A GUIDE TO YOUR 2016 OCCRA LARGE ROBOTICS KIT v.3

TABLE OF CONTENTS

A GUIDE TO YOUR 2016 OCCRA ROBOTICS KIT: PART I.

Written by Mike McIntyre
STEM Coordinator
Oakland Schools Career Focused Education

I. INTRODUCTION

Robots are machines that are automated, multitasking and programmable. If you are a high school student, you and your teammates will be building a large (maximum = 120 lb.) robot for competition in the “Go 4 It!” Game. The robot that you are building for this year’s Oakland County Competitive Robotics Association tournaments *could* fit these criteria but, since we are going to continue using student drivers and local gyms with minimal barriers, your large machine will be only “semi-autonomous” at best. For middle school students who are using the smaller VEX robot, yours will be fully autonomous for part of each match. Your robots have a "brain" (microprocessor) and are capable of learning (being programmed), receiving input information from sensors (joysticks, limit switches, potentiometers, range finders, etc.), and running autonomously. A robot’s “program” is nothing more than a set of instructions that guide the robot’s actions. These instructions can be written in many different programming “languages” but, in OCCRA, we are asking teams to use the language that we supply: “RobotC.” We are going to allow some programming of the *large* robots again this year but there will *not* be a totally autonomous period as there has been in FIRST competitions through the years.

Teams do not need to program their large robots if they do not want to; there are four default programs already loaded in your robot’s microcontroller that automatically assigns the CORTEX’s (robot brain) outputs to the Joystick’s (operator station) inputs, so nobody needs to learn the “RobotC” programming language unless they are planning to use pneumatics, sensors, or they just really want to! The VEX robots, however, **need** to be programmed to be competitive. We will use the VEX control system for both high school and middle school OCCRA competitions. For both high school and middle school programmers, there will be a separate instructional workshop devoted to this. The remainder of this manual will be devoted to understanding the science behind the robots and learning the workings of the components in the *large* robot’s kit of materials. Some of this information will also pertain to the VEX robots but there will be separate material (mostly web-based) that focuses on the smaller VEX robot, engineering design principles, best practices in robot building, programming, etc.

Figure 1a. The Joystick Control Arrangement

The students who drive your large robot will use a set of 2 **Joystick Modules** (with 2 “sticks” on each) mounted on a control panel and including a **VEXNet key** (shown attached in Figure 1a and separately in Figure 1b.) One of the joystick modules will serve as the primary joystick module and will have the USB VEXnet key attached, the other joystick module will be the “partner” joystick and will not have a USB VEXnet key installed. The VEXNet key functions like other Bluetooth devices: it is both a radio transmitter and receiver. The Joystick modules contain the processors, and the VEXnet 802.11g USB key provides the wireless communication.

Figure 1b. the VEXnet 802.11g USB key

Over the years, we have used different control systems in OCCRA. The current VEXnet System, as with all the other systems, requires a special computer program that controls the hardware devices. This special computer program is known as “firmware.” The firmware contained in the Joystick Module’s microprocessor provides the control program for the Joystick, the firmware installed in the Cortex’s microprocessor controls the Cortex, etc. Changing the VEX firmware is usually only done about once a year for OCCRA. We need to have the firmware updated each season because it allows the VEX manufacturer to fix any bugs in the program, add new features to the devices, etc.

The Cortex Module and the Joystick Module both have firmware that was installed in them at the factory. Since the firmware software changes over time, it is very important that you are using the current version for the 2016 season**. It is extremely important that the Joysticks, VEXNet keys and the Cortex modules are using the same version of the firmware!** The two modules will not link up properly and nothing on your robot will work if you do not take care of this first! The latest version of the firmware can be downloaded for free from the VEX website. We will be checking this at the first tournament but please do not wait until then to take care of this!

If you have ever driven a remote-controlled car you have used the same basic system that will control the robots in this year's competition. The rules governing the "sanctioned" layout are no longer very specific: you are allowed to make some modifications to the basic layout this year, but do not attempt to modify the VEXnet components themselves.

There is a Quick Start Guide for using the VEX CORTEX Microcontroller and VEX Joystick that can be found on the internet. Refer to the VEX Wiki: https://www.vexrobotics.com/wiki/VEX_Cortex_Microcontroller for updates to this document. Some of the text has been copied in Appendix V for your convenience.

The operator controls, or “joysticks,” that sit on your control panel can be attached with Velcro, rubber bands, etc. so that they can be lifted and held by the drivers during a match, but secured to the control board during transport. Moving a joystick is essentially like turning a dimmer switch on a light, while pushing one of the joystick’s buttons is like flipping a switch: both send an electrical signal from the Joystick to the Cortex. Sensor inputs on the Joystick Module detect which switch has been thrown [See Fig. 1a]. The microprocessor inside the Joystick then assigns and forwards that information through the unique routing or "**channel**" that had been assigned to that particular switch. For example, the four black buttons on the right send data via “Channel 8.”

After the Joystick Module assigns **data** from a switch or **sensor** and forwards that information through a channel routing, the signal is then sent to the transmitter: the VEXnet 802.11g USB key. (Use only the 2.0 WHITE keys, not the BLACK key that’s pictured in Fig. 1c!) This transmitter then sends a radio signal to your robot. Your robot's receiver, another VEXnet USB key that’s attached to the **Cortex** (see Figure 1c.), listens to the frequency of its own transmitter, and carries the signal to the onboard computer, the Cortex. The Cortex is the robot’s controller. Using some built-in instructions (the **“Robot-C” program**), the Cortex controller sends an electrical signal to the **output** terminal that the program has designated. This signal is sent to either an electric **relay** ("**Spike**") or **speed controller** ("**Victor**" or “Talon”) which then activates one of the robot's motors or valves. Your 'bot then slam-dunks another ball!

Figure 1c. The Robot’s Brain: Battery, Cortex Microcontroller, and VEXnet Key

Throughout the building of your robot and all the competitions, please **adhere closely to all safety rules and procedures**. Hand tools can be dangerous if not used properly; make sure that tools are only used for their intended purpose and that all people using tools know the correct techniques and procedures. There is a standardized list of “legal” tools that teams may use this year. Teams that desire any of these tools but lack

the resources to buy them should follow the instructions in the game manual for borrowing them. We request that all people in work areas wear **safety goggles**. (They will be required in the pit area of all league competitions.) The powerful motors, pneumatic cylinders, and high-energy batteries in this kit must be treated with a healthy respect. **Remember: Always disconnect the power to your robot and release the stored up air pressure while working on it or transporting it**.

Never let wires from the positive and the negative terminals of your battery come into contact: the "**short circuit**" that results can deliver large currents of electricity. Fires and burns are possible if you are not careful. If anybody has strong **nickel** or **latex allergies** they must use extra caution as the OCCRA kit contains both materials. To make this a rewarding, exciting, SAFE experience for all involved we need *everybody's* cooperation. See APPENDIX II for more details.

Figure 2.
Use a vice or clamp to hold your work securely and ALWAYS WEAR THOSE SAFETY GLASSES!

II. THE SCIENTIFIC BASIS OF YOUR ROBOT

1. BACKGROUND ELECTRICAL TERMS AND CONCEPTS

You have probably already learned in your science classes about the particle nature of matter and the existence of **atoms**. Any high school text on general science or physical science would give you a good introduction to the scientific principles relevant here. All matter contains tiny particles called atoms. Atoms contain even smaller (subatomic) particles. Scientists have learned that there are three main particles inside of atoms. The particles we are most concerned with here are called "**electrons**". They are the ones with the **negative charge** that are found in the outer layers of each atom. Atoms also contain particles (protons) that have a positive charge. Charge is the property of matter that causes electric phenomenon. Charges are classified as either **positive** or **negative**. If two pieces of matter have opposite charges, they will try to attract each other. If two pieces of matter have the same kind of charge, they want to repel each other.

When these electrons begin to move in one direction, jumping from atom to atom, the flow is called "**current**". The amount of current flowing is measured in terms of a unit called an "**Ampere**" or, more simply, an "Amp". The battery in your kit is capable of delivering well over 120 Amps (but we'll try not to let that happen!). Current, therefore, measures how much charge flows past each second.

Electrons do not all start flowing in one direction unless something is pushing them. A sealed lead/acid battery will supply the electrical "push" needed by your robot's electrical system. The battery contains two strips of different metals called **electrodes** (your kit uses lead/lead oxide) that are immersed in chemicals (a sulfuric acid solution). These ingredients react together causing more electrons to build up on one (the negative) electrode than on the other (positive) electrode. This difference (potential difference) in electron concentration creates a pushing effect that is known as "**electromotive force**" or "**emf**".

The negative battery terminal will have a **black wire** attached to it. We will use this electrode as the reference or "**ground**" level for our electrical system. The terminal strip where all of the negative black wires attach will therefore be called the "**negative terminal**" or "**grounding strip,"** even though it does not actually connect to the earth. Electrons all have the same negative charge and do not want to be crammed in together on this

negative terminal; given the opportunity, they repel each other until they have spread out evenly. A unit called "**volt**" measures the amount of emf. Consequently, emf is commonly referred to as "**voltage**". The kit battery should produce a fairly steady emf of 13.5 to 12.5 volts. **Batteries are constant voltage sources**. This means that they will maintain nearly the same voltage across the terminals as long as the chemicals inside are still reacting. When the battery's chemicals are nearly used up, the voltage begins to drop *dramatically*. Your battery should be well over 13 volts when fully charged. Never start a match if the battery's voltage is below 13 volts because this indicates that the battery's energy is almost depleted.

