The University of Western Australia
School of Electrical, Electronic and Computer Engineering

Digital Systems ELEC1300

Laboratory Manual

2009

John Dell

Thomas Bräunl

Table of Contents

Acknowledgments

Some of the experiments in this course are based on the experiments developed by Prof. Ricardo B. Uribe, of the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. They have been modified to include dynamic breaking and MOS transistors. The development of the laboratories at The University of Western Australia would not have been possible without the aid of Mr Kwah Heong-Yong and Mr Rob Creswell who worked tirelessly over the summer to get things working. The skills and advice of both the Electronics Workshops and the Mechanical Workshops staff have been essential in getting the equipment for this lab operational.

Introduction

The purpose of these laboratories is to introduce you to the concepts and application of computer engineering in a practical way, and in a manner that is also hopefully enjoyable. The laboratories build on the work covered in lectures, reinforcing the concepts needed in the design of systems.

In the laboratories you will develop an autonomous vehicle that will follow a line on a track. In the course of developing this vehicle, you will

- Investigate logic circuits
- Introduce the fundamentals of microprocessors
- Examine the operation and design of a simple control system

Safety

In general the voltages used in this laboratory are kept low (<20V) to minimise hazards. However there are a number of general laboratory rules which will be strictly enforced.

1) ***No student will be allowed entrance to the laboratory without adequate footwear***. Covered, low healed shoes are the minimum requirement. Shoes must be worn throughout the duration of the laboratory.

2) ***Long hair must be tied back***. This laboratory involves the use of motors. While these motors are small and slow running, they are quite powerful and could cause serious injury if mishandled.

3) ***No loose clothing to be worn***. (For the same reasons as 2).

4) ***The electric vehicle must be adequately mounted when not on the test track or on the floor***. The electric vehicle is quite heavy and has a number of sharp corners that could cause serious injury or result in damage if it fell from the bench onto the floor.

5) When changing the configuration of a circuit, ***turn off the power***.

6) ***Ground. Know and ensure your partner knows which nodes of your circuit are grounded***. It is very easy to get unintentional shorts in your circuit when you don't know which parts of the circuit are ground. This is particularly true with the protoboards.

7) ***Heat. Components can be at high enough temperature to cause burns***. A number of components in this laboratory are power components, designed to operate at relatively high temperatures. Mistakes in circuit layout (see 6 above) can result in unexpectedly high power dissipation in parts of the circuit. Only a few watts is required to get some components to temperatures high enough to cause burns.

8) ***Check your circuit (at least twice) before turning it on***.

9) ***Live components. The casings of some components can be live***. Most semiconductor components with metal casings have the casing attached to one of the terminals of the device. The casing can therefore be at a high voltage. Be careful when probing the components to avoid accidentally shorting out parts of the circuit.

10) ***No food or drink is permitted in the laboratory***.

EMERGENCY PHONE NUMBER: 2222

Laboratory Organisation

- Laboratories will be done in teams of two students each.
- You are to keep to the same team members for each laboratory and use the same laboratory bench each time.
- There are a total of 5 laboratories to be completed (one every two weeks).
- You are required to have a laboratory book that must be handed in at the end of each laboratory session. ***PUT YOUR NAME, STUDENT NUMBER, BENCH NUMBER, AND DAY AND TIME OF YOUR LABORATORY SESSION ON YOUR LABORATORY BOOK***.
- ***All entries in your laboratory notebook must be dated and in ink***[*].
- Marks will be allocated each week for your laboratory work.
- Failure to attend a laboratory constitutes grounds for failing the unit. If you have extenuating circumstances for not attending a laboratory, you should inform the Laboratory Supervisor before the scheduled laboratory session[†]. A medical certificate is required in the case of illness. Failure to attend a laboratory will result in a minimum penalty of an assessment of zero marks for the laboratory.
- There is no formal homework required for this laboratory, however you would be foolish not to read the laboratory notes prior to coming to the laboratory.
- Students are expected to ask questions. The laboratory notes are often incomplete and you must make decisions about what exactly is important for various parts of the laboratory.
- ***NOTES IN BOLD ITALICS ARE IMPORTANT AND SHOULD BE READ CAREFULLY - THEY WILL GENERALLY PREVENT YOU DESTROYING COMPONENTS OR SAVE YOU CONSIDERABLE TIME.***

---

* When you work in industry, your laboratory/design notebooks can become legal documents. If you are involved in an invention, the date of precedence of the invention (the date determining who can own the patent rights) can be set by the dates entered in your laboratory/design notebook. However, these only have legal standing if they are in ink. Get into the habit early of writing important documentation in ink.

† You should also inform your laboratory partner.

Laboratory Equipment and Materials

Materials

Item	Quantity
Vehicle	1
Protoboard	1
Wire kit	1
H-Bridge module	2
Quad 2 input NOR module	1
Quad 2 input NAND module	1
Motorola HC12 based control board and debug module	1

A variety of other components (resistors, transistors, sensors, diodes, LEDs etc) are available for use throughout the laboratory. You are requested to consider alternative circuits. Discuss these with your demonstrator. If warranted, components other than those above can be used to implement your designs.

Equipment

Item	Model	Quantity
Power supply	GPC-3030	1
Function Generator	GFG-8219A	1
Bench Multimeter	GDM-8135	1
Power Meter	MetraHit29S	1
Digital Oscilloscope	TDS210	1
Probes		2
PC		1

A set of cables is also provided to connect the equipment.

Laboratory Notebook

The laboratory/design notebook is essential in any experimental environment. The purpose of the laboratory notebook is to keep an accurate record of your experiment work. The notebook should include:

- **Diagrams of all circuits used in the laboratory.** Draw all circuits used including sufficient detail so that the circuit could be built by someone reading your notebook (ie it must include element values, device types, test points, source of signals and power - eg sensor, function generator, battery or bench power supply, etc). The aim is to allow you to accurately repeat the experiment if required.
- **Equipment used.** If you only use the standard equipment supplied on the bench, just record the bench number. If you use a different piece of equipment, record the equipment function, model number and serial number. If you need to change any of the standard equipment (eg due to equipment failure), also record all of these details for the substituted equipment.
- **Experimental technique.** Each time you use a new instrument, write down a brief description of how it used. You do not need to do this every time you use an instrument.
- **Answers to questions.** Answer all questions posed in the laboratory notes in you laboratory notebook. Mark clearly which question you are attempting to answer.
- **All experimental data.** Be sure to include units and scale settings. For example, an oscilloscope measurement would include as a minimum the "VOLTS/DIV" and "SEC/DIV" settings used to make the measurements, then a list of numerical values of the data recorded. Enter your results legibly and ensure that you and your partner agree on the value recorded. Make extensive use of tables and graphs.
- **Team members.** Record the name of your team members.

You may also include calculations, observations and even speculations in your notebook, however, ensure that these are clearly marked and kept separate from your experimental data and from your question answers.

Notebook errors should be crossed out (***not obliterated - do not use Whiteout - marks will be deducted***) and initialled and dated by the recorder. In addition, each page should be initialled and dated as it is used. ***Do not remove pages from your notebook (marks will be deducted)***. Do not try to save paper by squeezing too much information onto a single page. It is a false economy and will result in errors and loss of marks.

The notebook must be written in ink.

Experiment 1

Objective

- To become aware of the laboratory safety requirements and how to operate the standard instruments in the laboratory (oscilloscope, function generator, multimeter, power meter, power supply).
- Examine some real logic components.
- To examine the vehicle base.
- To build a simple logic circuit using the protoboard.

Before coming to the first laboratory session ensure that you have read the safety notes on page 4 of this manual. You should also read the operating procedures for the equipment in Appendix A.

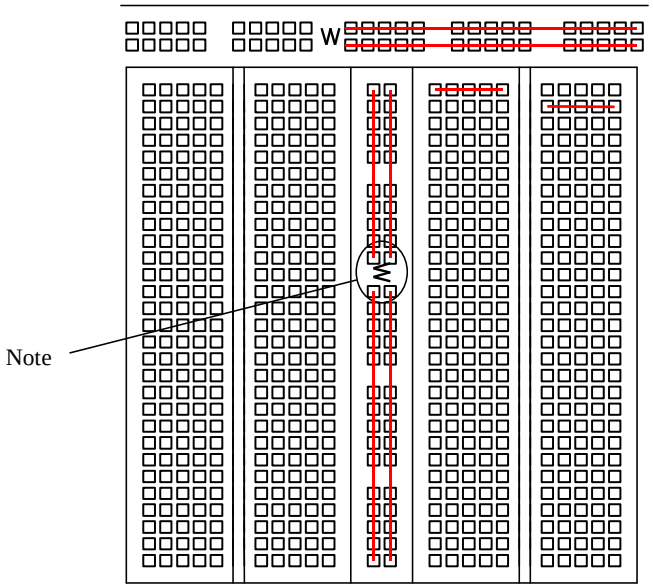
Procedures

Examination of the Protoboard

a) The protoboard is a set of socket strips and bus strips attached to a backboard. Three socket strips, mounted lengthwise, are used to place and interconnect modules and components. Bus strips, which are used for routing (bussing) commonly used signals or voltage levels, are also available on the board. Each strip is simply an array of holes with spring clips underneath them. Stripped #22 gauge solid wire, provided for you, is used for making interconnections.

i) You must only use this wire for making connections.

ii) Heavier gauge wire will damage the spring clips.



The diagram above is a schematic representation of the protoboard. Each square represents a hole in the socket strips on the board.

b) The matrix board is connected as follows:

   i) Each set of 5 holes running across the larger sections of the protoboard are connected together.

   ii) The long thin sections of protoboard are called bus strips, and consist of four lines of holes, two sets on either side of the W. The holes in each line from one end to the W are connected together.

c) Note that at the top of the protoboard are a set of connectors which are both screw terminals and also take banana plug leads, allowing you to make a connections between banana plug leads and wires leading to the protoboard. You use these to take the 0V, 5V and 12V connections from the back of the car onto the protoboard.

d) Place the protoboard on top of the vehicle with the screw terminals towards the back of the vehicle so that the power from the vehicle can be brought onto the protoboard using the small red, yellow and black banana-to-banana plug leads.

Logic for Control of the Vehicle

The aim of this laboratory is to design and construct a logic circuit to make the vehicle follow a white track on a black background. If the vehicle comes to broad white band, it is to stop. To achieve this behaviour, the vehicle is supplied with the following:

a) The vehicle is to have 4 input sensors labelled RS1, RS2, LS1 and LS2.

b) The sensors are to be placed on the underside of the sensor board on the front of the vehicle (the correct orientation of the sensors is with the writing to the front of the vehicle, with the sensor completely within one of the numbered blocks of four holes in the connector). LS1 is the left hand sensor closest to the centre of the vehicle. RS1 is the right hand sensor closest to the centre of the vehicle.

c) The sensor output is low except when it is over a reflective object.

d) The outputs from the sensors are to be connected from the top-side of the sensor board to the protoboard using the wires provided.

e) The circuit is to implement the following:

   i) If the sensors RS1 and LS1 are over the white band and the sensors RS2 and LS2 are over the dark background both motors should be running and the car should go straight.

   ii) If the white band turns to the LEFT while the vehicle is going straight, RS1 will go off the band towards the right, and the left hand motor should stop. While LS1 is still over the white track, the right hand motor should keep running. Therefore the vehicle will turn to the left, attempting to follow the band. Note that nothing has been said about RS2 and LS2 and therefore assume that they remain over the dark background.

   iii) If the white band turns to the RIGHT while the vehicle is going straight, LS1 will go off the band towards the left and the right hand motor should stop. While RS1 is still over the white track, the

left hand motor should keep running. Therefore the vehicle will turn to the right, attempting to follow the band.

iv) If all four sensors detect a white band (an obstacle), R and L are both low (logic 0) and the vehicle will stop.

v) All other combinations of inputs are invalid and we don't care what the vehicle does.

f) Based on the required behaviour described above, complete the following Karnaugh maps for the control signals to the left and right motors.

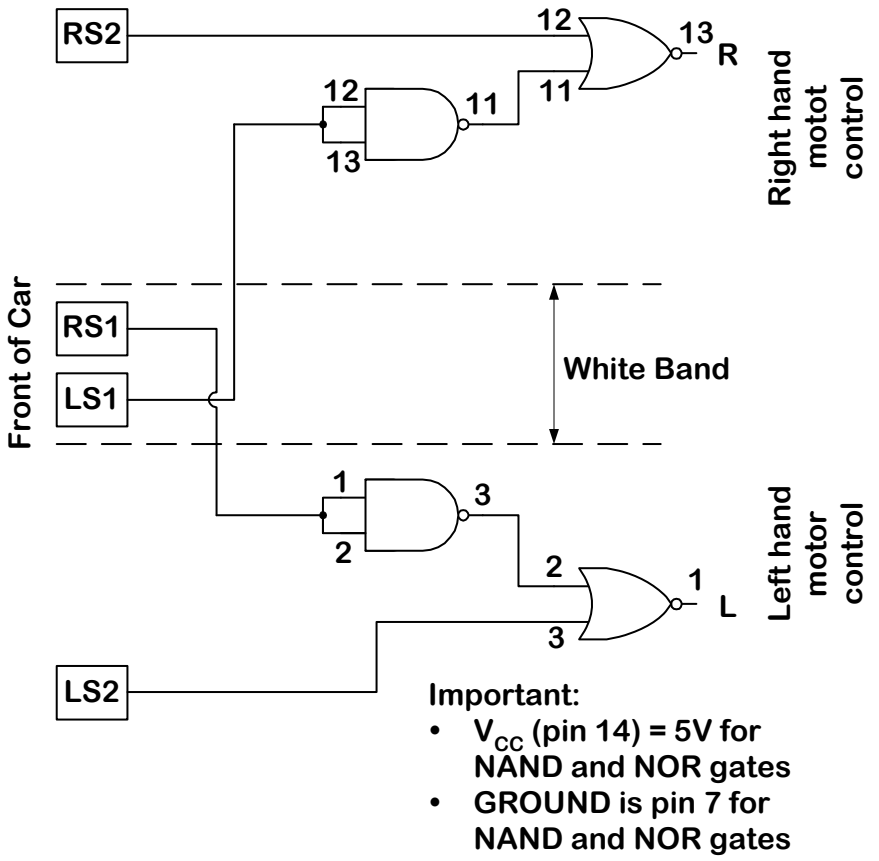
LS1,RS1 \ LS2,RS2	00	01	11	10
00				
01				
11				
10				

LEFT HAND MOTOR

LS1,RS1 \ LS2,RS2	00	01	11	10
00				
01				
11				
10				

RIGHT HAND MOTOR

g) Show that the circuit following page will implement this logic (do this either by showing the equivalence of the circuit to the Karnaugh maps above or using Boolean algebra to reduce the Karnaugh maps to a simple form).



Building the Circuit

You have been supplied with two modules, one of which is a quad 2 input NOR gate, the other is a quad two input NAND gate (that is, the NOR and NAND modules each contain 4 two input gates. The actual gates are in the black integrated circuit (IC) package. There are several other components on the modules. The small green components are light emitting diodes connected to the outputs of the gates, which glow when the output of the gate is high. The other components are resistors connected between the inputs and 0 Volts (logical '0'). In the implementation of logic circuits, it is important that unused inputs are given a definite value.

AFTER READING THIS SECTION (ALL THREE PAGES) CAREFULLY, construct the circuit shown above using the NOR and NAND modules supplied. ***The numbers on the circuit above refer to the pin numbers on the modules.***

NOTES:

1) ***WHEN YOU MAKE OR ALTER CIRCUITS, ENSURE THAT THE POWER FROM THE VEHICLE IS SWITCHED OFF - FAILURE TO DO SO CAN RESULT IS THE VEHICLE MOVING UNEXPECTEDLY OR DESTRUCTION OF MODULES.***

2) The pin-outs for the NOR and NAND modules are as shown in the figures on page 13.

3) ***YOU <u>MUST</u> CONNECT +5V TO PIN 14 OF THE NOR AND NAND MODULES, AND YOU MUST CONNECT 0V TO PIN 7 OF THE NOR AND NAND MODULES.*** These connections are not usually shown on logic circuits ***BUT ARE ALWAYS REQUIRED***.

