Microcontrollers for Ham Radio
Part 1 - BASIC Stamp

By Terry G. Glagowski / W1TR - 2012-03-22 23:17

• What is a Microcontroller?
• Some HAM Applications
• Popular Microcontrollers for Ham Radio
• The BASIC Stamp – Parallax
• Installing and Using the BASIC Stamp
• The Original BASIC Language
• The BASIC Stamp Integrated Development Environment
• Introduction to PBASIC Programming
• Some Example Programs
• Summary
What IS a Microcontroller?

- Microcontroller Chip
  - MicroCode
  - Controller
    - Counters
    - Timers
    - Registers
  - Interface Logic
  - Power Interface

- Additional Memory
  - Digital Interface
  - Communication Interface
  - Analog Interface
  - Power Interface

- Power Supply

- Microcontroller Board

- MicroCode

- Program ROM

- Data RAM

- Controller

- Counters
  - Timers
  - Registers

- Interface Logic

- Power Interface

- Communication Interface

- Digital Interface

- Analog Interface

- Additional Memory

- External Memory

- Communications (SIO, USB, Ethernet)

- Sensors

- Switches
  - Buttons
  - Logic Gates

- Relays
  - Lights
  - Motors
  - Logic Gates

- Amplifiers
Popular Microcontrollers for Ham Radio

- Arduino
- PIC
- Parallax Propeller
- BASIC Stamp (some versions use PIC chip)
- Older and **Commercial** Units
  - Z800 / Z80 / Z8 (Zilog) – Z8 had a BASIC interpreter
  - 8080 / 8085 / 8086 / 8088 / **Pentium family** (Intel/AMD)
  - 1802 (RCA)
  - 6800 / 68000 (Motorola)
  - 6502 (Rockwell)
  - TMS 9900 / **TMS320Cxx DSP** (Texas Instruments)
  - **TLSC870** (Toshiba)
Some HAM Applications

- Automatic Antenna Tuner
- Automatic Screwdriver Antenna Controller
- Automatic Amplifier Controller / Tuner
- Multi Radio to Multi Antenna Switch Controller
- Transceiver Controller (in every radio)
- DDS Synthesizer (PIC-EL)
- CW Keyer – ARRL PIC Kit
- Rotor Controller
- ??? Use Your Imagination ...
Update the Heathkit SA-2500 ATU

• Purchased the SA-2550 ATU in 1993
• HeathKit going out of business, deep discount
• Claim: automatic antenna tuner
• Reality:
  – Turn bandswitch to Set Coil Electrically, Comparator Circuit
  – Input and Output Capacitors Turn Randomly
  – Tuning Stops When SWR is Coincidentally Low Enough?
  – HUH? Not Really All That Automatic, It’s STUPID !!!
• Since 1993, Thought About a Microcontroller
• Procrastination Set In, then Life Circumstances!
• Article: “Tuner Transformation”, Peter Ferrand, 73 Magazine, April, 1987
  http://www.nostalgickitscentral.com/heath/73-index/articles/SA-2500-4-87.pdf
Heathkit SA-2500
SA-2500 Front Panel
Antenna Tuner “TEE” Circuit
### Sensor / Effector Signals

