Low-cost Real-time Decoding of Broadcast Time and Frequency Standard

Amey Deosthali and Srikanth Gummadi
Department of Electrical and Computer Engineering
The University of Texas, Austin, TX 78712-1084
E-mail: {amey,gummadi}@vision.ece.utexas.edu

March 16, 1998

Contents

Abstract

Radio-controlled time-keeping (called radio clock) has made enormous progress over the past few years. This project implements a low-cost, real-time radio clock for obtaining a secondary time and frequency standard, which can be used for accurate time synchronization and frequency calibration anywhere in the United States. Under normal operation it is possible to achieve an accuracy of 100μ seconds or better. The radio clock is modeled as a heterogeneous system having a mixture of **dataflow** and **finite state machine** domains, and it is implemented in on a low cost microcontroller.

1 Introduction

Precise frequency and global time information is needed by

- electric power companies,
- radio and television stations,
- telephone companies,
- air traffic control systems,
- computer networks,
- scientists monitoring data of all kinds, and
- navigators on ships and planes.

These users need to calibrate their timing to a reliable, and internationally recognized standard. The National Institute of Standards and Technology (NIST) provides this standard for most users in the United States using radio services such as WWV, WWVH and WWVB.

In this project, we focus on the design of a radio-controlled clock based on the decoding of the WWVB signal which provides accurate time synchronization and frequency calibration. The radio station WWVB is located near Ft. Collins, Colorado [1]. The radio-controlled clock automatically sets itself to the radio signal that is transmitted by the WWVB station. The clock in the WWVB station is governed by the US atomic clock which is accurate to 1 second in 1.8 million years. Apart from providing accuracy, the radio-controlled clock provides a convenient and automatic way of setting the time shown by clock to the correct time. This includes automatic setting for daylight savings time and

leap year. Thus users no longer need to run from clock to clock for time adjustments after a power-outage.

This survey is organized as follows. Section 2 underlines the main aim of our project. Section 3 describes the time code that is transmitted by radio station WWVB. Design of the radio clock is discussed in Sections 4 and 5. Section 6 highlights the implementation of radio clock on a microcontroller. We conclude this survey and point out future work in Section 7.

2 Aim

As the number of signal processing applications increase so does the need for providing cost-effective solutions. Although current devices exist that provide time and frequency references, they are very expensive. The main aim of this project is to provide a **reliable**, **low-cost**, end-product which can provide an accurate time and frequency reference. Microcontrollers provide effective cost efficient solutions for low bandwidth applications which do not require a lot of processing power. The radio clock is a perfect signal processing example that can be implemented on a microcontroller and thus reduce the cost.

3 WWVB Time Code

WWVB is a radio station located near Ft. Collins, Colorado [1]. WWVB continuously broadcasts time and frequency signals at 60 KHz, primarily for the continental United States. It should be noted that WWVB is not an acronym but a name for the radio station. All the TV and radio stations in the US are named by Federal Communications Commission (FCC) and they start with W or K.

The WWVB time code is synchronized with the 60 KHz carrier and is broadcast continuously at a rate of 1 pulse per second using pulse-width modulation. Each pulse is generated by reducing the carrier power 10 dB at the start of the second, so that the leading edge of every negative-going pulse is on time. Full power is restored either 0.2, 0.5, or 0.8 seconds later to convey either a binary "0", binary "1", or a position marker, respectively.

