

Project Number: LM1717
Project Title: Timecode Generator
Abstract
Microcontroller LM3S811

A Timecode Generator

Introduction:

This project creates a Timecode signal generator supplying the audio signal for syncing recording machines as well MIDI Timecode used by a lot of MIDI equipment to perform the audio task.

Several VTRs together performing live recordings of concerts, talk shows, musicals needs to be synchronized.

All these recorders are locked by the video signal itself running at the same speed storing different views of the show.

But when the editing team needs to mix these images from different recorders they need to be sure that all recorders run at same speed and also they need to know the time elapsed from show beginning.

In 1967, The US Society of Motion Picture and Television Engineers introduced SMPTE time code. The audio sync tone version of SMPTE is called linear or longitudinal time code or LTC. There is a MIDI version called MTC.

This project generates an audio signal that conforms to the standard, generates MTC too and is capable of being synchronized with external reference video.

In addition it can display the running Timecode and permits setting different modes to suit to several working schemas and could work in NTSC, PAL or SECAM as well film.

This project only uses 4 standard IC, led displays, transistors and can be ported easily to other members of the Stellaris family.

Source Code Sample:

```

// -----
// ----- BUILD 80 BIT SEQUENCE -----
// -----
void
BuildData(void)
{
  unsigned char uc_Byte;
  unsigned char uc_Bit;
  unsigned char uc_Par;

  // ----- MIX TC + UB -----
  for(uc_Byte = 0 ; uc_Byte < 8 ; uc_Byte++)
  {
    g_ucTC[uc_Byte] = ((g_ucTC[uc_Byte + 10]&15)<< 4) | (g_ucTC[uc_Byte +
18]&15);
  }

  // ----- FLAGS ADDER -----
  g_ucTC[1] |= (HWREGBITB(&g_ucKey[0], DF) << 2); // bit 10: drop frame
  g_ucTC[1] |= (HWREGBITB(&g_ucKey[0], CF) << 3); // bit 11: color frame
  g_ucTC[3] |= (HWREGBITB(&g_ucKey[0], BGB1) << 3); // bit 27: binary group
  g_ucTC[5] |= (HWREGBITB(&g_ucKey[0], BGB2) << 3); // bit 43: binary group

  // ----- 'ONE' COUNTER -----
  uc_Par = 0;
  for (uc_Byte = 0 ; uc_Byte < 10 ; uc_Byte++)
  {
    for (uc_Bit = 0 ; uc_Bit < 8 ; uc_Bit++)
    {
      uc_Par += HWREGBITB(&g_ucTC[uc_Byte], uc_Bit);
    }
  }

  // ----- PARITY GENERATOR -----
  if(uc_Par & 1)
  {
    g_ucTC[7] |= 8; // Bi-phase mark-correction bit
  }
}

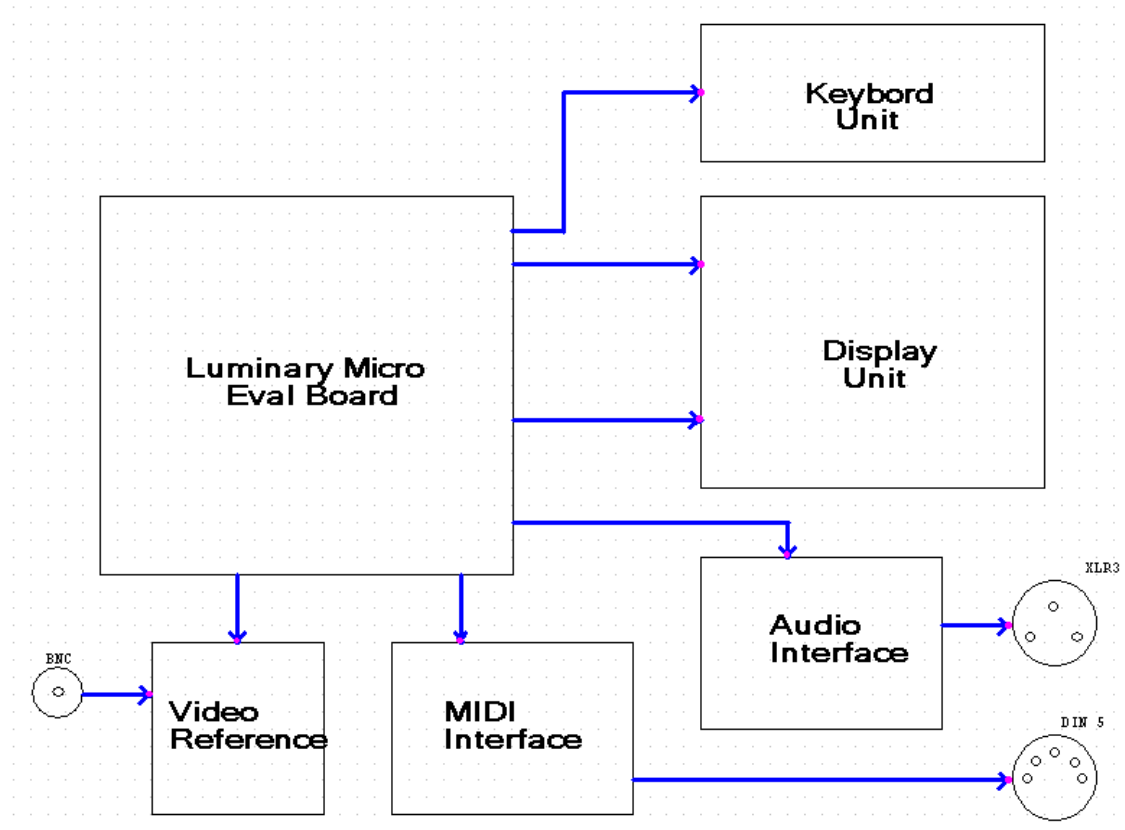
// -----
// ----- FLAGS & FRAME RATE SET -----
// -----
void
Rate(void)
{
  switch (g_ucKey[0] & 6)
  {
    case 0: // 24 Frame
      g_ucRate = 23;
      g_ulCount = 5208; // Value @ 20 MHz
      (HWREGBITB(&g_ucKey[0], DF) = 0);
  }
}