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Signal</th>
<th>Type</th>
<th>Direction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor 1</td>
<td>Input Capacitor Position</td>
<td>Digital</td>
<td>Input</td>
<td>Indicates Input Capacitor is at Home Position</td>
</tr>
<tr>
<td>Sensor 2</td>
<td>Coil Position</td>
<td>Analog</td>
<td>Input</td>
<td>Indicates Coil Position - Voltage from Potentiometer</td>
</tr>
<tr>
<td>Sensor 3</td>
<td>Output Capacitor Position</td>
<td>Digital</td>
<td>Input</td>
<td>Indicates Output Capacitor is at Home Position</td>
</tr>
<tr>
<td>Sensor 4</td>
<td>Forward Power</td>
<td>Analog</td>
<td>Input</td>
<td>Indicates Forward Power Level - Power Output</td>
</tr>
<tr>
<td>Sensor 4</td>
<td>Reflected Power</td>
<td>Analog</td>
<td>Input</td>
<td>Indicates Reflected Power Level</td>
</tr>
<tr>
<td>Sensor 5</td>
<td>SWR Limit</td>
<td>Analog</td>
<td>Input</td>
<td>Indicates Acceptable SWR Limit - Voltage from Potentiometer</td>
</tr>
<tr>
<td>Switch 1</td>
<td>Input Capacitor UP</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch UP Position - Increase Input Capacitance</td>
</tr>
<tr>
<td>Switch 1</td>
<td>Input Capacitor DOWN</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch DOWN Position - Decrease Input Capacitance</td>
</tr>
<tr>
<td>Switch 2</td>
<td>Coil UP</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch UP Position - Increase Inductance</td>
</tr>
<tr>
<td>Switch 2</td>
<td>Coil DOWN</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch DOWN Position - Decrease Inductance</td>
</tr>
<tr>
<td>Switch 3</td>
<td>Output Capacitor UP</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch UP Position - Increase Output Capacitance</td>
</tr>
<tr>
<td>Switch 3</td>
<td>Output Capacitor DOWN</td>
<td>Digital</td>
<td>Input</td>
<td>Center Off Switch DOWN Position - Decrease Output Capacitance</td>
</tr>
<tr>
<td>Switch 4</td>
<td>Band SW - Sig 1 thru Sig 10</td>
<td>Digital</td>
<td>Input</td>
<td>Indicates Band (i) is Selected (may need to do BCD encoding)</td>
</tr>
<tr>
<td>Switch 5</td>
<td>Band High / Low</td>
<td>Digital</td>
<td>Input</td>
<td>Indicates Band HI or LO is Selected</td>
</tr>
<tr>
<td>Button 1</td>
<td>Auto Tune</td>
<td>Digital</td>
<td>Input</td>
<td>Activates / Deactivates Automatic Tune Cycle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Effector</th>
<th>Signal</th>
<th>Type</th>
<th>Direction</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor 1</td>
<td>Forward</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 1 to Move Forward</td>
</tr>
<tr>
<td>Motor 1</td>
<td>Reverse</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 1 to Move Reverse</td>
</tr>
<tr>
<td>Motor 2</td>
<td>Forward</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 2 to Move Forward</td>
</tr>
<tr>
<td>Motor 2</td>
<td>Reverse</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 2 to Move Reverse</td>
</tr>
<tr>
<td>Motor 3</td>
<td>Forward</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 3 to Move Forward</td>
</tr>
<tr>
<td>Motor 3</td>
<td>Reverse</td>
<td>Digital</td>
<td>Output</td>
<td>Controls Motor 3 to Move Reverse</td>
</tr>
</tbody>
</table>
Control Algorithm

• Always Move Input Cap, Coil, Output Cap Re Switches
• Begin Automatic When AutoTune Button Pressed
• Stop Automatic When AutoTune Button Pressed Again
• Read Band and Hi/Lo Switches, Preset Coil
• Iteratively Tune Output Cap, Input Cap, Coil While Reading Forward and Reflected Power
• Adjust Components for Maximum Forward / Reflected Ratio
• Stop at Minimum, Less Than Acceptable Limit, or Abort
• More-Or-Less How you would do it manually!
What Microcontroller to Use?

