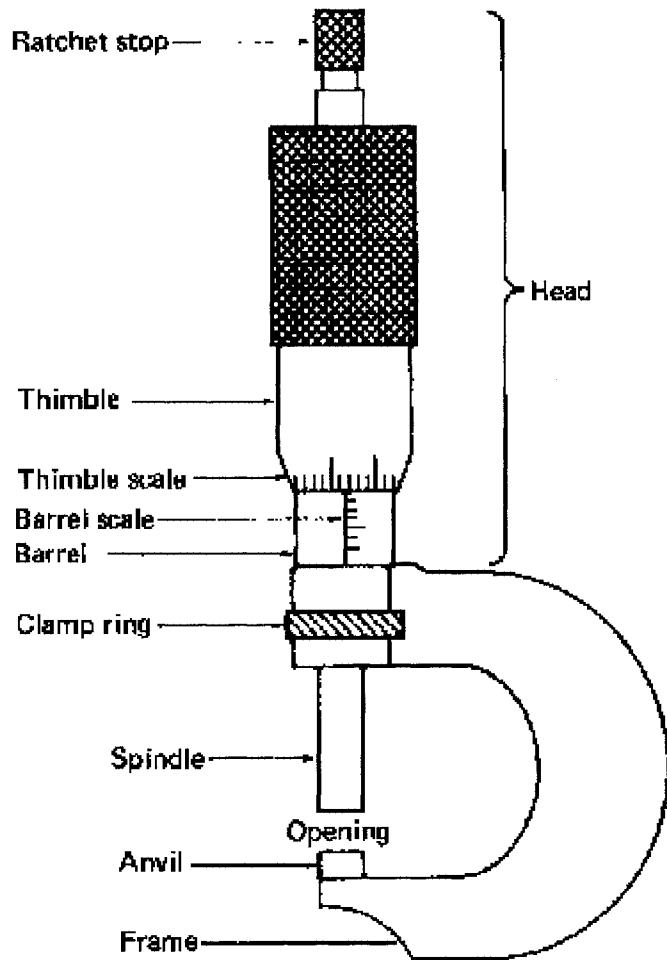


Figure 3.3: Parts of a Micrometer

To read the micrometer:

- Read the exposed number on the barrel.
- Read the number of divisions past the number. Each division represents 0.025 inches.
- Read the division on the spindle. These usually read to less than thousandths of an inch.

An example of reading the micrometer is shown in Figure 3.4.

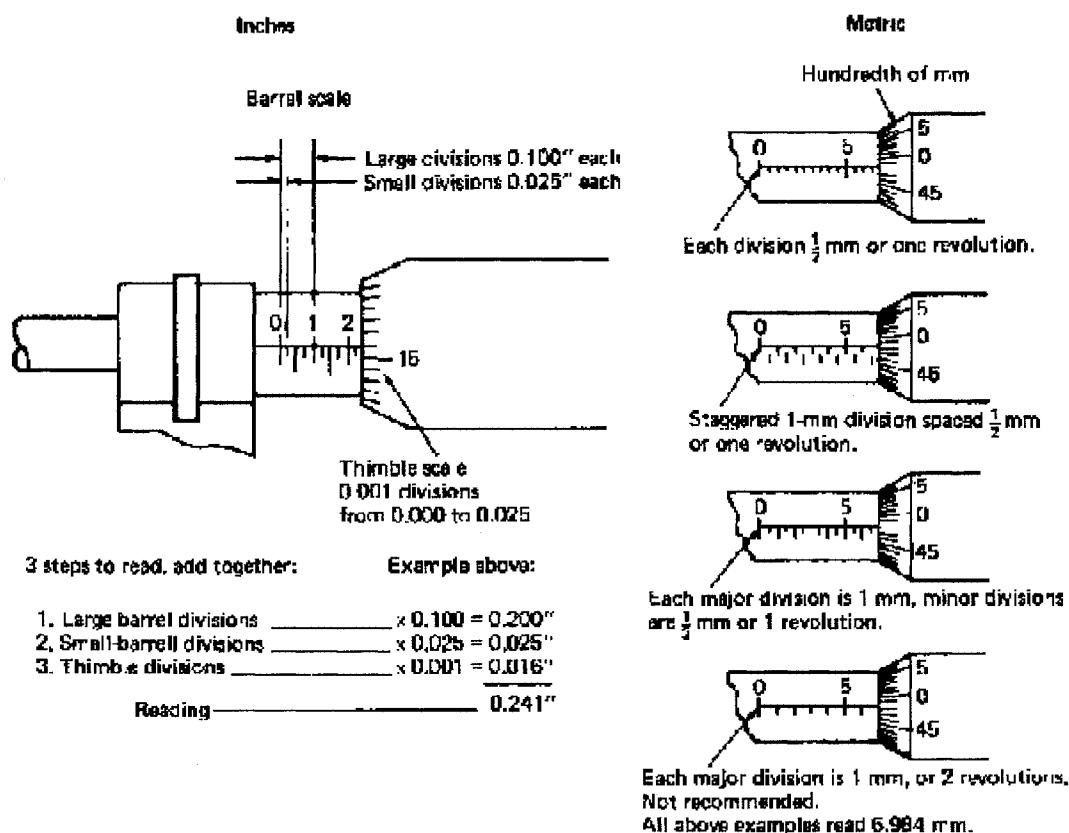


Figure 3.4: Reading a Micrometer - An Example

3.3 Part Layout

Always have your drawings completed before coming to the machine shop. The shop can be distracting and you are bound to make mistakes if you try to do drafting or design work there. One good way to lay a part out on a piece of stock is simply to plot out a 1:1 scale drawing of the part and glue it to the part surface. If you do it this way, you can be sure no errors will be made in transferring the part from the computer to the surface of the part. If a scale drawing is not available, the part can be marked by hand. First, apply a thin coat of blue die. If you apply too much, it will have a tendency to flake off when you try to scribe a line through it. You can use a square and a scribe to make a line perpendicular to an edge on your part. The scribed line will be about 0.002 inches wide, so you can make a reasonably accurate part by milling up to such a line by eyes.

3.3.1 Scribing Lines

You can make a line that is parallel to an existing edge on your part by offsetting a specified distance with a ruler or, more accurately, with parallel plates. Parallel plates come in widths spaced 1/16 inch apart and can be stacked up to achieve desired dimensions. Note how an extra parallel is held vertical to provide alignment with the edge of the part. Another way to scribe lines offset from an existing feature is with a caliper. First, set the desired offset and lock it in. Then, run the caliper along the edge of the part scribing a line with the point of the caliper as you go. This is quick and easy, but somewhat less accurate than using parallel plates.

3.3.2 Using A Center Punch

Locations of holes referenced to corners can be found by crossing two scribe lines. Hole positions should be marked with a center punch. If accuracy is important, use a magnifying glass to set the punch on the mark. Then, give the punch a couple of light taps with a hammer. The center punch creates a dimple that tends to guide the drill bit to the proper location. Without the dimple, the drill bit can "walk" away from the desired location.

3.3.3 Using a Compass

Circles and fillets can be marked using a compass. Just set one end at the center of the arc, and then scribe the arc. There are a number of useful "tricks" that can be learnt to draw simple geometric shapes with a compass. The compass can be used to accurately bisect an angle, trisect an angle, and draw a hexagon, amongst many other applications. There are plenty of references on the Internet regarding geometric construction.

3.4 Band Saw Machine

The band saw machine is useful for cutting stock to size and roughing out shapes. It contains a serrated blade that forms one continuous loop. The blade is stretched over two pulleys, the upper one idle, and the lower one driven by a variable speed electric motor. The SF4003 Machine Shop is equipped with two wood bandsaws, in the front shop, and a metal bandsaw in the rear shop. The wood bandsaws can be used to cut plastic, plexiglass and acrylic, in addition to wood.

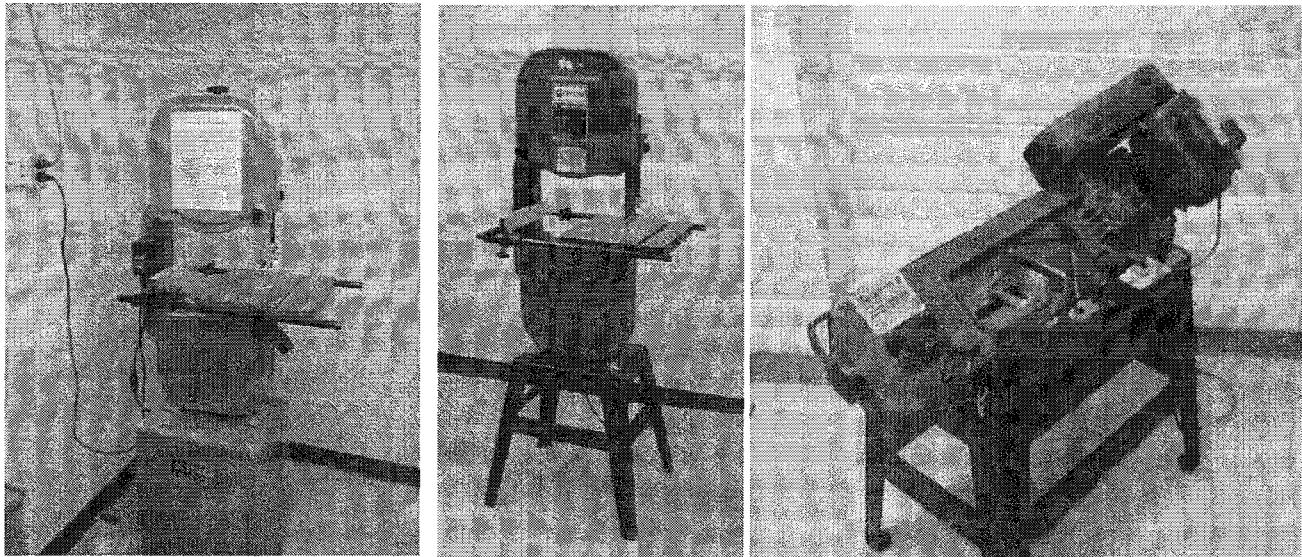


Figure 3.5: Wood and metal band saw machines in the Design workshop

3.4.1 Selecting and Installing A Blade

There are many different types of blades that can be installed in a band saw. They vary in tooth size, tooth shape, blade material, etc. If you are unsure of what type of blade to use for a particular task, then refer to a machinist handbook or manufacturer's documentation. The material to be cut should be at least three tooth widths in thickness. Therefore, it is often best to cut thin sheets with a shear rather than with a band saw. When the proper blade is found, it can be cut to length and welded into a continuous band using an electrical resistance welder built into the band saw. Upper and lower doors open to expose the pulleys. A tensioner allows the upper pulley to be raised and lowered. Lowering the upper pulley makes it easier to install a new blade. The blade must lie properly in the upper and lower blade guides. Once the blade is in place, the tension should be set and locked.

3.4.2 Operating a Band Saw

Before starting the band saw, you should adjust the blade guide/guard to the appropriate height. The less blade that is exposed, the safer you will be. Always set the blade guide just high enough to clear the part you are cutting.

The appropriate cutting speed varies widely for different jobs. For instance, mild steel should be cut at much lower speeds than Aluminum alloys. To adjust the blade speed the motor must be on. Motor speed can be varied with the variable speed control while motor rpm is monitored on the speed indicator dial.

When cutting with a band saw, proper technique is important. Do not lean excessively into the work and keep your hands braced against the table. One of the best features of the band saw is its ability to cut curved shapes. Watch for proper position of the hands in this clip.

3.4.3 Lubricating the Blade

If you are making a long or deep cut, lubricate the blade with stick wax. Just push the tube briefly into the running blade. Do not attempt to apply wax with your fingers.

3.4.4 Cutting Round Stock

The band saw has a tendency to spin round stock and the rough edges of the stock could cut your hands. To avoid this, secure the stock in a drill press vise before cutting.

3.5 Belt Sander Machine

A belt sander is useful for removing burrs and rough edges from parts. It is composed of a belt coated with abrasive riding about two pulleys. The lower pulley is driven by a motor. The upper pulley follows and allows tension in the belt to be adjusted. Belt sanders are effective on wood, most metals (aluminum, steel, brass, etc.), and some plastics. The small particles generated by the belt sander can be toxic. It is good to use a sander with a vent attached to it and to wear a mask when using the belt sander. Don't use the belt sander on printed circuit boards or fiberglass; they create toxic particles.

When smoothing edges and rounding corners, the part should be supported on the table of the machine. The part should be moved back and forth to achieve a better finish and to avoid hot spots on the belt.

To round off the edges of a part, one must hold the part on an angle with respect to the belt. Be sure that the belt is pointing down into the belt. If the part is pointing up, the belt could catch on the part and throw it down into your hand.

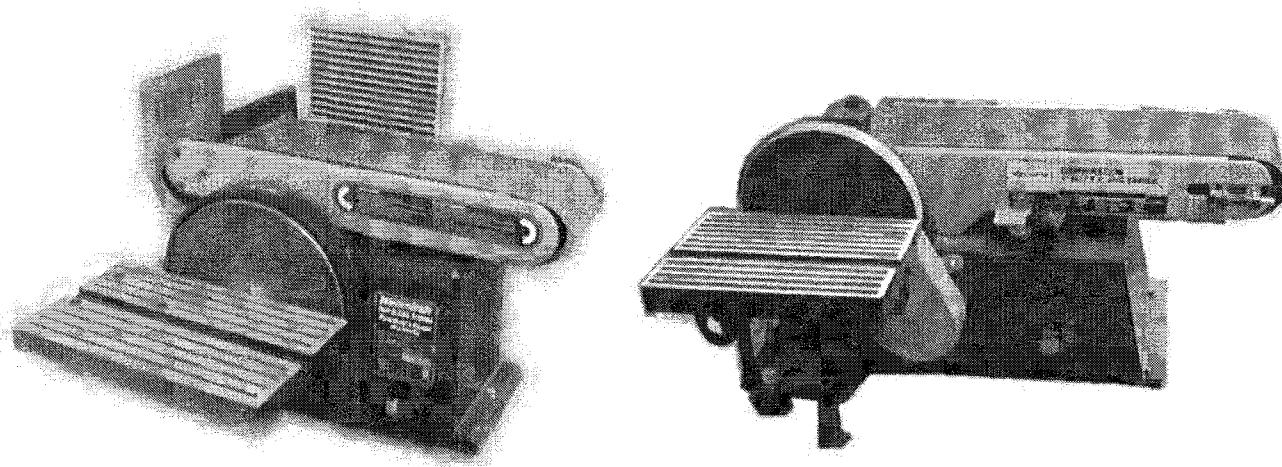


Figure 3.6: Belt sander machines in the Design workshop

3.6 Drill Press

A drill press is preferable to a hand drill when the location and orientation of the hole must be controlled accurately. A drill press is composed of a base that supports a column. The column in turn supports a table. Work can be supported on the table with a vise or hold down clamps, or the table can be swiveled out of the way to allow tall work to be supported directly on the base. Height of the table can be adjusted with a table lift crank than locked in place with a table lock. The column also supports a head containing a motor. The motor turns the spindle at a speed controlled by a variable speed control dial. The spindle holds a drill chuck to hold the cutting tools (drill bits, center drills, etc.).

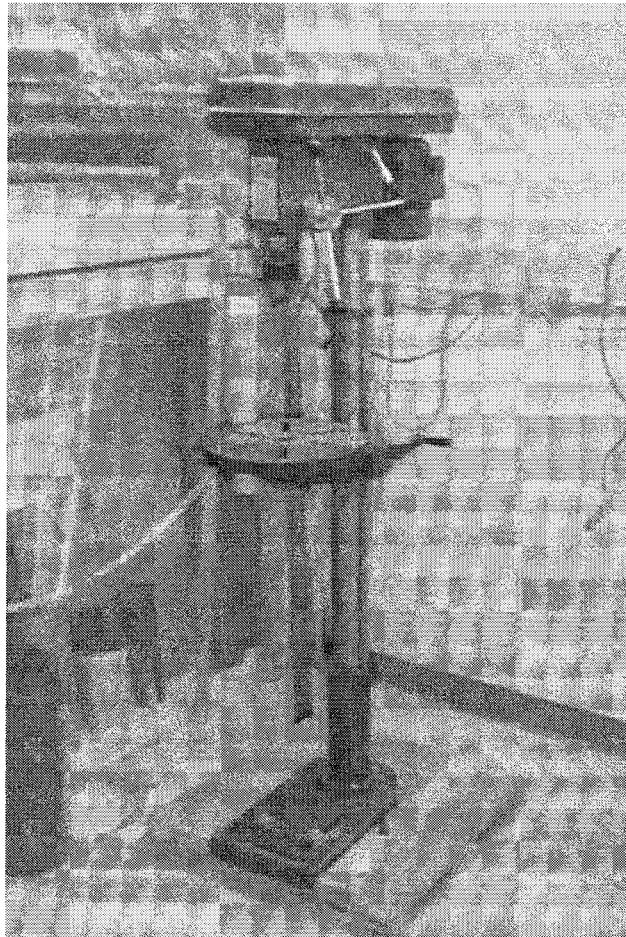
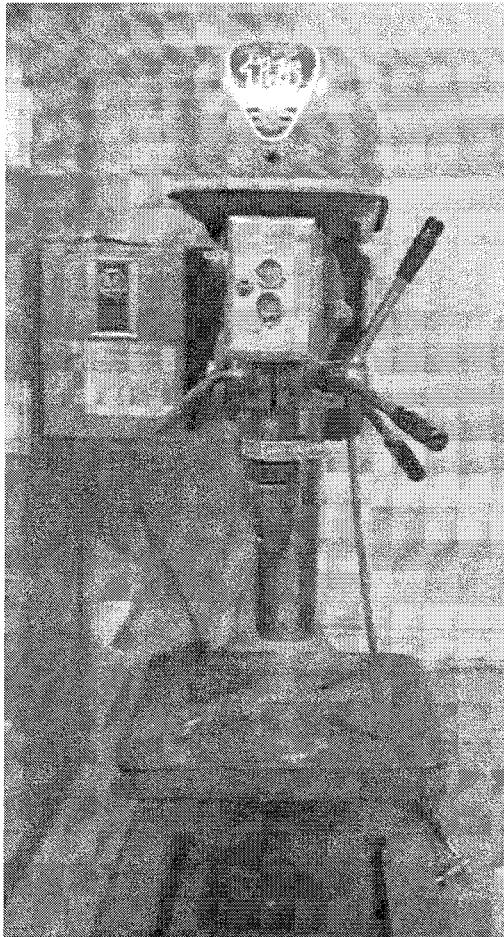
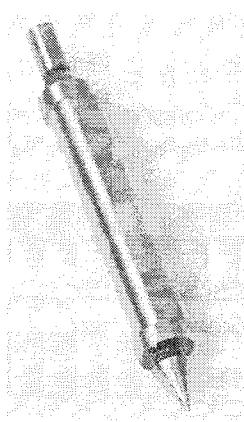


Figure 3.7: Drill press machines in the Design workshop

3.6.1 Using A Centre Finder

A center finder is useful for setting the spindle of a drill press or mill accurately over a known point. A center finder is made of two separate pieces spring loaded together. The center finder is used by installing it in the chuck, placing the pointed end into a center punch mark, and pressing down *lightly* on the quill. At first, the two parts will probably not be concentric.

The bed of the drill or mill can then be adjusted to until the two halves are concentric. Check for concentricity by running your fingers up and down the center finder. You should detect no steps. You should be able to locate the desired position to within 0.001 inches using this technique. Now the hole is ready to be drilled.



3.6.2 Drilling A Hole

First, a center drill should be used. A center drill has a thick shaft and very short flutes. It is therefore very stiff and won't walk as the hole is getting started. It doesn't cut as easily as a drill bit, so you should use cutting fluid.

Now the hole can be drilled. If the hole is large, it is a good idea to drill a smaller pilot hole before drilling the final one. Your hole will be more accurately positioned, rounder, and the bits will last longer. If the hole is deeper than it is wide, use coolant and back off occasionally to clear the chips. As you step up in drill size, you will need to reduce the spindle speed. If drilling a through hole, ensure that the bit will not drill the table after moving through your work. To set a desired depth of hole, there is a depth stop on the quill.

3.6.3 Deburring A Hole

Usually, the top edge of the hole will be fairly clean, but the bottom edge will have substantial burrs. To remove them, insert a deburring tool into the hole and run the tool about the edge of the hole with moderate pressure.

3.6.4 Reaming a Hole

A drilled hole will be accurate to about two thousandths of an inch in diameter. If greater precision is required for slip fits or interference fits, a reamer must be used. The straight flutes of a reamer cannot drill a hole. You must drill a hole slightly undersize to start. Be sure to drive the reamer down with a constant, slow speed.

3.6.5 Thread Standards

The threads cut by taps and dies conform to a standard for the shape of the threads. Often it is the *American National Standard* in National Coarse (NC) or National Fine (NF). NC has fewer threads per inch than NF. NC is most common while NF is favored in precision assemblies. Also in common use is the *Unified Thread System* with UNC (coarse) and UNF (fine). The only difference between National and Unified threads is the shape of the root and crest. The threads in the two different standards will mate. Fasteners are designated by their diameter, number of threads per inch, and shape. For example, 1/4-20NC means 1/4 inch diameter, 20 threads per inch, and National Standard. Threads smaller than 1/4 inch are designated by number from 0 (smallest) to 12 (largest). To convert number to diameter multiply by 0.013 and add 0.060.

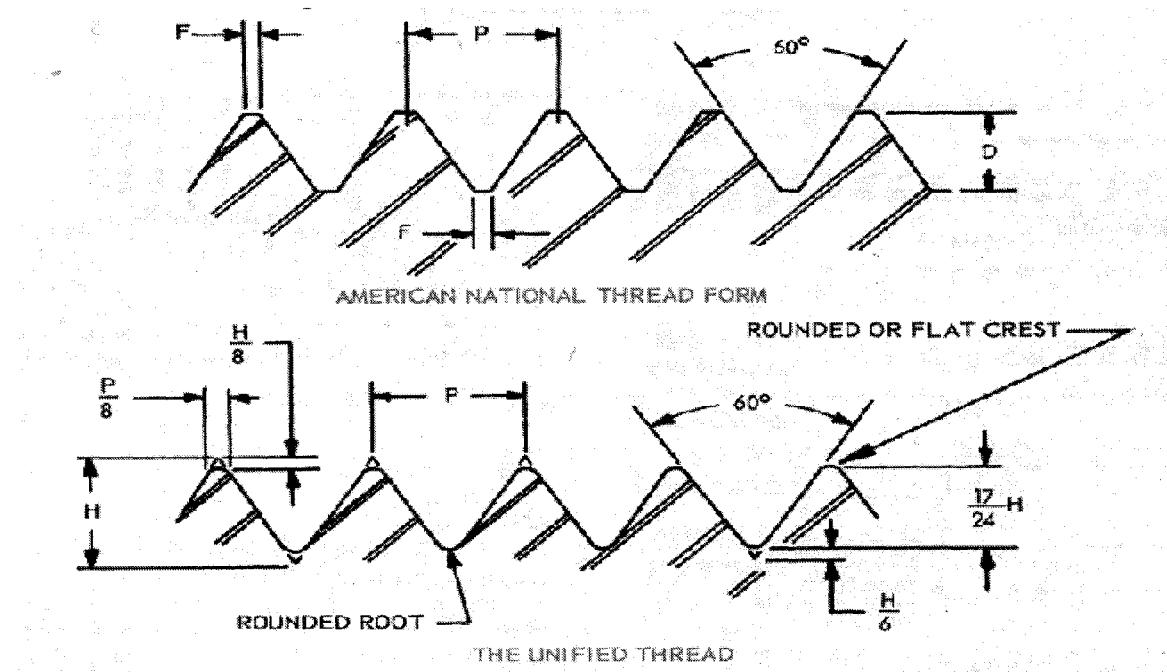


Figure 3.8: Thread standards

3.6.6 Tapping a Hole

If you want to cut threads in a hole, use a tap. Figure 3.9 shows what a tap looks like. It has cutting edges to cut the threads and straight flutes to allow chips to be expelled. Note that the end is tapered slightly to help the tap get started. Taps and dies are hard and brittle so you should be careful working with them (try not to drop them or force them into a hole when stuck). Be sure that the hole you drilled is the correct size for the tap you're using or it may break inside your part, refer to the chart below or machinist's handbook. The die in the picture is for cutting external threads on a shaft.

It is better to use the drill press to help maintain alignment as you tap the hole. First, use the centerfinder to place the spindle directly above the hole. Then install a tapered guide into the chuck. Put the tap in place and apply moderate pressure with the quill as you turn the tap. It's good practice to back the tap up a bit for every quarter turn of thread you cut.

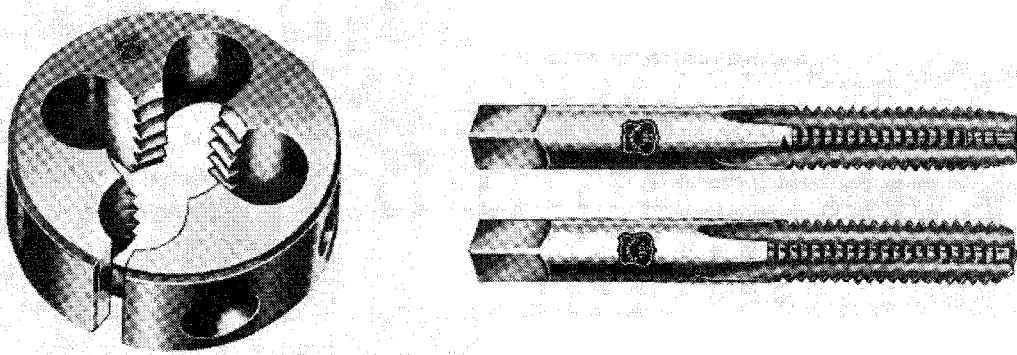


Figure 3.9: Two Taps and a Die

3.7 Lathe

The purpose of a lathe is to rotate a part against a tool whose position it controls. It is useful for fabricating parts and/or features that have a circular cross section. The spindle is the part of the lathe that rotates. Various work holding attachments such as three jaw chucks, collets, and centers can be held in the spindle. The spindle is driven by an electric motor through a system of belt drives and/or gear trains. Spindle speed is controlled by varying the geometry of the drive train.

The tailstock can be used to support the end of the workpiece with a center, or to hold tools for drilling, reaming, threading, or cutting tapers. It can be adjusted in position along the ways to accommodate different length workpieces. The ram can be fed along the axis of rotation with the tailstock hand wheel.

The carriage controls and supports the cutting tool. It consists of:

- A saddle that mates with and slides along the ways.
- An apron that controls the feed mechanisms.
- A cross slide that controls transverse motion of the tool (toward or away from the operator).
- A tool compound that adjusts to permit angular tool movement.
- A tool post T-slot that holds the tool post.

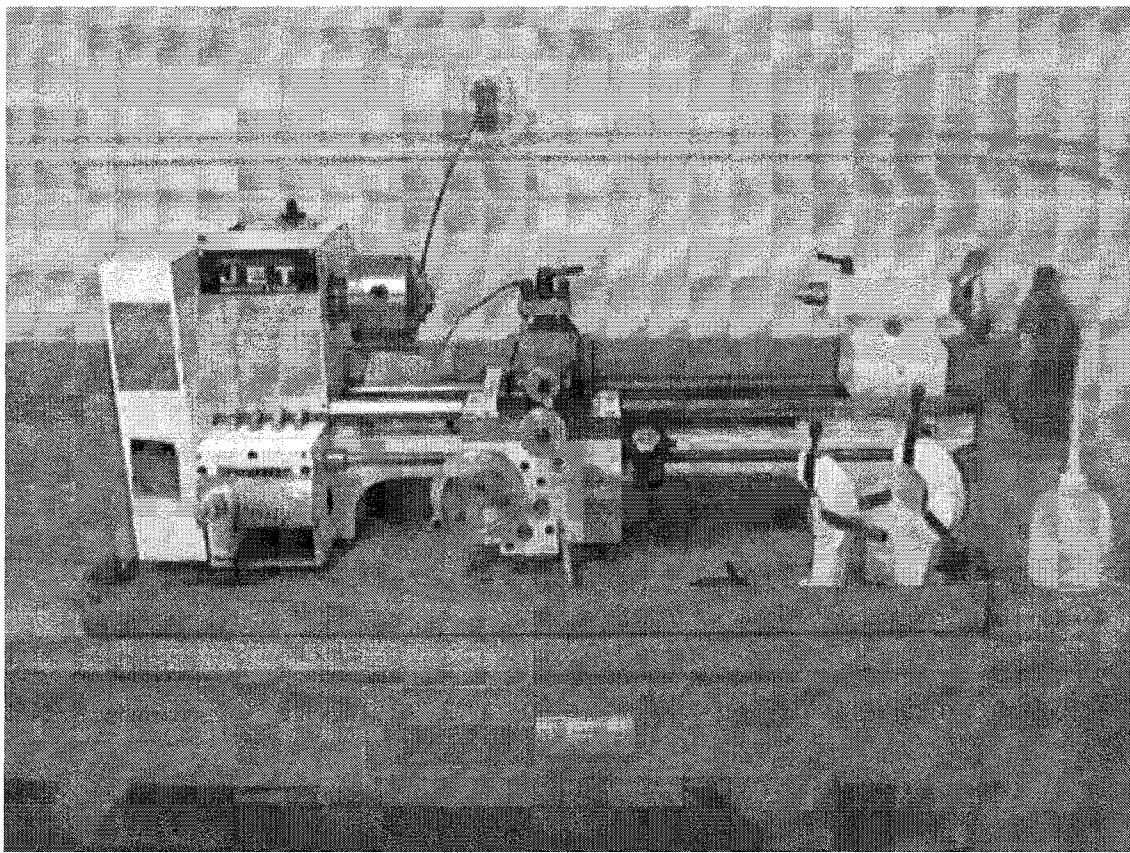


Figure 3.10: The lathe machine in the Design workshop

3.7.1 Choosing a Cutting Tool

Figure 3.11 shows a typical cutting tool and the terminology used to describe it. The actual geometry varies with the type of work to be done. The standard cutting tool shapes are shown in Figure 3.12.

- Facing tools are ground to provide clearance with a center.
- Roughing tools have a small side relief angle to leave more material to support the cutting edge during deep cuts.
- Finishing tools have a more rounded nose to provide a finer finish. Round nose tools are for lighter turning. They have no back or side rake to permit cutting in either direction.
- Left hand cutting tools are designed to cut best when traveling from left to right.
- Aluminum is cut best by specially shaped cutting tools (not shown) that are used with the cutting edge slightly above center to reduce chatter.

3.7.2 Installing a Cutting Tool

Lathe cutting tools are held by tool holders. To install a tool, first clean the holder, and then tighten the bolts. The tool post is secured to the compound with a T-bolt. The tool holder is secured to the tool post using a quick release lever.

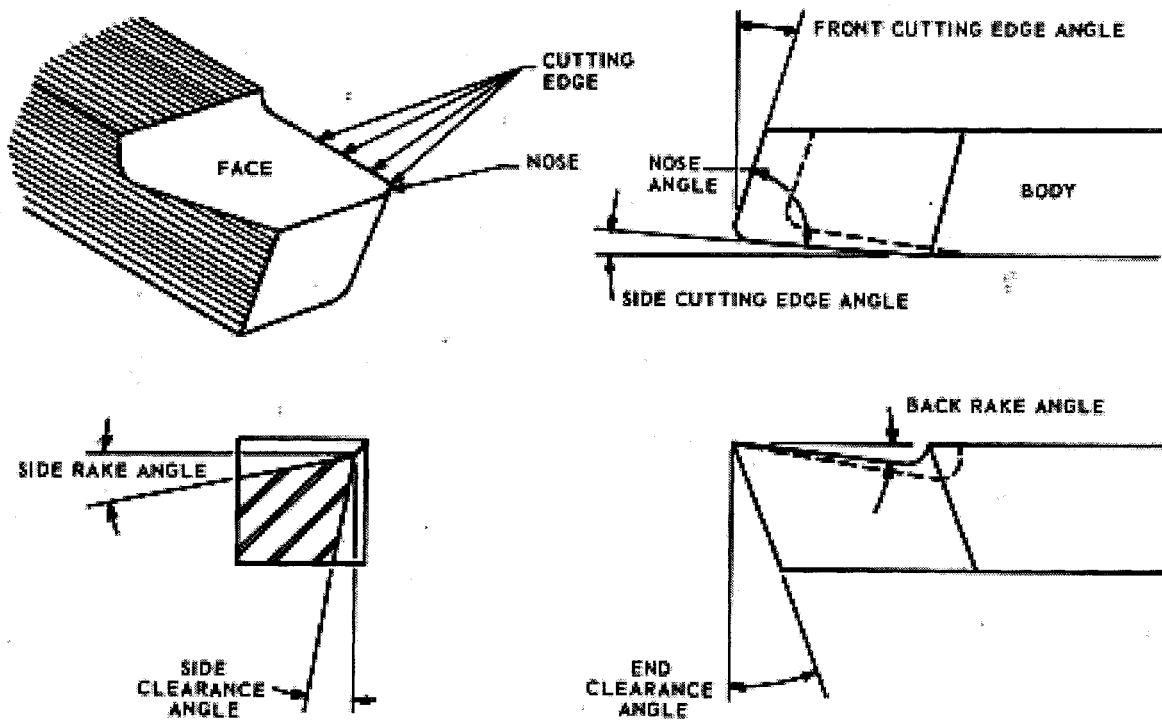


Figure 3.11: Cutting tool terminology

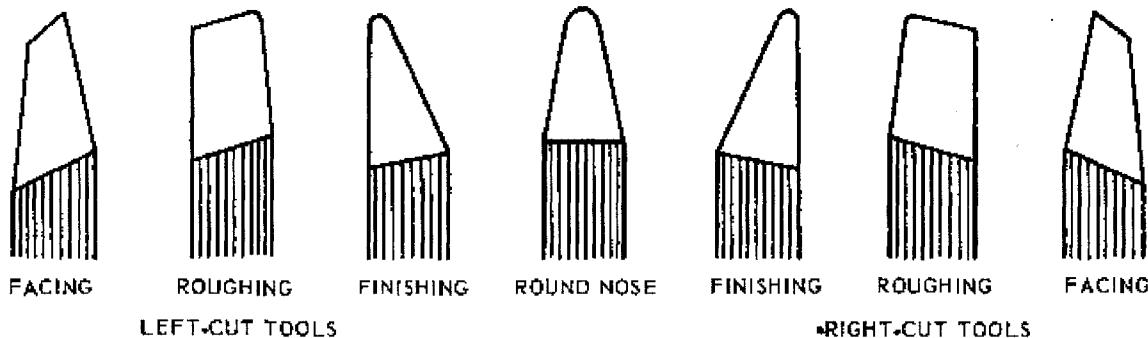


Figure 3.12: Standard Cutting Tools

3.7.3 Positioning the Tool

In order to move the cutting tool, the lathe saddle and cross slide can be moved by hand. There are also power feeds for these axes. Procedures vary from machine to machine. A third axis of motion is provided by the compound. The angle of the compound can be adjusted to allow tapers to be cut at any desired angle. First, loosen the bolts securing the compound to the saddle. Then rotate the compound to the desired angle referencing the dial indicator at the base of the compound. Retighten the bolts. Now the tool can be hand fed along the desired angle. No power feed is available for the compound. If a fine finish is required, use both hands to achieve a smoother feed rate.

The cross slide and compound have a micrometer dial to allow accurate positioning, but the saddle doesn't. To position the saddle accurately, you may use a dial indicator mounted to the saddle. The dial indicator presses against a stop (often a micrometer as shown in the clip below).

3.7.4 Feed, Speed, and Depth of Cut

Cutting speed is defined as the speed at which the work moves with respect to the tool (usually measured in feet per minute). Feed rate is defined as the distance the tool travels during one revolution of the part. Cutting speed and feed determines the surface finish, power requirements, and material removal rate. The primary factor in choosing feed and speed is the material to be cut. However, one should also consider material of the tool, rigidity of the workpiece, size and condition of the lathe, and depth of cut. For most Aluminum alloys, on a roughing cut (.010 to .020 inches depth of cut) run at 600 fpm. On a finishing cut (.002 to .010 depth of cut) run at 1000 fpm. To calculate the proper spindle speed, divide the desired cutting speed by the circumference of the work. Experiment with feed rates to achieve the desired finish. In considering depth of cut, it's important to remember that for each thousandth depth of cut, the work diameter is reduced by *two thousandths*.

3.7.5 Turning

The lathe can be used to reduce the diameter of a part to a desired dimension. First, *clamp the part securely in a lathe chuck*. The part should not extend more than three times its diameter. Then install a roughing or finishing tool (whichever is appropriate). If you are feeding the saddle toward the headstock use a right-hand turning tool. Move the tool off the part by backing the carriage up with the carriage hand wheel, then use the cross feed to set the desired depth of cut.

3.7.6 Facing

A lathe can be used to create a smooth, flat, face very accurately perpendicular to the axis of a cylindrical part. First, clamp the part securely in a lathe chuck. Then, install a facing tool. Bring the tool approximately into position, but slightly off of the part. Always *turn the spindle by hand* before turning it on. This ensures that no parts interfere with the rotation of the spindle. Move the tool outside the part and adjust the saddle to take the desired depth of cut. Then, feed the tool across the face with the cross slide. If a finer finish is required, take just a few thousandths on the final cut and use the power feed. Be careful clearing the ribbon-like chips. They are very sharp. Do not clear the chips while the spindle is turning. After facing, there is a very sharp edge on the part. Break the edge with a file.

3.7.7 Parting

A parting tool is deeper and narrower than a turning tool. It is designed for making narrow grooves and for cutting off parts. When a parting tool is installed, ensure that it hangs over the tool holder enough that the holder will clear the workpiece (but no more than that). Ensure that the parting tool is perpendicular to the axis of rotation and that the tip is the same height as the center of the part. A good way to do this is to hold the tool against the face of the part. Set the height of the tool, lay it flat against the face of the part, then lock the tool in place. When the cut is deep, the side of the part can rub against sides of the groove, so it's especially important to apply cutting fluid. In this clip, a part is cut off from a piece of stock.

3.7.8 Drilling

A lathe can also be used to drill holes accurately concentric with the centerline of a cylindrical part. First, install a drill chuck into the tail stock. Make certain that the tang on the back of the drill chuck seats properly in the tail stock. Withdraw the jaws of the chuck and tap the chuck in place with a soft hammer.

Move the saddle forward to make room for the tailstock. Move the tailstock into position, and lock the it in place (otherwise it will slide backward as you try to drill). Before starting the machine, turn the spindle by hand. You have just moved the saddle forward, so it could interfere with the rotation of the lathe chuck. Always use a center drill to start the hole. You should use cutting fluid with the center drill. It has shallow flutes (for added stiffness) and doesn't cut as easily as a drill bit. Always drill past the beginning of the taper to create a funnel to guide the bit in. If the drill bit squeaks, apply solvent more often. The drill chuck can be removed from the tail stock by drawing back the drill chuck as far as it will easily go, then about a quarter turn more. A pin will press the chuck out of the collet.

3.8 Milling Machine

Milling machines are versatile machines that are usually used to machine flat surfaces, but can also produce irregular surfaces. They can also be used to drill, bore, cut gears, and produce slots. The type of milling machine in the Engineering Design machine shop is a vertical spindle machine.

A milling machine removes metal by rotating a multi-toothed cutter that is fed into the moving workpiece. The spindle can be fed up and down with a quill feed lever on the head. The bed can also be fed in the x , y , and z axis manually. Once an axis is located at a desired position and will no longer be fed, it should be locked into its position. Most milling machines are equipped with power feed for one or more axes. Power feed is smoother than manual feed and, therefore, can produce a better surface finish. Power feed also reduces operator fatigue on long cuts. On some machines, the power feed is controlled by a forward-reverse lever and a speed control knob.

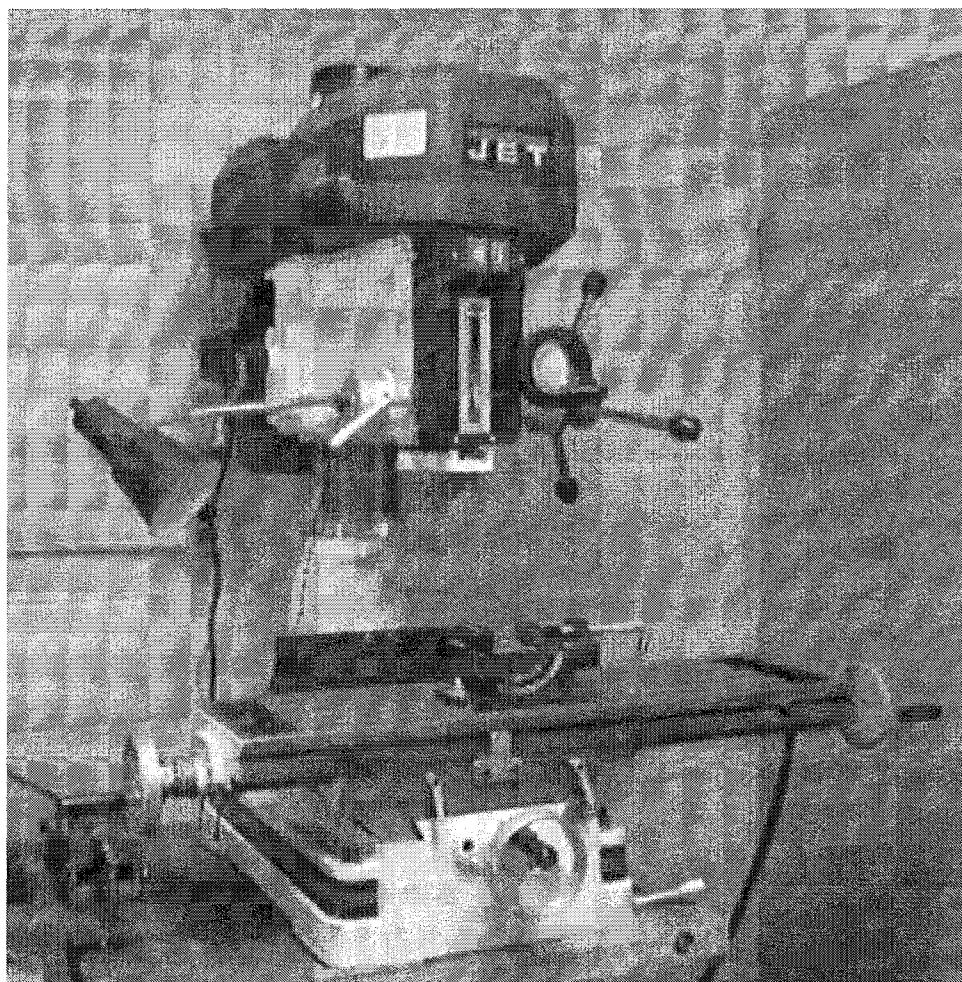


Figure 3.13: The milling machine in the Design workshop

3.8.1 Squaring the Vise

The work on a milling machine is often held in a vise clamped onto the bed. To make features aligned with the edges of the stock, it is necessary to align the vise with the feed axes of the mill. To do this, mount the vise on the bed and secure it with T-bolts, but only lightly so as to permit adjustment of the orientation of the vise. Mount a dial indicator in the spindle of the machine with the probe facing away from you. Lower the spindle and run the bed of the table back until the fixed jaw of the vise is in contact with the indicator and further until the indicator registers one half of a revolution. Set the bezel to zero. Use the cross feed to run the indicator across the face of the vise. If the vise is squared, the indicator will remain at zero. If the dial indicator does not read zero, tap lightly with a soft hammer to realign the vise reduce the indicator reading to half of its previous value. Iterate this procedure until the dial indicator reads zero through the full travel across the face of the vise. Tighten down the T-bolts be careful not to change the vise orientation. Recheck the alignment of the vise.

3.8.2 Tilting the Head

The head of a vertical milling machine can be tilted from side to side and from front to back. This allows for versatility of the machine, but these adjustments can drift. Occasionally, one should check and adjust the head so that the spindle will be normal to the plane of the table. Install a dial indicator into the spindle so that the dial is offset at least six inches from the axis of the spindle and the indicator probe is facing down. Lower the spindle until the dial indicator contacts the table then registers about one half of a revolution. Set the dial indicator toward you and set the bezel to zero. Rotate the spindle

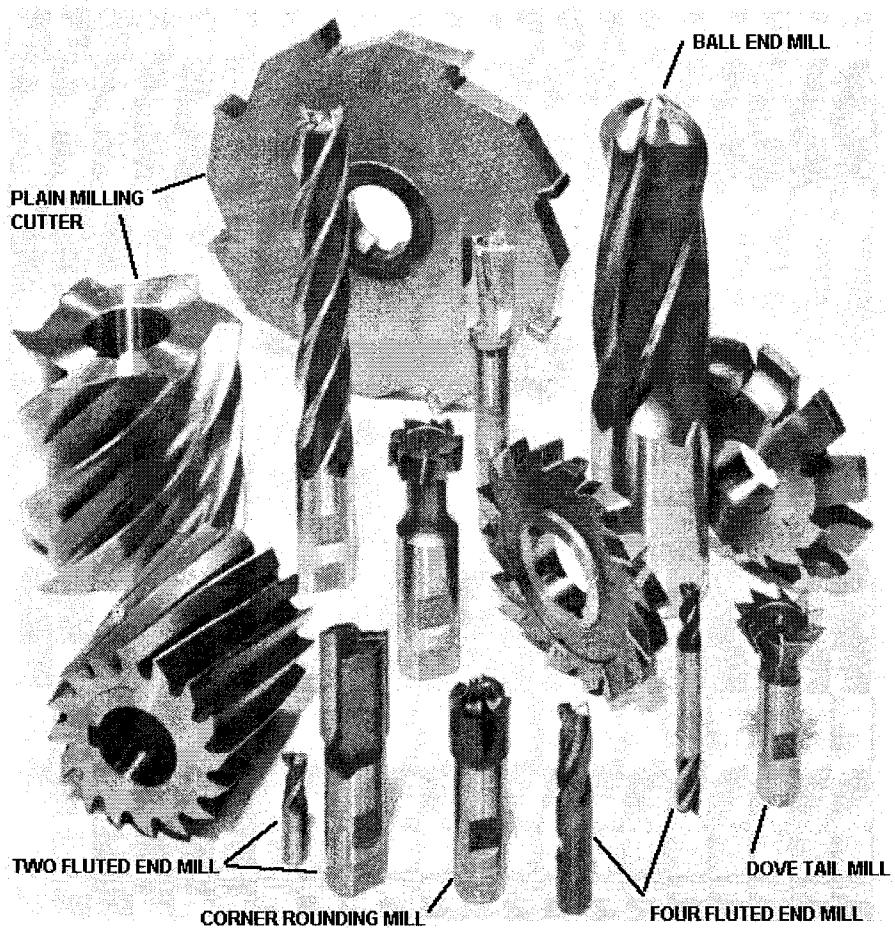


Figure 3.14: Different types of Milling Cutters

by hand 180 degrees. If the dial indicator still reads zero, the spindle is aligned front to back. If not, adjust the head until the dial reads half of the original reading and iterate the entire process until the error falls within acceptable limits. Repeat the process with the dial displaced left and right to align the head side to side.

3.8.3 Types of Milling Cutters

In vertical mills, milling cutters with solid shafts are usually used. Milling cutters with keyed holes are predominantly for use in horizontal mills. End mills are designed for cutting slots, keyways and pockets. Two fluted end mills can be used to plunge into work like a drill. End mills with more than two flutes should not be plunged into the work. Ball end mills can produce a fillet. Formed milling cutters can be used to produce a variety of features including round edges. Figure 3.14 shows a variety of cutting tools.

3.8.4 Installing Milling Cutters

End mills can be held by the spindle in several ways; a few of the ways are shown in Figure 3.15. On most machines, a draw bar is used to pull a spring collet into a taper in the spindle. To remove a tool, move the quill to the highest position and lock it in place. Then, engage the brake while loosening the draw bar with a wrench. Ensure that the draw bar's threads are still engaged in the collet. Tap on the end of the draw bar to release the collet from the spindle. If the threads of the draw bar are not engaged, the milling cutter will fall, and could be damaged. Finally, unscrew the drawbar from the collet. To install a tool, place the desired milling cutter in a collet that fits the shank of the cutter. Insert the collet into the spindle. Ensure that the key way on the collet mates properly with the key in the spindle. While holding the tool with one hand, start the threads of the draw bar into the collet by hand. Use a wrench to tighten the drawbar down with one hand while holding the brake.

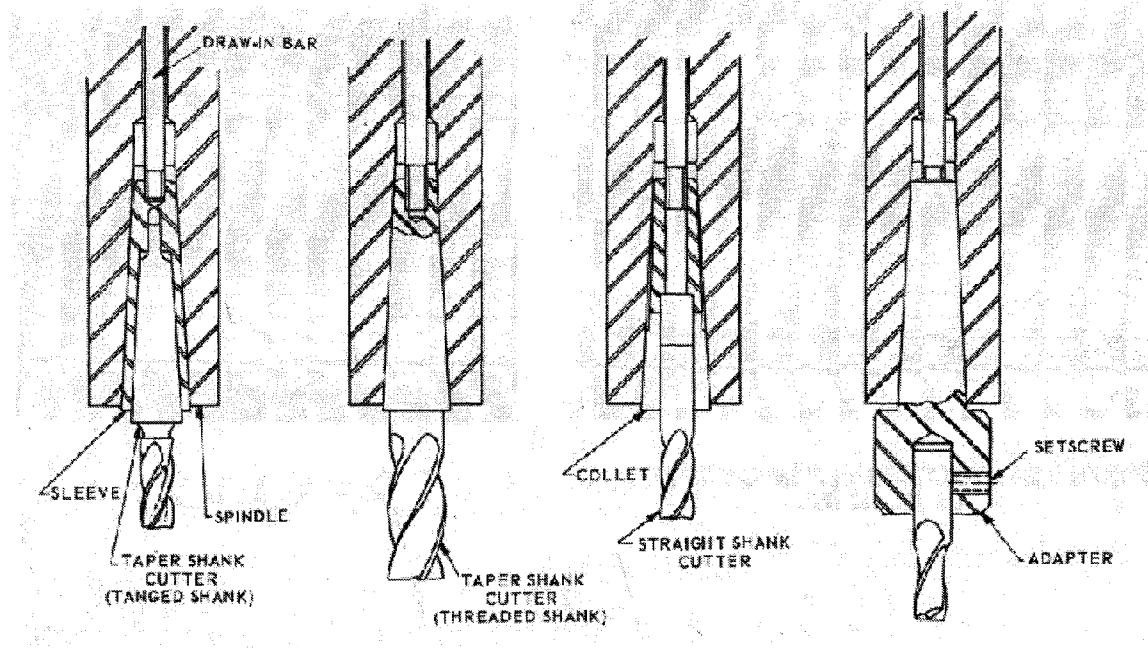
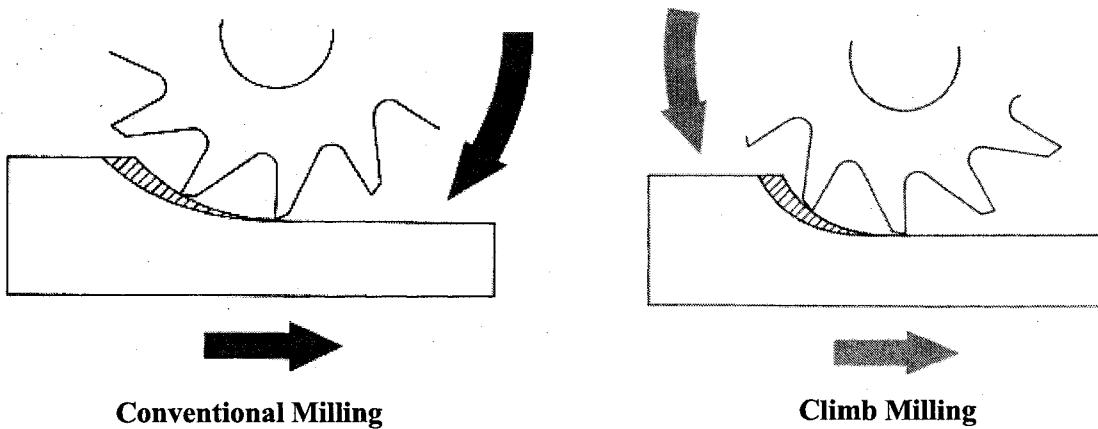


Figure 3.15: Methods of retaining the end mill



3.16: Two methods of milling

3.8.5 Climb vs. Conventional Milling

When milling, one should be aware of the difference between conventional, and climb milling. In conventional milling, the workpiece is fed into the rotation of the cutter. This type of cut requires lower forces and is preferred for roughing cuts. In climb milling, the work moves with the rotation of the cutter. This produces a better finish. However, this method is not recommended if the workpiece cannot be held securely or cannot support high forces. Figure 3.16 illustrates these two methods.

3.8.6 Calculating Speeds and Feeds

Cutting speed refers to the speed at which the tool point of the cutter moves with respect to the work measured in feet per minute. Feed is the rate at which the work moves into the cutter measured in feed per tooth revolution. Feeds and speeds affect the time to finish a cut, tool life, finish of the machined surface and power required of the machine. The cutting speed is mostly determined by the material to be cut and the material of the tool. To find the right speed for any task, refer to the Machinery's Handbook or other reference. To calculate the proper spindle speed, divide the desired cutting speed by the circumference of the tool expressed in feet. The feed rate depends on the width and depth of cut, finish desired and many other variables. To calculate the desired feed setting from the feed rate, multiply feed per tooth per revolution by number of teeth and rpm of the spindle.

3.8.7 Squaring Stock

To create a square corner on a part, first orient an already finished edge vertically in the vise and clamp lightly onto the part. Set a square block against the finished edge and the bottom of the vise. Lightly tap the part with a plastic hammer to align it with the square. Clamp the vise down securely. Now the top edge of the part is ready to be milled to horizontal.

3.8.8 Face Milling

It is often necessary to create a flat face on a large part. This can be done best with a facing cutter. Select a cutter about one inch wider than the workpiece so that the facing can be accomplished in one pass.

3.8.9 Milling Slots

End mills are designed to cut square slots. They will produce a slot to within two one-thousandths of an inch in one pass. If greater accuracy is required, use an end mill a little smaller than the desired slot. Measure the slot produced, and open it to the desired dimension with a second pass.

3.8.10 Work Holding

A work piece can be set up easily when the desired features are parallel with or perpendicular to the workpiece edges. When the features are at an angle to the edges, more ingenuity is required. In these cases, an angle plate is used to set the position of a vise within a vise. Thus a slot can be milled into a workpiece at any desired angle. To hold round stock more securely in a vise, use a v-block. The work can be held vertically or horizontally. Some parts do not fit well into a vise. These parts can be secured directly to the bed of the machine with hold down clamps. It is a good practice to create a gap between the bed and the work with parallels. The clamps should be tilted down slightly into the work

Round stock often cannot be held securely in the vise without damaging the work. A collet block is designed to hold a round stock. Square collet blocks allow the part to be indexed to put in features at 90-degree increments. To mill features at 60-degree increments, use a hexagonal block. To create circular features on a mill, a rotary table can be installed onto the bed. The table allows the workpiece to be rotated. A dial indicator allows precise control of the angle of rotation.

3.9 Grinding and Buffing

The primary purpose of a grinding wheel is to sharpen tools (e.g. drill bits). The hard abrasive of the wheel is made for removing very hard materials like high-speed steel. Never grind on the side of the wheel. Also, never grind a soft material such as Aluminum. The material will coat the wheel and prevent the abrasive from working properly. If Aluminum is pressed against the wheel for too long the wheel could heat excessively and explode. If the grinding wheel becomes coated with metal, dress the wheel prior to use.

A deburring wheel is made of a material similar to a pot scrubber; a mesh of abrasive fibers held together with adhesive. It is good for intermediate polishing. Even very hard materials such as tool steel can be removed with a deburring wheel. A tool that has been smoothed out in this manner can be used to burnish a part in the lathe to achieve a very fine finish. When working with the deburring wheel, it is critical to maintain proper orientation of the part. If a corner catches on the wheel, it can be thrown down forcefully. The part will probably be marred and you could easily be hurt. This applies to buffing and grinding as well.

A buffing wheel is made of cloth. By itself it is not abrasive. To make it work properly, abrasive must be applied. Abrasives come in a tube and are suspended in wax. The tube is pressed onto the wheel as it spins, melting the wax, which helps the abrasive adhere to the cloth. To buff a part, hold it against the wheel with a firm pressure. Keep moving the part about and gradually lighten the pressure as the finish gets finer and finer.

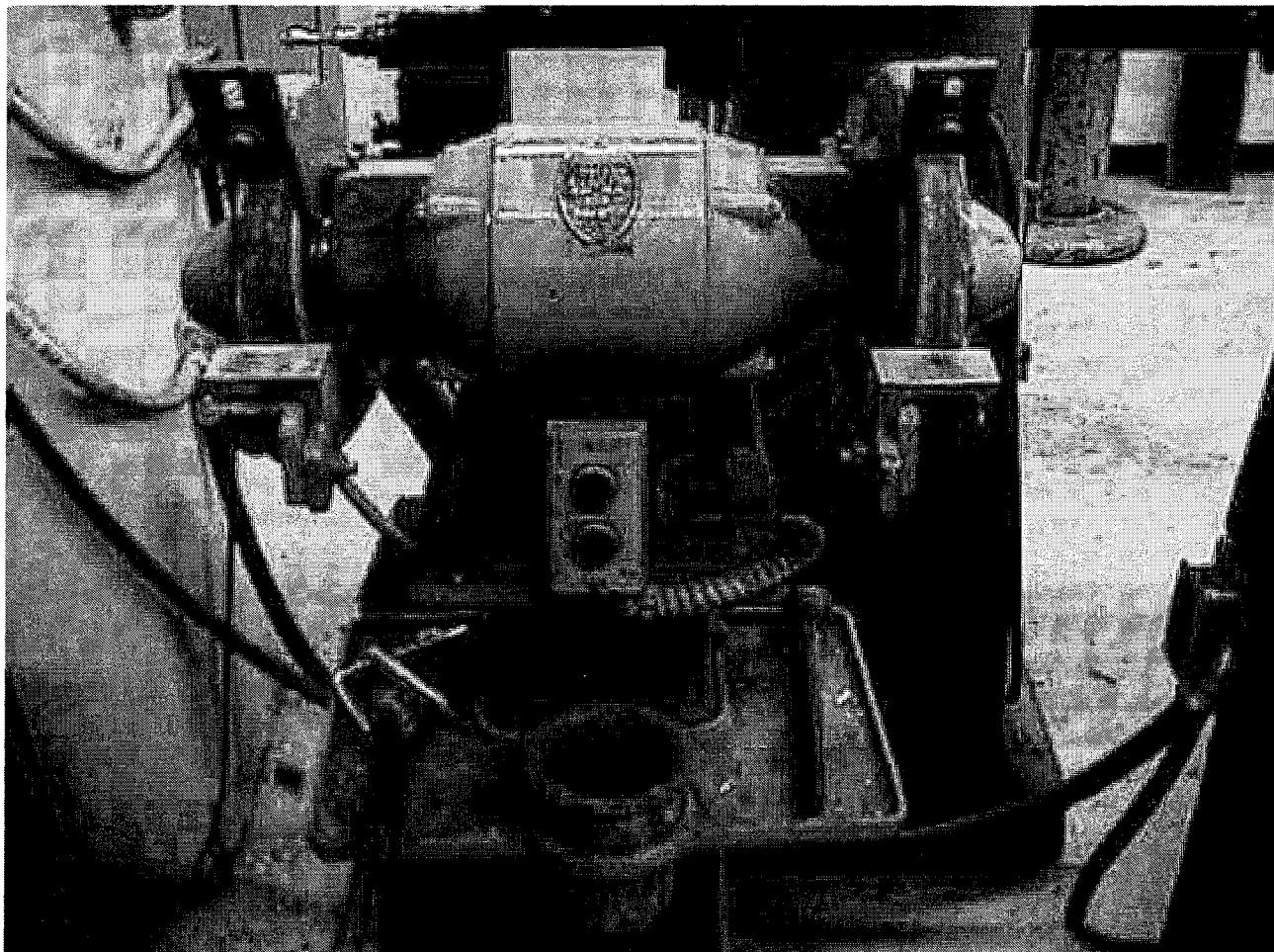


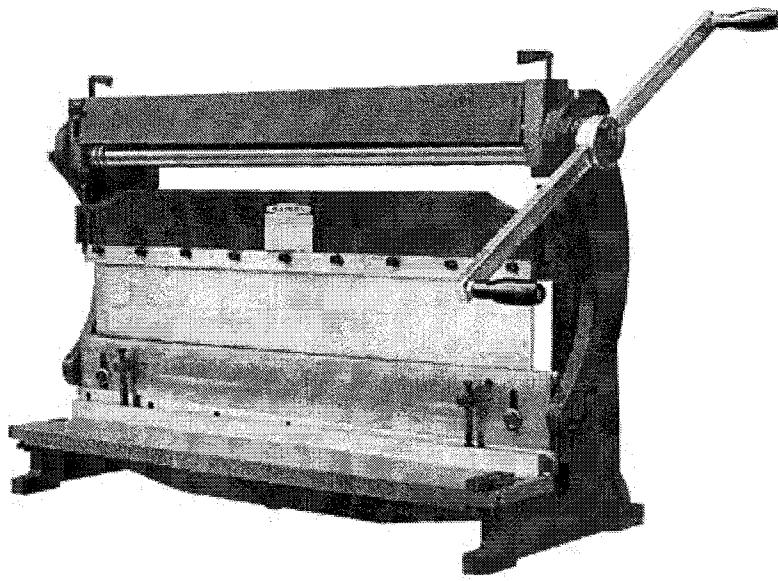
Figure 3.17: A grinding machine

3.10 Bending Brake

A bending brake is a simple device employed to bend thin materials. Adjustments are provided to compensate for various thickness of sheer. When laying out for a bend in a certain thickness of material, bending allowance must be taken into consideration.

When bending sheet metal or brass, add from 1/3 to 1/2 of the thickness of the stock, for each bend, to the some of the inside dimensions of the finished piece, to get the length of the straight bland.

Care must be exercised when handling sheet metals for they have extremely sharp edges. Leather gloves can be used when handling larger sheets. Keep people away to assure adequate clearance around the machine. And, do not exceed the machine capacity.



3.18: A bending and cutting machine

Chapter 4

Design

4.1 Fundamentals	1
4.1.1 System (in the context of engineering design).....	1
4.1.2 Engineering (<i>of a system</i>)	1
4.1.3 Design V-Diagram.....	2
4.1.4 Systems Design.....	3
4.1.5 Involved Disciplines in Systems Design	4
4.1.6 Systems Design Goals	4
Customer	4
4.1.7 Design Specifications	5
4.1.8 Design Process.....	6
Recognition of Need.....	7
Preliminary Design.....	7
Detailed Design.....	7
Design Refinement.....	7
Design Evaluation	7
Production Planning	7
Documentation	7
4.1.9 Design Process Details	8
<i>Conceptual Design</i>	8
<i>Preliminary Design</i>	9
<i>Detailed Design</i>	9
<i>Production Planning</i>	10
<i>Documentation</i>	10
4.2 Conceptual Design	11
4.2.1 Creation of Design Ideas	11
4.2.1.1 Thinking Process.....	11
4.2.1.2 Experimentation.....	12
4.2.1.3 Drawing	13
4.2.1.4 Reading.....	15
4.2.1.5 Writing.....	15
4.2.1.6 Analysis	15
4.2.1.7 Selecting Ideas	17
4.2.2 Basic Design Principles	18
4.2.2.1 Saint-Venant's Principle	18
4.2.2.2 Abbe's Principle	19
4.2.2.3 Reciprocity Principle	20
4.2.2.4 Self Principles	21
4.2.2.5 Stability Principle	21
4.2.2.6 Superposition Principle.....	21
4.2.2.7 The Golden Rectangle.....	22
4.2.2.8 Parallel Axis Theorem	22
4.2.2.9 Accuracy, Repeatability, and Resolution	22
4.2.2.10 Robustness Principle.....	23
4.2.2.11 Exact Constraint.....	25
4.2.2.12 KISS and MISS.....	25
4.3 Design Methodologies	26

4.3.1 Design Attributes.....	26
4.3.1.1 Design Objectives and Constraints.....	26
4.3.1.2 Design Functions	26
4.3.1.2.1 Methods of Identifying Design Functions	27
4.3.1.3 Metrics.....	29
4.3.1.4 Performance Specifications.....	30
4.3.2 Quality Function Deployment.....	32
4.3.3 Design for Optimization.....	36
4.3.3.1 Lagrange Multiplier Method.....	36
4.3.3.2 Search Methods	36
4.3.3.3 Linear Programming.....	39
4.3.3.4 Multicriterion Optimization.....	40
4.3.4 Design for Reliability	41
4.3.4.1 Component Reliability.....	41
4.3.4.2 System Reliability	43
4.3.4.3 Optimization of System Reliability	45
4.3.4.3.1 Reliability Optimization for A Given Cost Constraint.....	45
4.3.4.3.2 Cost Minimization for A Given Reliability Constraint.....	48
4.4 Communication in Design	50
4.4.1 Engineering Proposals.....	50
4.4.1.1 Process.....	51
Concept.....	51
Program	51
Expenses	51
4.4.1.2 Format	51
4.4.1.2.1 Title.....	51
4.4.1.2.2 Executive Summary.....	52
4.4.1.2.3 Introduction.....	52
Statement of Need / Problem	52
Goals and Objectives	53
Background and Survey	53
4.4.1.2.4 Technical Body	53
Methodology – Statement of Work (SOW)	53
Schedule	54
Budget.....	54
4.4.1.2.5 Conclusion	54
4.4.1.2.6 Appendices	55
4.4.2 Engineering Notebook.....	55
4.4.3 Technical Report	56
4.4.3.1 Report Covers	56
4.4.3.2 Report Preamble	57
4.4.3.3 Body of the Report	57
4.5 Resources	59

4.1 Fundamentals

4.1.1 System (in the context of engineering design)

A collection of hardware, software, people, facilities, and procedures united by some *apparent* interrelations and organized to accomplish some *common objectives*.

Objectives : typically address cost, schedule, and performance :

cheaper, faster, and better (more reliable, durable, secure, beautiful, ...)

System Definition with emphasis on boundaries

"The closed system has rigid, impenetrable boundaries, whereas the open system has permeable boundaries ... Boundaries are the demarcation lines or regions for the definition of appropriate system activity, for admission of members into the system, and for other imports into the system."

(Kast, F. E., Rosenzweig J.E., "The modern view: a systems approach", *Systems Behaviour*, Open University Press, London, 1972.)

"A system is an assemblage of objects, principles, or facts, united by some form of regular interaction or interdependence into an organized whole."

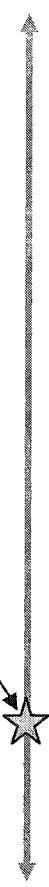
(Roe P.H., Soulis G.N., Handa V.K., "The Discipline of Design", University of Waterloo, 1992.)

"We call a thing a system when we wish to express the fact that the thing is perceived/conceived as consisting of a set of elements, of parts, that are connected to each other by at least one discriminable, distinguishing principle."

(Jordan N., "Some thinking about System", *Themes in Speculative Psychology*, London,

System Definition with emphasis on interrelations

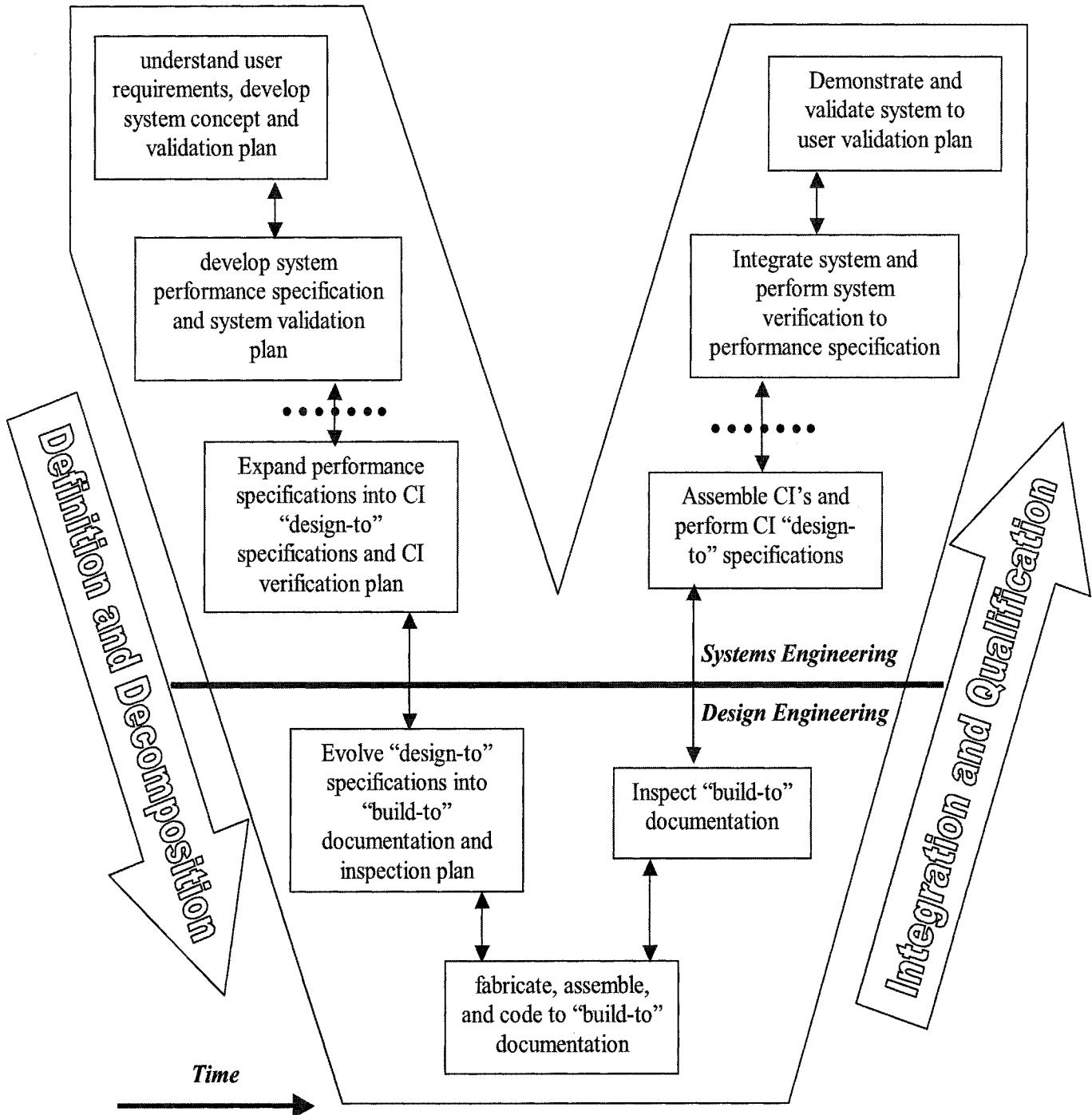
We are here.



4.1.2 Engineering (of a system)

Discipline for developing, matching, and trading off requirements, functions, and alternate system resources to achieve a cost-effective, life-cycle-balanced product based upon the needs of the customer.

4.1.3 Design V-Diagram



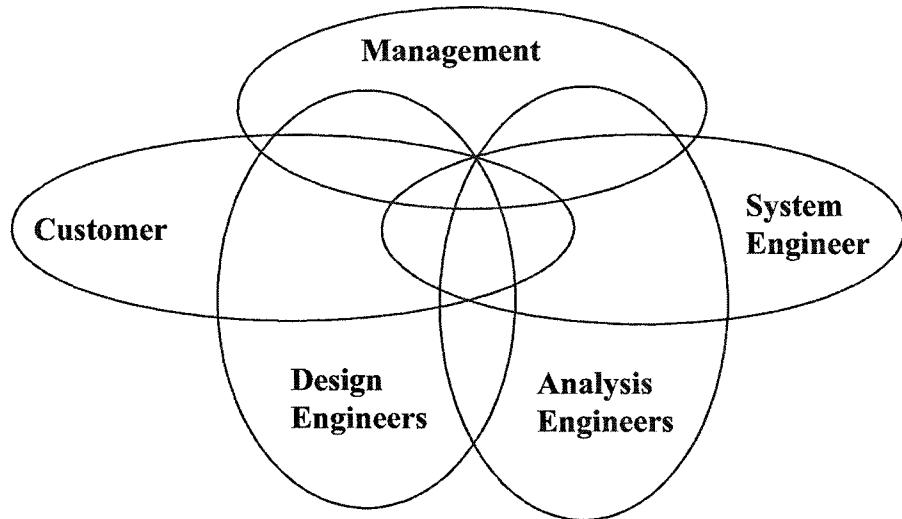
4.1.4 Systems Design

System design is the invention, disposition, or modification of the forms, parts or details of a *system* according to a plan to meet desired objectives or specifications. Five steps of systems design are:

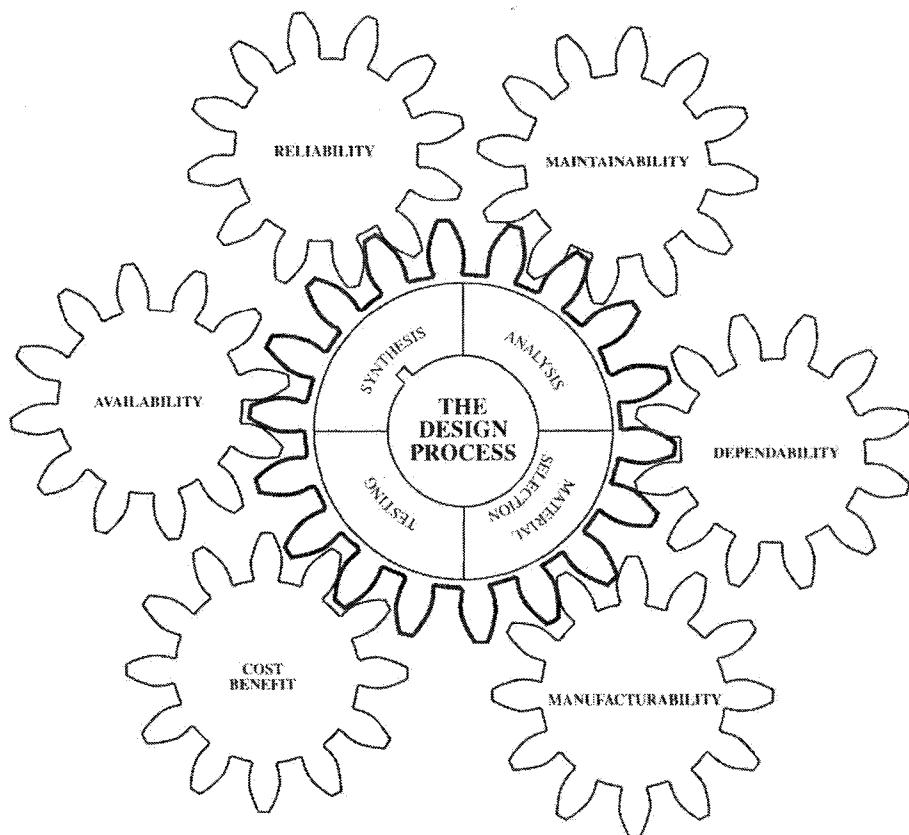
- Define the system-level design problem and solution (Conceptual Design)
- Develop the system functional architecture (Preliminary Design)
- Develop the system physical architecture (Full-Scale or Detailed Design)
- Develop the interface architecture (Integration)
- Develop the qualification process (Qualification)

Development Phase	Examples of Decisions during Systems Design
Conceptual Design	<ul style="list-style-type: none">• Should a conceptual design effort be undertaken?• Which system concept (or mixture of technologies) should be the basis of the design?• Which technology for a given subsystem should be chosen?• What existing hardware and software can be used?• Is the envisioned concept technically feasible based on cost, schedule, and performance requirements?• Should additional research be conducted before a decision is made?
Preliminary Design	<ul style="list-style-type: none">• Should a preliminary design effort be undertaken?• Which specific physical architecture should be chosen from several alternatives?• To which physical resource should a particular function be allocated?• Should a prototype be developed? If so to what level of reality?• How should validation and acceptance testing be structured?
Detailed Design	<ul style="list-style-type: none">• Should a full-scale design effort be undertaken?• Which configuration items should be bought instead of manufactured?• Which detailed design should be chosen for a specific component given that one or more performance requirements are critical?
Integration	<ul style="list-style-type: none">• What is the most cost-effective schedule for implementation activities?• What issues should be tested?• What equipment, people, facilities should be used to test each issue?• What models of the system should be developed or adapted to enhance the effectiveness of integration?
Qualification and Product Refinement	<ul style="list-style-type: none">• How much testing should be devoted to each subsystem?• What adaptive testing should be planned for each subsystem?• Should a product improvement be introduced at this time?• Which technologies should be the basis of the product improvement?• What redesign is best to meet some clearly defined deficiency in the system?• How should the refinement of existing systems be implemented given schedule, performance and cost criteria?

4.1.5 Involved Disciplines in Systems Design



4.1.6 Systems Design Goals



4.1.7 Design Specifications

Design specifications are a set of criteria that the designer uses as benchmarks to evaluate the progress toward a design, as well as the performance of the final design product. These specifications are the designer's translation of the client's needs or requirements into the product being designed. Three major categories of design specifications are:

Prescriptive Specifications: specify the quantitative attributes of the product such as weight, geometry, etc.

Example: "A step on a ladder is safe if it is made from Grade A fir, has a length that does not exceed 50cm, and is attached in a full-width groove slot at each end."

Procedural Specifications: identify specific procedure for calculating the product attributes or performance.

Example: "A step on the ladder is safe if its maximum bending stress is computed from

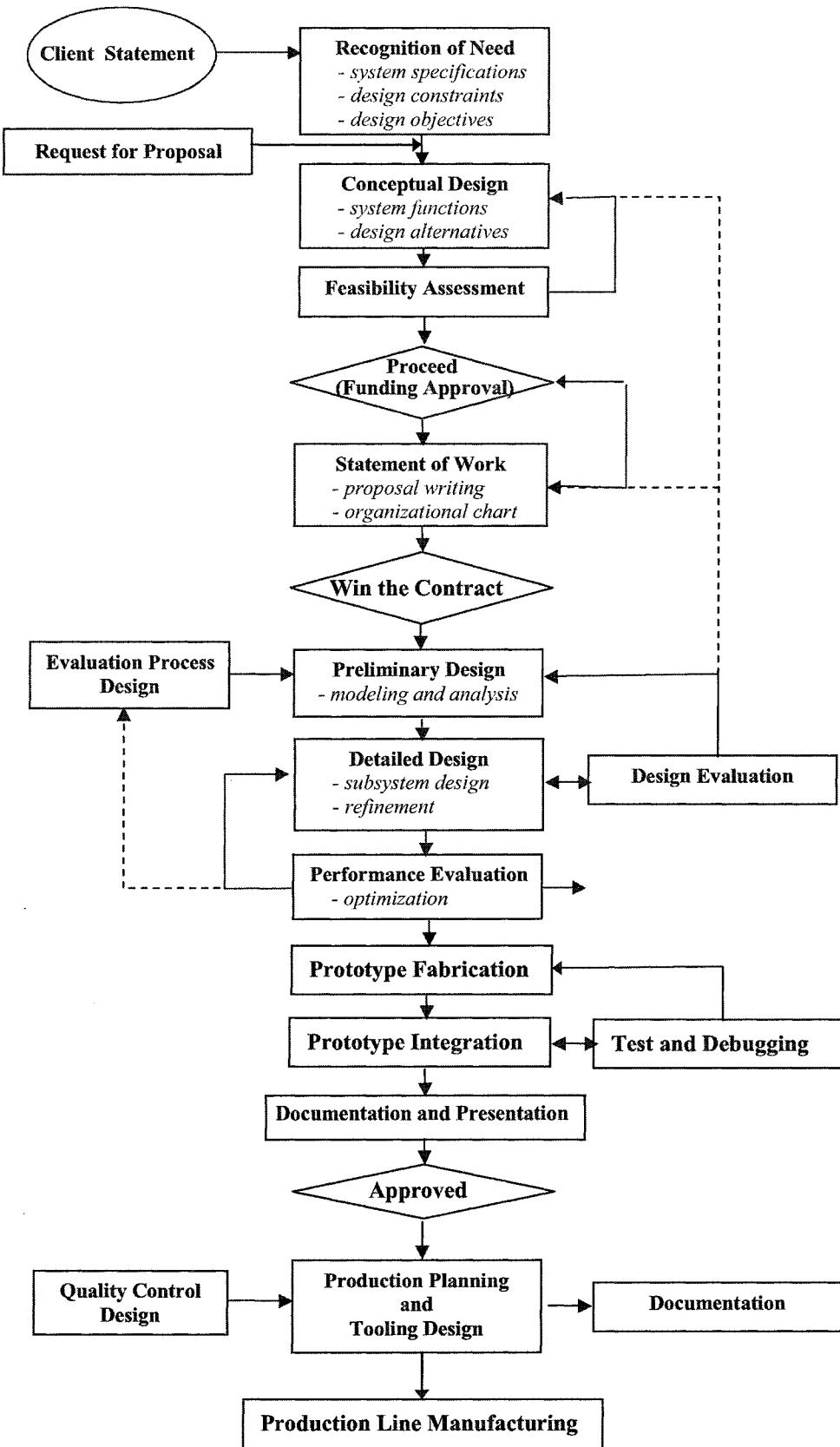
$$\sigma_{\max} = 0.75 \left[\frac{Mc}{I} \right],$$

and is such that σ_{\max} does not exceed σ_{allow} .

Performance Specifications: characterize the desired performance.

Example: "A step on a ladder is safe if it will support a 300kg gorilla."

4.1.8 Design Process



Example: Design of A “Safe” Ladder

Recognition of Need

- How is the ladder used?
- What colour(s) should it be?
- How much should it cost?
- How does the “safety and health” regulations define “safety”?
- Must the ladder support someone carrying something?
- How high should someone on the ladder be able to reach?
- How much load should the ladder support?
- What is the allowable load on a step?

Conceptual Design

- Should the ladder be portable?
- Can the ladder lean against a supporting surface?
- Could the ladder be a stepladder or an extension ladder?
- Could the ladder be made of wood, aluminum, fiberglass, etc.?

Preliminary Design

- What is the maximum stress in a step supporting the nominal load?
- How does the bending deflection of a loaded step vary with the material of which the step is made?
- How does the ladder weight affect the internal stress?

Detailed Design

- What is the code for the selected material?
- What are the dimensions of the ladder?
- What is the geometry of the steps?
- How are the step connections?

Design Refinement

- Is there a more economic solution?
- Is there a more efficient design (e.g., less material, more strength, etc.)?

Design Evaluation

- Can the ladder carry the nominal load?
- Can each step carry the allowable load?
- Can someone on the ladder reach the specified height?
- Are all the safety and health regulation satisfied?

Production Planning

- What are the tolerances for the ladder components?
- What is the manufacturing procedure for making the ladder?
- What tools/machines are required?

Documentation

- What are the justifications for this design?
- What was the design process?
- How is the ladder assembled?
- How is the ladder used?

4.1.9 Design Process Details

Recognition of Need

This phase is devoted to clarifying the objectives set out by the client and gathering the information needed to develop an engineering statement of what the client wants and how to achieve them.

Input: *client's statement*

Sources: *literature on state-of-the-art experts
codes and regulations*

Tasks: *clarifying design objectives
establishing user requirements
identifying constraints
establishing system specifications*

Methods: *objectives tree
pairwise comparison chart
weighted objectives tree
function-means tree
functional analysis
requirements matrix*

Means: *literature review
user surveys and questionnaires
structured interviews*

Output: *revised problem statement
detailed (weighted) objectives
design constraints
user requirements*

Conceptual Design

The goal of the *conceptual design* stage is to generate concepts or schemes of candidate design solutions.

Input: *revised problem statement
detailed (weighted) objectives
constraints
user requirements
functions*

Sources: *competitive products*

Tasks: *establishing design functions
generating design alternatives*

Methods: *performance specification method
quality function deployment (QFD)*

morphological chart

- Means: *brainstorming*
Analogy analysis
reverse engineering (dissection)
- Output: *conceptual design(s) or scheme(s)*
design functions

Preliminary Design

The goal of the *preliminary design* phase is to identify the principal attributes of the design concepts or schemes.

- Input: *conceptual design(s) or scheme(s)*
design specifications and functions
- Sources: *heuristics (rules of thumb)*
simple models
known physical relationships
- Tasks: *model, analyze conceptual designs*
test, evaluate conceptual designs
- Methods: *conventional analytic modeling*
black-box modeling
- Means: *laboratory experiments*
prototype development
simulation and computer analysis
proof-of-concept testing
- Output: *a selected design*
test/evaluation results

Detailed Design

The goal of the *detailed design* phase is to define the final design in details and to refine it if possible.

- Input: *selected design*
test/evaluation results
- Sources: *design codes*
handbooks
local laws and regulations
supplier's component specifications
- Tasks: *define and optimize the elected design*
- Methods: *discipline-specific CAD*
- Means: *formal design reviews*

public hearings (if applicable)
beta testing

Output: *detailed fabrication specifications*
final design review for client

Production Planning

The goal of the *production planning* phase is to specify the tolerances and identify means and procedure of manufacturing the final product.

Input: *detailed fabrication specifications*

Sources: *machine handbooks*
manufacturing handbooks
Suppliers' catalogues
standard codes

Tasks: *define the tolerances and manufacturing procedure*

Methods: *Monte-Carlo simulation*
heuristics

Means: *engineering drawings*

Output: *system tolerances*
manufacturing procedure
engineering drawings

Documentation

The *design documentation* phase is devoted to documenting the fabrication and manufacturing specifications and their justification.

Input: *fabrication specifications*
system tolerances and manufacturing procedure
engineering drawings

Sources: *feedback from clients and users*
Required deliverables
supplier's component specifications

Tasks: *document the completed design*

Output: *final report to client containing*
fabrication specifications and their justification
tolerances and manufacturing procedure
engineering drawings
software code for analysis
assemblage instructions

4.2 Conceptual Design

4.2.1 Creation of Design Ideas

The key success of every design is the core idea(s) created by the designer(s). Generating *new* ideas for daily design problems might seem close to impossible, but by applying effective techniques, some of which are addressed below (adopted from MIT's 2.007 pergatory.mit.edu/2.007/), there is always a possibility of modification of the existing solutions or creation of novel solutions for the design problem.

4.2.1.1 Thinking Process

Good ideas come from good thinkers. In the design context, thinking can be defined as specific processes, some of which are listed below, that can help generating design solutions. Combinations of these processes can be used as catalysts for other methods of conceptual design coming in sequel.

Systematic Variation: This thinking process is simply to consider ALL the possibilities for the system to be designed:

- Energy: How can it be applied, generated, stored?
 - Mechanical: springs, flywheels...
 - Hydraulic: piston, bladder, reservoir, propeller...
 - Electrical: line source, Battery, capacitor, magnet, optical...
- Material:
 - State: solid, liquid, gas
 - Behavior: rigid, elastic, plastic, viscous
 - Form: bar, sheet, powder, ...
- Motions:
 - Type: fixed, linear, rotary
 - Nature: uniform, non-uniform, transient
- Direction & Magnitude
- ...

AND, consider all combinations of the above.

Systematic thinking would be more effective if the following recommendations are followed:

- In your thinking process continually ask “WHAT?”, “WHY?”, “HOW?”, “WHERE?”, “WHO?”, ...
- Do leisurely things (e.g., long walks) that you know would inspire creative thinking.
- “Briefly” look at what other people have created. Look in your home, stores, www, patents, etc., but don’t go much into details.
- Get out of *traffic* and take alternate routes.
- Sketch ideas and the ideas’ principal components (coming in sequel).
- Cut out the principal components and pretend they are modular elements. Like toy building blocks, try different combinations of components to make different products.
- Pit one idea against another and imagine strategies for winning. Take the best from different ideas and evolve them into the best 2 or 3 ideas.

- Create an “Infomercial Sheet” or “Press Release” for your favorite ideas. Ordinary people should be able to read the press release and fully understand your idea without you having to explain it to them. These sheets will be shared with your teammates in the next stages.

Rohrbach’s Method: This technique is a team effort for generating new ideas, and is usually performed after individual thinking (above) and before group brainstorming (next). In this method, N (usually six) people circulate their “Press Releases” to the other N-1 (five) teammates for comments. No talking is allowed. People make written constructive comments on each other’s papers, until everyone has read everyone else’s press releases. This technique creates a collective mind, so that everybody knows what everyone else has been thinking. The group mind then works together in a more efficient manner when the team brainstorming is performed.

Brainstorming: Brainstorming helps individuals or teams solve personal creativity deadlocks, and ensure nothing has been overlooked. This technique is to discard any sort of constraints in the design problem, and generate *as many ideas as possible*. Brainstorming has its maximum effect when performed by a team of designers. However, the following group personality factors must be considered:

- Shy individuals getting run over.
- Aggressive individuals always driving.
- An individual’s personality often has nothing to do with creativity.

Psychologists have established the following rules to make the process work effectively:

- No more than 6 people.
- One person takes careful notes.
- Initially let everyone voice her/his suggestions, then distill ideas.
- No purely negative statements.
 - Simultaneously point out risks and possible countermeasures.
 - Not allowed “that sucks!”
 - Allowed: “A low pressure region seems to exist, so lets divert flow....”

Forward Steps: In this thinking process, you start with an idea, and vary it in as many ways as possible to create different ideas, until each gets to the end goal. This method is also called the *method of divergent thought*.

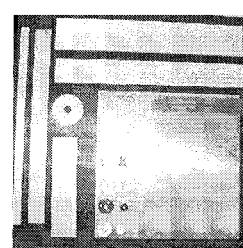
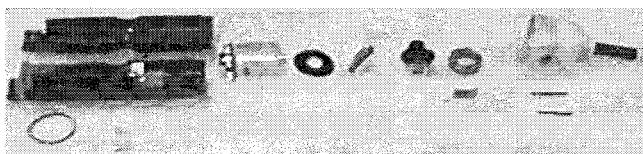
Backwards Steps: Start with the end goal and work backwards along as many paths as possible till you get to the beginning.

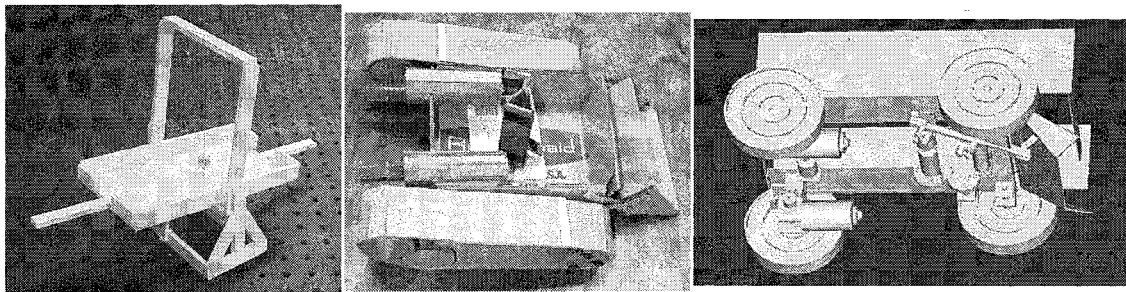
Nature’s Way: How would nature solve the problem?

Minimum Constraints: What is the minimal number of restrictions dictated by the customer and physics?

4.2.1.2 Experimentation

Playing with Parts: Lay out all the materials (physical or information sheets) you have in front of you and play with them, let them talk to you, what are their limits, how have others used them, etc. Place components on various places to obtain a physical feel of how they might fit (see figure below.) With a “competing” partner “drive” imaginary machines with your hands to represent possible motions. Mock fantasy competitions can highlight strategies.





Sketch Models: Sketch models are made from simple materials (e.g., cardboard, foam, hot-melt-glue, tape, string) and they allow you to literally “play” with possible strategies. Later, when you have a concept developed, they enable you to “test drive” your machine concept around the table. In the “real world” where design solutions are often very complex, sketch models are still important aids for “proof of concept”. They can be invaluable sales tools. Solid models and kinematic motion simulation packages, which also check for interferences and give forces and reactions, are the mainstay of the modern design world. Some sketch models are shown in figure above.

It is advised to create sketch models of “best” strategies and concepts and play them off against each other on the table. While you are creating a sketch model, create a systematic list of systems in each design, and their potential functional requirements. This will help you develop *Bench-Level Experiments*. Identify the physics, and strategy and concept issues (design parameters) that emerge. Very quickly, this will allow you to determine if you can realize a design with the hardware available. For example, what if your strategy relies on a force of 100 N? You might choose a different strategy, or you might be led to use a lead-screw and not worry about using a winch.

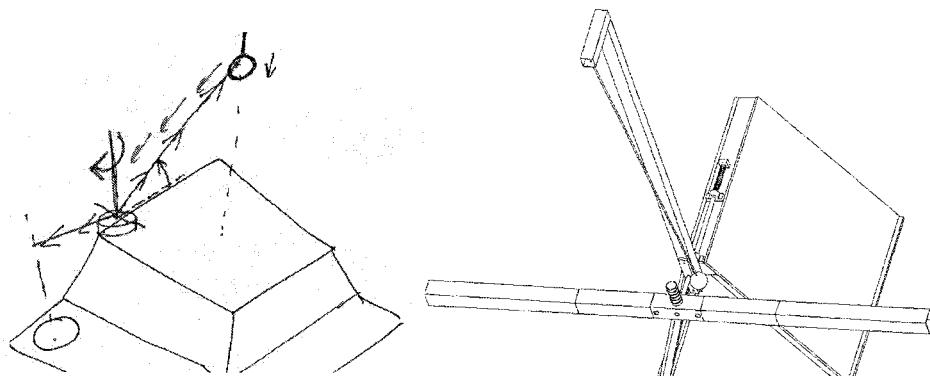
NOTE: Many real-world designs can be sketch-modeled, but some cannot, so use the technique when it is appropriate. Solid models are actually often made faster and more realistic than a sketch model. Finite element models and kinematic synthesis packages can provide “digital” feel of system performance. *Virtual Reality* tools will soon give designers the *dynamic* feel of their on-screen designs.

Bench-Level Experiment: Bench-level experiments are quick and easy tests performed during the strategy and concept phases of development to investigate force, friction, speed, etc. They are vital parts of the design process. Although analysis is potentially the quickest way to verify an idea, You might end up to *analysis paralysis* due to uncertainty or lack of experience. Analysis paralysis is most often relieved by a simple bench-level experiment.

For risky ideas, bench-level experiments can end up developing a bench-level prototype that is an actual, ready-to-use module of the final design. It often illustrates what works and what must be fixed, and whether or not the design idea is applicable.

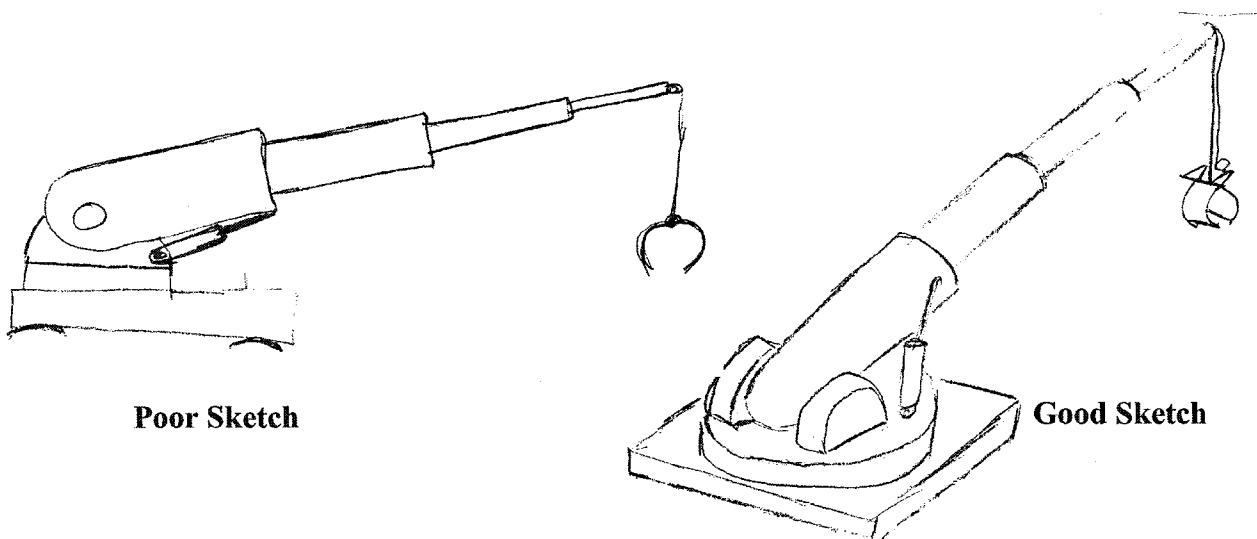
4.2.1.3 Drawing

Motion Diagrams: It is important to sketch the idea of a strategy without including any mechanical details. Motion diagrams only illustrate the motions of a mechanism with lines arcs, arrows, etc. Using different colors would be helpful. Often, a motion diagram makes you explore a scaled, evolved, or evolutionary alternative. Drawing this alternative on the same sketch in a different color would assist you compare the solutions.



Motion

Detailed



Sketching: While *strategies* are sketched with simple motion diagrams, *concepts* (possible ways to implement a strategy) are first hand sketched in 3D configurations. During sketching, you can determine critical geometries and references, which must be identified if a robust solid model is to be created later.

Mock battles could be well employed during sketching. Generally, it is not difficult to convince yourself the brilliance of your own idea when you are alone. Describing your idea in front of a group helps you discover hidden pitfalls and potentials. Sketch favorite *strategies* (motion diagrams) and *concepts* (simple stick figures) on a blackboard or sheets of paper and then discuss mock verbal battle scenarios. Whenever a weak point is discovered, try to overcome it by design, or use it to your advantage: your opponent may use this design, so if you have found a weak point you know how to defend (or attack) against it.

Solid Models: A solid model of a concept starts as simple parametric shapes that will essentially define volumes into which modules must fit. Details are added as the design progresses. It is also useful to create a solid model of the environment. A solid model of the environment lets you take measurements in the middle of the night, to make sure your mechanism will fit, and thus speeds the development process. A solid model can serve as a *Bench-Level Experiment*, to illuminate problems and thus help you guide the analysis.

4.2.1.4 Reading

Needless to say, handbooks and cookbooks of materials, machines, mechanisms, circuits, actuators, etc. are valuable resources for every design problem. One particular resource that is not yet completely discovered is the World Wide Web. It contains a huge volume of detailed information about any subject of design. For example, do you want to design a winch, and want to know how it works? Check www.howthingswork.com

You just have to find the right keywords and use an efficient search engine, you will find almost everything you are looking for. Of particular interest to design applications are web sites presenting patents in machines and machine components such as the following:

- www.delphion.com/home (Intellectual Property Network, lets you see the pictures)
- www.InventorsDigest.com (provides useful how-to information)
- www.uspto.gov (US Patent and Trademark Office)

4.2.1.5 Writing

Some people think in words. For them, writing vivid stories to describe their ideas would be very helpful. They write and therefore they can do. You will never know whether or not you belong to this category unless you give it a try. Get a pen and a piece of paper, and write down a story about how your design works.

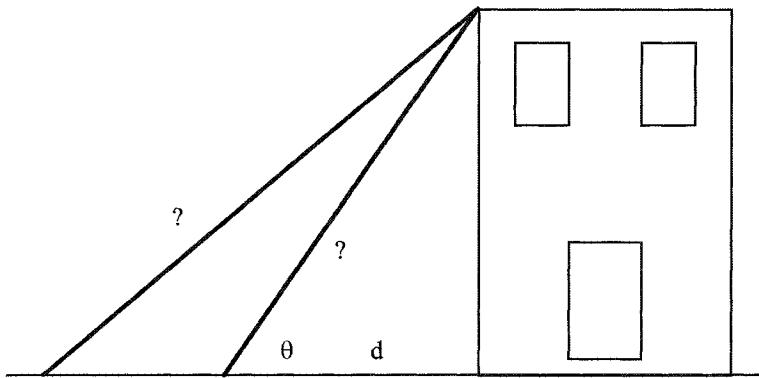
In addition to creative writing, preparing a table for organizing your thoughts would be quite helpful, as shown below. This table lists Functional Requirements (FR), Design Parameters (DP), Analysis methods (A), References (R), Risks (R), and Counter-measures (C), and thus is sometimes referred to as *FRDPARRC table*.

4.2.1.6 Analysis

Appropriate analysis is a critical part of defining a problem's bounds and generating creative concepts. While you are conducting the *sketch model*, create a systematic list of systems in each design, their FRs, and analysis methods that can be used to remove the uncertainty. Major systems to be analyzed are:

- Primary Structural Systems
- Kinematic (motion) and Dynamic (force) Systems

Functional Requirements (Events) Words	Design Parameters (Idea) Words & Drawings	Analysis Experiments, FEA, Equations, Spreadsheets...	References Historical documents, www...	Risk Words, Drawings, Analysis...	Counter-measures Words, Drawings, Analysis...
A list of independent functions that the design is to accomplish.	Ideally independent means to accomplish each FR. An FR can have several potential DPs. the “best one” must ultimately be selected	Economic, time & motion, power, stress... Each DP's feasibility must be proven. Analysis can be used to create DPs.	Anything that can help develop the idea including personal contacts, articles, patents, web sites....	High, Medium, Low (explain why) risk of development assessment for each DP	Ideas or plan to mitigate each risk, including use of off-the-shelf known solutions



- Bearing Systems
- Power & Actuator Systems
- Circuits

Identify the physics for each of the above systems, and detail the concepts (design parameters) that start to emerge. Very quickly, this will allow you to determine if you can realize a design with the hardware available. For example, what if your concept relies on a pneumatic cylinder force of 1000 N? You better choose a different design!

Remember to use analysis to design experiments, and experiments to answer questions when analysis is too difficult.

Example: Given a building of unknown height (see figure above), and two ladders, one long and one short, which ladder do you use to measure the height of the building?

FR: Determine building height

Possible DPs: 1) Ladder Length
2) Angle Measurement **OR** Distance Measurement

Analysis: Trigonometry

Option 1: Measure the angle θ , and by knowing the ladder length L , get the building height H .

$$H = L \sin \theta$$

The critical measurement is the angle, and the corresponding variation in building height is:

$$\partial H = L \partial \theta \cos \theta$$

The goal is to minimize ∂H , which is done by making θ large. Hence, the shorter ladder should be used.

Option 2: Measure the distance d along the ground, and by knowing the ladder length get the building height.

$$H = \sqrt{L^2 - d^2}$$

$$\partial H = \frac{-d(\partial d)}{\sqrt{L^2 - d^2}}$$

The answer is still the same, use the shorter ladder.

H	20	20	20	20
L	30	40	30	40
d	22.36068	34.64102	22.36068	34.64102
error	1%	1%	0.1	0.1
H	19.74715	20.58835	19.88763	19.82579
dH	0.252848	-0.58835	0.112369	0.174214

4.2.1.7 Selecting Ideas

There are many systematic methods available for evaluating design alternatives. First, you must ensure the feasibility of your ideas/strategies. The goal is to end up with only a few viable concepts, from which the “best” solution can be distilled. This is accomplished by breaking down each design into systems common to all alternatives, and raise practical questions about each system, and find answers using engineering analysis, physical models, and/or experiments:

- | | |
|--------------------------------------|---|
| <i>Structure:</i> | <ul style="list-style-type: none">- what is the overall physical framework?- Is there enough space in which to create the mechanism? |
| <i>Kinematics / Dynamics:</i> | <ul style="list-style-type: none">- What are the required / possible motions, speeds, and loads?- Can the mechanism actually generate the proper motion?- Is there any source of vibration? |
| <i>Bearings:</i> | <ul style="list-style-type: none">- How will you support moving components? |
| <i>Actuators, Sensors, Controls:</i> | <ul style="list-style-type: none">- How will you move components?- How will you sense the motion, force, torque, etc.?- How do you control the motion, force, torque, etc.? |
| <i>Power:</i> | <ul style="list-style-type: none">- Is there enough force, torque and power to create the required motions? |
| <i>Strength and Stability:</i> | <ul style="list-style-type: none">- Will components break or wear when stressed?- Will the machine tip over when it goes up an incline?- Will extending arms cause the machine to tip over? |
| <i>Manufacturing:</i> | <ul style="list-style-type: none">-How do you make sure you can make it? |

After electing few alternatives as a result of the feasibility study, you can apply different techniques of decision making to find the “best” solution. One simple method is the linear weighting scheme:

- 1) List the FRs as evaluation parameters.
- 2) Apply a relative importance weight to each evaluation parameter.
- 3) Pick one design as a “baseline” (all scores zero), and compare the rest with signed (+ or -) numbers.
- 4) When you find the “best” design, look at other alternatives that have higher weights and see how those characteristics can be transferred to the “best” design to make it even better.

4.2.2 Basic Design Principles

Engineering design often ends up to creation or modification of a physical system to perform some specific tasks. All machines and mechanisms follow several fundamental principles of physics whose knowledge and awareness would help the designer generate feasible and applicable ideas. These rules can be listed as follows (Slocum A., pergatory.mit.edu/2.007/):

- Saint-Venant's Principle
- Abbe's Principle
- Reciprocity Principle
- Self Principles
- Stability Principle
- Superposition Principle
- Golden Rectangle
- Parallel Axis Theorem
- Accuracy, Repeatability, Resolution
- Robust Principle
- Exact Constraint Design
- KISS and MISS

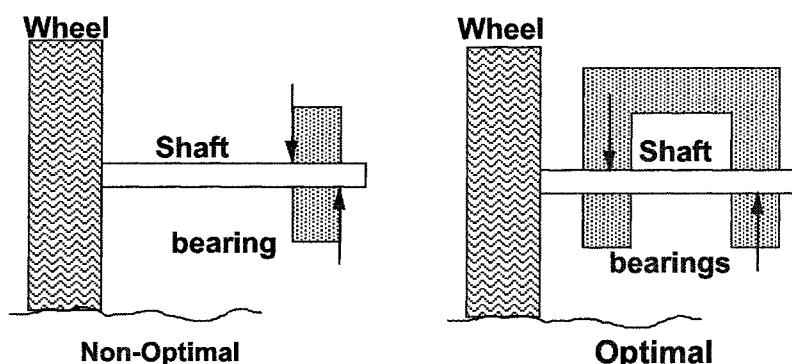
4.2.2.1 Saint-Venant's Principle

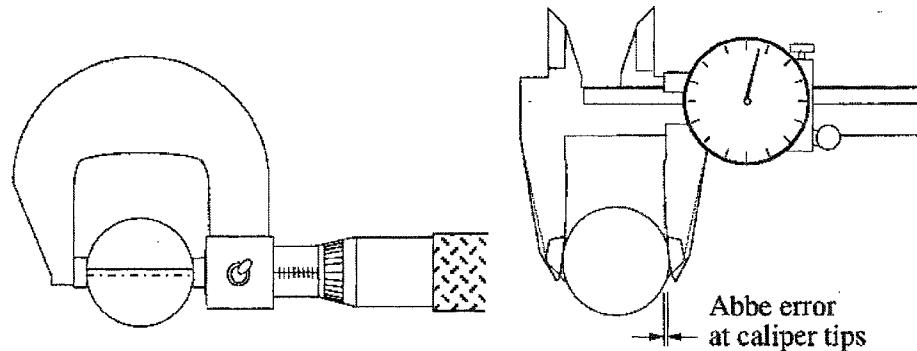
This principle simply states that local effects (strains) of loading do not affect global strains. Saint-Venant's principle is often invoked to justify approximate solutions to boundary value problems in linear elasticity. For example, bending strains at the root of a cantilever are not influenced by the local deformations of a point load applied to the end of the cantilever.

The engineering applications of Saint-Venant principle are profound for the development of conceptual ideas and initial layouts of design:

- If you want something to NOT feel the effects of something else, be several characteristic dimensions away. For example, if a plate is 5mm thick and a bolt passes through it, you should be 3 plate thickness away from the edge so that the bolt force not to cause any *warping* of the plate. Many bearing systems fail because anchoring bolts are too close to the bearings, and they warp the bearings.
- To have control over an object, apply constraints over several characteristic dimensions. For example, if a column is to be cantilevered, the anchor region should be 3 times the column base area. As another example, for holding a rotary shaft by bearings the characteristic dimension L/D must be at least 3 (see figure below).

It must be noted, however, that the above guidelines are only initial layout guidelines, and design solutions must be optimized using closed-form or finite element analysis.

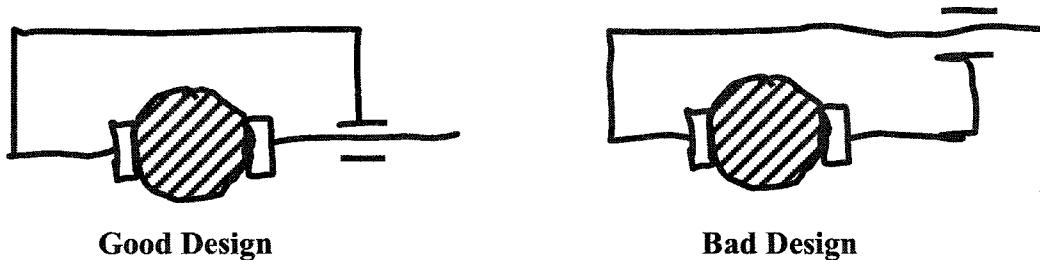




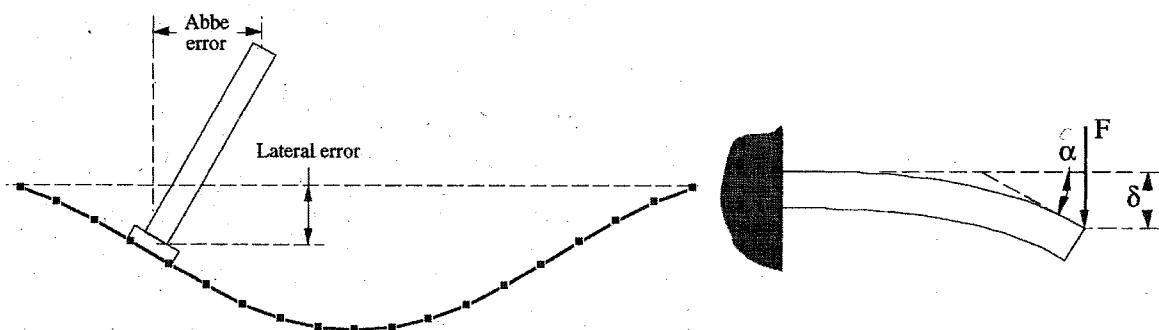
4.2.2.2 Abbe's Principle

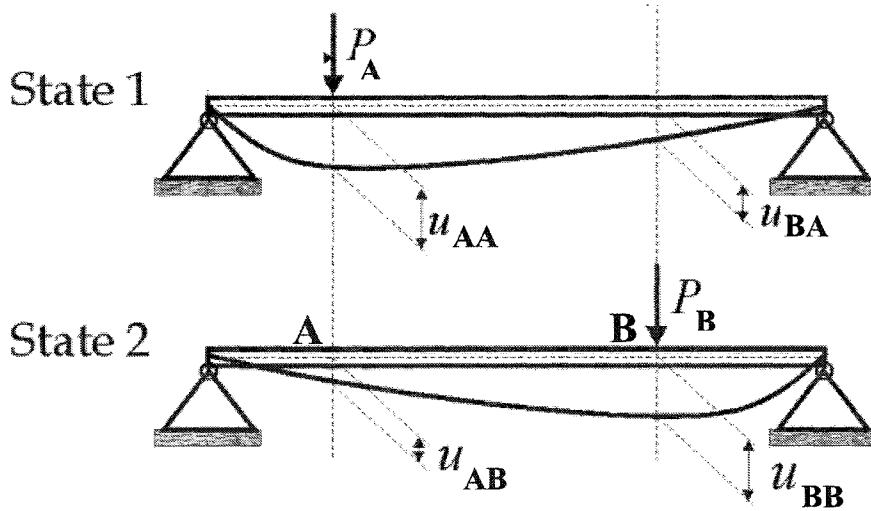
According to Abbe's principle, the scale (reference line of action) of a linear measuring system should be colinear with the spatial dimension or displacement to be measured or else the measurement must be corrected for the associated *Abbe errors*. Abbe error is the error in measuring a feature's spatial dimension (such as diameter or linear displacement), which results from a change of angular orientation between object and instrument component. This effect is observed when the measured feature or point of motion does not lie along the same line of the measurement reference (see figure above). The spatial separation between measured point and reference line is known as the *Abbe offset*.

Thinking of Abbe errors, and the system FRs is a powerful catalyst to help develop DPs, where location of motion axes is depicted schematically. For example, see figure below:



A direct implication of Abbe's principle is the fact that a small angular deflection in one part of a machine quickly grows as subsequent layers are stacked upon it. Thus, design solutions must consider not only linear deflections, but also angular deflections and their resulting Abbe errors. The intuitive rule of LAAAM (Lever Arms Amplify Angular Motions) is also a direct implication of Abbe's principle. Always measure (displacements, velocities, temperature, etc.) near the source. Always move the bearings and actuators near the moving parts.





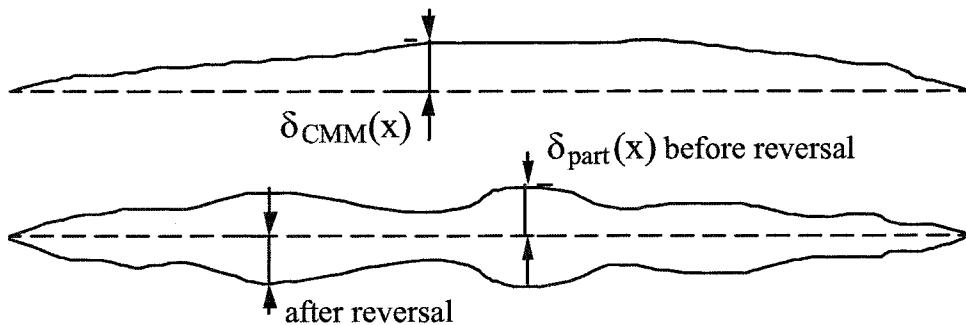
4.2.2.3 Reciprocity Principle

Maxwell's theory of reciprocity is primarily for linear systems. Let A and B be any two points of an elastic system. Let u_{BA} be the displacement of B in any direction U_B due to a force P_A acting in any direction U_A at A ; and let u_{AB} be the displacement of A in the direction U_B due to a force P_B acting in the direction U_B at B . Then, according to Maxwell's Reciprocity, $P_A \times u_{AB} = P_B \times u_{BA}$. The reciprocity principle can be extended in philosophical terms to have a profound effect on measurement and development of concepts. One application of the reciprocity principle in measurement and manufacturing is called *reversal method*. This method is used to take out repeatable (systematic) instrument errors from the measurements.

$$Z_{\text{probe before reversal}}(x) = \delta_{\text{CMM}}(x) - \delta_{\text{part}}(x)$$

$$Z_{\text{probe after reversal}}(x) = \delta_{\text{CMM}}(x) + \delta_{\text{part}}(x)$$

$$\delta_{\text{par}}(x) = \frac{-Z_{\text{probe before reversal}}(x) + Z_{\text{probe after reversal}}(x)}{2}$$

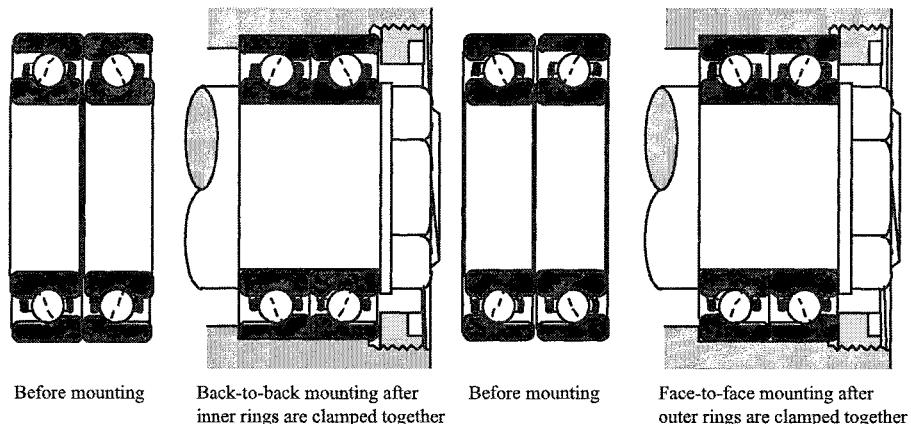


4.2.2.4 Self Principles

The common statement of self principles is that the manner in which a design reacts to inputs determines its output. Thus, investigating how a design affects itself can be used to advance the design. As an example, consider the *self-help* approach in design. In this approach, an initial effect is used to make the system (device) ready for the inputs, and the supplementary effects will be naturally induced by the inputs in the direction of enhancing the output. Airliner doors in the airplanes use this approach for having maximum air sealing. When the door is closed, several latches squeeze the seal making the cabinet airtight. As the plane ascends and the outside air pressure decreases, the higher inner air pressure presses on the door and causes the seal to seal even tighter. Other forms of self principles can also be considered, such as *self balancing*, *self reinforcing*, *self braking*, etc.

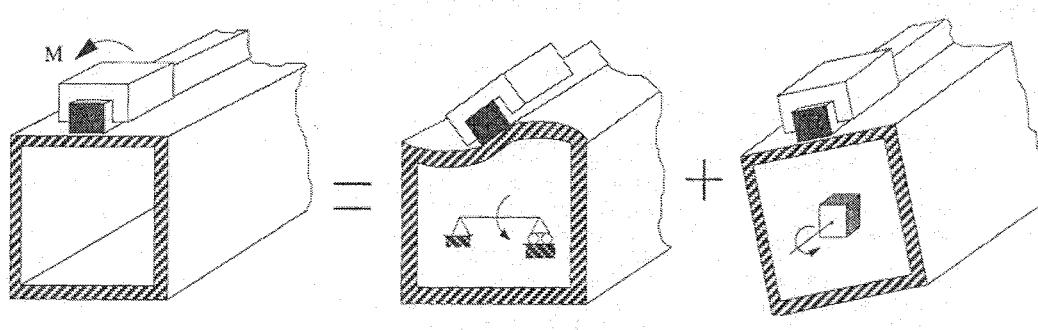
4.2.2.5 Stability Principle

Stability is an energy state of systems. All systems naturally tend to move towards the minimum level of their energy where a stable state occurs. If the system is configured at the maximum level of its energy, by applying a small input it will move off the state towards the minimum energy level. A neutral state can be configured for a system if the level of energy remains constant at all states. As an example, in many designs, bearings are mounted in the *face-to-face* mode to better tolerate shaft misalignments. However, axial thermal growth effectively adds to radial thermal growth causing an unstable state, so that it can cause the bearings to overload and seize at high speeds. On the other hand, bearings mounted in the *back-to-back* mode use axial thermal growth to cancel radial thermal growth, and thus remain thermally stable at high speeds.



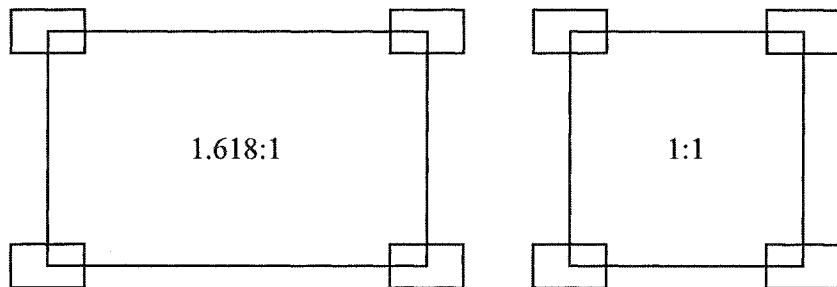
4.2.2.6 Superposition Principle

Superposition principle states that, with certain exceptions (large or plastic deformations), the effect (stress, strain, deflection, current, etc.) produced on a system by any final state of loading is the same whether the causes that constitute the loading are applied simultaneously or individually in any given sequence (see figure above). A direct implication of this principle in design is that even the most complicated problems can be broken up into simple, manageable ones. The key is to logically divide them up.



4.2.2.7 The Golden Rectangle

The Golden Rectangle is a rectangle with a length-to-width ratio of about 1.618. This rectangle has some interesting geometrical properties. For instance, when a square is cut from such a rectangle, the remaining rectangle is also a Golden Rectangle. In design, the proportion of the Golden Rectangle is a natural starting point for preliminary sizing of the structures and elements. As a rule of thumb, the higher the speed, the higher should the length-to-width ratio be.



4.2.2.8 Parallel Axis Theorem

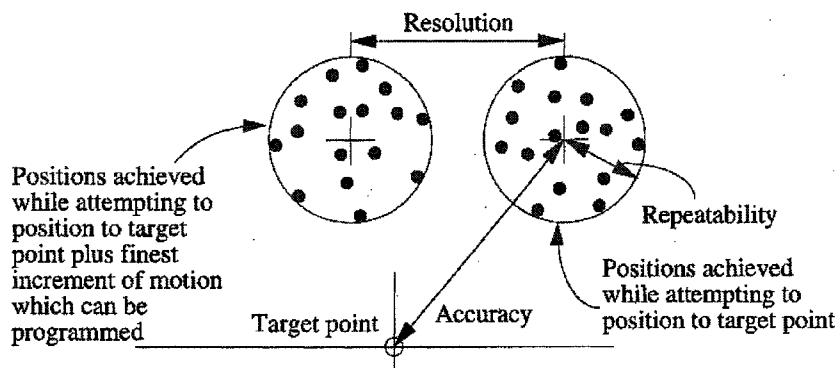
This theorem is primarily used for measuring the moment of inertia of a body (or cross section in 2D) against rotation about any axis knowing the moment of inertia of the body against rotation about its principal (neutral) axis.

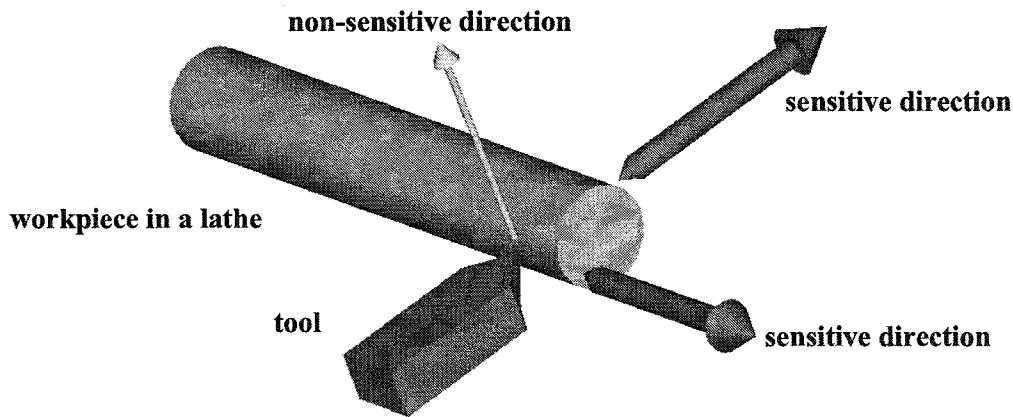
$$I_o = I_c + md^2$$

The neutral axis passes through the centre of mass (or centre of area in 2D) of the body. An important implication of this theorem in design is that a section stiffens with the square of the distance from the neutral axis. For example, when designing a laminate (1.5mm aluminum sheet separated by a wooden core), double the core and quadruple the panel stiffness.

4.2.2.9 Accuracy, Repeatability, and Resolution

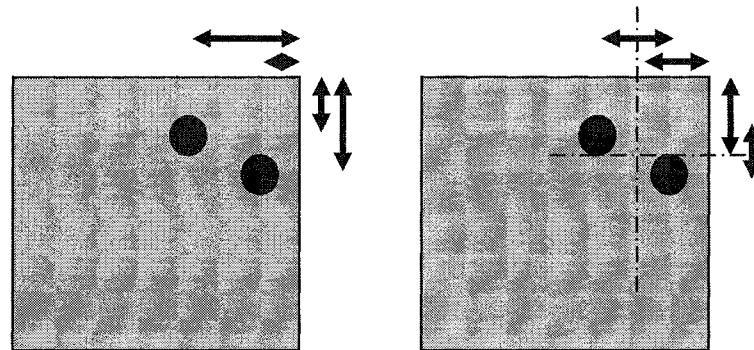
The outcome of any design must be accurate, repeatable, and within a certain range of resolution. Accuracy is the closeness of the real output to the desired output. Repeatability is the capability of having the same output whenever it is generated. And, resolution is the details through which the output must be interpreted (see figure below). These concepts are extended to system components, as well. The performance of all parts must be accurate, repeatable, and within a certain resolution.





Achievement of the above requirements in all directions may be impossible or very expensive. Hence, identification of sensitive directions along which precision is needed is an important aspect of design. An example is the location of the tool in a lathe (see figure above). In many cases of design, it is more important to obtain repeatability than accuracy, because the latter can often be satisfied by sensors and control systems.

Another important aspect of design, related to precision, is to assign reference features with respect to which measurements are performed. For example, which of the following models uses reference features correctly?



4.2.2.10 Robustness Principle

In order to reduce the errors, increase the repeatability, and thus enhance the robustness, the primary work zone must be as near as possible to the centers of action. The centers of action are points at which when a force is applied no moments are created. Most important centers of action are centers of mass, stiffness, and friction.

Centre of Mass: The centre of mass of a system of mass particles m_i move like a single particle of mass $M = \sum m_i$ under the influence of the resultant external force acting on the system. The location of the centre of mass of a system of bodies m_i with respect to a reference frame can be obtained by:

$$\bar{X}_c = \frac{\sum_i m_i X_i}{\sum_i m_i},$$

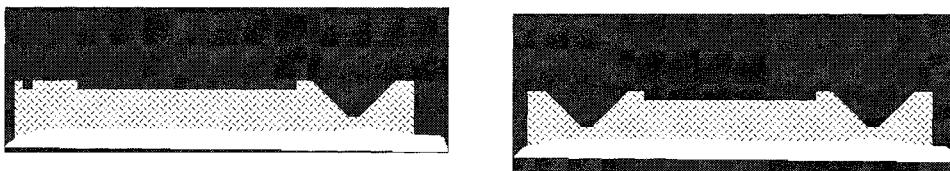
where \vec{X}_c is the location vector of the centre of mass, and \vec{X}_i is the location vector of the body i with respect to the reference frame.

When a force is applied at the centre of mass the object undergoes only linear acceleration and thus has no angular acceleration component (which would otherwise lead to Abbe errors).

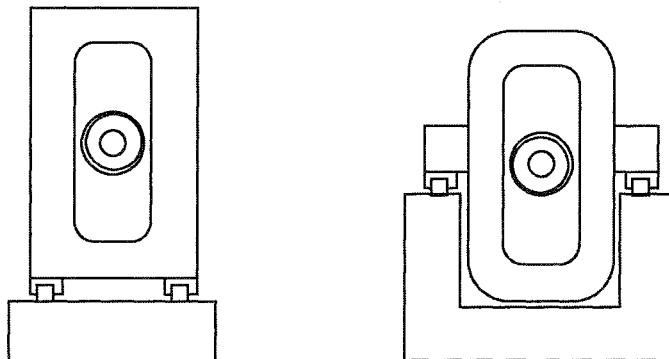
Centre of Stiffness: This is the point at which when a force is applied to a locked-in-place axis, no angular motion of the structure occurs. It is also the point about which angular motion occurs when forces are applied elsewhere on the body. A body supported by bearings behaves as if all the bearings are concentrated at the center of stiffness. The location of the centre of stiffness with respect to a reference frame is also obtained from the above equation by replacing m_i with k_i the stiffness coefficient of each bearing.

Centre of Friction: The centre of friction is the point at which when a force is applied to a moving structure to which no other external forces are applied, no angular motion of the structure occurs. It is calculated using force and moment balance equations that consider the effects of friction, bearing geometry, and center of gravity.

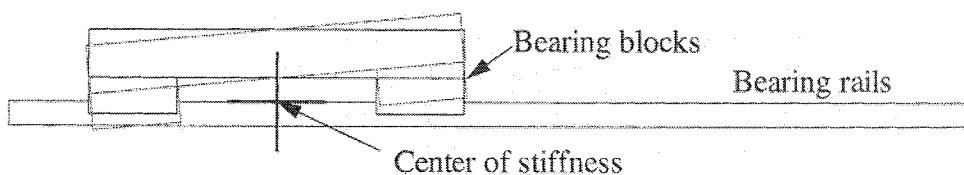
Example: If a load is applied to different positions on a Vee-and-Flat and a Double-Vee supported carriage (see figure below), how do the centre of friction and the centre of stiffness vary?



In designing a machine, if the actuators for a machine apply their forces near the center of mass, stiffness, and friction, errors will be minimized. However, it is often difficult to make all three coincide (see figure below). The practical goal hence must be to get them as close as possible to the centers of action.



Example: As another example, if a machine element (for example, a lead-screw nut in figure below) is located at the center of stiffness, then error motions of one machine element (wobble of the screw) will not cause pitch errors (Abbe errors) in another element (carriage).



4.2.2.11 Exact Constraint

A design must be *exact constraint*. An exactly constrained design has no chance of deforming or having its function impaired after the components are assembled; be it by fastener tightening or thermal expansion. Total number of independent variables that can completely identify the configuration of a system in the space is called *degrees of freedom of the system*. The number of degrees of freedom for a general mechanism can be calculated as:

$$d.o.f. = \lambda(n-1) - \sum_{i=1}^k (\lambda - f_i)$$

where, n is the number of rigid bodies in the system, k is the number of joints, and f_i is the number of degrees of freedom for joint i . A *joint* is the connection between two bodies, which provides some physical restriction on the relative motion between the two bodies. Six basic types of joint used in machines and mechanisms are shown in table below.

4.2.2.12 KISS and MISS

Keep it super simple and make it super simple!

Name of Pair	Geometric Form	Schematic Representations	Degrees of Freedom
<i>Revolute</i>			1
<i>Cylinder</i>			2
<i>Prismatic</i>			1
<i>Sphere</i>			3
<i>Helix</i>			1
<i>Plane</i>			3

4.3 Design Methodologies

The process of electing the right design solution and finalizing its details can proceed based on different approaches that reflect the focus point of the designer or the organization. Some important design methodologies are addressed in this chapter as listed below:

- Design for Quality (Quality Function Deployment)
- Design for Optimization
- Design for Satisfaction
- Design for Reliability

Before discussing different methods of design, we need to know what are the design variables and parameters that must be considered or perhaps adjusted to reach the design goals. This topic is discussed in the next section.

4.3.1 Design Attributes

In order to translate the language of the client to the terminology of the engineers and designers, four distinct aspects of design, i.e., objectives and constraints, functions, metrics, and performance specifications must be identified from the customer's *needs* and *desires*.

4.3.1.1 Design Objectives and Constraints

Objectives are expressions of the desired attributes and behaviours that the customer would find attractive. They are normally expressed as “*being*” statements that say what the design should *be*, as opposed to what the design must *do* (as in design functions).

Constraints are restrictions or limitations on the behaviour or values of the design parameters and/or variables. They are usually stated as clearly defined limits whose satisfaction can be framed into a binary choice. For example, in designing a ladder, several restrictions may exist, such as the ladder material be a conductor or not, or the step deflection be less than 1mm. Constraints limit the size of the design space, while objectives permit the exploration of the remainder of the design space. That is, constraints can be formulated so as to allow the designer to reject alternatives that are unacceptable, while objectives allow us to select among design alternatives that are at least acceptable. It is important to recall that both objectives and constraints refer to the object being designed, not to the design process.

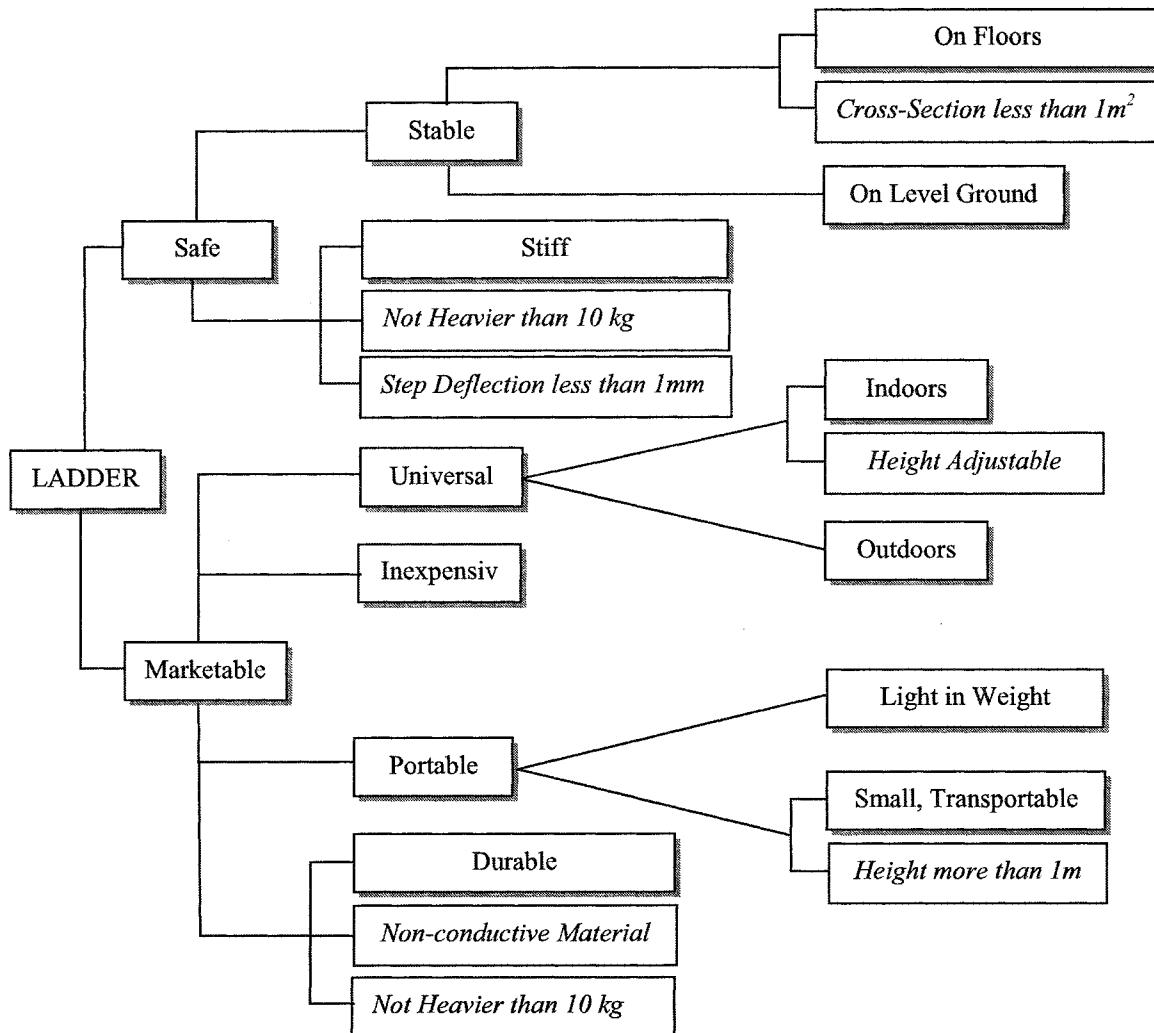
It is useful to build an *objectives-constraints tree* to represent the characteristics of the design output. This tree is a hierarchical, clustered structure through which the main objectives and constraints are decomposed into related or similar sub-goals that are clustered together. In some design methods such as Quality Function Deployment, this tree is represented in a tabular format as will be discussed in Section 2.

Example: Figure below shows the objectives-constraints tree of the design for a safe and marketable ladder.

4.3.1.2 Design Functions

The statement of a design function usually consists of a verb and a noun, and the verb will normally be an “action” verb, rather than a “*being*” verb. For example, “lift”, “raise”, “move”, “transfer”, or “light” are all action verbs. The noun in the statement of function will generally start off as a very specific referent or item, but an experienced designer will look for the more general case.

Example: For designing a bookcase, its functions must be listed first, such as: to resist forces of gravity associated with any object weighting less than some predetermined weight, to separate objects in different levels of height, etc. As you notice, the noun used for describing the above functions is more general than, for instance, “book”, because a bookcase can also be used to hold trophies, artifacts, or even piles of homework.

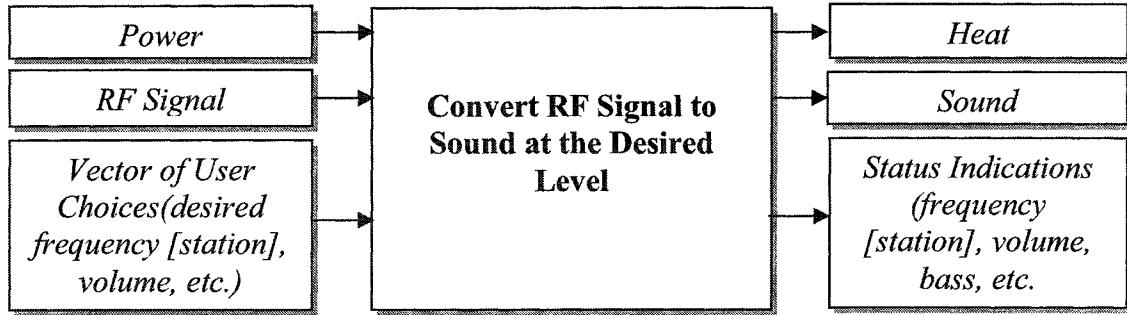


Example: An experienced designer always avoids tying a function to a particular solution. For example, if we were designing a cigarette lighter, we might be tempted to consider “applying flame to tobacco” as a function. This implies that the only way to light the tobacco is by using a flame (and that tobacco is the only material to be lit). Car lighters, however, use electrical resistance in a wire to achieve this function. Thus, a better way to characterize this function might be “to ignite leafy matter” or even “to ignite flammable materials.” (In the context of ethics in engineering design, it will be discussed whether it is ethical for an engineer to design a “better” cigarette lighter, in light of the well-documented health hazards associated with smoking.)

4.3.1.2.1 Methods of Identifying Design Functions

Although enumeration is the most natural method and the best starting way of determining functions of a to-be-designed product, in many cases we could get “stumped” very early in the process. Consider, for example, a bridge. If the bridge is used for highway traffic, we might note that its basic function is to act as a conduit for cars and trucks. Most of us would have to scratch our heads for a while before being able to go much further than this initial, single-entry list. There are, therefore, some more systematic methods to find out the desired design functions as discussed below.

Glass Box Method: In this method, the system or object being designed is graphically represented by a “black box” with inputs and outputs. It is important that all of the known inputs and outputs be specified, even those undesirable byproducts that are resulted from unwanted secondary functions. Once the “black box” has been initially drawn, the



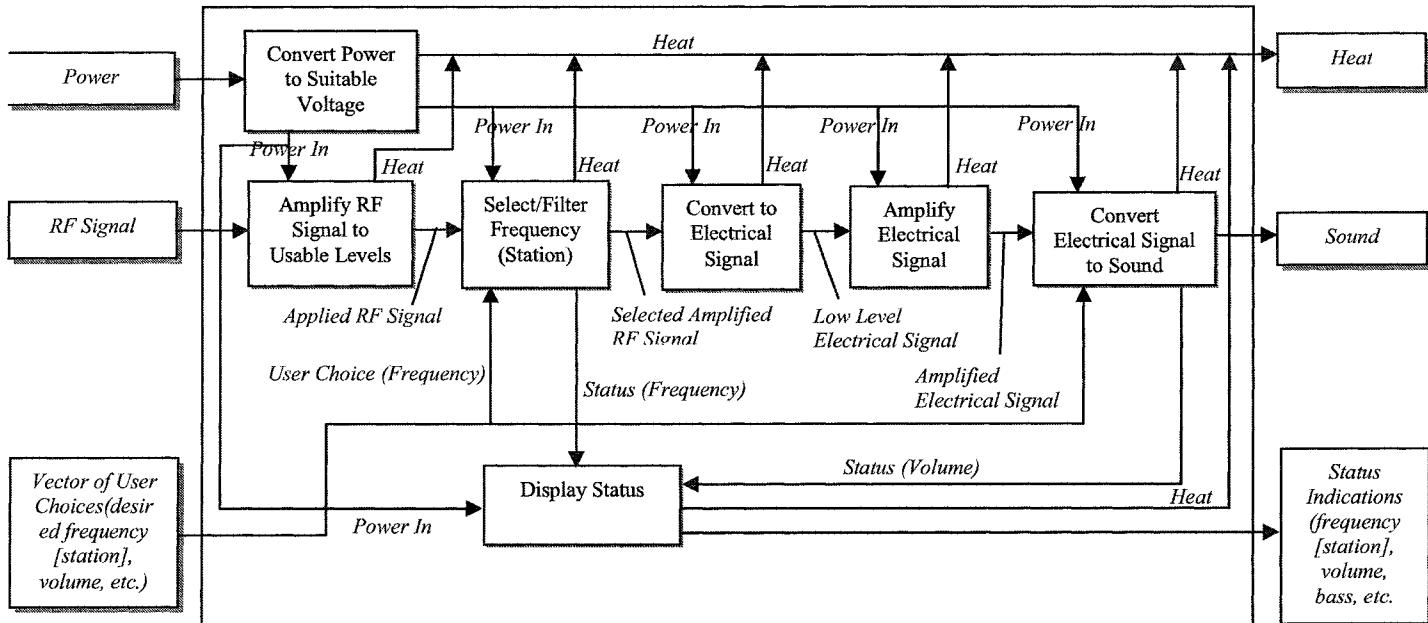
designer can ask questions such as “*What happens to this input?*”, or “*Where does this output come from?*” These questions will then be answered by removing the cover of the black box, thus making it into a “transparent box”, so the designer can see what is going inside, and expose transformations from inputs to outputs. Within a given box, especially after it is made transparent, we can also link more detailed “sub-inputs” to internal boxes that produce related “sub-outputs.”

Example: Radio Design

Figure above shows the black box of a system known to us a radio. It can be thought of as being contained in a box that transforms the RF signal into another signal that we hear as audible sound, i.e., music, talk and perhaps noise. The relevant questions to ask are: “*How does this happen?*”, “*What functions are performed in a radio?*”, etc. By taking the cover off the system, several new “black boxes” will show up within, as shown in figure below.

The new boxes include transforming the power from the outlet level to a desired level, filtering out unwanted frequencies, amplifying the signal, converting RF to electrical signal, etc. Thus, making the system black box transparent revealed a number of additional functions. If we were responsible for designing the radio, we would probably want to remove the covers of even more of the boxes we now see. On the other hand, if we were simply assembling a radio from known components, we might stop at this level. The process of opening the internal boxes continues until the designer fully understands how all inputs are transformed into corresponding outputs and what additional inputs or side effects are produced by these transformations.

The glass box method does not apply only to devices that have a physical box or housing. If, for example, we were designing a playground to be used in a rainy climate, our inputs would include the children, their parents or caregivers, and the rain. Our outputs would include entertained children, satisfied parents, and water. If we forget the water, our playground design will neglect the need for proper drainage.



Function-Means Tree: A function-means tree is a graphical representation of the basic and secondary functions associated with a design. The top level of the tree shows the basic function to be met. Each of the succeeding levels alternates between showing various means by which the primary function might be implemented and displaying the secondary functions made necessary by those means. In order to make the tree more readable, some graphical notion is employed to distinguish functions from means.

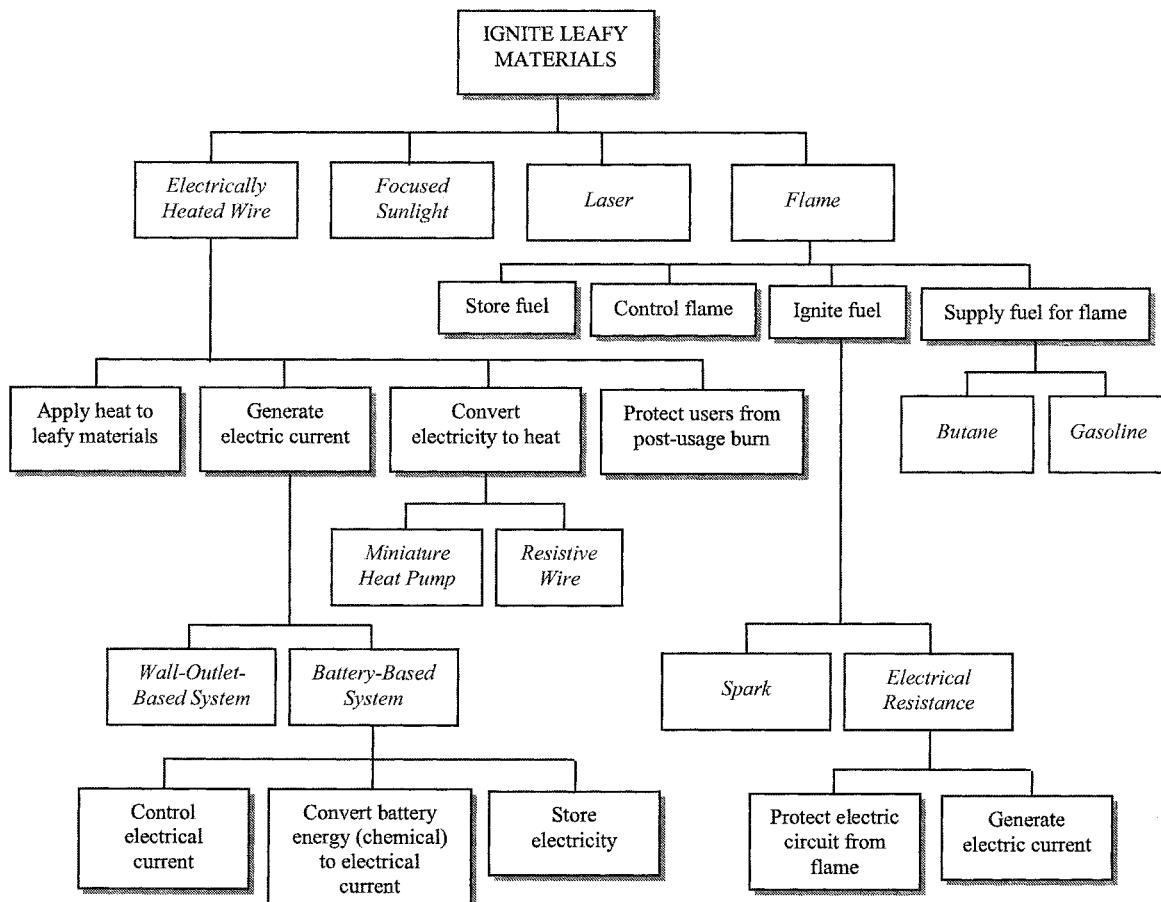
Figure below shows part of a function-means tree for a hand-held cigarette lighter. The top-level function has been specified in the most general terms possible. At the second level, two different means have been given: a flame-based lighter and a hot-wire-based lighter. These different means imply a different set of secondary functions, as well as some common ones. Once a function-means tree has been developed, one can list all of the functions noting which are common to all (or many) of the alternatives and which are particular to a specific means. Those functions that are common to all the means are likely to be inherent in the problem. Others will need to be addressed only if the associated concept is adopted during the alternatives evaluation phase.

4.3.1.3 Metrics

Metrics are standard measurements that are used to measure the extent to which a design realizes its objectives. Ideally, a metric gives the designer an exact gauge of the objective s/he is concerned with. In practice, designers often make difficult choices about what constitutes an appropriate metric, how one actually applies that metric, and how much it costs to measure the achievement of a design objective.

A three-step procedure of selecting metrics is as follows:

- 1) Identify both the scale and the units of something that is appropriate to measure the objective. The appropriate units that are applied to the concerned objective can be numerical, such as *kg*, *lb*, or *oz* for measuring weight or mass, or they can be subjective ranking, such as *advanced*, *intermediate*, and *elementary* for measuring the complexity level



of a system. There are four major types of measurement scales for design evaluation and measurement:

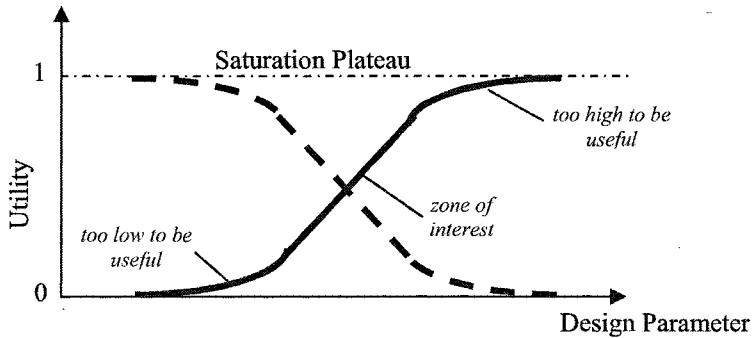
- a. **Nominal Scale:** assigns a category label to an objective with no explicit ordering on the category labels. For example, colors, smells, professions, etc., need nominal scales for expression.
 - b. **Ordinal Scale:** assigns values to an objective that are rank-ordered with respect to some characteristics, but no arithmetic transformations are meaningful. For example, “parents satisfaction” in a playground design might be measured as high, medium, and low. But, one would not say that the difference between high and medium satisfaction is equal to (or any other transformation) to the difference between medium and low satisfaction.
 - c. **Interval Scale:** defines a unit of measurement such that the distance between units is meaningful (usually equal). Thus, an interval scale permits statements not only as to whether one value ranks higher than another, but also as to the difference between the two in comparison to the difference between another pair of values. For example, temperature measured in degrees centigrade is an interval scale.
 - d. **Ratio Scale:** involves measures that have meaningful distances (equal units) throughout and a meaningful zero point, which indicates no possession of the characteristic being measured. Height, weight, and cost are examples of ratio scale. As an example to distinguish ratio scale from interval scale, one can say that an individual whose height is 152.4 centimeters is twice as tall as an individual who height is 76.2 centimeters. In contrast, a student who earns a score of 48 on a test cannot be said to be twice as proficient as a student who earns a score of 24 on the same test since a score of zero probably does not indicate absolutely no proficiency at all.
- 2) Identify a means of assessing the value of a design in terms of those relevant units. An important aspect of this step is to ensure that the plan for measuring the design objective is compatible with the type of scale and units selected in the first step. This could include, for example, laboratory tests, field trials, customer response to surveys, etc. Cost, in particular, could be quite difficult to measure, unless some factors of manufacturing techniques to be employed, number of units to be manufactured, and components to be included in the design are known.
 - 3) Evaluate whether this particular measurement and its subsequent evaluation is feasible. The feasibility is mainly due to time, energy, and cost needed for the measurement. For example, consider design of a computer notebook. “Low cost” might be one of the objectives of the design. However, it might well happen that the information required to accurately assess the manufacturing costs is not available to the design team without a significant and expensive study. An alternative option might be to estimate the manufacturing cost by adding up the costs of the individual components. Alternatively, the designers might depend upon expert input and simply rank the designs into ordinal categories such as *very expensive, expensive, moderately expensive, inexpensive, and very cheap*.

4.3.1.4 Performance Specifications

Design objectives (and constraints) and functions determine what the design outcome should be and what it must do, respectively. However, it is essential for developing and assessing design outcomes that functions and objectives be translated into measurable terms. Hence, the designer must find a way to measure the performance of a design in realizing a specific function or objective, and then establish the range over which that measure is relevant to the design. Finally, the designer must determine the extent to which improvements in performance over that range really matter. For example, if the design problem is to make a device that can produce musical sounds, the designer must specify how loudly, how clearly, and with what tones the device must produce the sound.

The process of assigning performance specifications can proceed through the following steps:

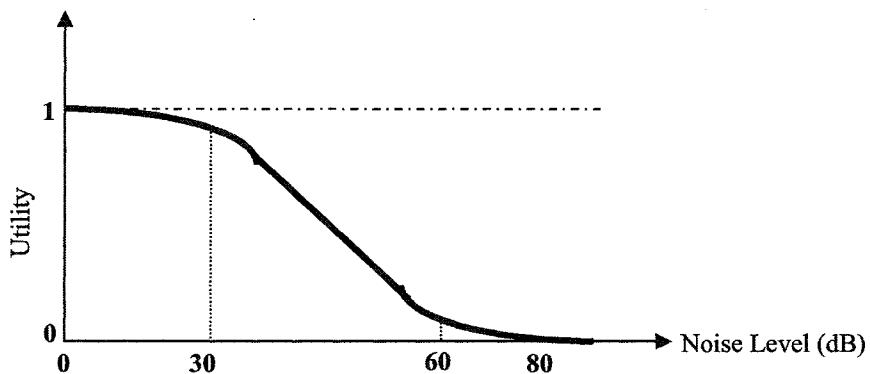
- 1) Determine design parameters that reflect the functions or objectives that must be measured and the units and scales in which those parameters are to be measured.
- 2) Establish the range of interest for each design parameter.
- 3) Then, form a *utility plot* for each design parameter. In this plot the utility or value of the design gain (usefulness) is plotted on the vertical axis over a normalized range from 0 to 1. The value of the design parameter is shown on the horizontal axis within the assigned range for the design parameter. Plot an S-curve (or inverse S-curve) as shown below to show the variation of the utility with the values of the parameter. For the entire S-curve, the utility is initially flat at low values of design parameter, increases over a range of interest, but at some points these incremental



gains decrease to small marginal improvement (degradation) as it approaches the plateau at saturation. Thus, utility values below the threshold and above the plateau are treated as equals, as no meaningful or worthwhile gains can be lost or achieved. The zone of interest lies between the threshold and the plateau, within which the design gains should be matched to and measured with respect to the specific design parameter.

Example: Suppose that a designer is asked to develop a Braille printer that is significantly more quiet than competing designs, so that it can be used in office settings (while none of the competing design are quiet enough for office environment). A natural question is hence “*how quiet does this design have to be?*” To answer this question, the designer must first determine the relevant units of noise measurement and the range of values of these unites that are of interest. The designer would also find out how much noise is generated by current printer designs and whether or not listeners can distinguish between different designs. If for example, one printer produces the same noise level made by gas lawn mower, while another generates noise at the level of a truck, it is unlikely that either design will be useful in an office setting, so there is no utility gained by distinguishing between these two designs. Sound intensity levels are usually measured in decibels (dB), which is a ratio scale. The designer may look for a source of information for sound intensities of different devices, such as table below. With this information, one can build a utility plot for the noise parameter in this design similar to figure below.

Level (dB)	Qualitative Description	Source / Environment
10	Very Faint	Hearing Threshold
20	Very Faint	Whisper, empty Theater
30	Faint	Quiet Conversation
40	Faint	Normal Private Office
50	Moderate	Normal Office Background noise
60	Moderate	Normal Private Conversation
70	Loud	Radio, Normal Street Noise
80	Loud	Electric Razor, Noisy Office
90	Very Loud	Band, Truck
100	Very Loud	Gas Lawn Mower, Boiler Factory

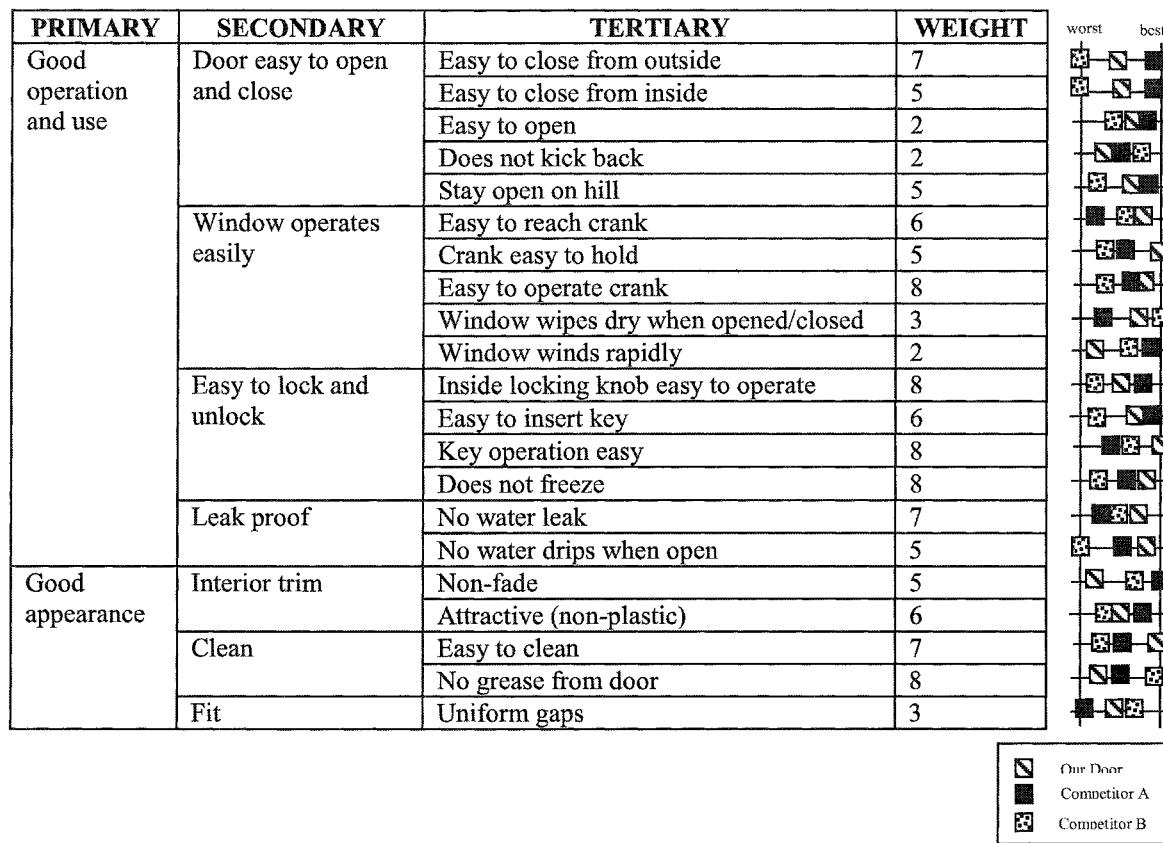


4.3.2 Quality Function Deployment

Over the past 40 years Japanese have developed many quality improvement techniques for design processes. One popular method is Quality Function Deployment (QFD), which was developed in Mitsubishi's Kobi shipyard in 1972, and was adopted later by Toyota, and is now used by almost half of the Japan's major companies. Automobile manufacturing companies in the United States and other countries began using this technique in the early 1980s. Currently, many major companies use it, including Ford, Crysler, General Motors, Hughes Aircraft, Boeing, Texas Instrument, and 3M.

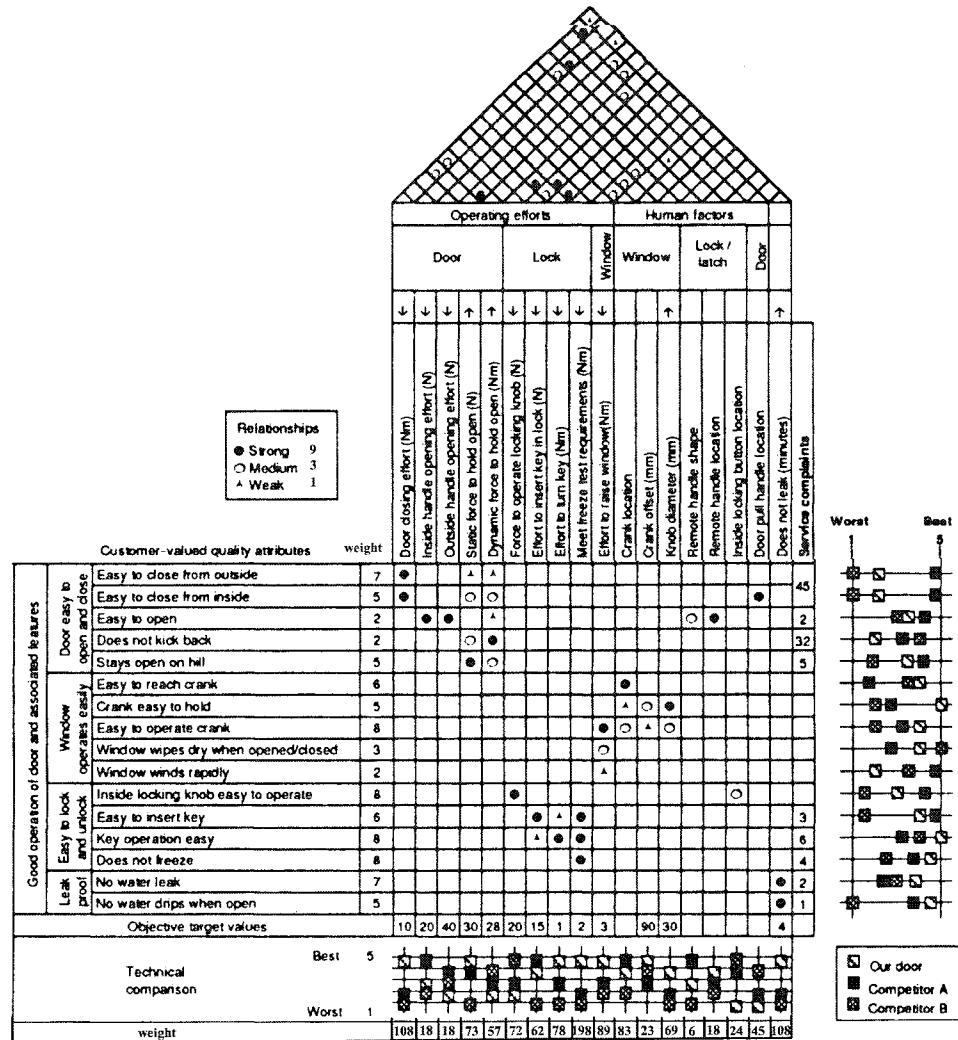
The focus of QFD is to get the idea of quality directly from the customer, introduced into the early phases of the design cycle and maintained throughout the entire life cycle of the product. The fundamental axiom is that although customers are only concerned with "if", and not "how" their requirements are satisfied, there is often a strong link between these two aspects, which can be easily overlooked by design team members due to the continuous pressure of solving technical problems. Hence, QFD provides a formalized method of linking customer-valued attributes to the engineering, manufacturing, and process decisions that designers need to make.

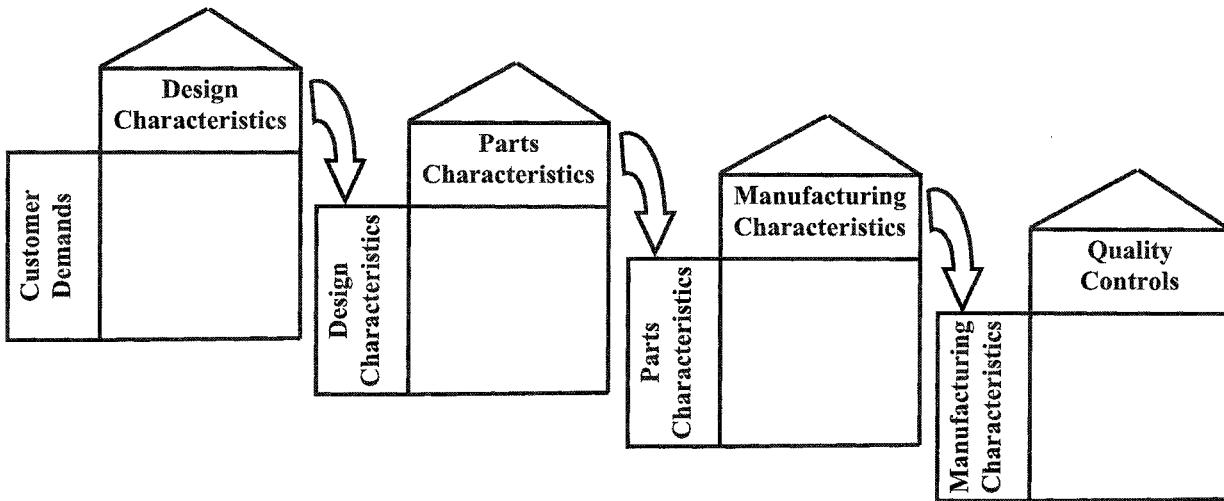
The QFD method uses multi-disciplinary project teams to prepare the data required for the completion of a series of charts or matrices. This method begins with interviews of customers to elicit those general characteristics most important to them, called Customer Demands (or Attributes). The customer, with the help of designer, will assign a weight to each demand, usually between 1 and 10, indicating its relative importance. As an example, a survey of automobile buyers identified attributes that were considered important when making decisions about which product to buy. For simplicity, let's focus on a small part of the survey relating to the door. Table below shows the objective tree of the automobile door in a tabular format. An additional indicator for the QFD charts is to express the success of the competitor products in meeting the needs of the customers. This is usually done by asking customers to rate products attributes by assigning success values to them. In the above example, let's assume there are two competitive products plus the one from the design company currently in the market. The rates are also shown in figure below.



In the next step, the design team should find design metrics and specifications by inspecting the design objectives and functions as discussed in the previous section. The customer's demands and design specifications will fill a matrix with rows as demands and columns as specifications. In this way, each demand is related to each specification as one entry of the matrix. If any row of this matrix remains blank, satisfaction of that demand cannot be assured, and the customer demand should be either eliminated or other performance specification added to the columns. The matrix entries illustrate the strength of the relationship (with some pre-defined symbols) between customer demands and design specifications. Thus, an important task in this step is to determine the strength of the relationships (or the degree of correlation) between the attributes and design specifications. This matrix is shown in the chart below for the automobile door example (only for one primary attribute). In this chart, three symbols have been used to show whether a particular specification has a strong, medium, or weak influence on an attribute.

In some cases, making a change to the magnitude of one specification has "knock-on" consequences for others. In the above example, the positioning of the handle that the customer uses to close the door from the inside will have obvious repercussions on the force required. A handle located near the door hinge will require a higher force applied to it than a handle further away from the hinge. These relationships can be very important particularly when changes to one characteristic aimed at improving an attribute results in detrimental changes to another. The triangular matrix sitting on top of the chart carries information on which specifications are linked with others. Again symbols have been used to illustrate the strength of this relationship.



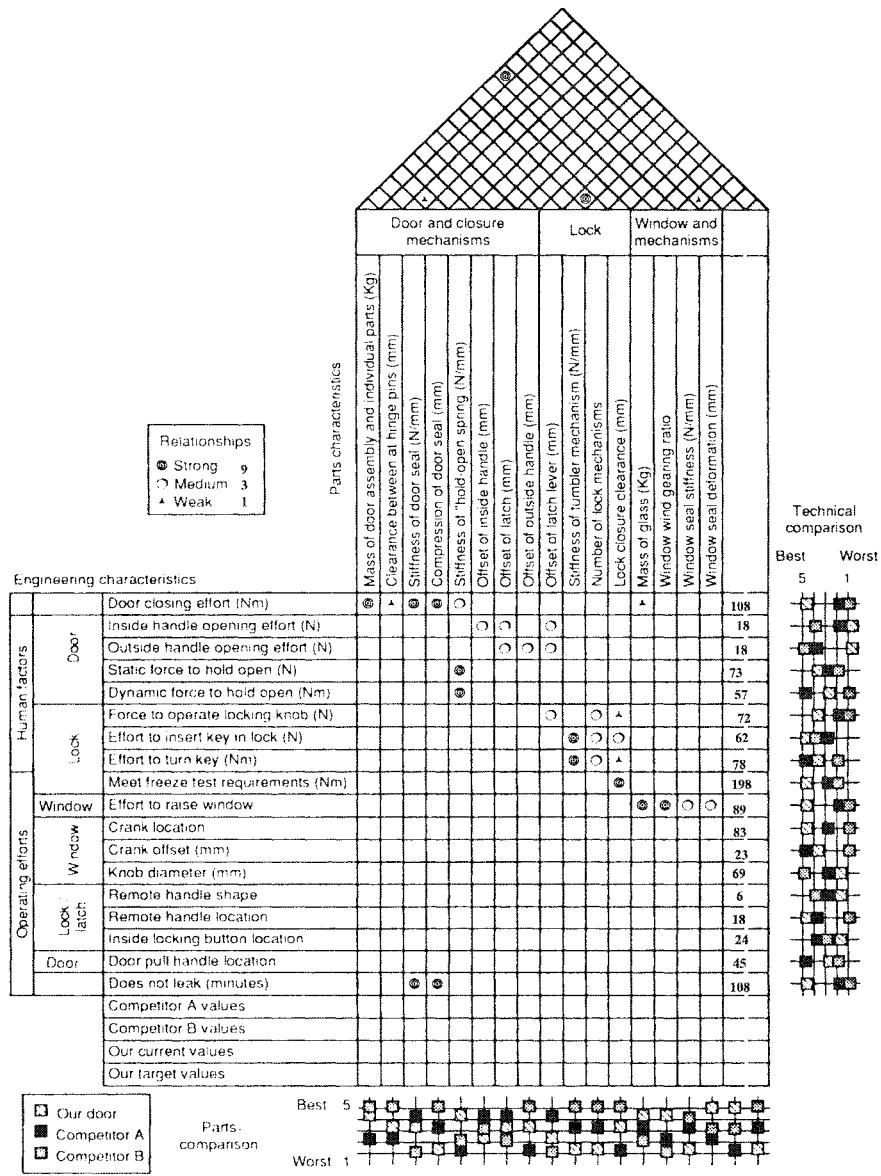


Hence, the purpose of QFD is two-fold. First, it helps the design team identify the engineering characteristics under its control, which might enable the designer to change customer-valued attributes for the better. Secondly, it informs the designer about the possible detrimental effects on some of the customer-valued attributes while making improvements to others.

In the next step of the QFD analysis, the designer multiplies the weights by the numerical values for the relationships, and adds the numbers in each column to get the scores for the design specifications at the bottom of the table. The total score for each column indicates the importance of the related specification in measuring the customer's satisfaction. Measures with low scores typically receive less consideration. However, this does not necessarily mean that these measures will not be used in the product design; they may still be necessary for contractual or other reasons. Nevertheless, to meet the goal of satisfying the customer, the design team has to pay strict attention to the measures with high scores.

In addition to the relationship between design objectives and specifications, the QFD chart presents the success values for the attributes in competition designs, as well as technical evaluation of the performance characteristics undertaken by design team members. Similar to the customer's perception of attributes satisfaction in competition designs, technical evaluation contains a comparison between the company's own product and other competitors. The chart can also show the number of claims that have been reported with regard to certain attributes, and the ideal values for design parameters which correspond to a utility value equal one, as discussed in the previous section.

QFD analysis does not stop by the first chart. The attribute-characteristic chart is merely the first in a sequence of several (usually four or more) charts that can be used to link customer-valued attributes to manufacturing processes and production requirements, as shown in figure above. The first chart, as explained before, links customer demands to the design characteristics that are under the control of the design team. The second chart, maps the relationships between design characteristics and parts characteristics. Parts characteristics are the indicators that describe features of the product's components that influence or determine the design specifications. For example, for a design characteristic such as "door closing effort" in the previous example, parts characteristics such as "weight of door assembly", "stiffness of hold-open spring", and "door seal stiffness" are listed. Through the second chart, the designer would be able to link the part characteristics to customer demands via design characteristics, and hence assess the impact of part modifications on customer perceptions. The second chart for the "car door" example is shown below. Since each design characteristic is frequently influenced by several product characteristics, the number of columns in the second chart can be large. The parts characteristics shown in chart below are a small portion of the entire chart. The third and fourth QFD charts provide the links between the parts characteristics and manufacturing specifications, and the link between manufacturing characteristics and quality control specifications, respectively.



4.3.3 Design for Optimization

In a design problem, two kinds of *effects* are inherently associated with any component or system:

- undesirable effects, such as cost, excessive weight, heat, large deflections, and vibrations, in many design problems;
- desirable effects, such as long useful life, efficient energy output, good power transmission capability, and high cooling capacity.

Optimum design can be defined as the *best* possible design from the standpoint of most significant effects; that is minimizing the most significant undesirable effects and/or maximizing the most significant desirable effects in design.

Application of optimization to a design problem requires formulation of an (or several) objective function(s) such as weight, cost, power, shape, etc., and the expression of design constraints as equalities or inequalities. The designer seeks the optimum solution for a given objective function U (the multi-objective optimization will be discussed in sequel) expressed in terms of the independent *design variables*, x_i as

$$U = U(x_1, x_2, \dots, x_n),$$

that fulfills several equality and/or inequality constraints, such as

$$\begin{aligned} h_i &= h_i(x_1, x_2, \dots, x_n) = 0 & i &= 1, 2, \dots, m; \\ g_i &= g_i(x_1, x_2, \dots, x_n) \geq, \leq 0 & j &= 1, 2, \dots, p. \end{aligned}$$

The mathematical solution of an optimization design can be through either *analytical* methods, such as Lagrange Multiplier Method, or *numerical* methods, such as linear and nonlinear programming techniques.

4.3.3.1 Lagrange Multiplier Method

This method is applied to nonlinear optimization problems with single objective function $U = U(x_1, x_2, \dots, x_n)$, defined in a closed-form, and several equality constraint functions $h_i(x_1, x_2, \dots, x_n) = 0$ ($i=1, 2, \dots, m$). It uses a function called Lagrange Expression (LE), which is a linear combination of the objective and constraint functions:

$$LE = U + \lambda_1 h_1 + \lambda_2 h_2 + \dots + \lambda_m h_m$$

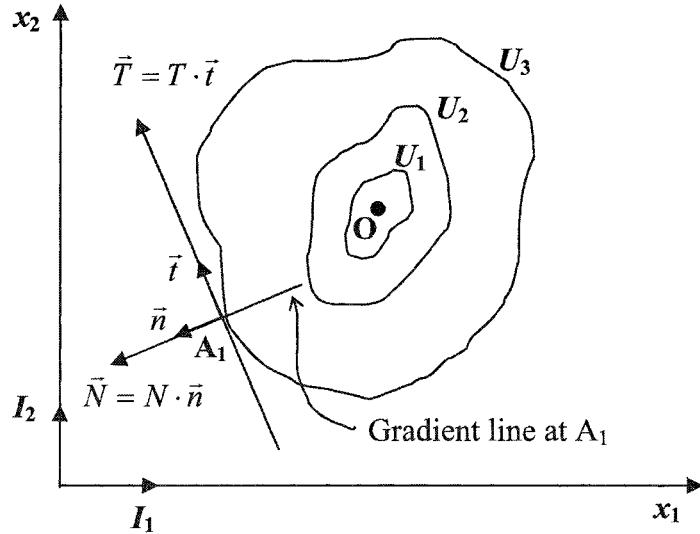
Constant factors λ_i , called *Lagrange multipliers*, are additional unknown parameters introduced into the Lagrange expression so that in determining the optimum values of x_1, x_2, \dots, x_n , the problem can be treated as an unconstraint optimization:

$$\text{extremize } U \Leftrightarrow \begin{cases} \frac{\partial LE}{\partial x_1} = 0; \frac{\partial LE}{\partial x_2} = 0; \dots; \frac{\partial LE}{\partial x_n} = 0 \\ \frac{\partial LE}{\partial \lambda_1} = 0; \frac{\partial LE}{\partial \lambda_2} = 0; \dots; \frac{\partial LE}{\partial \lambda_m} = 0 \end{cases}$$

4.3.3.2 Search Methods

Search methods are based on examining simultaneous or sequential trial solutions over the entire domain of feasible designs to determine which point is optimal. There are different strategies for searching the optimum point, which result in different rates of convergence in a particular design problem. The *steepest descent* technique is considered as a simple yet effective search method. Suppose an objective function with two variables x_1 and x_2 is given:

$$U = U(x_1, x_2)$$



For the given function U , contour curves of constant U are obtained by changing values of x_1 and x_2 as shown in figure below. Point O is the optimum point of function U . A_1 is an arbitrary starting point, called the *base point*. Consider the tangent and normal vectors with the unit vectors \vec{t} and \vec{n} at point A_1 , respectively, so that

$$\vec{T} = T \cdot \vec{t}$$

$$\vec{N} = N \cdot \vec{n}$$

The normal vector at point A_1 is called the gradient vector, defined as

$$\vec{N} = \vec{\nabla} U = \left(\frac{\partial U}{\partial x_1} \right) \vec{I}_1 + \left(\frac{\partial U}{\partial x_2} \right) \vec{I}_2$$

Thus, the normal unit vector can be written as

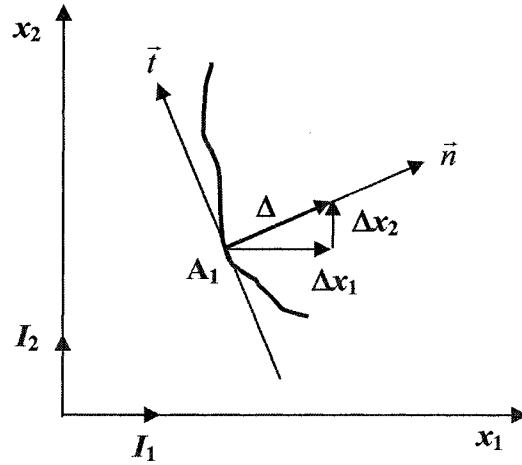
$$\vec{n} = \frac{\left(\frac{\partial U}{\partial x_1} \right) \vec{I}_1 + \left(\frac{\partial U}{\partial x_2} \right) \vec{I}_2}{\sqrt{\left(\frac{\partial U}{\partial x_1} \right)^2 + \left(\frac{\partial U}{\partial x_2} \right)^2}},$$

which indicates the direction of the gradient as the directional vector of the next step. The steepest search method can be used for multivariable problems with the increment in the gradient direction as shown in figure below. To move in the direction of the gradient vector \vec{N} , increment Δ is multiplied by the unit vector \vec{n} :

$$\vec{\Delta} = -\Delta \cdot \vec{n}$$

or

$$\vec{\Delta} = \frac{\left[-\left(\frac{\partial U}{\partial x_1} \right) \Delta \cdot \vec{I}_1 - \left(\frac{\partial U}{\partial x_2} \right) \Delta \cdot \vec{I}_2 \right]}{\sqrt{\left(\frac{\partial U}{\partial x_1} \right)^2 + \left(\frac{\partial U}{\partial x_2} \right)^2}}$$



The above equation can be written as

$$\vec{\Delta} = \Delta x_1 \vec{I}_1 + \Delta x_2 \vec{I}_2$$

where

$$\Delta x_1 = \frac{-\left(\frac{\partial U}{\partial x_1}\right)_{A_1} \Delta}{\sqrt{\left(\frac{\partial U}{\partial x_1}\right)_{A_1}^2 + \left(\frac{\partial U}{\partial x_2}\right)_{A_1}^2}} \quad \text{and} \quad \Delta x_2 = \frac{-\left(\frac{\partial U}{\partial x_2}\right)_{A_1} \Delta}{\sqrt{\left(\frac{\partial U}{\partial x_1}\right)_{A_1}^2 + \left(\frac{\partial U}{\partial x_2}\right)_{A_1}^2}}$$

Thus the new point A_2 is obtained with the following coordinates:

$$\vec{X}^2 = x_1^2 \vec{I}_1 + x_2^2 \vec{I}_2 = \vec{X}^1 + \vec{\Delta} = (x_1^1 + \Delta x_1) \vec{I}_1 + (x_2^1 + \Delta x_2) \vec{I}_2$$

The increment Δ and gradient direction are calculated in the same way until an increase in U is sensed. The direction of the gradient is then changed and Δ is reduced to $\Delta/2$. This procedure is repeated until the Δ is reduced to a value corresponding to that of the convergence criterion ε_1 . The next gradient direction is then determined, and the same steps are repeated until $\partial U/\partial x_1$ or $\partial U/\partial x_2$ is reduced to a value corresponding to that of the convergence criterion ε_2 . When x_1 and x_2 reach a minimum or maximum, their values will be equal to the roots of $\partial U/\partial x_1$ and $\partial U/\partial x_2$. Hence, the smaller the partial derivative $\partial U/\partial x_i$ is, the closer the x_i values are to the roots.

