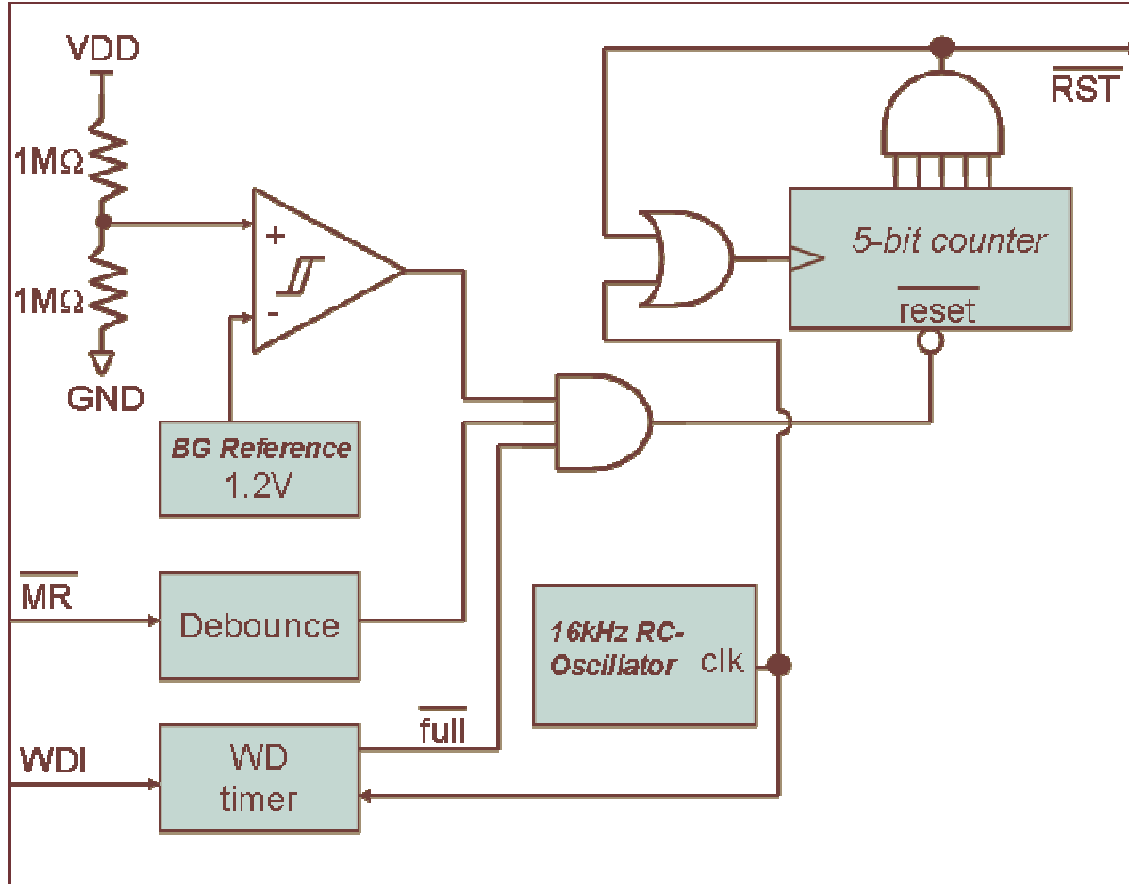


HOMEWORK ASSIGNMENT #4Due Friday, March 27th, 2009**1. (14 points) What is it?**

- A block diagram of a chip is given above... What is it used for?
- Under what 3 conditions does its active low output become active?
- What is the minimum amount of time that $\overline{\text{RST}}$ can be asserted?
- At what approximate threshold level VDD has to cross?
- What is the purpose of the WDI input?
- What is the purpose of the MR input?
- Would this be an appropriate chip to use with our processor?

2. (20 points) To Sleep or Not to Sleep?