The WWVB time code is sent in Binary Coded Decimal (BCD) format [2]. The binary

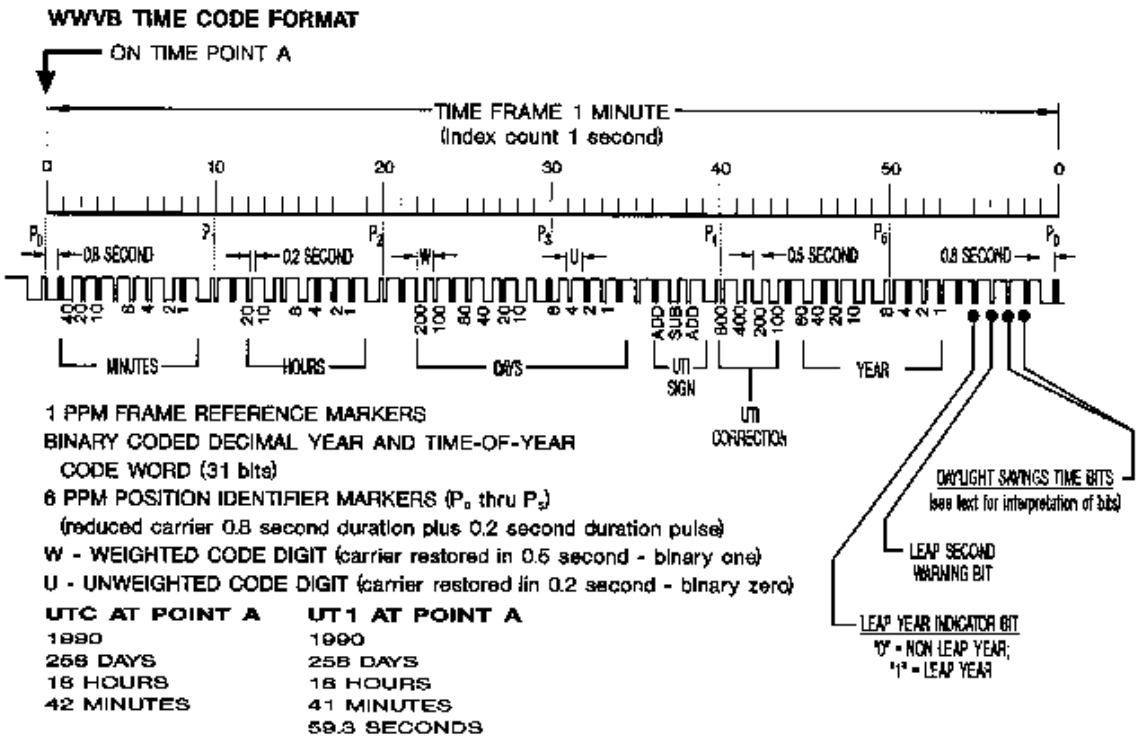
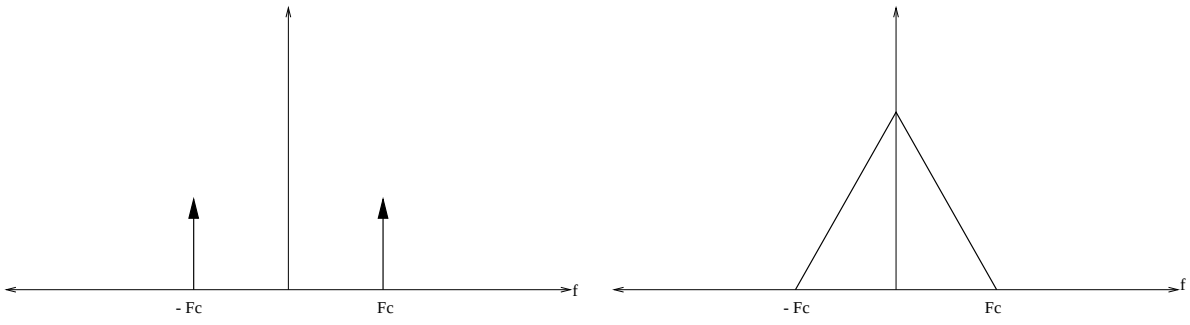


Figure 1: A sample WWVB time code format

to decimal weighting is 8-4-2-1. The most significant bit is sent first. The decimal number is obtained by multiplying each bit in the binary group by its weight and then adding the four products together. For example, the binary group 0101 is equal to 5 which comes from $0 \times 8 + 1 \times 4 + 0 \times 0 + 1 \times 1$.

Every minute, the WWVB time code sends the current minute, hour, day of year, 2 digits of the current year, bit, and Daylight Saving Time (DST) and leap year indicators. Two BCD groups are needed to express the hour (00 to 23), minute (00 to 59), and year (00-99); and three groups are needed to express the day of year (001 to 366). Some bits in the BCD groups are unused, but may provide additional information in the future. To represent units, tens, or hundreds, the basic 8-4-2-1 weights are simply multiplied by 1, 10, or 100 as appropriate. The coded information refers to the time at the start of the one-minute frame. Seconds are determined by counting pulses within the frame. Each minute begins with a frame reference pulse lasting for 0.8 seconds. A position identifier pulse lasting for 0.8 seconds is transmitted every 10 seconds.