Figure 3a. The sealed lead-acid batteries typically last for three matches, but don't count on it: replace and recharge frequently!!

During a match, there can be moments when a very large amount of current is being drawn and the voltage across the battery terminals drops way below 12V—this is why we ask teams to power their Cortex microcontrollers with the separate, blue, 7.2V VEX batteries AND use a back-up 9V battery. We don't want the CORTEX to get starved of electricity during a match! (Your robot starts misbehaving badly in these moments.)

Your kit comes with a **recharger**. Follow the directions that come with the recharger when hooking your battery up to it. It takes a few hours but the chemical reaction that went on inside the battery will be almost completely reversed. Your recharged battery will return to practically the same voltage that it had when it was new. Take care in organizing your battery management system throughout the year. Over the years many teams have lost matches because they took the field with a battery that was not fully charged. You should have a team member assigned the task of charging up the used battery and installing a fresh battery after each match. Remember: the robot can drain a battery in 10 to 20 minutes, but it takes many hours to fully recharge a dead battery. The person who handles battery management, the "Energy Czar," needs to also keep a close watch on the electrical connections. **ALL CONNECTIONS MUST BE TIGHT.** A loose wire will greatly reduce the amount of current that flows and will cause the robot to behave as though the battery was dying out!

Figure 3b. The newest type of charger in the OCCRA Kit of Materials

Many people confuse the terms "current" and "voltage": they are two very different creatures. To help understand the difference, imagine 2 identical barrels of water along side each other, connected near the bottom by a single pipe. (See Figure 3b.) If the barrel on the right is filled to the top with water and the barrel on the left is only half full, what happens? Water begins to flow from the right barrel, through the pipe, into the left barrel. How long does the flow last? The flow continues until there is no longer a difference in the water levels. In this example, the drops of water represented the electrons in the battery. The full barrel represented the battery's negative electrode and the partially full barrel represented the battery's positive electrode. If the amount of water flowing through the pipe each second had been measured, we could tell how much current we had. What caused the current to flow? The pressure difference in the 2 barrels, created by the difference in water depth caused the current to flow. This pressure difference represented the emf, or voltage. (In fact, voltage is often expressed as "**potential difference**".) What would have happened if the pipe had been clogged shut? We still would have had the pressure difference but there would have been no flow. Likewise, it's possible to have current with almost no voltage. Notice from the example that, to take a voltage reading, you have to compare two **points**. This explains why you hear people talk about measuring the voltage "across" the battery's electrodes or "across" the motor's terminals. If somebody says they measured how much voltage was "flowing through the wire", you'll know they're a little confused.

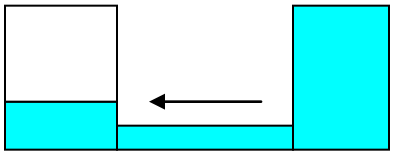


Figure 3c. The Current Is Caused by the Potential Difference

Current must have a **path** or route to follow. The complete path followed by the current is called a "**circuit**". The circuits on your robot begin with the negative battery electrode (**black** is the color code for wires that connect toward this **negative** terminal; **green** is also used sometimes). The current flows through the various circuits on the robot until it returns through the **positive (red) wires** and into the positive battery terminal. Positive wires are also color-coded **white** sometimes. (While electricity can also be thought of as a flow of positive charges in the reverse direction, the so-called "**Classic Current**", I will be using the "**Electron Flow Model**" in this paper.) Current can flow in a **closed** circuit but cannot flow in a circuit that has been cut, or **opened**. A **switch** is a device that interrupts the current by breaking the circuit (or, in some types of switches, closing the switch completes the circuit and *allows* the current to flow). **Fuses** and **circuit breakers** can also stop the flow of current; you will hear more on them later.

The **output power** of a motor can be defined as the rate at which the motor does work. The output power of a motor can be calculated by multiplying the motor's **torque** by its **rotational speed**. This will be explained in more detail later. The rate at which energy gets used up is called "**input power**". The metric unit used to measure the electrical input power (or any other type of power) is the **watt**. Electric power, which supplies your motor's input power, can be calculated by multiplying emf times current (watts = volts x amps). Look at Appendix I for a listing of all the motors in your kit and their power ratings. This chart should help you considerably when you allocate motors while planning your robot's design. A perfect motor would have equal input and output power. The motors in your kit are pretty slick but, sadly, are not perfect. The output power will always be a fraction of the input power. This fraction can be written as a percent and is called the "efficiency rating" of the motor. A DC motor that uses 4 watts of electric power but only delivers 3 watts of output mechanical power has an efficiency of 3/4 = .75 = 75%. Small electric motors often have efficiencies below 50%. A well-aligned set of sprockets or gears can deliver power with efficiencies over 95%. The same method for computing efficiencies can be used to analyze any machine, including gear trains and the sprocket drive systems used on your robot.

2. BACKGROUND PNEUMATIC TERMS AND CONCEPTS

You must understand certain basic scientific concepts before you can really understand how a pneumatic system works. In science, pushes and pulls are called **forces**. The **pound** is the unit of force that we use most often in OCCRA. Weight is an example of a force and can be measured in pounds (for example, your 2016 OCCRA robot may not weigh over 120 pounds). **Area** is the amount of space that a surface has and is measured in square units. In OCCRA, we generally size things in terms of inches, so our usual area unit is the square inch. A force being applied to an area causes **pressure**. Pressure can be calculated by dividing the applied force by the area on which it acts. The rules for OCCRA 2016 require that the air pressure used on your robot be 60 psi or less ("**psi**" is short for "**pounds per square inch**", the unit used in this booklet to measure air pressure). Anything that flows is a **fluid**, so liquids and gases are both fluids. Centuries ago, a young French scientist named Pascal discovered that anytime an enclosed fluid is subjected to a force, pressure is transmitted throughout the fluid equally in all directions. This very important discovery is called "**Pascal's Law**." The metric unit of pressure is named the "Pascal" in his honor. (FYI: 6.81 kiloPascals equals 1.00 psi)

Fluid power systems have a lot in common with electrical systems and mechanical systems. Pneumatic systems are fluid power systems that use a gas (typically air) to transmit power through a circuit and do useful work. The term "**circuit**" refers to the complete path followed by the fluid as it flows from its source to the various output devices. Instead of electrical energy, pneumatics uses the energy of compressed (pressurized) air to transmit power. The amount of electricity moving through a wire is a characteristic called "**current**" while the rate of a moving fluid is referred to as the **flow.** The **prime mover** in electrical systems is called the

"potential difference" (emf), while it is a **pressure difference** that **causes fluids to flow**. The following chart summarizes the comparisons between the three systems:

	Mechanical System	**Electrical System**	**Fluid System**
Prime mover	Force or torque	Potential difference (voltage)	Pressure difference
Rate	Speed	Current	Flow
Resistance	Friction	Electrical resistance	Fluid resistance
Power	Force x Distance or torque x rotational speed	Voltage x Current	Pressure x Flow

Think of pneumatics as a method of storing and transferring energy to perform work using compressed air. I am sure that you have all seen pneumatics being used before: large rides at amusement parks, air brakes on buses and trucks, dental drills, the carrier at the bank drive up window, the animated figures at Disney World, power tools at auto service stations…etc.

The basic sections of a pneumatic system are the **air production system** and the **air consumption system**. On your robot, the air production system includes the compressor, filter, pressure switch, check valve, relief valve, and accumulator tank. Industrial systems would also include coolers, auto drains, dryers…etc. The air consumption system is composed of the regulator, control valves and the cylinders. Since the output of the entire pneumatic system rests with the **cylinders** (**actuators**), it is worth our while to spend a little time on them.

