MagicBox

GAA Custom Electronics
Gordon Anderson

<u>Submitted:</u>
Cypress MicroSystems Contest
PSoC Design Challenge 2002

Abstract:

MagicBox is an accessory designed for the model airplane enthusiast. MagicBox connects between the aircraft receiver and the control servos as shown in figure 1. This device provides a number of functions that the pilot can select from a ten-position mode switch. Each function has adjustable parameters and options that are custom designed for the specific mode. The mode function allows the pilot to configure the MagicBox to his specific aircraft and needs. This device combines several functions into one package, you would need to buy 5 or 6 different modules to collect all of the functions available in the MagicBox. Each mode and a brief description are given below:

Mode	Description
0	*Position*, servo position control using pots (jumper D removed)
0*	*2 axis platform stabilization*, this mode is enabled by installing jumper D
1	*Cycle*, servo A cycling
2	*Isolate*, allows isolation and reversal of 4 servo channels
3	*Match*, allows servo output doubling with gain and offset adjustments
4	*Stable*, uses accelerometers to stabilize flight path
5	*Vtail*, supports rudder and elevator mixing
6	*Flaperon*, supports dual aileron, flaps, and differential
7	*Dualrate*, allows three channels of dual rates
8	*Glitch*, this is a fail safe and glitch buster function
9	*Mixer*, allows two channel mixing

MagicBox monitors the battery voltage on power up and will indicate, using a buzzer, if the voltage is low. The input signals are validated in every mode, if a signal is detected out of the valid range, it will be ignored. The servo outputs are opto-isolated, allowing you to use separate batteries for your servos. Jumpers internal to the MagicBox allow you to use the same battery if you wish.

One of the unique features of the MagicBox is the use of accelerometers for flight stabilization. The accelerometers are monitoring the attitude of the aircraft and will control the aileron and elevator servo positions to keep the aircraft on a straight and level flight path. This is a wonderful aid for the new pilot; if he gets in trouble, just let go of the controls and MagicBox will return the aircraft to level flight, like magic!

A block diagram of MagicBox is shown in figure 2. This is a very simple design with the Cypress PSoC microcontroller at the core. Servo signals from the receiver are sent into the microcontroller along with signals from a 2 axis accelerometer. Servo output signals are generated in the microcontroller and buffered through opto-isolators. This allows the servo’s load to be isolated from the receiver and the MagicBox.

Servo positions are controlled with a digital pulse. These pulses is nominally 1.5 mS in duration when the servo is in its center position. The servo extremes are 1.0 mS to 2.0 mS. The pulse rate expected by the servos is 40 Hz, or a pulse every 25 mS. The signal from the accelerometer is a pulse-width modulated signal that has been adjusted for a