Assume that you have an ADuC7026 that is operating with a perfect 32.768 kHz crystal and the 1275x PLL. The processing requirements are such that the processor will be notified every 2ms by an external circuit (using IRQ0) that it needs to do its computations. When the processor completes the computations, it can go back to a low-power mode (if applicable) until the next time it is notified. It has been determined that the computations take 6,000 HCLK cycles. How could you use the various power control operating modes and the HCLK divider (CD) to control the power consumption while still performing the required computations? Determine the best use

of the clock divider for each of the five (5) power control operating modes (including staying active continuously), and then select the best option to minimize overall power consumption. **HINT:** To get full credit you need to figure your strategy (and CD) for each mode, and estimate the power consumption for each mode.

3. (10 points) Timer1 as a counter, and Port1 as an output

Create a program that will setup Timer1 as an up counter, counting Port0.6 transitions. The pre-scale should be 1:1. Port1 should be initialized as an output, and will display the value of the lower 8-bits of the Timer1 count.

The main loop should simply read Timer1 value and write it to Port1 output.

Under *Peripherals* → *General Purpose Input/Output* you can get GUI windows that you can use to control Port0.6, and see the contents of Port1. When you run the program you should be able to click on the Port0.6 input, and see the count in the Port1 output increment every time Port0.6 is transitioned high.

Important: Submit your program source code file *teamX_cnt.s* in the homework #4 dropbox in Learn@UW. Also, submit a **paper copy** of *teamX_cnt.s* with the rest of the assignment.

4. (20 points) Storm Stopper (game at Rocky Rocco's Pizza)

You will use the Timer0 and the GPIO to implement a simple game.

First you need to setup Timer0 to give you approximately a 0.1 second time base. (HINT: at a CD=3 (default config) with a 32kHz clock into the 1275 multiplying PLL (also default config) the MSB of Timer0 will toggle about once every 0.1 seconds if the Timer0 pre-scale is 16).

You are not to use interrupts for this program. All actions will occur as the result of polling.

The 8-bits of Port0 will be configured as outputs. Think of them as connected to LED's. Start with a 1 seeded in the LSB of Port0. Every 0.1 seconds it should be shifted left one position. When it is at Port0.7 (MSB of Port0) then it should roll over to Port0.0 (LSB of Port0) next.

Port1 bits should be configured as output to display the score. Obviously score starts at zero.

Port2.0 (LSB of Port2) will be the stop_n/start input bit. By default this port pin will start with a logic 1. When Port2.0 is a 1 the loop is running, and the lit LED of Port0 is circulating around with a 0.1 second update rate.

When you click on Port2.0 peripheral input control to set it to zero you are stopping the lit LED from circulating.

The objective of the game is to stop it when it is on Port0.6. When your polling loop detects Port2.0 falls it should check to see where the LED was. If it is on Port0.5 or Port0.7 your score is increased by 1 point. If it is on Port0.6 your score is increased by 10 points. For all other locations your score remains unchanged for that try. At this point the score should be updated on

Port1. The program should now be in a spin loop waiting for Port2.0 to be raised. When it is raised the LED should resume movement.

You should have another counter, counting the number of tries. When there have been 5 tries to stop the LED the program should vector to a section to handle `game_over`. In the game over the program should spin until Port2.1 is lowered and raised. Once it is the code should reinitialize itself and start over (i.e. score cleared, try counter set to zero, and if Port2.0 is high the LED moving again).

Base your code on the Kiel uVision3 sample project on the course web page, renaming `main.s` to `teamX_prob4.s`. All references to the GPIO MMRs should use the definitions in `aduc7026.inc`. Note that the definitions are set up to include a base address for a group of registers, and smaller offsets within the groups. A typical usage of these definitions would be as shown below.

```
LDR R7, =(GPIO_MMR_BASE) ;load base address
MOV R0, #0
STR R0, [R7, # GPIOCON] ;set all Port0 pins to GPIO function
```

To test your program, you should open windows to view the pin states of P0, P1 and P3 – you can do so by selecting **Peripherals** → **General Purpose Input/Output** from the menu when running the debugger. By checking/unchecking the bits in the I/O Pins block for Ports 2, you can create the game inputs, even while your program is running.