Each of the pneumatic cylinders supplied in the OCCRA kit of materials contains a rod that attaches to a disk inside the cylinder to form a moveable **piston**. As this piston moves out from the cylinder (**extends**) and back into the cylinder (**retracts**) it exerts a push or a pull. The **force of an extending cylinder** depends on the **area of the piston** and the **pressure** driving it. You probably already know that the area of a rectangle is calculated by multiplying length times width. The shape of the end of the piston, however, is not a rectangle: it's a **circle**. The distance across a circle is called the **diameter**. When measuring the round piston inside the cylinder the pneumatics industry uses the term **bore**, which means the same thing as the inner diameter of the circle. The amount of area is measured in square units regardless of the area's shape. The formula for the area of a circle is expressed by the equation: $\mathbf{A = \pi r^2}$. The "r" is the symbol for the **radius** of the circle. The radius is always half of the diameter. Since the largest OCCRA cylinder has a bore of two inches, it has a radius of one inch. The symbol "π" is the Greek letter **"pi**" and is has a constant value of approximately 3.14. This means that the two-inch bore OCCRA cylinders have pistons with an area of 3.14 square inches. The force of an extending pneumatic cylinder is calculated by multiplying the area of the piston times the pressure of the air. Can you see now why we limit the air pressure in OCCRA to 60psi? If we used a pressure of 120 psi, the large cylinders would each extend with around 375 pounds of force! [since 120 x 3.13 = 375]

3. ELECTRICAL SYSTEMS AND OVERCURRENT PROTECTION

The flow of electricity always heats up the material that carries the current. The motors, wires, relay...etc. in your kit can all tolerate a certain amount of warming. A point can be reached, however, where the **current can cause a destructive level of heat**. To prevent this from happening, several protective measures have been taken in your kit. To start with, a size or "**gauge**" of wire has been chosen that will safely handle the anticipated current. Notice that the wires connecting to the battery are much thicker than the other wires in the kit. In general, the bigger a wire's diameter, the more current it can safely handle. The **battery wire is #6** (six gauge) wire that can handle up to **120 amps** of current. Most of the wires in your kit are **#14** stranded wire that should always carry less than **30 amps**. You have probably noticed that #14 wire is smaller than #6 wire and you have probably thought: "What's up with that?" Wire gauges work on an inverse scale, just like shotgun gauges: the bigger the gauge, the smaller the diameter. For teams using the old Victor speed controllers, the smallest gauge wires in the kit are those little skinny wires running to the fans on the older Victor 884 speed controllers are only about a #22 gauge.

The second way we prevent excessive heating is by the use of two different kinds of **over-current protection devices**: **fuses and circuit-breakers**. On the outside wall of your control box (where the battery cables enter) is a 120-amp circuit breaker. [Or teams may use the older 80-Amp breaker which can be seen in See Fig.4] This device is like a sentry that stands guard over your robot's electrical system. Should the entire system suddenly begin to draw 120 or more amps of current, the circuit breaker will turn itself off and save your robot from becoming a 120-pound toaster. A little plastic lever will swing out from the side of the circuit breaker to let you know that the breaker has been tripped off. To **reset the circuit breaker** (once the problem that caused the breaker to trip off has been fixed!), all you need to do is click the little black lever back up into its original position. This circuit breaker can also be used as an on/off switch to power down your robot in a hurry (some call it a "kill switch").

Figure 4. The 80-Amp or 120-Amp Circuit Breaker attaches to the side of the robot control box.

Pushing in the little **red button** that sticks through the hole in the plastic will **turn off all power** to your robot instantly. You would be wise to design your robot with a protective structure around the kill switch so that other robots do not accidentally bump your kill switch and shut you down during a match.

Make sure you leave enough of an opening, however, that a person's arm and hand can easily reach the switch in an emergency. Engineers from General Motors Powertrain, who helped design most of the electrical control system in your kit, suggested this safety device. Make sure everybody on your team knows this way to shut off power in an emergency! We are supplying 6 other circuit breakers to protect the 6 largest motors and their speed controllers: the drill motors, the CIM motors, and the wiper motors (either Denso, Bosch or Valeo). These “slow-blow” circuit breakers come with either 35 Amp, 30 or 20 Amp ratings.

Figure 5. (From the left) Slow-Blow 30-Amp Circuit breaker, 20-Amp Fuse from a Spike Relay, and 30-Amp Fuse from Power Block.

We had a lot of complaints from teams during the first OCCRA season in 2000 about the 30-amp fuses. The concern was that they blew out too easily during momentary rushes of current, leaving many robots dead on the field or spinning in circles. Drivers’ training can partially solve the overload problems that come from “**stalling**” motors, but we’ll get to that in a moment. The circuit breakers that we have added (See Fig.5) will trip off a little slower and will reset themselves in a few moments after they have had a chance to cool, so teams that overload a motor might still have some time to recover before the end of a match. Notice that the slow-blow circuit breakers have breakaway tabs so that their length can be adjusted: please do not break away any more of the tab: we already have them broken away to the length that fits best. The third way we protect your electrical system is with **fuses**. Little plastic 30-amp fuses are positioned like sentries on the main power distribution block and 20-amp fuses guard the relays. (See Fig.5) These fuses stand guard over each of the output circuits on your robot, ready to give their lives to protect the motors they serve. Unlike the circuit breaker, there is no resetting a fuse: if too much current flows through a fuse, the little guy is destroyed and must be replaced.

Pull out one of the fuses and take a close look at it. You should be able to see a little wire-like metal strip inside the plastic that runs between the two terminals. Now take a **multimeter** and set it to the "**Resistance**" or "**Ohm**" scale. (See Fig.6) A blown fuse, like the one pictured in Figure 6, will give an almost infinitely high resistance reading. Many digital multimeters show this with a “1” or an “OL” in the display.

If you don't have a **multimeter** you should go out and get one. They are very useful for troubleshooting problems and you can find them for under $15. Most schools have multimeters in their science departments.

Touching the two wire leads of your **ohmmeter**/tester to the 2 terminals of the fuse should show that the fuse is intact by registering almost zero ohms of resistance. Many ohmmeters will buzz to show that there is **continuity** (a continuous path for the electricity) in a circuit. A defective or "blown" fuse will register an infinitely high resistance on an ohmmeter.

Figure 6. Multimeter test of a blown fuse; when resistance is too high for a digital meter to read, it will often display a steady "1" or the letters "OL".

The final things to consider here are the reasons **why excessive current might flow** in your circuits. The first culprit is the dreaded "**short-circuit**" condition. Recall that electrons had to be pushed through the circuit since all elements in the circuit are going to be resisting the flow of electricity. What would happen if those resistances were suddenly removed? It is kind of like the time you were pushing real hard on a wrench to loosen a stubborn bolt and the bolt suddenly stopped resisting (came loose): what happened? You probably smashed your knuckles in the sudden surge that followed. Likewise, if the resistance of a circuit suddenly drops, a large increase in current results. Mathematically, **Ohm's Law** describes this: **Current equals Voltage divided by Resistance**. If there are 12 volts divided by 6 ohms you get 2 amps of current. Look what happens if the same voltage applies but the resistance drops to one-tenth of an ohm: now we suddenly have 120 amps of current (12/.1 = 120)! What might cause this dramatic loss of resistance? Usually it happens when somebody accidentally lets a metal object form a bridge between one of the positive and negative wires on the robot. It can also easily happen if there is a worn spot on a wire or if too much insulation has been removed from some wires. These are all things to watch out for.

A less obvious cause of excessive current flow occurs in the motors themselves. When electricity flows through a motor it encounters a kind of resistance (**reactance**) because of what's called "**counter-emf**". A turning motor creates an electromagnetic field that opposes the current flow inside it. It is kind of like when you push on a door to open it: the door also pushes back on your hand. With motors (as you'll soon see), they must be free to spin. If you put a voltage across a motor that is not free to spin (or spinning very slowly), there is very little resistance inside the motor itself and a large increase in current results. We say that the motor is "**stalled**". The **high current level** during a stall is referred to as "**stall current**". You will see this situation frequently during OCCRA matches: your robot is attempting to lift its arm to score, but an evil opposing robot blocks the arm. Your driver gives your robot arm's motor full power in an attempt to overcome the blocker, but nothing happens. Your motor is stalled and the current is surging in it. Welcome to the robotic version of Drivers' Training 101: if you are lucky, the fuse or the circuit breaker stops the current in time. If not, the magic white smoke that lives inside your motor begins to escape and your motor begins its new career as a paperweight.

4. MOTORS

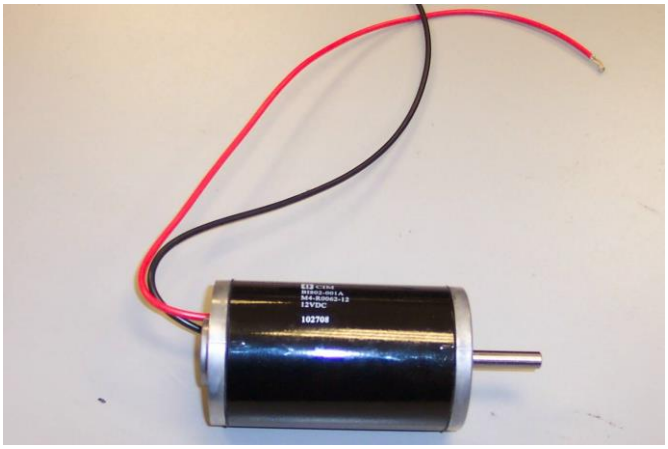
Your kit contains about a dozen electric motors. OCCRA allows your team to use any motor from 4 different sources (The Robot Space, AndyMark, VEX or Banebot) that produces less than 360Watts of power and costs less than $100. There are 5 different motors supplied in the Kit of Parts; you will be able to use as many of any kit motor that you would like to use! Teams are cautioned to think carefully about your choice of motors: larger is not always better! Your robot is still limited to 120 Amps of current and you only have 7 Victor speed controllers at your disposal. Likewise, you are facing a weight constraint, so you probably will not be able to use big CIM motors everywhere you want to!

