Frequency Measurement using Timer 2 on a MicroConverter

1.0 Introduction.

This technote describes the implementation of a single pin measurement of frequencies up to 500kHz using the Timer 2 counter input pin. The code accompanying this technote implements a frequency acquisition system. This could be combined with a voltage measurement via the adc so both frequency and voltage of the input signal are kept track of. The measured is displayed on a HD44780 compatible LCD screen. The 'display.c' code accompanying this technote implements the routines necessary to output to the LCD display.

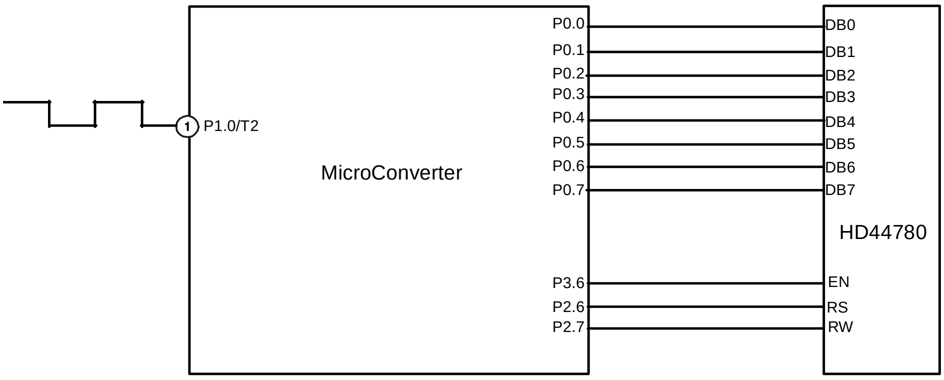


Figure 1. Setup of LCD Display

2.0 How the frequency is measured.

Since frequency is the number of cycles of a given waveform recorded over a second, we need to measure both the number of cycles and the time taken for that number of cycles. To simplify matters we measure the number of cycles over a second. This is the frequency of the waveform.

In order to measure the number of cycles we need to keep a record of the number of times a 1-0 transition occurs on the waveform. This can be done using the timer 2 counter input pin. This pin increments the timer 2 registers on a 1-0 transition.

To measure a second two different approaches may be taken.
The first way involves setting up Timer 0 with reload values such that it overflows when 10ms has elapsed. By counting 100 of these overflows we can measure a 1 second interval. An example of this method is given in freq.c

The time can also be measured using the Time Interval Counter which can be set up to interrupt the core after a second has elapsed. This has the advantage that the core is interrupt less that if a segmented count using Timer 0 and thus is free to carry out tasks in the meantime. An example of this method is given in tic.c

Since the timer 2 registers can only hold a 16 bit result the maximum frequency that is measurable is 65535 Hz , in order to extend this a track is kept of the number of times that timer 2 overflows during the 1 second interval. This is then factored in to calculate the actual frequency during this time.

Also implemented in both methods is a calibration function. If a 100kHz square wave is inputted to P1.0 and the INT0 button is pressed the software will calibrated the frequency measurement to this. This is necessary to compensate for errors that interrupt latency introduces at the higher frequencies, though since this error is only significant above 10kHz the calibration routine ignores the gain error below this frequency.