Important: Submit ONLY your program source code file `teamX_prob4.s` in the homework #4 dropbox in Learn@UW. Also, submit a **paper copy** of `teamX_prob4.s` with the rest of the assignment.

5. (16 points) Using the ADuC7026 Watchdog Timer

Using the sample project provided as a starting point, write code to configure the watchdog timer to give a ~25ms timeout while operating with the **Secure Clear Bit set**. Base your code on the Kiel uVision3 sample project on the course web page, renaming `main.s` to `teamX_prob6.s`.

When your code starts up, check if a watchdog reset has occurred (RSTSTA bit 1). If it has, branch immediately to the spin loop at the end of the program. Otherwise, set GPIO Port2 as all inputs, then configure and enable the watchdog timer. Use an initial seed value of 0x47 for the secure clear mode. After the watchdog timer is configured and operating, your code should enter an indefinite loop where it does the following;

- If pin P2.0 is read as a 1, continue to loop checking P2.0. Note that this will eventually cause a watchdog reset since we will not reset the watchdog timer.
- If pin P2.0 is read as a 0, check if the watchdog timer has counted down below one half (1/2) of its original load value. If it has, then reset the watchdog timer using the appropriate secure clear value (you will need to calculate it).

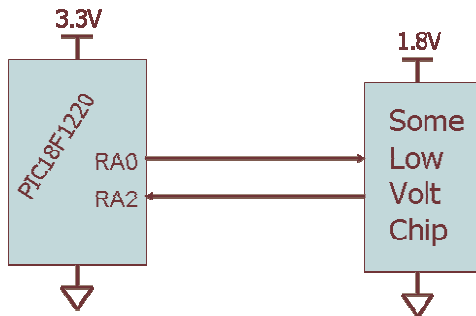
As a suggestion for organizing your work, you may want to get your code working without putting the watchdog timer into secure clear mode first (clearing the watchdog timer is easy

when not in secure clear mode). Only after that is done, add the secure clear mode functionality. All references to the GPIO MMRs should use the definitions in *aduc7026.inc*.

Important: Submit ONLY your program source code file *teamX_prob6.s* using the homework #4 dropbox in Learn@UW. Also, submit a **paper copy** of *teamX_prob6.s* with the rest of the assignment.

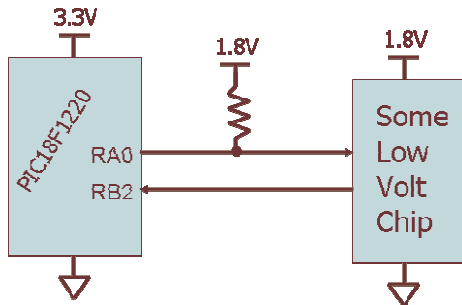
6. (10 points) Interfacing Different Voltages

Here we consider a very common situation where multiple power supply voltages are in use in a digital system.



A PIC chip is being interfaced with a 1.8V chip. There is one input and one output. The output from the 1.8V chip contains CMOS full push/pull driver (i.e. $V_{oh} = V_{DD}$, and $V_{ol} = V_{SS}$).

It was originally intended to hook it up as shown here. But luckily it was caught in a design review that this would have problems.



The proposed solution was to make the changes shown. Now the input from the low voltage chip comes into RB2 instead of RA2. The output to the low voltage chip is still from RA0, but it has this pullup resistor, and the guy in the design review said to: “run it open drain”.

- What were the problems with the original design? (what would have gone wrong)
- Why does changing to RB2 help?
- What did the the guy in the design review mean by “Run it open drain”. How does that solve this problem?

7. (10 points) Quiz Question

Design one original quiz question operating at Bloom’s Taxonomy level 3 for any material covered in Module 4. This must test one of the educational objectives (see <http://eceserv0.ece.wisc.edu/~morrow/ECE353/objectives.pdf>) in a specific problem. Explicitly state which particular educational objective you are attempting to test. Provide a complete, detailed solution to your question.