4) The sensors are the small black cubes that plug into the underside of the sensor board at the front of the vehicle.

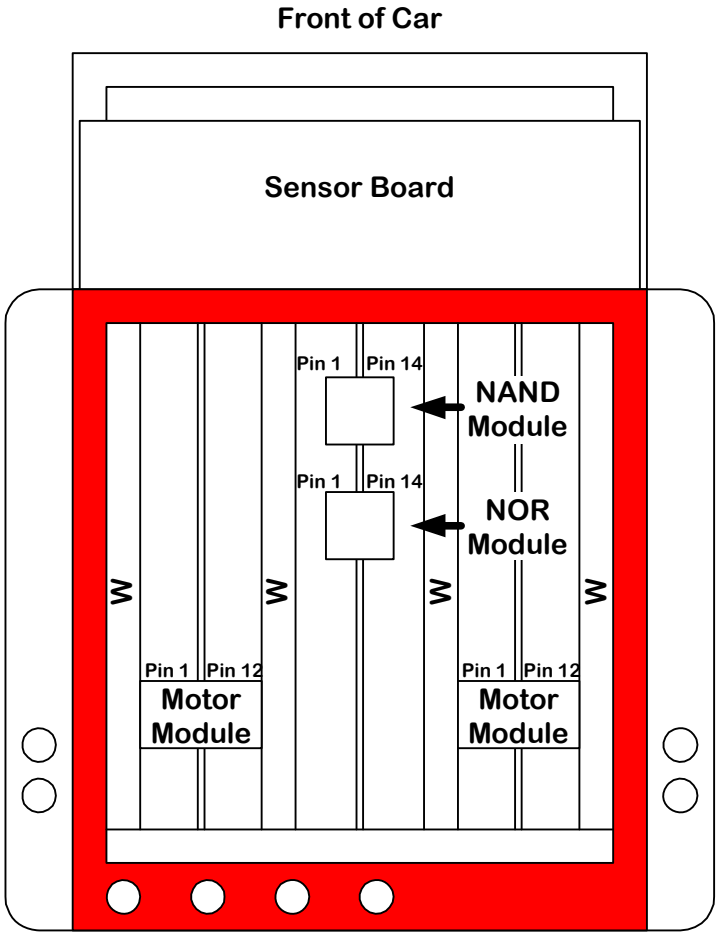
5) The correct orientation of the sensors is with the writing to the front of the vehicle, with the sensor completely within one of the numbered blocks of four holes in the connector.

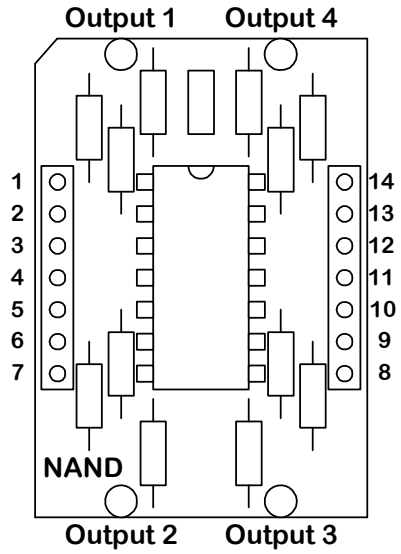
6) You can use any sensor positions you wish, but remember that LS1 and RS1 are to be the inner most sensors.

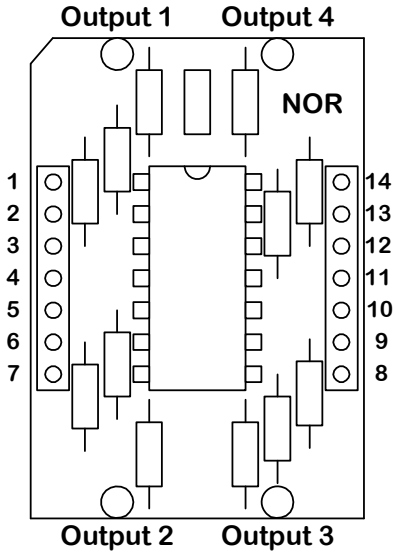
7) Orient the modules on the protoboard so that when you look down on the vehicle with the front of the car away from you, pin 1 is on the left hand side. Place the modules as shown in the diagram below.

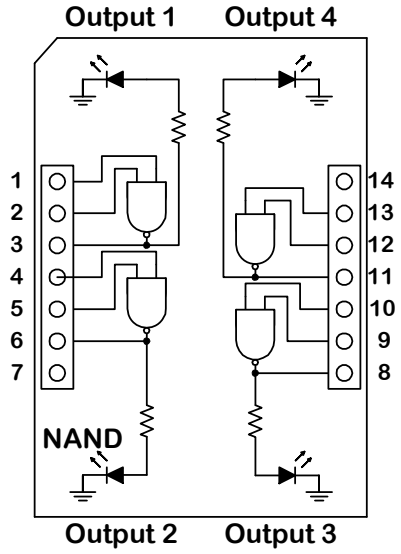
8) Use short wires and keep your wiring neat.

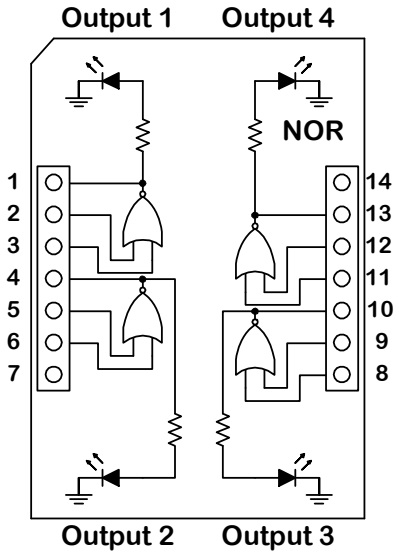
9) BEFORE TURNING THE POWER ON, ASK A DEMONSTRATOR TO CHECK YOUR CIRCUIT.











NAND MODULE

Remember - You *MUST* connect PIN 14 to VCC=5V and PIN 7 to GROUND (0V)

NOR MODULE

Remember - You *MUST* connect PIN 14 to VCC=5V and PIN 7 to GROUND (0V)

WHEN YOU MAKE CIRCUITS, ENSURE THAT THE POWER FROM THE VEHICLE IS SWITCHED OFF - FAILURE TO DO SO CAN RESULT IS THE VEHICLE MOVING UNEXPECTEDLY OR DESTRUCTION OF MODULES.

Measurements

1) By inputting all 16 possible combinations of input, complete the following Karnaugh maps. Note that you should not have any “DON’T CARES”, the

circuit will have a definite logic level output for all possible input combinations.

LS1,RS1 \ LS2,RS2	00	01	11	10
00				
01				
11				
10				

LEFT HAND MOTOR

LS1,RS1 \ LS2,RS2	00	01	11	10
00				
01				
11				
10				

RIGHT HAND MOTOR

2) Using one of the multimeters available to you set on **DC VOLTS**, measure the voltage between pin 13 and pin 7 of the NOR gate firstly when the pin 13 is a logical '1' and then when pin 13 is a logical '0'. Note the value in your laboratory notebook. **YOU MUST RECORD WHAT RANGE YOU USED FOR ALL MEASUREMENTS**.

3) Repeat the measurement for the voltage between pin 11 and pin 7 for the NOR gate when pin 11 is a logical '1' and when pin 11 is a logical '0'.

4) Repeat the measurement for the voltage between pin 12 and pin 7 for the NOR gate when pin 12 is a logical '1' and when pin 12 is a logical '0'.

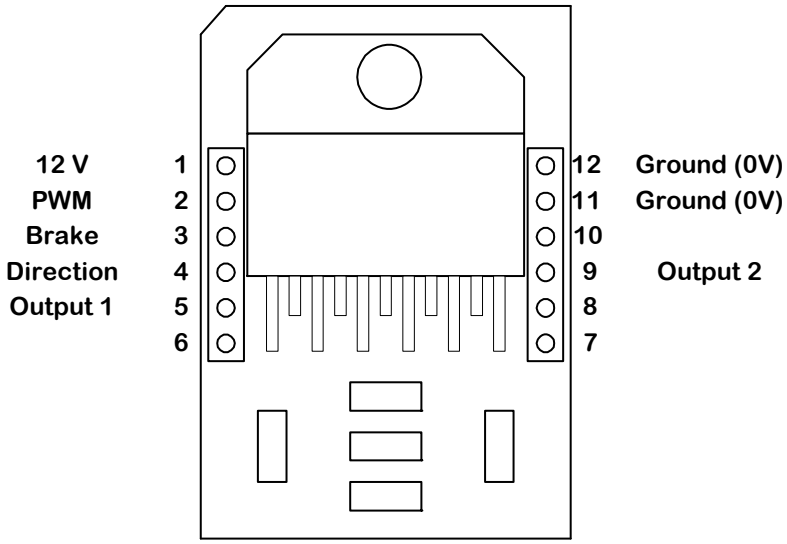
5) The two inputs of the NAND gates in this circuit are connected together. What is the logical relationship between the input to a NAND gate and the output of a NAND gate when it is connected this way?

DO THE FOLLOWING ONLY IF YOU ARE INTERESTED AND HAVE TIME

Attaching the Motors and Verifying Operation

To take the logic signal from the control logic you have constructed above, you need the use the motor modules. The motor modules operate from 12V and have three logic inputs. We will only use one of them in this experiment, the one labelled PWM (for Pulse Width Modulation). The other inputs are connected to a logical '0' via resistors. In the next laboratory, we will investigate the operation of these logic inputs.

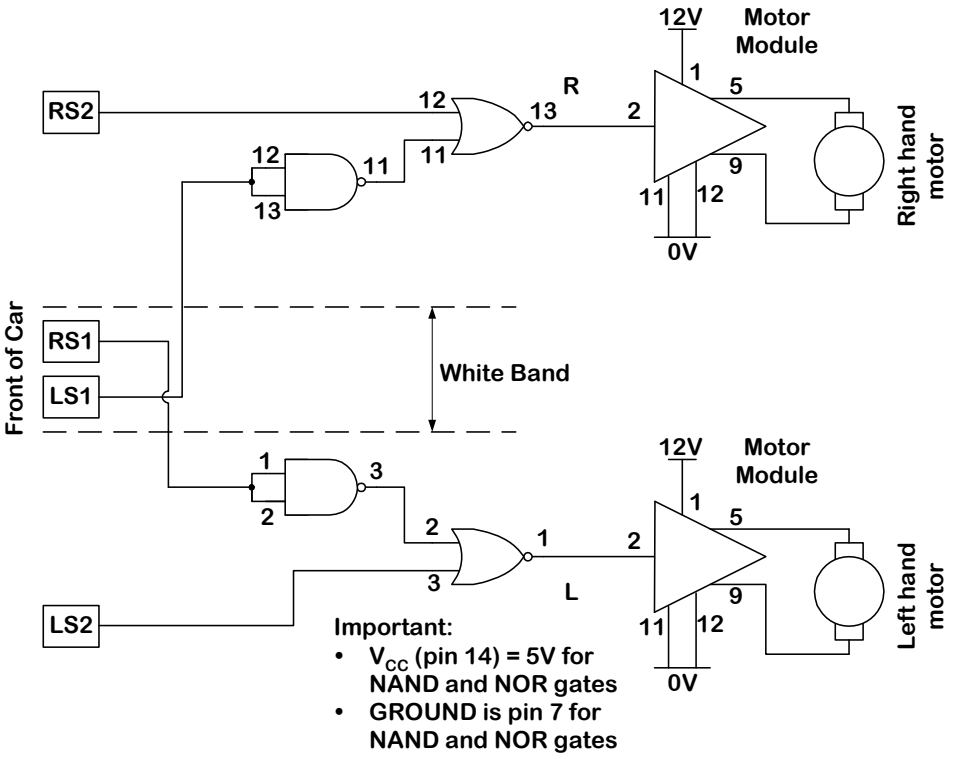
The motor modules are shown in the figure on the next page.



Motor Module

Modification of Your Circuit

- ***After reading all of the following instructions***, build the circuit show on the next page.
- ***Turn the power off at the back of the vehicle.***
- Pin 1 of the motor modules are connected to 12V (***note that all other circuits on the board work from 5V***).
- Pins 11 and 12 are BOTH connected to 0V.
- The motor terminals for the vehicle are the screw terminals on the side of the car (the WHITE and BLUE terminals, and the GREEN and YELLOW terminals). When you connect the motors to the motor modules determine the correct connections of OUTPUT 1 (Pin 5) and OUTPUT 2 (Pin 9) by trial and error - if you have the motors connected the wrong way around, the motor will run in the incorrect direction.
- When you have made the circuit, as the demonstrator to check the circuit.
- Once you are sure that your circuit is correct (if you are unsure, ask a demonstrator), turn the power on at the back of the car.
- Try your car on the track provided. You may have to move the sensors to get good results. You may also have to change the motor connections to give the correct direction of rotation. ***ALWAYS TURN OFF THE POWER WHEN MAKING CHANGES TO THE CIRCUIT***.


12V
Motor Module
RS2
R
Right hand motor
0V
Front of Car
RS1
LS1
White Band
L
Left hand motor
LS2
Important:
• V_{CC} (pin 14) = 5V for NAND and NOR gates
• GROUND is pin 7 for NAND and NOR gates

Experiment 2

Objective

- Examine some medium scale integrated circuits (multiplexers and comparators).
- To become aware of how pulse width modulation can be used to control speed of the motors.
- To examine the operation of the Dir and PWM controls on the motor driver modules.

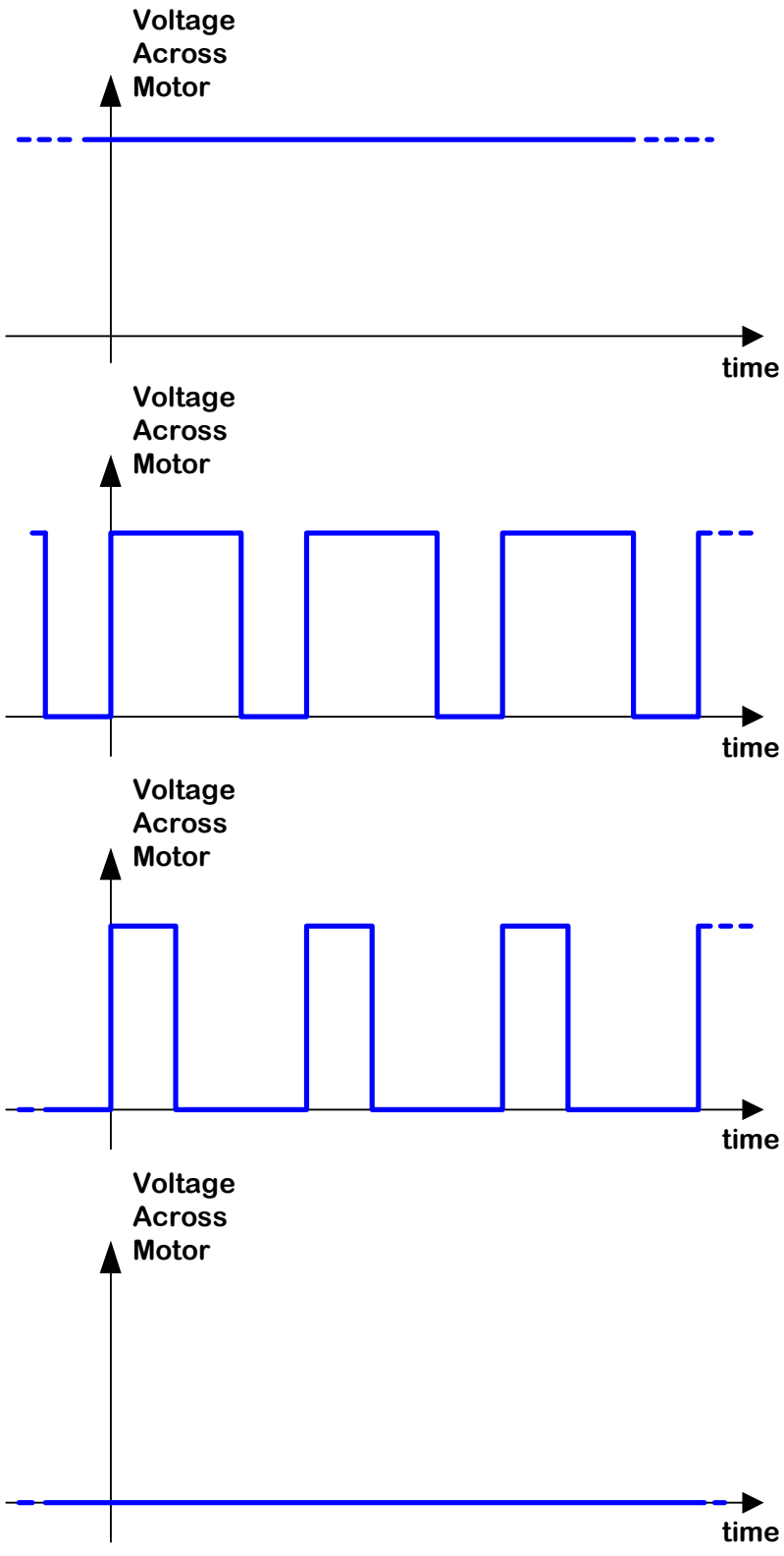
This laboratory will help you devise the strategy to control the vehicle using the microcontroller module to be used in Experiments 3, 4, and 5

Before coming to the first laboratory session ensure that you have read the safety notes on page 4 of this manual.