Motors are all capable of taking the electrical energy that you send to them and producing a rotational motion. The general principles for motors are fairly straightforward:
When electrons flow through a wire, they create a region around the wire called a "**magnetic field**". This magnetic field is pretty weak for a straight piece of wire. If, however, a wire carrying electric current is wound up into a coil, the magnetic field gets concentrated into a much smaller volume and the magnetic field therefore gets strengthened. The coil continues to produce this magnetism as long as the current continues to flow. If you reverse the direction of the current flow, the end of the coil that had created the **north magnetic pole** will now have a **south pole** instead! Likewise, the other end of the coil changes from being a south pole into being a north pole. The DC motors in your kit all contain coils of wire attached to a shaft forming an **armature**; the armature coils become magnetized when electricity gets sent through the motor. The motors are structured such that a stationary or "fixed" magnet's north pole faces the coil's magnetic north pole and the 2 poles want to push away from or "**repel**" each other. Since the fixed magnet can't move, the armature's nort h pole spins away instead. Through a simple (yet clever) device called a "**commutator"**, the direction of current flow in the armature gets reversed right before the armature's south pole reaches the fixed magnet's north pole. Once again, two north poles are suddenly facing each other and, once again, they repel each other causing the armature to continue spinning away. This process keeps repeating as long as there is current flowing through the armature coil.

The term that describes and measures a motor's ability to produce rotation is called "**torque**". Think of torque as being like the "turning strength" of a motor. [Torque is actually a little more complicated than this; it's the vector product of two measures: the radius vector from the axis of rotation to the point where the force gets applied, and the amount of the output force itself. This explains why torque measurements are always expressed in units like "foot-pounds" and "Newton-meters".]

Internal electromagnetic forces act on the armature coils to produce the torque that made the shaft spin. The output shaft of the motor likewise delivers a torque to objects it encounters. A motor spinning at a certain rate has a shaft that can deliver a fairly constant torque. If the motor shaft is attached to a sprocket, the sprocket will begin to rotate also. This **torque can be transmitted by chain to other sprockets** and devices. When you ride a bicycle, the force of your foot on the pedal causes a torque to be delivered to the front sprocket, making the sprocket rotate. A chain links the front and back bicycle sprockets so that the rotation of the front sprocket delivers a torque to the back sprocket to makes it rotate also. Your kit motors will produce the torques needed to produce motions of your robot.

The **most powerful motors** in your kit are the **Dewalt drill motors** and the **CIM motors**. (See Figs. 8a and 8b) Most teams will use two or more of these motors to drive their robot's wheels. If you use two of these motors to power your drive train, your robot will have over 600 Watts of power moving it along the carpet. If you use four CIM motors, your robot will have over 1,200 Watts of power! The question is: how much power do you really need to do the strategy that your team has devised? Maybe some of that power (and weight!) can be better used on an arm, or an elevator system, etc.



You can use different wheel sizes, sprocket ratios or gear ratios to create the balance of torque and speed that you want. It is important to remember, however, that **your total maximum power is a constant for whichever set of motors you choose to use.** Changing the sprocket ratio to make your robot push with greater strength (increasing its torque) means that the robot's speed will be proportionately reduced since rotational speed times torque equals power.

Figure 8a. The CIM Motor. We strongly recommend that you use the CIM motors to drive your robot's chassis.

Although your motor allocation depends upon the design strategy you are employing, only the CIMs or the Dewalts are likely to have enough power to move your robot around very fast. The three most powerful motors in your kit (the CIM motors, the drill motors and the Denso/Valeo wiper motors) must be hooked up to the **Victor speed controllers**, not the **Spikes**. These motors can potentially draw more current in normal operation than the 20 amps that the Spike relays can safely handle.

Figure 8b. The Dewalt Drill Motor (no longer stocked by OCCRA)

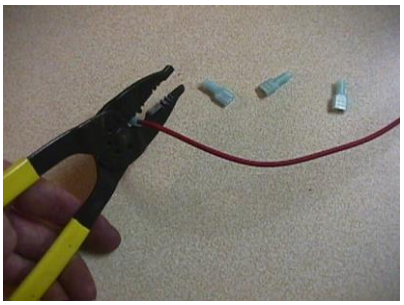
Motors can be built to run on alternating current (AC), like you would find in the wall outlets and lights of your house, or on direct current (DC). **Batteries** are the most common **DC source**. The motors in your kit are all **DC motors** and a **12V lead-acid battery** is used as the DC electrical power source.

The shafts of all motors need to be supported so that there are no side loads (also known as "**radial loads**") that could cause excessive wear on the motor's internal bearings and overloading of the motor itself. Your kit contains 3/8-inch diameter brass bushings that fit nicely on the motor and sprocket shafts. If you mount one of the bushings into a piece of wood or angle aluminum (easy to do: just drill a ½" hole and jam the bushing in!) and then align the mounting material and the bushing so that the shaft goes right into the bushing's opening, you will support the shaft and protect the motor. Look at Figure 9: A vertical wood mount supports the motor shaft from side-loading. Notice the brass bearing just to the right of the collar on the end of the shaft: the shaft spins inside of this bearing. The bearings that OCCRA supplies do not need grease or oil inside the bearing since there is oil impregnated into the bearing to make the shaft spin even easier. For heavier side-loads on a shaft (such as on your drive train) it is wise to mount supports with roller bearings or brass bushings on *both* sides of the load.

Figure 9. Supporting a Motor Shaft.

Refer to any text that deals with mechanical drive systems (such as *The Robot Builder's Bonanza*) and you will find suggestions for making axle and shaft hookups. A good text that shows you how to align sprockets, adjust chain tension...etc. will also enable you to mount your motors to perform more efficiently. The ***Toughbox*** transmissions in your kit make it easy to align sprockets and adjust chain. Use the slotted mounting holes on the transmission's aluminum mounting bracket (supplied with the ***Kit-bot*** chassis) to pull the chains snug enough so there is only about a chain's width (1/4") of chain movement when you grab the chain and wiggle it manually. The "How-To" video also covers this topic briefly. We cannot stress it enough: **motors should only be supplying your shafts with torque, not any support.** Support for motor shafts must come from somewhere else!

5. MECHANICAL ADVANTAGE



There are several types of **simple machines** that you should study before you design your robot. Many books group the simple machines into six classifications: levers, inclined planes, wheel and axles, screws, pulleys, and wedges. If a machine enables you to lift a 30-pound weight while using only 15 pounds of lifting force, we say that the machine has a "**mechanical advantage**" of "2". In other words, the machine made you twice as strong. If a different machine enabled you to lift the 30-pound weight using only 6 pounds of lifting force, this machine made you a lot stronger and we would say it had a mechanical advantage of "5". While machines can make work easier, you do not get something for nothing. Whenever you use any kind of a simple machine you face a trade-off: **the greater the mechanical advantage of the machine, the less movement the machine produces.** This usually means that the stronger it makes you, the less speed you will have.

Figure 10a. The Pliers is a First Class Lever.

Consider the effects of different **gear ratios**. Notice from Appendix I that the **high gear** gives you a lot **more speed** but the **low gear ratio** gives you a lot **more torque**. Is it more important for your robot to be strong or fast? This is the trade-off you always face when choosing gear ratios and sprocket ratios: the faster you design it, the weaker it will be! The lever is another simple machine that you will likely use on your robot. Imagine a stick that is 100 centimeters long and has a hook attached to each end. (See Figure 10b.) Now, imagine that there is a pivot point (fulcrum) attached 20 centimeters from the left end. The stick is free to rotate around the fulcrum, so this can be called a "**lever**." If a 10-Newton weight is hung from the left hook, the lever lets us lift the weight using only 2.5 Newtons of effort force on the right hook. So, the lever made us four times stronger. Note that the distance from the pivot to the weight is only 20cm while the distance from the pivot to the effort force is 80cm.

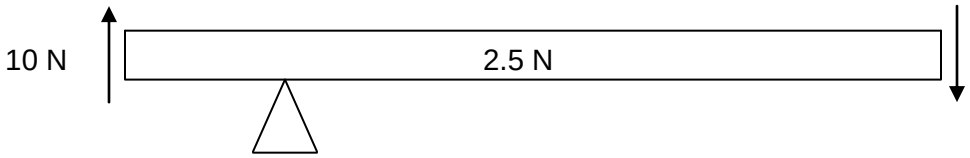


Figure 10b. The First Class Lever Can Give You a Mechanical Advantage.

Dividing 80cm by 20cm gives **the mechanical advantage** of "4" which we already found. Although the lever made us 4 times stronger, it is important to note that we "lost" something here: we had to pull the stick down for 4cm just to lift the weight up 1cm!

The pliers in Fig.10a is a **lever**, too: the farther from the **pivot** (**fulcrum**) that you grip the handle, the greater the crimping force on the other side of the pivot point.

Soon you will be running **roller chain** (See Fig.11) between **sprockets** of differing sizes. When a torque causes the first sprocket to rotate, the chain will transfer the energy to a second sprocket and the second sprocket will rotate also.

The first sprocket is called the "**driver**" sprocket and the second one is called the "**driven**" sprocket. There is a simple way to calculate the "**ideal" mechanical advantage** of any pair of sprockets: just compare the number of "**teeth**" on each sprocket.

