Discrete Logic Replacement

Generator

Author: *Kirill Yelizarov V.*
Moscow Power Engineering Institute
Moscow
email: tihonov@srv-vmss.mpei.ac.ru

OVERVIEW

The Logic Macro language discussed in this application note may be used to emulate several 7400 series TTL and 4000 series CMOS components, make a circuit of your own or be used as an internal bit processor. To show the advantages of this language, I built a generator with external timing components and a divide by two, using only macro language commands. With the help of this Macro language a simple signal processing can be made.

LOGIC MACRO LANGUAGE

The Logic Macro Language enables several old type 7400 and 4000 series ICs to be replaced without losing functionality. This is especially true with the 4000 series, when used on low frequencies or in static mode. Basically, this language operates only with bits, however, consider them as IC pins. To define a bit you have to write the following:

```
#define        Bit     GPIO,0
```

This will define a PIC bit 0 to be a Macro language pin. Not only general purpose registers may be used. Any RAM area that is previously defined or PICmicro™ flags may be used too.

```
IntD      equ   0x08  ;Some internal data
#define   LED1  IntD,0
```

At the moment the pins are defined, they are not defined as input or output pins. This is done later. By default, all internal pins are input and output at the same moment (the actual type depends on where the pin is placed in the macro function). External pins (microcontroller pins) may be set as input or output only.

Set Pin as Output (`LogOUT`)

This command is used to set the corresponding PICmicro pin to be output. On startup all pins are inputs. This command is working with general purpose in/out registers only. You can try to assign some pins from RAM to be set as output only, but the code will be skipped. Here is the instruction description:

```
LogOUT          x
```

where **x** is a Macro language pin. With the above stated definition this command will look like this:

```
LogOUT          Bit
```

And the pin **Bit** will be set as output.

Delay: 3 cycles

This language, besides the pin data, uses two RAM cells to keep TRIS register data and a Counter to make loops.

Set Pin as Input (`LogIN`)

This command is used to set the corresponding PICmicro pin to be input. On startup, all pins are inputs so there is no need of using this command on initialization, but it may be useful inside the circuit. This command is working with general purpose in/out registers only. You can try to assign some pins from RAM to be set as input only, but the code will be skipped. Here is the instruction description:

```
LogIN           x
```

where **x** is a Macro language pin. With the above stated definition this command will look like this:

```
LogIN           Bit
```

And the pin **Bit** will be set as input.

Delay: 3 cycles

Set Pin as Input or Output (LogSTATE)

This command is used to set the corresponding PICmicro pin to be input (third state of output) or output. If the y pin is low then x pin is output else x pin is input. Here is the instruction description:

```
LogSTATE        x,y
```

where **x** is a Macro language pin. With the above stated definition this command will look like this:

```
LogSTATE        Bit,EO
```

If EO pin is low, then **Bit** pin will be set as output.

Delay: 5 cycles

Logic SET Operation (LogSET)

```
LogSET          x
```

This macro sets pin x high. This is usually needed to initialize flip flops or other pin units in the circuit.

Delay: 1 cycle

Logic CLR Operation (LogCLR)

```
LogCLR          x
```

This macro sets pin x low. This is usually needed to initialize flip flops or other pin units in the circuit.

Delay: 1 cycle

Logic NOT Operation (LogNOT)

```
LogNOT          x,y
```

Logic inverter. Output level of y is inverted to the input level of x.

Delay: 4 - 5 cycles

Logic AND Operation (LogAND)

```
LogAND          x,y,z
```

Logic AND function. If x=y=1, then z=1. Else, if any of x or y is equal to 0, then z=0.

Delay: 4 - 7 cycles

Logic OR Operation (LogOR)

```
LogOR           x,y,z
```

Logic OR function. If x=y=0, then z=0. Else, if any of x or y is equal to 1, then z=1.

Delay: 4 - 7 cycles

Logic Repeater Operation (LogREP)

```
LogREP          x,y
```

Logic repeater is used mostly to output data through external pins. Output y is always equal to input x.

Delay: 4 - 5 cycles

Logic Switch (LogSW)

```
LogSW           x,y,z
```

Logic switch is used to output input data x to y pin if z is logic low (switch closed), if z is logic high (switch opened) then y stays unchanged.

