

MC68HC908QT4 IR CONTROLLER

DESCRIPTION: This project uses an RCA remote control to control the operation, (turn on or off), 4 exterior lights on my home.

BACKGROUND: My home has two usable side yards that are not lighted. I do not like motion detector operated lights or X10 so I designed the IR Controller to operate the lights, (obviously the controller may be used to control many other devices).

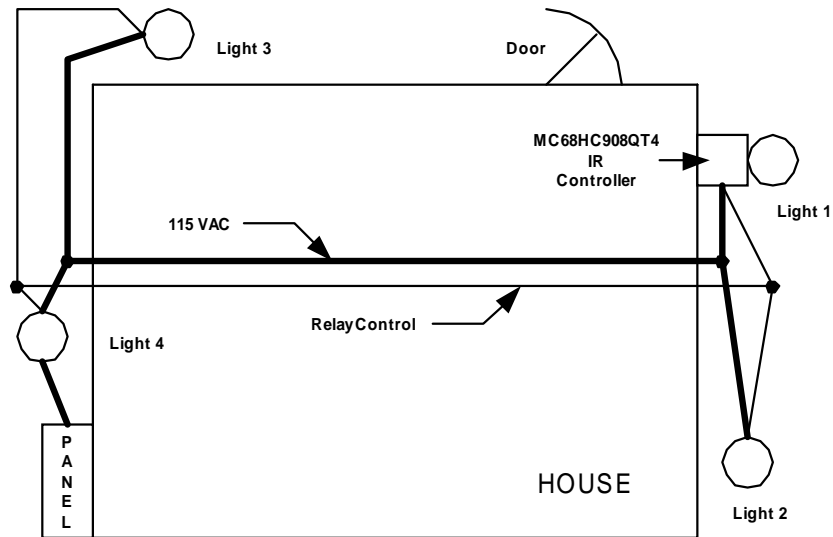


FIGURE 1 - Home and Lights Layout

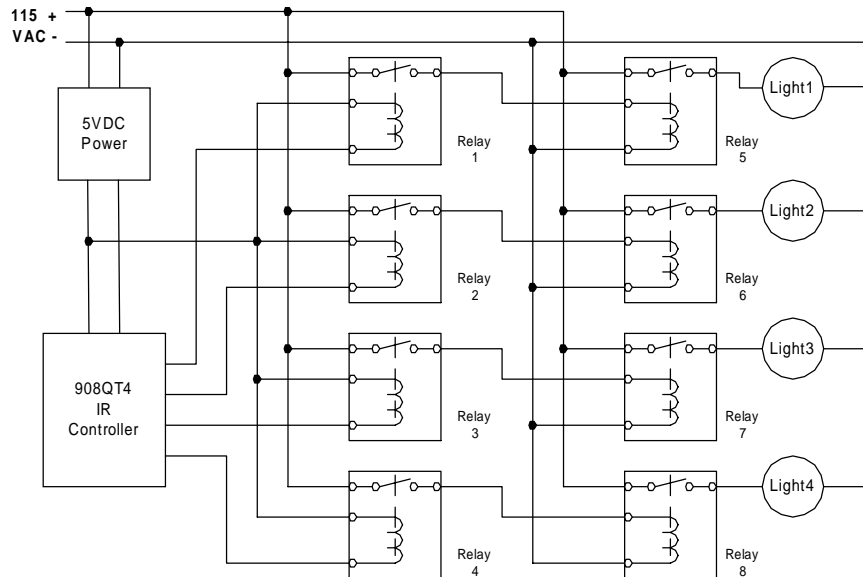


FIGURE 2 - Wiring Diagram

CIRCUIT:

The circuit will decode an RCA TV Remote Control to control 4 exterior lights. The circuit consists of (1) Sharp GP1UM281QK Remote Controller Receiver, (1) Motorola 68HC908QT4 microcontroller, (1) 74HCT244 Buffer, (4) 5VDC low power relays, (4) 115VAC power relays, (1) 115VAC to 5VDC Power Supply, and associated wiring and packaging.

