

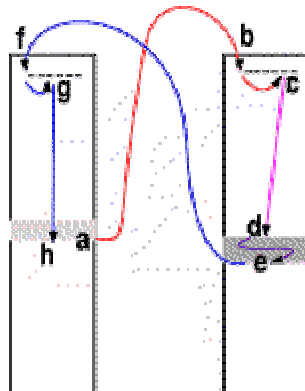
BASIC Stamp IISX Bank Switching

The BASIC Stamp IISX differs from the Stamp II in timing (baud constants) and architecture (banks). Check out our article the Stamp IISX at <http://www.solutions-cubed.com/Stamp/oct99.pdf>.

Also visit <http://www.emesystems.com/BSIISX.htm> for a great tutorial on IISX variable allocation, data acquisition scripting, using scratchpad RAM and bank switching using pointers.

Application notes for interfacing our Mini-Mods to the BASIC Stamp IISX demonstrate bank switching by using pointers and the GET and PUT commands. Pointers to routines are defined in single bank, which is responsible for calling routines in other banks. Automated data acquisition using multiple I/O devices is simplified to procedural scripts using this method.

Following is an approach developed by Tracy Allen of EME Systems. This approach works well if you want to fully utilize BASIC Stamp IISX program memory.



a) routine in one bank needs to call a routine in another bank. It puts the return index on the stack, and loads the pointer to the target routine into a special variable, and issues a RUN command for the bank where the target program exists. This bank number and index are known in advance as part of the programming ritual. **b,c)** the RUN command begins execution in the target bank. A branch instruction looks up the index to take program execution to an actual address for the target routine. **d,e)** the target routine executes, then pulls the return index and bank number off the stack, and RUNs the return bank. **f,g)** the RUN command begins execution back at the first bank, where a branch instruction looks up the return index to branch to the address to continue program execution. **h)** the program resumes execution where it left off.

Source: <http://www.emesystems.com/BS2SX.htm>

SOLUTIONS CUBED

Electronics Prototyping, Custom Design, Product Development

Commands for the systems I/O devices can be defined in banks 1 - 7, and pointers are defined as constants. Pointers to the commands in bank 0 must be saved to the IISX scratchpad RAM using the GET and PUT commands. Once entering bank 1, the program decides where to go by BRANCHing on the ROUTINENUM variable, defined in the main RAM of the IISX.

ROUTINENUM is defined as the nibble 1 of ROUTINE, which is defined as a byte. Thus, nibble 0 is the bank number. In order to call a function in bank 1, and return to the next section of code in bank 0, the user must place the pointer of the next TEST_PROGRAM (bank 0) into scratchpad RAM (into RAM location STACK con 62) by using the PUT command.

Next, assign the Pointer of the desired Motor Mind B command to ROUTINE, and execute the RUN command on it.

Here is the code:

In bank 0:

```
PUT STACK, POINTER_OF_NEXT_ROUTINE_IN_BANK_0
ROUTINE = POINTER_OF_ROUTINE_IN_BANK_1
RUN ROUTINE
```

Now in bank 1. To return to bank 0, we want to:

```
GET STACK, ROUTINE
RUN ROUTINE
```

ROUTINE will be the POINTER_OF_NEXT_ROUTINE_IN_BANK_0. Pretty simple, huh? Using this scheme for bank switching, make sure that your routines in each bank are placed in the sequence that you defined their pointers as. Otherwise, you will branch to the wrong routine.