EzyBus and a barrier level crossing on a single track line

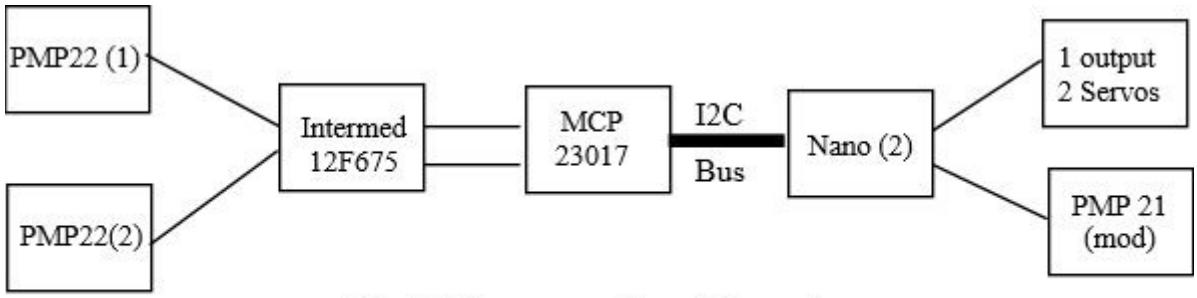
When developing a layout to demonstrate the EzyBus system the East of Scotland group (EOSAG) decided to include a level crossing at the rear of the layout for something to happen there. This required a means of detecting the approach of a train from either direction and equally when it had cleared the crossing. There was also a requirement for the flashing light to start appreciably before the barriers came down and, on re-opening to road traffic for the lights to stay on until the barriers had lifted.

Early on it was decided to use a laser TOTI either side of the crossing and, after some false starts, it was decided that an intermediate PIC circuit was required to combine the inputs from the wo TOTIs and to give the necessary timing, essentially as follows –

Detector A > HIGH; Lights flash, delay, barriers close
Detector B > HIGH; barriers open, delay, lights go out. And the same sequence in the opposite direction.

The original designs at their simplest used two inputs to the PIC (a 12F675) and two outputs, one for the lights and one for the barriers. This worked, sort of. There were unpredictable delays on triggering the system and if, perhaps due to a gap in the train, the first detector was re-triggered the system would re-set to the position with the barriers up. For various reasons, covid being just one, development on the crossing stopped at this point while other parts of the project were worked on.

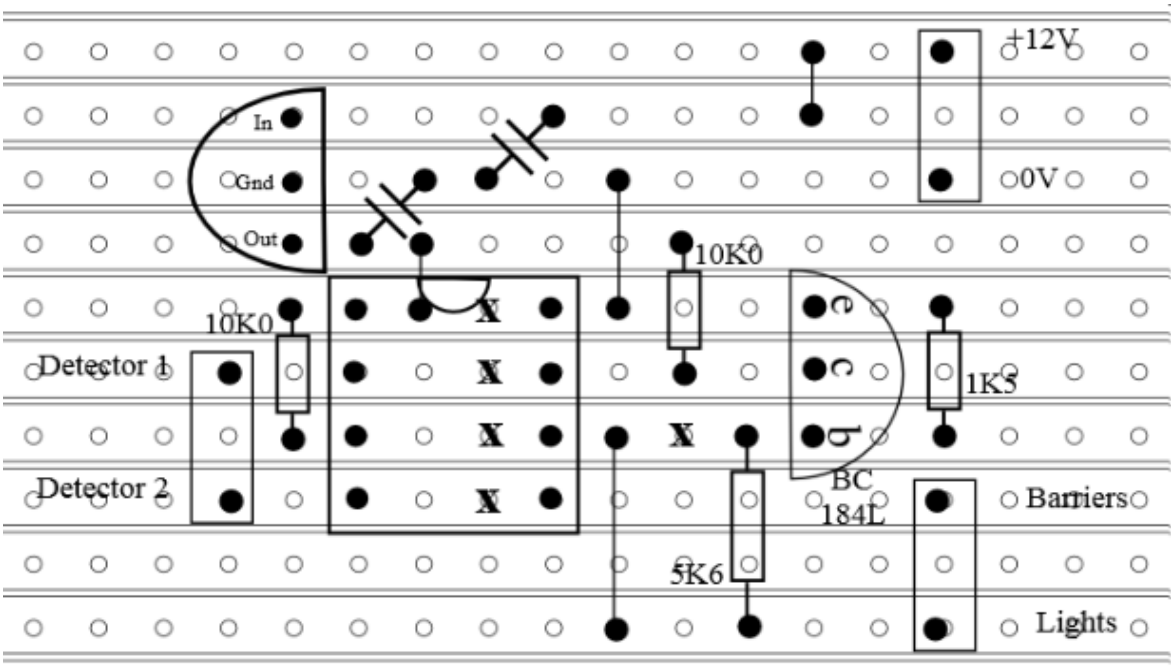
I returned to this in November 2020 and finally decided that another input was required to give the PIC the state of the lights. This was achieved by adding a transistor to the lights output and using the voltage on its collector as an input to the PIC. This inverts the signal so that a HIGH on the lights output gives a LOW input to the PIC. The block diagram for the complete system is –