Delay: 3 cycles (switch opened), 6 - 7 cycles (switch closed).

Logic Branch to Label (LogJMP)

```
LogJMP          x,y,label
```

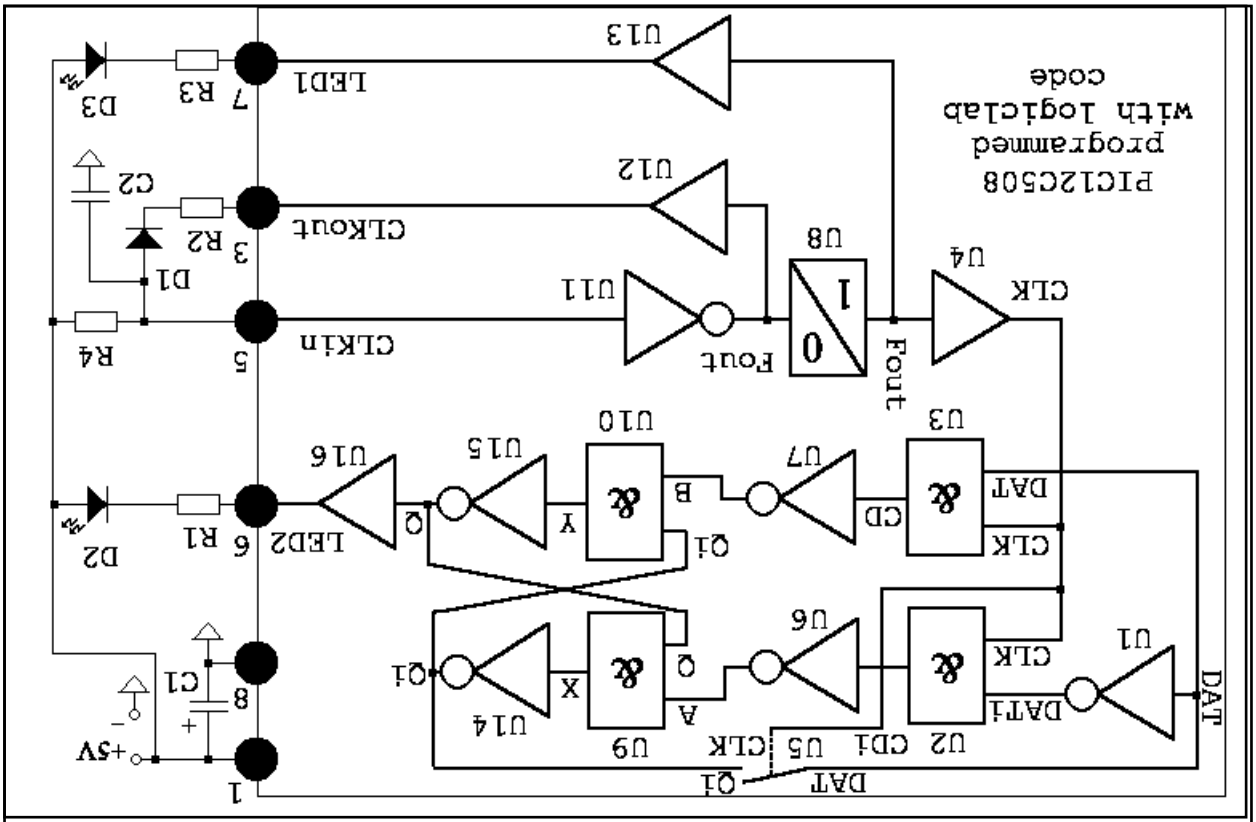
This macro checks state and branches to the label if pin x is equal to flag y. If the pin values are not equal then flag y is saved with value of x, and the program continues. This function is useful to skip some code if the input value of the circuit to be skipped is unchanged.

Delay: 5 - 6 cycles (branch to label), 7 cycles (skip)

A GENERATOR WITH A DIVIDE-BY-TWO

The schematic diagram with artificial circuit inside PICmicro is shown in Figure 1. Total scheme delay is about 140 µs (with a 4 MHz oscillator). So this scheme can run at approximately 7 kHz.

FIGURE 1: SCHEMATIC DIAGRAM



Note: Other PICmicro pins should be grounded.

