# RMV ELECTRONICS INC.

**Application Note:**

| | |
|---|---|
| | **Application #:** 00005 |
| | **Date**: September 1994 |
| **Description:** Using the ITC232-A for Pulse Counting. | **Status:** Draft version |

This circuit illustrates the of the ITC232-A in a pulse counter and control application. The maximum number of counts is 24 bits (~16 million) and a presettable compare value can be added to initiate a control action when the correct number of pulses has been received. A secondary counter (with two more able to be added on) with a maximum count of 16 bits each (~65 thousand) can also count the pulse stream and indicate when a preset number of pulses is reached. This counter could be used as a warning indicator before the primary counter reaches its action point.

## 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 details.

The circuit is implemented using timer/counter combinations. In this example we have used the Intel (or equivalent) 82C53 Programmable Interval Timer (PIT) and the LSI Computer Systems LS7166 MultiMode Counter (MMC). The 82C53 contains three independent 16 bit counters, each capable of running in one of several modes. The LS7166 is similar in nature and contains a single counter. The LS7166 was used because of it's enhanced operation and single 24 bit counter. If reduced count size and slower operation is adequate, the LS7166 can be removed and the other counters within the 82C53 can be used instead. This would reduce the chip count and lessen the circuit cost. The circuit also includes a single pole, double throw relay for immediate control action capability, along with a transistor driver.

Counter 0 in the PIT is used in terminal count mode. The counter value is set to the warning set point. When this number of counts has been reached, an IRQH signal will be produced, signaling the PC. The MMC is initialized as a standard up counter with compare (COMP) output enable. The count value when the action is to occur is entered in the compare registers. When the count value is reached, the borrow (BW) pin will be asserted. This pin both signals the PC by the generation of a IRQL signal, as well as turning on the transistor. This, in turn, will energize the relay. The form 'C' contacts on the relay can be used for a variety of purposes. The counts in the MMC (and the relay) are reset by simply bringing the reset pin to ground or through software and the ITC232-A. The actual count value can be read at any time.

By using one of more of the 82C53 counters in the same mode, several levels of warnings or other control actions can be initiated. If desired the out lines of each counter can also be tied to external hardware such as lights, relays, etc.

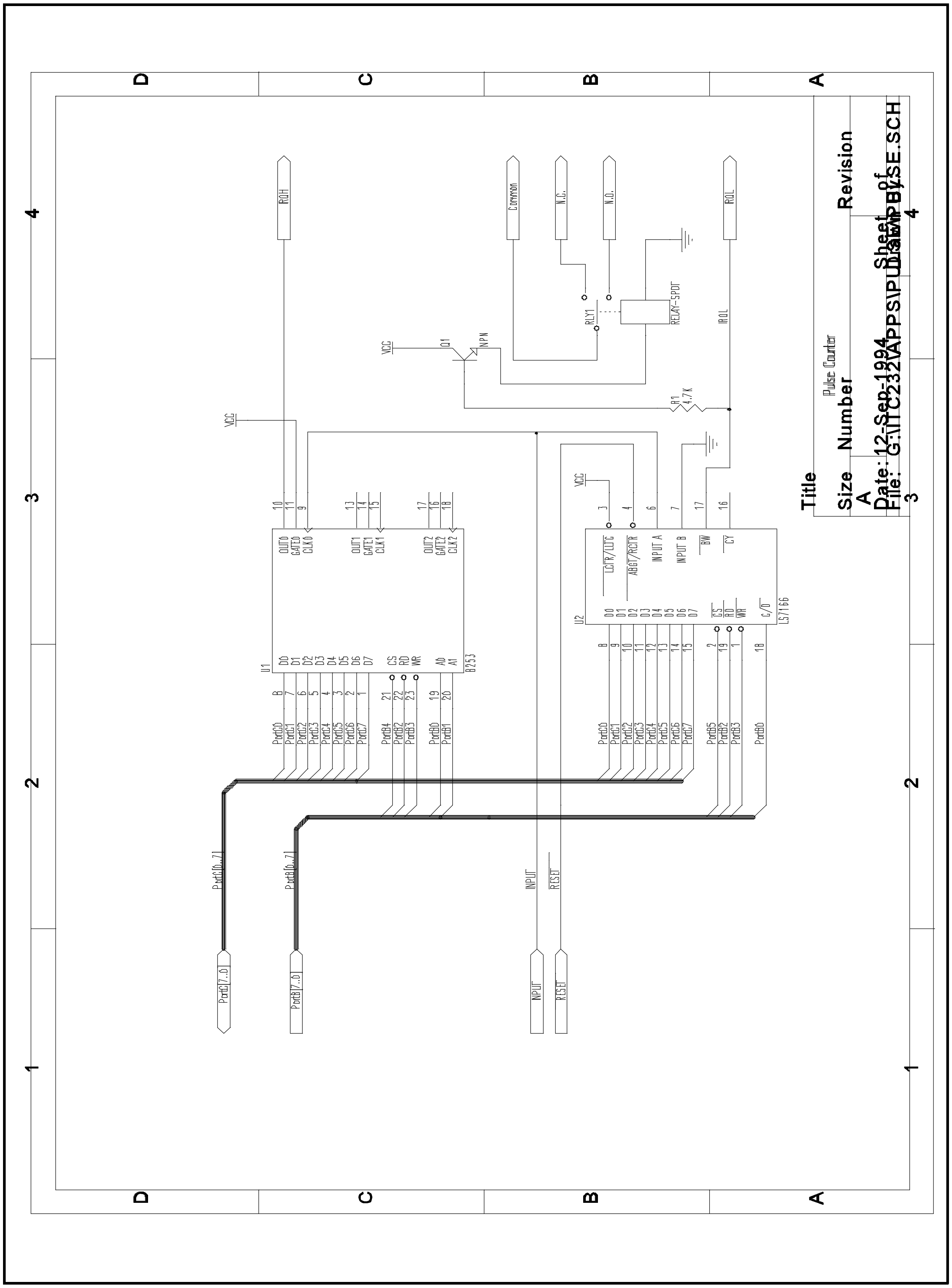
When using the LS7166 as the counter, the input pulse frequency is limited to 20 Mhz. Replacing the LS7166 with the 82C53 will reduce the upper pulse frequency to 8 Mhz. If higher frequency operation is desired, a prescaler can be inserted before the counter and the results calculations modified accordingly. For example, an ECL divide by 10 prescaler would allow operation to 200 Mhz. However, the counters are then operating effectively at 10 pulses per count and the warning and control numbers would have to be adjusted accordingly.

```
/*
Functional code listing for the pulse 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 PIT and MMC
Port B, bits 2 and 3 are the active low read and write lines
```

```
respectively
Port B, bit 4 is the active low chip select for the PIT
Port B, bit 5 is the active low chip select for the MMC
Port B, bit 6 is not used
Port B, bit 7 is an output for resetting the MMC counter
IRQH is used to indicate the PIT pulse counter has reached it's
preset
IRQL is used to indicate the MMC pulse counter has reached
it's preset
*/
/* User preset definitions here. Alternatively, the user can enter
the values
via the PC software, or by whatever other means may be
necessary.
#define PIT_preset      1000
#define MMC_preset      2000

/* Constant definitions */
#define PIT_CTR_0       0xec
#define PIT_CTR_1       0xed
#define PIT_CTR_2       0xee
#define PIT_CONTROL     0xef
#define MMC_DATA        0xdc
#define MMC_CONTROL     0xdd
#define BAUD_9600       0x0c
#define COMM_1          0x01
#define COMM_2          0x02
#define NULL            0x00

/*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 */
char    ITC_data(void); /*function that returns the data value in
response
                                        to a port read operation */


/* 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()
{
char    count_hi,count_mid,count_lo, input;
long    count_total, MMC_preset;
char    PIT_preset_lo, PIT_preset_hi;
char    MMC_preset_lo, MMC_preset_mid, MMC_preset_hi;
int     PIT_preset;
/*Initialize the PC serial port as for 9600 baud, no parity, 8 bits,
1 stop*/
Setup(BAUD_9600, 3, COMM_1);

/*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");

/*Set PIT counter 0 mode to 0 (terminal count), two byte xfer,
binary counting*/
out_data(PIT_CONTROL,0x30);
PIT_preset_lo = PIT_preset; /* lo byte only from user
parameter */
PIT_preset_hi = PIT_preset >> 8; /* hi byte */
out_data(PIT_CTR_0,PIT_preset_lo); /*counter 0 low byte*/
out_data(PIT_CTR_0,PIT_preset_hi); /*counter 0 hi byte*/

/*Reset MMC, then setup for counting up, normal mode,
compare enabled,
external counter reset enabled, binary count */
/* 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, 0xb0);
/* set quadrature control register */
out_data(MMC_CONTROL, 0xc0);

/* set user compare values in MMC preset */
MMC_preset_lo = MMC_preset; /* lo byte only from user
parameter */
MMC_preset_mid = MMC_preset >> 8; /* middle byte */
MMC_preset_hi = MMC_preset >> 16; /* hi byte */
out_data(MMC_CONTROL, 0x01); /*
out_data(MMC_DATA, MMC_preset_lo); /* comparator 0 low
byte*/
out_data(MMC_DATA, MMC_preset_mid); /*comparator 0
middle byte*/
out_data(MMC_DATA, MMC_preset_hi); /*comparator 0 hi
byte*/

/* main loop is here, the process is stopped by hitting any key */
while( !kbhit() )
{
/* deassert the counter reset line */
ITC_send("PWB$FF\r");

/* now wait for either an IRQL or IRQH interrupt from the
ITC232 */
do
   {
   input = Receive(0,0);
   }
   while(input == 0);  /* if nothing received, just wait */

if((char)input == 'H')
```

```
	{
	/* the PIT counter has reached terminal, send a message
*/
	printf("\nPIT counter preset value reached\n");

	}
else
	{
	/* the MMC counter has reached terminal, send a
message */
	printf("\nMMC counter comparator value reached\n");
	}

} /*end of while(kbhit... loop */
printf("\nInput Pulse Counter function halted\n");
/* reset the counter and release output control */
ITC_send("PWB$7F\r");
out_data(MMC_CONTROL, 0x20);

} /*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);


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


/* 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 */

}

/***************************************/
/*   ITC_send_no_wait                 */
/*    sends a string out the comm port */
/*    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 */
}
```


Title
Pulse Counter
Size
A
Number
Revision
Date: 12-Sep-1994
Sheet of
File: G:\ITC232\APPS\PULSE\PULSE.SCH
Drawn By:
U1
8253
OUT0
GATE0
CLK0
OUT1
GATE1
CLK1
OUT2
GATE2
CLK2
D0
D1
D2
D3
D4
D5
D6
D7
CS
RD
WR
A0
A1
U2
LS7166
LCTR/LLTC
ABGT/RCTR
INPUT A
INPUT B
BW
CY
C/D
PortC0
PortC1
PortC2
PortC3
PortC4
PortC5
PortC6
PortC7
PortB0
PortB1
PortB2
PortB3
PortB4
PortB5
PortC[0..7]
PortB[0..7]
INPUT
RESET
VCC
Q1
NPN
R1
4.7K
RLY1
RELAY-SPDT
ROH
ROL
Common
N.C.
N.O.