Using LabVIEW to Decode and Synchronize SMPTE Time Code with Data Acquisition

Samuel G. Stone
Electronics Engineer
NIOSH
Morgantown, WV 26505

Walter McKinney
Electronics Engineer
NIOSH
Morgantown, WV 26505

Travis Goldsmith
Electronics Engineer
NIOSH
Morgantown, WV 26505

David G. Frazer
Bioengineer
NIOSH
Morgantown, WV 26505

KEYWORDS

LabVIEW, SMPTE Time Code, Video Synchronization, Longitudinal Time Code

ABSTRACT

In many experiments involving both data and video collection it may be desired to synchronize the collected analog/digital data with the recorded video. This paper demonstrates two LabVIEW Virtual Instruments (VI), which use a single A/D channel to collect and decode SMPTE (Society of Motion Picture Television Engineers) time codes. This paper will demonstrate the proper sample rate selection for bi-phase data collection, as well as the decoding of Binary Coded Decimal (BCD) data. The output of the VI will be frame number, hours, minutes, and seconds in ASCII format, which can be included with the collected analog data. These Virtual Instruments demonstrate a simple, efficient, cost effective method for synchronization of data and video.

INTRODUCTION

In 1971, to simplify video and audiotape editing, the Society of Motion Picture and Television Engineers (SMPTE) developed a time "code" standard [1]. SMPTE 12M-1999 is the current time code standard, and in this paper "time code" will refer to this standard. Today, we can use a time code signal to synchronize video with data acquisition using LabVIEW. A time code's digital encoding applies an address to each video frame. There are 0 to 29 frames of time code data per second, 0 to 59 seconds, 0 to 59 minutes and 0 to 23 hours per day. Hence, there can be up to 30 X 60 X 60 X 24 or 2,592,000 unique addresses in every 24-hour period that must be decoded.

FRAME-RATE FORMATS

We need to clarify the differences between real time and National Television Systems Committee (NTSC) time. As defined by the SMPTE standard, in a system running at a frame rate of 30 frames per second, exactly one second of real time elapses during the scanning of 30 frames. An example of such a system is an 1125/60 high definition system [1]. In an NTSC system running at a vertical field rate of 60/1.001 fields per second (~59.94 Hz), one second of NTSC time elapses during the scanning of 30 frames. Because of the difference in the vertical scanning rates, the relationship between real time and NTSC time is 1 NTSC second = 1.001 real time seconds. To determine if a video system uses NTSC time, check the vertical refresh rate in the owner's manual. If the refresh rate is 59.94 Hz, then chances are that system uses NTSC time.

For video systems using NTSC time, this works out to be 29.97 frames of video per second. This equates to an error of approximately +108 frames (3.6 seconds of real time) over a one-hour period. To correct for this a method called the drop frame mode is used. In the drop frame mode, frame numbers 00 and 01 are dropped from the time code signal at the start of each minute except minutes 00, 10, 20, 30, 40, and 50. This brings the error after one hour down to 3.6ms. If high accuracy resolution is not a requirement, then the video system may be operated in the non-drop frame mode.

MODULATION of SMPTE LTC

SMPTE time code has two formats: Linear or Longitudinal Time Code (LTC) and Vertical Interval Time Code (VITC). This paper focuses on the LTC format, which is used much more extensively. Table 1 shows the format of the LTC word. An LTC word consists of eighty bits in Binary Coded Decimal (BCD) format with the least significant bit first. The eighty bit word is subdivided into 4 bits for the frame units, 2 bits for the frame tens, 4 bits for the seconds unit, 3 bits for the seconds tens, 4 bits for the minute units, 3 bits for the minute tens, 4 bits for the hours unit, 2 bits for the hours tens, and thirty-two bits for the control word. Most of the remaining bits are user bits. User bits can be thought of as spare bits. There are a total of 32 user bits grouped as 8 sections, each 4 bits wide. These spare data bits may be used to contain a secondary time value or some additional information, such as the date. Regardless of whether you are receiving time code at the rate of 29.97 seconds (NTSC) or at 30 seconds (real time), the only difference between the frames is the frame number. So, frame 00 carries the exact same time information as frame 29. In fact, if the user is able to decode one frame of time code, it is possible for the user to interpolate the remaining frames, since the time spacing between all thirty frames are known. Of course, things like the drop frame mode would have to be taken into account. At a minimum, the user should periodically resynchronize with the time code signal.