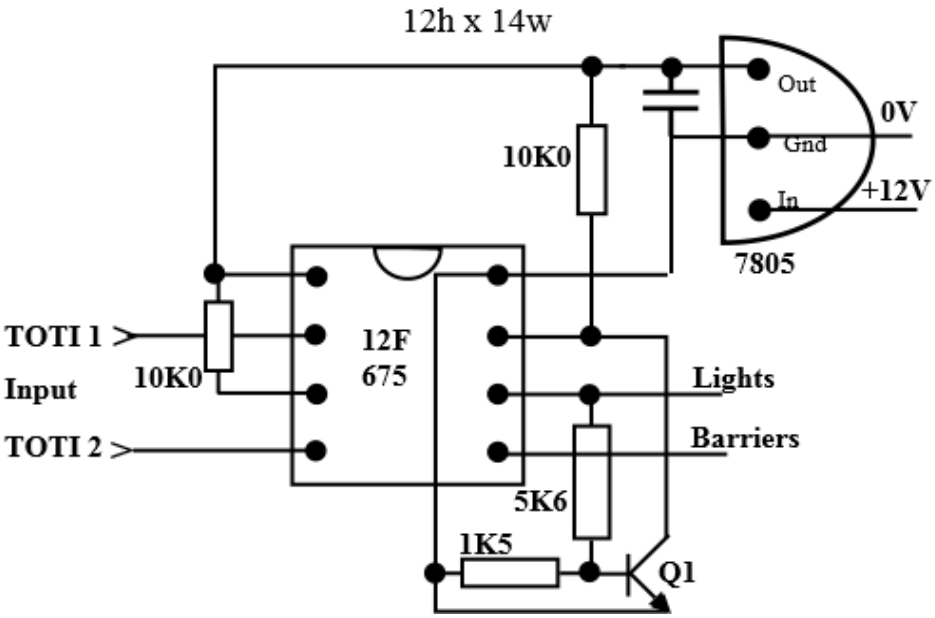
Block Diagram - Level Crossing Barriers and Lights

The laser TOTIs are standard PMP22 kits and the HIGH output on detection is used. The signals are combined in the Intermediate board and fed into EzyBus via the MCP23017 and use one or two servo outputs depending on whether or not two or four barriers are used, and one digital output to switch the lights on via a slightly modified PMP21. The PMP circuit is modified to the extent that it uses an NPN transistor in the ground side to switch the lights on and off.

The vero layout and the circuit diagram for the Intermediate board follow. Please note that while any general purpose NPN transistor can be used for Q1 I used a BC184L because they were in my box. ***The now more common BC547 has a different pin-out.***



Intermediate Board with Lights State



The program for this intermediate board is below and testing has shown that it works. The outputs from the 675 change as soon as the inputs do, it does not re-trigger should there be light transmission to the laser receiver while the train is passing, and it cannot be re-triggered until 20 seconds after the front of the train has cleared the second TOTI. This should allow enough time for any likely train to clear the crossing but can be changed merely by changing the two delay(20) statements to some other value.

NOTE. This program, and circuitry, was developed for a single track line and for trains capable of travelling in either direction and using a half barrier crossing. If a full barrier crossing is planned and it is felt that driving four servos off one output point on the Nano boards would be excessive then the barrier output can be fed to two inputs on the MCP23017. It has not been tested for twin track use, specifically against the possibility of two trains being in the detector region at the same time.

Chris Cosgrove

The program

```
;  JAL v2.4
;
;  Intermediate controller with Lights state v2.jal
;
;  This programme is for the intermediate board to control a barrier level
;  crossing as used in EzyBus. It takes its inputs from two Laser TOTIs and
;  outputs signals to switch the crossing lights and barriers.
;
-- A HIGH output is required to the EzyBus input switch unit to turn the lights
-- on. For the barriers it is a matter of convenience but this program assumes
-- HIGH for open and LOW for closed.
;
;  So, for the crossing open to road traffic -
;    Lights = LOW and Barriers = HIGH
;  For the crossing closed to road traffic -
;    Lights = HIGH and Barriers = LOW
;
;  This is written to overcome the lag in switching that has occurred with
;  all other versions and the re-triggering problem.
;
;  Chris Cosgrove, December 2020

-------------------------------------------------------------
--                          PIC 12F675 pinouts
--
--                                        +5v o    o  0v
-- Input from Detector 1 --> GP5  Input o    o  Input  GP0 <--  Lights state
-- Not connected           --> GP4  Input o    o  Output GP1 -->  Lights servo
-- Input from Detector 2 --> GP3  Input o    o  Output GP2 -->  Barriers servo
-- pin=HIGH means pin goes to +5v
-- -------------------------------------------------------------
```

```
include 12f675

pragma target clock 4_000_000     -- oscillator frequency
pragma target OSC INTOSC_NOCLKOUT
pragma target WDT  disabled       -- no watchdog
pragma target MCLR internal       -- reset internally
--
include delay
enable_digital_io()               -- disable analog I/O (if any)

;-----------------------------------------------------
; ========== Set up the variables ====================
;-----------------------------------------------------

Var bit Detector_1 is pin_GP5
Var bit Detector_2 is pin_GP3
Var bit Barriers is pin_GP2
Var bit Lights is pin_GP1
Var bit Lights_state is pin_GP0

pin_GP5_direction = input  ;From Detector 1
pin_GP3_direction = input  ;From Detector 2
pin_GP2_direction = output ;Barriers
pin_GP1_direction = output ;Lights
pin_GP0_direction = input  ;Inverted Lights output

;-----------------------------------------------------
;------------ set default variable states -----------
;-----------------------------------------------------

Lights = LOW     ; Default states for crossing,  Lights OFF, Barriers OPEN
Barriers = HIGH

;-----------------------------------------------------
;---------  Opening and closing procedures  ----------
;-----------------------------------------------------

procedure CloseCrossing() is   ; Closes crossing to road traffic
  Lights = HIGH
  delay_1ms(3000)
  Barriers = LOW
end procedure
```

```
procedure OpenCrossing() is    ; Opens crossing to road traffic
  Barriers = HIGH
  delay_1ms(3000)
  Lights = LOW
end procedure

procedure Crossing_stays_closed() is
  Lights = HIGH
  Barriers = LOW
end procedure

;-----------------------------------------------------
; Main Program
;-----------------------------------------------------

forever loop

;  From Direction 1
  if Detector_1 == HIGH & Lights_state == HIGH then
    CloseCrossing()
      while Detector_2 == LOW loop    ; Loop keeps barriers closed until train
        Crossing_stays_closed()              ; reaches second detector
      end loop
  end if

  if Detector_2 == HIGH & Lights_state == LOW then
    OpenCrossing()          ; The delay prevents re-triggering the lights before the
    delay_1s(20)              ; train has completely passed the detector
  end if

;  From Direction 2
  if Detector_2 == HIGH & Lights_state == HIGH then
    CloseCrossing()
      while Detector_1 == LOW loop     ; Loop as above
        Crossing_stays_closed()
      end loop
  end if

  if Detector_1 == HIGH & Lights_state == LOW then
    OpenCrossing()
    delay_1s(20)      ; Delay as above
  end if

end loop
```

```
-- -------------------------------------------------------
--
--  This program with the modified circuit (added a transistor to give Lights-
--  state) and with the addition of the 'while' loop appears to achieve all
--  required conditions :-
--
--  It triggers the operation of the crossing on the first beam break without
--  any lag.
--
--  Re-triggering the first detector (from either direction) no longer re-sets
--  the system as might be caused by a gap in the train.
--
--  It re-sets the system on the first beam break of the second detector with
--  delay to prevent re-triggering while the train passes.
```