RAMAIAH
Institute of Technology

Ramaiah Institute of Technology

MICROCONTROLLER (EC62)

HDB3 Line Coding

Team members:

1. Sinchana.S.R 1MS17EC106
2. Sudhanva.S 1MS17EC112
3. Sanjana.P.Chandankeri 1MS17EC139

ABSTRACT:

The HDB3 (High-Density Bipolar Order 3) code relies on the transmission of both positive and negative pulses and is the encoding technique used over G.703 E1 networks. With this technique, both timing and data can be transmitted over just two wires in each direction. In our mini project we have implemented HDB3 line coding technique in MSP430 by using Assembly Level Programming (ALP) and simulated the same using Proteus 8.8.

INTRODUCTION:

HDB3 is a development of AMI (Alternate Mark Inversion), a line code in which a logical 0 is represented by no change and a logical 1 is represented by pulses of alternating polarity. HDB3 prevents more than four AMI "no change" bits from being sent consecutively. This, in turn, prevents long runs of zeros in the data stream. Without HDB3, the receiving PLL (phase lock loop) circuit would have difficulty maintaining synchronization.

The **high density bipolar of order 3 (HDB3)** code replaces any instance of 4 consecutive 0 bits with one of the patterns "000V" or "B00V". The choice is made to ensure that consecutive violations are of differing polarity; i.e., separated by an odd number of normal + or − marks.

HDB 3 coding of 0000_2			
Parity of +/− bits since previous V	**Pattern**	**Previous pulse**	**Coded**
Even	B00V	+	−00−
		−	+00+
Odd	000V	+	000+
		−	000-

These rules are applied on the code as it is being built from the original string. Every time there are 4 consecutive zeros in the code they will be replaced by either 000−, 000+, +00+ or −00−. To determine which pattern to use, one must count the number of pluses (+) and the number of minuses (−) since the last violation bit V, then subtract one from the other. If the result is an odd number then 000− or 000+ is used. If the result is an even number then +00+ or −00− is used. To determine which polarity to use, one must look at the pulse preceding the four zeros. If 000V form must be used then V simply copies the polarity of last pulse, if B00V form must be used then B and V chosen will have the opposite polarity of the last pulse.

COMPONENTS USED:

3 LEDs(Green,Red,Yellow), MSP430F1111, SW-SPDT switch, 4V DC Power supply

CODE:

```
;R5 is where input signal is stored

;R6 HAS A COPY OF R5

;R7 IS USED TO TELL IF HDB3 IS NEEDED

;R9 CONTAINS THE HDB3 MODE

;R10 IS A COUNTER OF 4

;R13 SAYS IF R14 IS ODD OR EVEN

;R14 has a count of number of ones before current HDB3

;R15 has toggling control for +a and -a levels


; R15=1  -> -a

; R15=2  -> +a


; RED IS +A ---- BLUE IS -A ---- YELLOW IS 0

; P1OUT.1        P1OUT.2        P1OUT.3

;   0X02          0X04          0X08


#include "msp430.h"

LED1 EQU BIT1 ;Red LED ; BIT1 -> 0X0002u P1.1

LED2 EQU BIT2 ;Blue LED ; BIT2 -> 0X0004u P1.2

LED3 EQU BIT3 ;Yellow LED ; BIT3 -> 0X0008u P1.3


      NAME    main                  ; module name
```

```
        PUBLIC  main                ; make the main label visible
                                    ; outside this module
        ORG     0FFFEh
        DC16    init                ; set reset vector to 'init' label

        RSEG    CSTACK              ; pre-declaration of segment
        RSEG    CODE                ; place program in 'CODE' segment

init:   MOV     #SFE(CSTACK), SP    ; set up stack

main:   NOP                         ; main program
        MOV.W   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog timer
        MOV #0X01, R15 ; Previous level is at -a
        CLR R5
        CLR R6
        CLR R7
        CLR R8
        CLR R9
        CLR R10
        CLR R14

        MOV.B #0XFE, &P1DIR  ; Bit-1 of P1 is input, rest all are outputs. BIT-0 IS NOT USED
```

```
MAIN1:  BIT.B  #0x01, P1IN
        JZ INL ; Input is 0
        INC R14
        SETC ; Input is 1
        JMP INR

INL:    CLRC
INR:    RLC.B R5
        MOV R5, R6
        AND.B #0X0F, R6
        JZ HDB3C0
        BIT.B #0X08, R5
        JNZ BPH
        ;SIMPLY BIT IS ZERO
        MOV.B #LED3, &P1OUT
        JMP MAIN1

BPH:    CMP.B #0X01, R15; CHECKING IF THE LAST ONE WAS -A
        JZ BPP

BPN:    MOV.B #LED2, &P1OUT ;LAST 1 WAS +A, SO MOVING -A INTO OUTPUT
        MOV #0X01, R15
        JMP MAIN1

BPP:    MOV.B #LED1, &P1OUT ;LAST 1 WAS -A, SO MOVING +A INTO OUTPUT
```

```
        MOV.B #0X02, R15


        JMP MAIN1
HDB3C0:  BIT.B #0X08, R5
        JZ HDB3C1


        CMP #0X01, R15
        JZ ML
        MOV.B #0X01, R15
ML:     MOV.B #0X02, R15  ;ENSURING LAST BIT DOES NOT CHANGE DUE TO
ROLLING


HDB3C1   CMP #0X00, R7
        JZ HDB3


HDB3C2:  CMP #0X00, R9
        JZ MAIN1


        CMP #0X01, R9
        JZ CC1


        CMP #0X02, R9
        JZ CC2


        CMP #0X03, R9
```

```
        JZ CC3

        CMP #0X04, R9
        JZ CC4


CC1:    CMP #0X03, R10
        JZ EP2

        CMP #0X02, R10
        JZ EP3

        CMP #0X01, R10
        JZ EP4


CC2:    CMP #0X03, R10
        JZ EN2

        CMP #0X02, R10
        JZ EN3

        CMP #0X01, R10
        JZ EN4
```

```
CC3:    CMP #0X03, R10
        JZ OP2

        CMP #0X02, R10
        JZ OP3

        CMP #0X01, R10
        JZ OP4

CC4:    CMP #0X03, R10
        JZ ON2

        CMP #0X02, R10
        JZ ON3

        CMP #0X01, R10
        JZ ON4

HDB3:  MOV #0X01, R7 ; BLOCKING INCOMING HDB3 ENCODING
                ; FOR FURTHER 3 BITS

      MOV #0X04, R10 ; COUNTER LOOP OF 4 1 SEC CLOCKS

      MOV R14, R13
```

```
        AND.B #0X01, R13

        JNZ LODD

        CMP #0X01, R15
        JZ LEN
; //-------------------------------------------------------
        ;EVEN AND LAST LEV WAS +a
        ;PATTERN -00-

        MOV #0X01, R9
;------------------------------
EP1:   MOV.B #LED2, &P1OUT

        DEC R10
        CMP #0X00, R10
        JNZ MAIN1
;------------------------------

EP2:   MOV.B #LED3, &P1OUT

        DEC R10
        CMP #0X00, R10
        JNZ MAIN1
;------------------------------
```

```
EP3:   MOV.B #LED3, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

EP4:   MOV.B #LED2, &P1OUT
       MOV.B #0X01, R15
       MOV.B #0X09, R6

      CLR R7
      CLR R14
      CLR R9

      JMP MAIN1

;-------------------------------------------------------

       ;EVEN AND LAST LEV WAS -a
       ;PATTERN +00+

LEN:   MOV #0X02, R9
;------------------------------
```

```
EN1:   MOV.B #LED1, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

EN2:   MOV.B #LED3, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

EN3:   MOV.B #LED3, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

EN4:   MOV.B #LED1, &P1OUT
       MOV.B #0X02, R15
       MOV.B #0X09, R6
```