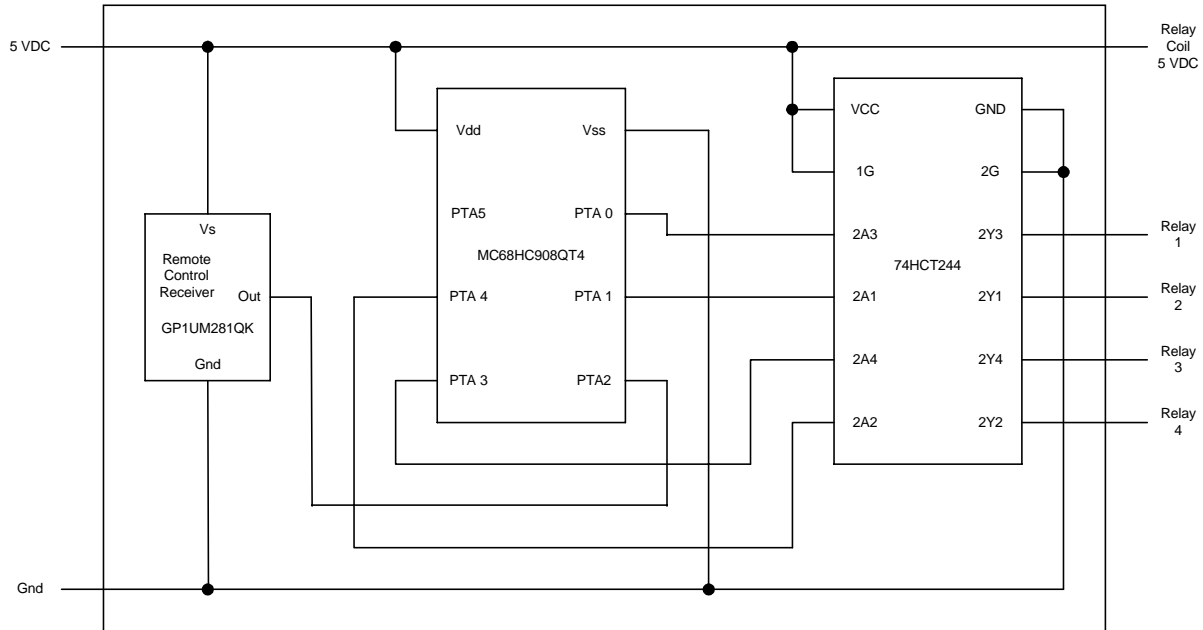


FIGURE 3 - IR Controller Circuit Diagram

SOFTWARE:

The software runs in a continuous loop waiting for a low input from the Remote Control Receiver. If this low is between 3.5 and 4 milliseconds the software measures the following high pulse. If this high pulse is greater than 3.5 mS then the software captures the following 24 code bits. If there are errors during the capture process the software goes back to the top waiting for a low input. If all 24 code bits are captured the software proceeds to decode the bits.

Figure 4 shows the pulse stream of the 1 key press. It can be seen from the figure that the second 12 bits are the same as the first 12 bits but inverted. Bits 9 - 12 are the bits of interest and contain the key press number code in binary.

The software compares bits 1 -12 against inverted bits 13 - 24 and if they match gets the value from bits 9 - 12. If there is a comparison error or key 1 - 4 was not pressed the software returns to the top. If key 1 - 4 was pressed the respected light output is toggled. There is a 0.4 second delay after a successful key decode to prevent inadvertent toggling of the output due to multiple key presses.

