

Re: universal programmer

Source: <http://sci.tech--archive.net/Archive/sci.electronics.design/2005-12/msg04503.html>

- *From:* "Abstract Dissonance" <Abstract.Dissonance@hotmail.com>
 - *Date:* Mon, 26 Dec 2005 14:39:37 -0600
-

"Jack W." <jackw@xxxxxxxx> wrote in message
[news:KvRrf.33098\\$0p5.310066@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:KvRrf.33098$0p5.310066@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)
>
> "Abstract Dissonance" <Abstract.Dissonance@hotmail.com> wrote in message
> news:11qrmkj4uu181bc@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>
>> "J. David" <jdavid@xxxxxxxx> wrote in message
>> [news:QIfrf.109779\\$dd6.1481029@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:QIfrf.109779$dd6.1481029@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)
>>>
>>> "Abstract Dissonance" <Abstract.Dissonance@hotmail.com> wrote in message
>>> news:11qoqlquql4kd3@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>>> Since I'm trying to get into MCU it seems I will need a programmer to
>>>> get anywhere. From looking online it seems that any "decent" programmer
>>>> is pretty expensive(1k+) and those that are cheap seem to offer very
>>>> few features and few chip support.
>>>>
>>>> I was thinking that it shouldn't be very difficult to program just
>>>> about any chip by using a computer if the computer had "access" to all
>>>> the pins on the chip. From looking at a few data sheets it seems that
>>>> its very easy to program a MCU by simply handling the procedure through
>>>> the software.
>>>>
>>>> heres a quote of how to program the PIC16C84
>>>>
>>>> The device is placed into a program/verify mode by
>>>> holding the RB6 and RB7 pins low, while raising the
>>>> MCLR pin from VIL to VIH (see PIC16C84 EEPROM
>>>> Memory Programming Specification (DS30189)). RB6
>>>> becomes the programming clock and RB7 becomes
>>>> the programming data. Both RB6 and RB7 are Schmitt
>>>> Trigger inputs in this mode.
>>>> After reset, to place the device into programming/verify
>>>> mode, the program counter (PC) points to location 00h.
>>>> A 6-bit command is then supplied to the device, 14-bits
>>>> of program data is then supplied to or from the device,
>>>> using load or read-type instructions. For complete
>>>> details of serial programming, please refer to the In-Circuit
>>>> Serial Programming Guide (DS30277).

Re: universal programmer

>>>> For ROM devices, both the program memory and Data
>>>> EEPROM memory may be read, but only the Data
>>>> EEPROM memory may be programmed.
>>>>
>>>>
>>>> It seems to me that if most chips follow a very similar method where
>>>> you use a few pins(lets say arbitrary too) to get it into program mode
>>>> and a serial communication on one pin to transfer the code then it
>>>> would be extremely easy to program any of those chips by using a
>>>> computer(with just about any number of pins). Is this the actual case?
>>>>
>>>> Lets suppose I have a device that simply lets me access any of the pins
>>>> on a chip from software.... could I then not use the software to
>>>> program the chip? If so, is this true of pretty much most of the MCU's
>>>> and EEPROMS or just a few? If not, what are the reasons why I couldn't
>>>> do this?
>>>>
>>>> Only thing I can think of that could prevent it from being so easy is
>>>> supplying the proper voltage to the right pin(but this shouldn't be
>>>> that hard) and getting the right clock into the chip. Other things
>>>> like having ot have resistors and stuff on certain pins for certain
>>>> reasons and such would surely scew this method up as then it might
>>>> become to complicated to do(well, it would be just easier to buy a
>>>> "universal programmer"). If there are no resistors needed and I can
>>>> generate the clock from the software in a computer and potentially the
>>>> voltage(or just allow the voltage to be selectable by the hardware and
>>>> on what pin it should be on) then I can't see why this would be that
>>>> difficult. Definately would be much cheaper to do than buying a 1000\$
>>>> programmer.
>>>>
>>>> Is it safe to say that the generalization of the above procedure for
>>>> the PIC16C84 applies to almost all other MCU's? And what about those
>>>> MCU's that cannot be programmed in that way, what makes them so
>>>> different?
>>>>
>>>> Any ideas?
>>>>
>>>> Thanks,
>>>> AD
>>>>
>>> For myself, using microcontrollers in DIP packages (no SMD) from 8 to 40
>>> pins, I use a 40-pin ZIF socket (where the uC to be programmed is
>>> inserted) and also a small 50-pin connector which can be a DB-50.
>>> The first 40 pins of this connector are connected directly to the ZIF
>>> socket pins and five other pins are connected to the programmer well
>>> known signals, that are: MCLR (or Vpp), Vcc, ground, DATA and CLOCK.
>>> These five signals are then hardware-strapped to the appropriate first
>>> 40-pins onto a 50-pin plug that fits into the connector for the
>>> appropriate uC to be programmed (many share the same programming
>>> pinouts).
>>> If different uC with a new programming pinout is released, I just buy a