```

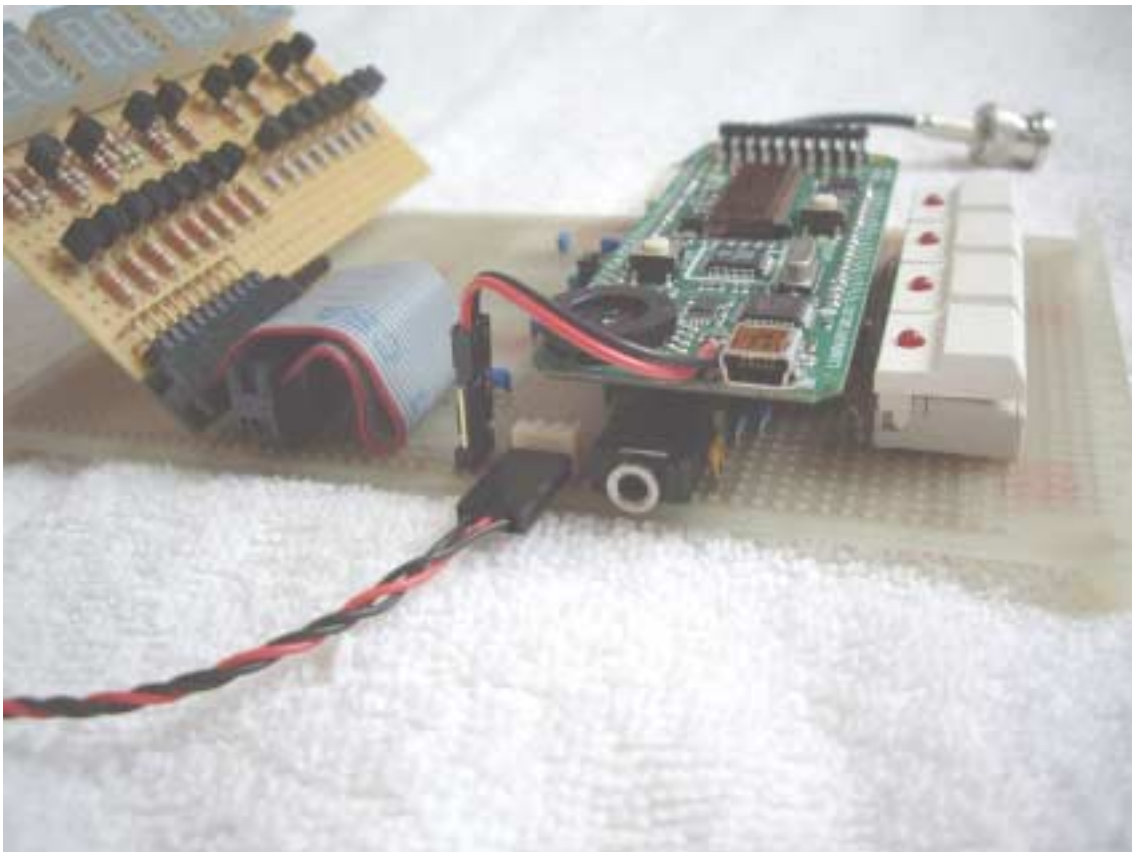
```
        g_ucTC[26]=32;
        g_ucTC[27]=32;
        g_ucTC[28]=64+4;
        g_ucTC[29]=64+2;
        break;
    case 2: // 25 Frame
        g_ucRate = 24;
        g_ulCount = 5000; // Value @ 20 MHz
        (HWREGBITB(&g_ucKey[0], DF) = 0);
        g_ucTC[26]=32;
        g_ucTC[27]=32;
        g_ucTC[28]=64+5;
        g_ucTC[29]=64+2;

        break;
    case 4: // 29.97 Frame
        g_ucRate = 29;
        g_ulCount = 4171; // Value @ 20 MHz
        (HWREGBITB(&g_ucKey[0], DF) = 1);
        g_ucTC[26]=64+15;
        g_ucTC[27]=64+13;
        g_ucTC[28]=64+0;
        g_ucTC[29]=64+3;
        break;
    case 6: // 30 Frame
        g_ucRate = 29;
        g_ulCount = 4167; // Value @ 20 MHz
        (HWREGBITB(&g_ucKey[0], DF) = 0);
        g_ucTC[26]=32;
        g_ucTC[27]=32;
        g_ucTC[28]=64+0;
        g_ucTC[29]=64+3;
        break;
    }
    SysTickPeriodSet(g_ulCount);
}
```

Block Diagram:



Complete Project View:



Schematic Diagrams:

