This is a Pic based level crossing controller using the minimum of components and built on Eurocard although Veroboard will do fine. It is easy to build and should not cost you more than £15 including 4 X SG90 tower pro servos.

The block the Level crossing is in should be at least twice the length (if poss) of the longest train with the level crossing in the middle, to facilitate trains going in opposite directions.

It detects the presence of a train through a toti or similar device producing an active low (0 volts) for the duration the train is in the level crossing block and triggers the following sequence.

-- when train is in the block yellow is on for 3 secs then off followed by
-- red lights flashing and sounder for 5secs when barriers are lowered.

-- barriers are staggered with 1/4 first and after a small delay 2/3
-- when barriers moving down red lights flash with sounder.
-- red lights flash with sounder while train in block

-- when level crossing cleared, red lights and sounder continue while barriers open
-- this time with 2/3 and 1/4 together.
-- red lights and sounder cease with barriers at 45 degrees.

-- when barriers are opening after a train has left the block and another train
-- enters the block while the barriers are in motion
-- barriers will lower immediately with red lights and sounder, may not be true to life
-- but will avoid embarrassment if detection is slow or block too short.

-- its up to YOU to deliver the logic state to the pic active low(ov on gp3 will do)
-- either by toti or other devices.

My own pic program is about 300 lines of code. It moves the Servos 90 degrees fixed. The Speed of movement is fixed.

The program is written in JAL (just another language) and is a high level language easy to understand and manipulate, it is free and available for download from the net. It is similar to basic so no steep learning curve and compiles your jal program to a hex file for burning to a pic with a programmer like pickit2 or similar.

Feel free to play with it and modify it.

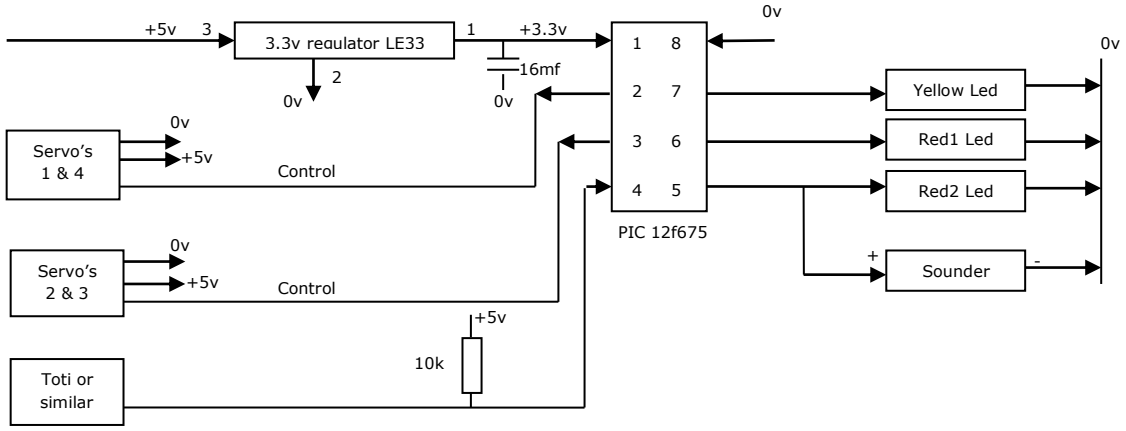
You the builder have to supply and build the traffic lights

There are no track side signals as I am reliably informed that level crossing barriers states (in real life) are integrated with the main track signals except in unusual circumstances.

In order to allow for non true mounting of the servos I removed the spline from the servo horn so that it can be positioned more easily, for demonstration purposes where an accidental knock would break them off, I have made them a flimsy push fit

Neil McNulty Wosag

July 2020



Construction info:

This Sounder is from Maplin, observe its polarity

Make sure you use an 8 pin dil socket for the pic

The servos are fed from 4 x (3 pin headers – 2mm pitch).

The pic is fed though a 3.3v regulator so no series resistors needed for leds.