MagicBox Connection Diagram

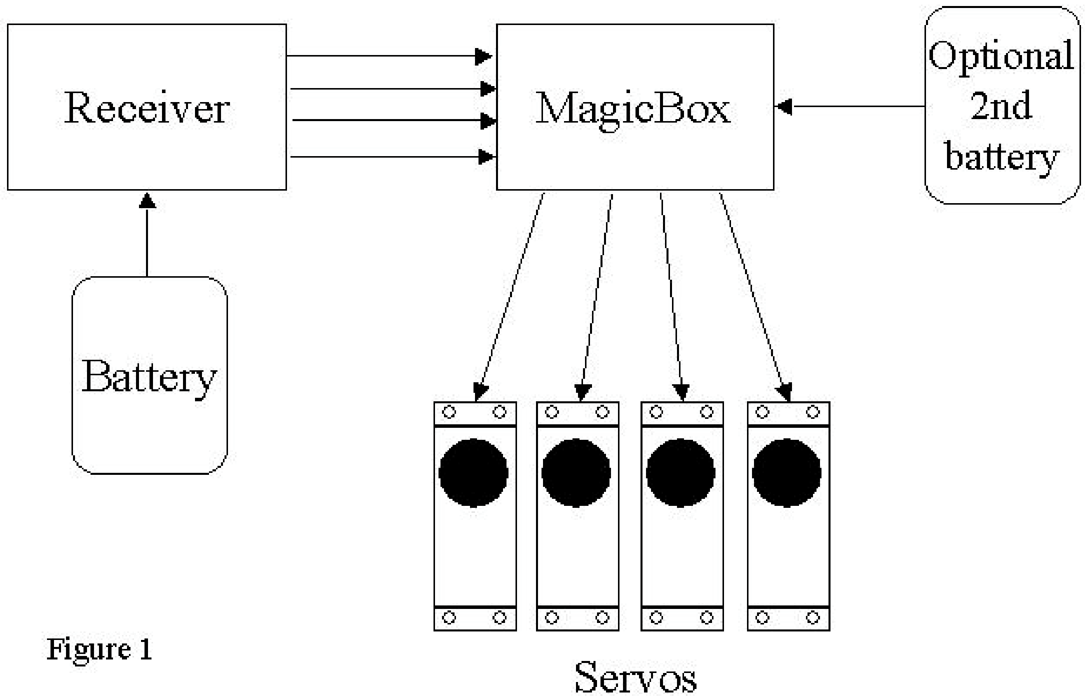


Figure 1

MagicBox Block Diagram

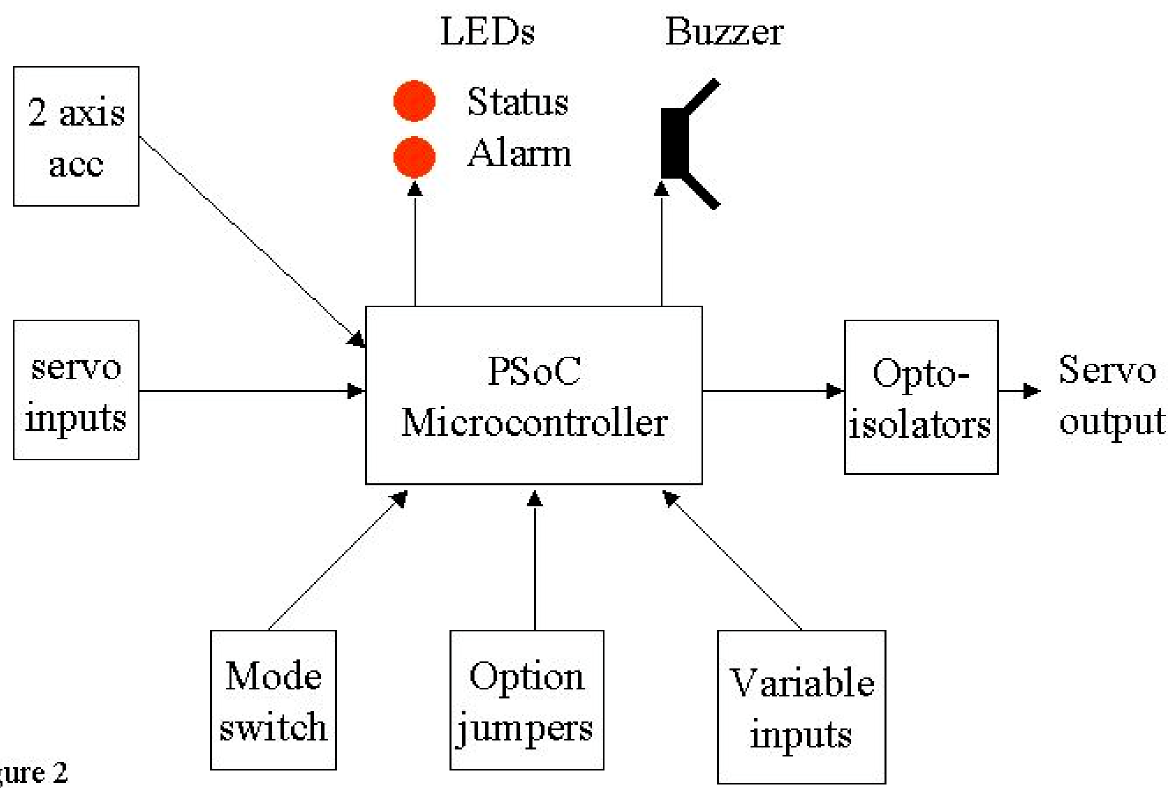


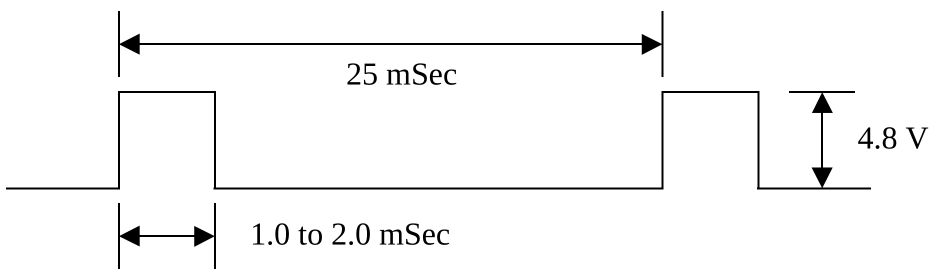
Figure 2

pulse rate of 1 KHz. The PSoC microcontroller is well suited to deal with the array of pulses that must be detected and generated. All 8 of the digital logic blocks and 4 of the analog blocks are used in the design of the MagicBox.

Background:

Modern radio control equipment includes programmable transmitters that allow the pilot to create sophisticated relationships between the pilot's controls and the servo positions. These relationships include things like v-tail mixing, elevons, dual ailerons with flaps and differential, etc. Radio control equipment spans the price range from complete systems for around $100 to over $1000. MagicBox was developed to provide these sophisticated functions to the lower cost radios and to provide unique functions not available in any commercial system.

All radio systems consist of a transmitter the pilot holds to fly the airplane and a flight pack mounted in the model. A picture of a typical radio system is shown in figure 4. The servos are used to move the control surfaces and they receive position information from the receiver. The servos connect to the receiver with a three-wire interface, ground, power, and the position information. The servos typically run from a 4 cell nicad battery pack, nominally 4.8 volts. The position information is a pulse stream with the pulse width determining the servo position. 1.5 mSec is the center position, 1.0 mS to 2.0 mS define the full range of motion of the servo. The position pulse is sent to the servo every 25 mS, or at a 40 Hz rate. A timing diagram for the servo signal is shown below:



The programmable control functions built into the high-end radio transmitters can be designed into a system that connects between the receiver and the servos. This is the design philosophy of MagicBox. The modes of operation for MagicBox are described in detail later in this document. These types of solutions have been developed by several manufacturers and the appendix contains references to several of these devices. One of the unique features of MagicBox is it combines 10 different functions into one device. The pilot can select the function he would like to use on one model, and then reuse MagicBox, in different ways, on other models. Servo testing and positioning functions have also been added to allow the pilot to test servos and servo installations.

One of the most sophisticated functions of the MagicBox is the flight stabilization mode. A two-axis accelerometer is used to measure the attitude of aircraft. The elevator and aileron control signals are sent into the MagicBox and it, then, develops the servo control signals. A third channel allows the pilot to enable and disable the stabilization function.

When enabled, MagicBox will sense the aircraft attitude and adjust the control surfaces to return the aircraft to straight and level flight. This is very helpful for new pilots, when they get in an out of control situation, enable the flight stabilization mode and let go of the controls. To help develop this mode and to help develop the control algorithms a platform stabilization mode was also developed. In this mode two servos are used to stabilize a two-axis platform. This platform can be seen in figure 5 and a short video clip is included on the CD showing the platform reacting to position disturbances.

Hardware:

A block diagram of MagicBox is shown in figure 2 and a detailed schematic is shown in figure 3. The hardware design is very simple, the heart of the design is the Cypress PSoC 26443 in the 28-pin dip package. Pictures of the MagicBox and the internal wiring are attached for reference. The system built is a prototype; in actual production surface mount parts would be used to reduce the size of the device. The prototype was built in a small Radio Shack enclosure and point-to-point wired on vector board. No batteries are included in the MagicBox, it is intended to receive power from the receiver and an optional isolated servo battery pack.

Referring to the schematic in figure 3, the general flow of information on the schematic is from left to right. The servo inputs, from the receiver are buffered, through resistors, and connected to PSoC inputs. Connectors are provided for each servo input. An analog devices ADXL202EB connector provides the interface to the two-axis accelerometer. This is an evaluation board containing the accelerometer, in a final production unit the accelerometer IC would be mounted directly on the MagicBox PCB.

The 4-position jumper block and the 10-position BCD mode switch are interfaced to the PSoC device using analog inputs. This was done to reduce the required IO pins. The jumper block and the mode switch use resistors selected in a near binary ratio and reverenced to ½ VCC. This allows the use of the 6-bit SAR ADC analog block to read and decode the jumper positions and the selected mode.

The servo input and output signals and the accelerometer signals require the accurate measurement of pulse widths. This requires an accurate system clock. A 32KHz external oscillator is used and the PSoC device is configured for a 12MHz internal PLL oscillator. An in-circuit programming connector was added to aid in the software development and version upgrade process.

The variable inputs are implemented using 10K trim pots connected to analog inputs of the PSoC device. The used of these inputs depends on the MagicBox mode. The supply voltage is monitored using a resistive divider and a 6-bit SAR ADC is used with the internal bandgap reference to monitor the supply voltage.

The PSoC device drives two LEDs to indicate status and alarm conditions to the pilot as well as a buzzer used to signal low battery and as an aid in locating a missing aircraft. The buzzer is connected to an 8-bit counter output and the counter’s frequency is set to