Background

The steering of the vehicle using the simple strategy developed in Experiment 1 is an ON-OFF (also called BANG-BANG) control system. As such, those of you who actually made the vehicle steer itself would have found that the vehicle motion was very erratic and "twitchy". To smooth the vehicle motion requires some form of speed control. In order to control the speed of a DC motor we can decrease the current through the motor using a resistor in series with the motor and the voltage source driving the motor. However, this has a serious side effect. The torque (the turning force) of the motor is related to the current through the motor. Decreasing the current using a series resistor decreases the torque dramatically so that while the motor might run slower, it can't actually do any work. An alternative approach (and one that is used in practice) is to keep full voltage across and current through the motor, but do so for only a short period of time and then turn off the supply. The motor then runs at a speed that is controlled by the ratio of the time that the voltage is applied to the time that the voltage across the motor is zero. If the voltage is applied for 100% of the time, the motor runs at full speed. If the voltage is only applied 2/3 of the time, the motor runs at approximately 2/3 full speed, etc. This concept is shown in the figure on the next page, and is called Pulse Width Modulation (PWM). It is a common concept in control applications where an output can be related to the average value of a periodic waveform.

The ratio of the time that the waveform is greater than zero (a MARK) to the time that the waveform is zero (a SPACE) is called the MARK SPACE RATIO. A mark space ratio of 1:1 implies that the waveform is greater than zero for 50% of the time.


Voltage Across Motor
time
Motor at maximum speed
Voltage Across Motor
time
Motor at 2/3 speed
Voltage Across Motor
time
Motor at 1/3 speed
Voltage Across Motor
time
Motor stopped

Pulse Width Modulator Module

The pulse width modulator module includes an oscillator, a four bit binary counter and a four bit binary magnitude comparator.

Oscillator (74LS629): The oscillator simply generates a periodic square wave (pin 7 of the 74LS629 IC - ***NOTE: NOT PIN 7 OF THE MODULE***). This is used as a "CLOCK" input to the counter.

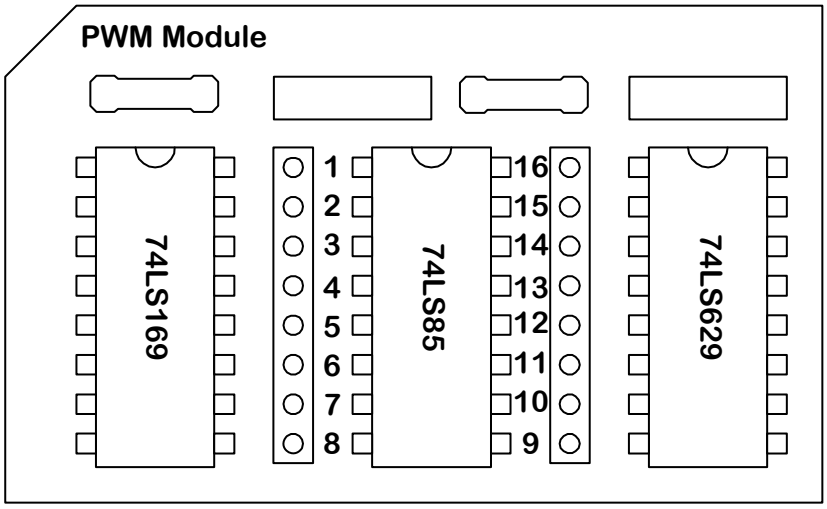
Counter (74LS169): This chip is a four-bit binary counter. It is designed to count, in binary, the number of logic transitions from '0' to '1' that occur on its "CLOCK" input (pin 2 of the 74LS169 IC - ***NOTE: NOT PIN 2 OF THE MODULE***). The output is four binary signals representing the binary count of the number of pulses counted. Since the output is only four bits wide, it actually counts modulo 16. That it is it counts 0, 1, 2, 3, …, 14, 15, 0, 1, 2, … (or 0000, 0001, 0010, 0011, …, 1110, 1111, 0000, 0001, 0010,…). Outputs from the counter called Q0, the least significant bit, Q1, Q2, and Q3, the most significant bit. (These are pins 14, 13, 12, and 11 of the 74LS169 IC respectively. ***NOTE: THESE ARE NOT THE MODULE PINS***). The outputs from the counter are used as one of the four bit binary numbers that are inputs to the magnitude comparator.

Magnitude Comparator (74LS85): This chip compares the magnitudes of two four-bit binary numbers, A and B. It has three outputs A>B, A=B, A<B (ie if A > B, then the output A>B will be '1' and the outputs A=B and A<B will be '0', etc). The input B comes from the output of the counter chip, the input A (A0, A1, A2, A3) comes from the module pins (pins 10, 12, 13, and 15 ***OF THE MODULE*** respectively).

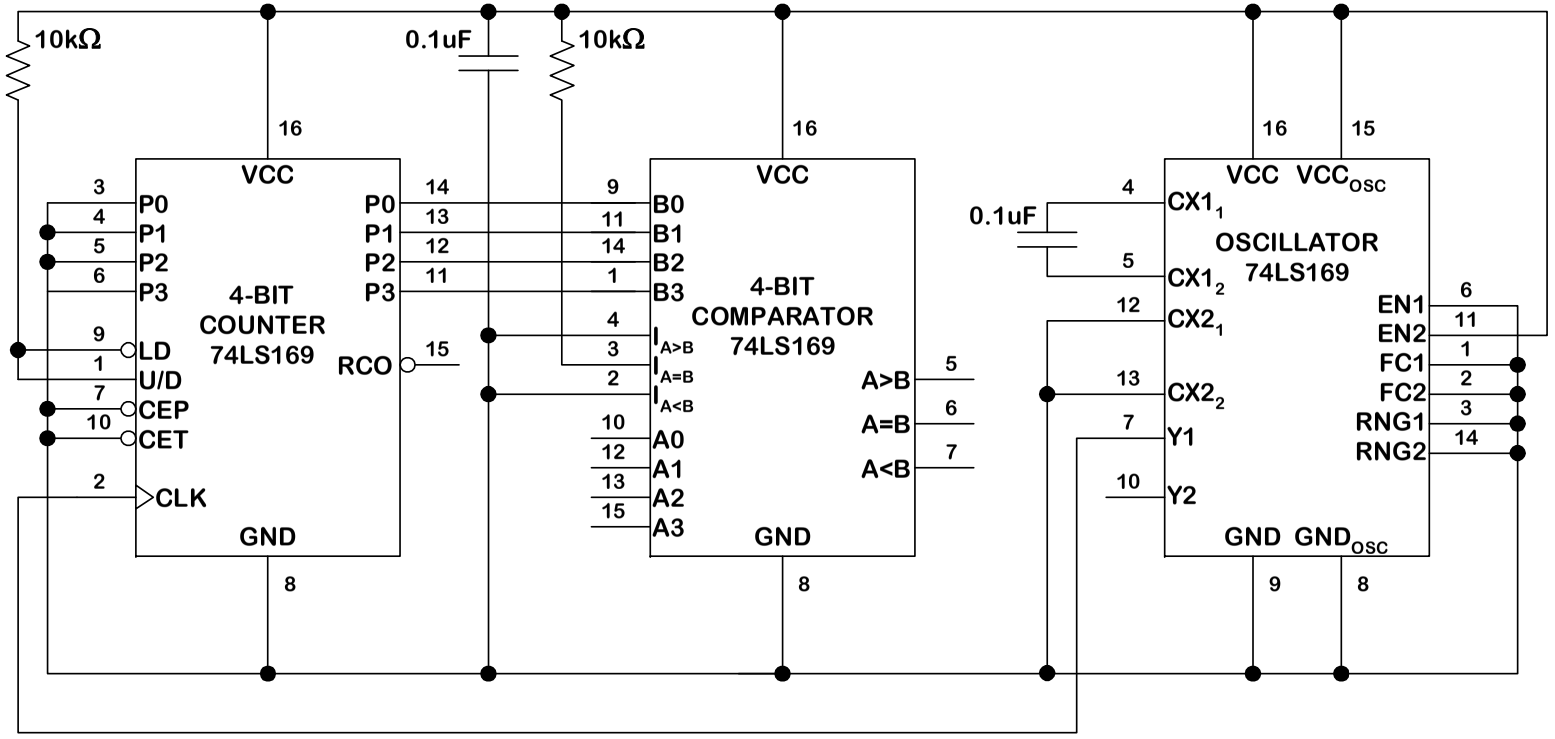
The layout for the PWM module is shown below and the complete circuit is shown on the next page.

NOTE: A_0 ***= Least Significant Bit (LSB)***

A_3 ***= Most Significant Bit (MSB)***



PWM Module

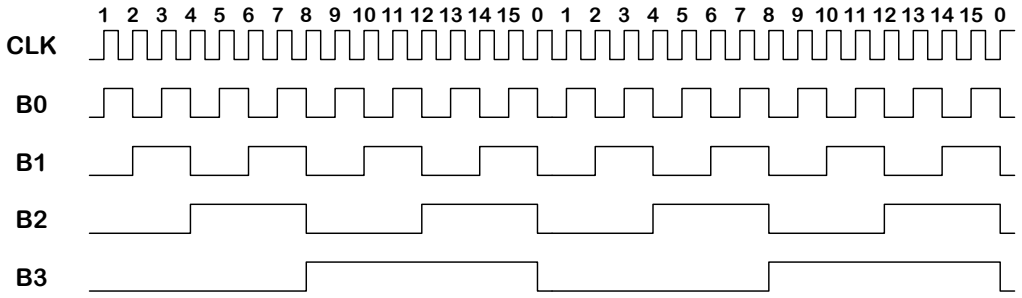


Notes:
Pin 16 of the module must be connected to 5V
Pin 8 of the module must be connected to 0V

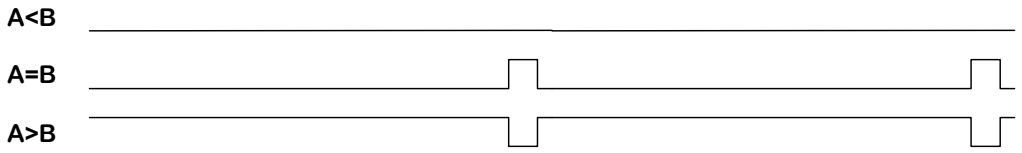
A inputs:
- **A0 Pin 10 of the module**
- **A1 Pin 12 of the module**
- **A2 Pin 13 of the module**
- **A3 Pin 15 of the module**

Outputs:
- **A>B Pin 5 of the module**
- **A=B Pin 6 of the module**
- **A<B Pin 7 of the module**

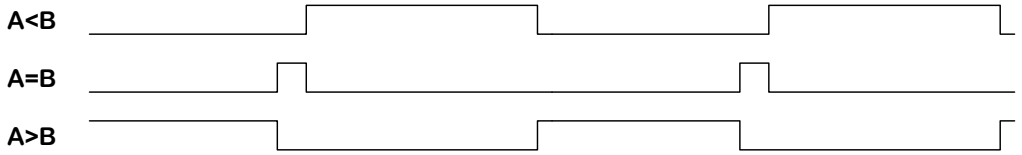
The diagram below shows how the module should behave for A=15(1111), 7(0111), 3(0011), 1(0001) and 0(0000).



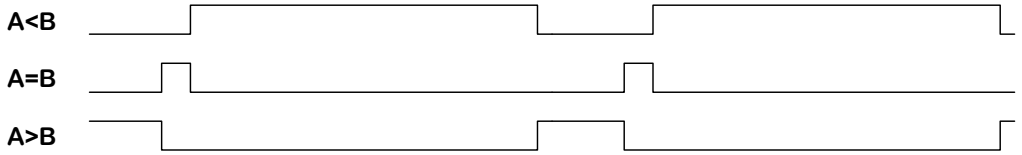
Output for an input A = 15
A0=1, A1=1, A2=1, A3=1



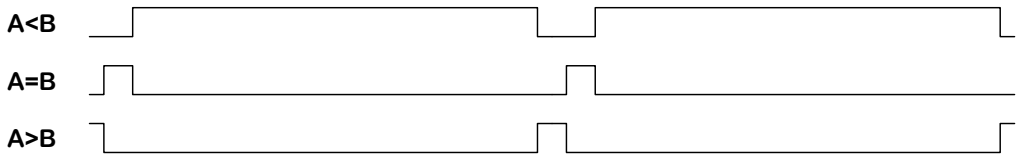
Output for an input A = 7
A0=1, A1=1, A2=1, A3=0



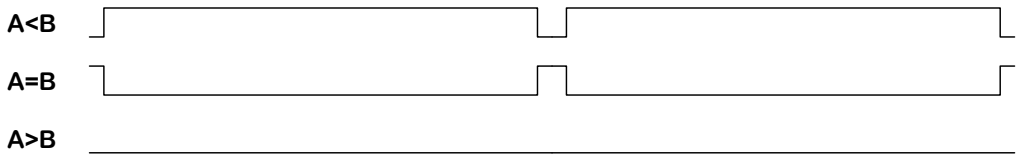
Output for an input A = 3
A0=1, A1=1, A2=0, A3=0



Output for an input A = 1
A0=1, A1=0, A2=0, A3=0



Output for an input A = 0
A0=0, A1=0, A2=0, A3=0



MUX Module

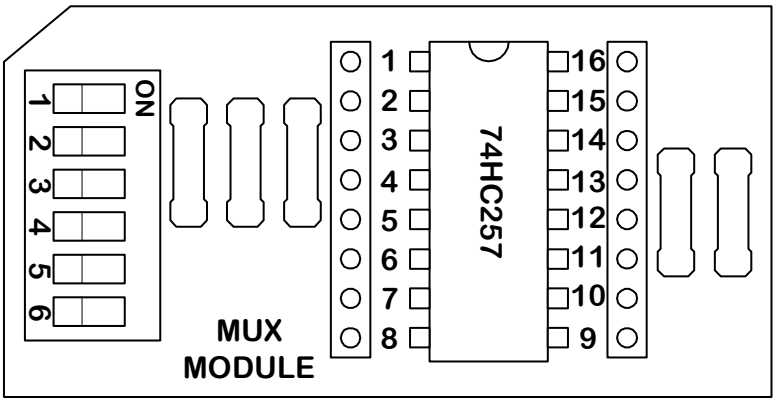
A multiplexer (mux) is a switch that selects which of a number of set of inputs is directed to the output. The MUX module supplied has four such multiplexers in one chip (74AC257) that select which of two inputs (I_0 or I_1) are directed to an output (Y) under the control of a logic input SEL. All four multiplexers share the same select input (that is, if the SEL input is '1' then $Y_A = I_{1A}$, $Y_B = I_{1B}$, $Y_C = I_{1C}$, and $Y_D = I_{1D}$, whereas if the SEL input is '0' then $Y_A = I_{0A}$, $Y_B = I_{0B}$, $Y_C = I_{0C}$, and $Y_D = I_{0D}$).