Re: universal programmer

```
>>> new 50-pin plug and hardware-strap it for the new pinouts, that's all!  
>>> Take a few minutes and soldering iron.  
>>> So far, I could have coped with the many different programming pinouts  
>>> Microchip (and others) have conceived to make our lives more difficult  
>>> to program their chips :)  
>>>  
>>> Johnny  
>>>  
>>  
>> But wouldn't it be nice to have a ZIF socket on your programmer that  
>> supports 100 pins? ;) Surely the complexity to handle 10 pins or 1000  
>> pins is virtually the same... most of those pins are useless and don't  
>> need to be messed with? If so then if one could "rearrange" the pins so  
>> that only the ones needed are put at the top then its like just having a  
>> 10 pins or so(or whatever you want) and one wouldn't need to actually  
>> handle all those useless pins.  
>>  
>> Thats my dream though. I figure that would could have some software code  
>> that looks like this:  
>>  
>> out Pin3, 1  
>> out Pin13, 0  
>> serial_out(Pin19, bufffer)  
>>  
>> while some other chip might be  
>>  
>> out Pin1, 1  
>> out Pin2, 0  
>> out Pin8, 1  
>> serial_out(Pin73, bufffer)  
>>  
>> (hypothetically... it would be more complicated than this but just to  
>> give an idea)  
>>  
>>  
>> But the problem is having "access" to all those pins... one needs a sorta  
>> router to route the pins used to the actual pins on the chip... I'm not  
>> sure if this is all that difficult or not... maybe the real issues just  
>> lie in the pratical side of things where certain chips have different  
>> characteristic like different voltages and clocks and all that(but I  
>> still feel it shouldn't be all that hard to do).  
>>  
>> Thanks,  
>> AD  
>>  
>  
> It looks like some kind of reprogrammable array. Maybe an FPGA could do  
> the trick.  
> I think I'm gonna dig more deeply... could be interesting!  
>  
>
```

Re: universal programmer

Re: universal programmer

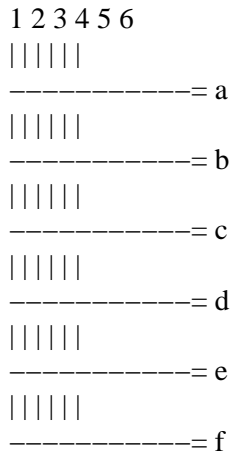
>

yeah, basically a "programmable matrix" one could view it as a matrix of transistors

```
+++
|+|+|
+++
|+|+|
```

or something like that. This way one could turn "make" paths by turning on and off certain combinations of transistors. For N wires there are N^2 transistors and hence N^2 "control" pins. But if one just wants permutations then this is reduced drastically since once one control pin is selected it should determine the state of many others.

now it might require 2N^2 transistors for analog(to get both polarities) and some other transistors for the switching logic but ultimately it will accomplish what I want. I'm sure there is a better way but this is what I thought up of trying to do it the "hard" way. I think its similar to an FPGA but the configuration is different(doesn't use and and or gates directly).



basically one can "route" pin x into pin y by tracing a path from the start to finish. There are many ways to do this but some are easier than others. The problem is that one doesn't want two inputs to be "combined" so we would need to add transistors to "control" the flow. i.e., if I wanted to map pin 1 to pin f I could do it like this



Re: universal programmer

|
-
|
-----= f

or pin 3 to pin b

1 2 3 4 5 6
|
-
|
-----= b

Then its just a matter of combining the above to get different permutations of the input.

I think the FPGA is very similar in its idea except it is designed for digital systems instead of analog. In the above it should work with analog or digital (but the controlling is digital ofcourse). I'm still reading up on FGPA's and other devices like it though.

Maybe the idea could be extended from RAM where the flip flop holds a state. But the output of the flip flop is linked to the base of a transistor and controls that transistors state... so we can "write to memory" and it will allow us to program the array.

There might be other ways to do what I'm talking about but its just some ways I've thought about doing it.

Thanks,
AD

• **References:**

- ◆ **universal programmer**
◇ From: Abstract Dissonance
- ◆ **Re: universal programmer**
◇ From: J. David
- ◆ **Re: universal programmer**
◇ From: Abstract Dissonance
- ◆ **Re: universal programmer**
◇ From: Jack W.

- Prev by Date: **Re: TOFP to TOFP converter?**
- Next by Date: **Re: 1-32 Khz in 1khz divisions**

Re: universal programmer

Re: universal programmer

- Previous by thread: *Re: universal programmer*
- Next by thread: *1206 thermals*
- Index(es):
 - ◆ *Date*
 - ◆ *Thread*