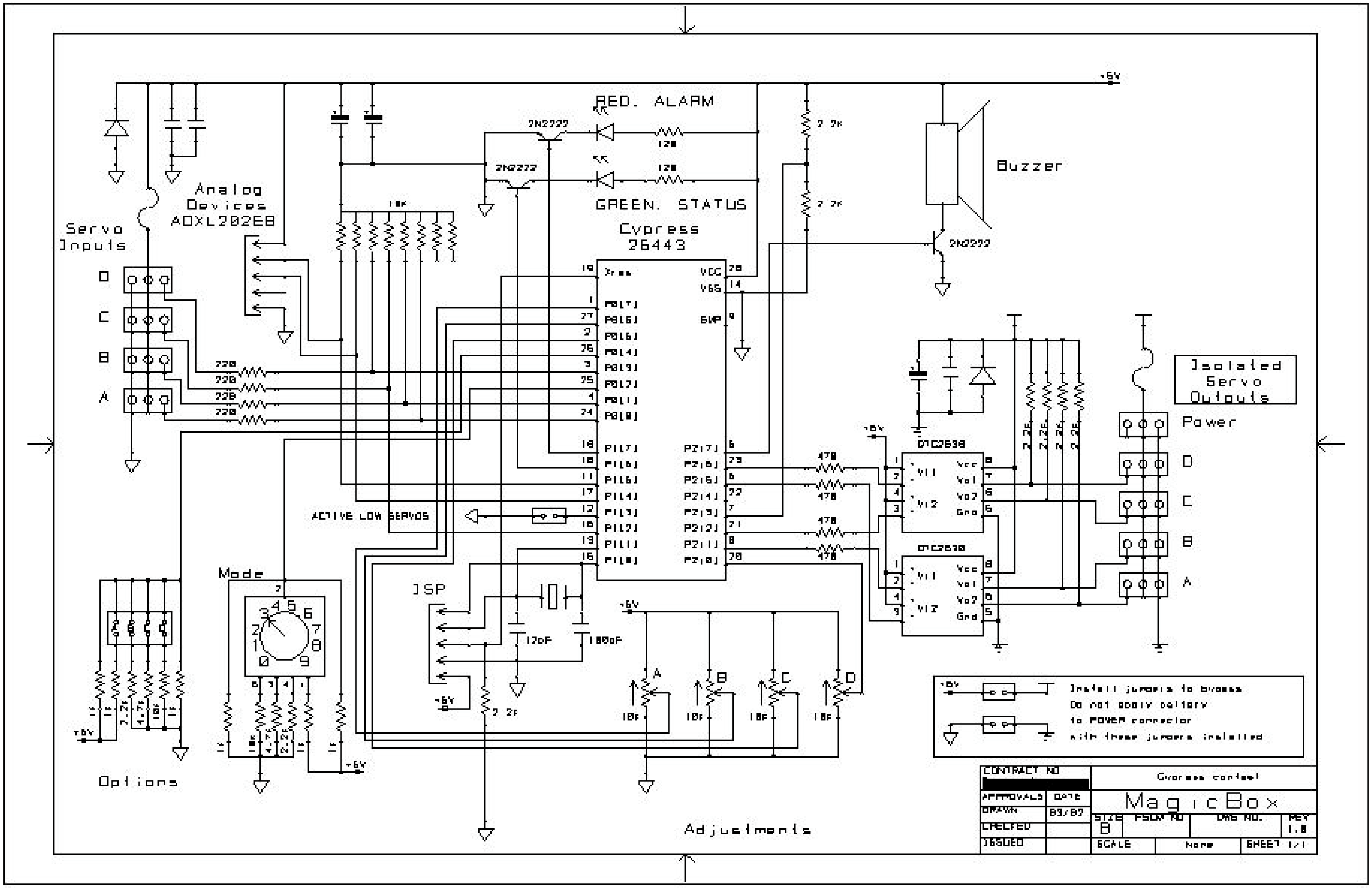
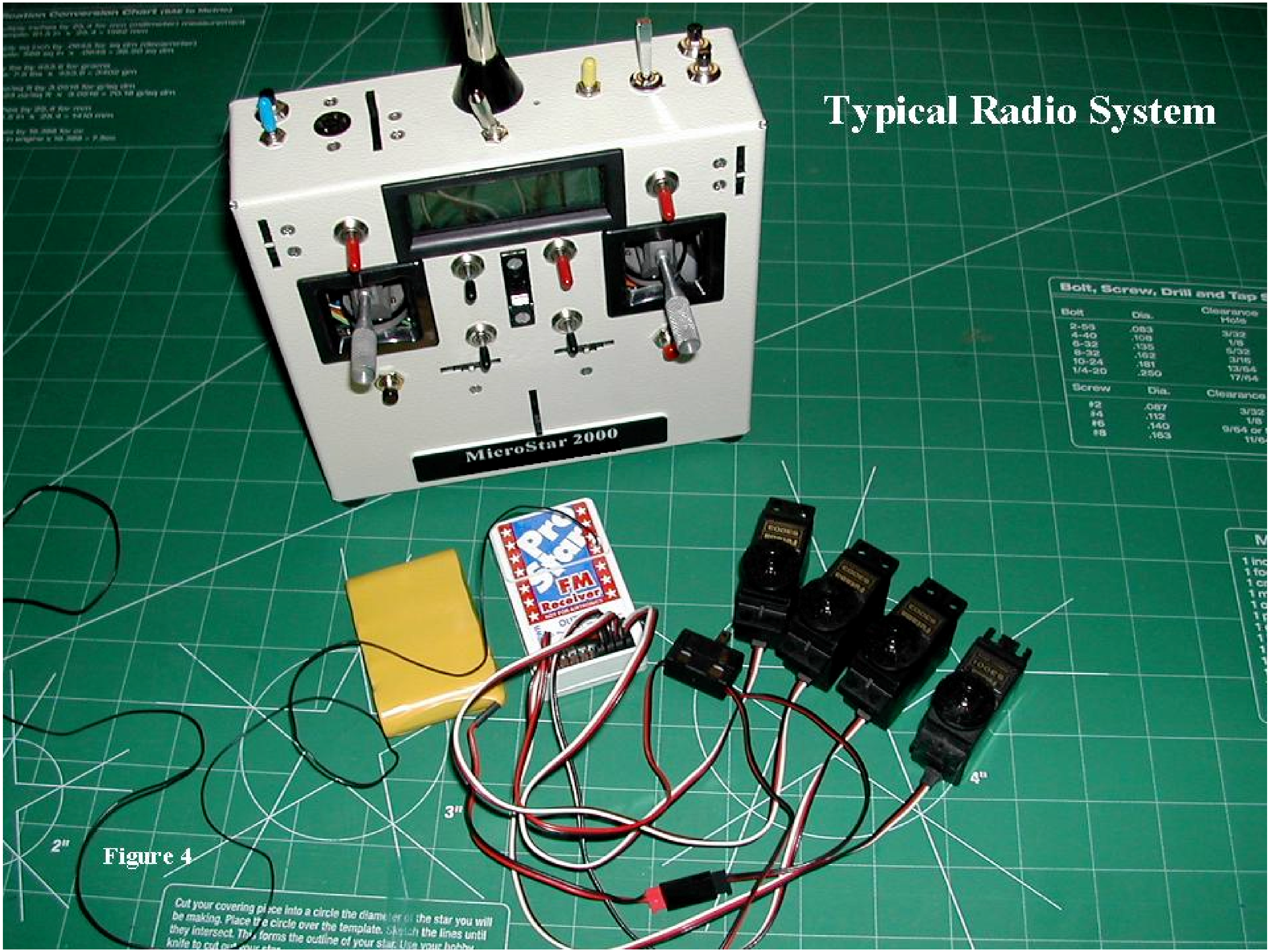

RED. ALARM
GREEN. STATUS
Buzzer
Analog Devices ADXL202EB
Servo Inputs
Cypress 26443
Isolated Servo Outputs
Power
ACTIVE LOW SERVOS
Mode
ISP
Options
Adjustments
Install jumpers to bypass
Do not apply battery
to POWER connector
with these jumpers installed
Cypress confdef
MagicBox

Figure 3



Typical Radio System

Figure 4


Stabilized Platform
Accelerometer is remote mounted under the platform using an extension cable.
MagicBox
Options
Variables
Mode
Futaba
Figure 5