(a) (b)

Figure 2: Spectrum of, (a) Sinusoidal signal, (b) Complex signal

4 Decoding the Broadcast Timer

The decoder has two main parts, a signal processing front-end, and a decision logic back-end. The signal processing front-end is modeled in the Synchronous Dataflow (SDF) domain. It has two main components, a sampler, which samples the continuous-time analog signal to convert it into a discrete-time digital signal, and a power estimator, which estimates the power of the received signal.

The decision logic back-end is modeled in the Finite State Machine (FSM) domain. It makes the decisions on the bits that are supplied by the signal processing front-end. The decision logic back-end is responsible for calculating the correct local time from the transmitted UTC (Universal Coordinated Time).

4.1 Signal Processing Front-end

4.1.1 Sampling of the Received Signal

In this application, the transmitted signal is a pure sinusoidal signal of 60 KHz and not a modulated signal. Figure 2 shows the difference in spectral contents between a complex signal and a pure sinusoidal signal. Sampling a pure sinusoid at a frequency which is lower than the Nyquist rate and following it by a low pass filter would result in another sinusoid but of a different frequency [3], as shown in Figure 3. Since our aim is to track the changes in signal power, frequency of the signal is irrelevant. The sampling frequency is restricted by three factors as follows :

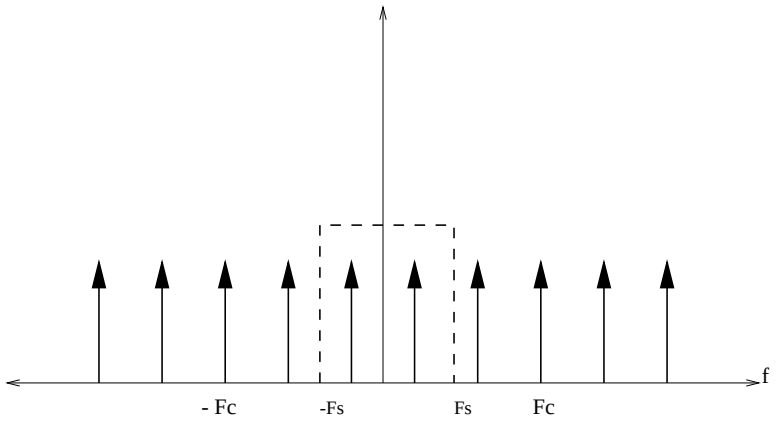


Figure 3: Sampling of a sinusoid signal at a frequency less than Nyquist frequency

- The transmitted frequency must not be a multiple of the sampling frequency. Otherwise the low pass component of the aliased signal would be the DC component. Thus all the information about the signal will be lost.
- The sampling frequency should be low enough that we can perform all the required calculations on the present sample before the next sample comes in.
- The sampling frequency should be as high as possible so that we get more number of samples to estimate the power (hence more accurate).

4.1.2 Estimating Signal Power

Accurate estimation of the power of the received signal is a very important task in the decoding of the WWVB signal. Each pulse is generated by reducing the carrier power by 10 dB at the start of the second and depending on the restoration of power either 0.2, 0.5, or 0.8 seconds later conveys either a binary "0", binary "1", or a position marker, respectively.

The signal power estimator has to be simple, efficient, accurate, and fast. In our implementation we have used a one pole infinite impulse response (IIR) filter to estimate the signal power [4]. The relation between the power $P(n)$ and the received signal $x(n)$ is shown in (1).

$$P(n) \; = \; \alpha \times P(n-1) \; + \; (1-\alpha) \times x(n)^2 \tag{1}$$

where α is the location of the pole of the IIR filter. When α is close to 1, the current estimate

of power depends more on the previous estimate of the power than on the instantaneous value of signal power (its like a weighted average) and is more accurate. Thus, if α is pushed close to the unit circle (typically $\alpha = 0.995$), then the IIR filter gives an accurate estimate of the signal power.

4.2 Decision Logic Back-end

The output of the signal processing front-end is a binary "0" or binary "1". After obtaining the binary bits we have to decode them so as to obtain the correct time. This would include adjusting the day and time depending on the following bits

- daylight savings time
- UT1 correction
- leap year

The displayed time would also be changed depending on which time zone (eastern, central, pacific, mountain) was selected. This decision making logic will be modeled in the Finite State Machine (FSM) domain.

