

```

/*****
/*
/*  FILENAME:    manch.c
/*
/*
/*  ABSTRACT:
/*      Manchester serial output transmit routine. This accepts a string
/*      of bytes to send in an array with seperately supplied length.
/*      then transmits the code out serially. The serial string starts
/*      with a sync stream of 1 bits followed by a 3T high 3T low pulse.
/*      Data bits then follow for each byte in the text string. After the
/*      last bit is sent there is a transmitted ending stream of zero
/*      bits. This uses a timing state machine to sequence the code that
/*      is called each 1 msec by the timer interrupt routine.
/*
/*
/*  CHANGE LOG:
/*      DATE      AUTHOR      SDR      DESCRIPTION
/*      01-may-11  M Karas      ---      Original setup for EVB1 with SiLabs
/*                                          C8051F344 processor as timer module.
/*
*****/

#include <string.h>
#include <stdio.h>

#include "c8051f340.h"          /* SFR declarations */
#include "evb1.h"
#include "manch.h"

/*
**
** global data for module
**
*/

unsigned char xdata manch_byte_array[MANCH_MAX_BYTES]; /* array of bytes to send out */
unsigned char xdata manch_byte_index;                  /* index to bytes being sent */
unsigned char xdata manch_byte_count;                  /* count of bytes to send */
unsigned char xdata manch_state;                       /* current state number of the
manchester code machine */
unsigned char xdata manch_cnt;                          /* manchester state machine counter */
unsigned char xdata manch_bits;                        /* bits to be shifted out for a byte */

/*
**
** manchester initialize routine
**
*/

void manch_init(void)
{
    BSET(MANCH_OMODE, MANCH_BIT); /* push pull = 1 */
    BSET(MANCH_IMODE, MANCH_BIT); /* digital = 1 */
    BCLR(MANCH_PORT, MANCH_BIT);  /* force the bit low initially */
}

```

```

    manch_byte_count = 0;
    manch_state = MANCH_STATE_IDLE;
}

/*
**
** routine to setup the manchester byte array
** setup the state machine to start on the new string.
**
*/

void manch_set_bin(unsigned char *bin_array, unsigned char bin_len)
{
    unsigned char index;
    unsigned char bin_byte;

    /* force stop of any current transmit process */
    manch_byte_count = 0;
    manch_state = MANCH_STATE_IDLE;

    /* loop to copy the bytes to be sent into bits array */
    index = 0;
    while(bin_len > 0)
    {
        bin_byte = *bin_array++;          /* get the next byte */
        manch_byte_array[index] = bin_byte;
        bin_len--;
        index++;
        if(index >= MANCH_MAX_BYTES)
        {
            break;
        }
    }
    manch_byte_count = index;              /* force to new string */
}

/*
**
** state machine to sequence out a manchester serial string to
** port bit to drive the transmitter module.
**
** (note that this is a timer interrupt context routine)
**
** this called once each millisecond from the timer ISR
**
*/

void manch_xmit(void) using 1
{
    switch(manch_state)
    {
        /* unrecognized state number - reset things and go to idle state */

```

```

default:
    manch_byte_count = 0;
    manch_byte_index = 0;
    manch_state = MANCH_STATE_IDLE;
    break;

/* idle state - hang till non-null manchester text string pointer to process */
case MANCH_STATE_IDLE:
    if(manch_byte_count != 0)                /* non zero count to start */
    {
        manch_byte_index = 0;                /* start send from beginning of array */
        manch_pin(1);                        /* high for 1st half of '0' */
        manch_cnt = MANCH_LEADIN_COUNT;      /* set the number of leadin pulses to do */
        manch_state = MANCH_LEADIN_L;
    }
    break;

/* leadin state second phase of each zero bit time */
case MANCH_LEADIN_L:
    manch_pin(0);                            /* low for 2nd half of '0' */
    manch_state = MANCH_LEADIN_H;
    break;

/* leadin state with sequence of zeros */
case MANCH_LEADIN_H:
    manch_cnt--;                            /* decrement the leadin bit count */
    if(manch_cnt == 0)
    {
        /* start the sync high period */
        manch_cnt = MANCH_TIMING_START;      /* sync width in T stats */
        manch_pin(1);                        /* start the high period of the sync pulse */
        manch_state = MANCH_STATE_SYNC_H;
        break;
    }
    /* launch next leadin pulse */
    manch_pin(1);                            /* high for 1st half of '0' */
    manch_state = MANCH_LEADIN_L;
    break;

/* state to process sync pulse high */
case MANCH_STATE_SYNC_H:
    manch_cnt--;                            /* decrement the sync high T count */
    if(manch_cnt == 0)
    {
        manch_cnt = MANCH_TIMING_START;      /* sync width in T stats */
        manch_pin(0);                        /* start the low period of the sync pulse */
        manch_state = MANCH_STATE_SYNC_L;
    }
    break;

/* state to process sync pulse low */
case MANCH_STATE_SYNC_L:
    manch_cnt--;                            /* decrement the sync low T count */