BILL OF MATERIALS

Capacitors

C1 - 47 µF (electrolytic)

C2 - 1 µF

Diodes

D1 - Any type diode

D2 - D3 Red light emitting diodes

Resistors

R1 - 330 Ohm

R2 - 22 kOhm

R3 - 330 Ohm

R4 - 3.0 MOhm

Miscellaneous

U1 - PIC12C508 programmed with Logiclab code

Generator

The generator itself is made of two units (U11 and U12). It can be clocked through unit U11 (CLKin pin) with the external generator. In this case, inverted clock can be monitored on unit U12 output (CLKout pin). With four external components, an internal generator can be achieved. The generator frequency can be calculated with the following formula:

$$F = 1/T$$

$$T = 2/3.R4.C2$$

Resistor R2 is used to discharge capacitor C2 and is used to regulate the high-to-low clock ratio. You may vary the value of R2, but it should be always 100 times smaller than R4 for proper generator work. Diode D1 prevents capacitor charge through the U12 pin transistor, when the pin is raised high. With the values shown in Figure 1, the generator frequency will be about 0.5 Hz.

Brancher (U8)

This brancher is used to increase generator accuracy to the highest possible value. It searches when the U11 output changes its level to opposite and continues to run the rest of the program, else it waits for the change level event. The highest frequency achieved is when the rest of the microcontroller code is done and the capacitor C2 is not yet charged or discharged.

D (data) Trigger

D trigger itself consists of the following units: U1, U2, U3, U6, U7, U9, U10, U14, U15. It has data input called DAT, clock input called CLK, and two outputs, Q and Q inverted. To make this trigger work as a divider by two Q inverted output is connected to DAT data input via logic switch.

Logic Switch (U8)

With the help of this switch data is sent from Q inverted line to the DAT line when CLK line is low only. This is needed to prevent auto generating of D trigger when the CLK line is high.

TIPS ON HOW TO WRITE PROGRAM AND MORE

First start with units whose input values are known. Don't forget to initialize flip-flops with opposite output data. It's better to start with generators and input pins and walk to the output pins. If the next unit pin needs a value that is unknown, then stop and start another circuit to get this unknown value. Don't use one and the same pin (RAM bit) for different unit pins, even if logic says they will never cross, especially in flip-flops. If part of the circuit is unstable (when input value of previous unit depends on the output value of the next unit), you should unite this code in a loop. For flip-flops, a counter of 3 in a loop is enough. Macro language and loops don't affect any PICmicro flags, so they may be also used as special pins (for example, use TO or PD bit to block generator on startup). This Macro language has a function to animate tri-stated pins. If such thing is needed inside the "chip" then a switch or a brancher may be used instead. Example 1 shows on how to use tri-stated external pin of the "chip."

EXAMPLE 1: USING A TRI-STATED EXTERNAL PIN

```
define            EOflag     RAMdata,6    ;io flag
define            EO         GPIO,3       ;input enable output pin
define            IOpin      GPIO,0       ;in/out pin
;enable flags
                  LogIN                   IOpin;set IOpin to third state
LogSET                       EOflag

Start:
LogJMP            EO,EOflag,Continue;if EO is equal to EOflag then jump to Continue
LogSTATE          IOpin,EO   ;EO pin changes its state then change IOpin state
Continue:
;*********************************
;                 some code
;*********************************
                  goto                    Start
```

QUICK CODE IDEAS

This Macro language is fully microcontroller independent and may be used on any PICmicro. If clocked by interrupts, watch dog timer or even sleep functions, this language may be used with other code or mixed with it on the same chip.