The FSM domain is well suited for the above application because the transition to the next state is dependent on the input received at this state. A simple logic can be implemented easily in SDF without having to use FSM domain, but the logic used for radio clock could not have been implemented since it has about 4 states with many transitions.

5 Generating a Frequency Reference

An accurate frequency reference can be built with the help of a Friedman interpolator [5]. The Friedman interpolator is an algorithm for the estimation of the frequency of a single sinusoid in white noise, based on the computation of the interval between zero crossings. In the following, only the negative to positive going zero-crossings of the sinusoid are considered.

At the arrival of the first positive sample following the zero-crossing, the estimate of the period of the sinusoid $T_e(n)$ is computed as

$$T_e(n) = [K(n) - \delta(n) + \delta(n-1)] \times T_s \tag{2}$$

where $K(n)$ is the number of sampling intervals between the positive samples following the $(n-1)^{th}$ and n^{th} zero-crossings, $\delta(n)T_s$, $\delta(n-1)T_s$ are the time intervals between these zero crossings and the next positive samples.

If $e(n)$ is the error made in the computation of the n^{th} zero crossing, it can be shown that

$$T_e(n) = T_{sin} + e(n) - e(n-1) \tag{3}$$

where T_{sin} is the actual period of the sinewave. The spectrum of $T_e(n)$ contains a DC component equal to the period of the incoming signal and the spectrum of the error signal which is concentrated in the high frequency region. Thus, by using an appropriate low-pass filter, the period and thus frequency can be computed to an arbitrary degree of accuracy.

We plan to modify the above algorithm to estimate the phase information of the received signal. By knowing the phase of the sinusoid we can know the exact time that has elapsed from the previous sample. Thus we will be able to generate an accurate frequency reference.

6 Implementation on a Microcontroller

The above designed radio clock will be implemented on a low-cost microcontroller. We will implement the decoder on a PIC microcontroller from Microchip Technology Inc. We chose PIC16C71 because of its low cost (about $1 in bulk) and its Harvard architecture (similar to the architecture of a DSP) [6].

PIC16C71 is an 18-pin microcontroller with an internal 8-bit analog to digital converter. All of the instructions are single cycle instructions (400 ns with a 10 MHz clock input) except for program branches which are two cycles [6]. It also has an 8-bit timer-counter and a watchdog timer. We will first be developing the application on a simulator and then transferring it to a PIC hardware board.

7 Conclusions and Future Work

Market forecasts suggest that radio-controlled time-keeping will soon be as popular as the ubiquitous quartz watches and clocks. This project is aimed at providing a low-cost solution

for an useful product that is used daily by many of us. The implementation of radio clock is a good example of providing a cost-effective solution by implementing signal processing algorithms on a microcontroller rather than on a DSP.

The decoder will be modeled in Ptolemy using the SDF and FSM domains. After the simulation of the radio clock in Ptolemy, it will be implemented on a PIC microcontroller. The implementation on the PIC microcontroller will be in two phases. In the first phase, the radio clock will be simulated in the PIC microcontroller simulator. In the second phase, the code will be downloaded on a PIC hardware board.

After the implementation of the time standard, we will implement a highly accurate frequency reference. This frequency reference should calibrate itself from the phase of the received WWVB signal. We will use the modified Friedman interpolator to keep the variation of the reference pulse as small as possible.

References

[1] T. E. Parker and J. Levine, "Impact of New High Stability Frequency Standards on the Performance of the NIST AT1 Time Scale," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 44, pp. 1239 – 1244, Nov 1997.

[2] NIST Boulder (Colorado) Laboratories, "http://www.boulder.nist.gov."

[3] P. E. Wellstead, "Aliasing in system identification," *International Journal of Control*, vol. 22, pp. 363 – 375, Sep. 1975.

[4] Z. Dusan, "Mean power estimation with a recursive filter," *IEEE Transactions on Aerospace Electronic Systems*, vol. 13, pp. 281 – 289, May 1977.

[5] V. Friedman, "A zero crossing algorithm for the estimation of the frequency of a single sinusoid in white noise," *IEEE Transactions on Signal Processing*, vol. 42, pp. 1565 – 1569, June 1994.

[6] Microchip Technology Inc., "http://www.microchip.com."