- Price
- Availability
- Ease of Programming
- Capabilities
  - Processing
  - I/O
  - Memory
Arduino

http://arduino.cc/en/Main/ArduinoBoardMega2560
# Arduino Mega 2560 Capabilities

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming Language</td>
<td>C / Assembly using Integ. Dev. Env.</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>54 (of which 14 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>16</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>256 KB of which 8 KB used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>8 KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>4 KB</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>
Programmable Interface Controller - PIC

www.microchip.com

• Originally Developed by General Instrument, 1975
• Just like McDonalds – over 10 billion sold!
• MANY variations of PIC chips – see data sheets on WEB
PIC Capabilities (8 bit)

- Bits: 8, 16, 32 versions
- Programming Language: MPLAB IDE - C, Assembly
- Operating Voltage: 5 VDC
- Memory: Flash, EEPROM 14KB, RAM 1024
- I/O Pins: 12
- Speed: 32 MHz
- Timers: (2) 8-bit, (1) 16 bit, Watchdog
- A/D: 8 ch, 10 bit
- Comparators: 2
- PWM 1
- USART 1
- MSSP 1
PIC Capabilities (16 bit)

- Bits: 8, 16, 32 versions
- Programming Language: MPLAB IDE - C, Assembly
- Operating Voltage: 3.6 Vdc
- Memory: Flash, EEPROM 256 b, RAM 512, ROM 4KB
- Speed: 8, 32 MHz
- 16 MIPS Performance
- Multiplier Unit: 16 x 16
- Divider Unit: 32 x 16
- Timers: (3)16 bit, Watchdog
- A/D: 7 ch, 10 bit, 500 ksp/s
- Comparators: 2
- PWM 1
- UART 1
- SPI 1
- I2C 1
Propeller Quick Start - Parallax
The Propeller Capabilities

- 32 bit 8 core multiprocessor P8X32A
- 160 MIPS
- 64KB EEPROM: 32KB Code, 32KB General
- 32 I/O pins
- Programming Language: SPIN, Assembly
The BASIC Stamp – Parallax

HomeWork Board – BS2 Module

http://www.parallax.com/go/wam

What's a Microcontroller? is our first-step BASIC Stamp 2 tutorial, introducing the essentials electronics and programming side by side. You will write PBASIC programs and build both simple and advanced breadboard circuits with LEDs, pushbuttons, potentiometers, light sensor, a servo, and much more!

Check the complete components listing.

Audience: Anyone interested in getting started with microcontrollers and electronics! What's a Microcontroller? is the gateway to our Stamps in Class program series and is found in hobbyist's homes as well as in middle, high and vocational schools, colleges and universities in pre-engineering and basic electronic courses. The text is designed to accommodate a wide range of ages and skill levels and is a great resource for STEM programs.

Kit Choices:
The What's a Microcontroller? text and hardware is available in many different kit forms; the BASIC Stamp Discovery Kit features the BASIC Stamp 2 and Board of Education:

- BASIC Stamp Discovery Kit - Serial (With USB Adapter and Cable)
- BASIC Stamp Discovery Kit (USB)
- BASIC Stamp Activity Kit Serial + USB (Text v3.0)
- BASIC Stamp Activity Kit - Serial (Text v2.2)
In this text, “BASIC Stamp” refers to the BASIC Stamp® 2 microcontroller module. Designed and manufactured by Parallax Incorporated, this module’s name is commonly abbreviated BS2, and it's the first in the series of modules shown in Figure 1-3. Each of the other modules is slightly different, featuring higher speed, more memory, additional functionality, or some combination of these extra features. To learn more, follow the “Compare BASIC Stamp Modules” link at www.parallax.com/basicstamp.

Figure 1-2
BASIC Stamp 2 Microcontroller Module

Figure 1-3
BASIC Stamp 2 Models, left to right: BS2, BS2e, BS2sx, BS2p24, BS2p40, BS2pe, BS2px
BASIC Stamp BS2 Capabilities

- Programmable in BASIC (PBASIC)
- Supports Windows, Mac, Linux
- First introduced in 1992
- CPU PIC16C57c 20 MHz, 4000 instr/sec
- RAM 32 Bytes, 6 I/O, 26 Variables
- EEPROM 2K Bytes 500 instructions
- I/O 16 bits bidirectional, + 2 SIO
  - Digital Output
  - Digital Input
  - Analog Output – Pulse Width Modulation (PWM)
  - Analog Input – read potentiometer position R/C network
  - Need A/D Chip or Comparator to read voltage input
The BASIC Language

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

• Beginners All-purpose Symbolic Instruction Code
• Invented at Dartmouth University in 1964
• By John Kemeny and Thomas Kurtz
• Designed to allow students to write programs easily
• For the Dartmouth Time Sharing System (DTSS)
• Used in mini computers and early home computers
  – Apple II
  – Comodore PET
  – Radio Shack TRS-80
  – S-100 Systems
  – IBM PC & XT
• BASIC is the “Trailor Trash” of the software programming world!
  It’s SIMPLE and IT WORKS!
• Computer Scientists DO NOT LIKE BASIC (or FORTRAN or Assembly Language)
  (Dykstra et.al. - structured programming: NO GOTOs, NO POINTERS)
• They prefer “structured languages” like ALGOL, Simula, Pascal, Modula
  and they barely tolerate “C” and “C++” but C# and Java seem acceptable!
Original BASIC Programming Statements

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

- **DIM** – defines data variables
- **DATA** – defines collection of data much like an input file but in memory
- **MAT** – defines a matrix (Was in Dartmouth BASIC, no longer in mainstream BASIC)
- **LET** – assigns expression to variable
- **IF ... THEN ... ELSE** – simple decision
- **FOR ... NEXT** – simple loop with iteration
- **WHILE ... WEND** – loop with termination condition
- **DO ... {WHILE} ... {UNTIL} ... LOOP** – loop with termination condition
- **GOTO** – change of program sequence
- **GOSUB** – subroutine call
- **RETURN** – return from subroutine
- **ON ... GOTO / GOSUB** – exception condition
- **PRINT** - output
- **INPUT** – input
- **TAB or AT** – position display cursor
- **REM** – remarks / comments for documenting code
- **END** – marks end of program
- **PBASIC includes a lot more for embedded control & monitoring**
Installing and Using the BASIC Stamp

• Go To URL: http://www.parallax.com/
• Download and Install BASIC Stamp Software
  Including BASIC Stamp Editor and USB Drivers
• Put a 9 volt battery in the kit
• Connect the USB cable to the kit
• Connect the USB cable to the computer
• Operate the BASIC Stamp Editor
Install BASIC Stamp Software

http://www.parallax.com/BasicStampSoftware
Integrated Development Environment

Tokenizer reduces storage requirements in Stamp. Size of the source program does not matter, number and size of variables, number and type of executable statements affect storage requirements. Long variable names have no impact on storage required. Documentation is stripped from the program code.
BASIC Stamp IDE Explorer & Editor Window

The HELLO Program for BASIC Stamp

```plaintext
' ID: Hello1.bs2
' Title: Hello Program for Basic Stamp 2 in PBASIC 2.5
' Author: T G Glagowski / W1TR
' Created: 18-Mar-2012 21:40 - Adapted from Parallax BASIC Stamp Manual:
'         Getting Started
' Description:
'         The BASIC Stamp sends message TO DEBUG Terminal.
'
' {$$STAMP BS2}
' {$$PBASIC 2.5}

DEBUG CR, "Hello, it's me, your BASIC Stamp!"
END
```
BASIC Stamp Debug Window

你好，我是你的BASIC Stamp！
Memory MAP

[Image of a memory map interface]
IDE Help Window

PBASIC Command Reference
(Alphabetical Listing)

Note: For BS1/BS2-compatible commands, syntax shown below is in BS2 format. Some commands may use slightly different formatting with the BS1.

Note: Requires \$PBASIC 2.5\ directive.

Note: Compound, multi-line command; syntax not shown.

Note: Command is accepted by the 24-pin BS2p, BS2pe, and BS2px, but only the 40-pin BS2p40 gives access to the auxiliary I/O pins.

- **AUXIO**
  - **BRANCH**
  - **BUTTON**
Debugging – Terminal Input / Output

• Controllability – putting the system into a known state
  – Setting variables
  – Starting program execution

• Observability – knowing what state the system is IN
  – Viewing variables (system state)
  – Viewing current instruction (algorithm state)
  – Viewing inputs, outputs (I/O state)
  – Stopping / pausing program execution

• Tracing – observing the algorithm and system state changes
  – dynamic behavior

• The DEBUG statement, output format options
• The DEBUGIN statement, input format options
• The PAUSE statement (DEBUGIN can be used also)
PBASIC 2.5 Programming BS2

- Variables: Name VAR Type, Types: Bit (1), Nib (4), Byte (8), Word (16) bits
- Constants: Name CON Constant, Constants: Decimal, %Binary, $Hex, “ASCII”
- Memory: INS, OUTS, DIRS, W0..W12 – 16 words, 32 bytes, 64 nibbles, 256 bits
- Assignment: Variable = Expression
- Output – DEBUG, OUTPUT*
- Input – DEBUGIN, INPUT*
- Statements Unique to BASIC Stamp*
  - Compiler Directives
  - EEPROM and RAM Access*
  - Timing – PAUSE, POLLWAIT
- Linear Sequence of statements
- Change of sequence – (GOTO, GOSUB, RETURN)
- Decision – IF...THEN...ELSE, SELECT...CASE, ON
- Loops – FOR...NEXT, DO...LOOP
- STOP
Expressions

• Data Values
  – Binary
  – HEXadecimal
  – Decimal
  – String
  – ASCII – American Standard Character Information Interchange
  – Constants
  – Variables
  – Arrays

• Operators
  – Binary
  – Unary
  – Precedence

• Algebraic Expressions
• Logical Expression
• String Expressions
• Inequality Expressions
Example Programs

• Hello Program
• Timer
• Digital Output
  – LEDs
  – Speaker / Sound
• Analog Output
• Digital Input
  – Button
  – Switch
• Analog Input
  – Potentiometer
  – Voltage (unable to make example, need more hardware)
The HELLO Program

• Use the DEBUG Statement to Print to Debug Window
• Become Familiar with:
  – Editor / Explorer
  – Debug Window
  – Memory Map
  – Help System
Developing is a Hybrid Activity

- Part Scientist, Engineer, Accountant (rules)
- Part Designer, Artist, Free Thinker (unruly)
- Each Organization Has Its Own Development Standards: Documentation, Naming, Formatting, etc.
- Each Developer Has His Own Style, Make One!
- Be Neat, Orderly, Concise, Consistent, Flexible
- Others May (Have To, Want To) Read Your Work: Document for Maintenance, Reuse
- Each Program is a Work of Art, Literature, Aesthetics
- Don’t Let Rules Get in the Way of the Solution, Creativity
General Overall Program Logic

Start

Initialize

Read Parameters

Finished?

Yes → Clean Up → Stop

No → Control Loop

Input

Process

Output
BASIC Stamp BS2 Circuitry

VIN
+5.5 to 15 VDC
Unregulated Input

VDD +5 VDC Regulated Input / Output

External Regulator May Be Used

General-purpose I/O pins:
Each can sink 25 mA and source 20 mA.

However, the total of all pins should not exceed 50 mA (sink) and 40 mA (source) if using the internal 5-volt regulator.

The total per 8-pin groups (P0 – P7 or P8 – 15) should not exceed 50 mA (sink) and 40 mA (source) if using an external 5-volt regulator.

VSS
System Ground

USB
SIN Serial TX
SOUT Serial RX
ATN Serial DTR

RES
Digital Output to LEDs

- **P15**: VSS (GND) with a 470 Ω resistor to the LED.
- **P14**: 220 Ω resistor to a Dual Color LED.
- **P13**: 220 Ω resistor to a Dual Color LED.
Pulse Width Modulation (PWM) Analog Output

1 VDC = 20% Duty Cycle (51)
2.5 VDC = 50% Duty Cycle (127)
4 VDC = 80% Duty Cycle (204)

\[
\begin{align*}
1 \text{ VDC} &= 20\% \text{ Duty Cycle (51)} \\
2.5 \text{ VDC} &= 50\% \text{ Duty Cycle (127)} \\
4 \text{ VDC} &= 80\% \text{ Duty Cycle (204)}
\end{align*}
\]
Digital Input From Buttons / Switches