MICROCHIP TOOLS USED

Assembler/Compiler Version

MPASM v01.50

APPENDIX A: SOURCE CODE

```
;Logic Lab
;Author: Kirill Yelizarov

         LIST                    P=PIC12C508, R=DEC
         INCLUDE                 <p12c508.inc>

         __CONFIG _IntRC_OSC & _WDT_OFF & _CP_OFF & _MCLRE_OFF



TrisReg             equ          0x07          ;Tris register
IntD1               equ          0x08          ;Some internal data
IntD2               equ          0x09          ;Some internal data
Count               equ          0x0a          ;Counter


;******* Logic Pins **********

#define  LED1       GPIO,0        ;LED F output
#define  LED2       GPIO,1        ;LED F/2 output
#define  CLKin      GPIO,2        ;External clock source input
#define  CLKout     GPIO,4        ;Clock output

#define  CLK        IntD1,0       ;trigger clock input
#define  DAT        IntD1,1       ;trigger data input
#define  DATi       IntD1,2       ;inverse data
#define  CDi        IntD1,3       ;DATi & CLK
#define  CD         IntD1,4       ;DAT & CLK
#define  Fout       IntD1,5       ;Generator output
#define  A          IntD1,6       ;inverse of CDi
#define  B          IntD1,7       ;inverse of CD

#define  X          IntD2,0       ;A & Q
#define  Y          IntD2,1       ;B & Qi
#define  Q          IntD2,2       ;data trigger output
#define  Qi         IntD2,3       ;inverse data trigger output
#define  FoutF      IntD2,4       ;Frequency flag

;***************** Logic Macros ************************

;--------Set pin as output------
;x - RAM cell
;b - bit
LogOUT  macro   x,b
         IF          x==GPIO                    ;Check if x is GPIO register
                     bcf          TrisReg,b     ;clear the bit
                     movf         TrisReg,W     ;move TRIS to W
                     tris         GPIO          ;and set new TRIS
         ENDIF
         endm
;----------------------------

;--------Set pin as input------
;x - RAM cell
;b - bit
LogIN    macro   x,b
         IF          x==GPIO                    ;Check if x is GPIO register
                     bsf          TrisReg,b     ;set the bit
                     movf         TrisReg,W     ;move TRIS to W
                     tris         GPIO          ;and set new TRIS
         ENDIF
         endm
;----------------------------
```

```
;--------Change pin IO state depending on f,fb pin------
;x,f - RAM cell
;b,fb - bit
LogSTATE  macro   x,b,f,fb
         IF          x==GPIO                      ;Check if x is GPIO register
                     bsf            TrisReg,b     ;set bit to be input
                     btfss          f,fb
                     bcf            TrisReg,b     ;clear the bit to be output
                     movf           TrisReg,W     ;move TRIS to W
                     tris           GPIO          ;and set new TRIS
         ENDIF
         endm
;----------------------------

;-----Perform a branch to label on some condition-----
;x,f - RAM cell
;xb,fb - bit
;lbl - label
;if x,xb is equal to f,fb the branch to label
;else set f,fb with x,xb value and continue
LogJMP    macro       x,xb,f,fb,lbl
         local       test1
         local       out
         btfsc       x,xb           ;goto test1 if pin x,xb is high
         goto        test1
         btfss       f,fb           ;goto label lbl if f,fb flag is low (f,fb = x,xb)
         goto        lbl
         bcf         f,fb           ;else clear f,fb because x,xb is low
         goto        out
test1:
         btfsc       f,fb           ;goto label lbl if f,fb flag is high (f,fb = x,xb)
         goto        lbl
         bsf         f,fb           ;else set f,fb because x,xb is high
out:
         endm
;-----------------------------------------------------

;------------Set pin high--------------------
;x - RAM cell
;b - bit
LogSET    macro       x,b
         bsf         x,b
         endm
;-------------------------------------------

;------------Set pin low---------------------
;x - RAM cell
;b - bit
LogCLR    macro       x,b
         bcf         x,b
         endm
;-------------------------------------------

;-----------Logic AND------------------------
;x,y,s - RAM cell
;xb,yb,sb - bit
LogAND    macro       x,xb,y,yb,s,sb
         local       log0
         local       log1

         btfss       x,xb           ;if x,xb=0 then s,sb is always 0
         goto        log0           ;goto set s,sb=0
         btfss       y,yb           ;if y,yb=0 then s,sb is always 0
         goto        log0           ;goto set s,sb=0
         bsf         s,sb           ;else set s,sb=1
         goto        log1           ;return
```

```