Servos 1 & 4 are wired in parallel as are servos 3 & 4

16mf is 16micro farad 10v or similar

Power Supply:

5 volts dc

The current drain from the power supply while 2 servos are in motion is about 220ma

and about 50ma when leds flashing + sounder

Component list:

Pic 12f675

8 pin dil socket

Le33 3.3v regulator

10k ohm resistor

16 micro farad 10v capacitor

6 leds - 4 red, 2 yellow, use a pin header for this connection as you have to remove led connection for icsp

Sounder

Pin Headers - 4 groups of 3 for servo outputs and 1 x 6 for led connectors 2mm pitch

If you use icsp to program the chip see pinout below.

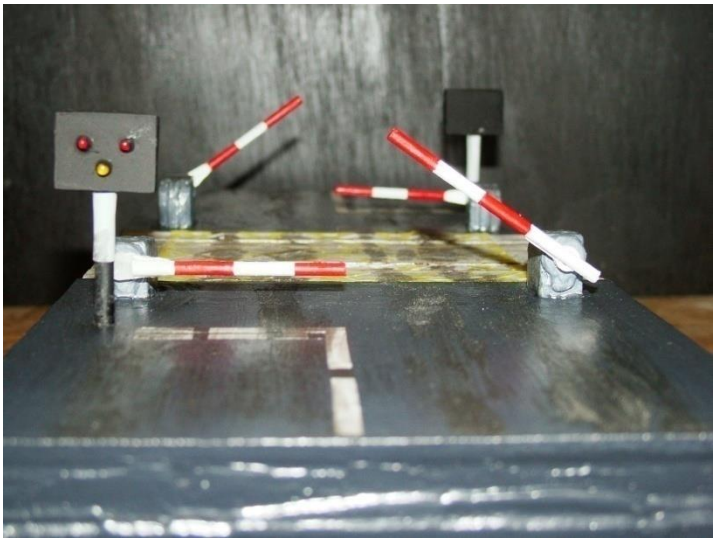
12f675

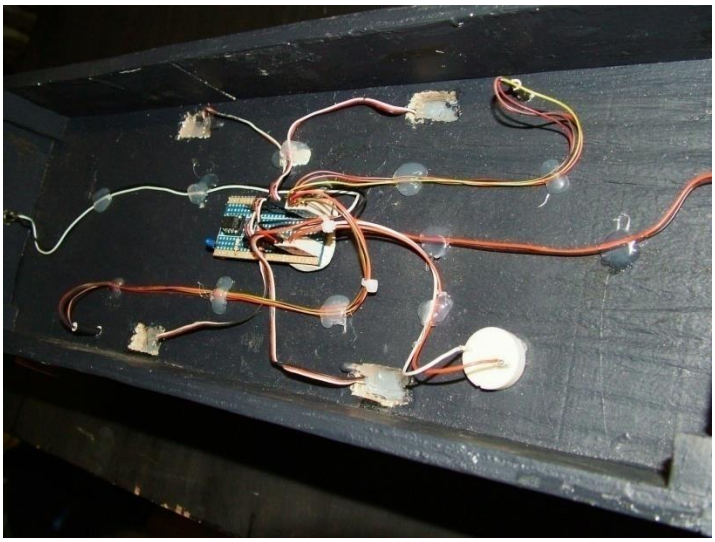
+5v o o 0v

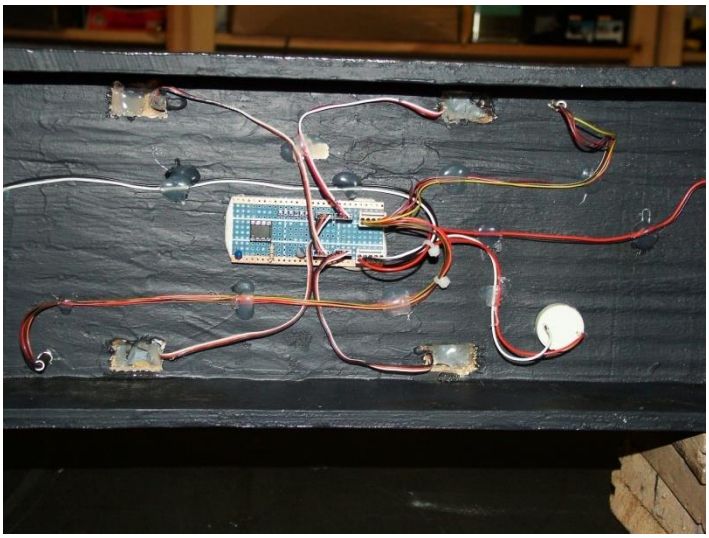
servos14 <--- GP5 output o o output GP0 --> **ICSP** Data / yell

servos23 <--- GP4 output o o Output GP1 --> **ICSP** Clock / red1

Toti/in/**ICSP**/mclr GP3 ---> GP3 input o o Output GP2 --> /red2







Click HERE to see video.

```
-- this version J is latest
-- JAL 2.4
-- This file is part of jallib  (http://jallib.googlecode.com)
-- Released under the BSD license (http://www.opensource.org/licenses/bsd-license.php)

-- ----------------------Description-----------------------------

-- Controller for Level crossing barriers or gates(4) with yellow and red flashing
lights
-- and sounder - sounder is fed from the flashing red leds, red2 output.

-- barriers are operated by 4 servo g9's each pair connected in parallel
-- whole sequence is triggered by train in level crossing block.

-- when train is in the block yellow is on for 3 secs then off followed by
-- red lights flashing and sounder for 5secs when barriers are lowered.
--
-- barriers are staggered with 1/4 first and after a small delay 2/3
-- when barriers moving down red lights flash with sounder.
-- red lights flash with sounder while train in block

-- when level crossing cleared, red lights and sounder continue while barriers open
-- this time with 2/3 and 1/4 together.
-- red lights and sounder cease with barriers at about 45 degrees.

-- when barriers are opening after a train has left the block and another train
-- enters the block while the barries are in motion
-- barriers will lower immeadiately with red lights and sounder, may not be true to
life
-- but will avoid embarrassment if detection is slow or block too short.

-- its up to YOU to deliver the logic state to the pic active low(ov on gp3 will do)
-- either by toti or other devices.

-- Intended for use with my merg canbus system but anything giving the appropriate
-- logic levels will do.

--  by Neil McNulty of Crookedholm
-- (bet you don't know where that is, well maybe you do by now)

--PIC 12f675 pinouts

--                                    +5v o    o  0v
--   servos14             <--- GP5 output o    o  output  GP0 -->  ICSP Data / yell
--   servos23             <--- GP4 output o    o  Output  GP1 -->  ICSP Clock / red1
--   Bk1in/icsp/mclr GP3  ---> GP3  input o    o  Output  GP2 -->  /red2 and sounder

-- NOTES: level crossing block should be at least the length of the average train
-- for bi-direction use as well as authenticity, will work fine even with double track
-- crossings or more

-- remove red and yellow connections if using icsp

-- servos will not twitch on power up.
-- if powered off when barriers closed, they will open on power up
-- if powered off when barriers up, nothing happens on power up
-- if powered off when servos in motion then you deserve what you get !
-- **if using icsp disconnect leds yell and red1 while doing so.
-- E&OE

---------------------------- system setups -------------------------

include 12f675
```

```
-- compiler instructions

pragma target clock 4_000_000       -- oscillator frequency
pragma target OSC INTOSC_NOCLKOUT
pragma target WDT  disabled         -- no watchdog
pragma target MCLR internal         -- reset externally

-- compiler instructions end

-- include commands are calls made to libraries that come with jal

include delay
include pic_data_eeprom

-- setting variables to chip pins

enable_digital_io()                 -- disable analog I/O (if any)

var bit yell is pin_GP0             -- output to yell

var bit red1 is pin_GP1             -- output to red1

var bit red2 is pin_GP2             -- output to red2 and sounder

var bit servos23 is pin_GP4         -- output to servos23  2&3 are in parallel

var bit servos14 is pin_GP5         -- output to servos14  1&4 are in parallel

var bit bk1in is pin_GP3            -- input from toti

--setting pin directions either inputs or outputs