4.3.3.3 Linear Programming

The method of Linear Programming (LP) is applicable to a linear objective function subjected to a number of linear constraints. Thus, the problem can be formulated as:

$$\text{Exteremize } U = c_1x_1 + c_2x_2 + \cdots + c_nx_n = CX ;$$

Subject to the linear constraints as

$$\begin{cases} f_i = b_{i1}x_1 + b_{i2}x_2 + \cdots + b_{in}x_n \leq, =, \geq a_i & i = 1, 2, \dots, m < n \\ x_1, x_2, \dots, x_n \geq 0 \end{cases},$$

where c_i are known respective weights or costs of the design variables, and constant factors b_{ij} , and a_i assign the weights and levels of constraints.

Linear programming problems with a low number of design variables could be solved graphically. However, for practical problems with relatively large number of variables algebraic techniques are used.

The algebraic method is designed to extend the graphical method results to multi-dimensional LP problem. The four-step procedure for the algebraic solution is mentioned below:

- 1) **Constructing the Boundaries of the Constraints Set:** Transform all inequalities (except the restricted condition on each variable, i.e., x_i 's ≥ 0 , or x_i 's ≤ 0) to equalities by adding or subtracting non-negate *slack* variables.
- 2) **Finding All the Vertices:** If the number of variables (including slack variables) is more than the number of equations, then set a number of (*total number of variables T minus number of equations E*) to zero. The variables to set to zero are the slack and *restricted variables* (any x_i 's ≥ 0 , or x_i 's ≤ 0) only. After setting these many variables to zero, then find the other variables by solving the resulting squared system of equations.
- 3) **Checking For Feasibility:** All slack variables must be non-negate and check for restricted condition on each variable, if any. Each feasible solution is called a *Basic Feasible Solution*, which is a corner point of the feasible region.
- 4) **Selecting the Optimal Corner Point:** Among all feasible solutions (i.e., feasible corner points), find the optimal one (if any) by evaluating the objective function. There might be multiple optimal solutions.

Each solution to any system of equations is called a *Basic Solution* (BS). Those BS that are feasible are called *Basic Feasible Solutions* (BFS). In every basic solution, the variables that are set equal to zero are called the *Non-Basic Variables* (NBV), all other variables computed by using the system of equations are called *Basic Variables* (BV).

Number of Basic Solutions: After converting all inequalities into equalities, let T = total number of variables including slack variables, E = number of equations, and R = total number of slack variables and restricted design variables, then the maximum number of basic solutions is:

$$\frac{R!}{(T-E)! \cdot (R+E-T)!}$$

where ! stands for Factorials.

EXAMPLE:

$$\text{Minimize } x_1 + 2x_2 ; \text{ subject to: } \begin{cases} x_1 + x_2 \geq 4 \\ -x_1 + x_2 \leq 2 \\ x_1 \geq 0 \end{cases}$$

Introducing the slack variables, we have:

$$\begin{cases} x_1 + x_2 - s_1 = 4 \\ -x_1 + x_2 + s_2 = 2 \end{cases}$$

Here we have 2 equations with 4 unknowns. Since only x_1 is restricted, we must set any two of the variables s_1 , s_2 , and x_1 to zero. Solving the 6 resultant systems of equations, we have:

x_1	x_2	s_1	s_2	$x_1 + 2x_2$	
0	4	0	-2	infeasible	
0	2	-2	0	infeasible	
1	3	0	0	7	

4.3.3.4 Multicriterion Optimization

In practical engineering design problems, there are often several criteria or design objectives that must be extremized. If the objectives are opposing, then the problem becomes finding the best possible design that still satisfies the opposing objectives. An optimum design problem must then be solved, with multiple objectives and constraints taken into consideration. This type of problem is known as either a *multiobjective*, *multicriteria*, or a *vector optimization* problem. As an example, in the design of an automobile an engineer may wish to maximize crash resistance for safety and minimize weight for fuel economy. This is a multiobjective problem with two opposing objectives, that is, a step towards improving one of the objectives, increasing crash resistance, is a step away from improving the other, decreasing weight.

A Multiple-Objective (MO) optimum design problem is solved in a manner similar to the Single- Objective (SO) problem. In a SO problem, the idea is to find a set of values for the design variables that, when subject to a number of constraints, yield an optimum value of the objective (or cost) function. In MO problems, the designer tries to find the values for the design variables which optimize the objective functions simultaneously, in this manner the solution is chosen from a so-called *Pareto optimal set*, and it yields a set of possible answers from which the engineer may choose the desired values of the design variables. In general, for multiobjective problems the optimal solutions obtained by individual optimization of the objectives (i.e., SO optimization) is not a feasible solution to the multiobjective problem.

A set of points is said to be *Pareto optimal* if, in moving from one point to another point in the set, any improvement in one of the objective functions from its current value would cause at least one of the other objective functions to deteriorate from its current value. The Pareto optimal set yields an infinite set of solutions, from which the engineer can choose the desired solution. In most cases, the Pareto optimal set is on the boundary of the feasible region.

One simple approach of solving a multicriterion optimization is called *Weighting Objective Method*. This method takes each objective function and multiplies it by a fraction of one, the *weighting coefficient* which is represented by w_i . The modified functions are then added together to obtain a single cost function, which can easily be solved using any SO method. Mathematically, the new function is written as

$$U(\bar{X}) = \sum_{i=1}^N w_i f_i(\bar{X})$$

where

$$0 \leq w_i \leq 1 ; \text{ and } \sum_{i=1}^N w_i = 1$$

In this method, the weighting coefficients are assumed beforehand. The coefficients are then varied to yield a set of feasible optima, the Pareto Optimal set. The designer is expected to pick the values of the variables from this set of solutions. In general, there is no guarantee that this method will yield the entire Pareto set.

More advance techniques have been suggested for multicriterion optimization. However, their efficiency and generality are still under investigation, and no universal algorithm has been established for this kind of optimization, yet.

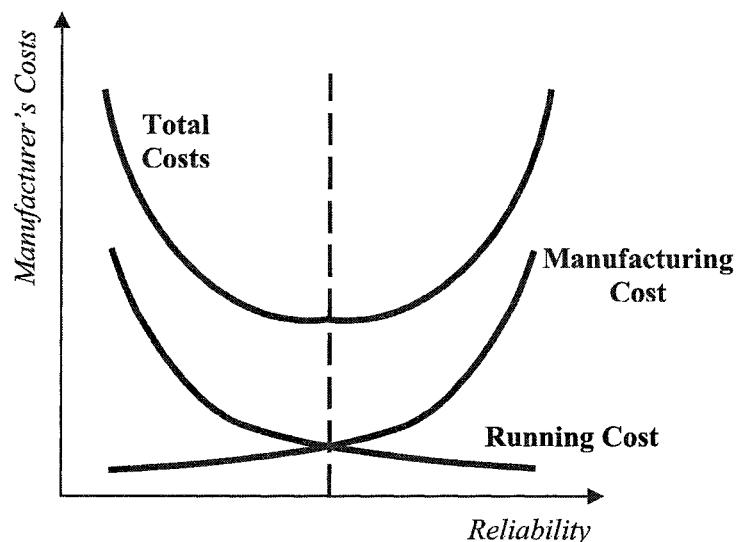
4.3.4 Design for Reliability

The cost of ownership of a product or system comprises two components. The first is the capital cost, and the second is the cost of operating, administrating, maintaining, and replacing the product or system. The second outlay, i.e., the *running cost*, can often exceed the capital cost, and is a function of *reliability*. As a matter of fact, because of the disastrous financial consequences of equipment failure, most customers specify reliability conditions tightly.

In general reliability adds cost to a product, and although unreliability carries with it a cost penalty, the optimum level of reliability is always a compromise between the two. The general relationship between reliability and cost is shown in figure below.

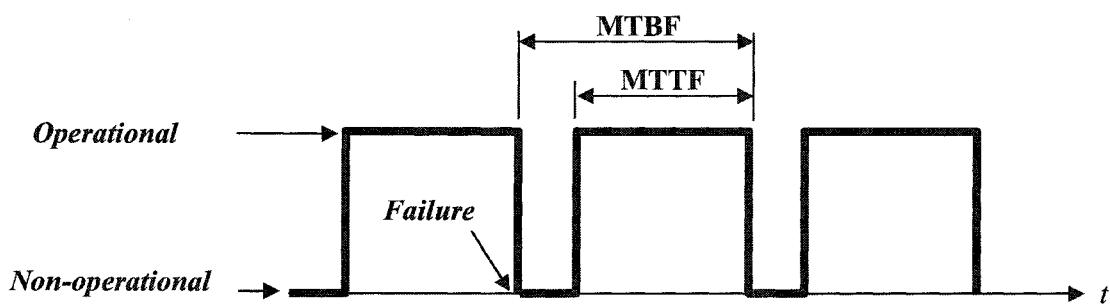
Reliability is defined as the probability that a system or component will successfully perform its required function for:

- A given range of operating condition; and/or
- A specific environmental condition; and/or
- A prescribed economic survival time.



4.3.4.1 Component Reliability

The reliability study is concerned with the causes, distribution, and prediction of *failure*. Failure is defined as the termination of the ability of a component (or system) to perform its required function. A method of describing the occurrence of failures is to state the mean time between successive failures. The two terms used, the *mean time between failures* (MTBF) and the *mean time to fail* (MTTF) are illustrated in figure below. The difference between the two terms



is the repair time:

$$\text{MTTF} + \text{mean time to repair} = \text{MTBF}$$

In the case of components that are not repaired but replaced, the repair time is considered zero and MTTF and MTBF are the same.

Reliability of a component (or system), $R(t)$, as a function of time is the probability of no failure by the time t :

$$R(t) = 1 - F(t) = 1 - \int_0^t f(t) dt ;$$

where $f(t)$ is the failure probability density function at any instant t , and $F(t)$ is the cumulative density function of the failure (or failure probability) up to the time t . Properties of the probability density function yield:

$$R(0) = 1 ; \text{ and } R(\infty) = 0$$

The *instantaneous failure rate* at each time t is defined as the ratio of the rate of failure probability change to the no-failure probability (or reliability):

$$\lambda(t) = \frac{\frac{dF(t)}{dt}}{R(t)} = \frac{f(t)}{R(t)}$$

Substituting the $R(t)$ formulation in the above yields:

$$\lambda(t) = -\frac{1}{R(t)} \frac{dR(t)}{dt} = -\frac{d}{dt} \ln R(t) .$$

Integrating both sides of the above equation yields

$$\ln R(t) = - \int_0^t \lambda(t) dt$$

or

$$R(t) = \exp \left[- \int_0^t \lambda(t) dt \right]$$

For a constant failure rate λ the reliability function is calculated as:

$$R(t) = e^{-\lambda t}$$

The constant *failure rate* λ is the frequency of system failures, and thus is the reciprocal of MTBF. As an example, records kept on 1000 airplane engines of the same type show an average life of 14000 flying hours. Hence, the probability that one such engine will survive a transatlantic flight of 7 hours is:

$$MTBF = \frac{1}{\lambda} = 14000 \text{ hours} \Rightarrow \lambda = \frac{1}{14000}$$

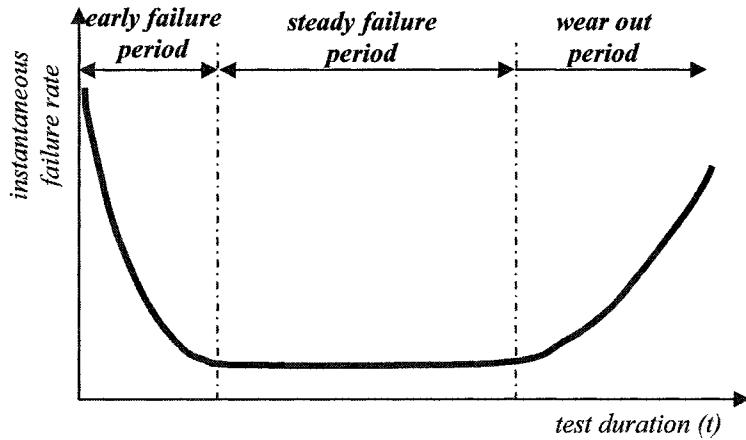
$$R(t) = e^{-\lambda t} \Rightarrow R(7) = \text{probability of surviving 7 hours} = e^{-7/14000} = 0.9995 \text{ or } 99.95\%$$

The real situation in the above example is obviously more complex since a modern passenger aircraft will always have more than one engine and could continue flying without one engine.

Different functions have been suggested for the failure probability density. One popular function, called Bathtub Hazard Distribution, is defined by

$$f(t) = bt^{b-1} \exp \left\{ - \left[e^{b^b} - t^b - 1 \right] \right\} \quad b > 0, t \geq 0$$

that results in the reliability function as



$$R(t) = \exp\left\{-\left[e^{t^b} - 1\right]\right\}.$$

The instantaneous failure rate will then be derived as

$$\lambda(t) = be^{t^b}$$

Typical *bathtub* instantaneous failure rate function is shown in figure above. During the *early failure period*, it is very likely that failures will occur due to various imperfections acquired during the manufacturing process, due to design faults or misuse. This period is often covered by the manufacturer's guarantee.

The *steady failure period* is usually relatively long and the failure rate is approximately constant. During this period, failures usually occur at a relatively low rate but from a wide range of causes.

The beginning of the *wear out period* corresponds to the end of the useful working life. All products wear out due to a variety of time-dependent mechanisms.

4.3.4.2 System Reliability

Performance of a system with n components depends on the performance of each and every component. Hence, the reliability of a system can be obtained from the knowledge of the reliabilities of its components, and it depends on the interrelationships among components.

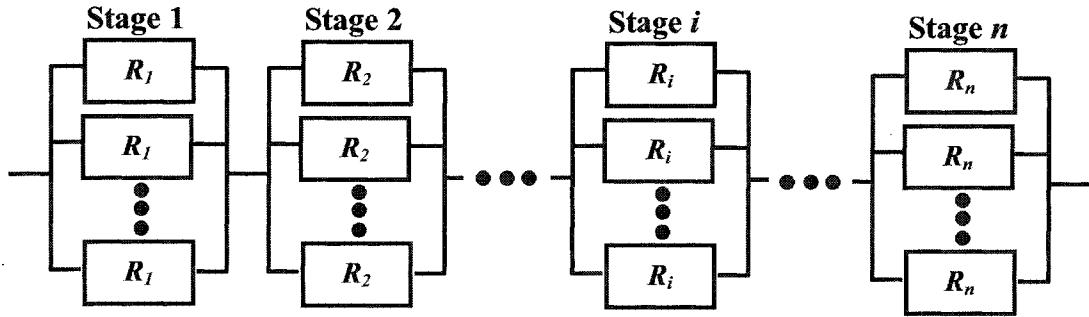
Serial Connections: Consider a system comprising n components connected in series such that failure of either causes system failure. The reliability of the system is then the product of the component reliabilities:

$$R_{system} = R_1 \cdot R_2 \cdot \dots \cdot R_n = \prod_{k=1}^n R_k$$

Parallel Connections: Consider another system again comprising n components but so connected that the system fails only if all components fail together. This is a parallel system, and the reliability of the system is the probability that either or all components succeed. Hence,

$$R_{system} = 1 - (1 - R_1) \cdot (1 - R_2) \cdot \dots \cdot (1 - R_n) = 1 - \prod_{k=1}^n (1 - R_k)$$

For parallel systems the system reliability increases as the number of components increases. However, increasing the number of parallel components increases the initial cost, weight and volume of the system as well as the maintenance requirement.



Hybrid Connections with Parallel Redundancy: In order to achieve the required degree of reliability, it may be necessary to duplicate components so that if one component fails there is another available to carry on working. The following are examples of this case, which is called *redundancy*.

- Altimeter in aircraft. One is insufficient in case of malfunction. Two would pose the problem of identifying which is giving the correct reading if they differ. Three are hence required. This is called *active redundancy*.
- In the operating rooms of hospitals. If the main electricity fails, provision is made for switching to an emergency supply. This is called *standby redundancy*.
- The spokes of a traditional bicycle wheel. Several spokes can break and the wheel will still function. This is called *partial redundancy*.

A system with parallel (active) redundancy is shown in figure above. The system consists of n stages in series where the components have identical reliabilities and are connected in parallel at each stage. The system reliability is the product of reliabilities of each stage:

$$R_{system} = \prod_{k=1}^n \left[1 - (1 - R_k)^{Y_k} \right]$$

where Y_k is the number of components in each stage and R_k is the common reliability of the components.

Parametric System Reliability Evaluation: By using the parametric method, the reliability of complex systems can be easily evaluated. This method introduces a new parameter, ϕ , defined as follows:

$$\phi = \frac{1 - R}{R}$$

where R is the reliability of a component. Substitution of this parameter in the previous formulations for system reliability yields:

Serial Connection:
$$(1 + \phi_{system}) = \prod_{k=1}^n (1 + \phi_k)$$

For a two element system, $n=2$, we have:

$$\phi_{system} = \phi_1 + \phi_2 + \phi_1 \phi_2$$

Assuming $\phi \ll 1$, the higher order terms of the above equations can be neglected, and the corresponding parameter for the series system can be approximated as

$$\phi_{system} = \sum_{k=1}^n \phi_k$$

Parallel Connections:

$$\frac{\phi_{system}}{1 + \phi_{system}} = \prod_{k=1}^n \frac{\phi_k}{1 + \phi_k}$$

For small values of ϕ , term $(1 + \phi)$ can be approximated as unity. Hence, the corresponding parameter for a parallel system becomes

$$\phi_{system} = \prod_{k=1}^n \phi_k$$

Hybrid Connections with Parallel Redundancy: The corresponding parameter for each stage is $\phi_k^{Y_k}$ where Y_k is the number of components in the k^{th} stage. Then, the corresponding parameter for a hybrid system is obtained using the formulation for a series system:

$$\phi_{system} = \sum_{k=1}^n \phi_k^{Y_k}$$

Once the parameter ϕ_{system} is known, the system reliability is calculated by

$$R_{system} = \frac{1}{1 + \phi_{system}}$$

Note that the parametric method assumes that $\phi \ll 1$, and hence is appropriate for determining the reliability of systems having components with high reliability.

4.3.4.3 Optimization of System Reliability

As explained above, the overall reliability of a series system can be increased by providing redundancy at each stage. Nevertheless, increasing the number of components at each stage increases the cost and weight of the system. Hence, to design a system with high performance requires compromise between reliability and cost or weight. The problem is to optimize the reliability with respect to cost or weight by a reasonable trade-off among the various components at each stage. Three cases of reliability optimization of a multi-stage parallel redundant system can be considered as below:

- Minimum reliability for a given cost constraint;
- Minimum cost for a given reliability constraint;
- Maximum reliability for given cost and weight constraints.

4.3.4.3.1 Reliability Optimization for A Given Cost Constraint

Consider a system in which the number of stages and component reliabilities is given. It is desirable that the total cost of the system not exceed a given cost limit C_L . Hence,

$$\sum_{k=1}^n C_k Y_k \leq C_L$$

where C_k is the cost of one component. The objective function is defined as

$$U = \sum_{k=1}^n \phi_k^{Y_k} .$$

The Lagrange expression for this optimization problem is

$$LE = \sum_{k=1}^n \phi_k^{Y_k} + \lambda \left(\sum_{k=1}^n C_k Y_k - C_L \right)$$

To optimize the number of components Y_i in the i^{th} stage, differentiate the Lagrange equation with respect to Y_i :

$$\frac{\partial LE}{\partial Y_i} = \ln \phi_i \phi_i^{Y_i} + \lambda C_i = 0$$

Since only one stage is considered, the summation in the above equation is removed. The above equation can be written in the form

$$Y_i = a_i \ln \lambda + b_i$$

where

$$a_i = \frac{1}{\ln \phi_i}$$

$$b_i = \frac{\ln K_i}{\ln \phi_i}$$

$$K_i = -\frac{C_i}{\ln \phi_i}$$

And, the expression for the Lagrange multiplier λ will be obtained in the following form:

$$\lambda = e^s$$

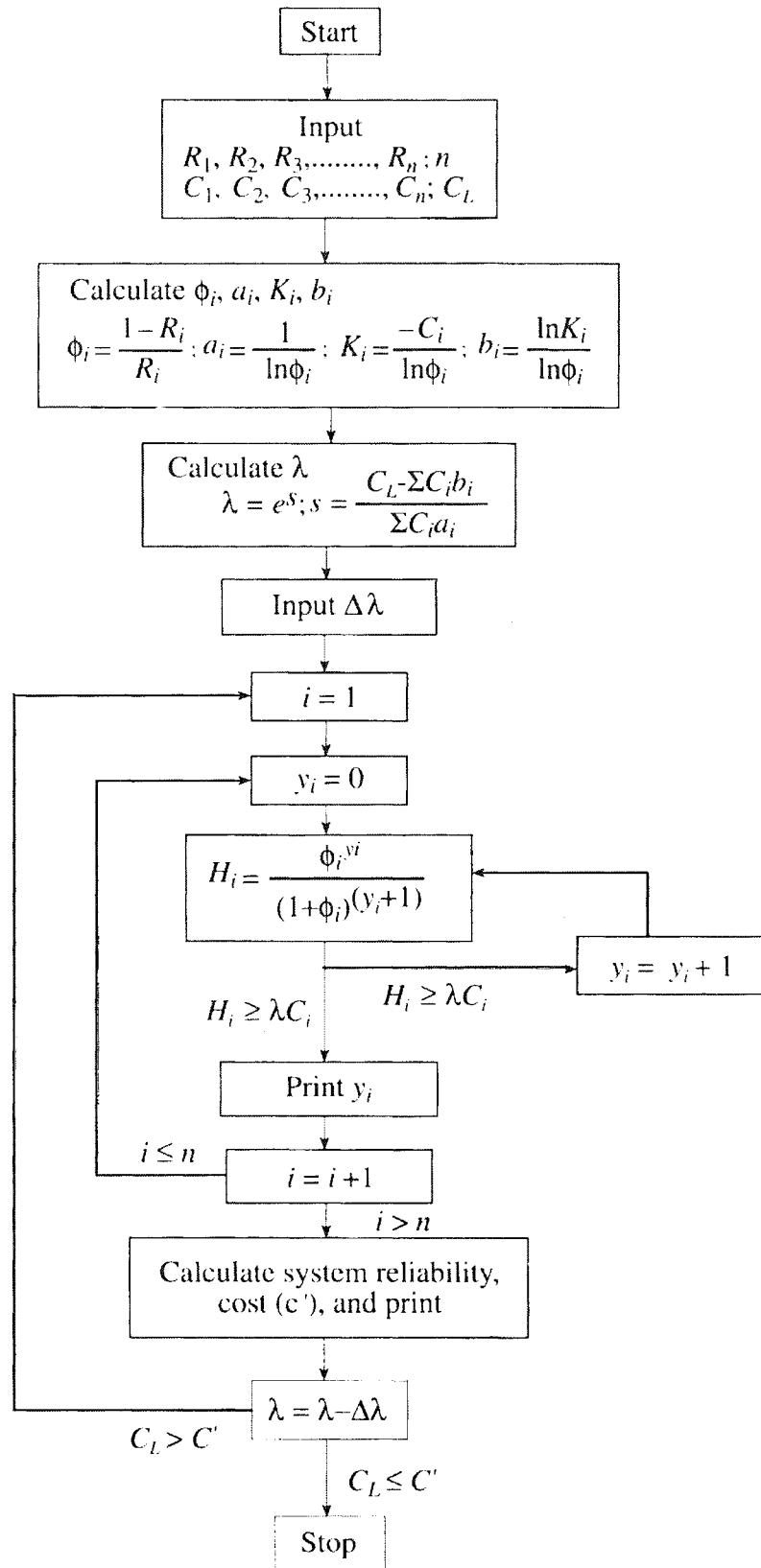
where

$$s = \frac{C_L - \sum_{k=1}^n C_k b_k}{\sum_{k=1}^n C_k a_k}$$

To obtain a sequence of numbers that will optimize the reliability for the given cost limit, it is assumed that a sequence $\bar{n} = (n_1, n_2, \dots, n_i, \dots, n_n)$ is produced for the system reliability of $R(\bar{n})$ and cost $C(\bar{n})$. Let there be another series $\bar{m} = (m_1, m_2, \dots, m_i, \dots, m_n)$ where $R(\bar{m}) > R(\bar{n})$ and $C(\bar{m}) > C(\bar{n})$. The sequence \bar{m} dominates \bar{n} , provided that the cost constraint is satisfied. A sequence of un-dominated numbers, \bar{n} , can be obtained from the smallest n_i that will satisfy the following inequality:

$$\frac{\phi_i^{Y_i}}{[1 + \phi_i]^{(1+Y_i)}} < \lambda C_i$$

By calculating λ from the previous equation, the smallest numbers that will satisfy the above inequality can be obtained. By changing the value of λ in decrements, a different sequence that corresponds to each λ can be obtained. By choosing proper decrements for λ optimum component numbers for each stage that satisfy the cost constraint can be obtained by using the iteration procedure shown in flowchart below. As seen from the flowchart, the value of λ changes with the cost and the reliability of the element. The decrements of λ should be chosen for different sets of data.



4.3.4.3.2 Cost Minimization for A Given Reliability Constraint

In this situation, the objective function that should be minimized is

$$U = \sum_{k=1}^n C_k Y_k$$

The constraint equation that should be satisfied is

$$\phi_L \leq \sum_{k=1}^n \phi_k^{Y_k}$$

where the limiting value of ϕ_L is given by

$$\phi_L = \frac{1 - R_L}{R_L}$$

where R_L is the reliability limit. The Lagrangian expression can be written as

$$LE = \sum_{k=1}^n C_k Y_k + \lambda \left(\sum_{k=1}^n \phi_k^{Y_k} - \phi_L \right)$$

To optimize the number of components Y_i in the i^{th} stage, differentiate the Lagrangian expression with respect to Y_i ,

$$\frac{\partial LE}{\partial Y_i} = C_i + \lambda \ln \phi_i \phi_i^{Y_i} = 0$$

For $i=1$, we have

$$\lambda \ln \phi_1 \phi_1^{Y_1} = -C_1 \Rightarrow \lambda = \frac{-C_1}{\ln \phi_1 \phi_1^{Y_1}}$$

Substituting λ into the differentiation equation yields

$$\phi_i^{Y_i} = K_i \phi_1^{Y_1}$$

where

$$K_i = \frac{C_i \ln \phi_1}{C_1 \ln \phi_i}$$

Using the above formulation, λ is calculated as

$$\lambda = -\frac{C_1}{S \ln \phi_1}$$

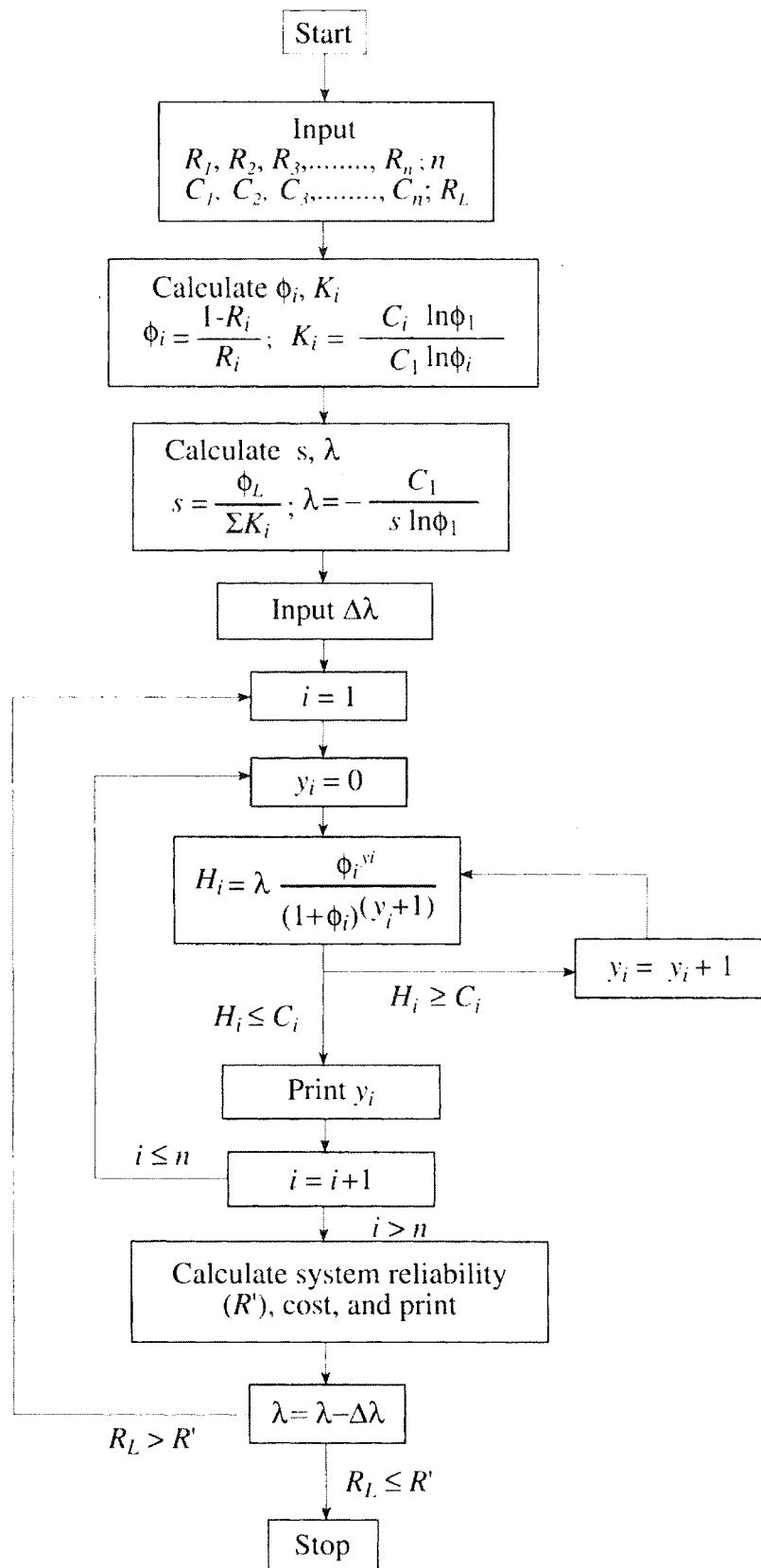
where

$$S = \frac{\phi_L}{\sum_{k=1}^n K_k}$$

From the theory of un-dominated modes, a sequence of numbers can be generated by finding the smallest number that will satisfy the following inequality:

$$\lambda \frac{\phi_i^{Y_i}}{(1 + \phi_i)^{1+Y_i}} < C_i$$

The algorithm for determining the optimum cost for a given reliability constraint is shown in figure below.



4.4 Communication in Design

In today's engineering environment, information flow is critical to both industry and business. Engineers who are incapable of preparing clear and understandable written communications tend to be relegated to passive roles in the design process. Engineers who lack the ability to write well will become information receivers, not information generators, and thus gradually find themselves out of the mainstream. Practically, engineers market their skills through the ability to communicate, particularly in written. Three important elements of written communication in design are: **a) the design proposal; b) the engineering notebook; and c) the design report.** The basic format of these communication means is addressed in sequel.

4.4.1 Engineering Proposals

The purpose of submitting a proposal is to put (*poser* means to place or put) forth (*pro* means forth) an idea or solution to a need for which the potential customer, manager or fund provider is known or thought to be interested. It can result **(a)** from a formal Request For Proposal (RFP), **(b)** from an informal suggestion from a potential customer that a proposal in an area would be favorably received, **(c)** in response to a request from in-house management, or **(d)** from an idea or need that is thought to be worthwhile, either for research or implementation, but requires funding and support from external resources.

Any proposal offers a plan to fill a need, and your reader will evaluate the plan according to how well your written presentation answers questions about *WHAT* you are proposing, *HOW* you plan to do it, *WHEN* you plan to do it, and *HOW MUCH* it is going to cost. To do this, you must ascertain the level of knowledge that your audience possesses, and take the positions of all your readers into account. You must also discern whether your readers will be members of your technical community, of your technical discourse community, or of both, and then use the appropriate materials and language to appeal to both. You might provide, for those outside of your specific area of expertise, an executive summary written in non-technical (easily accessible) language, or you might include a glossary of terms that explains technical language used in the body of the proposal, and/or attach appendices that explain technical information in generally-understood language.

The most basic composition of a proposal, as with any other written document, is simple; it needs a *beginning* (the Introduction), a *middle* (the Body of material to be presented) and an *end* (the Conclusion).

The **INTRODUCTION** presents and summarizes the problem you intend to solve and your solution to that problem, briefly and in a plain language, including the benefits the reader/group will receive from the solution and the cost of that solution.

The **BODY** of the proposal should explain the complete details of the solution: what are the alternative solutions, what is your detailed proposed solution, why your solution is preferred, how the job will be done, broken into separate tasks; what method will be used to do it, including the equipment, material, and personnel that would be required; when the work will begin; what are the steps and milestones, and when each of them will be completed. It should also present a detailed cost breakdown for the entire job.

The **CONCLUSION** should emphasize the benefits that the reader will realize from your solution to the problem and should urge the reader to action. It should be encouraging, confident and assertive in tone.

Proposals are informative and persuasive writing. The goal of the writer is not only to persuade the reader to do what is being requested, but also to make the reader believe that the solution is practical and appropriate. In persuasive proposal writing, the case is built by the demonstration of logic and reason in the approach taken in the solution.

Facts must lead logically and inevitably to the conclusion and/or the solution presented. Evidence and data should be given in a *descending order of importance*, beginning with the most important evidence and ending with the least important.

Any questions that the reader might pose should be anticipated and answered in a way that reflects the stated position of your proposal. It is important that the writer, also, considers all sides of the argument, providing other alternative solutions to the problem, but showing how the one chosen is superior to the others included.

4.4.1.1 Process

Writing a proposal does not stand alone. It must be part of a process of planning and of research on, outreach to, and cultivation of the available market and potential customers, managers, and funding organizations.

The first thing you will need to do in writing a proposal is to gather the documentation for it. You will require background documentation in three areas: *concept*, *program*, and *expenses*. If all of this information is not readily available to you, determine who will help you gather each type of information. Once you know with whom to talk, identify the questions to ask. This data-gathering process makes the actual writing much easier.

Concept

It is important that you have a complete sense of how your solution would work, with all details that the customer might be interested in. The need that the proposal is addressing must also be documented. These concepts must be well articulated in the proposal. Those who provide funds want to know that a project reinforces the overall direction towards solving the problem, and they may need to be convinced that the case for the project is compelling. You should collect background information on alternative solutions and all available resources that you have access to in the design process.

Program

Here is a checklist of the process information you require:

- the nature of the project, the subtasks, and how they will be conducted;
- the market, literature and idea surveys,
- the detailed steps of the project;
- the timetable for the project;
- the anticipated milestones and outcomes and how best to evaluate the results; and
- staffing needs and division of tasks among staff, including deployment of current knowledge and what must be gained in future.

Expenses

You will not be able to pin down all the expenses associated with the project until the program details and timing have been worked out. Thus, the main financial data gathering takes place after the narrative part of the proposal has been written. However, at this stage you do need to sketch out the broad outlines of the budget to be sure that the costs are in reasonable proportion to the outcomes you anticipate. If it appears that the costs will be prohibitive, you should then scale back your plans or adjust them to remove the least cost-effective expenditures.

4.4.1.2 Format

There are several formats to a proposal, but one that has the most flexibility and has achieved the widest acceptance is as follows.

4.4.1.2.1 Title

It is the cover page, and consists of the following:

- project title,
- customer's name (in your case, Course and Instructor),

- group identification (group number, members' name, student numbers, teaching assistant),
- organization name (your division and university),
- date.

4.4.1.2.2 Executive Summary

This is the first page of the proposal, and is usually not more than 300 words. Here you will provide the reader with a snapshot of what is to follow. Specifically, it summarizes all of the key information and is a sales document designed to convince the reader that this project should be considered for support. Be certain to include:

Problem: a brief statement of the problem or need your team is prepared to address (one or two sentences);

Solution: a short description of the project, including what will take place, how and where it will operate, and for how long (three or four sentences);

Funding Requirements: an explanation of the amount of money required for the project and what your plans are for funding it in the future (two sentences); and

Organization and Its Expertise: a brief statement of task subdivisions and activities of team members (two to three sentences).

4.4.1.2.3 Introduction

The Introduction is basically a sales pitch to interest the potential customer or funding organization in continuing further into the proposal. Hence, all proposals normally include information that establishes the background for the problem for which a solution is being proposed. Even when responding to an RFP, it is wise to include background material to ensure that the potential customer recognizes that the *proposer* understands the problems and is capable of developing a solution. Nevertheless, all statements should be concise referring directly to the aspects of the problem to be solved. The introduction part can consist of the following sections.

Statement of Need / Problem

The statement of need presents facts and evidence that support the need for the project and establishes that you understand the problems and, therefore, can reasonably address them. You want the need section to be succinct, yet persuasive. Consider the following points in stating the need and or problem:

First, decide which facts or statistics best support the project. Be sure the data you present are accurate. Information that is too generic or broad will not help you develop a winning argument for your project. Information that does not relate to the project you are presenting will cause the reader to question the entire proposal.

Second, determine whether it is reasonable to portray the need as acute. You are asking the reader to pay more attention to your proposal because either the problem you address is worse than others or the solution you propose makes more sense than others.

Third, give the reader hope. The picture you paint should not be so grim that the solution appears hopeless. The fund provider will wonder whether an investment in a solution will be worthwhile.

Fourth, avoid circular reasoning. In circular reasoning, you present the absence of your solution as the actual problem. Then your solution is offered as the way to solve the problem. For example, the circular reasoning for building fingerprint-ID lock might go like this: "The problem is that we have no lock in the market working with fingerprint ID. Building such a lock will solve the problem." A more persuasive case would cite what the proposed lock means to the ease of operation, increase of security, system durability, and market interest. The statement might refer to a survey that underscores the target customer's planned usage of the device and conclude with the connection between the proposed usage and potential benefits to enhance the product service.

The statement of need does not have to be long and involved. Short (one paragraph, for instance), concise information captures the reader's attention.

Goals and Objectives

After the problem has been described, a clear and concise statement of the *objectives* of the proposal must be given. It is often advised to include *goals* as well, but distinguish them from objectives. Goals are the abstract statements of what you hope to accomplish but usually are not quite measurable. They create the setting for what you are proposing. Objectives, on the other hand, are operational; they present specific things that you will be accomplishing in your project, and are measurable. Your objectives will form the basis for the activities of your project and will also serve as the basis for your evaluation of the project. Present measurable and specific objectives for your project. Your proposal is easier to understand (and the outcomes are much more clear) if you describe your objectives in measurable ways.

Background and Survey

In order to conduct an engineering project, you must be well aware of the information relevant to the scope of your project. This information includes the existing products or services similar or related to the subject of the project (*market survey*), history of applications and related research and theoretical foundations (*literature survey*), and alternate ideas and solutions for the problem (*idea survey* or *brain storming*). In this section, you reflect this awareness, and indicate to the reader that you know what you are proposing because you are familiar with what has preceded you. The survey you make must be deep and thorough, so that no (at least, obvious) alternate or competition solution is left out of your list.

After reviewing the background, position your project in relation to other efforts and show how your project will: **(a)** extend the work that has been previously done, **(b)** avoid the mistakes and/or errors that have been previously made, or **(c)** be unique since it does not follow the same path as previously followed.

4.4.1.2.4 Technical Body

The body of your proposal should explain the three major topics of *how*, *when*, and *how much*. Thus, it consists of the following sections

Methodology – Statement of Work (SOW)

This is the most important section of your proposal. There should be a very clear link between the methods you describe in this section and the needs/problems you have previously described. Be explicit in your writing and state exactly how the methods you have chosen will work, and how they help deal with the needs/problems that your proposal is focused on. Figures, schematics, and diagrams represent the design concepts much better than words, but they should be kept simple. All drawings must depict the system in the most descriptive views.

It is strongly advised that the product or service be presented in two separate categories: *physical description* and *functional description*. These categories may then be further subdivided as appropriate. For example, the physical description can be divided into materials, actuators, controller, circuits, etc. Once again, layouts and drawings can best illustrate what you propose, and better reduce the possibility of misinterpretation and confusion. A block diagram, showing all the functional details, should be prepared in a way that the contents of the major units and their interconnection become obvious to the reader.

If the product is complex, the integration of components into the overall system must be described in detail. Illustrations are helpful to the reader to show a typical equipment design. Information should be included on all important characteristics of the product, e.g., weight, size, power, range, accuracy, all input and output functions, operating controls, installation requirements, etc.

Mathematics should be used only where necessary to emphasize a point to substantiate a claim, and then must be clearly presented. Extensive calculations and supporting data, such as manufacturers' data sheets, component specification tables and graphs, copies of brief catalogues and brochures, performance analysis data, application notes, etc. and calculations are encouraged, and are best included in appendices so that the reader will not be distracted from the main ideas. If you want to describe the design solutions you rejected to show that your proposal is not your first or only idea, the place for these details is also in the appendix.

Never assume that your reader knows what the proposal is all about. Since you cannot be with the reader to explain confusing parts and justify claims, all statements should be made clear and easy to understand, and be supported with facts and/or clearly defined assumptions. If your proposal is confusing to the customer, s/he can only assume that you are confused, that you are not sure of what you are saying, and that your company shouldn't get the contract. Further, you

must take any opportunity in this section to explain why each part of your design is advantageous. Be objective in selecting the advantages; all advantages should be expressed in specific terms. Too often proposals broadly say, "The proposed design meets or exceeds the specification requirements, is reliable, and is light weight." Statements of this hand, if not accompanied by numbers and figures, may even have negative influence on the customer.

This section should also include a description of your development plans. The development plans must be stated in terms of specific steps to be followed. An important task not to be discarded in the statement of work is the evaluation plan. Evaluation plan is usually divided into *process evaluation* and *product evaluation*. Process evaluation plan explains the required steps for examining the trend of the project and whether or not it follows the budget and timing assignments. Product evaluation plan consists of specific activities that will take place to examine the performance of subsystems and the integrated system to assure the customer that the final product will meet the desired specifications. For design problems where either of two design solutions could be best, proposing of alternate designs should be avoided. Instead, limited parallel development of the two designs should be proposed. Your proposal should state that, from these parallel developments, the best approach will be incorporated in the final design. If you propose alternate design, you force the customer to choose the best one. If you don't know which design is better, how do you expect the customer to know? Alternate designs destroy the customer's confidence in your proposal's qualifications.

Schedule

For small projects, such as the ones in the Engineering Design course, the schedule in the proposal consists of a simple bar chart that lists all anticipated steps during the project in rows and shows the time span for each task based on the time unit. The time unit depends on the total length of the project; for example, for a 22-week project, it could be as short as days. For large-scale projects advanced techniques of planning and scheduling must be applied.

It is recommended to break down the tasks into clear and specific steps to better schedule the project. Schedules with general task items are not only uncertain in terms of their feasibility, but also do they give the impression to the customer that the *proposer* is not quite aware of detailed steps, and this can question the credibility of the entire proposal.

Budget

The budget for your proposal may be as simple as a one-page statement of projected expenses. Or, your proposal may require a more complex presentation, perhaps including a page on projected support and revenue and notes explaining various items of expense or of revenue.

As you prepare to assemble the budget, you must go back through the proposal narrative and make a list of all personnel and non-personnel (material, equipment, and facilities) items related to the operation of the project. Then, obtain the relevant costs from the related resources. Put the costs you have identified next to each item on your list.

Your list of budget items and the calculations you have done to arrive at a dollar figure for each item should be summarized on worksheets. You should keep these to remind yourself how the numbers were developed. These worksheets can be useful as you continue to develop the proposal and discuss it with the fund providers. They are also valuable tools for monitoring the project once it is under way and for reporting after completion of the grant.

A narrative portion of the budget is used to explain any unusual line items in the budget and is not always needed. If costs are straightforward and the numbers tell the story clearly, explanations are redundant. If you decide that a budget narrative is needed, you can structure it in one of two ways. You can create "Notes to the Budget," with footnote-style numbers on the line items in the budget keyed to numbered explanations. If an extensive or more general explanation is required, you can structure the budget narrative as straight text. Remember though, the basic narrative about the project belong elsewhere in the proposal, not in the budget narrative.

4.4.1.2.5 Conclusion

Every proposal should have a concluding paragraph or two. This is a good place to call attention to the future extension of the project. Further, you should address the possible *bottlenecks* of the project, and ways to prevent or overcome them. This section is also the place to make a final appeal for your project. Briefly re-iterate what you plan to do and why it is going to be successful.

4.4.1.2.6 Appendices

Appendices are usually divided into three parts: (a) bibliography and references, which include addresses of related companies that would be supplying resources to the project and related brochures and catalogues, (b) all graphs, charts, tables, data sheets, quotations, and other parts that support the main content of the technical body of the proposal, (c) letters of support and other evidence of your team's capabilities.

4.4.2 Engineering Notebook

During the design process, each engineer shall possess a notebook in which all the relevant thoughts and activities are reflected clearly and in detail. Getting the habit of recording all engineering activities might seem tedious and unnecessary to some students in the beginning. However, this is an important practice that every professional engineer must perform, for engineering notebooks are great helps as notepads, bookkeepers, diaries, brain storming tools, data files, and even legal documents when it comes to some specific circumstances such as proving the originality of an idea or patent conflicts. They are also the best references for writing the final technical report. The following notes are important in keeping an engineering notebook.

- Use a bound (stitched binding) notebook. Do not use a loose leaf or spiral bound notebook. A black or blue physics-type hardcover notebook is strongly recommended for the Engineering Design course. The notebook is not a transcription of notes; it is the original record.
- Keep your notebook contemporary. This simply means, write "everything" down, including notes from technical lectures, evolving sketches of the project, discussions with your teammates, shopping lists, trips to the electronic stores, sketches, flowcharts, graphs, etc. All calculations, data sheets, tests, and stages and versions of schedules for completing steps of the design shall be included.
- Have a reasonable classification of the material you include in your notebook. For example, have a separate section for course notes, another one for meetings, and so forth.
- Entries shall be in permanent ink, *not pencil*. Pencil will smear leaving the first pages unintelligible after a year of riding in a book bag.
- All entries shall be dated, and shall be kept up to date.
- All data is to be recorded directly into the notebook. The inclusion of all elaborate details is preferable. Notes and calculations should be done in the notebook, not on loose paper. In the case of an error, draw a single line through the incorrect data. Do Not Erase or use correction fluid. Again, all corrections should be initialed and dated.
- Use both sides of a page. Never leave any white space: "X" out or Crosshatch all unused space.
- All entries that are not purely the result of your imagination should be referenced in some way. The source or inspiration for all circuits should be noted. All entries that show a circuit, a design, a part or anything purchased from, copied from, adapted from, or inspired by a source other than your uninfluenced thoughts should contain a notation to indicate the source of the entry.
- Include all receipts for materials, telephone numbers and addresses of contacts and suppliers.
- Each notebook shall contain a table of content. Leave the first 6 pages blank for this purpose.
- Do not fill the notebook with endless printouts of electronic files that could be stored on a CD or USB key in the back of your notebook.
- The notebook should always be with you during the lecture and laboratory sessions.
- The following general format is required for your notebook:
 - At the front:
 - your name and information sufficient for return if it is lost;
 - a CD or USB key envelop with a CD or key in;

- a table of contents listing the page numbers of all entries up to the current status.
- Near the back:
 - a page of references to people (tape in their business cards);
 - a page of references for companies and suppliers;
 - a page of all full references for books cited in the notebook;
 - a page of all important and relevant web sites visited during the project;
 - a page where all library call numbers are scratched down during computer searches, even if the book is not accessed;
 - an envelope glued into the back cover for all project receipts.

4.4.3 Technical Report

In the majority of engineering projects, the final report constitutes the greatest writing challenge that the team faces. The final report is commonly the only document that describes how and why the work was accomplished, and what the results, recommendations, and conclusions are. It is often the mere document of the project that is maintained on file for future reference. From the external point of view, the final report may be the principal measure of the quality of the work, and from the internal point of view, the capability of the engineering team and even the performance of the entire department are judged by the quality of the report.

There are three important remarks with regard to the general format of an engineering final report:

- The final report of an engineering project may be read by individuals that are not intimately familiar with the work or the author. Their impressions are thus formed by the content of the report and their ability to understand what was accomplished, as well as its meaning. The challenge is to discuss the work accomplished in some terms that are understandable to a reader not directly involved in the effort, possibly not even an engineer or technical person, yet to contain the objective engineering spirit. It is very important to know the audience and to prepare the report so that it is completely understandable to that group of people.
- Establishment and maintenance of a logical content flow are crucial quality factors in a technical report. If the reader is to understand the report, the presentation of information must follow some thread of logic. The material presented should be introduced in chronological sequence within each division of the report, and periodic *signposts* should be provided so that the reader always knows where s/he has started from and where s/he is going on her/his journey through the report. This goal can be achieved by stating what is coming next in the report in each major subdivision and by beginning each paragraph with a topic statement.
- Formal engineering reports are normally written in third person, past tense. Hence, personal pronouns should very rarely be used. *Objectivity* must be present throughout the entire report. The literature must indicate the fact that the report has been prepared in a purely objective, impersonal manner. Results are judged on the basis of existing, applicable theory and previous experiments. No exaggeration or compliment of the results is allowed. Opinions are expressed only when existing knowledge fails.

Most of the issues addressed in writing the engineering proposals are also valid in writing the technical reports. The following is the accepted format for the final report of the Engineering Design course.

4.4.3.1 Report Covers

The cover of a final report is similar to that of the design proposal. However, some special cover pages will be provided to students that already contain the organization logo. Other information including project title, course name and number, instructor, group identification, and date shall be printed on this cover page.

4.4.3.2 Report Preamble

Use separate pages for each section of the preamble. These pages will be numbered with Roman numerals at the bottom of the page. The preamble consists of the following in order:

- **Title page:** Use the same format as for the cover.
- **Acknowledgments:** List all those who assisted the group.
- **Abstract:** Make this a concise but comprehensive summary of the project in 200 words or less. The problem to be solved should be identified, along with the solution and the most important results and conclusions. Figures and tables should not be referred to in the abstract. Equations are also very rarely used in an abstract. It is naturally best to write the abstract last, scanning the report body for essential information and presenting this information in as few words as possible.
- **Table of Contents:** Should include every titled section in the report from Acknowledgments to Appendices in the following order: Acknowledgments, Abstract, Table of Contents, Notation, Abbreviations, Introduction, Sections of the body of the report, Conclusions, References, Bibliography, Tables, Figures, Appendices.
- **Notation:** All symbols especially those used in formulae should be defined here with their units where applicable. Some logical order should be used.
- **Abbreviations:** All abbreviations used in the text should be spelled out in the first occurrence and abbreviated thereafter. All abbreviations should be defined in alphabetical order.

4.4.3.3 Body of the Report

- **Introduction:** The background, perspective and motivation for the current work should be given in the introduction. The format is similar to the proposal. This section may also refer to literature references which pertain the project but not in great detail at this stage.
- **Project Concept and Design Parameters:** Describe the project concept in detail. Do not assume the reader is familiar with the work.
- **Perspective:** Introduce any theory or history that relates to the problem. Use equations freely. Equations may be repeated as required in later sections for the convenience of the reader. Teams are allowed to give this section the name that is most appropriate for their project. Regardless of how this section is handled, make sure to communicate any limitations placed on the design by basic physics or the current state of technology.
- **Subsystems:** Each subsystem of the design shall be described in its own section of the report. Cover all aspects of the work in a logical sequence. Be sure to include the team's assessment of the problem, elaborate on theoretical concerns where appropriate, describe the solution chosen, and reference any supporting calculations or computer programs that appear in appendices. Be sure to note areas where the subsystem may have been improved or any other suggestions arising from the subsystem in question. You can include small figures in the text only if reducing their size does not reduce their technical value. Never reduce figures that were used to read design point data. Place all full page figures in the figure section described below. Leaving a space in the text, cutting and pasting figures in, and then photocopying (set on light) is the easiest method of inserting a figure cleanly. Scanning the figure and including it in the file is another option. Figures drawn by hand (with ink and a ruler or French curves) on full pages at a large scale and then photocopy reduced look particularly sharp and no CAD package is required.
- **Integration:** Summarize the stages of integration and any problems that were encountered and how they were solved.
- **Conclusions:** Summarize the work briefly and concentrate on results and inferences. Can the team suggest how the project may be improved? In what areas would the study best be extended? What are the limitations or restrictions imposed by using the presented methods? What specifications are most limiting or where could they reasonably be modified or best clarified to simplify or improve the design?

- **References:** Number the references in the order they are referenced in the text. Use the following style (IEEE format):

- [9] D. E. Byrd, *Feathered Flight*, London: Longmans, 2nd ed., 1985, pp. 201-205.
- [10] M. T. Head, "Optical Transmission at Negative Frequencies," *IEEE Trans. Robotics*, vol. 94, pp. 124, Jan. 1990.
- [11] E. E. Reber, R. L. Mitchell, and C. J. Carter, "Oxygen absorption in the Earth's atmosphere," Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (4230-46)-3, Nov. 1968.

Book titles are italic and the actual pages referenced should be given. If in different references, different sections of the same book are referenced then separate references should appear. The Ibid, Op cit, method may be used or full reference information may be given but in either case the page number referenced must be given. Journal paper titles are enclosed in quotes and the page number referenced may be omitted.

- **Bibliography:** This is a list of the material that does not appear in the References section but was important to background research. List bibliography entries alphabetically by the author's last name. The same format may be used for books and journals as in the references section.
- **Tables:** Students who complain they would not know what to put in a table do not understand that tables are for the concise summary of information for easy comparison - any information. If in a text description one finds a series of options and each option must be considered from more than one aspect, a table should be used to shorten the text section by putting the options down the side and aspects across the top. In this case, options can be compared easily. In this case, the text need only introduce the table and point out the main result. Number tables consecutively as they are referred to in the text. If the numbering scheme includes the section number from which it is referenced, i.e. Table 3.2 being the second table referenced in section 3, it makes integrating the report among several students easier. Please state units for all quantities including the applicable label such as radians, or degrees for non-dimensional quantities where there may be confusion.
- **Figures:** All graphs, drawings, sketches etc fall under the designation Figures. If anything is more clearly explained symbolically with lines and symbols than a quantity of text, the text should be removed and a figure inserted. Avoid fold outs-material larger than 8.5" × 11". If necessary, such material can be placed in a pocket at the back of the report along with computer disks.
- **Appendices:** Many types of information essential to the documentation of the project but too bulky to be placed in the regular sections of the report may be placed in appendices. Computer printouts, extensive sample calculations and such may be included in the appendices. In general, detailing the hardware and software in the final report depends on the nature of the projects and the designated readers. For AER 201Y, all teams are required to include a complete pin by pin description of the hardware as well as algorithmic explanation of the software in appropriate appendices if not already given elsewhere in the report. Teams are also required to produce a set of Standard Operating Procedures (SOP) for their project, including all the information necessary for a technically capable person to fully operate the project hardware. A list of all adjustment tools and supplementary materials should be included at the beginning of the SOP Appendix.

4.5 Resources

- M. N. Horenstein, *Design Concepts for Engineers*, NJ: Prentice Hall, 2nd ed., 2002.
- B. Hyman, *Fundamentals of Engineering Design*, NJ: Prentice Hall, 2nd ed., 2003.
- G. C. Andrews, J. D. Kemper, *Canadian Professional Engineering Practice and Ethics*, Toronto: Harcourt Canada, 2nd ed., 1999.
- A. Kusiak, *Engineering Design: Products, Process, and Systems*, London: Academic, 1999.
- K. S. Hurst, *Engineering Design Principles*, NY: John Wiley and Sons, 1999.
- I. C. Wright, *Design Models in Engineering and Product Design*, London: McGraw-Hill, 1998.
- B. S. Dhillon, *Advanced Design Concepts for Engineers*, PA: Technomic Publishing Company, 1998.
- R. Birmingham, *Understanding Engineering Design: Context, Theory, and Practice*, NY: Prentice Hall, 1997.
- A. Ertas, J. C. Jones, *The Engineering Design Process*, NY: John Wiley & Sons, 2nd ed., 1996.
- D. L. Marston, *Law for Professional Engineers*, Canadian and International Perspectives, McGraw-Hill, 3rd ed., 1996.
- J. Walton, *Engineering Design: From Art to Practice*, MN: West Publishing Company, 1991.
- W. L. Chapman, A. T. Bahill, A. W. Wymore, *Engineering Modeling and Design*, FL: CRC Press, 1992.
- C. Hales, *Managing Engineering Design*, NY: John Wiley and Sons, 1993.
- MIT 2.007 Design Course, <http://pergatory.mit.edu/2.007/>



Chapter 5

Circuits

5.1 Safety First!	1
5.1.1 Electric Shock.....	1
5.1.2 First Aid for Electric Shock.....	1
5.1.3 120 Volt AC Power	2
5.2 Electronic Components	4
5.2.1 Resistor	4
5.2.1.1 Ohm's Law	4
5.2.1.2 Kirchoff's Current Law.....	5
5.2.1.3 Kirchoff's Voltage Law	5
5.2.1.4 Voltage Divider.....	5
5.2.1.5 Voltage to Current Converter.....	6
5.2.1.6 Current to Voltage Converter.....	6
5.2.1.7 Power and Energy	6
5.2.1.8 Standard Values	7
5.2.1.9 Resistor Types.....	8
5.2.2 Capacitor.....	11
5.2.2.1 Specifications.....	11
5.2.2.2 Capacitor Types	12
5.2.3 Inductor.....	14
5.2.3.1 Characteristic of Inductors	14
5.2.3.2 High Frequency Coils	15
5.2.3.3 The Toroidal Coil.....	16
5.2.4 Diode	17
5.2.4.1 Types and Applications of Diodes	18
5.2.5 Transistor.....	22
5.2.5.1 Operation	23
Common Emitter Amplifier	26
5.2.5.2 Types of Bipolar Transistors.....	27
5.2.5.3 Junction Field-Effect Transistor (JFET)	30
5.2.5.4 Metal Oxide Semiconductor Field-Effect Transistors (MOSFET)	31
5.2.6 Switches and Relays	32
5.2.7 Batteries	34
Design Considerations	36
5.3 Digital Electronics.....	37
5.3.1 Number Systems.....	37
5.3.1.1 Binary Mathematics	38
5.3.1.2 Parity Method for Error Detection	39
5.3.2 Logic Gates.....	40
5.3.2.1 Digital Logic Gate ICs	42
5.3.3 Combinational Logic	44
5.3.3.1 Combinational Devices	45
5.3.3.1.1 Multiplexers (Data selectors) and Bilateral Switches.....	45
5.3.3.1.2 Demultiplexers (Data Distributors) and Decoders	46
5.3.3.1.3 Code Converters (Decoders)	46
5.3.3.1.4 Arithmetic Logic Units (ALUs)	47

5.3.3.1.5	Digital Comparators.....	47
5.3.4	Sequential Logic.....	48
5.3.4.1	SR Flip-Flops.....	48
5.3.4.2	JK Flip-Flops.....	49
5.3.4.3	D Flip-Flops.....	50
5.3.4.4	Registers	50
5.3.4.5	The 555 Timer IC	50
5.3.4.5.1	Basic Astable Operation	51
5.3.4.5.2	Basic Monostable Operation.....	52
5.3.4.5.3	Practical Notes	52
5.3.5	Practical Remarks.....	53
5.3.5.1	Powering Logic ICs	53
5.3.5.2	Power Supply Decoupling	53
5.3.5.3	Unused Inputs.....	53
5.3.5.4	Automatic Power-Up Clear (Reset) Circuits	54
5.3.5.5	Pullup and Pulldown Resistors	54
5.4	Signal Conditioning	56
5.4.1	Signal Protection	56
5.4.2	Signal Amplification	57
5.4.2.1	Various Op Amp Circuits	59
5.4.2.1.1	Buffer (Unity Gain Amplifier).....	59
5.4.2.1.2	Inverting Amplifier.....	60
5.4.2.1.3	Non-inverting Amplifier.....	60
5.4.2.1.4	Summing Amplifier	60
5.4.2.1.5	Difference Amplifier.....	61
5.4.2.1.6	Integrator.....	61
5.4.2.1.7	Differentiator	62
5.4.2.2	Op Amp Specifications and Types	62
5.4.2.3	Applications.....	64
5.4.2.3.1	Op Amp Output Drivers (for loads that are either ON or OFF).....	64
5.4.2.3.2	Comparator Output Drivers	65
5.4.2.3.3	Voltage-to-Current Converter	65
5.4.2.3.4	Precision Current Source	65
5.4.2.3.5	Current-to-Voltage-Converter.....	65
5.4.2.3.6	Over-voltage Protection	66
5.4.2.3.7	Sample-and-Hold Circuits	66
5.4.2.3.8	Peak Detectors	67
5.4.2.4	Practical Notes	67
5.4.2.4.1	Powering Op Amps.....	67
5.4.2.4.2	Protecting Op Amps.....	68
5.4.2.4.3	Voltage and Current Offset Compensation	68
5.4.3	Signal Filtering.....	69
5.4.3.1	Basic Notions.....	70
5.4.3.2	Basic Passive Filters	71
5.4.3.3	Filter Specifications	71
5.4.3.4	Integrated Active Filter Circuits	73
5.4.4	Signal Regulation	75
5.4.4.1	Voltage Regulator ICs	75
5.4.4.2	Some Applications.....	77
5.4.5	Analog / Digital Signal Conversion	77
5.4.5.1	ADC and DAC Basics	77
5.4.5.2	Digital-to-Analog Conversion ICs.....	78
5.4.5.3	Analog-to-Digital Conversion ICs.....	79
5.4.5.4	Triggering Logic Responses from Analog Signals	80
5.4.5.5	Using Logic to Drive External Loads	80
5.4.6	DC-DC Voltage Conversion	81

5.4.6.1	Conversion Principles	81
5.4.6.2	Converter Types.....	81
5.4.6.2.1	Buck Converter	82
5.4.6.2.2	Boost Converter	82
5.4.6.2.3	Buck-Boost Converter.....	83
5.4.6.2.4	Cuk Converter	83
5.4.6.2.5	Charge-pump Converter.....	84
5.4.6.2.6	Flyback Converter.....	84
5.4.6.2.7	Forward Converter	85
5.4.6.3	Efficiency.....	86
5.4.6.4	Operating Frequency.....	86
5.5	The AER201 Utility Board.....	87
5.6	Practical Notes.....	88
5.6.1	Device Safety and Handling Precautions.....	88
5.6.2	Basic Measurements	88
5.6.2.1	Measuring Voltages	88
5.6.2.2	Measuring Currents.....	89
5.6.2.3	Measuring Resistance	89
5.6.2.4	Measurement Errors.....	89
5.6.3	Troubleshooting the Circuits	90
5.6.4	Electronic Symbols.....	91
5.6.5	Device Data	92
5.7	Resources	101

5.1 Safety First!

There are two major hazards in working with circuits: chemicals and voltages. On the subject of chemicals, the following can be said in general:

- Nearly every chemical involved in circuit work is toxic.
- Do not trust lack of labeling; treat everything as toxic.
- Never open sealed units like capacitors or batteries.
- Never hold an electronic component, wire or anything in your mouth.
- Minimize breathing solder-flux fumes - work in a ventilated area.
- Never eat while working on circuits.
- Never set food on a table being used for circuits.
- Always wash your hands thoroughly with soap and water after working.

Any voltage above 30-50 Volts can be lethal. There are several ways to generate dangerous voltages and these should be avoided or care must be taken not to touch any of the potentially live contacts:

- During electrical contact with an inductor - even with an ohmmeter!
- Anything involving transformers, particularly in breaking connections,
- Remember the other, unused, windings of a transformer are live too,
- Even low voltage DC brush motors when running,
- AC voltages and diode networks,
- Capacitors in any high voltage equipment, even long after turning off.
- Never work alone!

When working in situations where these conditions may occur, be careful to clear debris - especially conductive debris like bits of wire, metal rulers, tools - away from the item being worked on. Make sure not to leave other contacts or lead wires dangling, particularly for a transformer, even for testing. The combination of dangling wires and conductive debris are the cause of many mysterious, high-voltage blue flashes where a student was not aware there was a circuit, let alone a short.

Be especially careful of DC voltages because they cause muscle lock and may prevent a person from releasing whatever they have touched. AC voltages cause muscle convulsions and are more likely to make a person twitch and release the connection. When working in an area with many sources of high voltage, Murphy's Law states a person will invariably twitch and involuntarily touch a more dangerous voltage.

5.1.1 Electric Shock

If a person appears to be experiencing a shock: either jumping in pain, yelling but making no movement to release, or locked rigidly, possibly quivering slightly, with a contorted facial expression or oddly distorted muscles in their arms -

Do Not Touch The Victim.

The person trying to help can experience a shock from them and may too become locked. The best procedure is to use a non-conductive rope, belt, scarf or such to loop the victim and pull them off the live contact. If nothing appears handy, the rescuer should use a quarter-back sack maneuver - that is move fast and bowl over the person receiving the shock. The rescuer should aim for the torso and make sure they have sufficient momentum to carry both clear of the live contact. The rescuer should be careful not to over-do it and give some thought to the landing spot before starting - an electrocuted person will not thank a rescuer who impales them on a piece of ready-rod.

5.1.2 First Aid for Electric Shock

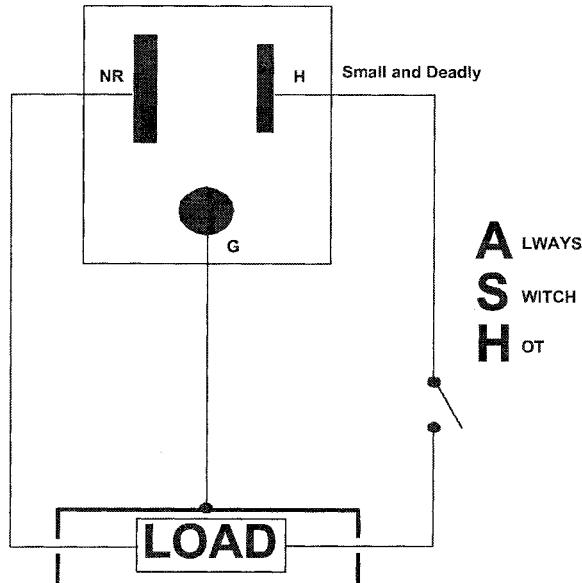
If the person is not fully conscious this is very bad, call 911 immediately (run). Keep the victim warm - use a blanket or several winter coats. Seek immediate medical attention (911) if there are any physical manifestations of the shock - shaking, burn marks, or chest pains. Even for shocks that appear to have had no impact at all, do not leave the victim alone: talk to them, work with them, walk them home. A shock victim can experience a heart attack some time after the incident. Never work alone!

5.1.3 120 Volt AC Power

There are three connections in a common 120V outlet shown in the figure above. The three connections are labeled **H**, **NR** and **G** for Hot, Neutral Return and Ground respectively.

The Ground is connected to the Earth, using a ground rod in the basement of the building. Ground carries current only in emergency situations; there is virtually no voltage drop along the length from any point in the building to the ground rod. The Ground is guaranteed (if it is properly wired by the electrician) to be at a very low voltage relative to exposed conductive surfaces in the building such as plumbing fixtures and conduit pipes. The Ground wire is a backup, should something go wrong in a piece of equipment - it is critical for safety. The Ground wire should be connected to the metal case containing the 120 Volt portion of a project's electronics. The Ground wire is frequently bare copper but is sometimes covered with green insulation.

The Neutral Return is connected to the Earth, probably at the nearest transformer station and certainly at the generating station. Under normal circumstances, the Neutral Return is at a non-hazardously low potential relative to exposed conductive surfaces in the building. Unfortunately, if the return wire is undersized for the current it is carrying or is very long, there may be a significant voltage drop along the wire. This leaves the end near the power distribution rail at a low potential and the end near the user at a higher potential. Touching the Neutral Return and plumbing at the same time is not a good idea. The Neutral Return should be connected directly to the load in the project, insulated from the case and be unbroken - there must be no switches, relays, or fuses on the Neutral Return path. Breaking this last rule makes it very likely accidents will lead to a shock hazards. The Neutral Return is normally covered in white insulation however it is sometimes bare. The Neutral Return should always be treated with respect because wiring errors by students or electricians in the building can swap it with the Hot line. Be particularly careful with extension cords. Errors swapping the Neutral Return and Hot line can go unnoticed for years until someone relies on a particular connection.



	Ground	Neutral Return	Hot Line
Usual Insulation Color	Bare Copper	White	Black
Alternate Color	Green	Bare	Red or Blue
Connected To	Building Ground	Hydro Ground	Transformer
Typical Voltage	Almost Zero	Usually, Low	120 Volts
Usage	Backup	Return Path	Source

The Hot wire carries a 60 Hz signal averaging around 120 Volts RMS, which is 340 Volts peak to peak or 169 Volts peak, relative to the Neutral Return. The Hot line should be broken by any switches, fuses or relays. **ASH** - *Always Switch the Hot* - or a short or fire is very likely result if there was any problem. To do otherwise poses a shock hazard to people. The Hot line is always insulated in some way, usually with continuous black insulation.

In a typical home or industrial setting, there are two Hot phases, 180 degrees out of phase - 240 Volts between them. The two phases are used for high power appliances such as electric stoves, hot water heaters, and air conditioners. It is common in a laboratory or kitchen to find a "split duplex". In this case the Ground and Neutral Return are the same for the two plugs in the duplex however one plug's Hot line is actually 240 Volts relative to the other. This allows a toaster and microwave to be plugged in at the same plug in a kitchen. The second phase is also insulated and can have either red or black insulation.

In the new Aerospace Design Lab, there are three phases of 120 Volts in the plugs along the ceiling, 120 degrees out of phase, which means there may be either 0 or 175 Volts between the Hot lines of any two plugs. To guard against problems, groups should please plug all equipment for their project into one power plug using a power bar.

It is quite possible for a human to survive a shock from single phase 169 Volts, provided they are not well grounded and do not grab a hot connection too firmly, yet in the same circumstances, particularly without immediate assistance, such a shock may be fatal. The Hot line should be treated with respect at all times.

As a student in AER 201, always observe the following safety precautions when dealing with 120 Volt circuits:

- Always wait until the instructor or TA can supervise before connecting.
- Never work alone.
- Never work on circuitry that has any chance of being live.
- Always shut off the apparatus, and unplug it.
- Be sure the correct plug has been unplugged - check it personally.
- Pull the cord out completely if removable.
- Keep positive control over the end of the plug while working.
- In AER 201Y, never work on a high voltage appliance (TV, monitor).
- Use a voltmeter to check for lethal residual charges in capacitors.

It is a good idea, particularly in a chaotic environment like the design lab, to take extra precautions when working on 120-Volt equipment. Students should tie the plug to a chair leg. This makes sure a partner or classmate cannot plug it in by accident when trying to connect their plug to a power bar or outlet.

When making connections with 120 Volt wiring, make sure all connections are firm. Make sure the insulation covers enough of the wire but does not enter the connector.

- Always make sure terminal screws are tight.
- When soldering, after the solder has cooled, pull on the conductor to make sure a good mechanical bond exists.
- When using marr or marrette connectors, make sure they are tight, all wires are held firmly and the insulation is neither in the connection nor exposing bare wire outside the lower bell of the connector.
- Never attempt to use a marr or marrette on stranded wire, always tin it until it is solid first and then treat it as solid.
- Do not use crimp connectors always solder or use screw terminals except for high heat connections where solder would melt and screw terminals would loosen due to repeated heating/cooling cycles.

Making mistakes with 120 Volt wiring is one of the most hazardous things AER 201 students can do in the lab. **Safety first, always.**

5.2 Electronic Components

5.2.1 Resistor

Resistance is simply defined as opposition to current flow. In any closed circuit, the amount of current that will flow is determined by two factors: 1) the amount of voltage, and 2) the amount of resistance present. A resistor is a semiconductor device manufactured in such a way that it only partially conducts electricity. It takes a certain amount of energy to make current flow through the resistor; the energy is lost from the circuit in the form of heat energy emitted from the resistor.

For a wire, the amount of resistance it contains depends on four factors:

- The material the wire is made of (ρ the resistivity factor in ohm-meters).
- The length of the wire (L in meters).
- The cross section of the wire (A in squared meters).
- The temperature of the wire.

$$R = \rho \frac{L}{A} \quad [\Omega]$$

The resistance of most materials that are good conductors *increases* slightly with increasing temperature (**positive temperature coefficient**). For copper, for instance, the resistance changes approximately 0.004 ohms per degree Celsius. For semi-conductor materials the resistance *decreases* with rising temperature (**negative temperature coefficient**).

5.2.1.1 Ohm's Law

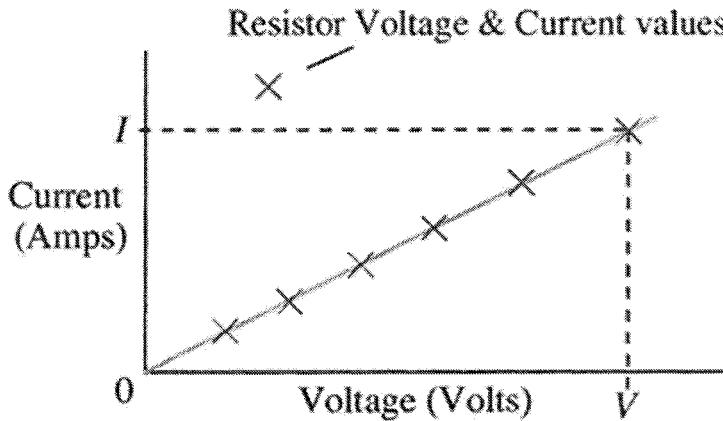
Voltage (V in volts) equals current (I in amperes) times resistance (R in ohms):

$$V = RI$$

There are three occasions when Ohm's Law is needed:

Voltage is Unknown, Current and Resistance are Known: This is the least common situation. This happens only when there are current sources and known resistors. This may happen in battery charging circuits where a resistor is placed in series with a fixed source of charging current. In FET circuits, where one resistance is known and the current conducted by the load is known, the voltage drop across the FET may be calculated. The main reason why this is the least common use of Ohm's Law is that voltages are so easy to measure in a powered circuit. Nothing needs to be disassembled or un-powered to make the measurement, unlike current or resistance.

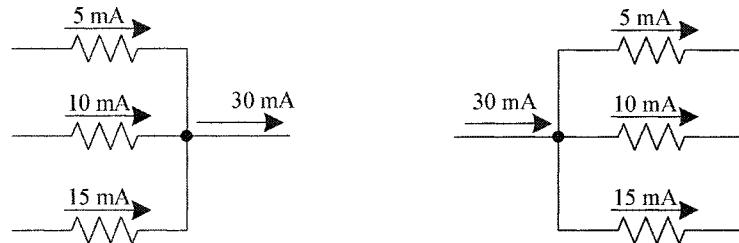
Current is Unknown, Voltage and Resistance are Known: This happens every time voltage is measured across a known resistor. The current may be calculated immediately. The circuit need not be disassembled to insert an ammeter. In reality, the ammeter in the kit multimeters is not essential and using it improperly only blows fuses. This is also useful in circuit layout to calculate the total current draw of a circuit to factor into power needs and battery life.



Resistance is Unknown, Voltage and Current are Known: This happens for most resistors during the design of a circuit. Outside constraints, some design operating point, or other element fix voltages and currents and resistance values must be calculated. Similarly, if the voltage and current over an element are known, then its operating resistance can be calculated. For items that are known to be non-linear, such items are not of much use.

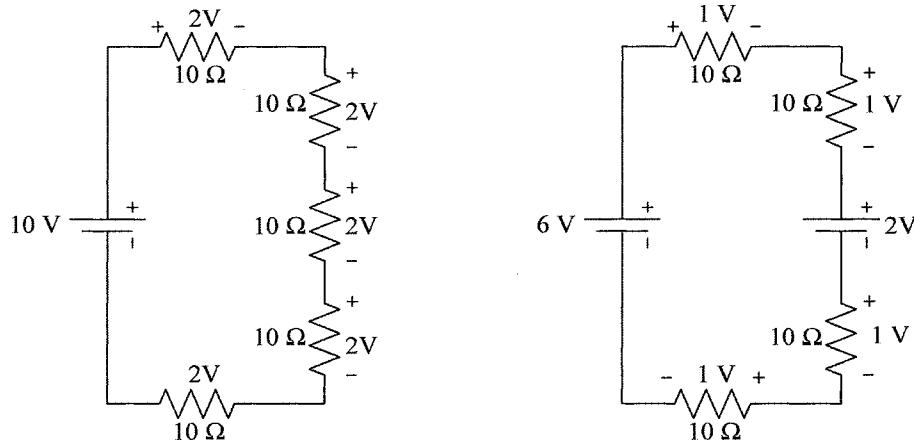
5.2.1.2 Kirchoff's Current Law

The algebraic sum of currents entering and leaving any point in a circuit must equal zero. In other words, no matter how many paths into and out of a single point all the current leaving that point must equal the current arriving at that point.



5.2.1.3 Kirchoff's Voltage Law

The algebraic sum of the voltages around any closed path is zero. In other words, the voltage that drops around any closed loop must equal the applied voltages.



5.2.1.4 Voltage Divider

The voltage divider rule can be derived from the circuit in figure below. From Ohm's Law and the Kirchoff's Current Law:

$$\begin{aligned}\frac{V_1}{R_1} &= \frac{V_2}{R_2} \\ \frac{V_1}{V_1 + V_2} &= \frac{R_1}{R_1 + R_2} \\ V_1 &= \frac{R_1}{R_1 + R_2} (V_1 + V_2)\end{aligned}$$

These relations become very important when designing amplifier circuits. Differentiating, holding R_2 constant, the second expression becomes:

$$\delta \left(\frac{V_1}{V_1 + V_2} \right) = \frac{R_1 R_2}{(R_1 + R_2)^2} \left(\frac{\delta R_1}{R_1} \right)$$

This says for a certain fractional variation in R_1 , we get a certain fractional variation in V_1 as a part of the total voltage drop. Differentiating the coefficient relating the two again and setting it to zero, we find the maximum variation in voltage for resistance comes when:

$$\frac{R_2^2 - R_1 R_2}{(R_1 + R_2)^3} = 0$$

so either $R_2 = R_1$ or $R_2 = 0$. If $R_2 = 0$ and we assume $V_1 + V_2$ is a constant, then V_1 is a constant, all derivatives are zero and not maximal. Differentiating again and taking $R_2 = R_1$ verifies this is a maximum. Therefore maximal fractional variation in divider midpoint voltage can be achieved for a given fractional change in R_1 if $R_2 = R_1$ and $V_2 = V_1$.

This is very important for sensors. If a sensor varies its resistance as a function of input, then putting it in series with a resistor such that the voltage is divided in half between them will maximize the signal drawn from the midpoint of the divider. This makes best use of a sensor in a divider type circuit.

5.2.1.5 Voltage to Current Converter

When driving the input (base) of a transistor gain stage: you must convert the input voltage to a current by using a voltage to current converter – a resistor.

5.2.1.6 Current to Voltage Converter

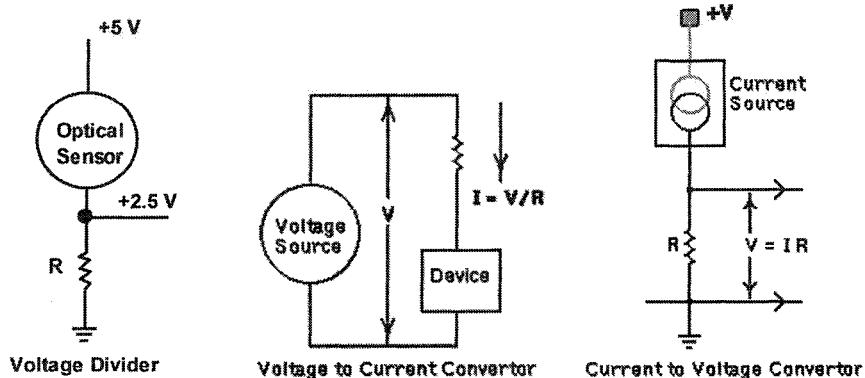
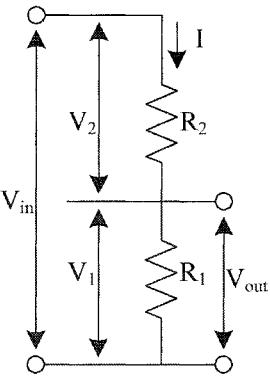
When deriving a voltage from the collector of a transistor amplifier stage: you convert the output (collector) current into a voltage by using a Current to Voltage converter in the collector circuit – a resistor.

5.2.1.7 Power and Energy

In resistors, we change the form of energy from electrical energy to heat energy. Power (P in watts) is the rate at which energy is converted. The first equation is a general physics equation. The second is the one you want. It's the same thing, only expressed in electrical terms.

$$P = VI = RI^2 = \frac{V^2}{R}$$

The **power rating** of a resistor is the maximum power that it can take before overheating. If the power rating is exceeded, the resistance may change and it may become an open circuit, or a short circuit. Usually, the power rating for small resistors is $\frac{1}{4}$ or $\frac{1}{2}$ watts. For larger resistors, 1W, 2W, or even 10W is available.



EXAMPLE:

To power a 5V circuit using a 12V supply, a three-terminal voltage regulator is usually used. However, if you try to drop the voltage from 12V to 5V using only a resistor, then you need to calculate the power rating of the resistor as well as the resistance value. At this time, the current consumed by the 5V circuit needs to be known. Here are a few ways to find out how much current the circuit demands.

- Assemble the circuit and measure the actual current used with a multi-meter.
- Check the component's current used against a standard table.

Assume the current consumed is 100 mA (milliamps) in this example. Since, 7V must be dropped with the resistor. The resistance value of the resistor becomes $7V / 0.1A = 70 \Omega$. The consumption of electric power for this resistor becomes $0.1A \times 0.1A \times 70\Omega = 0.7W$. Generally, it is safe to choose a resistor which has a power rating of about twice the power consumption needed.

5.2.1.8 Standard Values

Resistors are manufactured only in certain values of resistance; these values are known as standard values. By combining different standard value resistors, either in series, or parallel, or a combination of the two, nearly any other resistance value required can be obtained. The manufacturing process for resistors also leads to resistors having a certain tolerance. The tolerance of a resistor is simply the percentage range that the resistance of the resistor may differ from the value that is color coded on to the resistor. For instance, a 100 ohm resistor with a 5 % tolerance may have an actual resistance of anywhere from 95 to 105 ohms.

Resistor categorized based on the three major parameters:

- Resistance in Ohms (Ω)
- Heat Dissipation in Watts (W)
- Manufacturing tolerance (%)

The resistance value and tolerance are colour or print coded on each resistor according to the following:

Print and Colour Codes:

- R = ohms
- K = thousands of ohms
- M = millions of ohms

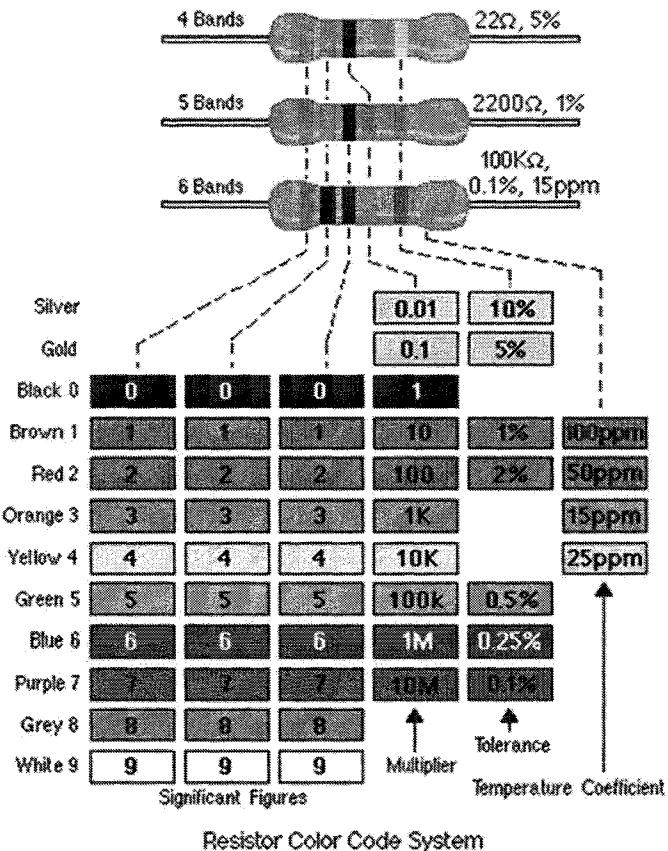
The position of the letter determines the decimal point. Tolerances are shown by using postfixes:

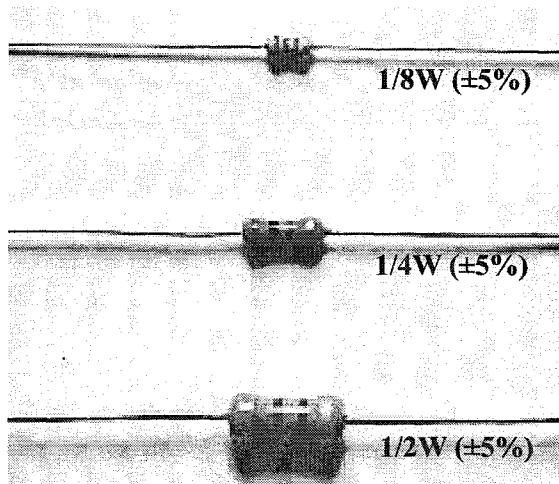
- J = $\pm 5\%$
- K = $\pm 10\%$
- M = $\pm 20\%$

So:

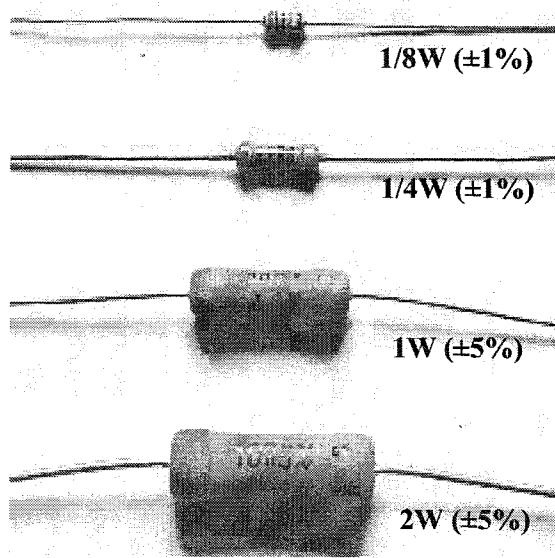
$$56K0K \equiv 56000 \pm 10\%$$

$$6M8J \equiv 6800000 \pm 5\%$$





Carbon Film Resistors



Metal Film Resistors

5.2.1.9 Resistor Types

5.2.1.9.1 Fixed Resistors

A fixed resistor is one in which the value of its resistance cannot change.

Carbon Film Resistors

This is the most general purpose, cheap resistor. Usually the tolerance of the resistance value is $\pm 5\%$. Power ratings of 1/8W, 1/4W and 1/2W are frequently used. Carbon film resistors have a disadvantage; they tend to be electrically noisy. Metal film resistors are recommended for use in analog circuits. The physical sizes of different resistors are shown in figure below.

Metal Film Resistors

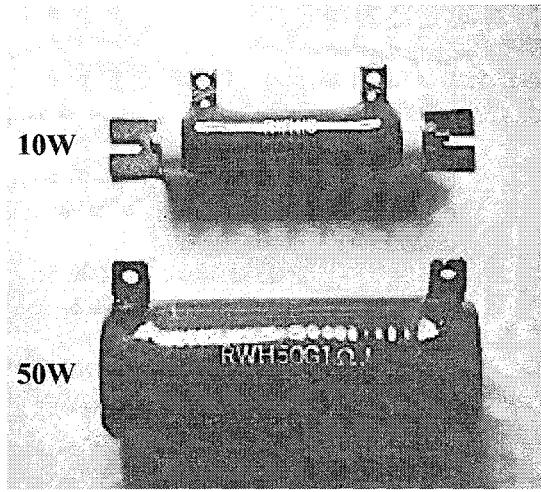
Metal film resistors are used when a higher tolerance (more accurate value) is needed. They have about $\pm 0.05\%$ tolerance. In most practical circuits, resistors that are about $\pm 1\%$ are sufficient. The metal film resistor is used for bridge circuits, filter circuits, and low-noise analog signal circuits.

Wire Wound Resistor

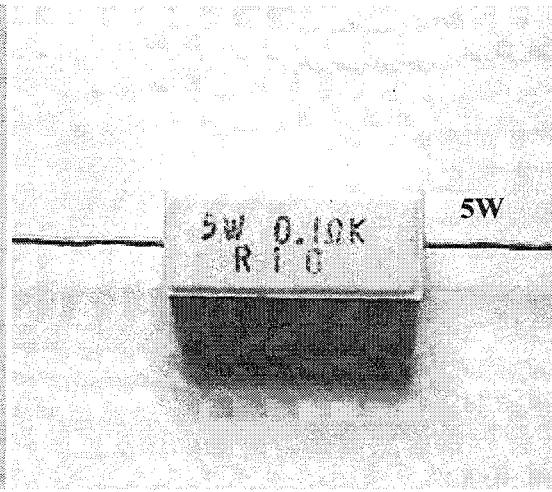
A wire wound resistor is made of metal resistance wire, and because of this, they can be manufactured to precise values. Also, high-wattage resistors can be made by using a thick wire material. They are temperature stable and can take a large power without failing (but can get very hot). Wire wound resistors cannot be used for high-frequency circuits. Since a wire wound resistor is a wire wrapped around an insulator, it behaves like a coil, hence can change the behavior of the high-frequency circuits.

Ceramic Resistor

These are wire wound resistors in a ceramic case, strengthened with special cement. They have very high power ratings, from 1 or 2 watts to dozens of watts. These resistors can become extremely hot when used for high power applications, and this must be taken into account when designing the circuit. These devices can easily get hot enough to burn you if you touch one.



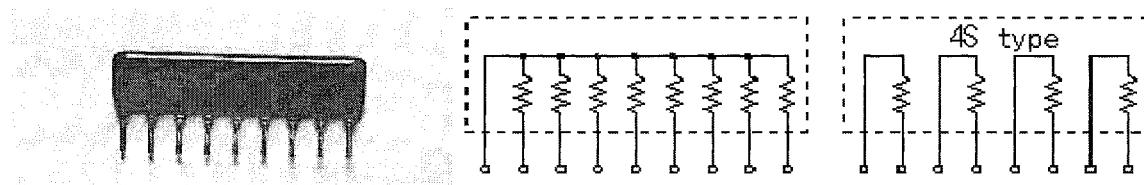
Wire Wound Resistors



Ceramic Resistor

Single-In-Line (SIL) Resistor Network

An SIL resistor network is made with many resistors of the same value, all in one package. One side of each resistor is connected with one side of all the other resistors inside. One example of its use would be to control the current in a circuit powering many light emitting diodes (LEDs). In the photograph below (left), 8 resistors are housed in the package. Each of the leads on the package is one resistor. The ninth lead on the left side is the common lead. Some resistor networks are marked as "4S" printed on the top of the chip. The 4S indicates that the package contains 4 independent resistors that are not wired together inside. The housing has eight leads instead of nine. The internal wiring of these typical resistor networks has been illustrated in figure below (right).

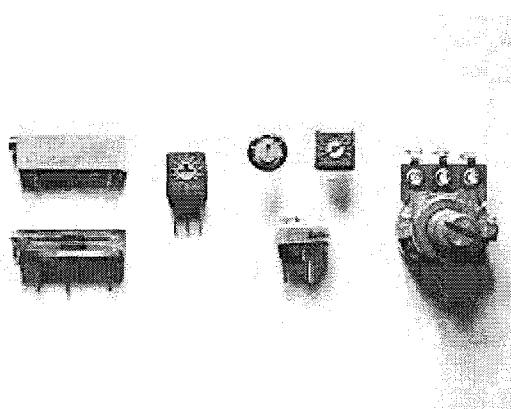
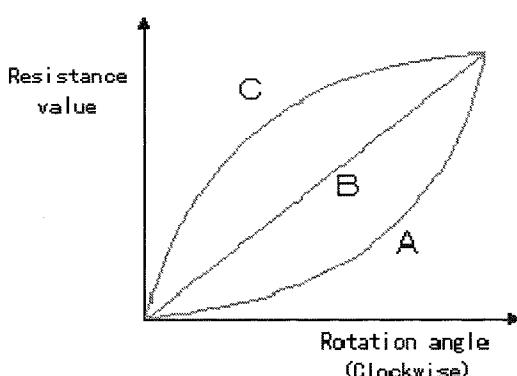


5.2.1.9.2 Variable Resistors

There are two general ways in which variable resistors are used. One is the variable resistor whose value is easily changed, like the volume adjustment of Radio. The other is semi-fixed resistor that is not meant to be adjusted by anyone but a technician. It is used to adjust the operating condition of the circuit by the technician. Semi-fixed resistors are used to compensate for the inaccuracies of the resistors, and to fine-tune a circuit. The rotation angle of the variable resistor is usually about 300 degrees. Some variable resistors must be turned many times to use the whole range of resistance they offer. This allows for very precise adjustments of their value. These are called **Potentiometers** or **Trimmer Potentiometers**.

In figure below (right), the variable resistors typically used for volume controls can be seen on the far right. Its value is very easy to adjust. The four resistors at the center of the photograph are the semi-fixed type. These ones are mounted on the printed circuit board. The two resistors on the left are the trimmer potentiometers.

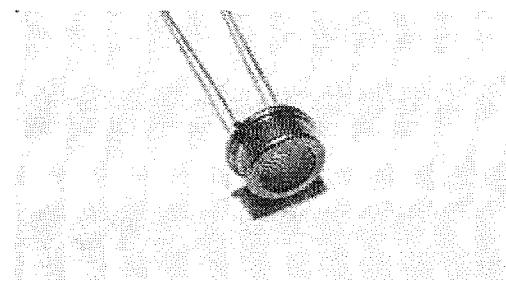
There are three ways in which the value of a variable resistor can change according to the rotation angle of its axis, as shown in figure below (left). When type "A" rotates clockwise, at first, the resistance value changes slowly and then in the second half of its axis, it changes very quickly. Type "A" is typically used for the volume control of a radio, for example. It is well suited to adjust a low sound subtly, and hence suits the characteristics of the ear. Type "A" variable



resistors are sometimes called **audio taper** potentiometers. In type "B", the rotation of the axis and the change of the resistance value are directly related. The rate of change is linear, throughout the sweep of the axis. This type suits a resistance value adjustment in a circuit, a balance circuit and so on. They are sometimes called **linear taper** potentiometers. Type "C" changes exactly the opposite way to type "A", and is only used in particular applications.

5.2.1.9.3 CdS Devices

Some components can change resistance value by changes in the amount of light hitting them. One type is the Cadmium Sulfide Photocell (CdS). The more light that hits this device, the smaller its resistance value becomes. CdS sensors vary according to light sensitivity, size, resistance value, etc. Figure on the right features a typical CdS photocell. Its diameter is 8 mm, 4 mm high, with a cylinder form. When bright light is hitting it, the value is about 200 ohms, and when in the dark, the resistance value is about 2M ohms.

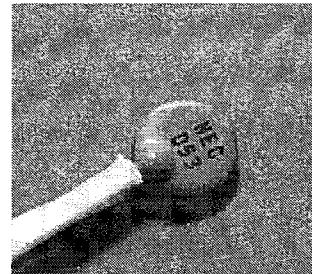


5.2.1.9.4 Thermistors

The resistance value of a thermistor changes according to temperature. There are mainly three types of thermistors.

➤ **NTC (Negative Temperature Coefficient Thermistor):** With this type, the resistance value decreases continuously as the temperature rises. This type is used for temperature control. The relation between the temperature and the resistance value of the NTC type can be calculated using the following formula

$$R = R_0 \exp\left(B\left(\frac{1}{T} - \frac{1}{T_0}\right)\right)$$



where T is the temperature ($^{\circ}\text{K}$), T_0 is the reference temperature ($298 \text{ }^{\circ}\text{K}$), R_0 is the resistance at the reference temperature, and B is the resistance coefficient.

- **PTC (Positive Temperature Coefficient Thermistor):** In this type, the resistance value increases suddenly when the temperature rises above a specific point.
- **CTR (Critical Temperature Resister Thermistor):** In this type, the resistance value decreases suddenly when the temperature rises above a specific point.

5.2.2 Capacitor

Capacitors store charge and hence voltage on them. For any capacitor that students are likely to purchase, its operation is linear, given by the following relation: charge (Q in Coulombs) equals voltage (V in Volts) times capacitance(C in Farads):

$$Q = VC$$

or similarly, for current:

$$I = C \frac{dV}{dt}$$

The higher the current, the faster the capacitor voltage rises. An interesting side effect is, if the voltage is reduced very quickly - by shorting the capacitor - very large currents can be produced. In practical terms, very little charge is stored in the average capacitor. For example, a $1\mu\text{F}$ capacitor charged to 20V only contains 20 micro Coulombs. If that charge were shorted using a 0.001 Ohm connection, a short, thick wire, the instantaneous current would be 20,000 Amps and would decline with voltage but would last only nanoseconds. The reason is, the amount of energy stored:

$$E = \frac{1}{2} CV^2$$

is very small, only 0.0002 Joules. A $100\mu\text{F}$ capacitor in a high voltage device, charged to 1000V contains 50J of energy. If shorted through a human body from one hand to another, that is 50J straight to the heart - more than enough to be lethal - and it is over in less than a second. This is the primary reason why students are prohibited from working with high voltage.

If a capacitor is being charged and discharged in a circuit by an oscillating wave form, and the wave is of size ΔV with frequency f , the current consumed by the circuit due to charging and discharging the capacitor is at least: $I = f \Delta V_{p-p} C$ (depending on inefficiencies in the circuit). Note this is peak-to-peak voltage so 120V RMS AC wall current would be 340V peak to peak. For a 1 MHz signal at 5 volts - a computer clock - and a $1\mu\text{F}$ capacitor, the current is at least 5 Amps! Keep capacitors in oscillating circuits reasonably small - in the nF range.

5.2.2.1 Specifications

Capacitors have two main specifications. The first is the capacitance. This, of course, needs to match whatever value is required by your circuit, within the appropriate tolerances. The second rating is the maximum voltage or break down voltage that you can apply to the capacitor. The breakdown voltage is the voltage that when exceeds will cause the dielectric (insulator) inside the capacitor to break down and conduct. When this happens, the failure can be catastrophic. Do not exceed this voltage. As with all maximum specs, you don't generally want to be pushing these devices up near their absolute max ratings, either. If your circuit may have up to ten volts across a particular capacitor, don't get one rated for twelve volts. Get one rated for twenty four volts. (Don't get one rated for 2000 volts, either - it probably won't work correctly in your circuit.)

In the majority of applications, the capacitance of a capacitor is generally very small, thus units such as the microFarad (10^{-6}F), nanoFarad (10^{-9}F), and picoFarad (10^{-12}F) are used. There are two ways in which the capacitance is marked on a capacitor. One uses letters and numbers, the other uses only numbers. In either case, there are only three characters used. For example, [10n] and [103] denote the same value of capacitance, which 10 nanoFarads. In the case that the value is displayed with the three-digit code, the 1st and 2nd digits from the left show the 1st figure and the 2nd figure, and the 3rd digit is a multiplier which determines how many zeros are to be added to the capacitance. Note that picofarad (pF) units are written this way. For example, when the code is [103], it indicates 10×10^3 , or $10,000\text{ pF} = 10\text{ nanofarad (nF)} = 0.01\text{ microfarad (\mu F)}$. If the code happened to be [224], it would be 22×10^4 = or $220,000\text{ pF} = 220\text{nF} = 0.22\mu\text{F}$. Values under 100pF are displayed with 2 digits only. For example, 47 would be 47pF.

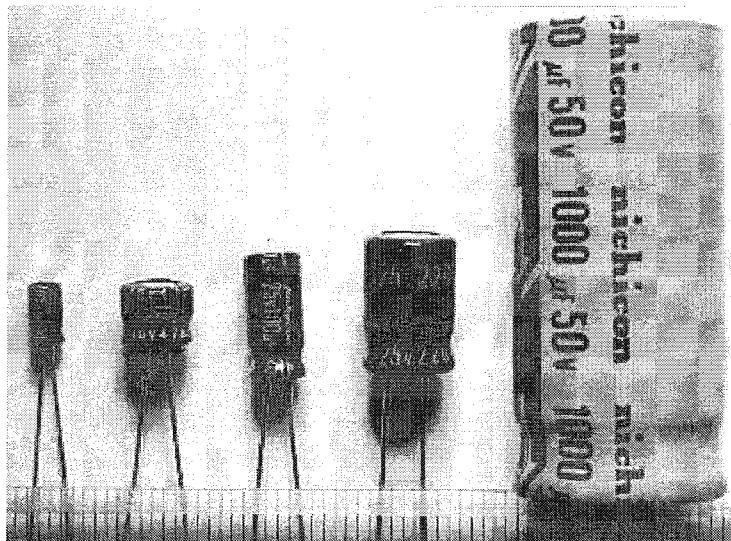
5.2.2.2 Capacitor Types

5.2.2.2.1 Polarized Capacitors

This type of capacitors has polarity. They have a positive and a negative electrode. This means that it is very important which way round they are connected. If the capacitor is subjected to voltage exceeding its working voltage, or if it is connected with incorrect polarity, it may burst. It is extremely dangerous, because it can literally explode. Make absolutely no mistakes. Generally, in the circuit diagram and on the capacitor, the positive side is indicated by a "+" (plus) symbol.

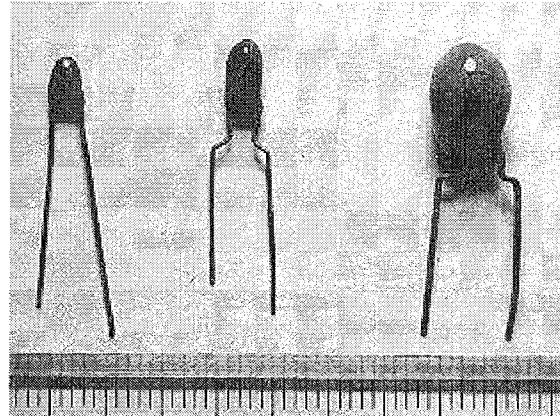
Electrolytic Capacitors

Aluminum is used for the electrodes by using a thin oxidization membrane. Large values of capacitance can be obtained in comparison with the size of the capacitor, because the dielectric used is very thin. They range in value from about $1\mu\text{F}$ to thousands of μF with maximum voltage of 6 to 300V. Mainly this type of capacitor is used as a ripple filter in a power supply circuit, or as a filter to bypass low frequency signals, etc. Because this type of capacitor is comparatively similar to the nature of a coil in construction, it is not suitable to use for high-frequency circuits. (It is said that the frequency characteristic is bad.)



Tantalum Capacitors

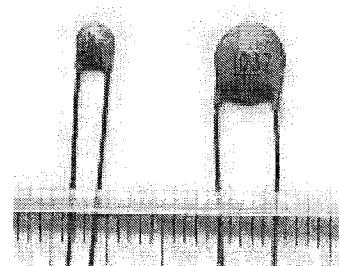
Tantalum Capacitors are electrolytic capacitors that use tantalum for the electrodes. Large values of capacitance similar to aluminum electrolytic capacitors can be obtained. Also, tantalum capacitors are superior to aluminum electrolytic capacitors in temperature and frequency characteristics. Tantalum capacitors are more expensive than aluminum electrolytic capacitors. They come in μF ranges and 6-50V. Capacitance can change with temperature as well as frequency, and these types are very stable. Therefore, tantalum capacitors are used for circuits that demand high stability in the capacitance values. Also, it is said to be common sense to use tantalum capacitors for analog signal systems, because the current-spike noise that occurs with aluminum electrolytic capacitors does not appear.



5.2.2.2.2 Unpolarized Capacitors

Ceramic Capacitors

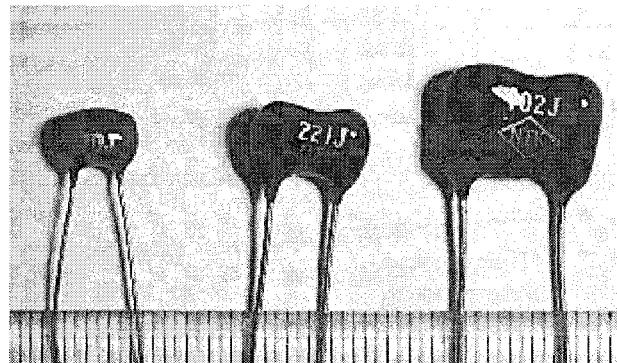
Ceramic capacitors are constructed with materials such as titanium and acid barium used as the dielectric. Internally, these capacitors are not constructed as a coil, so they can be used in high frequency applications. Typically, they are used in circuits which bypass high frequency signals to ground. These capacitors have the shape of a disk. Their capacitance is comparatively small, but their breakdown voltage can be very



high, say 1kV. Some ceramic capacitors are multi-layer with many-layered dielectric. These capacitors are small in size, and have good temperature and frequency characteristics. In general, Ceramic capacitors should not be used for analog circuits, because they can distort the signal. In this course, ceramic disks are best used only in MHz clock oscillator circuits, or on motors to absorb high frequency, high voltage spikes.

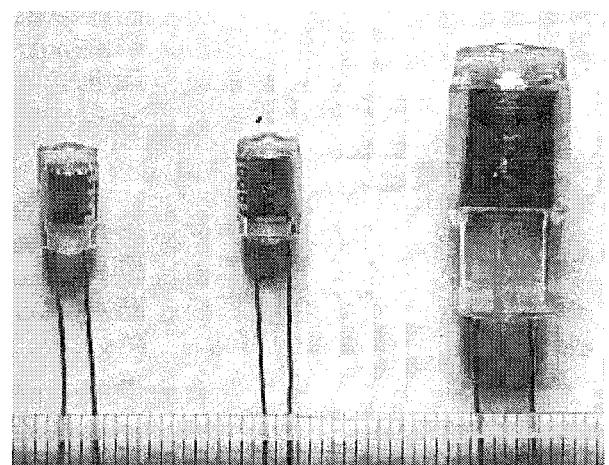
Mica Capacitors

These capacitors use Mica for the dielectric. Mica capacitors have good stability because their temperature coefficient is small. Because their frequency characteristic is excellent, they are used for resonance circuits, and high frequency filters. Also, they have good insulation, and so can be utilized in high voltage circuits. Mica capacitors do not have high values of capacitance, and they can be relatively expensive.



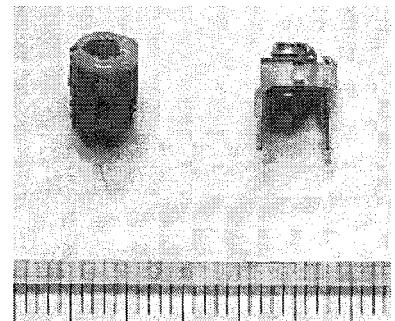
Polystyrene Film Capacitors

Polystyrene film is used as the dielectric. This type of capacitor is not for use in high frequency circuits, because they are constructed like a coil inside. They are used well in filter circuits or timing circuits which run at several hundred KHz or less. They are noted for their very low leakage. Their only application that justifies their expense in the course is sample and hold circuits required to maintain a voltage for a while. The component shown on the right has a red color due to the copper leaf used for the electrode. The silver color is due to the use of aluminum foil as the electrode.



5.2.2.2.3 Variable Capacitors

Variable capacitors are used for adjustment of frequency mainly. On the left in the figure is a "trimmer," which uses ceramic as the dielectric (3-27 pF). Next to it on the right is one that uses polyester film for the dielectric (5-40 pF). When adjusting the value of a variable capacitor, it is advisable to be careful. One of the component's leads is connected to the adjustment screw of the capacitor. This means that the value of the capacitor can be affected by the capacitance of the screwdriver in your hand.



5.2.3 Inductor

Inductors or coils generate currents by maintaining magnetic fields in their core. When an inductor is out in a circuit and voltage is applied, the current rises according to the formula:

$$V = L \frac{dI}{dt}$$

where V is the voltage across the inductor and L is the inductance measured in Henry (H). Like the Farad, in electronics circuits, the Henry is a large unit. Typically, palm-sized elements are measured in tens of milliHenrys. When V is positive, the current through the inductor increases. For an ideal inductor connected directly to a voltage supply, the current will constantly increase without bound. While even superconducting inductors cannot carry infinite currents, currents can become very large. For practical inductors made from copper wire, the inductor dies have some internal resistance so the equation is modified to be:

$$V = L \frac{dI}{dt} + RI$$

Thus, for practical inductors, a maximum current is reached, dictated by the internal resistance and applied voltage.

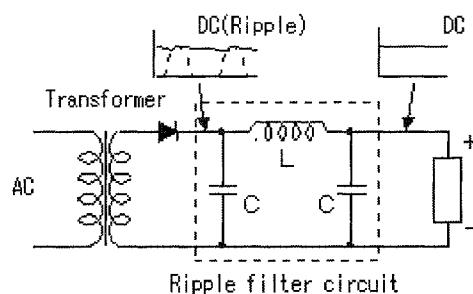
The inductance will tend to smooth sudden changes in current just as the capacitance smoothes sudden changes in voltage. Of course, if the current is constant there will be no inductance effect. So, unlike the capacitor which behaves like an open-circuit in DC circuits, an inductor behaves like a short-circuit in DC circuits.

Students could rightfully say, "Who buys an inductor these days?" With the revolution in microelectronics, a filter block is a tiny fleck of silicon many times smaller and lighter than any circuit using inductors. Why would AER 201Y ever go back to the dark ages of wires wound around iron cores? The answer is simple, electromagnetic devices, such as motors and solenoids, are still the easiest way of changing electrical signals into forces. It is a reality that any circuit that actuates something is probably going to include an inductor.

5.2.3.1 Characteristic of Inductors

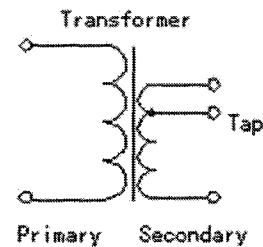
5.2.3.1.1 Current Stabilization Characteristic

When current begins to flow in the inductor, it resists the flow. When current decreases, the coil makes current continue to flow at the previous rate. This characteristic is used for the ripple filter circuit of a power supply where it transforms alternating current (AC) to direct current (DC). When a rectifier is used to make DC from AC, the output of the rectifier without a ripple filter circuit is ripple current. Ripple current is DC that has a large AC component. A ripple filter circuit often combines coil and capacitors. The coil resists the change of current and capacitors supplement the flow of current by discharging into the circuit if the input voltage drops. Thus, clear, ripple-free DC is obtained from the ripple filter circuit. Resistor is used instead of coil in simple ripple filter circuits.



5.2.3.1.2 Mutual induction

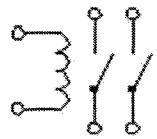
The electric power can be transferred between two coils by mutual induction. The transformer utilizes this characteristic. The input coil that gives the electric power is called the primary side, while the output coil that takes out the electric power is called



the secondary side. The output voltage is determined by the ratio of turns of wire between the primary coil and secondary coil. Some transformers have a tap (or several) on the secondary coil to provide multiple voltage levels.

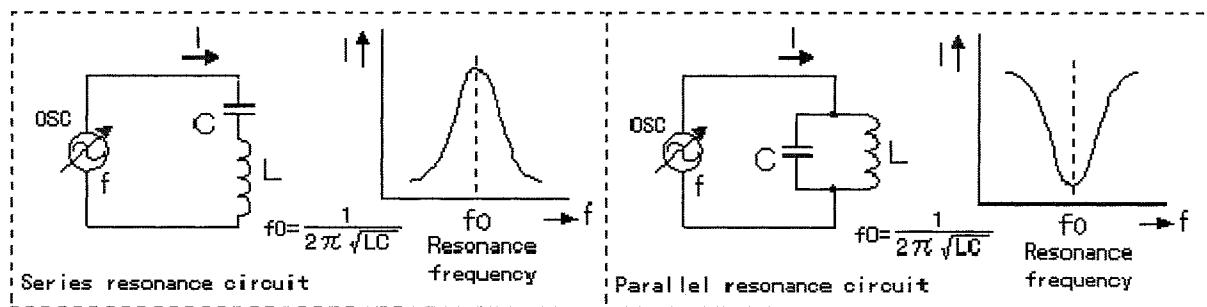
5.2.3.1.3 Electromagnet

When current flows through a conductor, a magnetic field is created. This field is much stronger in a coil. An electromagnet is just like a regular magnet. It attracts iron, nickel, and some other metals. Relays utilize this characteristic. When the current flows to the coil of relay, the magnetic field attracts a steel plate, and the switch that is attached to the steel plate goes ON.



5.2.3.1.4 Resonance

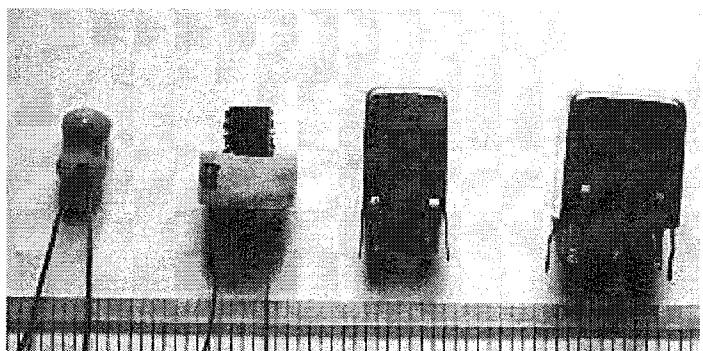
When a coil and a capacitor are combined, the resulting circuit has special characteristics. The impedance (resistance to current flow) of the circuit changes with the frequency of the voltage. Current will flow easily at a given frequency, but has difficulty flowing at another frequency. The tuning circuit that selects a particular radio station utilizes this characteristic.



5.2.3.2 High Frequency Coils

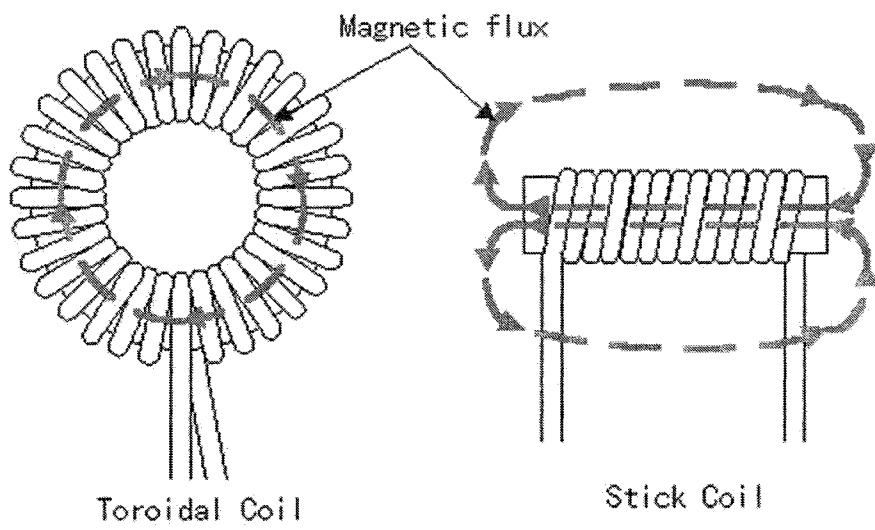
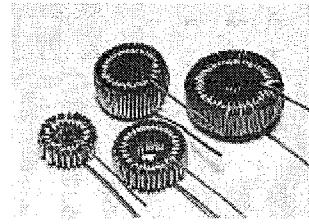
The photograph shows an example of a small coil component. The component on the left is wound with thin copper wire to a small barbell-shaped ferrite core, and has a value of $100\mu\text{H}$. It is used for high frequency resonance, or for deterring of high frequency. The value of the small coil like this is indicated with a color code, just like a resistor. The strength of this type of coil varies from $1\mu\text{H}$ to several hundred μH .

The second coil from the left has thin copper wire wound around a stick-shaped ferrite core. It is used the same as the component above. The value is $470\mu\text{H}$. The two devices on the right in the figure are high frequency transformers. They are used for intermediate frequency (455KHz) tuning of transistor radios, or for oscillator circuits. To shield the coils from magnetic flux, and to prevent the coils from interfering with other circuits, the high frequency coils are housed in a metal case called shield case. This case must be connected to ground. As for tuning or oscillation, this type of transformer can change its value of inductance.



5.2.3.3 The Toroidal Coil

The toroidal coil consists of copper wire wrapped around a cylindrical core. It is possible to make it so that the magnetic flux which occurs within the coil doesn't leak out, the coil efficiency is good, and that the magnetic flux has little influence on other components.



5.2.4 Diode

A diode is a semiconductor device that allows current to flow through it in only one direction. Hence, when negative voltages are applied, no current is conducted. At positive voltages, the current conducted is exponentially related to the voltage. The polarity of applied voltage that causes charge to flow through the diode is called *Forward Bias*. The polarity of applied voltage that cannot produce any current is called *Reverse Bias*.

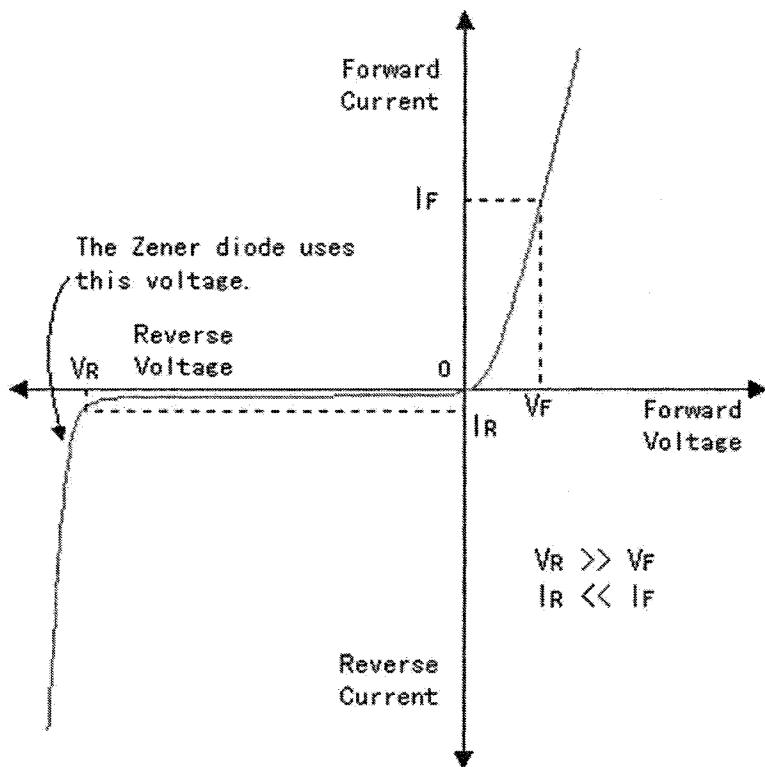
Figure below shows the electrical characteristics of a typical diode. Diodes are referred to as non-linear circuit elements because of the below characteristic curve. When a small voltage is applied to the diode in the forward direction, current flows easily. Because the diode has a certain amount of resistance, the voltage will drop slightly as current flows through the diode. A typical diode causes a voltage drop of about 0.6-1V (V_F) (In the case of silicon diode, almost 0.6V). Practically speaking, silicon diodes will conduct about 1mA at 0.7V and ten times as much current for every 0.1V more. A typical characteristic is given here:

V	0.5V	0.6V	0.7V	0.8V	0.9V	1.0V
I	.01mA	.1mA	1mA	10mA	100mA	1A

In nearly all cases of electronic circuits at low currents, it is safe to assume the diode will require a 0.7V forward voltage. In very low power circuitry where the supply voltages are less than 10V and the resistances are in the hundreds of $k\Omega$, a safer assumption may be 0.6V. In power circuitry where several Amps may be conducted, a 1V forward voltage drop might be a better assumption.

Since diode voltage does not change much with current, their power dissipation once forward biased depends almost entirely on the current conducted. Diodes are therefore rated in Amps. Their current rating is dependent on the heat dissipation capability of the package. For diodes or bridge rectifiers with a heat sink surface, a heat sink must be used or the current rating will be reduced.

The voltage drop of a diode needs to be taken into consideration in a circuit that uses many diodes in series. Also, the amount of current passing through the diodes must be considered. When voltage is applied in the reverse direction



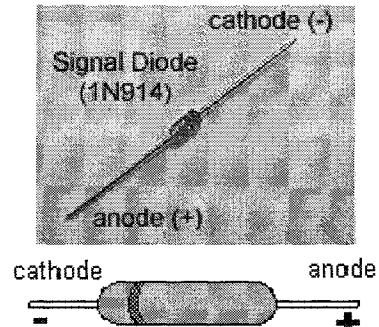
through a diode, the diode will have a great resistance to current flow. Different diodes have different characteristics when reverse-biased. A given diode should be selected depending on how it will be used in the circuit. The current that will flow through a diode biased in the reverse direction will vary from several mA to only μ A, which is very small. The limiting voltages and currents permissible must be considered on a case-by-case basis. For example, when using diodes for rectification, part of the time they will be required to withstand a reverse voltage. If the diodes are not chosen carefully, they will break down.

5.2.4.1 Types and Applications of Diodes

Although all diodes operate with the same general principle, there are different types suited to different applications. For example, the following devices are best used for the applications noted.

Signal Diodes

Signal diodes are physically small devices usually used where small currents, high voltages and high frequencies are involved. The name comes from the fact that these diodes are suitable for use in radio detectors to isolate the radio "signal". Signal diodes are very small and often glass encapsulated, with a red or black band on one end. (The glass is sometimes painted over to reduce unwanted photovoltaic effects.) The 1N914 is a good signal diode.



Power Diodes

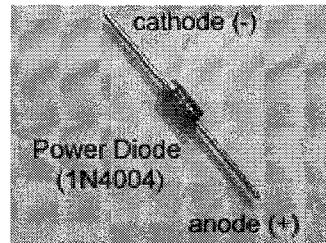
Where larger currents are involved, a larger junction is needed to dissipate the heat generated. A small junction would be in danger of literally melting with currents in excess of a few hundred milli-amps. Since the power diode has a large junction, it is not suited to high frequency applications. (High frequency, high current diodes are available, but the cost is substantial.)

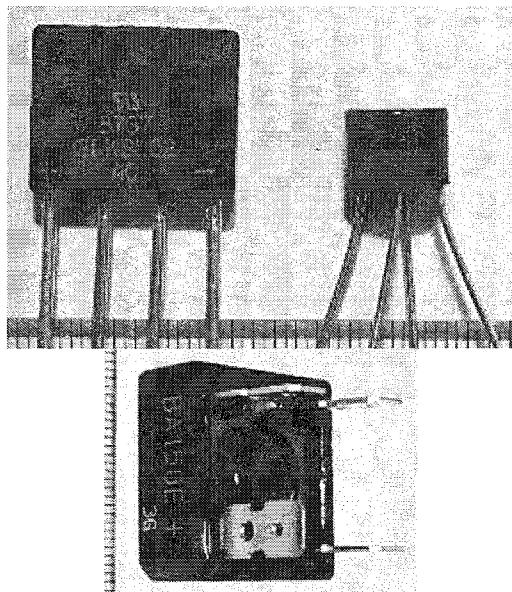
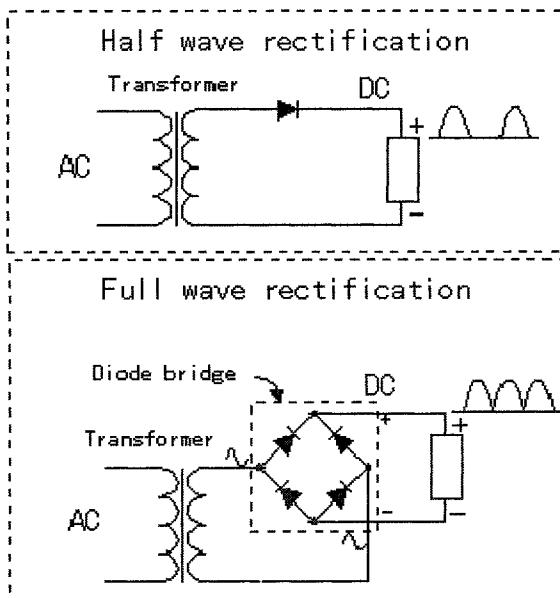
One advantage of the larger junction is its ability to withstand higher voltages without sustaining damage. While a signal diode may only be able to take 30 to 50 volts reverse potential, it is quite common to find power diodes rated up to several thousand volts maximum reverse bias (termed "Peak Inverse Voltage", or PIV.)

Power diodes are able to pass large loads varying from the 1N400X series rated at 1 amp up to industrial diodes capable of carrying hundreds of amps! These diodes come in a variety of encapsulations, the most common being a black cylinder of plastic about 3mm long with a white band indicating the cathode (negative) end. Large-current devices are often encased in metal to provide efficient heat transfer.

Diode Bridge

Power diodes are used to rectify the AC and make DC. It is possible to do only "half wave rectification" using one diode. When four diodes are combined, "full wave rectification" occurs. Devices that combine 4 diodes in one package are called diode bridges. They are used for full-wave rectification. The physical construction of bridge depends on the current demand. All have four leads - two for AC input and one each for positive and negative output. The bridge converts an alternating input current into a fixed polarity DC output.





The photograph on the top right shows two examples of diode bridges. The cylindrical device on the right in the photograph has a current limit of 1A. Physically, it is 7 mm high, and 10 mm in diameter. The flat device on the left has a current limit of 4A. It has a thickness of 6 mm, is 16 mm in height, and 19 mm in width. The photograph on the down right shows a large, high-power diode bridge. It has a current capacity of 15A. The peak reverse-bias voltage is 400V. Diode bridges with large current capacities like this one require a heat sink. Typically, they are screwed to a piece of metal, or the chassis of device in which they are used. The heat sink allows the device to radiate the excess heat. As for size, this one is 26 mm wide on each side, and the height of the module part is 10 mm.

Zener Diode (\rightarrow)

From the previous characteristic graph, all diodes will in fact break down if sufficient negative potentials are applied. Signal diodes such as the 1N914 (the small pink-glass ones) will start allowing reverse current, catastrophically, at about $-75V$. Regular rectifying diodes such as the 1N4001 through 1N4007 break down at $-50V$ to $-1000V$, respectively. Zener diodes are designed to breakdown at specific (low) voltages and not suffer any damage, so long as they do not over-heat. Thus, Zener diodes work the same way as regular diodes for most of their operation. They have a similar voltage and current curve when operated in the forward direction. The difference comes in the reverse direction. Here, they still act like regular diodes as long as the reverse breakdown voltage is not exceeded. However, once you reach the reverse breakdown voltage, the Zener diode starts to conduct electricity, and will continue to conduct electricity while maintaining almost the same reverse voltage across its terminals. **WARNING:** This means that if not used properly, it is very easy to burn up a zener diode. You must have some external component to limit the amount of current that flows when the breakdown voltage is exceeded. (A 10V Zener conducting 0.5A is a 5W power dissipation. If the Zener package only dissipates a maximum of 0.5W as the 1N5240B does, the Zener will heat rapidly and burn out.)

Why would one be interested in having a diode that breaks down in the reverse direction? One can use a Zener diode to limit the voltage at a certain point in a circuit, thus protecting an input. Electric circuits are designed with specific voltage ranges at various parts of the circuit. Certain circuit elements, notably inductors, are prone to generating large voltage spikes - noise - sufficient to push certain currents through the circuit, similar to floods and dams. What is needed is a "spillway" for the current once the voltage is pushed outside the normal operating range, but before it reaches the point where damage is done to the circuit or limiting the magnitude of noise. Here, the Zener diode is used to provide this path. The Zener begins to conduct at a certain voltage and provides the alternate path, saving the circuit or clipping the noise. Because Zeners behave as normal diodes in the forward direction, conducting at $0.7V$, two Zeners are normally placed in series, facing opposite directions so that the Zener voltage of the series is equal in the forward and reverse directions.

Furthermore, since in Zener diodes the breakdown occurs in a very controlled way, you can also use Zeners to make a known voltage source, which is constant over a wide range of power supply voltages. This has applications for circuits that need some sort of reference to make measurements by, and also to make regulator circuits for power supplies.

It must be remembered that Zeners are used in **REVERSE** mode, i.e. the anode connects to the negative supply.

Light Emitting Diodes (\rightarrow)

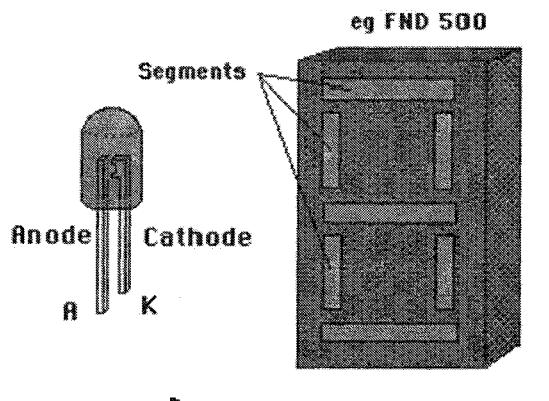
Light emitting diodes (LEDs) are among the most widely used of all types of diodes. The available colours vary from red, orange, yellow, green and blue. Sizes are 3mm, or 5mm. It is also possible to purchase "rectangular" LEDs and special-purpose LEDs, for example LEDs that have been molded to represent a small dot. Most LEDs have a coloured "lens", but it is possible to buy "water-clear" LEDs that have no colouring in the plastic lens. LEDs are also available in different "intensities" ranging up to several "candle-power." The most common (and cheapest) LED is the 5mm red LED. They are also available in "packages" arranged to produce letters and numerals. The price and availability of these packages depends to a large extent upon current industrial requirements. The once common "FND500" could be obtained for less than a dollar until quite recently. At the moment it costs considerably more, if it can be found at all!

Numerals are produced by arranging LEDs in a seven-segment arrangement as indicated in the figure. Integrated circuits (ICs) are available for "driving" displays directly. (The 4026 IC for example, will take pulses, count them and display the count on a seven-segment display, all for a few dollars!)

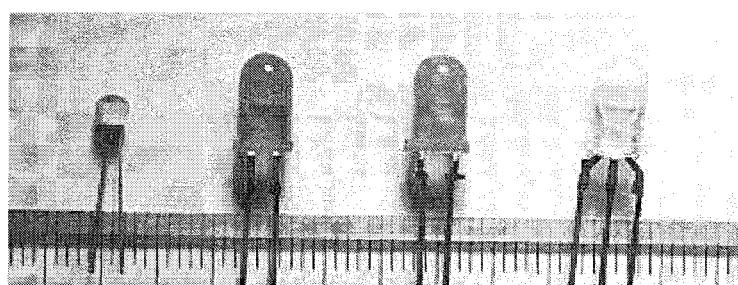
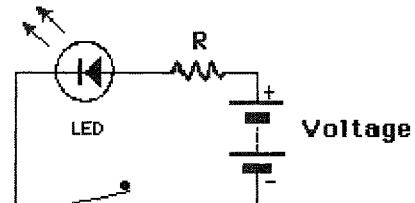
An LED may be thought of as a 1.6V globe for design purposes. If voltages of more than this are involved, a "dropping" resistor is needed, as indicated below. The maximum current flowing through an LED should be limited to around 20 milli-amps. The formula below shows how to calculate the resistance required to use LEDs with different voltages.

$$R_x = \frac{(V - 1.6)}{I}$$

where, R_x is the "dropping" resistance in ohms, V is the supply voltage in Volts, and I is the maximum current in amps ($=0.02$ A).

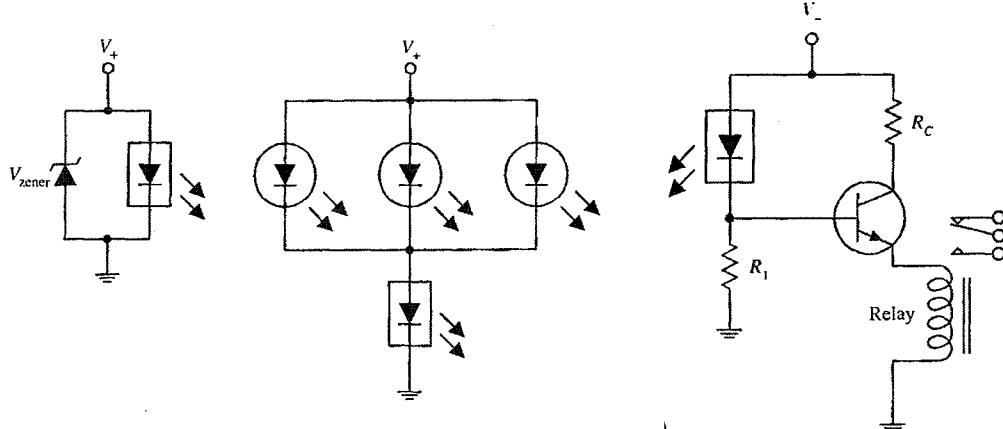
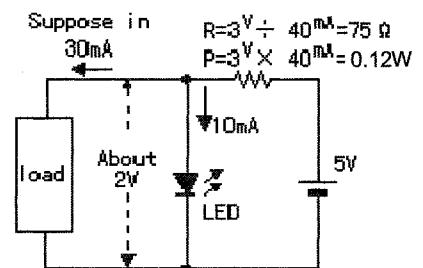


The device on the far right in figure below combines a red LED and green LED in one package. The component lead in the middle is common to both LEDs. As for the remaining two leads, one side is for the green, the other for the red LED. When both are turned on simultaneously, it becomes orange. As with all diodes, orientation of LEDs is critical. If you connect the legs in a wrong way around, it will not conduct. The figure on the right should provide a useful guide. This property is very useful when using a diode to provide protection against voltage "reversal" - also called "idiot-proofing". If a diode is built into the power section the rest of the circuit will be protected in the event of somebody connecting the power supply the wrong way around. The longer lead is the Anode side, and the short one is the Cathode side. The polarity of an LED can also be determined using a resistance meter, or even a 1.5 V battery. When using a test meter to determine polarity, set the meter to a low resistance measurement range. Connect the probes of the meter to the LED. If the polarity is correct, the LED will glow. If the LED does not glow, switch the meter probes to the opposite leads on the



LED. In either case, the side of the diode, which is connected to the black meter probe when the LED glows, is the Anode side. Positive voltage flows out of the black probe when the meter is set to measure resistance. It is possible to use an LED to obtain a fixed voltage as shown in figure on the right. The voltage drop (forward voltage, or V_F) of an LED is comparatively stable at just about 1.6-2V.

A special type of LEDs (e.g. NTE3130) contains a miniature IC within their package that causes the LED to flash from 1 to 6 times each second. They are used primarily as indicator flashers, but may also be used as simple oscillators. Flasher LEDs usually don't require a current-limiting resistor. Typically, a voltage between 3 and 7V is used to drive a flasher LED. To protect a flasher LED from excessive forward voltage, a zener diode (6V) can be placed in parallel as shown in figure below (left). A single flasher LED can be used to flash a number of ordinary LEDs, as shown in figure below (middle). Also, in figure below (right), a flasher LED is used to supply a series of ON/OFF pulses of current/voltage to the base of the transistor, thus driving the relay intermittently. R_1 sets the biasing voltage for the transistor and RC sets the collector current, as will be discussed in the next section.



Schottky barrier diode

Diodes are used to rectify alternating current into direct current. However, rectification will not occur when the frequency of the alternating current is too high. This is due to what is known as the "reverse recovery characteristic." The reverse recovery characteristic can be explained as follows: if the opposite voltage is suddenly applied to a forward-biased diode, current will continue to flow in the forward direction for a brief moment. This time until the current flow stops is called the "Reverse Recovery Time". The current is considered to be stopped when it falls to about 10% of the value of the peak reverse current. The Schottky barrier diode has a short reverse recovery time, which makes it ideally suited to use in high frequency rectification. The shottky barrier diode has the following characteristics.

- The voltage drop in the forward direction is low.
- The reverse recovery time is short.

However, it has the following disadvantages.

- The diode can have relatively high leakage current.
- The surge resistance is low.

Because the reverse recovery time is short, this diode is often used for the switching regulator in a high frequency circuit.



5.2.5 Transistor

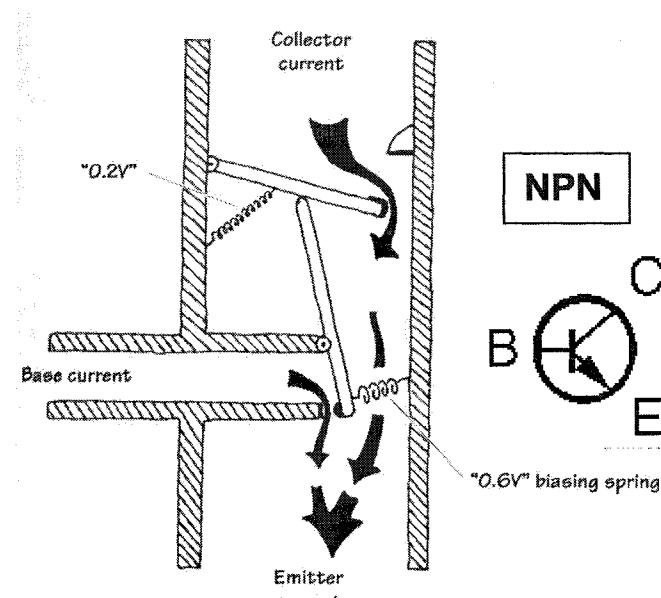
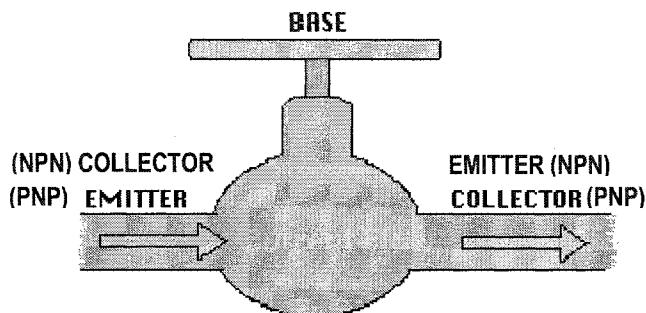
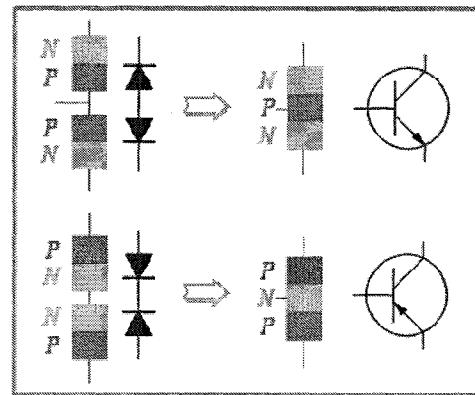
The basic configuration of a transistor is called "Bipolar Junction Transistor" (BJT). A BJT essentially consists of a pair of Diodes that are joined back-to-back. This forms a sort of a sandwich where one kind of semiconductor is placed in between two others. There are therefore two kinds of bipolar sandwich, the *NPN* and *PNP* varieties. The three layers of the sandwich are conventionally called the Collector, Base, and Emitter.

A bipolar transistor can be thought of as an electronic tap able to control a large flow of electrons with only small variations of the "handle". Voltage changes at the base of the transistor result in changes to the "flow" of electricity through the transistor. Thus, with an electronic "tap," the flow of electricity is controlled by varying the voltage between the **emitter** and **base** of the transistor. The main "flow" is the path between the emitter and the collector. The direction of the main flow, however, depends on the type of the transistor. A more detailed analogy of transistors with flow valves is illustrated in figures below.

In an NPN transistor, the base can be represented by the smaller tube entering the main device from the left side, as shown in figure on the right. The collector is represented by the upper portion of the wider tube, while the emitter is represented by the lower portion of the wider tube. When no current is applied through the "base", i.e., the base is open circuited, the lower lever

arm remains vertical while the top of this arm holds the upper main door shut. This state is analogous to a real NPN transistor off state. In the flow valve analogy, when a small current and pressure are applied to the base tube, the vertical lever is pushed by the entering current and consequently swings counterclockwise, provided there is enough pressure to overcome the force of the spring holding the lever vertical. This spring force is analogous to the minimum 0.6V biasing voltage, relative to the emitter, needed to turn the state on. When this lever arm swings, the upper main door is permitted to swing open a certain amount that is dependent on the amount of swing of the lever arm. In this state, water can make its way from the collector tube to the emitter tube, assuming that there is a slightly higher voltage on the collector side. Notice that in this analogy, the small base water current combines with the collector current. Also, in order to turn on the collector-emitter pathway, the base must be at least 0.6 volts "more positive" with respect to the emitter.

The mechanism of a PNP transistor is similar to the above. However, the main difference is the need for a "negative bias" at the base, relative to the emitter, to turn on the transistor. By allowing current to flow out the base tube, the lever moves, allowing the emitter-collector door to open. The degree of openness varies with the amount of swing in the lever arm, which corresponds to the amount of current escaping through the base tube. Again, note the 0.6V biasing spring at the lever.

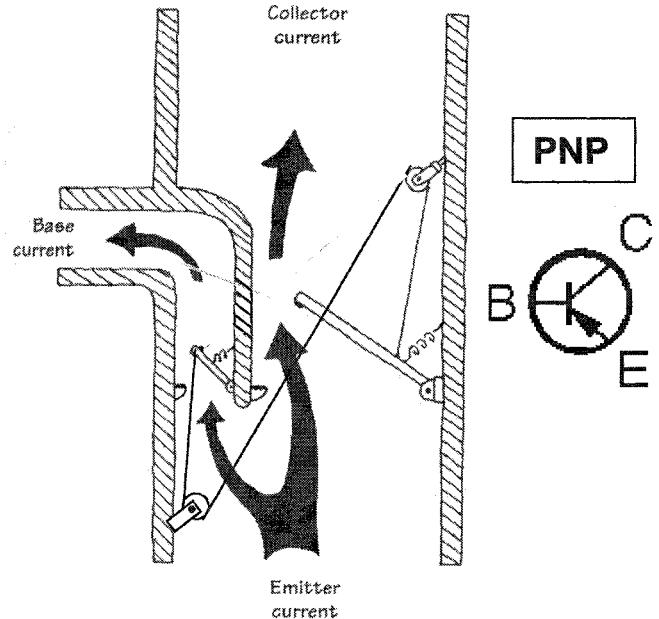


5.2.5.1 Operation

Two important rules of operating bipolar transistors are:

- 1) For an NPN transistor, the voltage at the collector V_C must be greater than the voltage at the emitter V_E by at least a few tenth of a volt. For PNP transistors, the emitter voltage must be greater than the collector voltage by a similar amount.
- 2) For an NPN transistor, there is voltage drop from the base to the emitter of about 0.6V. For a PNP transistor, there is 0.6V rise from base to emitter. In terms of operation, this means that the base voltage V_B of an NPN transistor must be at least 0.6V greater than the emitter voltage V_E . For a PNP transistor, V_B must be at least 0.6V less than V_E .

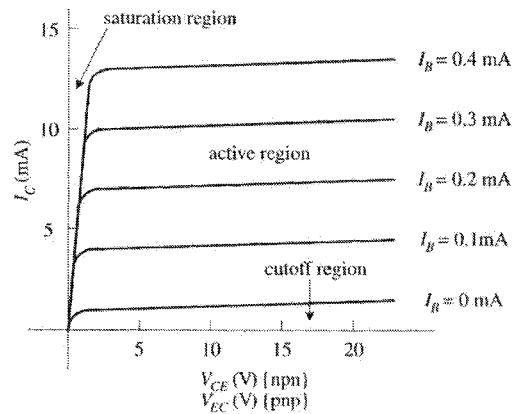
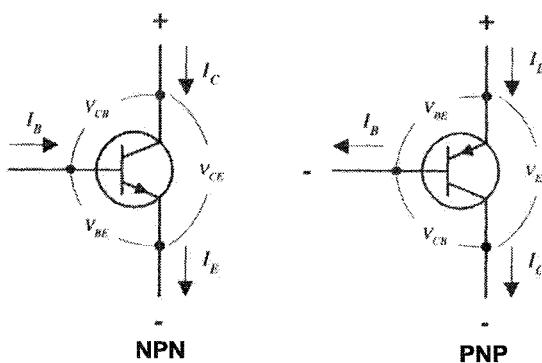
A typical characteristic curve for a bipolar transistor is shown in figure below. This characteristic curve describes the effects the base current I_B and the emitter-to-collector voltage V_{EC} have on the emitter/collector currents I_E and I_C . Some important terms used to describe a transistor's operation include saturation region, cutoff region, active mode region, bias, and quiescent point (Q-point). *Saturation region* refers to a region of operation where maximum collector current flows and the transistor acts much like a closed switch from collector to emitter. *Cutoff region* refers to the region of operation near the voltage axis of the collector characteristics graph, where the transistor acts like an open switch; only a very small leakage current flows in this mode of operation. *Active mode region* describes transistor operation in the region to the right of saturation and above cutoff, where a near-linear relationship exists between terminal currents (I_B , I_C , I_E). *Bias* refers to the specific DC terminal voltages and current of the transistor to set a desired point of active-mode operation, called the *quiescent point*, or *Q-point*.

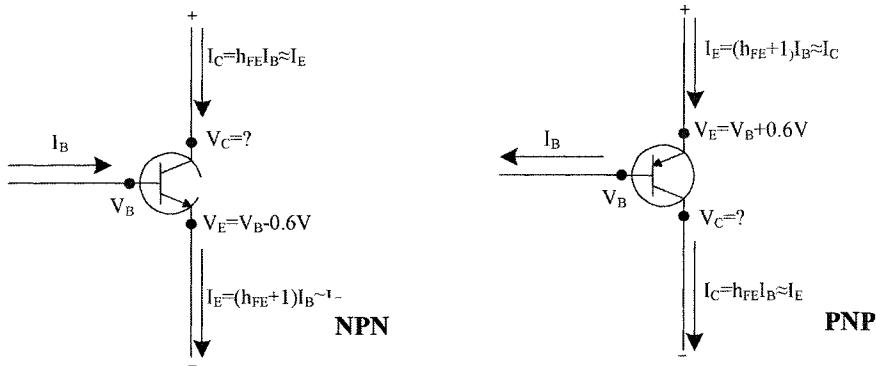


The fundamental formula used to describe the behavior of a bipolar transistor (at least within the active region) is

$$I_C = h_{FE} I_B = \beta I_B$$

where I_B is the base current, I_C is the collector current, and h_{FE} (also referred to as β) is the *current gain*. Every transistor has its own unique h_{FE} . It is often taken to be a constant, typically around 10 to 500, but it may change slightly with temperature and with changes in collector-to-emitter voltage. (A transistor's gain is given in transistor spec. tables.) A simple explanation of what the current-gain formula tells you is this: If you take a bipolar transistor with, say, an h_{FE} of 100 and then feed (NPN) or sink (PNP) a 1-mA current into (NPN) or out of (PNP) its base, a collector current of 100 mA will result. It is important to note that the current-gain formula applies only if rules 1 and 2 are met, i.e., assuming the





transistor is within the active region. Also, there is a limit to how much current can flow through a transistor's terminals and a limit to the size of voltage that can be applied across them.

By applying the law of conservation of current (follow the arrows in figure above), you get the following expression relating the emitter, collector, and base currents:

$$I_E = I_C + I_B$$

Combining the above equation with the current-gain equation will result in:

$$I_E = (h_{FE} + 1)I_B$$

In practice, adding 1 to h_{FE} is not significant as long as h_{FE} is sufficiently large (which is almost always the case). This means that you can make the following approximation:

$$I_E \approx I_C$$

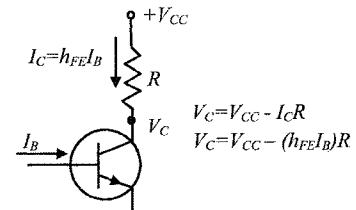
Figure above shows how all the terminal currents and voltages are related. In the figure, notice that the collector voltage has a question mark next to it. As it turns out, the value of V_C cannot be determined directly by applying the formulas. Instead, V_C depends on the network that is connected to it. For example, if you consider the setup shown in figure on the right, you must find the voltage drop across the resistor in order to find the collector voltage. By applying Ohm's law and using the current-gain relation, V_C can be calculated. The results are shown in the figure.

It is important to note that the above equations are valid within the voltage and current bounds provided by the transistor characteristic curves. If you apply the equations blindly, without considering the operating characteristics, you could end up with some wild results that are physically impossible.

Another characteristic of transistors is what is called *transresistance* R_{tr} . Transresistance represents a small resistance that is inherently present within the emitter junction region of a transistor. Two things that determine the transresistance of a transistor are temperature and emitter current flow. The following equation provides a rough approximation of R_{tr} :

$$R_{tr} \approx \frac{0.026V}{I_E}$$

In many cases, R_{tr} is insignificantly small, so that its effect can be neglected in the circuit. Nevertheless, in a real transistor, any restriction to the current flow causes heat to be produced. This happens with air or water in other things: for example, your bicycle pump becomes hot near the valve when you pump air through it. A transistor must be kept cool or it will melt. It runs coolest when it is fully OFF and fully ON. When it is fully ON there is very little restriction. Thus, even though a lot of current is flowing, only a small amount of heat is produced. When it is fully OFF, provided we can stop the base leakage, then NO heat is produced. If a transistor is half on then significant current is flowing through a restricted gap and heat is produced. To help dissipate this heat, the transistor might be clamped to a metal plate (heat sink), which draws the heat away and radiates it to the air.



Transistors are used in two basic ways:

- To provide amplification of a small changing voltage (a “signal”) present at the base. You can control a “big” flow of current with a “small” flow of current at the base. If the small current flow varies continually, then the corresponding changes in the large current flow will happen. However, The amount of current that can flow from emitter to collector is limited by the “pipe diameter”. So, no matter how much current we push into the base, there will be a point beyond which we cannot get any more current flow from emitter to collector. The only way to solve this problem is to use a larger transistor, e.g., a “power transistor”.
- As an electronic “switch”, i.e., they are either fully ON, or fully OFF. (e.g., to switch a relay.) If we put sufficient current into the base, the transistor will allow the maximum amount of current to flow from emitter to collector. The transistor is switched fully ON. If the current into the base is reduced to the point where it can no longer push the “handle”, the transistor will be OFF. Only a small “leakage” current from base will be flowing. To turn it fully off, we must stop all current flowing into the base.

Biassing Transistors

To keep the transistor operating within its useful range, a predetermined potential difference must be established between emitter and base and base and collector. Two common methods for “biasing” a transistor are to use either a voltage-divider circuit (left side) or a Zener diode regulator (right side), as shown in figure below. In the voltage-divider circuit, the base voltage is set by R_1 and R_2 , and is equal to

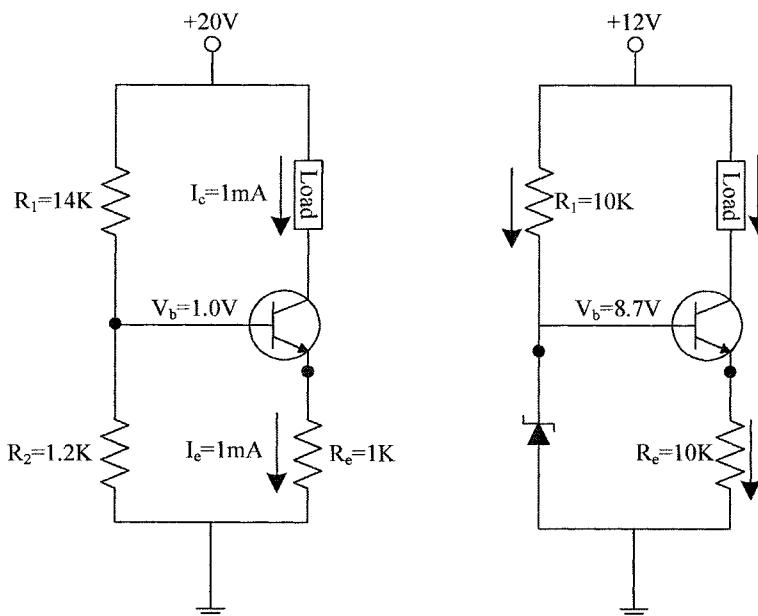
$$V_B = \frac{R_2}{R_1 + R_2} V_{CC}$$

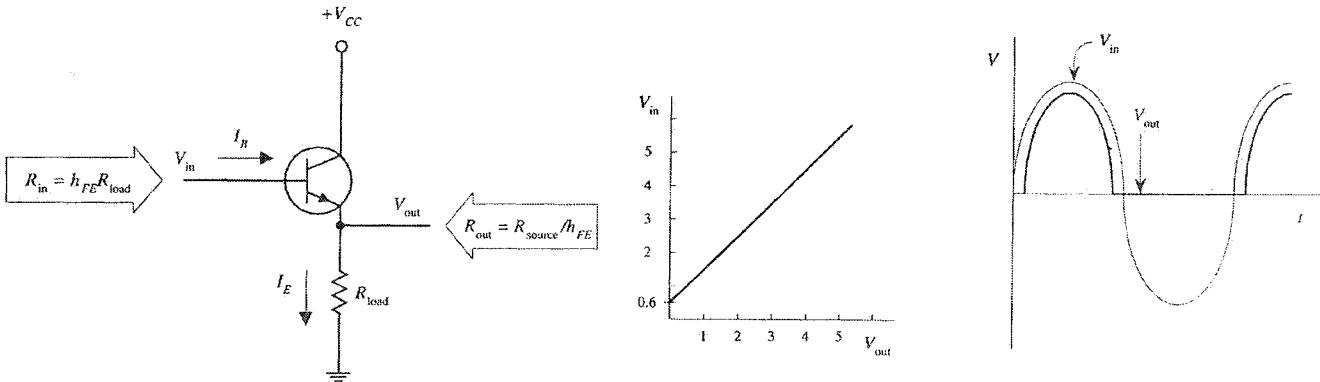
In the Zener diode circuit, the base voltage is set by the Zener diode’s breakdown voltage such that

$$V_B = V_{Zener}$$

In this way, by applying a small input voltage and current at the base (for NPN transistor), a larger collector/load current can be controlled. The collector/load current is related to the base voltage by

$$I_C \approx I_E = \frac{V_B - 0.6V}{R_E}$$





Emitter Follower (Common Collector) Amplifier

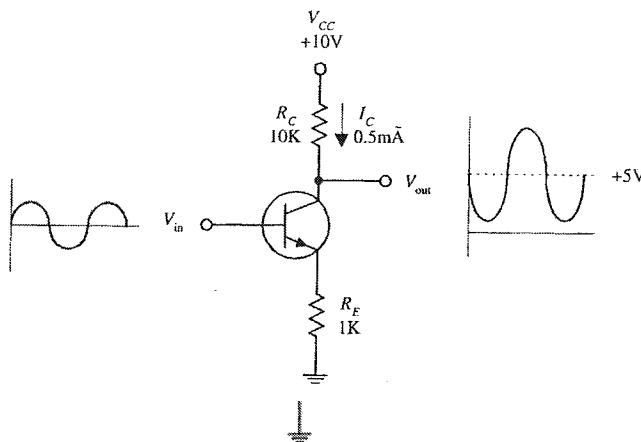
The circuit shown above (left) is called an emitter follower. In this circuit, the output voltage (tapped at the emitter) is almost a mirror image of the input (output "follows" input), with the exception of a 0.6V drop in the output relative to the input (caused by base-emitter pn-junction). Also, whenever $V_B \leq V_E + 0.6V$ (during negative swings in input), the transistor will turn off (the pn-junction is reversed-biased). This effect results in clipping of the output, as shown in the above graph (right). At first glance, it may appear that the emitter follower is useless; it has no voltage gain. However, if you look at the circuit more closely, you will see that it has a much larger input impedance than an output impedance, or more precisely, it has a much larger output current (I_E) relative to an input current (I_B). In other words, the emitter follower has current gain, a feature that is just as important in applications as voltage gain. This means that this circuit requires less power from the signal source (applied to V_{in}) to drive a load than would otherwise be required if the load were to be powered directly by the source. By manipulating the transistor gain equation and using Ohm's law, the input resistance and output resistance are:

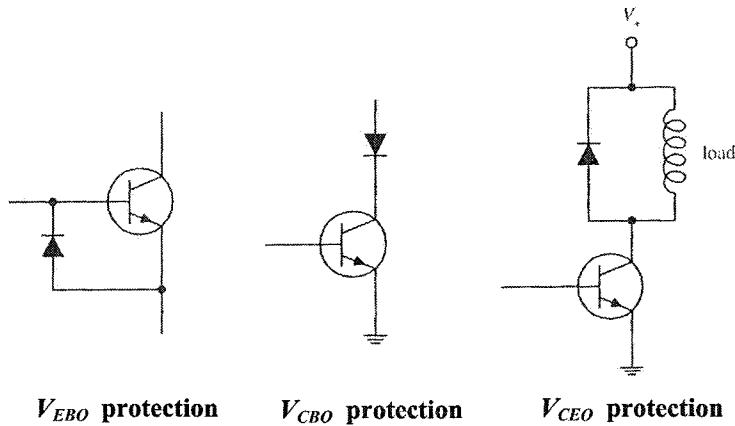
$$R_{in} = h_{FE} \cdot R_{load} \quad \text{and} \quad R_{out} = \frac{R_{source}}{h_{FE}}$$

Common Emitter Amplifier

The transistor configuration shown below is referred to as the common-emitter configuration. Unlike the emitter follower, the common emitter has voltage gain. The voltage gain can be calculated from the following (R_{tr} is the transresistance of the transistor):

$$\frac{V_{out}}{V_{in}} = \frac{R_C}{R_E + R_{tr}}$$





Transistor Characteristics

Transistors are featured by their characteristics, such as type (NPN, PNP, or FET), maximum collector-current rating (I_{Cmax}), collector-to-base breakdown voltage (V_{CBO}), collector-to-emitter breakdown voltage (V_{CEO}), emitter-to-base breakdown voltage (V_{EBO}), and maximum collector power dissipation rating (P_D). If these ratings are exceeded, the transistor may burn out. One method to safeguard against V_{EBO} is to place a diode from the emitter to the base, as shown in figure above (left). The diode prevents emitter-to-base conduction whenever the emitter becomes more positive than the base (e.g., input at base swings negative while emitter is grounded). To avoid exceeding V_{CBO} , a diode placed in series with the collector (figure above, middle) can be used to prevent collector-base conduction from occurring when the base voltage becomes excessively larger than the collector voltage. To prevent exceeding V_{CEO} , which may be an issue if the collector holds an inductive load, a diode placed in parallel with the load (figure above, right) will go into conduction before a collector voltage spike, created by the inductive load, reaches the breakdown voltage.

It is important to note that the current gain (h_{FE}) of a transistor is usually not a reliable parameter. It can vary with changes in collector current, collector-to-emitter voltage, and temperature.

5.2.5.2 Types of Bipolar Transistors

Transistors come in all shapes and sizes. Generally they have three “legs”, but not all three-legged components are transistors. The size of a transistor is usually determined by the amount of current they are required to handle. Large-current transistors are physically large and often have enhanced cooling features such as a metal case.

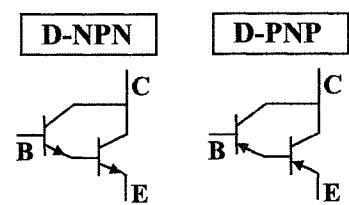
Major categories of bipolar transistors are listed as:

Small Signal Transistors: used to amplify low-level signals or for switching. Typical h_{FE} values range from 10 to 500, with maximum I_C ratings from 80 to 600 mA. Maximum operating frequencies range from about 1 to 300 MHz.

Small Switching Transistors: used primarily as switches. Typical h_{FE} values range from 10 to 200, with maximum I_C ratings from 10 to 1000 mA. Maximum operating frequencies range from about 10 to 2000 MHz.

Power Transistors: used in high-power amplifiers and power supplies. The collector is connected to a metal base that acts as a heat sink. Typical power ratings range from around 10 to 300 W, with frequency ratings from about 1 to 100 MHz. Maximum I_C values range between 1 and 100 A.

Darlington Transistors: These are two transistors in one. They provide more stability at high current levels. The effective h_{FE} of a Darlington transistor is about the multiplication of the gains of the two individual transistors, hence usually much larger than a single transistor. Note that the base-emitter voltage drop in a Darlington transistor is also about twice the voltage drop of each single transistor. Less base current is required to turn on a Darlington, so when replacing a regular transistor with a Darlington, increase the original base drive resistor by a factor of



10 to compensate for the higher gain.

Phototransistors: act as light-sensitive bipolar transistors (base is exposed to light.) When light hits the base region, a base current results. Depending on the type of phototransistor, the light may act exclusively as a biasing agent (two-lead phototransistor) or may simply alter an already present base current (three-lead phototransistor).

Transistor Array: This consists of a number of transistors combined into a single integrated package. For example, the transistor array shown in the figure is made of three NPN and two PNP transistors.

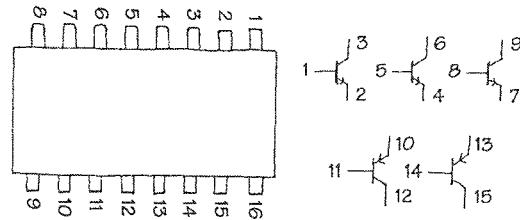
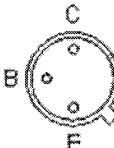
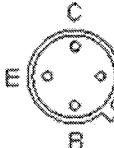
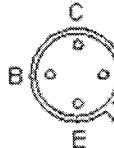
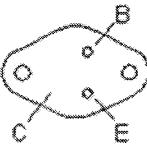
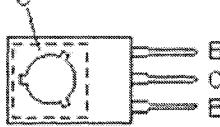
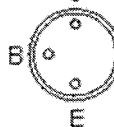
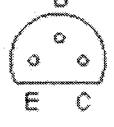
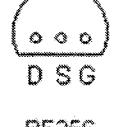
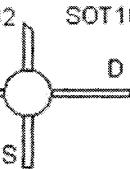
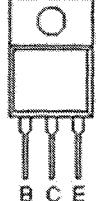
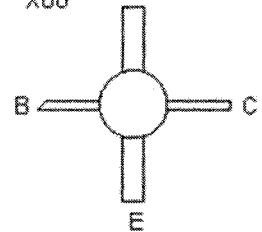
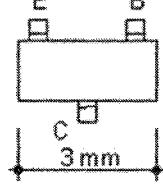


Table below lists some popular transistors that are used in small and medium-size projects. In general, transistors are manufactured in different package types. These types are coded so that the transistor pinouts can be identified in most cases. Figure in the next page shows a variety of transistor packages and their pinouts. Pinouts of a bipolar transistor can also be identified by using a digital multi-meter. The procedure for an NPN transistor is as follows:

- 1) Set the meter to the diode check position,
- 2) Connect V/Ω lead to "base".
- 3) Connect COM lead to "collector".
- 4) Meter must now show a forward voltage drop V_f between 0.5V and 0.75 V.
- 5) Short "base" to "emitter".
- 6) Meter reading should drop slightly something between 0.004V and 0.1V.
- 7) Connect COM lead to "emitter", and leave V/Ω lead to "base".
- 8) Meter must show V_f as in step 4.
- 9) Short "base" to "collector".
- 10) Meter reading should decrease as in step 6.

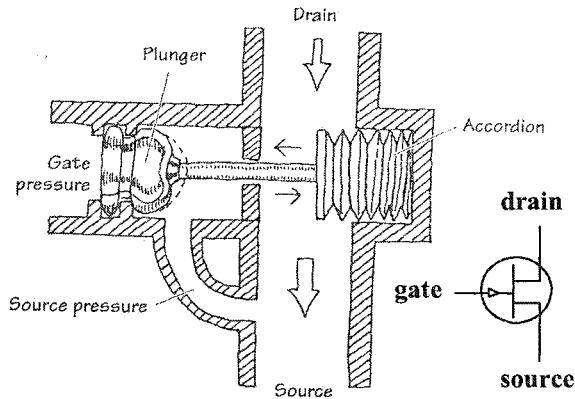
Darlington transistors show slightly larger drops in steps 6 and 10. PNP transistors can also be tested through the same procedure but with COM and V/Ω leads interchanged.

Item	Category	Type	Power	I_{Cmax}	V_{CEO}	Frequency	h_{FE}	Pkg.	Substitute
2N3904	regular	NPN	350mW	200mA	40V	300MHz	30-100	TO-92	2N2222
2N3906	regular	PNP	350mW	200mA	40V	250MHz	30-100	TO-92	2N2907
TIP29	regular	NPN	30W	1A	40V	3MHz	15-75	TO-220	TIP29A/B/C
TIP30	regular	PNP	30W	1A	40V	3MHz	15-75	TO-220	TIP30A/B/C
2N3055	regular	NPN	115W	15A	60V	2.5MHz	20-70	TO-3	
MJ2955	regular	PNP	115W	15A	60V	2.5MHz	20-70	TO-3	
TIP110	Darlington	NPN	50W	2A	60V	25MHz	500	TO-220	TIP111/112
TIP115	Darlington	PNP	50W	2A	60V	25MHz	500	TO-220	TIP116/117

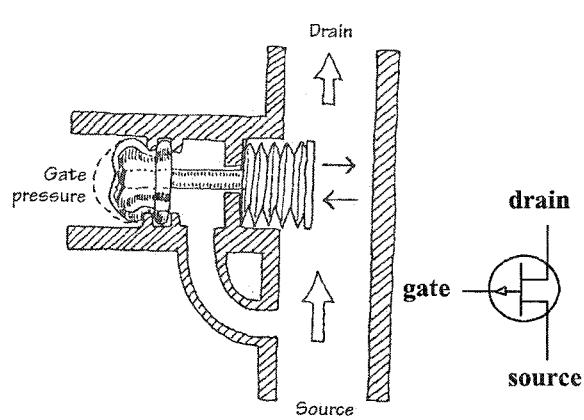
Package TO5, TO18		TO72				TO3 and similar	
							
NPN	PNP	NPN	PNP	NPN	PNP	NPN	PNP
BC107 BC108 BC109 BFX84 BFX50 2N706 2N2369 BC286	BC177 BC178 BC179 BC186 BC187 BCY71 BC287 2N2904	BF180 BF181 BF182 BF183 BF200	AF139 AF178 AF179 AF180 AF181	BF115 BF167 BF173 BF184 BF185	AF124 AF125 AF126 AF127	2N3055 BDY20 BD121 BD123 AD161	PNP3055 BDX16 OC26 AD149 AD162
TO126		TO-1		X-55		TO92	
							
NPN	PNP	NPN	PNP	NPN	PNP	NPN	PNP
BD135 BD131 BD437 BUP41	BD136 BD132 BD438	AC176 AC187	AC128 AC188	BC182 BC183 BC184 2N3707 2N3710	BC212 BC213 BC514 2N3702 2N3703	BC183L BC237B BC547 BC548 BC546 2N3705 2N3903	BC213L BC557 BC307B 2N4402
TO18							
Unijunction transistor		3N140 3N141 40673		BF256 2N3819 (connection FET)		BF960 BF961 3SK81	
2N2646 2N2647	2N4870 2N4871						
TO220		NPN	PNP	X80	E	SMD	SOT23
		BD539 BD743 TIP29C BU407 BUP30 2N6099 BD243C D44C10	BD540 BD744 TIP30C BD244C BD240C BD242C				
BD241C				BFR14	BFR49		
						1,3 mm	
						3 mm	
							PNP
							BC846B BC847B BC848B BC849B
							BC856B BC857B BC858B BC859B

5.2.5.3 Junction Field-Effect Transistor (JFET)

Junction field-effect transistors (JFET) are also three-lead semi-conductive devices similar to bipolar transistors. However, two main difference between JFETs and bipolar transistors are: a) JFETs do not require a biasing current; and b) they are normally ON when there is no voltage difference between its “gate” (corresponding to bias in bipolar) and “source” (corresponding to emitter in bipolar) leads. The “water analogy” of JFETs, both n-channel and p-channel, are shown in figures below.

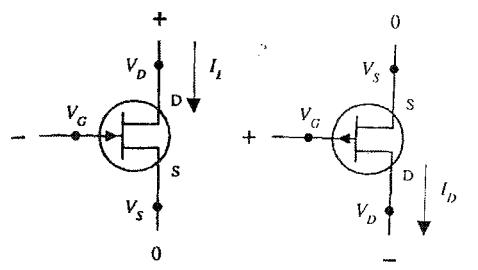


n-channel JFET



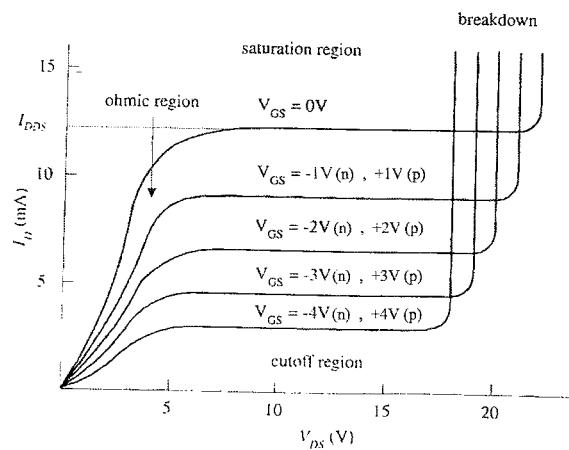
p-channel JFET

Figure below presents typical characteristic curves and current flow of JFETs. The characteristic curves describe how the drain current (I_D) is influenced by the gate-to-source voltage (V_{GS}) and the drain-to-source voltage (V_{DS}). When the gate voltage is set to the same voltage as the source ($V_{GS} = 0$), maximum current flows through the JFET. The corresponding current is called the *drain current for zero bias* (I_{DSS}). I_{DSS} is a constant and depends on the type of JFET. When V_{DS} is small, the drain current varies nearly linearly with V_{DS} . This region is called *ohmic (linear) region*, in which JFET behaves like a voltage-controlled resistor. The region in the curves where I_D flattens is called *active region*. In this region, the drain current is strongly influenced by the gate-source voltage V_{GS} but hardly at all by the drain-to-source voltage V_{DS} . Another thing to note is the value of V_{GS} that causes the JFET to turn off, i.e., practically no current flows through device. The particular V_{GS} that turns off the JFET is called *cutoff (or pinch-off) voltage* $V_{GS,off}$. As V_{DS} increases, there is a point where the JFET loses its ability to resist current, because too much voltage is applied across its drain-source terminals. This effect is referred to as *drain-source breakdown*, and the respective voltage is expressed as $V_{B,DS}$. For a typical JFET, I_{DSS} values range from about 1 mA to 1 A, $V_{GS,off}$ from around -0.5 to -10 V for an n-channel JFET (from $+0.5$ to $+10$ V



n-channel JFET

p-channel JFET



for p-channel), and $V_{B,DS}$ from about 6 to 50 V. Another parameter of JFETs is *on-resistance* ($R_{DS,on}$), which is the internal resistance of the device when in its fully conducting state ($V_{GS}=0$). This parameter typically ranges from 10 to 1000Ω.

Some useful formulations for JFETs are listed below. $V_{GS,off}$ and I_{DSS} are typically the known parameters.

Drain Current (Ohmic Region):
$$I_D = I_{DSS} \left[2 \frac{V_{DS}}{-V_{GS,off}} \left(1 - \frac{V_{GS}}{V_{GS,off}} \right) - \left(\frac{V_{DS}}{V_{GS,off}} \right)^2 \right]$$

Drain Current (Active Region):
$$I_D = I_{DSS} \left(1 - \frac{V_{GS}}{V_{GS,off}} \right)^2$$

Drain-Source Resistance:
$$R_{DS} \approx \frac{V_{GS,off}}{2I_{DSS}(V_{GS} - V_{GS,off})}$$

Drain-Source Voltage:
$$V_{DS} = V_{GS} - V_{GS,off}$$

5.2.5.4 Metal Oxide Semiconductor Field-Effect Transistors (MOSFET)

MOSFETs are popular transistors that in many ways resemble JFETs. For instance, when a small voltage is applied at their gate lead, the current flow through their drain-source channel is altered. However, unlike JFETs, MOSFETs have larger gate lead input impedances ($\geq 10^{14} \Omega$, as compared with $\sim 10^9 \Omega$ for JFETs), which means that they draw almost no gate current. Hence, the main merit is their extremely low on-resistance, and ability to control high currents with only a small steering current. Small on-resistance ensures low power dissipation, and minimal heating in these devices. The pay off is the very-low gate-to-channel capacitance, which makes these devices extremely fragile. For instance, you could blow out a MOSFET by simply walking across a carpet and then touching the gate of the MOSFET. The charge you pick up during your walk may be large enough to set yourself to a potential of a few thousand volts. Although the amount of current discharged during the contact is not significantly large, the gate-channel capacitance is so small that even a small current can be fatal to the MOSFET. Thus, when installing MOSFETs, it is essential to eliminate all static electricity from your work area.

5.2.6 Switches and Relays

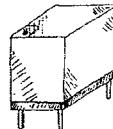
A switch is a mechanical device that interrupts or diverts electric current flow within a circuit. It is characterized by its number of *poles* and *throws*. A pole (P) is the path through which the current flows into the switch. A throw (T) is the path that the current flows out of the switch. A switch is specified by the number of its poles and throws. Letter "S" is used for single pole or throw, and "D" is used for double poles or throws and digits 3, 4, ..., are used for more. For example, DPST represents a switch with double pole and single throw, and SP3T represents a switch with single pole and three throws. Some popular switches are shown in figure below.

Two additional features that may come with switches are "momentary contact action" and "centre-off position." Momentary-contact switches (mainly for pushbutton switches) are used where only a brief connection or disconnection of a circuit is needed. They come in either Normally Closed (NC) or Normally Open (NO) configurations. Centre-off position switches have an additional OFF position located between the two ON positions.

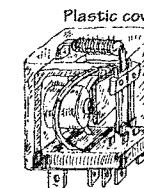
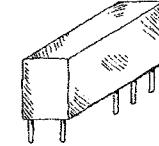
Relays are electrically actuated switches. Three basic types of relays are:

Mechanical Relays: In mechanical relays, a current sent through a coil magnet acts to pull a flexible, spring-loaded conductive plate from one switch contact to another. They are designed for switching relatively large currents with low speeds. The excitation voltage rating of DC-actuated relays is 6, 12, or 24V, with coil resistance of about 40, 160, and 650 Ω , respectively. Switching speeds range from about 10 to 100 ms, and current ratings range from about 2 to 15 A. Some *miniature* mechanical relays can have coil resistances up to 3000 Ω .

Miniature Relay

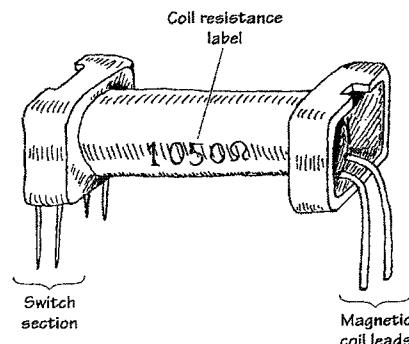


Subminiature Relay

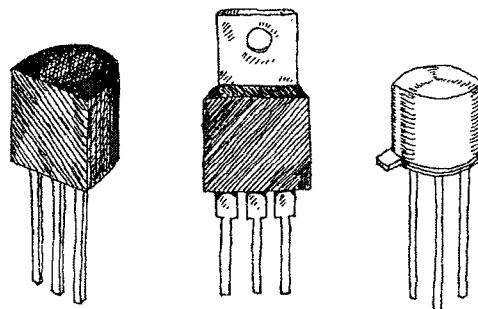


Reed Relays: In reed relays, two thin metal strips, or reeds, act as movable contacts. The reeds are placed in a glass-encapsulated container that is surrounded by a coil magnet. When current is sent through the outer coil, the reeds are forced together, thus closing the switch. The low mass of reeds allows for quick switching, typically around 0.2 to 2 ms. These relays are DC-actuated designed to switch moderate currents (typically 500 mA to 1 A), and come with excitation voltages of 5, 6, 12, and 25 V. Their coil resistance is around 250 to 2000 Ω .

Reed Relay



Solid-State Relays: These relays are made from semiconductor materials, thus do not cause "contact wear", and have extremely fast switching (typically 1 to 100 ns). However, they typically have high on-resistance,



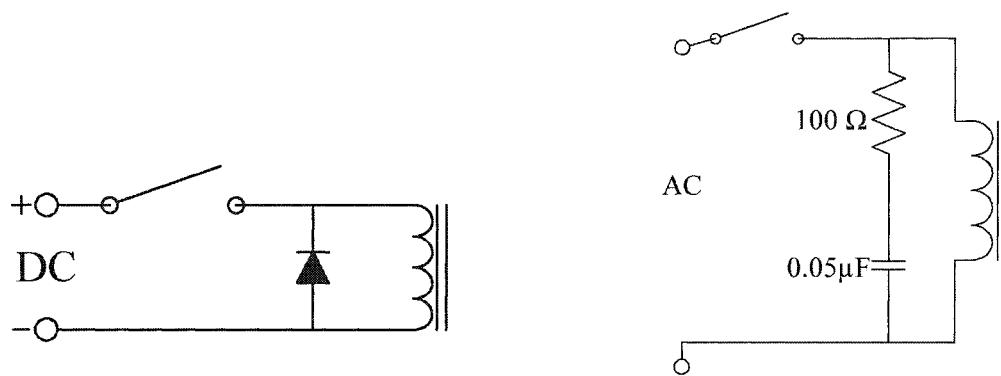
Solid-State Relays

they require fine-tuning, and they are less durable to overloads compared to electromechanical relays.

There are several practical notes that must be taken into account when using relays in a circuit:

To make a relay change states, the voltage across the leads of its magnetic coil should be at least within $\pm 25\%$ of the relay's specified control-voltage rating. Too much voltage may damage the magnetic coil, whereas too little voltage may not trigger the relay or may cause the relay to act erratically (flip back and forth).

The coil of a relay acts as an inductor. Hence, sudden changes of current through the coil can cause a very large voltage across its leads, causing a large surge of current through it. Surges in current that result from inductive behaviour can create menacing voltage spikes that can have some unpleasant effects on neighboring devices within the circuit. To get rid of spikes, *transient suppressors* must be used. A simple yet effective example of a transient suppressor is shown in figure below, where a diode in reverse bias is placed in parallel with the switch (or motor or any other device with inductance). This will eliminate voltage spikes by going into conduction before a large voltage can form across the coil. A proper general-purpose diode that works well for this application is 1N4004 diode.



5.2.7 Batteries

WARNING: Never short-circuit the two-leads of a battery. Some batteries (particularly Nickel-Cadmium and Lead Acid cells can deliver very high currents that may cause a fire or severe burns. Industrial accidents have been reported where metal watch bands have fused across the terminals of high capacity batteries. In addition, batteries can be destroyed or have their performance permanently impaired by a short circuit test.

Literally speaking, a battery is a series or parallel combination of *cells*, the fundamental electrochemical units that produce electric current. Unlike capacitors, batteries do not store electrical charge, they generate electric current through chemical reactions. Once these chemicals are depleted, the battery is flat. Rechargeable, or *secondary* batteries can be regenerated by putting back more energy than was taken out, (typically 50% more). Any battery can be recharged to some extent from a half-wave rectified source. However, some *primary* or one-time use batteries may explode if they are over-charged. Lithium and mercury cells are especially hazardous.

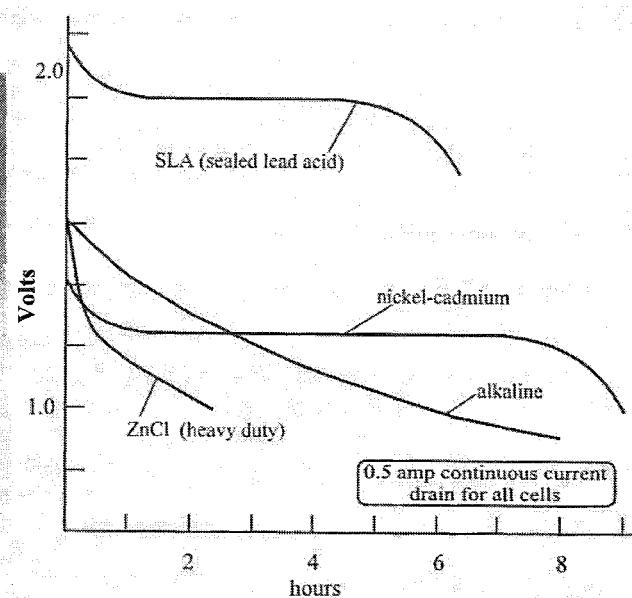
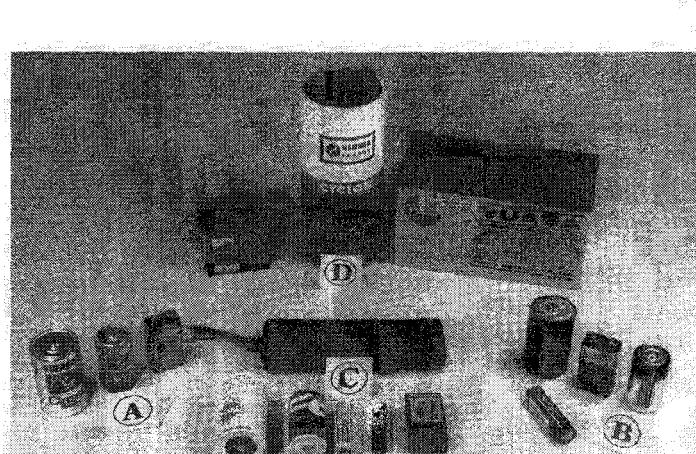
Some batteries commonly found in the market and used in small/medium projects are shown in figure below (left) and explained in the following.

a) Heavy Duty (Super Heavy Duty) or ZnCl (zinc-chloride) batteries are inexpensive one-time-use primary cells. They provide a continuously and rapidly decaying voltage output under a constant current load (figure below, right). The nominal voltage for various sized of this category is 1.5 V, and although these cells continue to provide power below 1 V, this low-voltage energy cannot be used efficiently if regulated outputs are necessary (in applications such as microcontrollers, op amps, and optical detectors.)

b) Regular Alkaline batteries are also primary cells, but their quality is higher than that of ZnCl cells hence providing better performance than their counterparts. Voltage decay is significantly slower than ZnCl cells under a constant current (figure below, right). There also exist rechargeable Alkaline batteries in the market, but since their total stored energy decreases with each recharge (for a maximum of about 25 charges), their repeated usage (for more than 4-5 times) may not give a reliable performance.

c) NiCd (nickel-cadmium or *Nicads*) cells are rechargeable secondary batteries that have a useful flat voltage region of predictable duration under a constant current load (figure below, right). They can be charged and recharged for quite a number of times, but they should be looked after carefully in terms of their storage and working conditions.

d) SLA (Sealed Lead Acid or *gel*) cells have similar performance as NiCd cells do, but they are typically rated at higher nominal voltages and thus powers, but they have shorter charge duration than NiCd cells.



Size	Outside Dimensions	Volume (cc)	Volume Ratio (D cell =1)
AAA	1 cm dia. \times 4.2 cm	3.3	0.067
AA	1.4 cm dia. \times 4.7 cm	7.2	0.15
C	2.5 cm dia. \times 4.6 cm	23	0.47
D	3.3 cm dia. \times 5.7 cm	49	1.00
9 volt	1.7 \times 2.5 \times 4.5 cm		~0.06

Each category of batteries comes in different sizes. Although dimensions vary even among cells of the same size mainly due to packaging, but typical outside dimensions for standard sizes of readily available consumer products are listed in table above (terminal projections are not included.) In terms of the weight, even for cells of the same type and size, cell weight may differ by 10% or more, so weigh each cell if weight is critical for your application. Some manufacturers package C size NiCd cells into a D size case, and these are often sold at D cell prices. If you suspect a D cell is a lightweight, it is probably a masquerading C cell. Battery and cell weights of the most commonly used items are listed in table below.

Discharge Rate (A)	NiCd	Alkaline	ZnCl	SLA
0.1	5.9	8.7	3.4	2.8
0.5	4.3	3.3	1.4	2.5
1.0	3.8	1.1	0.75	1.8
Initial Voltage (V)	1.25	1.58	1.60	2.15
Final Voltage (V)	1.00	1.00	1.00	2.00

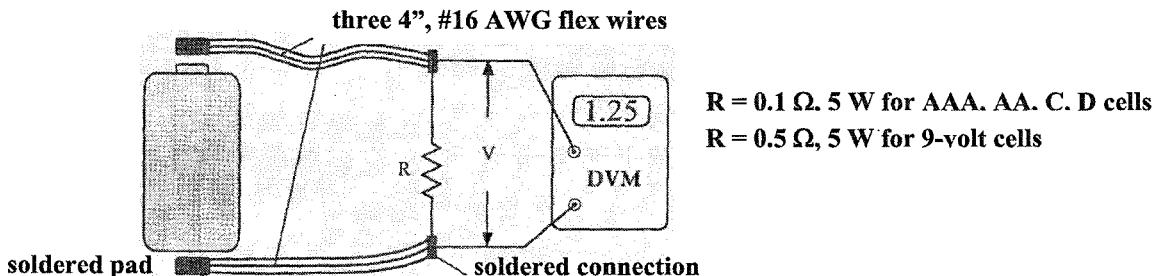
In terms of the electrical capacity, each battery is assigned a number in ampere-hours (Ah) that indicates the battery's energy capacity when discharged at a rate equal to one-tenth of the assigned rating. For example, if a D-size Nickel-Cadmium cell is rated as 4 Ah, this means that if this cell is discharged at 0.4 A, it will take 10 hours for cell voltage to drop below the low-level threshold (the low-level threshold is 1 V for 1.5V batteries.) Table below shows the ampere-hour capacity for D size of different cells.

Size	NiCd	Alkaline	ZnCl	SLA
AAA	10g	12g	10g	
AA	22g	22g	18g	
C	52g	66g	48g	
D	158g	142g	102g	180g
9 volt	36g	46g	38g	

The energy content of different sizes (of each type) is closely proportional to their volume. Thus, by having the Ah ratings for the D size and also the volume ratio of different sizes with respect to the D size, as given in the table on top of the page, the capacity of each size (of each type) can simply be calculated by multiplying the corresponding capacity for the D size by the volume ratio. For example, the capacity of the AAA size of an Alkaline cell is equal to $8.7\text{Ah} \times 0.067 = 0.58\text{ Ah}$.

Every battery has the equivalent of a resistor inside each cell, limiting the maximum current that the battery can provide. This "internal resistance" is an important parameter, because it governs voltage available at the battery's terminals when it is under load. **Measuring the battery's internal resistance is not trivial, since you must not short-circuit the battery leads through an ohmmeter.** The internal resistance can be measured safely and accurately as follows:

- 1) Connect a small resistor R to the battery leads as shown in figure below. Since the battery internal resistance r is usually less than 1Ω , short lengths of flexible heavy gage wire must be used for this test. Three separate wires are more flexible than one heavy wire, and soldered ends make good contact pads.



2) Touch soldered wire ends to the battery firmly by QUICKLY and record V from the multimeter. Do not wait for the multimeter reading to settle to a stable value; by that time the battery will be flat. Calculate the current as:

$$i = \frac{V}{R}$$

3) Measure the battery open-circuit voltage V_0 , with R removed from the battery. Calculate the amount of the internal resistance r as follows:

$$r = \frac{V_0}{i} - R$$

Design Considerations

1. Batteries are not made to precision tolerances. Allow ± 0.5 mm tolerance for both diameter and length for cylindrical batteries, and as much as ± 5 mm tolerance for lead acid and sealed lead acid cells.
2. Don't measure the length of a live cell with metal calipers! Be careful! batteries are different than other components. They can short from keys, from metal tools, or coins, so be careful of putting them in your purse, tool box, or pocket.
3. Open Cell Voltage (voltage measured with a high impedance meter) is a function of the cell chemistry and internal leakage. Even dead cells can register the same OCV as good cells. The test of a battery is its voltage under load.
4. When using battery packs, be careful not to inadvertently short the cells. A pack of cells wired in series will become shorted if the cases of adjacent batteries touch, since the outer case is a terminal. This can happen if the cells are shrink wrapped, film wrapped or painted and the batteries rub against each other. Brittle shrink wrap is known to shred under stress, leaving the bare cell walls to touch.
5. When using or designing battery holders make sure there is adequate provision for short cells, long cells, or wide cells. Keep sharp clip edges from touching the cell where they could cut the film or paint, causing a short between cells held by the same clip.
6. Position the cells away from heat sources if possible. Heat will cause an increase in self-discharge, and will also decrease the life of the battery. The old rule-of-thumb of twice the self-discharge every 10°C increase in temperature still holds.
7. If there are alternate ways to power your device, make sure to isolate the battery from the alternate sources of power to prevent inadvertent charging. In the case of high voltage batteries isolate the battery from terminals of alternate power sources.
8. Some cells, such as CMOS maintenance lithium batteries are designed for a very low rate of discharge. Some meters will load them down with too much current, and so will give incorrect readings. Make sure you are using a high impedance meter.
9. Be careful to match the cells in a battery pack. When a battery pack is near zero volts under load the weaker cells will go into reversal, and suffer damage and perhaps venting.
10. When using rechargeable cells, don't over-discharge below the rated discharge voltage. For NiCad this is about 1 volt per cell. For lead acid, it is about 1.75 volts per cell (10.5 volts for a 12 volt battery).
11. Take care when designing a battery holder to use batteries in parallel. A backwards battery will short out the pack, and customers typically expect the batteries to alternate orientation.
12. Batteries also expand and contract during charge and discharge. Potting a battery is not a good idea, unless there is some provision made for this dimension change.
13. Over the course of life, most batteries release hydrogen, and sometimes oxygen. Take this into account if you are designing a closed system, such as waterproof lights, weatherproof installations, etc. Some methods of releasing or absorbing the hydrogen, flooding with air or inert gas should be used. In closed cabinets some provision for ventilation is necessary to prevent hydrogen gas from accumulating

5.3 Digital Electronics

Digital electronics is based on digital (or binary) logic. The basic idea is that a two-valued (binary) number system can possibly be the basis for advanced and complicated circuits. The binary number system involves only two digits: 0 and 1. Hence, when dealing with digital logic every statement or condition could only be either “true” or “false”. While this approach may seem limited, it actually works quite effectively, and can be expanded to express very complex relationships and interactions among any number of individual conditions. One essential reason for basing logical operations on the binary number system is that it is easy to design simple, stable electronic circuits that can switch back and forth between two clearly-defined states, with no ambiguity attached. It is also readily possible to design and build circuits that will remain indefinitely in one state unless and until they are deliberately switched to the other state. This makes it possible to construct a machine that can remember sequences of events and adjust its behavior accordingly.

Digital logic may be divided into two classes: *combinational logic*, in which the logical outputs are determined by the combination of the logical input states at that particular moment (without using the history); and *sequential logic*, in which the outputs also depend on the prior states of outputs and inputs. In other words, sequential logic is a sort of combinational logic with memory. Both classes of logic implement with logic *gates* to combine logical input signals in various ways to produce the desired outputs.

5.3.1 Number Systems

The **decimal** system is based on the use of 10 symbols or digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor of 10 as we proceed from right to left.

[...	10^3	10^2	10^1	10^0]
	thousands	hundreds	tens	units

The **binary** system is based on only two symbols or states: 0 and 1. These are termed binary or *bits*. When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor of 2 as we proceed from right to left.

[...	2^3	2^2	2^1	2^0]
	bit 3	bit 2	bit 1	bit 0

For example, the decimal number 15 in the binary system is 1111. In a binary number, the bit 0 is termed the least significant bit (LSB) and the highest bit is named the most significant bit (MSB).

Example: Decimal to Binary Conversion: 109 (decimal) to binary

109/2 = 54 w/ remainder 1 (LSB)	_____		
54/2 = 27 w/ remainder 0	_____		
27/2 = 13 w/ remainder 1	_____		
13/2 = 6 w/ remainder 1	_____	Answer:	1101101
6/2 = 3 w/ remainder 0	_____		
3/2 = 1 w/ remainder 1	_____		
1/2 = 0 w/ remainder 1 (MSB)	_____	8-bit answer:	01101101

Example: Decimal to Binary Conversion: 10100100 (binary) to decimal

$$1 (\text{MSB}) \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 (\text{LSB}) \times 2^0 = 164 \text{ (decimal)}$$

The **octal** system is based on eight digits: 0, 1, 2, 3, 4, 5, 6, 7. When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor of 8 as we proceed from right to left.

[...	8^3	8^2	8^1	8^0]
-------	-------	-------	-------	---------

For example, the decimal number 15 in the octal system is 17.

The **hexadecimal** system is based on 16 digits/symbols: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. When a number is represented by this system, the digit position in the number indicates that the weight attached to each digit increases by a factor 16 as we proceed from right to left.

$$[\dots \quad 16^3 \quad 16^2 \quad 16^1 \quad 16^0]$$

For example, the decimal number 15 is F in the hexadecimal system. This system is generally used in the writing of programs for microprocessor-based systems since it represents a very compact method of entering data.

Octal and hexadecimal numbers can be translated easily to and from binary. This is so because a binary number, no matter how long, can be broken up into 3-bit groupings (from octal) or 4-bit groupings (from hexadecimal). You can simply add zero to the left side of a binary number if the total number of bits is not divisible by 3 or 4.

Example: Octal-Binary Conversion

537_8 to binary

$$\begin{array}{c} 5 \quad 3 \quad 7 \\ \hline 101 \quad 010 \quad 111 \end{array}$$

Answer: 101010111_2

111001100_2 to octal

$$\begin{array}{c} 111 \quad 001 \quad 100 \\ \hline 7 \quad 1 \quad 4 \end{array}$$

Answer: 714_8

Example: Hex-Binary Conversion

The **binary coded decimal** (BCD) system is a coding system in which each decimal digit is coded separately as a 4-bit

$3E9_{16}$ to binary

$$\begin{array}{c} 3 \quad E \quad 9 \\ \hline 0011 \quad 1110 \quad 1001 \end{array}$$

Answer: $0011 \ 1110 \ 1001_2$

$1001 \ 1111 \ 1010 \ 0111_2$ to hex

$$\begin{array}{c} 1001 \quad 1111 \quad 1010 \quad 0111 \\ \hline 9 \quad F \quad A \quad 7 \end{array}$$

Answer: $9FA7_{16}$

binary number. For example, the decimal number 15 in BCD is $0001 \ 0101$. BCD is commonly used when outputting to decimal (0-9) displays, such as those found in digital clocks and multimeters.

5.3.1.1 Binary Mathematics

Addition of binary numbers follows the following rules:

$$0 + 0 = 0$$

$$0 + 1 = 1 + 0 = 1$$

$$1 + 1 = 1 \quad \text{i.e., } 0 + \text{carry } 1$$

$$1 + 1 + 1 = 11 \quad \text{i.e., } 1 + \text{carry } 1$$

Example: In decimal system, the addition of 14 and 19 gives 33. In binary system, this addition becomes:

Augend $0 \ 1 \ 1 \ 1 \ 0$

Addend $1 \ 0 \ 0 \ 1 \ 1$

Sum $1 \ 0 \ 0 \ 0 \ 1$

When adding binary numbers A and B to give C , i.e., $A + B = C$, A is termed the *augend*, B the *addend*, and C the *sum*.

Subtraction of binary numbers follows the following rules:

$$0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = 10 - 1 + \text{borrow} = 1 + \text{borrow}$$

When evaluating $(0 - 1)$, a “1” is borrowed from the next column on the left containing a “1”.

Example: In decimal system, the subtraction of 14 from 27 gives 13. In binary system, this subtraction becomes:

Minuend	1 1 0 1 1
Subtrahend	0 1 1 1 0
Difference	0 1 1 0 1

When subtracting binary numbers A and B to give C , i.e., $A - B = C$, A is termed the *minuend*, B the *subtrahend*, and C the *difference*.

The numbers used so far are referred to as *unsigned*, because the number itself contains no indication whether it is negative or positive. A binary number is said to be signed when the most significant bit is used to indicate the sign of the number, a “0” being used for positive and a “1” for negative number. Another useful way of representing negative numbers is to use the *2s complement* method. The 2s complement of a binary number is obtained from its 1s complement. The 1s complement of a binary number is obtained by changing all the “1”s in the unsigned number into “0”s and “0”s into “1”s. Then, the 2s complement is obtained by adding 1 to the 1s complement. Then, to indicate that it is negative, we sign it with a “1” (if it is positive we sign the number with a “0”). Consider the representation of the decimal number -3 as a binary signed 2s complement number. We first write the binary number for the unsigned 3 as 0011, then obtain the 1s complement of 1100, and then add 1 to give the 2s complement of 1101, and finally sign it with a 1 to indicate it is negative. The result is thus 11101.

Example: In decimal system, subtraction of 43 from 75 gives 14. This subtraction can be carried on as adding the signed positive +75 to the signed negative -43 . In binary system, the signed positive number of 57 is 0011 1001. The signed 2s complement of -43 is obtained as:

Unsigned binary number for 43	010 1011
1s complement	101 0100
Add 1	1
Unsigned 2s complement	101 0101
Signed 2s complement	1101 0101

The, the two numbers can be added as:

Signed Augend	0011 1001
Signed Addend	1101 0101
Unsigned 2s complement	0000 1110 + carry 1

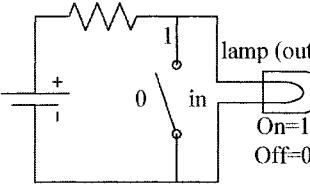
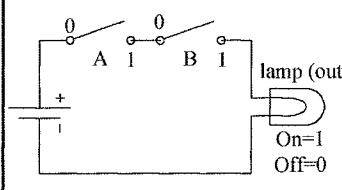
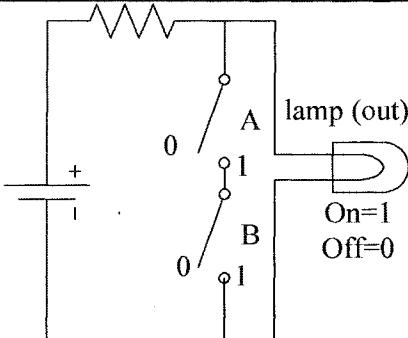
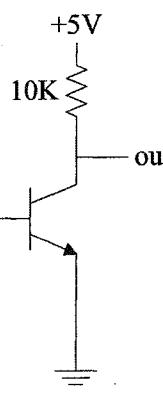
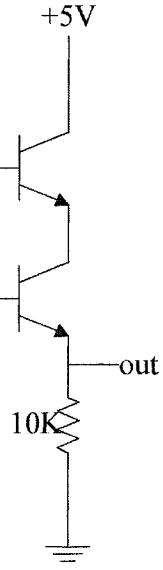
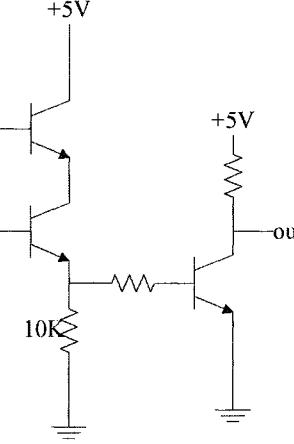
The “carry 1” is ignored. The result is hence 0000 1110, and since the MSB is 0 the result is positive. It is the decimal number +14.

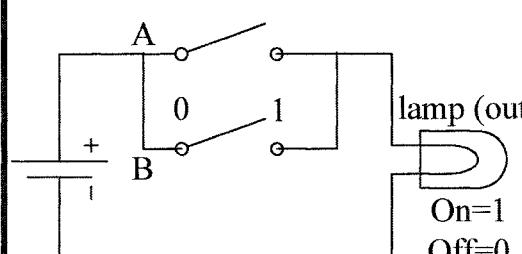
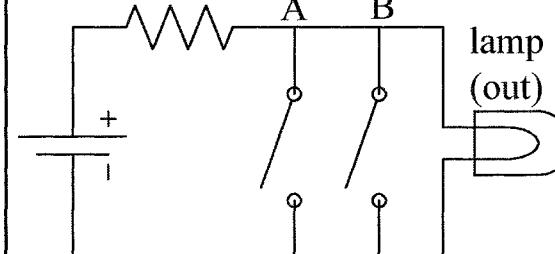
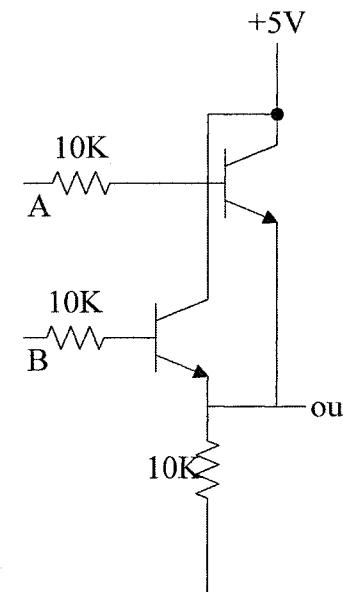
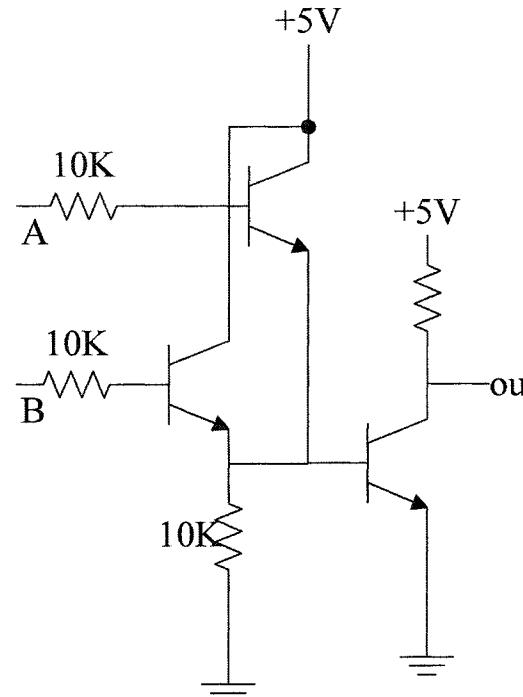
5.3.1.2 Parity Method for Error Detection

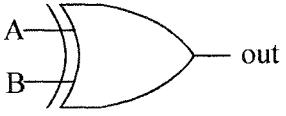
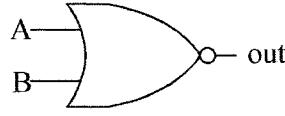
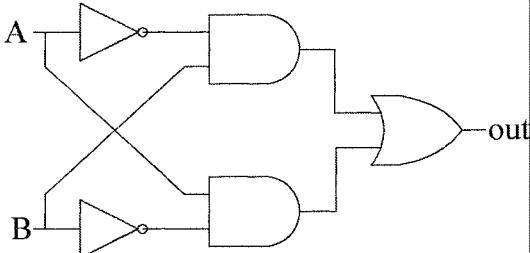
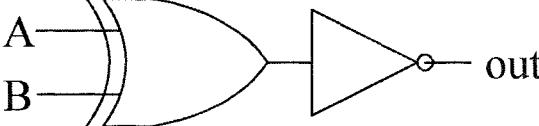
Transferring digital data from one location to another can result in transmission errors, due to electrical noise in the transmission process. Sometimes, a noise pulse may be large enough at some point to alter the logic level of the signal. For example, the sequence 1001 may be transmitted and received as though 1101. In order to detect such errors a *parity bit* is often used. The parity bit is an extra 0 or 1 bit attached to a code group at transmission. In the *even-parity* method, the value of the bit is chosen so that the total number of 1’s in the code group, including the parity bit, is an even number. For example in transmitting 1001 the parity bit used would be 0 to give 01001 and so an even number of 1’s. In transmitting 1101 the parity bit would be 1 to give 11101 and so an even number of 1’s. With *odd parity*, the parity bit is chosen so that the total number of 1’s, including the parity bit, is odd. Thus, if at the receiver the number of 1’s in a code group does not give the required parity, the receiver will know that there is an error and can request the code group be retransmitted.

5.3.2 Logic Gates

Logic gates are basic building blocks for digital electronic circuits. Some fundamental logic gates are represented in figure below.

Gate	INVERT (NOT)	AND	NAND																																				
Truth Table	<table border="1"> <thead> <tr> <th>in</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	in	out	0	1	1	0	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	out	0	0	0	0	1	0	1	0	0	1	1	1	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	out	0	0	1	0	1	1	1	0	1	1	1	0
in	out																																						
0	1																																						
1	0																																						
A	B	out																																					
0	0	0																																					
0	1	0																																					
1	0	0																																					
1	1	1																																					
A	B	out																																					
0	0	1																																					
0	1	1																																					
1	0	1																																					
1	1	0																																					
Switch Analogy																																							
Transistor Analogy																																							
Description	A NOT gate or inverter outputs a logic level that's the opposite (complement) of the input logic level.	The output of an AND gate is HIGH only when both inputs are HIGH	Combines the NOT function with an AND gate; output only goes LOW when both inputs are HIGH.																																				

Gate	OR 	NOR 																														
Truth Table	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	out	0	0	0	0	1	1	1	0	1	1	1	1	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	out	0	0	1	0	1	0	1	0	0	1	1	0
A	B	out																														
0	0	0																														
0	1	1																														
1	0	1																														
1	1	1																														
A	B	out																														
0	0	1																														
0	1	0																														
1	0	0																														
1	1	0																														
Switch Analogy																																
Transistor Analogy																																
Description	The output of an OR gate will go HIGH if one or both inputs goes HIGH. The outputs only goes LOW when both inputs are LOW	Combines the NOT function with an OR gate: output goes LOW if one or both inputs are LOW, output goes HIGH when both inputs are LOW.																														

Gate	XOR 	NOR 																														
Truth Table	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	out	0	0	0	0	1	1	1	0	1	1	1	0	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>out</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	out	0	0	1	0	1	0	1	0	0	1	1	0
A	B	out																														
0	0	0																														
0	1	1																														
1	0	1																														
1	1	0																														
A	B	out																														
0	0	1																														
0	1	0																														
1	0	0																														
1	1	0																														
Equivalent Circuit																																
Description	The output of an XOR gate goes HIGH if the inputs are different from each other. XOR gates only come with two inputs.	Combines the NOT function with an OR gate: output goes HIGH if the inputs are the same.																														

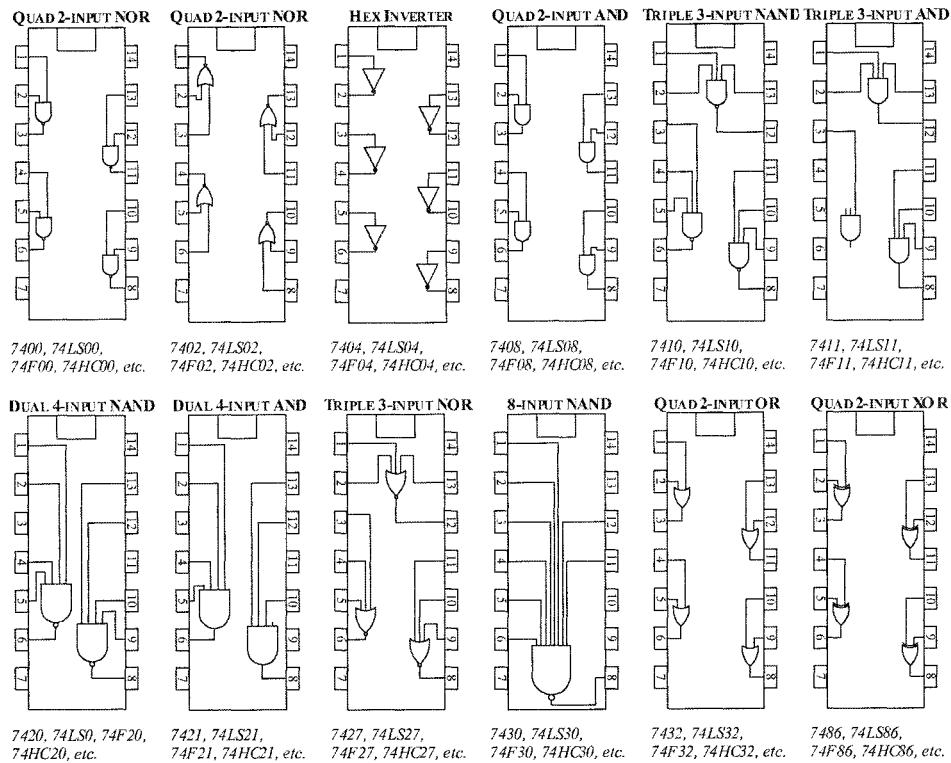
5.3.2.1 Digital Logic Gate ICs

There are different technologies for fabrication of digital logic gates. The two most popular technologies include TTL (Transistor-Transistor Logic) and CMOS (Complementary MOSFET) logic. TTL incorporates bipolar transistors into its design, while CMOS incorporates MOSFETs. Both technologies perform similar basic functions, but certain characteristics (power consumption, speed, output drive capacity, etc.) differ, as shown in Table below.

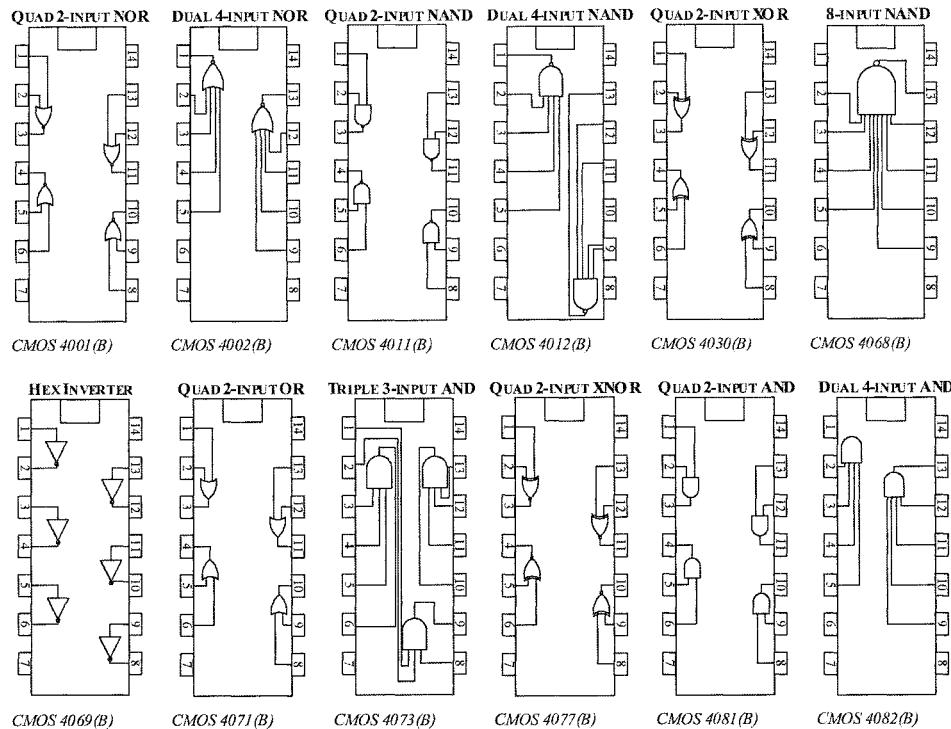
		TTL		CMOS	
Supply Voltage		4.75-5.25V		5-15V	
Max. Supply Current		-100 mA		-0.02 mA	
		Input	Output	Input	Output
0 State	Voltage	0.8V	0.5V	1.5 V	0.05 V
	Current	-0.4 mA	8 mA	-0.0001 mA	0.5 mA
1 State	Voltage	2.0 V	2.0 V	3.5 V	4.95 V
	Current	0.02 mA	0.4 mA	0.0001 mA	-0.2 mA
Max. Operating Frequency		33 MHz		10 MHz	
Active Power Consumption		8 mW		0.1 mW	

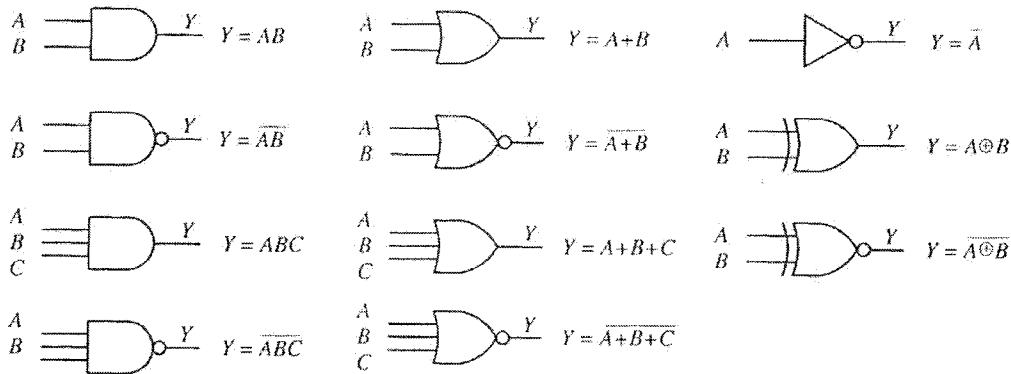
A logic IC, be it TTL or CMOS, typically houses more than one logic gate. Each of the gates within the IC shares a common supply voltage that is implemented via two supply pins, a positive supply pin ($+V_{CC}$ or $+V_{DD}$) and a ground pin (GND). The vast majority of TTL and many CMOS ICs are designed to run off a +5V supply. Some popular 7400 TTL and 4000B CMOS series are presented in figure below.

7400 Series



4000(B) Series





5.3.3 Combinational Logic

Combinational logic involves combining logic gates together to form circuits capable of enacting more useful, complex functions. To make designing combinational circuits easier, a special symbolic language called *Boolean Algebra* is used, which only works with “true” (1) and “false” (0) values for variables and statements. Boolean expressions for the logic gates are shown in figure above.

Similar to the conventional algebra, Boolean algebra also has a set of logical identities that are used to simplify the Boolean expressions and hence make circuits simpler and more compact. These identities can be summarized in the following list:

- 1) $A + B = B + A$
- 2) $AB = BA$
- 3) $A + (B + C) = (A + B) + C$
- 4) $A(BC) = (AB)C$
- 5) $A(B + C) = AB + AC$
- 6) $(A + B)(C + D) = AC + AD + BC + BD$
- 7) $\overline{1} = 0$
- 8) $\overline{0} = 1$
- 9) $A \cdot 0 = 0$
- 10) $A \cdot 1 = A$
- 11) $A + 0 = A$
- 12) $A + 1 = 1$
- 13) $A + A = A$
- 14) $AA = A$
- 15) $\overline{\overline{A}} = A$
- 16) $A + \overline{A} = 1$
- 17) $\overline{A}\overline{A} = 0$
- 18) $\overline{A + B} = \overline{A}\overline{B}$
- 19) $\overline{AB} = \overline{A} + \overline{B}$
- 20) $A + \overline{A}B = A + B$
- 21) $\overline{A} + AB = \overline{A} + B$
- 22) $A \oplus B = \overline{AB} + A\overline{B} = (A + B)(\overline{AB})$
- 23) $\overline{A \oplus B} = AB + \overline{AB}$

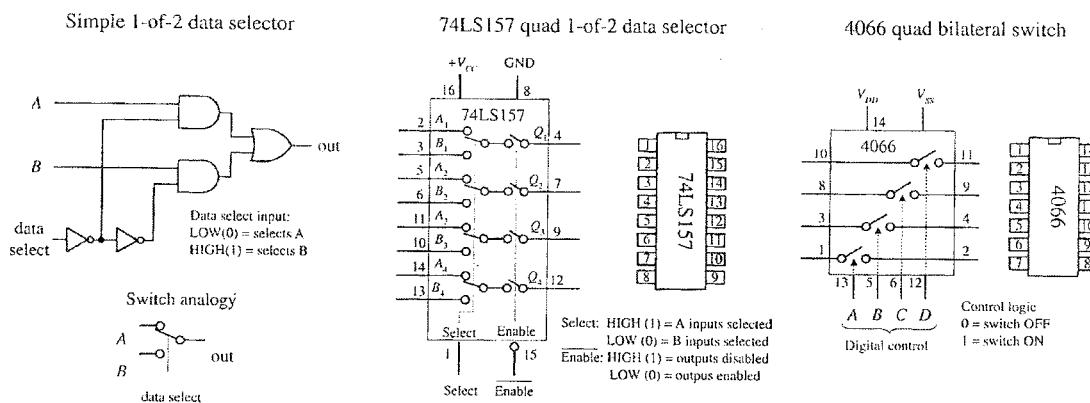
5.3.3.1 Combinational Devices

A wide variety of tasks can be performed by combining basic logic gates. These functions are usually carried out by different IC chips that contain all the necessary logic. Some of the most popular applications are mentioned in sequel.

5.3.3.1.1 Multiplexers (Data selectors) and Bilateral Switches

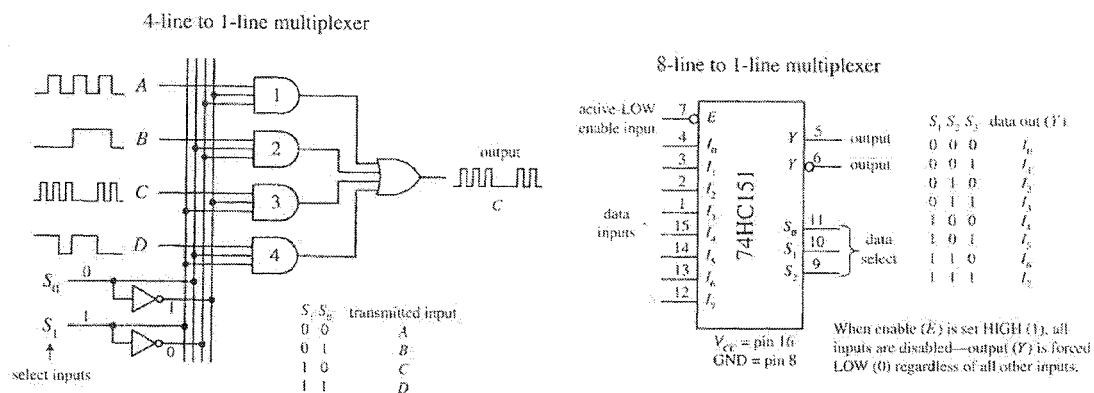
Multiplexers or data selectors act as digitally controlled switches. (The term *data selector* appears to be the accepted term when the device is designed to act like an SPDT switch, while the term *multiplexer* is used when the throw count of the "switch" exceeds two, e.g., SP8T.) A simple 1-of-2 data selector built from logic gates is shown in figure below. The data select input of this circuit acts to control which input (*A* or *B*) gets passed to the output: When data select is high, input *A* passes while *B* is blocked. When data select is low, input *B* is passed while *A* is blocked.

There are a number of different types of data selectors that come in IC form. For example, the 74LS157 quad 1-of-2 data selector IC, shown in figure above, acts like an electrically controlled quad SPDT switch. When its select input is set high (1), inputs *A*₁, *A*₂, *A*₃, and *A*₄ are allowed to pass to outputs *Q*₁, *Q*₂, *Q*₃, and *Q*₄. When its select input is low (0), inputs *B*₁, *B*₂, *B*₃, and *B*₄ are allowed to pass to outputs *Q*₁, *Q*₂, *Q*₃, and *Q*₄. Either of these two conditions, however, ultimately depends on the state of the enable input. When the enable input is low, all data input signals are allowed to pass to the output; however, if the enable is high, the signals are not allowed to pass. This type of enable control is referred to as *active-low* enable, since the active function (passing the data to the output) occurs only with a low-level input voltage.



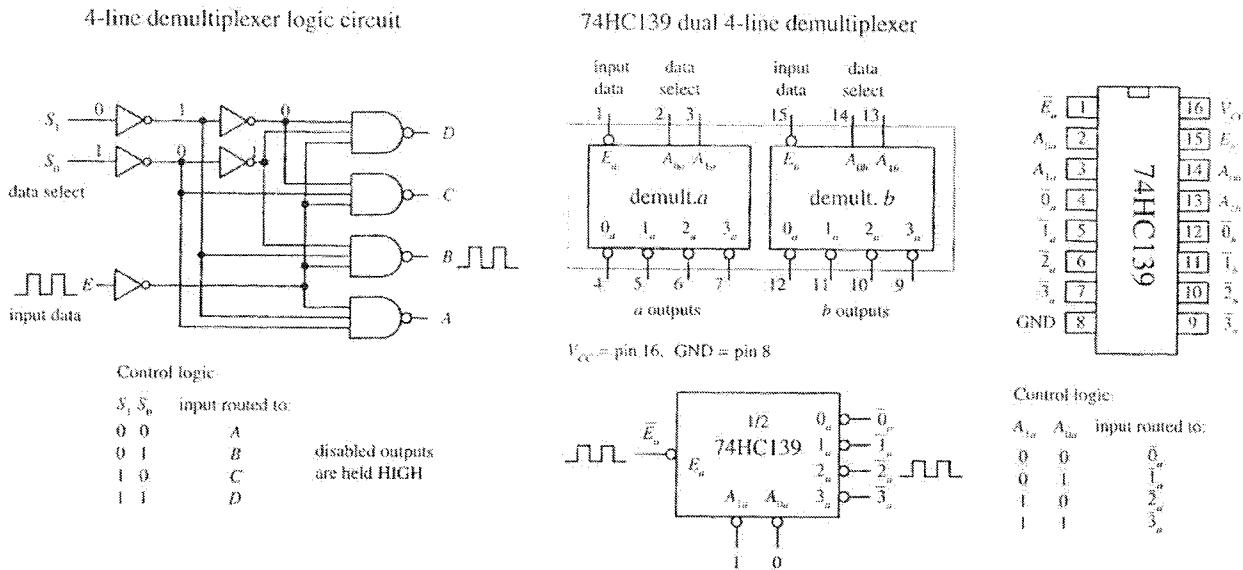
Another useful device shown to the far right in figure above is called *bilateral switch*. Unlike the multiplexer, this device merely acts as a digitally controlled quad SPST switch or quad transmission gate. Using a digital control input, you select which switches are ON and which switches are OFF. To turn on a given switch, apply a high level to the corresponding switch select input; otherwise, keep the select input low.

Figure below shows a 4-line-to-1-line multiplexer built with logic gates. This circuit resembles the 2-of-1 data selector shown in figure above, but requires an additional select input to provide four address combinations.



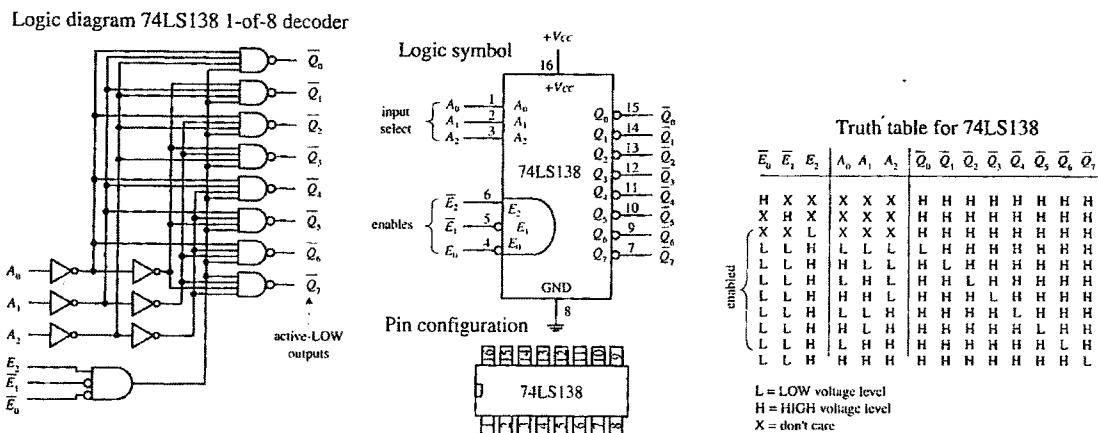
5.3.3.1.2 Demultiplexers (Data Distributors) and Decoders

A demultiplexer (or data distributor) is the opposite of a multiplexer. It takes a single data input and routes it to one of several possible outputs. A simple four-line demultiplexer built from logic gates is shown in figure below. To select the output (A , B , C , or D) to which you want to send the input signal (applied at E), you apply logic levels to the data select inputs (S_0 , S_1), as shown in the truth table. Notice that the unselected outputs assume a high level while the selected output varies with the input signal. An IC that contains two functionally separate four-line demultiplexers is the 74HC139, shown on the right side of the above figure. If you need more outputs, check out the 75xx154 16-line demultiplexer. This IC uses four data select inputs to choose from 1 of 16 possible outputs.



5.3.3.1.3 Code Converters (Decoders)

A decoder is somewhat like a demultiplexer, but it does not route input data to a specific output via data select inputs. Instead, it simply uses the data select inputs to choose which output (or outputs) among many is to be made high or low. The number of address inputs, the number of outputs, and the active state of the selected output vary from decoder to decoder. As an example, the 74LS138 1-of-8 decoder shown in figure below uses a 3-bit address input to select which of 8 outputs will be made low; all other outputs are held high. Like the demultiplexer in the previous figure, this decoder has active-low outputs. This means that when an active-low output is selected, it is forced to a low logic state; otherwise, it is held high. By placing a load (e.g. warning LED) between +VCC and an active-low output, you can sink current through the load and into the active-low output when the output is selected. There are other types of IC's that provide active-high

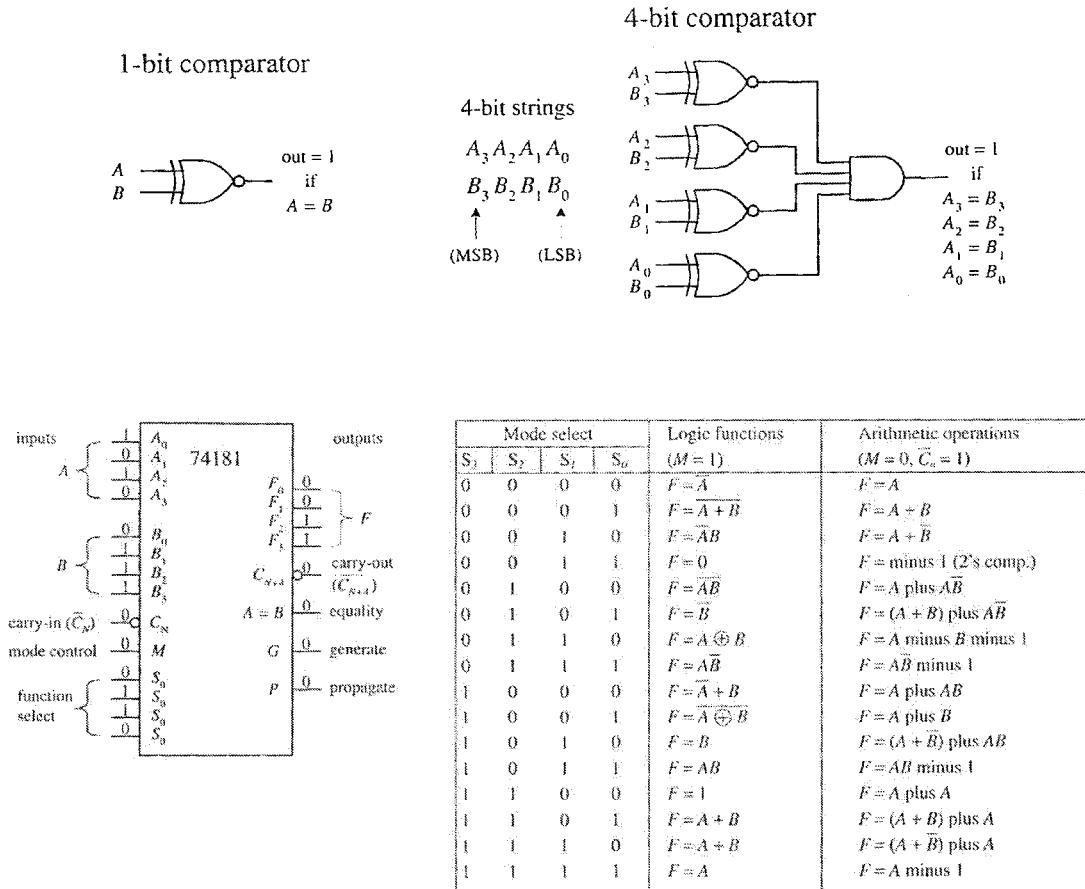


outputs, and by placing a load between an active-high output and ground the current can be sourced from the output and drained through the load when the output is selected.

Other common decoders include the 7442 BCD-to-DEC decoder, the 74154 1-of-16 (Hex) decoder, and the 7447 BCD-to-seven-segment decoder.

5.3.3.1.4 Arithmetic Logic Units (ALUs)

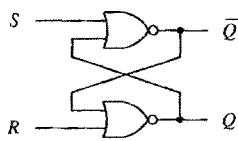
An arithmetic/logic unit (ALU) is a multipurpose IC capable of performing various arithmetic and logic operations. To choose a specific operation to be performed, a binary code is applied to the IC's mode select inputs. The 74181 chip, shown in figure below is a 4-bit ALU that provides 16 arithmetic and 16 logic operations.



5.3.3.1.5 Digital Comparators

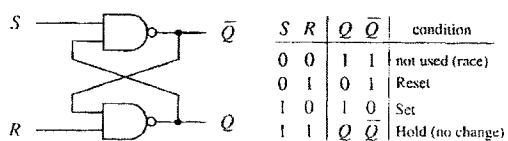
A digital comparator is a circuit that compares two binary words to determine if they are exactly equal. The two words are compared bit by bit and a "1" output is given if the words are equal. To compare the equality of two bits, an XOR gate can be used as shown in figure above. A 4-bit comparator, also shown in figure above, is basically four 1-bit comparators in one. When all individual digits of each number are equal, all XOR gates give a high, which in turn enables the AND gate making the output high.

Cross-NOR SR flip-flop

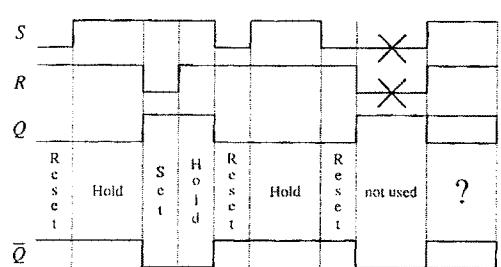
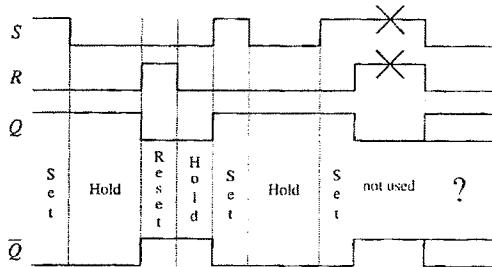


S	R	Q	\bar{Q}	condition
0	0	Q	\bar{Q}	Hold (no change)
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	not used (race)

Cross-NAND SR flip-flop



S	R	Q	\bar{Q}	condition
0	0	1	1	not used (race)
0	1	0	1	Reset
1	0	1	0	Set
1	1	Q	\bar{Q}	Hold (no change)



Going from $S=1, R=1$ back to the hold condition ($S=0, R=0$) leads to an unpredictable output. Therefore $S=1, R=1$ isn't used.

Digital comparator IC's can generally not only determine if two words are equal but which one is greater than the other. For example, the 74HC85 4-bit *magnitude comparator*, shown in figure below, compares two 4-bit words A and B , giving a "1" output from pin 5 if A is greater than B , a "1" output from pin 6 if A equals B , and a "1" output from pin 7 if A is less than B .

5.3.4 Sequential Logic

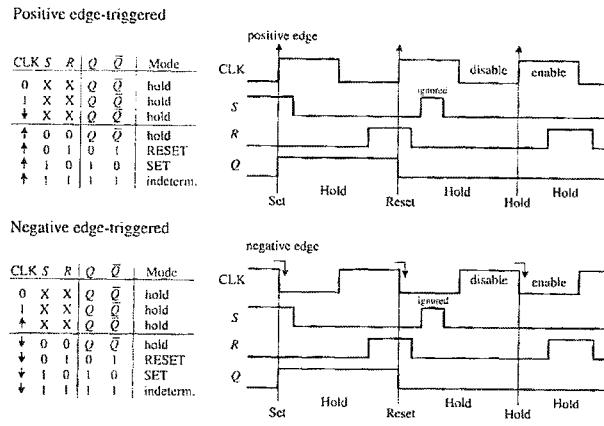
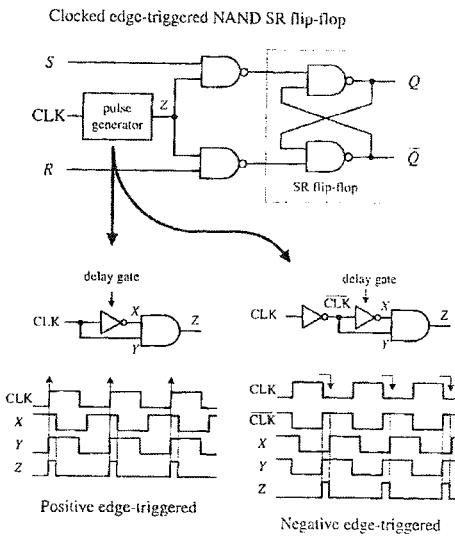
The logic circuits introduced so far were all examples of combinational logic where the output is determined by the combination of the input variables at a particular instant of time. Where a system requires an output that depends on earlier values of the inputs and outputs, a *sequential logic* system is required. Thus, the main difference between the combinational and sequential circuits is the existence of some sort of memory in the latter, where the data bits are stored and retrieved. In a sequential circuit, a series of steps are required to enable the storage device, load a group of data bits (in parallel or serial), perform the operation, and restore the data in the memory. Therefore, these circuits usually require a clock generator, which generates a series of high and low voltages that can set bits into action. The clock also acts as a time base on which all sequential actions can be referenced. Some of the popular sequential circuits are addressed below.

5.3.4.1 SR Flip-Flops

An elementary data storage circuit is called *SR (set-reset) flip-flop*, also referred to as a *transparent latch*. Two basic types of SR flip-flops are *cross-NOR* and *cross-AND* circuits both shown above.

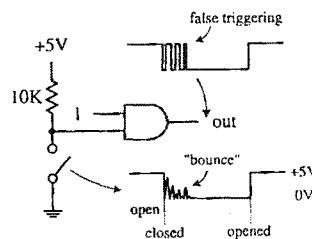
A fine application of SR flip-flops is in the switch debouncer circuit. Consider the left switch/pullup resistor circuit in figure below to drive an AND gate's input high or low (the other input is fixed high). When the switch is open, the AND gate should receive a high. When the switch is closed, the gate should receive a low. Practically, this logic may not occur, due to the switch bounce. When the switch is closed, the metal contacts bounce a number of times before coming to rest due to inherent spring-like characteristics of the contacts. Though the bouncing typically lasts no more than 50 ms, the results can lead to unwanted false triggering, as shown in the left circuit below. A simple way to eliminate the switch bounce is to use the switch debouncer circuit, shown at right in figure below. This circuit simply uses an SR flip-flop to store the initial switch contact voltage while ignoring all trailing bounces. In this circuit, when the switch is thrown from the B to A position, the flip-flop is set. As the switch bounces alternately high and low, the Q output remains high because when the switch contact bounces away from A , the S input receives a low (R is low too), but that is just a hold condition; the output stays the same. The same debouncing feature occurs when the switch is thrown from position A to B .

It is often necessary for set-and-reset operations to occur at particular times. With an un-clocked or *asynchronous* system the outputs of logic gates can change state any time when any one or more of the inputs change. With a clocked or

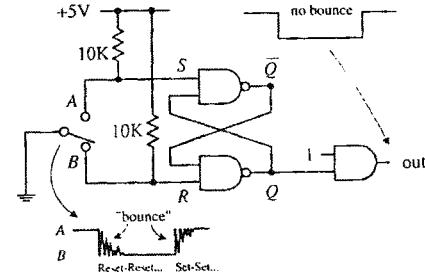


synchronous system the exact times at which any output can change state is determined by a signal termed the *clock signal*. There are different methods to trigger the flip-flop by a clock, such as *level triggering*, *edge triggering*, or *pulse triggering*. As an example, an edge-triggered RS flip-flop, samples the inputs only during either positive (positive edge-triggered) or negative (negative edge-triggered) clock edge. Any changes that occur before or after the clock edge are ignored, i.e., the flip-flop will be placed in hold mode. Figure above shows both positive and negative edge-triggered flip-flops. 74LS279A is a quad SR latch that is commonly used in switch debouncers.

Example of switch bounce



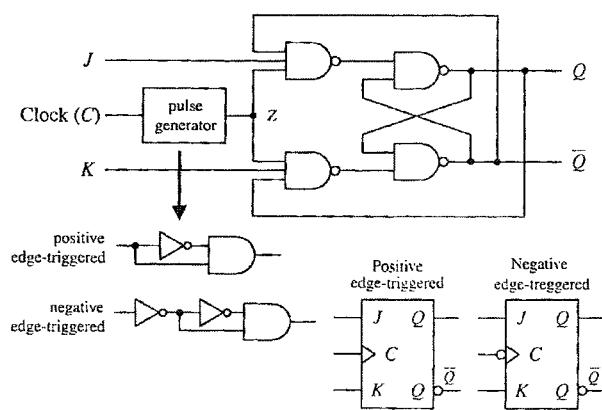
Switch debouncer circuit



5.3.4.2 JK Flip-Flops

For many applications the indeterminate state that occurs with SR flip-flop when $S=1$ and $R=1$ is not acceptable, and another form of flip-flop is used. The JK flip-flop has a toggle mode when $J=1$ and $K=1$, see figure below. Toggle means that Q and \bar{Q} outputs switch to their opposite states at each active clock edge.

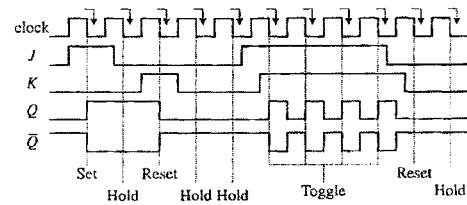
Edge-triggered JK flip-flops

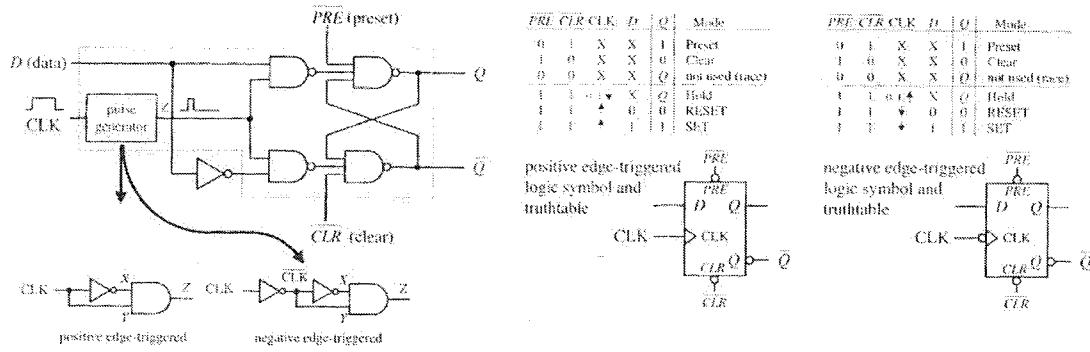


Positive edge-triggered Negative edge-triggered

C	J	K	Q	\bar{Q}	Mode	C	J	K	Q	\bar{Q}	Mode
0	X	X	Q	\bar{Q}	hold	0	X	X	Q	\bar{Q}	hold
1	X	X	Q	\bar{Q}	hold	1	X	X	Q	\bar{Q}	hold
\downarrow	X	X	Q	\bar{Q}	hold	\uparrow	X	X	Q	\bar{Q}	hold
\uparrow	0	0	Q	\bar{Q}	hold	\downarrow	0	0	Q	\bar{Q}	hold
\uparrow	0	1	0	1	Reset	\downarrow	0	1	0	0	Reset
\uparrow	1	0	0	1	Set	\downarrow	1	0	0	0	Set
\uparrow	1	1	\bar{Q}	Q	Toggle	\downarrow	1	1	\bar{Q}	Q	Toggle

Timing diagram for negative-edge triggered JK flip-flop



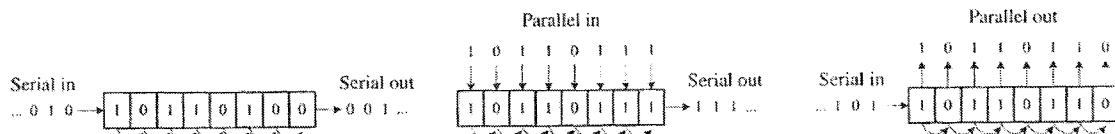


5.3.4.3 D Flip-Flops

The Data or D flip-flop is basically a clocked SR or JK flip-flop with the D input being connected directly to the S or J inputs and via a NOT gate to the R or K inputs; in the symbol for D flip-flop this joined R and K inputs is labeled D. Figure above shows the operation mechanism for an edge-triggered D flip-flop (e.g., 74HC74). With an edge-triggered D flip-flop, transitions in Q occur only at the edge.

5.3.4.4 Registers

Data words traveling through a digital system must frequently be held temporarily, copied, and bit-shifted to the left or right. A device that can be used for such purposes is the *shift register*. A shift register is made of a row of flip-flops connected so that digital data can be shifted down the row either in the left or right direction. Most shift registers can handle parallel movement of data bits as well as serial movement, and can also be used to convert from parallel to serial or from serial to parallel. Figure below shows the three types of shift register arrangements: serial in / serial-out, parallel-in / serial-out, and serial-in / parallel-out.



A universal shift register IC is the 74194 4-bit chip, which can accept either serial or parallel inputs, provide serial or parallel outputs, and shift left or right based on input signals applied to select control pins.

5.3.4.5 The 555 Timer IC

In digital electronics, square-wave oscillators, called *clocks* or *timers*, are used to drive bits of information through logic gates and flip-flops at a rate of speed determined by the frequency of the clock. The 555 timer IC is a useful precision timer that can act as either a timer or an oscillator. In timer mode, known as *monostable mode*, the 555 simply acts as a "one-shot" timer; when a trigger voltage is applied to its trigger lead, the chip's output goes from low to high for a duration set by an external RC circuit. In oscillator mode, better known as *astable mode*, the 555 acts as a rectangular-wave generator whose output waveform (low duration, high duration, frequency, etc.) can be adjusted by means of two external RC charge/discharge circuits. The 555 timer IC is easy to use (requires few components and calculations) and inexpensive and can be used in a number of applications. For example, with the aid of a 555, it is possible to create digital clock waveform generators, LED and lamp flasher circuits, tone-generator circuits (sirens, metronomes, etc.), one-shot timer circuits, bounce-free switches, triangular-waveform generators, frequency dividers, etc.

5.3.4.5.1 Basic Astable Operation

When a 555 is set up in astable mode, it has no stable states, i.e., the output jumps back and forth. The time duration V_{out} remains low (around 0.1V) is set by the R_1C time constant and the $\frac{1}{3}V_{CC}$ and $\frac{2}{3}V_{CC}$ levels. The time duration V_{out} remains high (around $V_{CC} - 1.5V$) is set by the $(R_1+R_2)C$ time constant and the two voltage levels, as shown in figure right. The following two practical expressions can be obtained:

$$t_{low} = 0.693 R_2 C$$

$$t_{high} = 0.693(R_1 + R_2)C$$

The duty cycle (the fraction of the time the output is high) is given by

$$\text{Duty Cycle} = \frac{t_{high}}{t_{high} + t_{low}}$$

The frequency of the output waveform is

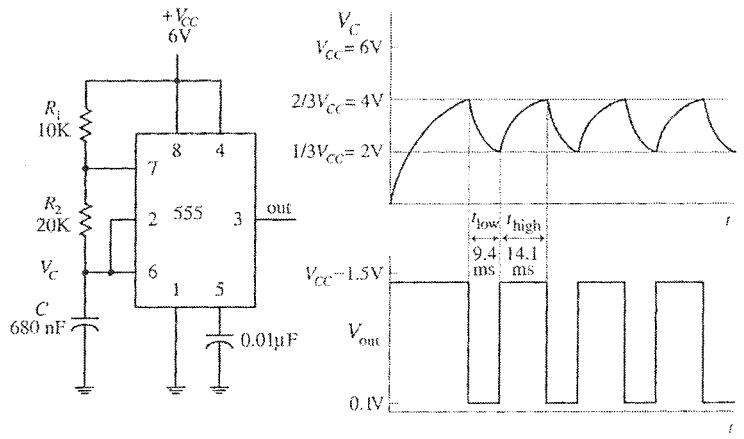
$$f = \frac{1}{t_{high} + t_{low}}$$

For reliable operation, the resistors should be between approximately 10 k Ω and 14 M Ω , and the timing capacitor should be from around 100 pF to 1000 μ F. The graph on right would give you a general idea of how the frequency responds to the component values.

There is a slight problem with the above circuit. You cannot get a duty cycle that is below 0.5 (or 50 percent). In other words, you cannot make t_{high} shorter than t_{low} . For this to occur, the R_2C network (used to generate t_{low}) would have to be larger than $(R_1 + R_2)C$ network (used to generate t_{high}). Simple arithmetic tells us that this is impossible. How do you remedy this situation? You attach a diode across R_2 as shown in the figure right. With the diode in place, as the capacitor is charging (generating t_{high}), the preceding time constant $(R_1 + R_2)C$ is reduced to R_1C because the charging current is diverted around R_2 through the diode. With the diode in place, the high time become

$$t_{high} = 0.693 R_1 C$$

To generate a duty cycle less than 0.5 (50 percent), simply make R_1 less than R_2 .

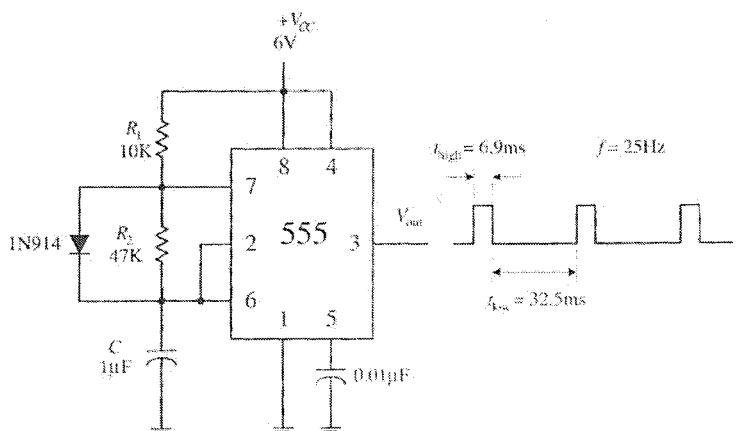
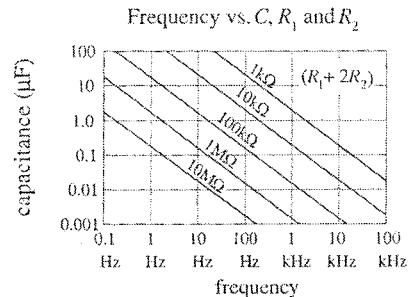


$$t_{low} = 0.693(20\text{K})(680\text{nF}) = 9.6\text{ms}$$

$$t_{high} = 0.693(10\text{K} + 20\text{K})(680\text{nF}) = 14.1\text{ms}$$

$$f = \frac{1}{9.6\text{ms} + 14.1\text{ms}} = 42\text{Hz}$$

$$\text{duty cycle} = \frac{14.1\text{ms}}{14.1\text{ms} + 9.6\text{ms}} = 0.6$$

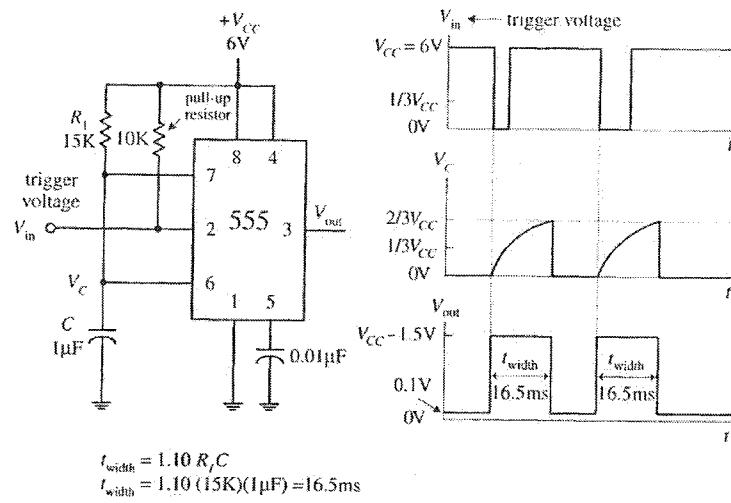


5.3.4.5.2 Basic Monostable Operation

The monostable circuit only has one stable state, i.e., the output rests at 0V (in reality, more like 0.1V) until a negative-going trigger pulse is applied to the trigger lead, pin 2. The negative-going pulse can be implemented by momentarily grounding pin 2, say, by using a pushbutton switch attached from pin 2 to ground. After the trigger pulse is applied, the output will go high (around $V_{CC} - 1.5V$) for the duration set by the R_1C network. The width of the high output pulse can be obtained as:

$$t_{width} = 1.10 R_1 C$$

For reliable operation, the timing resistor R_1 should be between around 10 k Ω and 14 M Ω , and the timing capacitor should be from around 100 pF to 1000 μ F.

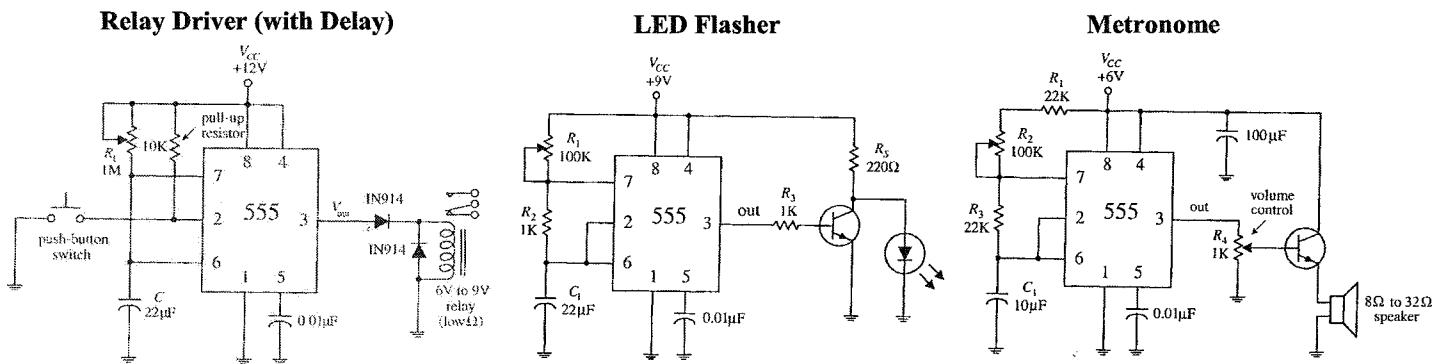


5.3.4.5.3 Practical Notes

555 ICs are available in both bipolar and CMOS types, which differ in terms of maximum output current, minimum supply voltage/current, minimum triggering current, and maximum switching speed. With the exception of maximum output current, the CMOS 555 surpasses the bipolar 555 in all regards. A CMOS 555 IC can be distinguished from a bipolar 555 by noting whether the part number contains a C somewhere within it (e.g., ICL7555, TLC555, LMC555, etc.). Note that there are hybrid versions of the 555 that incorporate the best features of both the bipolar and CMOS technologies. If you need more than one 555 timer per IC, check out the 556 (dual version) and 558 (quad version).

To avoid problems associated with false triggering, connect the 555's pin 5 to ground through a 0.01 μ F capacitor. Also, if the power supply lead becomes long or the timer does not seem to function for some unknown reasons, try attaching a 0.1 μ F or larger capacitor between pins 8 and 1.

Figure below shows three applications of using the 555 timer.



5.3.5 Practical Remarks

5.3.5.1 Powering Logic ICs

Most TTL and CMOS logic devices will work with $5V \pm 0.25V$ (5 percent) supplies. The battery supplies should be avoided when using certain TTL families such as 74xx, 74S, and 74F, which dissipate considerably more current than other series including CMOS 74HC. Nevertheless, Low-power, low-voltage 74LV, 74LVC, 74LVT, and 74BCT series, which require from 1.2 to 3.6 V with as low as $2.5 \mu\text{W}/\text{gate}$ power dissipation, are ideal for small battery-powered applications.

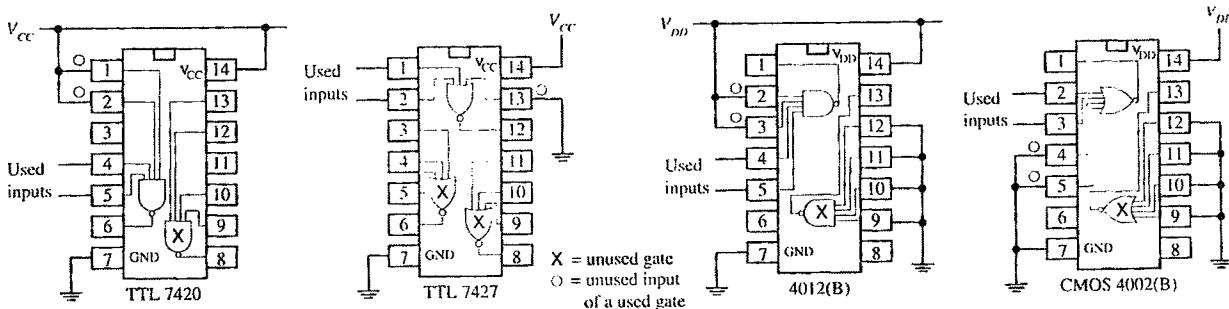
5.3.5.2 Power Supply Decoupling

When a TTL device makes a low-to-high or a high-to-low level transition, there might be an interval during which a drastic change in power supply current occurs, which results in a sharp, high-frequency current spike within the supply line. If a number of other devices are linked to the same supply, the unwanted spike can cause false triggering of these devices. The spike can also generate unwanted electromagnetic radiation. To avoid unwanted spikes within TTL systems, decoupling capacitors can be used. A decoupling capacitor, typically tantalum, from 0.01 to $1 \mu\text{F}$ ($> 5 \text{ V}$), is placed directly across the V_{cc} -to-ground pins of each IC in the system. The capacitors absorb the spikes and keep the V_{cc} level at each IC constant, thus reducing the likelihood of false triggering and generally electromagnetic radiation. Decoupling capacitors should be placed as close to the ICs as possible to keep current spikes local, instead of allowing them to propagate back toward the power supply. You can usually get by with using one decoupling capacitor for every 5 to 10 gates or one for every 5 counter or register ICs.

5.3.5.3 Unused Inputs

Unused inputs that affect the logical state of a chip should not be allowed to float. Instead, they should be tied high or low, as necessary (floating inputs are liable to pickup external electrical noise, which leads to erratic output behavior). For example, a four-input NAND TTL gate that only uses two inputs should have its two unused inputs held high to maintain proper logic operation. A three-input NOR gate that only uses two inputs should have its unused input held low to maintain proper logic operation. Likewise, the CLEAR and PRESET inputs of a flip-flop should be grounded or tied high, as appropriate.

If there are unused sections within an IC (e.g., unused logic gates within a multi-gate package), the inputs that link to these sections can be left unconnected for TTL but not for CMOS. When unused inputs are left unconnected in CMOS devices, the inputs may pick up unwanted charge and may reach a voltage level that causes output MOS transistors to conduct simultaneously, resulting in a large internal current spike from the supply (V_{DD}) to ground. The result can lead to excessive supply current drain and IC damage. To avoid this fate, inputs of unused sections of a CMOS IC should be grounded. Figure below illustrates what to do with unused inputs for TTL and CMOS NAND and NOR ICs. As a last note of caution, never drive CMOS inputs when the IC's supply voltage is removed. Doing so can damage the IC's input protection diodes.

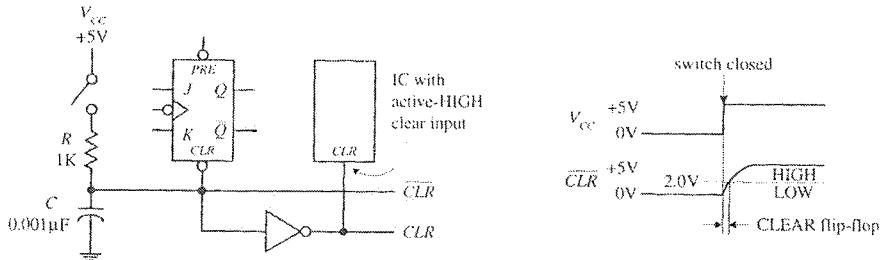


Connect unused inputs of a used NAND gate HIGH to maintain proper logic function. Don't assume they'll be naturally HIGH for TTL. Connect unused input of a NOR gate LOW to maintain proper logic function. Inputs of unused TTL gates can be left unconnected.

Connect unused inputs of a used NAND gate HIGH to maintain proper logic function. Connect unused inputs of a used NOR gate LOW to maintain proper logic function. Inputs of unused CMOS gates should be grounded.

5.3.5.4 Automatic Power-Up Clear (Reset) Circuits

In sequential circuits it is usually a good idea to clear (reset) devices when power is first applied. This ensures that devices, such as flip-flops and other sequential ICs, do not start out in an unknown mode (e.g., counter IC does not start counting at, say, 1101 instead of 0000). One technique used to provide automatic power-up clearing is shown in Figure below.



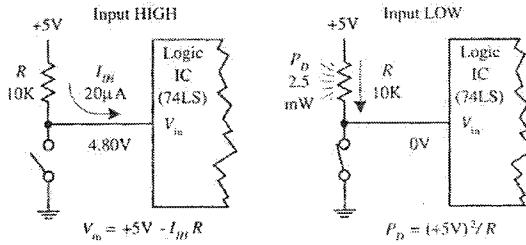
Assume that one of the devices in a circuit has a JK flip-flop that needs clearing during power-up. In order to clear the flip-flop and then quickly return it to synchronous operations, you would like to apply a low (0) voltage to its active-low clear input; afterwards, you would like the voltage to go high (at least above 2.0V for a 74LS76 JK flip-flop). A simple way to implement this function is to use an RC network like the one shown in the previous figure. When the power is off (switch open), the capacitor is uncharged (0V). This means that the CLR line is low (0V). Once the power is turned on (switch closed), the capacitor begins charging up toward V_{CC} (+5V). However, until the capacitor's voltage reaches 2.0V, the CLR line is considered low to the active-low clear input. After a duration of $t = RC$, the capacitor's voltage will reach 63 percent of V_{CC}, or 3.15V; after a duration of $t = 5RC$, its voltage will be nearly equal to +5V. Since the 74LS76's CLR input requires at least 2.0V to be placed back into synchronous operations, you know that $t=RC$ is long enough. Thus, as a rough estimation, if you want the CLR line to remain low for 1 μ s after power-up, you must set $RC = 1 \mu$ s. Setting $R = 1 \text{ k}$ and $C = 0.001 \mu\text{F}$ does the job.

This automatic resetting scheme can be used within circuits that contain a number of reset table ICs. If an IC requires an active-high reset (not common), simply add an inverter and create an active-high clear line, as shown in the previous figure. Depending on the device being reset, the length of time that the clear line is at a low will be about 1 μ s. As more devices are placed on the clear line, the low time duration will decrease due to the additional charging paths. To prevent this from occurring, a larger capacitor can be used.

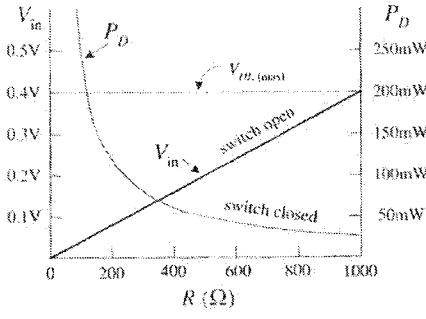
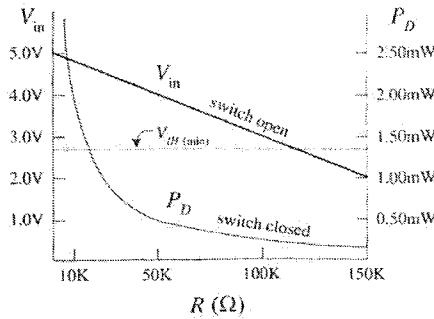
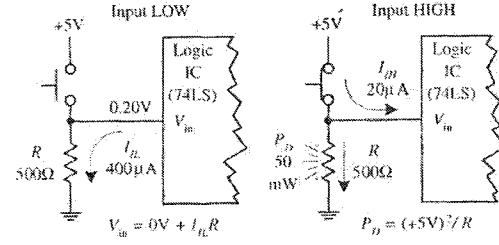
5.3.5.5 Pullup and Pulldown Resistors

Pullup or Pulldown resistors are used to keep an input high or low that would otherwise float if left unconnected. The next figure shows typical logic circuits with pullup (left) and Pulldown (right) resistors. It is important to have an idea about the size of these resistors. For example, the idea for a pullup resistor value is that it must be small enough so that the voltage drop across it does not weight down the input voltage below the minimum high threshold voltage ($V_{IH,min}$) of the IC. At the same time, you do not want to make it too small; otherwise, when you ground the pin, excessive current will be dissipated. In the next figure (left), a 10K pullup resistor is used to keep a 74LS device's input high. To make the input low, close the switch. To figure out if the resistor is large enough so as not to weigh down the input, use $V_{in} = +5V - RI_{IH}$, where I_{IH} is the current drawn into the IC during the high input state, when the switch is open. For a typical 74LS device, I_{IH} is around 20 μA . Thus, by applying the simple formula, you find that $V_{in} = 4.80 \text{ V}$, which is well above the $V_{IH,min}$ level for a 74LS device. Now, if you close the switch to force the input low, the power dissipated through the resistor ($P_D = V^2 / R$) will be $(5V)^2 / 10\text{k} = 25 \text{ mW}$. The graph shown in the figure (below left) provides V_{in} versus R and P_D versus R curves. As you can see, if R becomes too large, V_{in} drops below the $V_{IH,min}$ level, and the output will not go high as planned. As R gets smaller, the power dissipation rises quickly. To determine what value of R to use for a specific logic IC, you look up the $V_{IH,min}$ and $I_{IH,max}$ values within the data sheets and apply the simple formulas. In most applications, a 10K pullup resistor will work fine.

Using a pullup resistor to keep input normally HIGH



Using a pulldown resistor to keep input normally LOW



In some situations, a pulldown resistor is needed to keep a floating terminal low. Unlike a pullup resistor, the pulldown resistor must be smaller because the input low current I_{LL} (sourced by IC) is usually much larger than I_{HH} . Typically, a pulldown resistor is around 100 to 1 KΩ. A lower resistance ensures that V_{in} is low enough to be interpreted as a low by the logic input. To determine if V_{in} is low enough, use $V_{in} = 0V + I_{LL}R$. As an example, use a 74LS device with an $I_{LL} = 400 \mu A$ and a 500Ω pulldown resistor. When the switch is open, the input will be 0.20 V, well below the $V_{LL,max}$ level for the 74LS (0.8 V). When the switch is closed, the power dissipated by the resistor will be $(5 V)^2 / 500 \Omega = 50 \text{ mW}$. The graph shown in the previous figure (below right) provides $V_{LL,max}$ versus R and P_D versus R curves. As you can see by the curves, if R becomes too large, V_{in} surpasses $V_{LL,max}$ and the output will not be low as planned. As R gets small, the power dissipation rises rapidly. If you have to use a pulldown resistor / switch arrangement, be aware of the high power dissipation through the resistor when the switch is closed.

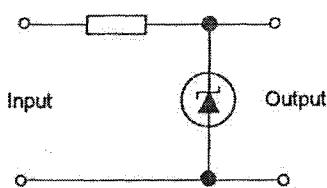
5.4 Signal Conditioning

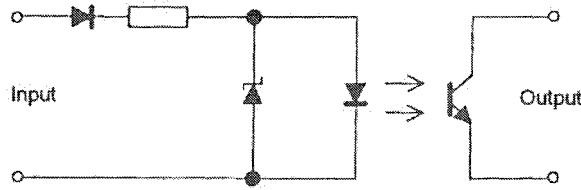
The output signal from the sensor of a measurement system and/or the input signal to an actuator have generally to be processed in some way to make them suitable for the next stage of the operation. The signal may be, for example, too small and have to be amplified, contain interference that has to be removed, be non-linear and require linearisation, be analogue and have to be made digital, be digital and have to be made analogue, be a resistance change and have to be made into a current change, be a voltage change and have to be made into a suitable size current change, etc. All these changes can be referred to as *signal conditioning*. For example, the output from a photoresistor is a small voltage, a few milli-Volts. A signal-conditioning module might then be used to convert this into a suitable size current signal, provide noise rejection, linearisation, etc. The following are some of the processes that can occur in conditioning a signal:

- 1) *Protection* to prevent damage to the next element, e.g. a microprocessor, as a result of high current or voltage. As a result, there can be a series of current-limiting resistors, fuses to break if the current is too high, polarity protection and voltage limitation circuits, etc.
- 2) Getting the *level* of the signal right. The signal from a sensor might be just a few milli-Volts. If the signal is to be fed into an analogue-to-digital converter for inputting to a microprocessor then it needs to be made much larger, Volts rather than milli-Volts. Operational amplifiers are widely used for amplification.
- 3) Getting the signal into the *right type of signal*. This can mean making the signal into a DC voltage or current. Thus, for example, the resistance change of a strain gauge has to be converted into a voltage change. This can be done by the use of a Wheatstone bridge and using the out-of-balance voltage. It can also mean making the signal digital or analogue.
- 4) Eliminating or reducing *noise*. For example, filters might be used to eliminate mains noise from a signal.
- 5) Signal *manipulation*, e.g. making it a linear function of some variable. The signals from some sensors, e.g. a flow meter, are non-linear and thus a signal conditioner might be used so that the signal fed on to the next element is linear.

5.4.1 Signal Protection

There are many situations where the connection of a sensor to the next unit, e.g. a microprocessor, can lead to the possibility of damage as a result of perhaps a high current or high voltage. A high current can be protected against by incorporating a series of resistors in the input line to limit the current to an acceptable level, and also a fuse to break if the current does exceed a safe level. High voltages, and wrong polarity, may be protected against by the use of a Zener or signal diode circuit as shown in figure below. Zener diodes behave like ordinary diodes up to some breakdown voltage when they become conducting. Hence, to allow a maximum voltage of 5 V but stop voltages above 5.1 V getting through, a Zener diode with a voltage rating of 5.1 V might be chosen. When the voltage rises to 5.1 V the Zener diode breaks down, and its resistance drops to a very low value. The result is that the voltage across the diode, and hence that outputted to the next circuit drops. Because the Zener diode is a diode with a low resistance for current in one direction through it and a high resistance for the opposite direction, it also provides protection against wrong polarity. It is connected with the correct polarity to give a high resistance across the output and so a high voltage drop. When the supply polarity is reversed, the diode has low resistance and so little voltage drop occurs across the output.





In some situations, it is desirable to completely isolate circuits and remove all electrical connections between them. This can be done using an optoisolator. This involves converting an electrical signal into an optical signal, and then passing it to a detector that then converts it back into an electrical signal. As an example, the input signal can be fed through an infrared LED. The infrared signal is then detected by a phototransistor. A protection circuit for a microprocessor is thus likely to be like that shown in figure above; to prevent the LED having the wrong polarity or too high an applied voltage, it is also likely to be protected by the Zener diode circuit shown in the previous figure. Further, if there is an alternating signal in the input a diode would be put in the input line to rectify it.

To protect circuits from excessive current flows, which are often caused by shorts or sudden power surges, fuses are used. A fuse contains a narrow strip of metal that is designed to melt when current flow exceeds its current rating, thereby interrupting power to the circuit. In practice, a fuse's current rating should be around 50 percent larger than the expected nominal current rating. The additional leeway allows for unexpected, slight variations in current and also provides compensation for fuses whose current ratings decrease as they age. Fuses usually come in *fast-action (quick-blow)* and *time-lag (slow-blow)* types. A fast-action fuse will turn off with just a brief surge in current, whereas a time-lag fuse will take a while longer (a second or so). Time-lag fuses are used in circuits that have large turn-on currents, such as motor circuits, and other inductive-type circuits.

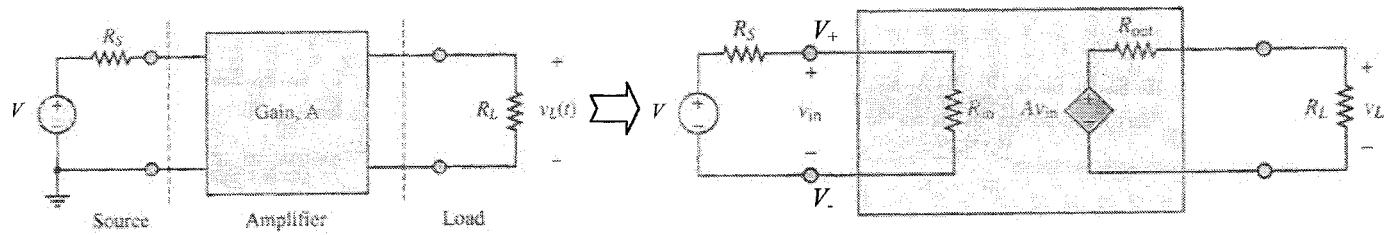
5.4.2 Signal Amplification

The task of signal amplification is mostly performed by operational amplifiers. The term operational amplifier, abbreviated as *op amp*, was coined in the 1940s to refer to a special kind of amplifier that, by proper selection of external components, can be configured to perform a variety of mathematical operations. Depending on the nature of input and output signals, there can be four types of amplifier gain:

- Voltage (voltage out/voltage in)
- Current (current out/current in)
- Transresistance (voltage out/current in)
- Transconductance (current out/voltage in).

Since most op amps are voltage amplifiers, we will limit our discussion to voltage amplifiers. The simplest model for an amplifier (linear) is shown in figure below where A is the gain of the amplifier. Ideally the load voltage would be given. Practically, the “black box” gain on the left is represented by the equivalent circuit shown on the right side of the figure. The load voltage is the amplified version of V_s by the gain A

$$V_L = AV$$



Using the voltage divider rule will result in

$$V_L = A \left(\frac{R_{in}}{R_S + R_{in}} \cdot \frac{R_L}{R_{out} + R_L} \right) V$$

An operational amplifier is a high-gain linear amplifier that has

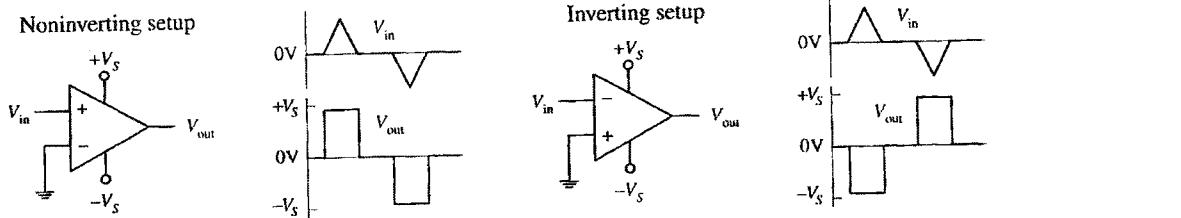
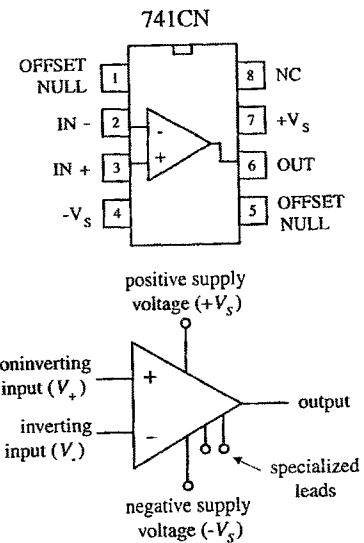
- Very high open-loop gain: A is 10^4 to 10^7 ;
- Very high input resistance: $R_{in} > 1 \text{ M}\Omega$;
- Low output resistance: $R_{out} = 10 - 100 \Omega$.

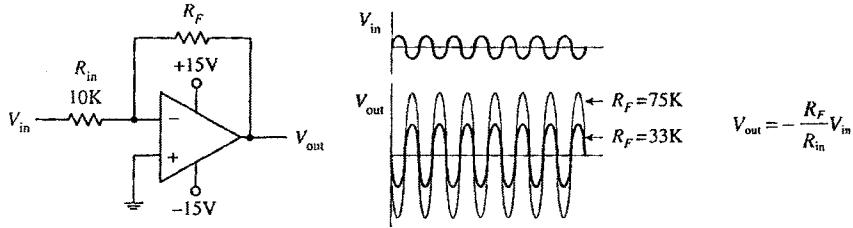
This leads to the following three circuit-simplifying assumptions:

1. **$V_+ = V_-$** : The output voltage is equal to $A(V_+ - V_-) = AV_{in}$, where A is a very large number, typically between 10^4 and 10^7 . Thus, even a small difference between V_- and V_+ will cause a very large output. However, the output has a practical upper limit established by the power supply; therefore, to keep the output from exceeding its limits, the difference between V_- and V_+ must be very small, thus they are virtually the same.
2. **Input current is zero**: The input resistance of an op amp is very high; therefore, it can be modeled as an open circuit. Thus, no current flows between the inputs. In reality the input impedance is between 10^6 and $10^{12} \Omega$.
3. **Output resistance is zero**: A low-output resistance means that the output voltage will not be pulled down even if the load draws a lot of current. In real op amps the output resistance is typically between 10 and 100 Ω . This assumption only holds if the load being driven is considerably higher than the output resistance of the op amp, which is the case in most applications.

From a practical point of view, an operational amplifier is an integrated-circuit device with a non-inverting input V_+ , an inverting input V_- , two DC power supply pins (positive $+V_S$ and negative $-V_S$), an output terminal, and a few other specialized leads used for fine-tuning. Figure to the right shows the pin connections and schematics for a typical SN741 operational amplifier.

By itself, the op amp's operation is simple. If the voltage applied to the inverting pin is more positive than the voltage applied to the non-inverting pin, the output saturates toward the negative supply voltage ($-V_S$). Conversely, if the non-inverting pin is more positive than the inverting pin, the output saturates toward the positive supply voltage ($+V_S$), as illustrated in Figure below. Thus, a mere op amp switches from one maximum output state to another whenever there is a voltage difference between its inputs. However, the significance of this device appears when a so-called *negative feedback* is applied. When voltage is fed back from the output terminal to the inverting terminal (this is referred to as *negative feedback*), the gain of an op amp can be controlled, i.e., the op amp's output is prevented from saturating.





For example, a feedback resistor R_F placed between the output and the inverting input, as shown in the figure above, acts to convey the state of the output back to the op amp's input. This feedback information basically tells the op amp to readjust its output voltage to a value determined by the resistance of the feedback resistor. The circuit in the above figure is called an *inverting amplifier*, and has an output equal to $-V_{in} / (R_F / R_{in})$. The negative sign means that the output is inverted relative to the input, a result of the inverting input. The gain is then simply the output voltage divided by the input voltage, or $-R_F / R_{in}$ (the negative sign indicates that the output is inverted relative to the input). As you can see from this equation, if you increase the resistance of the feedback resistor, there is an increase in the voltage gain. On the other hand, if you decrease the resistance of the feedback resistor, there is a decrease in the voltage gain. Hence, As a result of negative feedback, whenever an op amp senses a voltage difference between its inverting and non-inverting inputs, it responds by feeding back as much current/voltage through the feedback network as is necessary to keep this difference equal to zero ($V_+ - V_- = 0$)

By adding other components to the negative-feedback circuit, an op amp can be made to do a number of useful tasks besides pure amplification. Other interesting op amp circuits include voltage-regulator circuits, current-to-voltage converters, voltage-to-current converters, oscillator circuits, mathematical circuits (adders, subtractors, multipliers, differentiators, integrators, etc.), waveform generators, active filter circuits, active rectifiers, peak detectors, sample-and-hold circuits, etc. Some of these circuits are explained in sequel.

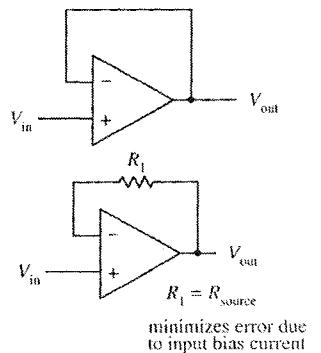
5.4.2.1 Various Op Amp Circuits

5.4.2.1.1 Buffer (Unity Gain Amplifier)

In this case, $V_{in} = V_+$ and $V_{out} = V_-$. Considering the fact that, due to the negative feedback, the output will attempt to make $V_+ - V_- = 0$, the input/output gain of such a circuit becomes 1.

$$Gain = \frac{V_{out}}{V_{in}} = 1$$

Although there is no amplification in this circuit, it should be noted that an op amp's input impedance is huge, while its output impedance is extremely small. This feature makes this circuit useful for circuit-isolation applications. In other words, the circuit acts as a buffer. With real op amps, it may be necessary to throw in a resistor in the feedback loop (lower circuit). The resistor acts to minimize voltage offset errors caused by input bias currents (leakage). The resistance of the feedback resistor should be equal to the source resistance that will be discussed in sequel.



5.4.2.1.2 Inverting Amplifier

Since V_+ is grounded, V_- will also be 0 V. By using the Ohm's law, currents I_1 and I_2 are found as:

$$I_1 = \frac{V_{in} - V_-}{R_1} = \frac{V_{in} - 0V}{R_1} = \frac{V_{in}}{R_1}$$

$$I_2 = \frac{V_{out} - V_-}{R_2} = \frac{V_{out} - 0V}{R_2} = \frac{V_{out}}{R_2}$$

Because an ideal op amp has infinite input impedance, no current will enter its inverting pin. Therefore, Kirchhoff's junction rule can be applied to get $I_1 = -I_2$. Substituting the calculated values of I_1 and I_2 into this expression results in

$$Gain = \frac{V_{out}}{V_{in}} = -\frac{R_2}{R_1}$$

The negative sign indicates that the signal that enters the input will be inverted (shifted 180°). Notice that if $R_1 = R_2$ the gain is -1 (the negative sign means the output is inverted). In this case you get a so-called *unity-gain inverter*, or an *inverting buffer*. When using real op amps that have relatively high input bias currents (e.g., bipolar op amps), it may be necessary to place a resistor with a resistance equal to $R_1 \parallel R_2$ between the non-inverting input and ground to minimize voltage offset errors.

5.4.2.1.3 Non-inverting Amplifier

Since $V_+ = V_{in}$ it concludes that $V_- = V_{in}$, and by using the voltage divider relation we have

$$V_- = \frac{R_1}{R_1 + R_2} V_{out} = V_{in}$$

Therefore,

$$Gain = \frac{V_{out}}{V_{in}} = \frac{R_1 + R_2}{R_1} = 1 + \frac{R_2}{R_1}$$

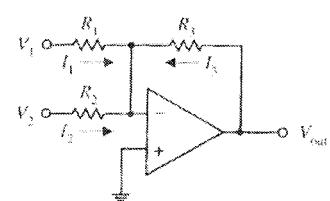
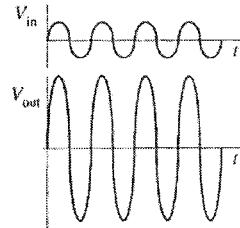
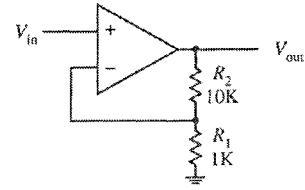
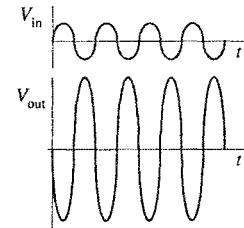
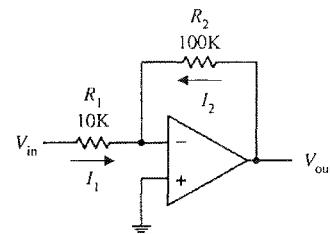
Unlike the inverting amplifier, the output of this circuit is non-inverted. With real op amps, to minimize voltage offset errors due to bias current, set $R_1 \parallel R_2 = R_{source}$.

5.4.2.1.4 Summing Amplifier

Since V_+ is grounded, $V_+ = V_- = 0V$. Solving for I_1 , I_2 , and I_3 by applying Ohm's law results in

$$I_1 = \frac{V_1 - V_-}{R_1} = \frac{V_1 - 0V}{R_1} = \frac{V_1}{R_1}$$

$$I_2 = \frac{V_2 - V_-}{R_2} = \frac{V_2 - 0V}{R_2} = \frac{V_2}{R_2}$$



$$I_3 = \frac{V_{out} - V_-}{R_3} = \frac{V_{out} - 0V}{R_3} = \frac{V_{out}}{R_3}$$

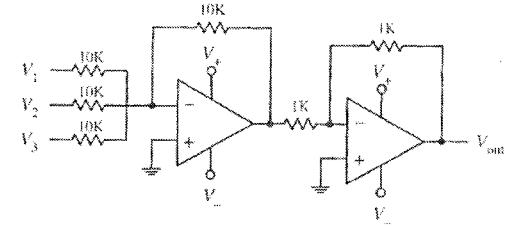
Considering the fact that no current enters the op amp's inverting terminal, the Kirchhoff's junction rule can be applied to combine I_1, I_2, I_3 into one expression as

$$I_3 = -(I_1 + I_2) = -I_1 - I_2$$

Therefore,

$$V_{out} = -\frac{R_3}{R_1}V_1 - \frac{R_3}{R_2}V_2 = -\left(\frac{R_3}{R_1}V_1 + \frac{R_3}{R_2}V_2\right)$$

Note that the sum is negative. To obtain a positive sum, you can add an inverting stage, as shown in figure right. Here, three inputs are added together to yield the output $V_{out} = V_1 + V_2 + V_3$. Again, for some real op amps, an additional input-bias compensation resistor placed between the non-inverting input and ground may be needed to avoid offset error caused by input bias current. Its value should be equal to the parallel resistance of all the input resistors.



5.4.2.1.5 Difference Amplifier

Assuming that no current enters the inputs, the voltage at the non-inverting terminal is obtained by the voltage divider relation:

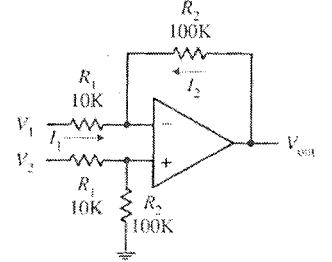
$$V_+ = \frac{R_2}{R_1 + R_2}V_2$$

By applying Kirchhoff's current junction law to the inverting input ($I_1 = I_2$):

$$\frac{V_1 - V_-}{R_1} = \frac{V_- - V_{out}}{R_2}$$

And, knowing that $V_+ = V$ and substituting the V_+ term in for V in the previous equation results in

$$V_{out} = \frac{R_2}{R_1}(V_2 - V_1)$$



5.4.2.1.6 Integrator

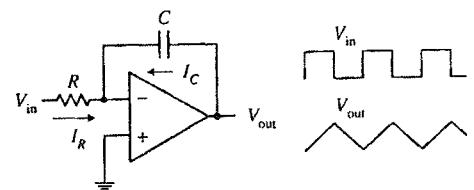
It is known that $V_+ = V_- = 0V$ and $I_R + I_C = 0A$. The current I_R can be obtained using Ohm's law:

$$I_R = \frac{V_{in} - V_-}{R} = \frac{V_{in} - 0V}{R} = \frac{V_{in}}{R}$$

I_C is found by using the capacitance current relation:

$$I_C = C \frac{dV}{dt} = C \frac{d(V_{out} - V_-)}{dt} = C \frac{d(V_{out} - 0V)}{dt} = \frac{dV_{out}}{dt}$$

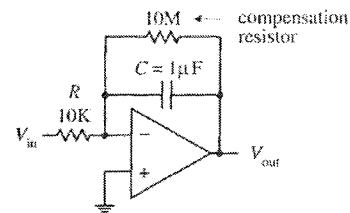
Substituting these values into $I_R + I_C = 0A$ and rearranging will result in



$$dV_{out} = -\frac{1}{RC} V_{in} dt$$

$$V_{out} = -\frac{1}{RC} \int V_{in} dt$$

Such a circuit is called an *integrator*; the input signal is integrated at the output. One problem with this circuit is that the output tends to drift, even with the input grounded, due to non-ideal characteristics of real op amps such as voltage offsets and bias current. A large resistor placed across the capacitor can provide DC feedback for stable biasing. Also, a compensation resistor may be needed between the non-inverting terminal and ground to correct voltage offset errors caused by input bias currents. The size of this resistor should be equal to the parallel resistance of the input resistor and the feedback compensation resistor.

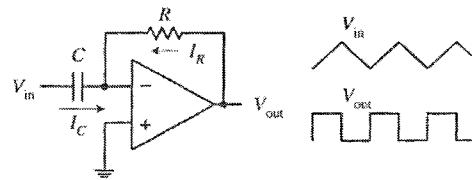


5.4.2.1.7 Differentiator

Similar to the integrator circuit, the currents I_C and I_R can be found as

$$I_R = \frac{V_{out} - V_-}{R} = \frac{V_{out} - 0V}{R} = \frac{V_{out}}{R}$$

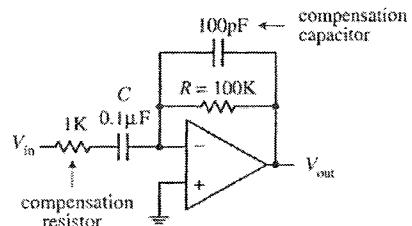
$$I_C = C \frac{dV}{dt} = C \frac{d(V_{in} - V_-)}{dt} = C \frac{d(V_{in} - 0V)}{dt} = \frac{dV_{in}}{dt}$$



Substituting these values into $I_R + I_C = 0$ A and rearranging will result in

$$V_{out} = -RC \frac{dV_{in}}{dt}$$

Such a circuit is called a *differentiator*; the input signal is differentiated at the output. However, The differentiator circuit shown in the previous figure is not practical. It is extremely susceptible to noise due to the op amp's high AC gain. Also, the feedback network of the differentiator acts as an RC low-pass filter that contributes a 90° phase lag within the loop and may cause stability problems. A more practical differentiator is shown in figure right. Here, both stability and noise problems are corrected with the addition of a feedback capacitor and input resistor. The additional components provide high-frequency rolloff to reduce high-frequency noise. These components also introduce a 90° lead to cancel the 90° phase lag. The effect of the additional components, however, limits the maximum frequency of operation. At very high frequencies, the differentiator becomes an integrator. Finally, an additional input-bias compensation resistor placed between the non-inverting input and ground may be needed to avoid offset error caused by input bias current. Its value should be equal to the resistance of the feedback resistor.



5.4.2.2 Op Amp Specifications and Types

For choosing the right op amps for a specific task, it is essential to understand the content of op amp data sheets. These sheets provide crucial information about the performance of the op amp, which should be closely reviewed before selecting the op amp. Some important specification are explained below.

Common-Mode Rejection Ratio (CMRR): The input to a difference amplifier, in general, contains two components: a common-mode and a difference-mode signal. The common-mode signal voltage is the average of the two inputs, whereas the difference-mode signal is the difference between the two inputs. Ideally, an amplifier affects the difference-mode signals only. However, the common-mode signal is also amplified to some degree. The *common-mode rejection ratio* (CMRR), which is defined as the ratio of the difference signal voltage gain to the common-mode signal voltage gain

provides an indication of how well an op amp does at rejecting a signal applied simultaneously to both inputs. The greater the value of the CMRR, the better is the performance of the op amp.

Differential-Input Voltage Range: Range of voltage that may be applied between input terminals without forcing the op amp to operate outside its specifications. If the inputs go beyond this range, the gain of the op amp may change drastically.

Differential input impedance: Impedance measured between the non-inverting and inverting input terminals.

Input Offset Voltage: In theory, the output voltage of an op amp should be zero when both inputs are zero. In reality, however, a slight circuit imbalance within the internal circuitry can result in an output voltage. The input offset voltage is the amount of voltage that must be applied to one of the inputs to zero the output.

Input Bias Current: Theoretically, an op amp should have an infinite input impedance and therefore no input current. In reality, however, small currents, typically within the nano-amp to pico-amp range, may be drawn by the inputs. The average of the two input currents is referred to as the *input bias current*. This current can result in a voltage drop across resistors in the feedback network, the bias network, or source impedance, which in turn can lead to error in the output voltage. Input bias currents depend on the input circuitry of an op amp. With FET op amps, input bias currents are usually small enough not to cause serious offset voltages. Bipolar op amps, on the other hand, may cause problems. With bipolar op amps, a compensation resistor is often required to center the output, as will be discussed in sequel.

Input Offset Current: This represents the difference in the input currents into the two input terminals when the output is zero. The input terminals of a real op amp tend to draw in different amounts of leakage current, even when the same voltage is applied to them. This occurs because there is always a slight difference in resistance within the input circuitry for the two terminals that originates during the manufacturing process. Therefore, if an op amp's two terminals are both connected to the same input voltage, different amounts of input current will result, causing the output to be offset. Op amps typically come with offset terminals that can be wired to a potentiometer to correct the offset current, as will be discussed in sequel.

Voltage Gain (A): A typical op amp has a voltage gain of 10^4 to 10^6 (or 80 to 120 dB; $1\text{dB} = 20 \log_{10}A$) at DC.

Unity Gain Frequency (f_T): This index indicates how high of a frequency can the op amp amplify as well as how much amplification it can provide at that frequency. With real op amps, the voltage gain decreases as the signal frequency increases. The gain drops to 1 dB at a frequency called the *Unity-gain frequency* (f_T), typically from 1 to 10 MHz. This is a result of high-frequency limitations in the op amp's internal circuitry.

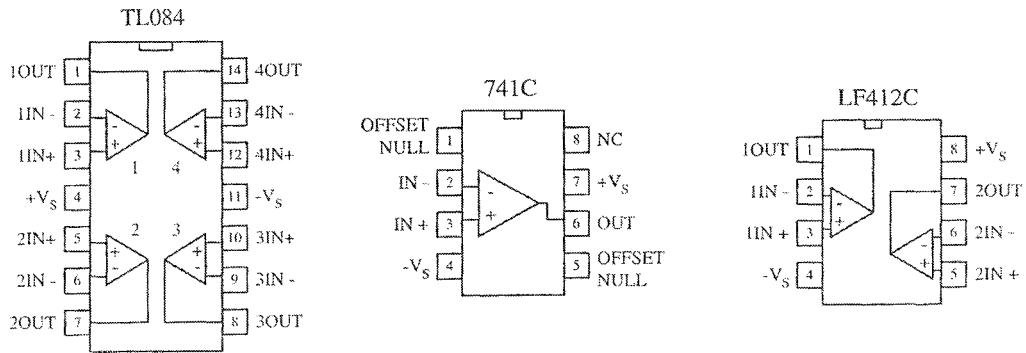
Output Voltage Swing: This is the peak output voltage swing, referenced to zero, that can be obtained without clipping.

Slew Rate: This represents the maximum rate of change of an op amp's output voltage with time. The limitation of output change with time results from internal or external frequency compensation capacitors slowing things down, which in turn results in delayed output changes with input changes (propagation delay). At high frequencies, the magnitude of an op amp's slew rate becomes more critical. A general-purpose op amp like the 741 has a 0.5 V/ μs slew rate, which is relatively small compared with the high-speed HA2539's slew rate of 600 V/ μs .

Supply Current. This represents the current that is required from the power supply to operate the op amp with no load present and with an output voltage of zero.

There is a large selection of general-purpose and precision op amps to choose from. Op amps fall into one of the following categories (based on input circuitry): bipolar, JFET, MOSFET, or some hybrid thereof (e.g., BiFET). In general bipolar op amps, like the 741 (industry standard), have higher input bias currents than either JFET or MOSFET types. This means that their input terminals have a greater tendency to "leak in" current. Input bias current results in voltage drops across resistors of feedback networks, biasing networks, or source impedances, which in turn can offset the output voltage. The amount of offset a circuit can tolerate ultimately depends on the application. Some general-purpose op amps are shown in figure below.

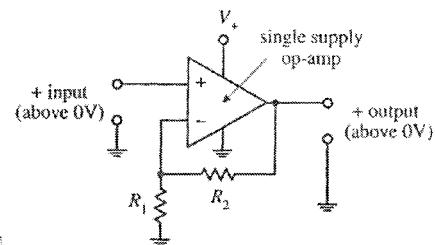
Precision op amps (such as OP-27, LT1057, AD713) are specifically designed for high stability, low offset voltages, low bias currents, and low drift parameters. A simple way to avoid problems associated with input bias current is to use a FET op amp. A typical JFET op amp has a very low input bias current, typically within the lower pico-amp range as compared with the nano-amp range for a typical bipolar op amp. Some MOSFET op amps come with even lower input bias currents, often as low as a few tenths of a pico-amp. Though FET op amps have lower input bias current than bipolar op amps, there are other features they have that are not quite as desirable. For example, JFET op amps often experience an



undesired effect called *phase inversion*. If the input common-mode voltage of the JFET approaches the negative supply too closely, the inverting and non-inverting input terminals may reverse directions, i.e., negative feedback becomes positive feedback causing the op amp to latch up. This problem can be avoided by using a bipolar op amp or by restricting the common-mode range of the signal. Some other general features. This problem can be avoided by using a bipolar op amp or by restricting the common-mode range of the signal. Some other general features of bipolar and FET op amps are: offset voltage (low for bipolar, medium for JFET, medium to high for MOSFET), offset drift (low for bipolar, medium for FET), bias matching (excellent for bipolar, fair for FET), bias/temperature variation (low for bipolar, fair for FET).

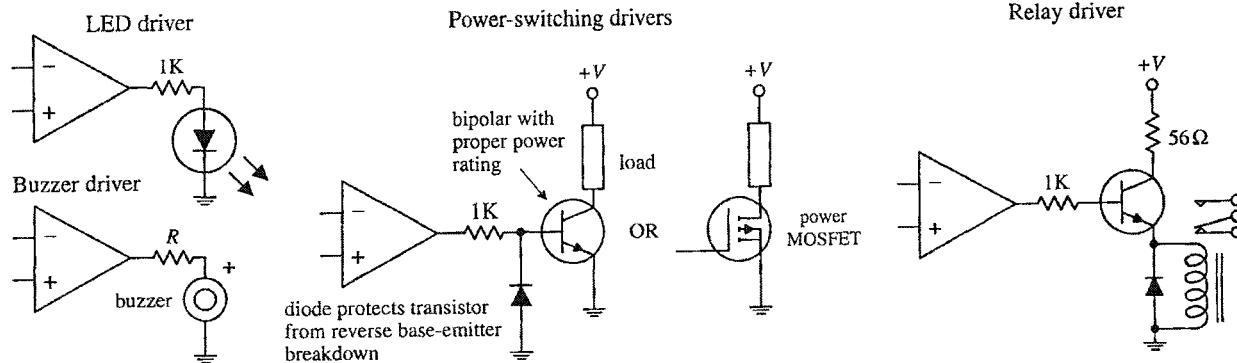
Another feature to look when purchasing an op amp is whether the op amp is internally or externally frequency compensated. An externally compensated op amp requires external components to prevent the gain from dropping too quickly at high frequencies, which can lead to phase inversions and oscillations. Internally compensated op amps take care of these problems with internal circuitry.

Another useful type of op amps are *single-supply* op amps (such as AD820 and LT1078). These op amps are designed to be operated from a single positive supply, and allow input voltages all the way down to the negative rail (normally tied to the ground). The circuit in right shows a simple DC amplifier that uses a single-supply op amp. It is, however, important to note that the output of this type of amplifier cannot become negative, thus it cannot be used for, say, AC-coupled audio signals. These op amps are usually used in battery-operated devices.

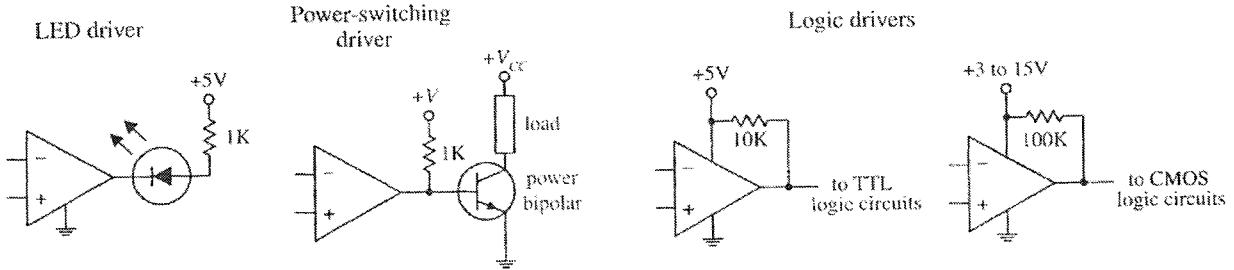


5.4.2.3 Applications

5.4.2.3.1 Op Amp Output Drivers (for loads that are either ON or OFF)



5.4.2.3.2 Comparator Output Drivers



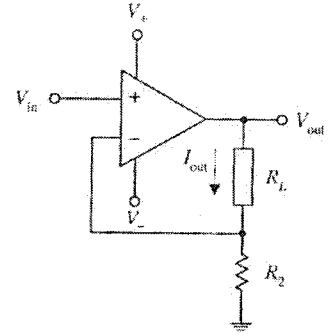
5.4.2.3.3 Voltage-to-Current Converter

This circuit is a simple current source whose output current is determined by an input voltage applied to the non-inverting terminal of the op amp. The output current and voltage are determined as:

$$V_{out} = \frac{(R_L + R_2)}{R_2} V_{in}$$

$$I_{out} = \frac{V_{out} - V_{in}}{R_L} = \frac{(R_L + R_2)/R_2 - 1}{R_L} V_{in} = \frac{V_{in}}{R_2}$$

V_{in} can be set with a voltage divider.

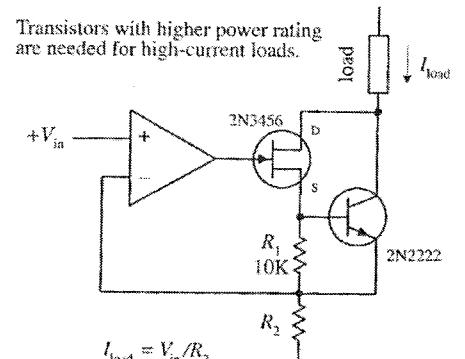


5.4.2.3.4 Precision Current Source

In a precision current circuit, as shown in figure right, a JFET is used to drive a bipolar output transistor used to sink current through a load. Unlike the preceding current source, this circuit is less susceptible to output drift. Use of the JFET helps achieve essentially zero bias current error (a single bipolar output stage will leak base current). This circuit is accurate for output currents larger than the JFET's $I_{DS,ON}$ and provided V_{in} is greater than 0 V. For large currents, the FET-bipolar combination can be replaced with a Darlington transistor, provided its base current does not introduce significant error. The output current or load current is determined by

$$I_{load} = \frac{V_{in}}{R_2}$$

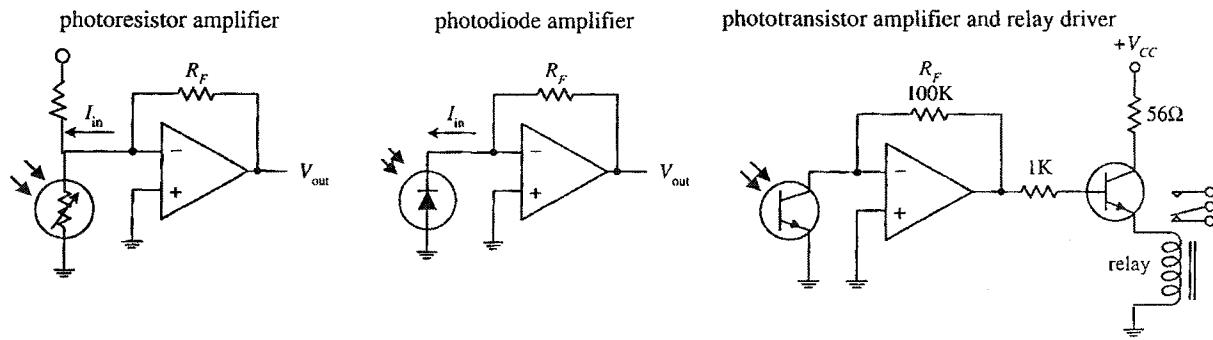
R_2 acts as an adjustment control. Additional compensation may be required depending on the load reactance and the transistors' parameters. Make sure to use transistors of sufficient power ratings to handle the load current in question.



5.4.2.3.5 Current-to-Voltage-Converter

Circuits shown in figure below (light-activated circuits) transform a current into a voltage. The feedback resistor R_F helps establish a voltage at the inverting input and controls the swing of the output. The output voltage is obtained by

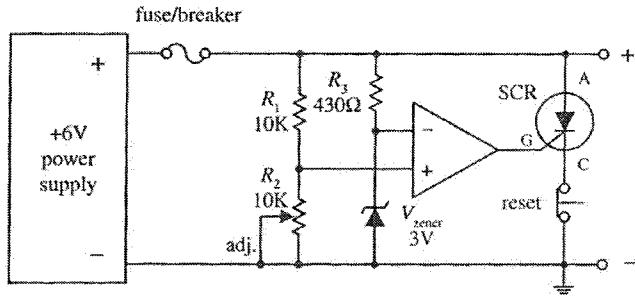
$$V_{out} = I_{in} R_F$$



In these circuits, the generated output voltage is proportional to the amount of input current drawn through the light sensor.

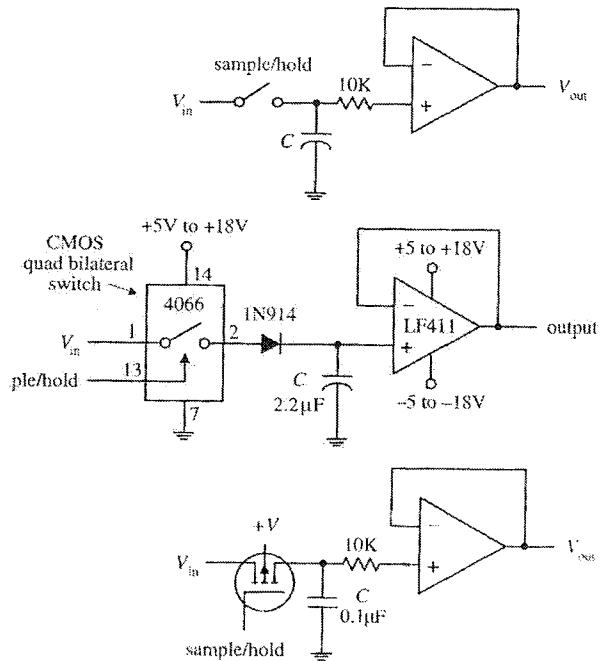
5.4.2.3.6 Over-voltage Protection

The circuit shown in figure on the right acts as a fast-acting over-voltage protection control used to protect sensitive loads from voltage surges generated in the power supply. First, assume that the supply is doing what it should, i.e., generating a constant +6V. In this case, the voltage applied to the op amp's non-inverting input is set to 3V (by means of the R_1 - R_2 voltage divider; the pot provides fine-tuning adjustment). At the same time, the inverting input is set to 3V by means of the 3V Zener diode. The op amp's differential input voltage in this case is therefore zero, making the op amp's output zero (op amp acts as a comparator). With the op amp's output zero, the SCR is OFF, and no current will pass from anode (A) to cathode to ground. Then, assume that there is a sudden surge in the supply voltage. When this occurs, the voltage at the non-inverting input increases, while the voltage at the inverting input remain at 3V (due to the 3V Zener). This causes the op amp's output to go high, triggering the SCR ON and diverting all current from the load to ground in the process. As a result, the fuse (breaker) blows and the load is saved. The switch is opened to reset the SCR.



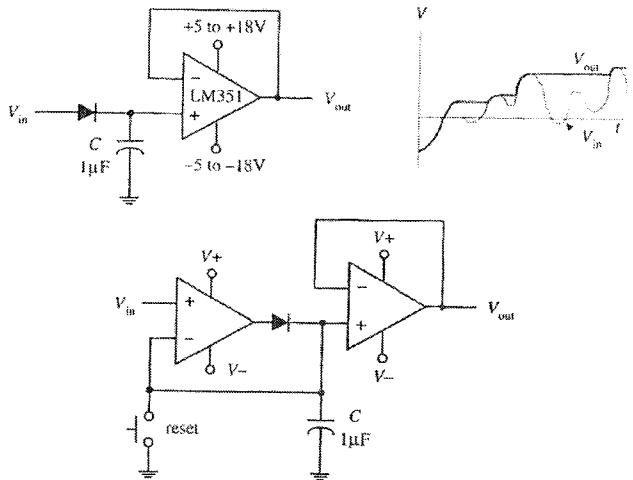
5.4.2.3.7 Sample-and-Hold Circuits

Sample-and-hold circuits are used to sample an analog signal and hold it so that it can be analyzed or converted into a digital signal. In the first circuit (top), a switch acts as a sample/hold control. Sampling begins when the switch is closed and ends when the switch is opened. When the switch is opened, the input voltage present at that exact moment will be stored in C . The op amp acts as a unity-gain amplifier (buffer), relaying the capacitor's voltage to the output but preventing the capacitor from discharging (recall that ideally, no current enters the inputs of an op amp). The length of time a sample voltage can be held varies depending on how much current leaks out of the capacitor. To minimize leakage currents, use op amps with low input-bias currents (e.g., FET op amps). In the other two circuits, the sample/hold manual switch is replaced with an electrically controlled switch. The lower circuit uses a bilateral switch, while the middle circuit uses a MOSFET. Capacitors best suited for sample/hold applications include Teflon, polyethylene, and polycarbonate dielectric capacitors.



5.4.2.3.8 Peak Detectors

The circuits shown on the right act as peak detectors. They follow an incoming voltage signal and store its maximum voltage within C. The op amp in the upper circuit acts as a buffer. It “measures” the voltage in C, outputs that voltage, and prevents C from discharging. The diode also prevents the capacitor from discharging when the input drops below the peak voltage stored on C. The second circuit is a more practical peak detector. The additional op amp makes the detector more sensitive; it compensates for the diode voltage drop (around 0.6V) by feeding back C's voltage to the inverting terminal. Also, this circuit incorporates a switch to reset the detector. Often, peak detectors use a FET in place of the diode and use the FET's gate as a reset switch. Reducing the capacitance of C promotes faster response times for changes in V_{in} .



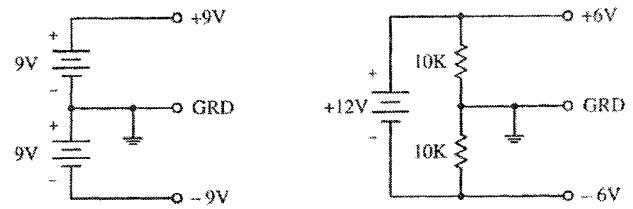
5.4.2.4 Practical Notes

5.4.2.4.1 Powering Op Amps

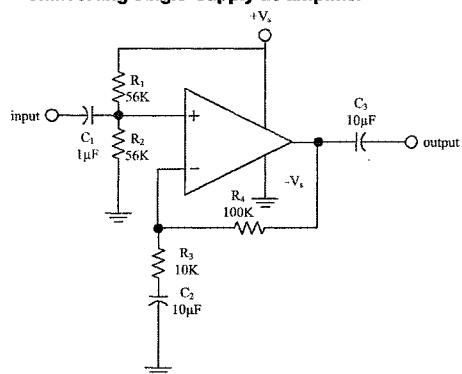
Most op amp applications require a dual-polarity supply. If you are using batteries to power an op amp, one of the arrangements shown in figure right can be useful.

It is often desirable to avoid “split supplies”, especially with small battery-powered applications. One option in such a case is to use a single-supply op amp. However, the output cannot go negative, hence this option cannot be used for applications where a negative output voltage is also needed or the signal is AC (such as AC-coupled audio signals). Another option is to use a conventional op amp and apply a DC level to one of the inputs of the op amp using a voltage-divider network. This, in turn, provides a DC offset level at the output. Both input and output offset levels are referenced to ground (the negative terminal of the battery). With the input offset voltage in place, when an input signal goes negative, the voltage applied to the input of the op amp will dip below the offset voltage, but will not go below ground (provided you have set the bias voltage large enough, and provided the input signal is not too large; otherwise, clipping occurs). The output, in turn, will fluctuate about its offset level. To allow for input and output coupling, input and output capacitors are needed. The two circuits shown in figure below illustrate non-inverting and inverting AC-coupled amplifiers (designed for audio) that use conventional op amps that run off a single supply voltage.

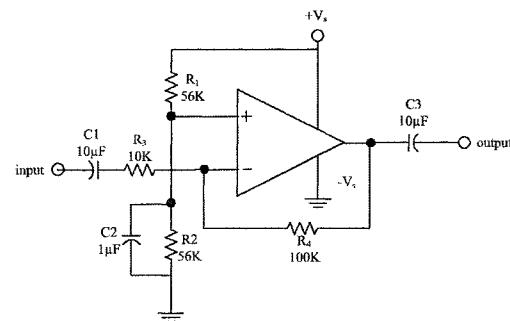
When using conventional op amps with single supply voltages, make sure to stay within the minimum supply voltage rating of the op amp, and also make sure to account for maximum output swing limitations and maximum common-mode input range.



noninverting single-supply ac amplifier



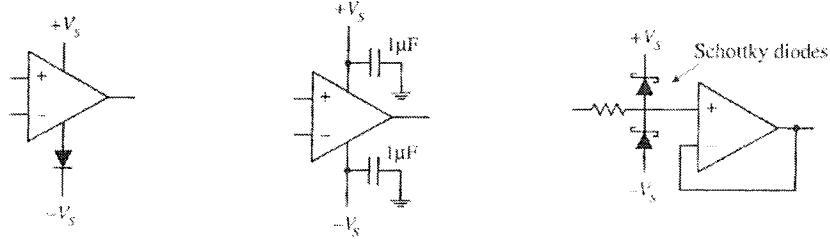
inverting single-supply ac amplifier



5.4.2.4.2 Protecting Op Amps

Never reverse an op amp's power supply leads. This can damage the op amp IC. One way to avoid this is to place a diode between the op amp's negative supply terminal and the negative supply, as shown in figure below (left). Also, keep wires running from the power supply to the op amp's supply terminals short and direct. This helps prevent unwanted oscillations / noise from arising in the output. Disturbances may also arise from variations in supply voltage. To eliminate these effects, place bypass capacitors from the supply terminals to ground as shown in figure below (middle). A $0.1\mu\text{F}$ disk capacitor or $1.0\mu\text{F}$ tantalum capacitor should do the job.

Op amps can experience a serious form of latch if the input signal becomes more positive or negative than the respective op amp power supplies. If the input terminals go more positive than $+V_s + 0.7\text{V}$ or more negative than $-V_s - 0.7\text{V}$, current may flow in the wrong direction within the internal circuitry, short-circuiting the power supplies and destroying the device. To avoid this potentially fatal latch-up, it is important to prevent the input terminals of op amps from exceeding the power supplies. This feature has vital consequences during device turn-on; if a signal is applied to an op amp before it

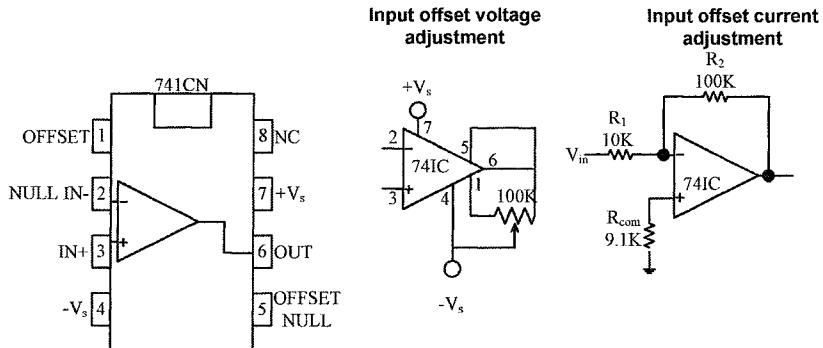


is powered, it may be destroyed at the moment power is applied. A "hard wire" solution to this problem involves clamping the input terminals at risk with diodes (preferably fast low-forward-voltage Schottky diodes), as shown in figure above (right). Current-limiting resistors also may be needed to prevent the diode current from becoming excessive. This protection circuitry has some problems, however. Leakage current in the diodes may increase the error. See manufacturers' literature for more information.

5.4.2.4.3 Voltage and Current Offset Compensation

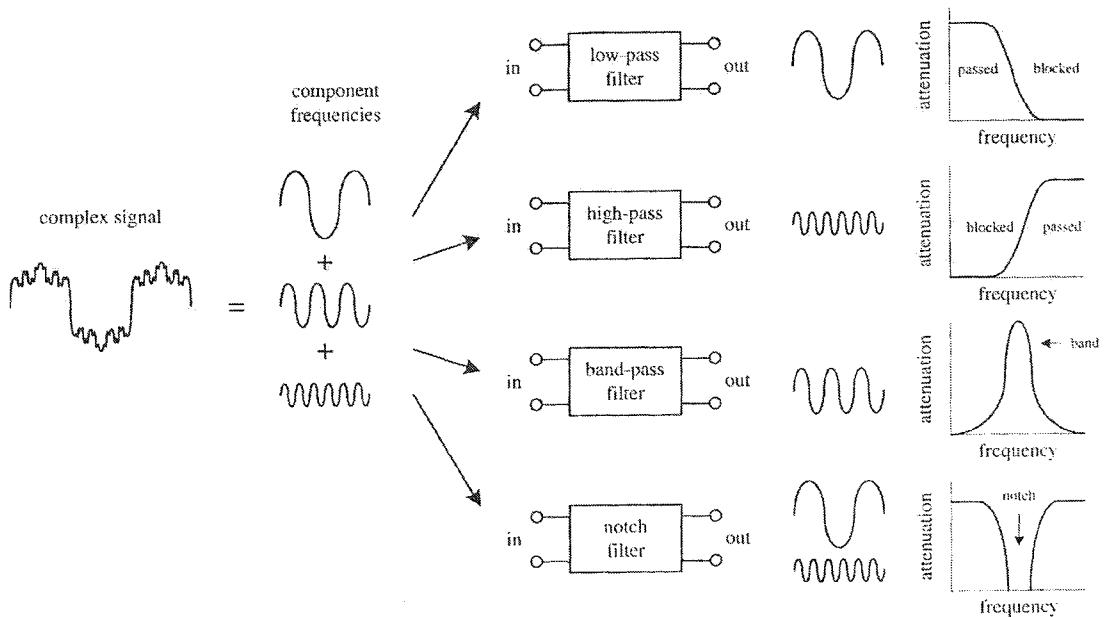
Theoretically, the output voltage of an op amp should be zero when both inputs are zero. In reality, however, a slight circuit imbalance within the internal circuitry can result in an output voltage (typically within the micro to milli-Volt range). The input offset voltage is the amount of voltage that must be applied to one of the inputs to zero the output. To zero the input offset voltage, manufacturers usually include a pair of offset null terminals to the op amp chip. A potentiometer is placed between these two terminals, while the pot's wiper is connected to the more negative supply terminal, as shown in figure below. To center the output, the two inputs can be shorted together and an input voltage applied. If the output saturates, the input offset needs trimming. Adjust the pot until the output approaches zero.

With FET op amps, the input bias errors are usually so small (in the pico-amp range) that the output voltage error is insignificant, and hence no compensation is required. However, with bipolar op amps, the input bias currents are in the nano-amp range), and compensation is often necessary. For instance, in the inverting amplifier, the bias current introduces a voltage drop equal to $V_{in} = I_{bias} (R_1 \parallel R_2)$, which is amplified by a factor of $-R_2 / R_1$. In order to compensate this error, a compensation resistor with a resistance equal to $R_1 \parallel R_2$ is placed between the non-inverting terminal and ground, as shown in figure below (right). This resistor makes the op amp "feel" the same input driving resistance.



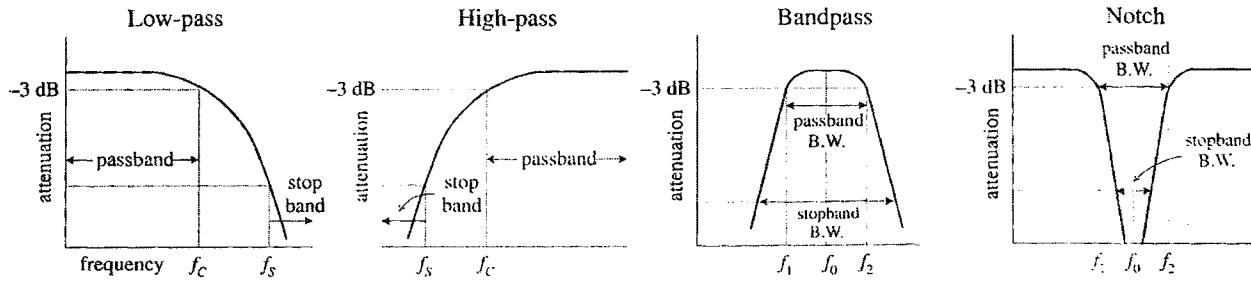
5.4.3 Signal Filtering

The term *filtering* is used to describe the process of removing a certain band of frequencies from a signal and permitting others to be transmitted. The range of frequencies passed by a filter is known as the *pass band*, the range not passed as the *stop band* and the boundary between stopping and passing as the *cut-off frequency*. Filters are classified according to the frequency ranges they transmit or reject. A *low-pass filter* has a pass band that allows all frequencies from 0 up to some frequency to be transmitted. A *high-pass filter* has a pass band that allows all frequencies from some value up to infinity to be transmitted. A *band-pass filter* allows all the frequencies within a specified band to be transmitted. A *band-stop filter (notch filter)* stops all frequencies with a particular band from being transmitted. The performance of these filters is illustrated in figure below.



Filters have many practical applications in electronics. For example, within a DC power supply, filter can be used to eliminate unwanted high-frequency noise present within the AC line voltage, and they act to flatten out pulsing DC voltage generated by the supply's rectifier section. In radio communication, filters make it possible for a radio receiver to provide the listener with only the desired signal while rejecting all others. Likewise, filters allow a radio transmitter to generate only one signal while attenuating other signals that might interfere with different radio transmitters' signals.

Two major classes of filter are *passive filters* and *active filters*. Passive filters are designed using passive elements (e.g., resistors, capacitors, and inductors) and are most responsive to frequencies between around 100 Hz and 300 MHz. The lower frequency limit results from the fact that at low frequencies the capacitance and inductance values become exceedingly large, meaning prohibitively large components are needed. The upper frequency limit results from the fact that at high frequencies parasitic capacitances and inductances wreak havoc. When designing passive filters with very steep attenuation falloff responses, the number of inductor and capacitor sections increases. As more sections are added to get the desired response, the greater is the chance for signal loss to occur. Also, source and load impedances must be taken into consideration when designing passive filters, as that the current that is drawn by the load can change the frequency characteristic of the filter. Active filters, unlike passive filters, are constructed from op amps, resistors, and capacitors. No inductors are needed. Active filters are capable of handling very low frequency signals (approaching 0 Hz), and they can provide voltage gain if needed (unlike passive filters). Active filters can be designed to offer comparable performance to LC filters, and they are typically easier to make, less finicky, and can be designed without the need for large-sized components. Also, with active filters, a desired input and output impedance can be provided that is independent of frequency. One major drawback with active filters is the relatively limited high-frequency range. Above around 100 kHz, active filters can become unreliable, as a result of the op amp's bandwidth and slew-rate requirements. At the high radio frequencies, it is best to use a passive filter.



5.4.3.1 Basic Notions

When describing the performance of a filter, a response curve is usually used, which is the signal attenuation (V_{out} / V_{in}) versus frequency curve, as shown in figure above. Attenuation is often expressed in decibels (dB), as it can take on many orders of magnitude. Attenuation in decibels is given by

$$A_{dB} = 20 \log \left| \frac{V_{out}}{V_{in}} \right|$$

The frequency may be expressed in either angular form ω (expressed in rad/s) or conventional form f (expressed in Hz). The two forms are related by

$$\omega = 2\pi f$$

Filter response curves may be plotted on linear-linear, log-linear, or log-log paper. In the case of log-linear graphs, the attenuation need not be specified in decibels. Some important definitions are explained below.

-3dB Frequency (f_{3dB}): This parameter represents the input frequency that causes the output signal to drop to -3 dB relative to the input signal. The -3dB frequency is equivalent to the cutoff frequency, i.e., the point where the input-to-output power is reduced by one-half, or the point where the input-to-output voltage is reduced by $1/\sqrt{2} = 0.707$. For low-pass and high-pass filters, there is only one -3dB frequency. However, for band-pass and notch filters, there are two -3dB frequencies, typically referred to as f_1 and f_2 .

Center Frequency (f_0): On a linear-log graph, band-pass filters are geometrically symmetrical around the filter's resonant frequency or center frequency, provided the response is plotted on linear-log graph paper (the logarithmic axis representing the frequency). On linear-log paper, the central frequency is related to the -3dB frequencies by the following expression:

$$f_0 = \sqrt{f_1 f_2}$$

For narrow-band band-pass filters, where the ratio of f_2 to f_1 is less than 1.1, the response shape approaches arithmetic symmetry. In this case, we can approximate f_0 by taking the average of -3dB frequencies:

$$f_0 = \frac{f_1 + f_2}{2}$$

Passband: This parameter represents those frequency signals that reach the output with no more than -3 dB worth of attenuation.

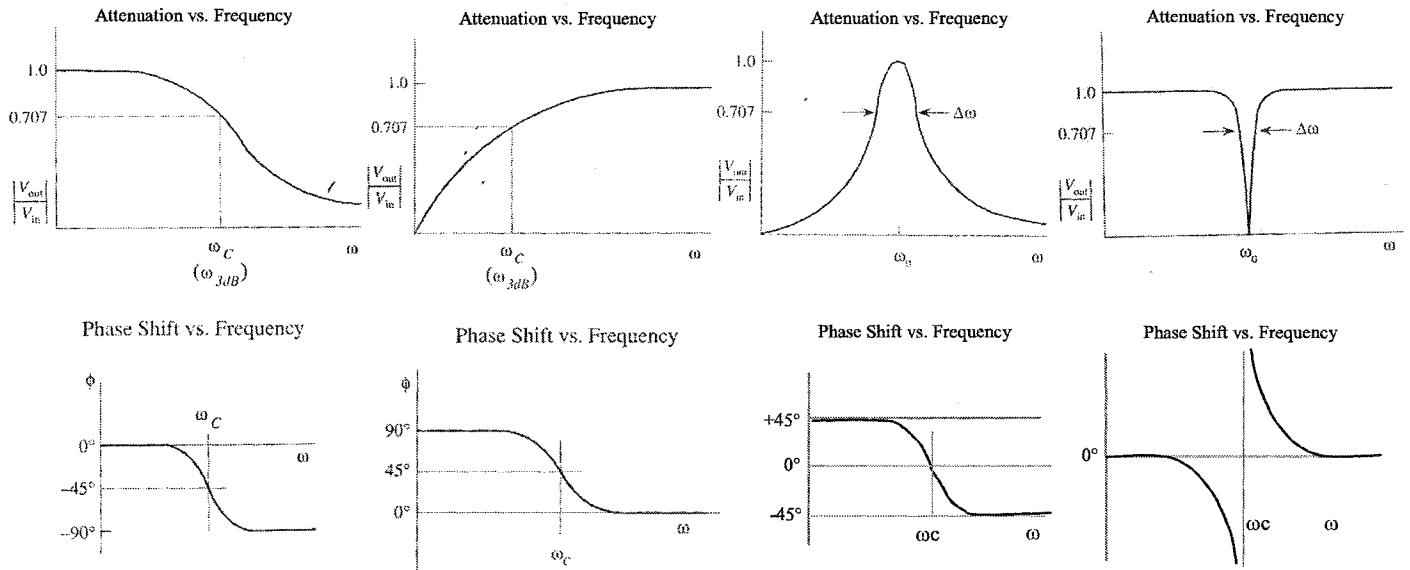
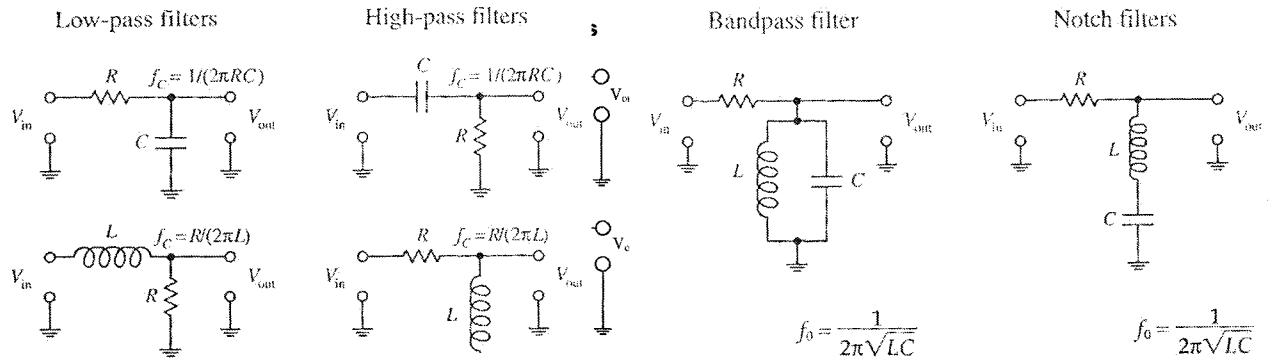
Stop-band Frequency (f_s): This is a specific frequency where the attenuation reaches a specified value set by the designer. For low-pass and high-pass filters, the frequencies beyond the stop-band frequency are referred to as the *stop band*. For band pass and notch filters, there are two stop-band frequencies, and the frequencies between the stop bands are also collectively called the *stop band*.

Quality Factor (Q): This parameter represents the ratio of the center frequency of a band pass filter to the -3dB bandwidth (distance between -3dB points f_1 and f_2):

$$Q = \frac{f_0}{f_2 - f_1}$$

For a notch filter, use $Q = (f_2 - f_1) / f_0$, where f_0 is often referred to as the *null frequency*.

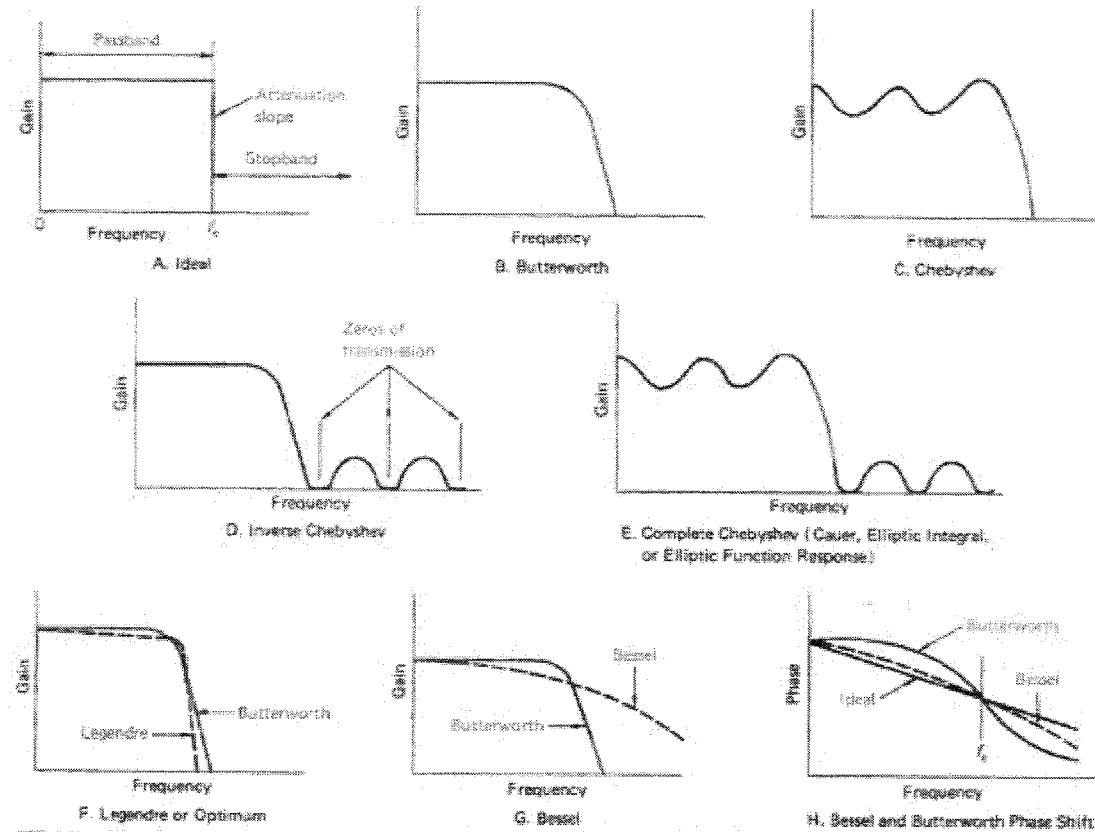
5.4.3.2 Basic Passive Filters



5.4.3.3 Filter Specifications

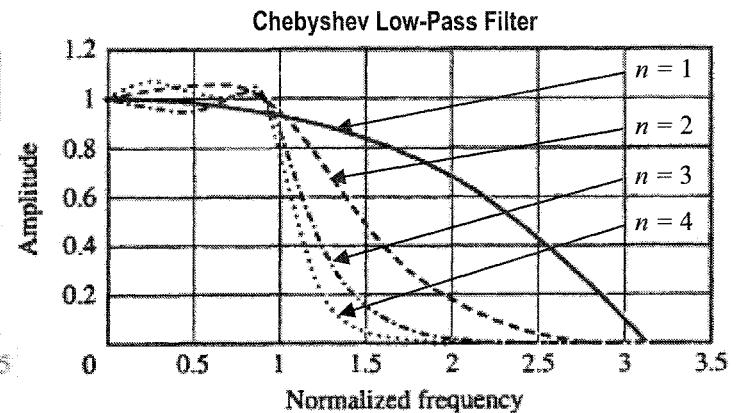
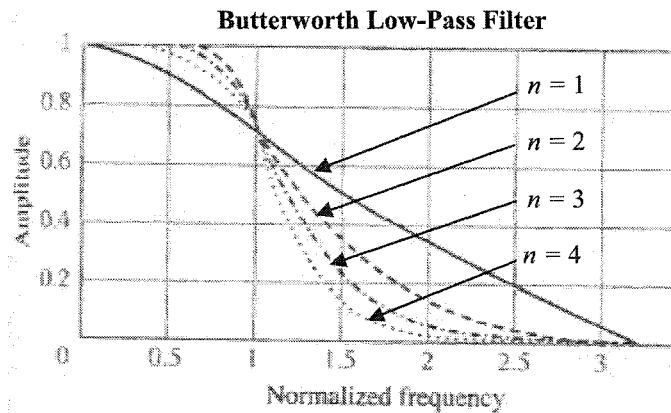
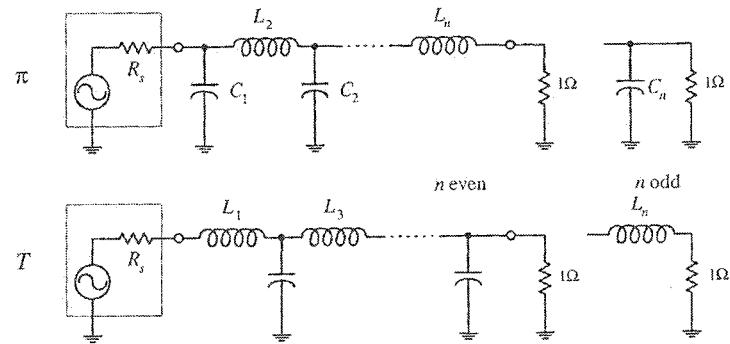
In order to design a passive filter circuit, two important decisions must be made *a priori*: the shape or form of the desirable response, and the order of the filter. The behaviour of a filter varies based on the type of the filter selected to be designed. Figure below illustrates the typical performance of some popular filters, and lists the cons and pros of each type. In many applications, the Butterworth filter would mostly satisfy the design requirements. However, if steeper falloffs are required, chebyshev filters can be used at the expense of ripples present in the Passband. Bessel filters minimize the phase shift, but compromise both flat passbands and steep falloffs. The second factor to specify for designing a filter is its order or the number of LC or RC sets to cascade in the filter circuit. By cascading two or more

basic filters, higher order filters can be obtained. In general, higher-order filters can produce faster transition from passband to stop-band. However, the pay back is a greater phase shift and amplitude distortion. For example, figure on the next page shows the typical response of low-pass Butterworth and Chebyshev filters with different orders.



Comparison of Filter Types

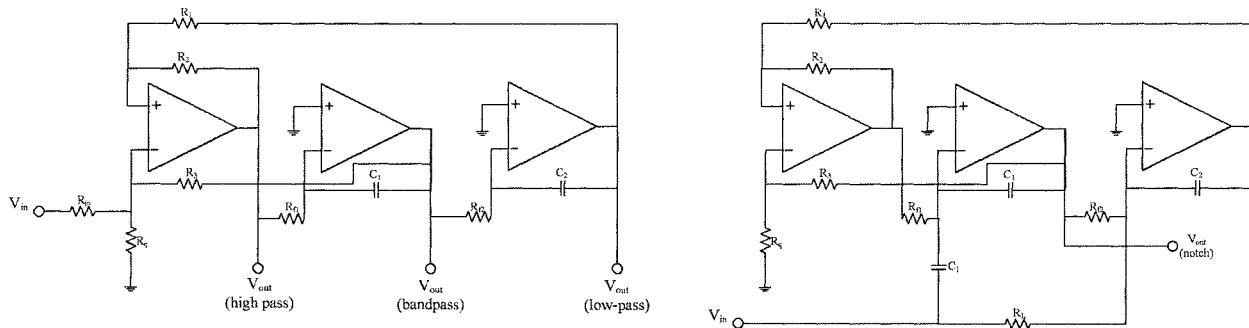
Name of Filter Type	Distinguishing Characteristic	Remarks
Butterworth	Maximally flat amplitude response	Most popular general-purpose filter
Chebyshev	Equal-amplitude ripples in passband	Attenuation slope steeper than in Butterworth near cutoff
Inverse Chebyshev	Equal-amplitude ripples in stopband	No passband ripple; zeros of transmission in stopband
Complete Chebyshev (also called Cauer, elliptic function, elliptic integral, or Zolotarev)	Equal-amplitude ripples in both passband and stopband	Zeros of transmission in stopband
Legendre	No passband ripple, but steeper attenuation slope than Butterworth	Not maximally flat
Bessel (also called Thomson)	Phase characteristic nearly linear in pass region, giving maximally flat group delay	Good for pulse circuits because ringing and overshoot minimized; poor attenuation slope

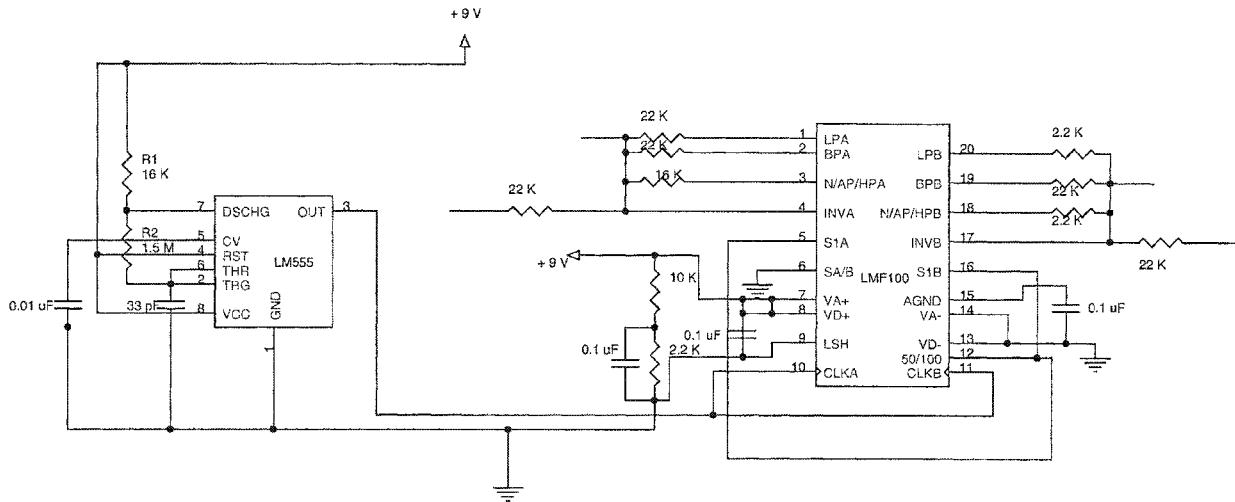


5.4.3.4 Integrated Active Filter Circuits

A number of active filter ICs are available in the market. Two of the major categories of integrated filter circuits include the state-variable and switched-capacitor filter. Both these filter ICs can be programmed to implement all the second-order filters. To design higher-order filters, a number of these ICs can be cascaded together. Typically, only few resistors are needed to program these ICs. Using IC filters allows for great versatility, somewhat simplified design, good precision, and limited design costs.

An example of a state-variable filter IC is the AF100 made by National Semiconductor (figure below). This IC can provide low-pass, high-pass, band-pass, and notch filtering capabilities. The state-variable filter can also provide voltage gain. For AF100, the low-pass gain is set using resistors R_1 and R_m (gain = $-R_1 / R_m$). For the high-pass filter, the gain is set by resistors R_2 and R_m (gain = $-R_2 / R_m$, the negative sign indicates that the output is inverted relative to the input.) Setting the gain for the band-pass and notch functions is more complex. Other parameters, such as Q can be obtained by using design formulas provided by the manufacturer.





Switched-capacitor filters are functionally similar to state-variable IC filters. However, instead of using external resistors to program the desired characteristics, switched-capacitor filters use a high-frequency capacitor-switching network technology. The capacitor-switching networks act like resistors whose values can be changed by changing the frequency of an externally applied clock voltage. The frequency of the clock signal determines which frequencies are passed and which frequencies get rejected. Typically, a digital clock signal is used to drive the filter. This is a useful feature if you are looking to design filters that can be altered by digital circuits. An example of a switched-capacitor IC is National Semiconductor's LMF100 (available in the design lab.), shown in Figure above. By using a few external resistors, a power source, and an applied clock signal, you can program the filter for low-pass, high-pass, and band-pass functions. Similar to the state-variable ICs, manufacturers will provide you with necessary formulas needed for selecting the resistors and the frequency of the clock signal.

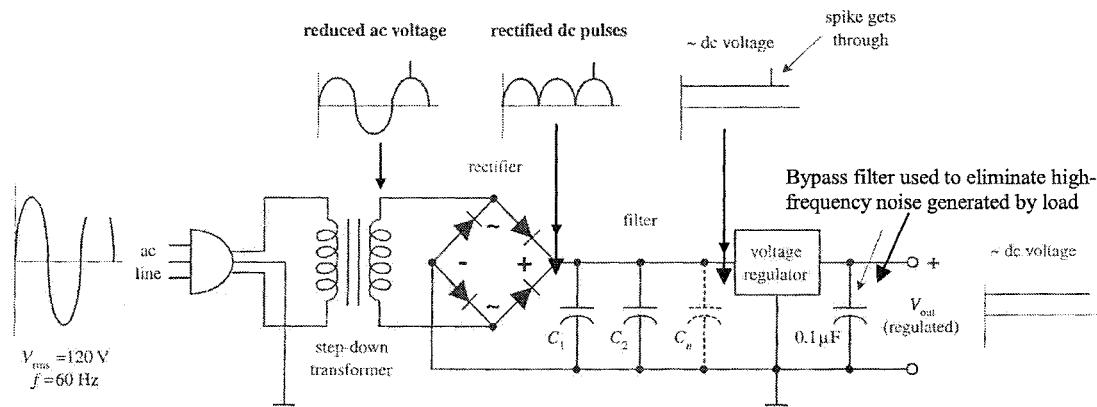
Switched-capacitor filters come in different filter orders. For example, the MF4 is a fourth-order Butterworth low-pass filter, and the MF6 is a sixth-order low-pass Butterworth filter; both are made by National Semiconductor. These two ICs have unity passband gain and require no external components, but they do need a clock input. As an important note, the periodic clock signal applied to a switched-capacitor filter can generate a significant amount of noise (around 10 to 25 mV) present in the output signal. Typically, this is not of much concern because the frequency of the noise, which is the same as the frequency of the clock, is far removed from the signal band of interest. Usually a simple RC filter can be used to get rid of the problem.

5.4.4 Signal Regulation

In the majority of applications, a DC power supply is required that can maintain a fixed voltage while supplying enough current to drive a load. Batteries make good DC supplies, but their relatively small current capacities make them impractical for driving high-current circuits. An alternative solution is to take a 120V AC, 60-Hz line voltage and convert it into a usable DC voltage. The steps of converting the AC line voltage into a usable (typically lower-level) DC voltage are as follows. First, use a transformer to step down the AC voltage. Next, the transformed voltage is applied through a rectifier network to remove the negative swings (or positive swings if you are designing a negative voltage supply). Once the negative swings are eliminated, a filter network is used to flatten out the rectified signal into a nearly flat (rippled) DC voltage pattern. Figure below shows the process in action. One major problem with the supply voltage after the filters is that it is *unregulated*. This means that if there are any sudden surges within the AC input voltage (spikes, dips, etc.), these variations will be expressed at the supply's output. Using an unregulated supply to run sensitive circuits (e.g., digital IC circuits) is not desirable. The current spikes can lead to improper operating characteristics (e.g., false triggering, etc.) and may destroy the ICs in the process. An unregulated supply also has a problem maintaining a constant output voltage as the load resistance varies. If a highly resistive (low-current) load is replaced with a lower-resistance (high-current) load, the unregulated output voltage will drop (Ohm's law). These are the reasons why there is a need to a special circuit (bundled in an IC) that can be placed across the output of an unregulated supply to convert it into a regulated supply, i.e., a supply that eliminates the spikes and maintains a constant output voltage with load variations, as shown in figure below. This special circuit is called a *voltage regulator*. A voltage regulator is designed to automatically adjust the amount of current flowing through a load, so as to maintain a constant output voltage by comparing the supply's DC output with a fixed or programmed internal reference voltage.

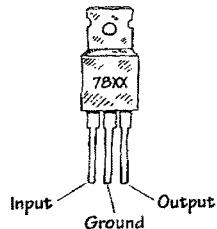
5.4.4.1 Voltage Regulator ICs

Different kinds of voltage regulators can be found in the market. Some are designed to output a fixed positive or negative voltage, and some other are designed to be adjustable. The specification tables for regulators typically provide the following information: output voltage, accuracy (percent), maximum output current, power dissipation, maximum and minimum input voltage, temperature stability ($\Delta V_{out} / \Delta T$), and output impedance (at specific frequencies). The following introduces some popular regulator ICs and their typical circuits. For LM78XX series, the "XX" digits represent the output voltage, e.g., 7805 has 5V output. As a general rule the input voltage of fixed regulators should be limited to 2 to 3 volts above the output voltage. The LM78XX series can handle up to 30 volts input, but the power difference between the input voltage/current ratio and output voltage/current ratio appears as heat. If the input voltage is unnecessarily high the regulator will get very hot. Unless sufficient heat sinking is provided the regulator will shut down. Both LM78XX and LM79XX series can handle a maximum output current of 1.5A if properly heat-sunk. Capacitors are added to the circuit to remove input or output noise, as shown in the figure.

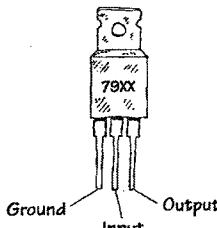


Fixed Regulator ICs

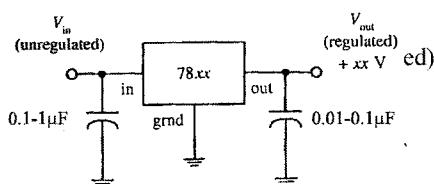
Positive voltage regulator



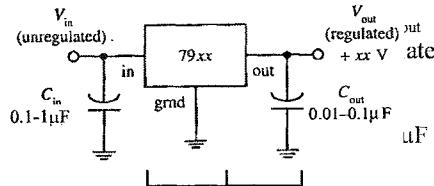
Negative voltage regulator



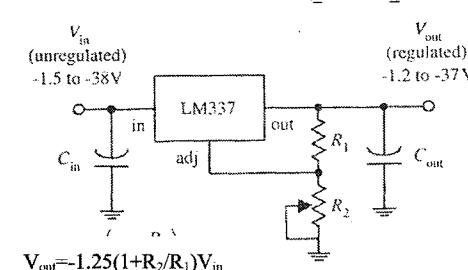
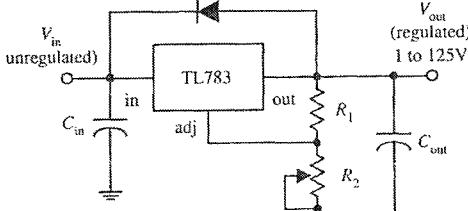
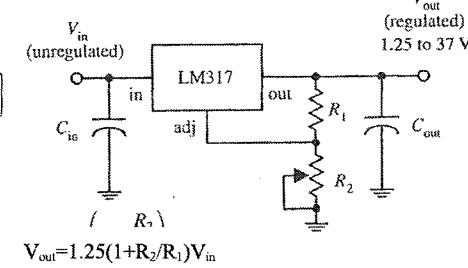
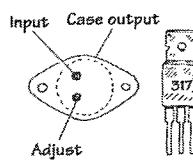
positive voltage regulator



negative voltage regulator

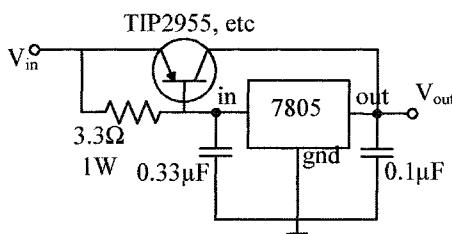


Adjustable Regulator ICs

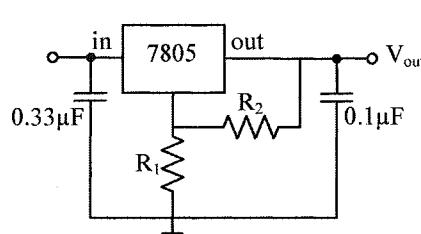


In adjustable regulators, such as LM317, TL783, and LM337T, the reference voltage terminal (called *adjust terminal*) floats. By applying a reference voltage, relative to its input voltage, to the adjust terminal the regulator's output can be altered. The reference voltage is applied by means of a voltage divider (R_1 and R_2). Increasing R_2 forces the adjust voltage upward, and hence pushes the regulator's output to a higher level. The LM317 is designed to accept an unregulated input voltage of up to 37 V, and can output a maximum current of 1.5 A. The TL783 is another positive adjustable regulator that can output a regulated voltage of from 1 to 125 V, with a maximum output current of 700 mA. The LM337T, unlike the previous two regulators, is an adjustable negative voltage regulator. It can output a regulated voltage of from -1.2 to -37 V, with a maximum output current of 1.5 A. C_{in} should be included if the regulator is far from the power source; it should be around 0.1 μ F. C_{out} is used to eliminate voltage spikes at the output; it should be around 0.1 μ F or larger.

The output current of a power supply based on a voltage regulator can be increased using a power transistor such as the 2955 series, as shown in figure below. These transistors can pass several amps quite safely. Further, It is possible to increase the output voltage of a regulator circuit using a pair of voltage-divider resistors (R_1 and R_2 in the previous figure right), or a Zener diode. It is not possible to obtain a voltage lower than the stated rating, i.e., you could not use a 12V regulator to make a 5V power supply, but you could use a 5V regulator to make a 12V supply.



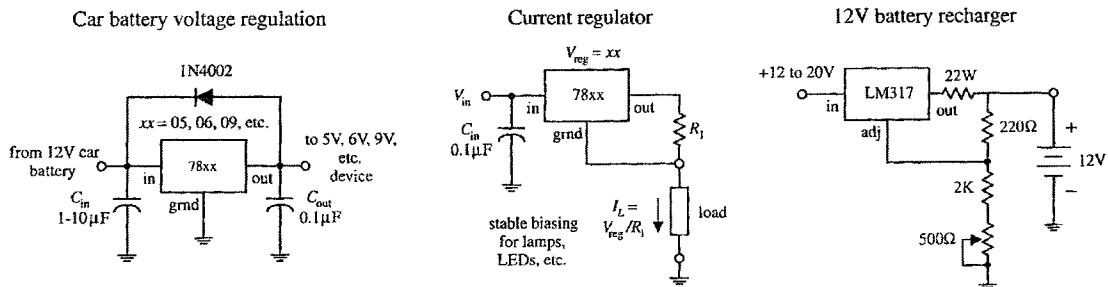
Increasing the Output Current



Increasing the Output Voltage

5.4.4.2 Some Applications

In addition to power supplies, some other applications that use voltage regulators are shown in figure below.



