RMV ELECTRONICS INC. **Application Note:**	**Application #:** 00006
	Date: September 1994
Description: Using the ITC232-A for Quadrature Encoded Counting.	**Status:** Draft version

This circuit describes how to economically decode and count the output of a quadrature (incremental) shaft encoder. The basic principle of quadrature encoding is the use of two output pulse streams 90 degrees out of phase. By properly decoding these signals, the direction and distance traveled of a shaft can be obtained. The circuit is inherently noise immune and highly accurate.

OPERATIONAL DESCRIPTION

Please refer to the schematic diagram and the functional software listing. Also refer to the appropriate manufacturers' data sheets and the ITC232-A user's manual for further detail.

The circuit is implemented using a commercially available counter, the LSI Computer Systems LS7166 MultiMode Counter (MMC). The LS7166 was used because of its built in quadrature decoding capability, maximum input frequency (>1 Mhz), and 24 bit counter. In conjunction with the ITC232-A, a complete quadrature subsystem is easily and economically available.

The interface between these devices and the ITC232-A is through all of Port C, 6 bits of Port B, and both interrupt lines. Port C is used as a bidirectional data bus, while the Port B lines act as address and bus control lines. A sequence of port writes and reads effectively emulates the operation of a microprocessor data bus, which both the PIT and MMC were designed for.

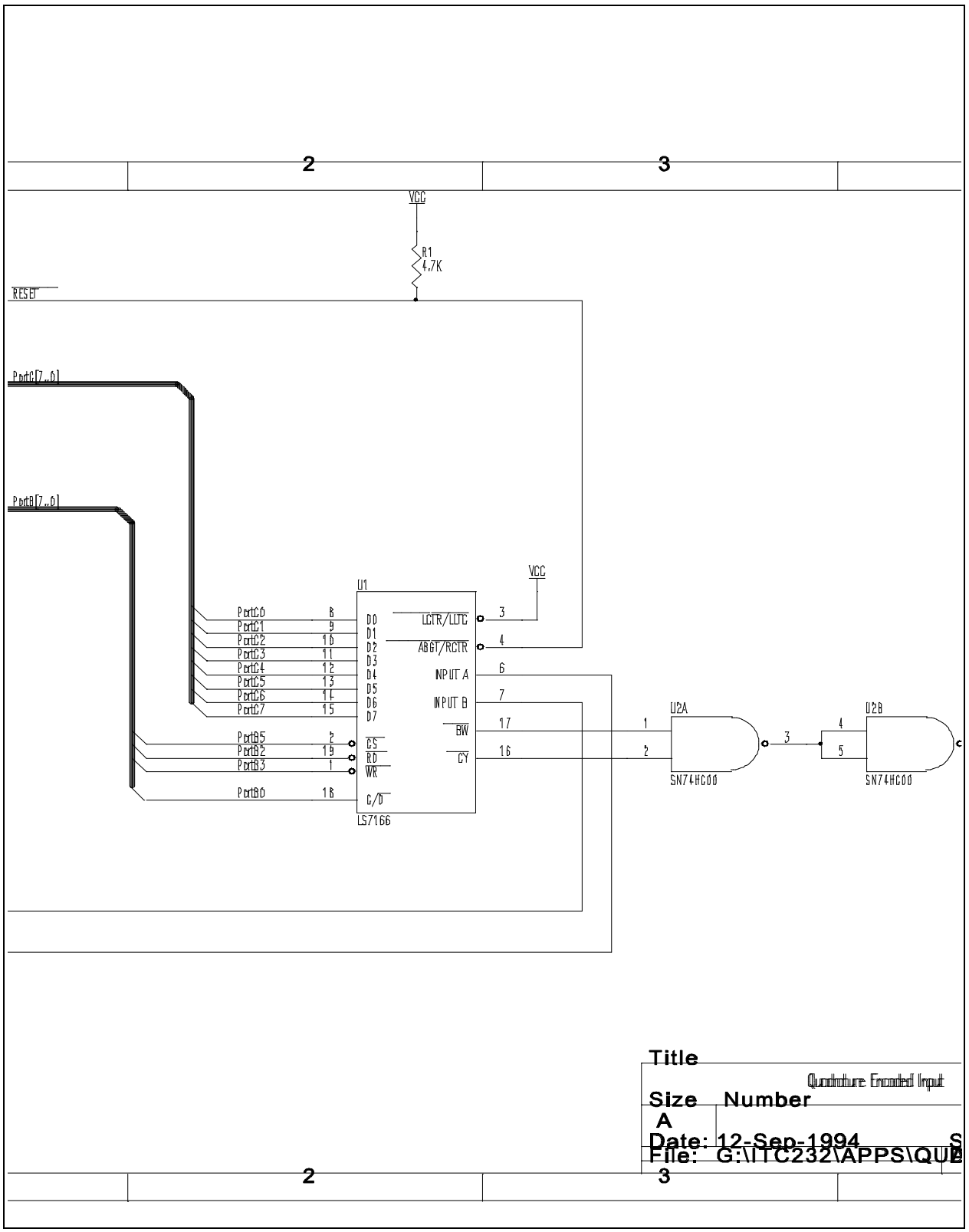
The MMC is initialized in quadrature counter mode. The carry and borrow outputs are combined so that either produces an ITC232-A interrupt. The carry output toggles each time the counter overflows (i.e. counting up over the maximum) while the borrow output toggles each time the counter underflows (i.e. counting down through 0). Alternatively, the interrupt can be programmed to occur whenever the quadrature counter reaches a preset value. The counters can be reset by external hardware simply by grounding the COUNTER RESET line or by software commands through the ITC232-A.

```
/*
Functional code listing for the quadrature counter application
circuit.

ITC232 port assignments are as follows (refer to schematic
diagram)
Port C is a bidirectional 8 bit data bus
Port B, bits 0 and 1 are address lines used to select particular
registers
on the MMC (only bit 0 used for the MMC, bit 1 is free for
expansion)
Port B, bits 2 and 3 are the active low read and write lines
respectively
Port B, bit 4 is not used
Port B, bit 5 is the active low chip select for the MMC
Port B, bit 6 and 7 are not used
IRQL is used to indicate an overflow or underflow condition
/*


/* Constant definitions */
#define MMC_DATA        0xdc
#define MMC_CONTROL     0xdd
char    ITC_data(void); /*function that returns the data value in
response
                                                    to a port read operation */
#define BAUD_9600       12
#define COMM_1          1
#define COMM_2          2
#define COMM_3          3
#define COMM_4          4

#define NULL            0

/* include files */
#include <stdio.h>
#include <math.h>
#include <conio.h>

/*function declarations*/
/* function for sending data thru the ITC232 to peripherals */
void    out_data(unsigned char a, unsigned char b);
/* function for receiving data from ITC232 peripherals */
char    in_data(unsigned char a);
void    ITC_send(char *ptr); /* function to send a string to
ITC232 */
void    ITC_send_no_wait(char *ptr); /* function to send a string
to ITC232 */
```

```
/* external function for setting up serial ports */
extern void far Setup(unsigned int Rate, unsigned char Parms,
unsigned char SelBase);
/* Setup(Divisor, Parms, SelBase) */
/* Divisor = 115200/desired baud rate */
/* Parms   = 0 | a | bcd | e | fg     */
/*          a   = 0 for break */
/*              = 1 for space */
/*          bcd = 000 none parity */
/*              = 001 odd parity  */
/*              = 011 even parity */
/*          e   = 0 for 1 stop bit */
/*              = 1 for 2 stop bits */
/*          fg  = 10 for 7 bits per character */
/*              = 11 for 8 bits per character */
/* SelBase = 1 for COM1 */
/*          2 for COM2 */
/*          3 for COM3 */
/*          4 for COM4 */

/* external functions for installing an comm interrupt, removing
same,
   and restoring previous port operation */
extern void far Restore();
extern void far SetInt();
extern void far RemInt();
extern unsigned int far Receive( unsigned char Rfl, unsigned
char Rvalue);
extern unsigned int far SendChar ( unsigned char Sfl, unsigned
char Tvalue);

main()
{
unsigned char    count_hi, count_mid, count_lo, input;
long    count_total;

/*Initialize the PC serial port as for 9600 baud, no parity, 8 bits,
1 stop*/
Setup(BAUD_9600, 3, COMM_1);
SetInt();

/*Initialize the ITC232 for Configure Results Ascii Program*/
ITC_send("CRAP\r");

/*Set Port B outputs bit 6 thru 0 high*/
ITC_send("PWB$7F\r");

/*Set Port B direction as bits 7 and 5 thru 0 as outputs, bit 6 as
input*/
ITC_send("PCB$BF\r");

/*Reset MMC, then setup for quadrature counting (X1), carry
and borrow
enabled, count reset enabled, binary counting */
/* send master reset */
out_data(MMC_CONTROL, 0x20);
/* set input control register */
out_data(MMC_CONTROL, 0x48);
/* set output control register */
out_data(MMC_CONTROL, 0x80);
/* set quadrature control register */
/*out_data(MMC_CONTROL, 0xc1);*/

/*bring chip select low */
itoa((a & 0x7f), value, 16);
strcpy(string,"PWB$");

printf("\nProcessing...hit any key to exit\n");

while( !kbhit() )
        {
        /* if no keys hit, continuously print count values */
        /* request a transfer of the counts to the output
latches */
        out_data(MMC_CONTROL, 0x03);
        /* start getting count values */
        count_lo = in_data(MMC_DATA);
        count_mid = in_data(MMC_DATA);
        count_hi = in_data(MMC_DATA);
        /* combine the three byte wide counter reads into a
single value */
        count_total = count_hi;
        count_total = (count_total << 8) + count_mid;
        count_total = (count_total << 8) + count_lo;
        printf("\ncounter = %ld", count_total);
        }

/*end of while(kbhit()) loop */

printf("\nQuadrature Input Measurement function halted\n");
RemInt();
Restore();

} /*end of main*/

/****************************************/
/*   out_data                           */
/* sets up 'bus' for a write operation */
/****************************************/
void out_data(a,b)
unsigned char a,b;
{
char string[8], value[3];

/* set string ends to NULLS */
string[7] = value[2] = NULL;

/* set port C to output */
ITC_send("PCC$FF\r");

/*setup data value on Port C*/
strcpy(string,"PWC$");
itoa(b, value, 16); /* convert the number to be sent to an ascii
hex string */
if(b < 0x10)
        strcat(string,"0");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/*setup register address */
itoa((a | 0x7c), value, 16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);



strcat(string,value);
strcat(string,"\r");
ITC_send(string);
```

```
/*bring write low */
itoa((a & 0x77), value, 16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/*bring write high*/
itoa((a & 0x7f),value,16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/*bring chip select high */
itoa((a | 0x7c), value, 16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/* then return to idle state */
ITC_send("PWB$7F\r");
}

/*****************************************/
/*   in_data                             */
/*  sets up 'bus' for a read operation */
/*****************************************/
char in_data(a)
unsigned char a;
{
char string[8], value[3], read_data;

/* set string ends to NULLS */
string[7] = value[2] = NULL;

/* set port C to inputs */
ITC_send("PCC$00\r");

/* setup register address */
itoa((a | 0x7c & 0x7f), value, 16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/* bring chip select low */
itoa((a & 0x7f), value, 16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


}

/*****************************************/
/*   ITC_send_no_wait                    */
/*    sends a string out the comm port */
/* bring read low*/
itoa((a & 0x7b),value,16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/* read Port C */
ITC_send_no_wait("PRC$\r");
read_data = ITC_data();

/*bring read high*/
itoa((a & 0x7f),value,16);
strcpy(string,"PWB$");
strcat(string,value);
strcat(string,"\r");
ITC_send(string);


/* then return to idle state */
ITC_send("PWB$7F\r");

/*return value read*/
return read_data;
}

/*****************************************/
/*   ITC_send                            */
/*    sends a string out the comm port  */
/*    then waits for the ITC '>' prompt */
/*****************************************/
void ITC_send(str)
char *str;
{
int rval, tval;

while(*str != 0)    /* loop until end of string (NULL) */
            {
   do
      {
      tval = SendChar(0, *str);   /* wait until tx buffer empty */
      }
   while(tval == 0xff00);

   str++;      /* after sending character, go send the next one */
            }
do             /* after final character sent, went until '>' sent back
*/
   {
   do
      {
      rval = Receive(0,0);    /* go get receive buffer status
and/or data */
      }
   while(rval == 0);   /* loop until a character received */
   }
while((char)rval != '>');    /* continue looping until prompt
character rx'd */


/*    does not wait for any response   */
/*****************************************/
void ITC_send_no_wait(str)
char *str;
{
```

```
int rval, tval;

while(*str != 0)    /* loop until end of string (NULL) */
            {
   do
      {
      tval = SendChar(0, *str);   /* wait until tx buffer empty */
      }
   while(tval == 0xff00);

   str++;      /* after sending character, go send the next one */
            }
}

/*****************************************/
/*   ITC_data                            */
/*  retrieves the data recv'd from a   */
/*  port read operation                 */
/*****************************************/
char ITC_data()
{
char    string[3], value, *ptr, result;
int rval=0;

ptr = string;
string[2] =  NULL;
do
   {
   do
      {
      rval = Receive(0,0);
      }
   while(rval == 0);
   switch(value = (char)rval)
      {
      case 'O':       /* if characters are O, K, $, or >, ignore */
         break;
      case 'K':
         break;
      case '$':
         break;
      case '>':
         break;
      case 'L':
         printf("\nOverflow or Underflow has occured\n");
         break;
      default:        /* otherwise characters should be results */
         *(ptr++) = value; /* put results in string */
      }
   }
while(value != '>');    /* continue looping until prompt character
rx'd */

/* string should now contain ascii hex value of result */
/* convert from hex string to a number */
if(isdigit(string[0]))
   result = (string[0] - 0x30) * 0x10;
else
   result = (string[0] - 0x37) * 0x10;
if(isdigit(string[1]))
   result = result + (string[1] - 0x30);
else
   result = result + (string[1] - 0x37);
return result;    /* return results converted to a number */
}
```


2
3
VCC
R1
4.7K
RESET
PortC[7..0]
PortB[7..0]
U1
VCC
PortC0
PortC1
PortC2
PortC3
PortC4
PortC5
PortC6
PortC7
PortB5
PortB2
PortB3
PortB0
D0
D1
D2
D3
D4
D5
D6
D7
CS
RD
WR
C/D
LCTR/LLTC
ABGT/RCTR
INPUT A
INPUT B
BW
CY
LS7166
U2A
U2B
SN74HC00
SN74HC00
Title
Quadrature Encoded Input
Size Number
A
Date: 12-Sep-1994
File: G:\ITC232\APPS\QUA
2
3