MagicBox PSoC Configuration

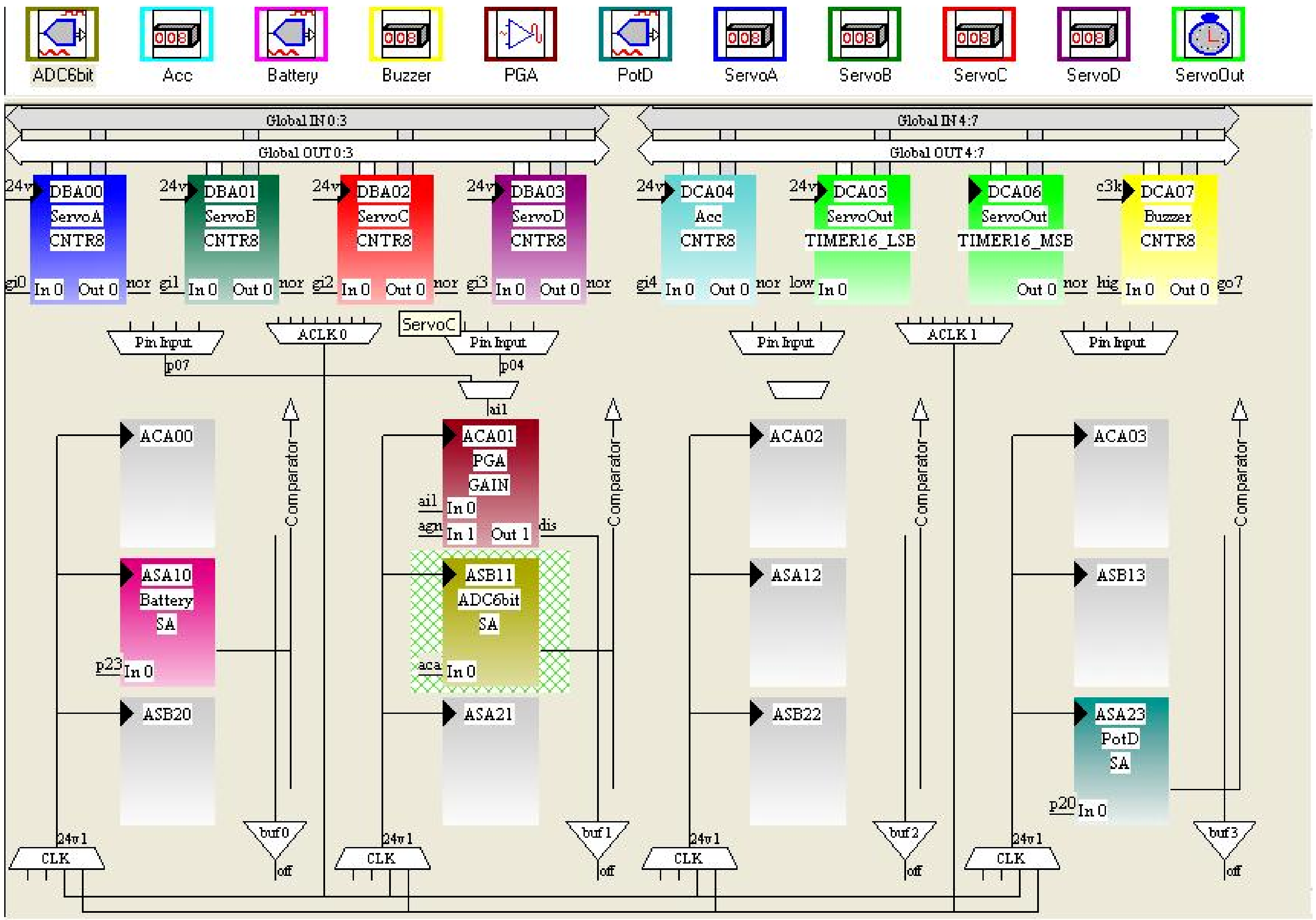


Figure 6

the resonate frequency of the buzzer. Ulta-bright LEDs were used to enable viewing in bright sunlight.

The servo outputs are buffered through opto-isolators. This allows the pilot to power the servos with a different battery pack and isolate the servo load from the receiver. Jumpers are provided to use separate batteries or use the receiver pack to power everything.

Figure 6 shows the internal configuration of the PSoC device. The servo input pulse widths are measured using 8-bit counters. The counters are gated by the servo-input signals. One counter is used for each servo channel, DBA00 through DBA03. The counter clock is 24V2 with a divider of 8. This clock is divided from the output of 24V1 witch is divided by 12 from the 24MHz source. This gives 24V1 a 2MHz output and 24V2 a 250KHz output. This enables a servo input position resolution of 4 uS.

The two-axis accelerometer produces a pulse-width modulated output signal on each axis. This device was configured to generate a 1 mSec output frequency. One 8-bit counter was used the read both axis. This counter, DCA04, is gated by the accelerometer input signal and the PSoC device is reconfigured, on the fly, to route the axis being measured to the counter gate input. This counter also uses the 250KHz clock, 24V2.

All four servo outputs are generated using the 16-bit timer in DAC05 and DAC06. This timer runs from the 2MHz 24V1 clock, enabling .5 uSec servo output pulse timing resolution. The timer is configured to generate at interrupt at each servo output pulse width time and software then develops the servo output pulses on the proper output pin.

The last digital output block, DAC07, is configured as a counter and used to generate the drive frequency for the buzzer. This frequency was set to the resonate frequency of the buzzer to generate the loudest sound possible.

Of the 12 analog blocks, 4 were used to monitor a total of 7 analog input signals. ASA10 is configured as a 6-bit SAR ADC and used to monitor the battery voltage, the reference is set to the internal bandgap for this measurement. ASA23 is used to measure the position of Pot D, this is also configured as a 6-bit SAR ADC input channel. ACA01 and ASC11 are used to measure the 5 remaining inputs. These two blocks create a 6-bit SAR ADC with a 8 channel mux on the input. The PSoC reconfiguration capability was used to route the proper signal to the ADC. All 5 of these inputs and PotD are measured using VCC as the reference. This produces an ADC output that is a ratio of the supply voltage.

MagicBox draws 70 milli-amps at 5.0 volts. The opto-isolators use most of this current. If the isolators are removed, only 17 milliamps are used by the remaining circuitry.

This completes the discussion of the hardware design. The table below summarizes the PSoC resources used:

Digital Blocks:	8, 100%
Analog Blocks:	4, 33%

Reconfiguration: Used for changing the ADC reference, steering the accelerometer gate signals and the ADC mux inputs.

Software:

All project files, including the source code, are included in the design entry package. Each source file header describes the functions available and how they are used. This discussion will outline the general software design and detail the critical functions used in the MagicBox.

In the main loop of the program, the IO ports are initialized, variables are initialized and the Mode switch and Jumper block are read, decoded and saved for later reference. Its important to note that the Mode switch and Jumper block are only read on power up, so if you change a position you will need to restart MagicBox to register the change. The Status and Alarm LEDs are flashed a few times to indicate the power up sequence is finished, then the battery voltage is measured and the buzzer is used to indicate the battery status.

After power up processing is finished the main loop will jump to the proper mode's main loop processing routine. A separate source code file exists for each function in the MagicBox. Additionally a 40 Hz interrupt is generated by the routine that develops the servo output position pulses. A mode specific service routine is called, from the main source file. This mode specific service routine is found in each of the mode's source files. Reading of the input servo positions and generation of the servo output positions are functions common to most modes and control the design methodology of the MagicBox. This software design is important because we are generating and measuring pulse widths that require resolution of a few microseconds. Concurrent interrupts could cause non-deterministic behavior and position jitter.

Servo input positions are measured using an 8-bit counter gated by the servo input signal. An 8-bit counter is used for each servo position and the data is monitored using a combination of polling and interrupt service routines. The file servo.asm contains the source code for the routines being discussed. Servo input signals are sent serial, that is, no two servo pulses overlap, this fact is used in the design of the servo position monitoring code. The function AcquireServoInputs is called by every mode that needs to read the servo positions, this routine performs the following functions:

1.) Stops all servo counters and sets the period register to 255.
2.) Starts all servo counters after first initializing the servo position's most significant bytes. The interrupt service routines for each counter are also initialized. This interrupt fires when the counter reaches zero, the service routine will then increment the most significant byte of the position word for the respective servo. For example SAP is the word saving the servo position for input servo A. The byte at SAP+1 will be incremented by the interrupt service routine for the counter.

3.) The next function called is the servo wait routine. This routine waits for all servo data to be read by all counters. This is done using the following procedure:
    a.) Wait for any input servo signal to go high, logic level 1.
    b.) Wait for all servo input signals to go low, logic level 0.
    At this point the counters contain the complement of the least significant byte of the position information and the most significant byte of the position information is saved in the servo position word's most significant byte.
4.) The last function that is called reads the counter data from each counter and calculates the servo position value. This value is determined by taking the complement of the counter value and placing it in the servo position word's least significant byte, SAP for example. This gives the pulse width in 4 microsecond units, since the servo output functions have a resolution of 0.5 microseconds, this function also multiplies the servo position by 8.

The result of calling AcquireServoInputs is to update the servo position words, SAP,SBP,SCP, and SDP, with valid 16 bit numbers indicating the input servos position, in 0.5 microsecond units.