- P11
  - VSS (GND)
  - 10 K Ω
  - 470 Ω
- Push Button
- Contact Bounce

- VDD (+5 Reg)

- P10
  - 10 K Ω
  - 470 Ω
- VSS (GND)

- P9
  - 10 K Ω
  - 470 Ω
- VSS (GND)
Speaker Output

![Diagram of speaker output with components labeled: P8, 22 µF, 33 Ω, 8 Ω, VSS (GND), and 0.0022 µF.](image)
Analog Input – Potentiometer
Analog Input – Voltage

Basic Stamp BS2 has NO A/D
Need A/D Chip, Comparator, or OpAmp

VDD (+5 Regulated)
VSS (GND)
Vin
GND
Vref
B1
B8
Sign
ENB
A/D Converter
P0-P8

VIN Analog

VSS (GND)

Comparator

PWM Analog Output

VDD (+5 REG)
P5

P6

470 Ω

0.22 µF

VSS (GND)

VIN Analog

VSS (GND)

VSS (GND)
Microcontroller Summary (1)

• **Arduino**
  – Available
    • @ Radio Shack, $70 - [www.radioshack.com](http://www.radioshack.com)
    • @ Arduino - [http://store.arduino.cc/ww/index.php](http://store.arduino.cc/ww/index.php)
    • @ Amazon, Digi-key, Mouser, Newark
  – Additional modules needed / available
  – Programmable in C / assembly Language, w/IDE

• **PIC**
  – Available
    • @ ARRL: Kit, $140 – [www.arrl.org](http://www.arrl.org) - Makes CW Keyer
    • @ Microchip - [www.microchip.com](http://www.microchip.com)
    • @ Amazon, Allied Electronics, Avnet, Digi-Key, Mouser, Newark
  – Programmable in C / assembly language, w/IDE (MPLAB)(simulator)
  – PICBASIC language to program PIC is available from meLabs
  – Version of PIC in BASIC Stamp
Microcontroller Summary (2)

• **Parallax Propeller**
  
  – Available
  
  • @ Radio Shack, $40 – [www.radioshack.com](http://www.radioshack.com)
  
  • @ Parallax - [www.parallax.com](http://www.parallax.com)

  – Hosts on Windows, Mac, Linux

  – Programmable in SPIN, Assembly Language

  – 32 bit 8 core processor 64KB EEPROM

• **Parallax BASIC Stamp**
  
  – Available
  
  • @ Radio Shack, $40 – [www.radioshack.com](http://www.radioshack.com)
  
  • @ Parallax - [www.parallax.com](http://www.parallax.com)

  – Hosts on Windows, Mac, Linux

  – Programmable in PBASIC

  – BS2 Uses PIC16F57 chip, 20MHz Clock, 32Byte RAM, 2K EEPROM

  – Higher level chips use Parallax proprietary SX chips
Question: Can the BASIC Stamp BS2 Be Used for the Automatic Antenna Tuner Project?

- Consult the Excel Spreadsheet of Antenna Tuner I/O’s
- 3 Motors, 2 Directions Each, 6 out
- Forward Power Analog In, 1 out, 1 in (external comparator)
- Reflected Power Analog In, 1 out, 1 in (external comparator)
- Coil Position Pot Analog In, 1 in/out
- SWR Limit Pot Analog In, 1 in/out
- Capacitor(s) Home Position, 2 in
- AutoTune Start/Stop, 1 in
- **Total 15 I/O – YES, but no band switch (11 in), No Cap/Coil UP/DN (6 in)**
- Is there enough Data Memory? Program Memory? (Dunno?)
  - MARGINAL!
- Maybe Use BS2PX which has an analog comparator, need analog switch!
- Maybe Arduino, PIC would be better?