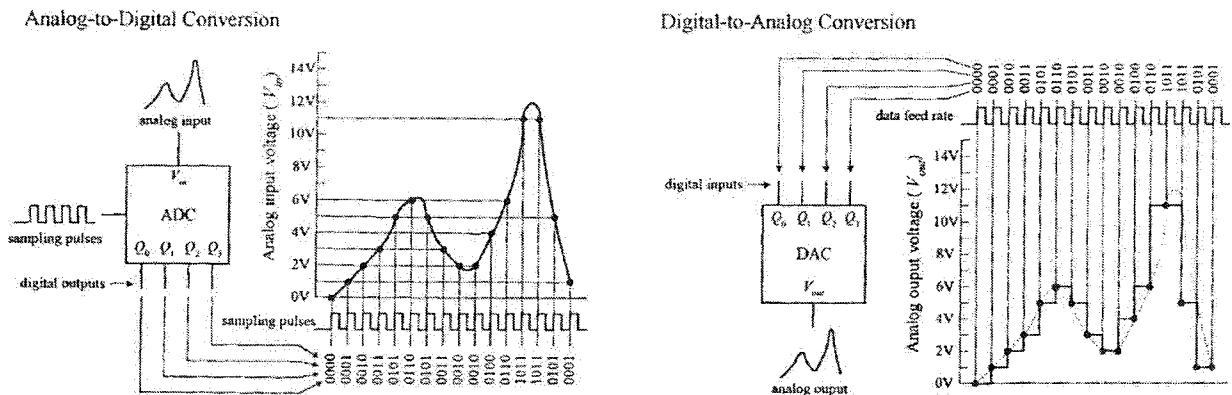
5.4.5 Analog / Digital Signal Conversion

5.4.5.1 ADC and DAC Basics

Figure below shows the basic idea behind analog-to-digital and digital-to-analog conversion. In the analog-to-digital figure (left), the ADC receives an analog input signal along with a series of digital sampling pulses. Each time a sampling pulse is received, the ADC measures the analog input voltage and outputs a 4-bit binary number that is proportional to the analog voltage measured during the specific sample. With 4 bits, we get 16 binary codes (0000 to 1111) that correspond to 16 possible analog levels (e.g., 0 to 15 V).

In the digital-to-analog conversion figure (right), the DAC receives a series of 4-bit binary numbers. The rate at which new binary numbers are fed into the DAC is determined by the logic that generates them. With each new binary number, a new analog voltage is generated. As with the ADC example, we have a total of 16 binary numbers to work with and 16 possible output voltages.

As you can see from the graphs, both these 4-bit converters lack the resolution needed to make the analog signal appear continuous (without steps). To make signals appear more continuous, a converter with higher resolution is used. This



means that instead of using 4-bit binary numbers, we use larger-bit numbers, such as 6-bit, 8-bit, 10-bit, 12-bit, 16-bit, or even 18-bit numbers. If our converter has a resolution of 8 bits, we have $2^8 = 256$ binary number to work with, along with 256 analog steps. Now, if this 8-bit converter is set up to generate 0V at binary 00000000 and 15V at binary 11111111

(full-scale), then each analog step is only 0.058V high ($\frac{1}{256} \times 15\text{V}$). With an 18-bit converter, the steps get incredibly tiny because we have $2^{18} = 262,144$ binary numbers and steps. With 0V corresponding to binary 00000000000000000000 and 15V corresponding to 1111111111111111, the 18-bit converter yields steps that are only 0.000058V high! As you can see in the 18-bit case, the conversion process between digital and analog appears practically continuous.

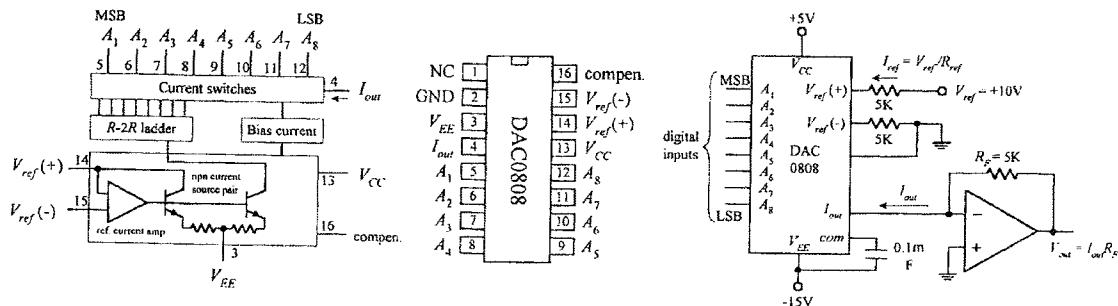
5.4.5.2 Digital-to-Analog Conversion ICs

Digital-to-Analog Converting devices can be purchased from a number of different manufacturers (e.g., National Semiconductor, Analog Devices, Texas Instruments, etc.). The typical resolutions for these ICs are 6, 8, 10, 12, 16, and 18 bits. Most often DAC ICs come with an external reference input that is used to set the analog output range. There are some DACs that have fixed references, but these are becoming rare. Often you will see a manufacturer list one of its DACs as being a *multiplying* DAC. A multiplying DAC can produce an output signal that is proportional to the product of a varying input reference level (voltage or current) times a digital code. As it turns out, most DACs, even those that are not specifically designated as multiplying DAC on the data sheets, can be used for multiplying purposes simply by using the reference input as the analog input. However, many such ICs do not provide the same quality multiplying characteristics, such as a wide analog input range and fast conversion times, as those which are called *multiplying* DACs. Multiplying is most commonly applied in systems that use ratiometric transducers (e.g., position potentiometers, strain gauges, pressure transducers, etc.). These transducers require an external analog voltage to act as a reference level on which to base analog output responses. If this reference level is altered, say, by an unwanted supply surge, the transducer's output will change in response, and this results in conversion errors at the DAC end. However, if we use a multiplying DAC, we eliminate these errors by feeding the transducer's reference voltage to the DAC's analog input. If any supply voltage / current errors occur, the DAC will alter its output in proportion to the analog error. DACs are capable of producing unipolar (single-polarity output) or bipolar (positive and negative) output signals. In most cases, when a DAC is used in unipolar mode, the digital code is expressed in standard binary. When used in bipolar mode, the most common code is either offset binary or 2's complement. Offset binary and 2's complement codes make it possible to express both positive and negative values.

The DAC0808 is a popular 8-bit DAC that requires an input reference current, and supplies 1 of 256 analog output current levels. Figure below shows a diagram of the device, along with its IC pin configuration and a sample application circuit. In the application circuit, the analog output range is set by applying a reference current (I_{ref}) to pin 14 ($+V_{ref}$). In this example, I_{ref} is set to 2 mA via an external +10V / $5k\Omega$ resistor combination. Note that another $5k\Omega$ resistor is required between pin 15 ($-V_{ref}$ and ground.) To determine the DAC's analog output current (I_{out}) for all possible binary inputs, the following formula can be used:

$$I_{out} = I_{ref} \left(\frac{A_1}{2} + \frac{A_2}{4} + \dots + \frac{A_8}{256} \right) = \frac{\text{decimal equivalent of input binary number}}{256}$$

At full scale (all A 's high or binary 255), $I_{out} = I_{ref} (255/256) = (2 \text{ mA})(0.996) = 1.99 \text{ mA}$. Considering that the DAC has 256 analog output levels, we can figure that each corresponding level is spaced $1.99 \text{ mA}/256 = 0.0078 \text{ mA}$ apart. To convert the analog output currents into analog output voltages, we attach the op amp. Using the op amp rules, we find that



the output voltage is $V_{out} = I_{out} \times R_f$. At full scale, $V_{out} = (1.99\text{mA})(5\text{k}\Omega) = 9.95\text{V}$. Each analog output level is spaced $9.95\text{V} / 256 = 0.0389\text{V}$ apart.

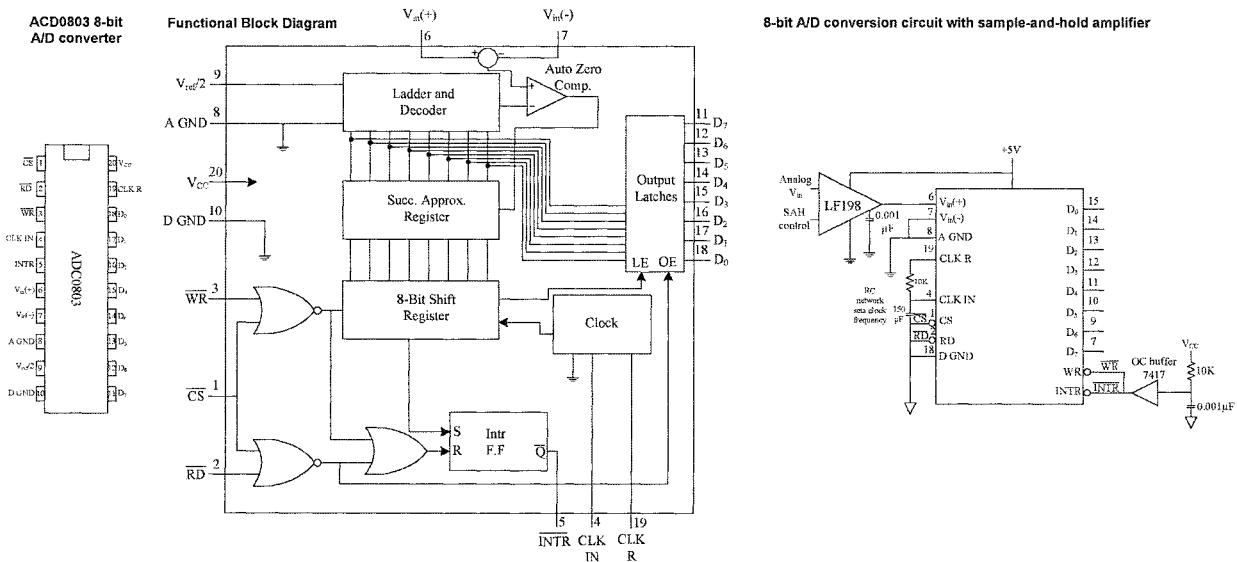
The DAC0808 can be configured as a multiplying DAC by applying the analog input signal to the reference input. In this case, however, the analog input current should be limited to a range from $16 \mu\text{A}$ to 4 mA to retain reasonable accuracy. See the National Semiconductor's data sheets for more details.

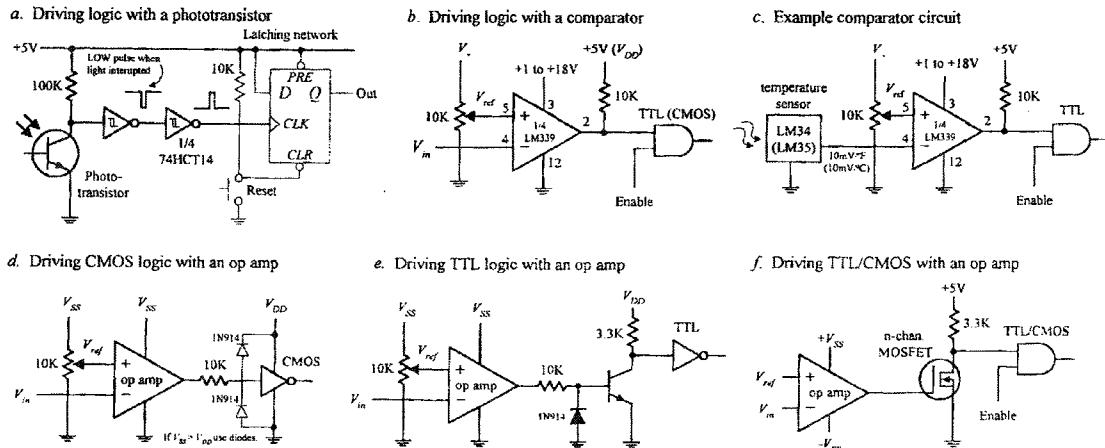
5.4.5.3 Analog-to-Digital Conversion ICs

There are a number of techniques used to convert analog signals into digital signals. The most common approach is called *successive approximation*. In this conversion technique, each bit of the binary output is found, one bit at a time, MSB first. This technique yields fairly fast conversion times (from around 10 to 300 μs) with a limited amount of circuitry.

The ADC0803 is an 8-bit successive approximation type ADC that contains an on-chip clock circuit. Figure below shows the pinouts, functional block diagram, and a typical analog-to-digital conversion circuit. In the example circuit, a conversion is initiated when the \overline{WR} input is set low, while an end-of-conversion is indicated when the \overline{INTR} output goes low. When the \overline{INTR} output is connected directly to the \overline{WR} input, the ADC is set up for continuous conversion. The resistor, capacitor, and buffer act as an automatic reset circuit that is used to ensure that the \overline{WR} input is taken low when power is first applied. The RC circuit connected to the $CLK\text{-R}$ and $CLK\text{-IN}$ inputs acts to set the converter's clock frequency, where $f = 1/1.1 RC$.

The ADC0803 has both a positive analog input (pin 6) and a negative analog input (pin 7). Having two inputs allows for positive, negative, or differential input signals. When working with positive analog input signals, pin 7 is grounded, while pin 6 is used as the analog input. When working with negative analog input signals, pin 6 is grounded, while pin 7 is used as the analog input. Differential signals are based on the voltage difference between pin 6 and pin 7. The $V_{ref}/2$ input is used to determine which analog input voltage will generate the maximum digital output of 11111111 (decimal 255). If this input is left unconnected, the supply voltage (+5 V) sets the analog range from 0 to +5 V. When a voltage is applied to the $V_{ref}/2$ input, a different analog input range can be obtained. For example, to generate a maximum digital output code of 11111111 when an analog input voltage of +4V is applied, a +2V level is set at the $V_{ref}/2$ input ($V_{ref}/2 = +4\text{V} / 2 = +2\text{ V}$). Also included in the example circuit is an LF198 sample-and-hold amplifier. This device is used to prevent any errors due to input signal variations during the conversion process. The LF198 contains two unity-gain non-inverting amplifiers, a logic-controlled switch, and an external capacitor. When the LF198's control input is made high, the internal logic-controlled switch is closed, and the external capacitor charges up quickly to equal the analog input voltage. When the logic-controlled switch is made low, the capacitor will retain the analog input voltage level. This stored voltage level can then be used by the ADC without worrying about analog input variations. You can find many ADCs out there that have the sample-and-hold section built into the IC.



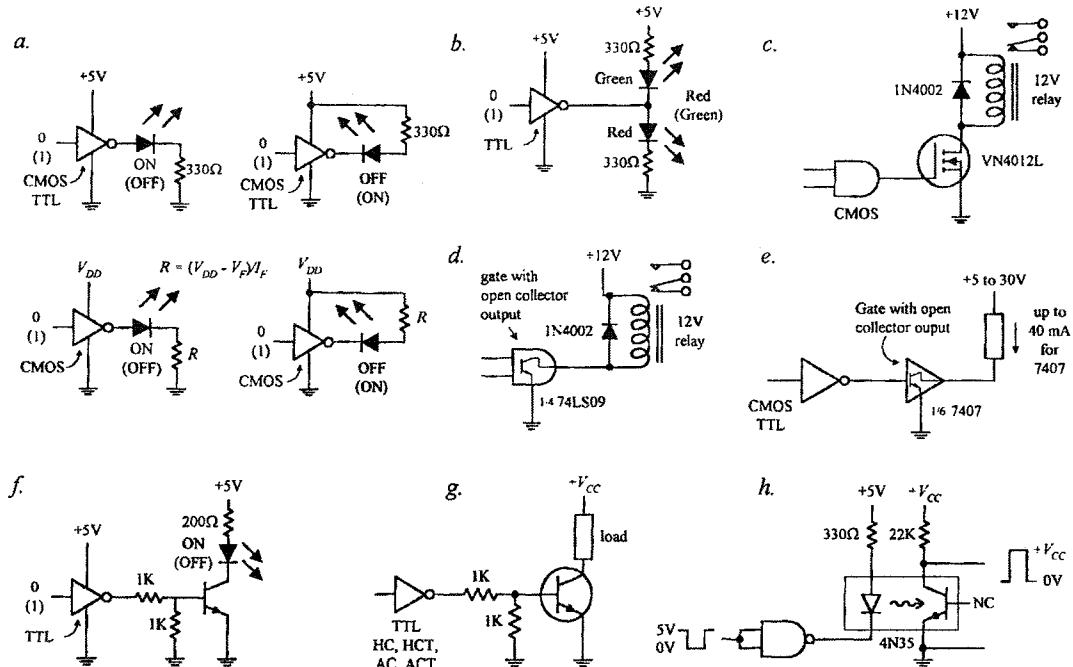


5.4.5.4 Triggering Logic Responses from Analog Signals

There are times when you need to drive logic from simple ON/OFF signals generated by analog devices. For example, you may want to latch an alarm (via a flip-flop) when an analog voltage, say, one generated from a temperature sensor, reaches a desired threshold level. Or you may want to count the number of times a certain analog threshold is reached. For simple ON/OFF applications such as these, it is common to use a comparator or op amp as the interface between the analog output of the transducer and the input of the logic circuit. Often it is possible to simply use a voltage divider network composed of a transducer of variable resistance and a pull-up resistor. Some sample networks are illustrated in figure above.

5.4.5.5 Using Logic to Drive External Loads

Driving simple loads such as LEDs, buzzers, or any device that assumes either an ON or OFF state is straightforward. When driving such loads, it is important to first check the driving logic's current specifications, i.e., how much current, say, a gate can sink or source. After that, you determine how much current is needed to drive the device. If the device draws more current than that the logic can source or sink, a high-power transistor typically can be used as an output switch. Some sample circuits used to drive various loads are shown in figure below.



5.4.6 DC-DC Voltage Conversion

DC power sources are becoming ever more common due to the use of batteries to increase portability. However, power sources such as batteries or USB ports come in a limited availability of voltages. Usually, with the added convenience of portability comes a limit to battery sizes which limit the voltage to low values. For this reason, DC-DC converters are used to transform one voltage to another much like a transformer steps an AC source up or down.

5.4.6.1 Conversion Principles

The objective of DC-DC voltage conversion is to change one voltage to another with maximum efficiency. It is important to note that the energy provided to the converter is the energy that will be available. As expected, no energy is manufactured by the converter. The principles of DC-DC voltage conversion use a very fundamental physical law,

$$P_{in} = P_{out}$$

However, some power is wasted by the converter itself and thus,

$$P_{in} = P_{out} + P_{losses}$$

where P_{in} is the power fed into the converter, P_{out} is the power taken out and P_{losses} is the power wasted by the converter during conversion. With an ideal converter, there would be no wasted power and thus like an ideal transformer,

$$V_{in} \times I_{in} = V_{out} \times I_{out}$$

Rearranging this,

$$\frac{V_{out}}{V_{in}} = \frac{I_{in}}{I_{out}}$$

Therefore, it can be seen that V_{out} and I_{out} are inversely proportional to each other. With an increase in voltage, there is a decrease in current. With a realistic DC-DC converter, wasted energy must be taken into account and thus the efficiency of a converter is given by,

$$\text{Efficiency}(\%) = \frac{P_{out}}{P_{in}} \times 100$$

Using latest components and circuit techniques, modern DC-DC converters achieve an efficiency of over 90%. Most converters achieve atleast 80-85% efficiency which compares with most standard AC transformers.

5.4.6.2 Converter Types

Many types of converters could be found today, each suitable for different types of applications. These can be categorized into two general converters: step-up, and step-down. A second major characteristic is whether there is full dielectric isolation between the input and output of the converters.

Non-isolating converters do not have dielectric isolation between the input and the output. These are usually used where the input output ratio is less than 4:1. There are five main types of non-isolating converters: **buck**, **boost**, **buck-boost**, **Cuk**, and **charge-pump** converters. *Buck* converter is used for step-down while *boost* converter is used for step-up conversion. The *buck-boost* and *Cuk* converters can be used for step-up or step-down but are just voltage polarity reversers or inverters. The *charge-pump* converter is a low power converter for step-up and voltage inversion.

Isolating converters offer dielectric isolation between the input and the output. This is more suitable for some applications as it completely isolates the input from the output. For example, when using a non-isolated converter, noise is created in the input line due to the converter. If it is desired that the input line must also be separately used for another application, the noise could be a problem. Therefore, it is preferred that an isolated converter be used to completely isolate the output line from the input line presenting no electrical interference. Two main types of isolating converters: **flyback type** and **forward type**. Like, non isolating converters these rely on storing energy in the magnetic field of an inductor.

5.4.6.2.1 Buck Converter

A basic schematic for a buck converter is shown to the left. It uses a switching power MOSFET (Q1), flywheel diode (D1), inductor (L), and output filter capacitor (C1). A switching control circuit monitors the output voltage and ensures the voltage is at a fixed level by switching the MOSFET on and off at a fixed operating frequency but a varying duty cycle.

The basic principle of this circuit is a two state process: Q1 on state stores energy in the inductor and Q1 off state transfers that energy to a capacitor via a diode.

First, current begins flowing from the input source through Q1 and L and into C1 when Q1 is turned on. As current flows through the inductor, the magnetic field builds up and begins to store energy. The voltage drop across L opposes or bucks part of the input voltage.

Then, when Q1 is turned off, the inductor begins to supply current to the load via the diode as it opposes any drop in current by suddenly reversing its EMF.

The DC output voltage is a fraction of the input voltage in this configuration and this fraction is the duty cycle of the switching control circuit that controls the MOSFET switch:

$$\frac{V_{out}}{V_{in}} = D$$

where D is the duty cycle and is related by

$$\frac{T_{on}}{T} = D$$

where T is the inverse of the operating frequency. Hence, by varying the duty cycle, the output voltage could be varied as well. For example, a duty cycle of 50% gives a step down ratio of 2:1. This can be used to turn a 24V input voltage to a 12V output voltage. This also means that the current is doubled according to power conservation laws stated earlier in an ideal case.

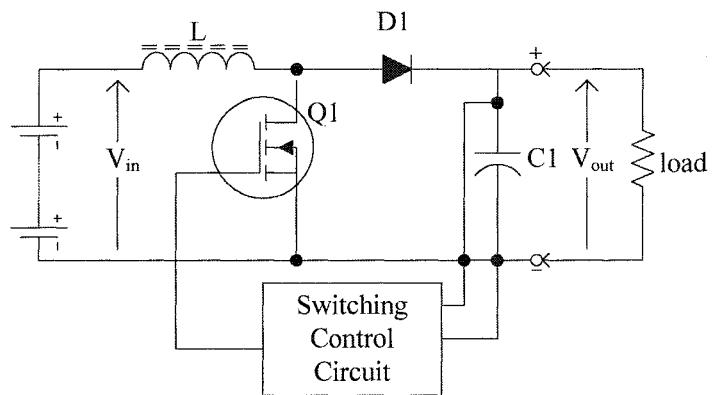
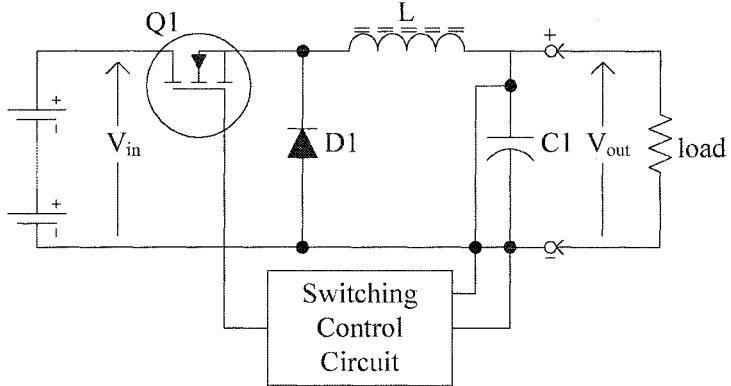
5.4.6.2.2 Boost Converter

The operation of a boost converter is similar to a buck converter. It differs only in the configuration of the components. Q1 is used as a high speed switch which uses the duty cycle to control the output voltage.

First, current begins flowing through L and Q1 when the latter is turned on. During this process, energy is stored in the inductor's magnetic field and current does not flow through D1. In this state, load current is supplied by C1.

Next, when Q1 is turned off, L reverses its EMF immediately and opposes any drop in current. This causes the inductor to add to the source voltage and results in boosting as current flows through L, D1 and the load while recharging C1 as well. Therefore, this results in the output voltage being higher than the input voltage with the ratio given by

$$\frac{V_{out}}{V_{in}} = \frac{1}{(1 - D)}$$



Also given by

$$\frac{V_{out}}{V_{in}} = \frac{T}{T_{off}}$$

For example, to attain a 2:1 voltage boost, a duty cycle of 50% is required. This also results in halving of the current due to the relation stated earlier. It is important to notice again that the energy is always conserved. This circuit can be used to boost low voltage battery or USB power sources to drive higher voltage devices.

5.4.6.2.3 Buck-Boost Converter

This circuit combines the functionality of the above mentioned circuits to provide either a step-up or a step-down output voltage. The configuration is different again to attain this purpose.

First when Q1 is turned on, current flows through the inductor and energy is stored in its magnetic field. D1 is reversed biased and thus it blocks current from flowing into the load. The load is supplied by C1 during this phase.

Next, the inductor reverses its EMF when Q1 is turned off to resist current drop. This generates a voltage to forward bias the diode while charging C1 and supplying current to the load.

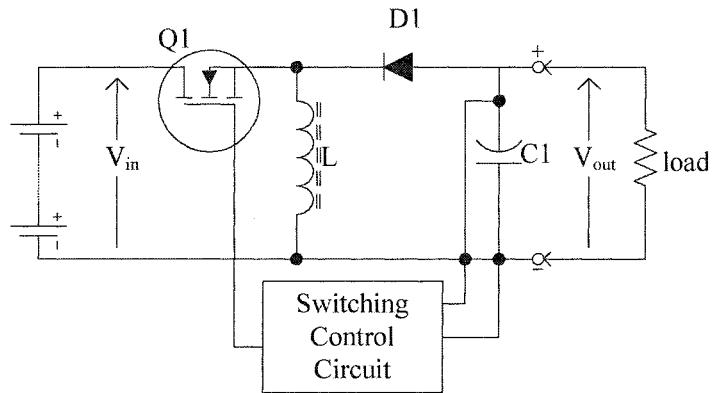
In this configuration, the ratio is,

$$\frac{V_{out}}{V_{in}} = -\frac{D}{(1-D)}$$

Also equivalent to,

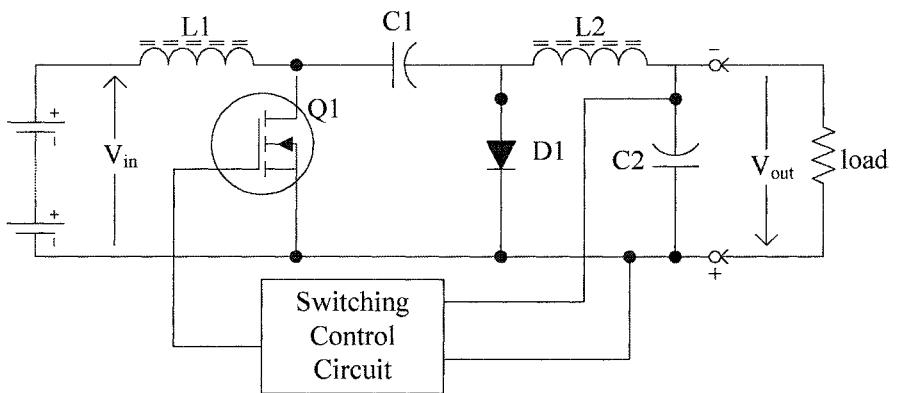
$$\frac{V_{out}}{V_{in}} = -\frac{T_{on}}{T_{off}}$$

If T_{on} is less than T_{off} , voltage is stepped down and the circuit becomes a buck converter and when T_{on} is greater than T_{off} , voltage is stepped up and the circuit becomes a boost converter. However, it is crucial to notice that the polarity is also reversed during this process. Therefore, the input voltage is multiplied by a negative number. Therefore, with a 50% duty cycle, the magnitude of the input voltage and the output voltage remains the same but the polarity is reversed whereby the circuit becomes an inverter.



5.4.6.2.4 Cuk Converter

This converter gets its name from its originator, Slobodan Cuk of Cal Tech university of California. It is similar to the other circuit discussed above with an additional inductor and a capacitor. The configuration delivers an inverted output like the buck-boost converter. But it is important to note that the output current passes through C1 and as ripple current. Therefore, a large electrolytic capacitor with a high ripple current and low ESR (equivalent series resistance) is used



to minimize losses.

First, by turning Q1 on, current begins to flow through L1 and Q1 while storing energy in L1's magnetic field. Q1 is then turned off and the inductor begins to reverse its EMF to maintain current flow. C1 is charged in the process as current flows through L1 and D1. This results in C1 having a higher voltage than V_{in} as energy is transferred from L1 in addition to the input voltage.

During the next cycle, Q1 is turned on. This results in C1 discharging its voltage to the load through L2. C2 is used as a smoothing filter. Simultaneously, energy is stored in L1 in preparation for the next state.

The ratio between the output voltage and the input voltage is given by the relation,

$$\frac{V_{out}}{V_{in}} = -\frac{D}{(1 - D)}$$

which is also equal to,

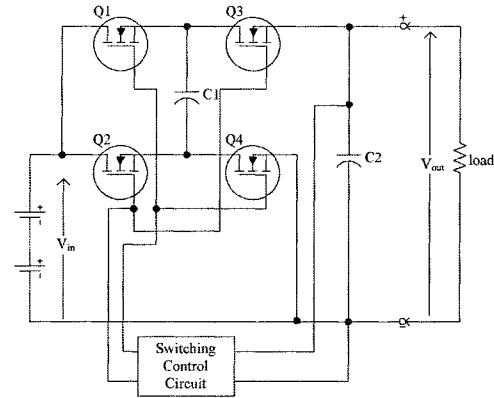
$$\frac{V_{out}}{V_{in}} = -\frac{T_{on}}{T_{off}}$$

This is exactly the same relations for a buck-boost converter – the minus sign indicating polarity reversal. The major advantage to using Cuk converter rather than a buck-boost converter is that the former has a much lower current ripple due to the fact that there are series inductors at both the input and the output. The ripple can be completely eliminated by carefully adjusting the two inductor values.

5.4.6.2.5 Charge-pump Converter

The primary principle of operation for the converters discussed so far dependent on storing energy in the magnetic field of the inductor. Contrarily, charge-pump converter uses a capacitor to store energy as electric charge. These are developed using voltage doubling and voltage multiplying rectifier circuits.

The circuit uses four MOSFET switches and a capacitor (charge bucket) to perform this operation. In the first state, Q1 and Q2 are turned on allowing current to pass from V_{in} , through C1 charging it. In the second state, the switches are turned off while Q2 and Q3 are turned on. This puts C1 in series with the input voltage source and C2. This transfers some charge from C1 to C2 causing it to charge to twice the input voltage. These two states are repeated at a high frequency and current to the load is provided by C2 when Q2 and Q3 are turned off.

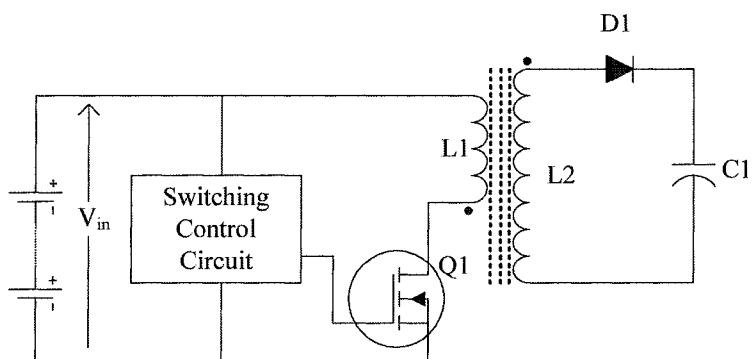


The load attains its energy as ripple current through C1. Hence, it must be a large capacitor with low ESR and tolerance against heavy ripple current. A slight modification of this circuit can result in an inverted output voltage, rather than a doubled voltage.

The main advantage to using a capacitor based converter such as a charge pump converter is that it is cheaper and more compact than inductor based ones. However, it is limited to low current application since a capacitor is being used.

5.4.6.2.6 Flyback Converter

The flyback converter's operation is similar to the buck-boost converter. However, it differs as it uses a transformer to store energy rather than an inductor. At first, Q1 is switched on and this causes current to flow through L1 from the source, storing energy in its magnetic field. Q1 is then switched off causing L1 to reverse its EMF creating a flyback back-EMF pulse. This flyback pulse induced is



stronger in L2 than in Q1 which is chosen to have a very high breakdown voltage. This is because the back EMF pulse is of a greater amplitude and thus Q1 must be able to resist a higher back EMF than forward EMF supplied by the primary source. The current is transferred through the diode to be used by the load and to recharge C1.

The amount of voltage Q1 must withstand is given by,

$$V_p + V_p \left(\frac{N_1 d}{N_2 (1 - d)} \right)$$

where V_p is voltage applied to the primary winding, N_1 is the number of turns in primary winding, N_2 is the number of turns in secondary winding, and d is duty ratio of switch.

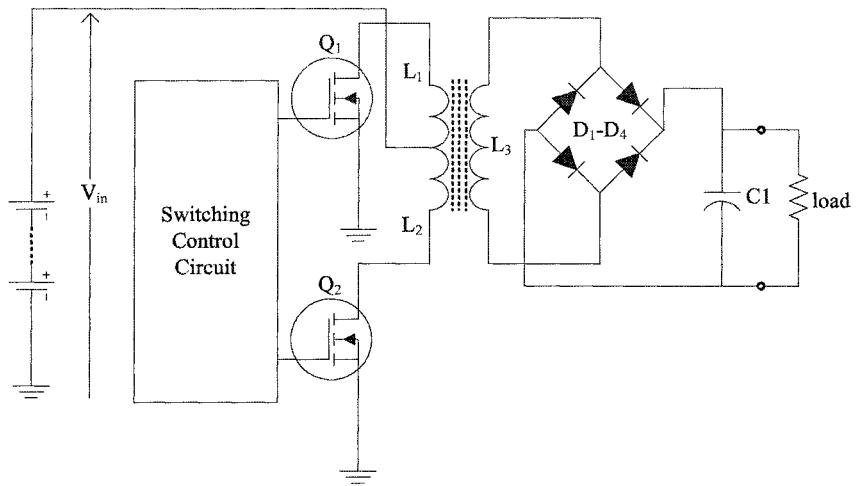
During the first phase, energy stored is the magnetic field of L1. Then in the second phase, energy is transferred to the load and C1 through L2. The ratio between L2 and L1 depends on the number of windings, winding inductance, length of time Q1 is turned on and many other factors. A significant factor is the turn ratio. Usually, flyback converters have a fairly high turns ratio in order to allow for a high voltage step-up ratio.

Due to the way in which flyback converter operates, the magnetic flux never reverses polarity. Therefore, a large transformer core is required even for relatively small power levels. Hence, flyback converters are used for low power applications such as Geiger counter tubes, cathode ray tubes, and generating high voltages for insulation testers.

Further, a third small winding can be added to sense the flyback pulse amplitude which is a reasonable approximation of V_{out} . This information can be used by the switching control to maintain a desired output voltage.

5.4.6.2.7 Forward Converter

This converter differs from its predecessor, the flyback converter in that it is a much more traditional approach – uses one phase to transfer energy between input and output. One of the most common types of forward converter is a push-pull converter shown in the figure. In this configuration the positive end of the source voltage is connected to a center tapping between L1 and L2 while two MOSFET switches are connected to opposite ends of L1 and L2 collectively.



During operation, Q1 and Q2 are alternately turned on. This means that the input voltage flows across L1 and then L2 periodically. This sequence is repeated in tens or hundreds of kilohertz of frequency causing a DC input source to be turned into a high frequency AC square wave. This results in the delivery of an AC square wave with a peak voltage every half-cycle equal to,

$$V_{ac(pk)} = V_{in} \times \frac{L3}{L1}$$

where L3 and L1 are the number of turns. Therefore, if L3 has ten times more windings than L1, then the peak output voltage will be ten times higher than the input voltage. The rectifier circuit (D1-D4) converts the AC voltage into a high voltage DC voltage. This is fed into the load and also is used to charge the capacitor. Ignoring the diode voltage drops, the DC output voltage is given by,

$$V_{out} = V_{in} \times \frac{L3}{L1}$$

In summary, in order to transfer energy via a transformer, the forward converter turns DC energy to AC energy and once transfer is complete, it again rectifies it back to DC to be used by the load. Also, once energy is turned into AC, the

transformer can be used to step-up or step-down the voltage. This can be directly controlled by the winding ratio between L3 and L1. Further, we can attain multiple outputs from the same source by having multiple secondary windings. The forward converter is also more suitable for high power application as it has much less tendency to cause saturation than in the flyback converter. This allows for smaller transformers for the same power level. Applications include car hifi amplifiers, multi-voltage switch mode power supplies, and many others.

5.4.6.3 Efficiency

Efficiency of a DC-DC converter is a crucial criterion. In the real world, energy is always wasted in the converter when going from one voltage to another or during inversion. Losses occur due to inductor resistance, transformer resistance, on resistance in MOSFETs, forward voltage drop in rectifier diodes, eddy current and hysteresis losses in the inductor and transformers, etc. During design, careful attention must be placed upon minimizing these losses to allow for the highest possible efficiency. For example, modern MOSFET switches are the most efficient switches for high DC currents and thus they are a natural choice in the design described above. They are open circuit with they are off and they have a few milliohms of resistance when they are on. This results in little power waste. Furthermore, the diodes used in DC-DC converters are either Shottky or 'hot carrier' metal -semiconductor junction type. Compared to silicon diodes, these diodes have lower forward voltage drop resulting in less power waste.

In order to minimize this drop in forward voltage even more, **synchronous rectification** is used in modern circuit. This is done by using MOSFET switches in place of the diodes and timing it in such a way that it allows forward current but blocks backward current (much like a diode). This can be used during rectification as well. The circuits described above can all use MOSFETs by simply replacing all diodes with a suitable MOSFET switch and turning it on in the same converter phase to allow for forward current to pass through but turning it off in all other phases where forward current is not required to pass through. For example, a buck-type step down converter can use this method to increase the efficiency from 89% (using Shottky diodes) to 94% (using controlled MOSFET switches) even for small voltage outputs.

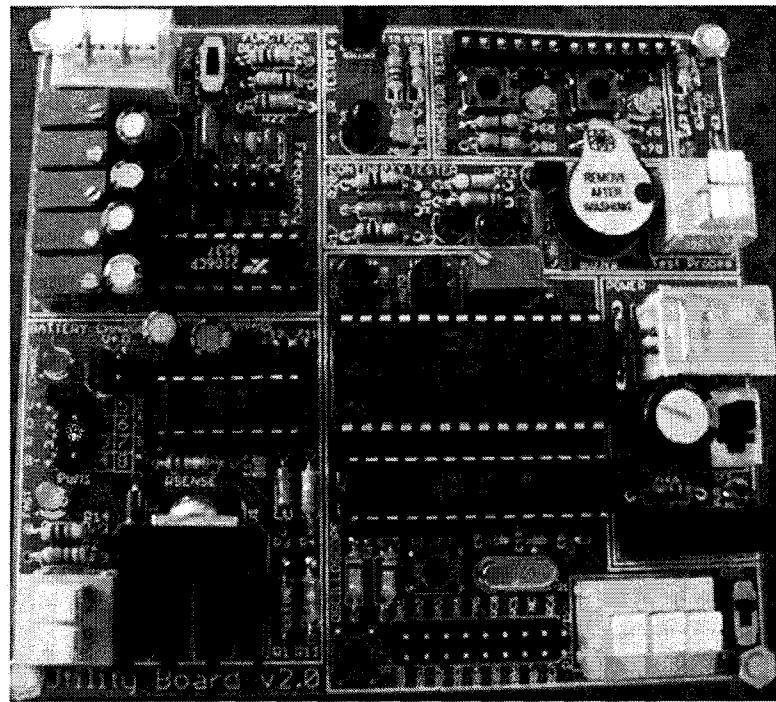
5.4.6.4 Operating Frequency

Modern DC-DC converters operate at a much higher frequency than traditional AC transformers (which operate at 50-60 Hertz). This allows for the use of small inductors, transformers, and capacitors in order to store energy. This allows for more compact design to be used in small devices. However, power losses are great in iron based inductor or transformer core. Thus, ferrite materials are used instead which allows for very efficient operation at many hundreds of kilohertz. Work is being progressed to develop materials and circuitry that work efficiency at very high frequencies.

5.5 The AER201 Utility Board

The AER201 Utility Board is a multifunction Printed Circuit Board (PCB) that aids students in their design projects. It contains seven distinct modules, some of which are connected to each other – a power module, a transistor tester, an infrared emitter tester, a continuity tester, a function generator, a battery charger, and a USB PC oscilloscope. Each module is explained briefly in the following. For more information, please refer to the board user manual.

The power module supplies power for all of other modules including the battery charger. Upon connecting the board to a USB port (PC or a wall outlet), the battery module is powered with +12V. This allows the other modules to be turned off while charging overnight or for extended periods of time. Upon switching on the board, the board is simultaneously supplied with 5V, +12V and -12V to power all other modules. The +12V and -12V is attained from the USB port using a DC-DC converter. This makes it very simple to power the board.



The transistor tester module checks the functionality of NPN or PNP transistors of any pin configuration. It determines in which of three states the transistor is in: **a**) functional; **b**) base-collector short or collector-emitter short; and **c**) base-emitter short.

The IR emitter tester module allows the user to connect an infrared emitter and determine its functionality through an indicator LED. Alternatively, the user may position the Utility Board near an IR emitter in an external circuit, but in either case, the detector on the Utility Board must be within 10 mm distance from the emitter.

The continuity tester tests the connection between traces, wires, and solder points. The tester is rated at 30Ω . Therefore, it detects resistance greater than 30Ω as being a short circuit. When two points are electrically connected, the buzzer beeps indicating a connection. The buzzer is a piezo electric device that produces a high decibels using low current usage.

The function generator module on the Utility Board uses the XR2206 controller to produce square, triangular, or sinusoidal waves of up to 100kHz or as low as 1Hz, with a tested range of -10V to +10V. Using jumpers and trimmers, one can adjust offset, frequency, duty cycle, and amplitude. One common use for a function generator is to produce a clock signal for the Driver Board, for example.

The battery charger module allows the user to charge between 1 and 8 NiMH, and NiCd rechargeable battery cells, with a typical charge time of 3 hours.

The oscilloscope module's primary function is a dual channel PC oscilloscope with a maximum sampling rate of 30kHz. It contains a PIC18F2455 microcontroller which runs as an HID USB device and interfaces to a Windows PC. The secondary function of this module is as a regular PIC driver. 18 input/output pins are included in the module to allow the user to run custom code on the PIC, with the option of communicating with a PC through the USB interface for advanced users. The mounted oscillator clock frequency is 20 MHz.

5.6 Practical Notes

5.6.1 Device Safety and Handling Precautions

Scuffing across a carpet while wearing sneakers on a dry day can result in a transfer of electrons from the carpet to your body. In such a case, it is entirely possible that you will assume a potential of 1000 V relative to ground. Handling a polyethylene bag can result in static voltages of 300 V or more, whereas combing your hair can result in voltages as high as 2500 V. The drier the conditions (lower the humidity), the greater the chance is for these large voltages to form. The amount of electrostatic discharge that can result from an electrostatically charged body coming in contact with a grounded object is not of much concern, in terms of human standards. However, the situation is entirely different when subjecting certain types of semi-conductive devices to similar discharges. Devices that are particularly vulnerable to damage include field-effect transistors, such as MOSFETs and JFETs. For example, a MOSFET can be destroyed easily if an electrostatically charged individual touches its gate; the gate-channel breakdown voltage will be exceeded, and a hole will be blown through the insulator that will destroy the transistor. The vulnerability of some common devices can be summarized as follows:

Extremely Vulnerable: MOS transistors, MOS ICs, junction FETs, laser diodes, microwave transistors, metal film resistors.

Moderately Vulnerable: CMOS ICs, LS TTL ICs, Schottky TTL ICs, Schottky diodes, linear ICs.

Somewhat Vulnerable: TTL ICs, small signal diodes and transistors, piezoelectric crystals.

Not Vulnerable: Capacitors, carbon-composite resistors, inductors, and many other analog devices

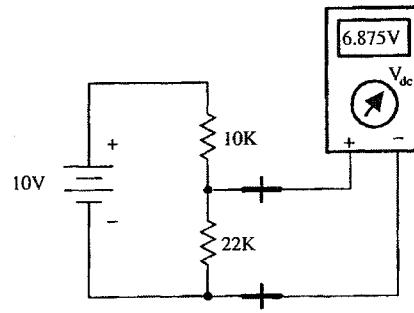
Devices that are highly vulnerable to damage are often marked with "Caution, components subject to damage by static electricity." If you see a label like this, use the following precautions:

- Store components in their original packages, in electrically conductive containers (e.g., metal sheet, aluminum foil), or in conductive foam packages.
- Do not touch leads of ESD-sensitive components.
- Discharge the static electricity on your body before touching components by touching a grounded metal surface such as a water pipe or large appliance.
- Never allow your clothing to make contact with components.
- Ground tabletops and soldering irons (or use a battery-powered soldering iron). You should also ground yourself with a conductive wrist guard that is connected with a wire to ground.
- Never install or remove an ESD-sensitive component into or from a circuit when power is applied. Once the component is installed, the chances for damage are greatly reduced.

5.6.2 Basic Measurements

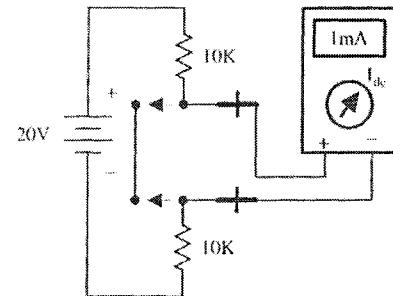
5.6.2.1 Measuring Voltages

For measuring voltages with a Volt-Ohm Multimeter (VOM), turn the selector knob to the voltage setting. If you want to measure a DC voltage, the knob is turned to the appropriate DC voltage-level setting. If you wish to measure an AC voltage, the knob is turned to the AC voltage setting (V_{AC} or V_{rms}). Note that the displayed voltage in the V_{AC} setting is the rms voltage ($V_{rms} = 0.707 V_{peak-to-peak}$). Once the VOM is set correctly, the voltage between two points in a circuit can be measured by touching the VOM's probes to these points (the VOM is placed in parallel). For example, figure right shows the procedure used to measure the voltage drop across a resistor.



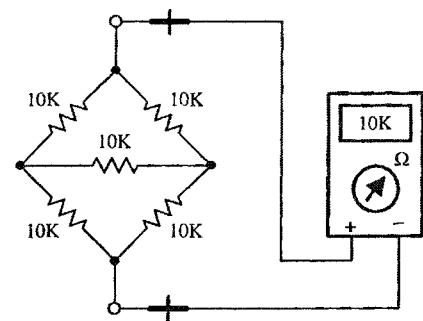
5.6.2.2 Measuring Currents

Measuring currents with a VOM is almost as easy as measuring voltages. The only difference (besides changing the setting) is that you must break the test circuit at the location where you wish to make a current reading. Once the circuit is open, the two probes of the VOM are placed across the break to complete the circuit (VOM is placed in series). Figure right shows how this is done. When measuring AC currents, the VOM must be set to the rms current setting.



5.6.2.3 Measuring Resistance

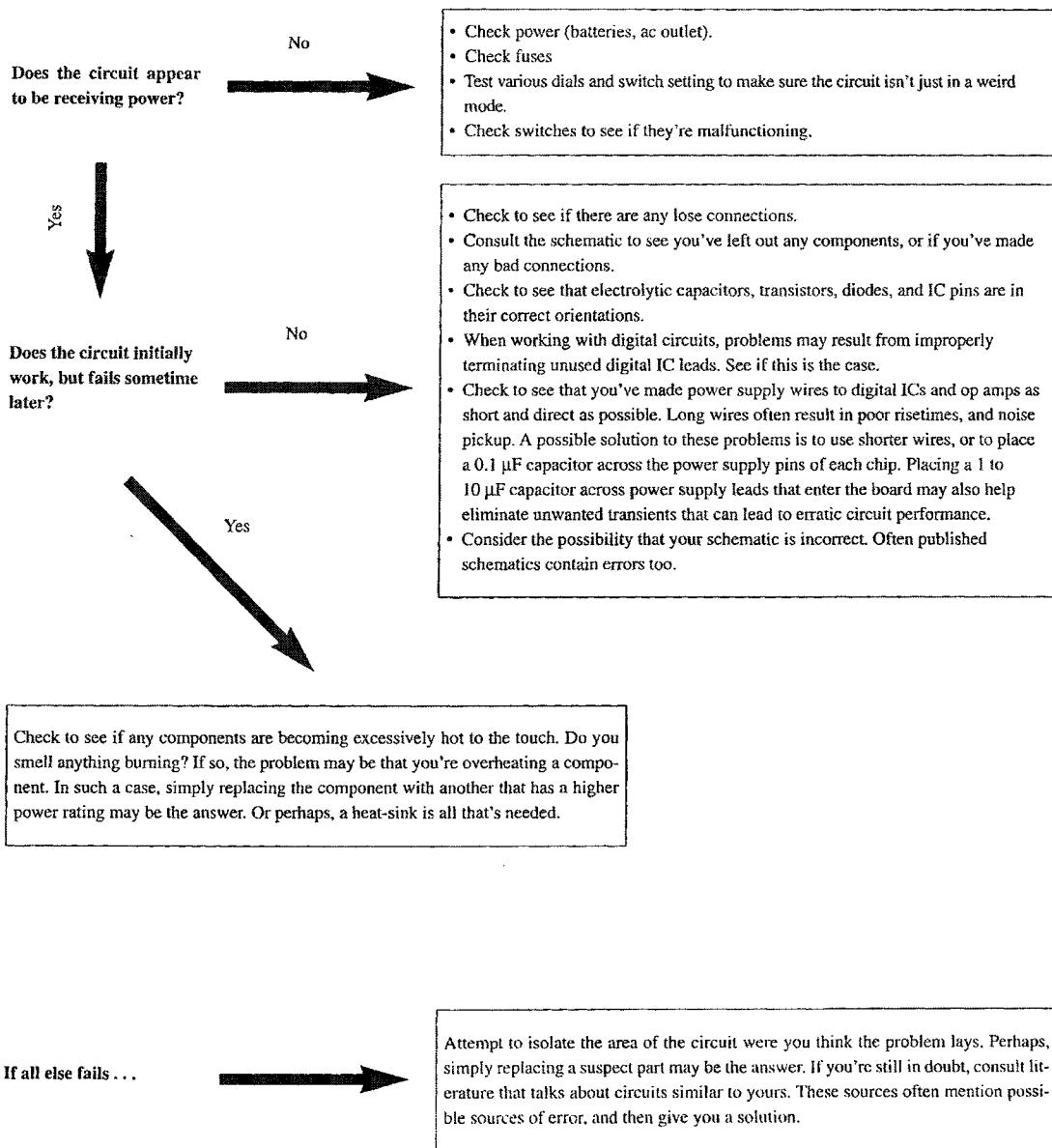
Measuring resistances with a VOM is simple enough; remove the power to the resistive section of interest, and then place the VOM's probes across this section. Of course, make sure to turn the VOM selector knob to the ohms setting beforehand.



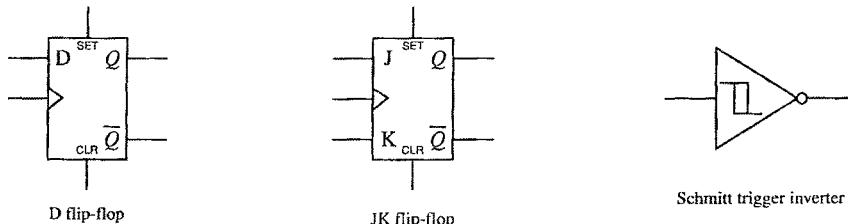
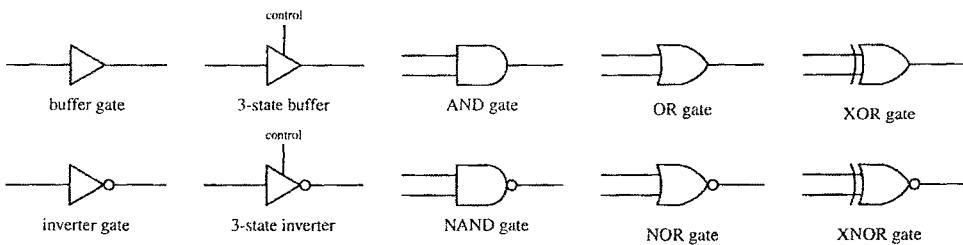
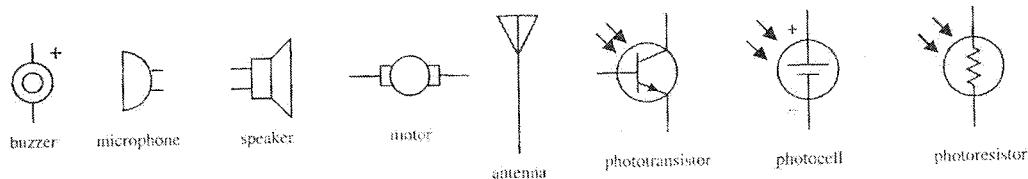
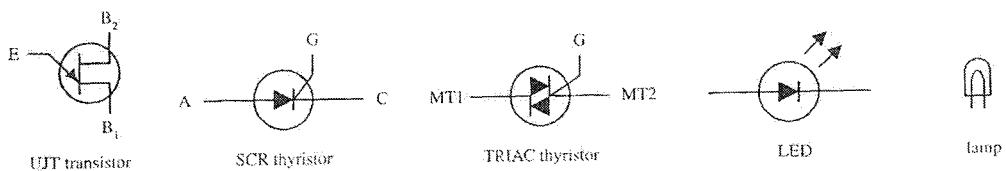
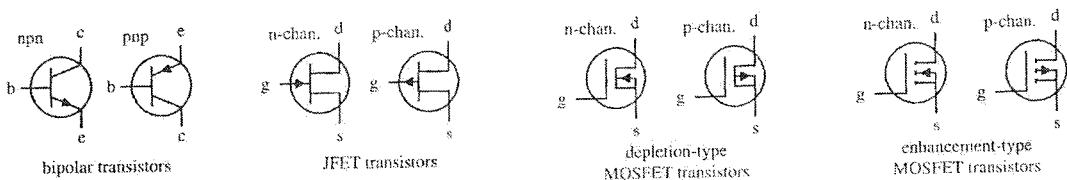
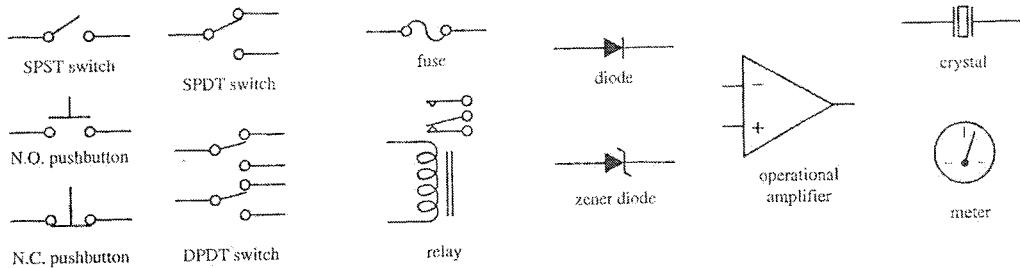
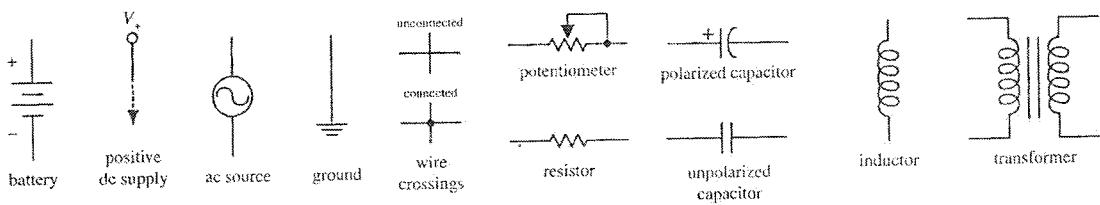
5.6.2.4 Measurement Errors

When measuring the current through (or voltage/resistance across) a load, the reading obtained from the VOM will always be different when compared with the true value present before the meter was connected. This error comes from the internal resistance of the VOM. For each setting (ammeter, voltmeter, ohmmeter), there will be a different internal resistance. A real ammeter typically will have a small input resistance of around $2\text{ k}\Omega$, while a voltmeter may have an internal resistance of $100\text{ k}\Omega$ or more. For an ohmmeter, the internal resistance is usually around $50\text{ }\Omega$. It is crucial to know these internal resistances in order to make accurate measurements. To minimize the percentage error, an ammeter's input resistance should be less than the equivalent resistance of the original circuit by 20 times or more. Conversely, a voltmeter should have an input resistance that is larger than the equivalent resistance of the original circuit by 20 times or more. The same goes for the ohmmeter; it should have an input resistance that is at least 20 times the equivalent resistance of the original circuit. By following these simple rules, it is possible to reduce the error to below 5 percent. Another approach (perhaps a bit more tedious) is to look up the internal resistances of your VOM, make your measurements, and then add or subtract the internal resistance afterwards.

5.6.3 Troubleshooting the Circuits



5.6.4 Electronic Symbols



5.6.5 Device Data

Standard Resistance Values for 5% Carbon-Film Resistors

Ω			$K\Omega$				$M\Omega$				
1.0	8.2	33	120	470	1.8	6.8	27	100	390	1.5	6.2
1.1	9.1	36	130	510	2.0	7.5	30	110	430	1.6	6.8
1.2	10	39	150	560	2.2	8.2	33	120	470	1.8	7.5
1.3	11	43	160	620	2.4	9.1	36	130	510	2.0	8.2
1.5	12	47	180	680	2.7	10	39	150	560	2.2	9.1
1.6	13	51	200	750	3.0	11	43	620	620	2.4	10
1.8	15	56	220	820	3.3	12	47	680	680	2.7	

Selection of Zener Diodes

Voltage (Volts)	Power (W)							
	0.25	0.4	0.5	1.0	1.5	5.0	10.0	50.0
1.8	1N4614							
2.0	1N4615							
2.2	1N4616							
2.4	1N4617	1N4370,A						
3.0	1N4619	1N4372,A	1N5987					
3.3	1N4620	1N5518	1N5988	1N3821	1N5913	1N5333,B		
3.6	1N4621	1N5519	1N5989	1N3822	1N5914	1N5334,B		
3.9	1N4622	1N5520	1N5844	1N3823	1N5915	1N5335,B	1N3993,A	1N4549,B
4.7	1N4624	1N5522	1N5846	1N3825	1N5917	1N5337,B	1N3995,A	1N4551,B
5.6	1N4626	1N5524	1N5848	1N3827	1N5919	1N5339,B	1N3997,A	1N4553,B
7.5	1N4100	1N5527	1N5997	1N3830	1N3786	1N5343,B	1N4000,A	1N4556
10.0	1N4104	1N5531	1N6000	1N4740	1N3789	1N5347,B	1N2974,A	1N2808,B
12.0		1N5532	1N6002	1N3022,B	1N3791	1N5349,B	1N2976,A	1N2810,B
14.0	1N4108	1N5534	1N5860			1N5351,B	1N2978,A	1N2812,B
16.0	1N4110	1N5536	1N5862	1N3025,B	1N3794	1N5353,B	1N2980,A	1N2814,B
20	1N4114	1N5540	1N5866	1N3027,B	1N3796	1N5357,B	1N2984,A	1N2818,B
24	1N4116	1N5542	1N6009	1N3929,B	1N3798	1N5359,B	1N2986,A	1N2820,B
28	1N4119	1N5544	1N5871			1N5362,B		
60	1N4128		1N5264,A,B			1N5371,B		
100	1N4135	1N985	1N6024	1N3044,A,B	1N3813	1N5378,B	1N3005,B	1N2838,B
120		1N987	1N6026	1N3046,B	1N5951	1N5380,B	1N3008A,B	1N2841,B

Selection of Diodes

Device	Type	Material	Peak Inverse Voltage (V)	Average Forward Current (mA)	Surge Current	Forward Voltage Drop (V)
1N34	Signal	Ge	60	8.5		1.0
1N34A	Signal	Ge	80			1.0
1N60	Signal	Ge	80	0		1.0
1N67A	Signal	Ge	100	4.0		1.0
1N191	Signal	Ge	90	5.0		1.0
1N914	Fast Switch	Si	75	75	0.05	1.0
1N4001	Rectifier	Si	50	1000	30	1.1
1N4002	Rectifier	Si	100	1000	30	1.1

1N4003	Rectifier	Si	200	1000	30	1.1
1N4004	Rectifier	Si	400	1000	30	1.1
1N4005	Rectifier	Si	600	1000	30	1.1
1N4006	Rectifier	Si	800	1000	30	1.1
1N4007	Rectifier	Si	1000	1000	30	1.1
1N4148	Signal	Si	75	10		1.0
1N4152	Fast Switch	Si	40	20		1.0
1N5400	Rectifier	Si	50	1000	200	
1N5401	Rectifier	Si	100	3000	200	
1N5402	Rectifier	Si	200	3000	200	
1N5404	Rectifier	Si	400	3000	200	
1N5406	Rectifier	Si	600	3000	200	
1N5408	Rectifier	Si	1000	3000	200	
1N4448	Signal	Si	75		0.72	0.72
1N5817	Schottky	Si	20	1000	25	0.45
1N5818	Schottky	Si	30	1000	25	0.55
1N5819	Schottky	Si	40	1000	25	0.60
SB1100	Schottky	Si	100	1000	30	0.85
1N5820	Schottky	Si	20	3000	80	0.475
1N5821	Schottky	Si	30	3000	80	0.400
1N5822	Schottky	Si	40	3000	150	0.525
P600A	Rectifier	Si	50	6000	400	
P600B	Rectifier	Si	100	6000	400	
P600D	Rectifier	Si	200	6000	400	
P600M	Rectifier	Si	1000	6000	400	

General Purpose Bipolar Transistors

Device	Type	V _{CEO} MAX (V)	V _{CBO} MAX (V)	V _{EBO} MAX (V)	I _C MAX (mA)	P _D (W)	I _C MAX (mA)	I _C MAX (mA)	F _T
2N918	NPN	15	30	3.0	50	0.2 (3mA)	20		600
2N2102	NPN	65	120	7.0	1000	1.0	20	40	60
2N2219	NPN	30	60	5.0	800	3.0	35	100	250
2N2219A	NPN	40			150			100 (min)	300
2N2222	NPN	30	60	5.0	800	1.2	35	100	250
2N2222A	NPN	40			150			100 (min)	250
2N2484	NPN	60			50			100 (min)	60
2N2857	NPN	15			40			30 (min)	1200
2N2905	PNP	40	60	5.0	600	0.6	35		200
2N2907	PNP	40	60	5.0	600	0.4	35		200
2N3019	NPN	80			1000			100 (min)	100
2N3053	NPN	40	60	5.0	700	5.0	0	50	100
2N2904	NPN	40	60	6.0	200	0.625	40		300
2N2907A	PNP	40			600			75 (min)	200
2N3439	NPN	350			1000			40 (min)	15
2N3467	PNP	40			1000			40 (min)	175
2N3704	NPN	30			600			100 (min)	250
2N3904	NPN	40			10			100 (min)	200
2N3906	PNP	20			10			100 (min)	250
2N3906	PNP	40	40	5.0	200	1.5	60		250

2N4037	PNP	40	60	7.0	1000	5.0		50	
2N4125	PNP	30			200			50 (min)	200
2N4126	PNP	25			200			120 (min)	200
2N4401	NPN	40	60	6.0	600	0.625	20	100	250
2N4403	PNP	40	40	5.0	600	0.625	30	100	200
2N4400	NPN	40			600			20 (min)	200
2N5087	PNP	50			50			40 (min)	40
2N5088	NPN	30			50			50 (min)	50
2N5415	PNP	200	200	4.0	1000	10.000		30	15

General Purpose Power Bipolar Transistors

Device	Type	V _{CEO} MAX (V)	I _C MAX (A)	P _D (W)	H _{FE} (min)	F _T (MHz)
2N5172	NPN	25	10	360	100	100
2N5210	NPN	50	50	625	250	30
2N5307	NPN-D	40	0.3	400	2000	60
2N5308	NPN-D	40	0.3	400	7000	60
2N5400	PNP	120	0.6	625	30	100
2N5401	PNP	150	0.6	625	40	100
2N5415	PNP	260	50	5	30	15
2N6036	PNP-D	80	4	40	750	25
2N6038	NPN-D	60	4	40	750	25
2N6043	NPN-D	60	8	75	1000	4
2N6052	PNP-D	100	12	150	750	4
2N6284	NPN-D	100	20	160	750	4
2N6287	PNP-D	100	20	160	750	4
2N6388	NPN-D	80	10	65	1000	20
2N6668	PNP-D	80	10	65	1000	20
TIP29C	NPN	100	1.0	30	40	
TIP30C	PNP	100	1.0	30	40	
TIP31C	NPN	100	3	40	25	
TIP32C	PNP	100	3	40	25	
TIP35C	NPN	100	25	125	25	
TIP36C	PNP	100	25	125	25	
TIP102	NPN-D	100	8	80	2500	
TIP107	PNP-D	100	8	80	2500	
TIP110	NPN-D	60	4	60	500	
TIP112	NPN-D	100	4	50	500	
TIP117	PNP-D	100	4	50	500	
TIP120	NPN	60	5	65	1000	>5
TIP142	NPN-D	100	10	125	500	
TIP147	PNP-D	100	10	125	500	
D4H11	NPN	80	10	50	40	50
D44H10	PNP	80	15	83	40	50
MJ11015	PNP-D	120	30	200	1000	4
MJ11016	NPN-D	120	30	200	1000	4
MH11032	NPN-D	120	50	300	1000	
MJ11033	PNP-D	120	50	300	1000	
MJE13007	NPN	400	8	80	8	4

MJE200	NPN	25	5	15	70	65
MJE210	PNP	20	5	15	70	65

Small-Signal JFETs

Device	Type	BV _{GS} (V)	MAX V _{GS} , OFF (V)	INPUT C (pF)	MAX I _D (mA)	APPLICATION
2N4338	N-JFET	50	-1	6	0.6	
2N4416	N-JFET	30	-6	4	15	VHF/UHF amp, mix, osc.
2N5114	P-JFET	30	10	25	90	Switch: R _{on} =75Ω (max)
2N5265- 2N5270	N-JFET	60	-3	7	1	Series of 6, 2N5358-64 P-JFET complement
2N5432	N-JFET	25	-10	30		Switch: R _{on} =5Ω (max)
2N5358- 2N5364	P-JFET	40	3	6	1	2N5265-70; N-JFET complement
2N5364		40	8	6	18.6	
2N5457- 2N5459	N-JFET	25	-6	7	5	General purpose, 2N5460-2 P-JFET complement
2N5459		25	-8	7	16	
2N5460- 2N5462	P-JFET	40	6	7	5	2N5457-9; N-JFET complement
2N5462		40	9	7	16	
2N5484	N-JFET	25	-6	5	30	HF/VHF/UHF amp, mix, osc.
MPF106						
2N5486	N-JFET	25	-2	5	15	VHF/UHF amp, mix, osc.
U304	P-JFET	30	10	27	50	Analog switch copper common-gate VHF/UHF
U310	N-JFET	30	-6	2.5	60	Amp
U350	Quad N-JFET	25	-6	5	60	Matched JFETs

Power JFETs

Device	Type	BV _{GS} (V)	R _{DS(ON)} , MAX (Ω)	V _{GS} MAX (V)	I _{GS} MAX (A)	P _D	CASE
IRFZ30	N-channel	50	0.050	4	30	75	TO-220
IRFZ42	N-channel	50	0.035	4	50	150	TO-220
VN0610L	N-channel	60	6	2.5	0.27		TO-92
VN10KM	N-channel	60	6	2.5	0.3	1	TO-237
IR511	N-channel	60	0.6	4	2.5	20	TO-220AB
MTP2955E	P-channel	60	0.12		11.5	125	TO-220AB
ZVN2110B	N-channel	100	4		0.85	5	TO-39
ZVN3310B	P-channel	100	20		0.3	5	TO-39
IRF510	N-channel	100	0.6	4	2	20	TO-220AB
IRF520	N-channel	100	0.27	4	5	40	TO-220AB
IRF150	N-channel	100	0.055	4	40	150	TO-204AE
ZVN0120B	N-channel	200	16		0.42	5	TO-39
ZVN1320B	P-channel	200	80		0.1	5	TO-39
IRF620	N-channel	200	0.8	4	5	40	TO-220AB
IRF220	N-channel	200	0.4	4	8	75	TO-220AB
IRF640	N-channel	200	0.18	4	10	125	TO-220AB
VP1320N3	P-channel	200	0.6	3.5	0.15		TO-92
IR9640	P-channel	200	0.5	4	11		TO-220
IR820	N-channel	500	3	4	2.5		TO-220
VP0650N3	P-channel	500	25	4	0.1		TO-92

Infrared Emitters

Device	Type	MIN. OUTPUT mW\cm ²	If (mA)	Vf (V)	BEAM ANGLE (degrees)	CASE STYLE
OP131	GaAs	3.0*	100	1.75	18	CAN-4.7
OP132	GaAs	4.0*	100	1.75	18	CAN-4.7
OP133	GaAs	5.0*	100	1.75	18	CAN-4.7
OP133 W	GaAs	5.0*	100	1.75	50	CAN-4.7
OP140A	GaAs	0.4	20	1.60	40	SIDE LENS
OP140C	GaAs	0.2	20	1.60	40	SIDE LENS
OP169A	GaAs	0.18	20		46	END LOOKER/LENS
OP240A	GaAlAs	0.6	20	1.80	40	SIDE LENS
OP240B	GaAlAs	0.4	20	1.80	40	SIDE LENS
OP265A	GaAlAs	2.7	20	1.80	18	T1.050 FLAT
OP290A	GaAlAs	210.0	1500	4.0	50	T1 ¾ DOME
OP293A	GaAlAs	16.0	100	2.0	60	TO 46 (Plastic)
OP295B	GaAlAs	33.0	1500	4.0	20	T1 ¾ DOME
OP296C	GaAlAs	1.6	100	2.0	20	T1 ¾ DOME

Photosensors

Device	Light current Ic (mA)		Test VCE	Respons e Ee	Response Angle (degrees)	Case Style
	Min	Max				
OP505A	4.30		5.0	.5	18	T1
OP509A	5.70		5.0	5.0	50	END LOOKER/LENS
OP550A	2.25		5.0	1.0	60	LATERNAL EXT. LENS
OP550C	0.25	1.55	5.0	1.0	60	LATERNAL EXT. LENS
OP593A	3.0		5.0	1.7	130	TO18 (Plastic)
OP598B	5.0	10.0	5.0	1.7	25	TO18 (Plastic)
OP804SL	7.0	22.0	5.0	5.0	25	TO18 (Plastic)
OP805SL	15.0		5.0	5.0	25	TO18 (Plastic)

555 Timers

Type	Supply Voltage (V)		Supply Current (μ A) (Vcc=5V)		Trig Current (nA) (Thres. Current)		Typical Freq. (MHz)	Iout,max (mA) (Vcc=5V)	
	Min	Max	Typ.	Max.	Typ.	Max.		Source	Sink
SN555	4.5	18	3000	5000	100	500	0.5	200	200
ICL7555	2	18	60	300		10	1	4	25
TLC555	2	18	170		0.01		2.1	10	100
LMC555	1.5	15	100	250	0.01		3		
NE555	4.5	15		6000					200

Op Amps

Type	Per Pkg.	Supply Voltage (V)		Input Offset, Max. (nA)	Input Offset, Typ. (μV)	Slew Rate (V/μS)	Freq., Typ. (MHz)	Max. Output (mA)	Comments
		Min.	Max.						
324A	4	3	32	30	2	0.5	1	20	Popular, single-supply bipolar
349	4	10	36	50	1	2	4	15	
355B	1	10	36	0.2	3	5	2.5	20	JFET, popular
741	1	10	36	200	2	0.5	1.2	20	Classic chip; 1458 (quad), 348 (dual)
1436	1	10	80	10	10	2	1	10	High-voltage
1463		30	80			165	17	1000	High-voltage
4558	2	8	36	200	2	1	2.5	15	Bipolar
AD841K	1	10	36	200	0.5	300	40	50	Bipolar, high-speed
AD848J	1	9	36	15	0.5	300	250	25	Bipolar, high-speed
AD744C	1	9	36	0.02	0.1	75	13	20	High-speed, low distortion JFET
CA3410A	4	4	36	0.01	3	10	5.4	6	High-Speed MOSFET
HA5141A	1	2	40	10	0.5	1.5	0.4	1	Low-power, single-supply bipolar
HA2541	1	10	35	7μA		280	40	10	High-speed, low distortion
HA2542	1	10	35	7μA		375	120	100	High-power
HA2544	1	10	33	2μA	6	150	33	35	High-speed
HA5151	1	2	40	30	2	4.5	1.3	3	Bipolar
ICL7641B	4	1	18	0.03		1.6	1.4	5	MOSFET, low voltage, general purpose
LF351	1	10	36	0.1	5	13	4	10	JFET, 353=ual, 247=quad
LF411	1	10	36	0.1	0.8	15	4	15	General purpose, low noise JFET
LM10	1	1	45	0.7	0.3	0.12	0.1	20	Low voltage, precision
LM11	1	5	40	10pA	0.1	0.3	0.5	2	Precision, low bias
LM12	1	20	80			9	0.7	10000	High-power
LM308	1	10	36	1	2	0.15	0.3	5	Precision, low-bias bipolar
LM312	1	10	40	1	2	0.15	0.3	5	Compensated 308
LM318	1	10	40	200	4	7	15	10	Classic Chip
LM343	1	10	68	10	2	2.5	1	10	High-voltage
LM344	1	10	68	19	2	30	10	10	High-voltage
LM833	2	10	36	200	0.3	7	15	10	Bipolar
LM6364	1	5	36	2μA	2	300	160	30	High-speed, bipolar
LT1006A	1	2.7	44	0.5	0.04	0.4	1	20	Bipolar, single-supply, precision
LT1028A	1	8	44	50	0.01	15	75	20	Bipolar precision, low-noise
LT1013C	2	4	44	20	0.06	0.4	0.8	25	Bipolar, single-supply

MC33078	2	10	36	150	0.15	7	16	20	Bipolar
MC34071A	1	3	44	50	0.5	10	4.5	25	Bipolar
MC34181	1	3	36	0.05	0.5	10	4	8	JFET, high-speed, low power, low distortion
NE5534	1	6	44	300	0.5	6	10	20	Bipolar
OP-07E	1	6	44	3.8	0.03	0.17	0.6	10	Precision, low noise
OP-37E	1	8	44	35	0.01	17	63	20	Precision, low noise
OP-77E	1	6	44	1.5	0.01	0.3	0.6	12	Improvement over OP-27
OP-90E	1	1.6	36	3	0.05	0.01	0.02	6	Low-power
OP-97E	1	4.5	40	0.1	0.01	0.2	0.9	10	Low-power
TL051C	1	10.5	36	0.1	0.6	24	3	30	JFET, high-speed, low power, low distortion
TL061C	1	4	36	0.2	3	3.5	1	5	JFET, low-power
TLC272A	2	3	18	1pA		4.5	2.3	10	CMOS, low-power
TLC279C	4	3	18	0.1pA	0.4	4.5	2.3	10	Quad CMOS

4000 Series Logic IC

Device	Description	Device	Description
4001	Quad 2-input buffered NOR gate	4051	8-channel analog multiplexer/demultiplexer
4002	Dual 4-input NOR gate	4052	Dual 4-channel analog multiplexer/demultiplexer
4006	18-stage static shift register	4053	Triple 2-channel analog multiplexer/demultiplexer
4007	Dual complementary pair and inverter	4063	4-bit magnitude comparator
4009	Hex inverting buffer	4066	Quad bilateral switch
4010	Hex noninverting buffer/converter	4068	8-input NAND gate
4011	Quad 2-input buffered NAND gate	4069	Input protected hex inverter
4012	Dual 4-input NAND gate	4071	Quad 2-input buffered OR gate
4013	Dual D flip-flop	4072	Dual 4-input buffered OR gate
4014	8-bit static shift register	4073	Triple 3-input buffered AND gate
4015	Dual 4-bit static shift register	4075	Triple 3-input buffered OR gate
4016	Quad bilateral switch	4077	Quad 2-input EXCLUSIVE-NOR gate
4017	5-stage decade counter/divider	4078	8-input buffered NOR gate
4018	Presettable divide-by-N counter	4081	Quad 2-input buffered AND gate
4020	14-stage ripple carry binary counter	4082	Dual 4-input buffered AND gate
4021	8-bit static shift register	4093	Quad 2-input NAND Schmitt trigger
4022	Divide-by-8-counter/divider with 8 decimal outputs	4094	8-stage shift-and-store bus register
4023	Buffered triple 3-input NAND gate	4099	8-bit addressable latch
4024	7-stage ripple carry binary counter	4511	BCD to 7-segment latch/decoder/driver
4025	Triple 3-input NOR gate	4512	8-channel buffered data selector
4027	Dual JK flip-flop	4514	4-bit latched/4-to-16 line decoders
4028	BCD to decimal decoder	4515	4-bit latch/4-to-16 line decoder
4029	Synchronous up/down counter, binary/decade counter	4516	Binary up-down counter
4030	Quad EXCLUSIVE-OR gate	4518	Dual 4-bit decoder counter
4040	12-stage ripple carry binary counter	4520	Dual 4-bit counter

4042	Quad D-clocked latch	4521	24-stage frequency divider and oscillator
4043	Quad cross-couple NOR R/S 3 state latch	4528	Dual monostable multivibrator
4047	Monostable/astable multivibrator	4538	Dual precision monostable multivibrator
4049	Hex inverting buffer	4584	Hex Schmitt-trigger
4050	Hex noninverting buffer	4585	Magnitude comparator

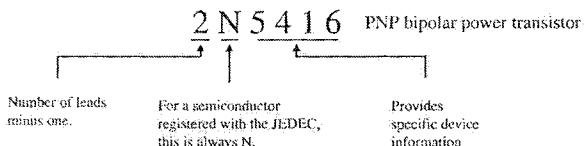
7000 Series IC

Device	Description	Device	Description
7400	Quad 2-input NAND gate	74138	3-line to 8-line decoder/demultiplexer
7402	Quad 2-input NOR gate	74139	Dial 2-line to 2-line decoder/demultiplexer
7404	Hex Inverter	74145	BCD-to-decimal decoder/driver
7406	Hex Inverter/buffer driver	74148	8-line to 3-line priority encoder
7408	Quad 2-input positive AND gate	74150	16-bit data selector
7410	Triple 3-input NAND gate	74151	Data selector/multiplexer
7411	Triple 3-input AND gate	74153	Dual 4-line to 1-line data selector/multiplexer
7414	Hex Schmitt-trigger inverter	74155	Dual 2-line to 4-line decoder/demultiplexer
7420	Dual 4-input NAND gate	74156	Dual 2-line to 4-line decoder/demultiplexer
7421	Dual 4-input AND gate	74158	Quad 2-line to 1-line multiplexer
7426	Quad 2-input NAND gate	74159	4-line to 16-line decoder/demultiplexer
7427	Triple 3-input NOR gate	74161	Synchronous 4-bit binary counter
7430	8-input NAND gate	74163	Fully synchronous 4-bit binary counter
7432	Quad 2-input OR gate	74164	8-bit parallel-out serial shift register
7433	Quad 2-input NOR buffer	74166	Parallel load 8-bit shift register
7438	Quad 2-input NAND buffer	74169	Synchronous 4-bit up/down counter
7442	4-line to 10-line decoder (1 of 10)	74378	Hex D-type flip-flop
7445	BCD-to-decimal decoder/driver	74390	Dual decade counter
7447	BCD 7-segment decoder/driver	74293	Dual 4-bit decade and binary counter
7451	Dual 2-wide 2-input AND-OR-invert gate	74173	Quad D-type flip-flop 3-state output
7473	Dual JK master slave flip-flop with clear	74174	Hex D-type flip flop with clear
7474	Dual D-type positive edge-triggered flip-flop	74175	Quad D-type flip-flop with clear
7475	4-bit bistable latch	74191	Synchronous up/down 4-bit counter
7485	4-bit magnitude comparator	74193	Synchronous 4 bit up/down counter
7486	Quad 2-input exclusive OR gate	74194	4-bit bi-directional shift register
7492	Divide-by-12 counter	74195	4-bit parallel access shift register
74107	Dual JK master slave flip-flop	74221	Dual monostable multivibrator with Schmitt-trigger input
74109	Dual JK positive-edge-trig. flip-flop	74240	Octal buffers/drivers with 3-state output
74112	Dual JK negative-edge-triggered flip-flop	74241	Octal bus/line driver
74121	Monostable multivibrator	74243	Quad bus transceiver
74122	Retriggerable monostable multivibrator	74244	Octal buffer, line driver, line receiver
74123	Dual retriggerable monostable multivibrator	74245	Octal bus transceiver with 3-state output
74124	Dual voltage controlled oscillator	74251	Data selector/multiplexer with 3-state output
74125	Quad 3-input buffer with 3-state output	74253	Dual 4-line-to-1 line data selector/multiplexer with 30state output
74126	Quad 3-state buffer	74257	Quad 2-to-1-line selector/multiplexer with 3-state output
74128	Quad 2-input NOR driver	74258	Quad 2-to-1-line selector/multiplexer with 3-

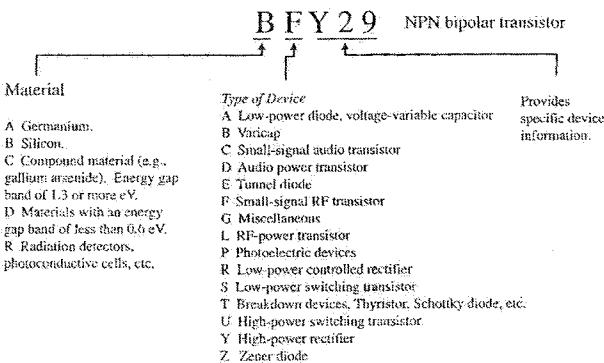
			state output
74132	Quad 2-input NAND with Schmitt trigger	74259	8-bit addressable latch
74136	Quad 2-input XOR gate	74521	Octal comparator
74266	Quad 2-input XOR gate	74540	Octal buffer/line driver
74273	Octal D-type flip flop with clear	74541	Octal buffer/line driver
74279	Quad S-R latch	74573	Octal D-type flip-flop 3-state output
74280	9-bit odd-even parity generator/checker	74574	Octal D-type flip-flop 3-state output
74283	4-bit binary full adder	74590	8-bit binary counter with 3-state output
74298	Quad 2-input multiplexer with storage	74595	8-bit shift registers with output latches
74348	8-line-to-3-line priority encoder with 3-state output	74640	Octal inverting bus transceiver
74365	Hex bus driver with 3-state output	74645	Octal non-inverting bus transceiver
74367	Hex buffer/driver, true data	74646	Octal noninverting bus transceiver output
74373	Octal transparent latch	74646	Octal bus transceiver and register 3-state output
74374	Octal D-type flip-flop	74670	4x4 register file with 3-state output
74375	Quad latch	74688	8-bit identity comparator

Semiconductor Codes

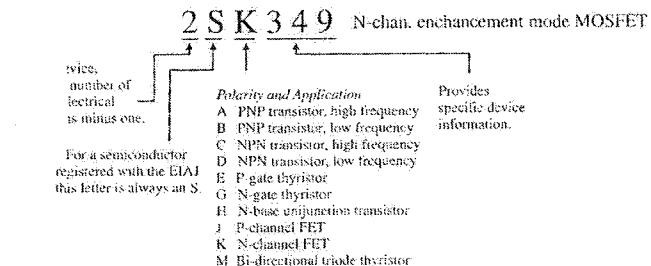
JOINT ELECTRON DEVICE ENGINEERING COUNCIL (JEDEC) CODE



EUROPEAN SEMICONDUCTOR CODE (PRO ELECTRON CODE)



JAPANESE INDUSTRIAL STANDARD (JIS)



5.7 Resources

T. E. Kissell, *Industrial Electronics*, NJ: Prentice Hall, 2nd ed., 2000.

B. Davies, *Practical Robotics*, Toronto: WERD Technology Inc., 1997.

P. Scherz, *Practical Electronics for Inventors*, New York: McGraw-Hill, 2000.

Electronics Tutorials, <http://www.williamson-labs.com/home.htm>: Basic Tutorials on different electronic components.

The Scots Guide to Electronics, http://www.st-andrews.ac.uk/~www_pa/Scots_Guide/intro/electron.htm: Simple graphical demonstration of various electronic components.

Electronics Learning Resources, <http://newton.ex.ac.uk/teaching/CDHW/Electronics2/ElectronicsResources.html>: Many useful theoretical sources can be found on this site.

Electronics Resources, <http://dmoz.org/Science/Technology/Electronics/Tutorials/>: A comprehensive list of electronics resources.

Hobby Circuits, <http://imagineeringezine.com/e-zine/hcircuits.html#Photo-Flash%20Circuits>: Thousands of simple and complicated circuits.

Hobby Electronics, http://www.interq.or.jp/japan/se-inoue/e_menu.htm: Another good source of different circuits.

BatteryUniversity, <http://www.batteryuniversity.com>: Tutorials and resources on batteries.

Chapter 6

Electromechanical Design

6.1	Introduction	1
6.2	Actuators	1
6.2.1	Solenoids	1
6.2.1.1	Selection Factors	2
6.2.1.2	How to Select A Solenoid	4
6.2.1.3	Minimizing Unnecessary Energy Consumption	6
	Voltage Reduction	6
	Resistance Increase	7
6.2.2	DC Motors	8
6.2.2.1	Impedance Matching	11
6.2.2.2	DC Motor Specifications	12
	Gearbox	12
6.2.2.3	Thermal Characteristics	14
6.2.2.4	Speed Control of DC Motors	14
6.2.2.5	Directional Control of DC Motors	15
6.2.2.6	Torque Regulator	18
6.2.2.7	Noise Reduction Techniques	19
6.2.3	Brushless DC Motors	20
6.2.3.1	Construction of a Brushless DC Motor	20
	Winding Configuration	21
	Hall Effect Sensors	21
6.2.3.2	Commutation of Brushless DC Motors	21
	Sensored Brushless DC Motors	21
	Sensorless Brushless DC Motors	22
6.2.3.3	Hall Effect Sensors on Brushless DC Motors	22
6.2.3.4	Driving Brushless DC Motors	23
	Speed Control of Brushless DC Motors	23
	Direction Control of Brushless DC Motors	23
6.2.4	Stepper Motors	24
6.2.4.1	Stepper Motor Types	24
	Variable Reluctance Steppers	25
	Unipolar Permanent Magnet Stepper Motors	25
	Bipolar Permanent Magnet Stepper Motors	26
	Hybrid Permanent Magnet Stepper Motors	26
6.2.4.2	Driving Stepper Motors	27
6.2.4.3	Controlling the Stepper Driver	28
6.2.4.4	Identifying the Stepper Motors	30
6.2.4.5	Stepper Motor Specifications	31
6.2.4.6	Thermal Characteristics	32
6.2.5	Servo Motors	33
6.2.5.1	Servo Motor Construction	33
6.2.5.2	Driving Servo Motors	34
6.2.6	Some Common Problems in Motors	35
6.2.7	The AER201 Driver Board	36
6.3	Transmission Systems	37

6.3.1 Gears	37
6.3.2 Flexible Mechanisms	39
6.3.3 Lead Screws	40
6.3.4 Continuously Variable Transmission	40
6.3.5 Shaft Coupling	41
6.3.6 Mounting Small Electric Motors	44
6.4 Stiffness and Deflection	45
6.4.1 Flexible Elements in Parallel and Series	45
6.4.2 Shafts	45
6.4.3 Gears	45
6.4.4 Belts	46
6.4.5 Links	46
6.5 Drive Mechanisms	47
6.5.1 Differential Drive	47
6.5.2 Steering System	47
6.5.3 Synchro Drive	48
6.5.4 Legs	48
6.6 Useful Mechanisms	49
6.6.1 Rotary-Longitudinal Motion Converters	49
6.6.2 One-way Motion Controls	52
6.7 Measuring Moment of Inertia	54
6.7.1 Method 1	54
6.7.2 Method 2	55
6.7.3 Method 3	55
6.7.4 Method 4	57
6.9 Adhesives	61
6.10 Resources	63

6.1 Introduction

The electromechanical subsystem is the greatest subsystem of the projects and here are the top ten reasons why:

- Greatest span for originality,
- Greatest range of complicating factors (friction, forces in all directions, torques....),
- Greatest reliance on past knowledge and practical experience,
- Greatest savings can be realized in surplus,
- Greatest problems for finding to-order parts,
- Greatest requirement for dynamic problem solving,
- Greatest requirement for non-linear thinking to recognize a needed part currently in another application,
- Greatest reliance on Common Sense,
- Greatest trap for over-confidence,
- Greatest span for failure.

Due to the breadth of issues of interest to the Electromechanical students, most issues arising in the design will be dealt with at lectures, during question and answer sessions, and individually. However, some general notes on actuators are addressed in this chapter.

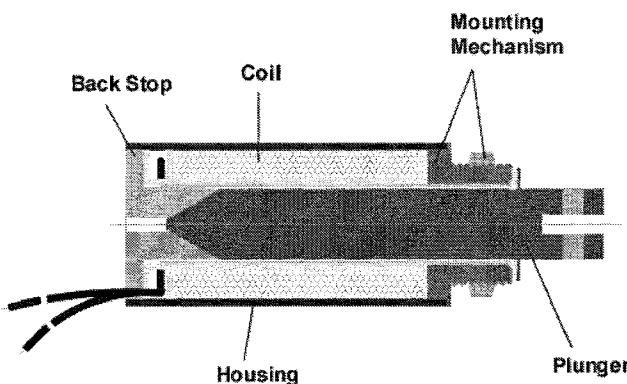
6.2 Actuators

Electromechanical systems employ actuators or drives that are part of the physical process being monitored and controlled. Actuation is the result of a direct physical action on the process, such as removal of a workpiece from a conveyor system or application of a force. It has a direct effect upon the process. Actuators take low-power signals transmitted from the computer or microcontroller and produce high-power signals that are applied as input to the process. There are many types of actuating devices, some of the most common ones being solenoids, electrohydraulic actuators, DC or AC motors, stepper motors, piezoelectric motors, and pneumatic devices.

Electrical actuators convert electrical command signals into mechanical motions. In this chapter, emphasis is placed on solenoids, DC motors, and stepper motors, because of their popularity in electromechanical systems, and their fitness to the Engineering Design projects.

6.2.1 Solenoids

A solenoid is an electromechanical device employed to convert electrical energy into linear (and sometimes rotational) movement. The resulting linear work produced could either be pull or push in configuration, and is usually with a short stroke (up to 25 mm). A solenoid consists of a long wire, wound with a helical pattern, usually surrounded by a steel frame, having a steel core inside the winding. When carrying a current, the solenoid becomes an electromechanical device, in which electrical energy is converted into mechanical work.



The core of a solenoid is usually made of two sections, a movable actuator, or plunger, and a fixed end core or backstop. The efficiency of a solenoid is a factor of mechanical geometry, electrical configuration and magnetic permeability of core, plunger and housing. The plunger is free to travel in the center of winding in a linear direction. When the coil is energized by the electric current, a magnetic force is created between plunger and end core, causing the plunger to travel. The higher the permeability of steel used, the better the performance.

The force available from a simple solenoid is given by the following expression:

$$force = \frac{1}{2} \frac{N^2 I^2}{x^2} A \mu_0$$

where N is the number of turns on the coil, I is the current through the coil, A is the cross-sectional area of the air gap, x is the length of the air gap (stroke), and μ_0 is the absolute permeability of air ($4\pi \times 10^{-7}$ wb/amp.m or kg.m/(amp) 2 .s).

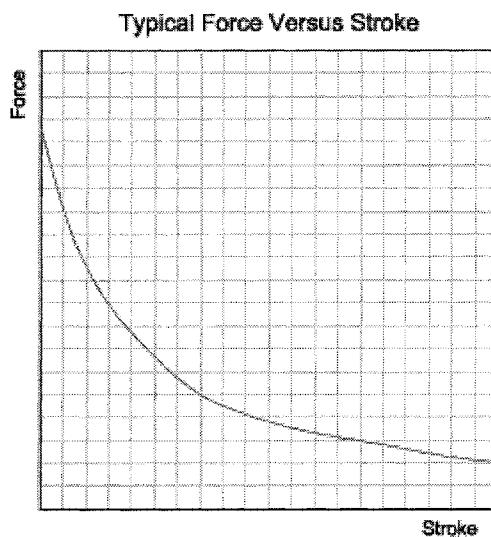
It is essential for a solenoid to lose its magnetic force when input electrical power is removed. This is to allow the plunger to resume its original position. Any remaining magnetic field is called residual magnetism. Material of the central guide and plating of plunger must be chosen to introduce minimum friction and low wear. Glass filled nylon and brass for the guide and electro-less nickel plating or other low friction coatings for the plunger are often good candidates.

Design and selection of a solenoid requires basic knowledge of mechanical and electrical interrelationships. In many cases it is essential to make tradeoffs among a variety of mechanical, electrical, thermal, acoustical, and physical properties.

6.2.1.1 Selection Factors

To obtain the optimum performance, reliability and life of a solenoid, selection considerations should include the following factors:

- **Force or Torque:** Pull, push or rotary load, developed by plunger when the coil is activated by an external voltage.
- **Stroke:** The distance a plunger must travel before it is stopped. In general, the force initially developed by the solenoid is dependent upon the stroke; i.e., the longer the stroke, the smaller the force. The force versus stroke relationship must be known for any particular solenoid to be used. This relationship is usually shown as the characteristic curve above. This relationship can be changed for a given solenoid, by changing the geometry of the interior components. For example, by changing the plunger tip geometry, the force versus stroke relationship can be totally altered, as illustrated in the figure on the next page.
- **Temperature:** When considering a solenoid application for your project, it is very important to determine what effect heat will have on the final output force. Heat can be the greatest enemy to proper performance, and can cause premature failure. When a solenoid is energized by an outside voltage supply, some heat is generated which increases



the temperature of the coil. This temperature rise has some undesired effects, since resistance of the coil winding varies with temperature: i.e., by increasing temperature, resistance rises, which in turn, reduces the electrical current. This will result in a reduction in force output. Resistance of copper changes by approximately 0.39 percent per degree centigrade, in the neighborhood of 20°C. An extreme increase in temperature can result in damages to the winding and other components. Usually the limiting factor for operating temperature is the rated temperature of insulating material used in the solenoid. The standard rated temperatures are as follows:

Class "A" 105°C maximum

Class "B" 130°C maximum

Class "F" 155°C maximum

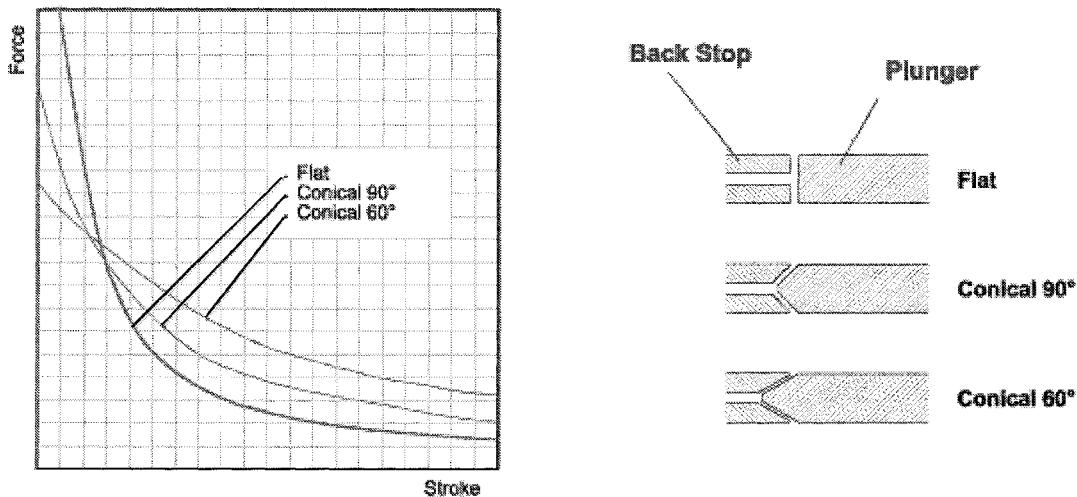
Class "H" 180°C maximum

Temperature rise in a solenoid can be reduced by dissipating the heat generated in the coil. This can be done by mounting the solenoid on a metal surface (heat sink) large enough to dissipate the excess energy, or by forced air cooling, or where the space permits, by using a larger size solenoid. Two other methods for inputting a smaller amount of energy and as a result generating less heat are to use the solenoid *intermittently*, or to use a solenoid with *multiple windings*.

- **Duty cycle:** Higher performances may be obtained from a solenoid by introducing higher input power, in which case more heat will be generated. One way to reduce the amount of excess heat generated at a higher input power is to use the solenoid *intermittently*, that is, the solenoid is used in "ON-OFF" intervals. Duty cycle is the ratio of "ON" time over the "TOTAL" time for any one cycle of operation. Usually duty cycle is expressed in percentage.

$$\text{Duty Cycle (\%)} = \frac{\text{"ON" time}}{\text{"ON" + "OFF" time}}$$

- **Size:** The space requirements assign the dimensions of the solenoid envelope. In general, the higher force asked, the bigger solenoid must be expected and considered in the design.
- **Power Supply:** The applied power in a solenoid can be either DC or rectified AC. In a DC solenoid, current remains constant throughout the entire stroke while force continues to increase. For continuous duty, the solenoid has to be mounted on a heat sink large enough to dissipate the coil heat buildup, or a mechanical or solid-state cutout switch can be used to reduce power after the solenoid plunger seats. AC solenoids are rated for continuous duty and a cycle rate of approximately thirty (30) times per minute at full stroke and normal ambient temperature. Higher cycle rates can be obtained when the solenoid is operated at less than full stroke or when the coil is designed for lower than standard force. Large AC solenoids should be cycled at a slower rate. Force curves are relatively flat until the end of the stroke. Current is highest at the start of the stroke and decreases to a low level for continuous duty when the plunger seats. Current is inversely proportional to the voltage. Plungers must always seat at the end of the stroke or coils will overheat.



6.2.1.2 How to Select A Solenoid

In catalogues, for each solenoid, a coil data table, and a set of force versus stroke curves are presented. By knowing the size, stroke and required force, a number for ampere turns can be obtained from the curves, which in turn, by reference to coil data, points out the amount of power consumption. By choosing the proper input voltage to the solenoid from the related column, the wire size will be obtained, which defines the type of solenoid to be used.

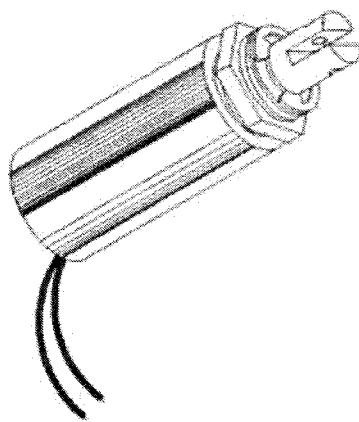
Example:

In an electro-mechanical system, space allows for a 1in diameter by 2in long solenoid, which must exert 50 ounces of force at a stroke of 0.2in. The solenoid will be used intermittently and the input voltage is approximately 12V DC. A proper choice is S-20-100 from *Magnetic Sensor Systems*. The data and force-stroke curve is shown below. It shows that approximately 1500 ampere-turns is required in order to apply 50 ounces at 0.2 inches. The closest ampere-turns recorded in the chart is 1590AT. A simple interpolation will show the proper voltage for the solenoid.

$$11.9V \times \frac{1500 \text{ AT}}{1590 \text{ AT}} = 11.2V$$

The AWG number (referring to the wire size) in the chart corresponding to 11.9V is 26. Hence, the solenoid number to order is S-20-100-26.

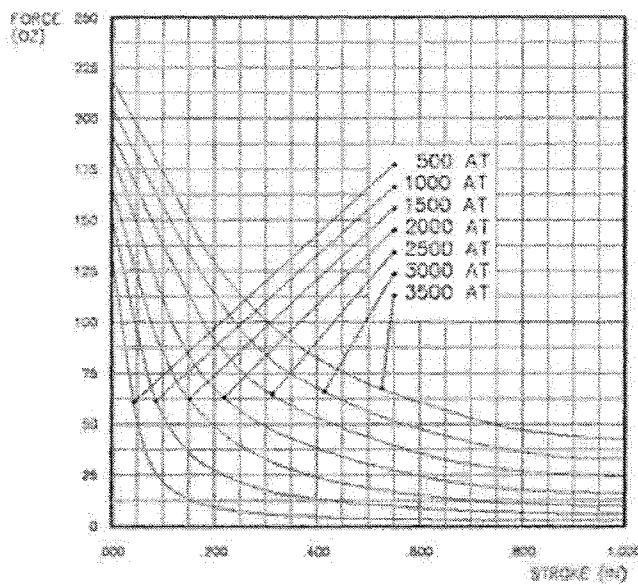
In this example no safety factor was considered. Considering a 30% safety factor is a good engineering practice. The force will be dropped slightly as the solenoid gets warmer. It should also be noted that the force versus stroke curves are typical, under nominal conditions.



Series S-20-100
1" DIA X 2"

TOTAL WEIGHT: 6.2 OUNCES
PLUNGER WEIGHT: 1.6 OUNCES

duty cycle maximum "ON" time, (Sec.)	1 sec	1/2 400	1/4 35	1/10 10
watts approximate ampere turns	10 1110	20 1590	40 2220	100 3510
AWC number	resistance	volts DC	volts DC	volts DC
23	1.52	3.9	5.5	7.8
24	2.95	5.4	7.7	10.9
25	4.01	6.3	8.0	12.7
26	7.10	8.4	11.8	16.9
27	9.92	10.0	14.1	19.9
28	17.1	13.1	18.5	26.2
29	27.5	16.6	23.5	33.2
30	46.2	21.5	30.4	43.0
31	72.0	26.8	37.9	53.7
32	110	33.2	48.9	66.3
33	175	41.8	59.2	83.7
34	270	52.0	73.5	104
35	450	67.1	94.9	134
36	665	81.5	115	163
37	1040	102	144	204
38	1550	125	176	249
39	2250	188	235	332
40	4350	209	295	417



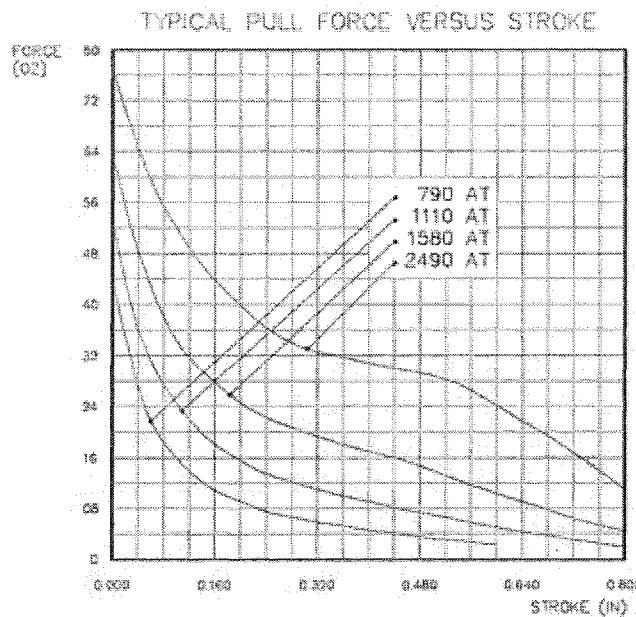
6.2.1.3 Minimizing Unnecessary Energy Consumption

Voltage Reduction

One method of consuming less energy is to apply a voltage to the solenoid coil to provide the necessary force to pull the plunger in, then, when seated, drop the voltage down to a level sufficient to maintain the plunger in a seated position. As an example, we may consider the following condition:

Example:

One particular design requires 20 ounces of force to move a linkage by 0.6 inches. Space allows for a 3/4 inch diameter by 1.5 inch long solenoid using a 24 volt DC supply.



duty cycle	1	1/2	1/4	1/10
maximum "ON" time, (Sec.)	∞	180	30	8
watts	8	16	32	80
approximate ampere turns	790	1110	1580	2490
AWG number	resistance	volts DC	volts DC	volts DC
23	0.63	2.4	3.3	4.7
24	1.05	3.0	4.2	6.0
25	1.48	3.7	5.2	7.4
26	2.68	4.5	6.4	9.1
27	4.43	5.9	8.2	11.7
28	7.03	7.5	10.5	15.0
29	11.0	9.4	13.2	18.8
30	20.4	11.9	16.7	23.8
31	28.2	14.9	21.0	29.8
32	45.1	18.6	26.2	37.2
33	70.5	23.7	33.3	47.4
34	98.8	29.3	41.1	58.5
35	191	38.2	63.7	76.4
36	269	47.8	67.2	95.6
37	458	59.0	83.0	118
38	692	74.9	105	150
39	1243	99.7	140	199
40	2066	126	177	253

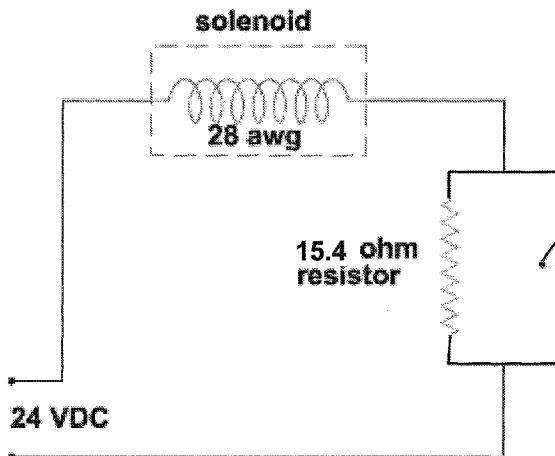
A good choice for this application would be the S-15-75 solenoid from *Magnetic Sensor Systems*. Using the force-stroke graph, it is observed that 24 ounces at 0.6 inches stroke can be obtained from a solenoid with approximately 2490AT (slightly more force than actually needed). By referring to the coil data, 80 watts will be required to generate the 2490 ampere-turns, but the solenoid cannot be left on for more than 8 seconds. The closest voltage in the related column is 23.6V DC, which refers to a coil with 28 AWG with an internal resistance of approximately 7 Ohms. Therefore, the proper solenoid is S-15-75-28.

When the plunger is seated, which means when the stroke is zero, the graphs indicate that only approximately 500 ampere-turns will be required to generate approximately 24 ounces of force. If we consider 790AT for hold, the voltage can be dropped to 7.5V. This results in 8 watts of power consumption. In this case the voltage can be left on indefinitely and power consumption will be dropped by approximately 90% in order to maintain the initial required force.

Resistance Increase

In cases where the voltage cannot be dropped, similar results could be achieved by placing a resistor in series with the coil when the plunger is seated.

In the previous example, in order to switch to 790AT at 24V DC input, the current must be dropped from 3.4 amperes to 1.07 amperes. A 15.4 ohm resistor can be switched in the circuit in series with the solenoid. However, it must be noted that the total power consumption in this case is actually 25.6 watts; where 17.6 watts of this amount is being dissipated as heat through the 15.4 ohm resistor, and the balance; 8 watts is being used by the solenoid.



6.2.2 DC Motors

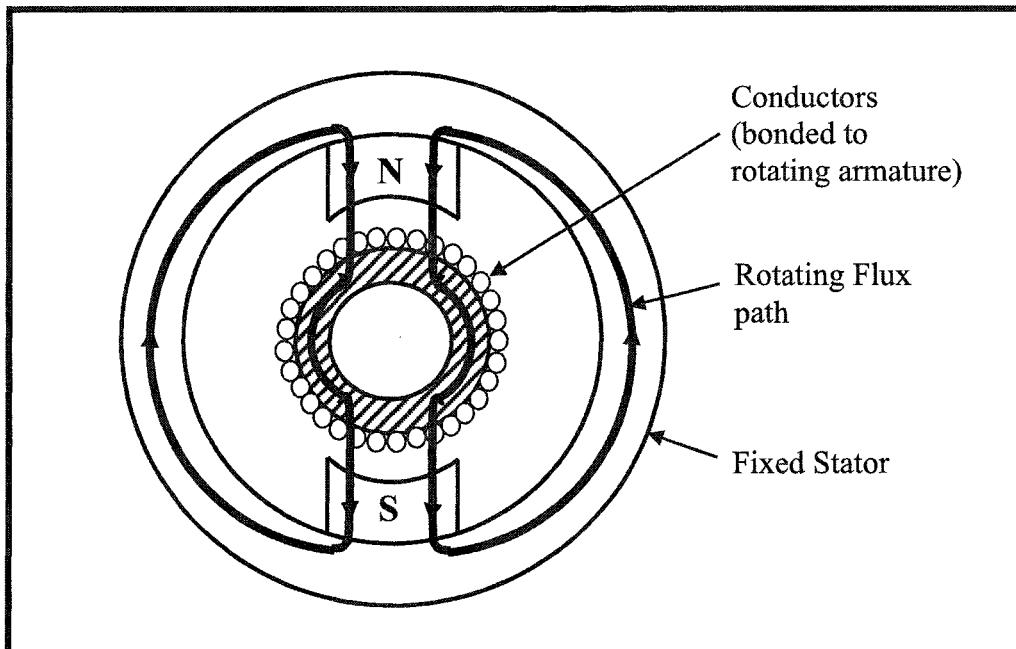
DC motors have the wonderful property of simplicity, at least as viewed by the user. When a voltage is applied to the motor, it turns. The higher the voltage, the faster it turns. This property has made DC brush motors very popular since their invention by Faraday in the mid-nineteenth century. The dominance of DC motors was challenged in the early twentieth century by the widespread adoption of AC electric power distribution. AC motors were cheaper to build and could run off the available power supply, so displaced DC motors in large numbers of applications. The inherent controllability of DC motors, however, prevented their complete demise. Machine control, among other applications, continued to require DC motors. Developments in power electronics in the mid and late twentieth century made possible the synthesis of arbitrary, high-power waveforms. With this ability, the controllability of AC-style motors can be vastly increased. The simplest of these motors is the stepping motor, which has successfully replaced DC motors in many positioning applications. DC-brushless motors and AC servomotors represent a convergence of this trend.

As with any motor, a DC motor depends on the generation of a force when a current-carrying conductor is placed in a magnetic field. In DC brush motors, the current-carrying conductor is connected to the rotor (the armature). The stator is used to provide a magnetic field that is fixed in space. In the simplest possible arrangement, one coil is attached to the armature, free to rotate in a magnetic field supplied by the stator. The rotor has many closely spaced slots on its periphery. These slots carry the rotor windings. The rotor windings (armature windings) are powered by the supply voltage. An arrangement of commutation segments and brushes ensures the transfer of DC current to the rotating winding. The rotation direction depends on the direction of current flow in the armature; if the current flow were to be reversed, the direction of the torque applied to the rotor will also change.

A DC motor works on the principle that a current carrying conductor in a magnetic field, created by a permanent magnet or electromagnet, experiences a force. The motor consists of a fixed **stator** that establishes a magnetic field across a turning **rotor** (or **armature**), as shown in figure below. The rotor consists of a shaft and windings through which current moves to power the motor. If the stator produces a radial magnetic flux ϕ (in webers) and the current in the rotor is i_a (in amperes), then there will be a torque τ_m (in Newton-meters) on the rotor causing it to rotate:

$$\tau_m = K_1 \phi \ i_a$$

where K_1 is a physical constant. In addition, whenever a conductor moves in a magnetic field, a voltage V_b (in volts), called **back emf**, is generated across its terminals, which tends to oppose the current flow in the conductor. This voltage



is proportional to rotational velocity of the armature ω_m (in rad/sec):

$$v_b = K_2 \phi \omega_m$$

where K_2 is another physical constant. Considering a permanent magnet DC motor, the flux ϕ remains constant, and the torque on the armature is then controlled merely by the rotor current i_a :

$$\tau_m = K_t i_a$$

where K_t is the torque constant (in N.m/amp), and also:

$$v_b = K_v \omega_m$$

where K_v is the back emf constant.

The physical schematic of the DC motor connection is shown in figure below, representing the electric circuit as well as the mechanical load on the rotor. The major components of the electric circuit are a voltage source v_a , the inductance of the armature windings l_a , and the generated back emf v_b . The dynamics of the circuit is described by a first-order differential equation given by

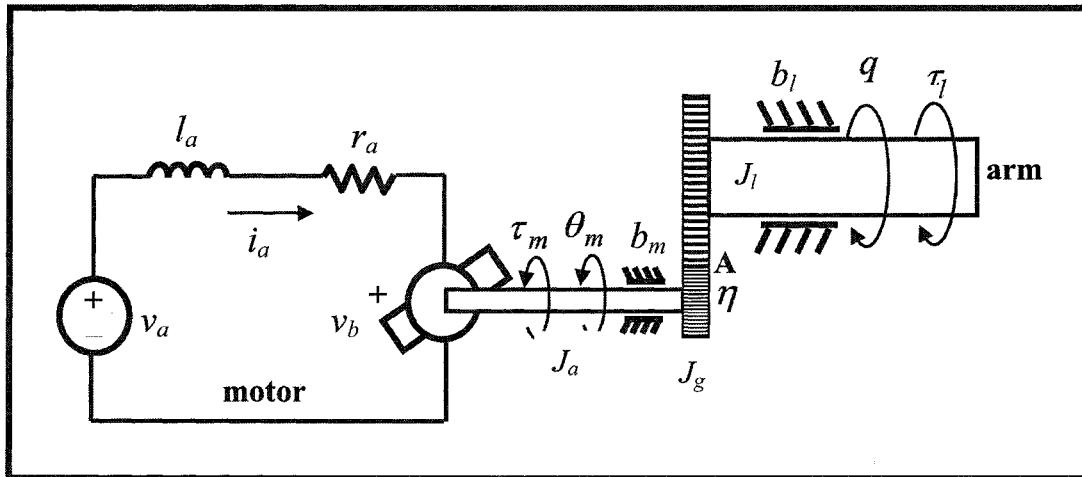
$$l_a \frac{di_a}{dt} + r_a i_a = v_a - v_b$$

$$l_a \frac{di_a}{dt} + r_a i_a = v_a - K_v \omega_m$$

The rotor current i_a is proportional to the motor torque τ_m by the torque constant K_t . The torque constant of a DC motor can be measured by *stalling* the rotor (not allowing it to rotate) and measuring the stalled torque τ_{m0} and the supplied voltage v_{a0} . With no rotation, v_b and di_a / dt are both zero, and we have:

$$v_{a0} = r_a i_{a0} = r_a \frac{\tau_{m0}}{K_t} \Rightarrow K_t = \frac{r_a \tau_{m0}}{v_{a0}}$$

The second part of the schematics is representing the mechanical system consisting of motor shaft, bearings, gear system, and arm. The torque-speed characteristics of motors are fixed by their design parameters. In general, there are many more potential load situations than there are standard motors available, so it may not always be possible to match a motor optimally to its intended load. A gearbox or other impedance-transforming device, such as pulleys or lead screws, can be used to improve the match. These devices are the mechanical equivalents of electrical transformers. The gearbox or other transformer is first used to match the needed static characteristics of the motor-load system. The speed range and torque



range are the most important of these static properties. The impedance requirement is usually to magnify the torque and consequently reduce the speed on the load side. Hence, the simplest configuration of a gearbox is to mount a smaller gear (with diameter d_m) on the motor shaft and engage it with a larger gear (with diameter d_l) mounted on the load shaft. The ratio of the gear diameters is called *gear ratio* η .

$$\eta = \frac{d_l}{d_m}$$

In typical mechanisms such as robot manipulators, the gear ratio η typically has values in the range 20 to 200 representing a torque magnification from 1 to 20 up to 200.

Assuming no *slippage* at the gears' contact point A, the linear velocity of this point remains the same on both gears. Hence,

$$(V_A)_m = (V_A)_l \Rightarrow \frac{d_m}{2} \omega_m = \frac{d_l}{2} \omega_l \Rightarrow \frac{d_l}{d_m} = \frac{\omega_m}{\omega_l}$$

$$\eta = \frac{\omega_m}{\omega_l}$$

The above equation represents the kinematic relation between motor shaft and load shaft.

For obtaining the torque relation in a gear mechanism, note that the action and reaction forces at the contact point A have equal amount (assuming no mass for the contact point). Thus,

$$(F_A)_l = (F_A)_o \Rightarrow \frac{2\tau_l}{d_l} = \frac{2\tau_o}{d_m} \Rightarrow \frac{\tau_l}{\tau_o} = \frac{d_l}{d_m}$$

$$\eta = \frac{\tau_l}{\tau_o}$$

The moments of inertia of motor shaft and gear system are J_a and J_g (in $\text{kg} \cdot \text{m}^2$), respectively, and we set $J_m = J_a + J_g$.

The friction at the bearings could be modeled as a viscous force proportional to the rotational velocity with factors b_m for shaft bearings and b_l for arm bearings. The dynamics of the motor is described by the following differential equation:

$$J_m \frac{d\omega_m}{dt} + b_m \omega_m = \tau_m - \tau_o = \tau_m - \frac{1}{\eta} \tau_l$$

The dynamics of the load is also expressed by the following differential equation:

$$J_l \frac{d\omega_l}{dt} + b_l \omega_l = \tau_l$$

Combining the two dynamic equations and using the gear kinematic relation we obtain:

$$\begin{aligned} & \left(J_m + \frac{1}{\eta^2} J_l \right) \frac{d\omega_m}{dt} + \left(b_m + \frac{1}{\eta^2} b_l \right) \omega_m = \tau_m \\ & \left\{ \begin{array}{l} J \frac{d\omega_m}{dt} + b \omega_m = \tau_m = K_t i_a \\ l_a \frac{di_a}{dt} + r_a i_a = v_a - K_v \omega_m \end{array} \right. \end{aligned}$$

where,

$$J = \left(J_m + \frac{1}{\eta^2} J_l \right) \quad , \quad b = \left(b_m + \frac{1}{\eta^2} b_l \right)$$

are called combined inertia and combined viscous, respectively, *as seen* by the motor.

A reasonable assumption for many electromechanical systems is that the "electrical time constant" $\frac{L_a}{r_a}$ is much smaller than the "mechanical time constant" $\frac{J_m}{b_m}$. In other words, the circuit delay due to the inductance is much shorter than the dynamic inertia of the mechanical load. Neglecting the inductance of the armature coil will result in:

$$i_a = \frac{v_a - v_b}{r_a} = \frac{v_a - K_v \omega}{r_a},$$

and hence a linear relation is obtained between the rotor speed and the motor torque:

$$\tau_m = \frac{K_t}{r_a} (v_a - K_v \omega_m).$$

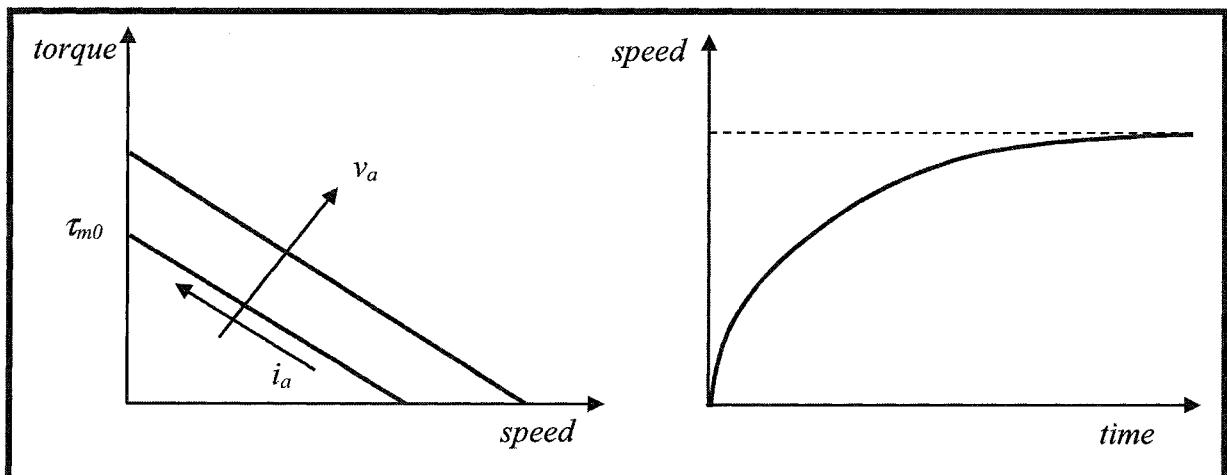
A typical torque-speed graph of DC motors is shown in figure below. Note that increased torque on a motor's shaft always results in a loss of speed, unless more current is supplied to the armature. When a motor is stalled, total input power is dissipated by the armature windings. In practice, high torques are always developed by gearing down a high speed motor. The dynamic equation of the motor can then be reduced to:

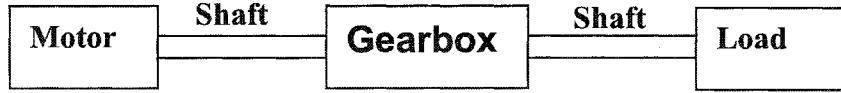
$$J \frac{d\omega_m}{dt} + \left(b + \frac{K_t K_v}{r_a} \right) \omega_m = \frac{K_t}{r_a} v_a.$$

The above equation shows that after applying the voltage on the armature the motor requires some time to reach its nominal speed. This delay depends on motor and load characteristics included in J and b .

6.2.2.1 Impedance Matching

A gearbox is usually used to match the speed and torque range characteristics of the motor-load system. By adjusting the gear ratio, it is possible to optimize some features of the system. For instance, if the maximum power transfer to the load so as to achieve maximum acceleration is the most important dynamic operating characteristic, the classic impedance-matching rule can provide an optimum gear ratio. For this case it will be assumed that the load is a pure inertia and that the motor itself is also a pure inertia. Losses due to friction, compliance, and so on will be neglected.





The kinetic energy of the load is

$$\mathcal{E}_l = \frac{1}{2} J_l \omega_l^2,$$

which can be differentiated with respect to time to get the power:

$$P_l = \frac{d}{dt} \mathcal{E}_l = J_l \omega_l \frac{d\omega_l}{dt}.$$

Reflect this to the motor:

$$P_l = \frac{J_l}{\eta^2} \omega_m \frac{d\omega_m}{dt}$$

The current during this process can be considered to be constant, since maximum power can be achieved only by delivering maximum current to the motor. The (constant) acceleration is proportional to applied torque divided by total inertia:

$$\frac{d\omega_m}{dt} = \alpha_m = \frac{\tau_m}{J}, \text{ and}$$

$$\omega_m = \alpha_m t,$$

where t is the time. Since the acceleration is constant, the power to the load is

$$P_l = \frac{J_l \tau_m^2}{\eta^2 J^2} t = \frac{J_l \tau_m^2}{\eta^2 \left(J_m + \frac{1}{\eta^2} J_l \right)^2} t = \frac{\eta^2 J_l \tau_m^2}{(\eta^2 J_m + J_l)^2} t.$$

Optimize the above by finding a maximum P_l value with respect to the gearbox η .

$$\frac{\partial P_l}{\partial \eta} = 0 \Rightarrow J_m = \frac{1}{\eta^2} J_l.$$

But this is just the load inertia as reflected at the motor. In other words, adjust η so that when viewed from the motor, the impedance (inertia) of the motor matches the impedance of the load as viewed through the gearbox.

6.2.2.2 DC Motor Specifications

An understanding of speed, torque, current, and efficiency allows you to decide what motor to buy for a particular application, and this is always more efficient and less expensive than a “suck it and see” approach. However, normally, when you obtain a motor from a commercial source, you will not immediately know all the information necessary to characterize the motor in terms of the theoretical development above. Some motors include the *stall torque*, the no-load speed, and the resistance, while others include only the rated voltage and the power output at maximum power. In these cases, there are several things you can do to experimentally determine the motor constant, internal resistance of the motor, etc. Here are some basic steps to take to characterize your motor:

- 1) When looking for a high torque DC brush motor in a store's surplus bins, use the following checklist:

- A high torque motor should feel heavy, because it has large powerful magnets. Motor density is not an absolute test because rare earth magnets can reduce a motor's density. However, if the motor is being sold for less than \$10, it will probably have regular ceramic magnets.
 - Large brush size means a motor handles large currents, and since torque is proportional to current this is a good sign too. Can the brushes be replaced? Replaceable brushes are used on many high current motors.
 - If a motor has a 1/16" diameter shaft put it back in the bin. It is common sense that a manufacturer will use a hefty shaft for high torque output.
 - Turn the motor shaft with your fingers. If the shaft 'cogs' (resists rotation with a jerky motion), it probably has good torque too. Unfortunately this is not a foolproof test either, since motors with an even number of commutator segments cog more readily. However, when coupled with the above three criteria cogging is a useful guide.
 - Is the armature wound with heavy gage wire? Note that high torque requires large currents. Small gage armature wires are not used in high torque motors. Heavy wiring also increases motor density.
 - What size tabs are used for the motor connections? Large heavy gage tabs are indicative of high torque.
 - Top quality motors such as those made by AstroFlite have machined bodies, ball bearings and adjustable timing. Cheap housings sometimes also reflect the care and attention paid to internal design features.
- 2) Look on the package of the motor, and obtain all the information you can from there. Then use the given information in the above equations to calculate other values that you don't know. For example:
- Usually at least the voltage, v_a , and the no-load speed, ω_m , are given. If they are, you can do some measurements and use the following equation to calculate K_v . First, measure the internal resistance of the motor, r_a , by an ohmmeter. Then apply the nominal voltage, v_a , to the motor, and measure the current, i_a , by putting an ammeter in series in the circuit. Now, assuming that the motor inductance, l_a , is quite small (if you don't know its value), you can calculate K_v as follows. Be sure to use the correct units.
- $$K_v = \frac{v_a - r_a i_a}{\omega_m}$$
- If the stall torque, τ_{m0} , is given, you can calculate K_t from:
- $$K_t = \frac{r_a \tau_{m0}}{v_{a0}}$$
- If the stall torque is not given, you can measure it by a simple experiment. Attach a large gear to your motor, and hang a weight from a string that will be wrapped up as the motor turns. At the bottom of the string, hang an object of known mass, or a cup that can hold water. Then, apply a voltage, and continue to increase the voltage until the motor just supports the weight of the object. By calculating the weight of the object and the radius of the gear, you can calculate the stall torque of the motor. This type of test is called the *locked rotor* test. Another way to perform this test is by applying a constant voltage and varying the applied weight until the stall torque is reached.
- If you have access to a strobe light (available in the Design lab.) or a tachometer, you can measure the no-load speed.
- 3) Finally, if you cannot obtain the information from the motor itself, and for some reason you cannot obtain the information using the above tests, you can get an idea of the performance of your motor by some estimations or by using information from other similar motors. For example, most DC motors have an efficiency ($power_{out} / power_{in}$) between 60% and 80% when operated in their nominal speed range ($\sim 50\% \text{ rpm}_{\max}$). Therefore, a useful estimate of the output torque may be obtained from the following equation:

$$\tau_m = \frac{P_{out}}{\omega_m} = \frac{(0.6, 0.8) \times (v_a \cdot i_a)}{\omega_m}$$

6.2.2.3 Thermal Characteristics

The main enemy of motors is heat. High temperatures can destroy the components of the motor, and even if they do not cause permanent damage, they can change the operating parameters and cause significant deviations in motor performance. The heat results from mechanical and electrical dissipative mechanisms, and appears in the motor's dynamic equations as a force (or torque) that resists the motion (i.e., a damping force). The main sources of heat production are winding electrical losses (i.e., resistive heating), eddy currents, magnetic hysteresis, windage, friction, short-circuit currents, and brush contact resistance. The dissipation in the windings comes from the resistance of the coils and the associated *ohmic* heating. A field-wound motor will also have ohmic loss from the field coils. They are proportional to the square of the current flow and rise rapidly as the motor is loaded.

The eddy current losses are associated with currents induced in the iron core due to the changing magnetic fields, while the hysteresis losses, also associated with the iron core, come from the shifting of magnetic domain boundaries as the armature rotates. Both of these are motor-speed dependent rather than load dependent. Ironless-core motors do not exhibit either phenomenon, so can be made more efficient than iron-core motors.

Windage and friction cover mechanical sources of dissipation and also tend to be speed dependent. They are heavily dependent on the specific motor design. The brushes contribute to two loss mechanisms. They contribute mechanical friction, and there is also electrical dissipation due to the sliding interaction between the brushes and the shaft. The brush contact resistance is a complex combination of factors, including temperature, speed, and armature current.

Short-circuit currents occur every time the commutator moves from one sector to the next. They are a combination of large currents caused by the rapid change in voltage on the coils, which act as inductances, and the momentary short circuit that occurs when both ends of a coil are connected briefly to the same side of the electrical circuit. The amount of loss is most closely correlated with motor speed.

6.2.2.4 Speed Control of DC Motors

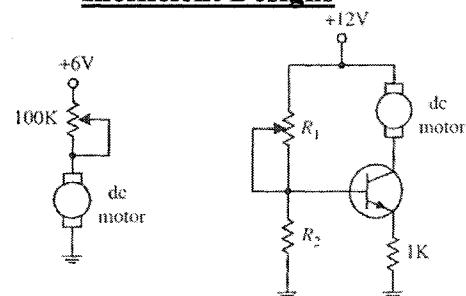
Theoretically, one can control the speed of a DC motor simply by limiting the current flow using a potentiometer, as shown in the circuit to the left in the figure. According to Ohm's law, as the resistance of the potentiometer increases, the current decreases, and the motor will slow down. However, using a potentiometer to control the current flow is inefficient. As the resistance increases, the amount of electric energy that must be converted into heat increases. Producing heat in order to slow a motor is not acceptable practically; it consumes supply power and may lead to potentiometer meltdown. Another imperfect approach to control the speed of a motor is to use a transistor amplifier arrangement like the one shown to the right in the figure. Again, as the collector-to-emitter resistance increases with varying base voltage/current, the transistor must dissipate a considerable amount of heat. This can lead to transistor meltdown.

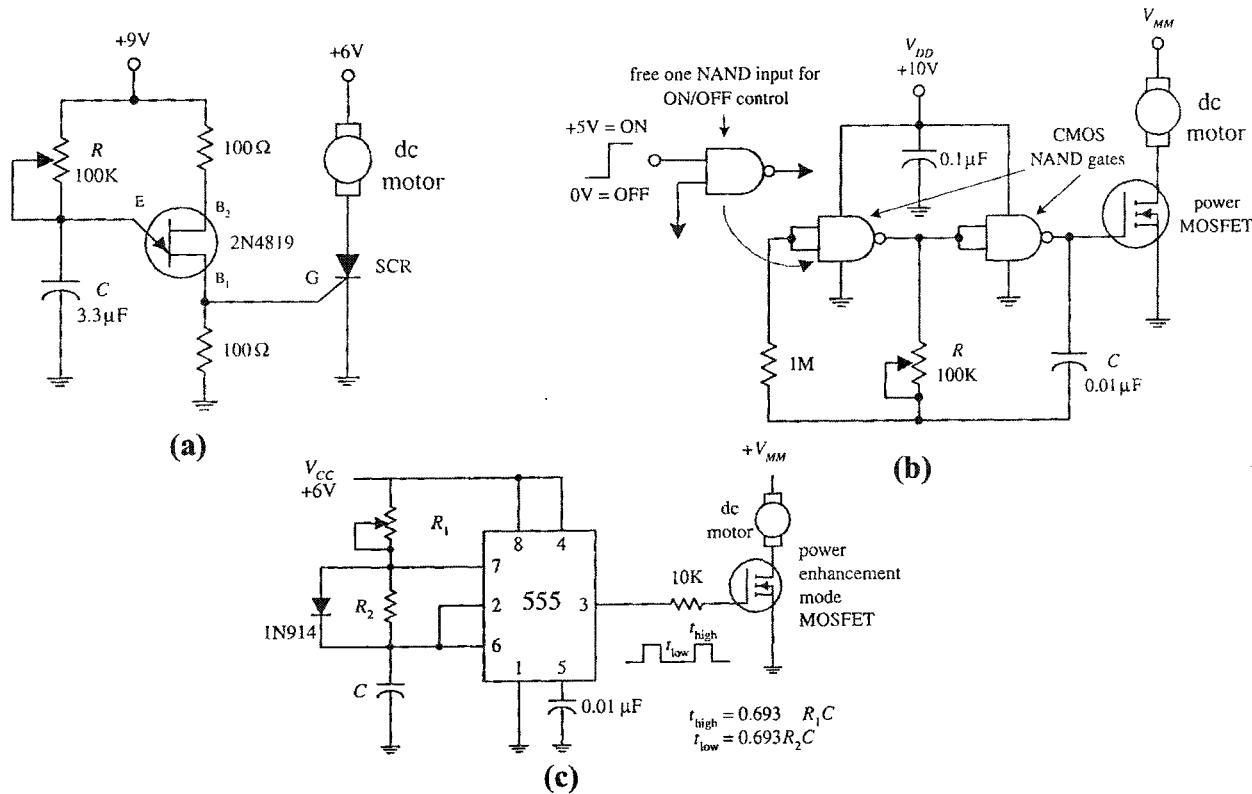
In order to conserve energy and prevent component meltdown, the practical approach involves sending the motor short pulses of current. By varying the width and frequency of the applied pulses, the speed of the motor can be controlled. Controlling the speed of a motor in this way prevents any components from experiencing continuous current stress. Three simple circuits used to provide the desired motor-control pulses are shown in figure above. This technique is called Pulse-Width Modulation, and is particularly useful for low speed control, where DC motors generally have poor torque performance.

In the first circuit (a), a UJT relaxation oscillator generates a series of pulses that drives an SCR ON and OFF. To change the speed of the motor, the oscillatory frequency of UJT is adjusted by changing the RC time constant.

In the second circuit (b), a pair of NAND gates make up the relaxation oscillator section, while power MOSFET is used to drive the motor. Like the previous circuit, the speed of the motor is controlled by the oscillator's RC time constant. Notice that if one of the input leads of the left NAND gate is pulled out, it is possible to create an extra terminal that can be used to provide ON/OFF control that can be interfaced with CMOS logic circuits.

Inefficient Designs



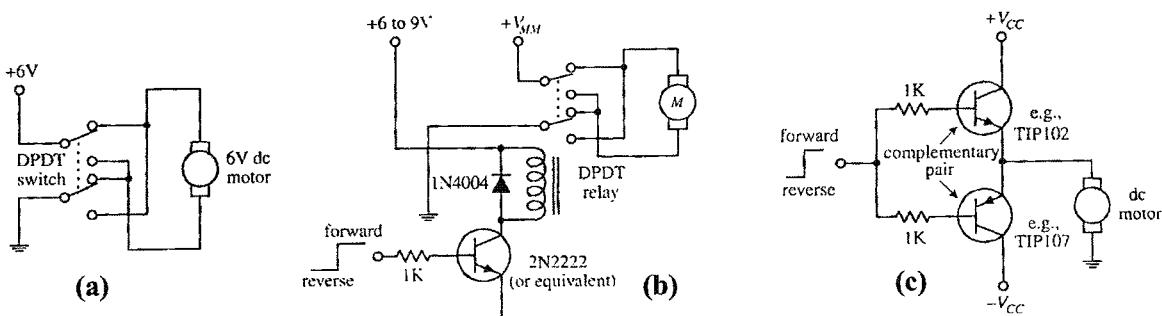


The third circuit (c) is a 555 timer that is used to generate pulses that drive a power MOSFET. By inserting a diode between pins 7 and 6, as shown, the 555 timer is placed into low-duty cycle operation. R_1 , R_2 , and C set the frequency and ON/OFF duration of the output pulses.

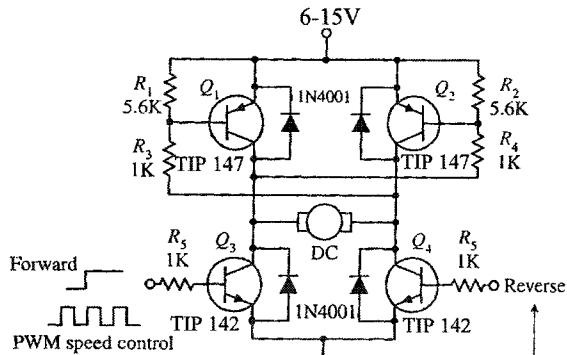
6.2.2.5 Directional Control of DC Motors

By changing the polarity of motor leads the direction of a motor can change. A simple manual-control approach is to use a DPDT switch, as shown in figure below (a). Alternately, a transistor-driven DPDT relay can be used as in the circuit (b). The relay can be eliminated by using a push-pull transistor circuit such as (c) in figure above. This circuit uses a complementary pair of transistors (similar betas and power rating). One is an NPN power Darlington, and the other is a PNP power Darlington. When a positive voltage (e.g., +5 V) is applied to the input, the upper transistor (NPN) conducts, allowing current to pass from the positive supply through the motor and into ground. If a negative voltage is applied to the input, the lower transistor (PNP) conducts, allowing current to pass through the motor from ground into the negative supply terminal. Relay circuits are typically used for low-speed operations, and they have a low rate of power loss, while transistor circuits are efficient for high-speed operation but with more loss of power.

A common circuit for controlling both direction and speed of a DC motor is the so-called *H-bridge* circuit. Figure above shows two simple versions of the H-bridge circuit. The left H-bridge circuit is constructed with bipolar transistors,

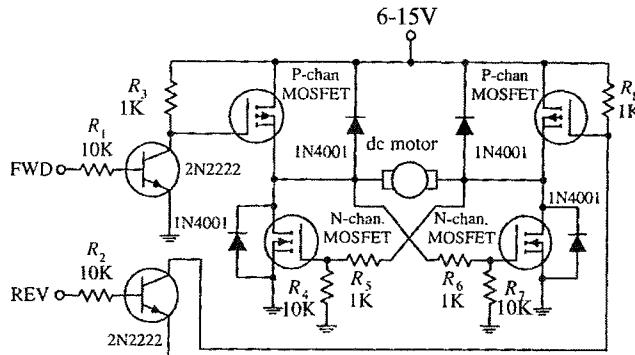


Bipolar H-Bridge



A signal must not be applied here when a signal is being applied to the forward lead.

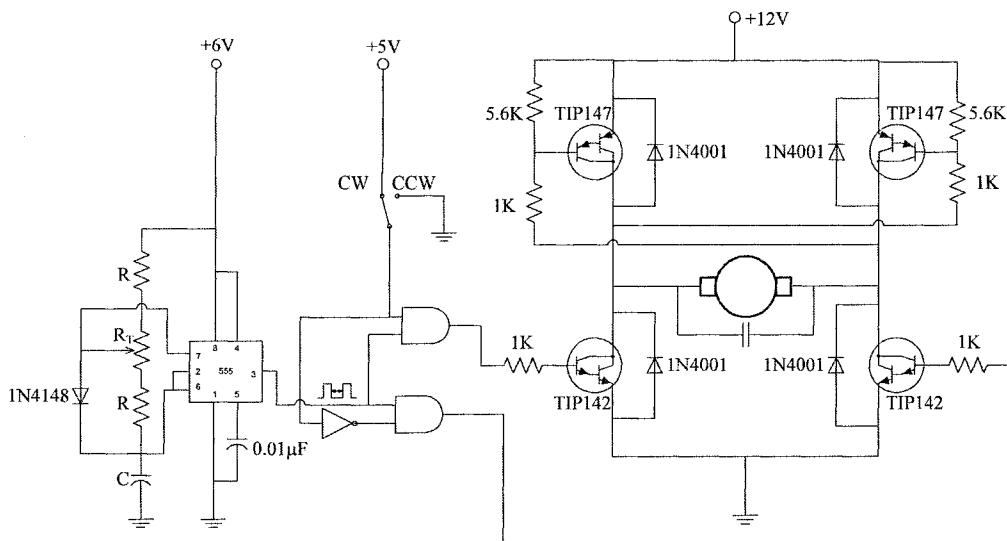
MOSFET H-Bridge

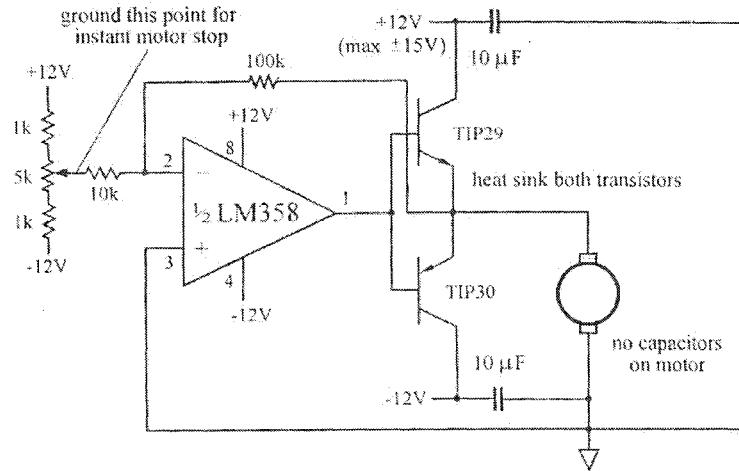


whereas the right H-bridge circuit is constructed from MOSFETs. To make the motor rotate in the forward direction, a high (+5V) signal is applied to the forward input, while no signal is applied to the reverse input (applying a voltage to both inputs at the same time is not allowed). The speed of the motor is controlled by pulse-width modulating the input signal. In the bipolar H-bridge, when a high voltage is applied to the base of Q_3 , it conducts, which in turn allows the PNP transistor Q_2 to conduct. Current then flows from the positive supply terminal through the motor in the right-to-left direction (*forward direction*). To reverse the motor direction, the high voltage signal is removed from the base of Q_3 , and placed on the base of Q_4 . This sets Q_4 and Q_1 into conduction, allowing current to pass through the motor in the opposite direction. The MOSFET H-bridge works in a similar manner. The diodes within the H-bridge circuits help dampen transient spikes that are generated by the motor's coils so that they do not damage the other components within the circuit. All transistors (except the bipolar within the MOSFET circuit) should have high power ratings.

An advantage of the H-bridge driver circuit is that a Pulse Width Modulated (PWM) signal can be inputted to the bridge to vary the motor speed. By increasing the duty cycle of the input PWM the average DC voltage increases, and so does the motor speed. The PWM frequency should be significantly larger than the "mechanical resonance frequency" of the motor, i.e., inverse of the mechanical time constant, so that the motor actually does not "sense" the pulses directly. High PWM frequencies can, however, cause voltage "spikes" at the motor leads, which will in turn prevent the shaft from rotating effectively. Using a capacitor across the motor leads can lesson this problem. The range of the applicable PWM frequency depends on the motor characteristics, and it is usually 5-20KHz for commercial servomotors. Figure below shows a bipolar H-bridge with the pulse generator and logic for driving the motor in both directions.

Making an H-bridge circuit is not a difficult task. However, it is sometimes useful to know that there are several DC motor-driver IC's available in the market. For example, National Semiconductor's LMD18200 motor-driver IC is a high-

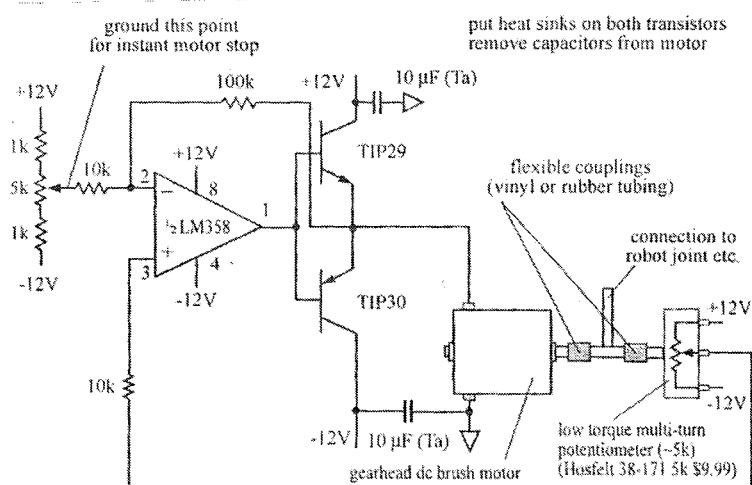




current, easy-to-use H-bridge chip that has a rating of 3 A and 12 to 55 V. This chip is TTL and CMOS compatible and includes clamping diodes, shorted load protection, and a thermal warning interrupt output lead. The L298 chip, from Unitrode, is another popular motor-driver IC. This chip is easy to use and is cheaper than the LMD18200 IC, but it cannot handle as much current, and does not provide as many additional features. Further, some "prefab" motor-driver boards are also available that are capable of driving a series of motors.

In some applications, it is important to stop the motor immediately, thus a brake mechanism is needed. Although mechanical brakes can always be used, an H-bridge circuit that can short both ends of the DC motor to ground can also be an option. Note that the H-bridges shown in the previous page do not have brake capability. An excellent brake circuit is a driver circuit with feedback, such as the one shown in figure below, can also have an excellent braking performance. The above circuit is a bi-directional variable speed DC brush motor controller that has almost instant start and stop performance. About 2 V is dropped across the driver, but torque loss between 12V and 10V operation is often a reasonable trade off for circuit simplicity. Tantalum capacitors are recommended for decoupling power supply lines, but electrolytics may be used if higher noise can be tolerated on the power buss. Both transistors may need heat sink when operating motors at low speeds. Pin 2 is driven by a voltage divider in the above figure, but input to pin 2 may be interfaced to a microcontroller. With the addition of an external feedback potentiometer, the circuit can become a servo-controlled unit. "Snappier" operation is obtained by increasing the resistor feedback ratio (e.g., making 100K larger). However, if the feedback ratio is too large, motor control will be erratic and overshoot will occur. For higher-voltage motors (24 V or more), the use of a high-voltage amplifier, such as MC1436, is essential in place of LM358.

The above driver circuit, when applied to a geared DC brush motor, can be converted to a servo-controlled motor, by adding appropriate feedback from a suitable position control sensor, such as a multi-turn potentiometer, as shown in figure below. In this configuration, the slightest change in voltage at pin 2 is translated into proportional rotation of the motor's output shaft.



6.2.2.6 Torque Regulator

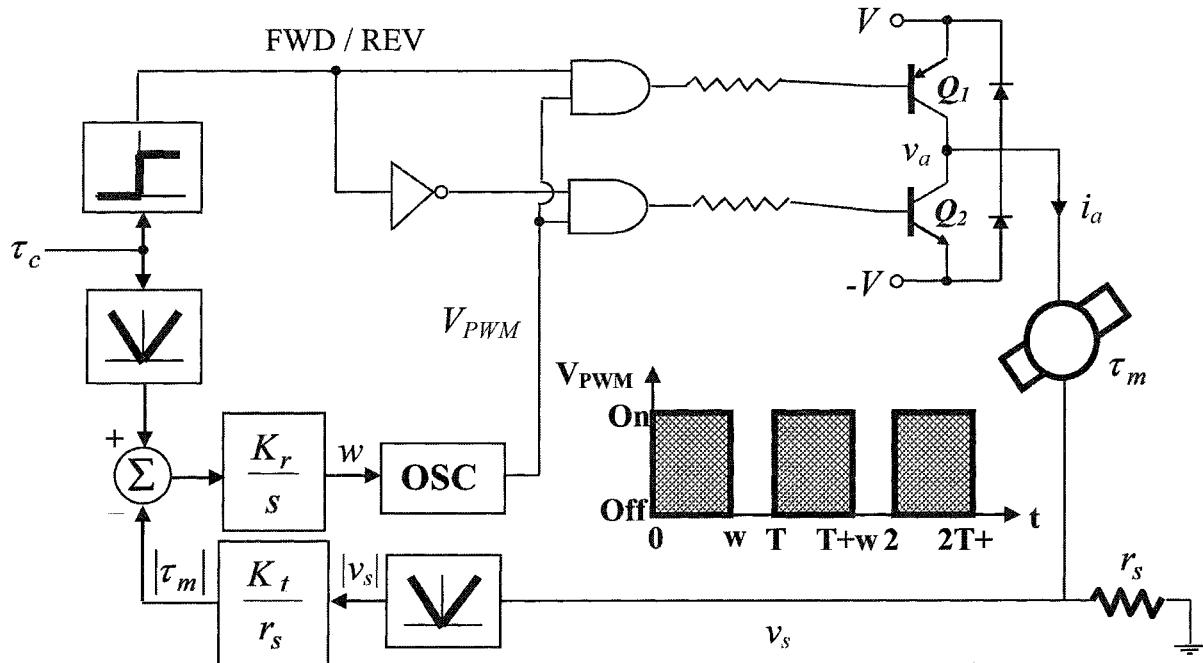
The torque of a DC motor is usually controlled indirectly by varying the applied armature voltage v_a . To regulate the torque (or armature current), a small current-sensing resistor r_s (a value of $r_s \leq 0.1r_a$ is usually satisfactory) is inserted in series with the armature winding in order to generate a feedback voltage v_s as shown in the circuit below. The key control element in the feedback system is the oscillator OSC. The output of the oscillator, V_{PWM} , is a *pulse-width-modulated* signal (PWM) with pulse width w and period T , where T is selected to be small (less than one twentieth) compared to the motor time constant $T_m = J_m/b_m$. The signal V_{PWM} switches drive transistors Q_1 and Q_2 on and off. Depending upon the sign of τ_c , one of the transistors will be turned off while the other is pulsed on and off by V_{PWM} . Thus, the sign of τ_c controls the direction in which the motor shaft turns, while the pulse width w controls the torque τ_m developed at the shaft. Torque is controlled by changing the average value of the armature voltage $v_a(t)$. Given a DC supply voltage of V , the average value of the magnitude of the armature voltage is

$$\bar{v}_a = \frac{wV}{T}$$

where $0 \leq w \leq T$. It is the average armature voltage, or DC component of $v_a(t)$, that is important, because a DC motor is essentially a low-pass filter that removes the fundamental and higher harmonics of the periodic signal $v_a(t)$ as long as the PWM period T is small in comparison with the motor time constant T_m . If $0 \leq w \leq T$, the motor torque τ_m is equal to the control torque τ_c in the steady state. When the pulse width exceeds the period, saturation happens and no further torque can be transferred to the motor. Thus, saturation of the PWM signal arises when the magnitude of the control torque τ_c exceeds the maximum torque τ_{max} that can be delivered to the load shaft. A DC motor produces maximum torque when the motor *stalls*; hence,

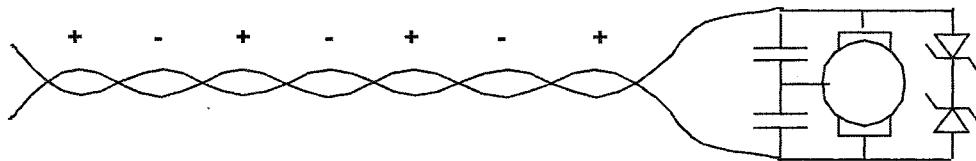
$$\tau_{max} = \tau_{m0} = \frac{K_t v_{a0}}{r_a}.$$

The control torque must satisfy $|\tau_c(t)| \leq \tau_{max}$ to avoid saturation of the PWM torque regulator circuit. This typically means that the speed of load shaft may have to be reduced, and consequently, the control gains cannot exceed beyond a limit, because otherwise more control torque may be required than the actuator is capable of delivering.



6.2.2.7 Noise Reduction Techniques

Typical problems with the existence of motors in the circuit are high frequency voltage spikes and electromagnetic (EM) interference. Some techniques to reduce these effects are shown in figure below. Capacitors leading from each motor terminal to the metal case of the motor will assist quenching spikes right after the motor. However, note that placing these capacitors even inches away will make the leads from the motor to those capacitors antennae broadcasting EM interference. Two head-to-head Zener diodes (especially for 12V and 18V supply) will clip voltage spikes. The leads of the motor should then be twisted so that magnetic field generated around the wires flip sign with every twist. At a distance, this is a line of dipoles and their field characteristic reduces faster with distance. Furthermore, in cases where the noise persists, the leads can be wrapped with a grounded conductive shield such as a tinfoil. If tinfoil is used, cut it into long narrow strips and carefully wrap it around the motor leads with a 50% lap. Make sure to connect the shield to ground or it will become an antenna as well. If the noise still persists, examine other routes of entry from the source to the undesired destination including common power supplies, test equipment, geometry, people's hands, etc.



6.2.3 Brushless DC Motors

A major maintenance problem in conventional DC motors is brush arcing. In brush DC motors, the magnetic polarity of the stator is fixed, and the polarity of the rotor is switched mechanically to obtain proper direction of motor torque. The armature voltage is supplied by a pair of brushes that maintain contact with split slip ring commutation. Brushes are the weak factors in DC motors, and they generate excessive noise, contact bounce, and maintenance problems due to rapid wear out. Brushless DC motors prevent brush arcing by putting the permanent magnet in the rotor and energizing the stator through angular positions.

Modern brushless DC motors use solid-state switching for commutation. In these motors, electrical commutation duplicates mechanical brush commutation. In brushless DC motors, the polarity of the rotor unit, which is a permanent magnet, is fixed relative to the rotor itself, and the polarity of the stator is switched by electronic means to achieve the same objective. Because the electrical commutation simulates the mechanical commutation in conventional systems, brushless DC motors exhibit similar torque-speed characteristics.

The advantages of brushless DC motors include high reliability and ability to generate relatively high torque at speeds up to 100,000rpm. Brushless DC motors are used in general-purpose applications as well as in servo systems for motion control applications. Motors in the range up to 1hp, operating at speeds up to 7,200rpm, are used in computer peripherals and also as drivers for fluid power devices.

Brushless DC motors are similar to AC synchronous motors, with the only difference in the type of back Electro-Magnetic Force (back EMF) produced. While an AC synchronous motor develops a sinusoidal back EMF, a brushless DC motor produces a rectangular or trapezoidal back EMF.

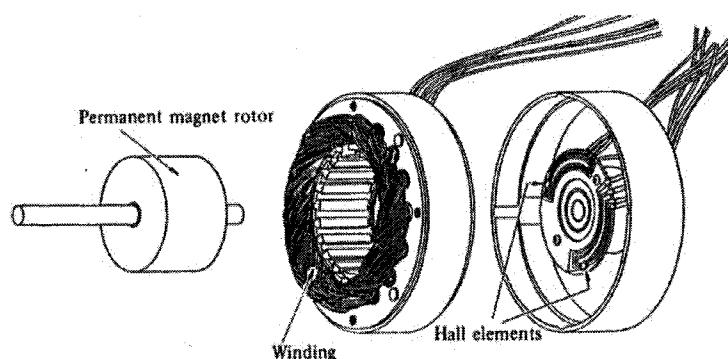
In general, a brushless DC motor is similar to a typical brush DC motor with the difference of having the role of rotor and stator reversed. In other words, the windings of the motor are attached to the stator while the permanent magnets are bonded to the rotor. Because the wires on the windings do not move, there is no need for brushes to transfer the electrical signal.

6.2.3.1 Construction of a Brushless DC Motor

There are two types of brushless DC motors: in-runner and out-runner. The difference between these two types of motors is not only in the physical construction but also in the motor characteristics.

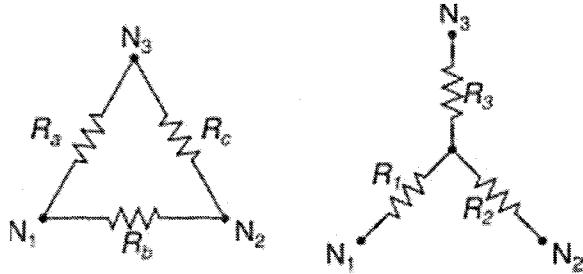
The in-runner brushless motor is the most common configuration, and it is constructed with the permanent magnets located on the rotor, and the windings of the motor are attached to the stator. On the out-runner or external rotor configuration, the rotor with the magnets spins on the outside surrounding the stator core.

In general, in-runner motors are good for applications where high speed is required, and they are more efficient than the out-runner motors although they produce more noise. Out-runner motors are used when high torque is required.



Winding Configuration

Brushless DC motors can also have two different winding configurations. The windings can either be Delta or Wye configured. In the Delta configuration the windings are connected in a triangular shape; in this configuration two windings are connected to a common node at all times. The Wye configuration connects all windings of the motor into a common node.



The main difference between the two configurations is the torque-to-speed ratio. A Delta configuration will provide low torque at low speeds, but also have the advantage of high peak speed. A Wye configuration provides high torque at low speeds, but with a lower peak speed. In broader terms, a Wye configuration is more efficient than Delta configuration, because they have less loss as they do not have a closed loop – no parasite currents.

The winding configuration does not affect the type of controller required to drive the motor. In other words, both winding configurations are treated the same way when driving the motor.

Hall-effect Sensors

In order to provide the electrical communication required for the brushless motor to run, most manufacturers supply motors with three Hall-effect position sensors, each delivering an alternating binary high and low as the rotor turns. The three sensors are off-set so that each aligns with one of the fields that is generated by one of the wound stator poles.

6.2.3.2 Commutation of Brushless DC Motors

Commutation in the brushless DC motor is done electronically, and there are numerous ways of achieving this effect. Commutation in a brushless DC motor implies a switching action between the windings of the motor, since this type of motor does not have a physical commutator. In order to achieve a good commutation, the motor requires a feedback mechanism to determine its position. Although it is possible to have a brushless DC motor commutated without sensors, the feedback is usually achieved with the use of an electric or magnetic encoder or with a Hall-effect sensor. Thus, brushless DC motors can be classified in Sensored and Sensorless types.

Sensored Brushless DC Motors

As the name implies, sensored brushless DC motors have some feedback system that allows the motor to detect the rotor position at all times. A sensored brushless motor is mainly used when the starting torque varies considerably or when high amounts of torque are required at startup. The use of sensors to detect position is of great advantage for low-speed applications and for determining the right phase sequence during startup. Similarly, a feedback mechanism provides the motor controller and the motor itself the synchronization needed at all speeds.

Several methods can be used to obtain the required feedback to the controller but the most common mechanisms include Hall-effect sensors and Encoders.

Sensorless Brushless DC Motors

An alternative way of driving brushless DC motors is without any kind of physical feedback. Sensorless drivers do not use physical sensors, but they use intrinsic motor properties to achieve the required feedback. This method is usually employed when torque is not critical with the additional advantage of providing a low-cost means of driving the motor. Driving a sensorless brushless motor implies that the controller does not know the position of the rotor, which makes it harder for the motor to start up since no sequence is specified by the controller.

A major disadvantage of sensorless brushless motors is the possibility of cogging on the motor when the controller attempts to provide the right sequence to a different phase. This is similar to providing an incorrect sequence to a stepper motor. Once the motor reaches some speed the cogging will disappear.

In order to determine the correct sequence for the motor windings, the sensorless brushless DC motor uses the back EMF effect. In this method two phase coils are energized to set the rotor to a known position. The coils are then energized based on a certain sequence, and the motor will gradually increase its speed while decreasing the commutation period. At the same time, the coil that is not being energized is monitored for back EMF.

In comparison, sensored brushless DC motors are slightly faster and more reliable than their sensorless counterparts, but their use depends merely on their applications.

CAUTION: It is possible to drive a sensorless or sensored brushless DC motor with a sensorless brushless DC motor driver, but it is NOT possible to drive a sensorless motor with a sensored driver.

6.2.3.3 Hall-effect Sensors in Brushless DC Motors

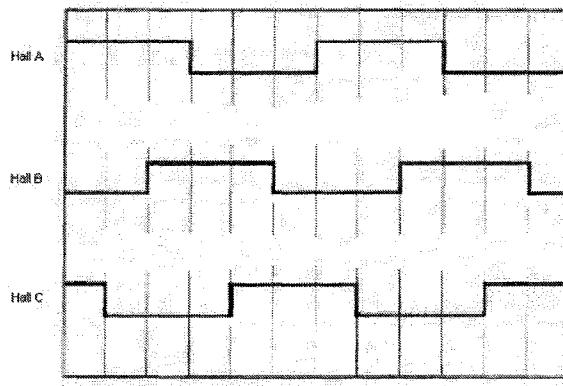
A Hall-effect sensor generates a voltage by detecting a change in the magnetic field. Hall-effect sensors are mainly used as proximity switches, speed detection or positioning sensors. For brushless DC motors the Hall Effect sensor serves as a feedback mechanism capable of detecting the position of the rotor at any moment. The number of phases in a brushless motor determines the number of sensors required, thus a three-phase motor will require three Hall Effect sensors, one for each phase.

A Hall Effect sensor usually works in conjunction with a transistor acting as a switch. The sensor produces a change in voltage each time a magnet on the motor passes by and delivers this change in potential to the base of the transistor. In turn, the transistor closes the circuit and allows current to flow to the next phase sequence to the motor specified by the controller.

Hall Effect sensor IC's come in different packages, and most of them include a voltage regulating mechanism to allow them to work in a variety of voltage ranges (usually from 5 to 24 V). Although many brushless motors include Hall Effect sensors as an integral component, separate commercial IC's, such as US5881, are also available.

When using a microcontroller or a special driver IC, the Hall Effect sensors provide the correct input to sequence the motor. The following graph is a representation of the input provided by the sensors.

The combination of Hall-effect sensors A, B and C provides the controller with the exact position of the motor. For example, on the first rotation of the motor, sensors A and C are high and sensor B is low providing a "101" logic. Based on this logic the driver circuit turns ON the windings A and B in the motor. As the sensors continue sequencing the coils alternate accordingly to the sensor inputs.

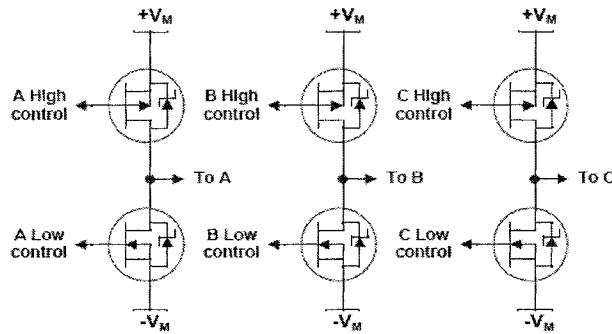


6.2.3.4 Driving Brushless DC Motors

Driving a Brushless DC motor is more complicated than driving a brush DC motor because of the electronic commutation requirement. The circuit is usually divided in two main sections: the controller and driver sections.

The controller section is in charge of receiving any signals from the sensors, process the information to obtain the order of commutation and deliver the right signals to the driver circuit. The controller portion of the circuit can be achieved with the use of a microcontroller or a brushless pre-driver IC. It is possible to obtain the right logic with the use of shift registers and some logic gates, but caution must be taken when implementing the circuit.

The driver section consists of three half-bridge drives, each connected to a phase of the motor. The upper section of the bridge (high side) is connected directly to the power line, and the lower section (sink side) is connected to ground. The microcontroller or IC delivers the correct signals to sequence the motor. Typically the driver circuit can be accomplished using either NPN transistors or MOSFETs acting like electronic switches.



Speed Control of Brushless DC Motors

Similar to a brush DC motor, the speed of a brushless DC motor can be controlled by changing the applied voltage or by using PWM. Using PWM to control the speed in a brushless DC motor increases the efficiency of the motor by 30 to 40%. Also, PWM can limit startup current, thus controlling the torque of the motor.

When using PWM to control the speed of a brushless DC motor it is important to maintain a high frequency. The rule of thumb is to have a frequency of about 10 times the motor's frequency. When the duty cycle of PWM is varied within the sequences, the average voltage supplied to the stator reduces, thus reducing the speed.

Usually PWM can be applied to the high side of the driver circuit only, while maintaining the lower side of the circuit at either high or low depending on the required sequence.

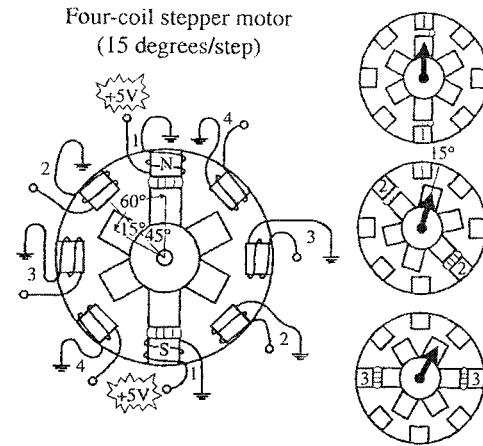
Direction Control of Brushless DC Motors

Similar to stepper motors (See next section), controlling the direction on a brushless DC motor can be achieved by reversing the sequence sent to the driver circuit. Due to the electrical commutation, the microcontroller or IC receiving the sensor information must also receive the information in the reverse order.

6.2.4 Stepper Motors

In recent years, the stepper motor has emerged as a cost-effective alternative to DC motors in motion control applications. The stepper motor is an actuator that translates electrical pulses into precise, equally spaced angular movements of the rotor in the form of steps. It is, indeed, a special type of brushless DC motors, in which the rotor is positioned by magnetically aligning the rotor and stator teeth that occur when the air gap between the two sets of teeth is minimized and aligned.

Figure right illustrates a simplified model of a 15°-per-step stepper motor. The stationary section of the motor, called the *stator*; has eight poles that are spaced 45° apart. The moving section of the motor, called the *rotor*, is made from a ferromagnetic material (a material that is attracted to magnetic fields) that has six teeth spaced 60° apart. To make the rotor turn one step, current is applied, at the same time, through two opposing pole pairs, or coil pairs. The applied current causes the opposing pair of poles to become magnetized. This in turn causes the rotor's teeth to align with the poles, as shown in the figure. To make the rotor rotate 15° clockwise from this position, the current through coil pair 1 is removed and sent through coil pair 2. To make the rotor rotate another 15° clockwise from this position, the current is removed from coil pair 2 and sent through coil pair 3. The process continues in this way. To make the rotor spin counterclockwise, the coil-pair firing sequence is reversed. By arranging different sequences and teeth configurations a wide variety of steppers with different degrees per step, from 0.1° to 90°, can be obtained.

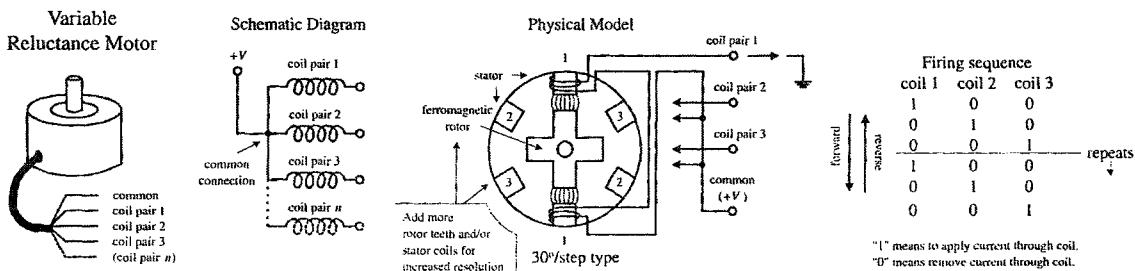


Stepper motors also differ from DC motors in their torque-speed relationship. DC motors generally are not very good at producing high torque at low speeds, without the aid of a gearing mechanism. Stepper motors, on the other hand, work in the opposite manner. They produce the highest torque at low speeds. Stepper motors also have another characteristic, holding torque, which is not present in DC motors. Holding torque allows a stepper motor to hold its position firmly when not turning. This can be useful for applications where the motor may be starting and stopping, while the force acting against the motor remains present. This eliminates the need for a mechanical brake mechanism. For a majority of motion control applications, stepper motors provide a low-cost alternative. The major component of cost is the drive circuit. Steppers do not simply respond to a clock signal, they have several windings that need to be energized in the correct sequence before the motor's shaft will rotate. Reversing the order of the sequence will cause the motor to rotate the other way. If the control signals are not sent in the correct order, the motor will not turn properly. It may simply buzz and not move, or it may actually turn, but in a rough or jerky manner. A circuit that is responsible for converting step and direction signals into winding energization patterns is called a *translator*. Most stepper motor control systems include a *driver* in addition to the translator, to handle the current drawn by the motor windings.

Stepper motors are very well suited for use in open-loop applications because of their accuracy and non-cumulative position error characteristics. The stepper motor is inherently a discrete device; so it is easy to control from a digital computer algorithm, stability is rarely an issue, and the brushless design results in less wear. Compared to DC servomotors, stepper motors produce considerably less torque, lower speeds, and higher vibrations, and they require more complicated driving software or hardware. Nonetheless, for many applications their benefits outweigh their drawbacks.

6.2.4.1 Stepper Motor Types

Two basic types of stepper motors are: *Variable Reluctance (VR)* and *Permanent Magnet (PM)* stepper motors. PM motors are also either *Unipolar* or *Bipolar*. There are also *hybrid* PM stepper motors, which are indistinguishable from unipolar or bipolar permanent magnet motors from the controller's point of view.

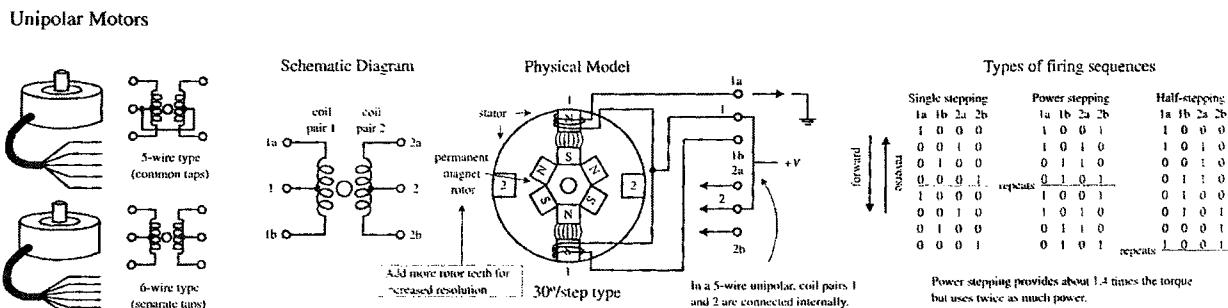


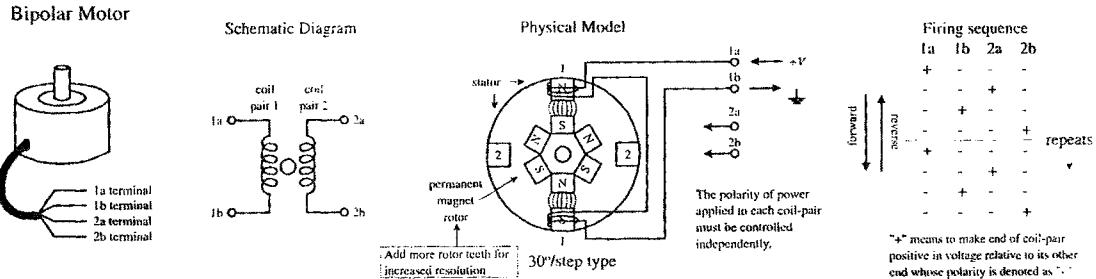
Variable Reluctance Steppers

In VR motors the stator windings are excited in a sequence that will cause the rotor to align to a position that minimizes magnetic reluctance between the stator and rotor. Figure above shows the physical model and schematic diagram of a 30°-per-step variable-reluctance stepper. This stepper consists of a six-pole (or three-coil pair) stator and a four-toothed ferromagnetic rotor. Variable-reluctance steppers with higher angular resolutions are constructed with more coil pairs and/ or more rotor teeth. Notice that the ends of all the coil pairs are joined together at a common point. This joining of the coil ends occurs internally within the motor case. The common and the coil pair free ends are brought out as wires from the motor case. These wires are referred to as the *phase wires*. The common wire is connected to the supply voltage, whereas the phase wires are grounded in sequence according to the table shown in figure above. Variable reluctance stepper motors are the simplest to control over other types of stepper motors. Their drive sequence is simply to "energize" each of the windings in order, one after the other (see drive pattern table above). This type of motor feels like a DC motor when the shaft is spun by hand; it turns freely and you cannot feel the steps. This type of stepper motor is not permanently magnetized like its unipolar and bipolar counterparts.

Unipolar Permanent Magnet Stepper Motors

Unipolar PM steppers have a similar stator arrangement as the variable-reluctance steppers, but they use a permanent-magnet rotor and different internal wiring arrangements. Figure below shows a 30°-per-step unipolar stepper. It consists of a four-pole (or two-coil pair) stator with center taps between coil pairs and a six-toothed permanent-magnetic rotor. The centre taps may be wired internally and brought out as one wire or may be brought out separately as two wires. The centre taps typically are wired to the positive supply voltage, whereas the two free ends of a coil pair are alternately grounded to reverse the direction of the field provided by that winding. As shown in the figure, when current flows from the centre tap of winding 1 out terminal 1a, the top stator pole "goes north," while the bottom stator pole "goes south." This causes the rotor to snap into position. If the current through winding 1 is removed, sent through winding 2, and out terminal 2a, the horizontal poles will become energized, causing the rotor to turn 30°, or one step. In figure below, three firing sequences are shown. The first sequence provides full stepping action, which was discussed above. The second sequence, referred to as the *power stepping sequence*, provides full stepping action with 1.4 times the torque but twice the power consumption. The third sequence provides half stepping (e.g., 15° instead of the rated 30°). Half stepping is made possible by energizing adjacent poles at the same time. This pulls the rotor in between the poles, thus resulting in one-half the stepping angle. Unipolar steppers with higher angular resolutions are constructed with more rotor teeth. Also, unipolar steppers come in either five- or six-wire types. The five-wire type has the center taps joined internally, while the six-wire type does not.





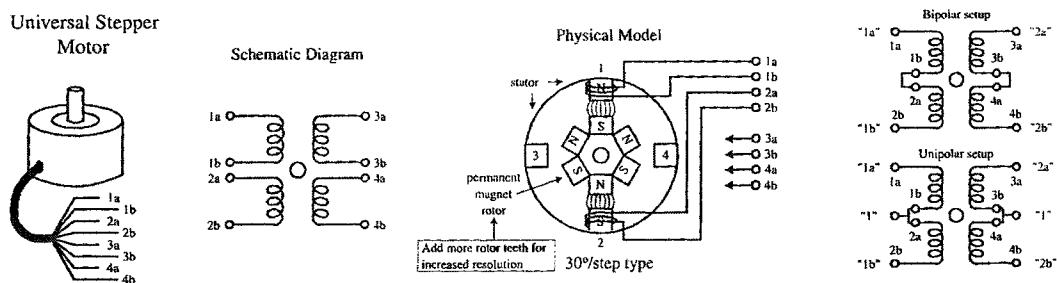
Bipolar Permanent Magnet Stepper Motors

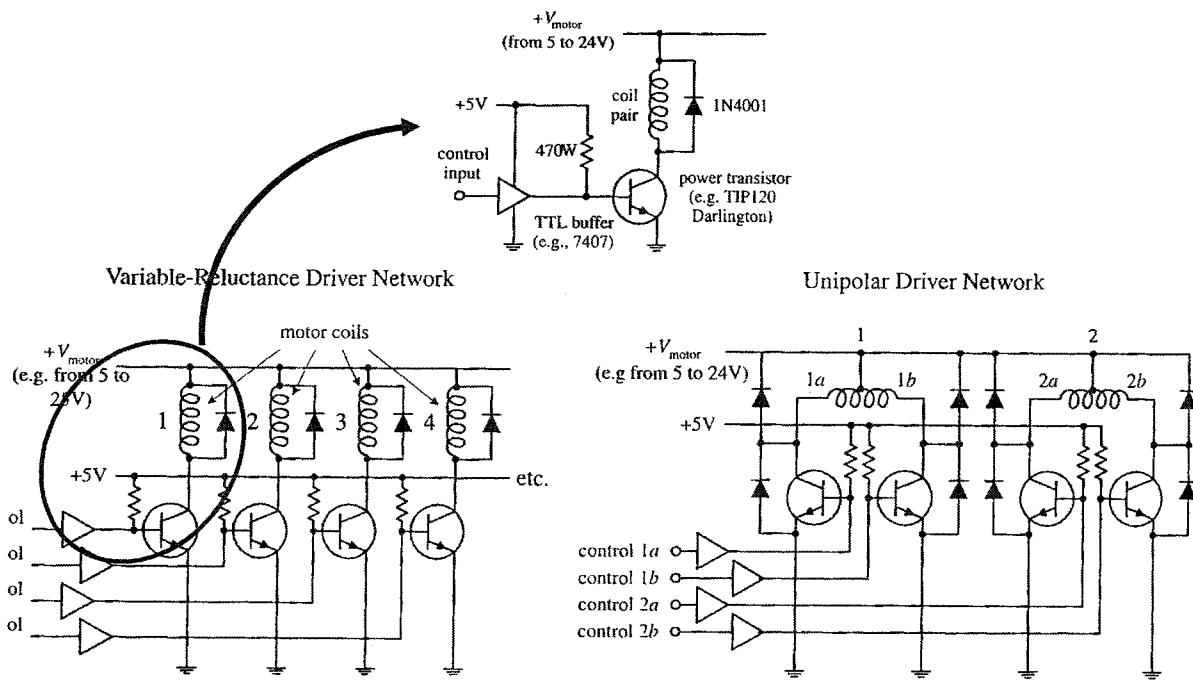
These steppers resemble unipolar steppers, but their coil pairs do not have centre taps. This means that instead of simply supplying a fixed supply voltage to a lead, as was the case in unipolar steppers (supply voltage was fixed to centre taps), the supply voltage must be alternately applied to different coil ends. At the same time, the opposite end of a coil pair must be set to the opposite polarity (ground). For example, in figure above, a 30°-per-step bipolar stepper is made to rotate by applying the polarities shown in the firing sequence table to the leads of the stepper. Notice that the firing sequence uses the same basic drive pattern as the unipolar stepper, but the “0” and “1” signals are replaced with “+” and “-“ symbols to show that the polarity matters. The circuitry used to drive a bipolar stepper requires an H-bridge network for every coil pair. Bipolar steppers are more difficult to control than both unipolar steppers and variable-reluctance steppers, but their unique polarity-shifting feature gives them a better size-to-torque ratio. Bipolar steppers with higher angular resolutions are constructed with more rotor teeth.

