

# Re: Purchase microcontroller dev. kit

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2006-09/msg06148.html>

---

- *From:* [nico@xxxxxxxxxxx](mailto:nico@xxxxxxxxxxx) (Nico Coesel)
  - *Date:* Thu, 28 Sep 2006 17:52:35 GMT
- 

David Brown <david.brown@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

Nico Coesel wrote:

David Brown <david.brown@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

Nico Coesel wrote:

"David L. Jones" <altzone@xxxxxxxx>  
wrote:

```
try to compile the following on an AVR:  
const char a[]={ "hello world" }  
char b[30];  
char *p;
```

```
p=a;  
strcpy(b,p);  
strcat(b, "?");  
printf("%s %s", a, b);
```

It won't work because the different memory areas will screw-up the pointers. Some very smart and expensive compilers may be able to solve these problems at the expense of speed.

That works perfectly well using avrgcc. Even on compilers which don't support it directly, the way to deal with code like that is clearly explained in user manuals and FAQs. You weren't suggesting that a beginner should just wade in without looking at the manuals?

## Re: Purchase microcontroller dev. kit

Of course, no one doing embedded development on a small microcontroller would write code like that anyway – using printf is a sure sign you don't know what you are doing.

Is it? All of my projects use printf (which usually comes in several grades to tailor memory requirements) to print status information, debug information and responses to commands send to a tiny command line interpreter (a little more than a string compare on a list with strings). And this is not even my idea.

Being able to have a sensible dialog with your hardware makes testing and debugging much easier. Having a command line interface is also very handy for field service. For instance: implement a command to have it print the version & build date, and presto, you know exactly which software is in the device.

Communication with your embedded system is an important aid to debugging, and a small CLI is often useful. But printf is not the way to go about it on a small microcontroller – the overhead is too big. It's frequently (though not always) a sign that the programmer is used to PC programming and PC "printf debugging", and has not thought about the costs. Look through any FAQ or mailing list for small microcontrollers – they are full of questions about why the code is so big, with the answer being either the use of printf, or the use of floating point. Of course, as with floating point, there are times when printf is the right choice, even on a small microcontroller – but it's a rarity, and not something to encourage in newbies.

The amount of program space used to be a problem and I did have my own printing routines, but in the end printf turned out to be more efficient to use. Most devices have plenty of flash nowadays. A decent (basic, non-floating point) printf implementation may cost around 1500 bytes of flash. Not really a problem anymore IMHO. It may become an issue in extremely cost sensitive products, then again, time to market is also an important factor.

—

Reply to nico@nctdevpunt.nl (punt=.)  
Bedrijven en winkels vindt U op [www.adresboekje.nl](http://www.adresboekje.nl)