The next vital function is the interrupt service routine used to develop the servo output signals. This service routine is also responsible for the generation of the 40 Hz real time interrupt which is the "heart beat" of the MagicBox. The output servo signals are developed using a 16-bit timer. This timer is configured to generate an interrupt when the counter reaches zero. An array called ChanTimes, declared in the main routine, contains the count value for each output servo plus the value needed to create a fixed output rate of 40 Hz. The interrupt service routine uses pointers to keep track of the signal its outputting and the next signal to send. After the last servo signal is output, a real-time 40 Hz processing routine is called. This routine contains code to flash the LEDs, buzz the buzzer, and decrement timeout timer registers used in various routines. This code can be found in the ServoOutINT.asm file. The real time service routine also calls the main loop which in turn calls the proper mode's real time routine. The servo output functions occur in the background and each mode needs only to update the ChanTimes array to make the servos move to the desired positions. Flags are used to synchronize the main loop processing with the servo output processing.

This section discusses the code found in the file Isolate.asm. This is the source file for the isolate mode. In the isolate mode the servo positions are read, optionally reversed and sent to the servo outputs. Since the MagicBox allows you to opto-isolate the servo outputs this function serves as a 4 channel isolation mode. This file contains two functions; ModeIsolate, and ISRisolate. This is the convention used in all modes. Each of these functions are described below:

ModeIsolate:

This is the isolate mode's main processing loop. A flag is set indicating the servo position information is valid, then the acquire servo input data function is called. When this routine returns, with servo positions, the data is processed after setting a flag indicating the old servo positions are no longer valid. For each input channel the value is read and validated to ensure its in a valid range and then it is placed in a servo output variable, ServoA for example. The output can then be reversed, if the reverse jumper is detected. The loop is repeated after all channels are processed. You will notice that the servo input position variables, SAP for example, do not map to the same output servo position letter, ServoD for example. This is due to hardware connector positioning and the order desired for the physical device.

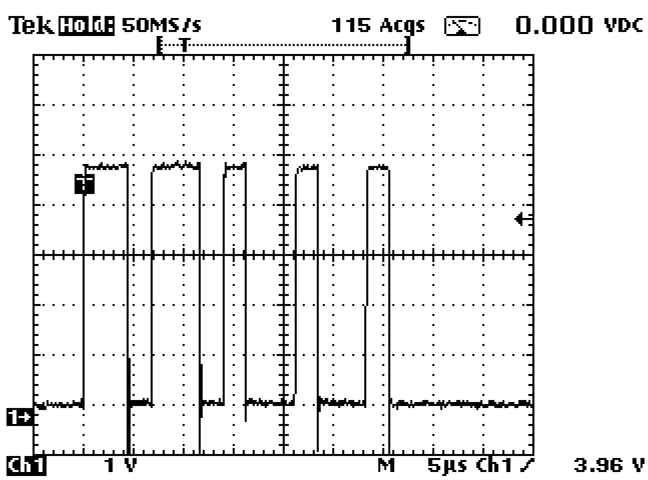
ISRisolate:

This is the mode-specific interrupt service routine. This routine is very simple; it checks the data flag and if the information is valid, the servo position information is used to update the ChanTimes array. The values are validated and the final value is calculated. This final value is used to ensure we generate a 40 Hz servo signal output rate.

This describes how the software was designed to create the MagicBox. The source code contains more information and comments on each of the routines. Please refer to this source code for more detailed information.

Development Environment:

All software was developed using the design kit supplied by Cypress as a result of entering the contest. All of the code was developed in assembly language. The code was debugged by writing test software and routines, programming the part and observing the behavior of the system. This worked for all but a few very stubborn bugs. Because I did not have a in-circuit emulator, I created a simple routine called SendByte. This routine would serialize a byte and send it out the Servo B output channel. I used a scope to look at this signal. A narrow output pulse is a 0 bit and a wide output pulse is a 1 bit. An example scope output can be seen below:



This trace is for an 8-bit value of A5 hex. Reading from left to right, first pulse wide thus it’s a 1 the next bit is narrow and represents a 0 etc. The last narrow pulse is the stop bit and is not decoded.

Mode Descriptions:

The following pages describe each mode in detail. The servo inputs and outputs are defined and the jumpers and variable usage are also described. There are a few things that are always performed by the MagicBox regardless of its mode and some important things you need to remember:

1.) The mode switch and option jumpers are only read on system startup. If you change the mode or move a jumper it will not register until you cycle power.
2.) MagicBox does not contain a battery, it requires power. Usually power is received from the connections to your receiver. The servo output can use the same power or use a second battery. Two jumpers are provided to enable this isolation (please see figure 7). If the isolation jumpers are install then the receiver power is used for the servos. If the isolation jumpers are removed then a second battery will be required to power the servos. Opto-isolators will electrically isolate servos if the isolation jumpers are removed.
3.) Servo input and output signals are always validated and no signals narrower than 0.9 milliseconds or wider than 2.1 milliseconds will ever be sent to a servo. This provides a level of glitch protection in every mode.
4.) On power up the Status and Alarm LEDs will flash to indicate the system is initialized.
5.) The receiver side battery is tested and the buzzer will sound to let you know the battery status. Here is what you will hear:
    a.) Three short beeps if the battery is over 5 volts
    b.) No beeps if the voltage is under 5 volts but is over 4.5 volts
    c.) One long beep if the voltage is under 4.5 volts. If you hear this you should not fly. Charge you batteries instead.

After these tests and functions are complete the MagicBox will enter the mode you have selected as discussed in the following pages.

Servo Position Control

The servo position control mode allows you to use the variable controls to set the positions of all four servos. The servos will hold the static position you set them to and will not perform any automatic motions. This mode is intended to allow you to test servos and aid in servo installation. Battery power will need to be applied to the power connector on the MagicBox and the isolation jumpers will need to be installed or a second battery will need to be connected to one of the servo input channels.

The Servo inputs and output, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 0

Servo Inputs		Servo Outputs	
A	Not used	A	Drives servo A
B	Not used	B	Drives servo B
C	Not used	C	Drives servo C
D	Not used	D	Drives servo D

Jumpers		Variables	
A	Not used	A	Sets position for servo A
B	Not used	B	Sets position for servo B
C	Not used	C	Sets position for servo C
D	No not install	D	Sets position for servo D

LEDs: Not used

Buzzer: Not used

Stabilized Platform

The stabilized platform mode uses two servos and the two-axis accelerometer to create a stabilized platform. A picture of this system is shown in figure 5. The accelerometer is mounted under the platform, two servos are used to create a two axis-gimbal. A short video clip of the stabilized platform in operation is shown on the CD provided with this design package. The two-axis accelerometer is placed so both its axis are orthogonal to the earth's gravity vector. If the platform tilts the angle is detected and the servo position is adjusted to return the platform to its zero position. While this mode is not useful as a model airplane flight function, it was very useful in the development of the control algorithms needed for the flight stabilization mode.

The platform is stabilized using proportional and integral gain in a classic feedback system. The gains and setpoints can be controlled using the variables provided. The servo directions can also be reversed in the event the gains polarities are reversed.

Battery power will need to be applied to the power connector on the MagicBox and the isolation jumpers will need to be installed or a second battery will need to be connected to one of the servo input channels.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 0

Servo Inputs		Servo Outputs	
A	Not used	A	Drives X axis servo
B	Not used	B	Drives Y axis servo
C	Not used	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Install to reverse X axis servo	A	Sets proportional gain
B	Install to reverse Y axis servo	B	Sets integral gain
C	Not used	C	Sets zero position for X axis
D	Must be installed	D	Sets zero position for Y axis

LEDs: Not used

Buzzer: Not used

Servo Cycle

The servo cycle mode is designed to set and exercise a servo. Only the servo connected to servo output A will be controlled. The servo will be moved between three positions; A to B to C to B and then back to A. This cycle will be repeated as long as power is applied. A delay of 1 second will occur at each position. The variable control pots allow you to define the positions A, B and C.

Battery power will need to be applied to the power connector on the MagicBox and the isolation jumpers will need to be installed or a second battery will need to be connected to one of the servo input channels.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 1

Servo Inputs		Servo Outputs	
A	Not used	A	Drive the servo being cycled
B	Not used	B	Not used
C	Not used	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Not used	A	Sets servo position A
B	Not used	B	Sets servo position B
C	Not used	C	Sets servo position C
D	Not used	D	Not used

LEDs: Not used

Buzzer: Not used

Isolate

The Isolate mode allows the pilot to isolate the receiver power from the servo power and isolates the servo signals. The jumpers allow the pilot to reverse any combination of servo channels. Installing the jumper reverses that channel's servo output. You do not need to drive all four channels, any combination of servo inputs can be used. Servo output signals will only be developed for the channels that are driven with valid input signals.

MagicBox receives its power from the receiver. The connectors that bring the servo input signals into the MagicBox also contain the power it needs. The servo outputs can be driven with a separate battery connected to the power connector. If this optional second battery is used the isolation jumpers must be removed. These jumpers are internal to the MagicBox. If you wish to use the receiver battery to power your servos install the isolation jumpers and do not connect a battery to the power connector.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 2

Servo Inputs		Servo Outputs	
A	Servo A input from receiver	A	To Servo A
B	Servo B input from receiver	B	To Servo B
C	Servo C input from receiver	C	To Servo C
D	Servo D input from receiver	D	To Servo D

Jumpers		Variables	
A	Install to reverse Servo A	A	Not used
B	Install to reverse Servo B	B	Not used
C	Install to reverse Servo C	C	Not used
D	Install to reverse Servo D	D	Not used

LEDs: Not used

Buzzer: Not used

Match

The Match mode is designed to take one servo input and drives two servos. The two output servos can each be reversed relative to the input servo. One of the output servos can be adjusted to match the other output servo. This is done with an offset adjustment and a gain adjustment. The offset is used to define a different center position and the gain adjustment allows you to match the range of motion of the two servos. This mode is useful when you are using two servos to drive your elevators or ailerons and you need to adjust for mismatches in the control linkages. The Match mode provides two of the output matching functions, allowing two input channels, each able to drive two output servos.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 3

Servo Inputs		Servo Outputs	
A	Servo 1 input from receiver	A	To Servo A from input 1
B	Servo 2 input from receiver	B	To Servo B from input 1, adjusted
C	Not used	C	To Servo C from input 2
D	Not used	D	To Servo D from input 2, adjusted

Jumpers		Variables	
A	Install to reverse Servo A	A	Servo B gain
B	Install to reverse Servo B	B	Servo B offset
C	Install to reverse Servo C	C	Servo D gain
D	Install to reverse Servo D	D	Servo D offset

LEDs: Not used

Buzzer: Not used

Stable

In this mode the aircraft's flight path is stabilized in the roll and pitch axis by allowing the MagicBox to control the Aileron and Elevator servo positions. A two-axis accelerometer is used to measure the aircraft's attitude and send correction signals to the servos to keep the flight path level. A third servo input allows this function to be enabled or disabled by the pilot. Each axis has a correction gain adjustment and offset adjustment. The gain adjustment should be set low and the gain gradually increased until you achieve the flight performance that best fits your preference and the model you are flying. The offset adjustment will allow you to define the attitude in level controlled flight. Jumpers are also provided to reverse the servo channel and to reverse the correction direction.

Orientation of the MagicBox in the aircraft is important. Make sure it is mounted parallel with the axis of the fuselage and held in position. Use a soft foam rubber mounting system to hold it firmly in place and provide vibration isolation.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 4

Servo Inputs		Servo Outputs	
A	Elevator servo input from receiver	A	To Elevator servo
B	Aileron servo input from receiver	B	To Aileron servo
C	Enable input	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Install to reverse Elevator input	A	Elevator correction gain
B	Install to reverse Elevator correction	B	Elevator correction offset
C	Install to reverse Aileron input	C	Aileron correction gain
D	Install to reverse Aileron correction	D	Aileron correction offset

LEDs:

Status – Flashes when the elevator (pitch) axis is close to level.
Alarm – Flashes when the aileron (roll) axis is close to level.

Buzzer: Not used

V-Tail

Some aircraft are designed with a V-shaped tail instead of the classically horizontal and vertical stabilizers. This requires coupling of the elevator and rudder signals to create the proper stabilizer effects. Elevator requires both controls to move in the same direction while rudder requires the controls to move in opposite directions. This mode takes the elevator and rudder signals, as inputs, and generates the proper servo control signals to enable V-Tail operation. Jumpers are provided to allow the reversal of one or both of the servo drive signals.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 5

Servo Inputs		Servo Outputs	
A	Elevator servo input from receiver	A	To tail Servo A
B	Rudder servo input from receiver	B	To tail Servo B
C	Not used	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Install to reverse Servo A	A	Not used
B	Install to reverse Servo B	B	Not used
C	Not used	C	Not used
D	Not used	D	Not used

LEDs: Not used

Buzzer: Not used

Flaperon

Many model aircraft use two servos to control the ailerons. This allows flaps and ailerons to be combined by dropping both ailerons to simulate lowering of the flaps. Also, a function called aileron differential is often employed. This is where the ailerons move less in the down direction than in the up direction. This mode supports all of these features. An aileron input signal and an optional flap input signal are connector to the MagicBox. The two aileron servo signals are generated from these inputs. Jumpers allow you to: 1.) reverse the servo outputs, 2.) enable the flap function and 3.) reverse the flap input signal direction. Gain adjustments allow you to define the amount of flaps and the level of aileron differential you wish to employ.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 6

Servo Inputs		Servo Outputs	
A	Aileron servo input from receiver	A	To right aileron Servo A
B	Flap servo input from receiver	B	To left aileron Servo B
C	Not used	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Install to reverse Servo A	A	Down aileron gain, differential
B	Install to reverse Servo B	B	Flap gain
C	Install to enable flaps	C	Not used
D	Install to reverse flap input signal	D	Not used

LEDs: Not used

Buzzer: Not used

Dualrates

One of the most popular model airplane functions is dualrate controls. This allows the pilot to change the sensitivity of the flight controls. This mode allows you to control the sensitivity of three channels using a fourth channel as a rate selection input. The input channels from the receiver are connected to the MagicBox and the servo drive signals are generated. If the enable channel signals a low rate request, MagicBox will reduce the sensitivity of the controls by an amount defined by the variable adjustments. Jumpers are provided to allow you to reverse any of the servo output signals and to reserve the enable input signal.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 7

Servo Inputs		Servo Outputs	
A	Servo A	A	Servo A
B	Servo B	B	Servo B
C	Servo C	C	Servo C
D	Low rate enable	D	Not used

Jumpers		Variables	
A	Install to reverse Servo A	A	Servo A low rate
B	Install to reverse Servo B	B	Servo B low rate
C	Install to reverse Servo C	C	Servo C low rate
D	Install to reverse enable input	D	Not used

LEDs: Not used

Buzzer: Not used

Glitch

The Glitch mode inspects the input signal and "looks" for a glitch. A glitch can be an input signal that is out of its valid pulse width range of 1.0 mSec to 2.0 mSec, or it could be a rapid and unexpected change in servo position. Several algorithms were tested during the development of this mode. The system implemented in the MagicBox consists of a buffer that holds the last 4 servo positions. This position information is investigated and a servo signal is generated only if the last 4 values have not changed or have all moved in the same direction. This greatly reduces the servo jitter seen in weak signal conditions and the erratic motions seen with interference and multi-path signal loss. This glitch removal algorithm is applied to all 4 servo inputs. The jumpers allow the pilot to reverse any combination of servo channels.

A fail-safe function is also supported in this mode. If a valid servo output is not generated for 6 seconds, all of the servos will be positioned to a location defined by the variable adjustments. When a fail-safe condition is detected, the buzzer will beep at a constant rate until the signal is again detected. The buzzer can help in locating a "downed" aircraft.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 8

Servo Inputs		Servo Outputs	
A	Servo A	A	Servo A
B	Servo B	B	Servo B
C	Servo C	C	Servo C
D	Servo D	D	Servo D

Jumpers		Variables	
A	Install to reverse Servo A	A	Fail safe position for Servo A
B	Install to reverse Servo B	B	Fail safe position for Servo B
C	Install to reverse Servo C	C	Fail safe position for Servo C
D	Install to reverse Servo D	D	Fail safe position for Servo D

LEDs:

Alarm – Flashes if an input signal from the receiver is out of its valid range.
Status – Flashes if a glitch is detected in an input signal data stream.

Buzzer:

Beeps at a constant rate if the input signal is lost.

Mixer

The mixer function allows the pilot to mix any two channels to enable functions like rudder and aileron mixing for coordinated turns etc. This mixing function can be enabled and disabled with a third input channel. The jumpers allow the pilot to reverse the servo output and to reverse the enable input. Servo B input is mixed with Servo A input and the mixed result is output on channel A. Servo B input signal is passed through to output B with no change. The mixed output is developed as defined below:

```
Mix = Servo B – Zero point
If Mix >=0 then
        Servo A = Servo A + Mix * Pos gain
Else
        Servo A = Servo A – Mix * Neg gain
End if
```

This algorithm allows different gains for the mix points above and below the zero position.

The Servo inputs and outputs, the jumpers and variable input usage are defined in the tables below:

Mode switch position: 9

Servo Inputs		Servo Outputs	
A	Servo A	A	Servo A + Servo B
B	Servo B	B	Servo B
C	Enable the mixer	C	Not used
D	Not used	D	Not used

Jumpers		Variables	
A	Install to reverse Servo A	A	Defined the zero point
B	Install to reverse Servo B	B	Positive mixer gain
C	Install to reverse enable input	C	Negative mixer gain
D	Not used	D	Not used

LEDs: Not used

Buzzer: Not used

Conclusions:

The PSoC microcontroller is a very unique design allowing the designer to configure the device to suit his/her specific needs. In this application I needed to deal with many pulse width signals and the ability to define a number of counters and timers were critical to the MagicBox. The small instruction set is very powerful and had a short learning curve. The fact that this device was developed, in assembly language, without the aid of an emulator is a strong statement for its ease of use.

A commercial version of the MagicBox would likely contain LEDs that could be remotely mounted, enabling the pilot to mount them externally. This will allow visualizing the status in flight. Surface mount technology will allow MagicBox to be reduced to a very small size. The device could be sold in several configurations:

Optional accelerometers
Optional opto-isolators

Several additional modes of operations are possible and could easily be implemented in this design, for example:

Servo speed limiting
Sequential servo control
Exponential and other non-linear rate adjustments

With small modifications to the hardware design MagicBox could provide an interface between computers and servos using an RS-232 interface. This is very useful is robotics applications.

I think the MagicBox hits a market niche not covered by any of the available systems. MagicBox could be manufactured at a very competitive price and I look forward to this opportunity.

Appendix

a.) List of similar devices currently available:

- Miracle Y servo reverser, MAXX products (847) 438-2233	$19.50
- Dual reverser, Precision Micro electronics (361) 814-6843	$27.90
- Elevon Mixer, Precision Micro electronics (361) 814-6843	$30.95
- Fail-safe, Precision Micro electronics (361) 814-6843	$24.95
- Flap Mixer, Precision Micro electronics (361) 814-6843	$30.95
- Servo Cycler, Precision Micro electronics (361) 814-6843	$24.95
- Dual servo reverser, ElectroDynamics (734) 422-5420	$29.95
- FMA co-pilot flight stabalization system (301) 831-8985	$149.00
- Gem 2000 electronic guardian, central hobbies (406) 259-9004	$26.95
- MatchBox, JR (877) 504-0233	$99.00
- Air Alert flight monitor, Hobbico (800) 637-6050	$15.99
- Futaba level system, (800) 637-6050	$49.99

MagicBox can do all of the things performed by the devices listed above. This is a short list of devices available and does not represent a comprehensive search of manufactures. I believe the MagicBox can be manufactured at a price competitive with these devices.


Alarm
Status
Options
Variables
A
B
C
D
A B C D
MagicBox
Mode
A B C D
A B C D Pwr
Signal
Pwr
Gnd
Servo inputs
Servo outputs

Top of MagicBox Prototype Circuit Board

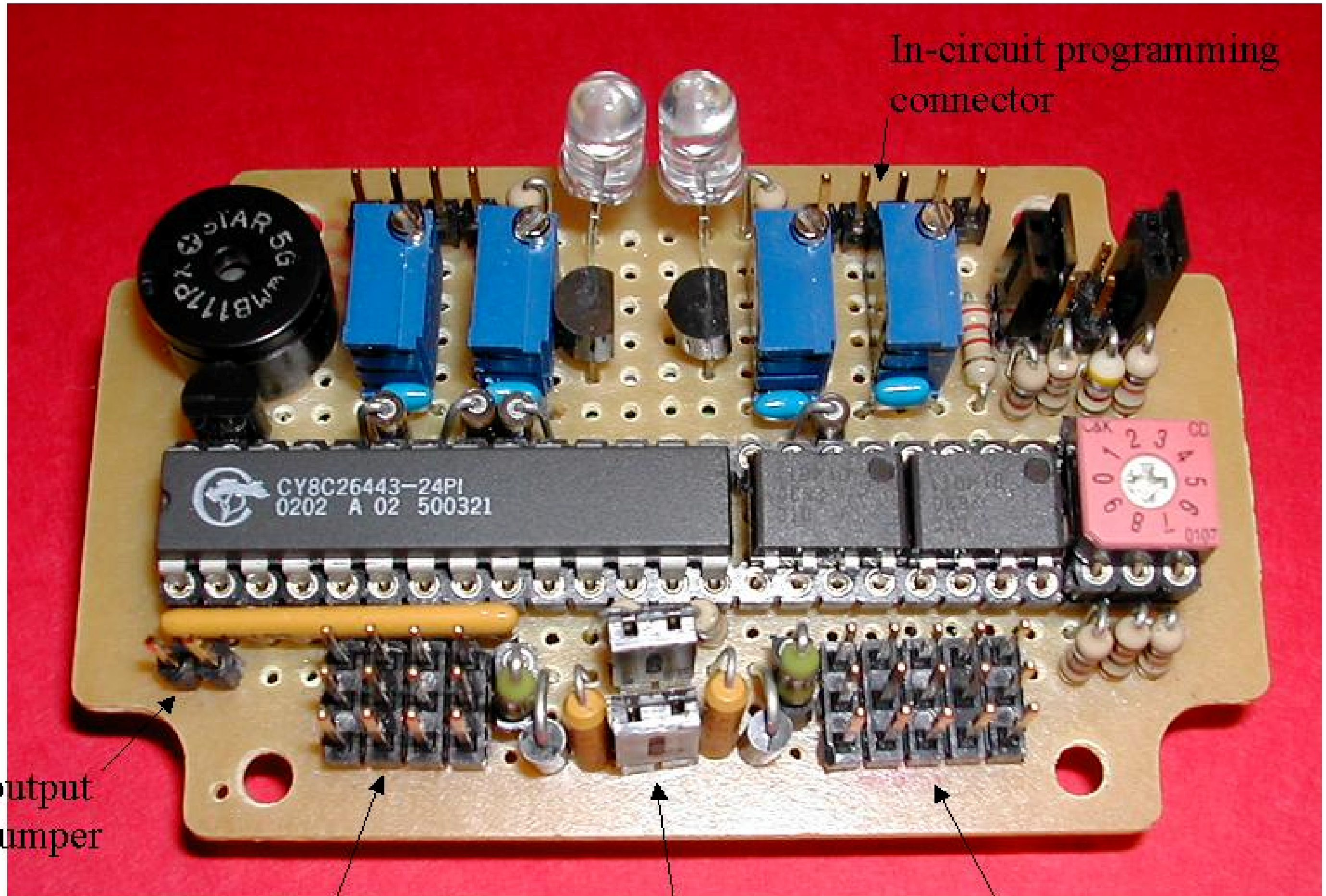
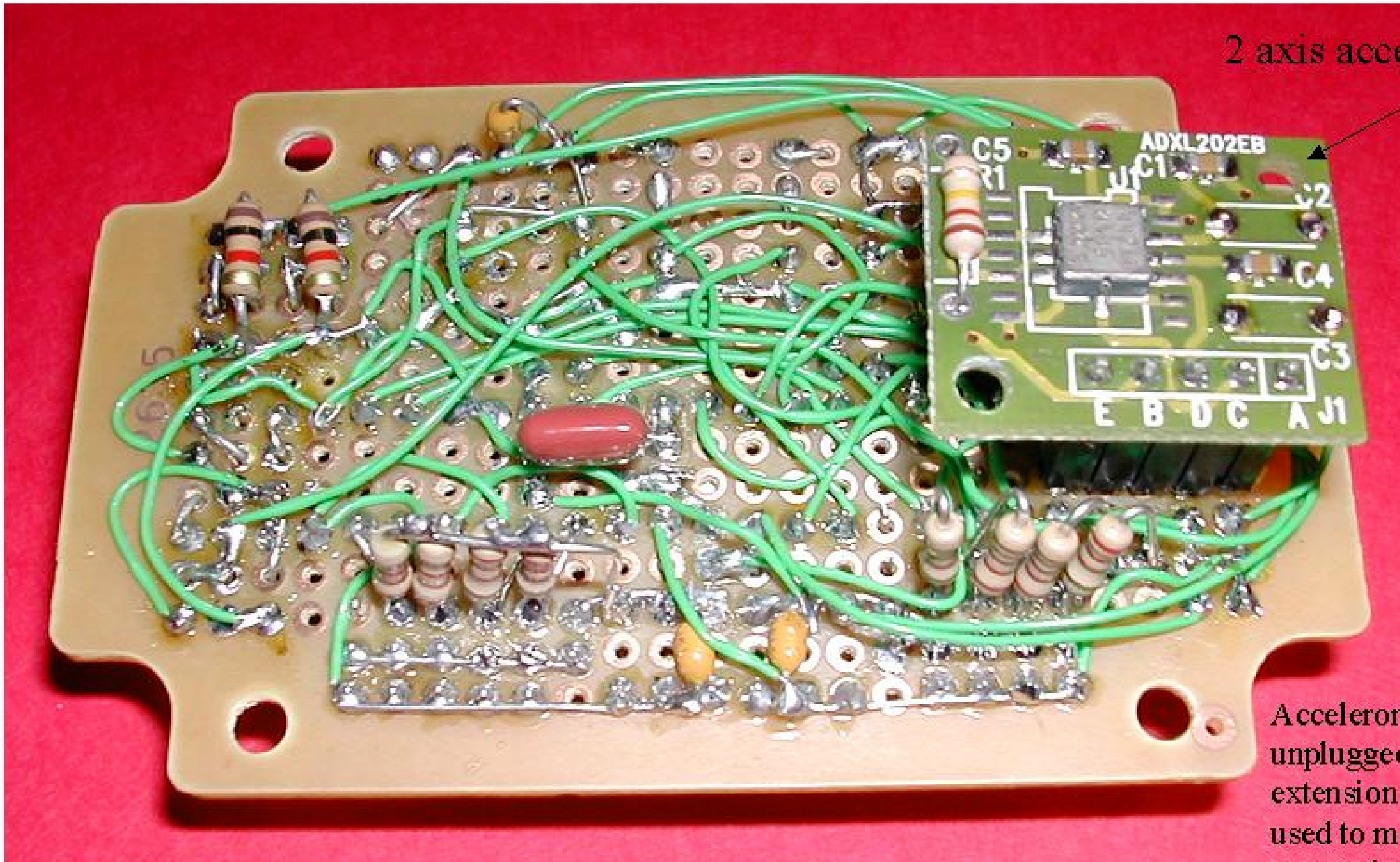


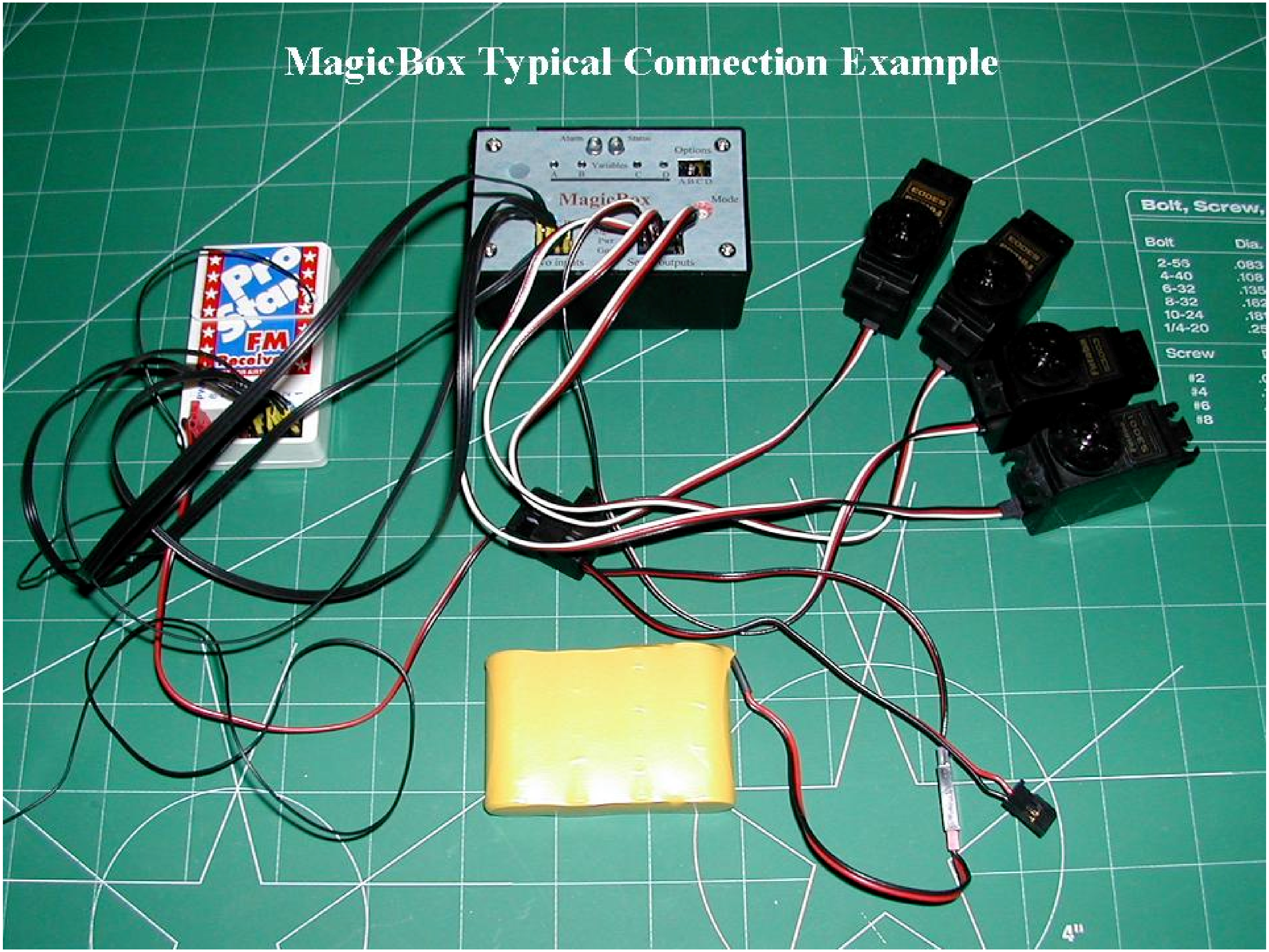
Figure 7

Bottom of MagicBox Prototype Circuit Board



2 axis accelerometer

Accelerometer can be unplugged and an extension cable can be used to mount it at a remote location. This is done in the stabilized platform mode.



MagicBox Typical Connection Example