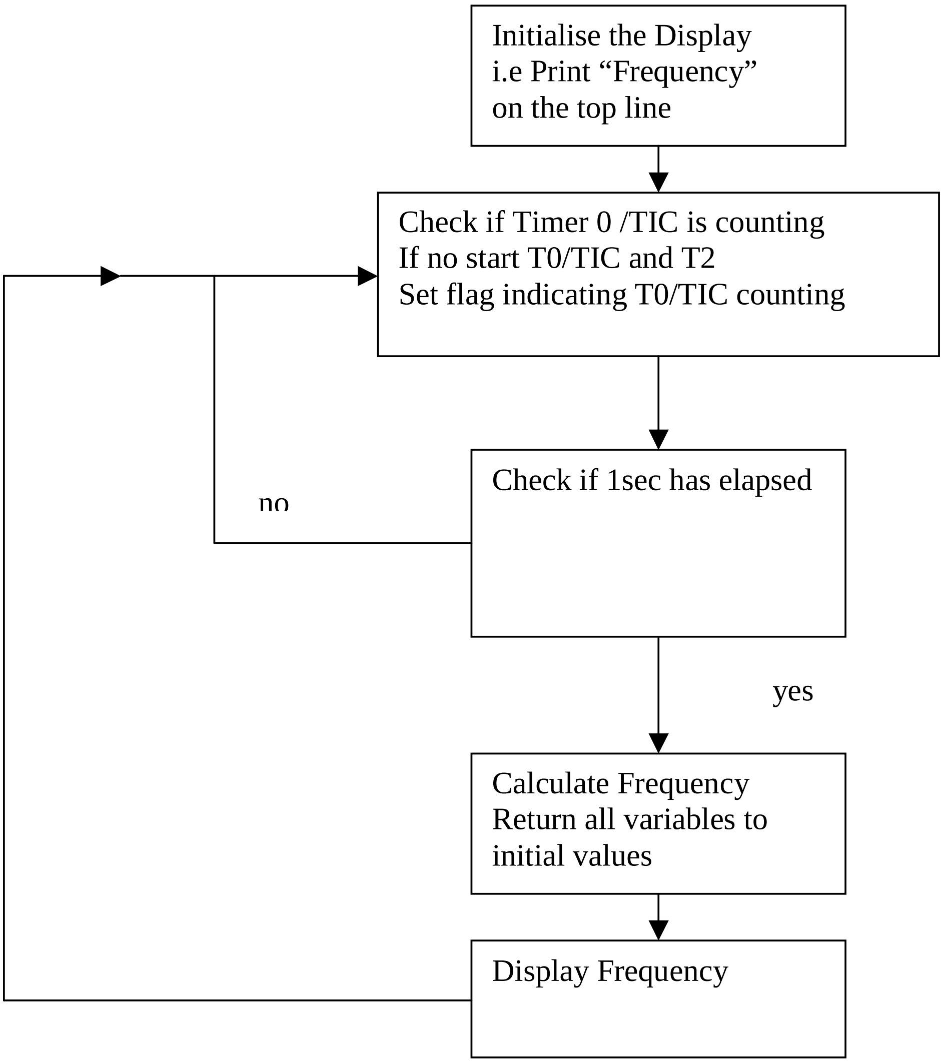
3.0 Program Example.

The above concept has been implemented in the following code example. In this code the resulting frequency is outputted to an LCD display. Though this code is written in c the concepts of communicating with the LCD is explained in technote uC014. This code has been written to be used on the ADuC816, ADuC824 and ADuC834 MicroConverters, thought he similar principles can be used on other MicroConverters.

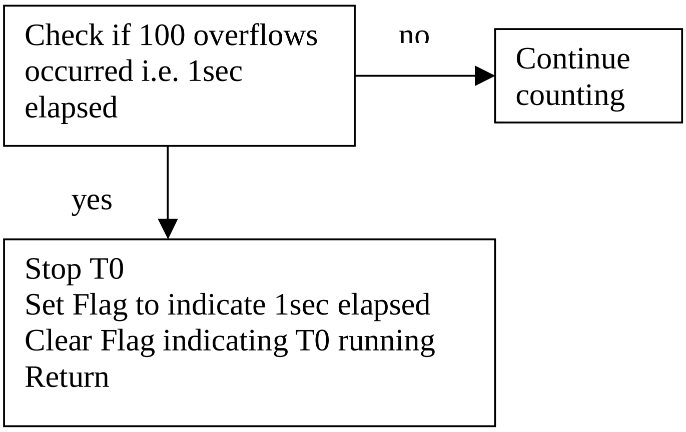
This c code example contains 3 files, freq.c, tic.c and display.c Freq.c implements the T0 timebase approach while Tic.c uses the Time Interval Counter to measure the second. Both freq.c and tic.c call the LCD display routines in display.c Also there is a project file for the Raisonance C compiler though most standard 8052 c compilers will be able to compile these files

4.0 Program Flow

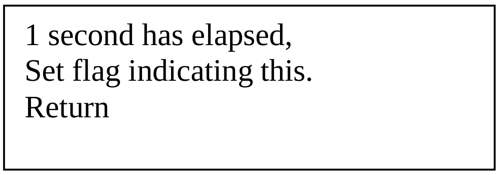
Shown below is a flow diagram for the Frequency measurement routine.



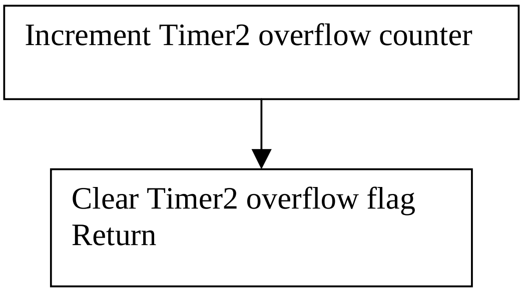
Flow chart for Timer 0 Interrupt Service Routine


Check if 100 overflows occurred i.e. 1sec elapsed
no
Continue counting
yes
Stop T0
Set Flag to indicate 1sec elapsed
Clear Flag indicating T0 running
Return

Time Interval Counter Interrupt Service Routine



Flow Chart Timer 2 Interrupt Service Routine


Increment Timer2 overflow counter
Clear Timer2 overflow flag
Return