Hybrid Permanent Magnet Stepper Motors

Hybrid (or universal) PM steppers represent a type of unipolar-bipolar motors. A universal stepper usually comes with four independent windings and eight leads. By connecting the coil windings in parallel, as shown in figure below, the universal stepper can be converted into a unipolar stepper. If the coil windings are connected in series, the stepper can be converted into a bipolar stepper.

In general, permanent magnet motors have a smaller step than variable reluctance motors, typical values being 1.8 degrees in PM steppers versus 15 degrees in VR steppers, which makes them more suitable for accurate positioning applications. However, the torque per unit volume of the PM motors is considerably lower than that of the VR motors. Typical torque ranges for PM motors are usually under 500 oz-in, and for VR motors under 2000 oz-in. This limits the range of applications for PM motors to a lower-torque region than that of VR motors. As a result, PM motors are available in smaller standard sizes (commercially known as size 23 or size 34). For example, a four-phase, size 23 motor typically produces under 100 oz-in. of torque with a speed range of up to 30,000 steps per second (sps), whereas a size 34 motor produces roughly three times the torque at one-third the speed.





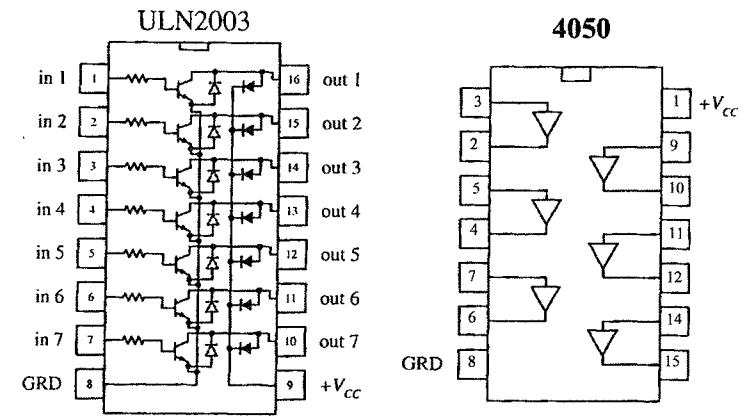
6.2.4.2 Driving Stepper Motors

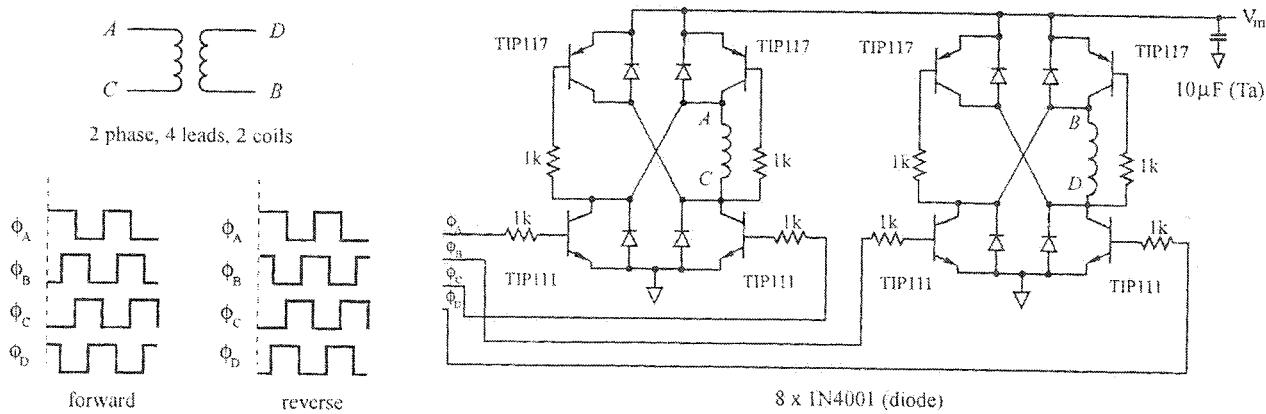
Running a stepper motor requires a driver circuit that can control the current flow sent through the coils within the stepper stator. The driver, in turn, must be instructed by a software code loaded on a microcontroller (or a PC) or a logic circuit. The control unit, either the logic circuit or the microcontroller, is usually referred to as the *translator*.

Figure above shows the driver networks for a variable-reluctance stepper and for a unipolar stepper. Both drivers use transistors to control current flow through the motor individual windings. In both driver networks, input buffer stages are added to protect the translator circuit from the motor supply voltage in the event of transistor collector-to-base breakdown. Diodes are added to both drivers to protect the transistors and power supply from inductive kickback generated by the motor coils. (Notice that the unipolar driver uses extra diodes because inductive kickback can leak out on either side of the centre tap. As you will see in a moment, a pair of diodes within this driver can be replaced with a single diode, keeping the diode count to four.) The above figure also shows the circuit for a single phase, in which a high-power Darlington transistor, a TTL buffer, and a reasonably fast protection diode (the extra diode should be included in the unipolar circuit) are used.

Some transistor-array ICs, such as the ULN200x series by Allegro Microsystems or the DS200x series by National Semiconductor, can be used to construct the stepper driver circuit. The ULN2003, shown in figure on the right, is a TTL-compatible chip that contains seven Darlington transistors with protection diodes included. The 4050 buffer IC can be used with the ULN2003 to construct a full-stepper driver. Other ICs, such as Motorola's MC1413 Darlington array IC, can drive multiple motor winding directly from the logic inputs.

The driver circuit for a bipolar stepper requires a pair of H-bridges. The H-bridge circuit acts to reverse the polarity applied across each specific coil pair within the stepper. The circuit shown in figure





above uses two H-bridge inverters to reverse coil currents as required for two-phase four-lead bipolar stepper motors. This type of driver circuit, needs only a unipolar power supply (V_m), for motor power. The control signals can be generated by logic gates or microcontrollers, as will be discussed in the next section. Motor direction is reversed when the positions of leads A and C are interchanged, and similarly for leads B and D. Small heat sinks are adequate for even large motors, and small steppers may be run from this circuit for extended periods without any heat sink. TIP111 and TIP 117 are 2amp Darlington transistors, but will run 4 amp motors because they are only operating here on a 50% duty cycle. It is reminded that H-bridges can be purchased as IC packages. The SGS Thompson's L293 dual H-bridge IC is a popular choice for driving small bipolar steppers drawing up to 1 A per motor winding at up to 36 V. The L298 dual H-bridge is similar to the L293 but can handle up to 2 A per winding. National semiconductor's LMD18200 H-bridge IC can handle up to 3 A, and unlike the L293 and L298, it has protection diodes built in.

6.2.4.3 Controlling the Stepper Driver

The proper sequence of commands to the stepper driver is generated through the translator. A translator can be either a PC/microcontroller with software directly generating the outputs needed to control the driver leads or a logic circuit that enacts the sequencing pulses used to drive the stepper driver. Also, In most cases, the translator can be a special IC that is designed to provide the proper firing sequences from its output leads when a clock signal is applied to one of its input leads; another input signal may control the direction of the firing sequence (the direction of the motor).

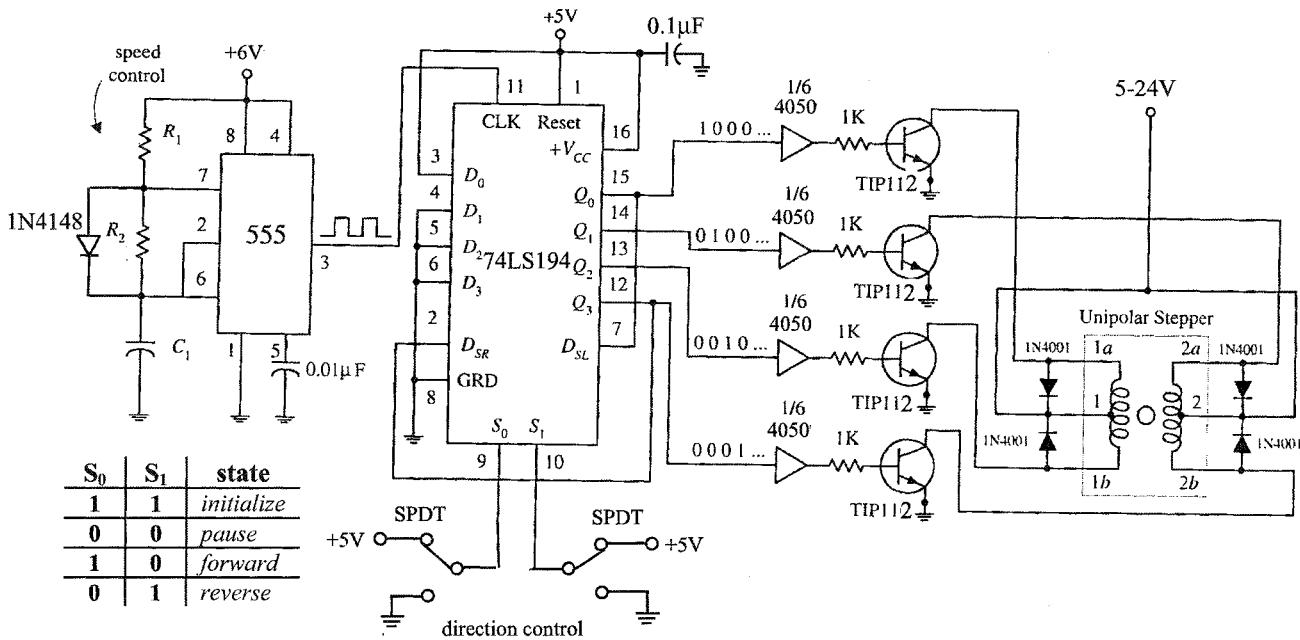
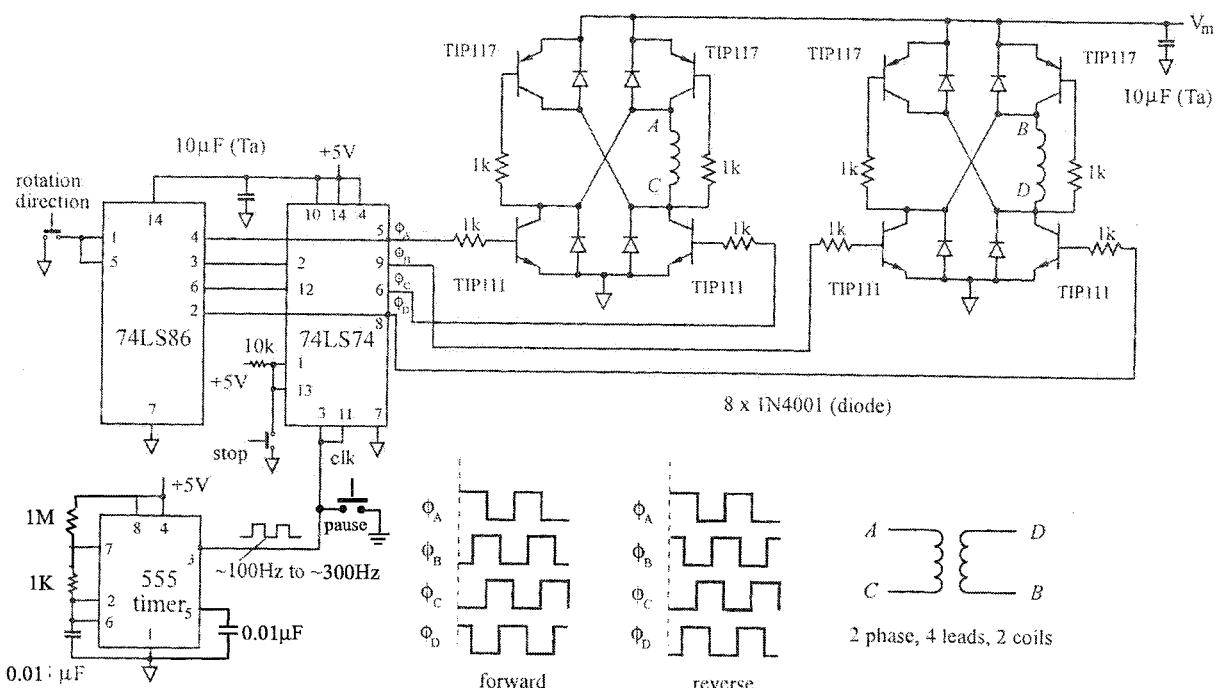


Figure above is an example of using digital logic circuit for controlling the driver of a unipolar stepper. The translator is a TTL 74194 shift counter. The 555 timer provides clock signals to the 74194, while the SPDT switches act to control the direction of the motor. After powering on the chips, both switches should be set high to initialize the shift register. Then, depending on which switch goes low, the motor rotates in one direction. Setting both switches low will pause the motor. The speed of the motor is dependent on the frequency of the clock, which in turn is dependent on the resistors R_1 and R_2 . The translator in this circuit can also be used to control a variable reluctance stepper by changing the driver to the one for a VR stepper, shown in the previous figures.

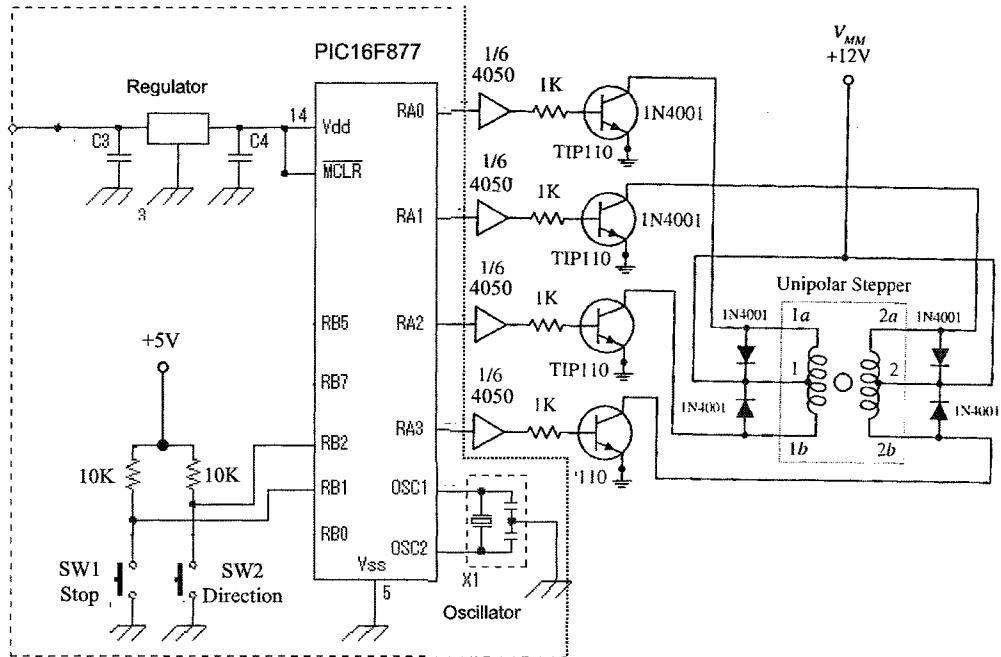
An alternative logic circuit is shown in figure below for controlling a bipolar stepper. In this circuit, phased logic level control signals are generated using the XOR gate 74LS86 and D flip-flop 74LS74 using a clock input. Good holding torque is provided whenever the "stop" mode is invoked at pin 13, and this may be done with a switch or a microcontroller command. Microcontroller logic signals applied to pin 1 are also adequate for changing motor shaft direction instantaneously.



Controlling the stepper driver can also be performed completely through the software. Using a PC or a microcontroller would then eliminate the logic circuit (translator), reducing the cost and simplifying the design. However, if the PC or microcontroller is not fast enough (due to code inefficiency or slow processor speed), or too many motors are driven simultaneously, things can begin to slow down. Interrupts and other system events can plague the control software more in this case. This method of controlling a stepper motor can be particularly useful where the hardware is not critical at first, and a simple interface is needed to allow more time to be spent on the development of the software before the hardware is refined. Figure below shows a schematic of the microcontroller circuit as a translator for the stepper driver.

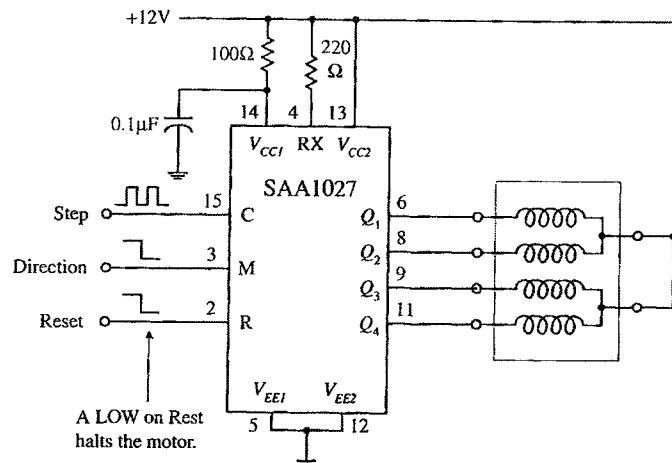
Sequencing signals are sent from port A (RA0-RA3) to the base of 4 transistors (via the buffer). Only 4 bits (in this case lower bits) of the 8-bit port are needed to switch the 4 transistors. In terms of the code for the circuit shown below, sending hex bytes 01, 02, 04, and 08 to the port in succession will cause the stepper motor to rotate 4 steps. After each step, a time delay is needed to allow the shaft to sit in its new position. This time delay assigns the speed of the motor. Two switches are used for stopping the motor and changing its direction using pins RB1 and RB2 as input. More circuits can be built around the PIC to control the motor speed and/or for power or half stepping.

Words of Caution: When making connections to I/O pins of a microcontroller, be sure to isolate the motor well. High voltage spikes of several hundred volts are possible as back EMF from the stepper motor coils. Always use clamping diodes to short these spikes back to the motor's power bus. The use of optical isolation devices (optoisolators) will add yet another layer of protection between the delicate control logic and the potential high-voltage that may be



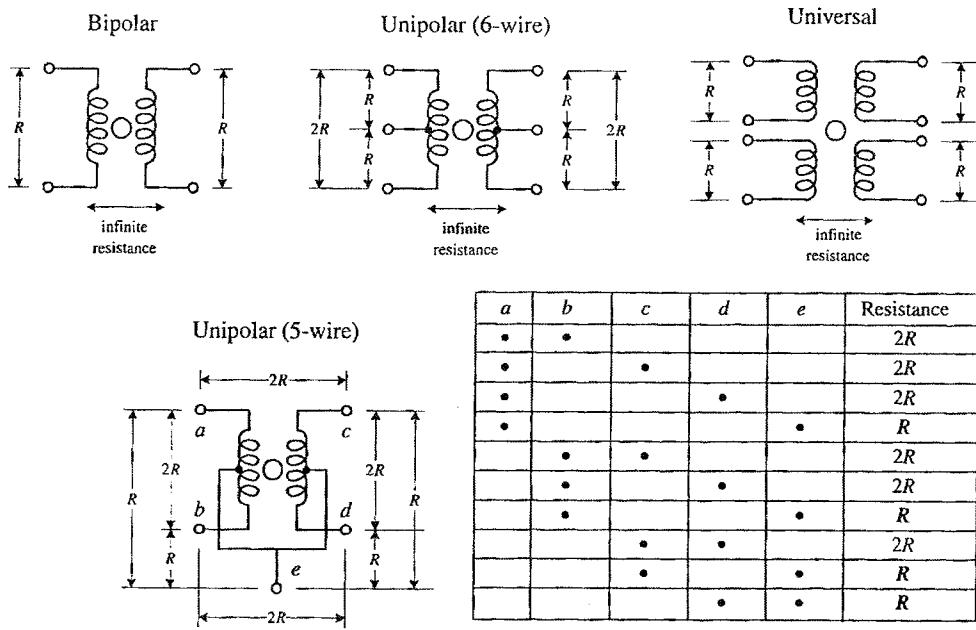
present in the power output stage. Whenever possible, use separate power supplies for the motor and the translator / microcontroller. This further reduces the chance of destructive voltages reaching the controller, and reduces or eliminates power supply noise that may be introduced by the motor.

There exists a number of stepper motor controller ICs that house both the translator and driver modules. A popular chip is SAA1027 from Philips. It is a bipolar IC that is designed to drive four-phase steppers. This chip has high-noise-immunity inputs, clockwise and counterclockwise capability, a reset control input, high output current, and output voltage protection. Its supply voltage runs from 9.5 to 18 V, and it accepts input voltages of 7.5 V minimum for high ("1") and 4.5 V maximum for low ("0"). It has a maximum output current of 500 mA. A driver circuit using this chip is shown in the figure. Pin 15 is the driving pin; a low-to-high transition at this pin causes the outputs to change state. The direction of the motor is controlled through pin 3. A LOW signal at pin 2 resets the counter to zero. An external resistor connector to pin 4 (RX) sets the base current of the transistor drivers. Its value is based on the required output current.



6.2.4.4 Identifying the Stepper Motors

Lacking a label on the motor, you can generally distinguish between PM and VR motors by rotating their shaft by hand when no power is applied. Permanent magnet motors show a "cog-like" resistance as you twist the rotor with your fingers, while variable reluctance motors almost spin freely (although they may cog slightly because of residual magnetization in the rotor). You can also distinguish between the two varieties with an ohmmeter. Variable reluctance motors usually have three (sometimes four) windings, with a common return, while permanent magnet motors usually have two independent windings, with or without center taps. Center-tapped windings are used in unipolar permanent magnet motors. Note that the vast majority of the steppers in the market are permanent magnet.



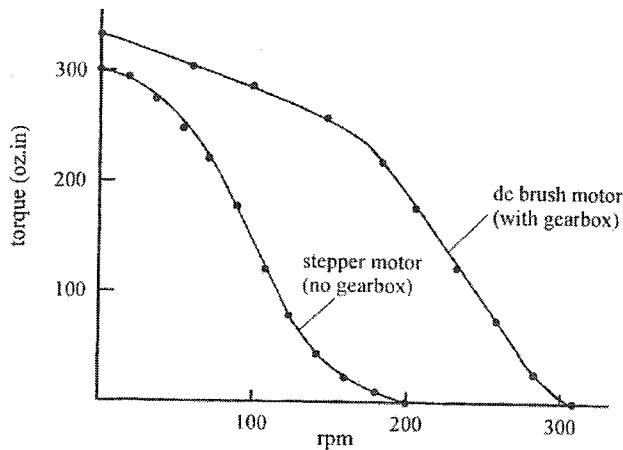
To identify the type of a PM motor, if it has four leads it is most likely a bipolar stepper. If the stepper has five leads then the motor is most likely a unipolar with common centre taps. If the stepper has six leads, it is probably a unipolar with separate centre taps. An eight-lead motor would most likely be a hybrid stepper.

The next step, after figuring the type of the motor, is to specify the leads. This can be easily done by measuring the resistance between various leads. First, determine which wire pair yields a low resistance value. This indicates that the two wires are ends of the same winding. If the two wires are not part of the same winding the resistance should be infinite. For unipolar stepper with 5 or 6 leads, you must continue measuring and comparing the resistance of different combinations of wires. Diagram above is a suitable guide for the identification process.

6.2.4.5 Stepper Motor Specifications

Primary considerations when choosing a stepper are torque, step resolution, size, weight, and cost. For amateurs, coil resistance is also one of the more important parameters because some surplus industrial motors have very low coil resistance, and these require special high current, low voltage power supplies. Three and five-phase motors should also be avoided, because these can only be operated from a dedicated computer stepping routine or an expensive commercial controller. An informed choice can be made after reading all material given in this section, but some important selection criteria are listed below:

- 1) Most stepper motors have larger diameter shafts than are found on the same sized ungeared DC motor. This is because steppers generate high torque at low speeds without the aid of gearing. It is not uncommon for a 1 lb stepper to have a 1/4" diameter shaft and deliver 50 oz.in torque up to 3000 steps/sec. This type of motor may also have a holding torque of 100 oz.in.
- 2) When selecting motors at a surplus store, use an ohmmeter to measure coil resistances, and if possible select motors with coils from 10 to 1000 Ω . Some industrial motors have 1 Ω coils and need large currents at low voltage, and this is inconvenient if only 12V is available. Low voltage steppers can be operated from 12V with appropriate power resistors, but this wastes power and generates heat.
- 3) Buy only two or four-phase motors. These have 4, 5, 6 or 8 leads. Four phase motors are the best choice since their driver circuits are simpler, and they usually have bifilar windings giving better performance. An ohmmeter is needed to tell whether a motor needs a 1, 3, 4 or 5-phase drive. Do not buy 3 or 5-phase motors unless you are prepared to write stepper routines, and also use a dedicated microprocessor or computer to run the motors.
- 4) Step resolution is marked or labeled on most motors. However, if a stepper motor shaft cogs you can count the number of clicks for one revolution, by holding and turning the shaft with fingers.



5) **"When should I use a stepper motor?"** No ironclad rules can be given concerning stepper use, but the following guidelines may be helpful:

- If instant starts, instant stops, instant direction reversal, and precise positioning are required where it is difficult to install an external feedback control element, then choose a stepping motor.
 - If rotation speeds faster than 200rpm with high torque are required, then select a geared DC brush motor.
 - Used with discretion steppers have no equal and their commercial uses are widespread, accounting for about 20% of the total electric motor market. However in a classroom, stepping motors are used in only 5% of projects, mainly due to availability, cost, and because geared DC brush motors are frequently a better option, employing simpler drive circuitry. Steppers cannot attain the very high speeds of DC brush motors, and may lose sync at high rotation rates, where rapid start-stop motions sometimes introduce undesirable resonant behavior.
- 6) **"Does a stepper have the same torque as a DC brush motor?"** From an energy conservation viewpoint, it is possible to design a stepper motor with the same low speed torque as an equivalent brush motor. Both stepper and brush motors will probably need different gearboxes, but the final speed/torque outputs will be comparable in similarly sized packages. In practice, most off-the-shelf steppers for light projects have lower torque than a comparably sized gearhead brush motor. Figure below compares torque/speed performance for two motors with the same dimensions. However, this is an unfair comparison in some respects, because steppers and brush motors are intended to serve different functions in motion control. A stepper delivers high torque at low speeds without a gearbox, so can often be directly coupled to drive a vehicle's wheels or similar mechanisms. In addition step monitoring provides an accurate record of distance traveled, if there is no wheel slippage. Similar performance can be obtained from a geared DC brush motor, and it will probably give superior torque and higher speeds. However, an encoder disk or other device must be installed to compute distance traveled.

6.2.4.6 Thermal Characteristics

One or more coils of a stepper motor are always energized, even when the motor is not moving. If the motor is being driven by an amplifier that acts as a voltage source, the current flowing through the coils will be maximum when the motor is not moving, because there will be no back emf to oppose the applied voltage. Therefore, the maximum heat generation also takes place with the motor stationary. The maximum heat-generation condition is also the worst case for heat removal, because with the rotor not moving, the convection cooling is at a minimum.

Temperature is thus a major operating limit for stepping motors. They are normally designed to run quite hot compared to other types of motors, but even so, temperature rise in the windings must be carefully controlled. In *normal* operating environment, observation of the manufacturer's voltage limits for the motor will usually ensure safe operation. With the thermal limit being the most restrictive operating limit, performance of a motor can be improved by providing more than normal cooling capability. The voltage, and thus the current, supply for the motor can then be increased without exceeding the temperature limit.

6.2.5 Servo Motors

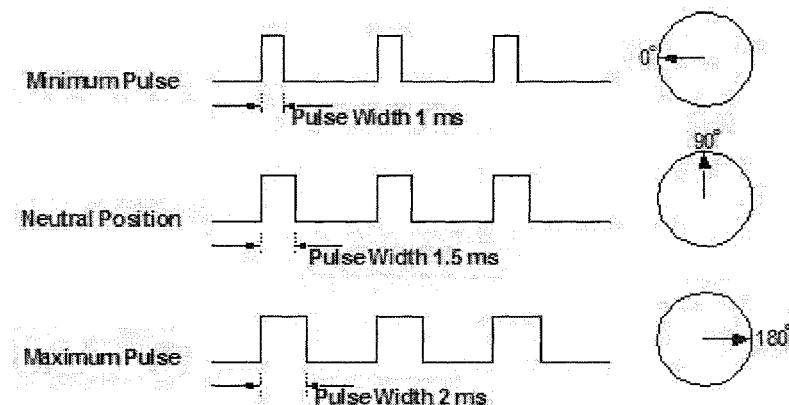
A servo is a device that uses a linear feedback mechanism, usually monitoring speed, current, or position, to control its performance. An example of such a device is the cruise control in a vehicle. A servo motor is a device that converts electrical energy into mechanical energy with the addition of a feedback mechanism, in a form of a potentiometer or optical sensor, to control the rotor displacement. An electric servo motor can consist of an either DC or AC motor. An AC servo motor has the characteristics of handling larger surge currents than a DC servo motor. AC servo motors are usually used when the application requires a fast and accurate response. To obtain these characteristics servo motors have small, high-resistance rotors. A small-diameter rotor results in a low inertia for fast start and stop, while a high-resistance rotor provides a rather linear relationship between speed and torque. Overall, motors are accurate and powerful for their size, and because of their built-in circuitry they are easy to use.

A typical DC servo motor has three wires, one for power voltage (4.5-6 V), one for ground and one for control voltage for sending a series of PWM pulses (3-5 V). The pulse frequency varies for different servo motors, but the typical frequency is 50 Hz. The desired position of the motor shaft is set with the duty cycle of the PWM. The width of the ON portion of the pulse sets the desired position of the shaft. The control circuit then tries to settle at this position by checking the shaft displacement using a potentiometer or encoder. Any error between the current and desired position of the shaft is corrected by generating a series of corrective voltages for the motor using the error signal (Proportional), its summation (Integral) and variation (Differential) in time. Servo motors have a specific range for the shaft position, usually 0 to 180 degrees. When the servo motor is pulsed with low pulse width it is said to have a minimum pulse and the motor will not rotate. A neutral pulse (90 degrees rotation), however, happens when the pulse width is increased. Similarly, a larger pulse width called maximum pulse provides an angle of about 180 degrees (if the motor allows for it). See figure below as an example. Note that the pulse width varies with the size and specification of the motor; therefore, make sure to check the specification before attempting to drive any servo motor.

When a servo is instructed to move to a definite position it will hold its position, despite applying load on the shaft or any other unpredictable situation, until instructed to do otherwise. Therefore, servo motors are good solutions for when accurate positioning of the shaft is required under varying and unpredictable loads. The holding torque in servos is usually higher than that of stepper motors. The maximum amount of torque exerted by the servo motor is represented by the motor torque rating. The servo motor will have to be reminded to stay in its position by continuously sending the correct pulse at a determined frequency; otherwise, it will eventually lose its position. Servo motors are often rated by their maximum holding torque and response time (turn rate), e.g., 0.71 N.m and 0.18 sec/60°.

The main advantages of servo motors include high output power given their size and weight, high efficiency (typically about 90% for light loads), high torque/inertia ratio (fast start/stop), and robust performance against noise, load and ambient variation, etc. On the other hand, servo motors usually require some tuning to stabilize the feedback loop, they are sensitive devices to the control signal due to their complexity, and they usually provide limited shaft displacement (between 0 and 180 degrees).

There are few servo motors in the market, called continuous servo motors, which allow for continuous rotation of their shaft. In these servo motors the control circuit is modified to have (limited) control over the shaft speed, instead of position, using a PWM.



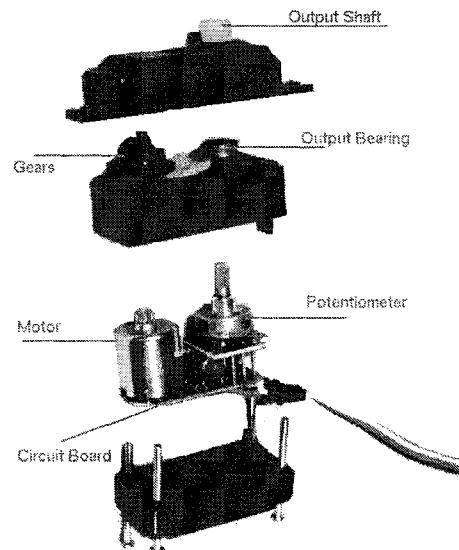
6.2.5.1 Servo Motor Construction

Servo motors are constructed from several modules including motor, gear train, output shaft bearing, feedback device, and control board. The feedback device is typically a shaft displacement sensor such as potentiometer (variable resistor). The motor, through a series of gears, turns the output shaft and the potentiometer simultaneously. The potentiometer signal is fed into the servo control circuit, and when the control circuit detects the correct position, it stops the motor.

The motor and gears in a servo motor vary depending on the required torque and life time. Inexpensive servos have plastic gears that will wear out after nearly 100 hours of use. More expensive servos have metal gears that are much more durable.

The feedback device is usually a potentiometer that is often the first thing to fail in a servo motors. If the potentiometer gets dirty or the contacts get oxidized, the servo will fail to work properly, sometimes by "jittering or hunting" since the feedback is inaccurate, or turning completely to one side and drawing lots of current, since the servo does not know where its output shaft is pointing. More expensive servos have "sealed" potentiometers protecting them against dust.

The last subsystem is the output shaft bearing. Cheap servos invariably have a plastic-on-plastic bearing that do not handle much load. Medium-priced servos generally have metal-on-metal bearings that stand up better under extended use, and expensive servos have ball bearings with best performance.

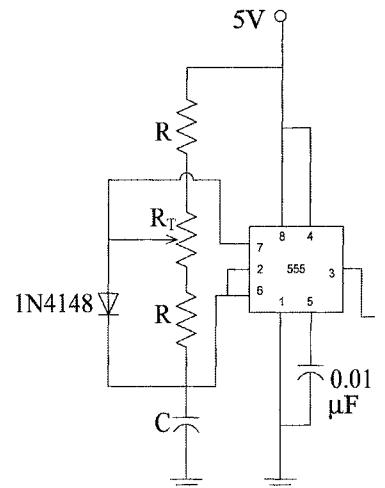


6.2.5.2 Driving Servo Motors

Controlling a servo motor is simple. The motor requires a pulse with a specific frequency, and a change in the pulse duty cycle can be achieved by using a timer circuit. Typical servo motors require a low duty cycle range to operate, between 4.5% and 12%. Always check the specifications of your servo motor before attempting to drive it, since a variation in the frequency can cause permanent damage to the motor.

In the circuit below, the values of R and C will assign the frequency of the output pulse. The variable resistance R_T (a trimmer) adjusts the PWM duty cycle.

A typical servo motor driver requires the PWM to have an up-time range from 0.9 ms to 2.1 ms while the down-time range varies from 19.1 ms to 17.9 ms (for a typical frequency of 50 Hz). For a good operation of the servo motor, the up-time of the pulse has to vary in order to achieve different angles. Depending on the manufacturer, some of the above-mentioned times may vary. For a servo motor with 90 degrees range, an up-time of 1 ms usually represents a 0 degree rotation; an up-time of 1.5 ms will rotate the motor to a 45-degree position; and an up-time of 2 ms will rotate the motor to the 90-degree position. Some servo controllers include two timers (usually 555). The first timer is used to set up the frequency while the second one is used to set up the duty cycle. A servo can also be driven using a microcontroller or any pulse generator that can have a low frequency and duty cycle characteristics.



6.2.6 Some Common Problems in Motors

Problem:

Our motor doesn't have enough power to perform the necessary task in our machine, but we have already bought the motor. What can we do?

Solution:

From a conservation of energy standpoint, if you don't have enough energy out of the system, add more into the system. In other words, increase the size of your batteries. If you add more batteries in series with the ones you already have, you will give the motor a higher top speed. On the other hand, if you need more torque, you can add more batteries in parallel with the existing ones, and your batteries will provide more torque at the same voltage. This is the way to go if your batteries are already supplying the rated voltage to the motor. Another obvious solution is to reduce the load on the motor. The final, and most commonly implemented solution to this problem is to use a gear train to gear your motor down or up. You can gain torque or speed at the expense of the other.

Problem:

Our machine works just fine, but it drains the batteries very quickly. Batteries are getting expensive! What do we do?

Solution:

Batteries are rated for a certain number of *amp-hours*. Alkaline batteries will typically have a larger rating than rechargables, for example. An amp-hour rating of 9 means that the battery can supply 1 amp for 9 hours, or 9 amps for 1 hour, or anywhere in between. If you know the characteristics of your motor, you can calculate the current that your motor is drawing, and from the amp-hour rating of your batteries, you know how long they will last. Because it is more of an issue with rechargables, they are more likely to supply this information when you purchase them. There are two solutions to this problem: reduce the load on the motor so that it draws less current, or get more batteries so that your battery pack can supply the same amount of current longer. Keep in mind that adding a gear train will in general NOT cure this problem, since the amount of power that is consumed by the motor will remain the same. If your machine was already moving forward, adding a gear train will reduce the torque load on the motor, but the motor will have to move faster to provide the same previous speed for the vehicle, and the motor will end up consuming the same amount of power. (Actually, the gear train will have some losses, and you might actually make the problem worse!)

Problem:

We are still in the design phase, and have no good idea how big of a motor we need to buy, or how big the batteries should be. What is a good way to figure this out?

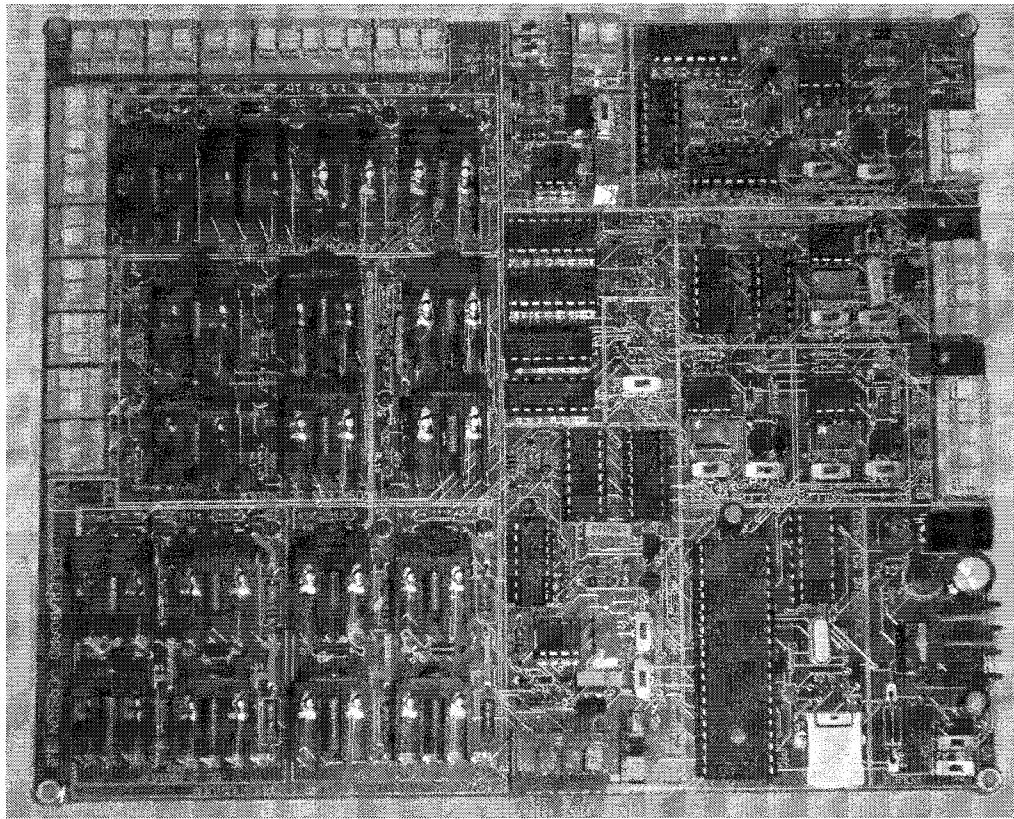
Solution:

The answer to this very common problem is not trivial- that's why it's a common problem. The following steps might help you make a good guess:

- 1) Gather the most complete and specific information you can about the task that the motor will need to perform. For example, get a good guess at the size of the machine, the amount of time it will run, the torque the motor will need to provide. If you can't quantify the applicable variables, then make a conservative 'best guess'.
- 2) Use the information gathered in (1) and physical relations to calculate the power that will be necessary, the torque that will be necessary, the time required, or whatever items are applicable.
- 3) Compare the information calculated in (2) to available motors and batteries. Start by finding the power you need, and then choose a voltage that is commonly available. Then use conservation of energy and the power equations to calculate how much current the motor might draw at this voltage. Finally, multiply this current by the time of operation and find the rating of the battery that might be required.

Always build some conservatism into the calculations. It's much better to have a battery last too long than not long enough!

6.2.7 The AER201 Driver Board



The AER201 Driver Board Version 3 is a Printed Circuit Board (PCB) that is capable of running 2 DC motors, 1 Unipolar stepper motor (UP motor), 1 bipolar stepper motor (BP motor), 1 servo motor (SRV motor) and 1 brushless DC motor (BLDC motor), either manually, by the PIC input or by PC interface. The board is capable of running simultaneously the following motor arrangements:

- 1 Unipolar Stepper, 1 Servo, 1 Brushless and 1 or 2 DC motors
- 1 Unipolar Stepper, 1 Servo, 1 Brushless and 1 Bipolar Stepper

The Driver Board can be powered with a DC voltage between 6V to 24V or by using the USB port. The maximum current that any one motor can draw is 5A, and the supply DC voltage for the motors can be up to 35V.

The board design is laid out in a way that the students can easily determine the type of circuit and relate to it (e.g, the H-bridges are laid out in a typical H configuration, and 555 timing circuits are all laid out in a similar way). The board is also divided into two major sections, the controller section (left half of the board) and the driver section (right half of the board). This configuration isolates the high current, noisy tracks of the motor drivers from the sensitive controller circuits. It also provides a visual separation between driver and controller sections of the board, which allows students to differentiate between the two. All the motor outputs terminals are located on the bottom right corner of the driver board while the power terminals are located on the right side of the board, so as not to become intertwined with the PIC input terminals that are located on the left side of the board. The power input for the controller section of the board is located on the top left corner.

The controller section of the board is divided into 8 modules: **i) Power Module**, **ii) Computer Interface**, **iii) Unipolar Stepper**, **iv) DC 1 Driver**, **v) DC 2 Driver**, **vi) Bipolar Stepper Driver** (consisting of DC1 and DC2 Drivers), **vii) Brushless DC Motor Driver** and **viii) Servo Motor Driver**. The driver section of the board is divided into 7 modules: **a) Unipolar Stepper controller**, **b) DC1 Controller**, **c) DC2 Controller**, **d) Bipolar Controller**, **e) Brushless Controller**, and **f) Mode Selector**.

For more information about operating the board, please refer to its user manual.

6.3 Transmission Systems

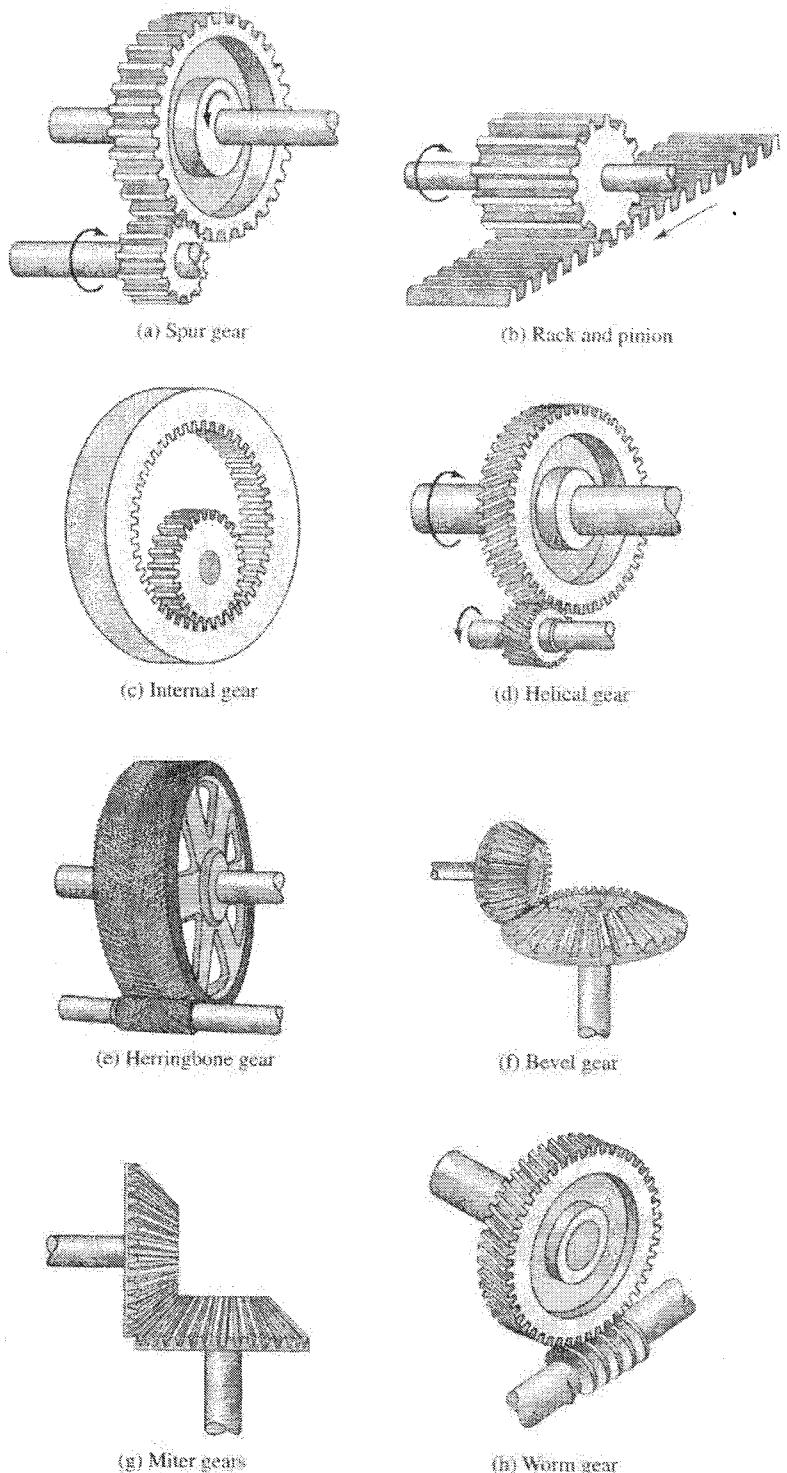
Many actuators are best suited to relatively high speeds and low torque, and therefore require a speed reduction system. Furthermore, actuators tend to be rather heavy. If they can be located remotely from the moving part and toward the base of the system, the overall inertia of the mechanism can be reduced considerably. This, in turn, reduces the size needed for the actuators. To realize these benefits, *transmission systems* are needed to transfer the motion from the actuator to the moving component. However, aside from the added complexity, the major disadvantage of reduction and/or transmission systems is that they introduce additional friction and flexibility into the mechanism. The optimal distribution of reduction stages throughout the transmission will ultimately depend on the flexibility of the transmission, the weight of the reduction system, the friction associated with the reduction system, and the ease of incorporating these components into the overall manipulator design.

6.3.1 Gears

Gears are the most common elements used to transmit torque from one shaft to another. They can provide for large speed reductions (torque amplifications) in relatively compact configurations. Gear pairs come in various configurations for parallel shafts (spur gears), orthogonal intersecting shafts (bevel gears), skew shafts (worm gears or cross helical gears), and other configurations. Figure on the right shows some popular types of gears. Different types of gears have different load rating characteristics, wear characteristics, and frictional properties.

The most basic type of gear is the spur gear, which has its teeth parallel to and in alignment with the centre of the gear. Early manual transmissions used straight-cut spur gears, which were easier to machine but were noisy and difficult to shift. Today these gears are used mainly for slow speeds to avoid excessive noise and vibration. They are commonly used in simple devices such as hand or powered winches.

Helical gears are like spur gears except that their teeth have been twisted at an



angle from the gear centre line. These gears get their name from being cut in a helix, which is a form of curve. This curve is more difficult to machine, but it is used because it reduces gear noise. Engagement of these gears begins at the tooth tip of one gear and rolls down the trailing edge of the teeth. This angular contact tends to cause side thrusts, which a bearing must absorb. However, helical spur gears are more quiet in operation and have greater strength and durability than straight spur gears, simply because the contacting teeth are longer. Helical spur gears are widely used in transmissions today because they are more quiet at high speeds and are durable.

Herringbone gears are actually double helical gears with teeth angles reversed on opposite sides. This causes the thrust produced by one side to be counterbalanced by the thrust produced by the other side. The two sets of teeth are often separated at the centre by a narrow gap for better alignment and to prevent oil from being trapped at the apex. Herringbone gears are best suited for quiet, high-speed, low-thrust applications where heavy loads are applied. Large turbines and generators frequently use herringbone gears because of their durability.

Bevel gears are shaped like a cone with its top cut off. The teeth point inward toward the peak of the cone. These gears permit the power flow to "turn a corner." Hence, the input and output shafts can be of any angle with respect to each other. Spiral bevel gears were developed for use where higher speed and strength are required as well as a change in the angle of the power flow. Their teeth are cut obliquely on the angular faces of the gears. The most commonly used spiral bevel gear set is the ring and pinion gears used in heavy truck differentials. Bevel-type gears are also used for slow-speed applications that are not subject to high impact forces. Hand-wheel controls that must operate some remote device at an angle use straight bevel gears. A special type of bevel gears, with a gear ratio of 1, is called Miter gears, which are used for turning the power flow 90 degrees.

The worm gear is actually a screw that is capable of high-speed reductions in a compact space. Its mating gear has teeth that are curved at the tips to permit a greater contact area. Power is supplied to the worm gear, which drives the mating gear. Worm gears usually provide right-angle power flows. The most common use for the worm gear is in applications where the power source operates at high speed and the output is at slow speed with high torque. Many steering mechanisms use a worm gear connected to the steering shaft and wheel and a partial (sector) gear connected to the steering linkage. Small power hand tools frequently use a high-speed motor with a worm gear drive.

Rack and pinion gears convert straight-line motion into rotary motion, and vice versa. Rack and pinion gears also change the angle of power flow with some degree of speed change. The teeth on the rack are cut straight across the shaft, while those on the pinion are cut like a spur gear. These gear sets can provide control of arbor presses and other devices where slow speed is involved. Rack and pinion gears also are commonly used in automotive steering boxes.

Internal gears have their teeth pointing inward and are commonly used in the planetary gear set used in automatic transmissions and transfer cases. These are gear sets in which an outer ring gear has internal teeth that mate with teeth on smaller planetary gears. These gears, in turn, mesh with a centre or sun gear. Many changes in speed and torque are possible, depending on which parts are held stationary and which are driven. In a planetary gear set, one gear is normally the input, another is prevented from moving, or held, and the third gear is the output gear. Planetary gears are widely used because each set is capable of more than one speed change. The gear load is spread over several gears, reducing stress and wear on any one gear.

The major disadvantages of using gearing are *backlash* and *friction*. Backlash, which arises from imperfectly meshed gears, can be defined as the maximum angular motion of the output gear when the input gear remains fixed. If the gear teeth are meshed tightly to eliminate backlash, there can be excessive amounts of friction. Very precise gears and very precise mounting minimize these problems but also increase cost.

The gear ratio, η , describes the speed reducing and torque increasing effects of a gear pair. For torque magnification systems, we will define $\eta > 1$ so that the relationships between input and output speeds and torque are given by

$$\omega_o = \frac{1}{\eta} \omega_i ;$$

$$T_o = \eta T_i ,$$

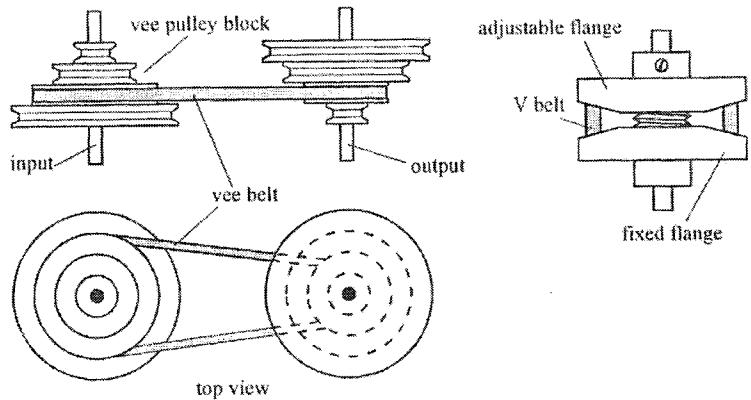
where ω_o and ω_i are output and input speeds respectively, and T_o and T_i are output and input torque.

6.3.2 Flexible Mechanisms

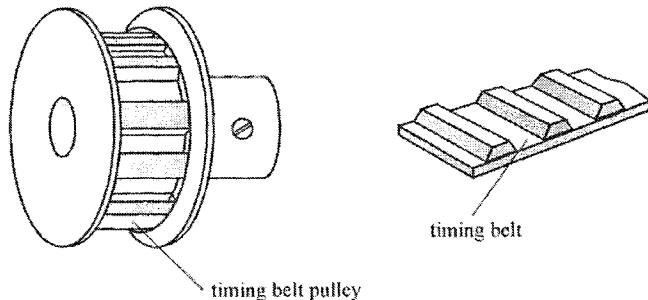
The second broad class of reduction elements includes flexible bands, cables, and belts. Because all of these elements must be flexible enough to bend around pulleys, they also tend to be flexible in the longitudinal direction. The flexibility of these elements is proportional to their length. Because these systems are flexible, there must be some mechanism for preloading the loop to ensure that the belt or cable stays engaged on the pulley. Large preloads can add undue strain to the flexible element and introduce excessive friction.

Cables or flexible bands can be used either in a closed loop or as single-ended elements that are always kept in tension by some sort of preload. In a joint that is spring loaded in one direction, a single-ended cable could be used to pull against it. Alternately, two active single-ended systems can oppose each other. This arrangement eliminates the problem of excessive preloads but adds more actuators.

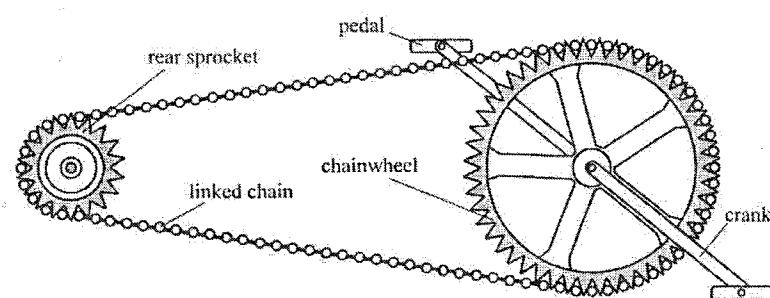
Stepped pulleys, shown in figure on the right (left) are still used in most drill presses for changing chuck speed. They allow a single belt to transfer power at either of several pulley combinations. In figure on the right (right), a variable pitch pulley is also shown that can adjust the belt speed by about 25%. A threaded hub with locking set screw is used for fine speed control, or to adjust belt tension, but this can only be done when the belt is stationary. Spring loaded variable pitch pulley wheels can be used to automatically regulate belt tension.



Slippage on V-belt systems can be reduced by applying belt dressing, but for critical applications positive drive from a timing belt or a chain is superior. Figure below shows a timing belt pulley and a section of belt capable of providing a no-slip drive that eliminates speed variations due to slippage. Timing belts do not rely on friction, hence they are more efficient, run cooler than V-belts, and can be used at surface speeds of ~ 50 m/s. Timing belts may be used whenever synchronous operation and high performance is required. They are lightweight, and need no lubrication.



Rope, string, leather straps or belts, nylon bands, o-rings, and coiled steel springs can all be used on pulley systems, but chain is one of the best methods for power transfer. Figure below shows a simple chain drive. Steel chain drives are used for bicycles, motorcycles, garage door openers, and even for vectored thrust on a Harrier jump-jet aircraft. Roller chains are similar to flexible bands but can bend around relatively small pulleys while retaining a high stiffness. As a result of



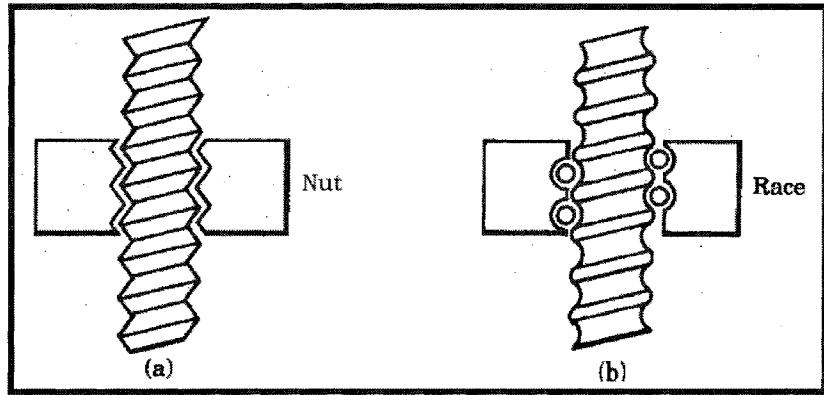
wear and high loads on the pins connecting the links, toothed belt systems may be more compact than roller chains for certain applications.

Band, cable, belt, and chain drives have the ability to combine transmission with reduction. If the input pulley has radius r_i and the output pulley has radius r_o (usually greater than r_i), the “gear” ratio of the transmission system is

$$\eta = \frac{r_o}{r_i}$$

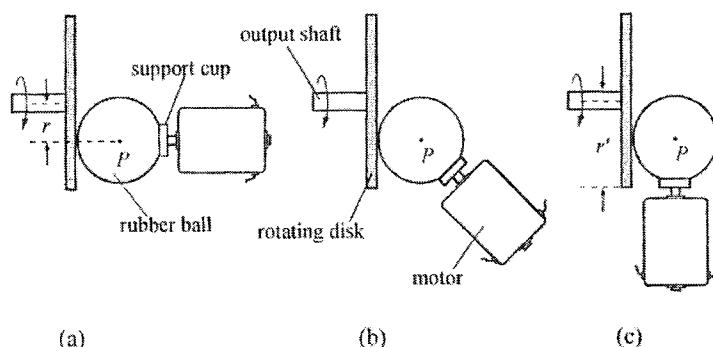
6.3.3 Lead Screws

Lead screws or ball bearing screws, as shown in Figure on the right, (a) and (b), respectively, provide another popular method of getting a large reduction in a compact package. Lead screws are very stiff and can support very large loads, and have the property that they transform rotary motion into linear motion. Ball bearing screws are similar to lead screws, but instead of having the nut threads riding directly on the screw threads, a recirculating circuit of ball bearings rolls between the sets of threads. Ball bearing screws have very low friction and are usually back drivable.

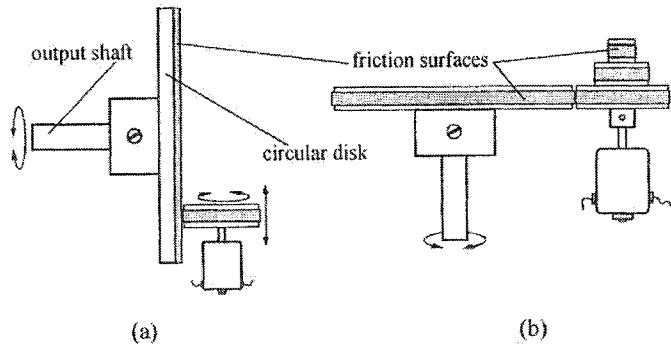


6.3.4 Continuously Variable Transmission

A chain drive can be made to vary its output continuously if it is coupled to a slipping clutch. However this is complicated and inefficient, and some simple alternatives are described below. A rubber ball attached to a motor's shaft in Figure below can be rotated about its center to vary the speed of a rotating disk, while power is extracted from its rotating shaft. In (a), minimum (~zero) output speed occurs when the ball/motor axis is normal to the disk. In (b), disk speed increases as the ball is revolved around its center p . And, in (c), maximum disk speed is attained when the ball has rotated through 90° , when full ball speed is transferred to the disk. Output shaft speed can be increased further by moving point p to a larger radius r' .



In figure below (a), a motor driven friction wheel is translated across the surface of a circular disk to vary the output shaft speed. Alternatively, a circular disk may be driven at its rim by a friction wheel as indicated in the figure (b); additional speeds are provided by the extra drive wheels. Wooden input and output disks are satisfactory for this purpose, and may

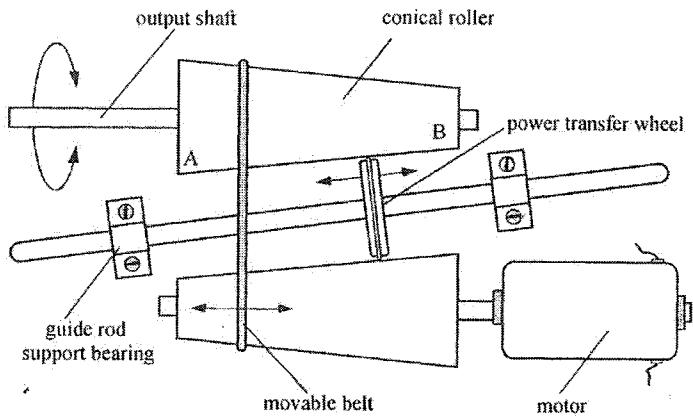


(a)

(b)

be cut with a band saw or jigsaw, then finished with a file (preferably while a disk is spinning). A thin layer of well-cured silicone rubber makes a good high friction coating for the wheel rims of figure below.

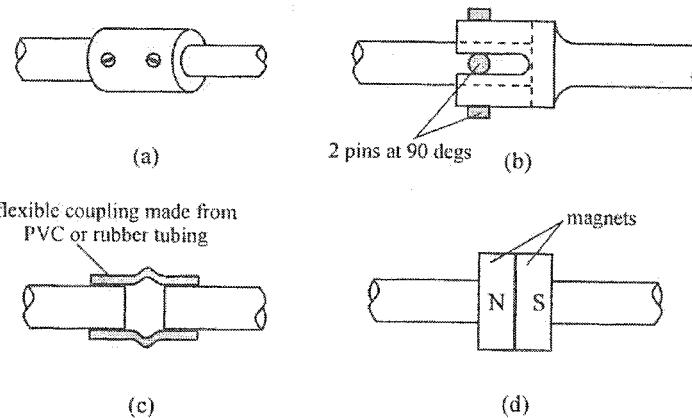
Auto manufacturers have used a cone drive assembly, shown in figure below, to make a shiftless automatic transmission. In this arrangement two cones perform just like two stepped pulley blocks, but with each block having an infinite number of pulleys with decreasing diameters. Output shaft speed is determined by the location of a friction belt which transfers power from the motor to output shaft. For automobiles a ratio of about 4: 1 (from high to low gear) is adequate, and cone angles shown in the figure also provide this ratio. A friction wheel sliding along a guide rod may also be used to transfer power instead of a movable belt, and either a belt or wheel is easily moved when the cones are rotating.



6.3.5 Shaft Coupling

Many mechanical devices are coupled to motors by either rigid or flexible shaft couplings. Devices for coupling rotating shafts are needed most frequently, and a few examples are shown in Figure below. Although a rigid coupling, shown in figure below (a), transmits torque from one shaft to another in a very simple and efficient way, it can be only used when both shafts are on the same axis, because shaft misalignment promotes wear in those bearings where devices are connected together. Mating shafts can be accurately aligned by checking for rotational eccentricities with a dial gage moved over a surface plate or a sheet of "float" glass. In general, amateurs should use flexible couplings, unless a dial gage is available for checking misalignment eccentricities on a rigidly coupled system.

The coupling (b) in figure below gives a bit more flexibility. Further, Longitudinal shifts are accommodated, but only small angular movements are permitted with this type of coupler. The flexible coupling, show in figure below (c), handles both angular and parallel displacement shaft misalignments, but at the expense of a lower torque limit set by the flexible element. Flexibility also causes some degree of uncertainty in shaft rotation, and twist in a coupling generates backlash when a motor's shaft rotation is reversed. Magnetic coupling in the figure (d) is simple, effective and easily made by



cementing magnets to both shaft ends, either rigidly with epoxy, or with silicone rubber for a flexible mount. Attracting magnetic faces must contact for optimum torque transfer, and this is best done by cementing parts when all elements are in their operating positions. Magnetic coupling can also be employed for transferring torque across a solid interface, and is often used when motive power must be coupled to the interior of an ultra high vacuum system.

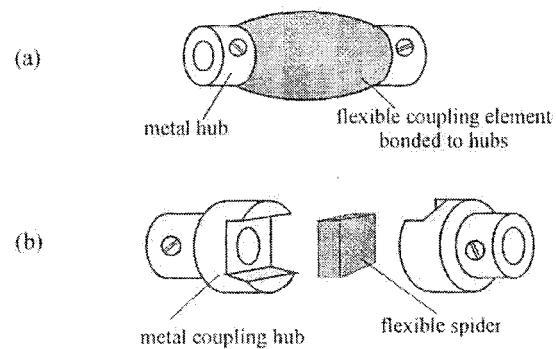
NOTE: When making a rigid coupling to hold shafts of different diameters, drill the smallest hole right through the axis, then, *without moving the drill or workpiece*, drill a hole for the large shaft to the required depth. This procedure ensures axial alignment for both holes.

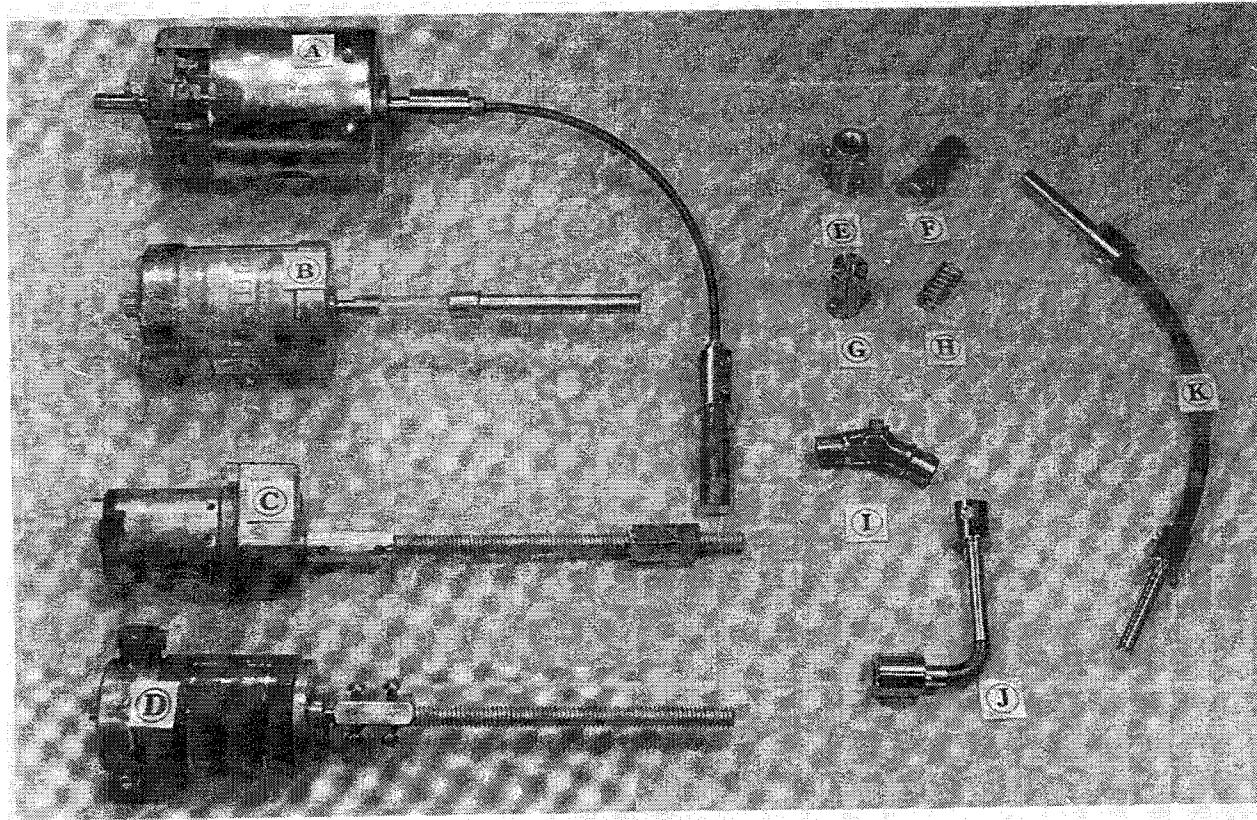
Fractional horsepower couplers consisting of metal end caps separated by a flexible insert provide good vibration isolation and tolerate shaft misalignment very well. Two designs that are used extensively for industrial applications are shown in figure on the right. They are consequently sometimes sold by surplus outlets at reasonable prices. Figure (a) shows a Buna N, or neoprene flexible element, which has a long operating life and tolerates up to seven degrees of angular misalignment. Another type is shown in figure (b). Cross or star rubberized insert modules are also sold.

When attaching couplers and pulleys to motor shafts, one must note that hardened set screws can easily disfigure a motor shaft making it impossible to remove a connector, pulley or other item attached to the shaft. To avoid this problem always file a small flat on a motor's shaft, making sure the set screw contacts only on this flattened area. Not only will this prevent shaft seizure, it gives a more secure bite and requires less screw force to prevent slippage.

A larger variety of different shaft couplings is shown in figure below. A flexible drive shaft in (A) is composed of two overlapping oppositely wound outer springs, on an inner shaft of longitudinal wires. Shafts of this type absorb vibration and can snake around obstacles, providing a simple drive connection normally requiring universal joints or gears. Potentiometers are sometimes remotely operated with these flexible shafts and shafts are sold by good electronics stores. Stiff springs, shown in H of figure below, soldered to brass coupling collars make inexpensive flexible drives. To avoid misalignment, attach both collars to a common shaft before soldering. It is tempting to use epoxy instead of solder for this task. However, adhesives only withstand modest torques, and will sometimes break under a shock load. Commercial flexible shafting is worth the extra cost in most cases. Drive performance from a spring wound shaft is asymmetric, and maximum torque or speeds are obtained when the outer spring is driven so it winds up. Object (K) in figure below illustrates a W flexible spring shaft, which is sold in lengths up to 6ft and operates down to a minimum turn radius of 6", up to 5000rpm at a maximum torque of 10lb-in. Flexible shafts of this type may be shortened after they have been well soldered at the cutting location.

Rubber or flexible vinyl tubing couplers are popular amongst students. They are inexpensive, easy to install, and can be used to connect two different size shafts, as shown on motor B in figure below. Free tubing length between shafts should be kept short to reduce backlash. Adhesive and clothesline wire tourniquets may be added if slippage occurs. Moulded





neoprene is used to join two metal collars in some commercial flexible couplings, and three-piece spider couplings with a rubber star insert are made to fit shafts from $\frac{1}{4}$ " to $4\frac{1}{2}$ " diameter. Item J in figure below shows a thin wall (0.001") stainless steel bellows often used for coupling motor shafts in ultra high vacuum systems, or where high temperatures are anticipated. For those applications where backlash cannot be tolerated and drive misalignment is <10 degrees, stainless steel bellows are usually the first choice. These devices are made from very thin material and are acceptable for use in ultra high vacuum conditions, and high temperatures. A small bellows fabricated from 0.001" stainless is shown in (J). Bellows couplers are used with resolvers, encoders, servo links and anywhere angular motion must be transferred with high fidelity. Nevertheless, stainless steel bellows are not cheap.

As mentioned before, all elements in a drive train must be concentric when rigid shaft connections are used, and such couplings must be turned on a lathe to achieve the required precision. Rigid couplings are shown in (F) of the above figure and it is installed on geared motor C. This latter coupling has been tapped to accept a $\frac{1}{4}$ -20 threaded rod, and is equipped with a $\frac{1}{4}$ -20 coupling nut rider. If screw adjusters are added to a standard coupling nut as shown on motor D, a plain or threaded rod can be attached to the motor's shaft so it runs concentrically, (4-40 brass screws work well on motor shaft couplings made from $\frac{1}{4}$ -20 coupling nuts). Brass adjusting screws are best because these do not mar the motor's shaft. If steel screws are used, their ends should be filed flat to reduce shaft damage. Always file a small flat on a shaft, and make sure one of the three adjusting screws is aligned with this surface. Coupling nuts with screw adjusters can be drilled out to accept larger motor shafts, or fitted with a brass insert to accept smaller size threaded rod. A modified coupling nut with an 8-32 insert is shown in G of the above figure. When using a coupling nut with screw adjusters, alignment for true running is made by trial and error. With a little practice shafts can be aligned with negligible wobble in a few minutes.

Zinc plated steel coupling nuts are made in 4-40, 6-32, 8-32, 10-24, $\frac{1}{4}$ -20, and larger sizes, and these can be obtained from comprehensive distributors. Hardware stores normally stock $\frac{1}{4}$ -20 (shown in E), and larger sizes. Threaded rod may be attached to coupling nuts, and then driven directly from a motor, (C in figure above). For most student projects $\frac{1}{4}$ -20 rod is preferred, because this is inexpensive and readily available. If finer thread is required, a brass screw can be soldered into a larger coupling nut, then threaded with an appropriate tap, as in G of figure above. Rigid couplers shown in C and F of the above figure may be made on a drill press from brass or aluminum rod.

Off-axis drive up to 30 degrees is possible with a universal joint. High quality double or telescoping universal joints are expensive (>\$30), but economy, or plastic ball and socket joints are available from about \$5. A pin type universal joint with brass collars is shown in (I) of the above figure.

Sometimes it is necessary to increase a motor's shaft diameter to match a particular fitting. If a piece of brass hobby tubing can be found to match the fitting, it may be attached to the motor's shaft as follows:

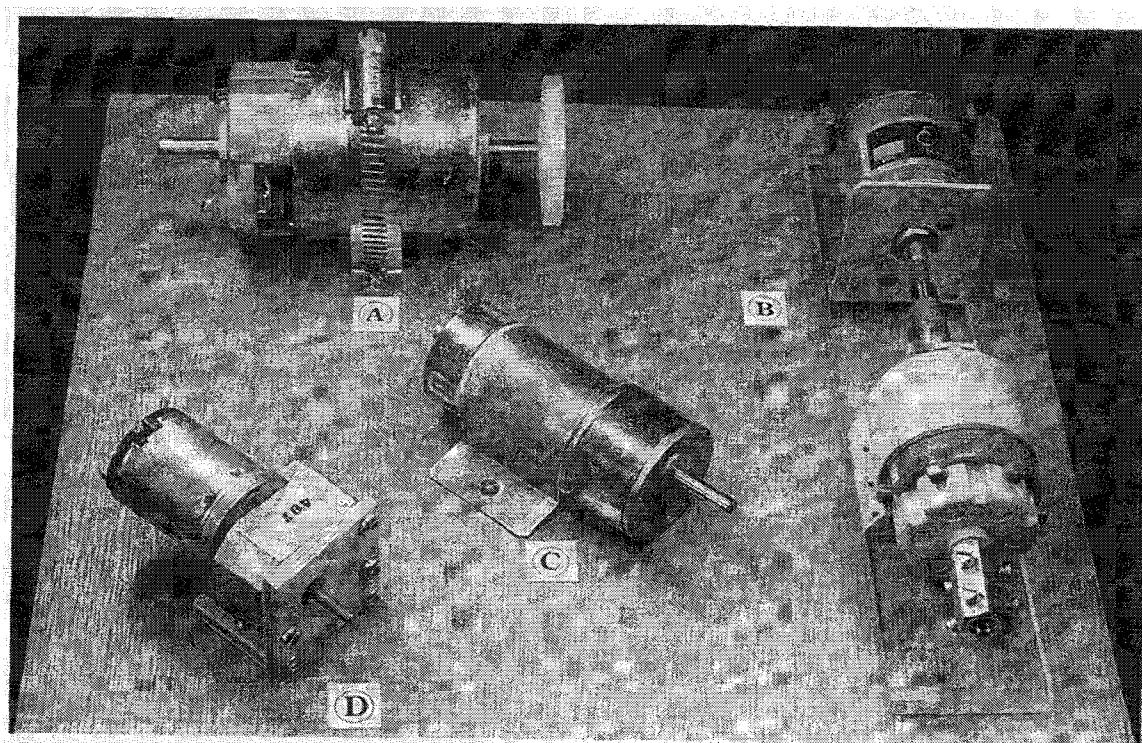
1. Trim a length of masking tape so its width matches the motor shaft length.
2. Wind this tape onto the motor shaft until it is a little larger than the tube's inside diameter.
3. Now repeatedly trim *short* lengths from the rolled tape until it just fits the tube.

If this is done carefully, the tubing will be concentric with the motor shaft. When glued, this assembly handles reasonably heavy drive loads. Keep the total tape thickness to <0.040" for best performance, or use additional hobby tubing sizes, one inside the other, to make a better initial fit.

6.3.6 Mounting Small Electric Motors

Reaction torques developed when a motor drives an external element tend to rotate a motor, and for this reason motors should be firmly attached to a suitable base. Ideally a mount should be simple, effective, and allow for easy installation of a new motor. Some mounting methods are shown in figure below, and these can be made with basic hand tools. Worm-drive adjustable stainless steel hose clamps make versatile motor holders, as shown in (A). Clamp sizes start at 7/32"-5/8", increasing by small increments to 5 5/8"-8 1/2". Clamps can be cut with aviation snips or a hacksaw. This DC brush motor is ungeared, and its dual shafts make it easier to install an encoder disk.

Item (B) in figure below shows a 12V DC brush motor attached to a gearbox via a flexible vinyl coupling. Item (C) shows a Pitman DC gearhead motor, which is clamped with a finger-bent galvanized steel band. Small gearhead motors are in great demand for design projects, and the style in (D) of figure below has been carried by various distributors.



6.4 Stiffness and Deflection

An important goal for design of most electromechanical is the overall stiffness of the structure and the drive system. Flexibilities in the structure or drive train will lead to **resonances** that have an undesirable effect on the system performance. In this section, some issues of stiffness and the resulting deflections under loads are addressed.

6.4.1 Flexible Elements in Parallel and Series

The combination of two flexible members of stiffness k_1 and k_2 when “connected in parallel” produces the net stiffness

$$k_{parallel} = k_1 + k_2,$$

and when “connected in series” produces the net stiffness

$$\frac{1}{k_{series}} = \frac{1}{k_1} + \frac{1}{k_2}.$$

In considering transmission systems, we often have the case of one stage of reduction or transmission in series with a following stage of reduction or transmission.

6.4.2 Shafts

A common method for transmitting rotary motion is through shafts. The torsional stiffness of a round shaft can be calculated as:

$$k = \frac{G\pi d^4}{32l}$$

where d is the shaft diameter, l is the shaft length, and G is the shear modulus of elasticity (about $7.5 \times 10^{10} \text{ N/m}^2$ for steel, and about a third as much for aluminum).

6.4.3 Gears

Gears, although typically quite stiff, introduce *compliance* into the drive system. An approximate formula to estimate the stiffness of the output gear (assuming the input gear is fixed) is given as:

$$k = C_g br^2$$

where b is the face width of the gears, r is the radius of the output gear, and $C_g = 1.34 \times 10^{10} \text{ N/m}^2$ for steel.

Gearing also has the effect of changing the effective stiffness of the drive system by a factor of η^2 . If the stiffness of the transmission system prior to the reduction (i.e., on the input side) is k_i , that is,

$$\tau_i = k_i (\delta \theta_i),$$

and the stiffness of the output side of the reduction is k_o , that is,

$$\tau_o = k_o (\delta \theta_o),$$

then we can compute the relationship between k_i and k_o , assuming a perfectly rigid gear pair as:

$$k_o = \frac{\tau_o}{\delta \theta_o} = \frac{\eta k_i (\delta \theta_i)}{\left(\frac{1}{\eta} \delta \theta_i \right)} = \eta^2 k_i.$$

Hence, a gear magnification has the effect of increasing the stiffness by the square of the gear ratio.

6.4.4 Belts

In a belt drive, stiffness is given by

$$k = \frac{AE}{l},$$

where A is the cross sectional area of the belt, E is the modulus of elasticity of the belt, and l is the length of the free belt between pulleys plus one third of the length of the belt in contact with the pulleys.

6.4.5 Links

As a rough approximation of the stiffness of a link, we might model a single link as a cantilever beam, and calculate the stiffness at the end point. For a round hollow beam, this stiffness is given by

$$k = \frac{3\pi E(d_o^4 - d_i^4)}{64l^3},$$

where d_i and d_o are the inner and outer diameters of the tubular beam, l is the length, and E is the modulus of elasticity (about $2 \times 10^{11} \text{ N/m}^2$ for steel, and about a third as much for aluminum). For a square cross section hollow beam, this stiffness is given by

$$k = \frac{E(w_o^4 - w_i^4)}{4l^3},$$

where w_i and w_o are the outer and inner widths of the beam (i.e., the wall thickness is $w_o - w_i$).

In this section we have examined some simple formulas for estimating the stiffness of gears, shafts, belts, and links. They are meant to give some guidance in sizing structural members and transmission elements. However, in practical applications, many sources of flexibility are very difficult to model. Often, the drive train introduces significantly more flexibility than the links. Furthermore, many sources of flexibility in the drive system have not been considered here (bearing flexibility, flexibility of the actuator mounting, etc). Generally, any attempt to analytically predict stiffness results in an overly stiff prediction because many sources are not considered. Finite element techniques can be used to predict more accurately the stiffness (as well as other properties) of more realistic structural elements.

6.5 Drive Mechanisms

An important aspect of a mobile robot's physical design is its drive system. It is responsible for moving the robot from place to place providing the appropriate motive force and steering mechanisms. Three most popular drive arrangements are shown in figure below, and they are explained in the following sub-sections.

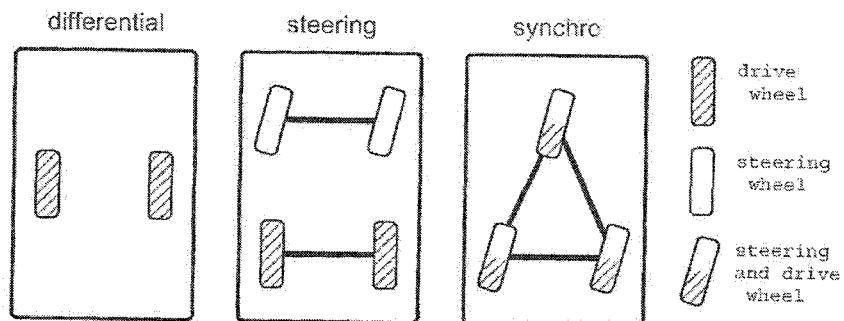
6.5.1 Differential Drive

A differential drive consists of two independently driven wheels arranged side by side. When both wheels are driven at the same rate in the same direction, the robot will move straight. When the wheels are driven at the same rate in opposite directions, the robot will spin in place. By varying the relative speeds of the two wheels, any turning radius can be achieved. Since this system minimizes the number of moving parts, it tends to be the simplest and most robust. The complexity is increased slightly, though, by the need for a caster in the front or back of the robot to keep it from tipping over. An especially useful property of this drive system is that the change in orientation of the robot depends only on the difference between the distances traveled by each of the wheels. It does not matter if the left wheel moves forward 10 cm and the right back 10 cm, or if only the left wheel moves forward 20 cm; the final location will be different, but the orientation will be the same (ignoring slippage). This greatly simplifies the process of turning, by making the final orientation of the robot easily predictable.

The differential drive is the most popular because of its simplicity, though it does have some limitations. Since it is difficult to calculate the final location of the robot if it turns and moves at the same time, most navigation algorithms consist of driving straight for a distance, turning through a specified angle in place, and then driving straight again. More sophisticated algorithms allow the robot to turn while moving, but typically, such turns are still constrained to easily calculated curves.

6.5.2 Steering System

Steering systems should already be familiar to you because they are widely used in automobiles. They usually consist of one or two steerable wheels at the front of the robot and two powered wheels at the rear. The turning radius is determined by the angle of the steerable wheels, but the robot must be moving in order to make a turn. This means that it cannot turn in place and can only make turns of limited sharpness while driving. The advantage of using a steering system comes from the separation of the steering and drive mechanisms. Such robots tend to be quick and fairly agile. They are well-suited to driving in open spaces or performing "follow" tasks since these tasks usually require making course corrections to the left and right as the robot drives. However, when a steering robot turns, the two rear wheels will take paths of different lengths. The outer one will travel a longer distance than the inner one, so it is helpful to use a differential in the gearbox to transfer force between the wheels in order to avoid slippage.



6.5.3 Synchro Drive

The synchro drive is an exotic mechanism where all the wheels are driven and steered together. Usually, there is one gearbox which turns the wheels to the desired orientation and then another which drives the robot in that direction. This may seem strange, but it allows the robot's instructions to be phrased in terms of world coordinates, instead of having to compute everything in terms of the robot's perspective. The robot can also be commanded to move in any direction, making this the most mobile of the drive systems. This mechanism simplifies control at the expense of complexity in construction. All the wheels must be both steerable and drivable, so building drive trains for such a system often requires an elaborate system of gears and chain drives. Also, since only the wheels turn, the robot's body remains in a fixed orientation, unless it is also turned. This makes it inconvenient for such a robot to have a "front" as many applications require.

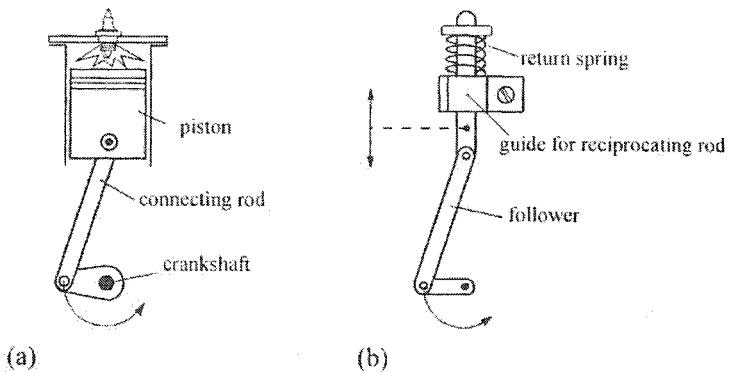
6.5.4 Legs

Robots with articulate legs can do everything that a wheeled robot can do and more. This includes walking in arbitrary directions, turning in place, and even climbing over otherwise inaccessible terrain. Unfortunately, though, legged robots are comparably difficult to build and comparably difficult to control. In fact, a great deal of research is currently being performed to study ways of making robots walk. If you think you are up to the challenge, a legged robot makes a very exciting project, but be warned that building legs can be much more difficult than it appears.

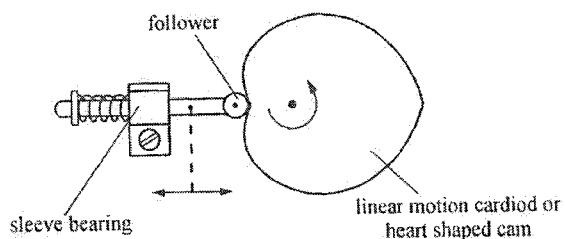
6.6 Useful Mechanisms

The following mechanisms are useful for the Engineering Design projects. (Source: Davis B., "Electric Motors and Mechanical Devices for Hobbyists and Engineers", WERD Technology Inc., Canada, 1997). For more detailed and advanced mechanisms the following reference can be very useful: Sclater N., Chironis N.P., "Mechanisms and Mechanical Devices Sourcebook", Third Edition, McGraw-Hill, 2001, Call No. TJ 181 .C399 2001 ENGI

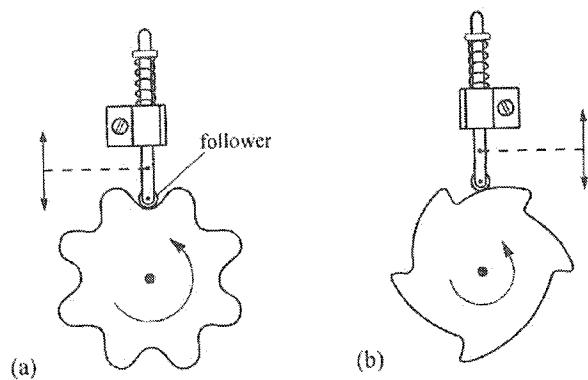
6.6.1 Rotary-Longitudinal Motion Converters



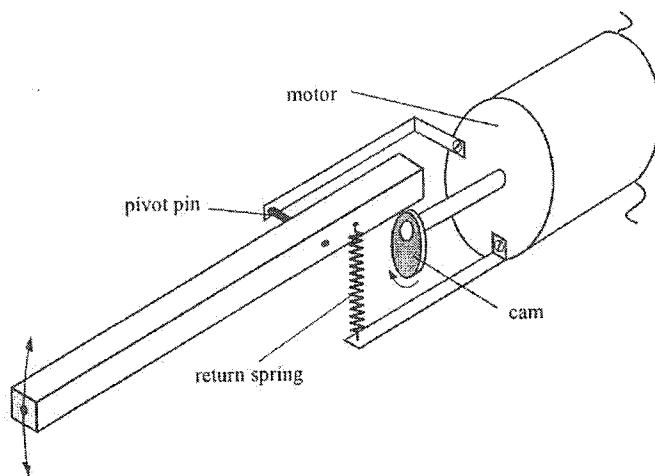
[a] Reciprocating piston movement is transformed into rotary camshaft motion by means of connecting rods and pivot pins. [b] A rotating link produces reciprocating motion in the follower by means of a pivoting arm. Low friction motion conversion is provided by both these devices, and these principles may be used to construct reliable movements for amateur applications.



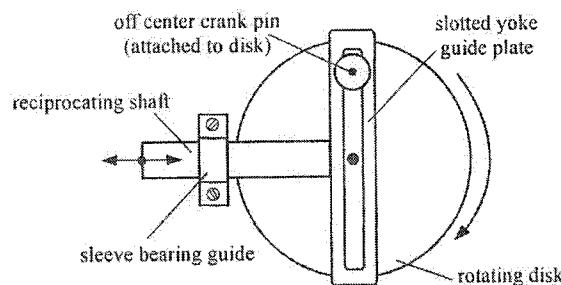
This cardioid shaped cam makes a follower move with a longitudinal stroke motion, producing a displacement that is linearly proportional to the cam rotation angle.



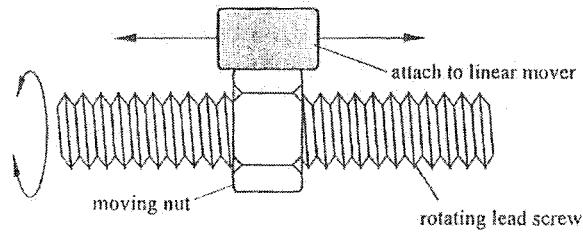
[a] Periodic cams are often used for event timing and may be driven by a motor or regulated clockwork spiral spring mechanism.
 [b] Profiles of irregular or aperiodic cams are usually smoothed where sharp profile changes occur, and this helps diminish follower bounce and cam wear.



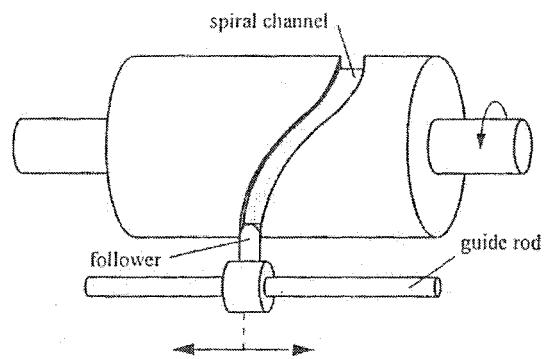
Cam force may be amplified or reduced by selecting a suitable arm ratio for this cam driven rocking beam.



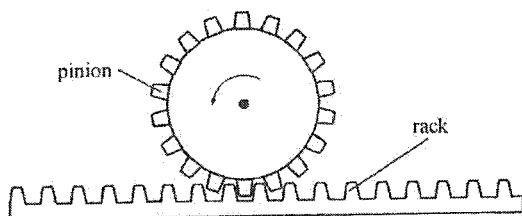
Scotch yokes are useful mechanisms for generating reciprocating motions from rotary action. These assemblies are easily made by amateurs and suitable guides may be fabricated from heavy music wire rods running inside brass hobby tubing. Guides are essential to prevent the follower from stalling or jamming during disk rotation.



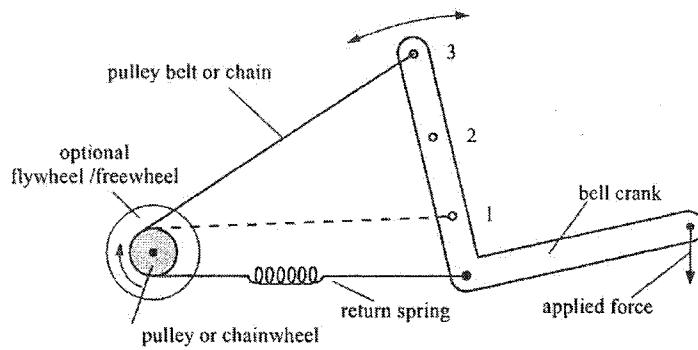
One of the most widely used mechanisms for creating powerful linear forces, this simple screw-nut combination is widely used in amateur robotics for good reasons. It is efficient, inexpensive, easy to attach to a motor shaft, develops high power push and pull forces, and motions are readily scaled by selecting a suitable screw pitch.



Large pitch spiral grooves on a rotating cylinder can move a follower precisely and rapidly. This type of kinematic movement is more difficult for hobbyists to build at home, but some units are available from surplus commercial instruments, such as older photocopiers.

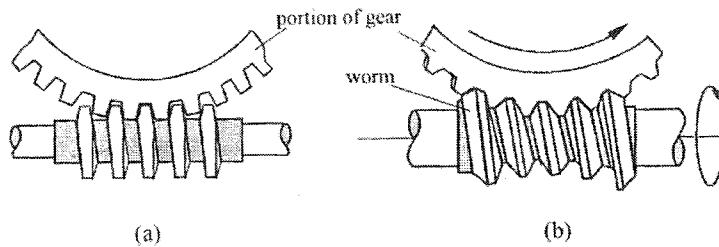


Rack and pinions may be employed for horizontal, inclined, or vertical movements. Higher torque and improved stability can be achieved by using double racks, i.e., with a second rack mounted above the pinion gear in this figure. *McMaster-Carr* sells spur gears and matching tracks in steel and nylon up to 6 feet long, 5 to 48 teeth/inch - some other sources are detailed in the text.

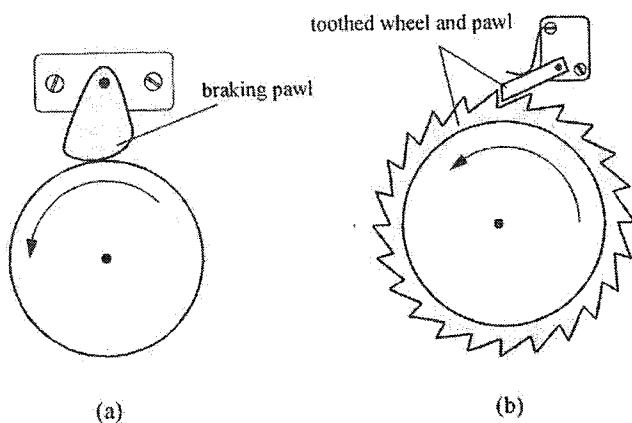


This lever-operated drive system has been used for bicycle propulsion. A pedal is normally attached to the right-hand side of the bell crank. Although a useful mechanism for converting reciprocating motion to rotary action, it has not found a commercial niche for reasons discussed in the text.

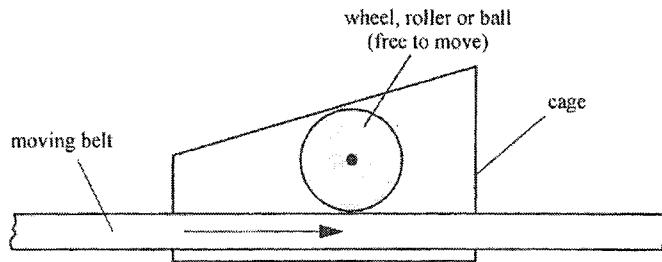
6.6.2 One-way Motion Controls



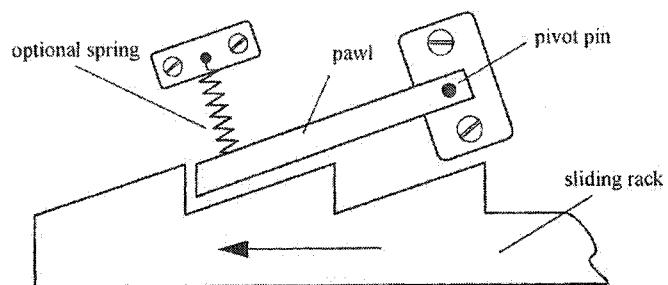
A worm and gear protects a motor from being driven backwards by a load. Because a gear cannot turn the worm drive, no appreciable torque can be transferred from load to motor, when the latter is stationary. [a] A straight worm is easier to manufacture but has a lower torque rating than a form fitting worm in [b], where better tooth contact is established.



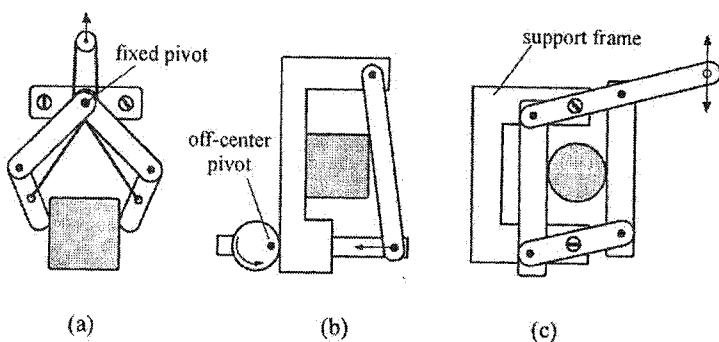
[a] Wheel rotation is unimpeded in the arrow's direction, but is repressed by the braking pawl when rotation is reversed. [b] Pawl and ratchets are commonly used safety mechanisms, and widely used for generating one-way motions in mechanical clocks.



Friction causes the ball to move left when the belt's direction is reversed. All motion is stopped once the ball is wedged between the cage ceiling and belt surface.



Gravity operated pawls are often used with horizontal slides, permitting operation without spring assistance.



[a] Cable-operated fingers can grasp the exterior of an object as shown here, or may be inverted for picking up a cylinder by its interior surface. [b] Cam action generated by an eccentrically pivoted circular roller develops good holding power. [c] Both upper and lower parallel members are secured to a stationary frame, but are free to swivel when the upper lever arm is moved to alter the clamping gap.

6.7 Measuring Moment of Inertia

In the design projects, one may occasionally need to have a good estimate of moments of inertia of various parts to calculate the amount of the required torque, for instance. Most common experimental apparatus for determining moment of inertia involve determining the natural frequency and corresponding period of oscillation for the experimental setup and test specimen. A number of various setups have been implemented successfully, but some are less complex and easier to assemble.

6.7.1 Method 1

The moment of inertia of an object can be determined by suspending the object as a pendulum so that rotational oscillations can occur about a pre-determined axis. The period of the free oscillations can be measured, and used with the geometry of the pendulum to determine the moments of inertia. Three types of pendulums are useful for this experiment: compound, torsion, and trifilar. The compound pendulum is shown in figure below. When using the compound pendulum the body is supported from two overhead points by wires. The distance l is measured between the axis of support O-O and a parallel axis C-C through the centre of gravity of the body. The moment of inertia about C-C is given by:

$$I_{cc} = ml^2 \left[\left(\frac{\tau_0}{2\pi} \right)^2 \left(\frac{g}{l} \right) - 1.0 \right]$$

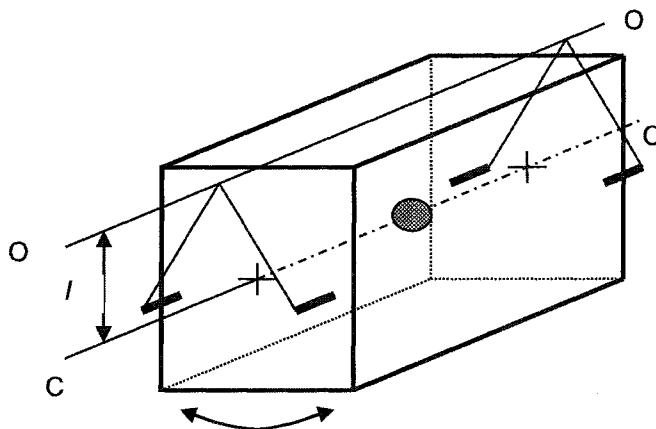
where:

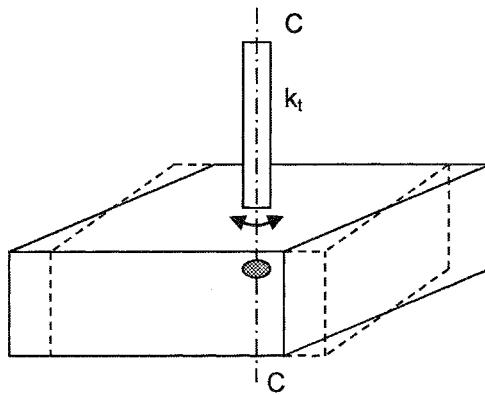
- τ_0 = period of oscillation [s]
- l = distance between axis of support and axis through the centre of gravity [in]
- g = gravitational acceleration [in/s²]
- m = mass [lb*in*s²]

A direct measurement of l may not be possible because the centre of gravity is often an inaccessible point. However, a change in l is easily measured. If the experiment is repeated with a different support axis O'-O', the length l becomes $l+\Delta l$ and the period of oscillation becomes τ_0' . The distance l can then be written in terms of Δl , and the two periods τ_0, τ_0' :

$$l = \Delta l \left[\frac{\left(\tau_0'^2 / 4\pi^2 \right) (g/\Delta l) - 1}{\left[\left(\tau_0^2 - \tau_0'^2 \right) / 4\pi^2 \right] [g/\Delta l] - 1} \right]$$

This value of l can be substituted into the first equation. The length of l should not be greater than the radius of gyration of the body about the C-C axis. For large values of l the equation of moment of inertia becomes sensitive to small changes in l and τ_0 .





6.7.2 Method 2

The second pendulum method of determining the moment of inertia involves a torsional pendulum as shown in figure above. The test specimen is attached to the end of a torsion spring. The moment of inertia about the desired axis can be determined if the stiffness of the spring k_t is known and the period of torsional oscillation τ is measured.

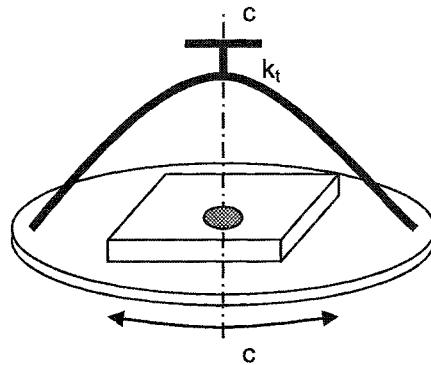
$$I_{cc} = \frac{k_t \tau^2}{4\pi^2} = \frac{k_t}{\omega_n^2}$$

where ω_n is the natural frequency.

A simpler method of determining the moments of inertia for a number of objects, or if the spring coefficient is unknown is to build a platform on the end of a torsion spring as shown in figure below. By repeating the experiment with different bodies placed on the platform, it becomes unnecessary to measure the torsional stiffness k_t . If a body with a known moment of inertia I_1 is placed on the platform and an oscillation period τ_1 results, the moment of inertia I_2 of a body which produces a period τ_2 is given by:

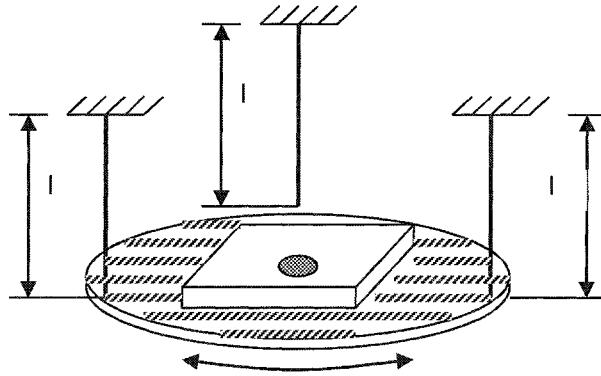
$$I_2 = I_1 \left[\frac{(\tau_2/\tau_0)^2 - 1.0}{(\tau_1/\tau_0)^2 - 1.0} \right]$$

τ_0 is the period of the pendulum and empty platform.



6.7.3 Method 3

The third method is similar to the torsion arrangement and called a trifilar pendulum and consists of a body suspended by three flexible wires as shown in figure below. To determine the moment of inertia, the distances of the wires to the



vertical axis C-C through the centre of gravity of the body are required. The angles between wires and the length of each wire are also required.

$$I_{cc} = \frac{mgR_1R_2R_3\tau^2}{4\pi^2l} \frac{R_1 \sin \phi_1 + R_2 \sin \phi_2 + R_3 \sin \phi_3}{R_2R_3 \sin \phi_1 + R_1R_3 \sin \phi_2 + R_1R_2 \sin \phi_3}$$

where:

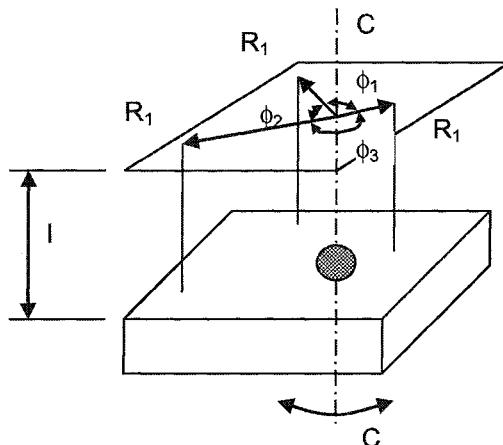
- R_1, R_2, R_3 = distance from axis through the centre of gravity to attachment point of wires
 ϕ_1, ϕ_2, ϕ_3 = angle between attachment points
 m = mass of test specimen
 l = length of wires
 τ = period of oscillation

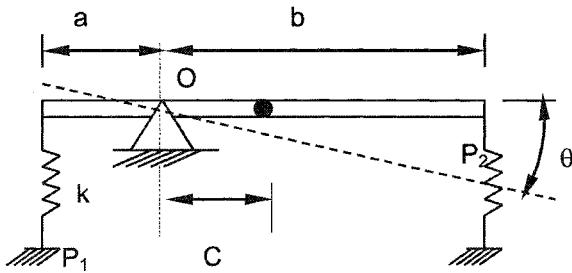
The trifilar pendulum can also be adapted for repeated use by installing a light platform, on three equally spaced wires. The body under investigation is placed on the platform with its centre of gravity equidistant from the wires. Thus $R_1=R_2=R_3=R$ and $\phi_1=\phi_2=\phi_3=120^\circ$. This arrangement is shown in figure above. When substituted into the previous equation, the moment of inertia simplifies to:

$$I = \frac{g\tau^2R^2(m_0 + m)}{4\pi^2l} - I_0$$

where:

- R = radius of platform [m]
 l = length of suspension wires [m]
 m_0 = mass of the unloaded platform [kg]
 m_1 = mass of the body being measured [kg]
 τ = period when loaded with unknown mass m_1 [s]
 I_0 = moment of inertia of platform [kg m^2]
 I = moment of inertia of test specimen [kg m^2]





6.7.4 Method 4

A fourth method of determining the moment of inertia of an object is a pivoted beam approach as shown in figure above. A beam is pivoted at a point O, in most cases the centre of gravity, with two uniform springs attached, at a distance a and b, to each end. The period of oscillation τ can be observed and plugged into the following equation to determine the moment of inertia. This method requires the test body to be pivoted and have springs attached, as a result, not all objects can be tested with this method.

$$I_0 = \frac{k(a^2 + b^2)\tau^2}{4\pi^2} = \frac{k(a^2 + b^2)}{\omega_n^2}$$

where k is stiffness of the springs and ω_n is the natural frequency.

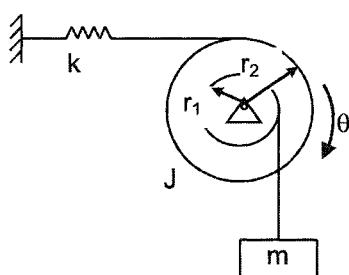
An energy method can also be used to determine the moment of inertia of test bodies. The experimental setup is shown in figure below, and involves a spring, mass, and pivot. The test specimen is pivoted about the investigated axis, and the moment of inertia is determined through the following equation.

$$I = \frac{kr_2^2\tau^2}{4\pi^2} - mr_1^2 = \frac{kr_2^2}{\omega_n^2} - mr_1^2$$

where:

- I = Total moment of inertia system
- k = spring constant
- m = mass of additional weight
- r_1 = inner radius
- r_2 = outer radius
- τ = period of oscillation
- ω_n = natural frequency

A number of other methods of determining the moment of inertia of an object are available. Each method is suited toward specific objects, with some methods producing more accurate results. The majority of experiments have simple setups and the results are easy to determine and compare.



6.8 Fabrication

The following items are some guides to various materials fasteners and bearings that may be considered. The rankings are based on strength, durability, ease of manufacturing/use, cost, mass, misuse, and aesthetics of the final product. The structural materials AER201Y students will find accessible are in order from best to worse:

- Sheets of Lexan (jig saw, circular saw)
- All-thread (ready rod) with doubled nuts as spacers/support (hacksaw)
- Sheet aircraft aluminum up to 1/32 inch thickness, 32 inch width (shear)
- Sheet regular aluminum up to 1/16 inch thickness, 32 inch width (shear)
- Plexiglas type Acrylic Sheet (jig saw)
- Plexiglas type Acrylic Rods (hacksaw)
- Plexiglas type Acrylic Blocks (hacksaw)
- Sheet steel up to 1/32 inch thickness, 32 inch width (shear)
- Blocks of wood greater than 1 inch cube as spacers/support (hacksaw)
- Good wood (oak, maple) boards (hacksaw, circular saw)
- unthreaded rod (hacksaw)
- Laminated (black or white) medium fiber compressed board sheet (jig saw, circular saw)\\[10pt]
- Aspenite, large chip board sheet (jig saw, circular saw)
- Structural Cardboard --- double or triple cored (xacto knife)
- Balsa Wood (xacto knife, hacksaw)
- Thermal Plastics (hacksaw)
- Metals thicker than \$1/32\$ inch or \$1/16\$ inch up to \$1/2\$ inch (hacksaw, jig saw)
- Other Acrylic materials and other very brittle plastics (nothing)
- Duct Tape (beer bottle opening front teeth)

Materials that should be avoided at all times:

- Stainless steel, tool steel (very expensive tools --- special hacksaws)
- Ceramics (high speed wet abrasive wheels)
- Metals thicker than \$1/2\$ inch (hacksaw)
- Compressed small fiber hard board sheet (garbage bin)

The options available for fasteners are, in order from best to worst:

- Velcro
- Adhesives if used properly --- epoxy
- Wood or metal screws in Lexan in a predrilled hole
- Pop rivets
- All-thread (ready rod) with doubled nuts as spacers/support

- Hot Glue in low stress applications
- Wood screws in wood
- Plastic Welding --- by adhesive
- ardox nails in wood
- Small nuts, bolts or screws
- Plastic Sonic Welding
- Metal screws in steel or aircraft aluminum
- Metal Spot Welding
- common nails in wood
- Metal screws in soft aluminum
- Crazy Glue --- because it is never used correctly
- Metal Welding

The options available for bearings are, in order from best to worst:

- wheels running on a flat surface
- greased Teflon
- greased nylon
- miniature ball bearing races (used in model cars?)
- lazy susan bearings
- recirculating ball bearings including transfer balls
- brass bearing inserts
- most other commercial ball bearing units

Bad choices for bearings are:

- none of the above
- threaded rod in a wood hole

The options available for geometry, in order from best to worst:

- medium sized holes up to 1/2 in sheet metal (punch)
- straight lines up to 32 inches in sheet metal (shear)
- cutting rods (hacksaw)
- small holes up to 3/8 inch in metal or plastic (drill press)
- straight lines in plastic sheet (jig saw)
- holes up to 1 inch in diameter in wood (wood hole saws)
- gently curved lines in plastic or wood (jig saw)
- almost straight lines in sheet metal (jig saw)
- holes larger than 1/2 and smaller than about 6 inches in anything (many drill holes, snips, file)

Some important properties of typical design construction materials are listed in table below. (Source: B. Davies, *Practical Robotics*, Toronto: WERD Technology Inc., 1997.) Young's modulus E is a measure of a material's stiffness, but weight is important too when building aircraft, robotic arms, etc. Therefore, a useful design parameter is E/ρ or specific modulus, which incorporates density ρ . Note that: 1 lb = 0.4536 kg ; 1 psi = 6895 Pa ; and 1 lb/in³ = 27680 kg/m³.

	density ρ (lbs/in ³)	density ρ (kg/m ³)	tensile strength (psi)	tensile strength (MPa)	Young's modulus E (psi)	Young's modulus E (GPa)	specific modulus (E/ρ)
aluminum	0.10	2700	10,000	70	12×10^6	80	1.2
brass	0.30	8500	60,000	400	15×10^6	100	0.5
copper	0.32	8900	22,000	150	17×10^6	120	0.5
fiberglass	0.065	1800	40,000	275	4×10^6	30	0.6
glass	0.094	2600	15,000	100	10×10^6	75	1.1
carbon composite	0.054	1500	400,000	2800	20×10^6	100-400*	3.7
Kevlar	0.049	1350	200,000	1400	12×10^6	80	2.5
Lexan	0.043	1200	8,000	55	0.3×10^6	2	0.1
nylon	0.041	1150	11,000	75	0.3×10^6	2	0.1
Plexiglas	0.042	1160	6,000	40	0.6×10^6	4	0.1
steel (mild)	0.28	7850	70,000	480	30×10^6	200	1.1
steel (piano wire)	0.28	7800	400,000	2800	30×10^6	200	1.1
wood (balsa)	0.008	215	2,000	14	1×10^6	7	1.3
wood (oak)	0.025	750	15,000	100	2×10^6	15	0.8
wood (pine)	0.015	410	6,000	40	1.5×10^6	12	1.0

6.9 Adhesives

Double-sided sticky foam tape, superglue, epoxy, hot-melt glue, contact cement, and silicone rubber happen to be very useful for design projects, not only for temporary assemblies but also in more permanent structures. For most design projects, 5-minute epoxy, Super (crazy) Glue (CA, cyanoacrylates), Weldbond™ space-age universal white glue, hot-melt glues, contact cement, and silicon seal (RTV rubber) fill most needs. Typical properties for adhesives often used are given in table below (Source: B. Davies, *Practical Robotics*, Toronto: WERD Technology Inc., 1997.) In addition, double-sided sticky tape (carpet tape) and conformable self-adhesive foam tapes could both find many uses in prototype fabrication. Metal plates stuck together with foam or carpet tape can only be separated with extreme difficulty, and foam tape will stick to irregular surfaces and support considerable loads. Removable Magic Tape™, manufactured by 3M, is very handy whenever a temporary non-marking tack is required. This tape is especially useful when delicate items such as cleaved fiber optic ends must be held firmly yet moved when necessary. Modeling clay, Plasticine™, Blu-Tak™, and similar sticky products are handy as temporary plugs for leaks, holding oddly shaped items, or as a retrieval tool when attached to a long wire. Permanent and temporary thread-lock adhesive compounds prevent fasteners from working loose with vibration. Reusable or temporary lockers are best for medium-size projects, since they work on wood, metal, and plastics, and unlike serrated or lock washers will not mar the finish on a special surface. These compounds are usually stocked at auto suppliers and hardware stores. Five-minute epoxy on threads is very good as a locking compound, and may be removed with debonder or by heating the screw with a soldering iron's tip.

BEST = ★★★	white glue	hot-melt glue	contact cement	super glue (CA)	epoxy	silicone seal
cardboard, paper	★★★	★★★	★★		★	★★★
wood	★★★	★	★★	★★	★★	★★★
fabric	★★	★★★	★★	★	★	★★
hard plastic			★★★	★★	★	★★
polyethylene, vinyl, rubber, Teflon			★★★	★★	★	★★
metal			★★	★★	★★	★★★
glass, ceramic	★★	★	★★★	★★	★★	★★★
water resistance		★★★	★★	★★★	★★★	★★★
tensile strength		★	★		★★	stretches well
flexibility	★	★	★★	★	★	★★★
fills gaps		★★★		★	★★★	★★★
viscosity	★★	★	★★	★★★	★	★
solvent (procure)	water	none	acetone	acetone	alcohol	none
solvent (post-cure)	hot water	none	acetone	acetone	debonder	softener
max. temperature	150 °F	140 °F	150 °F	200 °F	150 °F	600 °F
cure time	> 1h	~ 60s	~ 1h	< 60s	5min – 12h	24h

The following explains 5 steps to a better bond:

1) Geometry

Adhesives are characterized by their strength per unit area. To give a strong bond, there must be a large adhesive area. Figure below shows a good bond geometry with adequate bond area and one of the most common bad bond geometries, trying to glue an edge. It is clear the edge bond is much weaker in all ways however it is most vulnerable to bending loads. The flat bond has three main failure directions where peel is the weakest direction, pull is stronger and shear the strongest. For a material that is sufficiently stiff in bending, the peel failure is not a big problem. For a material that is very weak in bending but strong in tension, peel is a serious problem. For instance, gluing a piece of fishing line to a plate is a bad peel geometry. Gluing an L bracket to a flat plate is better than an edge-bond however if you assume a small amount of flexibility in the L material, bending it one way as noted causes a peel failure at the bottom. Gluing a T bracket or two L's back to back solves this problem.

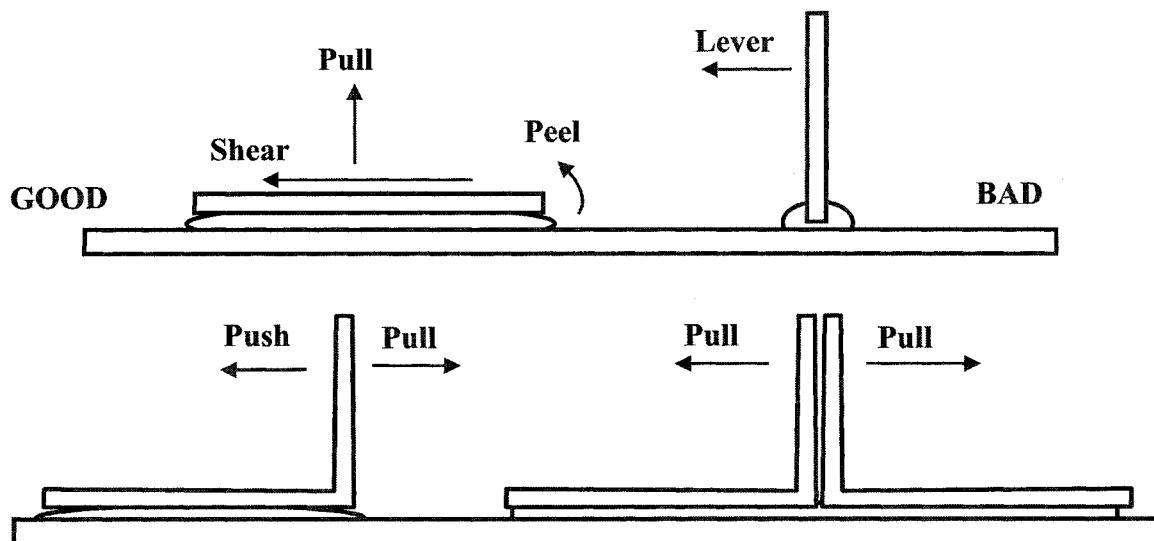
2) Roughen

Use fine sandpaper to remove surface contaminants and expose the raw material. Fine sandpaper also greatly

increases the surface area of the pieces to be bonded. Use coarse sandpaper to create deep scratches and produce a good mechanical bond. Always use semi-random circular motions to create scratches in all directions. The bond created then has no preferred direction and is less vulnerable to peel or shear in certain directions.

3) Clean

Clean the area to be bonded with alcohol and a clean wipe. Wipe the dirt *up* off the surface by rotating the wipe as it is drawn across the surface as shown in Figure 5.6. This exposes the surface to a constant contact with a clean portion of the wipe and can remove 99.9% or more of surface contaminants. Any other form of wiping, short of washing under a stream of fresh solvent, leaves far more contaminant behind. Contaminants and poor surface preparation are the source of mechanical weakness. Good quality crazy glue is as strong as aluminum if used properly. Unfortunately, unlike epoxy, crazy glue does not fill gaps well (the Aircraft cement Zap-a-Gap can do a reasonable job) so unless the surfaces fit together well, epoxy is best. As well epoxy does not pose as great a danger for skin and eye bonding as crazy glue so it is safer.

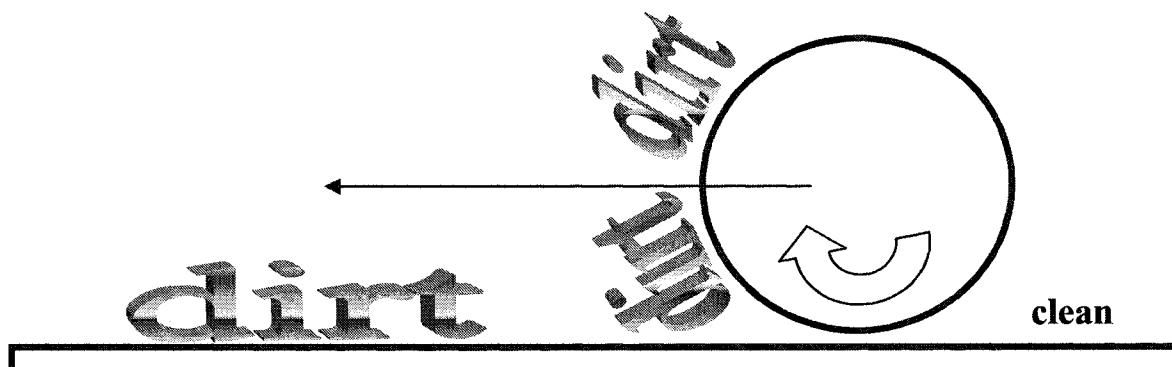


4) Bond

Place a small amount of glue between the surfaces as per the instructions for the glue. For rubber cement or carpenter's wood glue, coat both surfaces with a thin skim of glue. For rubber cement allow it to skin over before bonding. For all other glues, place a small quantity in the middle of the area to be glued and press the parts until it is extruded out all sides; wipe off the excess. For carpenter's wood glue, pressure is an important part of the bonding process so use a clamp. For crazy glue, the heat from your fingers is important to polymerization so hold the parts together. For epoxy, not being disturbed is most important so prop up the piece and leave it alone.

5) Wait

This is the most important part of the process. If students do not wait for a bond to form and instead test it before it is complete the bond will be permanently weakened.



6.10 Resources

- N. Slater, N. P. Chironis, *Mechanisms and Mechanical Devices Sourcebook*, MacGraw-Hill, 3rd ed., 2001.
- B. Davies, *Electric Motors and Mechanical Devices for Hobbyists and Engineers*, Toronto: WERD Technology Inc., 1997.
- J. L. Fuller, *Robotics: Introduction, Programming, and Projects*, NJ: Prentice Hall, 2nd ed., 1999.
- B. Davies, *Practical Robotics*, Toronto: WERD Technology Inc., 1997.
- D. M. Auslander, C. J. Kempf, *Mechatronics: Mechanical System Interfacing*, New NJ: Prentice Hall, 1996.
- W. Bolton, *Mechatronics*, Malaysia: Longman, 2nd ed., 1999.
- D. Benson, *Easy Step 'n: An Introduction to Stepper Motors for the Experimenter*, CA: Square 1, 2001.
- F. G. Martin, *Robotic Explorations*, NJ: Prentice Hall, 2001.
- Motor/Generator Links, <http://science.uniserve.edu.au/school/curric/stage6/phys/motors/power.html>: A good list of resources on electric motors on the Web.
- Curriculum for the Electric Motors, http://www.cpo.com/CPOCatalog/EM/em_curr.htm: A comprehensive set of tutorials about electric motors.
- Machines and Mechanisms, <http://www.mos.org/sln/Leonardo/InventorsToolbox.html>: Simple machines and mechanisms.

Chapter 7

PIC Microcontrollers

7.1	Choosing the Right Device	1
7.1.1	Embedded Controllers	1
7.1.2	FPGAs	1
7.1.3	Microprocessors vs. Microcontrollers	2
7.1.4	Microcontroller Industry	3
7.1.5	PIC Comparison	4
7.1.5.1	Low-end Devices	5
7.1.5.2	Mid-range Devices	5
7.1.5.3	High-end Devices	5
7.2	PIC Microcontrollers	6
7.3	Architecture	8
7.3.1	Block Diagram	12
7.3.2	Pin-Out Description (PIC16F877)	16
7.3.3	Pin-Out Description (PIC16F887)	18
7.3.4	Pin-Out Diagrams (40-pin PDIP)	21
7.3.5	Arithmetic and Logic Unit (ALU)	23
7.4	Memory Organization	24
7.4.1	PIC16 Program Memory	24
7.4.1.1	PIC16F877 Configuration Settings	25
7.4.1.2	PIC16F887 Configuration Settings	26
7.4.1.3	PIC16F1937 Configuration Settings	27
7.4.2	PIC18 Program Memory	29
7.4.2.1	Fast Call/Return Mode	31
7.4.2.2	Configuration Settings	31
7.4.3	PIC16 Data Memory	34
7.4.3.1	PIC16F877 Memory Map	34
7.4.3.2	PIC16F887 Memory Map	35
7.4.3.3	Indirect Addressing	38
7.4.4	PIC18 Data Memory	40
7.4.4.1	Special Function Registers	42
7.4.4.2	Indirect Addressing	44
7.4.5	PIC16 EEPROM Data Memory	44
7.4.5.1	EECON1 and EECON2	45
7.4.5.2	Writing to the EEPROM	45
7.4.5.3	Reading from the EEPROM	46
7.4.6	PIC18 EEPROM Data Memory	46
7.4.6.1	Writing to the EEPROM	46
7.4.6.2	Reading from the EEPROM	47
7.5	Additional Core Features	47
7.5.1	8x8 Hardware Multiplier	47
7.5.2	PLL and Internal Oscillator	47
7.5.3	Interrupts	48
7.5.3.1	Enabling Interrupt	48
7.5.3.2	Interrupt Support	50
7.5.3.2.1	Creating Interrupt Service Routine	50
7.5.3.2.2	Context Saving During Interrupts	51
7.5.3.2.3	Firmware Engineer's Responsibilities	51
7.5.4	PIC16 I/O Ports	51

7.5.4.1	PIC16F877	51
7.5.4.2	PIC16F887	53
7.5.4.3	PIC16F1937	54
7.5.5	PIC18 I/O Ports	56
7.5.6	Timers	57
7.5.6.1	PIC16 Timer Module	57
7.5.6.2	PIC18 Timer Module	57
7.6	Programming	59
7.6.1	MPASM Assembler	59
7.6.2	Source Code Format	59
7.6.3	PIC16F877/887 Instruction Set	60
7.6.3.1	Mnemonics	61
7.6.4	PIC16F1937 Instruction Set	64
7.6.4.1	Mnemonics	64
7.6.5	PIC18 Instruction Set	69
7.6.5.1	Mnemonics	70
7.6.6	Directives	74
7.6.7	Addressing	76
7.6.8	Programming Procedure	77
7.6.9	Simple Operations	78
7.6.9.1	Branching in PIC18	79
7.6.10	PIC16 Code Template	80
7.6.10.1	PIC16F877	80
7.6.10.2	PIC16F887	81
7.6.10.3	PIC16F1937	82
7.6.11	PIC18 Code Template	83
7.6.12	Macros	84
7.6.13	Tables	86
7.6.13.1	Tabling and the PCLATH Register	86
7.6.13.2	Tabling and Displaying String on LCD	87
7.6.14	Using Tables in PIC18	87
7.6.15	Data Parsing	89
7.6.16	Programming Checklist	91
7.7	Example	92
7.7.1	Sample LED Blinking Code	92
7.7.2	Sample Switch Debouncing Code	97
7.8	Interface with Keypad	99
7.9	Interface with LCD	103
7.9.1	Basic Overview of the Pins	103
7.9.2	Test Circuit	105
7.9.3	When Power Is First Applied	105
7.9.4	Addressing	106
7.9.5	Shifting the Display	106
7.9.6	Character Entry Mode	107
7.9.7	Interface with the PIC	107
7.9.7.1	PIC16F877 Sample Code	107
7.9.7.2	PIC16F887 Sample Code	110
7.9.7.3	PIC16F1937 Sample Code	112
7.9.7.4	PIC18 Sample Code	114
7.10	Capture/Compare/PWM (CCP) Module	118
7.10.1	Control Registers	118
7.10.1.1	CCPxCON Register	118
7.10.1.2	T2CON Register	119
7.10.2	Capture Mode	119
7.10.3	Compare Mode	119
7.10.4	PIC PWM Modules	120

7.10.4.1	Timing	120
7.10.4.1.1	PWM Period	120
7.10.4.1.2	PWM Duty Cycle	120
7.10.4.1.3	Resolution	121
7.10.4.1.4	Example	121
7.10.4.2	Setup	121
7.10.4.3	Analog Output Program (PIC16F877)	121
7.10.4.4	Analog Output Program (PIC16F887)	122
7.10.4.5	Analog Output Program (PIC16F1937)	123
7.10.4.6	Analog Output Program (PIC18)	123
7.10.4.7	Enhanced PWM Module	125
7.10.4.7.1	ECCP Control Register	125
7.11	A/D Module	126
7.11.1	Control Registers	126
7.11.1.1	ADCON0 Register (PIC16F877)	126
7.11.1.2	ADCON0 Register (PIC16F887)	127
7.11.1.3	ADCON0 Register (PIC16F1937)	128
7.11.1.4	ADCON0 Register (PIC18)	129
7.11.1.5	ADCON1 Register (PIC16F877)	130
7.11.1.6	ADCON1 Register (PIC16F887)	130
7.11.1.7	ADCON1 Register (PIC16F1937)	131
7.11.1.8	ADCON1 Register (PIC18)	132
7.11.1.9	ANSEL Register (PIC16F887)	132
7.11.1.10	ANSELH Register (PIC16F887)	133
7.11.1.11	ADCON2 Register (PIC18)	133
7.11.2	Operation	134
7.11.3	Timing	134
7.11.3.1	A/D Conversion Clock	134
7.11.3.2	Example	135
7.11.3.3	Acquisition Time Requirement	135
7.11.4	Simple A/D Converter (PIC16F877)	135
7.11.5	Simple A/D Converter (PIC16F887)	136
7.11.6	Simple A/D Converter (PIC16F1937)	138
7.11.7	Simple A/D Converter (PIC18)	139
7.12	USART/EUSART Module	141
7.12.1	Control Registers	142
7.12.1.1	TXSTA Register	142
7.12.1.2	RCSTA Register	143
7.12.1.3	BAUDCON Register (PIC16F887 and PIC16F1937)	144
7.12.1.4	BAUDCON Register (PIC18 Only)	145
7.12.2	Baud Rate Generator	145
7.12.3	Setup	146
7.12.4	Hardware	147
7.12.4.1	Sample Asynchronous TX Program	147
7.13	I ² C Bus	149
7.13.1	Overview	149
7.13.2	Theory	149
7.13.2.1	Communication – Hardware	149
7.13.2.2	Communication – Protocol	149
7.13.3	Master Device	149
7.13.3.1	Overview	149
7.13.3.2	Layered Framework	150
7.13.3.2.1	Hardware Layer	150
7.13.3.2.2	Process Layer	150
7.13.3.2.3	User Layer	150
7.13.3.3	PIC-RTC Programming	151

7.13.3.4	PIC16-PIC18 Extending I/O Set Programming.....	152
7.13.3.5	PIC16-PIC18 Communication for Register Watching	153
7.14	Universal Serial Bus Basics.....	154
7.14.1	Introduction.....	154
7.14.2	Hierarchy of Layered Framework for Device	154
7.14.3	USB Communication	154
7.14.4	Enumeration	156
7.15	Some Practical Notes.....	157
7.16	Computer Boards.....	158
7.16.1	Simple Configuration Board	158
7.16.2	The PIC DevBugger.....	160
7.16.2.1	Features	161
7.16.2.2	Programming.....	161
7.17	Resources.....	163

7.1 Choosing the Right Device

This chapter first gives an overview of the different types of processing units available for development purposes. It then reviews some basic features of the Peripheral Interface Controller (PIC) microcontrollers from Microchip Technology Inc., which are recommended for the design projects due to their fast operation, low power, low cost, and ease of programming.

Choosing the correct processor for the job is as important as doing the job properly. Different processors and microcontrollers exist ranging in very different configurations. Many of these will be able to do the job that is required, however some may be better optimized for the needs of the product based on price, size, pin-out configurations, etc. This section attempts to quickly guide the engineer through some considerations when choosing a device to work with.

7.1.1 Embedded Controllers

Embedded control has been incorporated into thousands of applications, and is seeing continued growth in a wide variety of markets. Embedded controllers are processors and I/O units that control a system without the need for an external computer. As technology progresses forward, more and more appliances and products are benefitting from automation and controls, and this increasing demand is driving the development of cheaper and more powerful embedded controllers. New embedded controllers offer the versatility required for almost every application. The most common types of controllers are microcontrollers, microprocessors, Complex Programmable Logic Devices (CPLDs) and Field Programmable Gate Arrays (FPGAs). Microcontrollers are simple, easy to implement and used for low-cost, integrated control applications. Microprocessors are used in high performance applications, from personal computers to mainframe systems. They must be implemented with a complex controlling circuit and external memory and I/O devices. The CPLDs and FPGAs are programmable logic devices; they can be programmed at a hardware level to emulate many different types of circuits.

7.1.2 FPGAs

Field Programmable Gate Arrays (FPGAs) fall into a niche of their own, and are beginning to see a wide variety of applications in telecommunication, digital signal processing, aerospace, vision systems, speech recognition, cryptography, etc. These programmable logic chips are configured via a Hardware Description Language (HDL), either Verilog or Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL). One of the major advantages of FPGAs is their versatility, since they can be configured to behave as any Application-specific Integrated Circuit (ASIC). The cost of an FPGA ranges from ten to hundreds of dollars, making them less applicable than microcontrollers in non-demanding tasks. However, FPGAs have seen considerable growth in low-volume applications, where the alternative of developing ASIC becomes too expensive to be profitable.

Two manufacturers dominate the FPGA industry – Xilinx and Altera. Together they control over 80% of the market. The FPGAs come in a variety of classes according to their processing power. Since they do not have a dedicated architecture, they can be programmed to efficiently perform low level logic functions, as well as to emulate microcontrollers and microprocessors. Using the design tools, ISE® and Quartus II, that Xilinx and Altera provide, one can configure the internal FPGA circuit and simulate its function and performance in a variety of ways.

Typically FPGAs consist of 4 main internal components – Logic Blocks, Input/Output Blocks, Embedded Memory, and Multiplier Blocks.

Logic Blocks are composed of programmable Logic Elements (LEs) consisting of Look-Up Tables (LUTs) and storage elements such as flip-flops. LEs can implement low level logic functions (such as AND and NOR) to more complex functions of several variables as well as store data.

I/O Blocks consist of a large number of I/O pins that can be programmed based on the internal logic configurations. These pins typically support bidirectional data flow and tri-state operation.

Memory Blocks usually consists of Static Random Access Memory (SRAM) for internal temporary data storage.

Multiplier Blocks are hardware embedded multipliers that greatly enhance the efficiency and speed of arithmetic operations.

Logic Elements are the smallest units of logic in an FPGA. The number of LEs in an FPGA chip usually determines the processing potential of the FPGA. A low-end Altera Cyclone II FPGA, for example, can have from 4,608 to 68,416 LEs, while a high-end Altera Stratix IV FPGA can have as many as 680,000 LEs. High-end FPGAs are widely used in the aerospace industry as well as other areas such as High Definition video manufacturing.

Programming an FPGA (also known as configuring) can be done in many ways. Different manufacturers have different ways of configuring their FPGA devices. However, most FPGAs support JTAG (Joint Test Action Group) Boundary Scan programming. Using this protocol, configuration programs can be downloaded directly onto the FPGA from an external PC.

7.1.3 Microprocessors vs. Microcontrollers

The microcontroller is perhaps the most widely seen modern electronic component. Many of us carry them with us at all times without realizing we even have one. One of the best known examples of embedded microcontrollers is in phone SIM cards. In essence, a microcontroller is a computer without a keyboard, monitor and mouse. It contains a CPU and Memory, as well as Input/Output devices. Most modern microcontrollers also have built-in peripherals such as A/D Converters and Timers.

A microprocessor is generally a device which contains the ability to execute instructions at high speeds while also being able to access external memory. Some common examples of today's microcontrollers are the Intel Pentium series of chips and AMD's Athlon chips. While these chips are significantly faster (3Ghz as opposed to 3-30mhz), they are not necessarily better suited to do the jobs required by AER201 students.

A small comparison between the Microprocessor and the Microcontroller has been prepared below:

Micropocessors	Microcontrollers
General purpose: Personal Computers to Mainframes, High-end Game Systems (Nintendo), Plant Control Units, etc.	Special purpose, "embedded systems", targeted for particular applications: Automobiles, Palm Pilot, Function Control Units, Vending Machines, Sensors, Appliances, etc.
\$50 - \$500	\$1 - \$50
Specialized for computation, requires additional systems, External RAM, ROM, and I/O Conversion	Integrated for minimal external systems and circuitry, often including I/O (even A/D), memory (RAM and ROM), and communication systems.
≥32 bits	4, 8, 16, 32 bits
Maximum Flops/\$	Minimum total system cost for application
Examples: Intel-Pentium, Motorola Power-PC	Examples: Motorola 'HC05, Microchip PIC18FXXXX

7.1.4 Microcontroller Industry

Microcontrollers serve in embedded control solutions for products and appliances in a wide range of markets worldwide. These applications include:

- Automotive subsystems
- Remote control devices
- Handheld tools
- Electronic Appliances
- Portable Computers
- Mobile and “Smart” phones
- Motor controls
- Security Systems

The increasing demand for embedded control has made the microcontroller industry a major part of the semiconductor market. The different types of available microcontrollers range from 4-bit to 32-bit architectures. Low-end 4-bit microcontrollers are the least expensive, usually less than \$1.00, but often do not meet the performance requirements of the user. High-performance 16-bit and 32-bit microcontrollers are often too expensive for most embedded control systems, typically costing between \$6.00 and \$12.00 each. As a result, 8-bit microcontrollers have found the widest application due to its good performance for a low cost. However, as embedded control systems become more complex, high-end 16-bit and 32-bit microcontrollers have seen considerable market growth as new applications opened up while costs are driven down. Although 8-bit microcontrollers are still the most common type in usage, their growth has leveled off. In 2008, the 8-bit microcontroller market was down 5% to about \$5 billion, while the 16-bit and 32-bit MCU market reached \$4.3 billion and \$4 billion respectively.

Major manufacturers of microcontrollers include Renesas Technology, Freescale Semiconductor, NEC, Fujitsu, Infineon Technologies and Microchip Technology and Atmel. Figure 7.1-1 shows the market share controlled by these companies as of 2008. In this course, we will be focusing on the PICmicro microcontrollers from Microchip Technology.

Since 2002, Microchip Technology Inc. has been the biggest producer of 8-bit microcontrollers worldwide, offering a wide range of unique products with many features to suit most embedded system requirements.

Company	2008		2008		2007		2007	
	Rank	\$M	Share	Rank	\$M	Share	Y/Y %	
Renesas Technology	1	2,770	20.1%	1	2,944	21.2%	-6%	
Freescale Semiconductor	2	1,518	11.0%	2	1,743	12.6%	-13%	
NEC	3	1,330	9.7%	3	1,298	9.3%	3%	
Fujitsu	4	1,065	7.7%	4	1,115	8.0%	-5%	
Infineon Technologies	5	983	7.2%	5	1,023	7.4%	-4%	
Microchip Technology	6	812	5.9%	6	778	5.6%	4%	
STMicroelectronics	7	645	4.7%	7	662	4.8%	-3%	
Texas Instruments	8	601	4.4%	8	607	4.4%	-1%	
Atmel	9	511	3.7%	9	458	3.3%	12%	
NXP Semiconductors	10	286	2.1%	10	303	2.2%	-6%	
Other		3,229	23.5%		2,936	21.2%	10%	
Total		13,749			13,866		-1%	

data based on estimates, Company Reports

Figure 7.1-1

7.1.5 PIC Comparison

The Microchip PICmicro™ family of microcontrollers has been chosen to be used as the production microcontroller. However, even within the PICmicro family there are many different architectures including PIC10F, PIC12F, PIC16F, PIC18F. Within these general families, the number of microcontrollers branches further.

Currently there are over 250 unique PICmicro part numbers available. Table below gives a brief overview of the different families of products available. Detailed information about these MCUs are available from the Microchip Technology Inc. website: www.microchip.com.

<i>Part Number</i>	<i>Architecture</i>	<i>Features</i>	<i>Applications</i>
10F2xx	Base-line		Simple Interfacing Osc/Reset
10F5xx	Base-line		Simple Interfacing Osc/Reset
12F508	Base-line		Simple Interfacing Osc/Reset
12F510	Base-line	ADC/Vref	Basic Applications
16F5xx	Base-line	Comparator/ADC	Basic Applications
16F81x	Base-line	CCP (PWM), SPI/I ² C	Basic Applications
16F6xx	Mid-Range		Analog Monitoring
16F7xx	Mid-Range		Analog Monitoring
16F887	Mid-Range		Analog Monitoring
16F9xx	Mid-Range		Analog Interfacing
16F1xxx	Enhanced Mid-Range		Application Development
18Fxx5x	High-End	USB Support	Serial Interfacing/Communication
18Fxx8x	High-End	CAN Bus Support	Automotive Control Network
18Fxx9x	High-End	LCD Driver	Advanced Applications
18Fxx3x	High-End	ECCP (PWM)	Motor Control
18Fxx2x	High-End	General Purpose/Low Power	Application Development
18FxxJ6x	High-End	Ethernet Support	Communication
18FxxKxx	High-End	Up to 16 MIPS	Advanced Applications

Table 7.1-1

7.1.5.1 Low-end Devices

These devices include the 10Fxxx, 12F5xx and 16F5xx families. They should be considered for specific applications with the following requirements:

- Simple interface functions
- Limited variable memory
- Simple digital interfaces

Many of the low-end MCUs only have space for 512 instructions with the maximum being 2K words. The latest devices should be compatible with the mid-range family of devices in terms of register and resource names. For the most part, these devices are not capable of carrying the application requirements of the AER201 projects simply due to programming word restrictions and should be used for simpler projects such as customized timer chips.

7.1.5.2 Mid-range Devices

The mid-range architecture is perhaps the most common architecture visible on the market today. Due to its' cost/performance ratio, it is very popular for hobbyist projects as well as professional applications. Its instruction set is simple and easy to learn and use. The mid-range PICmicro MCU architecture uses the low-end's architecture while enhancing it with more registers. This allows the end user to implement more advanced I/O peripheral devices.

7.1.5.3 High-end Devices

The high-end PIC18s expand on the mid-range architecture by allowing for a greater number of I/O pins and significantly larger memory maps, as well as adding new peripheral features. These high performance microcontrollers feature an extended instruction set to enhance program optimization and built-in C programming optimizing features. The PIC18 architecture also comes with built-in hardware multipliers for more efficient arithmetic operations. The PIC18 architecture is code-compatible with the mid-range PIC architectures. This means that any programs developed on the PIC16 should be compatible with the PIC18 (with some small exceptions, such as commands to change the page bits on the PIC16).

Perhaps the most exciting feature of the PIC18 family is the ability to read and write to/from the stack. This allows for the implementation of a Real Time Operating System (RTOS) environment.

7.2 PIC Microcontrollers

The Peripheral Interface Controller (PIC) chips, referred to as PICmicro™ Micro Controller Units (MCUs), are 8-bit general-purpose microcontrollers, which offer a price/performance ratio that allows them to be considered for any traditional 8-bit MCU application as well as some traditional 4-bit applications (Base-Line family), dedicated logic replacement, and low-end DSP applications (High-End family). The PICmicro architecture is based on the RISC (Reduced Instruction Set Computer) design. Its code is very efficient, allowing the PIC to run with typically less program memory than its competitors, such as Intel 8052 and Motorola 68HC11. Nearly all instructions execute in the same number of clock cycles, which makes timing control much easier. These features and the price-performance combination make PICmicro MCUs an attractive solution for the majority of mechatronics applications.

PICmicro 8-bit devices are grouped by the size of their Instruction Word. The three current PICmicro families are:

1. **Base-Line:** 12-bit Instruction Word length
2. **Mid-Range:** 14-bit Instruction Word length
3. **High-End:** 16-bit Instruction Word length

Different versions of the above PICmicro families are available that are equipped with various combinations of EEPROM, Enhanced FLASH program and data memory. Since the cost of the MCUs is proportional to the manufacturing cost, PIC microcontrollers with higher specifications are not necessarily more expensive. The recommended chips for the AER 201 projects are PIC16F877, PIC16F887, PIC16F1937 and PIC18F4620 with the following specifications:

PIC Micro-controller	Data RAM (Bytes)	Data EEPROM (Bytes)	Speed MHz	I/O Ports	ADC 10-Bit Ports	Serial I/O	PWM	Brown Out	Comparators	Timers	ICSP	Flash Program Memory (Bytes)	Flash Program Memory (Words)
16F877	368	256	20	33	8	USART/MSSP	2	Yes	-	3+WDT	Yes	14336	8192
16F887	368	256	20	36	14	EUSART/MSSP	Up to 5	Yes	2	3+WDT	Yes	14336	8192
16F1937	512	256	32	36	14	EUSART/MSSP	Up to 15	Yes	2	5+WDT	Yes	14336	8192
18F4620	3968	1024	40	36	13	EUSART/MSSP	Up to 5	Yes	2	4+WDT	Yes	65536	32768

Table 7.2-1

The chip comes in a variety of pin configurations, but the preferred one is the 40-pin Dual In-line Package (DIP) that is more convenient for amateur installation and implementation. The core features of PIC18F4620, and also PIC16F877, PIC16F887, and PIC16F1937 [in brackets] are:

- High-performance RISC CPU
- 75 core instructions [35 instructions for PIC16F877/887, 49 instructions for PIC16F1937]
- C compiler optimized architecture with optional extended instruction set designed to optimize re-entrant code (PIC18F and PIC16F1937 only)
- Mostly single cycle instructions
- Up to 32K words (16 bits) of FLASH Program Memory [8K words (14 bits) for PIC16F877/887/1937]; Up to 3968 bytes (8 bits) of data memory (RAM) [368 bytes for PIC16F877/887, 512 bytes for PIC16F1937]; Up to 1024 bytes (8 bits) of EEPROM data memory [256 bytes for PIC16F877/887/1937]
- Self-programmable under software control (PIC18F only)
- Interrupt capability (up to 20 sources) with 2 interrupt priority levels [up to 14 sources for PIC16F877/887/1937]
- 31 level hardware stack [8 level stack for PIC16F877/887, 16 level stack for PIC16F1937]
- 8 x 8 single cycle hardware multiply (PIC18F only)
- Direct, indirect and relative addressing modes (Enhanced core register locations and linear mapping for PIC16F1937)
- Power-on Reset (POR)

- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Extended Watchdog Timer (WDT) with programmable period from 4 ms to 131 s (1 ms to 256 s for PIC16F1937)
- Programmable code-protection
- Run mode down to 11 μ A currents typical
- Sleep mode down to 100nA currents
- 4 crystal oscillator modes
- 31 kHz to 8 MHz internal oscillator (PIC18F and PIC16F887/1937 only)
- Low-power, high-speed NanoWatt technology (PIC18F and PIC16F887/1937 only)
- Fully static design
- Single 5V In-Circuit Serial Programming (ICSP) capability via two pins
- In-Circuit Debugging via two pins
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial and Industrial temperature ranges

And, the peripheral features of PIC18F4620 and PIC16F877 are:

- Three programmable external interrupts [one external interrupt for PIC16F877/887/1937]
- Four input change interrupts [Eight for PIC16F887/1937]
- Enhanced Capture/Compare/PWM (ECCP) module supporting up to 4 PWM outputs [2 CCP modules for PIC16F877, 3 ECCP and 2 CCP for PIC16F1937]
- Master Synchronous Serial Port (MSSP) with SPI (All Modes) and I²C (Master/Slave)
- Enhanced Addressable Universal Synchronous Asynchronous Receiver-Transmitter (EUSART) with 9-bit address detection [USART for PIC16F877]
- 10-bit, up to 13-Channel Analog-to-Digital converter [8 Channels for PIC16F877, 14 Channels for PIC16F887/1937]
- Dual Analog Comparators with input multiplexing (PIC18F and PIC16F887/1937 only)
- Programmable 16-Level High/Low-Voltage Detection (HLVD) (PIC18F only)
- One 8-bit and three 16-bit timers/counters [two 8-bit and one 16-bit timers for PIC16F877/887, four 8-bit and 1 16-bit timers for PIC16F1937]
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (PIC18F and PIC16F877 only)
- Brown-out detection circuitry for Brown-out Reset (BOR)
- Integrated LCD Controller with internal voltage reference selections and variable clock input (PIC16F1937 only)
- SR Latch (555 Timer) with multiple set/reset input options (PIC16F1937 only)
- Voltage Reference Module (PIC16F1937 only)
- Capacitive Sensing Module (mTouchTM) with up to 16 selectable channels (PIC16F1937 only)

7.3 Architecture

The high performance of the PICmicro chips can be attributed to a number of architectural features as explained below.

Harvard Architecture

PICmicro chip design is based on the *Harvard* architecture to achieve an exceptionally fast execution speed for a given clock. The program memory and data memory in the Harvard architecture are separate, and accessed from separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. The comparison of Harvard vs. von-Neumann architectures is shown in Figure 7.3-1. To execute an

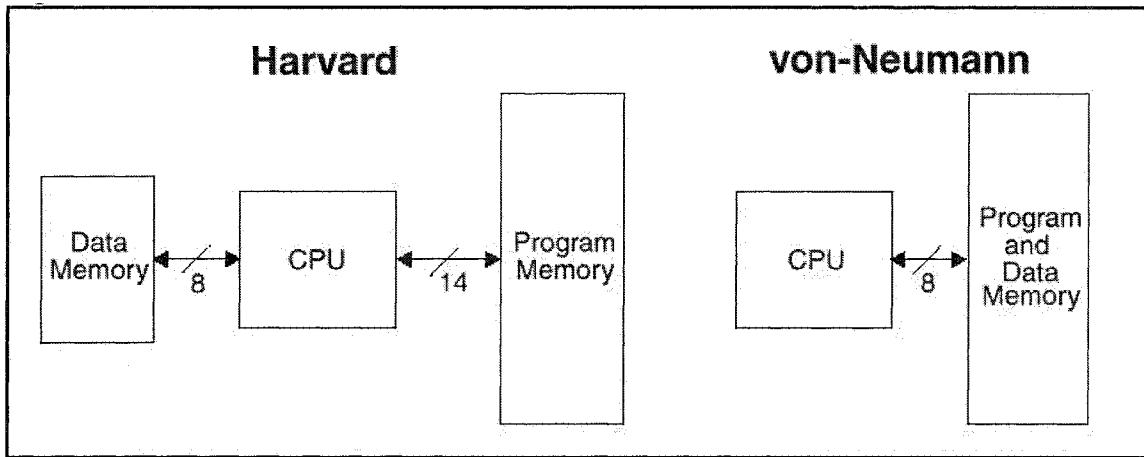
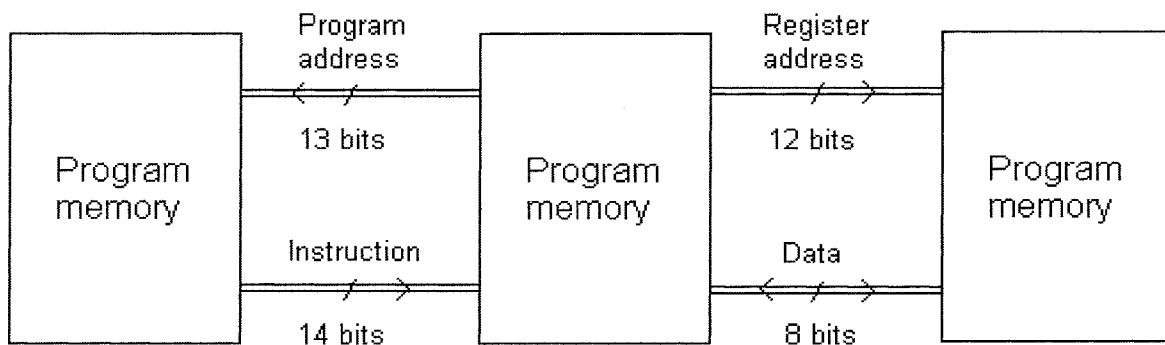


Figure 7.3-1

instruction, a von Neumann machine must make one or more (generally more) accesses across the 8-bit bus to fetch the instruction. Then data may need to be fetched, operated on, and possibly written. As a result, the bus can be extremely congested in von-Neumann architecture. With Harvard architecture, on the other hand, *instructions* are fetched from program memory using buses that are distinct from the buses used for accessing *variables* in data memory, I/O ports, etc. On the PIC16F877/887/1937, every instruction is coded as a *single* 14-bit word and fetched over a 14-bit-wide bus. Consequently, as instructions are fetched from successive program memory locations, a new instruction is fetched every cycle, as depicted in figure 7.3-2. Hence, separated buses allow one instruction to execute while the next instruction is fetched. However, one drawback to the Harvard architecture is that it is very difficult to bring the memory address and data busses out to device pins, so adding external program memory is difficult at best. For this reason, most Harvard machines have only internal program memory. For example, the PIC16F877/887 contains 8k words of FLASH program memory, 368 bytes of data RAM, and 256 bytes of data EEPROM. While this seems like a rather limited amount of code and data space, the PIC's extremely compact coding makes the most of it; 8×1024 instruction word memory actually means 8×1024 instructions, no less. Even immediate-mode instructions, where an operand is part of the instruction itself, takes only one memory location, as do CALL and GOTO instructions. Also, PIC16F1937 has an enhanced architecture which increases the memory up to 512 bytes of data RAM.



Instruction Pipeline

In the PICmicro family, the CPU executes each instruction during the cycle following its fetch, *pipelining* instruction fetches and instruction executions to achieve the execution of one instruction every cycle. This is illustrated in Figure 7.3-3. It can be seen that while each instruction requires two cycles (a fetch cycle followed by an execute cycle), the overlapping of the execute cycle of one instruction with the fetch cycle of the next instruction leads to the execution of a new instruction every cycle.

This lockstep progression is broken whenever an instruction includes a branch operation, as illustrated in Figure 7.3-3. In this

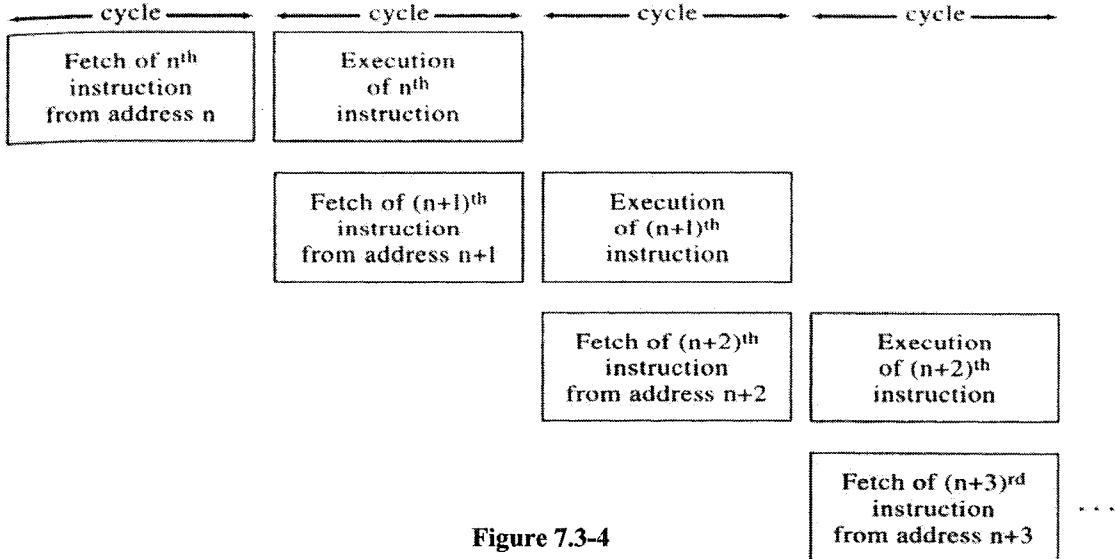


Figure 7.3-4

example, an instruction is fetched during the second cycle, **goto NewAddress**, whose job is to change the normal flow of instruction fetches from one address to the next address. During the third cycle, the CPU carries out the sequential fetch from address n+2. At the end of that third cycle, the CPU executes the **goto NewAddress** instruction by changing the program counter to **NewAddress** instead of simply incrementing it to n+3. On the fourth cycle while it is fetching the instruction at **NewAddress**, it ignores the instruction automatically fetched from address n+2. While this (n+2)th instruction is *located* in the program immediately after the (n+1)th **goto NewAddress** instruction, it is *never executed* immediately after the execution of that (n+1)th **goto NewAddress** instruction.

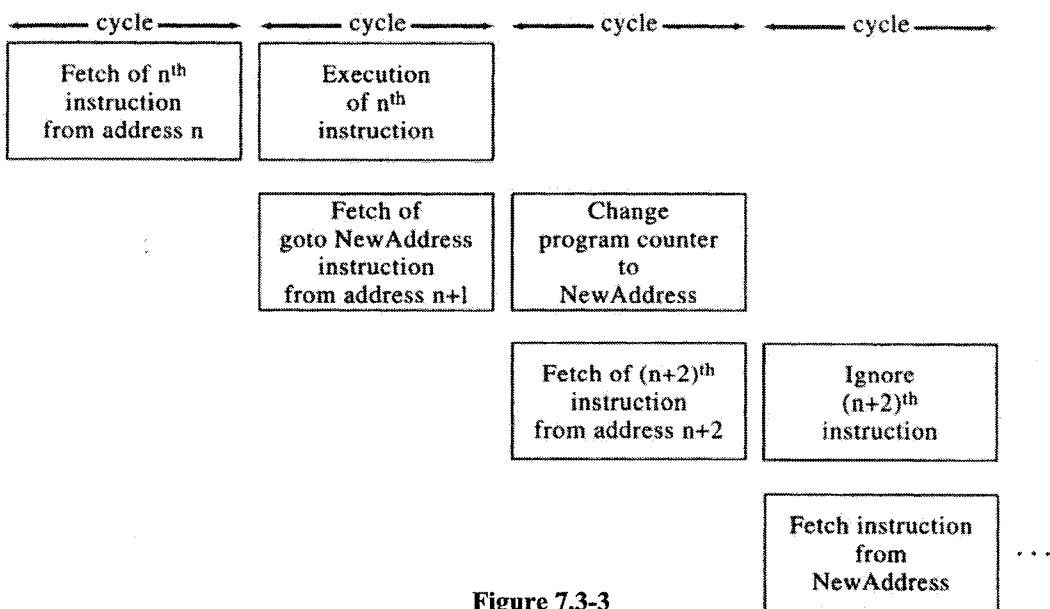


Figure 7.3-3

Long, Single Word Instruction

The 16F series of PIC processors use a 14-bit instruction word. Other PIC processor series use 8-, 12- or 16-bit cores. Long word instructions have a wider (more bits) instruction bus than the 8-bit Data Memory Bus. This is possible because the two buses are separate. The number of bits in the instruction word does not affect how data is stored; the data path is 8 bits wide, and we still treat data as bytes (or bits) and handle it in 1- or 8-bit chunks. This further allows instructions to be sized differently than the 8-bit wide data word, which allows a more efficient use of the program memory, since the program memory width is optimized to the architectural requirements. The 14-bit instruction path is split out by decode logic into different control signals. These control signals determine exactly how data flows through the PIC's internal busses and registers. As an example of how this works, let us look at one PIC instruction and see how the processor uses the instruction. The following table summarizes the ADDWF (Add W with F) instruction.

Mnemonic	Description	Cycles	14-Bit opcode	Flags
ADDWF f, d	Add W and F	1	00 0111 dfff ffff	C,DC,Z

The "fff ffff" bits are used to select a RAM address, known as a "file register". The "d" is a destination bit, which selects where the result is gated into -- 0 for the W register, or 1 to send the data back to the register we used. This means that you can manipulate a register or data variable pretty much "in place" using only one instruction. The first few bits, "00 0111", mean that this is the ADDWF instruction. The first two bits designate it as a byte-oriented instruction, and the next four select the function performed by the MUX, ALU and WREG. Again, these bits directly control gates that control the flow of data through the chip, as well as selecting the ALU function.

This is one reason why the PIC instruction set is so small. With single word instructions, the number of words of program memory locations equals the number of instructions for the device. This means that all locations are valid instructions. Typically in the von Neumann architecture, most instructions are multi-byte. In general, a device with 4-kBytes of program memory would allow approximately 2k of instructions in von Neumann architecture. The PIC RISC architecture also lets us perform the exact same operation on RAM data memory, an I/O port, or any control register, such as those used for port pin direction control (TRISA, TRISB), timer/counter control, interrupt control, EEPROM address or data, and indirect addressing (just to name a few).

With the Program Memory bus being 14-bits wide, the entire instruction is fetched in a single machine cycle (T_{CY}). The instruction contains all the information required and is executed in a single cycle. There may be a one-cycle delay in execution if the result of the instruction modifies the contents of the Program Counter. This requires the pipeline to be flushed and a new instruction to be fetched.

Orthogonal (Symmetric) Instructions

Orthogonal instructions make it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of "special instructions" make programming simple yet efficient. In addition, the learning curve is reduced significantly. The mid-range instruction set uses only two non-register oriented instructions, which are used for two of the cores features. One is the SLEEP instruction that places the device into the lowest power use mode. The other is the CLRWDT instruction that verifies the chip is operating properly by preventing the on-chip Watchdog Timer (WDT) from overflowing and resetting the device.

Register File Architecture

The register files/data memory can be directly or indirectly addressed. All special function registers, including the program counter, are mapped in the data memory.

Reduced Instruction Set

When an instruction set is well designed and highly orthogonal (symmetric), fewer instructions are required to perform all needed tasks. With fewer instructions, the whole set can be more rapidly learned.

The PIC18 devices also use Harvard architecture to separate the program memory from the data memory. To provide enhanced performance and more processing capabilities, PIC18s are equipped with more program memory and data memory. PIC18s also feature a more comprehensive instruction set of 75 instructions to greatly increase runtime efficiency and allow shorter and simpler algorithms. Therefore, every instruction is coded as a 2-byte or 16-bit word and fetched over a 16-bit-wide bus. The PIC18F4620 contains 64kBytes of FLASH program memory, 3968 bytes of data RAM, and 1024 bytes of data EEPROM.

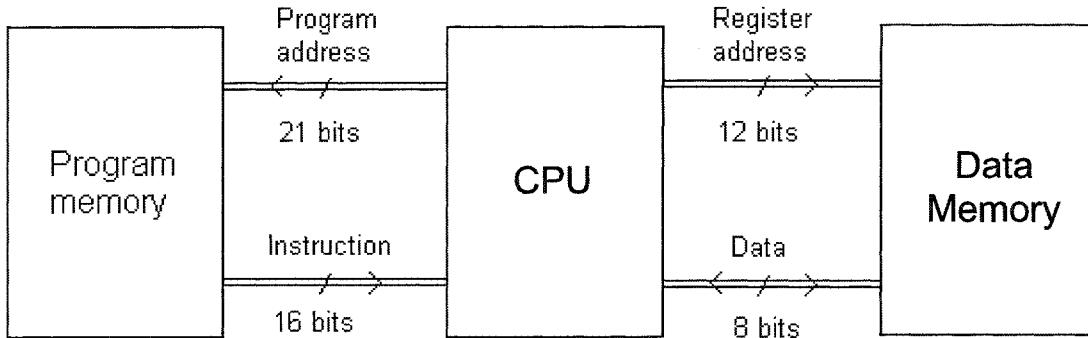


Figure 7.3-5

The 18F series of PIC processors use a 16-bit instruction word. The program memory bus is 2-bits wider than the PIC16 because of the greater number of instructions in the instruction set. The format of the instructions is almost identical to the PIC16. For example, consider the PIC18 ADDWF instruction:

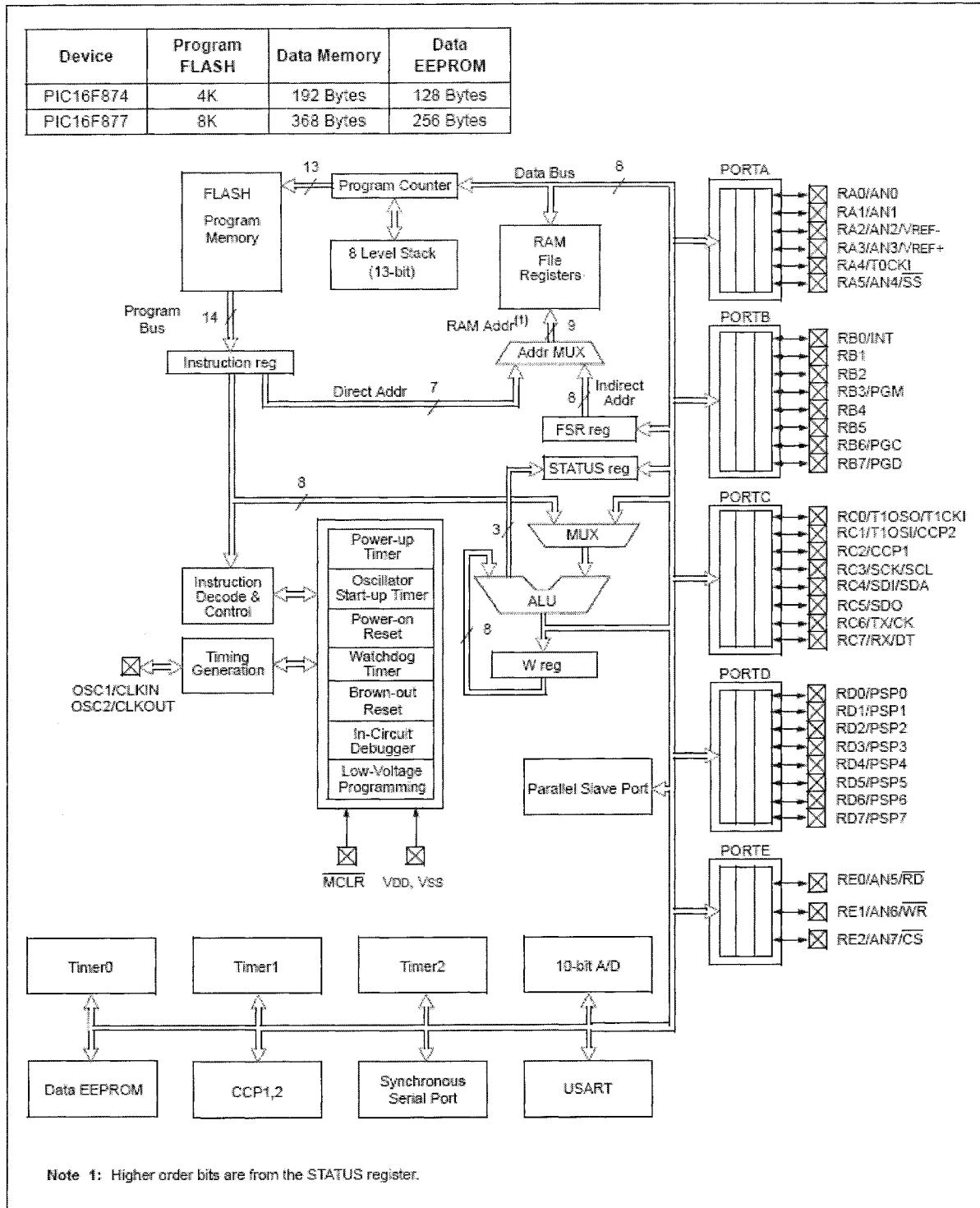
Mnemonic	Description	Cycles	16-Bit opcode	Flags
ADDWF f, d, a	Add W and F	1	0010 01da ffff ffff	C,DC,Z,OV,N

The "ffff ffff" bits are used to select a file register address. The "d" is a destination bit, which selects where the result is gated into -- 0 for the W register, or 1 to send the data back to the register we used. The "a" is the RAM access bit, which specifies which part of the RAM is to be used – 0 for a location in the Access RAM and 1 for a location in the Banked RAM. Data memory organization will be further discussed in section 7.4.4. The first six bits, "0010 01", stands for the ADDWF instruction.

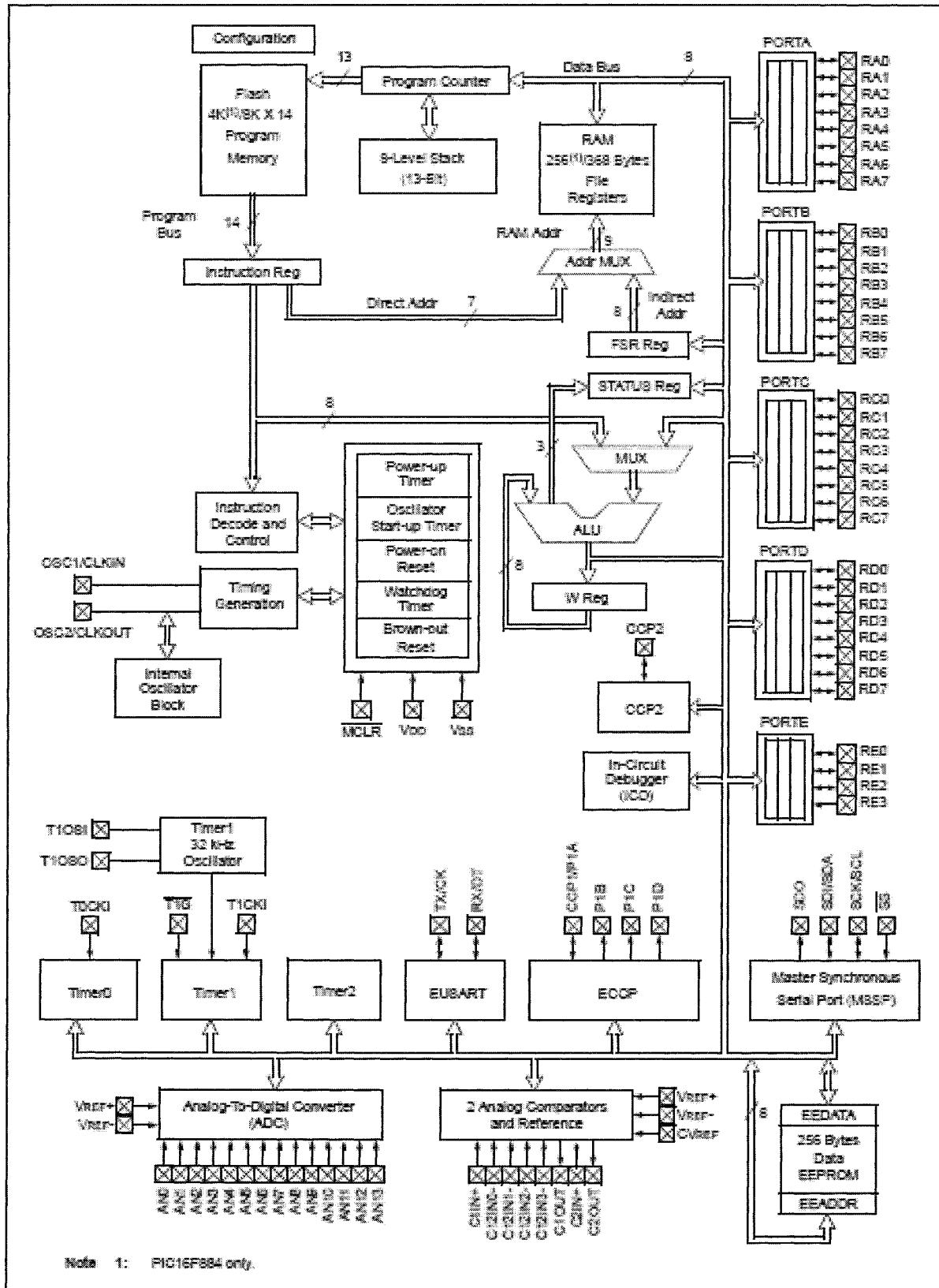
Most instructions in PIC18 devices are single word instructions, but 4 instructions are 2 words long. These instructions contain full program memory or data memory addresses to eliminate program memory paging or selecting the right data bank, reducing potential errors and debugging time. The instruction set will be discussed in detail in section 7.6.5.