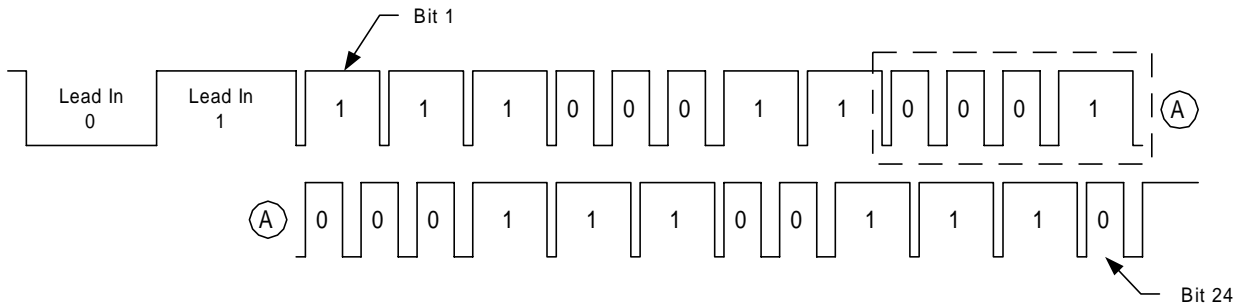


FIGURE 4 - RCA Remote - 1 Key Pulse Stream

PARTIAL CODE SAMPLE:

```

; MOTOPROJ.ASM
; This file decodes an RCA Remote Control to toggle
; 4 Light Outputs.
;
;
; ----- V+ ----- GND -----
; ----- PTA5 ----- PTA0 ----- Light_1
; ----- PTA4 ----- PTA1 ----- Light_2
; Light_4----- PTA3 ----- PTA2 ----- IR_In
; ----- MC68HC908QT4 -----
;
;
XDEF Entry,main

Include 'QT4.inc' ; MC68HC908QT4

Bits_Base EQU $E0 ;start location of storage of each of the
;24 remote code bits in indexed memory

IR_In EQU 2 ;Remote Controller Receiver output
;to HC08 pin 2 input

DEFAULT_RAM SECTION SHORT

;*****
;* Global Variables *
;*****

LOOP1 ds 1 ;for delay routine
LOOP2 ds 1 ;for delay routine
Puls_Cnt_Lo ds 1 ;counter for remote hi pulses
Puls_Cnt_Hi ds 1 ;counter for remote lo pulses
Bit_Cnt ds 1 ;temp counter used during acquiring of code bits and
;during moving of bits from indexed memory to Byte_1 -
Byte_4

Byte_1 ds 1 ;bits 1-6 of decoded remote code

```

```

Byte_2    ds 1           ;bits 7-12 of decoded remote code
Byte_3    ds 1           ;bits 13-18 of decoded remote code
Byte_4    ds 1           ;bits 19-24 of decoded remote code
           ; bytes 3 and 4 are same but inverted of bytes 1 and 2
           ; remember to deal with 2 hi bits in bytes 1 - 4
           ; as only 6 code bits are stored in each byte

```

```

DEFAULT_ROM          SECTION

```

```

;*****
;* Entry:  This is where the application starts.          *
;*****

```

```

Entry:

```

```

main:  rsp
       clra           ; Initialize A,X so that interrupt
       clrx           ; processing doesn't stop with
                   ; uninitialized register warning
                   ; when push A,X on the stack

```

```

       lda  $FFC0      ; load the oscillator TRIM value
       sta  OSCSTRIM   ; and use it to calibrate oscillator.
                   ; Automated programming tools such as
                   ; the Mon08 Cyclone set the correct
                   ; trim value in location $FFC0.

```

```

       mov  #$FF, PTAPUE ; enable all pullups on inputs

```

```

;*****
; set up I/O's and configs
;*****

```

```

       mov  #$FF,PORTA  ; all outputs to OFF, active low
       mov  #$FF,DDRA   ; PTA0 - PTA5 outputs, PTA2 is ALWAYS an input

```

```

       mov  #$06,INTSCR ; Disable IRQ interrupts (unused)
       mov  #$01,CONFIG1 ; Disable WDT, Low Voltage Reset set at 3V,
                   ;STOP instruction disabled
                   ;CONFIG2 = $00 from Power Up

```

```

mainloop:

```

```

       clr  Puls_Cnt_Lo
       clr  Puls_Cnt_Hi

```

```

       brset IR_In, PORTA, mainloop ;wait for start of lo pulse

```

```

CNT_LO_LEAD:

```

```

       mov  #7, LOOP1
       jsr  DELAY1      ;delay 20us
       inc  Puls_Cnt_Lo
       beq  mainloop    ;lo pulse too long start again

       brclr IR_In, PORTA, CNT_LO_LEAD ;still lo keep counting