A bit is a binary digit with a value of either 0 or 1. However, SMPTE LTC represents 0's and 1's not in the convential way, but by using Bi-Phase modulation. Figure 1 shows a section of a LTC signal. In Bi-Phase modulation the binary 0 is represented by a "single" transition at the start of the bit period where a bit period is defined as 417.1 μs in duration at the 30 frames per second rate. A binary 1 is represented by a transition at the beginning of the bit period and another transition in the middle of the bit period. This makes for a very robust encoding method, as the noise immunity is quite high.

Table 1

Time Address	BCD Weight	Bit #	User & Control Bits
Frame Units	1	0	
	2	1	
	4	2	
	8	3	
		4	User # 1
		5	
		6	
		7	
Frame Tens	1	8	
	2	9	
		10	Drop Frame Flag
		11	Color Frame Bit
		12	User # 2
		13	
		14	
		15	
Seconds Units	1	16	
	2	17	
	4	18	
	8	19	
		20	User # 3
		21	
		22	
		23	
Seconds Tens	1	24	
	2	25	
	4	26	
		27	Unassigned
		28	User # 4
		29	
		30	
		31	
Minute Units	1	32	
	2	33	
	4	34	
	8	35	
		36	User # 5
		37	
		38	
		39	
Minute Tens	1	40	
	2	41	
	4	42	
		43	Unassigned Bit
		44	User # 6
		45	
		46	
		47	
Hours Unit	1	48	
	2	49	
	4	50	
	8	51	
		52	Users # 7
		53	
		54	
		55	
Hours Tens	1	56	
	2	57	
		58	Unassigned Bits
		59	
		60	Users # 8
		61	
		62	
		63	
		64	Control Word
		65	
		66	
		67	
		68	
		69	
		70	
		71	
		72	
		73	
		74	
		75	
		76	
		77	
		78	
		79	

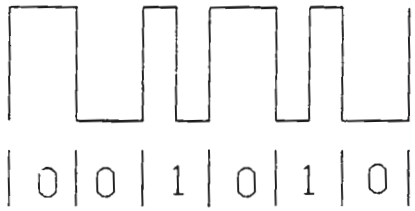

Figure 1

While the bi-phase modulation provides good noise immunity, it does make decoding the time code signal rather difficult, as the program needs to detect transition within the bit period and not just the logic levels. Shannon's sampling theorem states that we must sample the signal at least twice the frequency of interest [2]. The SMPTE LTC time code signal has a frequency range of ~1200 Hz (all zeros) to ~2400 Hz (all ones) so in theory, the minimum sample rate we can use with LabVIEW is then 4800 Hz. The problem with sampling at 4800 Hz is what happens if the sample occurs right on a transition edge? Samples that occur on the transition can be very difficult to decode in LabVIEW, as you will not know if you are on the rising or trailing edge of the signal. If decoding every frame of time code data is not a requirement then the Simple Time Code VI described below is one solution to this problem.

A SIMPLE TIME CODE VI

Figure 2 shows the front panel controls of a simple time code VI that will decode SMPTE LTC time code. The key to this VI is that you pick the number of frames per second that you want to decode. If you are able to decode four to six frames a second, this may be all that is required for a low-resolution synchronization of video with the collected analog data. As stated before, the only difference between the different frames per second is the frame number. So if you know when four to six frames occurred in time, then you can interpolate for the remaining frames if needed. This is easier to do if you operate in the non-drop frame mode. This VI lets the user specify the number of frames to collect each second, and the sample rate for the VI is the minimum frequency of 4800 Hz. If the sample happens to occur on a transition edge, the VI checks for a valid control word; if the control word is not valid, it skips that frame and moves on to the next set of data until a valid control word is found. The front panel controls in Figure 2 show what five frames of collected analog time code looks like. Notice toward the end of the graph, in this instance, the A/D just happens to sample on a transition edge.