```

```
    if(manch_cnt == 0)
    {
        manch_pin(1);                /* setup the 1st half high of the fake zero
        */
        manch_state = MANCH_STATE_FAKE0;
    }
    break;

/* state for the 2nd half of the fake zero */
case MANCH_STATE_FAKE0:
    manch_pin(0);                    /* setup the 2nd half low of the fake zero */
    manch_state = MANCH_STATE_NEXT_BYTE;
    break;

/* state for next character in the string */
case MANCH_STATE_NEXT_BYTE:
    if(manch_byte_index >= manch_byte_count)    /* check for byte array done */
    {
        manch_pin(1);                /* high for 1st half of '0' */
        manch_cnt = MANCH_LEADOUT_COUNT;    /* set the number of leadout pulses to do */
        manch_state = MANCH_LEADOUT_L;
    }
    manch_bits = manch_byte_array[manch_byte_index]; /* get next byte to send out */
    manch_byte_index++;                /* increment array index for next time */

    /* setup bits transmit loop (NOTE: We send the bit 7 first */
    manch_cnt = 8;                    /* send the byte as 8 bits */

    /* no break -- just fall through to the NEXT BIT state */

/* state to process the next bit in the curent byte */
case MANCH_STATE_NEXT_BIT:
    if((manch_bits & 0x80) != 0)        /* get ready for a 1 bit */
    {
        /* 1 bits are low in 1st half of bit cell and high in second half */
        manch_pin(0);                /* low for 1st half of '1' */
        manch_state = MANCH_STATE_ONE_BIT;
    }
    else
    {
        /* 0 bits are high in 1st half and low in second half */
        manch_pin(1);                /* high for 1st half of '0' */
        manch_state = MANCH_STATE_ZERO_BIT;
    }
    manch_bits <<= 1;                /* shift bit pattern left */
    manch_cnt--;                    /* decrement the bits count */
    break;

/* state to process 2nd half of the '1' bit */
case MANCH_STATE_ONE_BIT:
    manch_pin(1);                    /* high for 2nd half of '1' */
    if(manch_cnt == 0)                /* done with all the bits ? */
    {
```

```

        manch_state = MANCH_STATE_NEXT_BYTE;
    }
    else
    {
        manch_state = MANCH_STATE_NEXT_BIT;
    }
    break;

/* state to process 2nd half of the '0' bit */
case MANCH_STATE_ZERO_BIT:
    manch_pin(0);                                /* low for 2nd half of '0' */
    if(manch_cnt == 0)                            /* done with all the bits ? */
    {
        manch_state = MANCH_STATE_NEXT_BYTE;
    }
    else
    {
        manch_state = MANCH_STATE_NEXT_BIT;
    }
    break;

/* leadout state second phase of each zero bit time */
case MANCH_LEADOUT_L:
    manch_pin(0);                                /* low for 2nd half of '0' */
    manch_state = MANCH_LEADOUT_H;
    break;

/* leadout state with sequence of zeros */
case MANCH_LEADOUT_H:
    manch_cnt--;                                /* decrement the leadout bit count */
    if(manch_cnt == 0)
    {
        /* return to idle state now that done */
        manch_pin(0);                            /* idle with pin low */
        manch_byte_count = 0;                    /* force idle wait till next byte count */
        manch_state = MANCH_STATE_IDLE;
        break;
    }
    /* launch next leadout pulse */
    manch_pin(1);                                /* high for 1st half of '0' */
    manch_state = MANCH_LEADOUT_L;
    break;
}
}

/*
**
**
** support routine to turn the manchester transmitter input on or off
**
** This supports the transmitter connected to P2.6 on the EVB extension board
**
** (use only from timer task context)

```

```
**  
*/
```

```
void manch_pin(char onoff) using 1  
{  
    if(onoff == 0)                /* turn PIN off */  
    {  
        (MANCH_PORT &= ~(1 << MANCH_BIT));  
    }  
    else                          /* turn PIN on */  
    {  
        (MANCH_PORT |= (1 << MANCH_BIT));  
    }  
}
```