```

```

lo:

```

```

       lda  #130
       cmp  Puls_Cnt_Lo ;130 - puls_cnt_lo
       bcc  mainloop    ;pulse cnt > 130

```

```

       lda  #180
       cmp  Puls_Cnt_Lo ;pulse cnt < 180
       bcs  mainloop    ;lo pulse correct duration, continue

```

CNT_HI_LEAD:

```
mov #7, LOOP1
jsr DELAY1      ;delay 20us
inc Puls_Cnt_Hi
beq mainloop    ;hi pulse too long start again

brset IR_In, PORTA, CNT_HI_LEAD ;still hi keep counting

lda #130
cmp Puls_Cnt_Hi ;pulse cnt > 130
bcc mainloop

lda #197
cmp Puls_Cnt_Hi ;pulse cnt < 197
bcs mainloop

;hi pulse correct duration, continue
;and get code bits
```

PROTOTYPE PHOTOS:

To prototype the circuit I built it up on a Radio Shack project PCB. I used a linear power supply to supply the 5VDC power. I used a protoboard with LED's to simulate the 115VAC Power Relays, to verify the circuit operation.

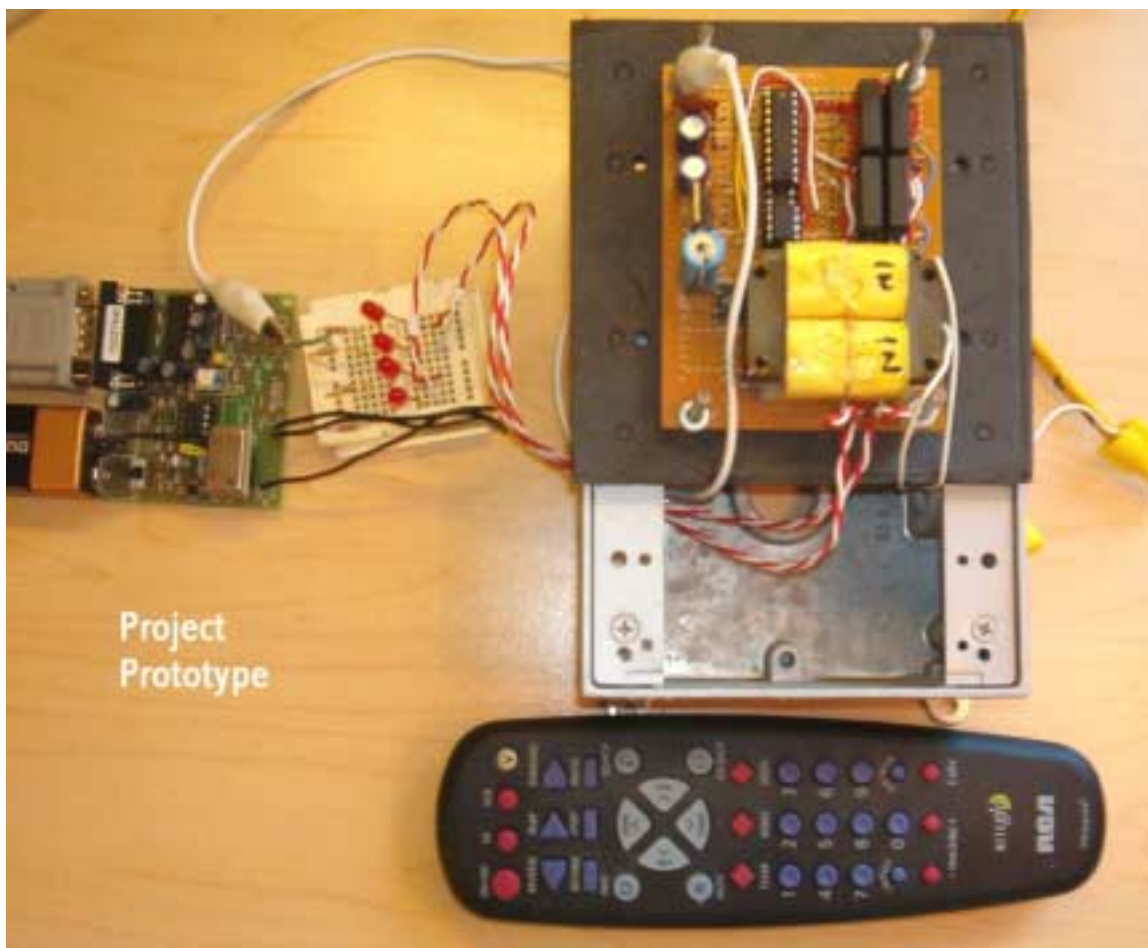


Photo 1 – Prototype Overview 1

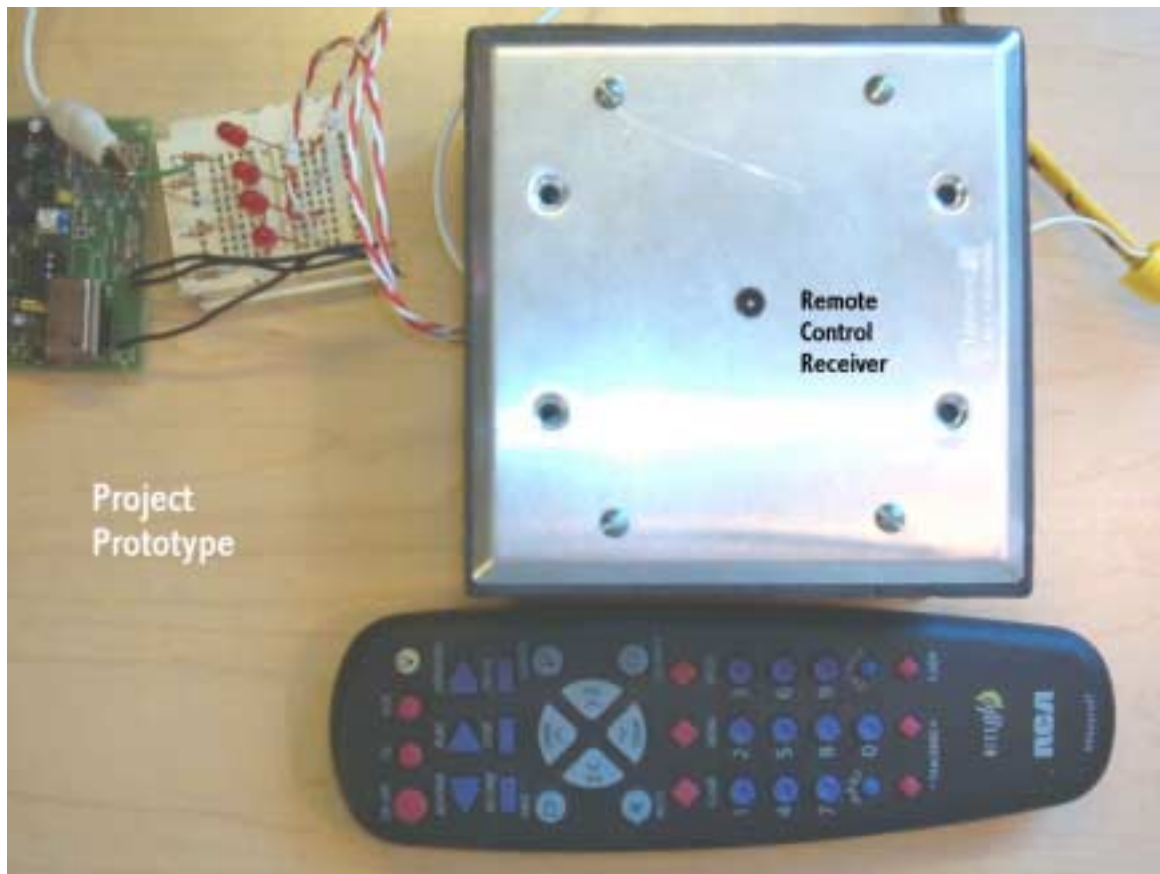


Photo 2 – Project Overview 2