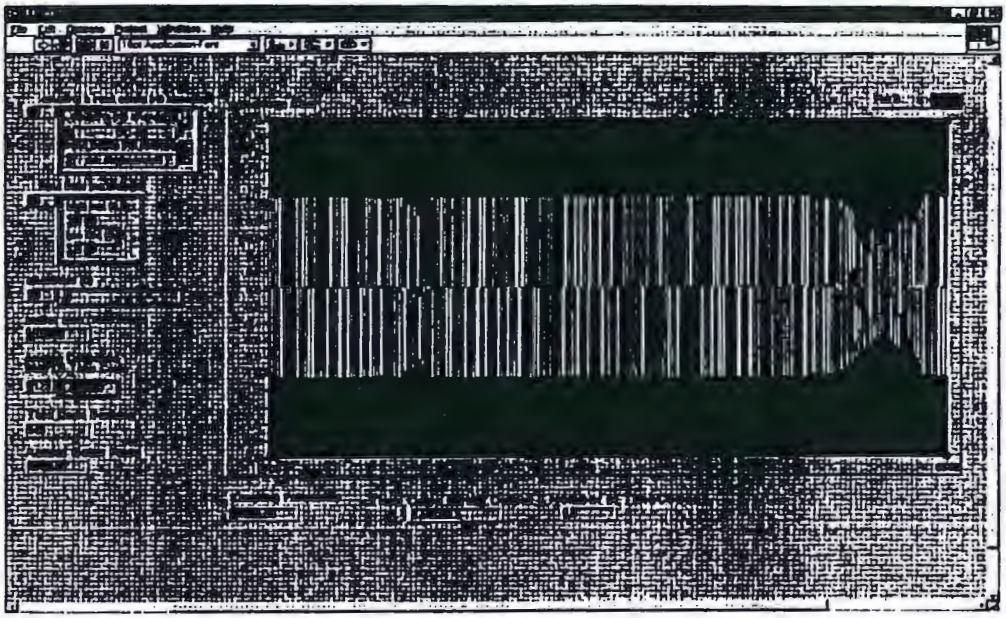

Figure 2

Figure 3 shows the wiring diagram for the simple time code VI. The READ subVI passes 190 bits of time code data to the FIND INDEX subVI. The reason for the 190 bits is that when the data collection starts, it can be anywhere in the time code data. The worst case would be if the data acquisition started one bit into the Control Word. Then the remaining 15 bits of that control word plus the next 80 bits (the next time code frame) would have to be collected to ensure that a valid index point had been found. These 95 bits have to be multiplied by two, because of the Bi-Phase modulation. Passing 190 bits of continuous data to the FIND INDEX subVI ensures that at a minimum, one frame of time code data has been collected.

The front panel for the FIND INDEX subVI is shown in Figure 4. Like its name, the FIND INDEX subVI sorts though the 190 bits of data to find the end of the Control Word. To do this the FIND INDEX subVI first converts the analog collected data to +1 and –1 value, and then it uses the cross correlation subVI to find the control word [3]. The Cross correlation subVI takes the 32 bits that make up the control word and then does a bit by bit correlation with the 190 bits that were passed from the READ subVI. If the maximum value of the cross correlation is 32 then a control word was found within the 190 bits and the FIND INDEX subVI provides the index value indicating the start of the control word. Because of the Bi-Phase modulation, there are actually two control word formats (one will be the complement of the other) because the control word can start on either a rising transition or a falling transition due to the bi-phase modulation. Thus the maximum and minimum values of the cross correlation is compared to the value 32. If a value of 32 is not found then the next 190 bits are read.

The ANALOG to LTC subVI converts the collected analog Bi-Phase data to a BCD format. The BCD to STRING subVI then indexes the LTC data and converts from BCD to an ASCII string output.