pin_GP0_direction =  output

pin_GP1_direction =  output

pin_GP2_direction =  output

pin_GP3_direction =  input

pin_GP4_direction =  output

pin_GP5_direction =  output

  -------------------------- program variables ---------------------

var byte counter                    -- used to count, doh !
var byte gatesareopen               -- current state

var byte movingvalue               -- holds current counter value when opening gates
--var byte pause                      -- delay variable
--var byte pause2                     -- delay variable
var byte pause1                     -- delay variable
var byte gatestates                 -- gates left open or closed info for eeprom

--------------------------- default variable states ----------------

yell = off                          -- start with led off
red1 = off                          -- start with led off
red2 = off                          -- start with led off
servos14 = on     --stops kick on power up --patent pending
servos23=  on     --stops kick on power up --patent pending
```

```
--pause2 =   3 --  was 4
pause1 = 200 --  was 220 this value tweaked to sync red lights while servos move
--pause  =   2

  -- ------------------------startup sequence ----------------------

  delay_100ms(15)   --1.5 sec delay

 -- read open or closed values from eeprom
      data_eeprom_read(0,gatestates)   -- previously stored value at address 0

 if gatestates == 0  then    -- was barrier down on power off  if yes raise barriers

  for 125 using counter loop
  servos23=  on
  delay_10us(125 + counter)      -- 1.25ms to 2.5ms
  servos23 = off
  delay_1ms(20)               --repeats every 20ms
  end loop

  delay_100ms(15)   --1.5 sec delay

  for 125 using counter loop
  servos14 = on
  delay_10us(250-counter)      -- 2.5ms to 1.25ms only count down
  servos14 = off
   delay_1ms(20)
  end loop

     end if


gatesareopen = true

gatestates = 1            -- makes eeprom state = 1 with barrier up
data_eeprom_write(0,gatestates)

-- --------------------------end startup sequence -------------------

  -- this routine synchronises the flashing red lights while the servos move
  -- by using the counter intervals to control the lights

  procedure reds_and_gates is

   if counter == 1 then
  red1 =on
  red2 =off
  end if

  if counter == 10 then
  red1 =off
  red2 =on
  end if

  if counter == 19 then
  red1 =on
  red2 =off
  end if

  if counter == 28 then
  red1 =off
  red2 =on
```

```
  end if

  if counter == 37 then
  red1 =on
  red2 =off
  end if

  if counter == 46 then
  red1 =off
  red2 =on
  end if

  if counter == 55 then
  red1 =on
  red2 =off
  end if

  if counter == 64 then
  red1 =off
  red2 =on
  end if

  if counter == 73 then
  red1 =on
  red2 =off
  end if

  if counter == 82 then
  red1 =off
  red2 =on
  end if

  if counter == 91 then
  red1 =on
  red2 =off
  end if

  if counter == 100 then
  red1 =off
  red2 =on
  end if

  if counter == 109 then
  red1 =on
  red2 =off
  end if

   if counter == 118 then
  red1 =off
  red2 =on
  end if

end procedure


-- when opening gates red lights go out and sounder stops at about 45 degrees

  procedure reds_and_gates_final is


   if counter == 1 then
  red1 =on
  red2 =off
```

```
end if

if counter == 10 then
red1 =off
red2 =on
end if

if counter == 19 then
red1 =on
red2 =off
end if

if counter == 28 then
red1 =off
red2 =on
end if

if counter == 37 then
red1 =on
red2 =off
end if

if counter == 46 then
red1 =off
red2 =on
end if

if counter == 55 then
red1 =on
red2 =off
end if

if counter == 64 then
red1 =off
red2 =off
end if

if counter == 73 then
red1 =off
red2 =off
end if

if counter == 82 then
red1 =off
red2 =off
end if

if counter == 91 then
red1 =off
red2 =off
end if

if counter == 100 then