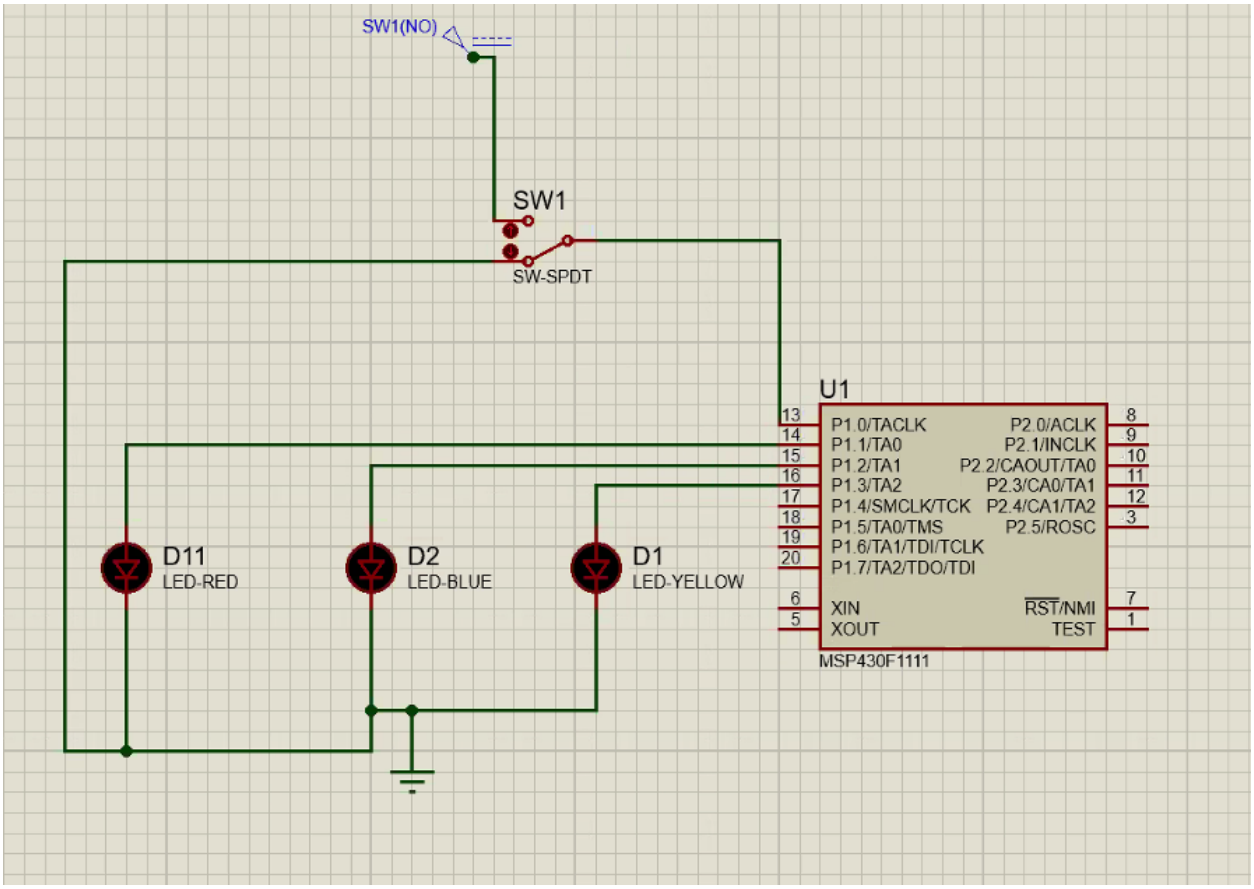
```
    CLR R7
    CLR R14
    CLR R9

    JMP MAIN1

;-------------------------------------------------------



LODD:  CMP #0X01, R15
    JZ LON
; //-------------------------------------------------------
    ;ODD AND LAST LEV WAS +a
    ;PATTERN 000+

    MOV #0X03, R9
;------------------------------
OP1:   MOV.B #LED3, &P1OUT

    DEC R10
    CMP #0X00, R10
    JNZ MAIN1
;------------------------------
```

```
OP2:   MOV.B #LED3, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

OP3:   MOV.B #LED3, &P1OUT

       DEC R10
       CMP #0X00, R10
       JNZ MAIN1
;------------------------------

OP4:   MOV.B #LED1, &P1OUT
       MOV #0X02, R15
       MOV.B #0X01, R6

      CLR R7
      CLR R14
      CLR R9

      JMP MAIN1

;-------------------------------------------------------
```

```
;ODD AND LAST LEV WAS -a
      ;PATTERN 000-

LON:   MOV #0X04, R9
;------------------------------
ON1:   MOV.B #LED3, &P1OUT

      DEC R10
      CMP #0X00, R10
      JNZ MAIN1
;------------------------------

ON2:   MOV.B #LED3, &P1OUT

      DEC R10
      CMP #0X00, R10
      JNZ MAIN1
;------------------------------

ON3:   MOV.B #LED3, &P1OUT

      DEC R10
      CMP #0X00, R10
      JNZ MAIN1
;------------------------------
```

```
ON4:   MOV.B #LED2, &P1OUT
       MOV.B #0X01, R15
       MOV.B #0X01, R6

      CLR R7
      CLR R14
      CLR R9

      JMP MAIN1

;-------------------------------------------------------

       JMP $                    ; jump to current location '$'
                                ; (endless loop)
   END
```

CIRCUIT DIAGRAM:



RESULT:

HDB3 line coder is successfully demonstrated using MSP430F1111 microcontroller. The code was written on IAR Embedded Workbench and the simulation environment is Proteus Professional 8.6