On the MUX module supplied to you, the inputs I_{1A}, I_{1B}, I_{1C} and I_{1D} are connected to switches 3, 4, 5 and 6 respectively, while the SEL input is connected to switch 0. ***When the switch is ON the input to the chip is a logic '0'.*** The inputs I_{0A}, I_{0B}, I_{0C} and I_{0D} are connected to pins 2, 5, 11 and 14 ***OF THE MODULE*** respectively. These inputs can be used for connecting to the sensors. Using the SEL input (switch 1), you can then select whether to use the sensors as input (SEL = '0', switch 1 ON), or switches 3, 4, 5, and 6 (SEL = '1', switch 1 OFF). The outputs Y_A, Y_B, Y_C and Y_D are connected to pins 4, 7, 9 and 12 respectively.

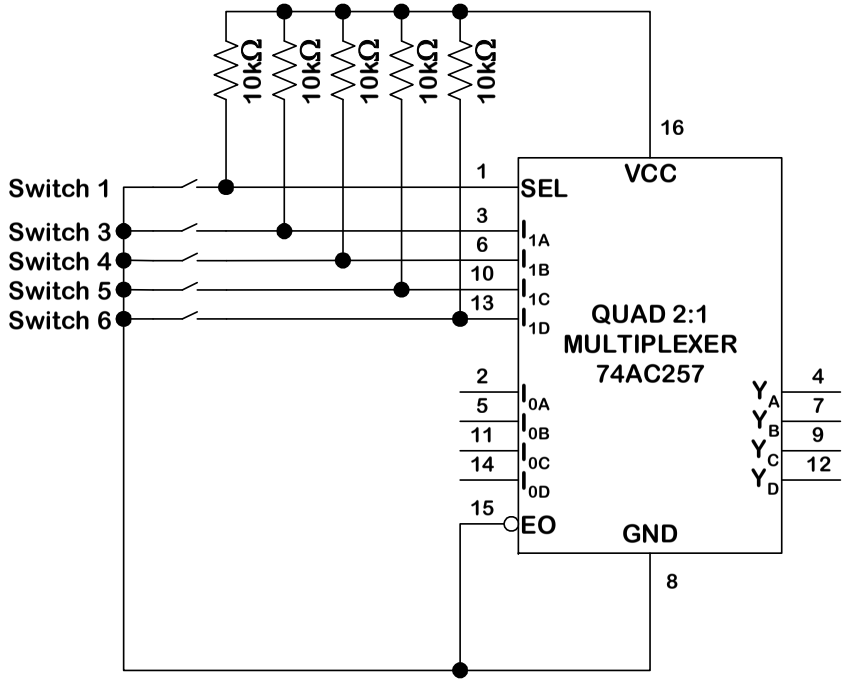
The module outline is shown below and the complete circuit is shown on the next page.

NOTE: ***Y_A = Least Significant Bit (LSB)***

Y_D = Most Significant Bit (MSB)



MUX Module

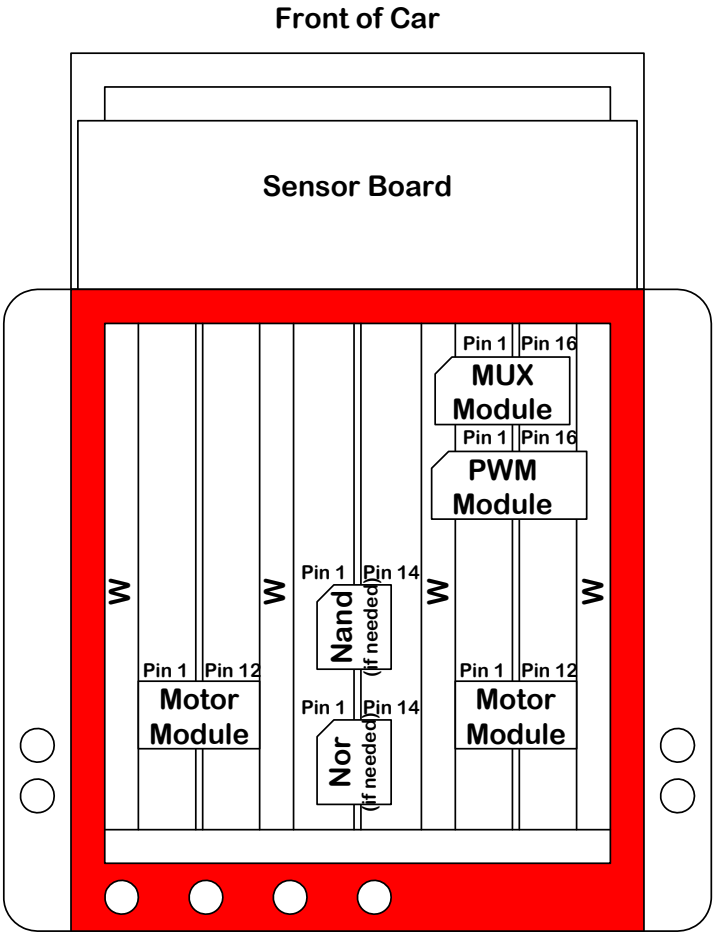


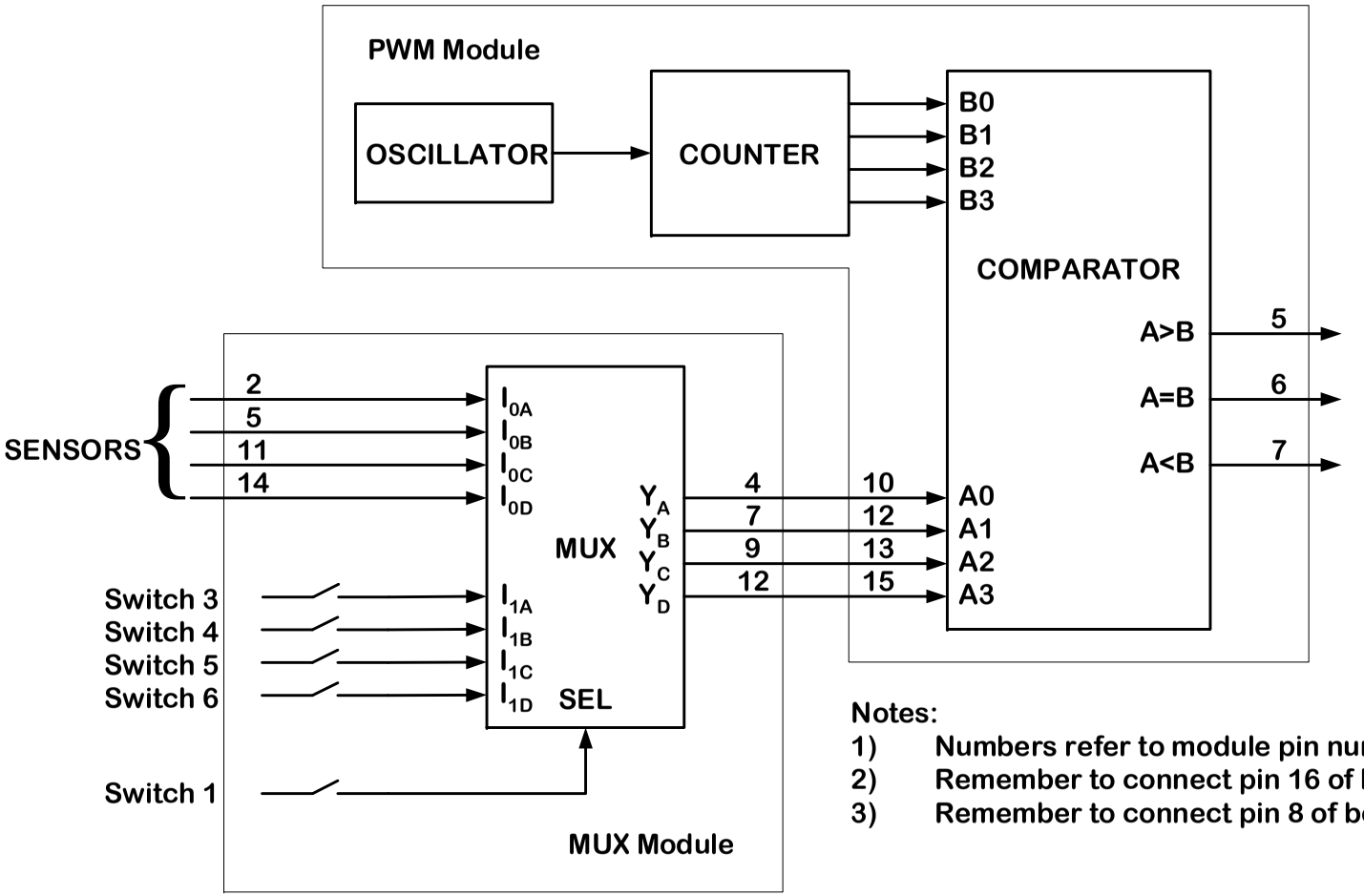
Notes:
Pin 16 of the module must be connected to 5V
Pin 8 of the module must be connected to 0V

I_0 inputs:	I_{0A}	Pin 2 of the module	Outputs:	Y_A	Pin 4 of the module
	I_{0B}	Pin 5 of the module		Y_B	Pin 7 of the module
	I_{0C}	Pin 11 of the module		Y_C	Pin 9 of the module
	I_{0D}	Pin 14 of the module		Y_D	Pin 12 of the module

Connecting the MUX Module and PWM Module

After reading ***ALL OF THE FOLLOWING INSTRUCTIONS***, construct the circuit shown on the following pages using the protoboard. It is strongly recommended that you use the suggested layout, particularly if you want to get the vehicle to steer. Remember to supply 5V to pin 16 and 0V to pin 8 of the MUX and PWM modules. You don't have to connect the sensors yet. You can also leave the Nand, Nor and Motor Modules off of the protoboard at this stage. ***ENSURE THAT THE POWER IS SWITCHED OFF WHILE YOU ARE CONSTRUCTING THE CIRCUIT OR MAKING CHANGES. ASK A DEMONSTRATOR TO CHECK YOUR WIRING BEFORE TURNING ON THE POWER.***





Notes:

1) Numbers refer to module pin numbers
2) Remember to connect pin 16 of both modules to 5V
3) Remember to connect pin 8 of both modules to 0V

Measurements

a) Set the select switch (switch 1) on the MUX module to OFF. This selects switches 3, 4, 5, and 6 as inputs to the MUX chip.

b) Using the oscilloscope and probes, examine the voltage waveform between 0V and the PWM module output A<B, 0V and the PWM module output A=B, and 0V and the PWM module output A>B as a function of the position of MUX module switches 3, 4, 5, and 6. To do this, connect the crocodile clip of the probe to pin 8 of the PWM module and connect the probe to pin 5 for A>B, pin 6 for A=B and pin 7 for A<B. Fill in the table shown below. The time high refers to the amount of time the particular output is high during one period of the waveform. You can use the measurement features of the oscilloscope to help make these measurements. ***Remember, the switches are connected so that when a switch is ON, the corresponding input to the MUX module is '0'. In addition, remember that the A_0 (connected to Y_A of the MUX) is the least significant digit, while A_3 (connected to Y_D of the MUX) is the most significant digit.*** (Therefore, to set A = 1010, switch 3 is ON, switch 4 is OFF, switch 5 is ON, and switch 6 is OFF.)

A (set by switches on MUX Module)	A<B Time high	A=B Time high	A>B Time high
0000			
0001			
0011			
0111			
1000			
1100			
1110			
1111			

c) ***TURN THE POWER OFF ON THE VEHICLE*** and then construct the circuit shown on the next page. This simply connects the motor driver modules to the MUX and the PWM modules. ***REMEMBER TO CONNECT*** 0V to pins 11 and 12 of the motor driver modules and 12V to pin 1 of the motor driver modules. Set switches 3, 4, 5, and 6 on the MUX module to ON. When you have finished wiring the vehicle and are sure you have wired it correctly (***ASK A DEMONSTRATOR IF YOU ARE UNSURE***), place the vehicle on a stand and turn on the power. If the motor is not running, you have made an error in your wiring. (If this is the case, turn the power off and check your circuit.)

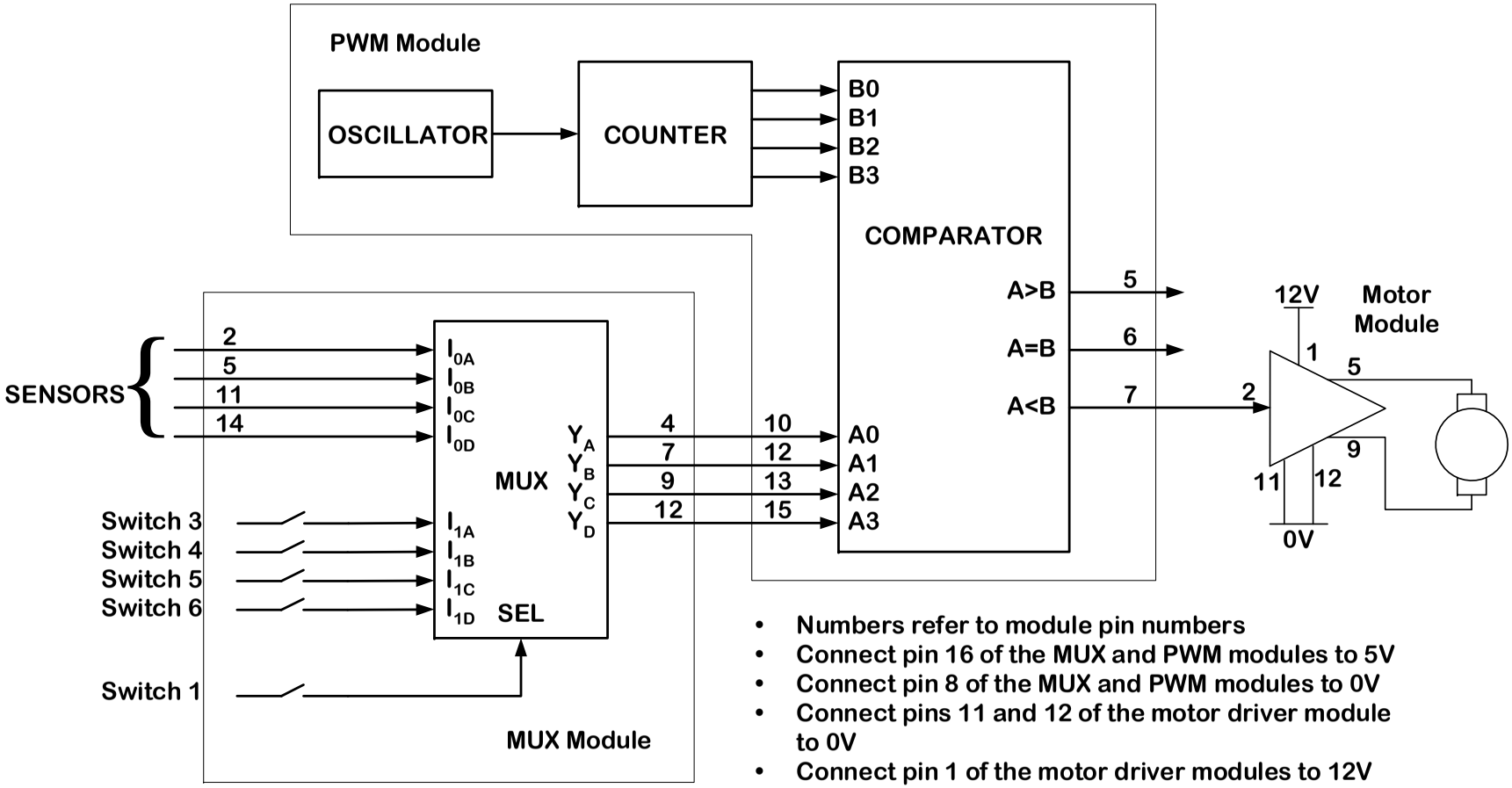
d) For the following switch positions, note the response of the motor and explain what is happening.

Switch 6	Switch 5	Switch 4	Switch 3	Response of motor
ON	ON	ON	ON	
ON	ON	ON	OFF	
ON	ON	OFF	OFF	
ON	OFF	OFF	OFF	
OFF	OFF	OFF	OFF	

e) Turn the vehicle power off and disconnect the PWM input to the motor driver module (pin 2 of the motor driver module) from the A<B output of the PWM module (pin 7 of the PWM module). Connect the A<B output of the PWM module to the DIR input of the motor driver module (pin 4 of the motor driver module). Connect pin 2 (PWM input) of the motor driver module to 5V. When you are sure that you have wired the circuit correctly (***ASK A DEMONSTRATOR IF YOU ARE UNSURE***), place the vehicle on a stand and turn on the power. Repeat part d) for the circuit in this configuration.

f) Using your observations in parts (d) and (e), devise a control strategy for making the vehicle smoothly follow a white band on a black background using PWM. You have available two MUX modules, two PWM modules, two motor control modules (one for each for each motor) and the NAND and NOR modules if needed.

g) IF TIME PERMITS AND YOU ARE INTERESTED, IMPLEMENT YOUR CONTROL STRATEGY.



- Numbers refer to module pin numbers
- Connect pin 16 of the MUX and PWM modules to 5V
- Connect pin 8 of the MUX and PWM modules to 0V
- Connect pins 11 and 12 of the motor driver module to 0V
- Connect pin 1 of the motor driver modules to 12V

Experiment 3

Objectives

- Learn to use microcontroller development tools, including the assembler and emulator/debugging tools
- Write a simple assembly program

Outline

In this lab, we will use the AVR Butterfly to drive the car around the track using a constant wheel speed and stopping at the white line. To begin, ignore the requirement that the car must stop at the white line. Later, we will build that in.

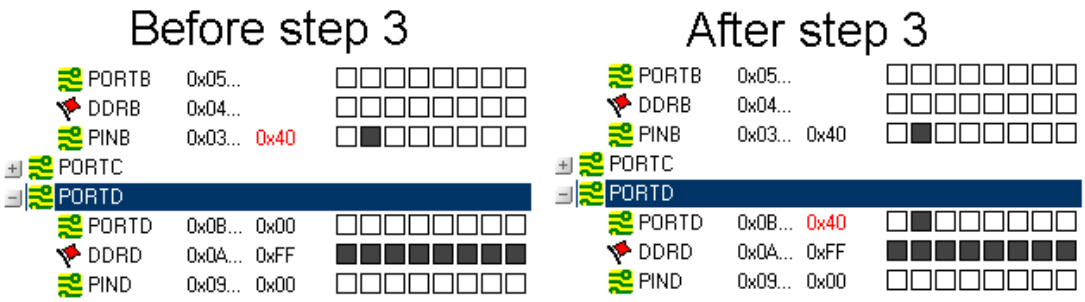
There are two sensors and two wheels. If the left sensor is over the white track then the right wheel turns on. Similarly, if the right sensor is over the white track then the left wheel turns on. The sensors should be regarded as independently controlling their respective wheels.

Setup

Working program code is being provided. Download all sample project files from the unit webpage (student.ee.uwa.edu.au/units/elec1300) and run it on the simulator as follows:

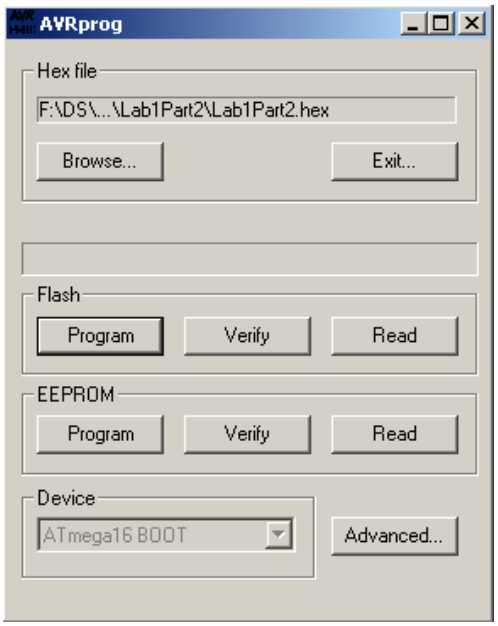
1. Start AVR studio
2. Click 'open' on the Welcome screen, click on Lab1Part1.aps and then 'open'.
3. Press F7 to build the project.
4. To simulate, click on 'debug' and then 'start debugging'

Press F10 to single step through the code. Just before executing step 3, manually set pin 6 of PINB to 1 and it turns black. You can do this by clicking on the square. If everything has been done correctly, you should see something like this:



To run the car on the white track, use the following steps. These steps will also be demonstrated for you during the lab session.

1. Place two sensors on the sensor array. Remember, that the writing should be towards the front and the sensor should fit nicely the square.
2. Connect the left sensor to PA2 and the right sensor to PA1.
3. Connect the left wheel to PP1 and the right wheel to PP2.
4. Connect up the battery with the banana leads.
5. Connect the serial cable from the PC to the car.
6. Turn on the car and hold the small black joystick down.
7. Whilst holding the small black joystick down, click on 'tools' in the menu bar, then 'avr prog'. A screen should appear like below:



8. Release the small black joystick and click on the 'browse' button.

   Click on the hex file named Lab1Part1.hex and then press 'open'.
9. In the flash box, click 'Program'. Please do not confuse this with the EEPROM box's 'Program'.
10. Once the programming has been verified as OK, close the screen.
11. Turn off the car and turn it back on.
12. Move the small joystick up. Your program should now be running.

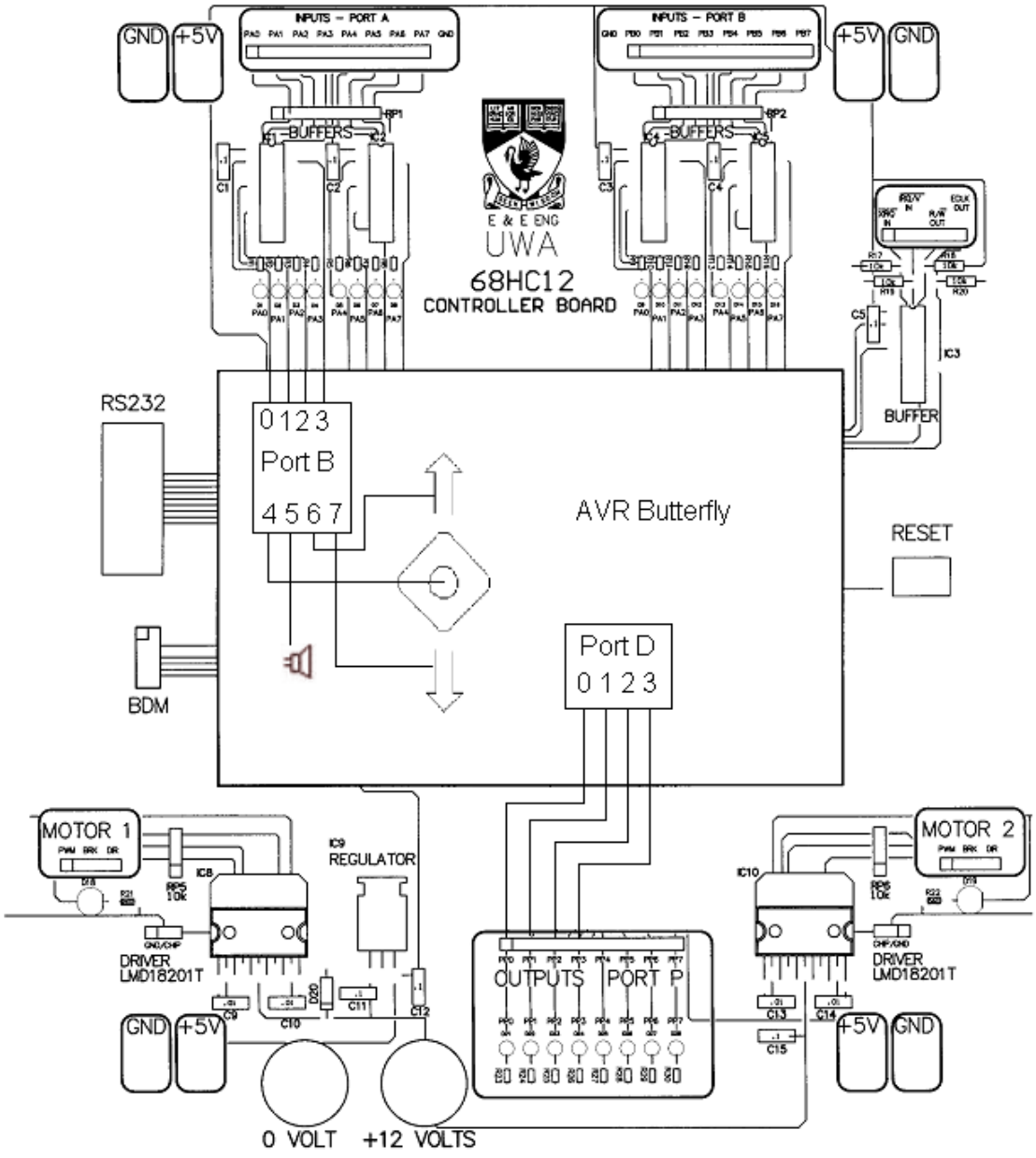
If everything was done correctly, the car should follow the white track.

Introductory Self-Test (not assessable)

The code is reproduced below:

```
.include "m169def.inc"

;1. Set the data direction register for port B to inputs.
;   Recall that the data direction register is called DDRB.
;   Also, 0 is for input and 1 is for output.

main:     LDI  R16, 0x00
          OUT DDRB, R16

;2. Set the data direction register for port D to outputs.
;   The data direction register for port D is called DDRD.

          LDI R16, 0xFF
          OUT DDRD, R16

;3. Read the inputs of port B and store them into port D.
;   Remember that when reading from port B, PINB must be used
;   and when writing to port D, PORTD must be used.

repeat:   IN   R16, PINB
          OUT PORTD, R16

;4. Go back to step 3.

          JMP repeat
```

The schematics are drawn below:



The schematics indicate that the sensors on PA0, PA1, PA2 and PA3 are connected to Port B bits 0, 1, 2 and 3 respectively. Bit 4 of Port B goes low if joystick is pressed, but is high otherwise. Bit 6 of Port B is low when the user moves the joystick up. Bit 7 of Port B is low when the user moves the joystick down. Bit 5 controls the speaker. Port D controls what goes onto the output port P. Bits of Port P are then connected to the motors.

Please make sure that you understand how the code works and the wiring. Try answering the following questions, which are ***not*** being marked. Compare your answers with the suggested answers on the next page. Please put up your hand and ask questions if you are stuck. You will find it very difficult to do part 2 if you don't understand the code or wiring.

1. Why is the left sensor connected to PA2 and the right sensor to PA1? Why should the left sensor ***not*** be connected to PA1 and the right Sensor ***not*** connected to PA2?

2. How is the sensor data retrieved?

3. Where is the sensor data stored once it has been retrieved?

4. How does the CPU control which wheel goes on or off?

5. What happens when you set DDRB to 0?

6. What happens when you set DDRD to 0xFF?

7. Will the program ever terminate? Why or why not?

Introductory Self-Test Answers

1. The right sensor should control the left wheel and the left sensor should control the right wheel. Given that the program simply transfers the data on PortA (the sensor data) to PortP(the wheels), this wiring is necessary to implement that strategy.

2. The sensor data is retrieved using IN R16, PINB. This transfers the data on PINB (the sensor data on PortA) to the register R16.

3. The sensor data is stored in register R16.

4. The CPU controls the wheels by setting the value given to PORTD. The value on PORTD sets the output pins on PORTP.

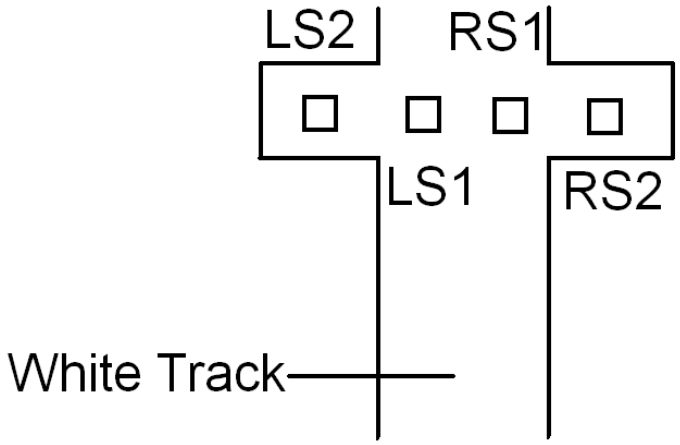
5. PORTB becomes an input port

6. PORTD becomes an output port.

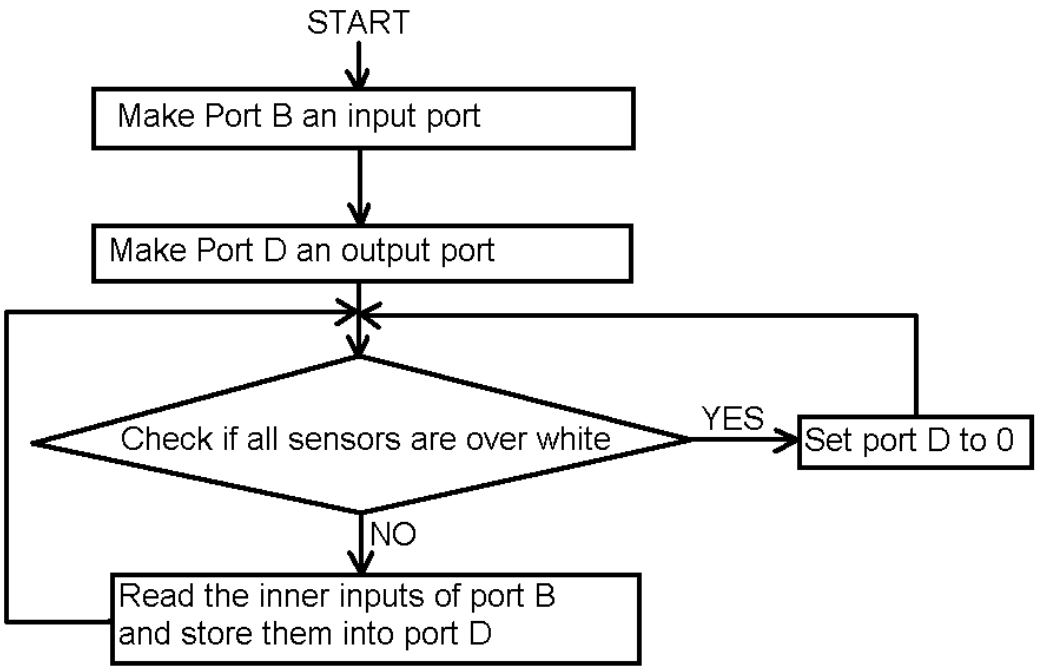
7. The program will never finish "naturally" because of the JMP instruction at the end (endless loop).

Lab Exercise

Include the requirement that the car must stop at a white line if all four sensors (LS2, LS1, RS1 and RS2) detect white. See the diagram below:



Obviously, this additional requirement means the code from part 1 needs to change. Write the assembly instruction(s) to implement the following flowchart:



Hints

- Make sure that you use an ‘and mask’ to filter out the unused bits of the sensor data stored in port B before you perform the check.
- The sensor pins are referred to in reverse in your program. For example, 0x02 refers to PA1 – the second leftmost pin on PortA.
- Simulate your code before trying it on the car. On the simulator, you can see exactly what is going on step by step. If it doesn’t work on the simulator, it won’t work on the car.

Questions

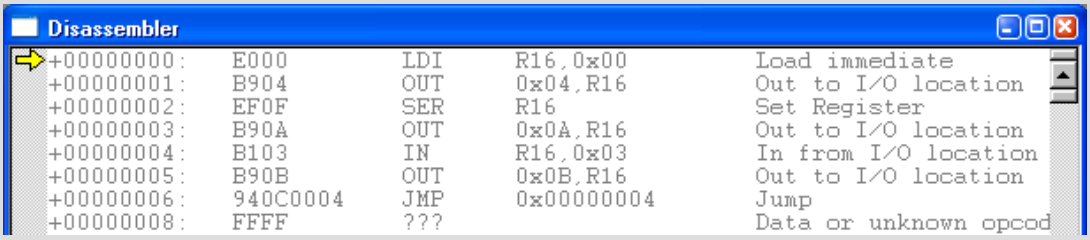
Please make sure that you understand how your code works. After the demonstration, questions will be asked about it that you or your partner need to answer orally. Some typical, but not necessarily tested, questions might be:

- Why have you inserted a compare instruction before the branching instruction?
- How did you get the car to stop?
- Explain why an ‘and mask’ needs to be used before comparing.
- Explain why you are comparing that register to that immediate value.
- Will you program ever terminate without user intervention? Why or why not?
- How does your wiring help to implement the control strategy?

Knowledge Test (not assessable)

The following is optional but highly recommended. If you want to gain a deeper understanding of what is going on, read further. Answers are on the next page.

Restart AVR and click on open project. Then choose the '.hex' file that was created on compilation of my code. This is typically in the 'default' subdirectory. You should see the disassembly which looks something like this:



The left column represents the memory address of the instructions. So, the load immediate instruction is at 0 in memory. The out instruction is at 1 in memory and so forth. Note that one instruction does not necessarily have to occupy only one address.

Each memory address can store 16 bits. This can be seen from the next column. For instance, B90A is 4 hex digits. Each hex digit represents 4 bits, so there must be 16 bits in total. 16 bits is equivalent to 2 bytes.

1. How many bytes are required for the JMP instruction?

The next column represents the opcode and operand which correspond to that assembly instruction. Recall from your lectures that the opcode represents what the instruction is doing. The operands, if any, are the parameters for the instruction.

Let's have a look at the first instruction LDI R16, 0x00. The documentation shows:

	Syntax:	Operands:	Program Counter:
(i)	LDI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

The leftmost 4 bits are 1110, which is E, and this is the opcode. The next 4 bits represent the most significant bits of the K operand. In this case, the K operand is 0, so the upper most 4 bits are 0. The next 4 bits represent the register which the K operand is to be stored. In this case it's R16, and given that the registers can only be between 16 and 31, we

expect that to be 0 as well. This is followed by the least significant 4 bits of the K operand which are 0. Hence, the opcode and operand is E000. This is shown in the disassembly.

2. What is the opcode for the SER instruction?

3. What are the operand(s) for the SER instruction?

	Syntax:	Operands:	Program Counter:
(i)	SER Rd	$16 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1110	1111	dddd	1111

You should also note that all the ports have been replaced by their respective memory addresses. For instance OUT PortD, R16 has been replaced by OUT 0x0B, R16.

4. What is the memory address of DDRB?

5. What is the memory address of DDRD?

Finally, we might be interested in how long it takes to execute an instruction. To do this, note that the processor is running at 16 MHz. This means it has 16 million clock cycles per second. For the LDI instruction, the documentation states that it takes 1 cycle. So, it would take 6.25×10^{-8} s to execute.

6. How long does it take to execute the JMP instruction?

7. Would it have been faster to use the RJMP instruction?

8. Would it have saved more memory to have used the RJMP instruction?

9. If you answered yes to both 7 and 8, why isn't the JMP instruction obsolete?

Answers to Knowledge Test

1. 4 bytes. One byte for each pair of hexadecimal digits.

2. The opcode is EFxF. The x represents the operand.

3. The operand is 0, representing the R16 register.

4. DDRB's address is 0x04.

5. DDRD's address is 0x0A.

6. 1.875×10^{-7} s.

7. Yes

8. Yes

9. Relative jumping only allows jumping within a preset range from the jumping instruction. However, jumping allows unlimited range.

Experiment 4

Objectives

- Create a rectangle signal
- Handle user input from the joystick.

Outline

In this lab, we will control the sound produced by the CPU using the joystick. To begin, some assembly code has been written to demonstrate how to make a sound and to read input from the joystick. Download these files, and run them on the Butterflies.

The code which makes a sound has a pulse width modulation ratio of 1:1. That is, the time the signal is high is equal to the time the signal is low. While the pulse ratio is always **constant,** the period time of the signal (and hence its frequency) may change. Changing frequency is what makes us perceive a different pitch of sound. Figure 1 illustrates this concept.

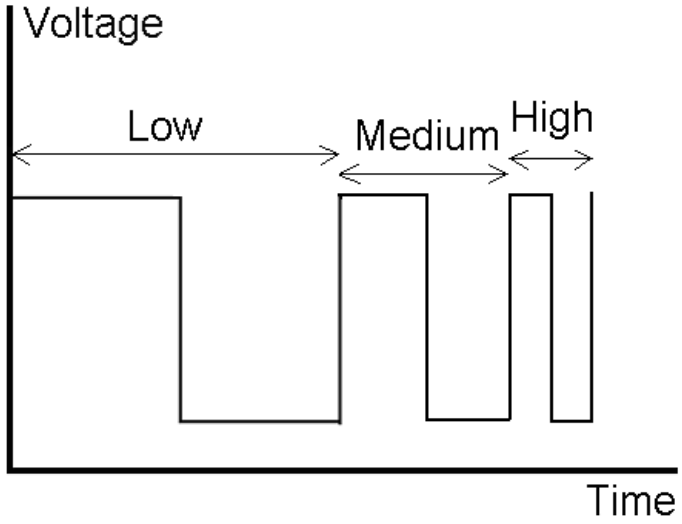


Figure 1: The high time equals the low time for all frequencies

Building Blocks

Part 1 - Sound

```
.include "m169def.inc"

;Set the speaker as output in the DDR.
main:          LDI  R16, 0x20
               OUT DDRB, R16

;Initialize the stack pointer
               LDI  R16,0x04
               OUT SPH, R16
               LDI R16, 0xFF
               OUT SPL, R16

;R20 is what controls the frequency of the sound produced
               LDI R20, 0x80

;The repeat loop generates a PWM signal.
;The frequency is changed by the time waiting in mywait.
repeat:        IN R16, PORTB
               ORI R16, 0x20
               OUT PORTB, R16

               CALL mywait

               IN R16, PORTB
               ANDI R16, 0xDF
               OUT PORTB, R16

               CALL mywait
               JMP repeat

;The mywait loop waits for a variable length of time
mywait:        MOV R17, R20
               LDI R18, 0xDB
               LDI R19, 0x64
iterate:       MUL R18, R19
               MUL R18, R17
               MUL R18, R19
               MUL R18, R17
               MUL R18, R19
               MUL R18, R17
               MUL R18, R19
               MUL R18, R17
               DEC R17
               BRNE iterate
               RET
```

Part 2 - Joystick

Source code is provided that illustrates the joystick use by turning on LEDs on PORTD. Pressing up with set D2 to zero (= PP2 off). Pressing down will set D3 to zero (=PP3 off). Pressing the joystick center will set D0 to zero (= PP0 off).

```
.include "m169def.inc"

main:           LDI  R16, 0x00
                OUT DDRB, R16

                LDI R16, 0xFF
                OUT DDRD, R16

repeat:         IN   R16, PINB
                LSR R16
                LSR R16
                LSR R16
                LSR R16
                OUT PORTD, R16

                JMP repeat
```

Introductory Self-Test (not assessable)

Make sure that you understand how both pieces of code work. Feel free to ask the lab demonstrator if you don't. You should be able to answer the following questions. These are ***not*** assessable. Compare your answers to the solutions on the next page:

For the sound code:

1. Which output pin controls the speaker?
   Give the port name and bit number.

2. a) Which instructions turn the speaker on?
   b) How do they accomplish this?
   c) What happens to the other bits on the port?

3. a) Which instructions turn the speaker off?
   b) How do they accomplish this?
   c) What happens to the other bits on the port?

4. a) What is the purpose of the 'myWait' subroutine?
   b) How does the 'myWait' subroutine achieve this?

5. How does the code produce a PWM ratio of 1:1?

For the joystick code:

1. Which input pin is affected by pressing the joystick up?
   Give the port name and bit number.
   Will pressing the joystick up set this bit to 0 or 1?

2. Which input pin is affected by pressing the joystick down?
   Give the port name and bit number.

3. Which input pin is affected by pressing the joystick center?
   Give the port name and bit number.

4. Which instruction reads the joystick inputs?
   In which register are these stored in the program?

5. Why is the LSR instruction used 4 times?

6. Which instruction sends the processed inputs to the LEDs?

Introductory Self-Test Answers

For the speaker code:

1. Bit 5 of Port B controls the speaker. Note the DDRB is initialized to 0x20, which is equivalent to 0x00100000.
   This sets Bit 5 of Port B to an output.

2. a) The instructions:
   `IN R16, PORTB, ORI R16, 0x20`
   and
   `OUT PORTB, R16`
   turn the speaker on.
   b) The first instruction reads what is currently on PORTB into R16. The second instruction is an 'or mask'. The 'or mask' is a bitwise or of every bit with the immediate value. The effect is that the 5th bit of R16 will always become a 1 after execution. The third instruction puts R16 into PORTB.
   c) The other bits retain the values before the 'or mask' is applied.

3. a) The instructions
   `IN R16, PORTB, ANDI R16, 0xDF`
   and
   `OUT PORTB, R16`
   turn the speaker off.
   b) The first instruction reads what is currently on PORTB into R16. The second instruction is an 'and mask'. The 'and mask' is a bitwise and of every bit with the immediate value. The effect is that the 5th bit of R16 will always become a 0 after execution. The third instruction puts R16 into PORTB.
   c) The other bits retain the values before the 'and mask' is applied.

4. a) The 'myWait' subroutine's purpose is to delay the processor. By delaying the processor, the speaker remains on or off for a fixed amount of time. This then gives the sound its frequency.
   b) The 'mywait' subroutine achieves its delay by performing unused multiply operations. The number of these operations is controlled by R20 because R20 controls the number of multiply loop iterations.

5. The PWM ratio is 1:1 because the '`myWait`' subroutine waits for an equal amount of time after switching bit 5 on and off.

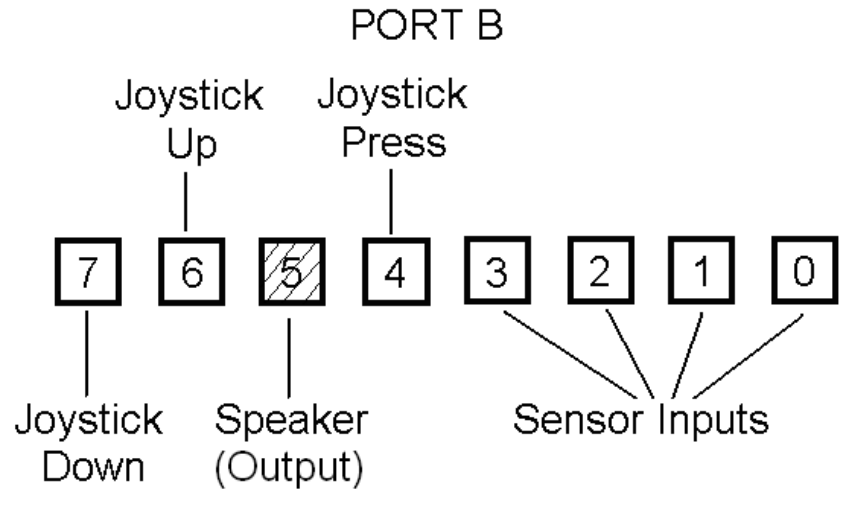
For the joystick code:

1. Bit 6 of Port B is controlled by pressing the joystick up. Pressing the joystick up will set this bit to **zero**. Therefore, it is a one when not pressed.

2. Bit 7 of Port B is controlled by pressing the joystick down.

3. Bit 4 of Port B is controlled by pressing the joystick center.

Note that all 8 bits of Port B have been accounted for and that bit 5 is an output whereas all the rest are inputs. See the diagram below:



4. The instruction
   `IN R16, PINB`
   reads the joystick inputs into R16.

5. The instruction
   `LSR`
   shifts the bits one place to the right. This is necessary because the joystick inputs are on bits 7, 6 and 4 whereas to display them on the LEDs, they need to be in bits 3, 2 and 0 of Port D.

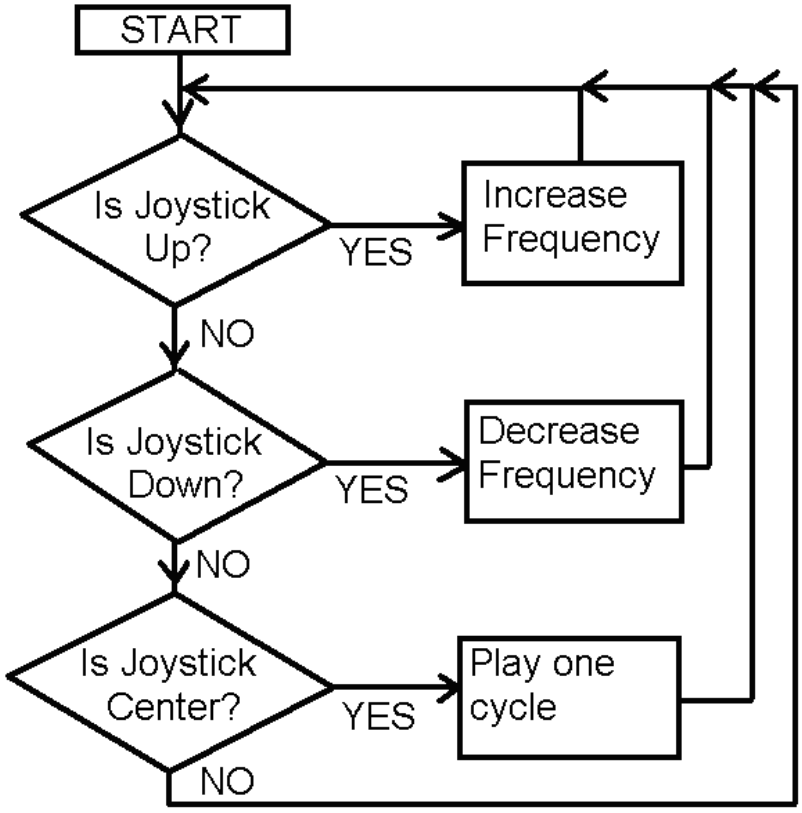
6. The instruction
   `OUT PORTD, R16`
   stores R16 into Port D.

Lab Exercise

Combine the two programs so that:

- The up button increases the frequency
- The down button decreases the frequency
- The sound only plays when the user presses the joystick center.
- Every joystick press should change the frequency by the same fixed amount, no matter how long the button has been pressed.
- Make sure there is no overflow/underflow. That is, when R20 is 0x00 and is decremented, R20 does not go to 0xFF.

The following flowchart may be of assistance:



Hints

Note that increasing R20 decreases frequency and vice versa. You may like to read up on the instructions BST, BRTC and BRTS. These are *very* useful.

Questions

After you present your work, be prepared to answer a few questions on the code you wrote. These may include, depending on your code:

1. How did you implement the diamond boxes on the flow diagram?

2. How did you implement the commands and how do they work?

3. Why is this particular command of importance in your code?

Extension - not assessable

If you get your code working, try setting R20 to the lowest value and hence the highest frequency. You may or may not be able to hear this sound depending on how well you've taken care of your hearing (i.e. no loud parties or rock concerts).

Generally, the younger the person, the more likely they will be able to hear the sound. Young people under 23 are normally able to hear high frequency sounds fairly well. However, once you get older, you will lose the ability to hear these high frequency sounds without a hearing aid.

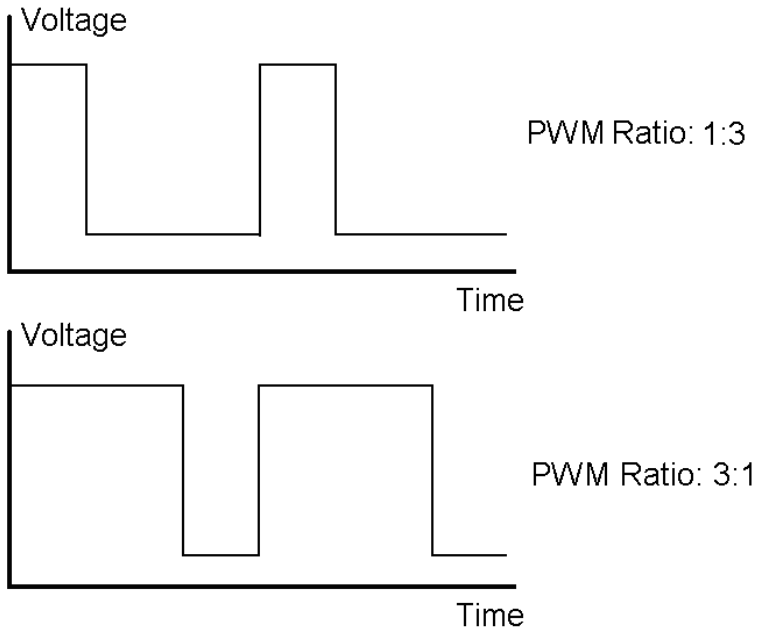
Experiment 5

Objectives

- Write an assembly program to generate a PWM signal
- Read input from multiple signals to control the PWM ratio.

Outline

Previously, a PWM signal maintained its ratio between high time and low time and varied its frequency to produce different sounds. In this lab, the frequency is fixed but PWM ratio will vary. This is illustrated in the diagram below:



The PWM signal will control the wheel speed of the car. The lower the ratio, the less time the motor is switched on and the slower the wheel rotates. Similarly, the higher the ratio, the more time the motor is switched on and the faster the wheel rotates.

Download the assembly code which demonstrates how to generate a PWM signal. Connect a wire from PP1 to the PWM pin of one of the motors. Run the code and observes what happens. Try changing the value of R16 and again observe what happens. The code is reproduced below:

```
.include "m169def.inc"

main:        LDI R16, 0xFF
             OUT DDRD, R16

             LDI  R16,0x04
             OUT SPH, R16
             LDI R16, 0xFF
             OUT SPL, R16

init:        LDI R16, 50
counter:     LDI R17, 0
delay:       LDI R18, 75
loop:        DEC R18
             BRNE loop

enddelay:    CP R17, R16
             BRLO settoone
             CALL turnOff
             RJMP continue
settoone:    CALL turnOn

continue:    INC R17
             CPI R17, 100
             BRNE delay
             RJMP counter

turnOff:     IN R19, PORTD
             BCLR 6
             BLD R19, 1
             OUT PORTD, R19
             RET

turnOn:      IN R19, PORTD
             BSET 6
             BLD R19, 1
             OUT PORTD, R19
             RET
```

Introductory self-test (not assessable)

Make sure that you understand the code and how it works. Try the following self test and check your answers over the page. These questions are ***not*** assessable.

1. What is the purpose of R16, R17, R18 and R19?

2. What is the purpose of the delay loop?

3. Once out of the delay loop, why are R17 and R16 compared?

4. What happens if R17 is greater than 16?

5. What happens if R17 is less than R16?

6. Which instruction resets R17 to 0?

7. Under what circumstances is this instruction executed?

8. How does the ‘turnOn’ subroutine work?

Introductory self-test answers

1. The purpose of R16 is for the user to input the desired PWM ratio. R17 is a counter which records how many times the delay loop has been executed – that always reset to 0 once it reaches 100. R18 is a loop counter that is used for waiting. R19 is a temporary register to alter the state of the pin.

2. The purpose of the delay loop is to delay the CPU for a small amount of time so that the PWM signal can be formed off it. This means that the PWM signal will be in discrete multiples of time for this delay loop.

3. R16 and R17 are compared to check if the counter is less than the desired PWM ratio. It's important to perform this check so that the correct PWM signal is generated.

4. If R17 is greater than R16, the motor is switched off. For instance if R17 was 30 and R16 was 20, then the program has been through the delay loop 30 times but the user only desires the signal to be high for the first 20 times, so therefore it should be off.

5. If R17 is less than R16, the motor is switched on. For instance if R17 was 10 and R16 was 20, then the program has been through the delay loop 10 times and the user wishes the signal to be high for the first 20 times, so therefore it should be on.

6. The instruction
   `LDI R17, 0`
   sets R17 to 0.

7. The instruction is executed whenever R17 reaches 100. When this happens, the instruction
   `BRNE delay`
   will not be taken and the program will jump to the instruction
   `LDI R17, 0`

8. The 'turnOn' subroutine works by storing Port D into R19. Then, the T flag of the status register is set to 1 using the BSET 6 instruction. The BLD R19, 1 instruction replaces the 1st bit in R19 with whatever is in the T flag. In this case it will always be 1. The modified R19 then replaces the contents of Port D. Finally, RET is called to return from the subroutine call.

Lab Exercise 1

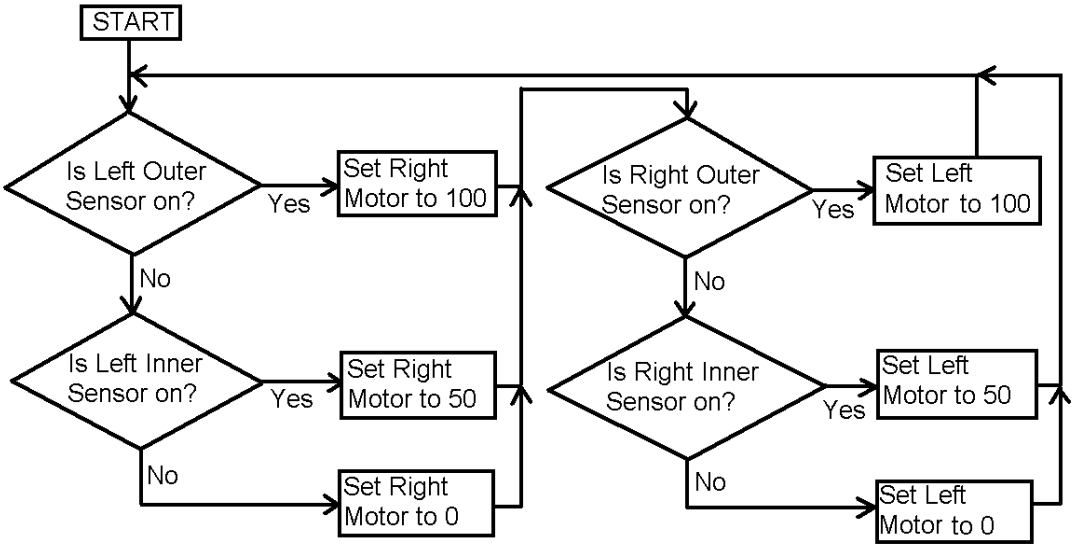
Extend the sample code so that both wheels are controlled independently by two variables stored in two different registers. For example, if the user wants to have the left wheel going at one quarter speed and the right wheel going at three quarter speed, then they set one register to 25 and the other to 75.

Hints

Think about what necessarily needs to be duplicated to generate two signals. Keep the rest common to both signals.

Lab Exercise 2

Extend your solution from Exercise 1 so that 4 sensors control the wheel speed (2 for each wheel). Once that is done, the car should be fully autonomous and able to drive around the track. It does not have to stop at the white line. Each wheel should be capable of achieving at least 3 different speeds depending on the sensor inputs. The following flowchart is highly recommended:



Hints

Implement each of the square boxes by changing the value in the registers for the code in part 1. That is, you should ***not*** have to modify any of the code in part 1; all you should be doing is changing the inputs to this sub-program.

The following sensor positions are recommended. Left Outer Sensor on square 5 on the 2nd back row. Left Inner Sensor on square 4 on the back row. This should be repeated symmetrically for the right sensors. The back row is the row closest to the wheels.

Questions

Be prepared to answer questions on your code. These will depend on the code. Some examples are given below:

- Why is it important that you branch at this point?
- Describe your program and how it works.
- What function do these instructions serve?

APPENDIX A - Description of Instruments

Detailed manuals are available from the Laboratory Demonstrators. They are not to be removed from the laboratory without permission.

GW DC Power Supply Model GPC3030

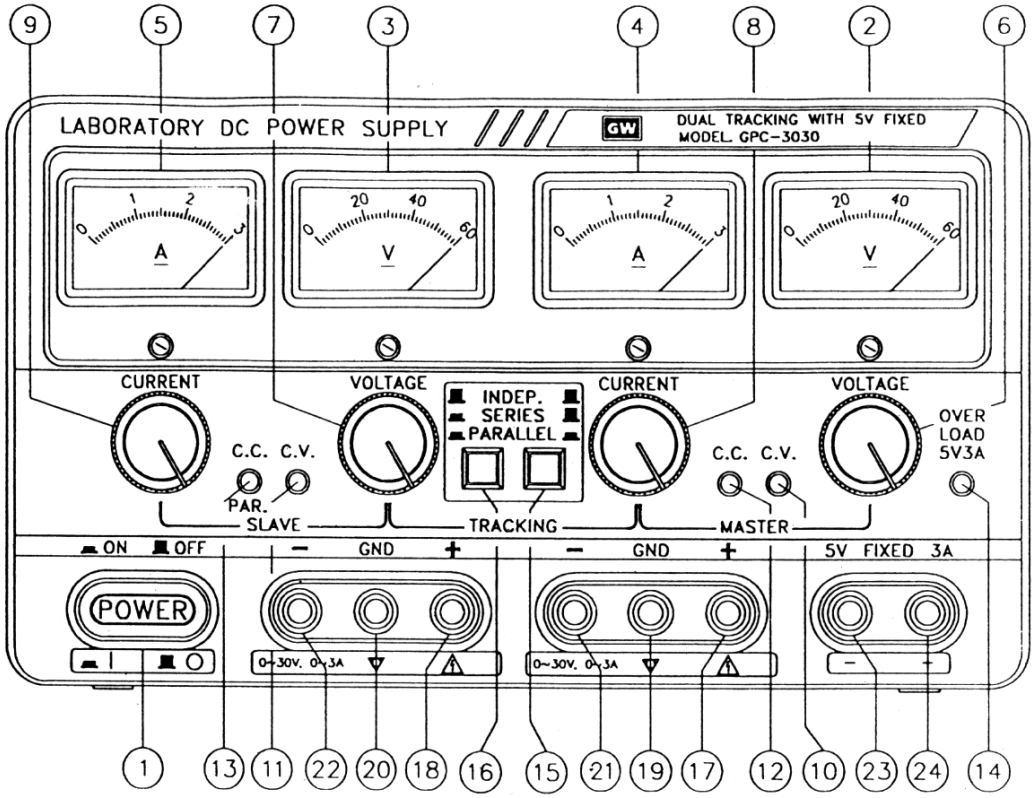


Fig. 4-1 Front Panel (Analog Type)

Specifications

2 0 - 30V, 0 - 3A supplies

1 fixed 5V 3A supply

Independent sources can be operated in master - slave mode in parallel and in series.

Parallel operation: 0 - 30V 0 - 6A

Series operation: 0 - 60V 0 - 3A

Operation

1) Ensure that the supply is plugged in and the mains switch is on.

2) Turn on the supply by pushing in the button on the bottom left of the supply.

5V - 3A Fixed Supply.

1) This supply is immediately available and ready to use. Note that the current limit is set and is not adjustable on this output.

0 - 30V 0 - 3A Independently Variable Supplies

2) Ensure that both buttons in the middle of the supply are out (ie independent mode of operation).

3) The output from the supply comes from the terminals labelled + and -. The terminal labelled GND is a connection to the instrument's chassis ground. The GND terminal is not normally used.

To set current limit.

1) Set the current control knob and voltage control knobs fully anticlockwise (ie to minimum).

2) Temporarily short out the supply.

3) Increase the voltage slightly until the constant current indicator (CC) is lit.

4) Adjust the current set control to the desired current limit.

5) Do not adjust the current control again until a different current limit is required.

6) Remove the short circuit.

7) Adjust the voltage control to the desired output voltage.

Function Description

1 Power switch: Turns the power supply ON/OFF.
2 Meter: Indicates the MASTER output voltage (Analog type)
Indicates the MASTER output voltage or current (Digital type)
3 Meter: Indicates the SLAVE output voltage (Analog type)
4 Meter: Indicates the MASTER output current (Analog type)
5 Meter: Indicates the SLAVE output current (Analog type)
6 Voltage control: for adjustment of the output voltage of the MASTER supply. Also functions as adjustment control for the maximum output voltage of the SLAVE supply when either parallel or series tracking operation.
7 Voltage control: for adjustment of the output voltage of the SLAVE supply when the independent operation.
8 Current control: for adjustment of the output current of the MASTER supply. Also functions as adjustment control for the maximum output voltage of the SLAVE supply when either parallel or series tracking operation.
9 Current control: for adjustment of the output current of the SLAVE supply.
10 C.V. indicator: lights when the MASTER supply is in the constant voltage operation. In either the Series or Parallel Tracking mode, both the MASTER AND SLAVE supplies are in the constant voltage operation.
11 C.V. indicator: lights when the SLAVE supply is in the constant voltage operation.
12 C.C. indicator: lights when the MASTER supply is in the constant current operation.

13 C.C. indicator: lights when the SLAVE supply is in the constant current operation. Also lights when the TRACKING PARALLEL mode is selected.

14 Overload indicator: lights when load on 5-volt supply becomes too large.

15 & 16 TRACKING Mode Switches:

Two push-button switches that select INDEPendent mode, SERIES tracking mode, or PARALLEL tracking mode as follows:

a. When both switches are disengaged (out), the unit is in the INDEPendent mode and the MASTER and SLAVE power supplies are completely independent from one another.

b. When the left switch is engaged (in) and the right switch is disengaged (out), the unit is in the TRACKING SERIES mode. In this mode, maximum voltage of both supplies is set using the MASTER VOLTAGE controls (voltage at output terminals of the SLAVE supply tracks the voltage at the output terminals of the MASTER supply). Also, in this mode of operation the positive terminal (red) of the SLAVE supply is connected to the negative terminal (black) of the MASTER supply. This allows the two supplies to be used as one 0 to double rating voltage supply.

c. When both switches are engaged (in), the unit is in the TRACKING PARALLEL mode. In this mode the MASTER and SLAVE supplies are wired together in parallel and both the maximum current and voltage are set using the MASTER controls. The MASTER and SLAVE outputs can be used as two individual (but tracking) power supplies or just the MASTER output can be used as a 0 to rating voltage supply with a 0 to double rating current capability.

17 "+" output terminal: Positive polarity output terminal for the MASTER supply.

18 "+" output terminal: Positive polarity output terminal for the SLAVE supply.

19 **GND terminal: Earth and chassis ground.**

20 **GND terminal: Earth and chassis ground.**

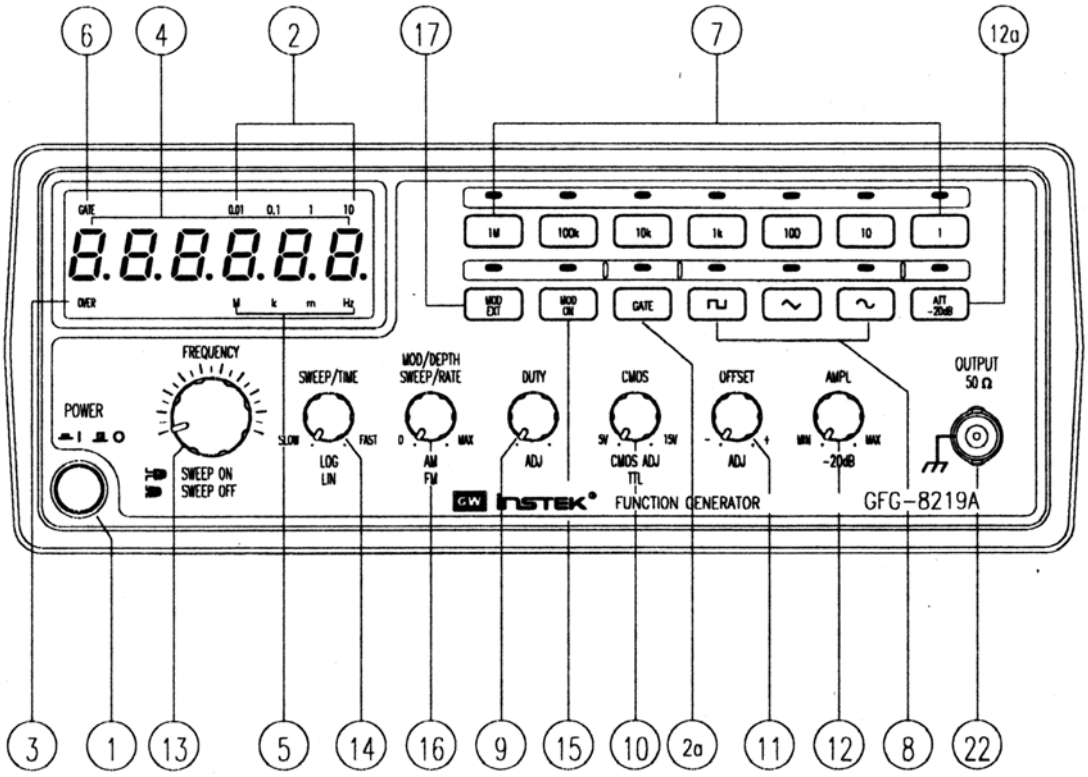
21 "-" output terminal: Negative polarity output terminal for the MASTER supply.

22 "-" output terminal: Negative polarity output terminal for the SLAVE supply.

23 "-" output terminal: Negative polarity output terminal for 5V supply.

24 "+" output terminal: Positive polarity output terminal for 5V supply.

GW Function Generator Model GFG-8219A



Specifications

Waveforms:	sine, square, triangle, ramp		
Frequency ranges:	Minimum frequency for 0.3MHz		
	Maximum frequency for 3MHz		
Frequency ranges:	sine	triangle	square
(<1% distortion)	200kHz	100kHz	100kHz
Output impedance:	50Ω (ie the function generator looks like an ideal voltage source in series with a 50Ω resistor)		
Output voltage:	up to ±10V in to 50Ω		
DC offset	±5V in to 50Ω		
IMPORTANT:	**Always connect the ground output of the function generator to ground of the circuit. *DAMAGE WILL RESULT IF THIS IS NOT DONE.***		

Operation

In general, the operation of the function generator is straightforward. To activate some functions requires the appropriate know to be pulled out (gently). The functions to which this applies are:

Duty cycle*/ramp generation: Pull out to make knob active. When pushed in, the duty cycle is always 50%

DC offset: Pull out this knob to make it active.

Function Description

1.	Power Switch	Connect the AC power, then press power switch.
2.	Gate Time Indicator	Press the power switch, Gate time indicator will start to flash (the gate time of internal counter is 0.01 second).
2a.	Gate Time Selector	Press this key to change gate time when use external counter mode. The change order is according to 0.0ls, 0.1s, 1s, 10s cycle by pressing these keys.
3.	Over Indicator	In the external counter mode, the indicator is illuminated when the output frequency is greater than the range selected.
4.	Counter Display	Shows the external frequency by 6 x 0.3" green display, and shows the internal frequency by 5 x 0.3 green display.
5.	Frequency Indicator	Indicate the current frequency value.
6.	Gate Time Indicator	Indicate the current Gate time (external. Counter mode use only).
7.	Frequency Range Selector	To select the required frequency range by pressing the relevant push button on the panel as shown in Table 1 and Table 2.

Table 1(for GFG-8215A/8216A/8217A/8219A)

Push bottom	1	10	100	1k	10k	100k	1M
Frequency Range	0.3Hz | 3Hz	3Hz | 30Hz	30Hz | 300Hz	300Hz | 3kHz	3kHz | 30kHz	30kHz | 300kHz	300kHz | 3MHz

Table 2(for GFG-825OA/8255A)

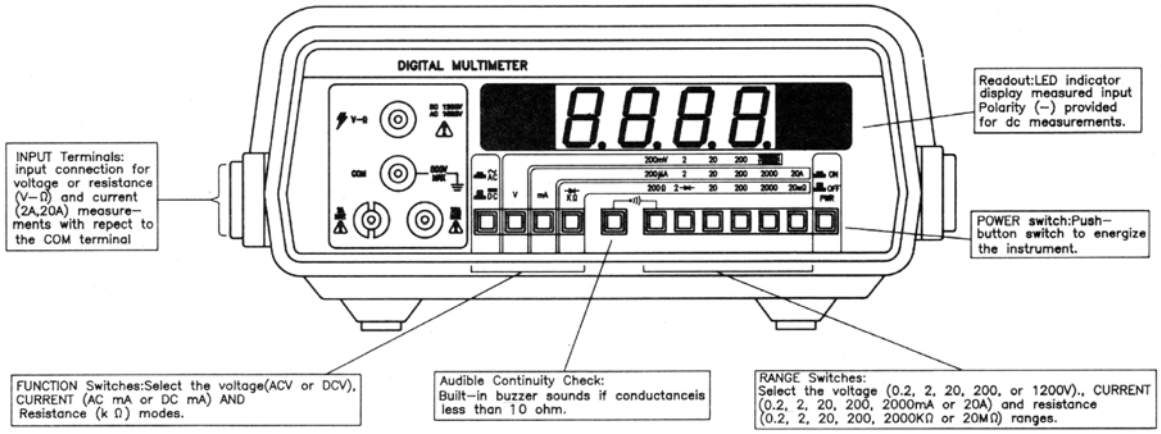
Push bottom	1	10	100	1k	10k	100k	1M
Frequency Range	0.5Hz | 5Hz	5Hz | 50Hz	50Hz | 500Hz	500Hz | 5kHz	5kHz | 50kHz	50kHz | 500kHz	500kHz | 5MHz

8.	Function Selector	Press one of the three push buttons to select the desired output waveform.
9.	Duty Function	Pull out and rotate the knob to adjust the duty cycle of the waveform.
10.	TTL/CMOS Selector	When push in the knob, the BNC terminal of (20) will output a TTL compatible waveform. If pull out and rotate the knob can adjust the CMOS compatible output (5-15Vpp) from the output of BNC (20).
11.	DC Offset Control	Pull out the knob to select any DC level of the waveform between ±10V, turn clockwise to set a positive DC level waveform and invert for a negative DC level waveform.

* Duty cycle refers to the ratio of the time pulse wave output is high to the time the output is low. A 50% duty cycle pulse waveform is a square wave.

12. Output Amplitude Control with Attenuation Operation	Turn clockwise for MAX. output and invert for a –20dB output. Pull the knob out for an additional 20dB output attenuation.
12a. 20dB Attenuation	Press the knob to adjust a –20dB output.
13. MANU/SWEEP Selector and Frequency Adjustment (Sweep On/Off)	Press and turn clockwise the knob for MAX frequency and invert for MIN frequency. (Keep the pointer within the scale range on the panel.). Pull out the knob to start the auto sweep operation; the upper frequency limit is determined by the knob position.
1. Sweep Time Control and LIN/LOG Selector	(1) Rotate the knob clockwise to adjust sweep time for MAX, or invert for MIN. (2) To proceed Linear sweep mode by pushing in the knob, or select LOG sweep mode by pulling out the knob.
15. Control MOD ON/OFF Selector	Pull out the knob, the output can be modulated by internal 400Hz Sine wave or an external signal via CVF/MOD in connector (21).
16. Sweep Width & Modulation Carrier &AM/FM Selector &FM Selector	(1) Sweep width can be controlled from 0 to 1000 times. (2) To adjust modulation ratio by turning the knob clockwise for MAX, or invert for MIN. (3) Press the knob to get AM function or pull it out for FM function.
17 INT/EXT MOD Selector	When press the button once, the indicator will lighten, then the EXT MOD has been selected. Press the key again, the indicator will be off, then INT MOD has been selected.

GW Digital Multimeter Model GDM-8135



Specifications

Number of digits:	3 ½

DC Voltage

Ranges	±199.9mV, ±1.999V, ±19.99V, ±199.9V, ±1999V
Maximum voltage	1200V
Input impedance	>10MΩ all ranges

AC Voltage

Ranges	±199.9mV, ±1.999V, ±19.99V, ±199.9V, ±1000V
Maximum voltage	1000V rms
Input impedance	10MΩ in parallel with 100pF all ranges
Maximum frequency	40kHz

DC Current

Ranges	±199.9µA, ±1.999mA, ±19.99mA, ±199.9mA, ±1.999A, ±19.99A
Maximum input	2A input 2A rms (fuse protected) 20A input 20A rms (not fuse protected)
Voltage burden	0.22V max up to 2A

AC Current

Ranges	±199.9µA, ±1.999mA, ±19.99mA, ±199.9mA, ±1.999A, ±19.99A
Maximum input	2A input 2A rms (fuse protected) 20A input 20A rms (not fuse protected)
Voltage burden	0.22V max up to 2A
Maximum frequency	40kHz

Resistance

Ranges	199.9Ω, 1.999kΩ, 19.99kΩ, 199.9kΩ, 1999kΩ, 19.99MΩ

Operation

To measure voltage:	Use COM and V-Ω terminals
To measure resistance:	Use COM and V-Ω terminals
To measure current (<2A):	Use COM and 2A terminals
To measure current (<20A):	Use COM and 20A terminals

MetraHit Precision Digital Multi/Power Meter

Specifications

Number of digits: 5 ½

DC Voltage

Ranges (resolution in brackets)	±300mV (1µV), ±3V (10µV), ±30V (100µV), ±300V (1mV), ±1000V (10mV)
Maximum voltage	1050V
Input impedance	20MΩ on 300mV range 10MΩ on other ranges

AC Voltage

Ranges (resolution in brackets)	±300mV (1µV), ±3V (10µV), ±30V (100µV), ±300V (1mV), ±1000V (10mV)
Maximum voltage	1000V rms
Input impedance	5MΩ in parallel with 50pF all ranges
Maximum frequency	100kHz

DC Current

Ranges (resolution in brackets)	±300µA (1nA), ±3mA (10nA), ±30mA (100nA), ±300mA (1µA), ±3A (100µA), ±10A (1mA)
Maximum input	mA input 0.36A rms (fuse protected) A input 10A rms (fuse protected)
Voltage burden	0.11V at 3A

AC Current

Ranges (resolution in brackets)	±300µA (1nA), ±3mA (10nA), ±30mA (100nA), ±300mA (1µA), ±3A (100µA), ±10A (1mA)
Maximum input	mA input 0.36A rms (fuse protected)

	A input	10A rms	(fuse protected)
Voltage burden	0.11V at 3A		
Maximum frequency	100kHz		

Resistance

Ranges (resolution in brackets)	300Ω (1mΩ), 3kΩ (10mΩ), 30kΩ (100mΩ), 300kΩ (1Ω), 3MΩ (10Ω), 30MΩ (100Ω)

Power

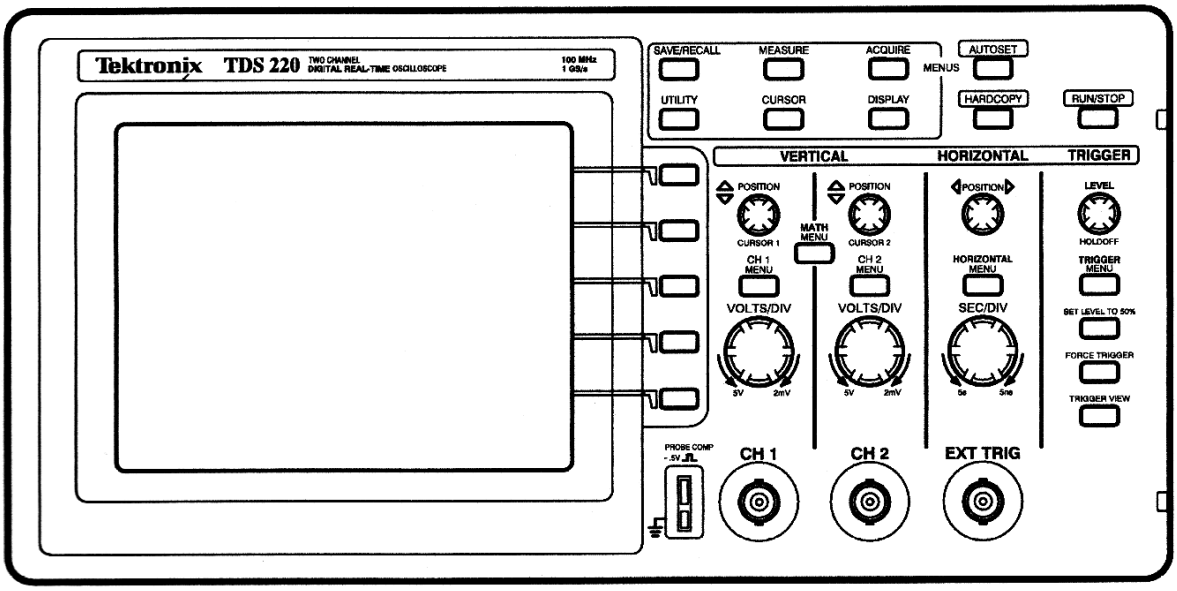
Ranges (mA setting) (resolution in brackets)	1mW (0.1µW), 10mW (1µW), 100mW (10µW), 1W (100µW), 10W (1mW), 100W (10mW), 1kW (100mW)
Ranges (A setting)	10W (1mW), 100W (10mW),1kW (100mW)

Capacitance

Ranges (resolution in brackets)	3nF (1pF), 30nF (10pF), 300nF (100pF), 3µF (1nF), 30µF (10nF), 300µF (100nF), 3,000µF (1µF), 30,000µF (10µF)

This meter can also measure energy and frequency, can log data and events in internal memory, and download data via an infrared link to a PC.

Tektronix Digital Oscilloscope Model TDS210



Function

The primary function of an oscilloscope is to display the time variation of a voltage signal. The voltage is usually plotted on the vertical axis while the time since triggering is plotted on the horizontal axis. The voltage signals are usually repetitive and appear on the screen as a stationary image provided the trace starts at the same point on the voltage waveform each time it displays. This is achieved by triggering the oscilloscope to start displaying the voltage when the input voltage reaches a certain value.

Operation

Ensure that the oscilloscope is plugged in and turned on at the mains.

Turn on the oscilloscope (white button on the top of the unit on the left-hand side). The instrument will undergo a self-test sequence. When the display indicates that it has passed the self-test the instrument is ready to use.

Push the button marked CH 1 MENU just above the VOLTS/DIV knob. Push the top menu item select button (the buttons along side the screen) several times until the Coupling is set to DC.

Connect the oscilloscope probe to CH1 input. Note that probes often include an attenuator that reduces the size of the signal reaching the oscilloscope. This is done to increase the input resistance of the instrument so that it does not interfere with the circuit being tested (oscilloscopes typically have 1MΩ input resistance; probes typically increase this to 10MΩ, but reduce the signal by a factor of 10). You can correct the display on the oscilloscope for the change in

sensitivity caused by the probe by pushing the CH 1 MENU button and the 4th menu select button repeatedly until the probe setting matches the probe you are using.

You can repeat the above instructions for CH2 if you need to measure two signals simultaneously.

Connect the probe attached to CH 1 to your signal source. Note that an oscilloscope is designed to measure voltage. This means you need to connect both the probe tip and the probe ground to your signal source. Push the AUTOSET button. This puts the oscilloscope into a sensible set up to measure your signal. If you have no signal attached, you may see just noise on the display. If this happens, just repeat the AUTOSET sequence after you have correctly connected the probe.

The voltage scales for CH 1 and CH 2 can be changed independently using the knobs above the input connectors for CH 1 and CH 2.

The position on the display of each channel can be adjusted using the POSITION knobs.

The time scale can be changed using right hand knob.

APPENDIX B - Circuit Inside Vehicle