

## Re: Learning to use PICS

**Source:** <http://sci.tech-archive.net/Archive/sci.electronics.basics/2004-12/0014.html>

---

**From:** Byron A Jeff ([byron\\_at\\_cc.gatech.edu](mailto:byron_at_cc.gatech.edu))

**Date:** 11/30/04

Date: 30 Nov 2004 10:22:34 -0500

In article <41AB98C1.8000002@netscape.net>, mike <spamme0@netscape.net> wrote:

-David Harper wrote:

-> All,

-> [Dave's questions about banking snipped.]

-

-If your objective is to understand the architecture of a 16F84, you're

-on the right track to understanding an obsolete and arcane part.

-There's only one reason to use paged/banked memory...Price! And that

-advantage has long since been nullified by technological advances.

That's not true. The PIC has a legitimate reason for having a banked memory architecture. It facilitates a uniform instruction set where virtually every instruction executes in one instruction cycle.

The non Harvard architecture of the chip was constructed so that instructions on file registers could hold a 7 bit address. As most parts have more than 128 addresses, banking is required in order to access them.

-

-While it's possible to understand the inner workings of a PIC16 part,

-the details of actually making it work in practice will drive you nutz.

-All it takes is one mistake in setting the page registers and your

-program won't work. And when you edit it, things move around and things

-that used to work are now broke.

Not true for the data memory, and mostly no longer true for program memory unless you cross a 2K page boundary.

-

-Now, any programmer who does this all day can manage it. Those of us

-who drag out a PIC once a year to do a quick project tend to get

-buried by those details. I want to hack out some code and have it work.

-I don't want to go searching for errors caused by my forgetting some

-processor quirk.

There's a simple solution to all of these problems, that David planned to take anyway: use a high level language that hides all of those details. Languages like JAL, XCSB, and PicBasic nullifies the vast majority of

banking issues.

And using the BANKSEL directive obviates most of the issues in assembly.

Finally if assembly programmers used the available linkers, it would handle virtually all of the banking issues that arise.

–

- One option is to let someone else manage all those details. A compiler
- should be smart enough to keep you out of trouble. I use PicBasic, but
- there should be lots of others.

Right.

–

- Second option is to use a processor that doesn't page memory. I think
- the PIC18 series fixes up most of this mess, but I haven't tried 'em yet.
- Think I might go with AVR if I didn't already have investment in tools
- for PIC.

Most but not all. The PIC 18 has two directly accessible pages, and a MOVFF instruction that can access memory from any bank.

–

- All boils down to what you're trying to accomplish. Can't think of any
- reason to want to understand an obsolete part. In sample quantities,
- all PICs are the same price...free.

Definite agreement on that statement.

BAJ