Figure 11. #35 Roller Chain

You do not have to actually count the teeth on your sprockets (or gears) in order to calculate the mechanical advantage: **the number of teeth** is often stamped onto each sprocket. You have different sizes of sprockets in your kit: 60-tooth, 45-tooth, 30-tooth, 24-tooth, 15-tooth, 12-tooth and 9-tooth. You may buy additional sprockets with a wide variety of tooth counts from outside vendors (subject to the $90 restriction for spare parts). To calculate the **mechanical advantage** of a pair of sprockets, simply **divide the number of teeth on the driven sprocket by the number of teeth on the driver sprocket**. This is called the "**sprocket ratio**". For example, suppose a driven sprocket having 60 teeth is attached to your wheel, and the driver sprocket having 15 teeth is attached to a motor. This arrangement has a mechanical advantage of four. This means that the wheel's shaft (the driven sprocket) has four times more torque than the motor's shaft (the driver sprocket). This increase in strength comes at a price: the wheel's driven sprocket only spins one-fourth as fast as the motor's driver sprocket. You will need to choose the sprocket ratios for your robot based on what you need to accomplish.

It might seem from this explanation that you are limited to a **maximum sprocket ratio** of approximately 6.7 (60 divided by 9.) This is true if you are only using two sprockets at a time, but it is possible to use more than two sprockets and axles. For example, suppose Sprocket "A" is the driver for a Sprocket "B". Now imagine that the axle that is holding "B" (the driven sprocket) has an additional Sprocket "C" on it. Finally, suppose that Sprocket C has its own separate chain that is driving Sprocket "D" (a fourth sprocket with its own axle). We now have a three-axle system where the sprocket ratio of "B to A" is ***multiplied*** by the sprocket ration of "D to C"! For example, suppose Sprockets A and C each has 15 teeth while Sprocket B has 60 teeth and D has 45 teeth. The sprocket ratio of B to A is 60/15 = 4 and the ratio of D to C is 45/15 = 3. Therefore, the ratio of the entire system is 4 x 3 = 12. Sprocket A has to rotate 12 times to get Sprocket D to rotate once, but Sprocket D's axle has 12 times the torque that the axle of Sprocket A supplied.

Figure 12a. Sprocket with Split-tapered Bushing

Some of the new chassis kits use belt drive systems, instead of sprocket and chain, to transmit power from the transmissions to the wheels. The rules discussed above still apply but the terms "pully and belt" are used in place of the terms "sprocket and chain." Each pully has notches called "teeth" embedded in their aluminum wheels. These function the same as teeth on a sprocket. Likewise, the timing belts have indentations to mesh with the pulleys. To calculate the **mechanical advantage** of a pair of pulleys, simply **divide the number of teeth on the driven pulley by the number of teeth on the driver pulley**. This is called the "**pulley ratio**".

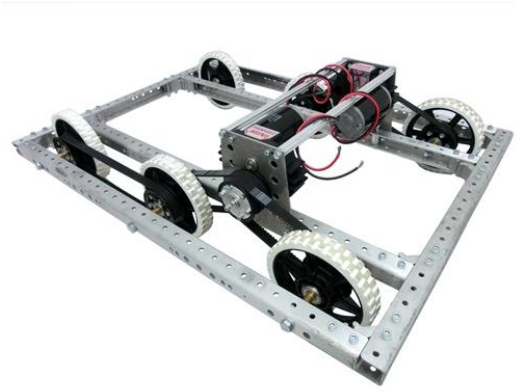


Figure 12b. The entire C-Base Chassis Kit by AndyMark, complete with 6-wheel belt drive

The 9-tooth sprocket was included in older OCCRA kits to attach directly to a drill's output shaft by threading onto it, so it did not need an attachment mechanism. Now, most teams use CIM motors with kit transmissions or Gear motors with attached transmissions to get the job done. The 12-tooth sprocket is too small to be used with the split-tapered bushings that mount inside the other sprockets. ("G" bushings are used with the 15-tooth sprockets and "H" bushings are used with the 30, 45, and 60-tooth sprockets.) The 12-tooth sprockets will

need to be pinned to shafts or mounted onto ¼" shafts using devices called "**tran-torques**." An explanation of how tran-torques and bushings attach to shafts will come later in the section on kit motors.

III. THE CONTROL SYSTEM

1. THE CONTROL PANEL AND JOYSTICK MODULE

Your drivers will control the actions of your robot from the Joystick Modules on the control panel [see Figure 13a.) Your control station consists of a board, 2 joystick controllers connected by an RJ-22 phone-style cable, and a VEXnet 802.11g USB key that functions as a radio transmitter and receiver. For a power supply, teams have all been given an AC to DC converter that plugs directly into the joysticks. Each team has its own sets, but OCCRA will supply them at the drivers' stations at tournaments. We suggest that you still keep freshly charged AAA batteries inside the Joystick Modules as a back-up in case your power gets interrupted somehow. Teams were given a 1" by 6" by 20" piece of board to serve as the mounting surface for your control panel but any comparable material can be used. When carrying the board to and from the drivers' stations, teams should have their Joystick Modules securely attached to the board with Velcro, rubber bands, etc.

Figure 13a. Partnered Joysticks

Since your **control panel** mounting-surface is probably going to be made of unfinished particleboard, your team may decide to **decorate** it or **paint** it in your school colors. This is allowed as long as you remove all of the electronic components first and then return them to the layout arrangement later. You should spend no more than a week planning and designing before you start fabricating your robot's chassis/drive system; this would be a good time to have a sub team working concurrently on your control panel so that it is finished by the time you need it for driving. Returning OCCRA teams are allowed to use the control panels from previous years. Except for the chassis and the electrical control box, the robots must be totally rebuilt each year.

Moving a joystick is essentially like turning a dimmer switch on a light, while pushing one of the joystick's buttons is like flipping a switch: both send an electrical signal from the Joystick to the Cortex. Sensor inputs on the Joystick Module detect which switch has been thrown [See Fig. 14.] The microprocessor inside the Joystick then assigns and forwards that information through the unique routing or "**channel**" that had been assigned to that particular switch.

After the Joystick Module assigns **data** from a switch or **sensor** and forwards that information through a channel routing, the signal is then sent to the transmitter.

Figure 13b. The Joystick Module

Each Joystick Module has a twelve buttons and two “sticks” on it that supply the input information for the robot. If you are using one of the **default programs**, the sticks and buttons of each joystick control the ten Motor Outputs on the Cortex, according to the chart in Appendix V. Notice that not all of the buttons do not get used in the default programs. There are 4 different default programsthat come with the Cortex; which one gets used depends upon the location of a little, 3-pin “jumper” which will be described later. There are no **digital** outputs to the Spike relays in the default program, so teams wishing to use digital control of devices (as done with pneumatics) will need to modify the default program. The joysticks provide a second type of input signal to the robot: moving the joystick forward or backward provides an **analog** input that controls the **motor outputs’** signal to the Victor Speed Controllers which, like the Spike Relays, can send current to the motors for forward or reverse operation. Unlike the relays, the **speed controllers can deliver varying amounts of current** to the motors, depending upon how far the joystick has been pushed forward or pulled back (more on that later).