red1 =off
red2 =off
end if

if counter == 109 then
red1 =off
red2 =off
end if

 if counter == 118 then
```

```
  red1 =off
  red2 =off
  end if

end procedure



procedure yellowandclosegates  is
             -- yellow light on for 3secs then the reds flash --
 Yell=on
delay_100ms(30)   --3 sec delay
  yell=off

          -- red lights flashing 5 secs

for 15 using counter loop

 red1 =on
 red2 =off
  delay_1ms(pause1)
 red1 =off
 red2 =on
 delay_1ms(pause1)

 end loop

              -- now move servos 1 and 4

for 125 using counter loop

servos14 = on
delay_10us(125+counter)          --counts up to 250 which is 1.25ms to 2.5ms
                                    -- variable pulse width for servo
servos14 = off
delay_1ms(20)                     --repeat every 20ms

 reds_and_gates    -- calls routine

end loop

  delay_1ms(50)  -- sync transition from moving to stopped

--delay before servos 2 and 3 move


 for   8 using counter loop
 red1 =on
 red2 =off
  delay_1ms(pause1)
 red1 =off
 red2 =on
 delay_1ms(pause1)

 end loop

-- servos 2/3

for 125 using counter loop

   servos23 = on
   delay_10us(250 - counter)
   servos23 = off
```

```
   delay_1ms(20)

  reds_and_gates

end loop

 delay_1ms(50)  -- sync transition from moving to stopped

gatesareopen= false

end procedure

procedure closeall is  -- if train into block during the raising of barriers


for movingvalue using counter loop   -- increment previous barrier position
                                     -- only in opposite direction

  servos23 = on
  delay_10us((125 + movingvalue -1) - counter)  -- continue from previous barrier
position
  servos23 = off

 servos14 = on
 delay_10us((250 - movingvalue -1)+ counter)   -- the -1 as movingvalue should start at
0
                                  -- variable pulse width for servo
 servos14 = off
 delay_1ms(20)                      --repeat every 20ms

 reds_and_gates

end loop

  delay_1ms(50)  -- sync transition from moving to stopped

  gatesareopen = false

end procedure



procedure finalsequence is

-- now move all servos together


 for 125 using counter loop

  movingvalue=counter    --

   servos23=  on
   delay_10us(125 + counter)
   servos23 = off

   servos14 = on
   delay_10us(250-counter)
   servos14 = off
   delay_1ms(20)   --repeats every 20ms

  reds_and_gates_final

 if bk1in==off then   -- toti input low another train, what next ?
```

```
  closeall             -- run procedure closeall
  return               -- quit this procedure and return to whatever called it (forever
loop)
  end if
  end loop

  red1=off

gatesareopen = true

 gatestates = 1
data_eeprom_write(0,gatestates)

end procedure

procedure flashreds  is

 red1=on
 red2=off
delay_1ms(pause1)
 red1=off
 red2=on
delay_1ms(pause1)

end procedure

procedure write_to_eeprom  is

     data_eeprom_write(0,gatestates)   -- write gatestates to eeprom at posn 0

end procedure

     ----------------------------------------------------
--                    M A I N   P R O G R A M  LOOP
     ----------------------------------------------------
forever loop

if bk1in==off  & gatesareopen == false then flashreds    --train in block and gates
closed

elsif bk1in==off  & gatesareopen == true then yellowandclosegates--train in but gates
not closed

elsif bk1in==on & gatesareopen == false then finalsequence -- if train out and

--gates closed then final sequence i.e raise barriers with red flashing lights.

end if

-- sets eeprom value to 0 if gates closed

       if gatesareopen == false then
           gatestates= 0
           write_to_eeprom

           end if

end loop

                ---------------------------------------
--                              SIMPLES
                ---------------------------------------
```