Arduino Frequency Counter - Hardware Description

Input Buffer

The two NAND gates U3A and U3B form the input buffer to the counter. D1 and D2 provide over voltage protection to U3A and resistor R1 limits the current through the two protection diodes. To improve sensitivity U3A is used in a quasi linear mode by the feedback resistor R9 and the bias arrangements formed by RV1 and R7. The second NAND gate U3B in conjunction with U3A forms a schmitt trigger with positive feedback applied by resistor R8. Capacitor C16 is required to prevent edge jitter on the square wave output at low frequencies. Input impedance is essentially set by the value of R7 and for this reason needs to be high. With RV1 set to around half VCC an input sensitivity of around 250mV at low frequency (10KHz) rising to approximately 480mV at 100MHz was observed. These figures were obtained using NXP HC logic devices. No instability was noted with this circuit configuration.

Switching between the standard input and the pre-scaler is achieved via U3C, U4A and pin 4 of U3B. A low level signal from the Arduino microcontroller to “Sel” switches the main input while a high signal to “Sel” selects the prescaler.

Timing System

The real time clock (RTC) DS3231 module U5 is controlled by the Arduino via an I2C bus (SDA, SCL); control, time and date information is sent across this bus. The RTC is programmed to provide a 1Hz square wave and this is further divided by 2 using half of a 74HC74 D type flip-flop (U1B). The resulting one second pulse is used to gate the incoming frequency to be measured by U4C. The output of U4C clocks U7B (74HC393) via U4D. The one second timing pulse is monitored by the Arduino (A3/pin 7), when this line goes low the incoming frequency is inhibited and the Arduino stops counting the output from the '393 counter. The Arduino then transitions into processing mode whereby it updates the display with the measured frequency.

The circuit was designed such that the Arduino can provide a timing pulse instead of the RTC module if so required. The reason for this addition was to enable the device to measure frequencies using a shorter time pulse albeit with poor accuracy and lower resolution. Such functionality can be useful when a fast changing frequency is being monitored. To change the timing mode a low level signal is sent to the 74HC74 preset pin (pin 10 U1B) which then enables NAND gate U4B. The Arduino can then generate a timing pulse via A2/pin 6 (U6).

Counting System

The measured frequency is counted by U7, a 74HC393 8 bit divide by 256 counter. The output from this counter is then passed to the Arduino's T1 16 bit counter. The T1 counter is rated to 20MHz and provides plenty of bandwidth when “front ended” by the '393 counter. The T1 counter is configured such that when it overflows it triggers an interrupt within the Arduino which in turn increments a software counter (overflow count register).

When the timing pulse returns to a low state the counting period is complete. At this point the frequency is the number of pulses counted by the '393, the T1 counter and the value in the overflow count register. To obtain the count from the '393 the Arduino first ascertains the state of pin 8 of the counter, if the state is low the count must be less than 128 if high it is 128 or greater. The Arduino then sends 128 pulses to counter's clock input via “Flush Counters” and calculates the value held in the counter. With this information plus the count from the T1 counter and the overflow count register the measured frequency is determined. If the frequency calculated is below a defined value (typically 100) the count display reverts to time and date, thus the display shows useful information when no input is present.

Switches

Three switches are used to set various parameters within the system SW1, SW2 and SW3. These switches are conditioned to suppress key bounce by the addition of two 10k resistors and a 10nF capacitor. This conditioning circuit was taken from the Bourns application note on encoders. The header P4 was included in case a user does not want to use small PCB mounted switches but rather a remote mounted set of switches.

Power Supply

The circuit can be powered from either an external source up to 9v or from a USB connection to the Arduino. When powered via header P3 the Arduino's internal 5v regulator is used to supply the circuit. This regulator is rated at 500mA however it does not have a heat sink so its power dissipation is limited. The frequency counter consumes 60 to 80mA in operation hence a supply of of 9v (or less) is recommended.

Prescaler

The prescaler was added as an optional extra. The circuit configuration is taken from the MC12080 application note. The prescaler input is protected by diodes D3, D5 and resistor R2; at VHF frequencies the input is essentially 50 ohms. Resistor R10 was included (typically 100k) as there have been reports of instability and a resistor shunting C7 is supposed to reduce stability problems. These reports may be due to issues with the layout of the MC12080.

The output from a MC12080 is only 1.2 volts which isn't enough to drive a TTL input, this is the reason for the two bias resistors R5 and R6 which places the input to U3D at half VCC. Although not a particularly elegant solution it does seem to work.

Arduino Pin Connections

To enable possible serial output from the counter Rx and Tx (pins 16 and 17) are routed to the header P5. The initial design used A6 and A7 as the output pins to drive the “Flush and Clear” lines to the '393 counter. Unfortunately it was later found that these two pins can only be used as inputs, they are not general GPIO pins. This seems to be a *feature* of the Atmel ATmega328 processor used within the Arduino Nano. The header P6 is of nugatory value to the design and future releases of the circuit will see it deleted.

AOD 16th November 2016