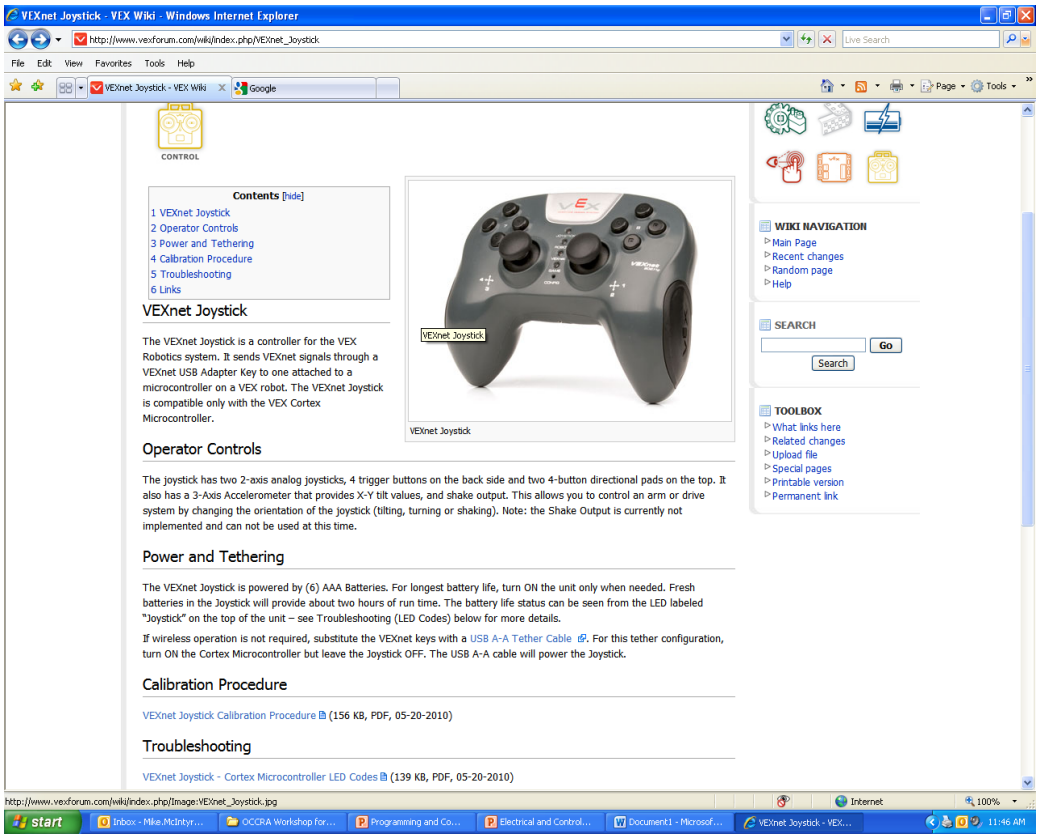


Figure 13c. Joystick Module

There is a constant current source aboard the joystick control module that routes current through the joysticks. Moving a joystick forward and backward (the "Y' axis) causes a variable resistor to change the voltage that is dropped across an internal resistance that gets read as that particular analog input. Moving the same joystick left and right (the "X" axis) causes a different variable resistor to change the amount of voltage read as a different analog input of the operator interface. In OCCRA, we now allow changes to the default program, so teams will have a way to use the inputs caused by sideways joystick motion. **In the default “tank drive” programs, moving your joysticks along the X-axis will not do anything.** Pushing one of the buttons on a joystick controller flips a switch. This, too, provides an electrical signal that the joystick module interprets, converts, and sends to the Cortex so that the appropriate motor controller gets sent an electrical signal.

Make sure that there are **no perpendicular forces acting on the VEXnet USB Keys or the cables** that run from the joysticks. Even fairly small movement can make for a bad connection and a very quirky robot.

The **“Cortex Microcontroller and VEXnet Joystick User Guide”** comes with the OCCRA Kit of Parts but can also be found at: http://content.vexrobotics.com/docs/VEXnet_Cortex_UserGuide_081811.pdf All teams are *strongly* encouraged to read through this document! It explains: the Cortex Microcontroller and VEXnet joystick pairing procedure, the basic connections of the VEX control system, how batteries are utilized, how joysticks can be calibrated, and how the default program operates (including a mapping of the joystick inputs to the motor outputs.)

When the Cortex and joystick controller have been properly linked, the VEXnet LED will be blinking fast green on both the Cortex and Joystick. [See Figure 14.] The VEXnet light is the only LED that determines a valid link. It usually takes 5 to 10 seconds to successfully establish a link. Once the units are linked, the Robot and Joystick LED Indicators will show the battery levels in their respective unit. A green Robot or Joystick LED indicates that their respective batteries are fully charged batteries. As the battery levels decrease, these LEDs will change to yellow and then red.

Figure 14. Linked Joystick and Cortex

When driving your robot, each movement of the joystick causes a signal to be sent via the VEXNet USB Key. This signal turns on one of the outputs leaving from the Cortex Controller aboard the robot. These outputs go to various motors via the relays and speed controllers and cause the robot to move.

What follows is a more in-depth explanation for those of you who are inquisitive.

The electrical message from the joysticks goes into its internal processor module as a 0 to 5 volt signal which gets encoded into what is called an "RS422" signal (a 0 to 5 volt digital signal, similar to what computers use). This signal consists of bursts of data referred to as "packets" of information that get sent out at 19.2 "kilobaud" (the transmission rate expressed in terms of how many thousands of bits of information are sent per second). A bit is the most basic unit of information storage capacity that is used in computers; it represents either a binary 0 (the "LOW", or near-zero voltage) or a binary 1 (the "HIGH" voltage value of one) as I explained earlier. Every 25 milliseconds you get another packet that contains 26 bytes at a time with 10 bits per byte. Two of the bits are unused and just designate the start of each byte (the start bit is always a LOW) and the stop of each byte (the stop bit is always a HIGH). If you were to examine one of the bytes that are inside one of the packets, then, you would find that the first bit is a LOW. Then you would find that there are 8 data bits: these are where the data is stored that describes the joystick's position.

Only 24 of the 26 bytes in the packets actually contain any data. There are 2 that are set aside to identify the start of each packet. A movement of a joystick sends a signal reflected by a changed voltage. The location of the respective bytes within the packet lets the joystick's processor know which channel is doing the talking. It's almost like the packet's information is stored in a bunch of drawers. The computer knows that the contents of the first drawer are always the value for channel one, the contents of the second drawer is always the value of channel 2...etc. Inside of each packet, 12 of the bytes are assigned to the analog channels and 2 of the bytes are digital stuff (so there are 16 bits of digital information). Also, the team number, channel number and a lot of other miscellaneous information, including bi-directional information (the robot can also send information back to the joystick module!) are part of each packet. For example, analog sensor #1 is in the exact same location within the packet as analog output #1...etc. There is no stop signal for the packets: the processor simply sits and waits for the next "start" byte to arrive. After a certain amount of idle time, it knows that the packet has ended and it is time to prepare for the next packet. *Although you are allowed to alter the default program, I suggest to programming teams that you design your robot so that it can, with only minor rewiring, still run using the default program. Do not expect OCCRA officials to hold up matches because you are having trouble with your program. Likewise, do not expect OCCRA staff to assist you with debugging your program at tournaments.*

For those of you who know something about how computer code and binary counting works, suppose that the joystick was at the full forward position at a certain instance. At that moment in time, the 8 data bits within the byte that's being encoded would all be at the binary "1" value. This is the equivalent of the decimal value 255 or "FF" in hexadecimal. This message will eventually reach the CPU inside the Cortex where, (assuming the

CPU is running the default program), it will be interpreted to mean: “tell the speed controllers to send maximum possible voltage to the motor."

2. THE VEXNet 802.11 USB KEY

OCCRA is now using only the white VEXnet 2.0 transmitters. One of these transmitters is attached to your joystick module and sends a radio signal to your robot. Your robot's receiver, another VEXnet USB key that’s attached to the **Cortex** (see Figure 13a.), listens to the frequency of its own transmitter, and carries the signal to the onboard computer, the Cortex.

The Joystick Module sends the information from the drivers to the VEXnet transmitter key [See Fig.15a] This message gets encoded and sent out as a 900 MHz electromagnetic signal. It travels through the air where it is received by the robot's VEXnet USB transmitter key and is then decoded by the cortex. You don't even have to use the robot's VEXnet USB transmitter keys to communicate between the Joystick Modules and the Cortex: a direct tether connection using a USB A to A cable can carry the message instead. The tether line that came with your kit is just a USB A-A computer cable that’s about six feet long. When you want the Joystick to control the robot without using the VEXnet USB transmitter keys, simply hook the tether to the ports where the USB keys normally go. When attached, the tether automatically overrides the VEXNet USB Keys.

Figure 15a VEXnet Key 2.0

How the VEXNet USB Keys accomplish the encoding and decoding is outside the realm of this paper. The important thing to understand is that the signal sent from the joystick module to the Cortex can either go by direct cable connection (the tether line) or by radio waves.

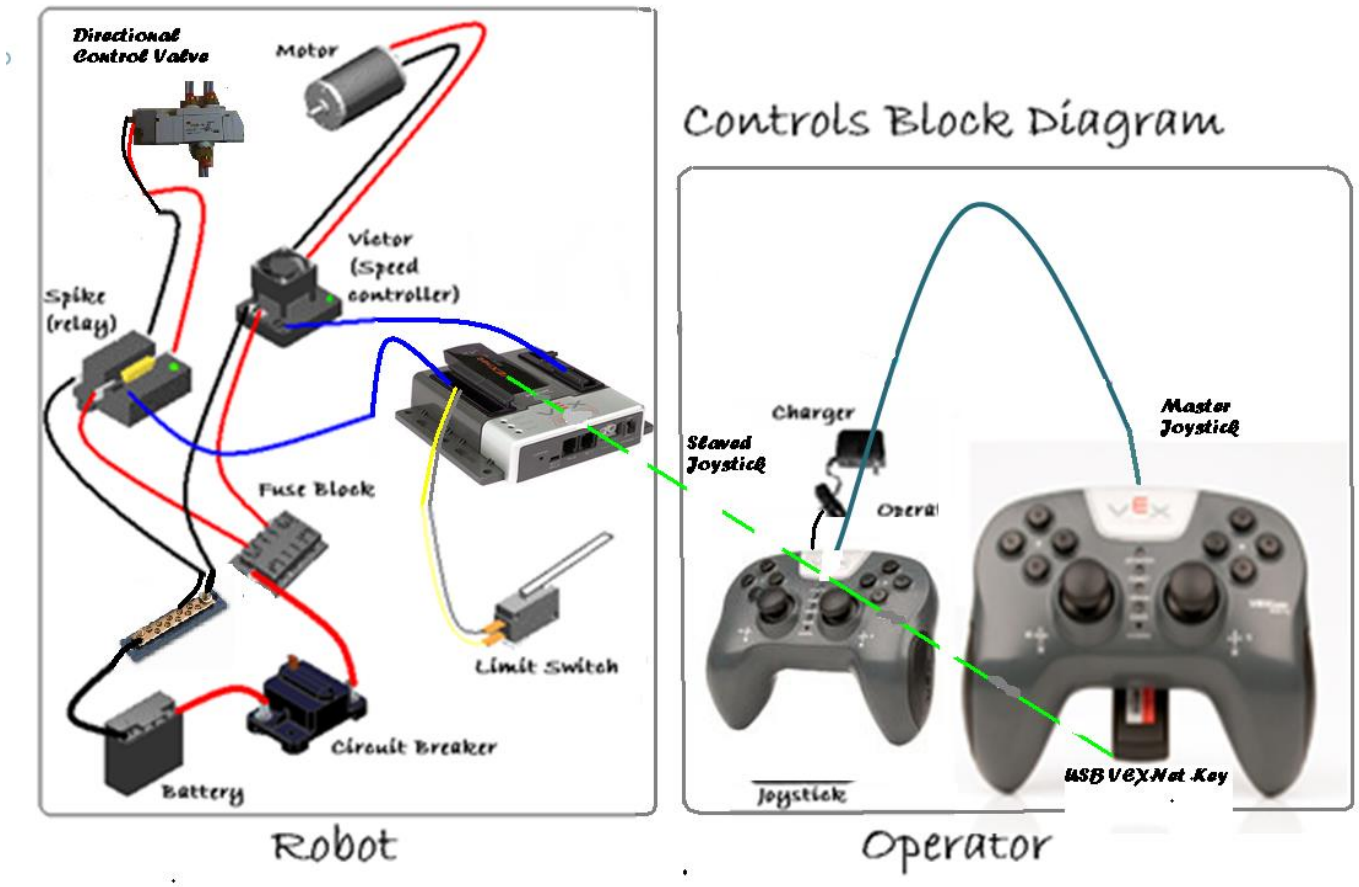


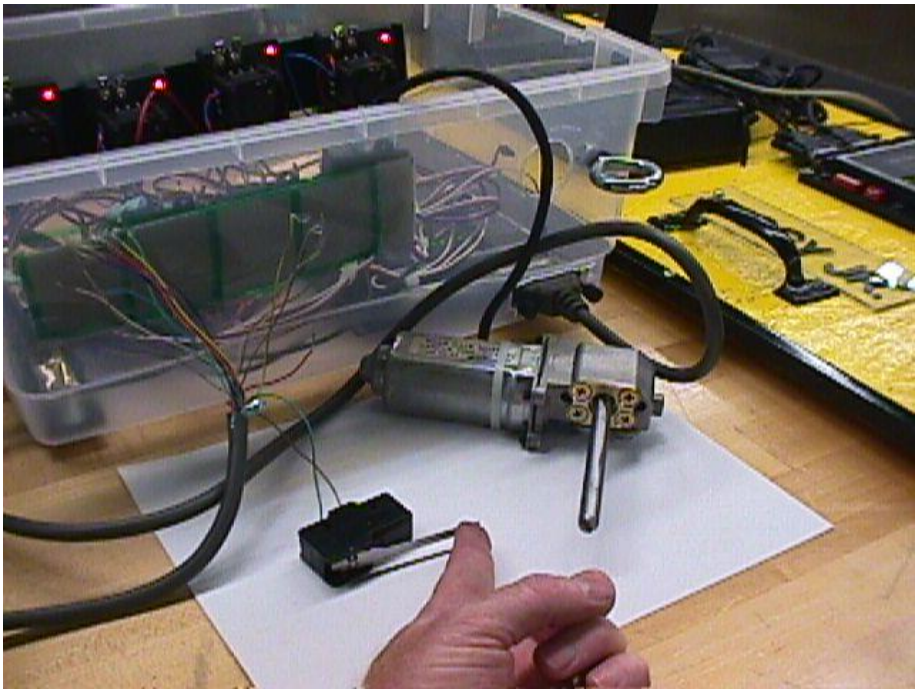
Figure 15b.

3. THE CORTEX CONTROLLER

The Cortex robot controller takes in the information that was sent by the Joystick Module, decides what to do with each piece of information, then sends electrical signals to the speed controllers and relays so that motors can be controlled. [See Fig.16] The default program that comes with your onboard computer (Cortex) can run the controller and listen for two inputs that control two of the relay outputs. This program will enable your student drivers to send instructions to your robot and make your 'bot do some rather amazing maneuvers. If the default program was altered, these robots could even be programmed to be totally autonomous. For OCCRA 2016, you can once again customize the basic robot program to fit your robot's needs, but I'll get into that later.

The Cortex is the robot's controller. Using some built-in instructions (the **"RobotC" program**), the Cortex controller sends an electrical signal to the **output** terminal that the program has designated. This signal is sent to either an electric **relay** ("**Spike**") or **speed controller** ("**Victor**") which then activates one of the robot's motors or valves.

The Cortex's CPUs (central processing units) have multiple jobs to do. One CPU acts as Master Processor and Output Processor to communicate with the inputs and outputs. The other CPU processes all of the data and, based on the software program that it is running, tells the Output Processor what to do with all the information. The Output Processor then sends a PWM code signal to the Victors and a relay code signal to the Spikes. The Cortex robot controller has a lot do and it does it 40 times each second.



For a complete list of the many capabilities you can consult the Users Manual as listed earlier in the section.

Figure 16a. A quick test of an input for the Cortex Robot Controller: if the limit switch is pressed, the Keyang motor stops its forward rotation but can still run in reverse.

4. SPIKE RELAY MODULES

The relay modules are small, square, electronic boxes that are found inside of the robot's control box, fastened to the walls with Velcro tape. [See Fig.16b and c] They have the name "SPIKE" written on them. You should have seven of these Spikes inside of your control box. The Spikes are relays, so they act like remote controlled electrical switches. They enable weak electrical signals from the Cortex Controller to turn on or off large current to the motors; the Spike's signals can also be used to control directional control valves (DCVs) in your pneumatic system. The Spikes can drive your motors in forward, reverse, or off. The output terminals on the Spike that go to the motor are labeled "M+" and "M-". If you have a motor that runs the opposite direction from the way you want, one solution is to swap the wires connected to the M+ and M- terminals of the Spike.
You can hook up the smaller of your kit motors (the ones that draw less than 20 amps of current) to the Spikes, but never hook the CIMs, Gearmotors, drill motors or wiper motors since they can draw well over 20 Amps of current under normal operating conditions. If you have a design feature on your robot that requires two motors to always run simultaneously, you can run two motors off of the same Spike using "splitter" attachments on the Spike's output terminals.

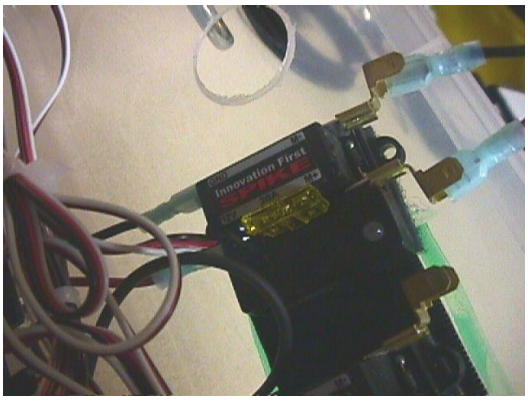


Figure 16b. A Spike Relay With Brass Splitters Attached to the Outputs.

When running wires from a motor to a Spike or to a Victor speed controller, make sure that you provide "strain relief". This is done by taking the entire bundle of output and input wires before they enter the hole in the control box and forming them into a loop. Now, snuggly wrap a plastic tie-wrap around the base of the loop so that it keeps its shape. Finally, take a second plastic tie-wrap and secure the first tie-wrap (the one holding the loop the wires) to the strain relief bolt on the side of the control box. Now you can connect the wires to the

Spikes and speed controllers. ***This strain relieving is very important: do not skip this step!*** This will save the Spike and speed controller connections from experiencing traumatic forces should there be a tug on one of the wires.

The Spike, also referred to as an “H-Bridge Relay Module,” is small enough to be remotely mounted almost anywhere on your robot but OCCRA rules require it (and all other controllers) to be safely mounted inside the clear plastic control box. Each Spike is opto-isolated at the signal input to protect the Cortex Controller against motor noise and return currents.

The Spikes get their input power from the 12-volt battery that hooks up to the terminals labeled "12V"(the positive terminal) and "GND" (the negative, or ground terminal). Your system has been pre-wired for you. However, if you do any work on the wiring, it is very important that you do not connect the 12V and GND backwards: it will destroy the Spike! There is a 20-amp fuse built into the Spike to help protect the relay from excess current flowing through it, but don't count on it. If there is a short circuit on the Spike's output, it is likely that the relay will be destroyed anyhow.