7.3.1 Block Diagram

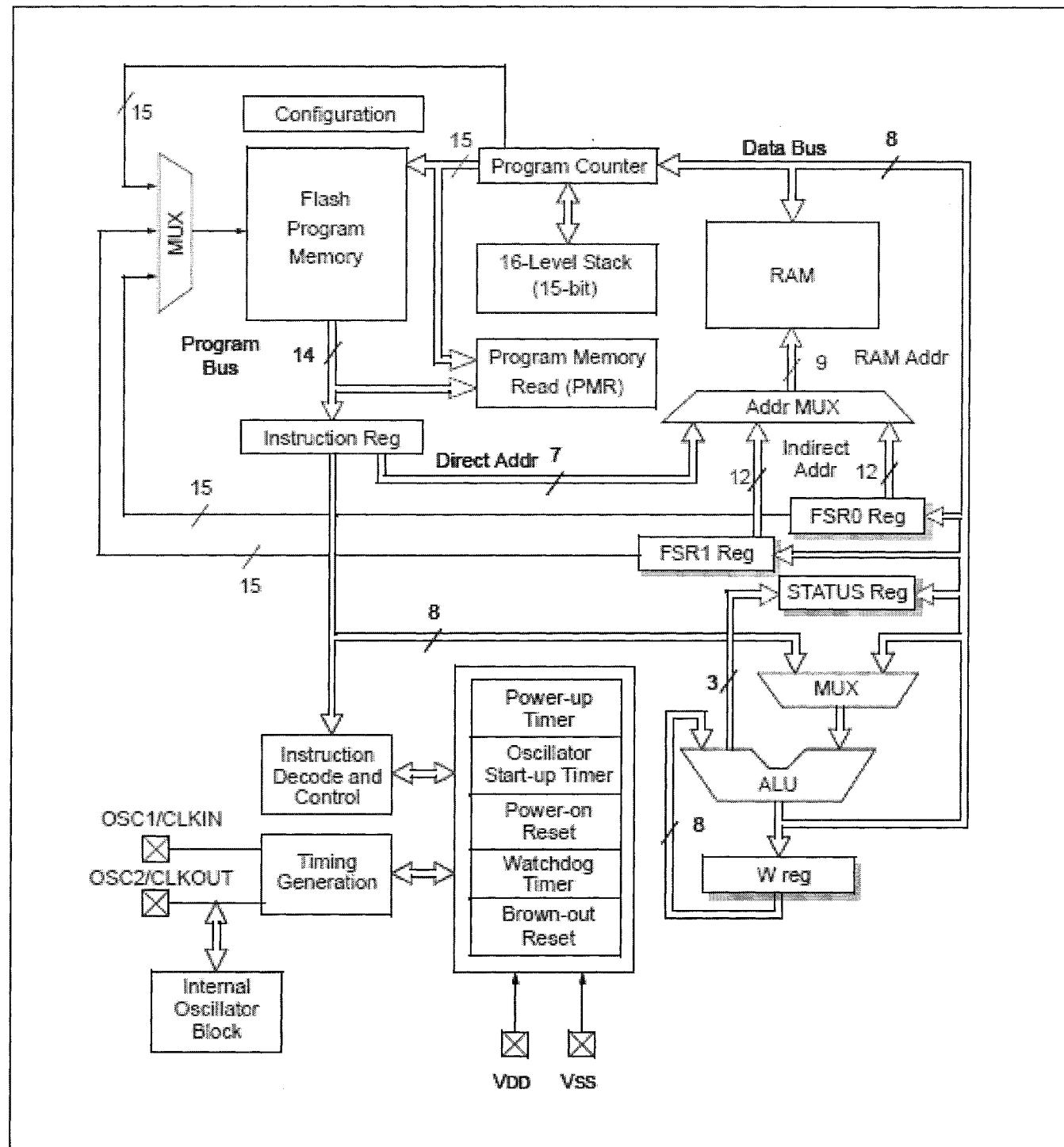
PIC16F877



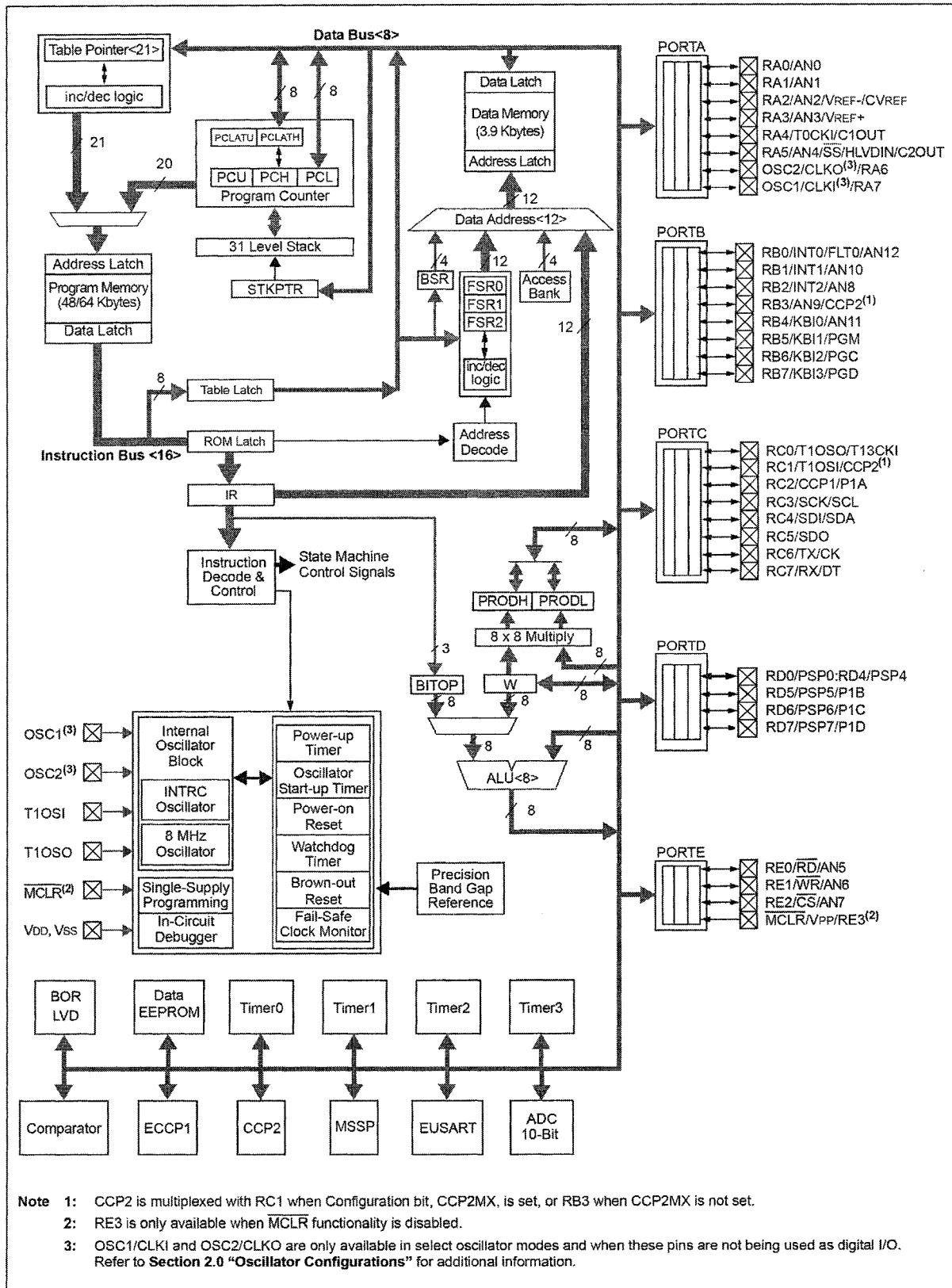
PIC16F887



PIC16F1937



PIC18F4620



7.3.2 Pin-Out Description (PIC16F877)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS ⁽⁴⁾	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
RA0/AN0	2	3	19	I/O	TTL	PORTA is a bi-directional I/O port. RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/V _{REF} -	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/V _{REF} +	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/SS/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
RB0/INT	33	36	8	I/O	TTL/ST ⁽¹⁾	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST ⁽²⁾	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST ⁽²⁾	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC is a bi-directional I/O port. RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I ² C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I ² C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.

RD0/PSP0	19	21	38	I/O	ST/TTL ⁽³⁾	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD1/PSP1	20	22	39	I/O	ST/TTL ⁽³⁾	
RD2/PSP2	21	23	40	I/O	ST/TTL ⁽³⁾	
RD3/PSP3	22	24	41	I/O	ST/TTL ⁽³⁾	
RD4/PSP4	27	30	2	I/O	ST/TTL ⁽³⁾	
RD5/PSP5	28	31	3	I/O	ST/TTL ⁽³⁾	
RD6/PSP6	29	32	4	I/O	ST/TTL ⁽³⁾	
RD7/PSP7	30	33	5	I/O	ST/TTL ⁽³⁾	
RE0/RD/AN5	8	9	25	I/O	ST/TTL ⁽³⁾	PORTE is a bi-directional I/O port. RE0 can also be read control for the parallel slave port, or analog input5.
RE1/WR/AN6	9	10	26	I/O	ST/TTL ⁽³⁾	
RE2/CS/AN7	10	11	27	I/O	ST/TTL ⁽³⁾	
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
Vdd	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
 — = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).

4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

7.3.3 Pin-Out Description (PIC16F887)

Name	Function	Input Type	Output Type	Description
RA0/AN0/ULPWU/C12IN0-	RA0	TTL	CMOS	General purpose I/O.
	AN0	AN	—	A/D Channel 0 input.
	ULPWU	AN	—	Ultra Low-Power Wake-up input.
	C12IN0-	AN	—	Comparator C1 or C2 negative input.
RA1/AN1/C12IN1-	RA1	TTL	CMOS	General purpose I/O.
	AN1	AN	—	A/D Channel 1 input.
	C12IN1-	AN	—	Comparator C1 or C2 negative input.
RA2/AN2/VREF-/CVREF/C2IN+	RA2	TTL	CMOS	General purpose I/O.
	AN2	AN	—	A/D Channel 2.
	VREF-	AN	—	A/D Negative Voltage Reference input.
	CVREF	—	AN	Comparator Voltage Reference output.
	C2IN+	AN	—	Comparator C2 positive input.
RA3/AN3/VREF+/C1IN+	RA3	TTL	CMOS	General purpose I/O.
	AN3	AN	—	A/D Channel 3.
	VREF+	AN	—	A/D Positive Voltage Reference input.
	C1IN+	AN	—	Comparator C1 positive input.
RA4/T0CKI/C1OUT	RA4	TTL	CMOS	General purpose I/O.
	T0CKI	ST	—	Timer0 clock input.
	C1OUT	—	CMOS	Comparator C1 output.
RA5/AN4/SS/C2OUT	RA5	TTL	CMOS	General purpose I/O.
	AN4	AN	—	A/D Channel 4.
	SS	ST	—	Slave Select input.
	C2OUT	—	CMOS	Comparator C2 output.
RA6/OSC2/CLKOUT	RA6	TTL	CMOS	General purpose I/O.
	OSC2	—	XTAL	Crystal/Resonator.
	CLKOUT	—	CMOS	Fosc/4 output.
RA7/OSC1/CLKIN	RA7	TTL	CMOS	General purpose I/O.
	OSC1	XTAL	—	Crystal/Resonator.
	CLKIN	ST	—	External clock input/RC oscillator connection.
RB0/AN12/INT	RB0	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN12	AN	—	A/D Channel 12.
	INT	ST	—	External interrupt.
RB1/AN10/C12IN3-	RB1	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN10	AN	—	A/D Channel 10.
	C12IN3-	AN	—	Comparator C1 or C2 negative input.
RB2/AN8	RB2	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN8	AN	—	A/D Channel 8.
RB3/AN9/PGM/C12IN2-	RB3	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN9	AN	—	A/D Channel 9.
	PGM	ST	—	Low-voltage ICSP™ Programming enable pin.
	C12IN2-	AN	—	Comparator C1 or C2 negative input.

RB4/AN11	RB4	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN11	AN	—	A/D Channel 11.
RB5/AN13/T1G	RB5	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	AN13	AN	—	A/D Channel 13.
	T1G	ST	—	Timer1 Gate input.
RB6/ICSPCLK	RB6	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPCLK	ST	—	Serial Programming Clock.
RB7/ICSPDAT	RB7	TTL	CMOS	General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.
	ICSPDAT	ST	TTL	ICSP™ Data I/O.
RC0/T1OS0/T1CKI	RC0	ST	CMOS	General purpose I/O.
	T1OS0	—	XTAL	Timer1 oscillator output.
	T1CKI	ST	—	Timer1 clock input.
RC1/T1OS1/CCP2	RC1	ST	CMOS	General purpose I/O.
	T1OS1	XTAL	—	Timer1 oscillator input.
	CCP2	ST	CMOS	Capture/Compare/PWM2.
RC2/P1A/CCP1	RC2	ST	CMOS	General purpose I/O.
	P1A	ST	CMOS	PWM output.
	CCP1	—	CMOS	Capture/Compare/PWM1.
RC3/SCK/SCL	RC3	ST	CMOS	General purpose I/O.
	SCK	ST	CMOS	SPI clock.
	SCL	ST	OD	I ² C™ clock.
RC4/SDI/SDA	RC4	ST	CMOS	General purpose I/O.
	SDI	ST	—	SPI data input.
	SDA	ST	OD	I ² C data input/output.
RC5/SDO	RC5	ST	CMOS	General purpose I/O.
	SDO	—	CMOS	SPI data output.
RC6/TX/CK	RC6	ST	CMOS	General purpose I/O.
	TX	—	CMOS	EUSART asynchronous transmit.
	CK	ST	CMOS	EUSART synchronous clock.
RC7/RX/DT	RC7	ST	CMOS	General purpose I/O.
	RX	ST	—	EUSART asynchronous input.
	DT	ST	CMOS	EUSART synchronous data.
RD0	RD0	TTL	CMOS	General purpose I/O.
RD1	RD1	TTL	CMOS	General purpose I/O.
RD2	RD2	TTL	CMOS	General purpose I/O.
RD3	RD3	TTL	CMOS	General purpose I/O.
RD4	RD4	TTL	CMOS	General purpose I/O.
RD5/P1B	RD5	TTL	CMOS	General purpose I/O.
	P1B	—	CMOS	PWM output.
RD6/P1C	RD6	TTL	CMOS	General purpose I/O.
	P1C	—	CMOS	PWM output.

RD7/P1D	RD7	TTL	CMOS	General purpose I/O.
	P1D	AN	—	PWM output.
RE0/AN5	RE0	TTL	CMOS	General purpose I/O.
	AN5	AN	—	A/D Channel 5.
RE1/AN6	RE1	TTL	CMOS	General purpose I/O.
	AN6	AN	—	A/D Channel 6.
RE2/AN7	RE2	TTL	CMOS	General purpose I/O.
	AN7	AN	—	A/D Channel 7.
RE3/MCLR/VPP	RE3	TTL	—	General purpose input.
	MCLR	ST	—	Master Clear with internal pull-up.
	VPP	HV	—	Programming voltage.
VSS	VSS	Power	—	Ground reference.
VDD	VDD	Power	—	Positive supply.

Legend: AN = Analog input or output CMOS = CMOS compatible input or output OD = Open Drain
 TTL = TTL compatible input ST = Schmitt Trigger input with CMOS levels
 HV = High Voltage XTAL = Crystal

7.3.4 Pin-Out Diagrams (40-pin PDIP)

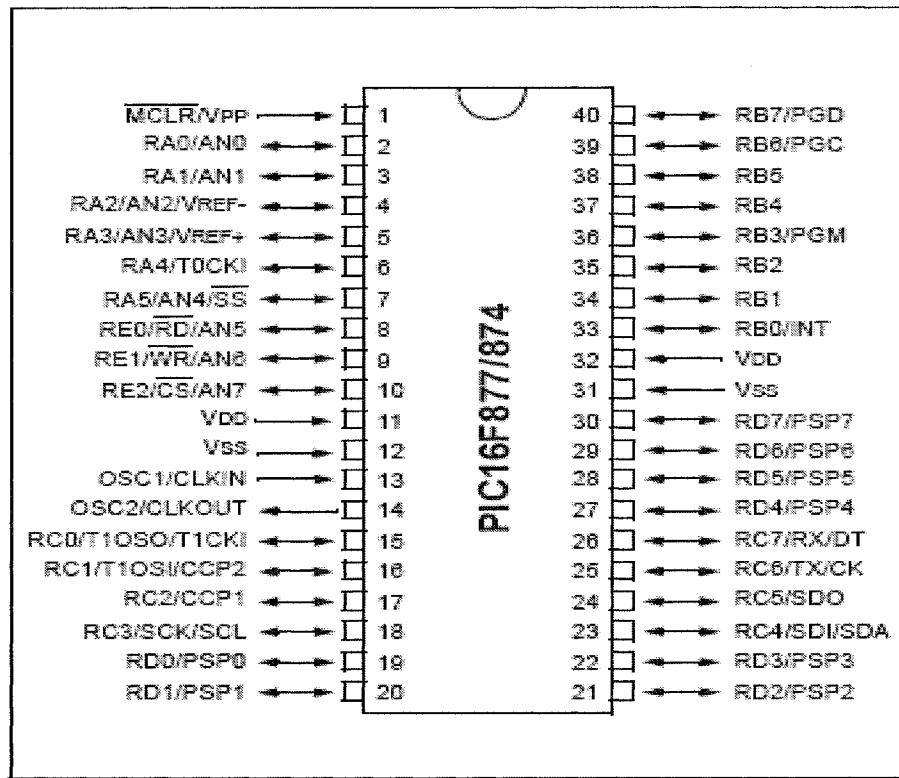


Figure 7.3-6a

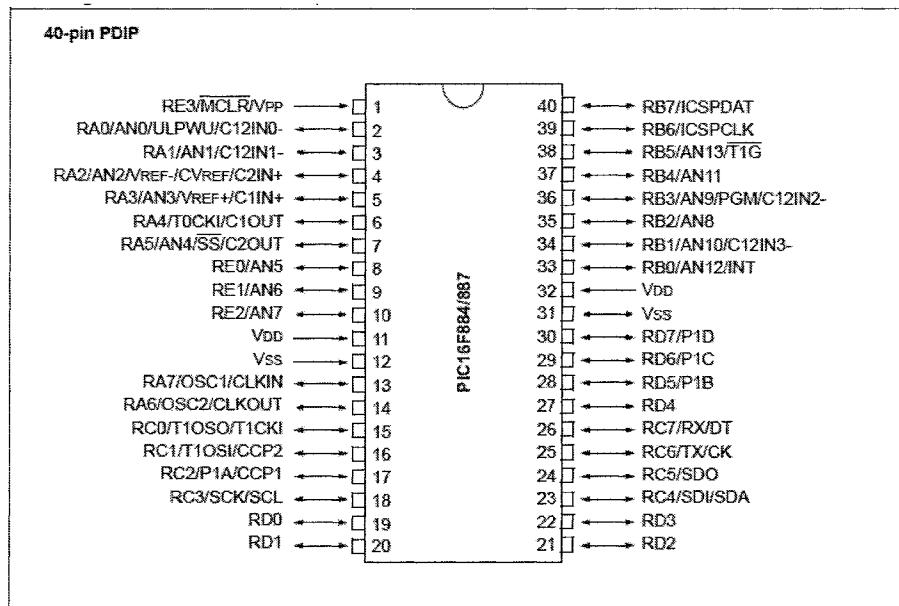
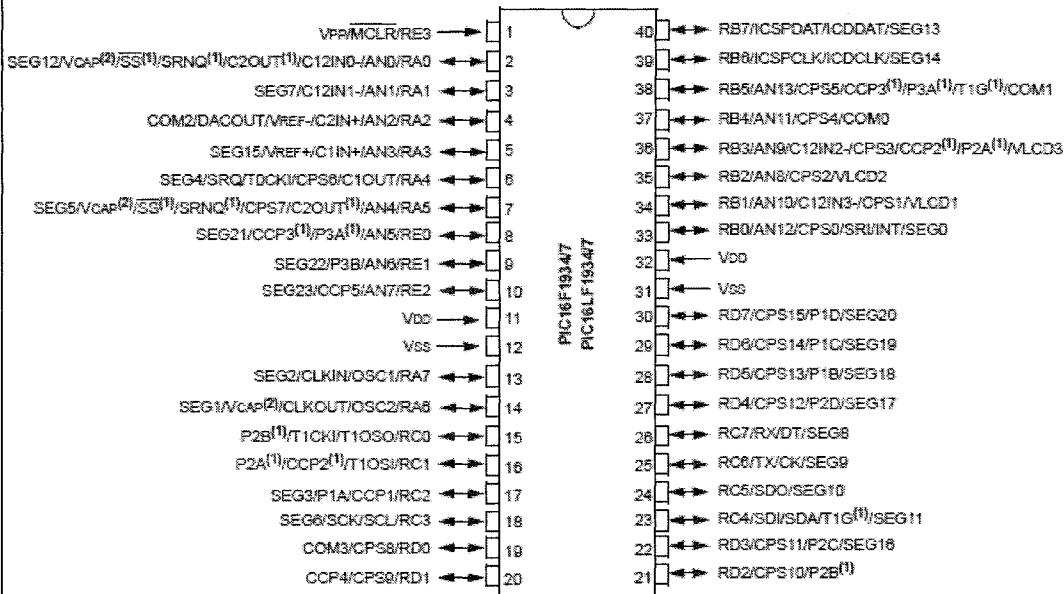


Figure 7.3-6b

48-Pin PDIP



Note 1: Pin function is selectable via the APFCON register.

2: PIC16F1934/7 devices only.

Figure 7.3-6c

40-Pin PDIP

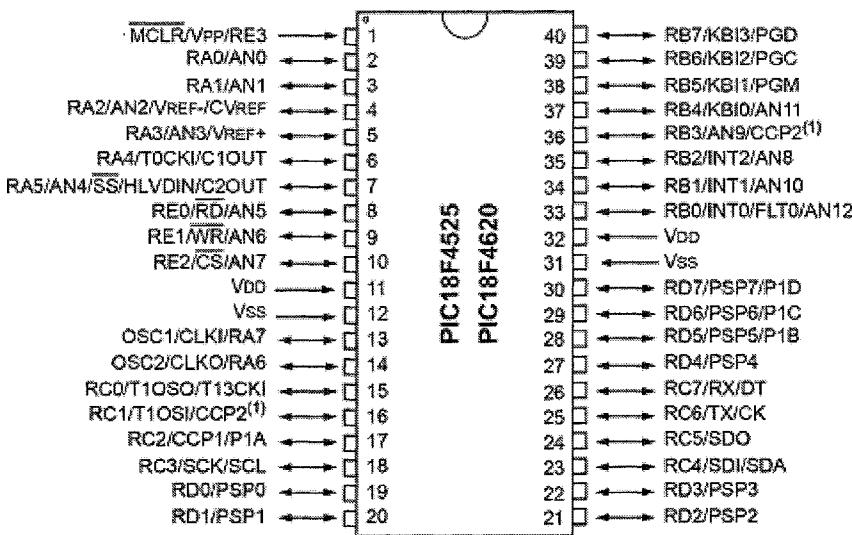


Figure 7.3-6d

7.3.5 Arithmetic and Logic Unit (ALU)

To operate on data memory and control status bits, an Arithmetic and Logic Unit (ALU) is required. The PICmicro chips contain an 8-bit ALU and an 8-bit working register. The ALU performs various arithmetic and Boolean functions between the data in the working register and any register file. The exact function to be performed is determined by several bits in the opcode. They include CLEAR, ADD, SUBtract, INCrement, DECrement, AND, OR, XOR, COMplement, Rotate Right, Rotate Left, SWAP nibbles, NOP, and a few other special cases. From one side of the ALU, a MUX (Multiple Xor) gate controls the data coming from the memory, and from the other side, the W register returns results of the previous instructions to be manipulated with the new data entry. The W register is an 8-bit working register used for ALU operations only, and is not an addressable register. The ALU also controls status bits (which are found in the STATUS register). The result of some instructions force status bits (like Carry C, or Zero Z bit) to a value depending on the state of the result. The results of the subsequent operations can be placed into the W register, or directly sent to the RAM locations or an I/O port or control register. The operation of the ALU and W register is illustrated in Figure 7.3-7.

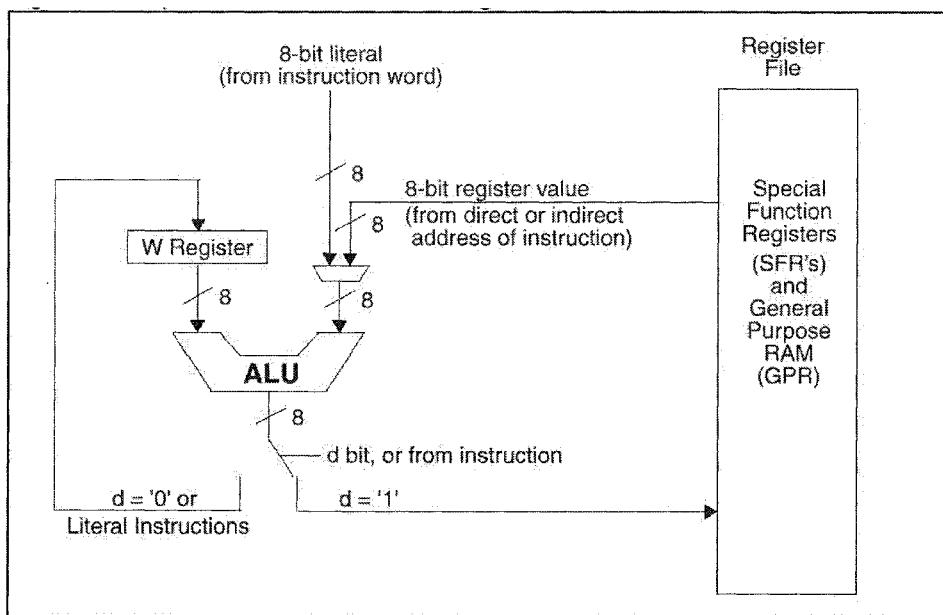


Figure 7.3-7

7.4 Memory Organization

There are three memory blocks in PICmicro chips: the FLASH Program Memory and RAM Data Memory, which have separate buses so that concurrent access can occur, and the EEPROM data memory. A brief description of these blocks is given below.

7.4.1 PIC16 Program Memory

The PIC16F877, PIC16F887, and PIC16F1937 devices have a 13-bit program counter capable of addressing 8k words (each word consists of 14 bits) of FLASH memory ($2^{13} = 8 \times 1024$). Figure 7.4-1 illustrates how different locations of the program memory are addressed by the program counter.

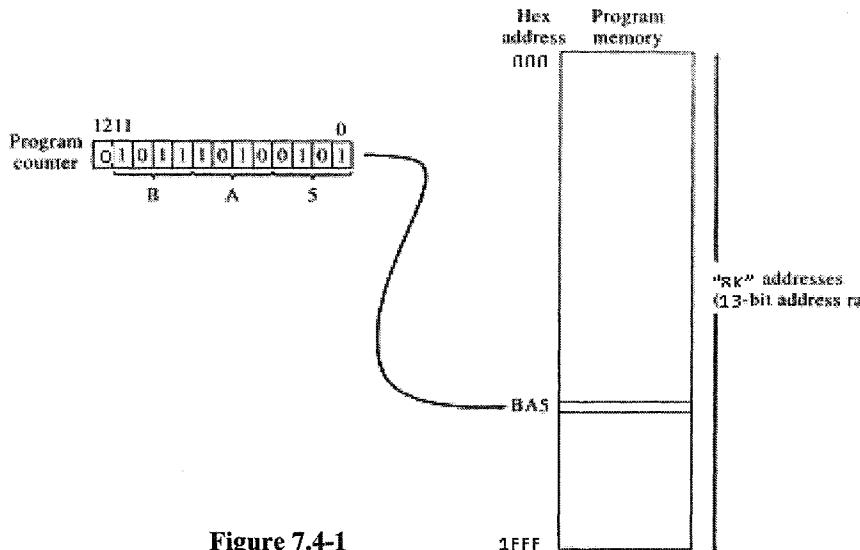


Figure 7.4-1

Two addresses in the program memory address space are treated in a special way by the CPU. When the CPU starts up from its reset state, its program counter is automatically cleared to zero. This is illustrated in figure on the right with the content of address H'0000' being a **goto Mainline** instruction (RESET vector). The second special address, H'0004', is automatically loaded into the program counter when an interrupt occurs (Interrupt vector). As shown in 7.4-2, a **goto IntService** instruction can be assigned to this address, to cause the CPU to jump to the beginning of the interrupt service routine, located elsewhere in the memory space. Further, in some occasions, the program code can be simplified somewhat if any tables that are created are assigned to addresses in the range H'0005'-H'00FF'. For most applications, these 250 locations provide more than enough room.

There is a special area in program memory, called *configuration word*, where some basic features of the chip are set. This word is mapped in program memory location H'2007'. This address is beyond the user program memory space, and can be accessed only during the programming. The configuration bits can be programmed (read as '0') or left un-programmed (read as '1') to select various device configurations. The erased or un-programmed value of the configuration word is 3FFFh. The following configurations can be assigned by the configuration word.

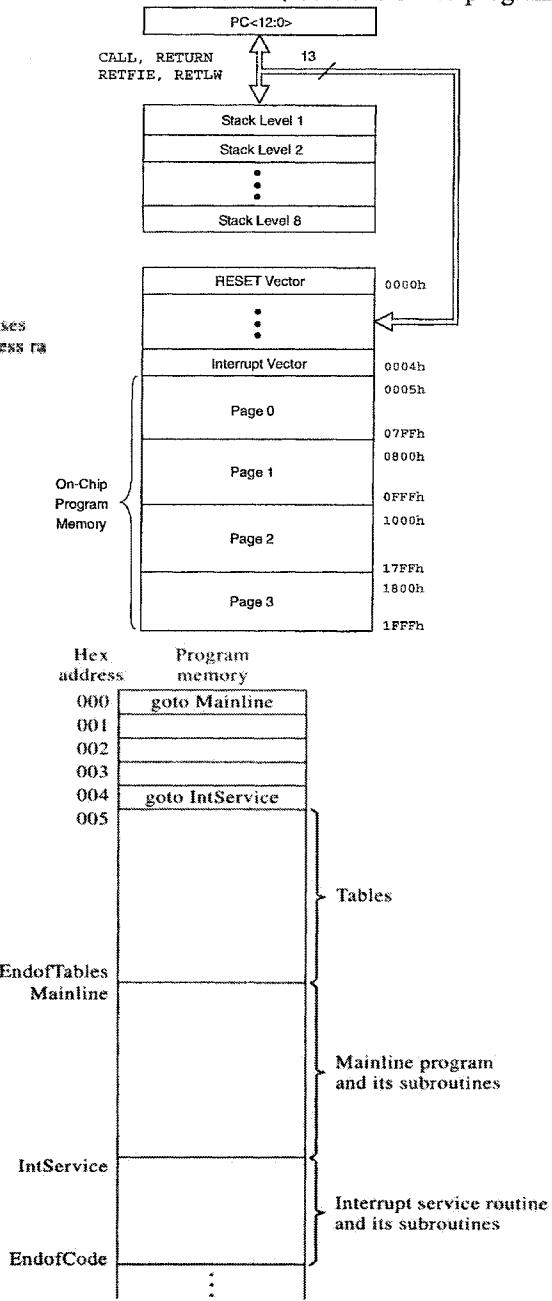


Figure 7.4-2

7.4.1.1 PIC16F877 Configuration Settings

13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP1	CP0	DEBUG	-	WRT	CPD	LVP	BODEN	CP1	CP0	PWRTE	WDTE	POSC1	POSC0

bit 13-12, **CP1:CP0:** FLASH Program Memory Code Protection bits

bit 11 **DEBUG:** In-Circuit Debugger Mode

 1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins

 0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger.

bit 10 **Unimplemented:** Read as '1'

bit 9 **WRT:** FLASH Program Memory Write Enable

 1 = Unprotected program memory may be written to by EECON control

 0 = Unprotected program memory may not be written to by EECON control

bit 8 **CPD:** Data EE Memory Code Protection

 1 = Code protection off

 0 = Data EEPROM memory code protected

bit 7 **LVP:** Low Voltage In-Circuit Serial Programming Enable bit

 1 = RB3/PGM pin has PGM function, low voltage programming enabled

 0 = RB3 is digital I/O, HV on MCLR must be used for programming

bit 6 **BODEN:** Brown-out Reset Enable bit (3)

 1 = BOR enabled

 0 = BOR disabled

bit 5-4 11 = Code protection off

 10 = 1F00h to 1FFFh code protected (PIC16F877, 876)

 10 = 0F00h to 0FFFh code protected (PIC16F874, 873)

 01 = 1000h to 1FFFh code protected (PIC16F877, 876)

 01 = 0800h to 0FFFh code protected (PIC16F874, 873)

 00 = 0000h to 1FFFh code protected (PIC16F877, 876)

 00 = 0000h to 0FFFh code protected (PIC16F874, 873)

bit 3 **PWRTE:** Power-up Timer Enable bit (3)

 1 = PWRT disabled

 0 = PWRT enabled

bit 2 **WDTE:** Watchdog Timer Enable bit

 1 = WDT enabled

 0 = WDT disabled

bit 1-0 **POSC1:POSC0:** Oscillator Selection bits

 11 = RC oscillator

 10 = HS oscillator

 01 = XT oscillator

 00 = LP oscillator

7.4.1.2 PIC16F887 Configuration Settings

CONFIG1: Configuration Word Register 1

—	—	DEBUG	LVP	FCMEN	IESO	BOREN1	BOREN0	CPD	CP	MCLRE	PWRTE	WDTE	FOSC2	FOSC1	FOSC0
bit 15								bit 8	bit 7						bit 0

- bit 15-14 **Unimplemented:** Read as '1'
- bit 13 **DEBUG:** In-Circuit Debugger Mode bit
 1 = In-Circuit Debugger disabled, RB6/ICSPCLK and RB7/ICSPDAT are general purpose I/O pins
 0 = In-Circuit Debugger enabled, RB6/ICSPCLK and RB7/ICSPDAT are dedicated to the debugger
- bit 12 **LVP:** Low Voltage Programming Enable bit
 1 = RB3/PGM pin has PGM function, low voltage programming enabled
 0 = RB3 pin is digital I/O, HV on MCLR must be used for programming
- bit 11 **FCMEN:** Fail-Safe Clock Monitor Enabled bit
 1 = Fail-Safe Clock Monitor is enabled
 0 = Fail-Safe Clock Monitor is disabled
- bit 10 **IESO:** Internal External Switchover bit
 1 = Internal/External Switchover mode is enabled
 0 = Internal/External Switchover mode is disabled
- bit 9-8 **BOREN<1:0>:** Brown-out Reset Selection bits(1)
 11 = BOR enabled
 10 = BOR enabled during operation and disabled in Sleep
 01 = BOR controlled by SBOREN bit of the PCON register
 00 = BOR disabled
- bit 7 **CPD:** Data Code Protection bit(2)
 1 = Data memory code protection is disabled
 0 = Data memory code protection is enabled
- bit 6 **CP:** Code Protection bit(3)
 1 = Program memory code protection is disabled
 0 = Program memory code protection is enabled
- bit 5 **MCLRE:** RE3/MCLR pin function select bit(4)
 1 = RE3/MCLR pin function is MCLR
 0 = RE3/MCLR pin function is digital input, MCLR internally tied to VDD
- bit 4 **PWRTE:** Power-up Timer Enable bit
 1 = PWRT disabled
 0 = PWRT enabled
- bit 3 **WDTE:** Watchdog Timer Enable bit
 1 = WDT enabled
 0 = WDT disabled and can be enabled by SWDTEN bit of the WDTCON register
- bit 2-0 **FOSC<2:0>:** Oscillator Selection bits
 111 = RC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, RC on RA7/OSC1/CLKIN
 110 = RCIO oscillator: I/O function on RA6/OSC2/CLKOUT pin, RC on RA7/OSC1/CLKIN
 101 = INTOSC oscillator: CLKOUT function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN
 100 = INTOSCO oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN
 011 = EC: I/O function on RA6/OSC2/CLKOUT pin, CLKIN on RA7/OSC1/CLKIN
 010 = HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN
 001 = XT oscillator: Crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN
 000 = LP oscillator: Low-power crystal on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN

- Note**
- 1: Enabling Brown-out Reset does not automatically enable Power-up Timer.
 - 2: The entire data EEPROM will be erased when the code protection is turned off.
 - 3: The entire program memory will be erased when the code protection is turned off.
 - 4: When MCLR is asserted in INTOSC or RC mode, the internal clock oscillator is disabled.

CONFIG2: Configuration Word Register 2

—	—	—	—	WRT1	WRT0	BOR4V	—	—	—	—	—	—	—
bit 15							bit 8	bit 7					bit 0

bit 15-11	Unimplemented: Read as '1'
bit 10-9	WRT<1:0>: Flash Program Memory Self Write Enable bits
	00 = 0000h to 0FFFh write protected, 1000h to 1FFFh may be modified by EECON control
	01 = 0000h to 07FFh write protected, 0800h to 1FFFh may be modified by EECON control
	10 = 0000h to 00FFh write protected, 0100h to 1FFFh may be modified by EECON control
	11 = Write protection off
bit 8	BOR4V: Brown-out Reset Selection bit
	0 = Brown-out Reset set to 2.1V
	1 = Brown-out Reset set to 4.0V
bit 7-0	Unimplemented: Read as '1'

7.4.1.3 PIC16F1937 Configuration Settings

CONFIG1: Configuration Word Register 1

FCMEN	IES0	CLKOUTEN	BOREN1	BOREN0	CPD	CP	MCLRE	PWRTE	WDTE1	WDTE0	FOSC2	FOSC1	FOSC0
bit 13							bit 7	bit 6					bit 0

bit 13	FCMEN: Fail-Safe Clock Monitor Enable bit
	1 = Fail-Safe Clock Monitor is enabled
	0 = Fail-Safe Clock Monitor is disabled
bit 12	IES0: Internal/External Switchover bit
	1 = Internal/External Switchover mode is enabled
	0 = Internal/External Switchover mode is disabled
bit 11	CLKOUTEN: Clock Out Enable bit
	1 = CLKOUT function is disabled. I/O or oscillator function on RA6/CLKOUT
	0 = CLKOUT function is enabled on RA6/CLKOUT
bit 10-9	BOREN<1:0>: Brown-out Reset Enable bits(1)
	11 = BOR enabled
	10 = BOR enabled during operation and disabled in Sleep
	01 = BOR controlled by SBORN bit of the PCON register
	00 = BOR disabled
bit 8	CPD: Data Code Protection bit(2)
	1 = Data memory code protection is disabled
	0 = Data memory code protection is enabled
bit 7	CP: Code Protection bit(3)
	1 = Program memory code protection is disabled
	0 = Program memory code protection is enabled
bit 6	MCLRE: RE3/MCLR/VPP Pin Function Select bit
	If LVP bit = 1: This bit is ignored.
	If LVP bit = 0: 1 = RE3/MCLR/VPP pin function is MCLR; Weak pull-up enabled. 0 = RE3/MCLR/VPP pin function is digital input; MCLR internally disabled; Weak pull-up under control of WPUE3 bit.
bit 5	PWRTE: Power-up Timer Enable bit(1)
	1 = PWRT disabled
	0 = PWRT enabled
bit 4-3	WDTE<1:0>: Watchdog Timer Enable bit
	11 = WDT enabled
	10 = WDT enabled while running and disabled in Sleep
	01 = WDT controlled by the SWDTEN bit in the WDTCON register
	00 = WDT disabled

- Note 1:** Enabling Brown-out Reset does not automatically enable Power-up Timer.
2: The entire data EEPROM will be erased when the code protection is turned off during an erase.
3: The entire program memory will be erased when the code protection is turned off.

CONFIG2: Configuration Word Register 2

LVP ⁽¹⁾	DEBUG ⁽³⁾	—	BORV	STVREN	PLLEN	—	—	VCAPEN<10> ⁽²⁾	—	—	—	WRT1	WRT0
bit 13						bit 7	bit 6						bit 0

bit 13	LVP: Low-Voltage Programming Enable bit(1) 1 = Low-voltage programming enabled 0 = High-voltage on MCLR/VPP must be used for programming
bit 12	DEBUG: In-Circuit Debugger Mode bit(3) 1 = In-Circuit Debugger disabled, RB6/ICSPCLK and RB7/ICSPDAT are general purpose I/O pins 0 = In-Circuit Debugger enabled, RB6/ICSPCLK and RB7/ICSPDAT are dedicated to the debugger
bit 11	Unimplemented: Read as '1'
bit 10	BORV: Brown-out Reset Voltage Selection bit 1 = Brown-out Reset voltage set to 1.9V 0 = Brown-out Reset voltage set to 2.5V
bit 9	STVREN: Stack Overflow/Underflow Reset Enable bit 1 = Stack Overflow or Underflow will cause a Reset 0 = Stack Overflow or Underflow will not cause a Reset
bit 8	PLLEN: PLL Enable bit 1 = 4xPLL enabled 0 = 4xPLL disabled
bit 7-6	Unimplemented: Read as '1'
bit 5-4	VCAPEN<1:0>: Voltage Regulator Capacitor Enable bits(2) 00 = VCAP functionality is enabled on RA0 01 = VCAP functionality is enabled on RA5 10 = VCAP functionality is enabled on RA6 11 = No capacitor on VCAP pin
bit 3-2	Unimplemented: Read as '1'
bit 1-0	WRT<1:0>: Flash Memory Self-Write Protection bits 4 kW Flash memory PIC16(L)F1934 only: 11 = Write protection off 10 = 000h to 1FFFh write-protected, 200h to FFFh may be modified by EECON control 01 = 000h to 7FFFh write-protected, 800h to FFFh may be modified by EECON control 00 = 000h to FFFFh write-protected, no addresses may be modified by EECON control 8 kW Flash memory (PIC16(L)F1936 and PIC16(L)F1937 only): 11 = Write protection off 10 = 000h to 1FFFh write-protected, 200h to 1FFFh may be modified by EECON control 01 = 000h to FFFFh write-protected, 1000h to 1FFFh may be modified by EECON control 00 = 000h to 1FFFh write-protected, no addresses may be modified by EECON control

Note 1: The LVP bit cannot be programmed to '0' when Programming mode is entered via LVP.

2: Reads as '11' on PIC16LF193X only.

3: The DEBUG bit in Configuration Word is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'.

In the next section, it will be shown how to set the configuration word by using a special directive. The FLASH program memory is readable and writable during normal operation over the entire V_{DD} range. A write operation causes an erase-then-write operation to take place on a specified word. Access to program memory allows for checksum calculation. The values written to program memory do not need to be valid instructions. Therefore, up to 14-bit numbers can be stored in memory for use as calibration parameters, serial numbers, etc. The flash program memory allows non-intrusive read access, but write operations cause the device to stop executing the instructions, until the write completes. Writing to any word of the program memory will cease the execution of instructions until the write is complete. The program memory cannot be accessed during write time. However, in the write duration, the oscillator continues to run, the peripherals continue to function, and interrupt events will be detected and essentially "queued" until the write is complete. When the write completes, the next instruction in the pipeline is executed and the branch to the interrupt vector will take place if the interrupt is enabled and occurred during the write.

The **Mainline** program begins immediately following the tables. A microcontroller mainline program typically has the structure

```

Mainline
    call  Initial      ;Initialize everything
MainLoop
    call Task1        ;Deal with Task1
    call Task2        ;Deal with Task2
    .
    .
    call  LoopTime    ;Force looptime to a fixed value
    goto MainLoop     ;Repeat

```

The mainline program begins execution when the PIC comes out of reset. It continues running until one of the PIC's interrupt sources request service. At that point the execution of the mainline code is temporarily suspended. The CPU begins the execution of the interrupt service routine by automatically loading the program counter with H'0004'. At the completion of the interrupt service routine, the CPU returns to where it left off in the mainline program. As shown in Figure 7.4-3, bits 10,...,0 of the **call** instruction are loaded into the program counter. At the same time, bits 4 and 3 of a special register called **PCLATH** ("program counter latch") are loaded into bits 12 and 11 of the program counter. As long as the program memory is less than 2048 (i.e., 2K) words, bits 4 and 3 of **PCLATH** can be left initialized to H'00', and then the 11 address bits in the **call** instruction will uniquely identify the starting address of any subroutine located up to address H'7FF'. For programs larger than this, it is necessary to ensure that bits 3 and 4 of **PCLATH** are set or cleared appropriately each time a subroutine is called. The **goto** instruction, which also has an 11-bit address field, requires an identical treatment.

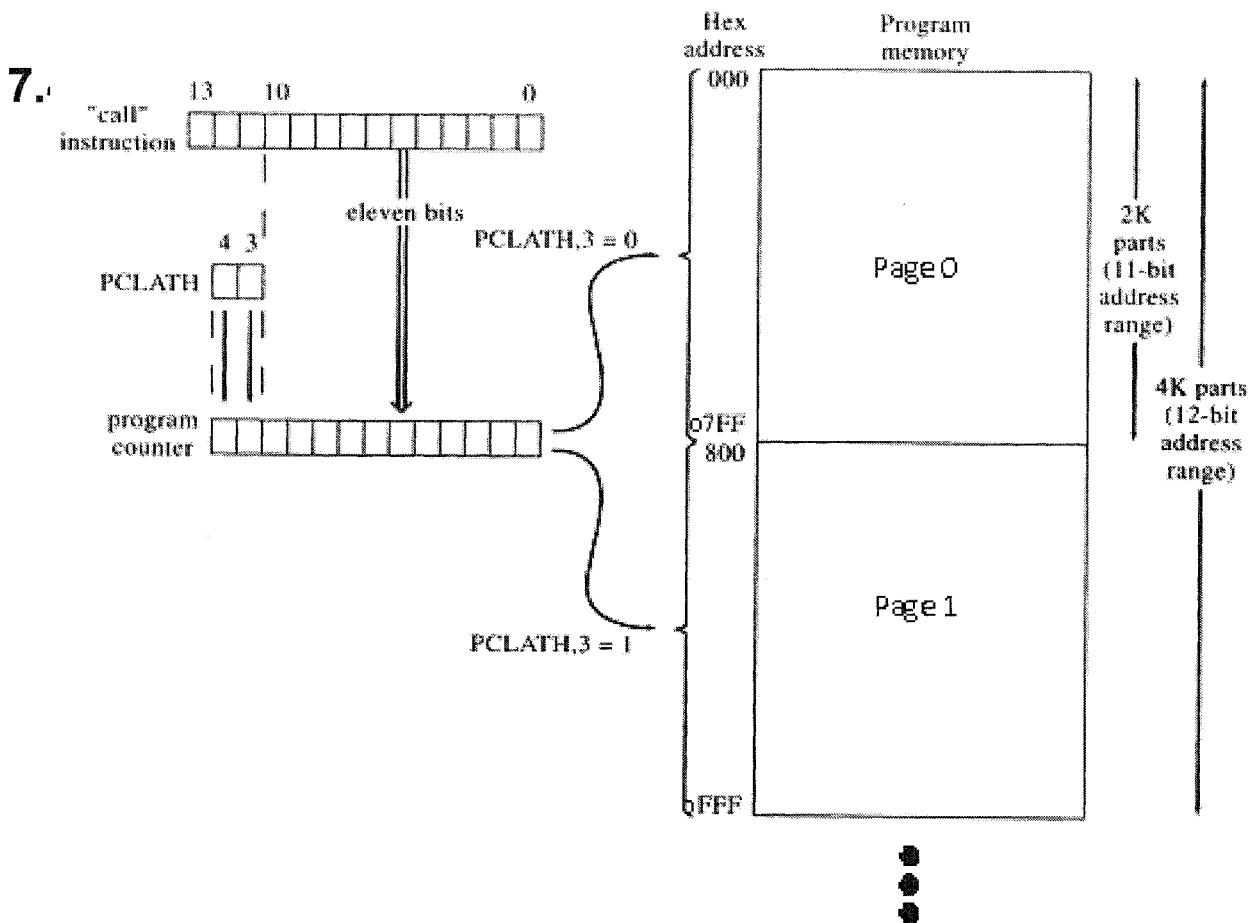


Figure 7.4-3

PIC18 Program Memory

The PIC18F4620 device has a 21-bit program counter capable of addressing 2 Mbytes of FLASH memory ($2^{21} = 2,097,152$). Figure 7.4-4 and Figure 7.4-5 illustrate how different locations of the program memory are addressed by the program counter.

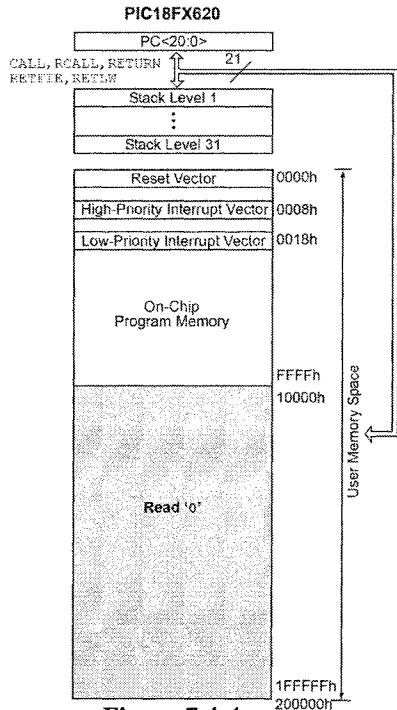


Figure 7.4-4

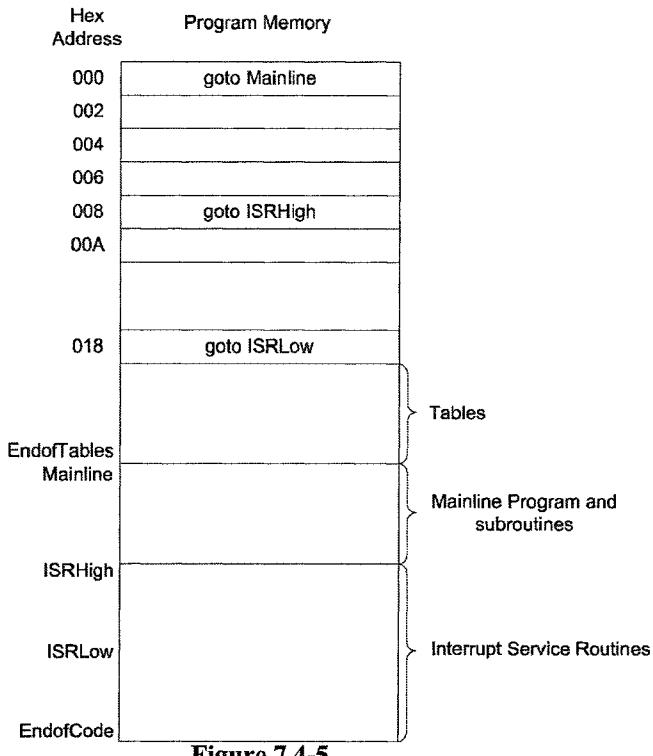


Figure 7.4-5

There are three special program memory addresses – the reset vector and 2 interrupt vectors. The reset vector is at program memory H'0000' and is the program entry point of the CPU. The second and third special addresses are Interrupt vectors. PIC18F4620 has two interrupt priority levels, a high priority and low priority interrupt. Each interrupt source can be assigned an interrupt priority level. High priority interrupts will interrupt any low priority routines in progress, allowing fast response time and flexibility. When a high priority interrupt occurs, the address H'0008' is automatically loaded into the program counter in order to service the interrupt. Similarly, when a low priority interrupt occurs, H'0018' is loaded into the program counter. As shown in Figure 7.4-4, a **goto ISRHigh** instruction can be assigned to high priority interrupt address to cause the CPU to jump to the beginning of the interrupt service routine, located elsewhere in the program memory. The same can be done for the low priority interrupt vector.

Same as PIC16, the FLASH program memory is readable and writable during normal operation over the entire V_{DD} range. The values written to program memory do not need to be valid instructions. Since PIC18 has a 16bit wide FLASH memory word, up to 2 bytes of data can be stored in each program memory word. This allows data such as tables to be stored in FLASH memory more efficiently. The flash program memory allows non-intrusive read access, but writing or erasing the program memory can only be done in blocks of 64 bytes. Writing to the FLASH memory during runtime is not recommended for most users as the process may corrupt the existing program.

A typical PIC18 program follows the same structure as the PIC16, with a few key differences. The **call** and **goto** instructions are two-word instructions that contain the full address of the target program memory location. These instructions are capable of addressing up to 2Mbytes of memory locations. This eliminates the need for manually setting the **PCLATH** register to switch pages when the program is longer than 2Kwords, as done in the PIC16. To make up for the slight loss in efficiency resulting from the 2-word instructions, the PIC18 also includes an **rcall** (relative call) instruction. This one-word instruction is able to call any program memory location within 1Kwords from the current program memory location. For calling neighboring functions, **rcall** will use less program memory than **call**.

7.4.2.1 Fast Call/Return Mode

Typically, it is good programming practice to save the states of several key registers when entering a subroutine and loading these registers back to their respective states upon exiting the subroutine. These key registers may include the W (working) register, the STATUS register and the BSR (Bank Select Register). A special opcode, denoted “s” can be included in the **call** and **return** instructions to enable the Fast Call/Return mode. In this mode, the contents of the W, STATUS and BSR registers are immediately loaded into 3 “shadow registers”, WS, STATUS, and BSRS. Upon returning to the main routine, the contents of the WS, STATUS, and BSRS shadow registers can be reloaded back into their respective registers. Setting “s” to 1 enables the Fast Call/Return mode. The default state of “s” is 0.

Example:

```
call    SUB1, 1      ;enable Fast Call/Return Mode
return  1           ;loads the values in the shadow registers back into W, STATUS and BSR
```

7.4.2.2 Configuration Settings

The configuration words of the PIC18 are mapped in program memory locations starting from H'300000'. Due to their wider range of features, PIC18s have 7 configuration words. However, not all bits in the configuration words are used. The following configurations can be assigned to the PIC18.

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0	00-- 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	PWRREN	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX	1--- -011
300006h	CONFIG4L	DEBUG	XINST	—	—	—	LVP	—	STVREN	10-- -1-1
300008h	CONFIG5L	—	—	—	—	CP3 ⁽¹⁾	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3 ⁽¹⁾	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3 ⁽¹⁾	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEVO	REV4	REV3	REV2	REV1	REVO	xxxx xxxx ⁽²⁾
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 1100 ⁽²⁾

Legend: x = unknown, u = unchanged, - = unimplemented. Shaded cells are unimplemented, read as '0'.

Note 1: Unimplemented in PIC18FX525 devices; maintain this bit set.

2: See Register 23-12 and Register 23-13 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

IESO: Internal/External Oscillator Switchover bit

- 1 = Oscillator Switchover mode enabled
- 0 = Oscillator Switchover mode disabled

FCMEN: Fail-Safe Clock Monitor Enable bit

- 1 = Fail-Safe Clock Monitor enabled
- 0 = Fail-Safe Clock Monitor disabled

FOSC3:FOSC0: Oscillator Selection bits

- 11xx = External RC oscillator, CLKO function on RA6
- 101x = External RC oscillator, CLKO function on RA6
- 1001 = Internal oscillator block, CLKO function on RA6, port function on RA7
- 1000 = Internal oscillator block, port function on RA6 and RA7
- 0111 = External RC oscillator, port function on RA6
- 0110 = HS oscillator, PLL enabled (Clock Frequency = 4 x FOSC1)
- 0101 = EC oscillator, port function on RA6

0100 = EC oscillator, CLK0 function on RA6
 0011 = External RC oscillator, CLK0 function on RA6
 0010 = HS oscillator
 0001 = XT oscillator
 0000 = LP oscillator

BORV1:BORV0: Brown-out Reset Voltage bits(1)

11 = Minimum setting
 ...
 00 = Maximum setting

BOREN1:BOREN0: Brown-out Reset Enable bits(2)

11 = Brown-out Reset enabled in hardware only (SBOREN is disabled)
 10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)
 01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled)
 00 = Brown-out Reset disabled in hardware and software

PWRDEN: Power-up Timer Enable bit(2)

1 = PWRT disabled
 0 = PWRT enabled

WDTPS3:WDTPS0: Watchdog Timer Postscale Select bits

1111 = 1:32,768	1011 = 1:2,048	0111 = 1:128	0011 = 1:8
1110 = 1:16,384	1010 = 1:1,024	0110 = 1:64	0010 = 1:4
1101 = 1:8,192	1001 = 1:512	0101 = 1:32	0001 = 1:2
1100 = 1:4,096	1000 = 1:256	0100 = 1:16	0000 = 1:1

WDTEN: Watchdog Timer Enable bit

1 = WDT enabled
 0 = WDT disabled (control is placed on the SWDTEN bit)

MCLRE: MCLR Pin Enable bit

1 = MCLR pin enabled; RE3 input pin disabled
 0 = RE3 input pin enabled; MCLR disabled

LPT1OSC: Low-Power Timer1 Oscillator Enable bit

1 = Timer1 configured for low-power operation
 0 = Timer1 configured for higher power operation

PBADEN: PORTB A/D Enable bit

(Affects ADCON1 Reset state. ADCON1 controls PORTB<4:0> pin configuration.)

1 = PORTB<4:0> pins are configured as analog input channels on Reset
0 = PORTB<4:0> pins are configured as digital I/O on Reset

CCP2MX: CCP2 MUX bit

1 = CCP2 input/output is multiplexed with RC1
 0 = CCP2 input/output is multiplexed with RB3

DEBUG: Background Debugger Enable bit

1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins
 0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug

XINST: Extended Instruction Set Enable bit

1 = Instruction set extension and Indexed Addressing mode enabled
 0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)

LVP: Single-Supply ICSP™ Enable bit

1 = Single-Supply ICSP enabled
 0 = Single-Supply ICSP disabled

STVREN: Stack Full/Underflow Reset Enable bit

1 = Stack full/underflow will cause Reset
 0 = Stack full/underflow will not cause Reset

CP3: Code Protection bit(1)

1 = Block 3 (006000-007FFFh) not code-protected
 0 = Block 3 (006000-007FFFh) code-protected

CP2: Code Protection bit

1 = Block 2 (004000-005FFFh) not code-protected
 0 = Block 2 (004000-005FFFh) code-protected

CP1: Code Protection bit

1 = Block 1 (002000-003FFFh) not code-protected
 0 = Block 1 (002000-003FFFh) code-protected

CP0: Code Protection bit

1 = Block 0 (000800-001FFFh) not code-protected
 0 = Block 0 (000800-001FFFh) code-protected

CPD: Data EEPROM Code Protection bit

1 = Data EEPROM not code-protected
 0 = Data EEPROM code-protected

CPB: Boot Block Code Protection bit

1 = Boot block (000000-0007FFh) not code-protected
0 = Boot block (000000-0007FFh) code-protected

WRT3: Write Protection bit

1 = Block 3 (006000-007FFFh) not write-protected
0 = Block 3 (006000-007FFFh) write-protected

WRT2: Write Protection bit

1 = Block 2 (004000-005FFFh) not write-protected
0 = Block 2 (004000-005FFFh) write-protected

WRT1: Write Protection bit

1 = Block 1 (002000-003FFFh) not write-protected
0 = Block 1 (002000-003FFFh) write-protected

WRT0: Write Protection bit

1 = Block 0 (000800-001FFFh) not write-protected
0 = Block 0 (000800-001FFFh) write-protected

WRTD: Data EEPROM Write Protection bit

1 = Data EEPROM not write-protected
0 = Data EEPROM write-protected

WRTB: Boot Block Write Protection bit

1 = Boot block (000000-0007FFh) not write-protected
0 = Boot block (000000-0007FFh) write-protected

WRTC: Configuration Register Write Protection bit

1 = Configuration registers (300000-3000FFh) not write-protected
0 = Configuration registers (300000-3000FFh) write-protected

EBTR3: Table Read Protection bit

1 = Block 3 (006000-007FFFh) not protected from table reads executed in other blocks
0 = Block 3 (006000-007FFFh) protected from table reads executed in other blocks

EBTR2: Table Read Protection bit

1 = Block 2 (004000-005FFFh) not protected from table reads executed in other blocks
0 = Block 2 (004000-005FFFh) protected from table reads executed in other blocks

EBTR1: Table Read Protection bit

1 = Block 1 (002000-003FFFh) not protected from table reads executed in other blocks
0 = Block 1 (002000-003FFFh) protected from table reads executed in other blocks

EBTR0: Table Read Protection bit

1 = Block 0 (000800-001FFFh) not protected from table reads executed in other blocks
0 = Block 0 (000800-001FFFh) protected from table reads executed in other blocks

EBTRB: Boot Block Table Read Protection bit

1 = Boot block (000000-0007FFh) not protected from table reads executed in other blocks
0 = Boot block (000000-0007FFh) protected from table reads executed in other blocks

The device ID memory locations store the part number of the PIC18 device. These locations are Read-Only and cannot be modified by the user.

7.4.3 PIC16 Data Memory

7.4.3.1 PIC16F877 Memory Map

File Address	File Address	File Address	File Address
Indirect addr. ^(*)	00h	Indirect addr. ^(*)	100h
TMR0	01h	OPTION_REG	101h
PCL	02h	PCL	102h
STATUS	03h	STATUS	103h
FSR	04h	FSR	104h
PORTA	05h	TRISA	105h
PORTB	06h	TRISB	106h
PORTC	07h	TRISC	107h
PORTD ⁽¹⁾	08h	TRISD ⁽¹⁾	108h
PORTE ⁽¹⁾	09h	TRISE ⁽¹⁾	109h
PCLATH	0Ah	PCLATH	10Ah
INTCON	0Bh	INTCON	10Bh
PIR1	0Ch	PIE1	10Ch
PIR2	0Dh	PIE2	10Dh
TMR1L	0Eh	PCON	10Eh
TMR1H	0Fh		10Fh
T1CON	10h		110h
TMR2	11h	SSPCON2	111h
T2CON	12h	PR2	112h
SSPBUF	13h	SSPADD	113h
SSPCON	14h	SSPSTAT	114h
CCPR1L	15h		115h
CCPR1H	16h		116h
CCP1ICON	17h		117h
RCSTA	18h	TXSTA	118h
TXREG	19h	SPBRG	119h
RCREG	1Ah		11Ah
CCPR2L	1Bh		11Bh
CCPR2H	1Ch		11Ch
CCP2CON	1Dh		11Dh
ADRESH	1Eh	ADRESL	11Eh
ADCON0	1Fh	ADCON1	11Fh
	20h		120h
General Purpose Register 96 Bytes	7Fh	General Purpose Register 80 Bytes	General Purpose Register 80 Bytes
		EFh	16Fh
		F0h	170h
		FFh	17Fh
			Bank 3
			1A0h
			General Purpose Register 80 Bytes
			1E1h
			1F0h
			1FFh
			Bank 3

 Unimplemented data memory locations, read as '0'.

 * Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
 2: These registers are reserved, maintain these registers clear.

Figure 7.4-6a

7.4.3.2 PIC16F887 Memory Map

File Address	File Address	File Address	File Address
Indirect addr. (1)	00h	Indirect addr. (1)	80h
TMR0	01h	OPTION_REG	81h
PCL	02h	PCL	82h
STATUS	03h	STATUS	83h
FSR	04h	FSR	84h
PORTA	05h	TRISA	85h
PORTB	06h	TRISB	86h
PORTC	07h	TRISC	87h
PORTD ⁽²⁾	08h	TRISD ⁽²⁾	88h
PORTE	09h	TRISE	89h
PCLATH	0Ah	PCLATH	8Ah
INTCON	0Bh	INTCON	8Bh
PIR1	0Ch	PIE1	8Ch
PIR2	0Dh	PIE2	8Dh
TMR1L	0Eh	PCON	8Eh
TMR1H	0Fh	OSCCON	8Fh
T1CON	10h	OSCTUNE	90h
TMR2	11h	SSPICON2	91h
T2CON	12h	PR2	92h
SSPBUF	13h	SSPADD	93h
SSPOON	14h	SSPSTAT	94h
CCPR1L	15h	WPUB	95h
CCPR1H	16h	IOCB	96h
CCP1ICON	17h	VRCON	97h
RCSTA	18h	TXSTA	98h
TXREG	19h	SPBRG	99h
RCREG	1Ah	SPBRGH	9Ah
CCPR2L	1Bh	PWM1CON	9Bh
CCPR2H	1Ch	ECCPAS	9Ch
CCP2ICON	1Dh	PSTRCON	9Dh
ADRESH	1Eh	ADRESL	9Eh
ADCON0	1Fh	ADCON1	9Fh
General Purpose Registers	20h	General Purpose Registers	A0h
96 Bytes	3Fh	80 Bytes	8Fh
	40h		80 Bytes
	6Fh	accesses	F0h
	70h	70h-7Fh	accesses
	7Fh		70h-7Fh
Bank 0	Bank 1	Bank 2	Bank 3
■ Unimplemented data memory locations, read as '0'.			
Note: 1: Not a physical register.			
2: PIC16F887 only.			

Figure 7.4-6b

The data memory is partitioned into four banks, which contain the register files. The term *register file* is PIC terminology used to denote the locations that an instruction can access via an address. The register file consists of two components: General Purpose Registers (GPR) and Special Function Registers (SFR).

The general-purpose register file is another name for the microcontroller's RAM. Data can be written to each 8-bit location, updated, and retrieved any number of times. The special-purpose register file contains *input* and *output ports* as well as the control registers used to establish each bit of a port as either an input or an output. It contains registers that provide the *data input* and *data output* to the variety of resources on the chip, such as the timers, the serial ports, and the analog-to-digital converter. It has registers that contain *control* bits for selecting the mode of operation of a chip resource as well as enabling or disabling its operation. It has registers containing status bits, which denote the state of one of these chip resources (e.g., "serial data transfer complete").

Banks in PIC16F877/887:

The register file structure is illustrated in Figure 7.4-6a and 7.4-6b, with addresses that span the 8-bit range from H'00' to H'1FF'. As illustrated in the figure, the register file is divided into four banks. These banks can be assigned (addressed) by a special register called STATUS. In the STATUS register, two bits control the bank (Page) selection, bits 6 and 5. These STATUS bits are referred to by their allocated names, i.e., RP1 (bit 6) and RP0 (bit 5). Their settings are as follows:

Bank	RP1	RP0
BANK0	0	0
BANK1	0	1
BANK2	1	0
BANK3	1	1

Each bank extends up to 80h (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers.

Banks in PIC16F1937:

The enhanced architecture of PIC16F1937 allows for a much larger data memory, resulting in 32 banks as opposed to 8 for the previously mentioned PICs. In order to access each bank without indirect accessing, the user has to load the number of the bank into PIC16F1937's Bank Selection Register (**BSR**). For instance, clearing **BSR** gives access to Bank0, while loading the decimal number d'5' gives access to Bank5. Due to this additional register, the **STATUS** register of PIC16F1937 does NOT have the **RP0** and **RP1** bits.

As mentioned previously, the General Purpose Registers are those non-dedicated physical blocks that can hold the data and hence be used for any operation. These blocks can be accessed either directly (by calling the address), or indirectly through the File Select Register (FSR). The Special Function Registers are those registers used by the CPU and peripheral modules for controlling the desired operation of the device. They can be classified into two sets of peripheral and core (CPU) registers. Some of the important SFRs are listed below. For further details refer to the PIC manual.

- **W** : working register used by many instructions as the source of an operand or the destination of an instruction.
- **INDF** : used for indirect addressing;
- **TMR0** : the counter/timer register;
- **OPTION_REG** : controls timer/counter operation and other functions;
- **PCL** : only the low-order byte of the program counter (PC);
- **STATUS** : power and sleep status, plus register bank selection for PIC16F877/887;
- **FSR** : used for indirect memory addressing;
- **PORTA, PORTB, PORTC, PORTD, PORTE** : the I/O ports;
- **TRISA, TRISB, TRISC, TRISD, TRISE** : the I/O port control registers. They set individual I/O pins for input or output operation.
- **INTCON** : the interrupt control register;
- **PCLATH** : holds the high-order bits of the program counter (PC);
- **EEDATA, EECON** : used for access to on-chip EEPROM data memory;
- **ADCON0** : used for A/D converters.

The **STATUS** register, in particular, has some important bits that must be noted carefully. The lowest bit is the **C**, or *carry*, bit. When two 8-bit operands are added together, a 9-bit result can occur. The ninth bit is placed in the carry bit. For example,

$$\begin{array}{rcl} B'10010001' & = & H'91' \\ \underline{B'10010001'} & = & \underline{H'91'} \\ B'100100010' & = & H'122' \end{array}$$

Here, the 8-bit result is $H'22'$ and the ninth bit produces $C = 1$. If the ninth bit had been zero, then $C = 0$ would have resulted. As a result of a subtract operation, the **C** bit acts as a *borrow* bit and is cleared to zero if a borrow occurs, and is set otherwise. For example,

$$\begin{array}{r} B'10010001' \\ - B'10010001' \\ \hline \end{array}$$

is implemented by forming the *two's complement* of the subtrahend

$$B'100000000' - B'10010001' = B'01101111'$$

and then adding this to the minuend:

$$\begin{array}{r} B'10010001' \\ \underline{B'01101111'} \\ \hline B'100000000' \end{array}$$

The one in the ninth bit is stored in the carry bit and indicates that the result of subtracting the original operands ($B'10010001'$ minus $B'10010001'$) did *not* produce a borrow.

The next bit (bit 1) of the **STATUS** register is **DC**, or *digit carry*, bit. This bit signals that a carry from the lower 4 bits occurred during an 8-bit addition. For example,

$$\begin{array}{r} B'0011 1000' \\ \underline{B'0011 1000'} \\ \hline B'0 0111 0000' \end{array}$$

Here, **DC** = 1 as a result of the carry from the bit 3 to the bit 4 position. This digit carry bit is useful when adding binary-coded-decimal (BCD) encoded numbers in which each 8-bit byte contains two 4-bit binary-coded-decimal digits. If the preceding example represented the sum of the BCD-encoded decimal number 38 to itself, then the **DC** = 1 result could be used as a signal to add 6 ($B'00000110'$) to the binary sum to convert it to the correct BCD result. The subtraction of two BCD-encoded numbers produces **DC** = 0 if a borrow results from the bit 3 to the bit 4 position. Otherwise **DC** = 1.

Bit 2 of the **STATUS** register is the **Z**, or *zero*, bit, which is affected by the execution of many (but not all) arithmetic and logic instructions. Before testing the **Z** bit following an instruction, it should be ascertained whether the instruction is indeed one of the group that affects the **Z** bit. For example, the instruction **decf** can be used to decrement a variable in RAM, setting the **Z** bit if the result is zero and clearing it otherwise. In contrast, the instruction **decfsz** can be used to decrement the same variable in RAM and skip over the next instruction if the result is zero but leave the **Z** bit unchanged.

The reset status bits, **NOT_TO** and **NOT_PD**, are bits 4 and 3 of the **STATUS** register, respectively. They are used in conjunction with the PIC's *sleep mode*. The microcontroller can put itself to sleep to save power during intervals when it has nothing to do. It can be awakened by the occurrence of any of three kinds of events. Upon wakeup, the CPU can check these two reset status bits to determine which kind of event awakened it and then respond accordingly.

For PIC16F877 and PIC16F887:

Bits 5 (**RP0**) and 6 (**RP1**) of the **STATUS** register are bank select bits, which are used in conjunction with the direct addressing mode to be discussed in the sequel.

Bits 7 **STATUS** register is used for indirect addressing, as will be explained in the next section.

7.4.3.3 Indirect Addressing

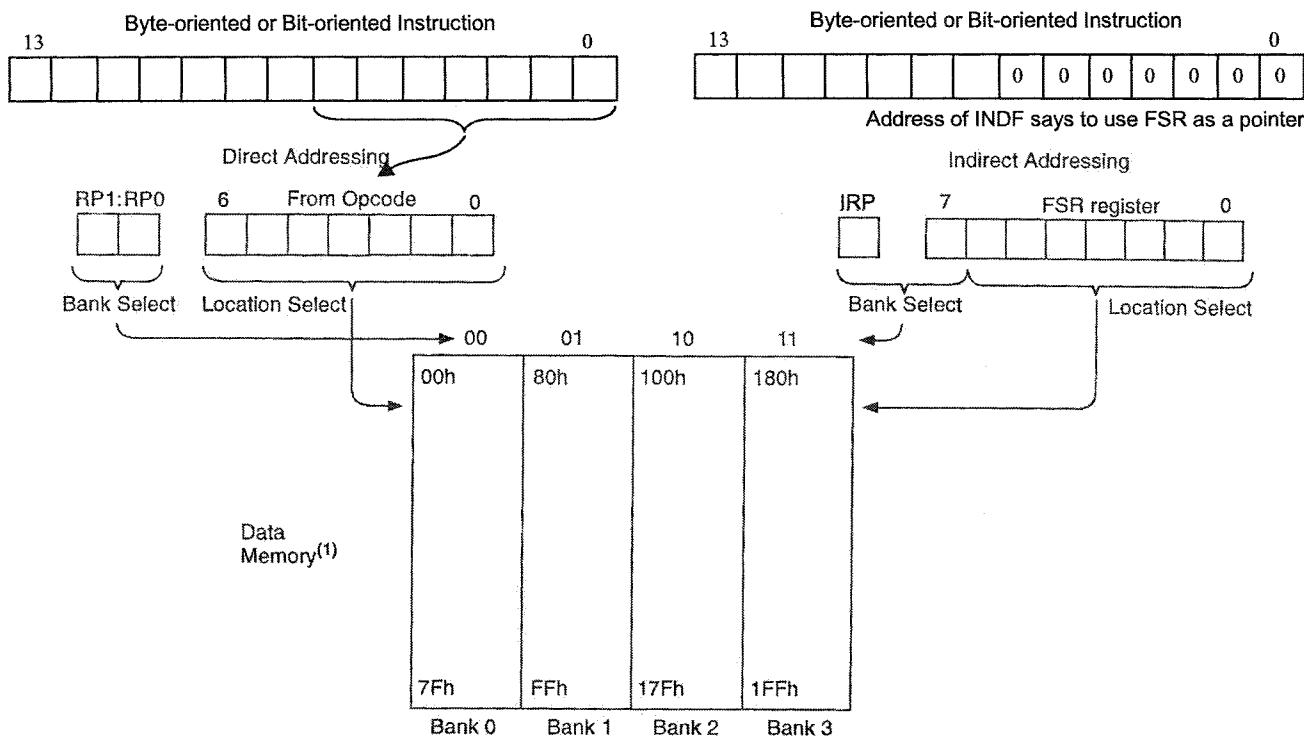


Figure 7.4-7

As previously mentioned, note that data memory is bank switched. If you need to access locations in more than one bank, you can either use the **RP0/RP1** bits in the **STATUS** register (or **BSR** register) or use the **FSR** register for indirect addressing. Every instruction that can employ the direct addressing mode can, as an alternative, employ the *indirect addressing* mode. In this alternative mode, the full 8-bit register file address is first written into **FSR**, a special-purpose register that serves as an address pointer to *any* address throughout the entire register file. A subsequent direct access of **INDF** (at location 0) will actually access the register file using the content of **FSR** as a pointer to the desired location of the operand. Please note that **INDF** is not a physical register.

Indirect Addressing in PIC16F877/887:

To use indirect addressing, the desired 8-bit address must first be written into **FSR** using direct addressing. To make this possible regardless of the value of the register bank select bits, **RP0** and **RP1**, the **FSR** register is accessible at each bank (note locations H'04', H'84', H'104' and H'184'). Consequently, a write to either H'04' or H'84' or H'104' or H'184' will write into the **FSR** register. Several other key registers, including **INDF**, are accessible at all of four banks, which allows these key registers to be accessed with direct addressing regardless of the value of the **RP0** and **RP1** bits. In addition, depending on whether the requested files registers are in Bank0,1 or Bank2,3, bit **IRP** of **STATUS** register (which is also accessible from all banks) must be cleared or set, respectively. Figure 7.4-7 shows the two methods of addressing.

	CLRF	STATUS,IRP
	MOVLW	0x20 ;initialize pointer
	MOVWF	FSR ;to RAM
NEXT	CLRF	INDF ;clear INDF register
	INCF	FSR,F ;inc pointer
	BTFS	FSR,4 ;all done?
	GOTO	NEXT ;no clear next
CONTINUE		;yes continue

Indirect Addressing in PIC16F1937:

Again due to its enhanced architecture, PIC16F1937 offers more extensive support for indirect referencing. For this PIC, there are 2 **FSR** registers, as well as 2 **INDF** registers. Also, the FSR registers are 16-bit wide, consisting of **FSR0/1L** and **FSR0/1H**. This allows the FSRs to be able to access additional memory besides Special Function Registers and General Purpose Registers.

Moreover, PIC16F1937 includes 3 additional instructions that improve the user's use of indirect addressing. Instructions will be discussed later, but for now, these 3 instructions are:

- **ADDSR** – Adds a literal between -32 and 31 to the specified FSR.
- **MOVIW** – Move a value from the specified INDF register into W.
- **MOVWI** – Move a value from W into the specified INDF register.

7.4.4 PIC18 Data Memory

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2 ⁽¹⁾	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2 ⁽¹⁾	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2 ⁽¹⁾	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2 ⁽¹⁾	FBCh	CCPR2H	F9Ch	__(2)
FFBh	PCLATU	FDBh	PLUSW2 ⁽¹⁾	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	__(2)
FF9h	PCL	FD9h	FSR2L	FB9h	__(2)	F99h	__(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	__(2)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON ⁽³⁾	F97h	__(2)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS ⁽³⁾	F96h	TRISE ⁽³⁾
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD ⁽³⁾
FF4h	PRODH	FD4h	__(2)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	__(2)
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	__(2)
FEFh	INDF0 ⁽¹⁾	FCFh	TMR1H	FAFh	SPBRG	F8Fh	__(2)
FEEh	POSTINC0 ⁽¹⁾	FCEh	TMR1L	FAEh	RCREG	F8Eh	__(2)
FEDh	POSTDEC0 ⁽¹⁾	FCDh	T1CON	FADh	TXREG	F8Dh	LATE ⁽³⁾
FECh	PREINC0 ⁽¹⁾	FCCh	TMR2	FACH	TXSTA	F8Ch	LATD ⁽³⁾
FEBh	PLUSW0 ⁽¹⁾	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADDR	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	__(2)
FE7h	INDF1 ⁽¹⁾	FC7h	SSPSTAT	FA7h	EECON2 ⁽¹⁾	F87h	__(2)
FE6h	POSTINC1 ⁽¹⁾	FC6h	SSPCON1	FA6h	EECON1	F86h	__(2)
FE5h	POSTDEC1 ⁽¹⁾	FC5h	SSPCON2	FA5h	__(2)	F85h	__(2)
FE4h	PREINC1 ⁽¹⁾	FC4h	ADRESH	FA4h	__(2)	F84h	PORTE ⁽³⁾
FE3h	PLUSW1 ⁽¹⁾	FC3h	ADRESL	FA3h	__(2)	F83h	PORTD ⁽³⁾
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Note 1: This is not a physical register.

2: Unimplemented registers are read as '0'.

3: This register is not available on 28-pin devices.

Figure 7.4-8

The PIC18F4620 has 16 data memory banks (0–15), implemented as static RAM. Each memory bank contains 256 file registers, so the PIC18F4620 is able to store a total of 4096 bytes of data. Two types of register file exist: General Purpose Registers (GPR) and Special Function Registers (SFR). The SFRs are implemented in such a way in the PIC18

that they are all located in the last 128 bytes of Bank 15 and can be accessed at any time without having to select data banks. A list of the SFRs is shown on Figure 7.4-, and a brief description of the SFRs is given in section 7.4.4.1.

Another special section of the data memory, called the Access Bank, is located in the first 128 bytes of Bank 0. The Access Bank can be accessed at any time without switching banks. It was created to allow faster access to the most frequently used memory locations and avoid errors caused by wrong bank switching. A special opcode called the RAM access bit, denoted “a”, is used to control whether or not the Access Bank is to be used.

The Data Memory structure is illustrated on Figure 7.4-. A special function register called the Bank Select Register (BSR) controls which data bank is selected. Only the last 4 bits of BSR are implemented to select one of the 16 memory banks available. The first 128 bytes of Bank 0 contains the Access RAM and the last 128 bytes of Bank 15 contains the SFRs. These can be accessed without setting the BSR. Most instructions use the lower 8 bits of the memory address and the 4 bits of the BSR to determine each memory location. The BSR can be set directly using the **movlb** instruction. In the PIC18 instruction set, only the **movff** instruction carries the full 12-bit address of each memory location and ignores the BSR completely.

The General Purpose Registers (GPRs) can be accessed by any instruction either directly or indirectly. Users must be careful to correctly set the BSR when addressing the GPRs directly. The GPRs can also be accessed indirectly through the File Select Registers (FSRs).

The **STATUS** register in PIC18s has the dedicated purpose of monitoring the status of the ALU. The lowest 3 bits are the same as the PIC16, the **Carry**, **Digital Carry**, and **Zero** bits. The 4th bit (bit 3) is the **Overflow** bit (OV). This flag is used in signed arithmetic operations. It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state. Bit 4 is the **Negative** bit (N), which indicates whether or not the result of a signed arithmetic operation is negative. The upper 3 bits are not implemented and are read as ‘0’.

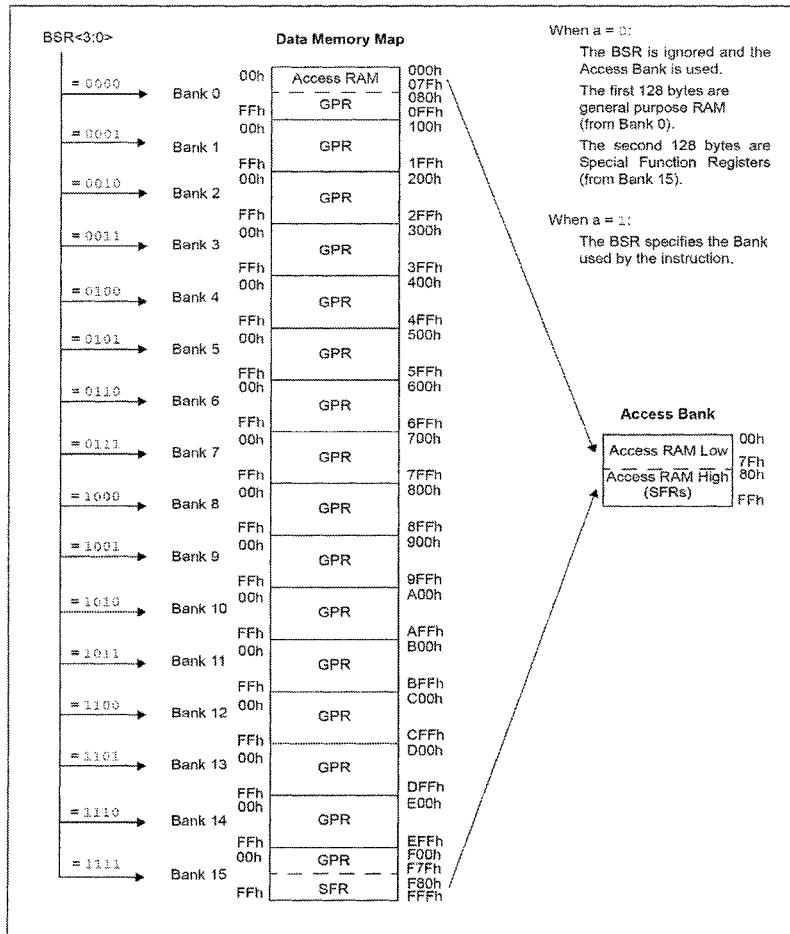


Figure 7.4-9

7.4.4.1 Special Function Registers

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
SPBRGH	USART Baud Rate Generator Register High Byte								0000 0000
SPBRG	USART Baud Rate Generator Register Low Byte								0000 0000
RCREG	USART Receive Register								0000 0000
TXREG	USART Transmit Register								0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000z
EEADR	—	—	—	—	—	—	—	—	EEPROM Addr Register High
EEDATA	EEPROM Address Register								0000 0000
EECON2	EEPROM Data Register								0000 0000
EECON1	EEPGD	CFG5	—	FREE	WRERR	WREN	WR	RD	xx-0 x000
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	11-1 1111
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	00-0 0000
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	00-0 0000
IPR1	PSPIP ⁽²⁾	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111
PIR1	PSPIF ⁽²⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000
PIE1	PSPIE ⁽²⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000
OSCTUNE	INTSRC	PLLEN ⁽³⁾	—	TUN4	TUN3	TUN2	TUN1	TUN0	00-0 0000
TRISE ⁽²⁾	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	0000 -111
TRISD ⁽²⁾	PORTD Data Direction Control Register								1111 1111
TRISC	PORTC Data Direction Control Register								1111 1111
TRISB	PORTB Data Direction Control Register								1111 1111
TRISA	TRISA7 ⁽⁵⁾	TRISA6 ⁽⁵⁾	Data Direction Control Register for PORTA						1111 1111
LATE ⁽²⁾	—	—	—	—	—	PORTE Data Latch Register (Read and Write to Data Latch)			---- -xxx
LATD ⁽²⁾	PORTD Data Latch Register (Read and Write to Data Latch)								xxxx xxxx
LATC	PORTC Data Latch Register (Read and Write to Data Latch)								xxxx xxxx
LATB	PORTB Data Latch Register (Read and Write to Data Latch)								xxxx xxxx
LATA	LATA7 ⁽⁶⁾	LATA6 ⁽⁶⁾	PORTA Data Latch Register (Read and Write to Data Latch)						xxxx xxxx
PORTE	—	—	—	—	RE3 ⁽⁴⁾	RE2 ⁽²⁾	RE1 ⁽²⁾	RE0 ⁽²⁾	---- xxxx
PORTD ⁽²⁾	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
PORTA	RA7 ⁽⁶⁾	RA6 ⁽⁶⁾	RA5	RA4	RA3	RA2	RA1	RA0	xx0x 0000
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000
ADRESH	A/D Result Register High Byte								xxxx xxxx
ADRESL	A/D Result Register Low Byte								xxxx xxxx
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	--00 0000
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0qqq
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000
CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx
CCP1CON	P1M1 ⁽²⁾	P1M0 ⁽²⁾	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000
CCPR2H	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0-00
PWM1CON	PRSEN	PDC6 ⁽²⁾	PDC5 ⁽²⁾	PDC4 ⁽²⁾	PDC3 ⁽²⁾	PDC2 ⁽²⁾	PDC1 ⁽²⁾	PDC0 ⁽²⁾	0000 0000
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1 ⁽²⁾	PSSBD0 ⁽²⁾	0000 0000
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111
TMR3H	Timer3 Register High Byte								xxxx xxxx
TMR3L	Timer3 Register Low Byte								xxxx xxxx
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000
TOSH	Top-of-Stack High Byte (TOS<15:8>)								
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								
STKPTR	STKFUL ⁽⁶⁾	STKUNF ⁽⁶⁾	—	SP4	SP3	SP2	SP1	SP0	00-0 0000
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000
PCLATH	Holding Register for PC<15:8>								
PCL	PC Low Byte (PC<7:0>)								
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					---0 0000
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								
TABLAT	Program Memory Table Latch								
PRODH	Product Register High Byte								
PRODL	Product Register Low Byte								
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								
WREG	Working Register								
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								
BSR	—	—	—	—	Bank Select Register				
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx
TMR0H	Timer0 Register High Byte								
TMR0L	Timer0 Register Low Byte								
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0100 q000
HLVDCON	VDIRMAG	—	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0	0-00 0101
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0
RCON	IPEN	SBOREN ⁽¹⁾	—	RI	TO	PD	POR	BOR	0q-1 11q0
TMR1H	Timer1 Register High Byte								
TMR1L	Timer1 Register Low Byte								
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000
TMR2	Timer2 Register								
PR2	Timer2 Period Register								
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
SSPBUF	MSSP Receive Buffer/Transmit Register								
SSPADD	MSSP Address Register in I ² C™ Slave Mode. MSSP Baud Rate Reload Register in I ² C Master Mode.								

7.4.4.2 Indirect Addressing

The PIC18 architecture expands the indirect addressing capabilities so users can create data structures like tables and arrays more efficiently. PIC18s have 3 FSRs (File Select Registers), FSR2:0. Each FSR acts as a pointer to a data memory location. Each FSR is a pair of SFRs (for example, FSR0H and FSR0L). Together these two registers contain 12 bits of addressable locations, enough to address the entire data memory (See Figure 7.4-10). Access to the physical registers pointed to by each FSR is achieved through the INDF virtual registers. There are 3 such registers, INDF2:0, which correspond to the 3 FSRs. Each INDF register contains the value of the register pointed to by each FSR.

To initiate indirect addressing, a value must be loaded into both registers of the FSR pair. Since the FSRs contain all 12 bits of each data memory address, no bank selection is needed in indirect addressing. If the FSR register containing the lower 8 bits overflows, it will automatically increment into the register containing the higher bits.

To further simplify indirect memory access, each FSR provides 4 additional operands. These are also virtual registers that cannot be indirectly read or written to. They are associated with the memory location pointed to by the FSR, but can also perform operations on FSR at the same time. These are:

POSTDEC – Accesses the FSR value, then decrements the FSR by 1

POSTINC – Accesses the FSR value, then increments the FSR by 1

PREINC – Increments the FSR by 1, then uses that value in the operation

PLUSW – Adds the signed value of the W register to the current value of the FSR and use the new value in the operation. By manipulating the value of W, the user can reach addresses a fixed value from the referenced address. This can be used to implement data structures like software stacks.

Using an instruction with one of the indirect addressing registers as the operand...

...uses the 12-bit address stored in the FSR pair associated with that register....

...to determine the data memory location to be used in that operation.

In this case, the FSR1 pair contains ECCh. This means the contents of location ECCh will be added to that of the W register and stored back in ECCh.

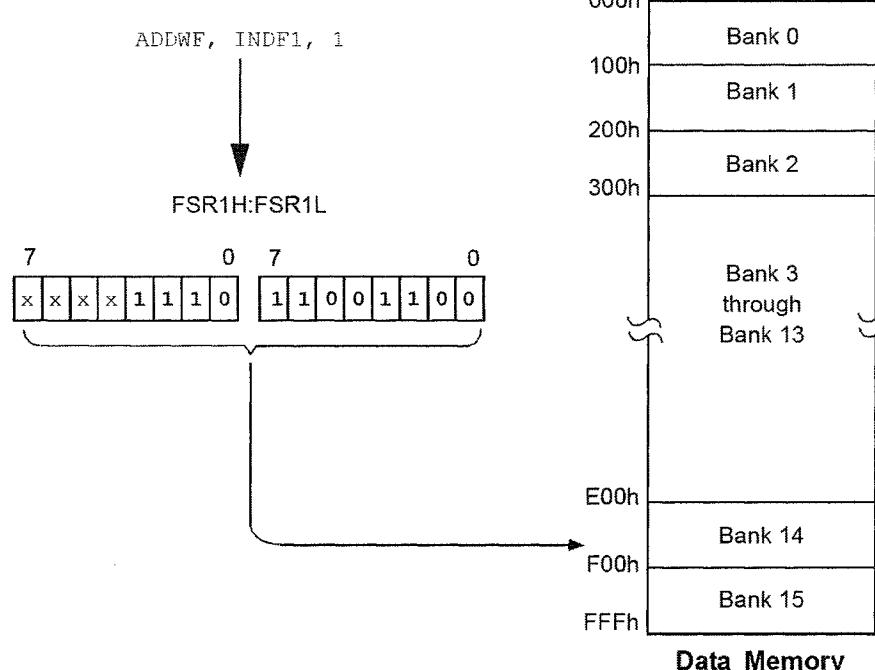


Figure 7.4-10

7.4.5 PIC16 EEPROM Data Memory

The PIC16F877, PIC16F887, and PIC16F1937 are also equipped with 256 bytes EEPROM data memory for fast erase/write cycles of frequently updated data. An on-chip timer controls the write time, which varies with voltage. The read and write operations take place on a single byte without interfering with the normal operation of the microcontroller. Read and write access to EEPROM memory (as well as FLASH program memory) take place indirectly through these

Special Function Registers: **EEDATA**, **EEADR**, **EECON1**, and **EECON2**. When interfacing to EEPROM data memory, the **EEADR** register holds the address to be accessed. Depending on the operation, the **EEDATA** register holds the data to be written, or the data read, at the address in **EEADR**. Since the PIC16F877 and PIC16F887 device has 256 bytes of EEPROM data memory and therefore uses all 8 bits of the **EEADR**. The **EECON1** register is the control register for configuring and initiating the access.

The PIC16F877 and PIC16F887 EEPROM is very useful for storing operational settings such as user names, passwords, statistics, etc. If the machine utilizes an LCD display, the EEPROM can be used to dramatically shorten the code required to output messages to the LCD by pre-storing the message in the EEPROM.

7.4.5.1 EECON1 and EECON2

The **EECON1** register is **EECON1 REGISTER (ADDRESS 18Ch)** the control register for configuring and initiating the access. The **EECON2** register is *not* a physically implemented register, but is used exclusively in the memory write sequence to prevent inadvertent writes. To access the EEPROM, the **EEPGD** bit (**EECON1<7>**) should be cleared (set to 0).

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD

bit 7

bit 0

Read operations only use one additional bit, **RD** (**EECON1<0>**), which initiates the read operation from the desired memory location. Once this bit is set, the value of the desired memory location will be available in the data registers. The bit is cleared automatically at the end of the read operation. The data will then be available at the **EEDATA** register in the very next instruction cycle after the **RD** bit is set.

To perform a write operation, two bits are controlled, **WREN** (**EECON1<2>**) and **WR** (**EECON1<0>**). When **WREN** is cleared, the write operation is disabled. This allows us to prevent writes from happening by mistake. The **WR** bit is set high to initiate a write sequence and will also be automatically cleared once the write sequence is complete. When the write sequence is complete, an **EIF** interrupt bit is set high. The **WRERR** bit is used to indicate that the device has been reset during a write operation. **WRERR** *must* be cleared after a Power-on-Reset and checked on any other reset.

7.4.5.2 Writing to the EEPROM

The steps to write to EEPROM data memory are:

1. Disable interrupts (if using interrupts).
2. Write the address to **EEADR**. Make sure that the address is not larger than the memory size of the PIC16F87X device.
3. Write the 8-bit data value to be programmed in the **EEDATA** register.
4. Clear the **EEPGD** bit to point to EEPROM data memory.
5. Set the **WREN** bit to enable program operations.
6. Disable interrupts (if enabled).
7. Execute the special five instruction sequence:
 - Write 55h to **EECON2** in two steps (first to W, then to **EECON2**)
 - Write AAh to **EECON2** in two steps (first to W, then to **EECON2**)
 - Set the **WR** bit.
8. Wait for the write operation to finish.
9. Clear the **WREN** bit to disable program operations.
10. At the completion of the write cycle, the **WR** bit is cleared and the **EIF** interrupt flag bit is set. Clear EEPROM write interrupt flag **EIF**.
11. Enable interrupts (if using interrupts).

BCF	INTCON, GIE	; Only disable interrupts
BCF	STATUS, RP0	; if already enabled,
BSF	STATUS, RP1	; otherwise discard
MOVF	ADDR, W	; Bank 2
MOVWF	EEADR	; Address to
MOVF	VALUE, W	; Write to
MOVWF	EEDATA	; Data to
BSF	STATUS, RP0	; Bank 3
BCF	EECON1, EEPGD	; Point to Data memory
BSF	EECON1, WREN	; Enable writes
MOVLW	0x55	; Write 55 th to
MOVWF	EECON2	; EECON2
MOVLW	0Xaa	; Write AAh to
MOVWF	EECON2	; EECON2
BSF	EECON1, WR	; Start write operation
BTFSC	EECON1, WR	; Wait for
GOTO	\$-1	; Write to finish
BCF	EECON1, WREN	; Disable writes
BCF	STATUS, RP0	; Bank0
BCF	STATUS, RP1	; Clear write interrupt flag
BCF	PIR2, EIF	; Only enable interrupts
BSF	INTCON, GIE	; if using interrupts,
		; otherwise discard

NOTE: The sample code is for PIC16F877 only. It should be modified according to the device, in order to meet the requirements for that specific device. For instance, bank selection should be done using BSR for PIC16F1937.

7.4.5.3 Reading from the EEPROM

The steps to reading the EEPROM data memory are:

1. Write the address to EEDATA. Make sure that the address is not larger than the memory size of the PIC16F877 device.
2. Clear the EEPGD bit to point to EEPROM data memory.
3. Set the RD bit to start the read operation.
4. Read the data from the EEDATA register.

BSF	STATUS, RP1	;
BCF	STATUS, RP0	; Bank 2
MOVF	ADDR, W	; Write address
MOVWF	EEADR	; to read from
BSF	STATUS, RP0	; Bank 3
BCF	EECON1, EEPGD	; Point to data memory
BSF	EECON1, RD	; Start read operation
BCF	STATUS, RP0	; Bank 2
MOVF	EEDATA, W	; W = EEDATA

NOTE: The sample code is for PIC16F877 only. It should be modified according to the device, in order to meet the requirements for that specific device. For instance, bank selection should be done using BSR for PIC16F1937.

7.4.6 PIC18 EEPROM Data Memory

The PIC18F4620 comes with 1024 bytes of EEPROM non-volatile data memory for internal data storage. The EEPROM is not mapped into either the program memory or the data memory, but can be accessed indirectly through the SFRs. Five SFRs control the read and write operation of the EEPROM: **EEDATA**, **EECON1**, **EECON2**, **EEADR**, and **EEADRH**. **EEADR** and **EEADRH** specify the 10 bit address of the EEPROM memory, since it is 1024 bytes in size. **EECON1** and **EECON2** control the read and write operations of the EEPROM and **EEDATA** holds the value being read or written to.

The structure and operation of the EEPROM is very similar to the PIC16. The only difference worth noting is bit 6 of the EECON1 register. This bit is called the Flash Program/Data EEPROM or Configuration Select bit (CFG5), and it allows access to either the configuration registers or the Flash memory/EEPROM. When it is set, configuration registers can be accessed. When it's cleared, the flash memory or data EEPROM can be accessed depending on the **EEPGD** bit. For a description of the other EEPROM control registers, refer to the PIC18F4620 Datasheet.

7.4.6.1 Writing to the EEPROM

The sample code below shows the procedure of writing to the Data EEPROM memory.

Required Sequence	MOVLW	DATA_EE_ADDRH	;
	MOVWF	EEADRH	; Upper bits of Data Memory Address to write
	MOVLW	DATA_EE_ADDR	;
	MOVWF	EEADR	; Lower bits of Data Memory Address to write
	MOVLW	DATA_EE_DATA	;
	MOVWF	EEDATA	; Data Memory Value to write
	BCF	EECON1, EPGD	; Point to DATA memory
	BCF	EECON1, CFG5	; Access EEPROM
	BSF	EECON1, WREN	; Enable writes
	BCF	INTCON, GIE	; Disable Interrupts
MOVLW	55h	;	
MOVWF	EECON2	; Write 55h	
MOVLW	0xAA	;	
MOVWF	EECON2	; Write 0xAA	
BSF	EECON1, WR	; Set WR bit to begin write	
BSF	INTCON, GIE	; Enable Interrupts	
BCF	EECON1, WREN	; User code execution	
BCF	EECON1, WREN	; Disable writes on write complete (EEIF set)	

7.4.6.2 Reading from the EEPROM

The following sample code shows the procedure of reading from the EEPROM memory.

MOVLW	DATA_EE_ADDRH	;
MOVWF	EEADDRH	; Upper bits of Data Memory Address to read
MOVLW	DATA_EE_ADDR	;
MOVWF	EEADR	; Lower bits of Data Memory Address to read
BCF	EECON1, EEPGD	; Point to DATA memory
BCF	EECON1, CFGS	; Access EEPROM
BSF	EECON1, RD	; EEPROM Read
MOVF	EEDATA, W	; W - EEDATA

7.5 Additional Core Features

7.5.1 8x8 Hardware Multiplier

All PIC18 devices include an 8x8 hardware multiplier embedded within the ALU. This greatly increases the efficiency of multiplication operations, since an 8x8 multiplication can be accomplished in a single cycle. The multiplier performs unsigned multiplication between the W register and a file register and stores the 16-bit result in the registers PRODH:PRODL. The operation does not affect any STATUS flags. The sample code shown on the right is the algorithm for performing an 8x8 signed multiplication operation. For 16x16 multiplication algorithms, refer to the PIC18F4620 Datasheet.

MOVF	ARG1, W	
MULWF	ARG2	; ARG1 * ARG2 ->
BTFS	ARG2, SB	; PRODH:PRODL
SUBWF	PRODH, F	; Test Sign Bit
MOVF	ARG2, W	; PRODH = PRODH
BTFS	ARG1, SB	; -ARG1
SUBWF	PRODH, F	
		; PRODH = PRODH
		; -ARG2

7.5.2 PLL and Internal Oscillator

The DevBugger board is equipped with a 10 MHz crystal oscillator. Although this will be fast enough for most applications, some users may want to run on a faster frequency. The PIC18F4620 comes with a Phase-locked Loop 4x frequency multiplier that is capable of producing clock speeds of up to 40 MHz using a 10 MHz oscillator. This feature can be enabled by setting the configuration parameter **OSC** to **HSPLL**.

The PIC18F4620 also has an internal oscillator block with two clock sources. This eliminates the need for an external crystal oscillator to drive the PIC and frees up two pins, RA6 and RA7, for digital I/O. The main clock source for the internal oscillator block is an 8 MHz oscillator and the secondary clock is a 31 kHz RC oscillator. The internal oscillator block also implements a postscaler, which can provide a range of frequencies from 31 kHz to 4 MHz. The internal oscillator block can also use the PLL frequency multiplier to generate a clock speed of up to 32 MHz. This must be done in software. The following steps illustrate the process of activating the internal oscillator and using the PLL.

1. Set the configuration parameter **OSC** to **INTIO67**. This will use the internal oscillator block as the primary oscillator and free up pin RA6 and RA7.
2. In the beginning of the code, set the value of the **OSCCON** register to **B'01110000'**. Bits 4-6 specify the internal oscillator frequency. At reset, the default internal oscillator frequency is set to 1 MHz. The value 111 sets the frequency to 8 MHz.
3. Set the **PLLEN** bit (**OSCTUNE<6>**). This will enable the PLL and change the oscillator frequency to 32 MHz.

When the oscillator frequency is changed, all the timing controls in the code also changes. All sample codes given are written assuming a 10 MHz clock. All delays must be changed corresponding to the oscillator speed. For custom length delays, refer to www.piclist.org.

7.5.3 Interrupts

The PIC18F4620 has multiple interrupt sources and an interrupt priority feature to allow interrupts to be processed as quickly as possible. The high priority interrupt vector is at 0x08 and the low priority interrupt vector is at 0x18. The high-priority interrupt events will interrupt any low-priority interrupts that may be in progress. There are ten registers which are used to control interrupt operation. These registers are: RCON, INTCON, INTCON2, INTCON3, PIR1 and PIR2, PIE1 and PIE2, IPR1 and IPR2. In general, each interrupt source has three bits to control its operation:

- Flag bit to indicate that an interrupt has occurred
- Enable bit to allow the program to service the interrupt when the Flag bit is set
- Priority bit to select high or low priority.

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When this is set, interrupts must be enabled by setting two global interrupt enable bits, GIEH (INTCON<7>) and GIEL (INTCON<6>), where GIEH enables all high-priority interrupts and GIEL enables all low-priority interrupts. When the interrupt flag enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When IPEN is cleared, the priority feature is disabled and the PIC is in Legacy mode. All interrupts generated go to the vector 0x08. In this case, the interrupt priority bits for each source have no effect and the global interrupt enable bit is GIE (INTCON<7>) and INTCON<6> is the PEIE (Peripheral Interrupt Enable) bit, which enables/disables all peripheral interrupts.

When an interrupt is being processed, the global interrupt enable bit is cleared to prevent further interrupts. High priority interrupts can interrupt low priority interrupts. When an interrupt occurs, the return address is pushed onto the stack and the program counter is loaded with the interrupt vector. In the interrupt service routine, the source of the interrupt can be found by polling each flag bit. Usually the flag bits must be cleared in software before the interrupt can be re-enabled. The `retfie` instruction returns to the main program and re-enables the global interrupt enable bit to re-enable the interrupts.

7.5.3.1 Enabling Interrupt

-First thing to do is to select Legacy or Priority mode interrupt operation in PIC18. This is done by setting or clearing IPEN bit in RCON register. By default, IPEN is clear, which means no priority levels and the PIC is in Legacy mode.

RCON Register

IPEN	SBOREN	—	RI	TO	PD	POR	BOR
------	--------	---	----	----	----	-----	-----

IPEN: Interrupt Priority Enable
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupt (PIC16 compatibility mode)

Peripheral Interrupt Set Up:

Peripheral Interrupt Priority Registers (IPR1, IPR2): If Priority Mode is selected, user can chose to set the peripheral interrupt priority using IPR1 and IPR2 Register. To select High Priority we set the bits to 1, low priority is 0.

IPR1 Register

PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
-------	------	------	------	-------	--------	--------	--------

PSPIP: Parallel Slave Interrupt Priority SSPIP: MSSP Interrupt Priority
ADIP: A/D Converter Interrupt Priority CCP1IP: CCP1 Interrupt Priority
RCIP: EUSART Rcv Interrupt Priority TMR2IP: Timer2 Interrupt Priority
TXIP: EUSART Tx Interrupt Priority TMR1IP: Timer1 Interrupt Priority

IPR2 Register

OSCIP	CMIP	---	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP
-------	------	-----	------	-------	--------	--------	--------

OSCIP: Oscillator Fail Interrupt Priority
 CMIP: Comparator Interrupt Priority
 --- Unimplemented Bit
 EEIP: Data EEPROM/Flash Write Operation Interrupt Priority
 BCLIP: Bus Collision Interrupt Priority
 HLVDIP: High/Low Voltage Detect Interrupt Priority
 TMR3IP: Timer3 Interrupt Priority
 CCP2IP: CCP2 Interrupt Priority

Peripheral Interrupt Flags Registers (PIR1, PIR2): We also need to ensure that the peripheral interrupt flags are cleared by clearing PIR1 and PIR2 Registers. The bits in these two registers are similar to those in IPR1 and IPR2 except instead of interrupt priority, they represent interrupt flags. If a bit is set to 1, an interrupt is pending or being processed. To clear these bits we set them to 0.

Peripheral Interrupt Enables Registers (PIE1, PIE2): Next we set the appropriate bits in peripheral interrupt enables register PIE1 and PIE2. Again the bits represent similar things as IPR1 and IPR2

Core Interrupt Set Up:

Core Interrupt Priority: When Priority Mode is selected for PIC18 interrupts, the PIC18 needs to know the priority of Core interrupt. Timer0 overflow interrupt priority as well as RB port change interrupt priority can be set by bits INTCON2<2> and INTCON2<0>.

INTCON2 Register:

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMROIP	—	RBIP
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	RBPU: PORTB Pull-up Enable bit 1 = All PORTB pull-ups are disabled 0 = PORTB pull-ups are enabled by individual port latch values
bit 6	INTEDG0: External Interrupt 0 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 5	INTEDG1: External Interrupt 1 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 4	INTEDG2: External Interrupt 2 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 3	Unimplemented: Read as '0'
bit 2	TMROIP: TMRO Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	Unimplemented: Read as '0'
bit 0	RBIP: RB Port Change Interrupt Priority bit 1 = High priority 0 = Low priority

PIC18 can have external interrupts as its interrupt source. We can set the priorities of these external interrupts using INTCON3 register. INT1 external interrupt priority can be set by bit INT1IP in INTCON3, INT2 external interrupt priority is set by bit INT2IP in INTCON3 register. Note that INT0 does not have an IP bit; it is always a high priority interrupt.

Core Interrupt Flags:

INTCON Register

GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
----------	-----------	--------	--------	------	--------	--------	------

TMR0IF: TMR0 Overflow Interrupt Flag
 INT0IF: INT0 External Interrupt Flag
 RBIF: RB Port Change Interrupt Flag

INTCON3 Register

INT2IP	INT1IP	---	INT2IE	INT1IE	---	INT2IF	INT1IF
--------	--------	-----	--------	--------	-----	--------	--------

INT2IF: INT2 External Interrupt Flag
 INT1IF: INT1 External Interrupt Flag

If any of the five bits shown above is set, it represents an interrupt has been raised and it is pending or being processed. To clear an interrupt flag simply set it to 0.

Core Interrupt Enables:

Similar to peripheral interrupt enables, the five core interrupt also needs to be enabled. In INTCON register:

- TMR0IE: TMR0 Overflow Interrupt Enable
- INT0IE: INT0 External Interrupt Enable
- RBIE: RB Port Change Interrupt Enable

In INTCON3 register:

- INT2IE: INT2 External Interrupt Enable
- INT1IE: INT1 External Interrupt Enable

Low priority interrupts or peripheral interrupts can be enabled by setting PIEI/GIE bit in INTCON register. When Legacy mode is selected (RCON<IPEN> = 0), setting this bit enables all unmasks peripheral interrupts, and otherwise disables all peripheral interrupts. If Priority Mode (RCON<IPEN> = 1) is selected, setting this bit enables all low priority peripheral interrupts, or otherwise disables them.

Last bit needs to be set to completely specify the PIC18 interrupt is GIE/GIEH<INTCON> bit. This bit enables high priority or global interrupts. When Legacy mode is chosen, setting this bit enables all unmasks interrupts, and in Priority Mode setting this bit enables all high priority interrupts.

7.5.3.2 Interrupt Support

7.5.3.2.1 Creating Interrupt Service Routine

It is the Programmer's responsibility to provide code for interrupt service routine. In another word, the programmer needs to tell the PIC what it needs to do when an interrupt occurs.

```

org 0x0000
goto start
org 0x0008
...
;high-priority interrupt service routine
org 0x0018
...
;low-priority interrupt service routine
retfie ;return from interrupt

```

```

start  ...
      ...
ende           ; your program

```

The PIC18 microcontroller unit reserves a small block of memory location to hold the reset handling routine and high-priority and low-priority interrupt service routines. The rest, the high-priority interrupt, and the low-priority interrupt service routines start at 0x0000, 0x0008, and 0x0018, respectively. The user program should start somewhere after 0x0018.

Note that interrupt flags should be cleared in the interrupt service routine for the next interrupt. Also context saving should be done which will be explained in the next section.

7.5.3.2.2 Context Saving During Interrupts

During interrupts, the return program counter address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the Fast Return Stack. If a fast return from interrupt is not used, the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. The following code template saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

```

MOVWF W_TEMP ; W_TEMP is in virtual bank
MOVFF STATUS, STATUS_TEMP ; STATUS_TEMP located anywhere
MOVFF BSR, BSR_TEMP ; BSR_TEMP located anywhere
;
; USER ISR CODE
;
MOVFF BSR_TEMP, BSR ; Restore BSR
MOVF W_TEMP, W ; Restore WREG
MOVFF STATUS_TEMP, STATUS ; Restore STATUS

```

7.5.3.2.3 Firmware Engineer's Responsibilities

When programming PIC18 interrupts, there are several things that a user needs to watch out for. First of all, the user must ensure that the interrupt source is enabled. Secondly, Interrupt priority levels must be set and interrupt flags must be manually clear during initialization, and during every interrupt service routine. Lastly, the user must save any registers/variables you access in the ISR's code if they are not handled automatically by the compiler.

7.5.4 PIC16 I/O Ports

7.5.4.1 PIC16F877

The PIC16F877 has 5 ports and a total number of 33 I/O pins, 6 pins for PORTA (6-bit wide), 8 pins for each of PORTB, PORTC, and PORTD (8-bit wide), and 3 pins for PORTE (3-bit wide). Each pin is bi-directional, i.e., it can be used as input or output, depending on the state of the corresponding bit in the TRIS register for that particular port. For example, if the TRISA register contains binary 00001111, then PORTA bits 0, 1, 2, and 3 are input ('1' bits set in TRIS) and bit 4 and 5 are output ('0' bit set). The other bits in the TRISA register are "unimportant", since the corresponding data latches and pins do not exist.

BCF	STATUS, RP0	;
BCF	STATUS, RP1	; Bank 0
CLRF	PORTA	; Initialize PORTA by ; clearing output data latches
BSF	STATUS, RP0	; Select Bank 1
MOVLW	0x06	; Configure all pins
MOVWF	ADCON1	; as digital inputs
MOVLW	0xCF	; Value used to initialize data direction
MOVWF	TRISA	; Set RA<3:0> as inputs ; RA<5:4> as outputs ; TRISA<7:6> are always read as ; '0'.

The user should also notice that the digital I/O pins are set to input by default, so it is always a good programming practice to initialize the TRIS to the appropriate value before each use.

An important note is the electrical limitations of the I/O ports. For the PIC16F877, the absolute maximum output or input pin current is 25 mA. However, if PORTA, PORTB, and PORTE are used together, the total current sunk or sourced by their pin must not exceed 200 mA. The same limit (200 mA) exists for PORTC and PORTD, when used simultaneously.

It is important to note that some pins of the I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin. Some important mode multiplexing for the PIC16F877 are indicated below:

- RA4 is a special pin. It has a Schmitt trigger input, which can be used as a timer/counter input to TMR0 for becoming the RA4/T0CKI pin, if you want to use it for that.
- RA4 also has an open-drain output configuration. This means that if you want to use it for output like the other pins, you will need to use an external pull-up resistor. You can pull the pin LOW under program control, or you can let it float, but to make it assume a HIGH level you will need an external pull-up resistor.
- RA0-RA3 are multiplexed with analog inputs and analog VREF input, depending on the state of the ADCON1 register bits (A/D Control Register1). This register is at address 9Fh, which in Bank1. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register. **On a Power-on Reset, these pins are configured as analog inputs and read as '0'.** Further, the TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. You must ensure the bits in the TRISA register are maintained set when using them as analog inputs. As an example, see the code for initializing PORTA on the right hand side.
- PORTB pins usually have weak internal pull-ups that can be enabled or disabled by the RBPU bit in the OPTION register
- PORTB pins 4 through 7 can cause an interrupt-on-change, if you have enabled the RBIE bit in the OPTION register. Only pins configured as inputs can cause this interrupt to occur, i.e., any RB4:RB7 pin configured as an output is excluded from the interrupt-on-change comparison. The input pins of RB4:RB7 are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB4:RB7 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF. This interrupt can also wake the processor from SLEEP mode. The user, in the interrupt Service Routine, can clear the interrupt through the following: a) any read or write of PORTB, which will end the mismatch condition; b) clear flag bit RBIF. This interrupt-on-change feature, along with the weak pull-ups, makes it ideal for keypad interfacing with wake-on-keypress feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.
- RB0 also is usually used for the INT interrupt pin (external interrupt input pin). This is configured using the INTEDG bit in the OPTION_REG register.
- RE0-RE2 pins have the same situation as RA0-RA3 pins. The same register ADCON1 also controls PORTE pins for analog input. To jointly set PORTA and PORTE for analog use, set ADCON1's bits 0 to 3 to 0000 (the default condition on power-up). To jointly use PORTA and PORTE for digital purposes, the bits are set to 0111.

The large number of available I/O port pins brings considerable flexibility into the project design. For example, assume that you want to design a remote combination-keypad lock for controlling access to a room. It needs to be battery-powered, so low power is a primary consideration. You also want to sound an alarm if the lock is tampered with or opened. You do not want to store combinations in the box. Instead, you would like to manage them from a central computer, so you intend to send the combination to a remote computer to authenticate the person and log access. You might want the following pins for your tasks:

- You could use RB7-1 for a 3x4 keypad. Using RB7-4's weak pullups and interrupt-on-change capability, you can put the CPU in ultra-low-power SLEEP mode when it is not in use for longer battery life.
- Use RB0 for a tamper switch to wake up the PIC, sound an alarm, and send a message to the central computer.
- Use RA4 for a multi-drop, shared serial bus along with any number of other units to send a serial data stream back to the central computer on one signal wire (plus ground, of course).
- Use RA2 and RA3 for status LED's and/or a piezo-beeper to give the user feedback when keys are pressed, blink red or green to indicate whether he or she has used a valid password, etc.

You are still left with plenty of unused pins. The code will easily fit into 1K word program memory with room to spare. All keypad debouncing and serial communication functions can be handled in software, with no external devices required. And you can run the whole thing on a few AA batteries. The CPU is happy with anything from 2 to 6 Volts. You could even store passwords locally in EEPROM, where they would be safe from power failures.

7.5.4.2 PIC16F887

The PIC16F887 has 5 ports and a total number of 35 I/O pins, 8 pins for PORTA (8-bit wide), 8 pins for each of PORTB, PORTC, and PORTD (8-bit wide), and 3 pins for PORTE (3-bit wide), plus one input-only pin in PORTE. Each pin is bi-directional, i.e., it can be used as input or output, depending on the state of the corresponding bit in the TRIS register for that particular port. For example, if the TRISA register contains binary 00001111, then PORTA bits 0, 1, 2, and 3 are input ('1' bits set in TRIS) and bit 4, 5, 6, and 7 are output ('0' bit set).

```

BANKSEL PORTA           ;Init PORTA
CLRF PORTA              ;
BANKSEL ANSEL            ;
CLRF ANSEL              ;digital I/O
BANKSEL TRISA            ;
MOVLW 0Ch                ;Set RA<3:2> as inputs
MOVWF TRISA              ;and set RA<5:4,1:0>
                           ;as outputs

```

The user should also notice that the digital I/O pins are set to input by default, so it is always a good programming practice to initialize the TRIS to the appropriate value before each use. **Note that for this PIC, the ANSEL registers (ANSEL and ANSELH) must be initialized to configure the Analog/Digital pins as digital.**

An important note is the electrical limitations of the I/O ports. For the PIC16F887, the absolute maximum output or input pin current is 25 mA. Also the total current sunk or sourced by all of the ports combined shall not exceed 90 mA limit.

It is important to note that some pins of the I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin. Some important mode multiplexing for the PIC16F887 are indicated below:

- RA4 is a special pin. It has a Schmitt trigger input, which can be used as a timer/counter input to TMR0 for becoming the RA4/T0CKI pin, if you want to use it for that. In addition, it can be the output of comparator, C1.
- RA0-RA3 and RA5 are multiplexed with analog inputs and analog VREF input. The VREF input is selected using the ADCON1 register. This register is at address 9Fh, which in Bank1. However, the analog state of each pin is indicated by setting the corresponding bit of ANSEL register in Bank3 high. For pins RA0-RA3 and RA5, these bits are ANS0-ANS4, which are ANSEL<4:0>. **On a Power-on Reset, these pins are configured as analog inputs and read as '0'.** Further, the TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. You must ensure the bits in the TRISA register are maintained set when using them as analog inputs. As an example, see the code for initializing PORTA on the right hand side. Moreover, pins RA0 and RA1 can be used as comparator C1 or C2 negative input, while RA3 and RA2 can be used as C1 and C2 positive input respectively. RA0 can also be multiplexed as Ultra low-Power Wake-up input.
- PORTB pins usually have weak internal pull-ups that can be enabled or disabled by the RBPU bit in the OPTION register. Pins RB0-RB5 can be configured as both analog or digital. To set these pin as analog, the corresponding bits of ANSELH register in Bank3 should be set. These bits are ANSELH<5:0>, named ANS8-ANS13. Also, pins RB1 and RB3 can be comparator C1 or C2 negative inputs.
- All PORTB pins can cause an interrupt-on-change, if you have enabled the RBIE bit in the OPTION register. In addition, a new register, IOCB (located in Bank1) should be configured for proper interrupt-on-change. If RBIE is set, setting each one bit of IOCB<7:0> high will enable interrupt-on-change for PORTB<7:0>. Only pins configured as inputs can cause this interrupt to occur, i.e., ant RB0:RB7 pin configured as an output is excluded from the interrupt-on-change comparison. The input pins of RB0:RB7 are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB0:RB7 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF. This interrupt can also wake the processor from SLEEP mode. The user, in the interrupt Service Routine, can clear the interrupt through the following: a) any read or write of PORTB, which will end the mismatch condition; b) clear flag bit RBIF. This interrupt-on-change feature, along with the weak pull-ups, makes it ideal for keypad interfacing with wake-on-keypress feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.
- RB0 also is usually used for the INT interrupt pin (external interrupt input pin). This is configured using the INTEDG bit in the OPTION_REG register.
- RB3, RB6 and RB7 can be used for ICSP, respectively for enabling it, ICSP clock, and data in/out.
- RE0-RE2 pins have the same situation as RA0-RA3 pins. The same register ANSEL in Bank3 also controls PORTE pins for analog input. To jointly use PORTA and PORTE for digital purposes, the ANSEL register should be cleared. Bits ANSEL<7:5> correspond to RE0-RE2, and are named ANS5-ANS7.
- RE3 can also be used a general purpose input (no output).

The large number of available I/O port pins brings considerable flexibility into the project design. For example, assume that you want to design a remote combination-keypad lock for controlling access to a room. It needs to be battery-powered, so low power is a primary consideration. You also want to sound an alarm if the lock is tampered with or opened. You do not want to store combinations in the box. Instead, you would like to manage them from a central computer, so you intend to send the combination to a remote computer to authenticate the person and log access. You might want the following pins for your tasks:

- You could use RB7-1 for a 3x4 keypad. Using RB7-4's weak pullups and interrupt-on-change capability, you can put the CPU in ultra-low-power SLEEP mode when it is not in use for longer battery life.
 - Use RB0 for a tamper switch to wake up the PIC, sound an alarm, and send a message to the central computer.
 - Use RA4 for a multi-drop, shared serial bus along with any number of other units to send a serial data stream back to the central computer on one signal wire (plus ground, of course).
 - Use RA2 and RA3 for status LED's and/or a piezo-beeper to give the user feedback when keys are pressed, blink red or green to indicate whether he or she has used a valid password, etc.

You are still left with plenty of unused pins. The code will easily fit into 1K word program memory with room to spare. All keypad debouncing and serial communication functions can be handled in software, with no external devices required. And you can run the whole thing on a few AA batteries. The CPU is happy with anything from 2 to 6 Volts. You could even store passwords locally in EEPROM, where they would be safe from power failures.

7.5.4.3 PIC16F1937

The PIC16F1937 has 5 ports and a total number of 36 I/O pins, 8 pins for PORTA (8-bit wide), 8 pins for each of PORTB, PORTC, and PORTD (8-bit wide), and 3 pins for PORTE (3-bit wide), plus one input-only pin in PORTE. Each pin is bi-directional, i.e., it can be used as input or output, depending on the state of the corresponding bit in the TRIS register for that particular port. For example, if the TRISA register contains binary 00001111, then PORTA bits 0, 1, 2, and 3 are input ('1' bits set in TRIS) and bit 4, 5, 6, and 7 are output ('0' bit set).

```

BANKSEL PORTA          ;
CLR PORTA              ;Init PORTA
BANKSEL ANSEL A        ;
CLR ANSEL A             ;digital I/O
BANKSEL TRISA          ;
MOVLW 0Ch               ;Set RA<3:2> as inputs
MOVWF TRISA             ;and set RA<5:4,1:0>
                           ;as outputs

```

The user should also notice that the digital I/O pins are set to input by default, so it is always a good programming practice to initialize the TRIS to the appropriate value before each use. Note that for this PIC, the ANSEL registers (**ANSELA**, **ANSELB**, **ANSELD**, and **ANSELE**) must be initialized to configure the Analog/Digital pins as digital.

An important note is the electrical limitations of the I/O ports. For the PIC16F1937, the absolute maximum output or input pin current is 25 mA.

It is important to note that some pins of the I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin. Some important mode multiplexing for the PIC16F1937 are indicated below:

- RA4 is a special pin. It has a Schmitt trigger input, which can be used as a timer/counter input to TMR0 for becoming the RA4/T0CKI pin, if you want to use it for that. In addition, it can be the output of comparator, C1, or Capture/Compare/PWM module. It can also be used for Timer0, for which the clock input can be indicated to be connected to RA4. Finally, it can be used as analog outputs for PIC16F1937's additional modules, such as SR latch and LCD Controller.
 - RA0-RA3 and RA5 are multiplexed with analog inputs and analog VREF input. The VREF input is selected using the ADCON1 register. This register is at address 9Eh, which in Bank1. However, the analog state of each pin is indicated by setting the corresponding bit of ANSELA register in Bank3 high. For pins RA0-RA3 and RA5, these bits are ANSA0-ANSA3 and ANSA5. **On a Power-on Reset, these pins are configured as analog inputs and read as '0'.** Further, the TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. You must ensure the bits in the TRISA register are maintained set when using them as analog inputs. As an example, see the code for initializing PORTA on the right hand side. Moreover, these

pins, RA0-RA3 and RA5 can be used as inputs or outputs of the comparator modules, or analog outputs for the LCD. Finally, RA2 and RA3 can even be used as A/D voltage reference input pins.

- PORTB pins usually have weak internal pull-ups that can be enabled or disabled by the RBPU bit in the OPTION register. Pins RB0-RB5 can be configured as both analog or digital. To set these pin as analog, the corresponding bits of ANSELB register in Bank3 should be set. These bits are ANSELB<5:0>, named ANSB0-ANS5. Also, pins RB1 and RB3 can be comparator C1 or C2 negative inputs.
- All PORTB pins can cause an interrupt-on-change, if you have enabled the RBIE bit in the OPTION register. In addition, a new register, IOCB (located in Bank7) should be configured for proper interrupt-on-change. If RBIE is set, each PORTB pin can be configured to raise an interrupt on the rising edge, or the falling edge. To enable interrupt on the rising edge, corresponding bits of IOCBP should be set high; to enable interrupt on the falling edge, corresponding bits of IOCBN should be set high. Setting both IOCBP and IOCBN bits of the same pin will result in interrupt for both edges. Only pins configured as inputs can cause this interrupt to occur, i.e., any RB0:RB7 pin configured as an output is excluded from the interrupt-on-change comparison. The input pins of RB0:RB7 are compared with the old value latched on the last read of PORTB. The “mismatch” outputs of RB0:RB7 are OR’ed together to generate the RB Port Change Interrupt with flag bit IOCIF of INTCON, as well as corresponding flag bit in register IOCBF, also in Bank7. This interrupt can also wake the processor from SLEEP mode. The user, in the interrupt Service Routine, can clear the interrupt through the following: a) any read or write of PORTB, which will end the mismatch condition; b) clear flag bit. This interrupt-on-change feature, along with the weak pull-ups, makes it ideal for keypad interfacing with wake-on-keypress feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.
- RB0 also is usually used for the INT interrupt pin (external interrupt input pin). This is configured using the INTEDG bit in the OPTION_REG register.
- For additional uses of RB1-RB7, please refer to the datasheet. There are several other functions associated with each of these pins, which include PWM outputs, LCD analog outputs, Capacitive Sensing inputs, and Comparator inputs.
- PORTC pins cannot be used as analog inputs. There are several other peripherals, however, some pins of which are multiplexed with this port. Examples include EUSART and I²C.
- PORTD pins cannot be used as analog input channels, but because they may be configured to be Capacitive Sensing inputs, they also have ANSELD bits associated with them. Each bit of ANSELD<7:0> corresponds to RD0-7, and is name ANSD0-ANSD7. Again, some of PORTD pins may be used for tasks such as PWM output, LCD analog output, or both.
- RE0-RE2 pins have the same situation as RA0-RA3 pins. A similar register, ANSELE in Bank3 also controls PORTE pins’ analog state. ANSELE<2:0>, which are named ANSE0-ANSE2, correspond to RE0-RE2.
- RE3 can also be used a general purpose input (no output).

The large number of available I/O port pins brings considerable flexibility into the project design. For example, assume that you want to design a remote combination-keypad lock for controlling access to a room. It needs to be battery-powered, so low power is a primary consideration. You also want to sound an alarm if the lock is tampered with or opened. You do not want to store combinations in the box. Instead, you would like to manage them from a central computer, so you intend to send the combination to a remote computer to authenticate the person and log access. You might want the following pins for your tasks:

- You could use RB7-1 for a 3x4 keypad. Using RB7-4's weak pullups and interrupt-on-change capability, you can put the CPU in ultra-low-power SLEEP mode when it is not in use for longer battery life.
- Use RB0 for a tamper switch to wake up the PIC, sound an alarm, and send a message to the central computer.
- Use RA4 for a multi-drop, shared serial bus along with any number of other units to send a serial data stream back to the central computer on one signal wire (plus ground, of course).
- Use RA2 and RA3 for status LED's and/or a piezo-beeper to give the user feedback when keys are pressed, blink red or green to indicate whether he or she has used a valid password, etc.

You are still left with plenty of unused pins. The code will easily fit into 1K word program memory with room to spare. All keypad debouncing and serial communication functions can be handled in software, with no external devices required. And you can run the whole thing on a few AA batteries. The CPU is happy with anything from 2 to 6 Volts. You could even store passwords locally in EEPROM, where they would be safe from power failures.

7.5.5 PIC18 I/O Ports

The PIC18F4620 has 5 ports and up to a total of 35 I/O pins, 8 pins for each of PORTA, PORTB, PORTC, and PORTD (8-bit wide), and 3 pins for PORTE (3-bit wide). Each port has 3 registers associated with its operation. These are:

- TRIS register (Data Direction Register) (1 = input, 0 = output)
- PORT register (level of the pins on the device)
- LAT register (output latch)

The PORT register should be used to read an input pin. The LAT register should be used when reading or writing to output pins.

It's important to note the electrical limitations of the I/O ports. For the PIC18F4620, the absolute maximum output or input pin current is 25 mA. However, the total current sunk or sourced by ALL pin must not exceed 200 mA.

CLRF	PORTA	; Initialize PORTA by clearing ; output
CLRF	LATA	; Alternate method to clear output
MOVLW	07H	; CONFIGURE A/D
MOVWF	ADCON1	; for digital inputs
MOVWF	07H	; Configure comparators
MOVWF	CMCON	; for digital input
MOVLW	0xCF	; Value used to initialize data ; direction
MOVWF	TRISA	; Set RA<7:6,3:0> as inputs ; RA<5:4> as outputs

Almost all pins of the I/O ports are multiplexed with alternate functions for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin. Some important mode multiplexing for the PIC18F4620 are indicated below:

- RA4 multiplexed with the Timer0 module clock input and a comparator output.
- RA6 and RA6 are multiplexed with the main oscillator pins. They can be enabled as I/O pins by selecting the oscillator in the configuration register.
- RA3:RA0 and RA5 are multiplexed with analog inputs, the analog Vref+ and Vref-, and the comparator voltage reference output. On a Power-on Reset, these pins are configured as analog inputs and read as '0'. To set these pins to digital I/O, they must be configured through the **ADCON1** register. To set them as digital inputs, the comparators must be turned off by setting the **CMCON** register. The **TRISA** register controls the direction of the RA pins even when they are used as analog inputs. They must always be correctly set to ensure functionality. The sample code in Figure X shows the process of initializing PORTA.
- PORTB pins have weak internal pull-ups that can be enabled or disabled by the **RBPU** bit in the **INTCON2** register.
- PORTB pins 4 through 7 can cause an interrupt-on-change if the **RBIE** bit in the **INTCON** register is enabled. Only pins configured as inputs can cause this interrupt to occur. The input pins of RB4:RB7 are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB4:RB7 are OR'ed together to generate the RB Port Change Interrupt with flag bit **RBIF** in the **INTCON** register. This interrupt can also wake the processor from **SLEEP** mode. The user, in the interrupt Service Routine, can clear the interrupt through the following: a) any read or write of PORTB (except for the **movff** instruction); b) Wait 1 instruction cycle; c) clear flag bit **RBIF**. Polling of PORTB is not recommended while using the interrupt-on-change feature.
- RB2:RB0 can be used for external interrupt input pins. They are configured using the **INTEDG2:0** bits in the **INTCON2** register.
- RB4:RB0 are configured to be analog inputs on Power-on Reset by default. This can be changed to digital inputs by programming the configuration bit, **PBADEN**.
- RB3 can be configured as the alternate peripheral pin for the CCP2 module by the configuration bit **CCP2MX**.
- All RC, RD, and RE pins have Schmitt Trigger input buffers.
- RD7:RD5 are multiplexed with outputs for the Enhanced CCP module.
- RE0-RE2 pins are also configured as analog inputs on Power-on Reset. The **PCFG3:PCFG0** bits on the **ADCON1** register controls which pins are set to digital or analog inputs. A **PCFG3:PCFG0** value of '0000' sets all A/D module pins to analog inputs and '1111' sets the pins to digital I/O.

7.5.6 Timers

7.5.6.1 PIC16 Timer Module

The PIC16F877 and PIC16F887 chip have three timer modules, TIMER0, TIMER1, and TIMER2. In addition to these 3, PIC16F1937 also has 2 other timers, TIMER4 and TIMER6. Timer0 module consists of one 8-bit register TMR0. It can be set either in timer mode or counter mode. In timer mode, it will typically increment every instruction cycle, and in counter mode, the module will increment either on every rising or falling edge of pin RA4/T0CKI. The mode is selected by clearing bit T0CS (OPTION_REG<5>), and the increment edge for the counter is determined by the bit T0SE (OPTION_REG<4>). Clearing bit T0SE selects the rising edge. The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). Bit T0IF must be cleared in software by TIMER0 module Interrupt Service Routine before re-enabling this interrupt.

TIMER1 module is a 16-bit timer/counter consisting of two 8-bit registers, TMR1H and TMR1L. It can be enabled/disabled by setting/clearing bit TMR1ON (T1CON<0>). Similar to TIMER1, TIMER2 can operate in either timer mode or counter mode. The operating mode is determined by the clock set bit, TMR1CS (T1CON<1>). Clearing this bit will set the timer mode. The counter mode operation can be either Synchronous or Asynchronous. In the counter mode, the timer increments on every rising edge of clock input on pin RC1/T1OSO/T1CK. If the control bit T1SYNCl is cleared, the external clock input is synchronized with internal phase clocks. If the control bit T1SYNC (T1CON<2>) is set, the external clock input is not synchronized, and the counter continues to increment asynchronous to the internal phase clocks.

For PIC16F887 and PIC16F1937, the TIMER1 has gate control. That is, it can be configured to be the T1G pin (RB5), or the output of the comparator C2. This allows for directly timing external events by T1G pin, or analog events using comparator C2. In order to control this additional feature, the T1CON register PIC16F887 has two more meaningful bits compared to PIC16F877: Bit 6 (TMR1GE) selects if Timer1 is gated, and Bit 7 (T1GINV) changes the T1G pin from being active-high to active-low. These two bits are unimplemented in PIC16F877. For PIC16F1937, there is an entire register dedicated for the gate control, because the input clock is selectable. Configuration of the TIMER1 gate in PIC16F1937, therefore, requires enabling the timer by setting TMR1ON of T1CON, and setting TMR1GE of T1GCON registers. The remaining bits of T1GCON are used for selecting the source and the mode of the Timer1 Gate.

TIMER2 has an 8-bit register TMR2. It can be used as the PWM time-base for the PWM mode of the CCP module. This module can be shut off by clearing control bit TMR2ON (T2CON<2>), to minimize power consumption.

TIMER4 and TIMER6, which exist in PIC16F1937 only, are identical to TIMER2 in operation.

7.5.6.2 PIC18 Timer Module

The PIC18F4620 has four timer modules, TIMER3:TIMER0. The Timer0 module can be configured to be an 8-bit as well as a 16-bit timer or counter. It consists of 3 registers, T0CON, TMR0L and TMR0H. T0CON is the control register for the timer and TMR0H:TMR0L are the data registers. 8-bit mode is selected by setting the T08BIT (T0CON<6>). In this mode, only TMR0L is used as the timer or counter. This module operates as a timer or a counter based on the clock source. In timer mode, the clock source is the internal instruction cycle clock. In counter mode, the clock source is the RA4/T0CKI pin. The clock source is selected by the Clock Source Select Bit, T0CS (T0CON<5>). Counter mode is enabled by setting the T0CS bit, and timer mode is enabled by clearing this bit. The timer/counter is turned on by setting the TMR0ON bit (T0CON<7>). An 8-bit prescaler is available for the Timer0 module, which can extend its increment time period by 2 through 256 in powers of 2 increments. The prescaler is assigned by clearing the PSA bit (T0CON<3>) and the prescaler value is determined by T0PS2:T0PS0 (T0CON<2:0>), with '111' being 1:256 and '000' being 1:2. In counter mode, the user can select the increment on the rising or the falling edge of the RA4 pin. Rising edge increment is selected by clearing the T0SE bit (T0CON<4>) and falling edge increment is selected by setting T0SE.

When operating in 16-bit mode, TMR0H is used for the upper 8 bits of the timer. TMR0H is a buffered version of the real high byte of Timer0, which is not directly readable or writable. When TMR0L is read, TMR0H is updated with the high byte of Timer0. This ensures the synchronization of the 2 bytes, which must be read separately. Data can be written to TMR0H, but is temporarily stored until a write occurs in the TMR0L register, at which time both values are simultaneously updated to TMR0. A TMR0 interrupt is generated when the TMR0 register overflows from 0xFF to 0x00

in 8-bit mode or from 0xFFFF to 0x0000 in 16-bit mode. This interrupt is enabled by setting the TMR0IE bit (INTCON<5>). The overflow sets the flag TMR0IF (INTCON<2>), which must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

The TIMER1 module is a 16-bit timer/counter with two 8-bit registers, TMR1H:TMR1L. It incorporates a selectable clock source and an additional Timer1 oscillator mode. It incorporates a low power on-chip crystal oscillator circuit suitable for 32 kHz crystals. This mode provides an extra clocking option when running in low-power modes. Read and write procedures are the same as the 16-bit mode on the Timer0 module. An overflow interrupt is generated when Timer1 overflows from 0xFFFF to 0x0000. This will set the Timer1 interrupt flag, TMR1IF (PIR1<0>), which must be cleared in software. The interrupt is enabled by setting the TMR1IE bit (PIE1<0>). Timer1 can operate in synchronous or asynchronous mode when the clock source is set to an external clock by setting the TMR1CS (T1CON<1>). If T1SYNC (T1CON<2>) is cleared, the external clock is synchronized with the internal phase clock; if set, the external clock is run asynchronous to the internal clock.

TIMER2 has an 8-bit register TMR2 as well as an 8-bit register PR2. It can be used as the PWM time-base for the PWM mode of the CCP module. Unlike the other timers, Timer2 generates an overflow interrupt when its value matches the value of PR2, which is default to 0xFF but is readable and writable. Timer2 provides up to 1:16 prescaler as well as up to 1:16 postscaler, which, together with the PR2 register, can provide flexible timing periods. Timer3 is a 16-bit timer/counter that is almost identical in operation to Timer1.

7.6 Programming

It is a usual exercise in microcontrollers to devise an assembly language, which uses mnemonic abbreviations and symbolic names for functions and memory locations and variables, instead of having to write the binary machine codes (opcodes) directly. Writing a program using the shorthand mnemonic codes is easier, since they are an abbreviated version of the operation performed by the instruction. Further, because the instructions describe the program operations and can easily be comprehended, they are less error prone than the binary pattern of machine code programming. Nevertheless, the assembly code must be compiled and translated to the machine code, which is the only language the microcontroller can recognize. This task can be performed either manually using the microcontroller manufacturer's data sheets or automatically using an *assembler* program. To make the programming even easier, a compiled or interpreted high-level language such as BASIC, FORTRAN, PASCAL, or C can also be used. Using these languages, there is no concern with addressing memory locations or other low-level tasks. However, after compilation or interpretation of these codes into the binary machine code, the resulting opcode is usually more redundant, and hence less efficient, than the one that is translated directly from an assembly code.

7.6.1 MPASM Assembler

Microchip provides a universal macro assembler for all PICmicro™ microcontrollers, called MPASM. Note that this name is also used for a DOS- or Windows-based PC application that provides a platform for developing and compiling assembly codes for PICmicro™ MCU's. First, A program code (such as *code.asm*) is written in the assembly language using the MPASM mnemonics, and then it is compiled by the MPASM compiler into the *code.hex* code ready to be loaded on the PIC device through a programmer. MPASM can also generate an object module, *code.o*, which can be linked with other modules using the MPLINK linker to form the final hex code. This is useful for creating reusable modules that do not have to be re-tested each time they are used. Related modules can also be grouped and stored together in a library using the MPLIB librarian. Required libraries can then be specified during the linking process, and only those routines that are needed will be included in the final hex code.

7.6.2 Source Code Format

The format of the source code is shown below. The maximum column width is 255 characters.

Label	Mnemonic	Operand	;	Comment
	LIST	P=16F84A		; set processor
	ORG	0		; pc=0
	CLRF	OC		; counter=0
LOOP	INCF	0C,1		; increment counter
	GOTO	LOOP		; goto loop
	END			

You can make one line only a comment, too. Generally, it is a good practice to use the TAB to separate the position of the label, the mnemonic, the operand, and the comment, as shown below:

Labels

- A label must start in column 1.
- All labels within a program must be unique. If there is no label then a space must be included in the label field.
- It may be followed by a colon (:), space, tab or the end of line.
- Labels must begin with an alpha character or an under bar (_).
- Labels may contain alphanumeric characters, the under bar and question mark.
- Labels may be up to 32 characters long.
- By default, labels are case sensitive, but case sensitivity may be overridden by a command line option.
- The label must not use any of the names reserved for registers or instruction codes. For example, since "B" becomes the reserved word that shows "Binary", it is not allowed to use it for labels.

Mnemonics

- Assembler instruction mnemonics, assembler directives and macro calls must begin in column two or greater.
- If there is a label on the same line, instructions must be separated from that label by a colon, or by one more spaces or tabs. Generally, the position of the instructions is arranged for using the tab.
- The only characters that are permitted to use are alpha characters and under bar.

Operands

- Operands give the address of the data to be operated on by the process specified by the mnemonics.
- Operands must be separated from mnemonics by one more spaces, or tabs.
- This field may remain empty if the instruction does not need any data or address.
- Multiple operands must be separated by commas. Spaces behind the commas are allowed to make it more visible.
- Numerical data in this field may be hexadecimal, decimal, octal, or binary. The default is hexadecimal.

Comments

- MPASM treats anything after a semicolon as a comment.
- All characters following the semicolon are ignored through the end of the line.
- String constants containing a semicolon are allowed, and are not confused with comments.
- Sometimes it is desired to comment out a block of code. This can be done very quickly by selecting the block of code in the project window, right click on the selected code, and go to Advanced->Comment Block. Uncomment block can be selected in the same drop down menu.

7.6.3 PIC16F877/887 Instruction Set

The PIC16F877 and PIC16F887 devices recognize 35 instructions to process the desired operations. Each instruction is carried by a 14-bit word, which is divided into a so-called OPCODE for specifying the instruction type and one or more operands for detailing the operation. Instructions can be classified into three categories of *byte-oriented* (to process a byte unit), *bit-oriented* (to process a bit unit), and *literal and control* (to process the fix numbers) instructions. General format of instructions of different kind is shown in Figure 7.6-1. For byte-oriented instructions, the field "f" represents a file register designator that specifies which file register is to be used by the instruction. The "d" field represents the destination designator that specifies where the result of the operation is to be placed; if "0" the result is placed in the W (Working) register (accumulator) and if "1" the result is placed in the file register specified in the "f" field of the instruction. For bit-oriented instruction, the "b" field represents a bit field designator that selects the number of the bit affected by the operation, and the "f" field represents the address of the file in which the bit is located. For literal and control instructions, the field "k" represents an 8- or 11-bit constant or literal value.

All instructions are executed within one single instruction cycle (T_{CY}), unless a conditional test is TRUE or the Program Counter (PC) is changed as a result of an instruction. In these particular cases, the execution takes two instruction cycles. Each instruction cycle (T_{CY}) is comprised of four oscillator periods, or Q cycles (Q1-Q4). The Q cycle time is the same as the device oscillator cycle time (T_{OSC}). For example, for an oscillator frequency of 20 MHz, the normal instruction execution time is 200 ns. The Q cycles provide the timing/designation for the Decode, Read, Process Data, Write, etc., of each instruction cycle. The four Q cycles that make up an instruction cycle (T_{CY}) can be generalized as:

- Q1: Instruction Decode Cycle or forced No operation;
- Q2: Instruction Read Data Cycle or No operation;
- Q3: Process the Data;
- Q4: Instruction Write Data Cycle or No operation.

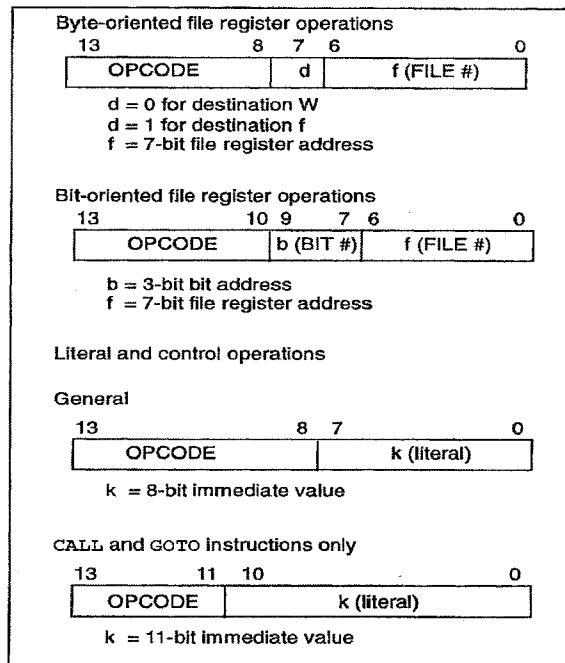


Figure 7.6-1

7.6.3.1 Mnemonics

A list of mnemonics for programming PIC16F877 and PIC16F887 is shown below. The byte-oriented instructions that require two parameters [e.g., `xorwf f,d`] expect the `f` to be replaced by the name of a special-purpose register (e.g. PORTA) or the name of a RAM variable (e.g., TEMPI), which serves as the *source* of the operand. The `d` parameter is the destination of the result of the operation. It should be replaced by: `F` if the destination is to be the source register; or `W` if the destination is to be the working register. For example, if `W` has been loaded with `B'00000001'`, then

`xorwf PORTA, F`

will toggle the least significant bit of port A (which presumably has been set up as an output line).

Mnemonic, Operands	Description	Cycles	14-Bit Opcode		Status Affected	Notes
			MSb	LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff	Z	1,2,3
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff	Z	1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSZ	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWD	-	Clear Watchdog Timer	1	00 0000 0110 0100	TO,PD	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 0xxx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	TO,PD	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, `d = 1`), the prescaler will be cleared if assigned to the Timer0 module.

3: If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

The bit-oriented instructions also expect two parameters (e.g., **btfss f,b**). Again **f** is to be replaced by the name of a special-purpose register or the name of a RAM variable. The **b** parameter is to be replaced by a bit number ranging from 0 to 7, or by a label that has been equated to such a number. For example, if the assembler has been told that

```
Z      equ      D'2'
```

that is, that **Z** is a label representing the number 2, then the instruction

```
btfss  STATUS, Z
```

will test the **Z** bit of the **STATUS** register (since the **Z** bit is bit 2 of the **STATUS** register) and will skip over the next instruction if the **Z** bit is set.

The literal instructions require an operand having a known value (e.g., **H'0F**) or a label that represents a known value. For example, if

```
MASK  equ      H'0F'
```

then

```
andlw  MASK
```

will force the upper bits of **W** to zero.

Every instruction fits in a single 14-bit word. As discussed earlier, this design decision was made to support a simple pipelining scheme. In addition, every instruction executes in a single cycle unless it changes the content of the program counter. Thus, each of the four instructions for which the "Cycles" are listed as 1(2) takes one cycle if the tested condition is not met and two cycles otherwise. For example, the instruction

```
btfss  PORTA, 2
```

will execute in two cycles if the tested bit is indeed set, resulting in an extra increment of the program counter to skip over the next instruction. If the tested bit is not set, then the instruction will execute in just one cycle, with the program counter incrementing to the next instruction.

Several examples are given below to illustrate the PIC Mnemonics:

Single-bit manipulation

```
bcf    PORTB, 0      ;Clear bit 0 of PORTB
bsf    STATUS, C      ;Set the carry bit
```

Clear/move

```
clrw      ;Clear the working register, W
clrf    TEMP1      ;Clear temporary variable TEMP1
movlw  5          ;Load 5 into W
movlw  10         ;Load D'10' or H'10' or B'10' into W
                  ;depending upon default representation
movwf  TEMP1      ;Move W into TEMP1
movwf  TEMP1, F    ;Incorrect syntax
movf   TEMP1, W    ;Move TEMP1 into W
swapf  TEMP1, F    ;Swap 4-bit nibbles of TEMP1
swapf  TEMP1, W    ;Move TEMP1 to W, swapping nibbles
                  ;and leave TEMP1 unchanged
```

Increment/decrement/complement

```
incf   TEMP1, F      ;Increment TEMP1
incf   TEMP1, W      ;W <- TEMP1 + 1; TEMP1 unchanged
decf   TEMP1, F      ;Decrement TEMP1
comf   TEMP1, F      ;Change 0s to 1s and 1s to 0s
```

Multiple-bit manipulation

```
andlw  B'00000111'  ;Force upper 5 bits of W to zero
andwf  TEMP1, F      ;TEMP1 <- TEMP1 AND W
andwf  TEMP1, W      ;W <- TEMP1 AND W
iorlw  B'00000111'  ;Force lower 3 bits of W to one
iorwf  TEMP1, F      ;TEMP1 <- TEMP1 OR W
xorlw  B'00000111'  ;Complement lower 3 bits of W
xorwf  TEMP1, W      ;W <- TEMP1 XOR W
```

Addition/subtraction

```
addlw 5      ;Add 5 to W
addwf TEMP1,F ;TEMP1 <- TEMP1 + W
sublw 5      ;W <- 5 - W (not W <- W - 5!)
subwf TEMP1,F ;TEMP1 <- TEMP1 - W
```

Rotate

```
rlf TEMP1,F  ;Nine-bit left rotate through C
               ;(C <- TEMP1,7 ; TEMP1,i+1 <- TEMP1,i
               ;TEMP1,0 <- C)
rrf TEMP1,W   ;Leave TEMP1 unchanged
               ;copy to W and rotate W right through C
```

Conditional branch

```
btfsc TEMP1,0 ;Skip the next instruction if bit 0 of
               ;TEMP1 equals zero
btfss STATUS,C ;Skip if C = 1
decfsz TEMP1,F ;Decrement TEMP1; skip if zero
incfsz TEMP1,W ;Leave TEMP1 unchanged; skip if
               ;TEMP1 = H'FF'; W <- TEMP1 + 1
```

Got/call/return/return from interrupt

```
goto There    ;Next instruction to be executed is
               ;labeled "There"
call Task1    ;Push return address; next instruction
               ;to be executed is labeled "Task1"
return        ;Pop return address off of stack
retlw 5       ;Pop return address; W <- 5
retfie        ;Pop return address; reenable interrupts
```

Miscellaneous

```
clrwdt      ;If watchdog timer is enabled, this
               ;instruction will reset it (before it
               ;resets , the CPU)
sleep        ;Stop clock: I reduce power; wait
               ;for watchdog timer or external signal
               ;to begin program execution again
nop         ;Do nothing; wait one clock cycle
```

7.6.4 PIC16F1937 Instruction Set

The PIC16F1937 device recognizes 49 instructions to process the desired operations. Each instruction is carried by a 14-bit word, which is divided into a so-called OPCODE for specifying the instruction type and one or more operands for detailing the operation. Instructions can be classified into three categories of *byte-oriented* (to process a byte unit), *bit-oriented* (to process a bit unit), and *literal and control* (to process the fix numbers) instructions. General format of instructions of different kind is shown in Figure 7.6-12. For byte-oriented instructions, the field "f" represents a file register designator that specifies which file register is to be used by the instruction. The "d" field represents the destination designator that specifies where the result of the operation is to be placed; if "0" the result is placed in the W (Working) register (accumulator) and if "1" the result is placed in the file register specified in the "f" field of the instruction. For bit-oriented instruction, the "b" field represents a bit field designator that selects the number of the bit affected by the operation, and the "f" field represents the address of the file in which the bit is located. For literal and control instructions, the field "k" represents an 8- or 11-bit constant or literal value.

Byte-oriented file register operations							
13 8 7 6 0							
OPCODE	d	f (FILE #)					
d = 0 for destination W							
d = 1 for destination f							
f = 7-bit file register address							
Bit-oriented file register operations							
13 10 9 7 6 0							
OPCODE	b (BIT #)	f (FILE #)					
b = 3-bit bit address							
f = 7-bit file register address							
Literal and control operations							
General							
13 8 7 0							
OPCODE		k (literal)					
k = 8-bit immediate value							
CALL and GOTO instructions only							
13 11 10 0							
OPCODE		k (literal)					
k = 11-bit immediate value							
MOVL instruction only							
13 7 6 0							
OPCODE		k (literal)					
k = 7-bit immediate value							
BRA instruction only							
13 9 8 0							
OPCODE		k (literal)					
k = 9-bit immediate value							
FSR Offset instructions							
13 7 6 5 0							
OPCODE	n	k (literal)					
n = appropriate FSR							
k = 6-bit immediate value							
FSR Increment instructions							
13 3 2 1 0							
OPCODE	n	m (mode)					
n = appropriate FSR							
m = 2-bit mode value							
Opcode only							
13 0							
OPCODE							

7.6.4.1 Mnemonics

A list of mnemonics for programming PIC16F1937 is shown below. The byte-oriented instructions that require two parameters [e.g., **xorwf f,d**] expect the **f** to be replaced by the name of a special-purpose register (e.g. PORTA) or the name of a RAM variable (e.g., TEMP1), which serves as the *source* of the operand. The **d** parameter is the destination of the result of the operation. It should be replaced by: **F** if the destination is to be the source register; or **W** if the destination is to be the working register. For example, if **W** has been loaded with B'00000001', then

xorwf PORTA, F

will toggle the least significant bit of port A (which presumably has been set up as an output line).

Note that many of these instructions are identical to those of PIC16F877/887, and the only difference is the additional 14 instructions which complete the enhanced instruction set.

Figure 7.6-2

Mnemonic, Operands	Description	Cycles	14-Bit Opcode				Status Affected	Notes	
			MSb	LSb					
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF f, d	Add W and f	1	00	0111	ffff	ffff	C, DC, Z	2	
ADDWFC f, d	Add with Carry W and f	1	11	1101	ffff	ffff	C, DC, Z	2	
ANDWF f, d	AND W with f	1	00	0101	ffff	ffff	Z	2	
ASRF f, d	Arithmetic Right Shift	1	11	0111	ffff	ffff	C, Z	2	
LSLF f, d	Logical Left Shift	1	11	0101	ffff	ffff	C, Z	2	
LSRF f, d	Logical Right Shift	1	11	0110	ffff	ffff	C, Z	2	
CLRF f	Clear f	1	00	0001	ffff	ffff	Z	2	
CLRW -	Clear W	1	00	0001	0000	00xx	Z		
COMF f, d	Complement f	1	00	1001	ffff	ffff	Z	2	
DECf f, d	Decrement f	1	00	0011	ffff	ffff	Z	2	
INCF f, d	Increment f	1	00	1010	ffff	ffff	Z	2	
IORWF f, d	Inclusive OR W with f	1	00	0100	ffff	ffff	Z	2	
MOVF f, d	Move f	1	00	1000	ffff	ffff	Z	2	
MOVWF f	Move W to f	1	00	0000	ffff	ffff		2	
RLF f, d	Rotate Left f through Carry	1	00	1101	ffff	ffff	C	2	
RRF f, d	Rotate Right f through Carry	1	00	1100	ffff	ffff	C	2	
SUBWF f, d	Subtract W from f	1	00	0010	ffff	ffff	C, DC, Z	2	
SUBWFB f, d	Subtract with Borrow W from f	1	11	1011	ffff	ffff	C, DC, Z	2	
SWAPF f, d	Swap nibbles in f	1	00	1110	ffff	ffff		2	
XORWF f, d	Exclusive OR W with f	1	00	0110	ffff	ffff	Z	2	
BYTE ORIENTED SKIP OPERATIONS									
DECFSZ f, d	Decrement f, Skip if 0	1(2)	00	1011	ffff	ffff		1,2	
INCFSZ f, d	Increment f, Skip if 0	1(2)	00	1111	ffff	ffff		1,2	
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF f, b	Bit Clear f	1	01	00bb	bfff	ffff		2	
BSF f, b	Bit Set f	1	01	01bb	bfff	ffff		2	
BIT-ORIENTED SKIP OPERATIONS									
BTFSCL f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		1,2	
BTFSST f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		1,2	
BIT-ORIENTED SKIP OPERATIONS									
BTFSCL f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		1,2	
BTFSST f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		1,2	
LITERAL OPERATIONS									
ADDLW k	Add literal and W	1	11	1110	kkkk	kkkk	C, DC, Z		
ANDLW k	AND literal with W	1	11	1001	kkkk	kkkk	Z		
IORLW k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z		
MOVLB k	Move literal to BSR	1	00	0000	001k	kkkk			
MOVLP k	Move literal to PCLATH	1	11	0001	1kkk	kkkk			
MOV LW k	Move literal to W	1	11	0000	kkkk	kkkk			
SUBLW k	Subtract W from literal	1	11	1100	kkkk	kkkk	C, DC, Z		
XORLW k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z		

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

Several examples are given below to illustrate the PIC Mnemonics:

Mnemonic, Operands	Description	Cycles	14-Bit Opcode		Status Affected	Notes
			MSb	LSb		
CONTROL OPERATIONS						
BRA k	Relative Branch	2	11	001k kkkk kkkk		
BRW -	Relative Branch with W	2	00	0000 0000 1011		
CALL k	Call Subroutine	2	10	0kkk kkkk kkkk		
CALLW -	Call Subroutine with W	2	00	0000 0000 1010		
GOTO k	Go to address	2	10	1kkk kkkk kkkk		
RETFIE k	Return from interrupt	2	00	0000 0000 1001		
RETLW k	Return with literal in W	2	11	0100 kkkk kkkk		
RETURN -	Return from Subroutine	2	00	0000 0000 1000		
INHERENT OPERATIONS						
CLRWDT -	Clear Watchdog Timer	1	00	0000 0110 0100	TO, PD	
NOP -	No Operation	1	00	0000 0000 0000		
OPTION -	Load OPTION_REG register with W	1	00	0000 0110 0010		
RESET -	Software device Reset	1	00	0000 0000 0001		
SLEEP -	Go into Standby mode	1	00	0000 0110 0011		
TRIS f	Load TRIS register with W	1	00	0000 0110 0fff	TO, PD	
C-COMPILER OPTIMIZED						
ADDFSR n, k	Add Literal k to FSRn	1	11	0001 0nkk kkkk	Z	2, 3
MOVIW n mm	Move Indirect FSRn to W with pre/post inc/dec modifier, mm	1	00	0000 0001 0nmm		
MOVWI n mm	Move W to Indirect FSRn with pre/post inc/dec modifier, mm	1	00	0000 0001 1nmm		
MOVWI k[n]	Move W to INDFn, Indexed Indirect	1	11	1111 0nkk kkkk	Z	2
MOVWI k[n]	Move W to INDFn, Indexed Indirect	1	00	0000 0001 1nmm		2, 3

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

- 2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.
- 3: See Table in the MOVIW and MOVWI instruction descriptions.

Single-bit manipulation

```
bcf    PORTB, 0      ;Clear bit 0 of PORTB
bsf    STATUS, C      ;Set the carry bit
```

Clear/move

```
clrw      ;Clear the working register, W
clrf    TEMP1      ;Clear temporary variable TEMP1
movlb    5          ;Load 5 into BSR
movlp    5          ;Load 5 into PCLATH
moviw    0 [INDF]   ;Move value pointed to by FSR into W (NOTE1)
movwi    -1[INDF]   ;Move W into next byte of value pointed to by FSR (NOTE1)
movlw    5          ;Load 5 into W
movlw    10         ;Load D'10' or H'10' or B'10' into W
                   ;depending upon default representation
movwf    TEMP1      ;Move W into TEMP1
movwf    TEMP1, F   ;Incorrect syntax
movf    TEMP1, W    ;Move TEMP1 into W
swapf   TEMP1, F    ;Swap 4-bit nibbles of TEMP1
swapf   TEMP1, W    ;Move TEMP1 to W, swapping nibbles
                   ;and leave TEMP1 unchanged
```

Increment/decrement/complement

```
incf    TEMP1, F    ;Increment TEMP1
incf    TEMP1, W    ;W <- TEMP1 + 1; TEMP1 unchanged
decf    TEMP1, F    ;Decrement TEMP1
comf    TEMP1, F    ;Change 0s to 1s and 1s to 0s
```

Multiple-bit manipulation

```
andlw   B'00000111' ;Force upper 5 bits of W to zero
```

```

andwf TEMP1,F      ;TEMP1 <- TEMP1 AND W
andwf TEMP1,W      ;W <- TEMP1 AND W
iorlw B'00000111'   ;Force lower 3 bits of W to one
iorwf TEMP1,F      ;TEMP1 <- TEMP1 OR W
xorlw B'00000111'   ;Complement lower 3 bits of W
xorwf TEMP1,W      ;W <- TEMP1 XOR W

```

Addition/subtraction

```

Addfsr 0,5          ;Add 5 to FSR0
Addwfc TEMP1,F      ;TEMP1 <- TEMP1 + W + CarryBit
addlw 5              ;Add 5 to W
addwf TEMP1,F      ;TEMP1 <- TEMP1 + W
subwfb TEMP1,F      ;TEMP1 <- TEMP1 - W - BorrowBit
sublw 5              ;W <- 5 - W (not W <- W - 5!)
subwf TEMP1,F      ;TEMP1 <- TEMP1 - W

```

Shift and rotate

```

lsrf TEMP1,F        ;Logical shift to the right through C
                     ;(C <- TEMP1,0 ; TEMP1,i <- TEMP1,i+1
                     ;TEMP1,7 <- 0)
asrf TEMP1,W        ;Arithmetic shift to the right through C
                     ;(C <- TEMP1,0 ; W,i <- TEMP1,i+1
                     ;W,7 <- TEMP1,7)
Lslf TEMP1,F        ;Logical shift to the left through C
                     ;(C <- TEMP1,7 ; TEMP1,i+1 <- TEMP1, i
                     ;TEMP1,0 <- 0)
rlf TEMP1,F         ;Nine-bit left rotate through C
                     ;(C <- TEMP1,7 ; TEMP1,i+1 <- TEMP1,i
                     ;TEMP1,0 <- C)
rrf TEMP1,W         ;Leave TEMP1 unchanged
                     ;copy to W and rotate W right through C

```

Relative branch

```

bra There            ;Relative branch to the label There
bra d'2'             ;PC <- PC + 2 (new address will be PC + 2 + 1)
brw                 ;PC <- PC + W (new address will be PC + W + 1)

```

Conditional branch

```

btfsC TEMP1,0        ;Skip the next instruction if bit 0 of
                     ;TEMP1 equals zero
btfsS STATUS,C       ;Skip if C = 1
decfsz TEMP1,F       ;Decrement TEMP1; skip if zero
incfsz TEMP1,W       ;Leave TEMP1 unchanged; skip if
                     ;TEMP1 = H'FF'; W <- TEMP1 + 1

```

Goto/call/return/return from interrupt

```

goto There           ;Next instruction to be executed is
                     ;labeled "There"
callw               ;Push return address (PC+1) on the return stack,
                     ;load W into PC<7:0>, PCLATH into PC<14:8>
call Task1           ;Push return address; next instruction
                     ;to be executed is labeled "Task1"
return              ;Pop return address off of stack
retlw 5              ;Pop return address; W <- 5
retfie               ;Pop return address; reenable interrupts

```

Miscellaneous

```

reset               ;Reset the CPU
clrwdt              ;If watchdog timer is enabled, this
                     ;instruction will reset it (before it
                     ;resets the CPU)
sleep               ;Stop clock: I reduce power; wait
                     ;for watchdog timer or external signal
                     ;to begin program execution again
nop                 ;Do nothing; wait one clock cycle

```

NOTE1: For MOVWI and MOVIW instructions, use the example below:

MOVIW	<code>++FSR0</code>	<code>; Preincrement FSR0 then INDF0 -> W</code>
MOVIW	<code>FSR0++</code>	<code>; INDF0 -> W then postincrement FSR0</code>
MOVIW	<code>--FSR0</code>	<code>; Predecrement FSR0 then INDF0 -> W</code>
MOVIW	<code>FSR0--</code>	<code>; INDF0 -> W then postdecrement FSR0</code>
MOVIW	<code>4[FSR0]</code>	<code>; FSR0+4 INDF0 -> W. FSR0 unchanged</code>

7.6.5 PIC18 Instruction Set

The PIC18F4620 incorporates an instruction set of 75 instructions to carry out its functions. Each instruction is carried by a 16-bit word, consisting of an OPCODE and one or more operands. For compatibility purposes, all PIC16 instructions are valid PIC18 instructions except for the two rotate instructions. Instructions are classified into four categories: *byte-oriented* (processes a byte unit), *bit-oriented* (processes a bit unit), *literal* (processes fix literals), and *control* instructions. General format of instructions of different kind is shown in Figure 7.6-2. For byte-oriented instructions, the field "f" represents a file register designator that specifies which file register is to be used by the instruction. The "d" field represents the destination designator that specifies where the result of the operation is to be placed; if "0" the result is placed in the W (Working) register (accumulator) and if "1" the result is placed in the file register denoted by "f" in the instruction. For bit-oriented instruction, the "b" represents a bit field designator that selects the number of the bit affected by the operation, and the "f" represents the address of the file in which the bit is located. For literal instructions, the "k" represents an 8-bit constant or literal value. In control instructions, the operand denoted by "n" is an 8- or 11- or 20-bit address of a program memory location.

Byte-oriented file register operations	Example Instruction											
<p>15 10 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>d</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>d = 0 for result destination to be WREG register d = 1 for result destination to be file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address</p>	OPCODE	d	a	f (FILE #)	0	ADDWF MYREG, W, B						
OPCODE	d	a	f (FILE #)	0								
<p>Byte to Byte move operations (2-word)</p> <p>15 12 11 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">f (Source FILE #)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">f (Destination FILE #)</td> <td>0</td> </tr> </table> <p>f = 12-bit file register address</p>	OPCODE	f (Source FILE #)			0	1111	f (Destination FILE #)			0	MOVFF MYREG1, MYREG2	
OPCODE	f (Source FILE #)			0								
1111	f (Destination FILE #)			0								
<p>Bit-oriented file register operations</p> <p>15 12 11 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>b (BIT #)</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>b = 3-bit position of bit in file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address</p>	OPCODE	b (BIT #)	a	f (FILE #)	0	BSF MYREG, bit, B						
OPCODE	b (BIT #)	a	f (FILE #)	0								
<p>Literal operations</p> <p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">k (literal)</td> <td>0</td> </tr> </table> <p>k = 8-bit immediate value</p>	OPCODE	k (literal)			0	MOVlw 7Fh						
OPCODE	k (literal)			0								
<p>Control operations</p> <p>CALL, GOTO and Branch operations</p> <p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">n<7:0> (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n<19:8> (literal)</td> <td>0</td> </tr> </table> <p>n = 20-bit immediate value</p>	OPCODE	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	GOTO Label	
OPCODE	n<7:0> (literal)			0								
1111	n<19:8> (literal)			0								
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>S</td> <td colspan="3">n<7:0> (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n<19:8> (literal)</td> <td>0</td> </tr> </table> <p>S = Fast bit</p>	OPCODE	S	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	CALL MYFUNC
OPCODE	S	n<7:0> (literal)			0							
1111	n<19:8> (literal)			0								
<p>15 11 10 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">n<10:0> (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<10:0> (literal)			0	BRA MYFUNC						
OPCODE	n<10:0> (literal)			0								
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">n<7:0> (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<7:0> (literal)			0	BC MYFUNC						
OPCODE	n<7:0> (literal)			0								

Figure 7.6-2

The instruction execution sequence is identical to that of the PIC16. Each instruction takes 4 oscillator cycles to complete.

7.6.5.1 Mnemonics

A list of mnemonics for programming the PIC18F4620 is shown below.

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	
DECFSZ	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	
MOVFF	f _s , f _d	Move f _s (source) to 1st word f _d (destination) 2nd word	2	1100	ffff	ffff	ffff	None	
				1111	ffff	ffff	ffff		
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with Borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap Nibbles in f	1	0011	10da	ffff	ffff	None	
TSTFSZ	f, a	Test f, Skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	
LITERAL OPERATIONS									
ADDLW	k	Add Literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND Literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR Literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move Literal (12-bit) 2nd word to FSR(f) 1st word	2	1110	1110	00ff	kkkk	None	
				1111	0000	kkkk	kkkk		
MOVLB	k	Move Literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move Literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply Literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from Literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR Literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
BIT-ORIENTED OPERATIONS									
BCF f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2	
BSF f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2	
BTFSC f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4	
BTFSS f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4	
BTG f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2	
CONTROL OPERATIONS									
BC n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None		
BN n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None		
BNC n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None		
BNN n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None		
BNOV n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None		
BNZ n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None		
BOV n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None		
BRA n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None		
BZ n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None		
CALL n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
CLRWDT —	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD		
DAW —	Decimal Adjust WREG	1	0000	0000	0000	0111	C		
GOTO n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
NOP —	No Operation	1	0000	0000	0000	0000	None		
NOP —	No Operation	1	1111	xxxx	xxxx	xxxx	None	4	
POP —	Pop Top of Return Stack (TOS)	1	0000	0000	0000	0110	None		
PUSH —	Push Top of Return Stack (TOS)	1	0000	0000	0000	0101	None		
RCALL n	Relative Call	2	1101	1nnn	nnnn	nnnn	None		
RESET	Software Device Reset	1	0000	0000	1111	1111	All		
RETFIE s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL		
RETLW k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None		
RETURN s	Return from Subroutine	2	0000	0000	0001	001s	None		
SLEEP —	Go into Standby mode	1	0000	0000	0000	0011	TO, PD		
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*	Table Read	2	0000	0000	0000	1000	None		
TBLRD*+	Table Read with Post-Increment		0000	0000	0000	1001	None		
TBLRD*-	Table Read with Post-Decrement		0000	0000	0000	1010	None		
TBLRD+*	Table Read with Pre-Increment		0000	0000	0000	1011	None		
TBLWT*	Table Write	2	0000	0000	0000	1100	None		
TBLWT*+	Table Write with Post-Increment		0000	0000	0000	1101	None		
TBLWT*-	Table Write with Post-Decrement		0000	0000	0000	1110	None		
TBLWT+*	Table Write with Pre-Increment		0000	0000	0000	1111	None		

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

Most byte-oriented instructions require two parameters, and the third parameter, 'a', is optional. For a byte-oriented instruction, the 'f' should be replaced by the name of a special purpose register (e.g. PORTA) or the name or address of a RAM variable (e.g. TEMP1 or 0x70). This is the source data for the operation. The 'd' parameter is the destination of the result of the operation and should be replaced by either F for the source file register, or W for the working register. The 'a' is the RAM access bit, and should take the value 0 for the Access RAM or 1 for Banked RAM. If omitted, the 'a' parameter will take a default value of 1, which selects the Banked RAM.

A special byte-oriented instruction, movff, copies the contents of one file register into another without the use of the W register. This is the only byte-oriented instruction that always takes two cycles to complete.

The bit-oriented instructions also expect two to three parameters (e.g., btfss f,b,a). Again f is to be replaced by the name of a special-purpose register or the name of a RAM variable. The b parameter is to be replaced by a bit number ranging from 0 to 7, or by a label that has been equated to such a number.

All instructions except for 4 fit into a single 16-bit word. The four 2-word instructions are **movff**, **goto**, **call**, and **lfsr**. These instructions contain the full addresses of the program or data memory locations they use, so no bank switching is needed. For the **goto** and **call** instructions, the full 21-bit program memory address is included in the instruction.

Several examples are given below to illustrate the PIC Mnemonics:

Single-bit manipulation

```
bcf    LATB,0      ;Clear bit 0 of PORTB
bsf    STATUS,C    ;Set the carry bit
btg    H'50',3,0    ;toggle bit 3 of the register address H'50' in the access
RAM
```

Clear/Set/Move

```
movlb B'1010'      ;Set BSR to Bank 10
clrf TEMP1        ;Clear temporary variable TEMP1 (in current Bank)
movlw 5             ;Load 5 into W
movlw 10            ;Load D'10' or H'10' or B'10' into W
                   ;depending upon default representation
movwf TEMP1         ;Move W into TEMP1
movwf TEMP1,f      ;Incorrect syntax
movf  TEMP1,W       ;Move TEMP1 into W
movff  TEMP1,TEMP2   ;Move TEMP1 into TEMP2
swapf TEMP1,f      ;Swap 4-bit nibbles of TEMP1
swapf TEMP1,W       ;Move TEMP1 to W, swapping nibbles
                   ;and leave TEMP1 unchanged
setf  TEMP1,1        ;TEMP1 <- H'FF'
```

Increment/decrement/complement

```
incf  TEMP1,f      ;Increment TEMP1
incf  TEMP1,W       ;W <- TEMP1 + 1; TEMP1 unchanged
decf  TEMP1,f      ;Decrement TEMP1
comf  TEMP1,f      ;Change 0s to 1s and 1s to 0s
negf  TEMP1          ;TEMP1 <- -TEMP1 (2's complement)
```

Multiple-bit manipulation

```
andlw B'00000111'   ;Force upper 5 bits of W to zero
andwf TEMP1,f      ;TEMP1 <- TEMP1 AND W
andwf TEMP1,W       ;W <- TEMP1 AND W
iorlw B'00000111'   ;Force lower 3 bits of W to one
iorwf TEMP1,f      ;TEMP1 <- TEMP1 OR W
xorlw B'00000111'   ;Complement lower 3 bits of W
xorwf TEMP1,W       ;W <- TEMP1 XOR W
```

Addition/subtraction

```
addlw 5             ;Add 5 to W
addwf TEMP1,f      ;TEMP1 <- TEMP1 + W
addwfc TEMP1,f     ;TEMP1 <- TEMP1 + W + C
sublw 5             ;W <- 5 - W (not W <- W - 5!)
subwf TEMP1,f      ;TEMP1 <- TEMP1 - W
subwfb TEMP1,f     ;TEMP1 <- TEMP1 - W - C
subfwb TEMP1,W      ;W <- W - TEMP1 - C
                   ;Be Careful not to mix up subwfb and subfwb!
```

Rotate/Multiply

```
rlcf  TEMP1,F      ;Nine-bit left rotate through C
rlncf TEMP1,F      ;(C <- TEMP1,7 ; TEMP1,i+1 <- TEMP1,i
rrcf  TEMP1,W      ;TEMP1,0 <- C)
rrncf TEMP1,F      ;Rotate left (no C)
mullw 6            ;(TEMP1,0 <- TEMP1,7 ; TEMP1,i+1 <- TEMP1,i)
mulwf TEMP1         ;Leave TEMP1 unchanged
                     ;copy to W and rotate W right through C
                     ;rotate TEMP1 right, no C
                     ;PRODH:PRODL <- 6 X W
                     ;W remains unchanged
                     ;PRODH:PRODL <- TEMP1 X W
                     ;TEMP1 and W are unchanged
```

Conditional branch

```
btfsz TEMP1,0      ;Skip the next instruction if bit 0 of
btfsz STATUS,C     ;TEMP1 equals zero
decfsz TEMP1,F     ;Skip if C = 1
incfsz TEMP1,W     ;Decrement TEMP1; skip if zero
dcfsnz TEMP1,F     ;Leave TEMP1 unchanged; skip if
incfsz TEMP1,W     ;TEMP1 = H'FF'; W <- TEMP1 + 1
                     ;Decrement TEMP1, skip if not zero
                     ;Leave TEMP1 unchanged; skip if W <- TEMP1 + 1 is not
zero               ;skip if TEMP1 = 0
```

Comparisons

```
cpfseq TEMP1        ;Compare TEMP1 with W, skip if equal
cpfsgt TEMP1        ;skip if TEMP1 > W
cpfslt TEMP1        ;skip if TEMP1 < W
```

Branching Operations

```
bc    Task1          ;Go to instruction labeled "Task1" if C (Carry bit) is
set
bnc   Task2          ;Go to instruction labeled "Task2" if C is not set
bn    Task3          ;Go to instruction labeled "Task3" if N (Negative) is set
bnn   Task4          ;Go to instruction labeled "Task4" if N is not set
bz    Task5          ;Go to "Task5" if the Zero bit is set
bnz   Task6          ;Go to "Task6" if the Zero bit is not set
bov   Task7          ;Go to "Task7" if the overflow bit is set
bnov  Task8          ;Go to "Task8" if the overflow bit is not set
bra   Next            ;Branch unconditionally to the instruction labeled "Next"
```

Goto/call/return/return from interrupt

```
goto  There           ;Next instruction to be executed is
                      ;labeled "There"
call   Task1          ;Push return address; next instruction
                      ;to be executed is labeled "Task1"
rcall  TaskA          ;call subroutine up to 1K words from the
                      ;current program location
return
retlw 5               ;Pop return address off of stack
retfie
                      ;Pop return address; W <- 5
                      ;Pop return address; re-enable interrupts
```

Miscellaneous

```
lfsr  1,B'001000110011' ;Load FSR1 with the 12-bit RAM address H'33'
                         ;in Bank 2
clrwdt
                     ;If watchdog timer is enabled, this
                     ;instruction will reset it (before it
                     ;resets the CPU)
daw
                     ;Decimal adjust W
                     ;adjusts the result of a packed BCD addition
                     ;to the correct packed BCD value
sleep
                     ;Stop clock: I reduce power; wait
                     ;for watchdog timer or external signal
                     ;to begin program execution again
nop
reset
                     ;Do nothing; wait one clock cycle
                     ;Software Device Reset
```

Program Memory Operations

tblrd*	;Byte stored at the Program Memory
tblrd*+	;location pointed to by TBLPTR is loaded into TABLAT
tblrd*-	;Value pointed to by TBLPTR is loaded into TABLAT,
tblrd+*	;then TBLPTR is incremented
	;Value pointed to by TBLPTR is loaded into TABLAT,
	;then TBLPTR is decremented
	;TBLPTR is incremented, then the program memory
	;is read into TABLAT

7.6.6 Directives

In addition to mnemonics, the MPASM assembler provides additional means for controlling the operation of the assembler. These commands are called *directives* or *pseudo-operations* since they appear in the same location as that of mnemonics in the code, but they are not translated into instructions in the machine code. These commands generally define symbols, assign programs and data to certain areas of memory, generate fixed tables and data, indicate the end of the program, etc. MPASM supports 58 directives, which are summarized below.