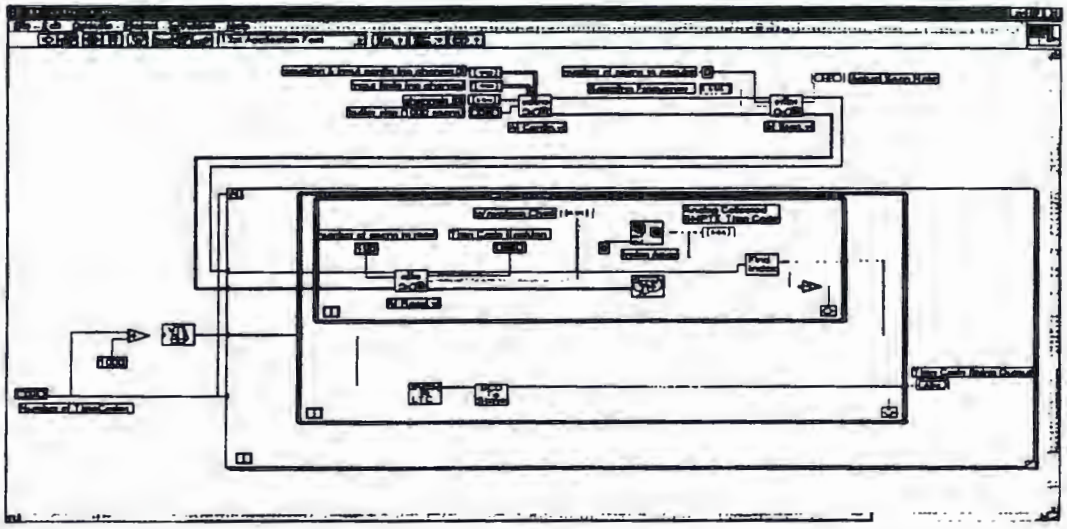

Figure 3

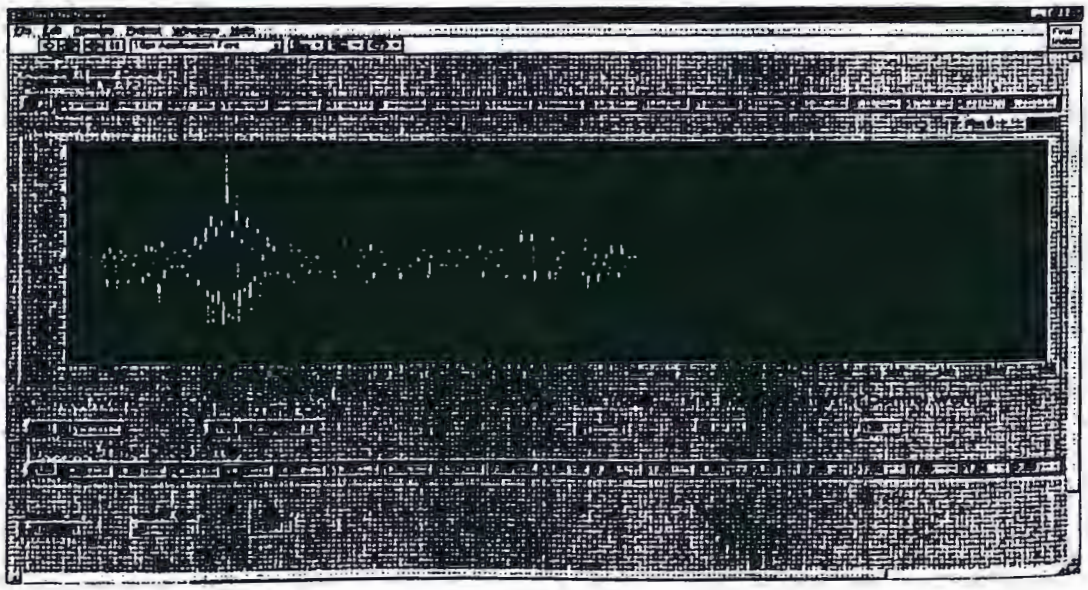

Figure 4

A MORE COMPLETE TIME CODE VI

The simple time code Virtual Instrument is adequate assuming it doesn't sample on a transition. The VI shown in Figure 5 solves this problem. It employs a six times over sampling technique and captures every frame of the time code data stream. The SMPTE time code standard mandates that one bit period be equal to 417.1 μs. This works out to be 2397.5 Hz for 30 frames of time code. For six times over sampling, the scan rate is 14385 Hz.

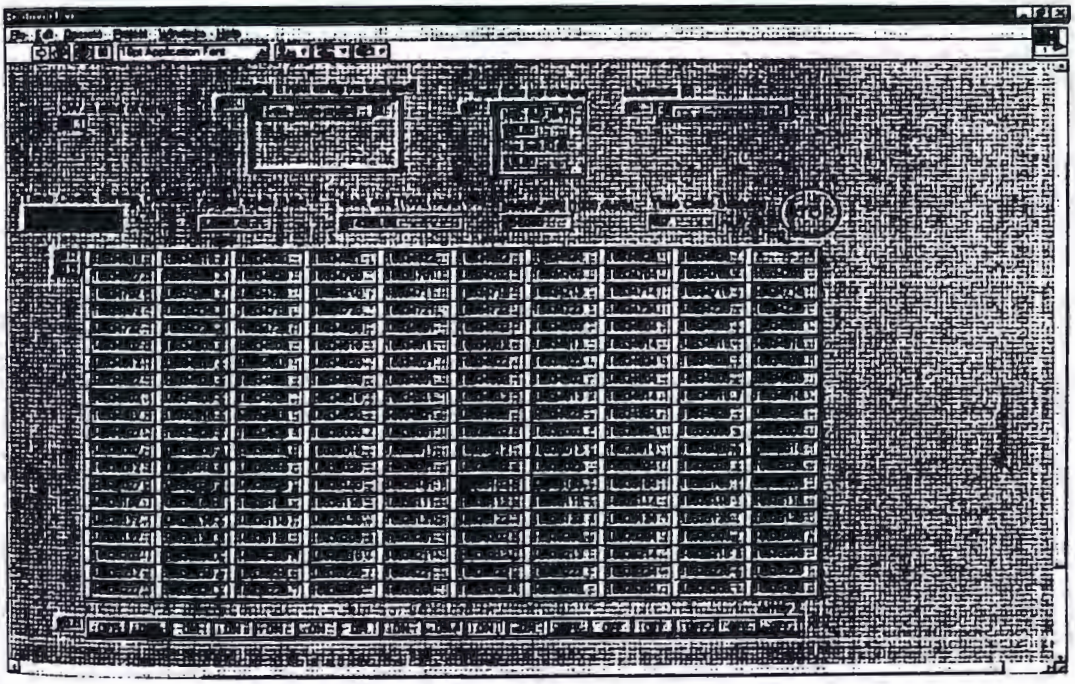

Figure 5

Figure 6 shows the wiring diagram for the complete time code VI. The READ subVI passes ten frames of time code data (4795 points) at a time to the ANALOG to LOGIC subVI. This VI converts the analog voltage values above zero volts to logic one and all other data to a logic zero. The DV CONV SubVI converts the bi-phase modulated time code signal to an eighty-bit BCD string. This is where the six times over sampling is used. To distinguish between a time code one or zero, the DV CONV SubVI checks the logic levels of consecutive bits. If it finds five bits or more in a row that are the same logic value, it outputs a logic zero. The logic is that a bi-phase zero does not have any transition in the middle of the bit period, so five or more consecutive logic levels indicate a logic zero.

The Strip Data subVI sorts though the string of data until it finds the control word. Because the DV CONV subVI converted the Bi-Phase data to logic one's and zero's, there is only one control word to compare. It then strips off the control word and passes the data to the BCD to String subVI which then

indexes the data and converts from BCD to ASCII string format. The primary disadvantage of using the complete time code subVI is the sample rate. Since, LabVIEW does not allow different sample rates for different channels, all channels must be sampled at the rate of 14385 Hz. Without some kind of averaging built into the analog data collection, this will result in enormous data files.