Figure 16c. The Spike Relay

The Spikes are controlled by a three-wire interface that connects to the Cortex Controller. These relay wires form a cable that is usually colored red, white, and black. The cable clips onto the Spike so that the black wire is on the "B" side and the white wire is on the other side, closer to the edge. When there is no signal coming from the Cortex, the motor will be off and in a self-braking mode (there will actually be +12V applied to both the M+ and M- outputs). The little LED indicator light will glow orange (amber) so you know that the Spike is getting power but is not getting any signal. If the Robot Controller sends both a forward and a reverse signal to the Spike at the same time, both motor outputs go to GND (ground), the motor is off, and the indicator LED turns off. This should never happen. If only the forward input signal is on, the green indicator lights up, and +12V is applied to the M+ terminal, making the motor spin forward. If only the reverse input signal is given, the red LED lights up, and +12V is applied to the M- terminal, making the motor rotate in the opposite direction.

To test the operational characteristics of Spikes, you have to have a digital output programmed into your Cortex’s user code (the default program that comes with the Cortex only has motor outputs in its program.) Programming the Cortex with a software program called “Robot-C” is too involved to explain here but can be learned via numerous on-line tutorials or by attending the OCCRA workshops that we run each year. Assuming you have a digital output mapped to a joystick controller’s button in your user code (robot program), hook up one of your Keyang motors to the Spike's M- and M+ terminals. The tri-colored relay cable from the designated Cortex digital output should run to the designated Spike. [This is a good time to suggest that you have somebody on your team label the ends of each of the wires on your robot with a simple code so that you always know which wire goes with which terminals: You really don't want to be frantically tracing wire routes in the pits while trying to get ready for a match!] You should slide the terminal connector of the black wire lead under the M- terminal, and then connect the red wire to the M+ terminal. Cradle the motor so that it can't fall or contact anything when the shaft begins to turn. Power up your control panel and connect your battery; then try pushing the designated joystick button (channel) that controls the digital output to which you are connected. Notice how the direction of the motor can be controlled through the Spike but the speed is constant. This is not the same kind of control that is possible with the Victor.

There are four default programs to chose from that assign all of the variables that the Cortex Controller has to take into account; to add digital outputs that will control the Spikes will require that some additions be made to the default program (explained later in this manual.) You may change the program but, as I said earlier, we suggest that you design your robot so that it can still run with the default program should your customized program develop “issues” while at a competition. If you are anxious to do some programming or just want to see the source code, go to the VEX website or open the Robot-C software on a computer and you can view all of the lines of code.

5. SPEED CONTROLLERS

The speed controllers are found inside the robot's control box, fastened to the walls with Velcro tape and/or bolts and zip ties. They are rectangular, black devices. Some, like the older Victors, have a fan on top and "VICTOR 883" or “VICTOR 884” written on them. [See Fig.16c] OCCRA also started using “TALON SR” speed controllers two years ago and has added the new “VICTOR SP” controllers to the allowed speed controller list for 2016 [See Figure 16d.] The Victor SP was co-developed through a collaborative partnership between *VEX Robotics* and *Cross the Road Electronics*, makers of the Talon SR. OCCRA will begin stocking only the Victor SP in its inventory. The Victor SP has been chosen because of its lower price and because of numerous improvements to its design: it is less than half the size of previous models and has a sealed enclosure that prevents debris from getting inside (this is a big problem with the older Victor: metal shavings that fell inside past the fan would cause “short-circuits” inside!) Other improvements on the Victor SP are: its full aluminum case with passive cooling fins so fans are not really needed, its electrically insulated body allows it to be directly mounted to the robot frame with no fear of electrical shorting (which is usually not an issue in OCCRA since we use a plastic control box), its LED indicators blink proportionately to output speed for easier debugging, its illuminated "Brake / Coast Calibration" button enables one-touch setting changes and calibration, its embedded power and output cables can’t shake loose during a match, etc.

The Victors control the amount and polarity (positive-negative orientation) of the electricity that is sent to the motors. Unlike the Spike relays that are either full forward, full reverse, or off, the Victors can output a range of power values. The Victors adjust the amount of power they deliver to the motors based upon the signal they receive from the Cortex Controller. When the driver moves the joystick to its end position, recall how the Joystick Module sent the signal to the Cortex Controller. The Cortex lets the Victor know that maximum power is needed by forwarding a coded signal. The Victor then opens up and allows maximum current to flow to the motor. The Victor, then, acts like a faucet regulating water flow through a garden hose. If the joystick is moved to a reverse position, the signal to the Victor has a reversed polarity. Unlike the water faucet, the Victor can reverse the direction of the current it sends. The Victor can make the motor spin in reverse.

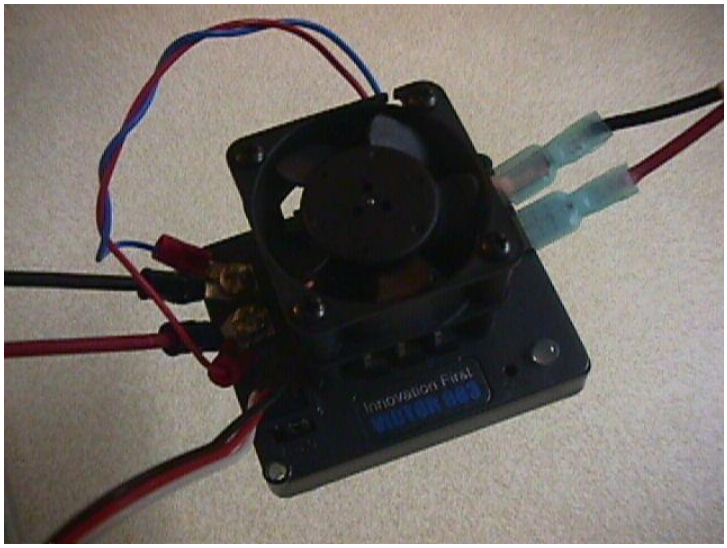


Figure 16c. A Victor Speed Controller with All the Hookups.

Try hooking up one of your motors to the M- and M+ terminal wires of one of your Victors. With older Victors, you slide the connector of the black wire lead under the M- terminal and tighten with a Phillips screwdriver, then connect the red wire to the M+ terminal. With the new Victor SP controllers, the wire leads are already attached internally. Cradle the motor so that it can't fall or contact anything when the shaft begins to turn. Turn on the power as you did with the Spikes earlier and then try moving the joystick that controls the output to which you are connected. With the default program, the left stick’s Y-axis on the main joystick will send a signal out of motor output #1, #2, and #3 (so put the signal cable into any one of these.) Notice how the speed and direction of the motor can be controlled through the Victor. Recall that this kind of control was not possible with the Spike.

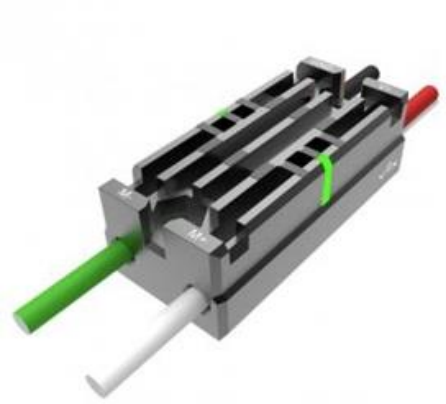


Figure 16d. The New Victor SP Speed Controller

I have already covered what you need to know about the Victor speed controllers, so what follows is only for those of you who crave detail:
Both the voltage and the current that the motors feel affect their performance. The level of continuous voltage, or "steady-state" voltage, across the motor's terminals determines the motor's rotational speed. Apply a certain voltage and the motors spin in a clockwise direction; reverse the direction, or "polarity," of the applied voltage and the same motor will now spin in the counterclockwise direction. The higher the voltage, the faster the motor will spin. To understand the effect of current you must first recall what is meant by "torque". The torque produced by a motor as it spins is determined by the amount of current that is flowing through the armature windings. Your kit motors will respond more or less linearly to both the voltage and the current.

Each of the Victors have a tiny CPU (computer) inside, complete with an FET (field effect transistor) switching architecture; older Victors also have an integral cooling fan. Basically, the Victors translate the PWM (power width modulation) signal that they receive from the Cortex Robot Controller into a PWM signal that the motors can use. This signal runs at around 2,000 Hz (this signal is audible: you can actually hear the Victors humming if you listen closely!) The transistors in the Victor, which are processing the signal, are actually turning on and off very quickly but at a constant frequency. The Victors are sending out a signal that can be thought of as train of square wave pulses. Although the frequency is constant, the width of each "on" pulse changes to reflect the desired output. Current only flows during the "on" pulses but the frequency is high enough that the motor's mechanical inertia doesn't have time to react to the changing wave pulse. Instead, the motor responds as if it was receiving a direct current that's the DC average of the square wave. A small movement of a joystick, for example, results in narrow signal pulses being sent by the Victor to the motor. This short "duty cycle" (the relative time spent in the "on" condition) means that the motor feels a small average voltage and therefore turns slowly. Greater movement of the joystick results in wider pulses and greater average voltages applied to the motor.

The directional control aspect of the Victors is due to its "H-bridge" circuitry. This name comes from the shape of the circuit when drawn schematically. The FET circuit acts kind of like a pair of DPDT (double-pole double throw) switches. Depending on the high-side/low-side conditions of the pair of switches, the motor can be given current flowing forward or in the reverse direction.