[]	: Optional Arguments	expr	: Expression
<>	: Variables. Text you supply	△	: Space
	: An OR selection		

Directive	Description	Syntax
_BDRAM	Specify invalid RAM locations	△ _badram △ <expr>[-<expr>][,<expr>[-<expr>]]
BANKISEL	Generate RAM bank selecting code for indirect addressing	△ bankisel △ <label>
BANKSEL	Generate RAM bank selecting code	△ banksel △ <label>
CBLOCK	Define a Block Constants	△ cblock △ [<expr>]
ENDC		△ endc
CODE	Begins executable code section	[<name>] △ code △ [<address>]
_CONFIG	Specify configuration bits	△ _config △ <expr>
CONSTANT	Declare Symbol Constant	△ constant △ <label>[=<expr>,...,<label>[=<expr>]]
DA	Store Strings in Program Memory	[<label>] △ da △ <expr>[,<expr>,...,<expr>]
DATA	Create Numeric and Text Data	[<label>] △ data △ <expr>[,<expr>,...,<expr>] [<label>] △ data △ "<text_string>"[,"<text_string>","..."]</text_string>
DB	Declare Data of One Byte	[<label>] △ db △ <expr>[,<expr>,...,<expr>] [<label>] △ db △ "<text_string>"[,"<text_string>","..."]</text_string>

DE	Define EEPROM Data	<code>[<label>] Δ de Δ <expr>[,<expr>,...,<expr>] [<label>] Δ de Δ "<text_string>"[,"<text_string>","...]</code>
#DEFINE	Define a text Substitution Label	<code>#define Δ <name>[<value>] #define Δ <name>[<arg>,...,<arg>]<value></code>
#UNDEFINE	Delete a Substitution Label	<code>#undefine Δ <label></code>
DT	Define Table	<code>[<label>] Δ dt Δ <expr>[,<expr>,...,<expr>] [<label>] Δ dt Δ "<text_string>"[,"<text_string>","...]</code>
DW	Declare Data of One Word	<code>[<label>] Δ dw Δ <expr>[,<expr>,...,<expr>] [<label>] Δ dw Δ "<text_string>"[,"<text_string>","...]</code>
END	End Program Block	<code>Δ end</code>
EQU	Define an Assembly Constant	<code><label> Δ equ Δ <expr></code>
ERROR	Issue an Error Message	<code>Δ error Δ "<text_string>"</code>
ERRORLEVEL	Set Error Level	<code>Δ errorlevel Δ "0 1 2 <+-><message_number>"</code>
EXITM	Exit from a Macro	<code>Δ exitm</code>
EXPAND	Expand Macro Listing	<code>Δ expand</code>
NOEXPAND	Turn off Macro Expansion	<code>Δ noexpand</code>
EXTERN	Declares an external label	<code>Δ extern Δ <label>[,<label>]</code>
FILL	Fill Memory	<code>[<label>] Δ fill Δ <expr>,<count></code>
GLOBAL	Exports a defined label	<code>Δ global Δ <label>[,<label>]</code>
IDATA	Begin initialized data section	<code>[<name>] Δ idata Δ [<address>]</code>
_IDLOCS	Specify ID locations	<code>Δ Δ _idlocs Δ <expr></code>
#IF	Begin Conditionally Assembled Code Block	<code>#if Δ <expr></code>
#ELSE	Begin Alternative Assembly Block to IF	<code>#else</code>
#ENDIF	End conditional Assembly Block	<code>#endif</code>
#IFDEF	Execute if Symbol has Been Defined	<code>#ifdef Δ <label></code>
#IFNDEF	Execute if Symbol has not Been Defined	<code>#ifndef Δ <label></code>
INCLUDE	Include Additional Source File	<code>Δ include Δ <<include_file>> "<include_file>"</code>
LIST	Listing Options	<code>Δ list Δ [<list_optin>,...,<list_option>]</code>
NOLIST	Turn off Listing Output	<code>Δ nolist</code>

LOCAL	Declare Local Macro Variable	$\Delta \text{local } \Delta \langle \text{label} \rangle [, \langle \text{label} \rangle]$
MACRO	Declare Macro Definition	$\langle \text{label} \rangle \Delta \text{macro } \Delta [\langle \text{arg} \rangle, \dots, \langle \text{arg} \rangle]$
ENDM	End a Macro Definition	Δendm
_MAXRAM	Specify maximum RAM address	$\Delta \text{__maxram } \Delta \langle \text{expr} \rangle$
MESSG	Create User Defined Message	$\Delta \text{messg } \Delta \text{ "} \langle \text{message_text} \rangle \text{ "}$
ORG	Set Program Origin	$[\langle \text{label} \rangle] \Delta \text{org } \Delta \langle \text{expr} \rangle$
PAGE	Insert Listing Page Eject	Δpage
PAGESEL	Generate ROM page selecting code	$\Delta \text{pagesel } \Delta \langle \text{label} \rangle$
PROCESSOR	Set Processor Type	$\Delta \text{processor } \Delta \langle \text{processor_type} \rangle$
RADIX	Specify Default Radix	$\Delta \text{radix } \Delta \langle \text{default_radix} \rangle$
RES	Reserve Memory	$[\langle \text{label} \rangle] \Delta \text{res } \Delta \langle \text{mem_units} \rangle$
SET	Define an Assembler Variable	$\langle \text{label} \rangle \Delta \text{set } \Delta \langle \text{expr} \rangle$
SPACE	Insert Blank Listing Lines	$\Delta \text{space } \Delta \langle \text{expr} \rangle$
TITLE	Specify Program Title	$\Delta \text{title } \Delta \text{ "} \langle \text{title_text} \rangle \text{ "}$
SUBTITLE	Specify Program Subtitle	$\Delta \text{subtitle } \Delta \text{ "} \langle \text{sub_text} \rangle \text{ "}$
UDATA	Begin uninitialized data section	$[\langle \text{name} \rangle] \Delta \text{udata } \Delta [\langle \text{address} \rangle]$
UDATA_ACS	Begins access uninitialized data section	$[\langle \text{name} \rangle] \Delta \text{udata_acs } \Delta [\langle \text{address} \rangle]$
UDATA_OVR	Begin overlayed uninitialized data section	$[\langle \text{name} \rangle] \Delta \text{udata_ovr } \Delta [\langle \text{address} \rangle]$
UDATA_SHR	Begin shared uninitialized data section	$[\langle \text{name} \rangle] \Delta \text{udata_shr } \Delta [\langle \text{address} \rangle]$
VARIABLE	Declare Symbol Variable	$\Delta \text{variable } \Delta$ $\langle \text{label} \rangle [= \langle \text{expr} \rangle, \dots, \langle \text{label} \rangle [= \langle \text{expr} \rangle]]$
WHILE	Perform Loop While Condition is True	$\Delta \text{while } \Delta \langle \text{expr} \rangle$
ENDW	End a While Loop	Δendw

7.6.7 Addressing

The PIC MCU supports both direct and indirect addressing modes. In the direct addressing, the address portion contains the actual file register address of the operand. For example, ADDWF 0x2E,W means add the contents of the W register and the register at location 0x2E, and store the results in the W register. In the indirect addressing, the address portion of the instruction does not contain the actual address of the data, but points to a register, which contains the address of the data. In the PIC devices, indirect addressing is done via the INDF and FSR (File Select Register). Any instruction using the INDF register actually accesses the register pointed to FSR.

7.6.8 Programming Procedure

Developing the software is to put several instructions to be executed by the microcontroller in a desired order. It is always a good practice to proceed in the following steps, even for a seemingly simple program.

- 1) **Define the Problem.** state quite clearly what functions the microcontroller must perform, the inputs and outputs required, any constraints regarding speed of operation, accuracy, memory size, etc.
- 2) **Create the Flowchart.** draw the algorithm in general blocks and detail them as much as you can. See Figure 7.6-3.
- 3) **Generate the pseudo-code.** Pseudo-code is a way of describing the steps of an algorithm in an informal way that can later be translated into a program. A pseudo-code can consist of the following parts:
 - *Hardware Definition:* This part specifies the kind of PIC to be programmed. The directive LIST of the assembler is used for this purpose.
 - *Register File Definition:* for the PIC16F877 device, 68 bytes can be used to label the registers. By attaching a label to each register, tasks of addressing and debugging become easier. The directive EQU of the assembler is used for this purpose.
 - *Initialization:* PIC instructions starts from this part. In this part, the I/O ports are set, the STATUS register and the content of the register file are initialized. The directive ORG can be useful for this purpose.
 - *Main Instructions:* All the desired processes are included in this part. It can be divided into several sub-parts that form subroutines. .
 - *Program End:* This is the flag for the end of the program. The directive END of the assembler is used.

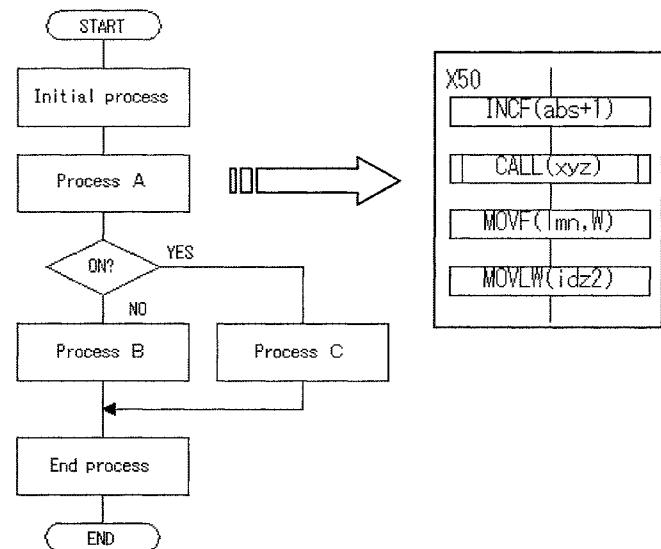


Figure 7.6-3

- 4) **Write the complete code.**
- 5) **Test the program with a debugger and/or simulator.** The preferred application for PIC MCU's is the MPLAB Integrated Development Environment (IDE) package available in the Engineering Design Lab. This package integrates several tools for editing, project managing, debugging, simulating, and compiling the MPASM codes. The package is also integrated with the DevBugger integrated development board for directly programming the PIC.

7.6.9 Simple Operations

With only the 35 instructions for the PIC16F877/887 devices, it is worthwhile to explore some commonly recurring sequences of instructions.

Either-or sequence

Assume that an instruction that affects the Z bit has just been executed. Then depending on the result, one instruction sequence or another is to be executed, continuing on after either case.

Decrement a 16-bit counter

Assume that the upper byte of the counter is called COUNTH and the lower byte is called COUNTL.

```
        movf  COUNTL,F      ;Set Z if lower byte = 0
        btfsc STATUS,Z      ;If so, decrement COUNTH
        decf  COUNTH,F
        decf  COUNTL,F      ;In either case decrement COUNTL
```

Note how the **movf** instruction is first used to test **COUNTL** for zero without changing it, and even without having to move it into **W**.

Test a 16-bit variable for zero

Using the same **COUNTH**, **COUNTL** variable as previously, the following sequence will either branch to an instruction labeled **BothZero** if the variable equals zero or to an instruction labeled **CarryOn** otherwise.

```
    movf  COUNTL,F      ;Set Z if lower byte = 0
    btfsc STATUS,Z      ;If not, then done testing
    movf  COUNTH,F      ;Set Z if upper byte = 0
    btfsc STATUS,Z      ;If not, then done
    goto BothZero        ;Branch if 16-bit variable = 0
CarryOn
```

Note that the same scheme will work for a variable of any size.

NOTE: For PIC16F1937, the additional 14 instructions can be used to increase efficiency. For instance, PIC16F1937 also provides the user with the ability to do relative branching, in addition to conditional one.

7.6.9.1 Branching in PIC18

Assume that an instruction that affects the Z bit has just been executed. Then depending on the result, one instruction sequence or another is to be executed, continuing on after either case. In PIC16 architecture, the following code would be implemented:

```
btfsc STATUS,Z      ;Test Z bit, skip if clear
goto Zset
Zclear
.
.
.
;Instructions to execute
;if Z = 0

goto Zdone
Zset
.
.
.
;Instructions to execute
;if Z = 1

.
.
.
Zdone           ;Carry on
```

In PIC18 architecture, the branching instructions can be used to simplify the code and increase efficiency:

```
bz      Zset      ;If Z is set, go to Zset
Zclear
.
.
.
;Instructions to execute
;if Z = 0

.
.
.
goto Zdone
Zset
.
.
.
;Instructions to execute
;if Z = 1

.
.
.
Zdone           ;Carry on
```

The branching instruction can branch to any program address within 127 instructions of the current location. It takes up less memory space than the goto instruction and is executed in less cycles.

These examples illustrate some of the tediousness of working with this “bare-bones” instruction set. However, it is important to notice three things:

1. Because of pipelining and because the chip can execute five cycles every microsecond (when used with a 20-MHz crystal), these sequences are executed quickly. For example, the 16-bit decrement sequence requires four cycles and therefore takes only 0.8 μ s to execute (whether or not the branch is taken).
2. Because the instruction set can operate directly on RAM variables, many operations avoid the “overhead” associated with other microcontrollers wherein operands in memory must be first loaded into an accumulator, then operated on, and then restored to memory.
3. Microchip Technology’s (free) assembler is a macro assembler. Consequently, sequences such as these can be prepared once and for all as macros. Thereafter, any such code sequence *looks* like a newly defined instruction. Furthermore, the assembler accepts *include* files. This means that any number of macro definitions can be put into one or more files and then have the assembler access these files as it assembles the file containing our application code. Any macros that are used will generate *object code*, the code that is loaded into the PIC for execution. Unused macros generate no object code. Even macro files of complex operations can be included, such as signed and unsigned multiply and divide routines for numbers of various lengths up to 32 bits long, available over the Internet from Microchip Technology. In fact, Microchip Technology supports users of its various microcontrollers with free data books, application examples, assembler and simulator software, as well as macro files over the Internet.

7.6.10 PIC16 Code Template

7.6.10.1 PIC16F877

- ; If interrupts are not used all code presented between the ORG 0x004 directive and the label main can be removed. In addition, the variable assignments for 'w_temp' and 'status_temp' can also be removed.

```
List      p=16f877           ; list directive to define processor
#include <p16f877.inc>      ; processor specific variable definitions
                                ↑ this file is included with MPLAB (in the MPLAB directory)

__CONFIG ( _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC & _WRT_ENABLE_ON & _LVP_OFF &
           _DEBUG_OFF & _CPD_OFF )
                                ↑ These are configuration bits
                                In the .asm file these configuration options should be on a single line (not wrapped)
```

; '__CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.

```
;***** VARIABLE DEFINITIONS
w_temp      EQU      0x70      ; variable used for context saving
status_temp EQU      0x71      ; variable used for context saving
```

⇒ Add your variables and constants here

```
;*****
ORG      0x000      ; processor reset vector
clrfl    PCLATH     ; ensure page bits are cleared
goto     main        ; go to beginning of program
                                } Code for reset

ORG      0x004      ; interrupt vector location
movwf   w_temp      ; save off current W register contents
movf    STATUS,w    ; move status register into W register
movwf   status_temp ; save off contents of STATUS register
                                } Save context during the interrupt.
```

; isr code can go here or be located as a call subroutine elsewhere ←

```
        movf    status_temp,w   ; retrieve copy of STATUS register
        movwf   STATUS        ; restore pre-isr STATUS register contents
        swapf   w_temp,f     ; restore pre-isr W register contents
        swapf   w_temp,w     ; restore pre-isr W register contents
        retfie
```

} **Restore context after completing the service**

main

⇒ Add your main code and subroutines here

```
END           ; directive 'end of program'
```

7.6.10.2 PIC16F887

; If interrupts are not used all code presented between the ORG 0x004 directive and the label main can be removed. In addition, the variable assignments for 'w_temp' and 'status_temp' can also be removed.

```
List      p=16f887          ; list directive to define processor
#include <p16f887.inc>      ; processor specific variable definitions
```

↑ this file is included with MPLAB (in the MPLAB directory)

```
_CONFIG _CONFIG1, _PWRTE_ON & _WDT_OFF & _INTOSC & _BOR_ON & _LVP_OFF & _CP_OFF & _MCLRE_ON &
IESO OFF & _FCMBN_OFF
_CONFIG _CONFIG2, _WRT_OFF & _BOR40V
```

↑ These are configuration bits

In the .asm file each of these two sets configuration options should be on a single line (not wrapped)

```
; '_CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.
```

```
;***** VARIABLE DEFINITIONS
w_temp      EQU      0x70      ; variable used for context saving
status_temp  EQU      0x71      ; variable used for context saving
```

⇒ Add your variables and constants here

```
;*****
ORG      0x000          ; processor reset vector
clrfl    PCLATH          ; ensure page bits are cleared
goto     main             ; go to beginning of program
```

}

Code for reset

```
ORG      0x004          ; interrupt vector location
movwf   w_temp           ; save off current W register contents
movf    STATUS,w          ; move status register into W register
movwf   status_temp        ; save off contents of STATUS register
```

}

Save context during the interrupt.

```
; isr code can go here or be located as a call subroutine elsewhere ←
```

```
movf    status_temp,w      ; retrieve copy of STATUS register
movwf   STATUS             ; restore pre-isr STATUS register contents
swapf   w_temp,f           ; restore pre-isr W register contents
swapf   w_temp,w           ; return from interrupt
```

}

Restore context after completing the service

main

⇒ Add your main code and subroutines here

```
END          ; directive 'end of program'
```

7.6.10.3 PIC16F1937

- ; If interrupts are not used all code presented between the ORG 0x004 directive and the label main can be removed. In addition, the variable assignments for 'w_temp' and 'status_temp' can also be removed.

```
List      p=16f1937      ; list directive to define processor
#include <p16f1937.inc>    ; processor specific variable definitions
```

↑ this file is included with MPLAB (in the MPLAB directory)

```
_CONFIG CONFIG1, _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _MCLRE_ON & _CP_OFF & _CPD_OFF &
_BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_ON
_CONFIG CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _BORV_19 & _LVP_OFF
```

↑ These are configuration bits

In the .asm file each of these two sets configuration options should be on a single line (not wrapped)

```
; '__CONFIG' directive is used to embed configuration data within .asm file.
; The labels following the directive are located in the respective .inc file.
; See respective data sheet for additional information on configuration word.
```

;***** VARIABLE DEFINITIONS

⇒ Add your variables and constants here

```
;*****
ORG      0x000      ; processor reset vector
clrfl    PCLATH     ; ensure page bits are cleared
goto     main        ; go to beginning of program
ORG      0x004      ; interrupt vector location
; isr code can go here or be located as a call subroutine elsewhere ←
retfie   ; return from interrupt
```

main

} **Code for reset**
} **PIC16F1937
has automatic
context-saving**

⇒ Add your main code and subroutines here

```
END      ; directive 'end of program'
```

7.6.11 PIC18 Code Template

; If interrupts are not used all code presented between the ORG 0x008 directive and the label main can be removed. In addition, the variable assignments for 'w_temp' and 'status_temp' can also be removed.

```
List      P=18F4620, F=INHX32, C=160, N=80, ST=OFF, MM=OFF, R=DEC
; list directive to define processor
#include <p18f4620.inc>      ; processor specific variable definitions
```

↑ this file is included with MPLAB (in the MPLAB directory)

```
CONFIG OSC=HS, FCMEN=OFF, IESO=OFF
CONFIG PWRT = OFF, BOREN = SBORDIS, BORV = 3
CONFIG WDT = OFF, WDTPS = 32768
CONFIG MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF, CCP2MX = PORTC
CONFIG STVREN = ON, LVP = OFF, XINST = OFF
CONFIG DEBUG = OFF
CONFIG CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF
CONFIG CPB = OFF, CPD = OFF
CONFIG WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
CONFIG WRTB = OFF, WRTC = OFF, WRTD = OFF
CONFIG EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF
CONFIG EBTRB = OFF
```

↑ These are configuration bits

; 'CONFIG' directive is the new directive used to embed configuration data within .asm file.
; See data sheet for additional information on configuration words.

```
;***** VARIABLE DEFINITIONS
w_temp      EQU      0x00      ; variable used for context saving
status_temp  EQU      0x01      ; variable used for context saving
```

⇒ Add your variables and constants here

```
;*****
ORG      0x000      ; processor reset vector
goto    main        ; go to beginning of program

ORG      0x008      ; High-priority interrupt vector
call    ISR_High   ; call high-priority ISR

ORG      0x018      ; High-priority interrupt vector
call    ISR_Low    ; call low-priority ISR
; High-priority ISR
ISR_High
movwf   w_temp      ; save off current W register contents
movff   STATUS,status_temp ; save off contents of STATUS register
...
movff   status_temp,STATUS ; retrieve STATUS register
swapf   w_temp,f    ; swapf does not change STATUS
swapf   w_temp,w    ; restore pre-isr W register contents
retfie
ISR_Low
...
main
```

} Code for reset

} Save context during the interrupt.

} Restore context after completing the service

⇒ Add your main code and subroutines here

```
END      ; directive 'end of program'
```

7.6.12 Macros

A macro is a function that is called from within the application code. This invocation is replaced by the code within the macro. Macros can be thought of as being similar to subroutines, but instead of calling a central routine, the routine is embedded into the source. The Microchip assembler includes macro capability. This means that new instructions can, in effect, be created that are really sequences of PIC instructions. For example, two instructions called **bank1** and **bank0** can be created that, when inserted into a source file, lead the assembler to make the substitution

```
bsf    STATUS, RP0
bcf    STATUS, RP1
```

in place of **bank1** and

```
bcf    STATUS, RP0
bcf    STATUS, RP1
```

in place of **bank0**.

A macro must be *defined* before it is invoked (i.e., used). The definition of the **bank1** macro takes the form

```
bank1  macro
       bsf    STATUS, RP0
       bcf    STATUS, RP1
       endm
```

Since the macro *definition* generates no object code, the definition can be inserted into a source (*.ASM) file at any place before it is invoked. For example, macro definitions might be added to the P1.asm code after the “Variables” section and before the “Vectors” section. Alternatively, macro definitions might be placed in a separate file called MACROS.INC and then “included” into the source file with

```
include "C:\MPLAB\MACROS.INC"
```

assuming the file is placed in the C:\MPLAB directory. In fact, any number of macro files may be added in this way, knowing that each macro in any of the files will add code to the object file only if it is invoked one or more times. Macro capability is an important feature, given the rudimentary instruction set of the PIC microcontrollers. As an example, Microchip Technology supports multiplication and division operations with dozens of macros for both signed and unsigned integers having lengths of 8, 16, 24, and 32 bits. These macros can be downloaded from Microchip web site. For example, to multiply two 8-bit unsigned integers, the following call would be inserted within the source code:

```
call    FXM0808U
```

The called subroutine is

```
FXM0808U
       clrf  ACCB1
       UMUL0808L
       retlw H'00'
```

where **UMUL0808L** is the macro whose definition must be included in the source code. This subroutine and macro combination requires the definition of four RAM variables;

```
cblock Bank0RAM
       :
       ;(Variables already defined)
       :
       ACCB1           ;Sixteen-bit result register
       ACCB0           ;Eight-bit multiplicand
       BARGB0          ;Eight-bit multiplier
       LOOPCOUNT       ;Loop counter to loop eight times
       endc
       AARGB0 equ ACCB0 ;Eight-bit multiplier
```

The combination requires the user to have loaded the two 8-bit numbers to be multiplied into **AARGB0** and **BARGB0**. It returns the 16-bit result in **ACCB1,ACCB0**. It introduces 21 words of program code and takes 54 to 73 cycles to execute, depending on the numbers being multiplied. Note that **AARGB0**, the source of the multiplier, becomes the least significant byte of the product, **ACCB0**. A macro definition can include one or more arguments. For example, a macro might be created to load a literal value into a register;

```

movlf  macro  literal,register
    movlw  literal
    movwf  register
    endm

```

Given this macro definition, its use as follows will initialize the RAM variable **BLNKCNT** with the value **MaxCount** (which has been equated to 50 in the P1.asm code);

```
    movlf  MaxCount, BLNKCNT
```

This could just as well have been expressed as

```
    movlf  50, BLNKCNT
```

Note that the preceding macro has the *side effect* of also changing the value of **W** to this same literal value. Side effects are a potential hazard of using macros, especially if the macro definition is stored in an **include** file, where it is not readily seen when debugging with the list file in hand. Given the PIC architecture, a conservative stance is to assume that all macros change both **W** and the **STATUS** bits (unless it is known otherwise, as is the case for the **bank1** and **bank0** macros defined earlier). A final issue arising with macros is how to handle a label within a macro definition. Consider how to create a *countdown* macro, which can be invoked with

```
    countdown COUNT,5
```

and which will initialize **COUNT** to 5 and then decrement **COUNT** to zero, to produce a variable delay. The intended code might first be written

```

    movlw  5
    movwf  COUNT
Again
    decfsz COUNT, F
    got0  Again

```

Now the macro definition can be created from this;

```

countdown macro COUNTREGISTER, InitialCount
    movlw  InitialCount
    movwf  COUNTREGISTER
Again
    decfsz COUNTREGISTER, F
    goto  Again
    endm

```

A problem arises with the label **Again** if this macro is used more than once in a source file. In that case, each invocation after the first will inject another **Again** label into the file, giving rise to an “address label duplicated” error for each of the invocations after the first. The solution is to define a *local* label, as follows;

```

countdown macro COUNTREGISTER, InitialCount
    local  Again
    movlw  InitialCount
    movwf  COUNTREGISTER
Again
    decfsz COUNTREGISTER, F
    goto  Again
    endm

```

Now the label **Again** has meaning only within the scope of the macro. It is purged from the symbol table when **endm** is reached.

One final warning should be raised; Do not precede a macro with one of the *skip* instructions. Note that the sequence

```

    btfsc  STATUS,C
    movlf  5, COUNT

```

using the two-instruction **movlf** macro defined earlier will not skip over these two instructions if **C** = 0, as intended. Instead it will skip to the second instruction of the macro

```
    movwf  COUNT
```

Rather than leaving **COUNT** unchanged, as was the intent of the original sequence, **COUNT** is loaded with whatever is in **W**.

7.6.13 Tables

In many cases, the purpose of a subroutine will be to communicate with the user, most often through a peripheral LCD screen based on the HD44780 protocol. These LCDs are communicated with one character at a time, and as such a simple word such as "HELLO" will take 10 calls to execute, one to move the letter into the **W** register, and one to call the 'write instruction' subroutine (which will then execute even more instructions)

Fortunately, the PIC assembly language allows us to utilize an idea known as 'tabling'. Traditional methods of implementing a table are to provide a subroutine that adds a constant **n** to a known point in the program memory space and stores this value in the **PCL** (*addwf PCL, f*). This causes the PIC to jump to the new memory location, where a *retlw* instruction is used to store the table value in the **W** register and return to the caller's code.

The most traditional way to do this is as follows:

```
Table      ; Return Table Value for contents of W
    addwf PCL, f      ; Add the Table Index to the program counter
TableEntries
    retlw "H"
    retlw "E"
    retlw "L"
    retlw "L"
    retlw "O"
    retlw 0
```

The zero value at the end of the *Table* subroutine is useful to indicate that the string has ended. 0 here is equivalent to the 0x000 ASCII character, commonly referred to as the null character. Please note that when *addwf PCL, f* is executed, the program counter is *already* incremented to the next instruction. Therefore, to return "H" in the table, a value of zero must be passed in **W**. This is similar to C++ character strings, where the first index is 0.

A more compact subroutine (at least visually) can be created using the *dt* assembler directive (see **Directives**, in the previous sections), which combines the *retlw* instructions. As such, the above sequence can be compacted to:

```
Table      ; Return Table Value for contents of W
    addwf PCL, f      ; Add the Table Index to the program counter
TableEntries
    dt "HELLO", 0
```

7.6.13.1 Tabling and the PCLATH Register

The example shown above is useful for many applications, but only under the condition that the table itself is located in the first 256 instructions of the program memory code. If the table is not located in the first 256 instructions or straddles the first 256-instruction boundary, then the execution will jump to an invalid address because the **PCLATH** register will not be set correctly for the table read.

This can be simply fixed using the following code:

```
Table      ; Return Table Value for contents of W
            ; anywhere in PICmicro Memory
    movwf Temp
    movlw HIGH TableEntries
    movwf PCLATH
    movf Temp,w
    addlw LOW TableEntries
    btfsc STATUS,C
        incf PCLATH,f
    movwf PCL
            ; Write the correct address to the Program
            ; Counter
```

7.6.13.2 Tabling and Displaying String on LCD

We can store the letters in a sentence in a table as follows:

```
Welcome_Msg
    addwf  PCL,F           ;PCL + work register = the nth letter
    dt      "Hello World!", 0 ;Sentence for output
```

The table would return the nth letters in the sentence corresponding to the numerical value in the working register added to the program counter latch (PCL). The returned value would be stored in the working register as an 8-bit ASCII value. As we go down the table, the last character that would be returned is "0" with ASCII code '00000000'. Reader can refer to section 7.6.11 in the Course Notes for more information regarding tables.

To send the ASCII code of the sentence to the LCD we use a very convenient macro called Display.

```
Display    macro   Message
    local    loop_           ;local variable defined for looping
    local    end_           ;local flag variable
    clrf    Table_Counter  ;variable to count letters in sentence
    clrw
loop_     movf   Table_Counter,W ;move counter into work register
    call    Message
    xorlw  B'00000000'      ;check WORK reg to see if 0 is returned
    btfsc  STATUS,Z
    goto   end_
    call    WR_DATA          ;send letter to LCD
    incf   Table_Counter,F  ;get next letter
    goto   loop_
end_
    endm
```

This macro takes in a parameter "Message", which can be replaced by the name of the table, for example: Display Welcome_Msg. The macro calls the table with value Table_Counter in Work Register which gets added to PCL and branches the program counter to get the desired character. For example, value 0 in Table_Counter would return the first character in Message, value 1 return the second character, ect.

After the ASCII value is returned from Message within the work register, it is send over to the LCD by simply calling WR_DATA subroutine.

7.6.14 Using Tables in PIC18

Tables are useful structures to store fixed data in program memory. PIC18 features a new way of creating tables and reading from tables that is more efficient both in the use of space and data retrieval time. This feature was possible because of the 16-bit instruction word length, which can fit exactly two bytes of data. Therefore, 8-bit characters can be stored as raw data in program memory rather as a part of an instruction. The Table Read instructions in the PIC18 instruction set are an efficient way of reading the program memory one byte at a time. This way, the table can be read without using subroutines to return each character.

To define a table, a special directive, **db** (declare byte), is used to write bytes of data into the FLASH program memory during programming. The **db** directive can be followed by bytes written as numbers, or by a text string. Remember to end each table with a termination character, such as 0 (null character), so the microcontroller knows when to stop. Note that this data will not be stored as instructions and the microcontroller may behave erratically if these bytes were executed as instructions. It is therefore important to put the tables in a location in the program memory that will never be executed as instructions.

Reading the table requires the data stored in program memory to be moved into RAM. In the PIC18, this is done through the Table Read operation, shown in Figure 7.6-4.

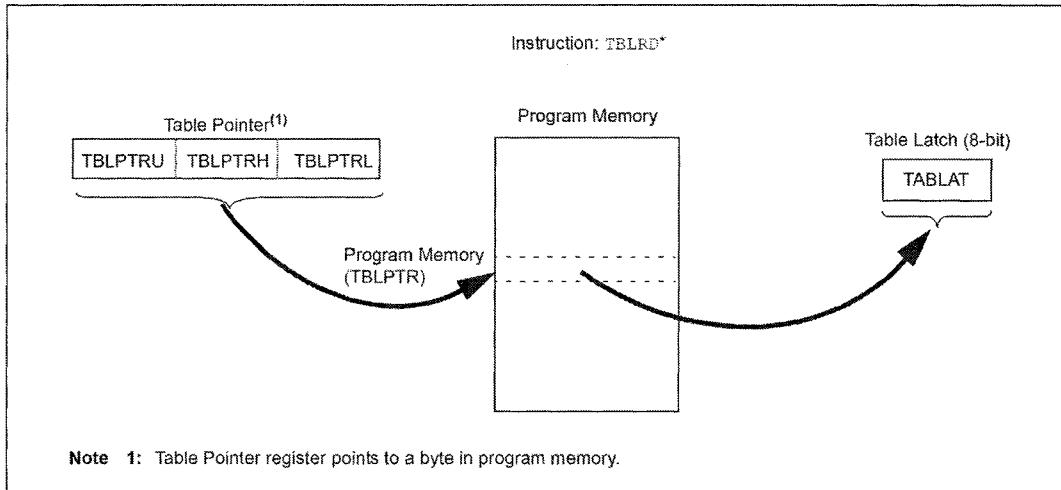


Figure 7.6-4

To read the table, the full address of the table must be loaded into a set of 3 registers called the Table Pointer (**TBLPTR**). The Table Pointer points to the starting location of the table and consists of **TBLPTRU** (bits 21:16), **TBLPTRH** (bits 15:8), and **TBLPTRL** (bits 7:0). Since the table pointer is able to address the entire program memory, the table can be any length and located anywhere in program memory. To load these three registers, three special directives, **upper**, **high**, and **low** are used to specify the three bytes in the address of a label. After the table pointer is updated, the **tblrd*** instruction is used to copy the byte at the location pointed to by **TBLPTR** into the Table Latch (**TABLAT**) register. After this, the byte can be retrieved from the **TABLAT** register for processing. To continue reading, use a variant of **tblrd*** such as **tblrd*+** or **tblrd*+**, which automatically increments the table pointer. The table can even be read backwards by using **tblrd*-**. The following sample code demonstrates the described procedure.

Writing to the program memory is also done through the Table Pointer and the Table Latch registers, but can only be done in blocks of 64 bytes. The Table Write instruction (**tblwt***) writes the contents of **TABLAT** into a set of 64 holding registers before the entire block is written in program memory. Writing to the program memory during runtime is not recommended.

<pre> Table db "TableData", 0 ReadTable movlw upper Table movwf TBLPTRU movlw high Table movwf TBLPTRH movlw low Table movwf TBLPTRL tblrd* movf TABLAT, W Again call WriteData tblrd+* movf TABLAT, W bnz Again </pre>	<pre> ;Label that declares the ;start of the table ;define table, ending with 0 ;Begin reading the table ;Move Table<20:16> into TBLPTRU ;Move Table<15:8> into TBLPTRH ;Move Table<7:0> into TBLPTRL ;Read byte located at TBLPTR and ;copy byte into TABLAT ;Move the byte into W ;Loop Label ;Process the byte just retrieved ;Increment TBLPTR, then read ;the next byte ;Move the byte into W ;Continue reading until '0' is ;detected </pre>
--	---

7.6.15 Data Parsing

A data parser analyzes the data's grammar and compares it to a predetermined/known grammatical structure or syntax. This is especially useful in assembly merely due to the fact that there is no "if" statement like some of the higher level programming languages. PIC16 and PIC18 I2C communications is an excellent example at demonstrating how data parsing is done and its benefits, and the following section would explain how data sent by PIC16 is parsed by PIC18 to set PIC18's IO pins.

In extended I/O function allows PIC16 to write the TRIS/PORT of PIC18 pins, or read from it. This is done by sending a sequence of binary data from the PIC16 to PIC18 through I2C bus. The PIC18 takes this byte of data, and calls a data parser to analyze it.

The following section will assume that PIC16 is trying to set PORTB3 of the PIC18, the corresponding sequence would be b'00010110'.

When the PIC18 receives the first transaction from PIC16, it uses the "btfs/s" mnemonics on the first bit to determine if the PIC16 is trying to write PIC18's pin or read from it.

```
btfs    instruction,7
call    handlewrite
btfs    instruction,7
call    handleread
```

The 7th bit is 0, thus we know that a write is wished to be performed. Next the second bit of the transaction is tested, this bit is used to symbolize if writing is done to a PORT or a TRIS when a write is desired, and has no significance if a read is desired.

```
handlewrite
btfs    instruction,6
call    hanldwriteport
btfs    instruction,6
call    hanldwritetris
```

Next, the bits 5 and 4 are used to extinguish between writing to PORTA/B/C.

```
Bit 5,4:      00-PORTA
               01-PORTB
               10-PORTC
```

Bit 5 and 4 is 01, thus a write to PORTB is selected. Bit 3,2,1,0 together, defines which bit in PORTA/B/C the user is trying to write.

```
handlewriteportb
btfs    instruction,3
bra    WBx467
btfs    instruction,2
bra    WBx3
btfs    instruction,0
setf    LATB
btfs    instruction,0
clrf    LATB
bra    donewriteport
WBx3
btfs    instruction,0
bsf    LATB,3
btfs    instruction,0
bcf    LATB,3
bra    donewriteport
WBx467
btfs    instruction,2
bra    WBx6
btfs    instruction,0
```

```

bsf      LATB, 4
btfss   instruction, 0
bcf      LATB, 4
bra     donewriteport
WBx6
btfsc   instruction, 1
bra     WBx7
btfsc   instruction, 0
bsf      LATB, 6
btfss   instruction, 0
bcf      LATB, 6
bra     donewriteport
WBx7
btfsc   instruction, 0
bsf      LATB, 7
btfss   instruction, 0
bcf      LATB, 7
bra     donewriteport
donewriteport
      return

```

The several set of code described above forms a complete data parser. It has a predetermined grammatical structure. By following this logic flow, the PIC18 can determine exactly what the PIC16 desires to do with its IO pins.

7.6.16 Programming Checklist

- ✓ Write instruction mnemonics in lowercase (e.g., `xorwf`). Write special register names, RAM variable names, and bit names in uppercase (e.g., `STATUS`, `RP0`). Write instruction and subroutine labels in mixed case (e.g., `Mainline`, `LoopTime`).
- ✓ Do not enable code protection! It cannot be undone. You will have to replace the chip.
- ✓ TRIS - make sure they are set properly for input/output pins.
- ✓ Disable WDT if not needed.
- ✓ Check MCLR is connected to V_{CC}
- ✓ Make sure "Osc" fuses are set for your type of oscillator, crystal (XT), RC (RC), resonator (HS), etc.
- ✓ Watch out for the difference between "`=`" and `EQU`. Theoretically, equated variables (they are in fact compiler directives) cannot be changed afterwards, but assigned "`=`" can.
- ✓ `END` is not `SLEEP` or Sometimes this thing just won't stop!

Question: Why is my PIC program running in a loop when I just want it to run once each time the PIC is reset? My program ends with

```
    movf ADRESH,w
    movwf PORTC
    end
```

Answer:

You are rolling off the end of memory and then restarting at `x'0000`. If you want it to quit at the `END`, you need to tell the processor to stop. The `END` just tells MPASM that it is the end of your source file. The PIC just goes on to the next instruction. A reasonable choice would be a `SLEEP` instruction, or an endless loop:

```
label: goto label
```

- ✓ Watch out for maximum speed limits during low voltage operation. When running the PIC at low voltage, the maximum clock speeds are not supported. Check the data sheet carefully.
- ✓ When your code doesn't work:
 - Set the entire program aside, find a way to separate just a small, but functional part of it and get that part working. Then, slowly add in the rest of the code from the original program, testing and debugging each addition, until you have the entire program working.
 - Comment each and every line of the code with what you think it is doing, and then check the datasheet against your comments.
 - The destination of the instruction may be specifying the `W` register (`d = 0`) instead of the file register (`d = 1`).
 - The register bank select bits (`RP1:RP0` or `IRP`) may not be properly selected. Also if interrupts are used, the register bank select bits may not be properly restored when exiting the interrupt handler.
- ✓ Do you seem not to be able to modify `STATUS` register flags? if the `STATUS` register is the destination for an instruction that affects the `Z`, `DC`, or `C` bits, the write to these bits is disabled. These bits are set or cleared based on device logic. Therefore, to modify bits in the `STATUS` register it is recommended to use the `BCF` and `BSF` instructions.
- ✓ Sample code for most common applications (including serial IO, LCD interfacing, keyboards, Internet interface, etc.) are available at <http://www.piclist.com/faq> .

7.7 Example

7.7.1 Sample LED Blinking Code

The example below, while quite rudimentary, will nevertheless exhibit the minimal external circuitry needed by the PIC microcontroller. It will also serve as a vehicle for explaining what needs to be included in program code. PIC microcontrollers tie up very few pins with overhead functions. The circuit in Figure 7.7-1 shows four pins needed to supply power, two pins connected to a 10-MHz crystal, and the *Master Clear* pin, which may either be connected as shown, or directly to the +5V supply (since the PIC includes an internal power-on reset circuit).

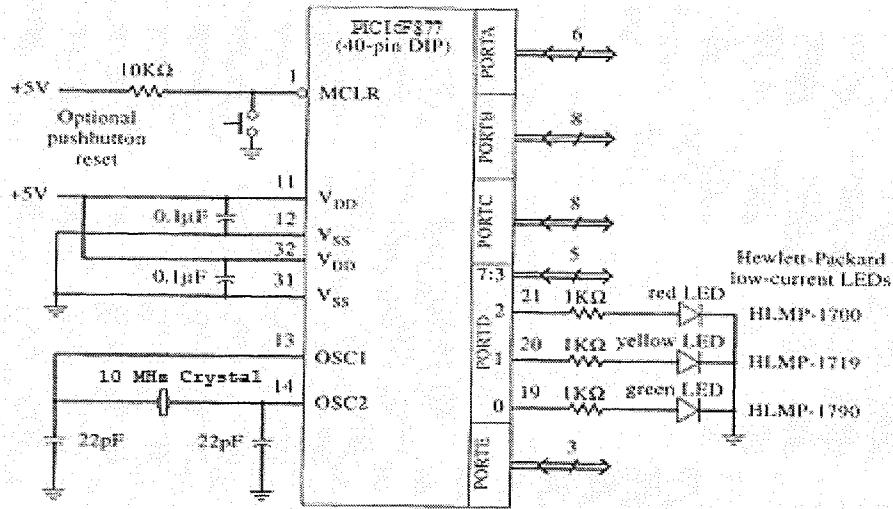


Figure 7.7-1

For this example, just three I/O pins are used. The schematic shows the LED circuitry being driven from **PORTD**, although any of the other ports could be used. In making the decision of what port pins to use for I/O functions, it is helpful to consider the alternative special functions associated with each port. Since **PORTD** has been chosen to drive the LEDs, it is worth noting that the *parallel slave* port alternative function associated with **PORTD** on PIC16F877 and PIC18F4620 allows its eight lines to be connected directly to the data bus of *another* microprocessor (PIC16F887 does NOT have a PSP peripheral). The other microprocessor can then read directly from the PIC or write directly to it. By using **PORTD** to drive these three LEDs, this alternative function will be ruled out.

To use any other port, or port pins, to drive these three LEDs, the code change will be minimal. The only change that

Output driven	Load current up to	Output voltage
High	3.0 mA	Will drop no more than 0.7 V from V_{DD}
Low	8.5 mA	Will rise no more than 0.6 V from V_{SS}

might trip up a newcomer to PIC microcontrollers would occur if one of the potential analog-to-digital converter pins of the PIC16F877 (i.e., **PORTA** or **PORTE** pins) were selected. The PIC16F877 include a register called **ADCON1**. The bits of **ADCON1** default to a state that causes every one of the eight possible analog-to-digital converter input pins to serve the A/D function, not the general-purpose I/O function. To drive LEDs with these pins, **ADCON1** must be initialized appropriately.

The circuit of the figure uses low-current LEDs. These units produce as much light with 2 mA as do many standard LEDs with 16 mA. Each of these LEDs exhibits a forward voltage drop of about 2 V. The supply voltage range of the PIC16F877 is 4.0 – 6.0 V for a 4MHz oscillator and 4.5 – 5.5 V for an oscillator between 4MHz and 20MHz. The absolute maximum output pin drive current (when driven either high or low) is 25mA. The PIC output pin drive specifications are shown in table above. For currents around 2mA, the PIC output pin voltage will drop about 0.7 V below the supply voltage. Considering the supply voltage range, each LED will be turned on with a current in the 1-3 mA range. The circuit of Figure 7.7-1 shows a 10-MHz crystal that will produce 0.4-μs-long internal clock cycles. While each PIC family member part is available in three speed grades (4 MHz, 10 MHz and 20 MHz), the 10-MHz part serves well for many applications. The assembly language code, P1.ASM, to blink the green LED on and off every second is shown

below. While this may seem to be a great deal of code for such a simple function, it will serve as a *template* for the writing of code for subsequent projects. It will also serve as a vehicle for introducing features of a PIC program.

```
;;;;, P1 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Toggle the green LED every half second (approximately)
; Count cycles to obtain timing.
; Use 10 MHz crystal for 0.4 microsecond internal clock period,
; or 4MHz crystal for 1 microsecond period.
;
;;;;, Program hierarchy ;;;;;;;;;;;;;;;
;
;Mainline
; Initial
; Blink
; TenMs
;
;;;;, Equates ;;;;;;;;;;;;;;;
Bank0RAM equ H'20' ;Start of Bank 0 RAM area
MaxCount equ 50 ;Number of loops in half a second
Green equ B'00000001' ;PORTD mask for green LED

;;;;, Variables ;;;;;;;;;;;;;;;
cblock Bank0RAM
BLNKCNT ;LED loop counter
COUNTH ;Two-byte counter for TenMs subroutine
COUNTL
Endc

;;;;, Vectors ;;;;;;;;;;;;;;;
org H'000' ;Reset vector
goto Mainline ;Branch past tables
org H'004' ;Unused interrupt vector
Stop
goto Stop ;Stop if an interrupt accidentally occurs

;;;;, Tables ;;;;;;;;;;;;;;;
; No tables needed
;;;;, End of Tables ;;;;;;;;;;;;;;;
;;;;, Mainline program ;;;;;;;;;;;;;;;
Mainline
call Initial ;Initialize everything
MainLoop
call Blink ;Blink LED
call TenMs ;Insert ten millisecond delay
goto MainLoop

page
;;;;, Initial subroutine ;;;;;;;;;;;;;;;
;
; This subroutine performs all initializations of variables and registers.

Initial
movlw MaxCount ;Initialize first blink of LED
movwf BLNKCNT ;to last about 0.50 seconds
movlw Green ;Turn on green LED; turn off others
movwf PORTD
bsf STATUS,RP0 ;Set register access to bank 1
clrfs TRISD ;Set up all bits of PORTD as outputs
```

```

        bcf      STATUS,RP0          ;Set register access back to bank 0
        return

;;;;;; Blink subroutine ;;;;;;;;
; This subroutine blinks a green LED every 0.5 second.

Blink
        decfsz  BLNKCNT,F        ;Decrement loop counter and return if not zero
        goto    BlinkEnd
        movlw   MaxCount          ;Reinitialize BLNKCNT
        movwf   BLNKCNT
        movlw   Green              ;Toggle green LED
        xorwf   PORTD,F

BlinkEnd
        return

;;;;;; TenMs subroutine ;;;;;;;;
;
; This subroutine and its call inserts a delay of exactly ten milliseconds into
; the execution of code.
; It assumes a 10 MHz crystal clock.
; Cycles = 25000
        TenMs
                ;24993 cycles
        movlw   0x86
        movwf   COUNTH
        movlw   0x14
        movwf   COUNTL

TenMs_0
        decfsz  COUNTH, f
        goto    $+2
        decfsz  COUNTL, f
        goto    TenMs_0
                ;3 cycles
        goto    $+1
        nop

                ;4 cycles (including call)
        return

```

The above program begins with some *comment* lines describing what it does. Each of these lines begins with a “;” in column 1, denoting that the entire line is to be treated as a comment to the reader and to be ignored by the assembler program. Further comment lines are used to present a *program hierarchy*. This lists the name of each subroutine and shows how each subroutine is related to others—who calls it and whom it calls. It also serves as a table of contents for the project code.

Four lines of assembler directives follow the program hierarchy. When used with no option, the *list* directive marks the lines that follow as lines to be included in the assembler’s output listing. (A *nolist* directive would mark subsequent lines to be left out of the output listing.) When used with options, the *list* directive serves as a catchall facility for passing along a variety of parameters to the assembler. Thus **P = PIC16F877** says to assemble this source file for the very specific pin out and on-chip facilities of the PIC16F877 chip. The **F = INHX8M** option tells the assembler which of three possible output formats to use when it creates the *hex file*, P1.HEX. This is chosen to match up with the requirements of Microchip’s PIC programmer so the assembled code can be programmed into a chip, ready to drive the target system with its LEDs. The **C = 160** and **N = 80** options tell the assembler to include 160 columns and 80 lines on each page of the list file output, P1.LST. This file is used to obtain a printed output listing. The **ST = OFF** and **MM = OFF** options tell the assembler to leave the *symbol table* and the *memory map* out of the list file. The symbol table lists the address associated with every label and every variable. It also lists the value equated to every name in the program. Depending on the size of a program and the debugging tools available, some users find the symbol table helpful. The memory map shows what program memory addresses have been used. The **R = DEC** option tells the assembler to treat undesignated numbers as *decimal* numbers. For example, the number 50 in the line

```
MaxCount    equ    50
```

will be treated as a decimal number as a result of the **R = DEC** option.

When the assembler is installed, the install program creates a directory called MPLAB™ into which it loads its many files and programs. Each PIC family member has associated with it an *.INC file, which lists the name and address of every register and the name and bit position of every bit defined by Microchip Technology for its special purpose register file. The line

```
Include      "C:\MPLAB\P16F877.INC"
```

tells the assembler how to deal with all of these names when it assembles the P1.ASM source file.

The next line uses the **_config** directive to specify the microcontroller's *configuration word*. This word is programmed into a chip along with the application program. The parameters shown in the program do the following:

- Turn off the *code protection* feature.
- Turn on the *power-up timer enable* feature.
- Select the *high speed crystal oscillator* option.
- Turn off the *watchdog timer* feature.
- Turn off the *brown-out reset* feature.

The **errorlevel** directive provides a mechanism for disabling a specific "error" or "warning" message that would otherwise be generated by the assembler. If error 302 were left enabled, then the

```
clrf  TRISD
```

instruction in the **Initial** subroutine would be flagged by the assembler. Note that **TRISD** is equated to the Bank 1 address H'0088'. Any direct addressing of an address over H'007F' needs to have the **RP0** and **RP1** bits set; this flagging serves as a warning to make sure this bit is indeed set.

The **Equates** section of the code permits the use of readable names in place of more cryptic numbers. For example, the **Blink** subroutine could have used

```
movlw      50
```

in lieu of

```
movlw      MaxCount
```

However, a subsequent change to a 10-MHz clock from the present 4-MHz clock would only require the change

```
MaxCount  equ  125
```

to still toggle the green LED every half second since $125 = 50(10/4)$. This becomes especially valuable when a name is used in several places in a program.

The **Variables** section uses a **cblock ... endc** directive to tell the assembler to assign consecutive addresses to the three variables listed. The **Bank0RAM** operand has been previously equated to address H'20', the beginning of the RAM area. Thus **BLNKCNT** will be assigned to address H'20', **COUNTH** to address H'21', and **COUNTL** to address H'22'.

As discussed previously, these PIC microcontrollers have two special program addresses, H'000' and H'004', which are automatically loaded into the program counter when the chip is reset or when an interrupt occurs. The **Vectors** section shows the handling of these two conditions.

The following **Tables** section is empty for this project. Recall the discussion in the preceding chapter in which it was pointed out that if tables are confined to addresses below H'0FF', then table use will be simplified. This empty **Tables** section is included here to serve as a reminder of where to put tables when they are created.

The **Mainline** program calls three subroutines and then loops back to execute the second and third subroutines repeatedly. The first subroutine, **Initial**, initializes variables and special-purpose registers appropriately. Thus **BLNKCNT** is initialized to 50 and **PORTD** is initialized to be an output port with its **Green** bit set and the remaining bits cleared.

The **page** directive included just before the **Initial** subroutine tells the assembler to insert a page break in the list file at this point. This directive provides a mechanism for cutting a page short, at a convenient break.

The **Blink** subroutine does nothing but decrement **BLNKCNT** for 49 out of every 50 times it is called. During the fiftieth time, it exclusive-ORs the least significant bit of **PORTD** with a one, thereby complementing it. The other bits of **PORTD** are exclusive-ORed with zeros, leaving them unchanged.

	Cycles
call TenMs	2
movlw 0x86 ; H'86' = 134 decimal	1
movwf COUNTH	1
movlw 0x14 ; H'14' = 20 decimal	1
movwf COUNTL	1
	Total: 6
 ;COUNTH = 134, COUNTL = 20	
TenMs_0	
decfsz COUNTH, f ; decrement COUNTH, skip if zero, 1 cycle	1
goto \$+2 ; go two addresses forward , 2 cycles	2
...	
goto TenMs_0 ; 2 cycles	2
	Total: 665
 ;now COUNTH = 1, COUNTL = 20 (...19...18...2)	
decfsz COUNTH, f ; decrement COUNTH, skip if zero, 2 cycles	2
...	
decfsz COUNTL,f ; decrement COUNTL, skip if zero, 1 cycle	1
goto TenMs_0 ; 2 cycles	2
	Total: 5
 ;now COUNTH = 0, COUNTL = 19	
decfsz COUNTH, f ; decrement COUNTH, skip if zero, 1 cycle	1
Goto \$+2 ; go two addresses forward , 2 cycles	2
...	
Goto TenMs_0 ; 2 cycles	2
	Total: 1275
 ;The above two sequences repeat 18 times until COUNTL =1, 1275+5= 1280*18 →	Total: 23040
 ;now COUNTH = 0, COUNTL = 1 →	Total: 1275
;now COUNTH = 1, COUNTL = 1	
decfsz COUNTH,f ; decrement COUNTH, skip if zero, 2 cycles	2
...	
decfsz COUNTL,f ; decrement COUNTL, skip if zero, 2 cycles	2
...	
goto \$+1 ; 2 cycles	2
nop ; 1 cycle	1
return ; 2 cycles	2
	Total: 9
 Total: 6 + 665 + 5 + 23040 + 1275 + 9 =	----- 25,000 Cycles

The final subroutine, **TenMs**, wastes 10 ms of time. Its call from within the mainline loop inserts a 10-ms delay into the loop. Since the remainder of the code executed in the mainline loop takes up so few cycles, the loop time is just slightly over 10 ms. Fifty times around the mainline loop takes just slightly over half a second.

This **TenMs** subroutine provides an excellent opportunity for counting the cycles taken by nested loops of instructions. This function is based on a 10MHz clock and the understanding that each PIC instruction takes 4 clock cycles to complete. As such, to generate a 10ms delay, the number of cycles required is equal to:

$$CYCLES = 10000000 \times \frac{cycles}{sec} \times \frac{1}{4} \frac{operations}{cycles} \times \frac{1}{1000} \frac{sec}{milisec} \times delay(milisec)$$

This is tabulated in figure above, where the left column lists instruction sequences in the order they are executed, the middle column shows the sequencing of values in COUNTL and COUNTH, and the right column shows the cycles taken by the instruction sequences.

The **TenMs** algorithm can be easily generalized into a function such as:

$$\text{Cycles} = 20 + (\text{COUNTH}-1)*5 + (\text{COUNTL}-2)*1280 + 1275$$

7.7.2 Sample Switch Debouncing Code

The following example explains what is de-bouncing as well as demonstrating how it can be achieved through software.

Though an input can be from any number of sources, one of the most common and the easiest to implement, would be a switch. Due to mechanical properties of a switch, when a switch is closed, there is a period of time in which the electrical connection "bounces" between open and closed. To a microcontroller, this "bouncing" can be interpreted as multiple button pushes as demonstrated below in figure 7.7-1. Therefore, switch inputs must be debounced to prevent multiple false detections while switch contact settles.

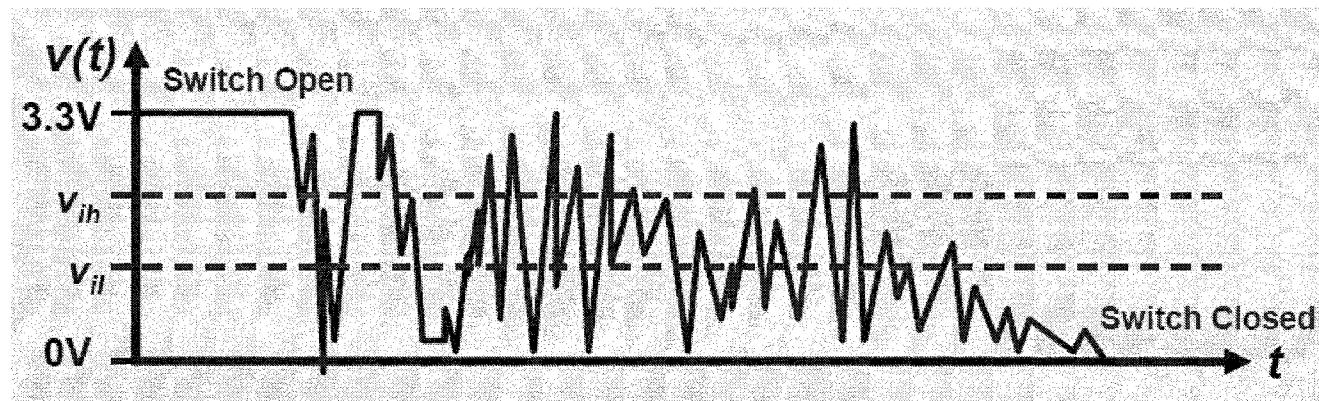


Figure 7.7-2

There are typically 2 methods employed to de-bounce a switch: one using timers and interrupts and one which polls the state of the switch. In this example, we will be de-bouncing the switch using the polling method.

A flow chart of de-bouncing mechanism of how to read from input pins is demonstrated below:

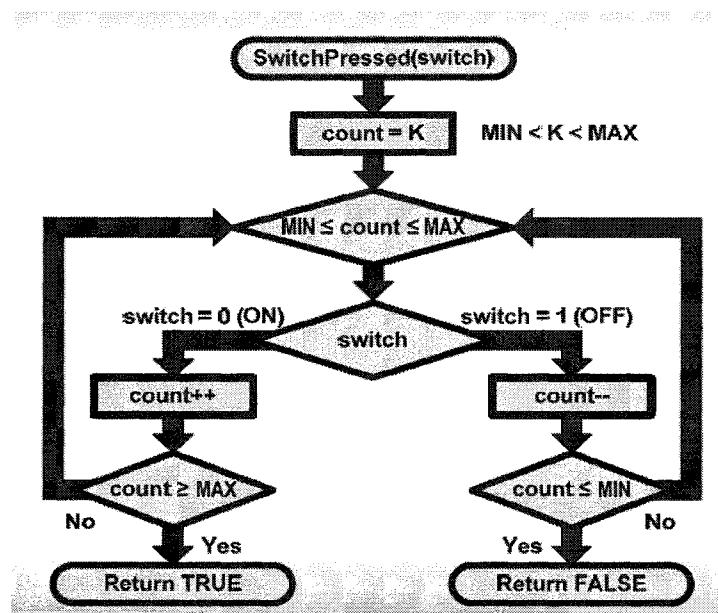


Figure 7.7-3

How the polling method works is that it polls from the input pins, and increments a parameter "count" when the switch is changed to a desired state. When the incrementation of "count" is reaches a desired value, it means that the bouncing period is over, and the code will return the state of the switch accordingly. In another word, the "count" is similar to a delay routine that waits for the bouncing period to settle down.

The following code utilizes the polling mechanism to de-bounce input pin PORTA,3 while reading its input.

```

;; LastStableState(lss) = 1
    movlw      D'1'
    movwf      LastStableState      ; Assume the Switch is up.
;; counter = 0
    clrf      Counter
MainLoop:
    clrw      ; need w=0 for counter calculation
    btfsc    LastStableState,0
    goto    LookingForDown
LookingForUp:
    ;; lss is 0: if current_state != 0 w=counter++ else counter = 0
    ;; i.e. if porta,3 is CLEAR skip the increment
    btfsc    PORTA,3
    incf      Counter,w          ; if it's SET, w = counter+1
    goto    EndDebounce
LookingForDown:
    ;; lss is 1: if current_state != 1 w=counter++ else counter = 0
    ;; i.e., if porta,3 is SET skip the increment
    btfss    PORTA,3
    incf      Counter,w          ; if switch low, w = counter+1
EndDebounce:
    movwf      Counter ; store either 0 or incremented value
    ;; W==counter, either 0 or incremented
    xorlw      5
    btfss    STATUS,Z
    goto    Delay1ms

    ;; zero flag was set: means W==5
    ;; if counter = 5
    ;; lss = !lss
    ;; counter = 0
    ;; if "active" take action
    ;;
switch_happened:
    comf      LastStableState,f ; after 5 straight, reverse the direction
    clrf      Counter
    btfsc    LastStableState,0 ; Was it a key-down press?
    goto    Delay1ms          ; no: take no action

    [down button action here]

Delay1ms:
    [delay routine here]
    goto MainLoop

```

7.8 Interface with Keypad

Being able to communicate with the microcontroller is an important feature that all Design projects must accommodate. Hence, interface with a keypad is a necessary task. Despite a wide variety of appearances for the commercial keypads, the majority of them are simply matrix switches. Figure 7.8-1 shows one typical switch and its schematics.

The easiest way to connect a keypad to the PIC would be a straight connection of the keypad wires to the PIC as shown in Figure 7.8-2. In the figure, **PORTB** has been used for the interface. First, we set up pins 4 through 7 as inputs, and pins 1 through 3 as outputs, and clear the output pins. Initialization of **PORTB** is done by setting the **TRISB** register, as shown in Figure 7.8-3. Since internal pullup resistors on **PORTB** are not needed, the MSB of **OPTION_REG** remains in its default value 1. For the particular 3×4 keypad shown in Figure 7.8-1, 7 bits (pins) are needed on the PIC, hence bit 0 of **PORTB** is not used. The output pins (1 – 3) are connected to the column wires, and the input pins (4 – 7) are connected to the row wires of the keypad. Pressing any key on the keypad will result in connecting the corresponding row and column pins. Thus, in the code, the output pins are sequentially set (set to +5V) in a loop and the keys are sequentially polled in each successive column by calling the **find_row1**, **find_row2**, **find_row3**, and **find_row4** subroutines. Within each **find_rowX** subroutine, we determine specifically which key has been pressed by testing whether the corresponding input pin (pins 4-7 on **PORTB**) has gone high. Upon determining the pressed key, the 4-bit hex-code of the key is returned from the subroutine. As an illustration, this information can be directly sent to another output port, such as **PORTD** to light up 4 LEDs, each representing one bit. Putting 100Ω resistors in all keypad circuits ensures that the current is limited at all time. Do not forget to use appropriate resistor for the LED circuits, as well.

Instead of connecting the keypad to the PIC directly and polling the keys through the code, another option would be to use a keypad encoder chip, such as MM74C922N. This chip translates the key-pressed on a 4×4 metric keypad into 4 bit hex-code, which can then be sent to the PIC. Figure 7.8-4 shows the chip pin assignment. The connection between the encoder chip and a 4×4 keypad is also illustrated in Figure 7.8-4a. When no keys are pressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output

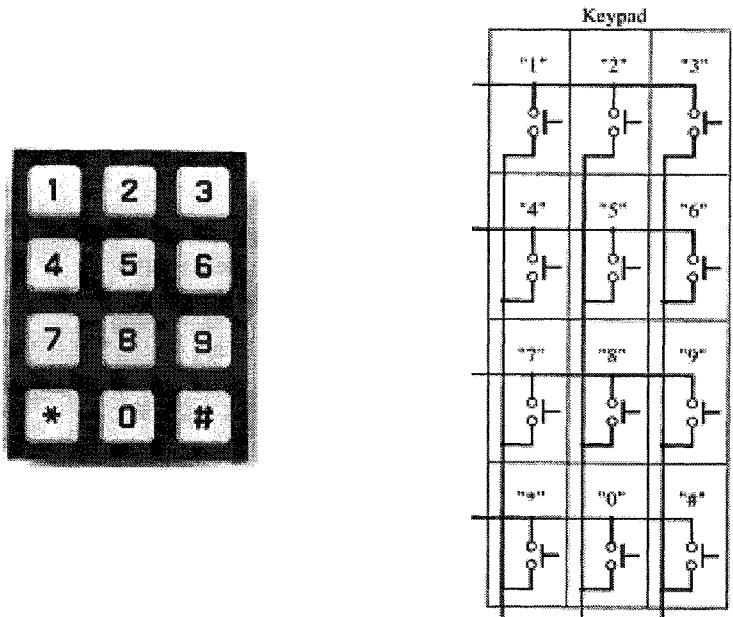


Figure 7.8-1

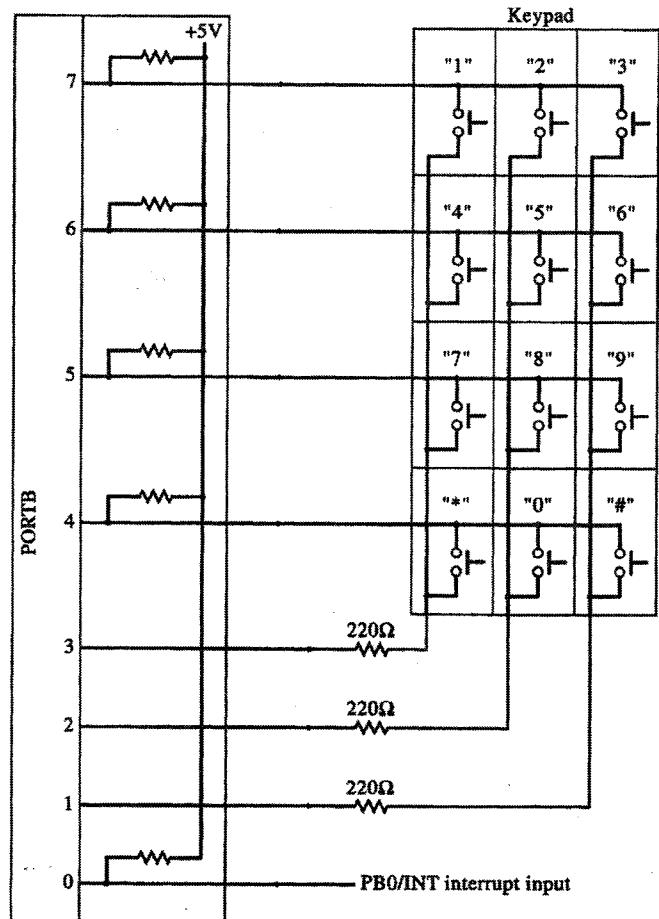


Figure 7.8-2

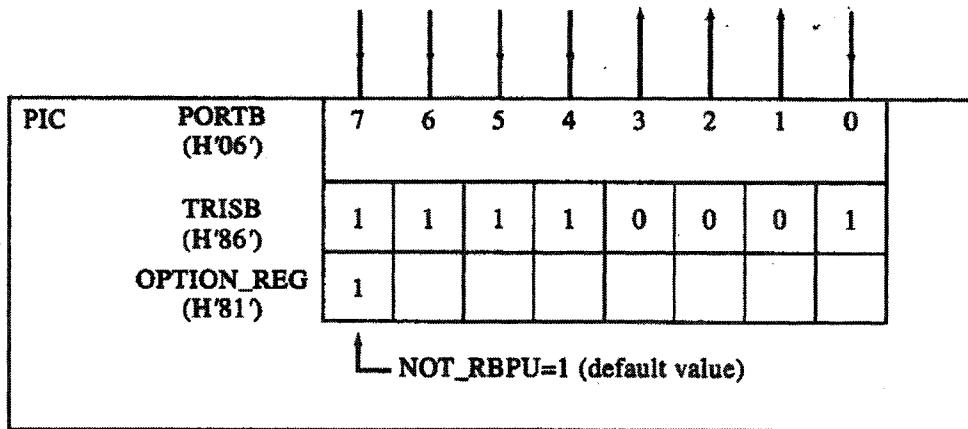


Figure 7.8-3

a logic “0”. These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2-4-bit decoder. When a key is pressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and Y1 will go low. This disables the counter and keeps X1 low. Y1 going low also initiates the key bounce circuit timing and locks out the other Y inputs. The key code to be output is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high. If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry.

The key may bounce several times, but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched. A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized. Both the keypad scan rate and the key debounce period can be controlled by altering the oscillator capacitor, C_{OSE} , and the key bounce-mask capacitor, C_{KBM} .

It is important to note that, the MM74C922N encoder assumes that the keys of the keypad are labeled from 0 to 15 starting with 0 at the 1st row and 1st column and ending with 15 at the 4th row and 4th column, which is compatible with most of the 4×4 keypads. Some other keypads might need some re-mapping in the code.

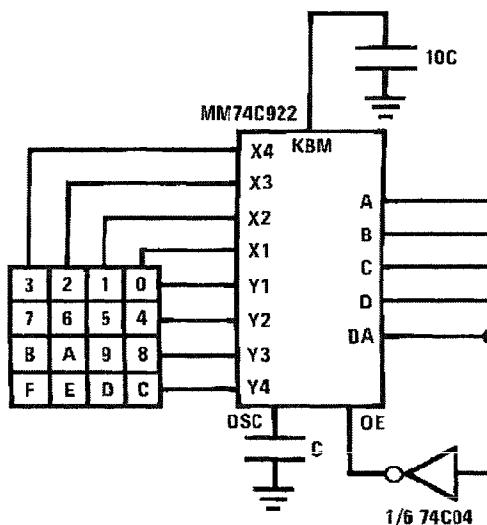


Figure 7.8-4a

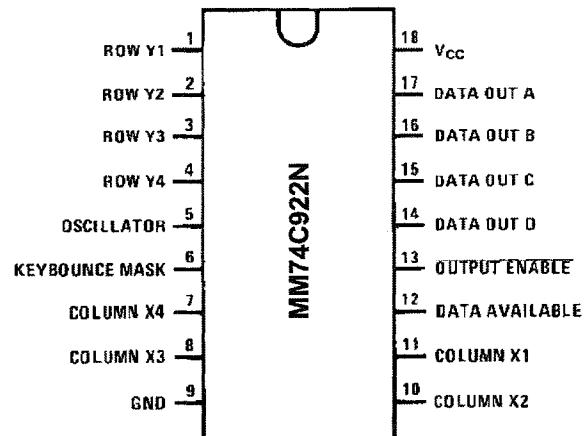


Figure 7.8-4b

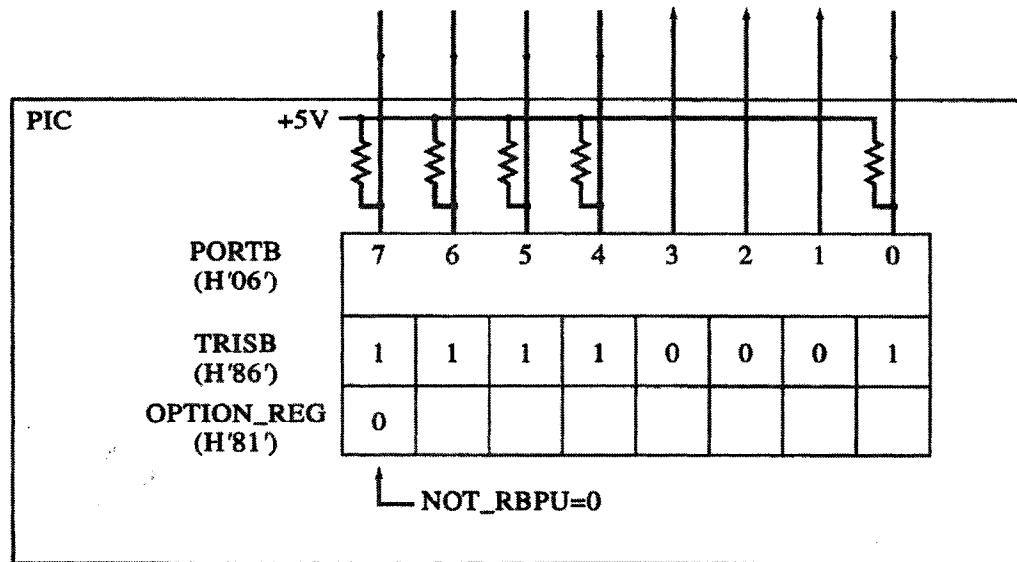


Figure 7.8-5

Interfacing with a keypad can also be done by using the interrupt service on the PIC. The PIC microcomputer has one pin, **RB0/INT**, that serves as its primary external interrupt input. This pin is bit 0 of **PORTB**. Before initializing the interrupt circuitry, **PORTB** itself should be initialized, as exemplified in Figure 7.9-5. The bits of the Bank 1 register **TRISB** set up the corresponding bits of **PORTB** as either inputs or outputs. All of the pins that are set up as inputs can include an optional weak pullup resistor by clearing the **NOT_RBPU** bit of **OPTION_REG**, as shown in Figure 7.8-5. This provides a useful input for a pushbutton switch or for an array of key switches, such as the keypad shown previously.

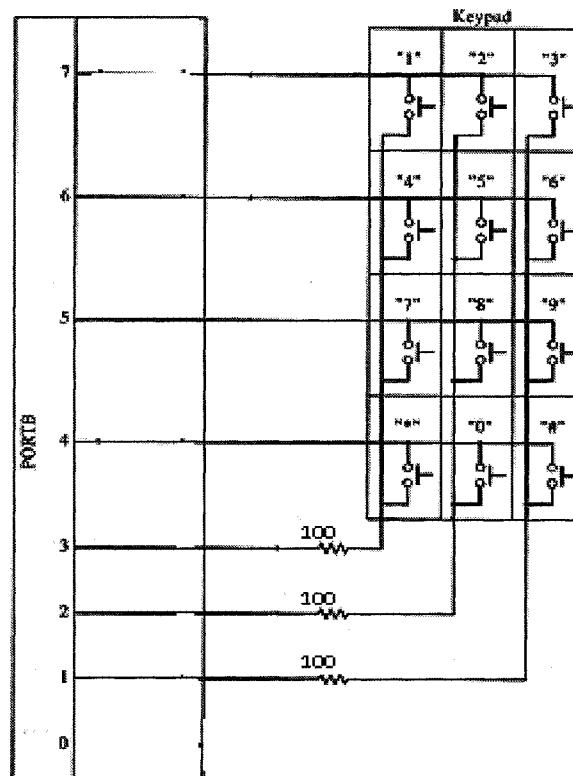


Figure 7.8-6

Figure 7.8-6 shows the circuit for connecting the keypad to the PIC, using the external interrupt input. The internal pullups of the circuit, shown in the figure, hold each input pin high until one of the three key switches attached to it is closed and the corresponding “column driver” output has been driven low. For example, bit 7 of **PORTB** will be driven low if the key switch labeled “1” is pressed and if the bit 3 output from **PORTB** is driven low. Otherwise the internal pullup resistor pulls the bit 7 input high. The figure also illustrates the use of bit 0 of **PORTB** as an interrupt input that can be used independently of the manner in which the other pins of **PORTB** are used. The setup for this independent interrupt input is shown in Figure 7.8-7. The presence or absence of the weak pullup resistor on all **PORTB** inputs is irrelevant to this bit 0 input since the device that drives this interrupt pin will override the weak pullup. The **INTEDG** bit of **OPTION_REG** permits us to set up this input to generate an interrupt on either a rising edge or a falling edge. In addition, when used as an interrupt input, this **PB0/INT** pin is automatically configured as a *Schmitt-trigger* input, triggering on the input edge regardless of its rise (or fall) time.

The **INTCON** register must be initialized with a one in its **INTE** (**RB0/INT** interrupt enable) bit as well as in its **GIE** (global interrupt enable) bit. When the interrupt occurs, there is no need to read **PORTB**. Rather, just poll the **INTF** (interrupt flag) bit of **INTCON** to determine if an edge occurring on this pin is the source of the interrupt. If so, then clear the flag with

```
bcf    INTCON, INTF
```

Then service the interrupt and go back to **IntService**’s polling routine to look for any other pending interrupts.

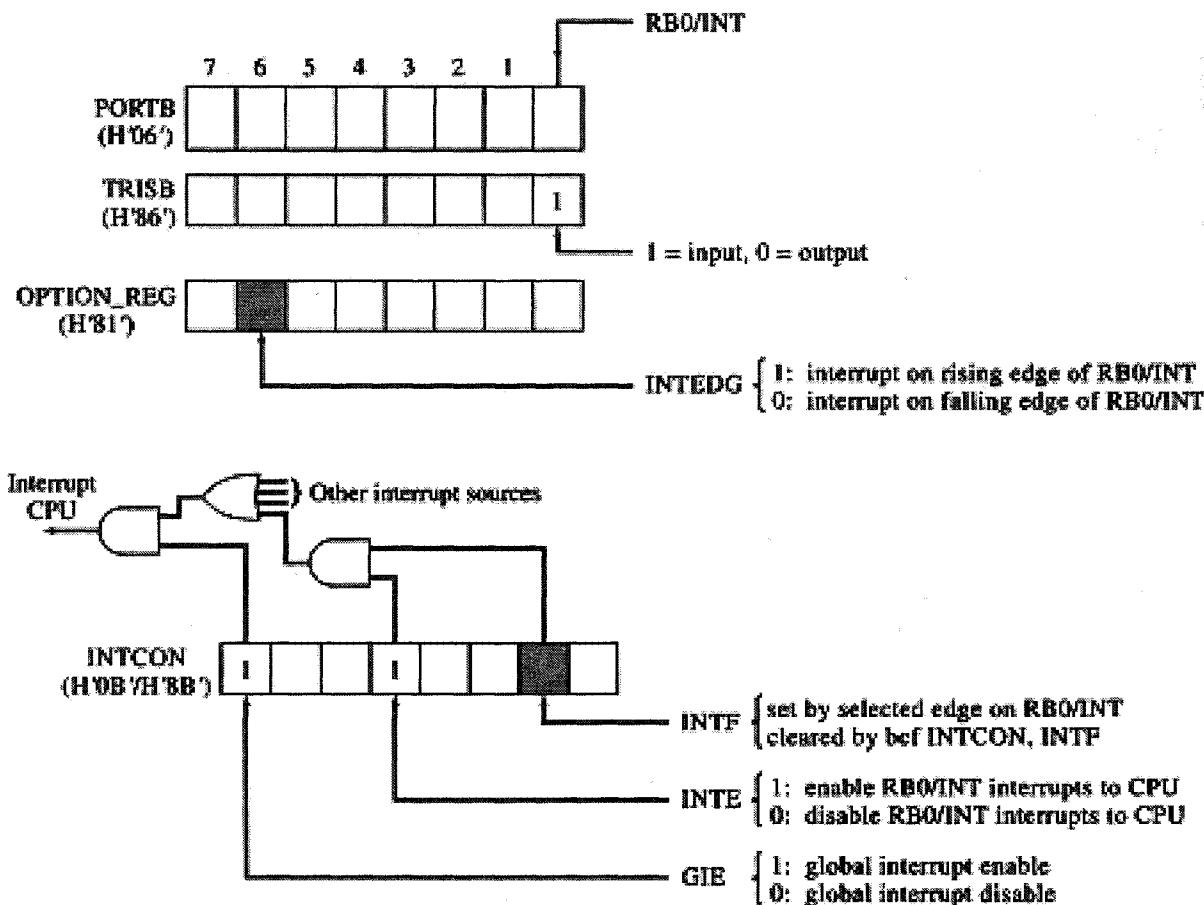


Figure 7.8-7

7.9 Interface with LCD

Dot matrix LCD displays are used to display alphanumeric characters and other symbols. These displays are used in cell phones, calculators, vending machines, and many other devices that provide the user with simple textual information. Dot matrix LCDs are also used in laptop computer screens; however, these displays incorporate special filters, multicolor back lighting, etc. In the Engineering Design course a simple alphanumeric LCD display (plus a keypad), is all one needs to communicate with the PIC.

An alphanumeric LCD screen is usually divided into a number of 5×8 pixel blocks, with vertical and horizontal spaces separating each block. Figure 7.9-1 shows a display with 20 columns and 4 rows of 5×8 pixel blocks. Other standard configurations come with 8, 16, 20, 24, 32, or 40 columns and 1, 2, or 4 rows. To generate a character within a given block requires that each pixel within the block be turned on or off. Now, as you can imagine, to control so many different pixels (electrode segments) requires a great deal of sophistication. For this reason, an intelligent driver IC is required.

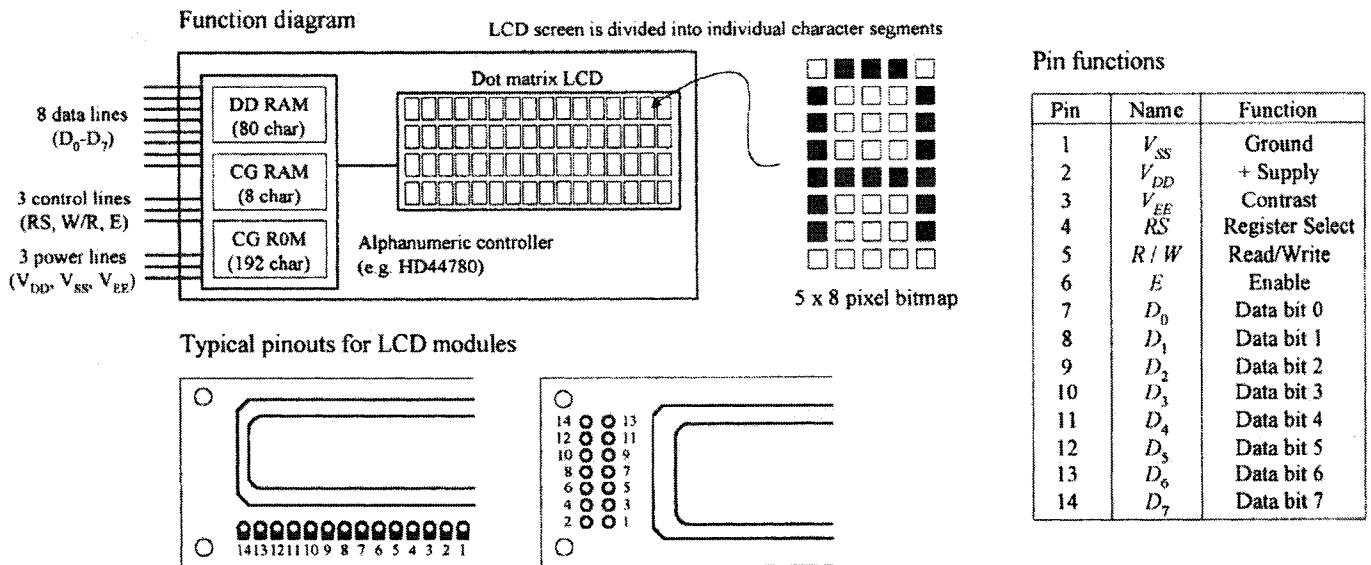


Figure 7.9-1

Almost all alphanumeric LCD modules are controlled by Hitachi's HD44780 (or equivalent, such as LM044L) driver IC. This driver contains a permanent memory (CG ROM) that stores 192 alphanumeric characters, a random access memory (DD RAM) used to store the display's contents, a second random access memory (CC RAM) used to hold custom symbols, input lines for data and instruction control signals, multiplexed outputs for driving LCD pixels, and additional outputs for communicating with expansion chips to drive more LCD pixels. This driver is included the LCD module.

7.9.1 Basic Overview of the Pins

The standard LCD module comes with a 14-pin interface: 8 data lines (D_0-D_7), 3 control lines (RS , W/R , E), and three power lines (V_{DD} , V_{SS} , V_{EE}). V_{DD} (pin 2) and V_{SS} (pin 1) are the module's positive and negative power supply leads. Usually V_{DD} is set to +5 V, while V_{SS} is grounded. V_{EE} (pin 3) is the display's contrast control. By changing the voltage applied to this lead, the contrast of the display increases or decreases. A potentiometer placed between supply voltages, with its wiper connected to V_{EE} , allows for manual adjustment. D_0-D_7 (pins 7-14) are the data bus lines. Data can be transferred to and from the display either as a single 8-bit byte or as two 4-bit nibbles. In the latter case, only the upper four data lines (D_4-D_7) are used. RS (pin 4) is the Register Select line. When this line is low, data bytes transferred to the display module are interpreted as commands, and data bytes read from the display module indicate its status. When the RS line is set high, character data can be transferred to and from the display module. R/W (pin 5) is the Read/Write control line. To write commands or character data to the module, R/W is set low. To read character data or status information from the module R/W is set high. E (pin 6) is the Enable control input, which is used to initiate the actual transfer of command or character data to and from the module. When writing to the display, data on the D_0-D_7 lines is

transferred to the display when the enable input receives a high-to-low transition. When reading from the display, data become available to the D_0 - D_7 lines shortly after a low-to-high transition occurs at the enable input and will remain available until the signal goes low again.

Figure 7.9-2 shows the instruction set and standard set of characters for a LCD module. We will go through some examples illustrating how to use the instructions and how to write characters to the display.

LCD Instruction Set

INSTRUCTION	R/S	R/W	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clear Display	0	0	0	0	0	0	0	0	0	1
Display & Cursor Home	0	0	0	0	0	0	0	0	1	X
Character Entry Mode	0	0	0	0	0	0	0	1	I/D	S
Display & Cursor On/Off	0	0	0	0	0	0	1	I/D	C	B
Display/Cursor Shift	0	0	0	0	0	1	D/C	R/L	X	X
Function Set	0	0	0	0	1	DL	N	F	X	X
Set CGRAM Address	0	0	0	1	A	A	A	A	A	A
Set Display Address	0	0	1	A	A	A	A	A	A	A
Poll the "Busy Flag"	0	0	BF	X	X	X	X	X	X	X
Write Character to Display ^a	1	0	D	D	D	D	D	D	D	D
Read Character on Display ^b	1	1	D	D	D	D	D	D	D	D

- I/D = Increment (I/D = 1)*/Decrement (I/D = 0) each byte written to display
- S = Display shift on (S = 1), Display shift off (S = 0)*
- D = Turn display on (D = 1), Turn display off (D = 0)*
- C = Show cursor (C = 1), Hide cursor (C = 0)
- B = Underline cursor (B = 0, C = 1), Blink cursor (B = 1, C = 1)
- D/C = Move display (D/C = 1), Move cursor (D/C = 0)
- R/L = Direction of shift: Shift right (R/L = 1), Shift left (R/L = 0)
- DL = Set data interface length: 8-bit interface (DL = 1)*, 4-bit interface (DL = 0)
- N = Number of display lines: 2 line mode (N = 1), 1 line mode (N = 0)*
- F = Character font format: 5 x 10 dot (F = 1), 5 x 7 dot (F = 0)*
- BF = Poll the Busy Flag: controller not busy (BF = 0), controller busy (BF = 1)
- A = CGRAM or display address bit
- D = Character data bit
- ^a = Write character to display at the current cursor position
- ^b = Read character on display at the current cursor position
- X = Don't care
- * = Initialization settings

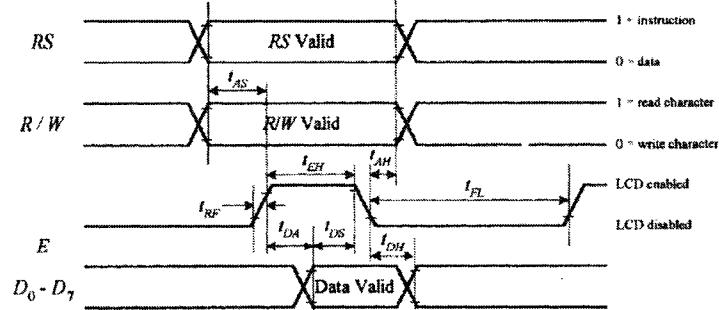
Standard LCD Character Table

RS	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
LSP	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 CG RAM (1)																
1 CG RAM (2)																
2 CG RAM (3)																
3 CG RAM (4)																
4 CG RAM (5)																
5 CG RAM (6)																
6 CG RAM (7)																
7 CG RAM (8)																
8 CG RAM (1)																
9 CG RAM (2)																
A CG RAM (3)																
B CG RAM (4)																
C CG RAM (5)																
D CG RAM (6)																
E CG RAM (7)																
F CG RAM (8)																

Steps used to Read and Write data to and from LCD module (HD44780 controlled)

HD44780 Timing Diagram

Steps for displaying a character	
Set to Write Character to Display mode: $R/W = 0$, $RS = 1$.	
Apply data bits (character code) to D_7 - D_0 .	
Briefly set $E = 1$, then set $E = 0$	



- t_{AS} (Address setup time) - For data inputs to be interpreted correctly, they must be held for a minimum time of t_{AS} (~140ns) prior to the Enable signal.
- t_{EH} (Enable high time) - E must be held HIGH for a minimum of t_{EH} (~450ns) for proper operation.
- t_{DH} (Data hold time) - Data inputs must be held stable for a time t_{DH} (~200ns) prior to the E signal for proper operation.
- t_{FL} (Data fall time) - Control lines RS and R/W lines must not change for a duration of t_{FL} (~10ms) after the E line goes LOW for proper operation.
- t_{DA} (Data hold time) - Data lines D_0 - D_7 must not change for a duration of t_{DA} (~20ns) after the E line goes LOW for proper operation.
- t_{EL} (enable low time) - E line must not be set HIGH again (for the next command), for at least t_{EL} (~500ns) for proper operation.
- t_{RS} (rise and fall time) - Rise and fall times are each ~25ns each.

Figure 7.9-2

7.9.2 Test Circuit

Figure 7.9-3 shows a simple test circuit that is quite useful for learning how to send commands and character data to the LCD module. In this circuit, switches connected to data inputs use pullup resistors in order to supply a high (1) when the switch is open or supply a low (0) when the switch is closed. The enable input receives its high and low levels from a debounced toggle switch. Debouncing the enable switch prevents the likelihood of multiple enable signals being generated. Multiple enable signals tend to create unwanted effects, such as generating the same character over and over again across the display. The 5-kΩ potentiometer is used for contrast. Note that in this circuit we've grounded the *R/W* line, which means we'll only deal with writing to the display.

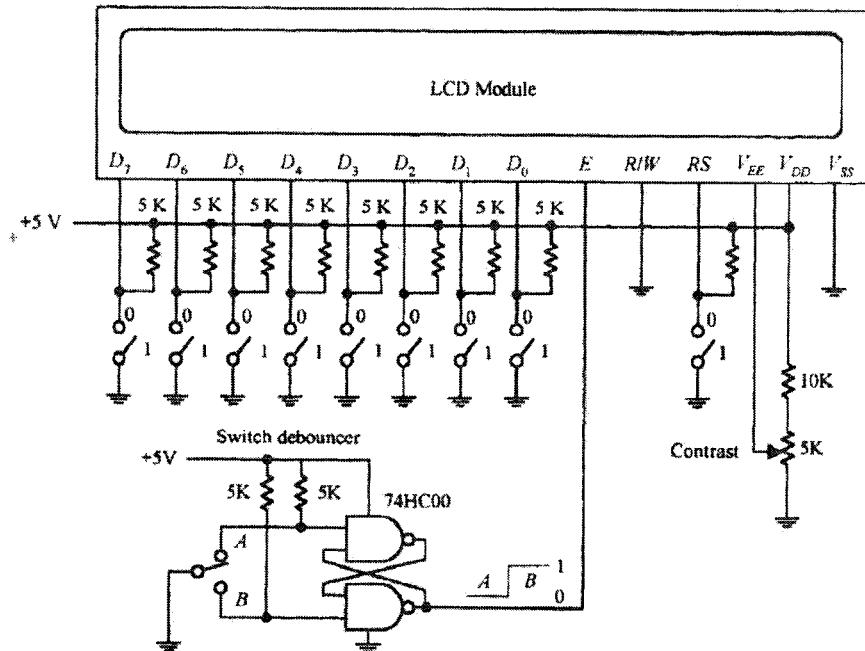


Figure 7.9-3

7.9.3 When Power Is First Applied

When power is first applied to the display, the display module resets itself to its initial settings. Initial settings are indicated in the LCD instruction set with an asterisk. As indicated, the display is actually turned off during the initial setting condition. If we attempt to write character data to the display now, nothing will show up. In order to show something, we must issue a command to the module telling it to turn on its display. According to the instruction set, the Display & Cursor On/Off instruction can be used to turn on the display. At the same time, this instruction also selects the cursor style. For example, if we apply the command code 0000 1111 to D_7-D_0 making sure to keep *RS* low so the module will interpret data as a command, a blinking cursor with underline should appear at the top leftmost position on the display. But before this command can take effect, it must be sent to the module by momentarily setting the Enable line (*E*) low.

Another important instruction that should be implemented after power-up is the Function Set command. When a 2-line display is used, this command tells the module to turn on the second line. It also tells the module what kind of data transfer is going to be used (8-bit or 4-bit; more on this later), and whether a 5 x 10 or 5 x 7 pixel format will be used (5 x 10 is found in some 1-line displays). Assuming that the display used in our example circuit is a 2-line display, we can send the command 0011 1000 telling the display to turn on both lines, use an 8-bit transfer, and provide a 5 x 7 pixel character format. Again, to send this command, we set *RS* low, then supply the command data to D_7-D_0 , and finally pulse *E* low.

Now that the module knows what format to use, we can try writing a character to the display. To do this, we set the module to character mode by setting *RS* high. Next, we apply one of the 8-bit codes listed in the Standard LCD Character

Set table to the data inputs D_7 - D_0 . For example, if we wanted to display the letter *Q*, we'd apply 01010001 (hex 51 or 51_{16}). To send the character data to the LCD module, we pulse *E* low. A *Q* should then appear on the display. To clear the screen, we use the Clear Display command 0000 0001, remembering to keep *RS* low and then pulsing *E* low.

7.9.4 Addressing

After power-up, the module's cursor is positioned to the far-left corner of the first line of the display. This display location is assigned a hexadecimal address of $H'00'$. As new characters are entered, the cursor automatically moves to the right to a new address of $H'01'$, then $H'02'$, etc. Though this automatic incrementing feature makes life easy when entering characters, there are times when it is necessary to set the cursor position to a location other than the first address location.

To set the cursor to another address location, a new starting address must be entered as a command. There are 128 different addresses to choose from, although not all these addresses have their own display location. In fact, there are only 80 display locations laid out on a single line in one-line mode or 40 display locations laid out on each line in two-line mode. Now, as it turns out, not all display locations are necessarily visible on the screen at one time. This will be made more apparent in a moment. Let's first try a simple address example with the LCD module set to two-line mode (provided that two lines are actually available).

To position the cursor to a desired location, we use the Set Address command. This command is specified with the binary code 1000 0000 + (binary value of desired hex address). For example, to send a command telling the cursor to jump to the 07_{16} address location, we would apply $(1000\ 0000 + 0000\ 0111) = 1000\ 0111$ to the D_7 - D_0 inputs, remembering to hold *RS* low and then pulsing *E* low. The cursor should now be located at the eighth position over from the left.

It is important to realize that the relationship between addresses and display locations varies from module to module. Most displays are configured with two lines of characters, the first line starting at address $H'00'$ and the second line at address $H'40'$. Figure 7.9-4 shows the relationship between address and display locations for various LCD modules. Note that the four-line module is really a two-line type with the two lines split, as shown in the figure.

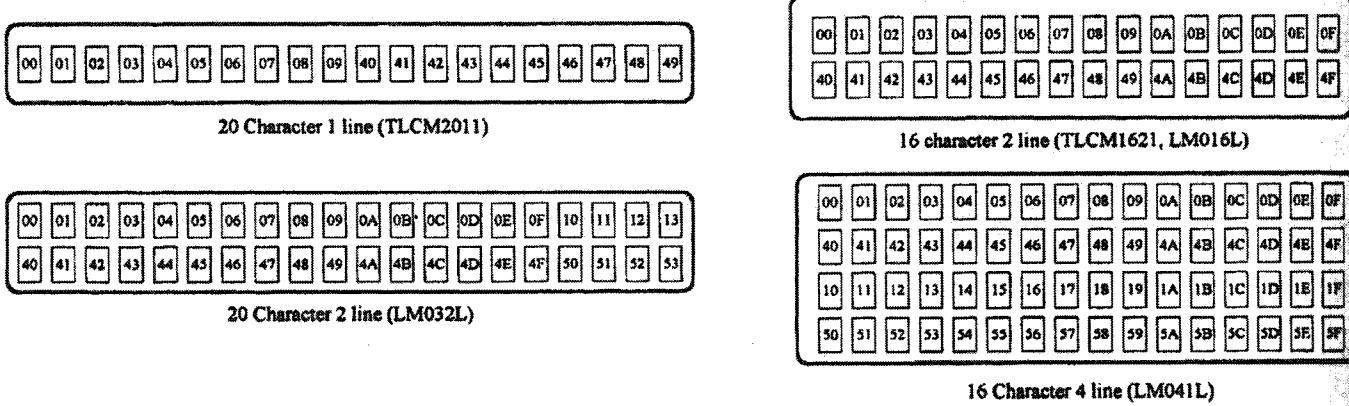


Figure 7.9-4

7.9.5 Shifting the Display

As mentioned before, regardless of size, LCD modules have 80 display locations that can be written to. With smaller displays, not all 80 locations can be displayed at once on the screen. For example, if we were to enter all the letters of the alphabet onto the first line of a 20-character display, only letters *A* through *T* would appear on the screen. Letters *S* through *Z*, along with the cursor, would be "pushed off" to the right of the screen, hidden from view. To bring these hidden characters into view, we can apply the Cursor/Display Shift command to shift all display locations to the left. The command for shifting to the left is 0001 1000. Every time this command is issued, the characters shift one step to the left. In our example, it would take 7 of these commands to bring *T* through *Z* and the cursor into view. To shift things to the right, we apply the command 0001 1100. To bring the cursor back to address $H'00'$ and shift the display address $H'00'$

back to the left-hand side of the display, a Cursor Home command (0000 0010) can be issued. Another alternative is to use the Clear Display command 0000 0001. However, this command also clears all display locations.

7.9.6 Character Entry Mode

If you do not want to enter characters from left to right, you can use the Character Entry Mode to enter characters from right to left. To do this, the cursor must first be sent to the rightmost display location on the screen. After that, the Character Entry Mode command 0000 0111 is entered into the module. This sets the entry mode to auto-increment/display shift left. Now, when characters are entered, they appear on the right-hand side, while the display shifts left for each character entered.

7.9.7 Interface with the PIC

The LCD module can be connected to the PIC through either 8-bit or 4-bit data transfer mode, as illustrated in Figure 7.9-5. In 8-bit data transfer mode, there is a one-to-one mapping between the LCD data pins and the PIC pins. Hence, transferring the data back and forth is straightforward. In 4-bit mode, only data lines D_4-D_7 are used. The other four lines, D_0-D_3 , are left either floating or tied to the power supply. To send data to the display requires sending two 4-bit chunks instead of one 8-bit word.

When power is first applied, the module is set up for 8-bit transfer. Ordinarily, to set up 4-bit transfer, the Function Set command with binary value 0010 0000 is sent to the display. Note that since there are only 4 data lines in use, all 8 bits cannot be sent. However, this is not a problem, since the 8-bit/4-bit selection is on data bit D_4 . From now on, 8-bit character and command bits must be sent in two halves, the first 4 most significant bits and then the remaining 4 bits. For example, to write character data 0100 1110 to the display would require setting RS high, applying 0100 to the data lines, pulsing E low, then applying 1110 to the data lines, and pulsing E low again.

However, it is also possible that the PIC was reset without the LCD being reset. In this case, it is possible that the LCD is in 4-bit mode, so that it is expecting two data chunks instead of one when the PIC attempts to perform initialization – or it even might be expecting the second chunk of a previous instruction. To put the LCD into a known state, it is necessary to first set it to 8-bit mode, by sending the sequence '0011' three times. If it is in 4-bit mode, and half of an instruction had been previously sent, the first '0011' will complete the previous instruction, and the next two will form a 'set to 8-bit mode' instruction. If it is in 4-bit mode, and beginning a new instruction, the first two '0011' will form a 'set to 8-bit mode' instruction, and the next '0011xxxx' (4 of the data lines are floating) sequence just forms another 'set to 8-bit mode' instruction. And, if it was already in 8-bit mode, then all three '0011' sequences will repeatedly set the LCD to 8-bit mode. Once this is completed, the LCD can be safely set to 4-bit mode as previously described.

The 4-bit transfer is frequently used when the LCD module is interfaced with a microcontroller that has limited I/O lines.

7.9.7.1 PIC16F877 Sample Code

```

;*****
; Test LCD
; Assembler : mpasm.exe
; Company   : ETT CO., LTD.
;*****  

list p=16f877           ; list directive to define processor
#include <p16f877.inc>      ; processor specific variable definitions  

_CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC &

```

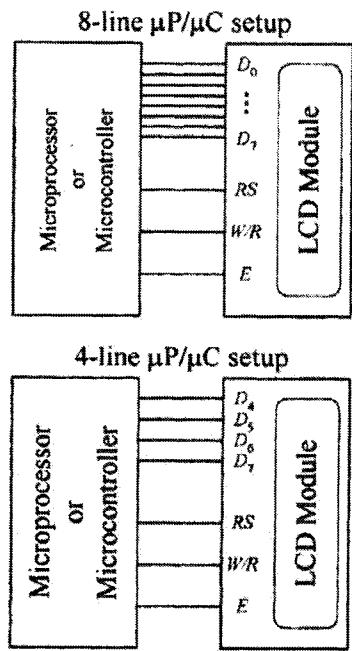


Figure 7.9-5

```

        _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_OFF & _CPD_OFF
#define RS      PORTD,2
#define E       PORTD,3

com      EQU      0x20          ; buffer for Instruction
dat      EQU      0x21          ; buffer for data
count1   EQU      0x22
count2   EQU      0x23
count3   EQU      0x24

        ORG      0x0000

;***** initial *****
init      bsf      STATUS,RP0      ; select bank 1
          clrf     TRISA          ; All port A is output
          clrf     TRISC
          clrf     TRISD          ; All port D is output
          bcf      STATUS,RP0      ; select bank 0
          call     delay
          call     delay
          movlw    B'00110011'      ; First two 0011 sequences
          call     WR_INS          ; (WR_INS sends two 4-bit chunks)
          movlw    B'00110010'      ; Third 0011 sequence, then set to 4-bit (0010xxxx)
          call     WR_INS
          movlw    B'00101000'      ; 4 bits, 2 lines,5X7 dot
          call     WR_INS
          movlw    B'00001100'      ; display on/off
          call     WR_INS
          movlw    B'00000110'      ; Entry mode
          call     WR_INS
          movlw    B'00000001'      ; Clear ram
          call     WR_INS

          movlw    "H"
          call     WR_DATA
          movlw    "E"
          call     WR_DATA
          movlw    "L"
          call     WR_DATA
          movlw    "L"
          call     WR_DATA
          movlw    "O"
          call     WR_DATA
          movlw    " "
          call     WR_DATA
          movlw    "W"
          call     WR_DATA
          movlw    "O"
          call     WR_DATA
          movlw    "R"
          call     WR_DATA
          movlw    "L"
          call     WR_DATA
          movlw    "D"
          call     WR_DATA

          goto    $                  ;Stall here (goto this line)

;*****
; Write command to LCD - Input : W , output : -
;*****
WR_INS   bcf      RS          ; clear RS

```

```

        movwf    com      ; W --> com
        andlw    0xF0      ; mask 4 bits MSB  W = X0

        movwf    PORTD    ; Send 4 bits MSB
        bsf      E         ;
        call     delay    ; ;____|____|
        bcf      E         ; ;____|____|
        swapf   com,w
        andlw   0xF0      ; 1111 0010
        movwf   PORTD    ; send 4 bits LSB
        bsf      E         ;
        call     delay    ; ;____|____|
        bcf      E         ; ;____|____|
        call     delay
        return

;***** *****
; Write data to LCD - Input : W , Output : -
;***** *****
WR_DATA  bsf      RS
        movwf   dat
        movf    dat,w
        andlw   0xF0
        addlw   4
        movwf   PORTD
        bsf      E
        call    delay    ; ;____|____|
        bcf      E         ; ;____|____|
        swapf   dat,w
        andlw   0xF0
        addlw   4
        movwf   PORTD
        bsf      E         ;
        call    delay    ; ;____|____|
        bcf      E         ; ;____|____|
        return
;***** *****
; Delay
;***** *****
        cblock 0x30
        d1
        d2
        endc

delay
        ;249993 cycles
        movlw   0x4E
        movwf   d1
        movlw   0xC4
        movwf   d2
delay_0
        decfsz d1, f
        goto   $+2
        decfsz d2, f
        goto   delay_0

        ;3 cycles
        goto   $+1
        nop

        ;4 cycles (including call)
return
END

```

7.9.7.2 PIC16F887 Sample Code

The following sample code demonstrates the program structure for PIC16F887. It performs the same function as the sample code in section PIC16F877 Sample Code 7.9.7.1.

```
;*****  
; Test LCD  
; Assembler : mpasm.exe  
; Company   : ETT CO., LTD.  
*****  
list p=16f887           ; list directive to define processor  
#include <p16f887.inc>    ; processor specific variable definitions  
  
_CONFIG _CONFIG1, _PWRTE_ON & _WDT_OFF & _INTOSC & _BOR_ON & _LVP_OFF & _CP_OFF &  
_MCLRE_ON & _IESO_OFF & _FCMEN_OFF  
  
_CONFIG _CONFIG2, _WRT_OFF & _BOR40V  
  
#define RS    PORTD,2  
#define E     PORTD,3  
  
com    EQU 0x20          ; buffer for Instruction  
dat    EQU 0x21          ; buffer for data  
count1 EQU 0x22  
count2 EQU 0x23  
count3 EQU 0x24  
  
        ORG 0x0000  
  
;***** initial *****  
  
init    bsf STATUS,RP0    ; select bank 1  
        clrf TRISA          ; All port A is output  
        clrf TRISC          ; All port D is output  
        bsf STATUS,RP1      ; select bank 3  
        clrf ANSEL          ; Initialize ANSEL to make all port A, port E digital  
        clrf ANSELH         ; Initialize ANSELH to make all port B digital  
        bcf STATUS,RP1      ; select bank 1  
        bcf STATUS,RP0      ; select bank 0  
        call delay          ;  
        call delay          ;  
        movlw B'00110011'    ; First two 0011 sequences  
        call WR_INS          ; (WR_INS sends two 4-bit chunks)  
        movlw B'00110010'    ; Third 0011 sequence, then set to 4-bit (0010xxxx)  
        call WR_INS          ;  
        movlw B'00101000'    ; 4 bits, 2 lines, 5x7 dot  
        call WR_INS          ;  
        movlw B'00001100'    ; display on/off  
        call WR_INS          ;  
        movlw B'00000110'    ; Entry mode  
        call WR_INS          ;  
        movlw B'00000001'    ; Clear ram  
        call WR_INS          ;  
  
        movlw "H"            ;  
        call WR_DATA         ;  
        movlw "E"            ;  
        call WR_DATA         ;  
        movlw "L"            ;  
        call WR_DATA         ;  
        movlw "L"            ;
```

```

call      WR_DATA
movlw    "O"
call      WR_DATA
movlw    "
call      WR_DATA
movlw    "W"
call      WR_DATA
movlw    "O"
call      WR_DATA
movlw    "R"
call      WR_DATA
movlw    "L"
call      WR_DATA
movlw    "D"
call      WR_DATA

goto     $           ;Stall here (goto this line)

;***** *****
; Write command to LCD - Input : W , output : -
;***** *****
WR_INS  bcf      RS      ; clear RS
        movwf    com    ; W --> com
        andlw   0xF0    ; mask 4 bits MSB W = X0

        movwf    PORTD   ; Send 4 bits MSB
        bsf      E       ;
        call     delay   ;
        bcf      E       ; —|__|—
        swapf   com,w   ;
        andlw   0xF0    ; 1111 0010
        movwf    PORTD   ; send 4 bits LSB
        bsf      E       ;
        call     delay   ;
        bcf      E       ; —|__|—
        call     delay   ;
        return

;***** *****
; Write data to LCD - Input : W , Output : -
;***** *****
WR_DATA bsf      RS
        movwf    dat
        movf    dat,w
        andlw   0xF0
        addlw   4
        movwf    PORTD
        bsf      E
        call     delay   ;
        bcf      E       ; —|__|—
        swapf   dat,w
        andlw   0xF0
        addlw   4
        movwf    PORTD
        bsf      E       ;
        call     delay   ;
        bcf      E       ; —|__|—
        return
;***** *****
; Delay
;***** *****
        cblock 0x30
        d1

```

```

d2
endc

delay
        ;249993 cycles
    movlw 0x4E
    movwf d1
    movlw 0xC4
    movwf d2
delay_0
    decfsz d1, f
    goto $+2
    decfsz d2, f
    goto delay_0

        ;3 cycles
    goto $+1
    nop

        ;4 cycles (including call)
return
END

```

7.9.7.3 PIC16F1937 Sample Code

The following sample code demonstrates the program structure for PIC16F1937. It performs the same function as the sample code in section PIC16F877 Sample Code 7.9.7.1.

```

;*****
; Test LCD
; Assembler : mpasm.exe
; Company : ETT CO.,LTD.
;*****
list p=16f1937          ; list directive to define processor
#include <p16f1937.inc>    ; processor specific variable definitions

__CONFIG _CONFIG1, _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _MCLRE_ON & _CP_OFF & _CPD_OFF &
_BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_ON

__CONFIG _CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _BORV_19 & _LVP_OFF

#define RS      PORTD,2
#define E       PORTD,3

com    EQU 0x20          ; buffer for Instruction
dat    EQU 0x21          ; buffer for data
count1 EQU 0x22
count2 EQU 0x23
count3 EQU 0x24

ORG    0x0000

;***** initial *****
init   movlb d'1'          ; select bank 1
        clrf TRISA          ; All port A is output
        clrf TRISC
        clrf TRISD          ; All port D is output

```

```

        clrf    ADCON1      ; Turn off A/D conversion
        clrf    ADCON0
        movlb  d'3'          ; select bank 3
        clrf    ANSELA        ; Initialize ANSELA to make all port A digital
        clrf    ANSELB        ; Initialize ANSELB to make all port B digital
        clrf    ANSELD        ; Initialize ANSELD to make all port D digital
        clrf    ANSELE        ; Initialize ANSELE to make all port E digital
        clrf    BSR           ; select bank 0
        call    delay
        call    delay
        movlw  B'00110011'   ; First two 0011 sequences
        call    WR_INS        ; (WR_INS sends two 4-bit chunks)
        movlw  B'00110010'   ; Third 0011 sequence, then set to 4-bit (0010xxxx)
        call    WR_INS
        movlw  B'00101000'   ; 4 bits, 2 lines, 5X7 dot
        call    WR_INS
        movlw  B'00000110'   ; display on/off
        call    WR_INS
        movlw  B'00000010'   ; Entry mode
        call    WR_INS
        movlw  B'00000001'   ; Clear ram
        call    WR_INS

        movlw      "H"
        call     WR_DATA
        movlw      "E"
        call     WR_DATA
        movlw      "L"
        call     WR_DATA
        movlw      "L"
        call     WR_DATA
        movlw      "O"
        call     WR_DATA
        movlw      " "
        call     WR_DATA
        movlw      "W"
        call     WR_DATA
        movlw      "O"
        call     WR_DATA
        movlw      "R"
        call     WR_DATA
        movlw      "L"
        call     WR_DATA
        movlw      "D"
        call     WR_DATA

        goto    $           ; Stall here (goto this line)

;*****
; Write command to LCD - Input : W , output : -
;*****
WR_INS  bcf    RS      ; clear RS
        movwf  com    ; W --> com
        andlw  0xF0    ; mask 4 bits MSB W = X0

        movwf  PORTD   ; Send 4 bits MSB
        bsf    E       ;
        call   delay   ;
        bcf    E       ; — |__| —
        swapf com,w
        andlw  0xF0    ; 1111 0010
        movwf  PORTD   ; send 4 bits LSB
        bsf    E       ;

```

```

call      delay      ; —|__|—
bcf      E          ; —|__|—
call      delay
return

;***** *****
; Write data to LCD - Input : W , Output : -
;***** *****
WR_DATA  bsf      RS
        movwf   dat
        movf    dat,w
        andlw   0xF0
        addlw   4
        movwf   PORTD
        bsf      E
        call    delay      ; —|__|—
        bcf      E          ; —|__|—
        swapf   dat,w
        andlw   0xF0
        addlw   4
        movwf   PORTD
        bsf      E          ;
        call    delay      ; —|__|—
        bcf      E          ; —|__|—
        return
;***** *****
; Delay
;***** *****
        cblock 0x30
        d1
        d2
        endc

delay
        ;249993 cycles
        movlw   0x4E
        movwf   d1
        movlw   0xC4
        movwf   d2
delay_0
        decfsz d1, f
        goto   $+2
        decfsz d2, f
        goto   delay_0

        ;3 cycles
        goto   $+1
        nop

        ;4 cycles (including call)
return
END

```

7.9.7.4 PIC18 Sample Code

The following sample code demonstrates the program structure for PIC18. It performs the same function as the sample code in section PIC16F877 Sample Code 7.9.7.1.

```
;***** *****

```

```

; Test LCD
; Displays characters to the LCD
; PIC18F4620
; Clock Speed: 10 MHz
;*****Configuration Bits*****
#include <p18f4620.inc>
list P=18F4620, F=INHX32, C=160, N=80, ST=OFF, MM=OFF, R=DEC

;*****Constant Defines*****
CONFIG OSC=HS, FCMEN=OFF, IESO=OFF
CONFIG PWRT = OFF, BOREN = SBORDIS, BORV = 3
CONFIG WDT = OFF, WDTPS = 32768
CONFIG MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF, CCP2MX = PORTC
CONFIG STVREN = ON, LVP = OFF, XINST = OFF
CONFIG DEBUG = OFF
CONFIG CPO = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF
CONFIG CPB = OFF, CPD = OFF
CONFIG WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
CONFIG WRTB = OFF, WRTC = OFF, WRTD = OFF
CONFIG EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF
CONFIG EBTRB = OFF

;*****Vectors*****
org 0x0000
goto Mainline
org 0x08 ; high priority ISR
retfie
org 0x18 ; low priority ISR
retfie

Table
db "HELLO WORLD", 0 ;Declare Table
Mainline
clrfa
clrfa
clrfa
clrfa
call delay5ms ; wait for LCD to start up
call delay5ms
movlw B'00110011'
call WR_INS
movlw B'00110010'
call WR_INS
movlw B'00101000' ; 4 bits, 2 lines, 5x7 dot
call WR_INS
movlw B'00001100' ; display on/off
call WR_INS
movlw B'00000110' ; Entry mode
call WR_INS
movlw B'00000001' ; Clear ram
call WR_INS

ReadTable
movlw upper Table ; Load the Table Pointer
movwf TBLPTRU ; with full address of Table
movlw high Table
movwf TBLPTRH
movlw low Table
movwf TBLPTRL

```

```

tblrd*           ; Read first Table entry into TABLAT
    movf    TABLAT, W
Again   call    WR_DATA
        ; Read Table Loop
        ; Write to LCD
tblrd+*         ; Increment TBLPTR then read
    movf    TABLAT, W
    bnz    Again
Stop    bra    Stop
        ; Branch back to loop if Table does not return 0
        ; End of Program

;*****
; Write command to LCD
; Input : W
; output : -
;*****
WR_INS
    bcf    RS           ; clear Register Status bit
    movwf  temp_lcd
    andlw  0xF0
    movwf  LATD
    bsf    E            ; mask 4 bits MSB
    swapf temp_lcd, WREG
    andlw  0xFO
    bcf    E            ; send 4 bits MSB
    movwf  LATD
    bsf    E            ; pulse enable high
    nop
    bcf    E
    call   delay5ms
    return

;*****
; Write data to LCD
; Input : W
; Output : -
;*****
WR_DATA
    bcf    RS           ; clear Register Status bit
    movwf  dat
    movf   dat, WREG
    andlw  0xF0
    addlw  4            ; store character
    movwf  PORTD
    bsf    E            ; mask 4 bits MSB
    swapf dat, WREG
    andlw  0xFO
    bcf    E            ; set Register Status
    addlw  4            ; send 4 bits MSB
    movwf  PORTD
    bsf    E            ; pulse enable high
    nop
    bcf    E
    call   delay44us
    return
        ; Wait for LCD to process

;*****
; delay44us (): wait exactly 110 cycles (44 us)
;*****
delay44us
    movlw   0x23
    movwf   delay1, 0

Delay44usLoop
    decfsz delay1, f
    bra    Delay44usLoop
    return

;*****
; delay5ms : wait for 5 ms
;*****
delay5ms
    movlw   0xC2

```

```

movwf  delay1,0
movlw  0x0A
movwf  delay2,0

Delay5msLoop
    decfsz delay1, f
    bra    d2
    decfsz delay2, f
d2    bra    Delay5msLoop
    return
end

```

There are several key differences between this code and the code for PIC16s, although most of the structure and syntax are the same. First, notice that the include file and the device list parameter are set for PIC18F4620. The PIC18 supports the hex file format INHX32, which is different from the format of PIC16. The configuration settings for PIC18 are quite extensive, since it is split into 7 words instead of one. For most cases, the configurations given in the sample code will suffice, but users should familiarize themselves with all the configuration settings described in section 7.4.2.2.

There are three vectors present near the start of the code. Notice the two interrupt vectors, high priority and low priority at address 0x08 and 0x18 respectively. Interrupts have not been enabled in this program, so the Interrupt Service Routines will not be called. The next section contains tables. The tables are saved in program memory byte by byte with the **db** directive, ending with 0. The table is retrieved with the table read instruction, which is overall a more efficient method compared to the PIC16 method of tabling.

7.10 Capture/Compare/PWM (CCP) Module

Each Capture/Compare/PWM (CCP) module contains a 16-bit register that can operate as a Capture, Compare, or PWM register. A standard CCP module has the following capabilities:

- **Input Capture:** Captures the 16 bit value of a timer when an event occurs on a CCP pin.
- **Output Compare:** Generate a signal on the CCP pin at a specified time.
- **Pulse Width Modulation:** 2 Pulse Width Modulated Outputs (10 bit accuracy).

There are two CCP modules, CCP1 and CCP2. Both the CCP1 and CCP2 modules are identical in operation, with the exception being the operation of the special event trigger. In the following sections, the operation of a CCP module is described with respect to CCP1. CCP2 operates the same as CCP1, except where noted. CCP1 Module: Capture/Compare/PWM Register1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. The special event trigger is generated by a compare match and will reset Timer1. Other CCP Modules are identical to CCP1 except that a special event trigger will reset Timer1 and start an A/D conversion (if A/D module is enabled).

For PIC16F877/887 and PIC18, the CCP module has 6 registers:

- Capture/Compare/PWM Register1 (LSB) (CCPR1L)
- Capture/Compare/PWM Register1 (MSB) (CCPR1H)
- CCP Control Register1 (CCP1CON)
- Capture/Compare/PWM Register2 (LSB) (CCPR2L)
- Capture/Compare/PWM Register2 (MSB) (CCPR2H)
- CCP Control Register2 (CCP2CON)

For PIC16F1937, the CCP module had 15 registers; 5 CCPs, each with 3 registers:

- Capture/Compare/PWM Registerx (LSB) (CCPRxL) (x = 1,2,3,4,5)
- Capture/Compare/PWM Registerx (MSB) (CCPRxH) (x = 1,2,3,4,5)
- CCP Control Registerx (CCPxCON) (x = 1,2,3,4,5)

7.10.1 Control Registers

7.10.1.1 CCPxCON Register

--	--	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7:6 **Unimplemented:** Read as '0'

bit 5:4 **CCPxX:CCPxY:** PWM Least Significant bits

Capture mode:

Unused

Compare mode:

Unused

PWM mode:

These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3:0 **CCPxM3:CCPxM0:** CCPx Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)

0100 = Capture mode, every falling edge

0101 = Capture mode, every rising edge

0110 = Capture mode, every 4th rising edge

0111 = Capture mode, every 16th rising edge

1000 = Compare mode, set output on match (CCPxIF bit is set)

1001 = Compare mode, clear output on match (CCPxIF bit is set)

1010 = Compare mode, generate software interrupt on match (CCPxIF bit is set, CCPx pin is unaffected)

1011 = Compare mode, trigger special event (CCPxIF bit is set, CCPx pin is unaffected); CCP1 resets

 TMR1; CCP2 resets TMR1 and starts an A/D conversion (if A/D module is enabled)

11xx = PWM mode

7.10.1.2 T2CON Register

--	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0
bit 7	Unimplemented: Read as '0'						
bit 6:3	TOUTPS3:TOUTPS0: Timer2 Output Postscale Select bits						
	0000 = 1:1 Postscale						
	0001 = 1:2 Postscale						
	0010 = 1:3 Postscale						
	•						
	•						
	•						
	1111 = 1:16 Postscale						
bit 2	TMR2ON: Timer2 On bit						
	1 = Timer2 is on						
	0 = Timer2 is off						
bit 1:0	T2CKPS1:T2CKPS0: Timer2 Clock Prescale Select bits						
	00 = Prescaler is 1						
	01 = Prescaler is 4						
	1x = Prescaler is 16						

7.10.2 Capture Mode

Capture mode is useful in many applications. It is capable of recording the time of an event arrival. It also can be used at measuring periods as well as pulse-width measurement. It is capable of generating interrupt and counting events. If a second CCP module is operating in compare mode, the capture mode can also be used as time reference or duty cycle measurement.

In Capture mode, CCP1H:CCP1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as one of the following:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

The type of event is configured by control bits CCP1M3:CCP1M0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set. The interrupt flag must be cleared in software. If another capture occurs before the value in register CCP1 is read, the old captured value is overwritten by the new value.

In capture mode, the CCP pins should have proper configuration. The RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit. Timer1 must be running in Timer mode, or Synchronized Counter mode, for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work. When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit CCP1IF, following any such change in operating mode.

7.10.3 Compare Mode

Compare mode is capable of generating a single pulse, a train of pulses or a periodic waveform. It can be used to start ADC conversion. If a second CCP module is operating in compare mode at the same time they can be used together to set time references.

In Compare mode, the 16-bit CCP1 register value is constantly compared against the TMR1 register pair value. When a match occurs, the RC2/CCP1 pin is:

- Driven high
- Driven low
- Remains unchanged

The action on the pin is based on the value of control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). At the same time, interrupt flag bit CCP1IF is set.

Timer1 must be running in Timer mode, or Synchronized Counter mode, if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work. When Generate Software Interrupt mode is chosen, the

CCP1 pin is not affected. The CCP1IF bit is set, causing a CCP interrupt (if enabled). In this mode, an internal hardware trigger is generated, which may be used to initiate an action. The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCP1 register to effectively be a 16-bit programmable period register for Timer1. The special event trigger output of CCP2 resets the TMR1 register pair and starts an A/D conversion (if the A/D module is enabled).

7.10.4 PIC PWM Modules

Pulse Width Modulation (PWM) module, which produces digital waveforms, can be used as digital-to-analog (D/A) converters and acts as an analog output of the microcontroller. This module can generate up to a 10-bit resolution PWM output with programmable PWM period and duty cycle.

The PWM module involves the configuration of 4 registers:

- **Timer2 Module's Period Register (PR2)** : contains the PWM period.
- **Capture/Compare/PWM Registerx (CCPRxL)** : contains the eight most-significant-bits of the PWM duty cycle.
- **Capture/Compare/PWM Control Registerx (CCPxCON)** : contains the 2 least-significant-bits of the PWM duty cycle, and controls the CCP operation mode.
- **Timer2 Control Register (T2CON)** : controls the operation of Timer2.

7.10.4.1 Timing

7.10.4.1.1 PWM Period

The PWM period is specified by writing to the PR2 register and it can be calculated by the following formula:

$$\text{PWM period} = [(PR2) + 1] * 4 * T_{osc} * (\text{TMR2 prescale value})$$

7.10.4.1.2 PWM Duty Cycle

The PWM duty cycle is specified by writing the eight MSBs to the CCPRxL register and the two LSBs to the CCPxCON<5:4> bits. Up to 10-bit resolution is available and this 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The PWM duty cycle in time can be calculated using the following equation:

$$\text{PWM active high duration (t)} = (\text{CCPRxL:CCPxCON<5:4>}) * T_{osc} * (\text{TMR2 prescale value})$$

$$\text{Duty Cycle (\%)} = \frac{t}{T_{PWM}} * 100\%$$

The PWM duty cycle can be changed at any time by writing to CCPR1L and CCP1CON<5:4>. However, the duty cycle value will not be updated until the end of current PWM period.

7.10.4.1.3 Resolution

The maximum PWM resolution (bits) for a given PWM frequency is given by the formula:

$$resolution = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

7.10.4.1.4 Example

With a 20MHz oscillator, set PR2 to be 255 and the timer prescaler to be 1,

- ⇒ $T_{OSC} = 0.05\mu s$
- ⇒ PWM period = $(255 + 1) * 4 * 0.05 * 1 = 51.2\mu s$
- ⇒ PWM frequency = 19.53kHz
- ⇒ Maximum resolution = 10bits

7.10.4.2 Setup

The following steps should be taken when configuring the PWM module operation:

1. Set the PWM period:
 - Write the period to *PR2*
2. Set the PWM duty cycle:
 - Write the 10-bit duty cycle time to *CCPRxL:CCPxCON<5:4>*
3. Make the CCPx pin an output
 - Clear *TRISC<2>* for CCP1, *TRISC<1>* for CCP2
4. Set TMR2 prescale value and enable Timer2
 - Configure *T2CON*
5. Configure CCPx module for PWM operation
 - Configure *CCPxCON<3:0>*

7.10.4.3 Analog Output Program (PIC16F877)

The following program uses the PWM module to generate a 78.12kHz digital wave with 50% duty cycle. This digital wave is outputted through the CCP1 pin (pin 17) of the PIC16F877 microcontroller. The measured DC voltage at the output would be about 2.5V.

```
list P=PIC16F877, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F877.INC"
_config ( _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC &
          _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_OFF & _CPD_OFF )
errorlevel -302

ORG      0
GOTO    INIT

INIT    BSF      STATUS,RP0      ;select bank 1
        BCF      INTCON,GIE     ;disable global interrupt
        BCF      INTCON,PEIE
        MOVLW   B'00111111'    ;configure PR2
        MOVWF   PR2
        BCF      STATUS,RP0      ;select bank 0
        MOVWF   CCP1CON
```

```

BSF      STATUS,RP0      ;select bank 1
CLRF     TRISC          ;configure PORTC as output
BCF      STATUS,RP0      ;select bank 0
MOVLW    B'00000100'    ;configure T2CON
MOVWF    T2CON
MOVLW    B'00110010'    ;Set the duty cycle to 50%
MOVWF    CCPR1L

ENDLP   GOTO    ENDLP      ;endless loop

END

```

7.10.4.4 Analog Output Program (PIC16F887)

The following program uses the PWM module to generate a 78.12kHz digital wave with 50% duty cycle. This digital wave is outputted through the CCP1 pin (pin 17) of the PIC16F887 microcontroller. The measured DC voltage at the output would be about 2.5V.

```

list P=PIC16F887, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F887.INC"
_CONFIG_CONFIG1, _PWRTE_ON & _WDT_OFF & _INTOSC & _BOR_ON & _LVP_OFF & _CP_OFF &
_MCLRE_ON & _IESO_OFF & _FCMEN_OFF
_CONFIG_CONFIG2, _WRT_OFF & _BOR40V
errorlevel -302

ORG      0
GOTO    INIT

INIT    BSF      STATUS,RP0      ;select bank 1
        BCF      INTCON,GIE      ;disable global interrupt
        BCF      INTCON,PEIE
        MOVLW    B'00111111'    ;configure PR2
        MOVWF    PR2
        BCF      STATUS,RP0      ;select bank 0
        MOVWF    CCP1CON
        BSF      STATUS,RP0      ;select bank 1
        CLRF     TRISC          ;configure PORTC as output
        BCF      STATUS,RP0      ;select bank 0
        MOVLW    B'00000100'    ;configure T2CON
        MOVWF    T2CON
        MOVLW    B'00110010'    ;Set the duty cycle to 50%
        MOVWF    CCPR1L

ENDLP   GOTO    ENDLP      ;endless loop

END

```

7.10.4.5 Analog Output Program (PIC16F1937)

The following program uses the PWM module to generate a 78.12kHz digital wave with 50% duty cycle. This digital wave is outputted through the CCP1 pin (pin 17) of the PIC16F1937 microcontroller. The measured DC voltage at the output would be about 2.5V. Note the difference between the PR2 values, which arises due to 32 MHz clock frequency of PIC16F1937, as opposed to 20 MHz of the previous examples.

```
list P=PIC16F1937, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F1937.INC"
    _CONFIG _CONFIG1, _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _MCLRE_ON & _CP_OFF &
    _CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_ON
    _CONFIG _CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _BORV_19 & _LVP_OFF
errorlevel -302

ORG      0
GOTO    INIT

INIT    MOVLB    D'0'      ;select bank 0
        BCF      INTCON, GIE   ;disable global interrupt
        BCF      INTCON, PEIE
        MOVLW    B'01100101'  ;configure PR2
        MOVWF    PR2
        MOVLB    D'5'      ;select bank 5
        MOVLW    B'00111100'
        MOVWF    CCP1CON
        MOVLB    D'1'      ;select bank 1
        CLRF     TRISC     ;configure PORTC as output
        clrf     BSR      ;select bank 0
        MOVLW    B'00000100'  ;configure T2CON
        MOVWF    T2CON
        MOVLB    D'5'      ;select bank 5
        MOVLW    B'00110010'  ;Set the duty cycle to 50%
        MOVWF    CCPR1L

ENDLP   GOTO    ENDLP     ;endless loop

END
```

7.10.4.6 Analog Output Program (PIC18)

The following program uses PWM to generate a 20kHz digital signal with 50% duty cycle. The key differences to be noted between this program and the previous program is that this PIC18 program uses the internal oscillator as its primary clock with PLL enabled for a Fosc of 32MHz. Because the ratio of the oscillator frequency to the output PWM frequency is fairly large, a 10-bit resolution is achieved. The signal is output through the CCP2 pin (RC1) on the PIC18F4620.

```
#include <p18f4620.inc>
list P=18F4620, F=INHX32, C=160, N=80, ST=OFF, MM=OFF, R=DEC

;;;;;Configuration Bits;;;;;;
CONFIG OSC=INTIO67, FCMEN=OFF, IESO=OFF
```

```

CONFIG PWRT = OFF, BOREN = SBORDIS, BORV = 3
CONFIG WDT = OFF, WDTPS = 32768
CONFIG MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF, CCP2MX = PORTC
CONFIG STVREN = ON, LVP = OFF, XINST = OFF
CONFIG DEBUG = OFF
CONFIG CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF
CONFIG CPB = OFF, CPD = OFF
CONFIG WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
CONFIG WRTB = OFF, WRTC = OFF, WRTD = OFF
CONFIG EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF
CONFIG EBTRB = OFF

org      0
goto    INIT

INIT
    movlw  B'01110000' ;Set internal oscillator frequency to 8MHz
    movwf  OSCCON
    bsf    OSCTUNE, 6 ;Enable PLL - oscillator speed = 32MHz
    bcf    INTCON, GIE ;disable global interrupt
    bcf    INTCON, PEIE
    movlw  B'01100011' ;configure PR2
    movwf  PR2
    movlw  B'00110010' ;configure CCPR2L
    movwf  CCPR2L
    movlw  B'00001100' ;configure CCP2CON
    movwf  CCP2CON
    clrf   TRISC      ;configure PORTC as output
    movlw  B'00000101' ;configure T2CON, set prescaler to 4
    movwf  T2CON

ENDLP  bra    ENDLP      ;endless loop

end

```

7.10.4.7 Enhanced PWM Module

The PIC16F887 and PIC18F4620 also incorporate one Enhanced PWM modules, implemented as ECCP. In addition, PIC16F1937 also has three ECCP modules. It supports a broader range of control options by allowing 2 or 4 output pins. The enhanced PWM is controlled by the CCP1CON register for PIC16F887 and PIC16F18, and the three ECCPs are controlled by CCP1CON, CCP2CON and CCP3CON for PIC16F1937. Enhanced PWM differs from standard PWM in that bits 6 and 7 are implemented for enhanced PWM configuration. The two-output mode is ideally suited for controlling a half-bridge circuit, and the four-output mode is for controlling a full-bridge circuit. For more information, refer to the PIC16F887 and PIC18F4620 datasheet. Note that for PIC16F1937, CCP4 and CCP5 do not have Enhanced feature.

7.10.4.7.1 ECCP Control Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6	P1M1:P1M0: Enhanced PWM Output Configuration bits <u>If CCP1M3:CCP1M2 = 00, 01, 10:</u> xx = P1A assigned as capture/compare input/output; P1B, P1C, P1D assigned as port pins <u>If CCP1M3:CCP1M2 = 11:</u> 00 = Single output, P1A modulated; P1B, P1C, P1D assigned as port pins 01 = Full-bridge output forward, P1D modulated; P1A active; P1B, P1C inactive 10 = Half-bridge output, P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins 11 = Full-bridge output reverse, P1B modulated; P1C active; P1A, P1D inactive
bit 5-4	DC1B1:DC1B0: PWM Duty Cycle bit 1 and bit 0 <u>Capture mode:</u> Unused. <u>Compare mode:</u> Unused. <u>PWM mode:</u> These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
bit 3-0	CCP1M3:CCP1M0: Enhanced CCP Mode Select bits 0000 = Capture/Compare/PWM off (resets ECCP module) 0001 = Reserved 0010 = Compare mode, toggle output on match 0011 = Capture mode 0100 = Capture mode, every falling edge 0101 = Capture mode, every rising edge 0110 = Capture mode, every 4th rising edge 0111 = Capture mode, every 16th rising edge 1000 = Compare mode, initialize CCP1 pin low; set output on compare match (set CCP1IF) 1001 = Compare mode, initialize CCP1 pin high; clear output on compare match (set CCP1IF) 1010 = Compare mode, generate software interrupt only; CCP1 pin reverts to I/O state 1011 = Compare mode, trigger special event (ECCP resets TMR1 or TMR3, sets CCP1IF bit) 1100 = PWM mode, P1A, P1C active-high; P1B, P1D active-high 1101 = PWM mode, P1A, P1C active-high; P1B, P1D active-low 1110 = PWM mode, P1A, P1C active-low; P1B, P1D active-high 1111 = PWM mode, P1A, P1C active-low; P1B, P1D active-low

7.11 A/D Module

The analog-to-digital (A/D) converter has 8 analog inputs for PIC16F877, 14 analog inputs for PIC16F887 and PIC16F1937, and 13 analog inputs for PIC18F4620. It allows the conversion of an input analog signal to a corresponding 10-bit digital number. The analog reference voltage is programmable to be the chip's supplied voltage or any input voltage reference.

The A/D module has 4 registers in common:

- **A/D Result High Register (ADRESH):** contains high-bits the A/D conversion result.
- **A/D Result Low Register (ADRESL):** contains low-bits the A/D conversion result.
- **A/D Control Register0 (ADCON0):** controls the operation of the A/D module.
- **A/D Control Register1 (ADCON1):** configures the functions of the port pins (For PIC16F887, this register only configures the output orientation and the reference voltages).

It has 2 extra registers for PIC16F887:

- **Analog Select Register (ANSEL):** configures the analog state of each configurable pin in PORTA, PORTE.
- **Analog Select High Register (ANSELH):** configures the analog state of each configurable pin PORTB.

It has 4 extra registers for PIC16F1937:

- **Analog Select Register for Port A (ANSELA):** configures the analog state of each configurable pin in PORTA.
- **Analog Select Register for Port B (ANSELB):** configures the analog state of each configurable pin in PORTB.
- **Analog Select Register for Port E (ANSELE):** configures the analog state of each configurable pin in PORTE.

It has 1 extra register for PIC18F4620:

- **A/D Control Register2 (ADCON2):** configures the A/D clock source, acquisition time and justification (PIC18 only)

7.11.1 Control Registers

7.11.1.1 ADCON0 Register (PIC16F877)

ADSC1	ADSC0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
bit 7						-	bit 0

bit 7:6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D RC oscillator)

bit 5:3 **CHS2:CHS0:** Analog Channel Select bits
000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

bit 2 **GO/DONE:** A/D Conversion Status bit
When ADON = 1
1 = A/D conversion in progress
(Setting this bit starts the A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is complete)

0 = A/D conversion not in progress

bit 1 **Unimplemented:** Read as '0'

bit 0 **ADON:** A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

7.11.1.2 ADCON0 Register (PIC16F887)

ADSC1	ADSC0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

bit 7:6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz maximum)

bit 5:2 **CHS3:CHS0:** Analog Channel Select bits
0000 = AN0
0001 = AN1
0010 = AN2
0011 = AN3
0100 = AN4
0101 = AN5
0110 = AN6
0111 = AN7
1000 = AN8
1001 = AN9
1010 = AN10
1011 = AN11
1100 = AN12
1101 = AN13
1110 = CVREF
1111 = Fixed Ref (0.6V fixed voltage reference)

bit 1 **GO/DONE:** A/D Conversion Status bit
When ADON = 1
1 = A/D conversion in progress
(Setting this bit starts the A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

bit 0 **ADON:** A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

7.11.1.3 ADCON0 Register (PIC16F1937)

---	CHS4	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

bit 5:2 **CHS4:CHS0:** Analog Channel Select bits

00000 = AN0
 00001 = AN1
 00010 = AN2
 00011 = AN3
 00100 = AN4
 00101 = AN5
 00110 = AN6
 00111 = AN7
 01000 = AN8
 01001 = AN9
 01010 = AN10
 01011 = AN11
 01100 = AN12
 01101 = AN13
 01110 = Reserved. No channel connected.
 •
 •
 •

11100 = Reserved. No channel connected.
 11101 = Temperature Indicator(3)
 11110 = DAC output(1)
 11111 =FVR (Fixed Voltage Reference) Buffer 1 Output(2)

bit 1 **GO/DONE:** A/D Conversion Status bit

When ADON = 1
 1 = A/D conversion in progress
 (Setting this bit starts the A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is complete)
 0 = A/D conversion not in progress

bit 0 **ADON:** A/D On bit

1 = A/D converter module is operating
 0 = A/D converter module is shutdown and consumes no operating current

Note 1: See Section 17.0 “Digital-to-Analog Converter (DAC) Module” of datasheet.

2: See Section 14.0 “Fixed Voltage Reference (FVR)” of datasheet..

3: See Section 16.0 “Temperature Indicator Module” of datasheet.

7.11.1.4 ADCON0 Register (PIC18)

---	---	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

bit 5:2 **CHS3:CHS0:** Analog Channel Select bits

0000 = channel 0, (AN0)
0001 = channel 1, (AN1)
0010 = channel 2, (AN2)
0011 = channel 3, (AN3)
0100 = channel 4, (AN4)
0101 = channel 5, (AN5)
0110 = channel 6, (AN6)
0111 = channel 7, (AN7)
1000 = channel 8, (AN8)
1001 = channel 9, (AN9)
1010 = channel 10, (AN10)
1011 = channel 11, (AN11)
1100 = channel 12, (AN12)

bit 1 **GO/DONE:** A/D Conversion Status bit

When ADON = 1

1 = A/D conversion in progress

(Setting this bit starts the A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is complete)

0 = A/D conversion not in progress

bit 0 **ADON:** A/D On bit

1 = A/D converter module is operating

0 = A/D converter module is shutoff and consumes no operating current

7.11.1.5 ADCON1 Register (PIC16F877)

ADFM	--	--	--	PSFG3	PSFG2	PSFG1	PSFG0
bit 7	bit 0						

bit 7 **ADFM:** A/D Result Format Select bit
 1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
 0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6:4 **Unimplemented:** Read as '0'

bit 3:0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	V_{REF+}	V_{REF-}
0000	A	A	A	A	A	A	A	A	V _{DD}	V _{SS}
0001	A	A	A	A	V _{REF+}	A	A	A	RA3	V _{SS}
0010	D	D	D	A	A	A	A	A	V _{DD}	V _{SS}
0011	D	D	D	A	V _{REF+}	A	A	A	RA3	V _{SS}
0100	D	D	D	D	A	D	A	A	V _{DD}	V _{SS}
0101	D	D	D	D	V _{REF+}	D	A	A	RA3	V _{SS}
011x	D	D	D	D	D	D	D	D	V _{DD}	V _{SS}
1000	A	A	A	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1001	D	D	A	A	A	A	A	A	V _{DD}	V _{SS}
1010	D	D	A	A	V _{REF+}	A	A	A	RA3	V _{SS}
1011	D	D	A	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1100	D	D	D	A	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1101	D	D	D	D	V _{REF+}	V _{REF-}	A	A	RA3	RA2
1110	D	D	D	D	D	D	D	A	V _{DD}	V _{SS}
1111	D	D	D	D	V _{REF+}	V _{REF-}	D	A	RA3	RA2

A = Analog input **D** = Digital I/O

7.11.1.6 ADCON1 Register (PIC16F887)

ADFM	--	VCFG1	VCGF0	--	--	--	--	--
bit 7	bit 0							

bit 7 **ADFM:** A/D Result Format Select bit
 1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
 0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference bit
 1 = V_{REF-} pin
 0 = V_{SS}

bit 4 **VCGF0:** Voltage Reference bit
 1 = V_{REF+} pin
 0 = V_{DD}

bit 3:0 **Unimplemented:** Read as '0'

7.11.1.7 ADCON1 Register (PIC16F1937)

ADFM	ADCS2	ADCS1	ADCS0	--	ADNREF	ADPREF1	ADPREF0
bit 7							bit 0

- bit 7 **ADFM:** A/D Result Format Select bit
 1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
 0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.
- bit 6-4 **ADCS<2:0>:** A/D Conversion Clock Select bits
 000 = Fosc/2
 001 = Fosc/8
 010 = Fosc/32
 011 = FRC (clock supplied from a dedicated RC oscillator)
 100 = Fosc/4
 101 = Fosc/16
 110 = Fosc/64
 111 = FRC (clock supplied from a dedicated RC oscillator)
- bit 3 **Unimplemented:** Read as '0'
- bit 2 **ADNREF:** A/D Negative Voltage Reference Configuration bit
 0 = VREF- is connected to Vss
 1 = VREF- is connected to external VREF- pin(1)
- bit 1-0 **ADPREF<1:0>:** A/D Positive Voltage Reference Configuration bits
 00 = VREF+ is connected to VDD
 01 = Reserved
 10 = VREF+ is connected to external VREF+ pin(1)
 11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module(1)

Note 1: When selecting the FVR or the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See the applicable Electrical Specifications Chapter for details.

7.11.1.8 ADCON1 Register (PIC18)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-q ⁽¹⁾	R/W-q ⁽¹⁾	R/W-q ⁽¹⁾
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7	bit 0						

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
 1 = VREF- (AN2)
 0 = Vss

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
 1 = VREF+ (AN3)
 0 = VDD

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7 ⁽²⁾	AN6 ⁽²⁾	AN5 ⁽²⁾	AN4	AN3	AN2	AN1	AN0
0000 ⁽¹⁾	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111 ⁽¹⁾	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input

D = Digital I/O

7.11.1.9 ANSEL Register (PIC16F887)

ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7	bit 0						

bit 7-0 **ANS<7:0>:** Analog Select bits

Analog select between analog or digital function on pins AN<7:0>, respectively.
 1 = Analog input. Pin is assigned as analog input.
 0 = Digital I/O. Pin is assigned to port or special function.

7.11.1.10 ANSELH Register (PIC16F887)

--	--	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **ANS<13:8>:** Analog Select bits

Analog select between analog or digital function on pins AN<13:8>, respectively.

1 = Analog input. Pin is assigned as analog input.

0 = Digital I/O. Pin is assigned to port or special function.

7.11.1.11 ADCON2 Register (PIC18)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **ADFM:** A/D Result Format Select bit

1 = Right justified

0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits

111 = 20 TAD

110 = 16 TAD

101 = 12 TAD

100 = 8 TAD

011 = 6 TAD

010 = 4 TAD

001 = 2 TAD

000 = 0 TAD⁽¹⁾

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits

111 = FRC (clock derived from A/D RC oscillator)⁽¹⁾

110 = Fosc/64

101 = Fosc/16

100 = Fosc/4

011 = FRC (clock derived from A/D RC oscillator)⁽¹⁾

010 = Fosc/32

001 = Fosc/8

000 = Fosc/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

7.11.2 Operation

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. We will discuss the requirement of the acquisition time in the next session. After this acquisition time has elapsed the A/D conversion can be started. The following steps should be followed for performing an A/D conversion:

1. Configure the A/D module:

- Configure voltage reference (*ADCON1*)
- Configure analog inputs and digital I/O (*ADCON1* for PIC16F877 and PIC18, *ANSEL* and *ANSELH* for PIC16F887, *ANSELA* and *ANSELB* and *ANSELE* for PIC16F1937)
- Select result format (*ADFM* bit – *ADCON1* for PIC16s, *ADCON2* for PIC18)
- Select A/D input channel (*ADCON0* – *CHS2:CHS0* for PIC16F877, *CHS3:CHS0* for PIC16F887 and PIC18, *CHS4:CHS0* for PIC16F1937)
- Select A/D acquisition time (*ADCON2* – *ACQT2:ACQT0*, PIC18 only)
- Select A/D conversion clock (*ADCON0* – *ADCS1:ADCS0* for PIC16F877/887, *ADCON1* – *ADCS2:ADCS0* for PIC16F1937, *ADCON2* – *ADCS2:ADCS0* for PIC18)
- Turn on A/D module (*ADCON0* – *ADON* / bit 0)
- Configure the TRIS register for the analog inputs

2. Configure A/D interrupt (if desired):

- Clear ADIF bit (*PIR1* – bit 6)
- Set ADIE bit (*PIE1* – bit 6)
- Set PEIE bit (*INTCON* – bit 6)
- Set GIE bit (*INTCON* – bit 7)

3. Wait the required acquisition time (if required)

4. Start conversion

- Set the GO/DONE bit (*ADCON0*)

5. Wait for A/D conversion to complete, by either:

- Polling for the GO/DONE bit to be cleared
- OR
- Waiting for the A/D interrupt

6. Read A/D result register pair (ADRESH:ADRESL), clear the ADIF bit, if required.

7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as T_{AD} . A minimum wait of $2T_{AD}$ is required before next acquisition starts.

7.11.3 Timing

7.11.3.1 A/D Conversion Clock

T_{AD} is defined as the A/D conversion time per bit. The entire 10-bit A/D conversion requires a minimum of $12T_{AD}$ to complete. The A/D conversion clock is software selected by *ADCS1:ADCS0* in *ADCON0* register. There are seven possible choices for T_{AD} ,

- $2T_{OSC}$
- $4T_{OSC}$ (PIC18 only)
- $8T_{OSC}$
- $16T_{OSC}$ (PIC18 only)

- $32T_{OSC}$
- $64T_{OSC}$ (PIC18 only)
- Internal A/D module RC oscillator (2-6 us)

where T_{OSC} is the device frequency.

Note: For correct A/D conversion, the A/D conversion clock (T_{AD}) must be at least 1.6us for PIC16 and 0.7us for PIC18.

7.11.3.2 Example

If a 20MHz oscillator is used,

⇒ Clock Period = 0.05us

- (i) $2T_{OSC}$ clock is selected, $T_{AD} = 0.1\text{us} < 1.6\text{us}$, incorrect conversion.
- (ii) $8T_{OSC}$ clock is selected, $T_{AD} = 0.4\text{us} < 1.6\text{us}$, incorrect conversion.
- (iii) $32T_{OSC}$ clock is selected, $T_{AD} = 1.6\text{us}$, correct conversion.
- (iv) Internal A/D module RC oscillator is selected, $T_{AD} = 2\text{-}6\text{us}$, correct conversion.

7.11.3.3 Acquisition Time Requirement

In order to measure the analog signal, the input charge holding capacitor must be allowed to fully charge to the input channel voltage level. The minimum required acquisition time depends on the impedance and the capacitance of the source as well as the internal sampling switch. In the sample code given in this chapter, an acquisition time of 20us is used. For more details of the calculation of the acquisition time requirement, please see the specification of the PIC.

The PIC18F4620 incorporates a programmable acquisition time feature, which allows the A/D converter to wait a preset amount of acquisition time before automatically starting the conversion. There are 7 levels of preset acquisition times ranging between $2 T_{AD}$ and $20 T_{AD}$. Using this feature, the user may not need to manually delay by the required amount before setting the GO/DONE bit to begin the conversion.

7.11.4 Simple A/D Converter (PIC16F877)

The following code presents an A/D converter that converts the analog input at AN0 and displays the high 8-bit of the 10-bit digital number on the LEDs at PORTB.

For pin connections, RA0 should be connected to an input analog signal which will be the source of the A/D converter. RB8:RB0 connects to 8 LEDs so that the result of the A/D conversion can be displayed through the LEDs.

Note: In order to use the RB5 as an I/O pin, low voltage programming cannot be used for this A/D converter.

```

list P=PIC16F877, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F877.INC"
__config (_CP_OFF & _PWRTE_ON & _XT_OSC & _WDT_OFF & _BODEN_OFF )
errorlevel -302

TIMCNT EQU 020H

;*****
; Initialize Part
;
; ADCON1 - Left justified result
; - AN0/RA0 as the only analog input
;
; TRISB - PORTB as output
;*****
;
```

```

ORG      0
GOTO    INIT

INIT    BSF      STATUS,RP0    ;select bank 1
        BCF      INTCON,GIE    ;disable global interrupt
        MOVLW   B'00001110'    ;configure ADCON1
        MOVWF   ADCON1
        CLRF    TRISB    ;configure PORTB as output
        BCF      STATUS,RP0    ;select bank 0
        goto    ADSTART

;*****MAIN PROGRAM*****
;*****MAIN PROGRAM*****
ADSTART CALL    AD_CONV    ;call the A2D subroutine
        MOVWF   PORTB    ;display the high 8-bit result to the LEDs

ENDLP   GOTO    ENDLP    ;endless loop

;*****AD CONVERT ROUTINE*****
;*****AD CONVERT ROUTINE*****
AD_CONV MOVLW   B'10000001'  ;configure ADCON0
        MOVWF   ADCON0
        CALL    TIM20    ;wait for required acquisition time
        BSF    ADCON0,GO  ;start the conversion

WAIT    BTFSC   ADCON0,GO  ;wait until the conversion is completed
        GOTO    WAIT    ;poll the GO bit in ADCON0
        MOVF    ADRESH,W  ;move the high 8-bit to W
        RETURN

;*****TIME DELAY ROUTINE FOR 20us*****
; - delay of 400 cycles
; - 400*0.05us = 20us
;*****TIME DELAY ROUTINE FOR 20us*****
TIM20    MOVLW   084H    ;1 cycle
        MOVWF   TIMCNT   ;1 cycle

TIMLP   DECFSZ  TIMCNT,F  ;(3*132)-1 = 395 cycles
        GOTO    TIMLP
        NOP
        RETURN   ;2 cycles

END

```

7.11.5 Simple A/D Converter (PIC16F887)

```

list P=PIC16F877, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F877.INC"
  _CONFIG _CONFIG1, _PWRTE_ON & _WDT_OFF & _INTOSC & _BOR_ON & _LVP_OFF & _CP_OFF &
  _MCLRE_ON & _IESO_OFF & _FCMEN_OFF
  __CONFIG __CONFIG2, _WRT_OFF & _BOR40V

TIMCNT  EQU    020H

```

```

;*****
; Initialize Part
;
; ADCON1 - Left justified result
; TRISB - PORTB as output
;*****
ORG      0
GOTO    INIT

INIT    BSF      STATUS,RP0      ;select bank 1
        BCF      INTCON,GIE      ;disable global interrupt
        Clrf    ADCON1      ;configure ADCON1
        BSF      STATUS,RP1      ;select bank 3
        Movlw   b'00000001'      ;configure ANSEL
        Movwf   ANSEL      ;set AN0/RA0 to analog
        Clrf    ANSELH      ;configure ANSELH to make all rest digital
        BCF      SRARUS,RP1      ;select bank 1
        CLRF    TRISB      ;configure PORTB as output
        BCF      STATUS,RP0      ;select bank 0
        goto    ADSTART

;*****
; MAIN PROGRAM
;*****
ADSTART CALL    AD_CONV      ;call the A2D subroutine
        MOVWF  PORTB      ;display the high 8-bit result to the LEDs

ENDLP   GOTO    ENDLP      ;endless loop

;*****
; AD CONVERT ROUTINE
;*****
AD_CONV MOVLW   B'10000001'      ;configure ADCON0
        MOVWF   ADCON0
        CALL    TIM20      ;wait for required acquisition time
        BSF    ADCON0,GO      ;start the conversion

WAIT    BTFSC  ADCON0,GO      ;wait until the conversion is completed
        GOTO    WAIT      ;poll the GO bit in ADCON0
        MOVF    ADRESH,W      ;move the high 8-bit to W
        RETURN

;*****
; TIME DELAY ROUTINE FOR 20us
;
; - delay of 400 cycles
; - 400*0.05us = 20us
;*****
TIM20   MOVLW   084H      ;1 cycle
        MOVWF   TIMCNT      ;1 cycle

TIMLP   DECFSZ  TIMCNT,F      ;(3*132)-1 = 395 cycles
        GOTO    TIMLP
        NOP
        RETURN      ;2 cycles

END

```

7.11.6 Simple A/D Converter (PIC16F1937)

```
list P=PIC16F1937, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F1937.INC"

__CONFIG _CONFIG1, _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _MCLRE_ON & _CP_OFF &
__CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_ON
__CONFIG _CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _BORV_19 & _LVP_OFF

TIMCNT EQU 020H

;*****
; Initialize Part
;
; ADCON1 - Left justified result
; TRISB - PORTB as output
;*****
ORG 0
GOTO INIT

INIT Movlb d'1'      ;select bank 1
      BCF INTCON, GIE ;disable global interrupt
      Clrf ADCON1    ;configure ADCON1
      Movlb d'3'      ;select bank 3
      Movlw b'00000001' ;configure ANSEL
      Movwf ANSELA   ;set AN0/RA0 to analog
      Clrf ANSELB    ;configure ANSELB to make PORTB all digital
      Clrf ANSELD    ;configure ANSELB to make PORTD all digital
      Clrf ANSELE    ;configure ANSELB to make PORTE all digital
      Movlb d'1'      ;select bank 1
      Clrf TRISB    ;configure PORTB as output
      Clrf BSR      ;select bank 0
      goto ADSTART

;*****
; MAIN PROGRAM
;*****
ADSTART CALL AD_CONV ;call the A2D subroutine
      MOVWF PORTB ;display the high 8-bit result to the LEDs

ENDLP GOTO ENDLP ;endless loop

;*****
; AD CONVERT ROUTINE
;*****
AD_CONV Movlb d'1'      ;select bank 1
      MOVLW B'10000001' ;configure ADCON0
      MOVWF ADCON0
      Clrf BSR      ;select bank 0
      CALL TIM20    ;wait for required acquisition time
      Movlb d'1'      ;select bank 1
      BSF ADCON0, GO ;start the conversion

WAIT BTFSC ADCON0, GO ;wait until the conversion is completed
      GOTO WAIT ;poll the GO bit in ADCON0
      MOVF ADRESH, W ;move the high 8-bit to W
      RETURN

;*****
; TIME DELAY ROUTINE FOR 20us
;
```

```

; - delay of 400 cycles
; - 400*0.03125us = 12.5us
;*****
TIM20  MOVLW    035H      ;1 cycle
      MOVWF    TIMCNT    ;1 cycle

TIMLP  DECFSZ   TIMCNT, F   ;160 cycles
      GOTO    TIMLP
      NOP
      RETURN   ;1 cycle
                  ;2 cycles

END

```

7.11.7 Simple A/D Converter (PIC18)

```

#include <p18f4620.inc>
list P=18F4620, F=INHX32, C=160, N=80, ST=OFF, MM=OFF, R=DEC

;;;;;Configuration Bits;;;;;;;;
CONFIG OSC=INTIO67, FCMEN=OFF, IESO=OFF
CONFIG PWRT = OFF, BOREN = SBORDIS, BORV = 3
CONFIG WDT = OFF, WDTPS = 32768
CONFIG MCLRE = ON, LPT1OSC = OFF, PBADEN = OFF, CCP2MX = PORTC
CONFIG STVREN = ON, LVP = OFF, XINST = OFF
CONFIG DEBUG = OFF
CONFIG CP0 = OFF, CP1 = OFF, CP2 = OFF, CP3 = OFF
CONFIG CPB = OFF, CPD = OFF
CONFIG WRT0 = OFF, WRT1 = OFF, WRT2 = OFF, WRT3 = OFF
CONFIG WRTB = OFF, WRTC = OFF, WRTD = OFF
CONFIG EBTR0 = OFF, EBTR1 = OFF, EBTR2 = OFF, EBTR3 = OFF
CONFIG EBTRB = OFF

;*****
; Initialize Part
;
; ADCON1 - AN0/RA0 as the only analog input
; ADCON2 - Acquisition time = 8 TAD
;           - Conversion clock = Fosc/64
; TRISB - PORTB as output
;*****
org      0
movlw   B'01110000' ;set internal oscillator frequency
movwf   OSCCON      ;to 8 MHz
bsf     OSCTUNE, 6  ;turn on PLL to enable 32MHz clock frequency
bra    INIT

INIT   bcf    INTCON, GIE  ;disable global interrupt
      movlw   B'00001110' ;configure ADCON1
      movwf   ADCON1
      movlw   B'00100110' ;configure ADCON2
      movwf   ADCON2
      clrf    TRISB       ;configure PORTB as output
      bra    ADSTART

;*****
; MAIN PROGRAM
;*****
ADSTART rcall   AD_CONV    ;call the A2D subroutine
      movwf   PORTB      ;display the high 8-bit result to the LEDs

```

```
ENDLP    bra      ENDLP      ;endless loop

;*****  
; AD CONVERT ROUTINE  
;*****  
AD_CONV  movlw    B'00000001' ;configure ADCON0  
         movwf    ADCON0  
         bsf     ADCON0,1  ;start the conversion

WAIT     btfsc   ADCON0,1  ;wait until the conversion is completed  
         bra     WAIT      ;poll the GO bit in ADCON0  
         movf    ADRESH,W ;move the high 8-bit to W  
         return  
         end
```

7.12 USART/EUSART Module

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules available on the PIC microcontroller. USART is a means of communication with peripheral devices such as Serial-Peripheral-Interface (SPI) terminals, or personal computers, when it is configured in the full-duplex Asynchronous mode. It can also communicate with other peripheral devices such as A/D or D/A devices, external serial EEPROMs, etc., when it is configured in the half-duplex Synchronous mode. On the PIC16F877, registers **TXSTA** and **RCSTA** control and perform tasks of transmission and reception, respectively. Bit **SPEN** (**RCSTA** $<7>$) and bits **TRISC** $<7:6>$ must be set in order to configure pins RC6/TX/CK (receive) and RC7/RX/DT for the USART. An 8-bit register, **SPBRG**, is dedicated for generating the baud rate for serial communication.

The Asynchronous mode can be selected by clearing bit **SYNC** (**TXSTA** $<4>$). In this mode, one START bit, eight data bits (can be nine), and one **STOP** bit are transferred from and to the two pins RC6 and RC7, respectively. Transmission and reception are functionally independent, but use the same format and baud rate. The baud rate values are set in the **SPBRG** register. An important bit is **BRGH** (**TXSTA** $<2>$) that selects the high baud rate. It is recommended to set this bit for rates equal to or higher than 9600.

The Synchronous mode is configured by setting bit **SYNC** (**TXSTA** $<4>$). This mode allows half-duplex communication, i.e., the data transmission and reception do not occur at the same time; when transmitting data the reception is inhibited and vice versa. The bit **SPEN** (**RCSTA** $<7>$) is set in order to configure the RC6 pin to clock (CK) and RC7 pin to data (DT) line.

The PIC16F887, PIC16F1937, and PIC18F4620 implement the Enhanced Universal Synchronous Asynchronous Receiver Transmitter module (EUSART). This module provides additional features to the USART module used on the PIC16F877 such as automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. It is ideally suited for Local Interconnect Network Bus (LIN Bus) systems. To enable USART, the **SPEN** bit (**RCSTA** $<7>$) and **TRISC** $<7:6>$ must be set. The baud rate control register **BAUDCON** controls enhanced features such as auto baud rate detect and 16-bit baud rate generator enable. The EUSART also features a 16-bit baud rate generator, **SPBRGH:SPBRG**, which will enable a wider range of baud rates for communication. Other features of EUSART are described in these PICs' Datasheet.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous – Master (half duplex)
- Synchronous – Slave (half duplex)

The USART module involves the configuration of the following registers:

- Transmit Status and Control Register (**TXSTA**)
- Receive Status and Control Register (**RCSTA**)
- Baud Rate Control Register (**BAUDCTL** for PIC16F887 only, **BAUDCON** for PIC18 only)
- Baud Rate Generator Register (**SPBRG**)

Data will be transmitted and received through:

- USART Transmit Register (**TXREG**)
- USART Receive Register (**RCREG**)

7.12.1 Control Registers

7.12.1.1 TXSTA Register

CSRC	TX9	TXEN	SYNC	SEND B	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7	CSRC: Clock Source Select bit <u>Asynchronous mode:</u> Don't care <u>Synchronous mode:</u> 1 = Master mode (clock generated internally from BRG) 0 = Slave mode (clock from external source)
bit 6	TX9: 9-bit Transmit Enable bit 1 = Selects 9-bit transmission 0 = Selects 8-bit transmission
bit 5	TXEN: Transmit Enable bit 1 = Transmit enabled 0 = Transmit disabled
bit 4	SYNC: USART Mode Select bit 1 = Synchronous mode 0 = Asynchronous mode
bit 3	SEND B: Send Break Character bit (PIC18 only) <u>Asynchronous mode:</u> 1 = Send Sync Break on next transmission (cleared by hardware upon completion) 0 = Sync Break transmission completed <u>Synchronous mode:</u> Don't care.
bit 2	BRGH: High Baud Rate Select bit <u>Asynchronous mode:</u> 1 = High speed 0 = Low speed <u>Synchronous mode:</u> Unused in this mode
bit 1	TRMT: Transmit Shift Register Status bit 1 = TSR empty 0 = TSR full
bit 0	TX9D: 9th bit of Transmit Data, can be parity bit

7.12.1.2 RCSTA Register

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D bit 0
------	-----	------	------	-------	------	------	---------------

bit 7	SPEN: Serial Port Enable bit 1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins) 0 = Serial port disabled						
bit 6	RX9: 9-bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception						
bit 5	SREN: Single Receive Enable bit <u>Asynchronous mode:</u> Don't care <u>Synchronous mode - master:</u> 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. <u>Synchronous mode - slave:</u> Don't care						
bit 4	CREN: Continuous Receive Enable bit <u>Asynchronous mode:</u> 1 = Enables continuous receive 0 = Disables continuous receive <u>Synchronous mode:</u> 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN) 0 = Disables continuous receive						
bit 3	ADDEN: Address Detect Enable bit <u>Asynchronous mode 9-bit (RX9 = 1):</u> 1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit						
bit 2	FERR: Framing Error bit 1 = Framing error (can be updated by reading RCREG register and receive next valid byte) 0 = No framing error						
bit 1	OERR: Overrun Error bit 1 = Overrun error (can be cleared by clearing bit CREN) 0 = No overrun error						
bit 0	RX9D: 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)						

7.12.1.3 BAUDCON Register (PIC16F887 and PIC16F1937)

R-0	R-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7	bit 0						

- bit 7 **SPEN:** Serial Port Enable bit
 1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
 0 = Serial port disabled (held in Reset)
- bit 6 **RX9:** 9-bit Receive Enable bit
 1 = Selects 9-bit reception
 0 = Selects 8-bit reception
- bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
 Don't care
Synchronous mode – Master:
 1 = Enables single receive
 0 = Disables single receive
 This bit is cleared after reception is complete.
Synchronous mode – Slave
 Don't care
- bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
 1 = Enables receiver
 0 = Disables receiver
Synchronous mode:
 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
 0 = Disables continuous receive
- bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
 1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
 Don't care
- bit 2 **FERR:** Framing Error bit
 1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
 0 = No framing error
- bit 1 **OERR:** Overrun Error bit
 1 = Overrun error (can be cleared by clearing bit CREN)
 0 = No overrun error
- bit 0 **RX9D:** Ninth bit of Received Data
 This can be address/data bit or a parity bit and must be calculated by user firmware.

7.12.1.4 BAUDCON Register (PIC18 Only)

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

- bit 7 **ABDOVF:** Auto-Baud Acquisition Rollover Status bit
 1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
 0 = No BRG rollover has occurred
- bit 6 **RCIDL:** Receive Operation Idle Status bit
 1 = Receive operation is Idle
 0 = Receive operation is active
- bit 5 **RXDTP:** Received Data Polarity Select bit (Asynchronous mode only)
Asynchronous mode:
 1 = RX data is inverted
 0 = RX data received is not inverted
- bit 4 **TXCKP:** Clock and Data Polarity Select bit
Asynchronous mode:
 1 = Idle state for transmit (TX) is a low level
 0 = Idle state for transmit (TX) is a high level
Synchronous mode:
 1 = Idle state for clock (CK) is a high level
 0 = Idle state for clock (CK) is a low level
- bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit
 1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
 0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored
- bit 2 **Unimplemented:** Read as '0'
- bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
 1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
 0 = RX pin not monitored or rising edge detected
Synchronous mode:
 Unused in this mode.
- bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
 1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
 0 = Baud rate measurement disabled or completed
Synchronous mode:
 Unused in this mode.

7.12.2 Baud Rate Generator

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator in the PIC16F877, an 8- or 16-bit baud rate generator in the PIC16F887, PIC16F1937, and PIC18. The SPBRG register controls the period of a free running 8-bit timer. In the other 3 PICs, the SPBRGH:SPBRG registers control the period of the free running 16-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) and BRG16 (BAUDCTL for PIC16F887 and PIC16F1937, BAUDCON<3> for PIC18 only) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Below is the formula for computing the baud rate.

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	$F_{osc}/[64 (n + 1)]$
0	0	1	8-bit/Asynchronous	$F_{osc}/[16 (n + 1)]$
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	$F_{osc}/[4 (n + 1)]$
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

In the PIC16F877, PIC16F1937, and PIC18, using the BRG in 16-bit mode can reduce baud rate errors and also provide low baud rates with a high oscillator rate. For possible baud rate values and errors, refer to the PIC datasheet.

7.12.3 Setup

The following steps should be taken when configuring the USART module operation for *Asynchronous Transmission*:

1. Set the baud rate:

- Initialize SPBRG or SPBRGH:SPBRG for the appropriate baud rate
- Set/clear the BRGH bit and/or the BRG16 bit to achieve the desired baud rate

2. Enable asynchronous serial port:

- Clear bit SYNC
- Set bit SPEN

3. If interrupts are desired,

- Set bit TXIE
- Set GIE and PEIE in the INTCON register

4. If 9-bit transmission is desired,

- Set bit TX9

5. Enable transmission

- Set bit TXEN which will also set bit TXIF

6. If 9-bit transmission is selected,

- Load the 9th data bit in TX9D

7. Load data:

- Write the 8-bit data to TXREG (starts transmission)

The following steps should be taken when configuring the USART module operation for *Asynchronous Reception*:

1. Set the baud rate:

- Initialize SPBRG or SPBRGH:SPBRG for the appropriate baud rate
- Set/clear the BRGH bit and/or the BRG16 bit to achieve the desired baud rate

2. Enable asynchronous serial port:

- Clear bit SYNC
- Set bit SPEN

3. If interrupts are desired,

- Set bit RCIE
- Set GIE and PEIE in the INTCON register

4. If 9-bit reception is desired,

- Set bit RX9

5. Enable reception

- Set bit CREN

6. When reception is completed,

- Flag bit RCIF will be set
- Interrupt will be generated if RCIE is set

7. If 9-bit transmission is selected,

- Read the 9th data bit from RX9D

8. Read the 8-bit data,

- From RCREG

9. If any error occurred,

- Clear the error by clearing bit CREN

7.12.4 Hardware

The built-in USART module makes the PIC very easy to communicate with the PC's serial port. However, the PIC microcontroller's I/O is 0V and +5V. We need +/-10V to meet the RS232 serial port standard. The easiest way to get these values is to use MAX232. MAX232 acts as a buffer between the microcontroller and the PC. For more details, please check the specification of MAX232.

7.12.4.1 Sample Asynchronous TX Program

The following program uses the USART module to send ASCII characters to the PC serial port. Config your terminal to 9600K, no parity, 1 stop bit. The received characters will be shown on the terminal. The only difference for PIC18 is that there is no bank switching.

```
list P=PIC16F877, F=INHX8M, C=160, N=80, ST=OFF, MM=OFF, R=DEC
include "P16F877.INC"
_config (_CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON & _HS_OSC &
         _WRT_ENABLE_ON & _LVP_OFF & _DEBUG_OFF & _CPD_OFF )
errorlevel -302

TIMCNT EQU 020H
LPCNT EQU 021H
TDATA EQU 022H

ORG 0
GOTO INIT

;-----
; Initialization
; - set baud rate to be 9600K
; - port B as output
;-----

INIT BSF STATUS,RP0 ;select bank 1
CLRF TRISB ;configure PORTB as output
BCF INTCON,GIE ;disable global interrupt
BCF INTCON,PEIE

movlw B'10000000'
movwf TRISC ;set RC7=USART RC, RC6=USART TX
```

```

MOVlw      d'15'      ;Baud rate 9600, assuming 10MHz oscillator
MOVwf      SPBRG
clr f     TXSTA      ;8 bits data, no parity, 1 stop

BCF       STATUS, RP0   ;select bank 0
CLRF      RCSTA
BSF       RCSTA, SPEN  ;enable single receive
BSF       RCSTA, CREN  ;continuous

BSF       STATUS, RP0   ;select bank 1
BSF       TXSTA, TXEN  ;enable tx

BCF       STATUS, RP0   ;select bank 0
MOVlw      H'01'
MOVwf      TDATA      ;Store the value 1 in TDATA

;-----
; Main Program
;-----

MAIN      CALL      S_WAIT
          CALL      S_WAIT
          CALL      S_WAIT
          CALL      S_WAIT
          CALL      S_WAIT
          MOVf      TDATA, W  ;Load TDATA into W
          MOVwf     TXREG    ;Send it over RS232
          MOVwf     PORTB    ;Display it on PORTB
          INCF      TDATA, F  ;Increment TDATA

ENDLP     goto     MAIN      ;Repeat

;-----
; Delay Subroutine
;-----

S_WAIT    MOVlw      081H
          MOVwf      LPCNT

TDEL      MOVlw      0FFH
          MOVwf      TIMCNT

TIMLP     DECFSZ   TIMCNT, F
          GOTO      TIMLP
          DECFSZ   LPCNT, F
          GOTO      TDEL
          RETURN

END

```

7.13 I²C Bus

7.13.1 Overview

The I²C is a widely used serial bus specification that allows communication between one or more master devices with one or more slave devices. The master is the device that selects the receiver by transmitting the address of the slave device and initiates both read and write transactions between itself and the slave. The I²C implementation on the DevBugger board includes one master PIC, and three slave devices, the RTC chip, the PIC18F2550, and the I²C female header pins at the top right corner of the board which can connect to an external I²C slave device.

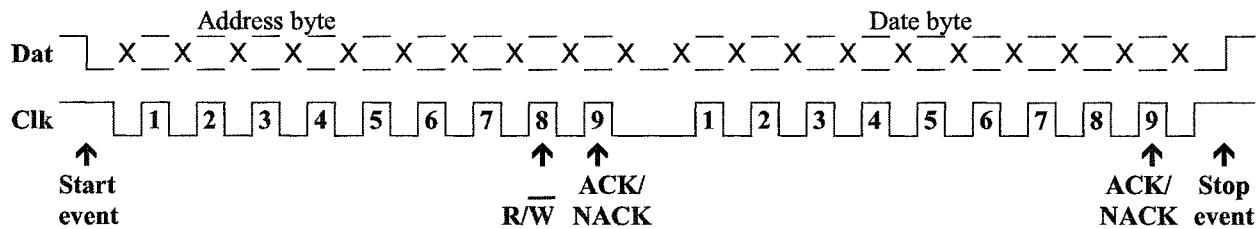
7.13.2 Theory

7.13.2.1 Communication – Hardware

As with other serial buses, the communication is done using a clock line and a data line, both pulled up to +5V and both connected to each master and slave device simultaneously. Thus, both lines are high in the idle state until a master begins a transaction, at which point it controls the clock line and the device that is transmitting data – whether master or slave – controls the data line.

7.13.2.2 Communication – Protocol

The I²C allows transmission of data in bytes, or groups of 8 bits in a defined sequence. Transactions can be reads or writes, but all begin with the master transmitting the address of the target slave, after which one or more bytes are sent or received. The diagram below explains the protocol in detail.



All transactions begin with the master initiating a start event – data goes low with the clock high – after which the 7-bit address of the desired slave is transmitted on the bus. Each slave device ignores this transmission unless the address transmitted matches the address of the slave, in which case it pulls the clock low on the 9th bit to acknowledge the master. The 8th bit transmitted signifies a write if low and a read if high. Based on this bit, the slave takes control of the data line to send data or receives another byte, followed by another acknowledge.

7.13.3 Master Device

7.13.3.1 Overview

All four of PIC16F877, PIC16F887, PIC16F1937, and PIC18F4620 have the Master Synchronous Serial Port implemented as a serial interface in the chip's hardware. This means that the hardware controls the lowest-level operations such as controlling the clock and sending a byte bit by bit. The significance of this is twofold – firstly, the programming is greatly simplified and secondly, the user's code running on the PIC can perform tasks in parallel with the hardware running its low-level operations, if so desired.

7.13.3.2 Layered Framework

The logic and algorithms in `i2c_common.asm` that operate the I²C have been simplified and organized into the 3-layer hierarchical structure shown in Figure 7.13-1. The lowest layer contains the hardware-level macros that control the I²C bus directly, through instructions such as start transaction, write, and stop transaction. The second layer, the process-level subroutines, combines and calls macros from the hardware level to execute complete send or receive transactions. Finally, the user level is where the send and receive subroutines are called. The process and hardware layers are contained in `"i2c_common.asm"`.

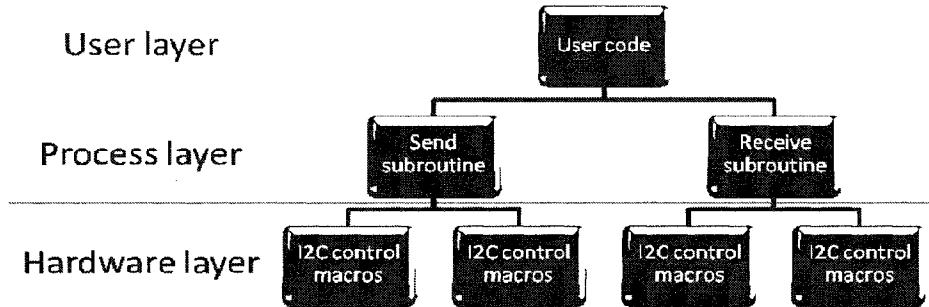


Figure 7.133-1

7.13.3.2.1 Hardware Layer

The I²C serial port on PIC microcontrollers is built into the hardware of the chip, allowing I²C communication to be run in parallel with the PIC's main code. The serial port is controlled mainly by five registers – SSPCON1, SSPCON2, SSPSTAT, SSPBUF, and SSPADD. The macros implemented on the hardware layer of the I²C communication framework use these registers to execute their events – start, stop, repeated start, read, write, acknowledge, no acknowledge, check acknowledge, and setup. Source code for this hardware layer has been developed to simplify I²C programming. It is designed in such a way that all of the lower-level MSSP control is centralized and handled by this code, leaving only the task of calling an I²C setup subroutine, a send subroutine, and a receive subroutine to the user when they need to use the I²C bus. All of this code is contained in a single source file which can be included in one's project in the MPLAB IDE – it is called `"i2c_common.asm"`.

7.13.3.2.2 Process Layer

The process-level subroutine layer has two elements: the send transaction and receive transaction subroutines. Send and receive on the process layer differ from write and read on the hardware layer because the former two cover the entire sending and receiving transactions, respectively, including the event initiation and termination and acknowledge checking. The write and read macros on the hardware layer are each only one event in the series of events called by the send and receive subroutines. Below is the full description of send and receive.

Macro	Send	Receive
1	Start event	Start event
2	Write slave address event	Write slave address event
3	Check acknowledge from slave	Check acknowledge from slave
4	Write data event	Read data event
5	Check acknowledge from slave	Send acknowledge to slave
6	Stop event	Stop event

7.13.3.2.3 User Layer

The user layer makes use of the `i2c_common_setup` subroutine, as well as the send and receive subroutines on the process

layer. The implementation of the user layer depends on the application of the I²C communication, as shown in table below.

Application	User layer
RTC chip communication	Macros are available in <i>rtc_macros.inc</i> that act as wrappers for the process layer.
PIC-PIC parallel processing	There are no pre-existing macros. The user must access the process layer directly.
PIC18 I/O pin usage	Macros are available in <i>p2p_macros.inc</i> that act as wrappers for the process layer.
External slave device	There are no pre-existing macros. The user must access the process layer directly.

7.13.3.3 PIC-RTC Programming

The DS1307RTC is an IC which can keep time not unlike how personal computers keep time even when they are powered off. The DS1307 can run on backup power using a 3V lithium disk battery. The DS1307 acts as a slave device and uses the I²C protocol to transmit data to master devices such as a microcontroller or processor. Aside from the timekeeping function of the RTC, the chip has other capabilities such as programmable square-wave output (not available on the DevBugger) as well as 56-byte non-volatile RAM for general data storage.

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00H	CH		10 Seconds			Seconds			Seconds	00-59
01H	0		10 Minutes			Minutes			Minutes	00-59
02H	0	12	10 Hour	10 Hour		Hours			Hours	1-12 +AM/PM 00-23
03H	0	0	0	0	0	DAY			Day	01-07
04H	0	0	10 Date			Date			Date	01-31
05H	0	0	0	10 Month		Month			Month	01-12
06H		10 Year				Year			Year	00-99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H-FFH

The table above shows the memory map for the DS1307 RTC. Register 00H to 06H are used for timekeeping, register 07H is the configuration register for the square-wave generator, and register 08H to 3FH are general data registers. The binary number in the timekeeping registers does not necessarily represent the value of time elapsed. Instead, the 4 least significant bits (LSB) represent the value of the ones digit of an ASCII (BCD) number while bits 4 to 6 represent the tens digit. Therefore, these binary numbers must be converted to their corresponding ASCII number. For example, if the binary number 0101 1001 is in register 01H, the actual value for the time elapsed in minutes is 59 and not 89 (note that bit 7, the most significant bit, is not used for timekeeping and will always be read back as zero). When reading from register 02H, it is important to keep in mind which mode the RTC is in. Bit 5 represents different meanings depending on whether or not the RTC is in 12 hour format or 24 hour format. When reading from the hours register, it is strongly advised to mask out unimportant bits to avoid reading the wrong value of time.

In order to read or write to these registers, the master device (PIC16 or PIC18) must follow the I²C protocol. The DS1307 RTC has a 7-bit slave address of 110 1000 and the PIC must broadcast this address to select the RTC on the bus. Once it has done that, the PIC master device will then send a byte of data which holds the address of the register it wants to write to or read from. If it is a write operation, the PIC master will send one more byte which holds the data it wants to write to the specified register. If it is a read operation, the RTC will send a byte of data holding the contents of the specified register to the PIC. Write operations are used to set the time for timekeeping and read operations are used to report the time.

There are two files that are available to students which handle all I²C communication with the RTC. These are *i2c_commons.asm* and *rtc_macros.inc*. The first file deals with all the hardware and I²C communication algorithms while the second file contains useful macros to control the RTC using the PIC as a master device. These macros are:

Macro	Parameters	Description
rtc_resetAll	None	Sets all timekeeping registers to zero
rtc_set	Address of register to be written, data to be written	Writes data to the specified register
rtc_read	Address of register to be read	Returns the data in the specified register into memory location 0x75 as a binary number and 0x77 and 0x78 as a two digit ASCII number.

7.13.3.4 PIC16-PIC18 Extending I/O Set Programming

Another very useful application of the I²C is the use of PIC18F2550's I/O pins. Using the I²C bus, the master PIC can send read or write instructions to the slave PIC to control its I/O pins as if they were its own with the only disadvantage of speed. This increases the master PIC's pin count by 7, by enabling access to the slave PIC through the I²C. For this purpose, an include file containing macros that resemble regular PIC I/O control operations has been developed – p2p_macros.inc – and must be included along with i2c_common.asm in the user's project to be able to use the PIC18F2550 I/O pin-related macros. The implementation of these macros utilizes a pre-defined instruction set, shown below.

Instruction	First Transaction						Second Transaction (if app.)					
	Type	Addr.	Data				Type	Addr.	Data			
1) Write to PIC18 pins	Write	0x10	0	0	P	P	D	D	D	V		
2) Write to PIC18 TRIS bits	Write	0x10	0	1	P	P	D	D	D	V		
3) Read PIC18 pins	Write	0x10	1	0	P	P	D	D	D	0	Read	0x11
4) Write to PIC18 ports	Write	0x10	0	0	P	P	0	0	0	V		
5) Write to PIC18 TRIS	Write	0x10	0	1	P	P	0	0	0	V		
6) Read PIC18 ports	Write	0x10	1	1	P	P	0	0	0	0	Read	0x11

Legend

P – Port

D – Identifier

V – Value

Below is the list of macros available in the macros include file. The function and usage of these macros do not require explanation as they behave almost identically to their counterparts in the PIC instruction set, except there are no input parameters.

Write to PIC18 pins/ports	Write to PIC18 TRIS	Read from PIC18 pins/ports
<ul style="list-style-type: none"> • bcf_PORTA_4 • bcf_PORTB_3 • bcf_PORTB_4 • bcf_PORTB_6 • bcf_PORTB_7 • bcf_PORTC_2 • bcf_PORTC_6 • bsf_PORTA_4 • bsf_PORTB_3 • bsf_PORTB_4 • bsf_PORTB_6 • bsf_PORTB_7 • bsf_PORTC_2 • bsf_PORTC_6 • clrf_PORTA • clrf_PORTB • clrf_PORTC • setf_PORTA • setf_PORTB • setf_PORTC 	<ul style="list-style-type: none"> • bcf_TRISA_4 • bcf_TRISB_3 • bcf_TRISB_4 • bcf_TRISB_6 • bcf_TRISB_7 • bcf_TRISC_2 • bcf_TRISC_6 • bsf_TRISA_4 • bsf_TRISB_3 • bsf_TRISB_4 • bsf_TRISB_6 • bsf_TRISB_7 • bsf_TRISC_2 • bsf_TRISC_6 • clrf_TRISA • clrf_TRISB • clrf_TRISC • setf_TRISA • setf_TRISB • setf_TRISC 	<ul style="list-style-type: none"> • btfsc_PORTA_4 • btfsc_PORTB_3 • btfsc_PORTB_4 • btfsc_PORTB_6 • btfsc_PORTB_7 • btfsc_PORTC_2 • btfsc_PORTC_6 • btfss_PORTA_4 • btfss_PORTB_3 • btfss_PORTB_4 • btfss_PORTB_6 • btfss_PORTB_7 • btfss_PORTC_2 • btfss_PORTC_6 • movf_PORTA_W • movf_PORTB_W • movf_PORTC_W

7.13.3.5 PIC16-PIC18 Communication for Register Watching

This application of the I2C can be thought of as an example of how one can use PIC to PIC communication. It is a finished product which can be used without any actual programming on the user's part. The design is simple:

- 1) PIC16: The user calls one of a number of macros to update the contents of a port or register during the running of their PIC16 code.
- 2) PIC18: The data is sent to the PIC18 through I2C, which is sent to the USB.
- 3) PC: The application (available through the website) displays this information.

There are four easy steps to set up this functionality.

1. Add *i2c_common.inc* and *Register_Watcher_macros.inc* to the PIC16 project.
2. Add `#include <Register_Watcher_macros.inc>` in the code.
3. Add *i2c_common_setup* in the code.
4. Call the macros as needed.
5. Make sure the programmer PIC18 has the firmware with register watching.

Below is a table listing the macros contained in *p2p_macros.inc* which must be called by the user's code.

Macro	Description
watch_PORTA	Sends the value of PORTA to the PORTA field in the application
watch_PORTB	Sends the value of PORTB to the PORTB field in the application
watch_PORTC	Sends the value of PORTC to the PORTC field in the application
watch_PORTD	Sends the value of PORTD to the PORTD field in the application
watch_PORTE	Sends the value of PORTE to the PORTE field in the application
watch_TRISA	Sends the value of TRISA to the TRISA field in the application
watch_TRISB	Sends the value of TRISB to the TRISB field in the application
watch_TRISC	Sends the value of TRISC to the TRISC field in the application
watch_TRISD	Sends the value of TRISD to the TRISD field in the application
watch_TRISE	Sends the value of TRISE to the TRISE field in the application
watch_register_0	These macros have an input parameter representing the address of the file register the user wants to monitor. This address is displayed along with the value of the register on the PC application. One can monitor up to 8 registers (registers 0 to 7).
watch_register_1	Example: <code>watch_register_0 0x05</code> <ul style="list-style-type: none">- The PC application displays 00000101 and the value of PORTA beside it (0x05 is the address of PORTA).
watch_register_2	
watch_register_3	
watch_register_4	
watch_register_5	
watch_register_6	
watch_register_7	

There are two things to keep in mind. First, the USB sends a packet every 10 ms, so this is the time-resolution for data updates on the PC application. Secondly, calling one of these macros is a one-time update of the value of the register. For registers with regularly changing values, it is necessary to call the macro repeatedly either in a loop or at crucial junctures in the code. Even so, most of the processing in one's code takes place in a matter of microseconds since one operation on the PIC16 takes 400 μ s – thus, one cannot actually see and monitor the contents of file registers that change this quickly.

One solution is to use breakpoints. Using the I2C, it is easy to implement a breakpoint subroutine that stops the PIC16's operation and waits for the user to press the programmer module's bootloader switch before continuing. Presented below is the implementation:

```
breakpoint
  <call whatever watching macros you want here, for e.g. watch_PORTA>
  bsf_TRISA_0 ; sets the pin the bootloader switch is connected to, to input
  btfsc_PORTA_0 ; waits for the user to press the switch and pull RA0 low
  goto $-1 ; if RA0 is not low, keep looping
  <put a bit of a delay here> ; so that the PIC is not at the next breakpoint by the time you release the switch.
  return ; continue with the program
```

The user should call this subroutine at key junctures in the code to stop execution, transmit register values, read them from the PC, and then push the bootloader switch when ready to continue operation.

7.14 Universal Serial Bus Basics

7.14.1 Introduction

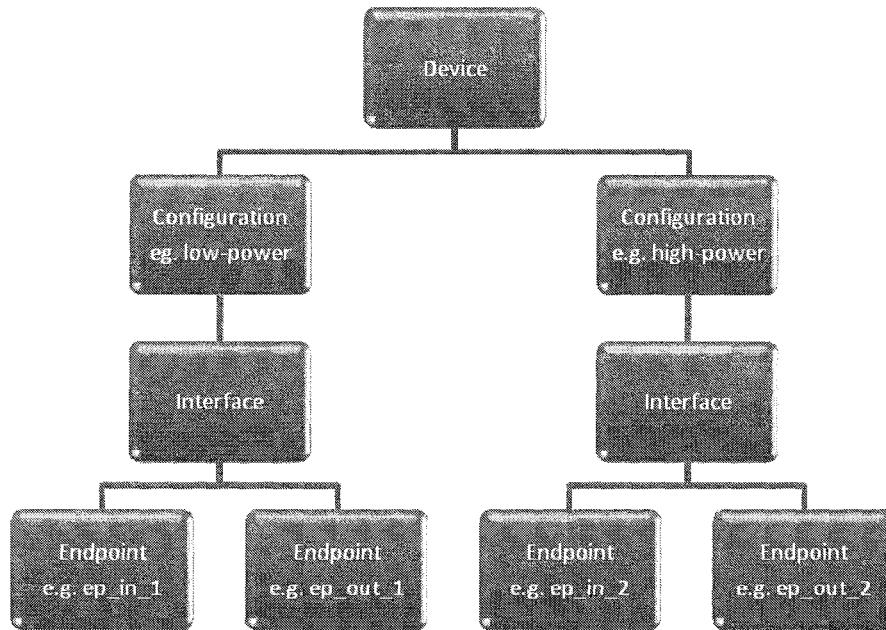
Basics. There are two elements on a USB connection. The host is the PC and the device is the PIC18, in our case. The physical cable consists of four wires – two for power (+5V and GND) and two signal wires. One of the main advantages of USB is that it supports “plug n’ play”. When a USB device is plugged in, the host detects it, retrieves information about it during enumeration, and loads the required drivers, all almost instantaneously as long as the drivers are installed. Assigning the correct driver is done using the device’s product ID and vendor ID, which is used to identify it.

Speed. Most USB ports on today’s computers are USB 1.1 while usually, 2 ports on each PC is USB 2.0. The following table lists the maximum speeds possible for each.

Type	Speed	Supported in USB 1.1	Supported in USB 2.0
Low speed	1.5 Mbps	✓	✓
Full speed	12 Mbps	✓	✓
High speed	480 Mbps		✓

7.14.2 Hierarchy of Layered Framework for Device

On the USB device, there is a hierarchy of layers starting from the device at the top to the endpoints through which data is actually moved.

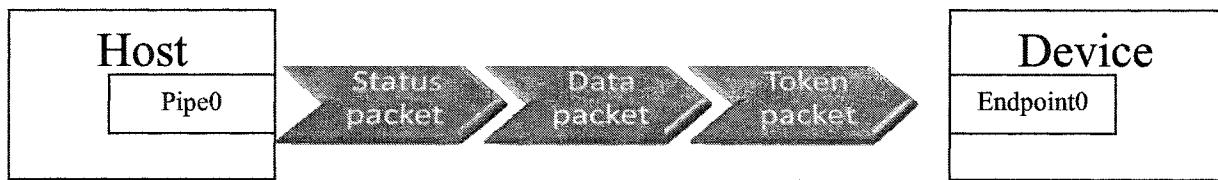


7.14.3 USB Communication

Transactions. A USB transaction has three parts: the token packet, the optional data packet, and the status packet, which represent the header, payload, and data acknowledge, respectively. Since USB is host-driven, the host initiates transactions by sending the token packet. It also sends packets to indicate the start of each frame, every millisecond. Packets on this layer are taken care of, so the developer does not have to worry about such low level transactions.

Endpoints. Endpoints can be thought of as the low-level interface between the hardware and firmware on the device. Packets are sent using specific input or output endpoints. Endpoint zero is used for control and status requests during enumeration.

Pipes. Pipes are the equivalent of endpoints for the host. It is the interface between the host and the endpoints. Since pipes are used by the PC, pipes have bandwidth, transfer type, direction, and maximum packet size associated with them. There are two types: stream pipes for bulk, isochronous, and interrupt transfers; and message pipes for control transfers.



Transfer types. There are 4 transfer types, control, interrupt, bulk, and isochronous. The transfer types are compared in the table below.

Transfer type	Max. rate (full speed)	Delivery guaranteed	Integrity guaranteed
Control	64 kBps	✓	✓
Interrupt	64 kBps	✓	✓
Bulk	64 kBps		✓
Isochronous	1023 kBps	✓	

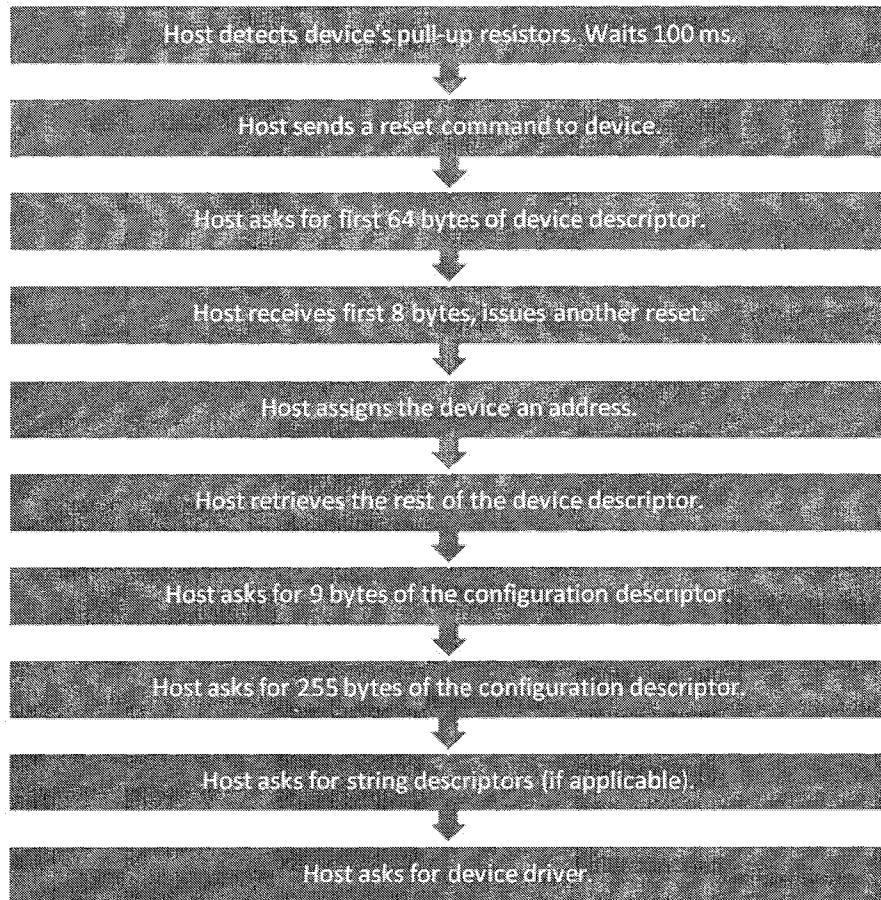
Control transfers are used for device enumeration and other USB device tasks. Interrupt transfers are used when timely delivery as well as data integrity is required, and a low packet size (compared to isochronous transfers) can be tolerated. Bulk transfers are used for bursts of data whose integrity is critical, but do not have to be delivered at a regular, fixed rate. Isochronous transfers are useful from streaming devices that require the delivery of large quantities of data at fixed intervals.

For control, interrupt, and bulk transfers, data integrity is ensured through cyclic redundancy checking and acknowledging reports through the status packet. For interrupt and isochronous transfers, timely delivery is ensured because the host assigns bandwidth to interrupt and isochronous endpoints up to 90% of the total available bandwidth on the bus. For the remaining 10% (or more), priority is given to control transfers and then any remaining space is given for bulk transfers.

Control, interrupt, bulk, and isochronous transfers use multiple transactions each containing a token, data, and status packet to carry out their USB communication.

7.14.4 Enumeration

Enumeration sequence. Upon attachment, the host initiates the enumeration process to retrieve information about the device. These steps are outlined below.



Descriptors. Descriptors contain information coded in the firmware of the device which is sent to the PC when requested during enumeration.

Type of descriptor	Description	Important info
Device	PID/VID, manufacturer, device class	Number of configurations
Configuration	Power requirements	Number of interfaces
Interface	Class of the interface	Number of endpoints
Endpoint	Transfer type and direction	
Report (HID)	Unique to HID	Usage class, size

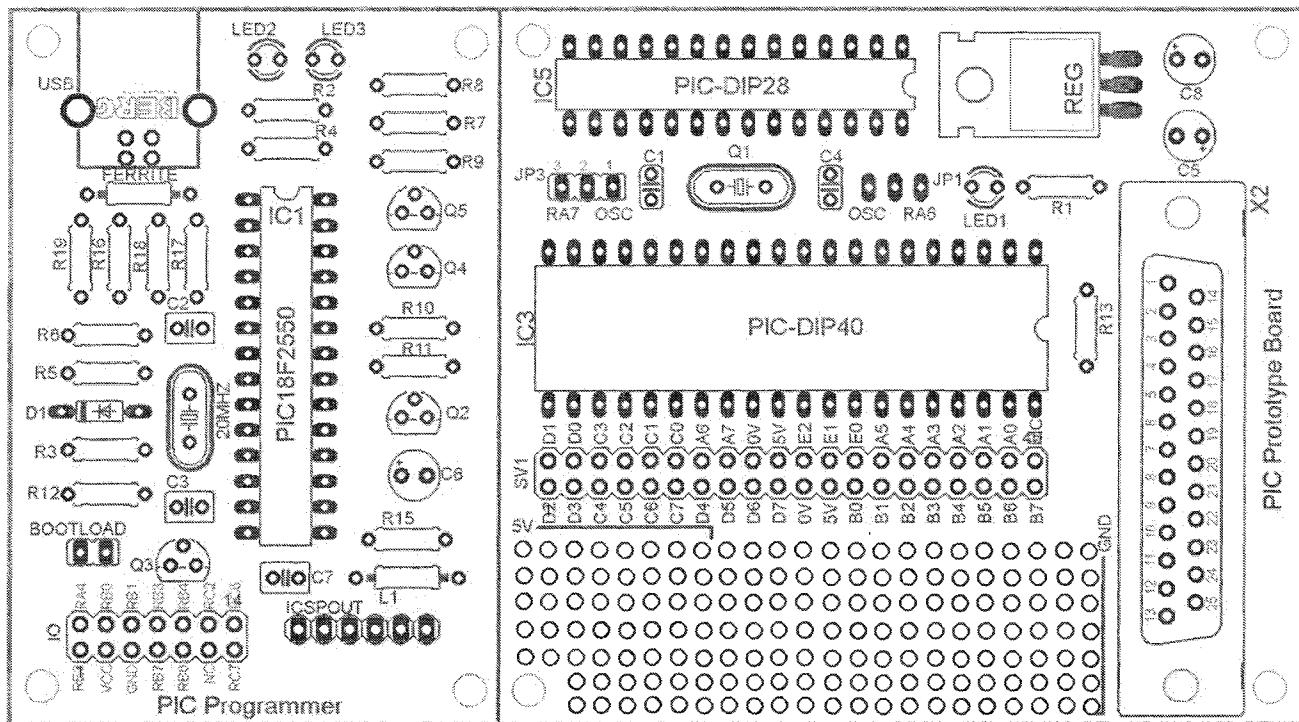
7.15 Some Practical Notes

- Electrostatic Shock – do not touch the pins without grounding yourself.
- Tie the pin used for a switch HI or LO with a ~10K resistor, or use internal pullups to avoid floating inputs when the switch is open. In general, do something with unused pins; do not just leave them float.
- Debounce the switches to avoid false switch states.
- Connect ALL V_{CC} and V_{DD} pins. Not just some of them.
- Despike your power supply to each chip. This means putting a .01 to .1 μF ceramic capacitor from the +5 V supply to ground right at the chip power line. Spikes occur when there are sudden current changes far from the power supply. If there are two or more V_{DD} pins, you need to put bypass capacitors on ALL pins. It is critical to de-couple power supply lines. You want the capacitors to absorb well at the third harmonic of the clock. 0.1 μF does well at 3 MHz, and 0.047 at 20 MHz.
- Make good power supply and ground connections. A skinny, daisy-chained wire-wrap connection from chip to chip is probably not going to be enough. The more robust you can make power supply and ground leads, the better.
- Connect all power supply and ground pins on chips that have more than one V_{DD} and V_{CC} connection.
- Keep the distance between the V_{DD} and V_{SS} pins on the same chip as short as possible. There should be one connection from a chip to the power supply. If two leads are on the chip, the best method is to make a T with equal left/right sides if possible.
- Keep your digital and analog circuitry physically separate whenever you can. Digital switching, especially at microprocessor bus, can throw all sorts of noise and trash into analog or audio circuitry via the power supply.
- Use IC sockets on the boards. After you have things debugged, you might consider soldering chips directly to the boards to save cost and eliminate connections that can oxidize or come loose, but during the design/testing phase you need to be able to swap chips without repeated desoldering.
- Do not trust data sheets labeled "Preliminary Information". This means that the data sheet was written before the chip was actually in production, and the device may change significantly by the time it is actually released. The device performance may be different; still more important, the pinout may be completely rearranged in the final design. Preliminary data can help you choose a device but do not set your design in stone based on it.
- Cover the window on any chip with EPROM even if you don't care about the EPROM part of the chip. PIC processors for example are sometimes affected by non-UV light entering the EPROM window and shining on the silicon. This can actually inject signals into the non-EPROM parts of the chip.

7.16 Computer Boards

7.16.1 Simple Configuration Board

For the final prototype, students may decide to assemble their own PIC driver (and programmer) board(s) for a number of reasons, including saving costs. For this purpose, a printed circuit board (PCB) has been designed and manufactured in the Design Lab that contains both a double-PIC driver and an MPLAB-compatible programmer. Students can purchase the board from the Design Shop. The board layout and information about the required parts are shown below.

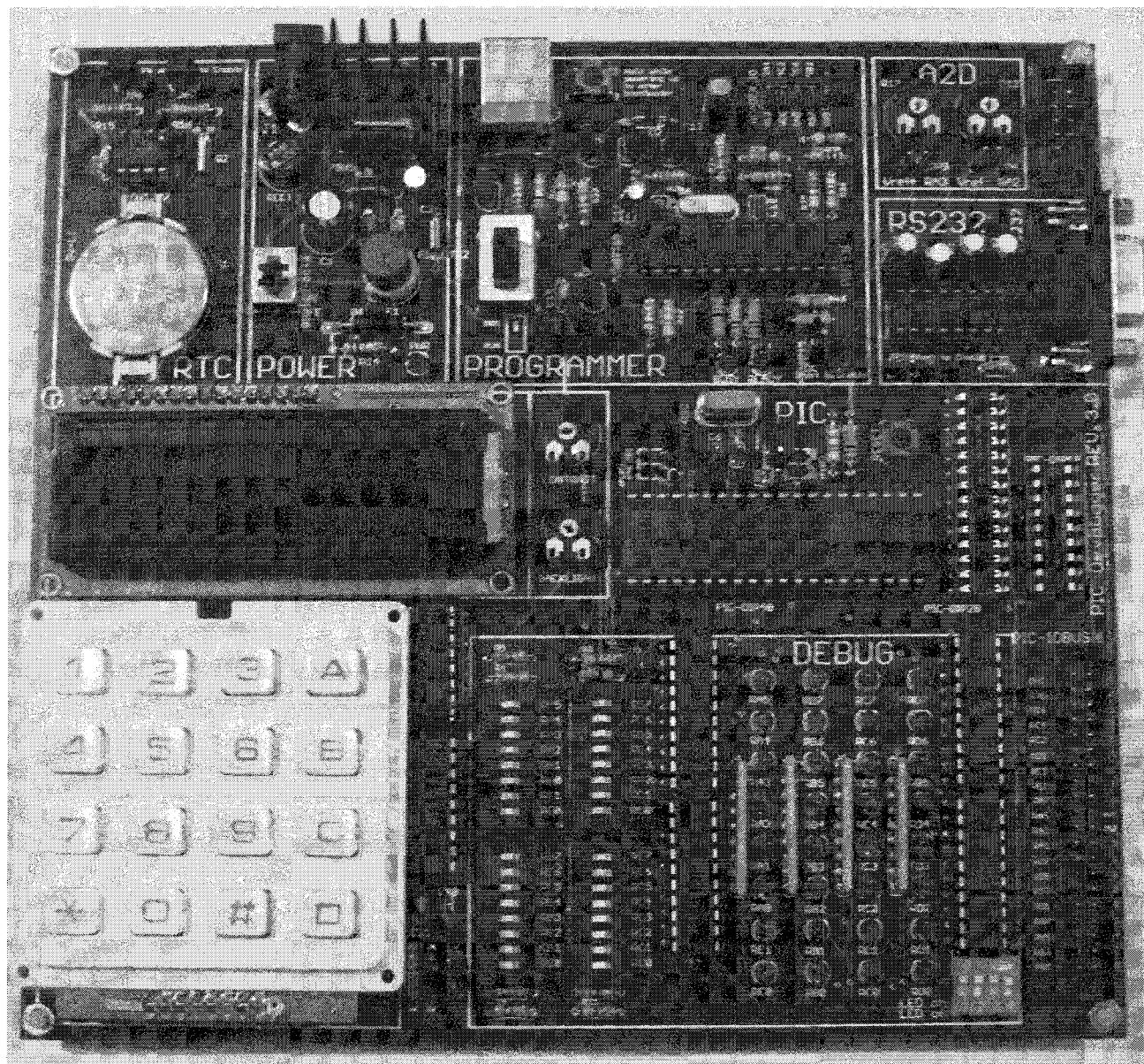


Part Name	Value	Device	Description
20MHZ	20MHz	CRYSTALHC49S	CRYSTAL
BOOTLOAD		JP1E	JUMPER
C1	22pF	C-US025-024X044	CAPACITOR, American symbol
C2	22pf	C-US025-030X050	CAPACITOR, American symbol
C3	22pf	C-US025-030X050	CAPACITOR, American symbol
C4	22pF	C-US025-024X044	CAPACITOR, American symbol
C5		CPOL-USE2.5-5	POLARIZED CAPACITOR, American symbol
C6	47uF	CPOL-USE2.5-5	POLARIZED CAPACITOR, American symbol
C7	0.47uf	C-US025-030X050	CAPACITOR, American symbol
C8		CPOL-USE2.5-5	POLARIZED CAPACITOR, American symbol
D1	1N4148DO35-7	1N4148DO35-7	DIODE
FERRITE		L-US0207/10	INDUCTOR, American symbol

IC1	PIC18F2550_28DIP	PIC18F2550_28DIP	USB Microcontrollers with nanoWatt Technology High-Performance, Enhanced Flash, 28/40/44-Pin,
IC3	PIC16F877P	PIC16F877P	MICROCONTROLLER
IC5	PIC16876P	PIC16876P	MICROCONTROLLER
ICSPOUT		PINHD-1X6	PIN HEADER
IO		MA07-2	PIN HEADER
JP1		JP2E	JUMPER
JP3		JP2E	JUMPER
L1	680uH	L-US0207/10	INDUCTOR, American symbol
LED1		LED3MM	LED
LED2	RED	LED3MM	LED
LED3	Green	LED3MM	LED
Q1		CRYSTALHC49S	CRYSTAL
Q2	2N3906	2N3906	PNP Transistor
Q3	2N3904	2N3904	NPN Transistor
Q4	2N3904	2N3904	NPN Transistor
Q5	2N3904	2N3904	NPN Transistor
R1	1k	R-US 0207/7	RESISTOR, American symbol
R2	470	R-US 0207/7	RESISTOR, American symbol
R3	1k	R-US 0207/7	RESISTOR, American symbol
R4	470	R-US 0207/7	RESISTOR, American symbol
R5	4.7k	R-US 0207/7	RESISTOR, American symbol
R6	2.7k	R-US 0207/7	RESISTOR, American symbol
R7	100k	R-US 0207/7	RESISTOR, American symbol
R8	10k	R-US 0207/7	RESISTOR, American symbol
R9	100	R-US 0207/7	RESISTOR, American symbol
R10	10k	R-US 0207/7	RESISTOR, American symbol
R11	10k	R-US 0207/7	RESISTOR, American symbol
R12	10k	R-US 0207/7	RESISTOR, American symbol
R13	10k	R-US 0207/7	RESISTOR, American symbol
R15	4.7k	R-US 0207/7	RESISTOR, American symbol
R16	100	R-US 0207/7	RESISTOR, American symbol
R17	100	R-US 0207/7	RESISTOR, American symbol
R18	4.7k	R-US 0207/7	RESISTOR, American symbol
R19	4.7k	R-US 0207/7	RESISTOR, American symbol
REG	7805T	7805T	Positive VOLTAGE REGULATOR
SV1		MA20-2	PIN HEADER
USB	PN61729-S	PN61729-S	BERG USB connector
X2		F25V	D-SUB 25

7.16.2 The PIC DevBugger

The PIC DevBugger development board was designed as a complete mobile solution to the PIC development, including a full-speed USB programmer and a number of peripheral modules. One especially useful feature of the board is the debugging module, which monitors all pin states and allows the user to simulate inputs to the PIC. DevBugger can host a variety of PIC microcontrollers, in particular a combination of PIC16F877 or PIC18F4620 and PIC18F2550 as parallel processors. It also includes an integrated keypad, LCD, and real-time clock modules. The board is ideal for students, as it can be used to quickly develop and test code. Its uses can vary from an exclusive development platform to a full-scale embedded processing/control system used in a final design.



7.16.2.1 Features

- User-changeable oscillator clock (10MHz crystal included)
- In-circuit USB High Voltage Programmer
- On-board RS232 peripheral including female socket and level converter
- Power supply compatible with voltages ranging from 7.5VDC to 17VDC
- Two-way power through DC adapter or USB
- Real Time Clock peripheral with 32.768khz crystal and battery socket
- I2C bus expansion socket
- On-board HD44780 LCD socket with contrast and backlight controls
- On-board 4x4 keypad socket and signal encoder
- 40-pin I/O bus with ribbon cable connector
- On-board A2D voltage reference settings
- Debugging Module with 32 indicator LEDs and signal-simulation buttons
- Programmer firmware can be modified to act as extra processor/memory

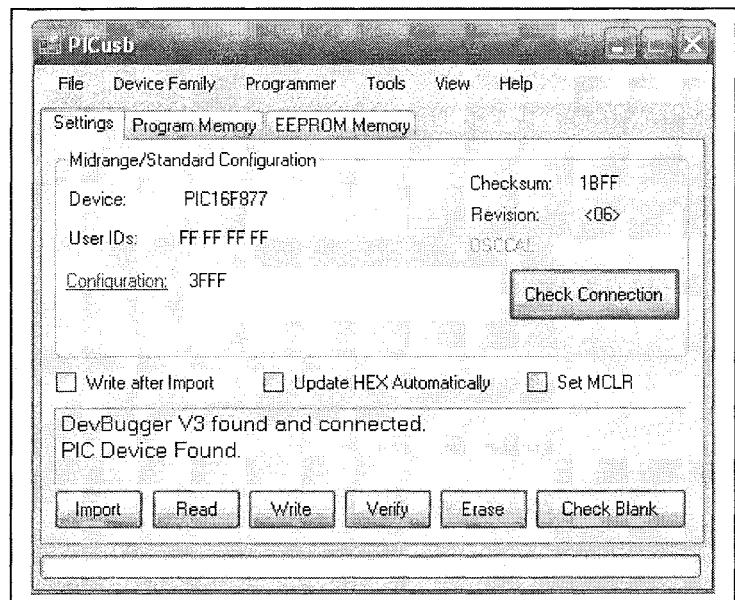
For more information about the board please refer to the DevBugger Manual.

7.16.2.2 Programming

An application called **PICusb** is provided, which is designed specifically to communicate with the DevBugger hardware.

The user interface for **PICusb** is fairly intuitive, with reasonable defaults for all settings. For those without prior experience with PIC programming software, the following steps can generally be used to load the HEX code onto the device:

1. Connect the *DevBugger* to the PC.
2. Flip the programming switch to PRG mode.
3. Turn on the DevBugger.
4. Click “Check Connection” on the **PICusb** window shown to the right. Wait for the application to detect the DevBugger and the PIC device.
5. Click “Import” or go to File→Import Hex to load a HEX file.
6. Press the Write button to load the HEX file to the PIC device. If no error message occurs, then the PIC should have been programmed successfully.



Note, if the DevBugger is disconnected at any point, it must be reconnected to the application by pressing the “Check Connection” button before programming.

If any errors occur or the application cannot find the DevBugger or the PIC device, make sure the programming switch is in PRG mode, turn off the power to reset the DevBugger and try again.

In some special cases, if a device appears to be malfunctioning and is not automatically detected, the problem may be resolved by going to Tools→Use VPP First Program Entry. This allows the programmer to program the PIC device if the code on the target PIC device is corrupt.

The following is a description of the common operations that may be performed upon the PIC device currently used in the *DevBugger*.

Import loads a HEX file into memory, for loading onto a PIC device or for verification against the code already loaded onto a PIC device.

Erase performs a bulk erase of the device, effectively returning it to its factory state. In some cases, this may fix a device that appears faulty.

Check Blank is used to verify that the chip is indeed ‘blank’; this is useful after performing an **Erase Chip** operation to verify that the operation succeeded.

Write programs the currently loaded HEX code into the target device. Several options for this operation are available. By default, all aspects of the HEX file are programmed into the device; however, checkboxes in the Program Memory tab and the EEPROM Memory tab can allow sections of the HEX file to be programmed.

Verify reads the contents of the PIC device and compares it against the loaded HEX file. If any differences are observed, an error is given and the operation fails.

Read is used to read the current program loaded into the target PIC device. This program can be retrieved and stored in a HEX file. It also retrieves the EEPROM data stored in the PIC.

Alternatively, the DevBugger programmer is fully supported by MPLAB IDE and the PIC device can be programmed through MPLAB. For more information about programming, refer to the DevBugger Manual.

7.17 Resources

J. B. Peatman, *Design with PIC Microcontrollers*, NJ: Prentice Hall, 1997.

M. Predko, *Programming and Customizing PICmicro Microcontrollers*, McGraw-Hill, 2nd ed., 2001.

J. B. Peatman, *Embedded Design with the PIC18F452 Microcontroller*, NJ: Prentice Hall, 2003.

D. Benson, *Pic 'n up the Pace*, CA: Square 1 Electronics, 1999.

D. Benson, *Pic 'n Techniques*, CA: Square 1 Electronics, 1999.

PIC16F877 Datasheet: This datasheet describes the architecture and functionality of the 16F877 microcontroller.

PIC16F887 Datasheet: This datasheet describes the architecture and functionality of the 16F887 microcontroller.

PIC16F1937 Datasheet: This datasheet describes the architecture and functionality of the 16F1937 microcontroller.

PIC18F4620 Datasheet: This datasheet describes the architecture and functionality of the 18F4620 microcontroller.

PIC18F2550 Datasheet: This datasheet describes the architecture and functionality of the 18F2550 microcontroller.

PIC Mid-Range Reference Manual: This manual describes the PIC16XXX family architecture and peripherals but not the specifics of each device. It is intended as a complement to the device datasheets.

PIC Assembly code templates : When you unzip *code.zip* contains templates for all PIC devices. The file *f877temp.asm* is the template for the 16F877 chip.

MPLAB Manual and Tutorial: This tutorial will instruct you how to use MPLAB for writing, compiling, and simulating the MPASM codes.

MPLAB 5.40 Package: The complete package has been installed on all machines in the engineering Design Lab. You can also download a version of this software.

[**www.piclist.com**](http://www.piclist.com) a huge source of information about different types of PIC microcontrollers.

[**www.gnupic.org**](http://www.gnupic.org) a collection of different source codes, compilers, simulators, programmers, etc. for PIC microcontrollers.

Chapter 8

Sensors

8.1	Introduction	1
8.2	Performance Characteristics	2
8.3	Photoelectric Sensors	4
8.3.1	Photoelectric Emitters	5
8.3.2	Photoelectric Receivers	6
8.3.2.1	Photoresistors	6
8.3.2.2	Photodiodes	7
8.3.2.3	Phototransistors	9
8.3.3	Optoisolators	10
8.3.4	Reflective Optosensors	11
8.3.5	Break-Beam Optosensors	12
8.3.6	Infrared Sensing	12
8.3.6.1	Modulation and Demodulation	13
8.3.6.2	IR Communication	15
8.3.6.3	“Seeing” IR	16
8.3.7	Applications	17
8.3.7.1	Shaft Encoding	17
8.3.7.2	Line Following	18
8.3.7.3	Proximity Sensing	18
8.3.7.4	Distance Sensing	19
8.3.7.5	Voice Communication	20
8.3.7.6	Motion Sensing	22
8.4	Inductive Sensors	23
8.4.1	Fundamentals	23
8.4.2	Application	26
8.5	Switch Sensors	27
8.6	Other Sensors	28
	Reed Switch	28
	Tilt Switch	28
	Force Sensitive Resistor (FSR)	28
	Thermistor	28
	Hall Effect Sensor	28
8.7	Transducers	29
	Microphone	29
	Digital Compass	29
	Gyroscope	29
	Accelerometer	29
	Humidity and Temperature Sensing	29
	Sonar Range Detector	29
	PIR Motion Sensor	29
8.8	Practical Notes	30
8.9	Odometry	31

8.1 Introduction

A *sensor* is defined as a device that is sensitive to one or several types of energy, such as motion, light, heat, electrical, magnetic, pressure, etc. The other two words frequently used related to sensors are *transducer* and *transmitter*. A *transducer* is defined as a device that can receive one type of energy and convert it to another type of energy, most frequently electrical energy. Hence, the output of a transducer is typically one of the features of the electrical circuits, such as voltage, current, resistance, frequency, capacitance, or inductance. In this course, both terms sensor and transducer are used interchangeably. As such, sensors/transducers represent part of the interface between the physical world and the world of electrical devices, such as computers. The other part of this interface is represented by *actuators*, which convert electrical signals into physical phenomena. A *transmitter* is a device that can convert a very small signal to a more usable signal for the control units. Transmitters usually use devices such as op amps to amplify and linearize the sensor output signal. They may also provide circuits calibrating the sensor signal with other parts of the electrical system.

Sensors can be categorized based on their applications and/or based on the input energy that they mostly convert to the output electrical signal. A list of some popular sensor applications in the industry is given in table below.

Application	Type	Explanation
Displacement, Position, and Proximity	Potentiometer	A resistive element with a movable contact (wiper), hence a resistor whose resistance varies as a function of displacement.
	Strain Gauge	Measures strain by means of the change in resistance of a metallic element bonded to the object.
	Capacitive	Probe and conductive target form parallel plates of a capacitor. Change in spacing and change in capacitance modulate the output of an oscillator.
	Linear (Rotary) Variable Differential Transformer (LVDT, RVDT)	Consists of a movable coil (primary) and two secondary coils symmetrically spaced on a cylindrical form. Core couples excitation voltage in primary to the two secondary coils. Phase and amplitude of the secondary outputs vary with the position of the core.
	Eddy Current Proximity	High frequency magnetic field induces eddy currents in a conductive target. Secondary coil senses changes of eddy currents as a function of displacement.
	Inductive Proximity	Similar to LVDT, but only two inductive coils, connected in a bridge circuit.
	Optical Encoder	A slotted disk mounted on the shaft rotates between a transmitter and receiver. Widely used in machine tools. Incremental vs. absolute optical.
	Pneumatic	Displacement or proximity of an object is transformed into a change of the pressure of compressed air.
	Hall-Effect Sensor	Induced voltage orthogonal to current flow in certain semiconductors when a semiconductor experiences a magnetic field (needs a magnet.)
	Microswitch	Simple pushbutton or lever switch to detect the limit or end of travel of the part.
Velocity, Acceleration	Encoder	A differentiator circuit makes the difference of the encoder signal output in constant time intervals.
	Tachogenerator	A DC or AC generator that can determine the speed of shaft by the amount of voltage the generator produces or the frequency of the output signal.
Force, Pressure	Strain Gauge Load Cell	Uses the electrical resistance strain gauges to monitor the strain produced in a member when stretched, compressed or bent by the application of a force.

	Piezoelectric Sensor	Consists of quartz or ceramic crystals that generate an electrostatic charge output when force is exerted on it.
	Tactile Sensor	A particular form of pressure sensor. Uses two layers of the piezoelectric film that are separated by a soft film which transmits vibrations. When pressure is applied to the upper layer, its vibrations are affected and the output alternating voltage will change.
Temperature	Bimetallic Strips	Two bands of dissimilar metals are attached together. Change of temperature will cause the total band to bend due to different elongation of the two metal bands.
	Resistance Temperature Detector (RTD)	Based on the increase of resistivity with temperature in most materials.
	Thermistor	Semiconductor sensor whose resistance changes exponentially with temperature.
	Thermocouple	Two wires of dissimilar metals joined at one end will produce an emf across their junction.
	Pyroelectric Sensor	Uses Pyroelectric, crystalline materials that generate charge in response to heat flow.

8.2 Performance Characteristics

Since the output of the sensors is typically an electrical signal, we tend to characterize sensors in the same way as electronic devices are characterized. The data sheets for many sensors are formatted just like those for the electronic products. However, due to lack of an international standard for sensor specifications, one may encounter a variety of interpretations of sensor performance parameters, and sometimes a great deal of confusion will emerge. It is important to realize that this confusion is not due to user's inability to explain the meaning of the terms. It is a result of the fact that different parts of the sensor community have gotten comfortable using different terminologies. Although it is critical to understand the data brought in the sensor data sheets, you should realize that data sheets are primarily marketing documents. They are designed to highlight the positive attributes of the sensor, emphasize some of the potential uses of the sensor, and might neglect to comment on some of the negative characteristics of the sensor. In many cases, the sensor has been designed to meet a particular performance specification for a specific customer, and the data sheet will concentrate on the performance parameters of greatest interest to this customer. In this case, the vendor and customer might have grown accustomed to unusual definitions for certain sensor performance parameters. As a potential new user of such a sensor, it is initially your problem to recognize this situation, and interpret things reasonably. So, expect that you will encounter odd definitions here and there, and expect that you will find that most sensor data sheets are missing some information that you might be most interested in. That is the nature of the business. Some general terms for characterizing the sensor performance are explained below.

Transfer Function: The functional relationship between physical input signal and electrical output signal. Usually, this relationship is represented as a graph showing the relationship between the input and output signal, and the details of this relationship may constitute a complete description of the sensor characteristics. For expensive sensors that are individually calibrated, this might take the form of the certified calibration curve.

Sensitivity: The sensitivity is defined in terms of the relationship between input physical signal and output electrical signal. The sensitivity is generally the ratio between a small change in electrical signal to a small change in physical signal. As such, it may be expressed as the derivative of the transfer function with respect to physical signal. As an example, a thermometer would have "high sensitivity" if a small temperature change resulted in a large voltage change.

Span or Dynamic Range: The range of input physical signals that may be converted to electrical signals by the sensor. Signals outside of this range are expected to cause unacceptably large inaccuracy. This span or dynamic range is usually specified by the sensor supplier as the range over which other performance characteristics described in the data sheets are expected to apply.

Accuracy: Generally defined as the largest expected error between actual and ideal output signals. Sometimes this is quoted as a fraction of the full scale output. For example, a thermometer might be guaranteed accurate to within 5% of FSO (Full Scale Output).

Hysteresis: Some sensors do not return to the same output value when the input stimulus is cycled up or down. The width of the expected error in terms of the measured quantity is defined as the hysteresis.

Nonlinearity (often called Linearity): The maximum deviation from a linear transfer function over the specified dynamic range. There are several measures of this error. The most common compares the actual transfer function with the “best straight line,” which lies midway between the two parallel lines which encompasses the entire transfer function over the specified dynamic range of the device. This choice of comparison method is popular because it makes most sensors look the best.

Noise: All sensors produce some output noise in addition to the output signal. In some cases, the noise of the sensor is less than the noise of the next element in the electronics, or less than the fluctuations in the physical signal, in which case it is not important. Many other cases exist in which the noise of the sensor limits the performance of the system based on the sensor. Noise is generally distributed across the frequency spectrum. Many common noise sources produce a white noise distribution, which is to say that the spectral noise density is the same at all frequencies.

Resolution: The resolution of a sensor is defined as the minimum detectable signal fluctuation. Since fluctuations are temporal phenomena, there is some relationship between the timescale for the fluctuation and the minimum detectable amplitude. Therefore, the definition of resolution must include some information about the nature of the measurement being carried out. Many sensors are limited by noise with a white spectral distribution.

Bandwidth: All sensors have finite response times to an instantaneous change in physical signal. In addition, many sensors have decay times, which would represent the time after a step change in physical signal for the sensor output to decay to its original value. The reciprocal of these times correspond to the upper and lower cutoff frequencies, respectively. The bandwidth of a sensor is the frequency range between these two frequencies.

8.3 Photoelectric Sensors

A photoelectric sensor is an electrical device that responds to a change in the intensity of the light falling upon it. Light is a quantum-mechanical phenomenon. It comes in discrete particles called *photons*. Photons are the elemental units of electromagnetic radiation, which have interesting features. They have no rest mass, but they do carry momentum (energy). A photon also has a wave-like characteristic. The wavelength of a photon λ (horizontal distance between consecutive electrical or magnetic field peaks) depends on the medium in which it travels as well as the source that has produced it. It is this wavelength that determines the color of a photon. A photon's frequency is related to its wavelength by

$$\omega = 2\pi c/\lambda$$

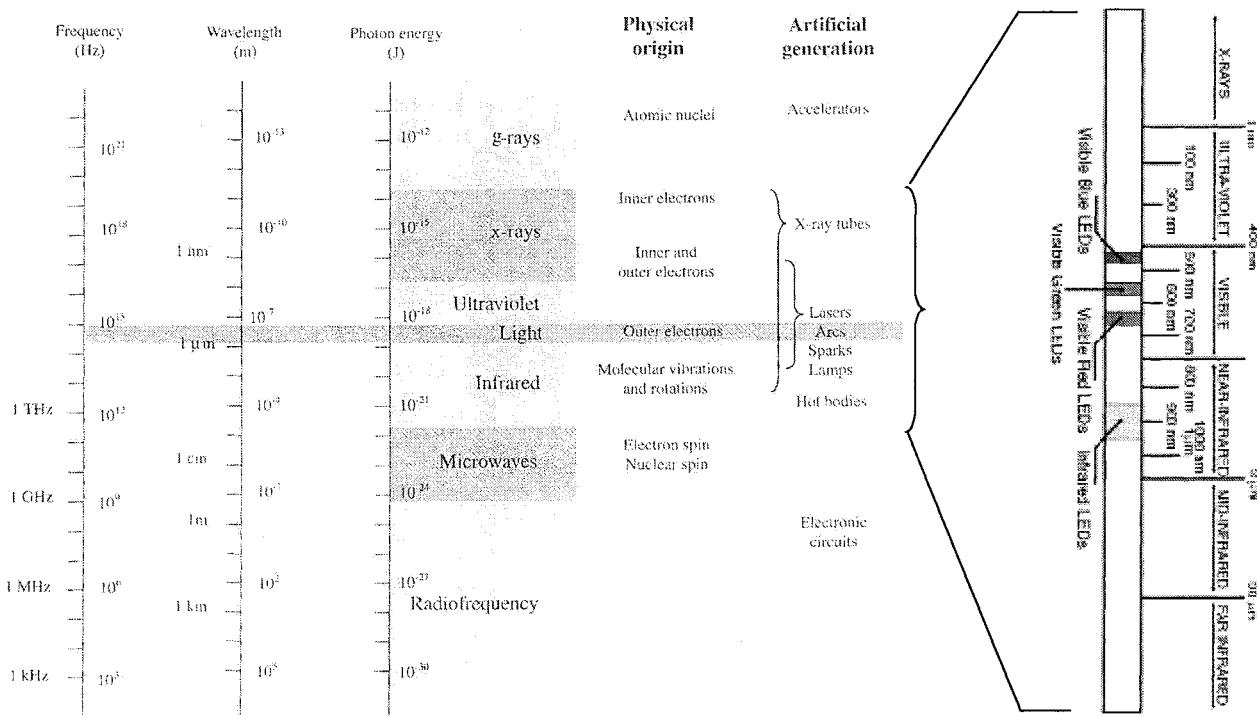
where c is the speed of the photon. In free space, c is equal to the speed of light ($c = 3 \times 10^8 \text{ m/s}$), but in other media, such as glass, c becomes smaller than the speed of light. A photon with a large wavelength (or small frequency) is less energetic than a photon with a shorter wavelength (or a higher frequency). The energy of a photon is equal to:

$$E = hc/\lambda$$

where h is Planck's constant ($h = 6.67 \times 10^{-34} \text{ J} \cdot \text{s}$). Visible light photons are perceived by brain in different colours ranging from red to blue, depending on their wavelength. It is important to note that there is no such a thing as white photon. Instead, when the combination of the various coloured photons interact with our eyes, our brain perceives what is called *white light*. Photons are not limited to visible light alone. There are also radio-frequency photons, infrared photons, microwave photons, and other kinds of photons that our eyes cannot detect.

The trick to "making" a photon is to accelerate/decelerate a charged particle. For example, an electron that is made to vibrate back and forth within an antenna will produce radio-frequency photons that have very long wavelengths (low energies) when compared with light photons. Visible light, on the other hand, is produced when outer-shell electrons within atoms are forced to make transitions between energy levels, accelerating in the process. Other frequency photons may be created by vibrating or rotating molecules very quickly, while still others, specifically those with very high energy (e.g. Gamma rays), can be created by the charge accelerations within the atomic nuclei. Figure below shows the breakdown of the electromagnetic spectrum.

In all photo-sensing cases, the energy of the photon determines how we detect it. Light detectors can be broken into two categories. The *Quantum* detectors convert incoming radiation directly into an electron in a semiconductor device, and process the resulting current with electronic circuitry. The *Thermal* detectors simply absorb the energy and operate by measuring the change in temperature with a thermometer. Here, the focus is on the Quantum detectors, which mostly offer the best performance for detection of optical radiation. In the quantum detectors, the photon is absorbed and an electron is liberated in the structure with the energy of the photon. Semiconductors feature the basic property that

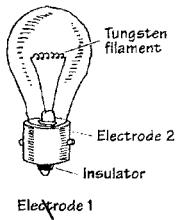
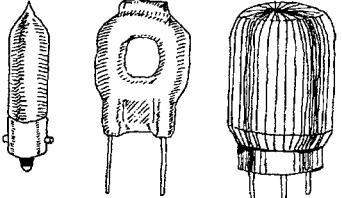
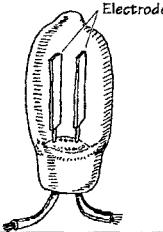
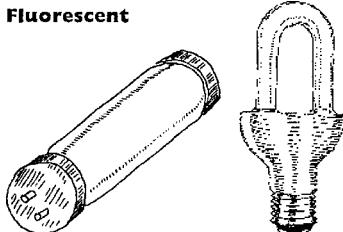
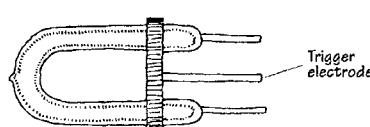


electrons are allowed to exist only at certain energy levels. Thus, if the photon carries an amount of energy that is "allowed" for an electron in the semiconductor, it can be absorbed. Once it is absorbed, the electron moves freely within the device, subject to electric fields (due to applied voltages) and other effects.

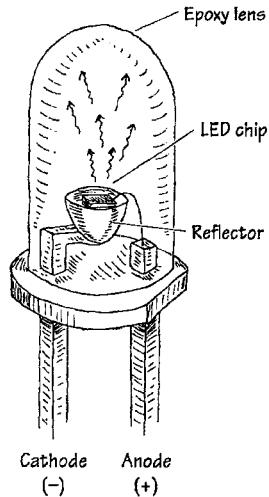
Photoelectric sensing requires an emitter (to generate light) and a receiver (which "sees" the light from the emitter). A variety of sensing "modes" exist, for the most part based on the emitter and receiver's physical orientation to each other.

8.3.1 Photoelectric Emitters

Two major sources of light energy are lamps and light-emitting diodes (LED). Different types of lamps have been described in table below. A lamp's brightness is measured in what is called the *Mean Spherical Candle Power* (MSCP). Bulb manufacturers place a lamp at the centre of an integrating sphere that averages the lamp's light output over its surface. The actual value of the MSCP for a lamp is a function of color temperature of the emitting surface of the lamp's filament. For a given temperature, doubling the filament's surface area doubles the MSCP.

Incandescent 	These lamps use a tungsten wire filament to produce a glowing white light when current passes through it. The filament is enclosed in an evacuated glass bulb filled with a gas such as argon, krypton, or nitrogen that helps increase the brilliance of the lamp and also helps prevent the filament from burning out (as would be the case in an oxygen-rich environment). Incandescent lamps are used in flashlights, home lighting, and as indicator lights. They come in a variety of different sizes and shapes, as well as various current, voltage, and candlelight power ratings.
Halogen 	Similar to a typical incandescent lamp, these lamps provide ultra bright output light. Unlike a typical incandescent lamp, the filament is coated on the inside of a quartz bulb. Within this bulb, a halogen gas, such as bromine or iodine, is placed. These lamps are used in projector lamps, automotive headlights, strobe lights, etc.
Gas-Discharge 	This lamp produces a dim, pale light that results from the ionization of neon gas molecules between two electrodes within the bulb. Types of gas-discharge lamps include neon, xenon flash, and mercury vapor lamps. Gas-discharge lamps have a tendency to suddenly switch on when a particular minimum operating voltage is met. For this reason, they are sometimes used in triggering and voltage-regulation applications. They are also used as indicator lights and for testing home ac power outlets.
Fluorescent 	This lamp consists of a mercury vapor-filled glass tube whose inner wall is coated with a material that fluoresces. At either end of the tube are cathode and anode incandescent filaments. When electrons emitted from an incandescent cathode electrode collide with the mercury atoms, ultraviolet (UV) radiation is emitted. The UV radiation then collides with the lamp's fluorescent coating, emitting visible light in the process. Such lamps require an auxiliary glow lamp with bimetallic contacts and a choke placed in parallel with the cathode and anode to initiate discharge within the lamp. These are highly efficient lamps that are often used in home lighting applications.
Xenon Flash Lamp 	This is a gas-discharge lamp that is filled with a xenon gas that ionizes when a particular voltage is applied across its electrodes. These lamps come with three leads: an anode, a cathode, and a trigger-voltage lead. Normally, a particular voltage is applied across the anode and cathode leads, and the lamp is off. However, when a particular voltage is applied to the trigger lead, the gas suddenly ionizes and releases an extremely bright flash. These lamps are used in photographic applications and in special-effect lighting projects.

Most photoelectric sensors use LEDs (Light Emitting Diodes) as a light source. An LED is a solid-state two-lead semiconductor, similar to a *pn*-junction diode, except that it emits a small amount of light (visible or infrared) when current flows through it in the forward direction, i.e., when the LED's anode lead is made more positive in voltage than its cathode lead (by at least 0.6 to 2.2V). LEDs can be built to emit a particular colour photon, such as green, blue, blue-green, yellow, red, or infrared light. The LED colours most commonly used in sensing are visible red and infrared. In those applications that sense colour contrasts, the choice of LED colour can be important. Infrared LEDs are designed to emit photons with a wavelength between 880 and 940 nm. They tend to have a narrower viewing angle when compared with a visible-light LED so that transmitted signal can be directed efficiently. Photon output is characterized in terms of output power per specific forward current. Typical outputs for infrared LEDs range from around 0.50mW/20mA to 8.0mW/50mA. Maximum forward voltages at specific forward currents range from about 1.6V at 20mA to 2V at 100mA. Since LEDs are solid-state, they will last for the entire useful life of a sensor. With the exception of infrared types, LEDs produce less light than incandescent and fluorescent light sources; often less light than the ambient light level around them. *Modulation* of the sensor's light beam provides the sensing power needed to sense reliably at these low-light levels. Most modern photoelectric sensors use modulated LED emitters. More technical information about LEDs is given in Chapter 4.



8.3.2 Photoelectric Receivers

Photoelectric receivers typically use one of the three light-sensitive electronic elements:

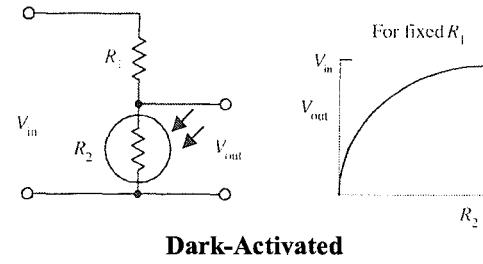
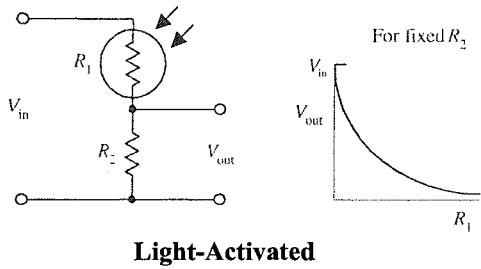
- *Phototransistors* are the most widely used receiver opto-elements in industrial photoelectric sensor design, because they offer the best tradeoff between light sensitivity and response speed compared to photoresistive and other photo-junction devices. Phototransistors respond well to both visible and infrared light.
- *Photoresistors* (or as a series, photocells) are used whenever greater sensitivity to visible wavelengths is required, as in some colour mark and ambient light detection applications. Photoresistors do not respond to infrared light. Photoresistors are easy to work with because electrically they are just resistors, but their response time is slow compared to photodiodes or phototransistors. Hence, photoresistors are suitable for detecting levels of ambient light or acting as break-beam sensors in low frequency applications.
- *Photodiodes* are generally reserved for applications requiring either extremely fast response time or linear response over several magnitudes of light level changes.

8.3.2.1 Photoresistors

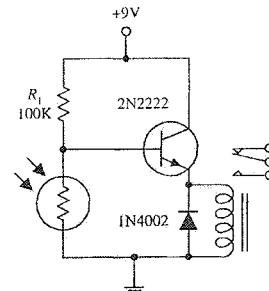
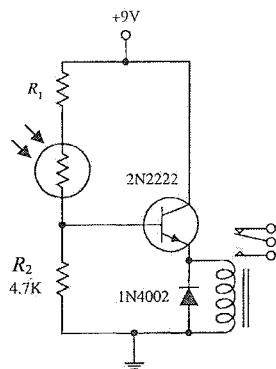
Photoresistors are light-controlled variable resistors. In terms of operation, a photo resistor is usually very resistive (in the Mega ohms) when placed in the dark. However, when it is illuminated, its resistance decreases significantly; it may drop as low as a few hundred ohms, depending on the light intensity. In terms of applications, photoresistors are used in light- and dark-activated switching circuits and in light-sensitive detector circuits. Photoresistors may require a few milliseconds or more to fully respond to changes in light intensity and may require a few seconds to return to their normal *dark resistance* once light is removed. In general, although photoresistors function in a similar manner, the sensitivity and resistance range of a photoresistor may vary greatly from one device to the next. Also, certain photoresistor may respond better to light that contains photons within a particular wavelength of the spectrum. For example, cadmium sulfide photoresistors (CdS) respond best to light within the 400- to 800-nm range, whereas lead sulfide photoresistors respond best to infrared photons.

One basic application of photoresistors is as the light-sensitive voltage divider circuit. Depending on the location of the photoresistor, the circuit can be light- or dark-activated, as illustrated in figure below. The output voltage is calculated as follows:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$



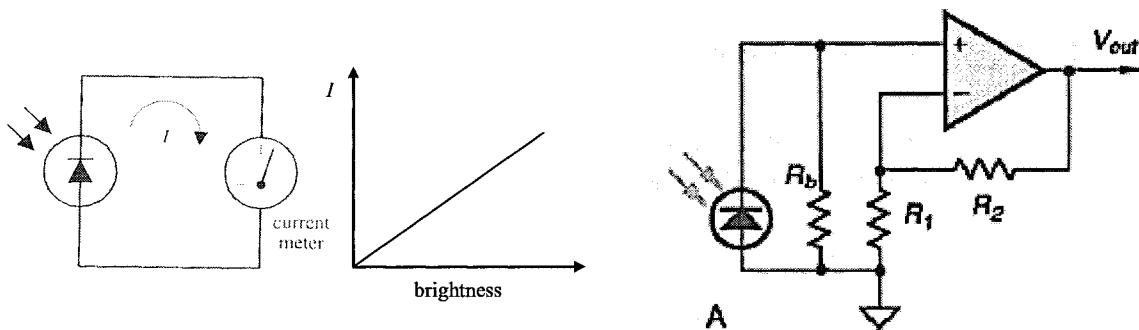
The following two circuits use light-sensitive voltage dividers to trip a relay whenever the light intensity varies. In the light-activated circuit, when light comes in contact with the photoresistor, its resistance decreases, causing an increase in the transistor's base current and voltage. If the base current and voltage are large enough, the transistor will allow enough current to pass from collector to emitter, triggering the relay. The dark-activated relay works in a similar but opposite manner. The value of R_1 in the light-activated circuit could be around $1\text{ K}\Omega$ but may need some adjusting. The resistor R_1 in the dark-activated circuit ($100\text{ K}\Omega$) may also need adjusting. A 6 to 9 V relay with 500Ω coil can be used in either circuit.



In some applications, it is required or recommended to use a pair of sensors, instead of a single one, and base the resultant command on the difference between the two received signals. This method could be very efficient in eliminating the effect of ambient light and/or directing towards a light source. In these cases, both resistors of the voltage divider circuits shown above can be photoresistors, and the output voltage will be proportional to the difference between their light sensing signals. Furthermore, to block the sensor from unwanted ambient light or to direct the sensing direction, one may have to build optical shield around the sensor. Optical shields are just black tubes, typically constructed out of construction paper or heat-shrink tubing, that mount in front of the photo sensor.

8.3.2.2 Photodiodes

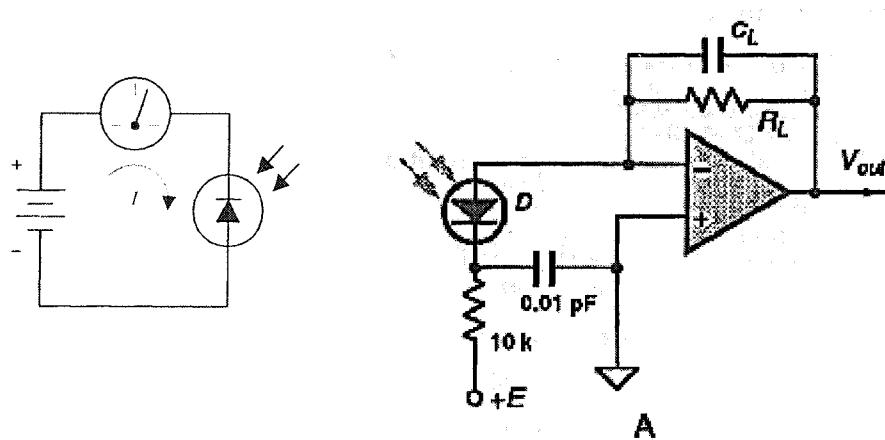
Photodiodes are two-lead devices that convert light energy directly into electric current. If the anode and cathode leads of a photodiode are joined together by a wire and then the photodiode is placed in the dark, no current will flow through the wire. However, when the photodiode is illuminated, it suddenly becomes a small current source that pumps current from the cathode through the wire and into the anode. Photodiodes are used most commonly to detect fast pulses of near-infrared light used in wireless communications. They are often found in light-meter circuits (e.g., camera light meters, intrusion alarms, etc.) because they have quite linear light/current responses. Photodiodes come in different shapes and sizes. Some come with built-in lenses, some come with optical filters, some are designed for high-speed responses, some have large surface areas for high sensitivity, and some have small surface areas. Note that when the surface area of a photodiode increases, the response time tends to slow down.



Photovoltaic Mode

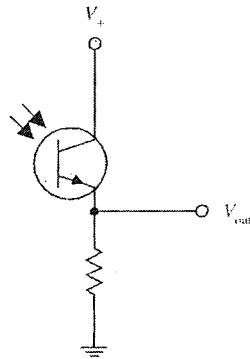
Photodiodes may be biased and operated in two basic modes: *photovoltaic* and *photoconductive*. In the photovoltaic mode, the diode acts to convert light energy directly into the electric current that can be measured by a meter. The input intensity of the light (brightness) and the output current are nearly linear. However, individual photodiodes may not produce enough current needed to drive a particular light-sensitive circuit. Practically, in this mode, the diode is attached to a virtual ground preamplifier as shown in figure below, and the arrival of light photons causes the generation of a voltage which is amplified by the op amp. The primary feature of this approach is that there is no DC-bias across the diode, and so there is no basic leakage current across the diode aside from thermally-generated currents. This configuration does suffer from slower response because the charge generated must charge the capacitance of the diode, causing an RC delay.

In the photoconductive mode, the photodiode is connected in reverse-biased direction with a battery. When dark, a small current called the dark current (within the nA range) flows through the photodiode. When the photodiode is illuminated, a larger current flows. This circuit, unlike the photovoltaic circuit, uses a battery for increased output current. A resistor placed in series with the diode and battery can be used to calibrate the meter. Note that the photodiode in the photoconductive circuit is reverse-biased to produce the current. Practically, in this mode, the diode is biased, and the current flowing across the diode is converted to a voltage (by a resistor), and amplified, as shown in figure below. The primary advantage of this approach is that the applied bias decreases the effective capacitance of the diode (by widening the depletion region), and allows for faster response. Unfortunately, the DC bias also causes some leakage current, so detection of very small signals is compromised.

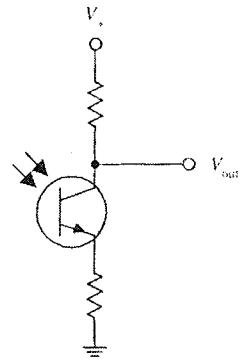


Photoconductive Mode

EMITTER FOLLOWER



COMMON EMITTER



8.3.2.3 Phototransistors

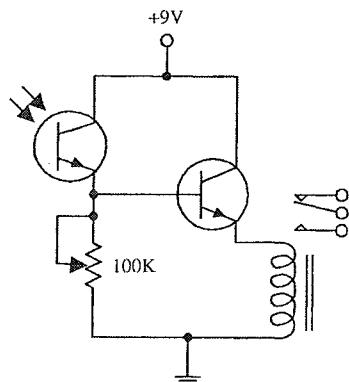
A common type pf phototransistor resembles a bipolar transistor with its base lead removed and replaced with a light-sensitive surface area. When this surface area is kept dark, the device is off, i.e., no current flows through the collector-to-emitter region. However, when the light-sensitive region is exposed to light, a small base current is generated that controls a much larger collector-to-emitter current. Another type of phototransistors is light-sensitive field-effect transistor, or photoFET. Unlike bipolar phototransistors, photoFETs use light to generate a gate voltage that is used to control a drain-source current. PhotoFETs are extremely sensitive to variations in light but are more fragile electrically than bipolar phototransistors.

Two-lead phototransistors may not be able to inject enough electrons into the base region to promote a desired collector-to-emitter current. For this reason, a three-lead phototransistor with a base lead may be used. The extra base lead can feed external current to help boost the number of electrons injected into the base region. In effect, the base current depends on both the light intensity and the supplied base current. In optoelectronic circuits, three-lead phototransistors are often used in place of two-lead devices, provided the base is left untouched. Similar to ordinary bipolar transistors, there exist Darlington phototransistors, or photodarlingtions, which are much more sensitive to light than ordinary phototransistors are, but they tend to have slower response time. Photodarlingtions may or may not come with a base lead.

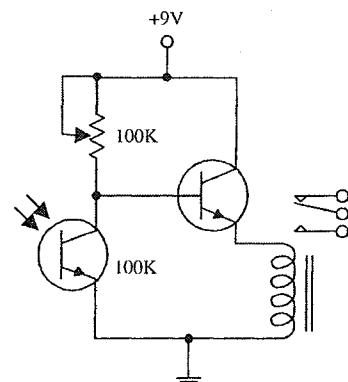
Similar to ordinary transistors, phototransistors can be used as either of two basic configurations: a) emitter-follower with current gain and no voltage gain; b) common-emitter with voltage gain, as shown in the above figure.

As an example, in circuits below, a phototransistor is used to control the base current supplied to a power-switching transistor that is used to supply current to a relay. In the light-activated circuit, when light comes in contact with the phototransistor, the phototransistor turns on, allowing current to pass from the supply into the base of the power-switching transistor. The power-switching transistor then turns on, and current flows through the relay, triggering it to switch states. In the dark-activated circuit, an opposite effect occurs. The 100KΩ potentiometers are used to adjust the sensitivity of both devices by controlling current flow through the phototransistor.

LIGHT ACTIVATED

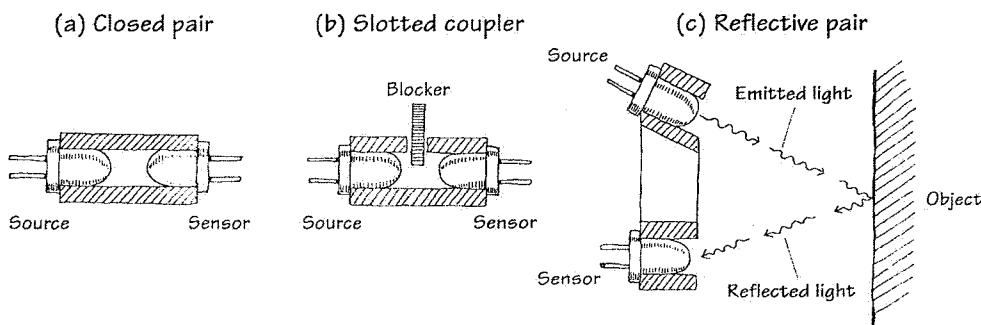


DARK ACTIVATED



8.3.3 Optoisolators

Optoisolators / optocouplers are devices that interconnect two circuits by means of an optical interface. For example, a typical optoisolator may consist of an LED and a phototransistor enclosed in a light-tight container. The LED portion of the optoisolator is connected to the source circuit, whereas the phototransistor portion is connected to the detector circuit. Whenever the LED is supplied current, it emits photons that are detected by the phototransistor. There are many other kinds of source-sensor combinations, such as LED-photodiode, and lamp-photoresistor pairs. In terms of applications, optoisolators are used frequently to provide electrical isolation between two separate circuits. This means that one circuit can be used to control another circuit without undesirable changes in voltage and current that might occur if the two circuits were connected electrically. Isolation couplers typically are enclosed in a dark container, with both source and sensor facing each other. In such an arrangement, the optoisolator is referred to as a *closed pair*. Besides being used for electrical isolation applications, closed pairs are also used for level conversions and solid-state relaying. A *slotted coupler / interrupter* is a device that contains an open slot between the source and sensor through which a blocker can be placed to interrupt light signals. These devices are frequently used for object detection, bounce-free switching, and vibration detection. A *reflective pair* is another kind of optoisolator configuration that uses a source to emit light and a sensor to detect that light once it has reflected off an object. Reflective pairs are used as object detectors, reflectance monitors, tachometers, and movement detectors.



8.3.4 Reflective Optosensors

The reflective optosensor is a device consisting of an emitter LED and a detector photodiode or phototransistor. These two elements are typically encased in a single plastic package, holding them in the ideal alignment for their application. The emitter LED generates a beam of light that is reflected off a surface and into the detector device. The detector and emitter are matched so that the peak sensitivity of the detector is at the same wavelength of the emissions of the emitter. The basic performance of a reflective optosensor, as well as some of the available commercial devices, is shown in figure below. In some of these products, the individual circular LED emitter and detector components are clearly distinguishable. In others, these components are rectangular and are formed into the overall device case packaging. As an example QRD1114 (or QRB1114) from Quality Technologies could be a good device for light projects. Most commercial reflective optosensors use infrared light. This is because the typical semiconductor junction used in LEDs is most efficient at an infrared wavelength. This would make it a bit difficult to ascertain that things are hooked up properly because the emitter's light is invisible. Also reflective sensors often include a filter in front of the detector device; these filters can easily be made to pass infrared wavelengths while trapping visible ones, thereby reducing interference from visible light.

There is a wide range of applications for reflective optosensors:

Object Detection: This is the case in most industrial applications. The reflective optosensor may be used to detect the presence of an object in the sensor's field of view. In addition to simply detecting the presence of the object, the data from a reflective optosensor may be used to indicate the object's distance from the sensor. These readings are dependent on the reflectivity of the object, among other things; i.e., a highly reflective object that is farther away may yield a signal as strong as a less reflective object that is closer. Of course, this only works if the object's surface colouring reflects at least some of the infrared light beam; objects that are completely black to infrared light cannot be detected.

Surface Feature Detection: Reflective optosensors are great for detecting features painted, taped, or otherwise marked onto the floor. Line following using a reflective sensor is a typical robot activity.

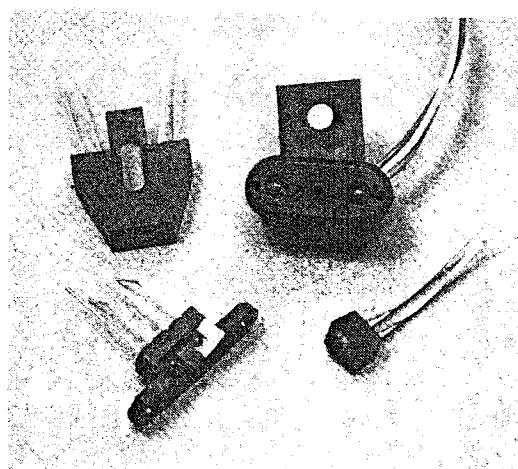
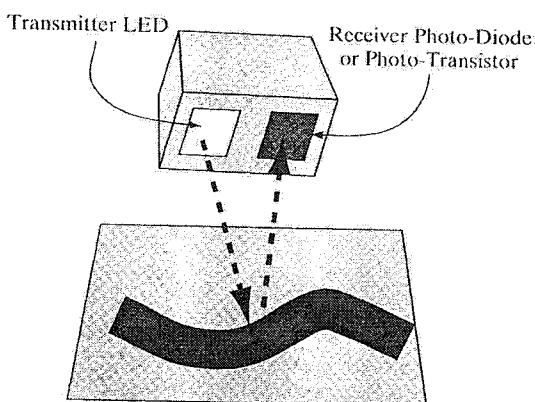
Wall Tracking: Related to the object-detection category, this application treats the wall as a continuous obstacle and uses the reflective sensor to indicate distance from the wall.

Rotational Shaft Encoding: Using a pie-shaped encoder wheel, the reflective optosensor can measure the rotation of a shaft (angular position and velocity).

Barcode Decoding: Reflective optosensors can be used to decode information from bar-code markers placed in a designated location.

It is often tricky to get reflective sensors to perform as desired. Like any sensor, they work best in highly controlled environments. Sensor calibration is necessary for nearly all sensor devices, but especially for reflectance situations in which readings may vary based on fluctuations in ambient lighting, distances between the sensor and the object being sensed, the reflectivity of the sensed object, and other factors.

Ambient light is always a problem unless the sensor and sensed object are both inside an opaque box. At a start, it is

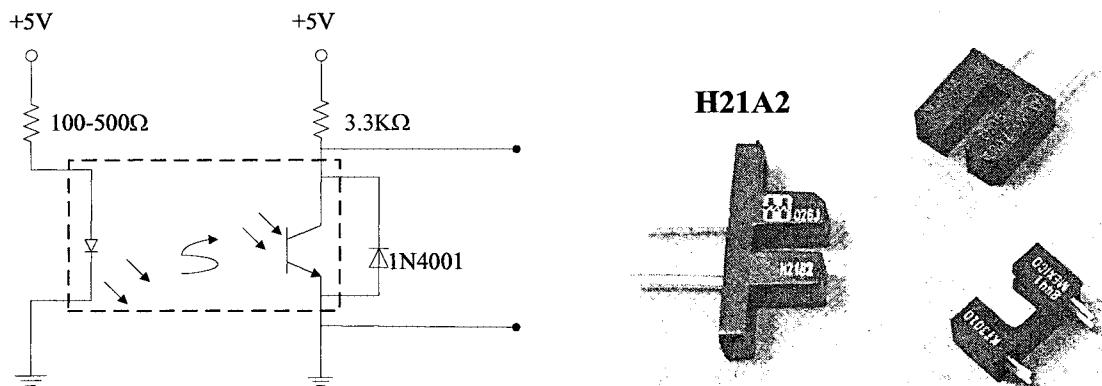


necessary to position the sensor so that ambient light does not directly reach the detector element. In many applications, it is necessary to provide some kind of sheath to further insulate the device from outside lighting. In some applications, active control of the sensor's own illumination source may be required. One effective solution is to switch the sensor's emitter light source on and off under program control. By taking two light level readings, one with the emitter on and one with the emitter off, the controller can subtract away the ambient light levels, yielding a much more accurate reflectance measurement.

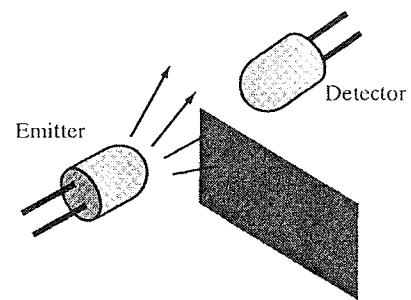
As an important note in using reflective (and other) optosensors, always remember to use a resistor (100-500 Ω) in the emitter LED circuit to limit its current. Otherwise, it will burn visibly, quite brightly indeed, for few seconds before expiring!

8.3.5 Break-Beam Optosensors

Break-beam devices consist of the light-emitting component aimed at a light-detecting component. When an opaque object comes between the emitter and detector, the beam of light is occluded, and the output of the detector changes. Industrial break-beam optosensors typically consist of an infrared emitter and infrared detector housed in a U-shaped plastic assembly. The gap between the vertical beams of the "U" is typically less than a half of an inch. These devices are sometimes called slotted switches. A good example is H21A2 optosensor from Fairchild, shown in figure below. Using these sensors is straightforward; only two resistors are needed for both the transmitter and receiver (at the collector) ends. Using a diode around the collector-emitter of the receiver phototransistor would also protect it against current surge. The circuit is shown in figure below.



For sensing objects between larger gaps, you can simply use discrete emitters and detectors. Any pair of compatible emitter-detector devices may be used: flashlight bulbs and photocells, red LEDs and visible-light-sensitive phototransistors, or infrared emitters and detectors. When covering larger distances, it may be necessary to use higher-powered or multiple emitters, lenses, or light shields to obtain satisfactory performance. Also, the light source can be switched on and off to determine the validity of the detector reading and minimize errors caused by ambient light. Object detection (between the emitter and detector) is the main application of break-beam optosensors.



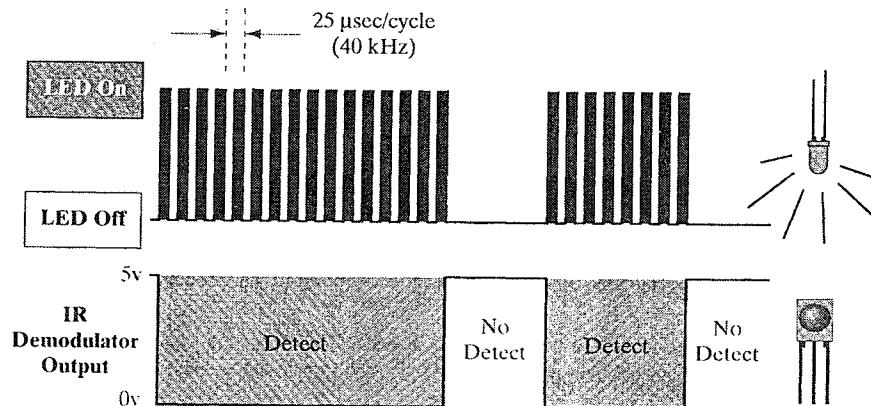
8.3.6 Infrared Sensing

Infrared (IR) transmitters and detectors offer advantages over visible light counterparts. They are quicker to respond to light changes, so they are well suited to applications such as motor shaft encoding. They are also more sensitive and less vulnerable to ambient light. Nevertheless, to use the IR sensing even more efficiently, a technique, called *modulation*, is applied that is basically rapidly turning on and off the IR source, so that it can be easily picked up from varying background illumination, even if the actual amount of modulated light is very small. In this way, great insensitivity to background ambient lighting can be achieved. This is how television remote controls work: IR LEDs in the remote

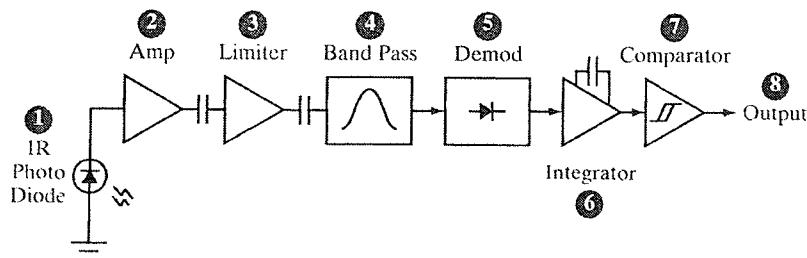
control transmit rapid flashes of light, which are decoded by the receiver device in the television. The advent of widespread use of IR remote control has made the demodulating circuits quite inexpensive and easy to use. What used to require several integrated circuits is now available in a single package and at a low cost of few dollars. One popular example is Sharp's GP1UD IR receiver modules that detect and process the IT modulated signal in different modulation frequencies.

8.3.6.1 Modulation and Demodulation

The basic principle of IR modulation is that, by flashing a light source at a particular frequency, the flashes of light at that same frequency can be detected (demodulation) even if they are very weak with respect to overall lighting conditions. Figure below illustrates this process. The upper graph indicates an IR LED being turned on in two successive bursts. Each burst consists of a number of very rapid on-off pulses of light (modulation). The lower graph shows the output from the IR detector. During the rapid on-off bursts, the demodulator indicates "detection." In between the bursts, the demodulator sees no IR activity and indicates "no detection." In other words, when the detector "sees" the flashes of light, it indicates "true." There are a few details worth considering. Most important, the demodulator is tuned to a specific frequency of light flashes. Commercial IR demodulators can be obtained for a variety of frequencies, ranging from about 32 kHz to 56.8 kHz, with 40 kHz being the most common one. All of these frequencies are high enough to avoid interference effects from common indoor lighting sources, such as fluorescent lights. Assuming that the demodulator is tuned to 40 kHz (40,000 flashes per second), as indicated in the figure, this corresponds to a flash period of 25 μ sec (0.000025 seconds). This means that the sum of the "on time" and "off time" for each flash must be 25 μ sec. (It happens that it is not critical that the on and off time be exactly equal; if each flash consisted of a 10 μ sec on pulse and a 15 μ sec off interval, that would work just fine.) Another detail in figure below is the voltage level output of the IR demodulator. Notice that it outputs 5V when there is no detection and 0V during detection. This response, i.e., negative true logic, is characteristic of most commercial IR demodulator devices. What is shown in the figure is an idealized representation of the demodulation process. Indeed, the diagram obscures a practical effect: it suggests that the IR demodulator detects the IR light bursts immediately as they occur. This is not actually the case in practice. The demodulator must receive several cycles of light, usually 5 to 10, at the proper frequency before indicating detection.

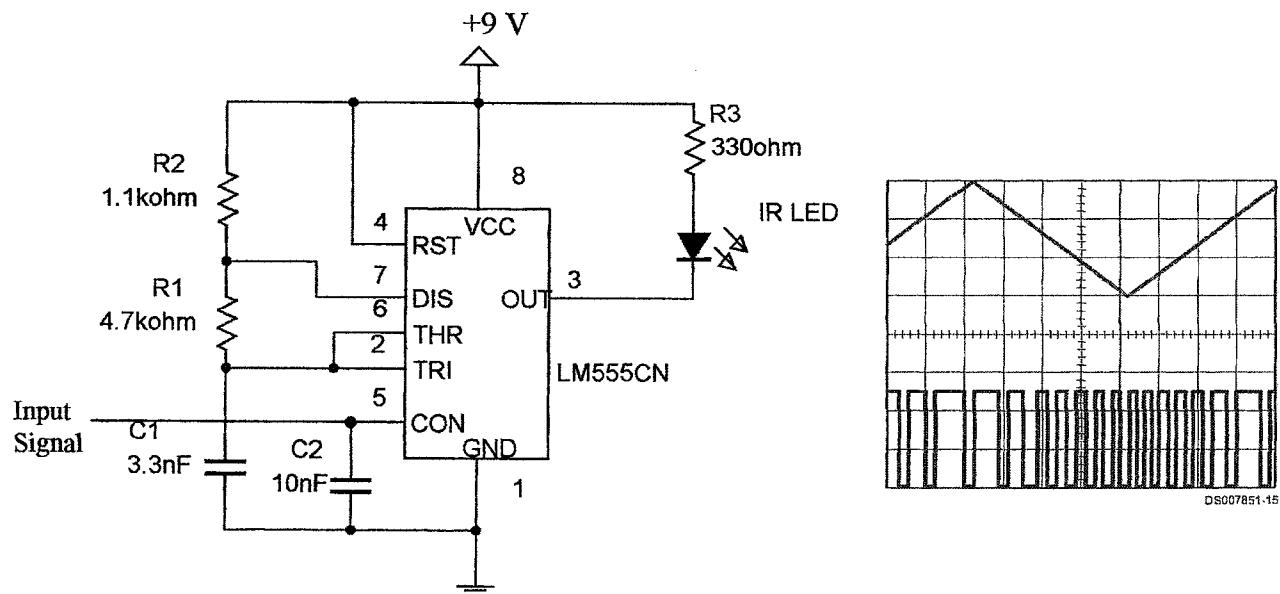


It would be beneficial to take a brief look at inside the commercial IR receiver modules. The architecture of most of the commercial IR detectors are similar to what is shown in figure below. In this block diagram, the information flow is from



left to right. First, the IR photodiode (1) receives infrared signals that are amplified (2) and limited (3). This normalized signal is then run into a band-pass filter (4), which allows only a specific frequency of signal (e.g., 40 kHz) to pass through. Actually, this is an idealization of the band-pass filter's performance; real band-pass filters have a centre frequency for which signals pass through best, and then a sloping curve that describes how much off-center frequencies are attenuated. The Gaussian-shaped illustration in the band-pass filter diagram suggests this attenuation curve. Ultimately, this means that even if an IR demodulator is tuned to a specific centre frequency, it will detect other frequencies, with a desensitized response as the frequency moves farther away from the centre point. After the band-pass, the signal contains predominantly a wave at the center frequency of the band-pass or no signal at all. This wave is then rectified by the demodulator (5). The demodulator is just a diode that turns the wave into a series of positive pulses. These pulses are fed into the integrator (6), which has a capacitor to store a brief history of incoming pulses. The more regular the pulse stream (corresponding to a steady signal at the center frequency of band-pass), the larger the output of the integrator. The integrator's output is given to the comparator (7), which measures it against an internal threshold. When the integrator's signal is above a threshold, the comparator signals "true" at the output (8); otherwise it signals "false." As indicated in the symbol for the comparator, it comparator functions with hysteresis, i.e., it has a bit of a latching function when interpreting the integrator's output. In a function with hysteresis, there are two separate thresholds: one for a rising input and one for a falling input. When the input is rising, it must reach above a higher threshold to be registered as high; when it is falling, it must fall below a lower threshold to be registered as low. By putting hysteresis in the comparator, small fluctuations of the input signal, which might happen to be about a single threshold, are ignored.

On the transmission side, one has to create the modulation frequency and have control over changing it to be able to apply ON/OFF conditions for the receiver. A 555-timer can be used for this purpose in astable operation where the input signal, to be modulated is applied to the control voltage terminal (pin 5). Figure below shows the circuit that has been set for a 40KHz modulation frequency. The pulse position varies with the modulating signal, since the threshold voltage and hence the time delay is varied by the input signal. This operation is sometimes called *pulse position modulation*, as will be discussed in the next section. Figure below also shows the modulated waveforms generated for a triangle signal.



8.3.6.2 IR Communication

What has been discussed so far was about detecting ON/OFF situations by using IR pairs of transmitter/detector in applications such shaft encoding, object detection, wall tracking, etc. IR signals are also used for signal/data communication. For this particular application, a variety of pulse modulation techniques has been implemented. Figure on the right represents some of the most popular methods of pulse modulation for IR signal/data transmission. A brief description of each method is given in here:

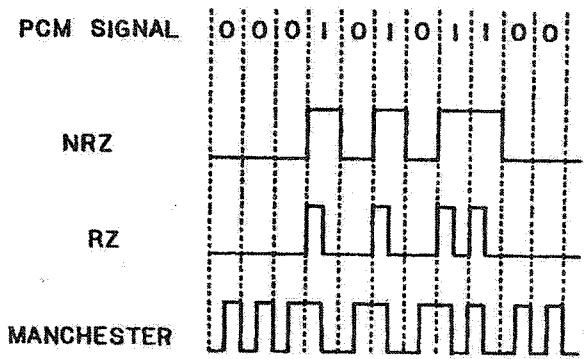
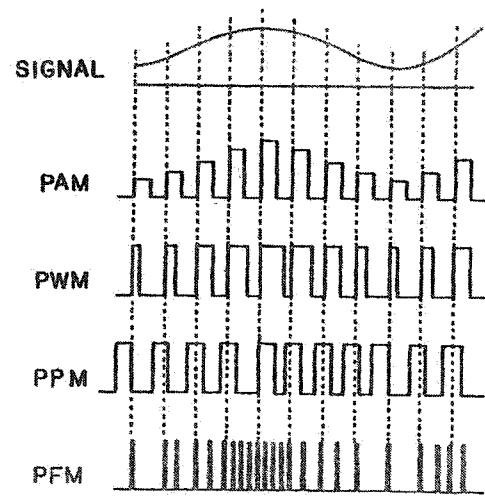
Pulse-Amplitude Modulation (PAM): In this modulation scheme the amplitude of the pulses is directly proportional to the amplitude of the modulating signal.

Pulse-Width Modulation (PWM): This method is also known as Pulse-Duration Modulation (PDM). The duration of individual pulses within a pulse train is made proportional to the amplitude of the modulating signal.

Pulse-Position Modulation (PPM): Here the amplitude of the input signal controls the relative position of individual pulses in a pulse stream. Unlike PAM and PWM, all the pulses in PPM have precisely the same amplitude and duration. This means the PPM receiver can be optimized for the processing of identically shaped pulses. This gives a higher degree of noise immunity than provided by PWM and especially PAM. The detection of PPM pulses by a receiver requires synchronization with the transmitter. This implies the necessity to transmit a clock signal along with the data or on a separate channel.

Pulse-Frequency Modulation (PFM): This modulation method resembles that used in FM radio in that the transmitter emits a steady train of pulses called the *carrier*. Information is superimposed on the carrier by altering the carrier's frequency. Detection of PFM is straightforward since, unlike PPM, no clock signal from the transmitter is required. Since the pulses have uniform duration and intensity, PFM offers many of the same advantages of PPM. PFM is particularly well suited for audio bandwidth lightwave links.

Digital Pulse-Code Modulation (PCM): All the above-mentioned pulse modulation methods require that such critical pulse parameters as amplitude, duration, or position be altered in response to an analog signal. PCM is a true digital modulation method, which involves the transformation of an analog signal into its binary equivalent. Voice, for example, is sampled at a sufficiently fast rate, and the amplitude at each sample point is converted into a binary word by an analog-to-digital converter. The binary word is then transmitted in serial form a bit at a time. In PPM, PFM, and PCM, the shape of each pulse is identical. PCM, however, offers two significant advantages over PPM and PFM. First, the pulses remain fixed in time, and a pulse is either absent or present. This greatly enhances the *noise* immunity of the signal. Second, the predictable spacing between pulses greatly simplifies time division multiplexing. The major disadvantages of PCM are system complexity and bandwidth limitations. Several pulse formats are used to implement PCM, the most important being return-to-zero (RZ) and nonreturn-to-zero (NRZ). A binary signal is either high (1) or low (0). A pulse, therefore, represents binary 1 and the absence of a pulse denotes binary 0. Figure on the right illustrates some of the important pulse formats. In the RZ mode, all bits return to zero before the next bit is transmitted. In the NRZ mode, a 1 remains high and a 0 remains low for the duration of the bit transmission interval. This means two consecutive 1 bits merge into a pulse having twice the duration of an individual bit position. The RZ format is more efficient than the NRZ format since only half the time is required to transmit a single bit position. Both the NRZ and RZ modes require synchronized clocking at both the transmitter and receiver. The Manchester format, also shown in the figure, is a modification of the RZ format in which half of every bit position is denoted by a pulse. If the bit position

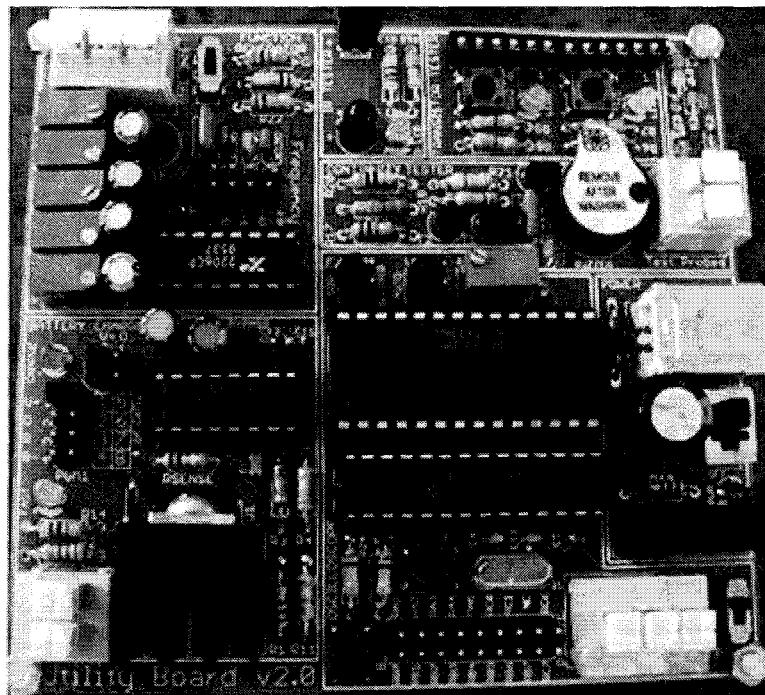
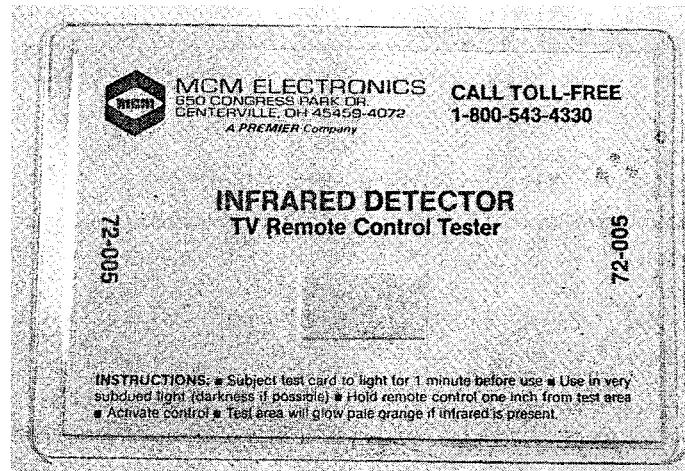


contains a 0, the first half of the pulse is low and the second half is high. If it contains a 1, the first half is high and the second half is low. Hence, Manchester coding eliminates the need for a separate clock channel. Like the NRZ format, however, it requires twice the time space of the RZ format.

8.3.6.3 “Seeing” IR

Working with infrared devices can be frustrating because infrared light is indeed invisible, making it hard to tell if a given infrared LED is emitting light. The CCD “retinas” of consumer video cameras are sensitive to infrared emissions, however. When looking at an IR LED through a camera with a video display of the image, if it appears lit up, then it is emitting infrared light. This is most easily demonstrated with household TV/VCR remotes; their IR LEDs are driven with brief but intense current bursts, making their light output quite powerful. Another method is to purchase an *infrared detector card*, shown on the right, from an electronics supplier like *Radio Shack* (catalog number 276-099) or *MCM Electronics* (catalog number 72-003 and 72-005). These credit-card-size devices contain a phosphorescent panel that glows visibly under infrared illumination. Aim the infrared LED at the sensitive portion of the card and a faint orange glow will be visible when the IR LED is turned on. It is necessary to provide subdued ambient light to see the glow, but it is quite definitive once you know what it looks like.

As mentioned in Chapter 5, The AER201 Utility Board that is included in the Project Kit also contains a module that enables the user to connect an infrared emitter and determine its functionality through an indicator LED. Alternatively, the user may position the Utility Board near an IR emitter in an external circuit, but in either case, the detector on the Utility Board must be within 10 mm distance from the emitter.



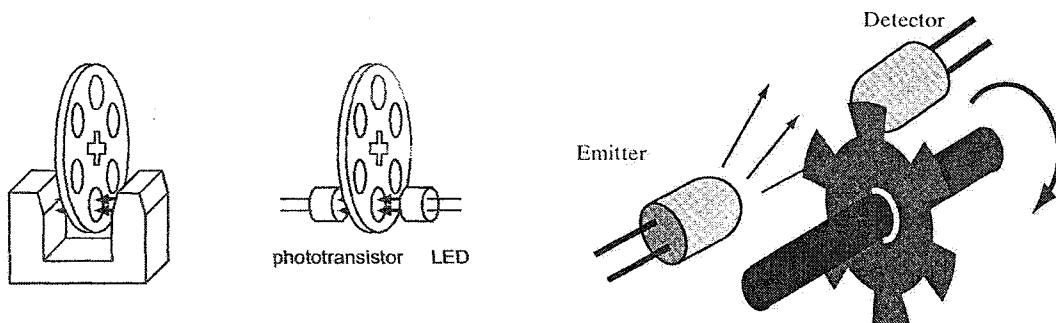
8.3.7 Applications

8.3.7.1 Shaft Encoding

Shaft encoders are suitable tools for vehicle navigation. When placed on a wheel or in the wheel's accompanying gearbox, encoders can quite accurately measure the distance and speed the wheel has traveled. This provides the mobile platform with a number of abilities, including measuring distances, driving straight, and turning accurately. An everyday example of shaft encoding is the speedometer on an automobile, which reports how fast the wheels are running, and its odometer, which keeps track of the number of total wheel rotations. As another example, consider the task of moving a robot on a straight line. If both sides of the robot are driven at the same rate, the robot will move in a straight line. For a robot with a differential drive system, the easiest way to do this is to place a shaft encoder into each of the drive trains and monitor the distance each wheel has traveled. Whenever one wheel gets ahead of the other, it is slowed down. The problem of driving straight can be solved with a very simple algorithm. Whenever the left side of the robot is ahead, the right wheel is driven faster, and the left is slowed down. When the right side of the robot is ahead, the opposite action is taken. The robot constantly corrects for small deviations allowing it to drive in a straight line. As another task, one easy way to make 90-degree turns, you can calibrate the amount that each wheel should travel, and the associated number registered by the break-beam sensor, for the desired turn. This number should remain fairly constant, assuming that there is no major wheel slippage. It should be noted, however, that this method of navigation is prone to some errors. The feedback provided by the encoders comes not from the environment, but rather from within the robot. You only get feedback regarding the amount that each wheel has rotated, not the actual distance that the wheel has traveled on the ground. Hence, slippage of the wheels on the floor is not accounted for, and can lead to measurement errors, especially when turning. In order to correct for these errors, additional feedback mechanisms might be needed in conjunction with the shaft encoders.

A basic application of shaft encoding is illustrated in figure below, which uses the break-beam sensors. An opaque circular disk with notches cut into its circumference is mounted on the shaft to be monitored. The disk is positioned so that as it rotates, the notches chop the light beam from the emitter to the detector. Software or hardware circuit is connected to the detector signal to decode and count the light pulses. The trick to making sense of the data from a shaft encoder is to execute a routine that repetitiously checks the sensor value, looking for rising or falling edges that indicate the counter wheel has turned one "click." The more often this routine checks the sensor value, the better it can keep up with the encoder wheel's transition. If the encoder wheel turns faster than the routine checks the sensor state, it will start missing transitions and lose track of the shaft's rotation. A simple way to look for transitions is to threshold the sensor data at some midrange point. By applying this hysteresis property, the data interpretation of the encoder will be robust against fluctuations of the sensor.

Measuring the velocity can be done simply by subtracting differences in the position readings after an interval of time has elapsed. Velocity measurement can be useful for a variety of purposes. A robot that has an unpowered trailer wheel with a shaft encoder can easily tell whether it is moving by looking at encoder activity on the trailer wheel. If the robot is moving, the trailer wheel will be dragged along and will have a nonzero velocity. If the robot is stuck, regardless of whether its main drive wheels are turning, the trailer wheel will be still. Further, as mentioned before, velocity information can be combined with position information to perform tasks like causing a robot to drive in the straight line or rotate a certain number of degrees.



It is also possible to build shaft encoders by using a reflective optosensor to detect black and white markings on an encoder wheel. Wheels can be used with any of the reflective optosensor devices, as long as the beam of light they generate is small enough to fit within the black and white pie-shaped markings. You can build your own wheel using a suitably stiff disk of even paper or cardboard that has pie sections of black and white, mount it on an axle, and aim the reflective sensor at it. Alternatively, the black markings can be made with any pen known to be "black" to an IR sensor, such as a blue Sharpie marker).

8.3.7.2 Line Following

Line following is usually accomplished by mounting one or more reflective optosensors on the underside of a robot. By measuring the intensity of the reflection, these sensors can determine whether they are over a light or dark area. By adding a little "intelligence," the robot can be made to follow lines. The number of sensors and the configuration used depends on the robot and application, but the method is almost always the same. The sensors detect when the robot begins straying from the line, and the controller issues orders to correct for the error. Developing this algorithm begins with constructing a chart of all possible combinations of sensor inputs and the appropriate action for each. This information can then easily be coded into the controller's code. As an example, figure below shows a table for constructing a program for a robot with three reflective optosensors arranged in a line across the robot with the left and right ones spaced wider than the line. Whenever the robot begins to drift, one of the outer sensors detects the line and tells the robot to correct itself. If the robot continues to stray, the middle sensor loses the line and tells it to turn sharper. It is interesting to note that in theory, some of the states cannot occur, but in practice, erroneous sensor readings and unexpected situations can cause them to arise. It is usually wise to assign best-guess actions to these states to make sure that the robot always has something to do. As a matter of fact, control systems often fail when something unexpected happens, and the available sensor information is not able to account for the situation. Most of the time, when a robot gets lost, it will wind up stuck on some obstacle. If none of the sensors register the collision, the robot will not even know that anything is wrong. It will continue trying to drive, oblivious to the fact that it is not getting anywhere. If the robot does not reach its goal after a certain amount of time, though, it can usually assume that a problem has occurred along the way. When an action times out, the robot only knows that something has gone wrong, but not what it is. Regardless, the robot can often act to increase its chances of recovering. In many cases, backing up a little or thrashing around may be sufficient to free the robot from an obstacle so that it can continue on its course. In any case, detecting that something has gone wrong can considerably increase the robots ability to make the best of a bad situation.

Left	Middle	Right	Action	Left	Middle	Right	Action
○	○	○	n/a	●	○	○	← LEFT! ← LEFT!
○	○	●	→ RIGHT! → RIGHT!	●	○	●	n/a
○	●	○	↑ straight	●	●	○	← left
○	●	●	→ right	●	●	●	n/a

Sensor Inputs
 ○ on ● off

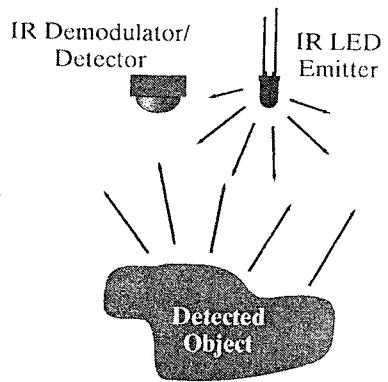
8.3.7.3 Proximity Sensing

Using the simple modulated output of an IR LED and an IR demodulator, it is possible to build an effective proximity sensor. With a proximity sensor, the light from the IR emitter is reflected back into detector by a nearby object, indicating whether an object is present. The idea is illustrated in figure below. An LED emitter and detector are pointed in the same direction so that when an object enters the proximity of the emitter-detector pair, light from the emitter is reflected off the object and into the detector. This kind of simple true-false proximity sensing is an ideal application for

modulated/demodulated IR sensing. Compared to simple reflected visible light magnitude sensing, modulated light is far less susceptible to environmental variables, such as amount of ambient light and reflectivity of different objects.

When constructing a proximity sensor, it is necessary to shield the light from the emitter from directly entering the detector. Especially because most IR detectors are extremely sensitive, with auto-gain circuits that amplify minute levels of light, shielding can be a real issue. If light from the emitter can bleed directly into the detector, the sensor will be rendered useless. One of the simplest and most effective ways to shield the emitter LED is with black heat-shrink tubing. The tubing can be placed around the base of the LED emitter and extend straight outward. After shrinking the tubing, it can be cut to length with a pair of scissors, providing an easy way of tuning the amount of light output.

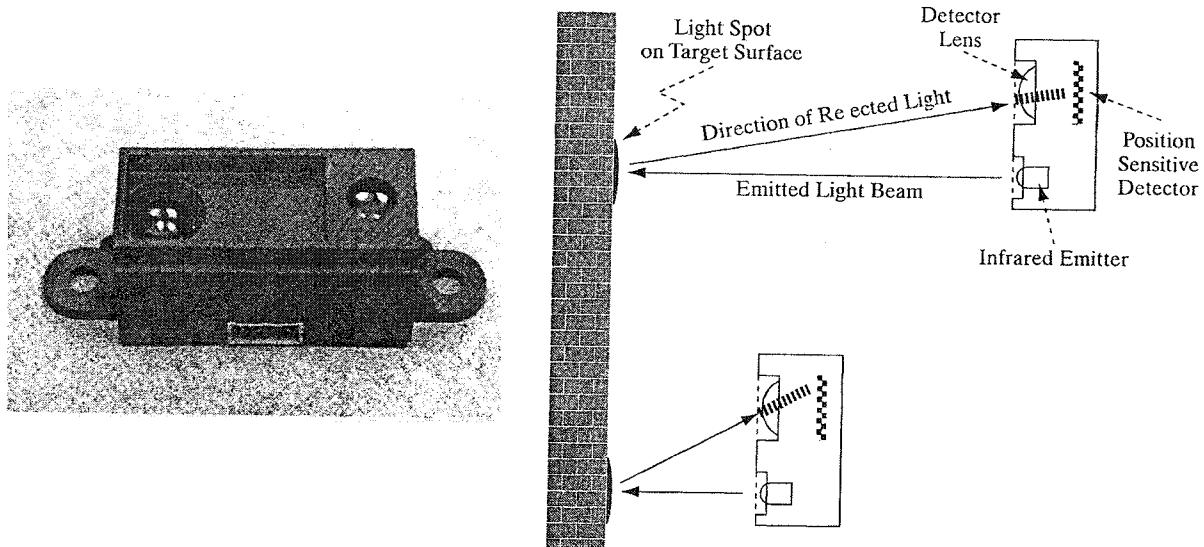
The commercial IR detector modules (such as GP1UD series) produce a digital output than can be simply connected to the microcontroller. The detectors require a 5-volt power supply. The detector's digital output is then wired to one of the microcontroller's input pins (analog or digital).



8.3.7.4 Distance Sensing

The proximity sensing discussed above provides only binary data, i.e., whether there is an object in view of the sensor or there is not. In some applications, it is required to know how far away the object is from the machine (sensor). For this purpose, more sophisticated sensors are required, such as Sharp's GP2D12 IR range finder, with analog output and GP2D02 with binary output. These devices combine a modulated IR emitter with a detector assembly that includes a focusing lens and a "position-sensitive detector." Figure below shows how these sensors work. An IR emitter LED projects a spot of modulated light onto the target surface. The light from this spot is focused by a detector lens onto a special linear position-sensitive detector element. Depending on the distance from the sensor unit to the target surface, the angle of incoming reflected light will change, and the spot of light will be projected to a different point along the position sensor. Thus, the location of the spot on the sensor corresponds to the distance to the target. The device's output is the position of the reflected spot (as an analog voltage for GP2D12 and a byte-value for GP2D02). As long as the target is sufficiently reflective so that the spot is registered by the detector, the resultant reading is independent of the target's actual reflectivity.

Connecting GP2D12 to the microcontroller is straightforward, but signal A/D conversion is required, which can be implemented by the microcontroller's A/D module. If the corresponding digital value of the analog signal is set to vary smoothly from 0 to 255 as the distance decreases, tests have shown that the following function



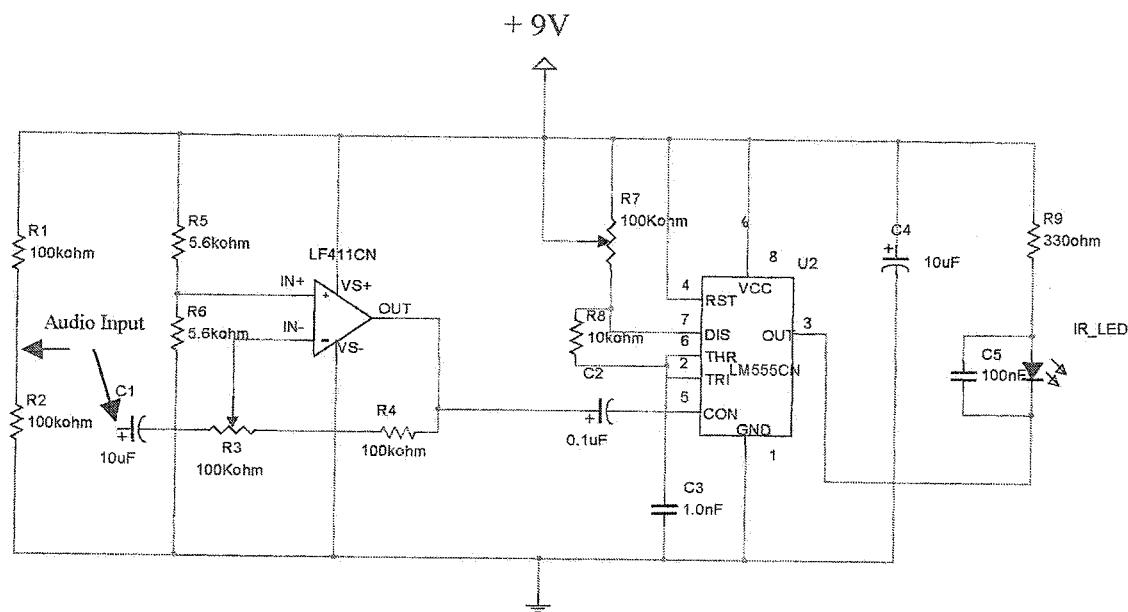
$$f(n) = -3.16'' + \frac{950''}{8.58 + n}$$

gives an accurate estimate of the distance as a function of the integer value n , and that the reading is fairly independent of lighting, object colour, or battery power level. Nonetheless, the numeric parameters may have to be recalibrated for each individual sensor if an accurate measurement is desired.

The Model GP2D02, however, uses a digital interface, and does not need any signal conversion when connected to the microcontroller. Nonetheless, there are some tricks in connecting this model to the controller that must be carefully noticed. The sensor has one input line and one output line. The input line is used to tell the device to take a reading and synchronize the clocking of data on its output line. In the path of the connection from the microcontroller to the input pin of the sensor, it is necessary to position a diode (such as 1N914, typically included with the sensor package) with the cathode pin connected to the microcontroller and the anode pin connected to the sensor input. This is because the GP2D02 device employs an unusual “open drain” input circuit, meaning that one must ever drive a +5V voltage to the input pin. The diode allows the microcontroller to pull the sensor’s input low (to zero volts); when the microcontroller pin is high, the diode blocks the voltage from driving the sensor input. There is a pull-up resistor internal to the sensor on its input line to assert the +5V level. Figure above shows the timing chart for operating this sensor. The V_{in} signal both enables the sensor and acts as a clock to serially shift the sensor reading out the output signal V_{out} . The process of taking a reading is as follows. First, V_{in} must be taken low for at least 70 milliseconds to allow the sensor to power up and perform the actual measurement. Then, the V_{in} line can begin clocking rapidly to make the result data available on the V_{out} line. After one initial set-up clock pulse, the data, beginning with the most significant bit of the result byte, are available on each rising edge of the V_{in} clock. After clocking out the last data bit, the V_{in} line should be left high to allow the sensor to enter a power-saving mode in which it draw a small amount of current, about 3 μ A. (Note that when taking a sensor reading, the GP2D02 consumes about 22mA.) After 1.5ms of idle time, the sensor automatically switches to power-saving mode.

8.3.7.5 Voice Communication

This is a simple but interesting example of using IR for signal (voice) communication. Figure below shows a circuit for straightforward PFM transmitter designed around the popular 555 timer IC. In operation, the 555 oscillates at a centre frequency determined by the time constant of R_5 and C_2 . Typically the centre frequency is 40 kHz. Low-level audio signals appearing at the microphone are amplified by the LF411 op amp and passed into the modulation input of the 555 where they alter the chip’s oscillation frequency. The pulse-frequency modulated signal appears at pin 3 where it is used to drive an infrared LED. For best results use an electret microphone. A crystal microphone may also be used. Better quantity op amps can be used in place of the 411 (such as LF353 and LM358) to give a lower transmitted noise level. The

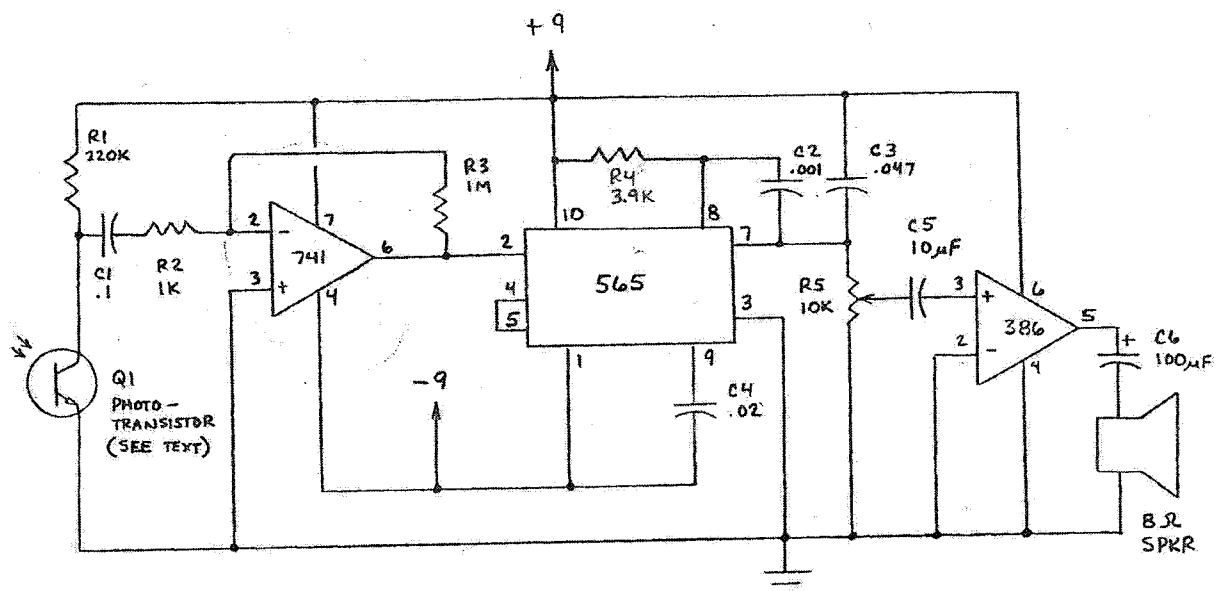


LED can be any infrared emitter. For high power operation, select one of the super-bright IR LEDs. The peak pulse current supplied to the LED by the 555 is about 50-60 mA. The pulse width is about 5 microseconds. For raising the pulse current a bipolar or FET transistor can be used to switch the diode.

Figure below shows the circuit for a PFM receiver designed around a 565 Phase-Locked Loop (PLL) IC. The PLL is tuned via R4 and C4 to the approximate centre frequency of the transmitter. Incoming optical signals are detected by the phototransistor (Q1). The resulting photocurrent is converted to a voltage by load resistor R1 and coupled via C1 into the 741 op amp. The signal is then amplified 1000 times (R3/R2) and passed into one of the phase comparator inputs of the PLL. The phase comparator generates an error voltage proportional to the difference between the PLL's on-chip VCO centre frequency and the instantaneous transmitter frequency. The error voltage is fed back to the VCO in a feedback loop, which causes the VCO to track the transmitter frequency. The error voltage represents the demodulated analog of the transmitter signal, so it is tapped via pin 7 for power amplification by the 386 audio op amp. R5 controls the gain of the system by controlling the amount of signal that reaches the 386.

Several modifications can be made to the basic receiver circuit. To reduce the effect of sunlight upon the detector, Q1 can be replaced by a silicon photodiode. The diode should be connected in the reverse direction. The gain of the 741 preamplifier can be altered by changing the ratio of R3/R2. And the center frequency of the PLL's VCO can be changed via R4 and C4. For best results, R4 should be at or near 4 K Ω .

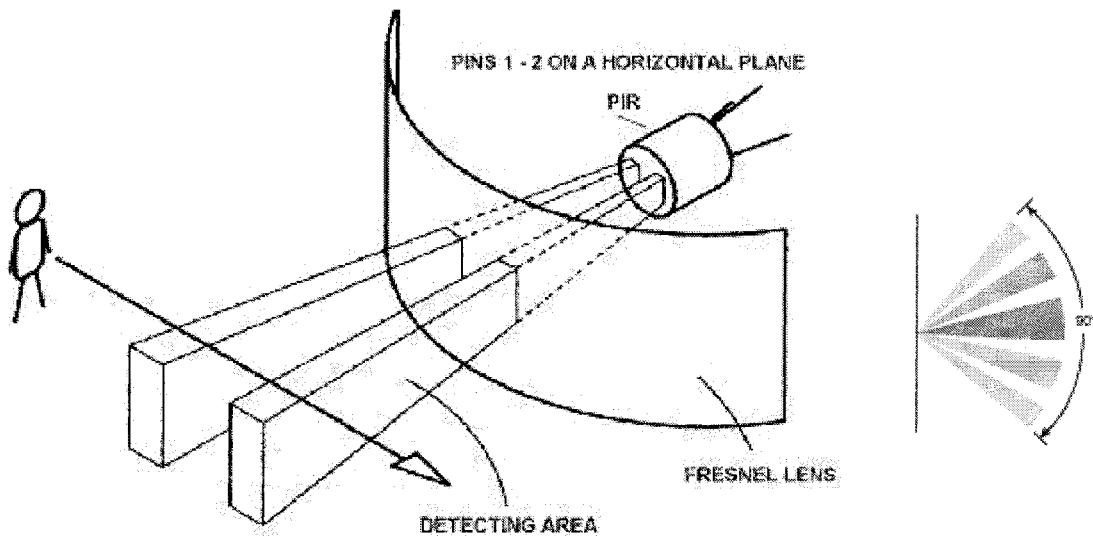
Before applying power to the transmitter and receiver, carefully inspect both circuits for possible wiring errors or omissions. To avoid severe interference when using a phototransistor in the receiver, place an opaque hollow tube over Q1 to keep artificial light from striking Q1's active region. A photodiode may not require this kind of protection. For initial tests, disconnect the transmitter's microphone and connect the audio output of a transistor radio to the circuit via a one-microfarad capacitor. The negative lead of the capacitor should be connected to R1. The radio output should be connected to the positive lead of the capacitor and the transmitter's ground connection. For the best results, use a radio with an earphone jack and make your connections with the help of clip leads soldered to an appropriate plug. Turn the radio on at low volume and select a station that gives clear reception. Then apply power to both the transmitter and receiver while pointing the transmitter's LED at the receiver's detector. At this point you should hear noise, oscillation, ideally, the sound of the radio from the receiver's speaker. In any event, carefully adjust the setting of R5 in the transmitter in an effort to match the centre frequency of the VCO in the receiver's PLL, in this case about 40 kHz. While adjusting R5 you will probably hear loud noises and whistles interspersed with very clear sounds of the radio. Select the best quality transmission point with R5 and then adjust the gain of the receiver (R5) for comfortable listening. You can then experiment with the setting of the transmitter's R1 to find the optimum gain of the transmitter's preamplifier. When the transmitter and receiver are properly adjusted for good transmission of the radio signal, you can experiment with the units. The most interesting demonstration is to gradually point the LED away from the receiver's detector while listening to the received signal. The amplitude of the signal will remain constant until a threshold point is reached where a sudden noise level appears. As the LED is moved still more, the noise will stop and the receiver will be silent.



8.3.7.6 Motion Sensing

All objects emit some level of infrared light, with hotter objects emitting more IR light than cooler objects. By utilizing this simple property of all materials, it is possible to sense motion in the proximity of a sensor. Note that this is not the same as proximity sensing, as that simply involves sensing the presence of an object and has nothing to do with the movement of the object.

Sensors used to detect motion by detecting IR levels are called PIR sensors (Passive IR). It consists of a lens and two or more different pins that can detect levels of IR. The pins are positioned such that they are pointing at slightly different angles with respect to each other, as illustrated in figure below. When no motion is present, the two pins detect equal amounts of IR from the environment. When an object moves within the range of the sensor, one pin will detect more IR than another, which results in a difference of IR levels between the pins. This difference is what the PIR detects, and it is what allows the sensor to detect motion. By having different number of pins arranged in different orientations, the range at which the sensor can detect motion can change. Many pins result in a wider view for the PIR sensor.



8.4 Inductive Sensors

8.4.1 Fundamentals

Inductive sensors are mostly used for proximity sensing when the presence or absence of an object must be detected with an electronic non-contact sensor. They are also used for motion detection and motion control applications. The main phenomenon behind an inductive sensor is *Faraday's law of induction* in a coil. Faraday's law of induction specifies that the induced voltage, or electromotive force (emf), is equal to the rate at which the magnetic flux through the circuit changes. If varying magnetic flux is applied to a coil, then electromotive force appears at every turn of the coil. This electromotive force plays as a resistance against variation of the magnetic flux. If the coil is wound in such a manner that each turn has the same area of cross section, the flux through each turn will be the same. The induced voltage equation can then be presented as follows:

$$V = -N \frac{d\phi}{dt}$$

where N is the number of turns and $\phi = B \cdot A$, where B is the magnetic field and A is the area of the coil. It follows that the voltage output can be changed by changing the flux enclosed by the circuit. This can be done by changing the amplitude of the magnetic field B or area of the coil A . The magnetic field B , produced by a current I in any wire, is proportional to the current and geometry of the coil, so that at any radius of r around wire the strength of the magnetic field is calculated as:

$$B = \frac{\mu_0 I}{2\pi r}$$

where μ_0 represents the effective permeability of the wire. Considering N turns of wires, as a coil, and being interested in the magnetic field just next to the coil surface would result in the following geometrical relationship:

$$\text{total length of the coil} = l = N \cdot (2\pi r)$$

The induced voltage can then be restated as:

$$V = -N \cdot A \cdot \mu_0 \cdot \frac{N}{l} \cdot \frac{dI}{dt} = -\left(\frac{\mu_0 N^2 A}{l}\right) \frac{dI}{dt} = -L \frac{dI}{dt}$$

L is the coil inductance in units of Henrys. Hence, if we have a simple coil, and try to suddenly cause a current to flow through it, the initial current flow causes a magnetic field to begin to form in the coil. Since this magnetic field is increasing, there is a change in the flux through the coil, and an opposing voltage appears. Eventually, the current rises to its limiting value, the magnetic field is stable, and the opposing voltage dies away.

If we assume that the current and voltage are both oscillating quantities:

$$V = V_0 e^{i\omega t}$$

$$I = I_0 e^{i\omega t}$$

then, we have

$$V_0 = -i\omega L I_0$$

If we recognize Ohm's law here ($V = IR$), then the effective resistance of the inductor is " R " = $-(i\omega L)$. The i implies that there is a shift in phase between voltage and current, and the ω implies that the effective resistance increases with frequency. Using this analogy, we can easily see how to build a large array of inductance measuring circuits, which are exactly like the resistance measuring circuits in voltage divider and/or bridge circuits.

To have a sense of the value of L , assume we have a coil with a 1cm diameter, a 1cm length, with 1000 loops of wire. Then,

$$L = \frac{(4\pi \times 10^{-7} \text{ H/m})(1000)^2 (\pi (0.5 \times 10^{-2} \text{ m})^2)}{(1 \times 10^{-2} \text{ m})} = 9.9 \times 10^{-3} \text{ H}$$

Now, if we want to measure the effective resistance of this coil, we will need to apply an oscillating voltage at some frequency. Normally, if we use a frequency of 1 kHz, the effective resistance of this device would only be about 60 ohms. Clearly, this is a small resistance, and measuring it to a high degree of accuracy will be difficult. We may go to a higher frequency to improve this situation. What is really preferred is an inductor with a higher inductance. In an inductor, we can fill the coil with a material, which has a higher magnetic permeability. For example, iron has a relative permeability of about 300. So, we generally see coils filled with metallic materials like iron.

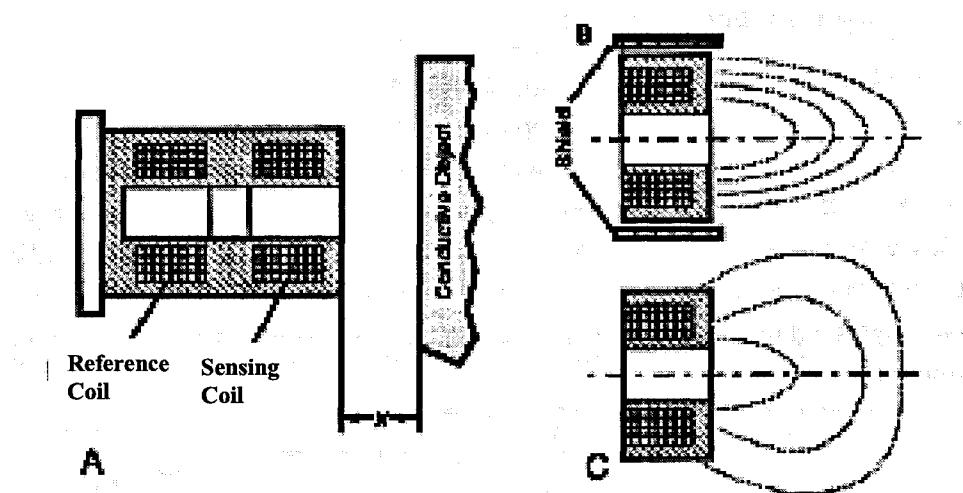
If we rewrite the formulation of the inductance of coil as follows:

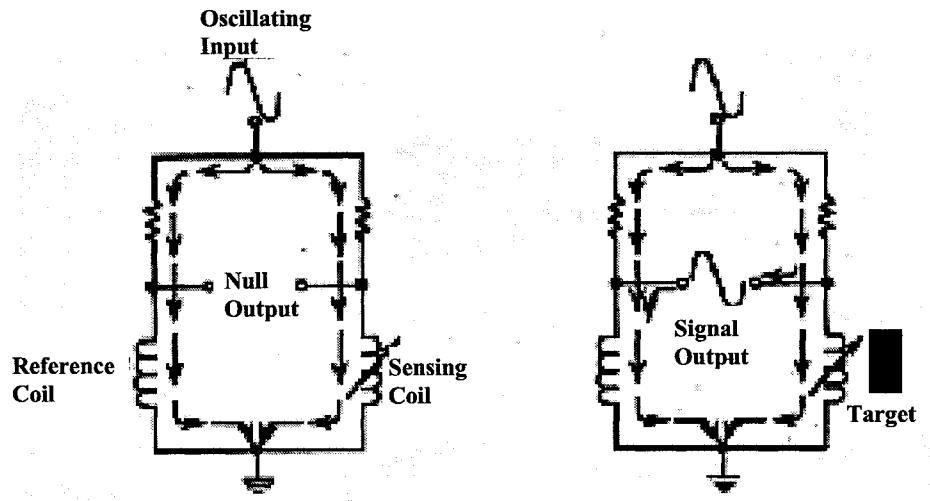
$$L = \mu_0 \cdot (N^2) \cdot \left(\frac{A}{l} \right)$$

it is clear that a change in the inductance can be caused by a change in the permeability of the magnetic material, in the number of turns, or in the geometric configuration of the coil. All three methods are used in practical sensing applications. The change of inductance by variations in the geometric configuration is the result of the coil arrangement. There are two parts of the coil mounted on iron cores. One part is stationary and the other part is movable. The displacement, which is to be measured, changes the position of the movable part of the coil, which results in the geometrical variation of the entire coil, and hence the coil inductance. Some sensors have also been designed that utilize variations in the number of turns.

Another very common approach of the inductive proximity sensors is shown in figure below. Here, a pair of coils are wired in a bridge circuit and biased with an AC signal. If a conducting object is positioned near the end of the device, it is closer to the sensing coil than the reference coil. The presence of a conductor has an important and complicated effect in this situation as explained in the following. A sheet of metal may be described as an assortment of essentially free electrons. If an electric field is applied to the metal, the electrons are free to move, and do so with very little resistance - hence the low resistance of a sheet of metal. In the presence of a magnetic field, the electrons feel a Lorentz force, which causes their trajectories to be curled into circles. These circular trajectories create a new magnetic field, which extends out of the sheet of metal back through the coils in the device. Since the sensing coil is closer to the metal sheet, the flux through the coil due to the sheet is larger in the sensing coil. These additional magnetic fields cause an additional opposing voltage in the coil, and the effect is to increase the inductance of the coils. Since the sensing coil is closer to the sheet, its inductance is increased more.

A bridge circuit is used to measure these changes in inductance, as shown in the next figure. Here, we can see that the effect of the sensing coil changing more than the reference coil is that there will be an AC voltage appearing at the bridge outputs. The amplitude of this difference is proportional to the inductance difference - which is related to the distance to the metal sheet in a very complicated way. The actual relation between distance and inductance change is too complicated



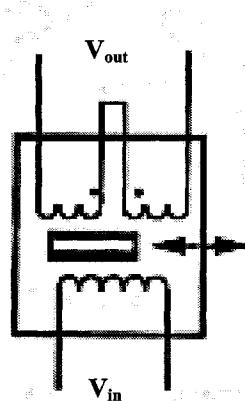


to derive in general, since it relies on the geometry of the situation, so this approach is not generally used for accurate position sensing. However, there are a number of important applications that benefit from a signal that indicates the presence of a metal object. Mine detecting can be one example. As another example, many automotive interchanges these days rely on this sort of inductive proximity sensing to detect the presence of a car in the left turn lanes or waiting in other lanes.

An important, but different sort of phenomenon, can be present if the metal object used to modify the inductance is a Ferromagnetic material. Ferromagnetism is an effect in which the magnetic moments of the bound electrons and the nuclei also interact with external fields. In a ferromagnet, the orientation of the magnetic moments in the material can become aligned with external fields, causing an effective field amplification that can be very large. In addition, this internal alignment can persist after the external field is removed, or even if its direction is changed. Because the field amplification may be as much as 1000X, the presence of ferromagnetic materials greatly increases the sensitivity of inductance coil bridges to their presence. Therefore, such sensors are very much more sensitive to ferromagnetic objects, and one often sees sensors which are tuned to detect a small piece of ferromagnet attached to a moving metal part.

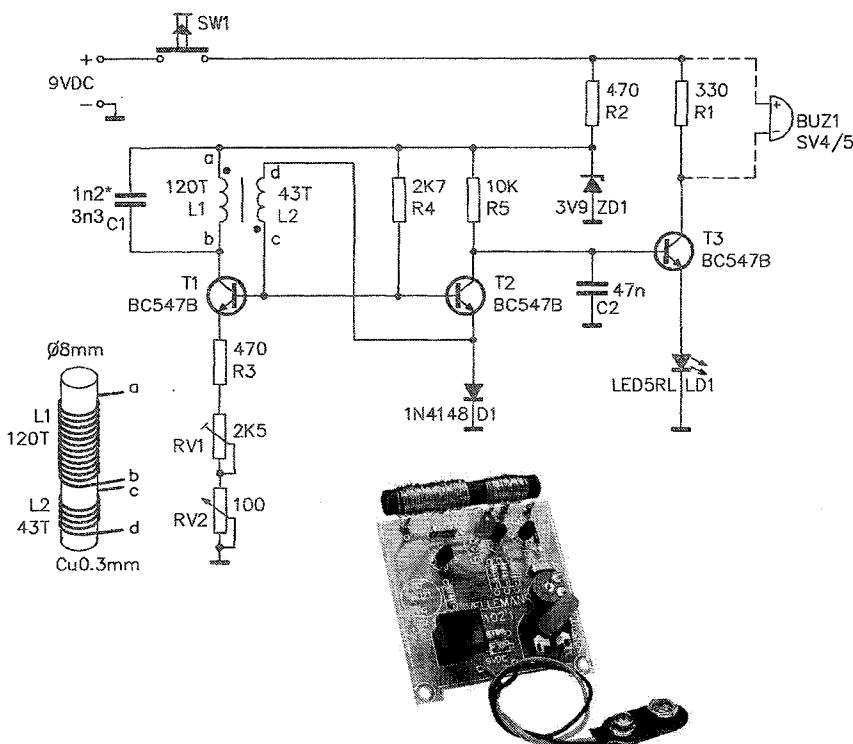
A common example of the use of a ferromagnetic element is as shown in figure below. In this system, the amount of magnetic field from one coil that is directed towards part of a second coil is dependent on the position of a ferromagnetic element. In the system shown in the figure, the two halves of the pick-up coil (wired to V_{out}) are wound in opposite directions. If the ferromagnet were not present, the flux in each half of the pick-up coil would be equal and opposite, and V_{out} would be zero. When the ferromagnet is positioned in the middle, there is also a complete cancellation of the flux. However, whenever the ferromagnet is displaced, the flux balance is changed, and the net effect is that there will be a voltage across the pickup coil whose amplitude is proportional to the displacement of the ferromagnet from its center position.

This sort of inductive position sensor is commonly used in a class of devices called Linear (or Rotary) Variable Displacement Transducer (LVDT or RVDT). These transducers can have very good accuracy (much better than 1% of the total range of motion, which is commonly called "stroke"), and are often used for precision position measurement applications such as flap and rudder position measurements in aircrafts.



8.4.2 Application

Circuit on the right shows a metal detector circuit that can be used for identifying pieces of metal from close distances. The Zener diode regulates the input voltage to the LC circuit to 3.9 V. The diode at the collector of the T2 transistor rectifies the oscillating wave. When there is no metal in the vicinity of the oscillator, the amplitude of the rectified wave is not sufficient to switch on the transistor T3. When a metal piece gets close to the windings, it increases the amplitude, and causes T3 to switch on. The circuit components are listed in table below. This circuit can be purchased as a kit from Velleman (available in the Design Lab.)



Label	Art. Nr.	Qty.	Description
C1	C3N3	1	3.3nF CERAMIC CAPACITOR
C2	C47N0	1	47nF CERAMIC CAPACITOR
D1	1N4148	1	1N4148 (IN914)
L1,L2	ROD8X50	1	AERIAL ROD 8 X 50mm , TWO WINDINGS
LD1	LED5RL	1	LED 5mm RED TD UNIVERSAL
R1	RA330E0	1	RESISTOR 1/4W 330E
R2,R3	RA470E0	2	RESISTOR 1/4W 470E
R4	RA2K7	1	RESISTOR 1/4W 2K7
R5	RA10K0	1	RESISTOR 1/4W 10K
RV1	K002SH	1	PIHER TRIM 2K2 SMALL HOR
RV2	E100LH	1	PIHER TRIM 100E LARGE HOR
SW1	S500	1	KEY SWITCH 500
T1...T3	BC547B	3	BC547B SI-NPN UN 50V-0.2A
ZD1	ZA3V9	1	ZENER DIODE 3V9-500mW
AXE	AXE	1	AXE 19mm FOR PIHER PT15
H7102B'1	H7102B'1	1	ASSEMBLING MANUAL
H7102P'1	H7102P'1	1	PARTSLIST
S509/BK	S509/BK	1	LOOSE KNOB S509/NE (BLACK)
SNAP9V	SNAP9V	1	BATTERY SNAP 9V "T" TYPE/LEADS 150mm
VK7102	VK7102	1	SUBKIT K7102 WINDING WIRE + PCB

8.5 Switch Sensors

The switch sensor is perhaps the most common sensor used in engineering applications. Switch sensors are used for a variety of purposes, including:

Contact (Touch) Sensing: Switch sensors can be used to indicate when a mechanism has made physical contact with another object. For example, a switch sensor can trigger when a robot's body runs into a wall or when a robot's gripper closes around a cube.

Limit Sensing: Related to simple contact sensing, a limit sensor detects when a mechanism has moved to the end of its range of travel, signaling to the control program that the motor should be turned off.

Shaft Encoding: The vehicle's axle may be fitted with a contact switch that clicks once per revolution. Software that counts the clicks can then determine the amount and speed of the axle's rotation.

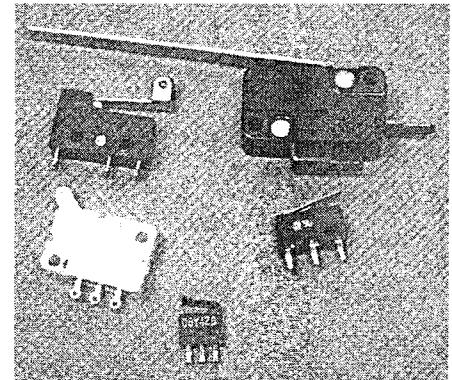
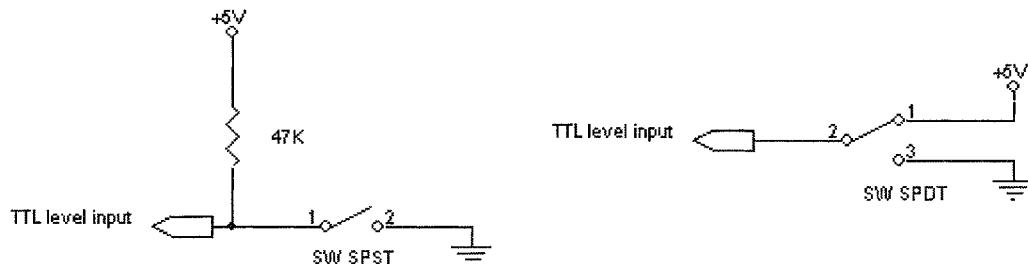
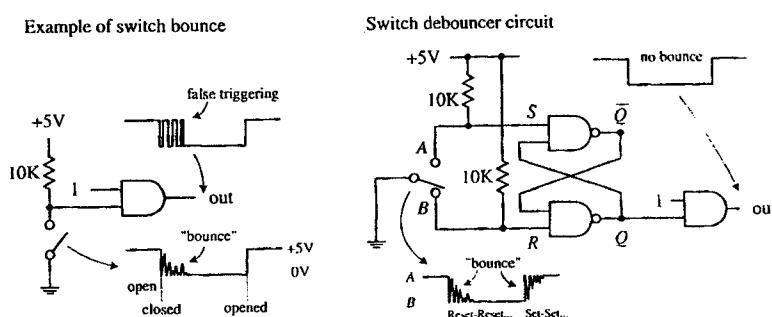


Figure below shows a number of common switches useful for touch sensing applications. The two basic kinds of switches are the microswitch, which is usually encased in a rectangular housing and often has an attached lever, and the pushbutton switch. Microswitches typically have three terminals: "NO" (normally open), "NC" (normally closed), and "C" (common). In the relaxed, unpressed state, the common terminal is connected to the NC contact; when pressed, the common terminal moves to the NO contact. A pushbutton switch is simpler. In a normally-open pushbutton, when the switch is pressed, the two contacts are connected. Normally-closed pushbuttons also exist, but they are less common.

The following figure shows how a single throw switch can be wired to a sensor input port. When the switch is opened, the sensor input is pulled to the +5V supply by the pull up resistor. When the switch is closed, the input is tied to ground, generating a zero voltage signal. A double throw switch makes life easier but they are less common.



One major issue with microswitches is the *bouncing* effect, which occurs in almost all mechanical contact switches. When the switch closes the circuit as a result of two contacting conductors, the contact is not a smooth transition but a series of random on-off sequences before the mechanical vibrations dissipate. Therefore, the switch output voltage shows random fluctuations for a short period of time whenever the switch closes. Though the duration of these fluctuations is short, it can affect the behaviour of any logic circuit or processor that receives the signal. The transition period can be disregarded in the processor's real-time code by having multiple readings in a longer period. But, a more effective solution is to use a *debouncing* logic circuit, e.g., SR latch, as discussed in Chapter 5 and shown in figure below.

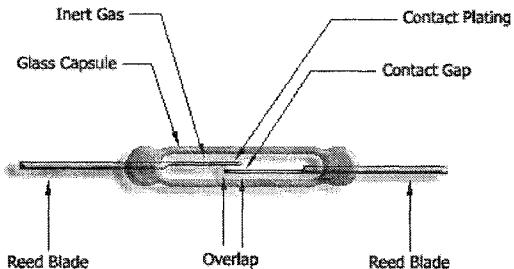


8.6 Other Sensors

There are a host of other types of sensors in the market for a wide range of applications, including the following:

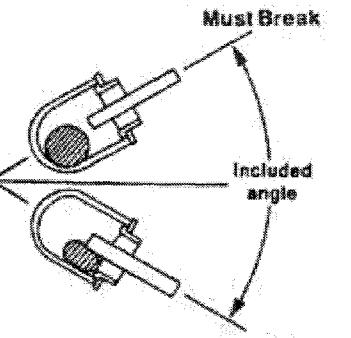
Reed Switch

This sensor becomes shorted in the presence of a magnetic field and is an open circuit otherwise. The switch is enclosed in a glass capsule with an inert gas to prevent oxidation of the plating inside. When a magnet is brought close to the switch, the two contact plating become opposite poles of a magnet and attracts each other. Since it is a switch, it does not matter which way reed blades are connected.



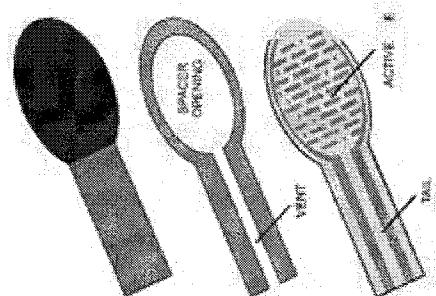
Tilt Switch

This sensor becomes shorted when it is tilted enough, and is an open circuit otherwise. As shown on the diagram to the left, a metal ball is located inside the switch. When the sensor is tilted enough, the ball moves so that it comes in contact with the two leads, thus shorting the circuit. When the ball rolls away from the two leads, the two leads are disconnected and become an open circuit.



Force Sensitive Resistor (FSR)

Force sensitive resistors are used to detect changes in the amount of physical force or pressure that is applied to it. The picture to the right shows the structure of a force sensitive resistor (FSR). As force is applied on the FSR, more connections are created between the two leads, thus decreasing the resistance of the FSR. In short, as the force on the FSR increases, the resistance decreases.



Thermistor

Thermistors are elements that change its resistance based on its temperature and generally follow a linear relation: $\Delta R = k \cdot \Delta T$. Thermistors come in two varieties: ones whose gain is positive $k > 0$, and ones with a negative gain $k < 0$.



Hall Effect Sensor

These sensors detect a magnetic field, due to the induced voltage that is orthogonal to the current flow through a special type of semiconductor (Hall effect). Thus, they can be used to signal when a magnet is in their vicinity.



8.7 Transducers

There exist numerous transducers, in a form of either Integrated Chips (ICs) or break-out boards, which provide ready signal for the processors. Some of these transducers are introduced below:

Microphone

This transducer amplifies the signal generated by the microphone by 100x, which is enough to be picked up by microcontrollers or processors. The board conveniently has 3 pins for power (V_{CC}), ground (GND), and analog output (AUD).

Digital Compass

This transducer acts like a compass and outputs a value based on the degree between the direction to north and the arrow etched on the board. The pins are for power, ground and 2 pins for I²C-based interface.

Gyroscope

This board detects the angular velocity and outputs a voltage proportional to it.

Accelerometer

This board outputs 3 voltage outputs, each proportional to acceleration in the X, Y and Z directions. The X,Y and Z axis' are etched on the board.

Humidity and Temperature Sensing

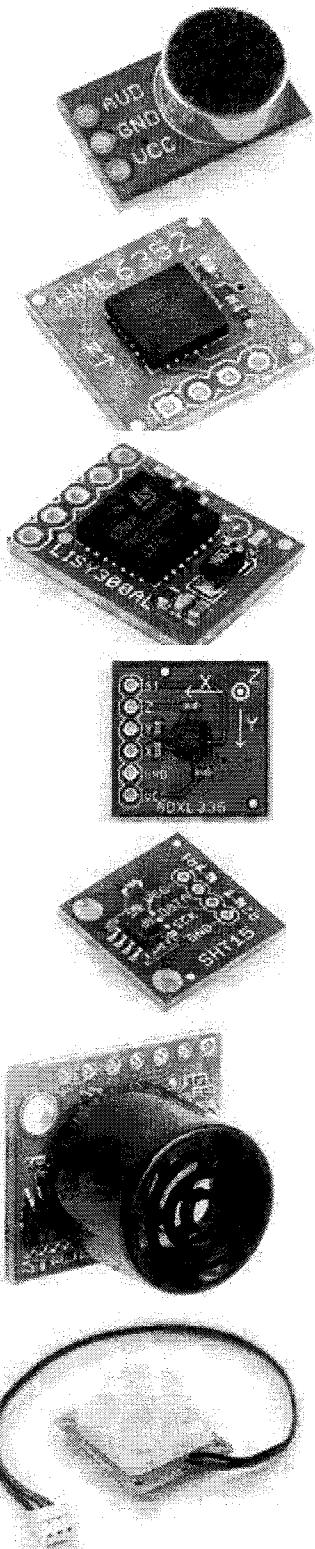
This board has two voltage outputs: one for temperature and the other for humidity.

Sonar Range Detector

This board detects the distance to a solid object using a sonar sensor, and it has an analog voltage output proportional to the distance and also an RS-232 interface.

PIR Motion Sensor

On this board, the output becomes high when the sensor detects any nearby motion, and low otherwise



8.8 Practical Notes

The real world is a noisy place compared to the digital world of a computer, and in engineering applications, this noise can make sensor readings unreliable. Sensor data mirrors the non-ideal nature of the machine's environment and must be interpreted by the computer. Below are a few of the common types of errors that appear when reading sensors:

- Glitchy data is often a problem due to faults in the sensor hardware. When a glitch occurs, the sensor may return an unexpected value or a value outside of the range of possible values. Often, loose connector plugs or shorted wires can cause spurious input by intermittently losing or making contact. This can be identified and fixed in software by ignoring values that fall outside of the expected input.
- Noisy data is a problem for most sensors. The world is full of flickering lights, uneven surfaces, and magnetic fields that can all cause errors in measurements. Since noise tends to be a random process, there is no way to predict it, but its effects can be minimized by averaging multiple values together when reading a sensor.
- Drifting data is often the result of sensors retaining some sort of memory. This problem is particularly prominent in light sensors, which are affected by changes in the ambient lighting of the room. For example, as a robot moves from place to place, the amount of light reaching the sensor can vary and change the range of the measurements. Some light sensors are also sensitive to heat and return different values as they warm up. These effects occur slowly over time, so they can be very difficult to detect. One option is to give the robot the ability to recalibrate itself whenever it finds itself in a known situation. This way, calibration values can be updated on the fly as the robot moves from one part of the environment to another.
- Most mechanical switches are afflicted with a phenomenon called bouncing. Wires are not ideal and instantaneous conductors, although they are often modeled that way. Instead, when a circuit is broken, the electricity continues to flow for a few microseconds. Since it has no place to go when it reaches the break, it bounces back and forth along the wire a few times before it settles down. This causes voltage levels to rise and fall, causing the sensor reading to transition a number of times (often dozens). A similar effect also causes the switch to bounce when the circuit is reconnected. In most circumstances, this does not cause a problem, but it can come into play for operations like counting the number of times a button is pressed. If sampling is performed too quickly, the computer can measure multiple counts when the button is pressed just once. This problem can be solved in software by ignoring other transitions that occur for a small amount of time after the first transition is detected. It can also be solved in hardware by the addition of appropriate circuitry.
- Light sensors are particularly sensitive to changes in the world. Differences in ambient lighting from one room to another can wreak havoc on the readings that the sensor returns. When the machine needs to be moved from one environment to another, calibration is often necessary to adjust the settings used to interpret the data. Even though most light sensors are analog in nature, you will often use them as if they were digital sensors. Rather than asking the question "How bright is it?", you will instead ask "Is it light or dark?" The goal of calibration, then, is to choose a threshold value that will separate light values from dark values. The most common way to choose a threshold is to take readings in a controlled situation, such as during a calibration routine. By averaging two readings, one for light and one for dark, a value can be chosen to separate the two conditions. Then, each time the sensor takes a reading, it can be compared to the threshold to determine if the sensor is seeing light or dark.

8.9 Odometry

Dead reckoning (derived from “deduced reckoning” of sailing days) is a simple mathematical procedure for determining the present location of a vessel by advancing some previous position through known course and velocity information over a given length of time. The vast majority of land-based mobile robotic systems in use today rely on dead reckoning to form the very backbone of their navigation strategy, and like their nautical counterparts, periodically null out accumulated errors with recurring “fixes” from assorted navigation aids. The most simplistic implementation of dead reckoning is sometimes termed *odometry*; the term implies vehicle displacement along the path of travel is directly derived from some onboard “odometer.” A common means of odometry instrumentation involves optical encoders directly coupled to the motor armatures or wheel axles.

Odometry is based on simple equations. Consider a differential drive vehicle, and suppose that at sampling interval i the left and right wheel encoders show a pulse increment of N_L and N_R , respectively. The conversion factor c_m that translates encoder pulses into linear wheel displacement can be obtained as:

$$c_m = \frac{\pi D_n}{n C_e}$$

where D_n is the nominal wheel diameter, C_e is the encoder resolution (in pulses per revolution), and n is the gear ratio of the reduction gear between the motor (where the encoder is attached) and the drive wheel. We can compute the incremental travel distance for the left and right wheel, $\Delta U_{L,i}$ and $\Delta U_{R,i}$, according to

$$\Delta U_{L/R,i} = c_m N_{L/R,i}$$

and the incremental linear displacement of the robot’s centrepoint C , denoted ΔU_i , according to

$$\Delta U_i = \frac{\Delta U_R + \Delta U_L}{2}.$$

Next, we compute the robot’s incremental change of orientation as

$$\Delta \theta_i = \frac{\Delta U_R - \Delta U_L}{b}$$

where b is the wheelbase of the robot, ideally measured as the distance between the two contact points between the wheels and the floor. The robot’s new relative orientation θ_i can be computed from

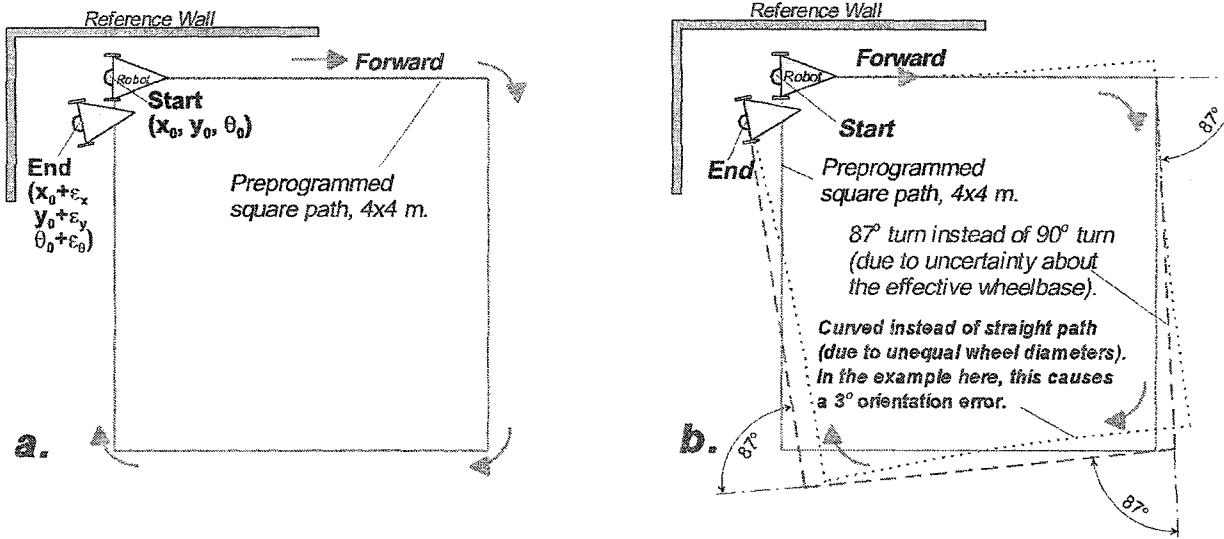
$$\theta_i = \theta_{i-1} + \Delta \theta_i$$

and the relative position of the centrepoint is

$$\begin{cases} x_i = x_{i-1} + \Delta U_i \cos \theta_i \\ y_i = y_{i-1} + \Delta U_i \sin \theta_i \end{cases}$$

where x_i and y_i are relative position of the robot’s centrepoint C at instant i .

Odometry is the most widely used navigation method for mobile robot positioning. It is well known that odometry provides good short-term accuracy, is inexpensive, and allows very high sampling rates. However, the fundamental idea of odometry is the integration of incremental motion information over time, which leads inevitably to the accumulation of errors. Particularly, the accumulation of orientation errors will cause large position errors which increase proportionally with the distance traveled by the robot. The basic assumption of Odometry is that wheel revolutions can be translated into linear displacement relative to the floor. This assumption is only of limited validity. One extreme example is wheel slippage: if one wheel was to slip on, say, an oil spill, then the associated encoder would register wheel revolutions even though these revolutions would not correspond to a linear displacement of the wheel. Along with the extreme case of total slippage, there are several other more subtle reasons for inaccuracies in the translation of wheel encoder readings into linear motion. All of these error sources fit into one of two categories: *systematic errors* and *non-systematic errors*.



Systematic Errors: Unequal wheel diameters, Average of actual wheel diameters differs from nominal wheel diameter, Actual wheelbase differs from nominal wheelbase, Misalignment of wheels, Finite encoder resolution, Finite encoder sampling rate.

Non-Systematic Errors: Travel over uneven floors, travel over unexpected objects on the floor, wheel-slipage due to: slippery floors, over-acceleration, fast turning (skidding), external forces (interaction with external bodies), internal forces (castor wheels), and non-point wheel contact with the floor.

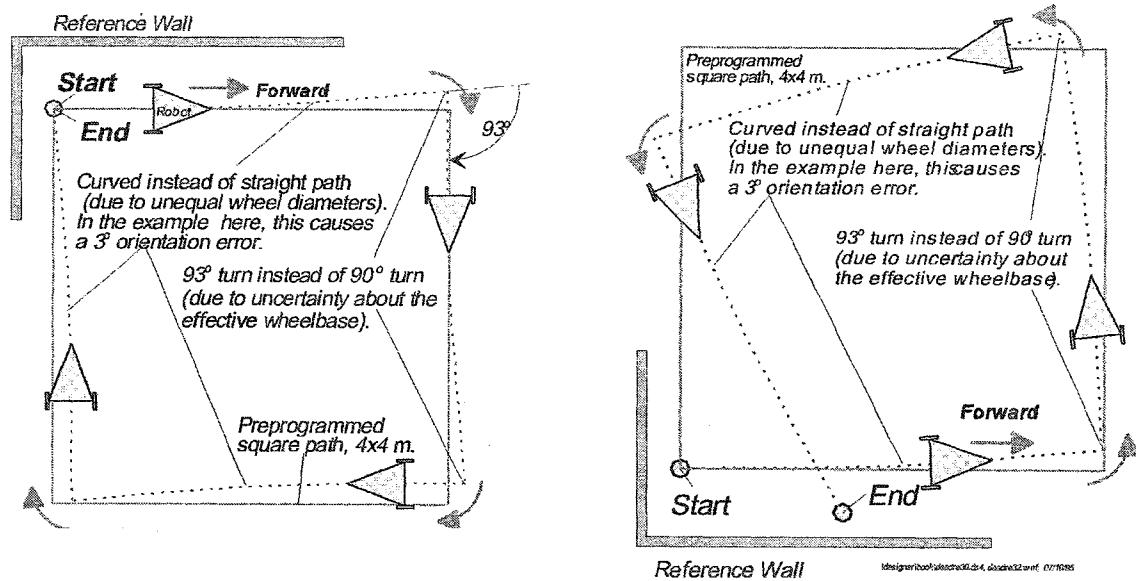
The clear distinction between systematic and non-systematic errors is of great importance for the effective reduction of odometry errors. For example, systematic errors are particularly grave because they accumulate constantly. On most smooth indoor surfaces, systematic errors contribute much more to odometry errors than non-systematic errors. However, on rough surfaces with significant irregularities, non-systematic errors are dominant. The problem with non-systematic errors is that they may appear unexpectedly (for example, when the robot traverses an unexpected object on the ground), and they can cause large position errors. Typically, when a mobile robot system is installed with a hybrid odometry/landmark navigation system, the frequency of the landmarks is determined empirically and is based on the worst-case systematic errors. Such systems are likely to fail when one or more large non-systematic errors occur.

Two major sources of systematic errors are: the error due to unequal wheel diameters: $E_d = \frac{D_R}{D_L}$, where D_R and D_L are the *actual* wheel diameters of the right and left wheel, respectively; and the error due to uncertainty about the effective wheelbase: $E_b = \frac{b_{actual}}{b_{nominal}}$, where b is the wheelbase of the robot. These two errors can be measured (and removed) by performing some simple experiments.

Figure above shows a 4×4 meter unidirectional square path. The robot starts out at a position x_0 , y_0 , and θ_0 , which is labeled START. The starting area could be located near the corner of two perpendicular walls. The walls serve as a fixed reference before and after the run: measuring the distance between three specific points on the robot and the walls allows accurate determination of the robot's absolute position and orientation. To conduct the test, the robot must be programmed to traverse the four legs of the square path. The path will return the vehicle to the starting area, but, because of odometry errors, not precisely to the starting position. Upon completion of the square path, the experimenter again measures the absolute position of the vehicle, using the fixed walls as a reference. These absolute measurements are then compared to the position and orientation of the vehicle as computed from odometry data. The result is a set of *return position errors* caused by odometry and denoted ϵ_x , ϵ_y , and $\epsilon\theta$.

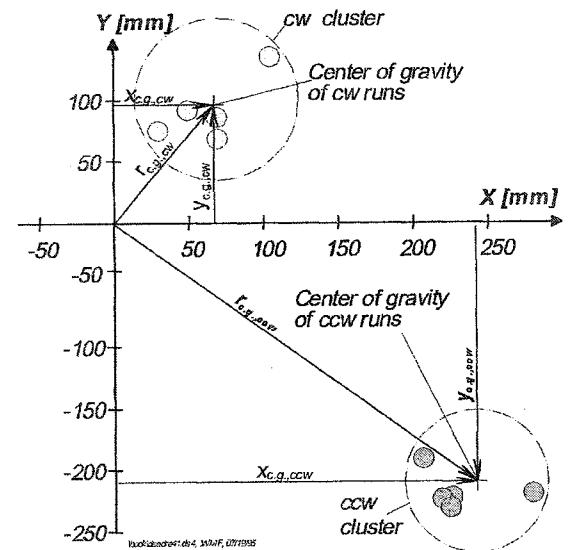
The path shown in the figure (a) comprises of four straight-line segments and four pure rotations about the robot's centrepoint, at the corners of the square. The robot's end position shown in the figure (a) visualizes the odometry error. While analyzing the results of this experiment, the experimenter may draw two different conclusions: The odometry error

is the result of unequal wheel diameters, Ed , as shown by the slightly curved trajectory in the figure (b, dotted line). Or, the odometry error is the result of uncertainty about the wheelbase, Ea . In the example of the figure (b), Ea caused the robot to turn 87 degrees instead of the desired 90 degrees (dashed trajectory). We can see from the figure (b) that either one of these two cases *could* yield approximately the same position error. The fact that two different error mechanisms might result in the same overall error necessitates further experiment to compensate both errors. Otherwise, it may lead to serious mistake. Imagine that in this example, the experimenter began "improving" performance by adjusting the wheelbase b in the control software only. According to the dead-reckoning equations for differential-drive vehicles derived above, the experimenter needs only to increase the value of b to make the robot turn more in each nominal 90-degree turn. In doing so, the experimenter will soon have adjusted b to the seemingly "ideal" value that will cause the robot to turn 93 degrees, thereby effectively compensating for the 3-degree orientation error introduced by each slightly curved (but nominally straight) leg of the square path. However, the concealed dual error will become obvious when the same square-path experiment is performed in the opposite direction. This is so because the two dominant systematic errors, which may compensate in only one direction, add up to each other and increase the overall error when run in the opposite direction, as illustrated in figure below. Thus, in performing the square-path experiment for Odometry calibration, driving the robot in both clockwise and counterclockwise direction is necessary.



The result of the bi-directional square-path experiment might look similar to the one shown in figure on the right. As this figure illustrates, the stopping positions for five runs each in clockwise (cw) and counterclockwise (ccw) directions were recorded. As obvious from the figure, the stopping positions after cw and ccw runs are clustered in two distinct areas. The distribution within the cw and ccw clusters are the result of non-systematic errors. However, when traveling over a reasonably smooth floor, the contribution of systematic errors to the total Odometry error can be notably larger than the contribution of non-systematic errors. To quantify the values of the errors, centres of gravity of the clusters can be computed, and calibration efforts should reduce the radial distance of these centres from the origin.

An effective method to reduce the Odometry systematic errors would be that, instead of attaching the encoders to the motors or wheels directly, a pair of accurately similar "knife-edge", non-load-bearing *encoder wheels* be added to the robot. Since these wheels are not used for transmitting power, they can be made very thin and with only a thin layer of rubber as tire. In this way, a good accuracy can be achieved for both the encoder wheelbase and wheel diameters.



Chapter 9

Suppliers

9.1	New Computer Hardware.....	1
9.1.1	New Computer Suppliers.....	2
9.2	Computer Surplus.....	3
9.3	Electronics Sellers.....	3
9.4	Mechanical Components.....	5
9.4.1	Hobby Stores	6
9.4.2	General Hardware Stores.....	6
9.4.3	Plastics.....	6
9.5	Web Information.....	7

We are lucky in the Greater Toronto Area to have access to most any component made in the world, the same day within 40 km of downtown. We are even more fortunate to have a large number of reasonably priced surplus shops of various sorts. Those students considering going home to another city for Reading Week are advised to do all their surplus shopping before they leave. Particularly in the electronics area, they may find themselves limited to Radio Shack, its small breadth and depth of stock, and high prices. Any students who have suggestions for suppliers, particularly surplus suppliers, in the GTA or other cities are asked to submit those to the instructor for inclusion in this chapter.

9.1 New Computer Hardware

We are lucky in the GTA to be at a PC computer nexus for the continent. There is no place in North America where system prices are cheaper, or the newest hardware is available sooner outside the main computer lines (Aptiva, Dell, AST, Packard Bell, etc.) in assembly shops. The large number of, and rabid competition between, assembly shops in and around Toronto keeps all system prices at a bare minimum. The price of all components is similarly very competitive.

Among the assembly shops, there are variations in both the price and quality of the products. The identical product may be found at several shops with a 20% variation in price. Products with very different parentages can be found for sale at the same price. In the clone business, there are several levels of "original" equipment, particularly in sound cards and video cards.

Commercial: This is an actual real component from the manufacturer, in a real box with full marketing graphics with full manuals and all the bells-and-whistles software. Occasionally, these products have noticeably better performance or greater capabilities. These are marketed to the consumer for system upgrades. Since the number of people who do their own upgrades is small and the cost is so high, these are seldom seen or purchased. Even the grayest clone shop will have what appear to be commercial equipment packaging on the wall but in fact sell OEM or worse.

Real OEM: These are the same components made by the real manufacturer, sometimes purchased in bulk by another company, packaged and resold. Typically the manual is written in broken English and the carton is plain or non-existent. The software is adequate - it contains all the necessary drivers - but usually does not contain all of the companion utilities. The marketing of these components is aimed at assembly shops (most computer stores) for inclusion in system packages. To protect the original manufacturer, shops are usually prohibited from selling these products outside systems but the majority do because these packages are significantly cheaper than commercial and the retail competition in the GTA is cut-throat.

Vanilla OEM: These are components either made by the real manufacturer or by another company under contract. They are typically the vanilla version of a very popular component, like the SoundBlaster16 (Creative Labs sound card) or Mach64 (ATI video card). Typically, these are fully functional but do not bet on these cards being 100% cloning. For example, the video cards in the Aerospace Design Lab PCs are Mach64 but because they are based on different ATI chips and different clock generators, Linux must use different software configurations. (I suspect at least 4 are in fact Chipset OEM, the next category.) Vanilla OEM are usually passed off as real OEM but are not, in fact, identical to the real commercial component. If they are made by the real manufacturer, a consumer can be more certain the component will be functionally equivalent.

Chipset OEM: These are the slippery slope to bottom-feeder clones and industrial fraud. The cards are based on some chip that can be traced to the real manufacturer in some way - usually it is made by the company that makes those chips for the real manufacturer, perhaps sold through the back door. Some other company, sometimes with permission of the real manufacturer, will create boards and put the chips on them and then sell them under some name that looks suspiciously like the original: SB16 or M64. Careful inspection of the component, box and documentation will determine that "ATI Technologies" or "Creative Labs" does not appear anywhere on them. The name of the actual manufacturer may be well hidden or non-existent. Make sure the exchange or refund policy is known before buying this type of hardware. Do not buy these types of components from a small industrial-unit store with no history. If the store disappears, so does any chance of restitution if there is a problem.

Clones: These manufacturers are at least honest about their role in the market - they sell their products under their own name. For example, there are dozens of clones of Novell's NE 2000 10Mbit network card. These products are typically poorly documented but otherwise very functional. Usually in the first paragraph, or under a heading "Software Drivers", they will say their component is register-compatible with a brand name or they will say to use the software driver for a brand name.

9.1.1 New Computer Suppliers

Hits Computers

3802 Victoria Park Ave., (near Active Electronics)
Toronto
(416) 499-2889 Fax: (416) 492-6501
2180 Steeles Ave. West, (at Keele)
Concord
(905) 660-3280 Fax: (905) 660-4336
420 Highway 7 East, (between Bayview&Leslie beside McDonalds)
Richmond Hill
(905) 889-6513 Fax: (905) 889-3125
Note: Getting pricey

Throne Computers

8994 Yonge St. (between Hwy 7 and 16th)
Richmond Hill
(905) 881-3299 Fax: (905) 881-6933
4810 Sheppard Ave East, (between McCowan and Markham Rd.)
Scarborough (416) 609-1668 Fax: (416) 609-8860
Note: Pretty cheap. Everything is OEM or lower.

MDG Computers

415 Horner Ave. Unit 2&3 (south east of QEW and 427)
Etobicoke
(416) 225-9343 Fax: (416) 225-8672
64 Jardin Dr. Unit #5? (Keele, N of 407, near Hits)
Concord
(905) 669-8841 Fax: (905) 669-7096 (ask for Dave)
Note: Cheapest around. If you want something, make sure it is written down on the order. Feel fortunate if you get an OEM.

Numerous Shops

College West of Spadina and at Queen and Spadina
Note: Great for components.

Numerous Shops

Alden Rd, 14th Ave, Woodbine and Steeles Area
Check Toronto Computes for Addresses
Not recommended for systems due to their short longevity

Because a particular store is or is not listed is not an endorsement or slight, these are just places the instructor has purchased computers or components from lately. In all cases, *buyer beware*. More established, and more costly stores are:

Grey Tech

550 Alden Rd, Unit 110 Markham (905) 470-1425 Fax (905) 470-9556

Dell

www.dell.ca

Expect these places to provide real commercial components or tell buyers something is OEM when it is. These places are just too expensive for most people.. If a student is considering buying a system from an assembly shop, the instructor would be happy to give his \$0.02 on their track record.

9.2 Computer Surplus

SCS

3804 Victoria Park Ave
Toronto
(416) 502 3090 Note: old Notebooks at decent prices.

PCUsed

(new address north side of) Dupont
(416) 922-6363
Large stock of used computers and occasionally they have free ones put out near the door.

Above All Surplus

Bathurst and Bloor
590 Bloor Street West,
Toronto
(416) 588-8119

Expect 50% of surplus computer boards on the floor to work - which is usually good considering the price. Parts behind the counter may be expected to work.

Friends and Relatives with old hardware: The best tactic is to assist them in purchasing new hardware. Go to the store, help them chose a system, help them select software for it. When the system arrives, help them configure it setting up several notable pieces of software - perhaps a financial program, dial-up access and a game. While they are dazzled by the speed and graphics, ask for that pile of junk in the corner that was their old hardware. This method does not pay if a student can find a reliable source of functioning PC surplus at reasonable prices.

9.3 Electronics Sellers

The first 7 are all very close together, the first 6 within walking distance of each other.

Active Electronics

3790 Victoria Park Avenue, Suite 100,
Toronto, Ontario
M2H 3H7
(416) 498-9886 Fax: (416) 498-9875
1350 Matheson Blvd., Unit 2
Mississauga, Ontario
L4W 4M1
(905) 238-8825 Fax: (905) 238-2817

First industrial plaza south of Steeles on the west side of Victoria Park at the intersection with Gordon Baker Road. Wide stock of new parts in neat little packages. The the sales staff sell by code number - they may as well be selling auto parts. The whole staff could not match the electronics knowledge of one salesperson from Active Surplus with a bad attitude. They also stock suppliers and parts that no one else stocks. If a sales person types a code number in the computer and says they do not have one, go check the shelf anyway. There is a good chance there is one of a different power rating, speed or insignificant variation in suffix on the part number.

Electrosonic

1100 Gordon Baker Road,
Toronto, Ontario
M2H 3B3
(416) 494-1555

Directly beside Active Electronics in the first building west on Gordon Baker. A huge stock of every part imaginable. Know the Electrosonic Part Number and Supplier Code - especially if it is busy - or a great deal of time will be spent figuring out their system.

Daiwa

3802 Victoria Park Avenue

North York, Ontario

M2H 3H7

(416) 499-2889, (416) 492-6501

1350 Matheson Blvd. East, Unit 7

Mississauga, Ontario

L4W 4M1

(905) 238-8701 Fax: (905) 238-1586

In the same plaza as Active Electronics at both locations.

SCS

3804 Victoria Park Avenue

North York, Ontario

M2H 3H7

Next to Daiwa, in the same plaza as Active Electronics. Limited to PC surplus parts and new and used PCs and notebooks. Do ask at the counter about equipment that is not on display. The owner aspires to have a larger surplus business and has a large stock room.

Saval Electronics

3791 Victoria Park,

Toronto

(416) 494-8999

1350 Matheson Blvd. E., Unit 3-4

Mississauga, Ontario

L4W 4M1

(905) 238-8640 Fax: (905) 238-9757

A block or two south of Active Electronics on the east side of Vic Park, or in Mississauga in the same building. Good, large electronics surplus store. A bit pricey for a surplus place, Daiwa, Above All and Active Surplus are better for price. Service could be better too.

Toronto Surplus and Scientific

608 Gordon Baker, (at Finch and 404)

Toronto

(416) 490-8865

A-1 Electronics

718 Kipling

(416) 231-4331

614 Yonge St.

(416) 928-1817

Call first, current stock unknown.

Creatron

255 College Street,

Toronto, Ontario

M5T 1R5

A good stock of electronic kits and parts in a rather small space.

College Home Hardware (formerly Honson Computer)

306 College Street,
Toronto, Ontario
M5T 1S2

Similar to Creatron, a good stock of electronic parts in a small space.

Active Surplus and Active Surplus Annex

345-347 Queen Street West,
Toronto, Ontario
M5V 2A4
(416) 593-0967, Fax: 593-0909

Go south from the University on St. George Street until it ends at Queen Street West. Active Surplus is two stores west of the intersection. The first stop for all surplus needs: electronic, mechanical, tools, wire - everything. Note the basement is not open until noon. Use the skill of the older man in the basement. He is very kind and patient and will readily test motors for polite students. Students should ask at the desk if they cannot not find something, particularly gear-head DC motors. They have them under the extension to the front counter behind some dirty Plexiglas sheet. Avoid PC components from Active Surplus or at least check the motherboards for footprints. This is the original Ferengi store - buyer beware. If a student comes in asking for something, they will leave with something and whether it is the right something or not, Active will have their money. Students may perceive that Active staff lied to them - they have never actually lied to me in 14 years and have actually been quite fair when full-price things performed significantly below standard. When I was unclear about what I wanted and explained it poorly, their advice was bad and I sometimes walked out with the wrong something. On a good day, for a polite student who identifies them self as a design student from UofT, for non-AS-IS¹ components, Active usually offers an in-store credit, that is, they still have the student's money: they will trade junk.

Above All Surplus

Bathurst and Bloor
590 Bloor Street West,
Toronto
(416) 588-8119

Located on the north side of Bloor opposite to Honest Ed's. Owned by a graduate of Active Surplus. Less mechanical gadgets, less dust, good surplus PC boards. This is a good store with a few very good quality motors, particularly Pitman motors.

Radio Shack

Check the telephone directory. Way too expensive and too small of a stock to be a first line supplier.

9.4 Mechanical Components

Canadian Bearings Ltd

1401 Courtney Park Dr, Mississauga (?), (905) 670-6700 1200 Tapscott Rd, Toronto (416) 298-7233 Good source for all precision mechanical parts. Pricey but feasible for the one or two critical mechanical components. Ask of they have recirculating ball screws.

Busy Bee Machine Tools

355 Norfinch Dr,
Toronto,
(416) 665-8008

Excellent supplier for tools and mechanical things used to support businesses using tools. They stock a wide variety of metal working and wood working tools.

9.4.1 Hobby Stores

National Hobby Inc

6089 Yonge St. (416) 222-8289

Keith's Hobby Shop Ltd.

5205 Yonge
Toronto
(416) 222-4721 Fax: (416) 222-4895

John's Photo and Hobby

2188 Danforth Ave (At Woodbine Subway)
Toronto
(416) 421-1850

Hobby Fun

668 Major Mackenzie Dr East (just west of Bayview), Richmond Hill
(905) 508-0231
Model car type hobbies. Miniature ball bearing races nearly any ID and OD - probably for model cars.

9.4.2 General Hardware Stores

Canadian Tire

839 Yonge (north of Bloor)
Toronto
(416) 925-9595

Home Depot

7 Curity
Toronto
(416) 752-5900
3155 Highway 7 (Just east of Woodbine)
Markham
(905) 940-5900
428 Ellesmere
Toronto
(416) 609-1800
2151 St.Clair Ave West
Toronto (416) 766-2800
140 Northview Blvd, (400 and Hwy 7)
Woodbridge
(905) 851-1800

9.4.3 Plastics

Cadillac Plastics

91 Kelfield
near Dixon Rd and Hwy 401
Toronto
(416) 249-8311
They sell scraps for a dollar a pound including lexan. They like design students. Service at the cash desk is slow.

Johnson Industrial Plastics

20 Fleeceline

(416) 252-9551 Fax: 255-7706

Small cash desk.

Peckover's Brass

2195 Langstaff Rd. (at Keele)

Concord

(905) 669-9444

Small cut-offs area and cash desk.

9.5 Web Information

Look for the list of links off of the AER 201 web page:

<http://aer201.aerospace.utoronto.ca/aer201>

Every student who has used the web to acquire information has a favorite search engine. Most are probably not aware of meta-search engines, of which there many, that make automated queries to a number of search engines and compile the results. The instructor's favorites are:

Savvy Search

www.savvysearch.com/

Google Search

www.google.com/

By selecting categories for the search, appropriate search engines are queried.