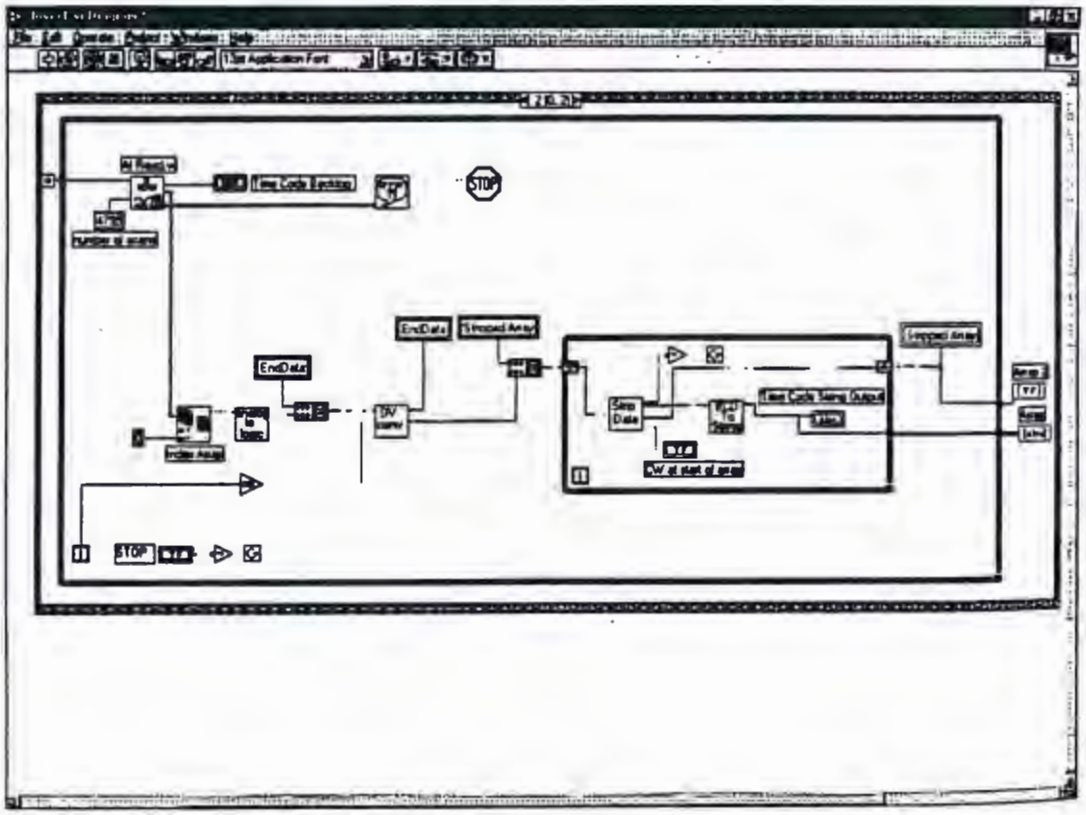

Figure 6

CONCLUSION

This paper has shown two LabVIEW subVI's that can be used to synchronize video data with collected analog data. The principal advantage to using these subVI's is that they do not require any external hardware as both subVI's can be implemented using only a spare A/D channel. The weakness of each subVI has been explored as well as the applications of each subVI. The format and modulation of a SMPTE LTC time code signal has been discussed. In addition, this paper has shown the proper way to determine the sample rate for each subVI.

REFERENCES

1. SMPTE Standard, SMPTE 12M-1999 for Television, Audio and Film - Time and Control Code
2. Proakis J., Manolakis D., Digital Signal Processing – Principles, Algorithms, and Applications, 3rd Ed., pp. 269-279
3. Proakis J., Manolakis D., Digital Signal Processing – Principles, Algorithms, and Applications, 3rd Ed., pp. 120-134

Technical Papers of ISA

Proceedings of the 46th International Instrumentation Symposium

Presented at:
Doubletree Hotel
Bellevue, Washington
30 April – 5 May 2000

AEROSPACE
20 00
46
TEST MEASUREMENT

ISA Volume 397

Sponsored by
Aerospace Industries Division
& Test Measurement Division of ISA

ISA is the international society for measurement and control®