log0:
         bcf         s,sb           ;set s,sb=0
log1:
         endm
;-----------------------------------------------

;---------------Logic OR-----------------------
;x,y,s - RAM cell
;xb,yb,sb - bit
LogOR    macro       x,xb,y,yb,s,sb
         local       log0
         local       log1

         btfsc       x,xb           ;if x,xb=1 then s,sb is always 1
         goto        log1           ;go to set s,sb=1
         btfsc       y,yb           ;if y,yb=1 then s,sb is always 1
         goto        log1           ;go to set s,sb=1
         bcf         s,sb           ;else set s,sb=0
         goto        log0           ;return
log1:
         bsf         s,sb           ;set s,sb=1
log0:
         endm
;-----------------------------------------------

;--------------Logic NOT------------------------
;x,s - RAM cell
;xb,sb - bit
LogNOT   macro       x,xb,s,sb
         local       log0
         local       log1

         btfsc       x,xb           ;if x,xb=1 then s,sb=0
         goto        log1           ;go to set s,sb=0
         bsf         s,sb           ;else set s,sb=1
         goto        log0           ;return
log1:
         bcf         s,sb           ;set s,sb=0
log0:
         endm
;-----------------------------------------------

;--------------Repeater---------------------------
;x,s - RAM cell
;xb,sb - bit
LogREP   macro       x,xb,s,sb
         local       log0
         local       log1

         btfsc       x,xb           ;if x,xb=1 then s,sb=1
         goto        log1           ;go to set s,sb=1
         bcf         s,sb           ;else set s,sb=0
         goto        log0           ;return
log1:
         bsf         s,sb           ;set s,sb=1
log0:
         endm
;-----------------------------------------------

;-------------------Switch-----------------------
;Macro to pass data from x,xb to s,sb if c,cb is low
;x,s,c - RAM cell
;xb,sb,cb - bit
LogSW    macro       x,xb,s,sb,c,cb
         local       log0
         local       log1
```

```
         btfsc      c,cb          ;if c,cb=1
         goto       log0          ;then skip
         btfsc      x,xb          ;if x,xb=1 then s,sb=1
         goto       log1          ;go to set s,sb=1
         bcf        s,sb          ;else set s,sb=0
         goto       log0          ;return
log1:
         bsf        s,sb          ;set s,sb=1
log0:
         endm
;-------------------------------------------------

;                      ------------ Test Code ------------

         org        0x00
         clrf       IntD1         ;clear bits
         clrf       IntD2         ;clear bits
         clrf       GPIO          ;Clear GPIO register
         clrf       TrisReg
         comf       TrisReg,F     ;Set all as input
         movlw      b'11000000'   ;Disable weak pull-up on Button pin
         option

;Set pins as output
         LogOUT     CLKout
         LogOUT     LED1
         LogOUT     LED2

;initialize flipflop (flip flops should be always initialized with opposite values)
         LogCLR     Qi
         LogSET     Q

Start:
         LogNOT     CLKin,Fout        ;lets build a generator
         LogREP     Fout,CLKout       ;output CLKin inversed level

         LogJMP     Fout,FoutF,Start  ;check the level is changed

         LogREP     Fout,LED1         ;Output to first LED to show the generator is working

         LogREP     Fout,CLK          ;Set clock data

;Data trigger set as a divider by 2

         LogSW      Qi,DAT,CLK    ;connect inverse output of Qi to Dat input on CLK low
         LogNOT     DAT,DATi      ;inverse DAT
         LogAND     CLK,DATi,CDi  ;Check CLK AND DATi and place result in CDi
         LogNOT     CDi,A         ;inverse result in CDi to A
         LogAND     CLK,DAT,CD    ;Check CLK AND DAT and place result in CD
         LogNOT     CD,B          ;inverse result in CD to B

;*********************** Flip Flop *******************************
         movlw      0x03          ;to make flip flop stabilize itself loop it three times
         movwf      Count
FlipFlop:
         LogAND     Q,A,X         ;check Q AND A and place result in X
         LogNOT     X,Qi          ;inverse result in X to Qi
         LogAND     Qi,B,Y        ;check Qi AND A and place result in Y
         LogNOT     Y,Q           ;inverse result in Y to Q
         decfsz     Count,F
         goto       FlipFlop
;************************** end of Flip Flop *****************************

         LogREP     Q,LED2        ;Output to LED2 to show Fout/2
         goto       Start
```

```
        org         0x1ff
        movlw       b’01110000’    ;set OSCCAL
        end
```

NOTES: