Sorry, the IR goggles still not working, but we can go ahead.

InfraRed is a radiation frequency below our eyes sensitivity, so we can not see it, by the same way we don't see sounds.

Even that we can not "see sound frequencies", we know that it exist, because we can hear it.

Even that we can not see or hear infrared, we can feel it at our skin temperature sensors.
Just go close to fire and you will understand what I mean, because heat emits lots of infrared.

In real, this is the way we protect ourselves from fire, feeling the infrared in the sensors all over our skin.

Infra-Red is interesting, because it is easily generated and doesn't suffer electromagnetic interference, so it is nicely used to communication and control. Some other light emissions could contains infrared as well, and that can interfere in this communication, the sun is an example, since it emits a wide spectrum or radiation.

The adventure of using lots of infra-red in TV/VCR remote controls and other applications, brought infra-red diodes (emitter and receivers) at very low cost at the market.

From now on you should think as infrared as just a "red" light.

This light can means something to the receiver, on or off can transmit different meanings.

Lots of things can generate infrared, anything that radiate heat do it, including out body, lamps, stove, oven, even the hot water at the faucet.

To allow a good communication using infra-red, and avoid those "fake" signals, it is imperative to use a "key" that can tell the receiver what is the real data transmitted and what is fake. As an analogy, our eyes can catch a weak blinking light in middle of strong steady lights, isn't ?

Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Fake Not Good Fake Not Good Fake Not Good Good Fake Fake Fake Not Good Fake Not Good Fake Not Good Fake Not Good Fake Fake Fake Not Good Fake Not Good Fake

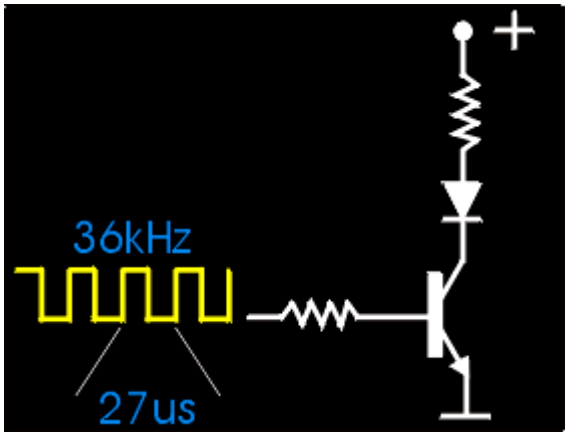
This is done using a pulsating infrared. The receiver will only look for this infrared frequency.
It is established that frequencies from 30 to 60kHz of pulsating infrared is a confortable way to transmit information in low speed.

This is why most of infrared remote controls use around 36 or 40kHz pulsating infrared.

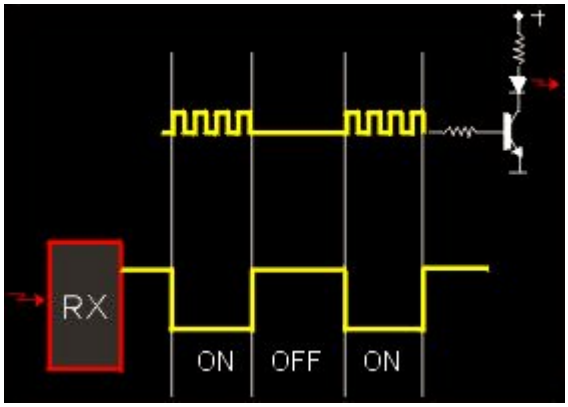
To generate a 40kHz pulsating infrared is quite easy, more difficult is to receive and identify this frequency. This is why some companies produce infrared receives, that contains the filters, decoding circuits and the output shaper, that delivers a square wave, meaning the existence or not of the 40kHz incoming pulsating infrared.

It means that those 3 dollars small units, have an output pin that goes high (+5V) when there is a pulsating 40kHz infrared in front of it, and zero volts when there is not this radiation.

A square wave of approximately 27uS (microseconds) injected at the base of a transistor, can drive an infrared LED to transmit this pulsating light wave. Upon its presence, the commercial receiver will switch its output to high level (+5V).



If you can turn on and off this frequency at the transmitter, your receiver's output will indicate when the transmitter is on or off.



Those IR demodulators have inverted logic at its output, when a burst of IR is sensed it drives its output to low level, meaning logic level = 1.

The TV, VCR, and Audio equipment manufacturers for long use infra-red at their remote controls. My first color TV in 1976 used an ultrasound (not infrared) remote control.

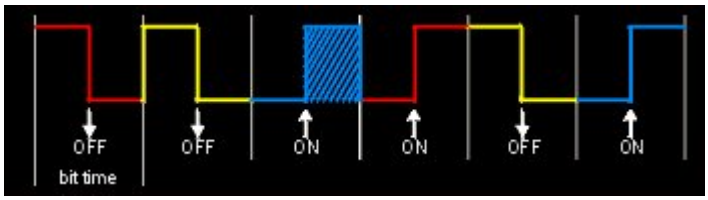
To avoid a Philips remote control to change channels in a Panasonic TV, they use different codifications at the infrared, even that all of them use basically the same transmitted frequency, from 36 to 50kHz. So, all of them use a different combination of bits or how to code the transmitted data to avoid interference.

Some standards were created. As illustrative material, we will only show one of them, the one used by Philips, even that we can cover the other ones in the future.

First of all, Philips adopted a standard that use fixed bit length and fixed quantity of bits.

Each time you press a button at the Philips remote control, it sends a train of 14 bits, 1.728ms per bit, the whole train is repeated every 130ms if you keep the button pressed

Each bit is sliced in two halves. The left and right half has opposed levels. If the bit to be transmitted is one (1), its left side is zero while its right side is one. If the bit to be transmitted is zero (0), its left side is one while the right side is zero.



(This is the right logic, reversed from what you can see at the IR receiver output.)

It means that the second half of the bit is actually the same meaning of the bit to be transmitted, as you can see at the shaded blue right side of the bit as on, means bit transmitted = 1.

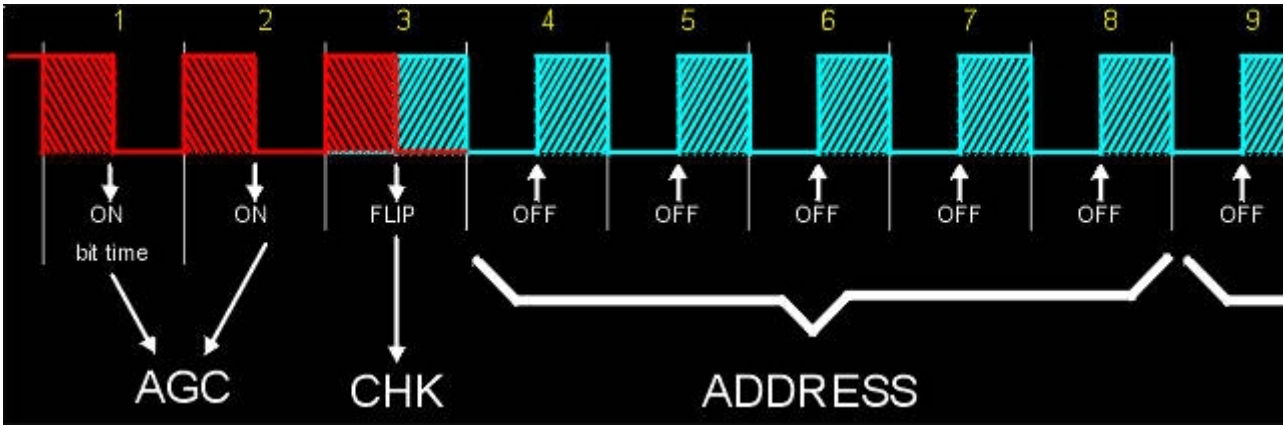
If you want to measure the correct logic level directly from the Receiver Output, you should measure at the first half of the bit.

The correct interpretation, is that it changes level exactly at the middle of bit time. At the IR Receiver output a bit Zero changes level from Low to Up, while a bit One changes level from Up to Low.

There are a minimum quantity of incoming 27μs pulses to the demodulator understand it is at the right frequency and then drop its output. The quantity of pulses used at the Philips remotes are 32 pulses per each half of the bit, 64 pulses per bit. So, a bit "0" to be transmitted it means 32 square pulses of 27μs each, then 32 x 27μs of silence. The bit "1" is the opposite, 32 x 27μs of silence followed by 32 square pulses of 27μs.

Our job here will be to decode the receiving of the waveform at the demodulator output. We could observe the direction of the changing at the middle of the bit, if going down, means bit 0, going up, means bit 1. But it is easy to sample the middle of the first half of the bit, so it directly tells us what is the bit state, as we will see next in this text.

The Philips remote control sends 14 bits in sequence as you can see below. (I am sorry to need to use more than your monitor width for the next picture, but it is to better view it).



(Figure 7)

You can see the 14 bits of the RC-5 system above. The RED bits are level "ON", while Blue are "OFF".

The first two bits, #1 and #2, are called *ACG* calibration. They are "ON" level, and serve to calibrate the IR Receivers Auto Gain Control.

In the Philips remotes, the bit #3 is the CHECK bit, every time you press a key at the remote, even pressing repeatedly the same key, this bit flips state. This feature is interesting. Suppose you pressed number "1" at the remote (trying to select channel 15 at TV) and holding it for 2 seconds, then your other hand just blocks the InfraRed signal. The TV would receive two trains of pulses, generated by your hand breaking a long train in two. Other systems would understand transmission of two keys "1" selecting channel "11", but this do not happens in the Philips system. This bit flips state every time you press a key, so blocking the signal with your hand doesn't change this bit, so the TV will understand that still the same key pressed. To select channel "11" you should press key "1" really twice.

The next 5 bits, #4 to #8, are used for SYSTEM ADDRESS, or to identify which kind of device should execute the COMMAND bits. For example, TV set uses ADDRESS ZERO. Bit #8 is the Less Significant Bit.

SYSTEM ADDRESS	EQUIPMENT
0	TV SET
2	TELETEXT
5	VIDEO RECORDER
7	EXPERIMENTAL
16	PREAMPLIFIER
17	RECEIVER / TUNER
18	TAPE / CASSETE RECORDER
19	EXPERIMENTAL

The next 6 bits, #9 to #14, are used for COMMAND information to the device selected at the ADDRESS bits. Bit #14 is the LESS SIGNIFICANT BIT, and it is last transmitted.

For example, STOP key uses COMMAND #54 (36h in hexadecimal), bits #9, #10, #12 and #13 should be ON, bits #11 and #14 should be OFF.

COMMAND (in decimal)	DESCRIPTION of FUNCTION
0-9	NUMERIC KEYS 0 - 9
12	STANDBY
13	MUTE
14	PRESETS
16	VOLUME UP
17	VOLUME DOWN
18	BRIGHTNESS +
19	BRIGHTNESS -
20	COLOR SATURATION +
21	COLOR SATURATION -
22	BASS UP
23	BASS DOWN
24	TREBLE +
25	TREBLE -
26	BALANCE RIGHT
27	BALANCE LEFT
48	PAUSE
50	FAST REVERSE
52	FAST FORWARD-
53	PLAY
54	STOP
55	RECORD
63	SYSTEM SELECT

This system can be automated if using the chips:
SAA3049 or TDA3048 for receivers and SAA3006, SAA3010 or SAA3027 as transmitters.

DECODING WITH A MICROCONTROLLER

To receive this signal using a microcontroller, follows the timing from the figure 7 above. Note that the Infrared Receiver invert the bit signal, low level means bit ON.

During inactivity (no Infrared present) the output of the Infrared receiver is UP (bit zero).

You can connect the IR receiver output to any input port pin or interrupt pin of your microcontroller, and keep polling it or prepare the interrupt routine to trigger your reading after the first low level sensed.

When you press a key at the remote, it transmits the train of pulses, and your microcontroller will receive bit #1 first. It will be sensed right after the middle of the bit when it changes from high to low level to means bit "1". This is the first time that your microcontroller will "see" the incoming IR signal.

You don't need to decode those first two bits, not even the CHK bit (except if you want to control as the TV do and described above), so you can skip those 3 bits and start to receive the ADDRESS bits. To do that, you need to skip 2.75 bits time, and you will be exactly at the middle of the right level of the first ADDRESS bit to be read (non inverted level).

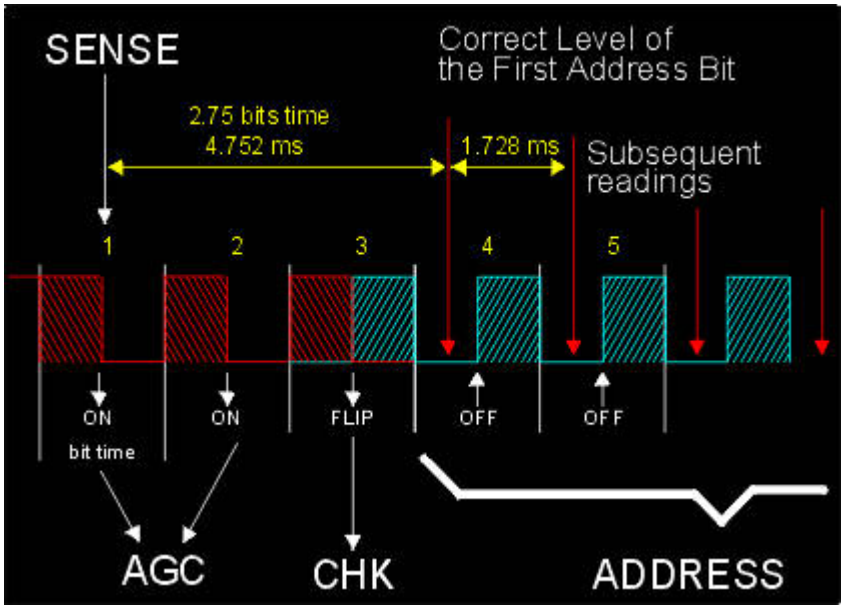


Figure 8

So, upon sensing the first low level, your software should wait 4.752 milliseconds and then start to read the next 11 bits spaced 1.728ms each. The first 5 bits are Address and the next 6 bits are Command, logic correct level, LOW = 0, HIGH = 1.

To make sure your software is waiting the correct timing, you need to use a dual channel oscilloscope, and this procedure to adjust your software:

At your bit reading routine use an available microcontroller port pin and generate a fast pulse UP and DOWN, then use one scope channel to display this pulse, and the other scope channel to show the incoming signal from the receiver. Press and hold key number ZERO at the remote, and sync the scope to show a complete wave form, don't worry with timing. The fast 11 pulses should always be in place of those RED down arrows at the figure 8 above. It means that the "bit reading" software routine will reading exactly in the middle of the correct bit level.

Your software will need to have two timing delays, the first to wait 4.752ms and the second to wait 1.728ms. Adjust the timing loop from the 4.752ms until the first fast pulse happens exactly as indicated above. Then adjust the 1.728 ms timing delay in such way that the last fast pulse (#11) bit reading happens exactly at the middle of the low part of the last bit (#14).

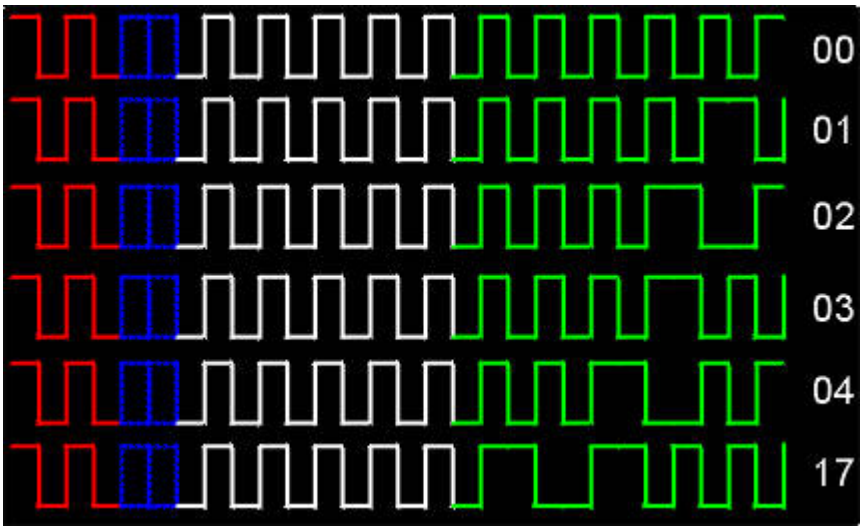
Check all other bits and fast pulses, they should be all matching ok. Small errors would be accepted since the reading would be happening in the middle of the bit, few errors for more or less is not a problem, but it is better to be the most possible in the middle of the low level of each bit. This is why you should adjust your 1.728ms timing routine looking at the last data bit and fast pulse, if they match somehow ok, all the other bits should be ok too.

Remember that any other remote key will generate a different pattern and it can confuse you. Use always key number ZERO for this software calibration.

Once you find the correct timing delays, you can replace the FAST pulse instructions with NOPS (check your chip instruction set to keep the same clock count wasted), or keep the fast pulse there just for fun, so you will be able to recheck it in case of problems.

Reading the 11 bits is easy. Just shift them left into a 8 bits register and ignore the high order 2 bits #7 and #6 (AND 03Fh instruction), keep only the COMMAND last 6 bits... You will not want to decode the ADDRESS bits, are you? The TV remote control will always send Address Zero, you know that, right?

Here few examples of the complete waveform (14 bits) at the Receiver Output:



Values at the right are the command in hexadecimal
Red: AGC pulses (ON)
Blue: Check bit (flipping)
White: Address (00)
Green: Command

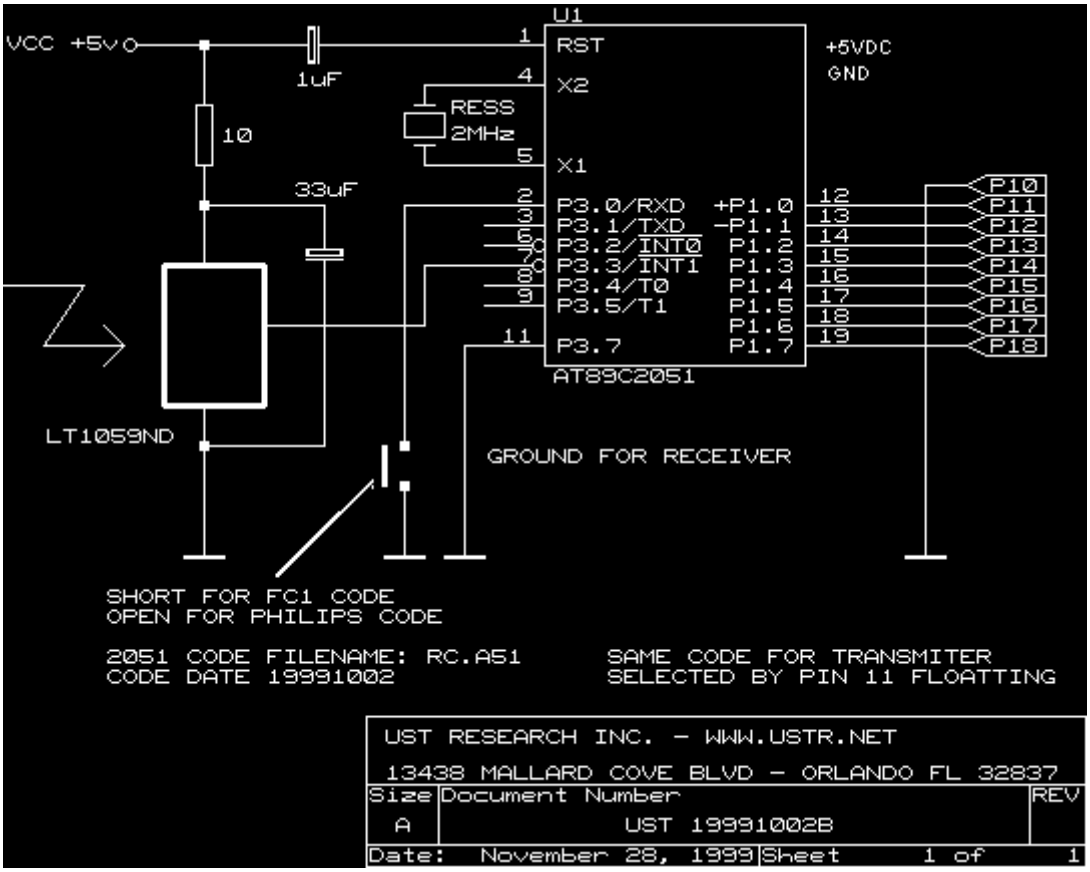
SAFETY

If you want to include some safety (recommended), check bits #7 and #6 for ZEROS, if it is a Philips TV remote. You can also not use the 4.752ms delay, instead, wait only 3/4 of a bit time, and then start to read from the bit #2. You will suppose the first one was a "1" and the second should be a "1" too, if not, discard them, wait 60ms and activate reception again, you should sync correctly at the beginning of the next train of pulses. But even this way, you could read wrongly a "1" bit followed by another if the reading start anywhere in the middle of the train.

To make sure your controller is never starting to read the train in the middle, you can discard the first reading always after a long silence (half second or more), so you will make sure that the second reading will be in sync. To do that, after idling for more than half second, enter in a special sync routine and upon sensing the first LOW level bit (can be the real first or not), wait 60ms and jump to the real receive routine and start to monitor the Receiver output again. It means that the first train will be used just to makey your receive routine to engage in a correct timming to read the second train of pulses.

Remember that there is a delay of 105ms between pulse trains, so doesn't matter which LOW bit you sensed, waiting 60ms will put your reception routine in the middle of the silence between pulse trains, allowing it to sync at the first bit of the next train of pulses.

AT89C2051 IR RECEIVER



8051 ASSEMBLER CODE

```
;  ÛÛÛÛÛÛÛÛ     ÜÛÛÛÛÛÛÜ            ÛÛÛÛÛÛÛÛÛ
;  ÛÛ    ßÜÛ   ÛÛß    ßÛ           ÛÛ
;  ÛÛ    ÜÜÜ   ÛÛ                   ÛÛ
;  ÛÛÛÛÛÛÛÛ    ÛÛ                   ÛÛÛÛÛÛÛÛÜ
;  ÛÛ   ÛÛ     ÛÛ                        ßÛÛ
;  ÛÛ    ÛÛ    ÛÛÜ    ÜÛ           ÛÛÜ   ÜÛÛ
;  ÛÛ     ÛÛ    ßÛÛÛÛÛÛß            ßÛÛÛÛÛÛß
;
; SOFTWARE FOR AT89C2051 TO DECODE PHILIPS REMOTE CONTROL
; RC-5 CODIFICATION - INFRA-RED RECEIVER LT1029-ND DIGIKEY
; CLIPPED PARTS FROM THE ORIGINAL SOURCE TEXT:
;   A) TRANSMITTER
;   B) SELECTION FOR PHILIPS / FC1 - (WORKING PHILIPS)
; AT89C2051 RUNNING AT 2MhZ <== IMPORTANT
; WAGNER LIPNHARSKI - NOVEMBER 1999 - WAGNER@USTR.NET
; UST RESEARCH INC. - ORLANDO, FL - WWW.USTR.NET
;
$MOD51
$NOPAGING
$LIST
;
; PIN 11 (P37) : 0 = RECEIVER CIRCUIT
;              : 1 = TRANSMITTER CIRCUIT
;
; PIN  2 (P30) : 1 = RECEIVER DECODES PHILIPS REMOTE
;              : 0 = RECEIVER DECODES FC1 REMOTE
;
; PIN 12 (P10) : PULLED UP = FC1 CODE TRANSMITTER
;              : GROUND    = PHILIPS CODE TRANSMITTER
;
; RECEIVER CONFIGURATION:
; -----------------------
; PIN 7 (P33) INPUT SIGNAL FROM IR RECEIVER
```

```
; PINS 12-19 DECODER OUTPUT (DRIVE TO GROUND)
; PIN 11 TO GROUND.
; CLOCK CERAMIC RESSONATOR 2MhZ
; RESET CAP 1uF TO +VCC
; PIN 9 (P35) CAN BE USED TO LITE A LED WHEN RECEIVING VALID IR
; PIN 8 (P34) SCOPE GATE TIME (SOFTWARE TIMMING ADJUSTMENT)
;
; ---___---___------_______---___---___--- PIN 7 P33 IRDATA
; _|_____|_____|_____|_____|_____|_____|_ PIN 8 P34 SCOPE READ GATE TIME
;
; --__  1 BIT
;             RX DATA
; __--  0 BIT
;

; TRANSMITTER CONFIGURATION:
; --------------------------
; PINS 2,3,6,7,8,9 & 11 TIED TOGETHER TO DRIVE IR LED
; PINS 16-19 KEYBOARD MATRIX DRIVE (4 PINS)
; PINS 14 & 15 KEYBOARD MATRIX RETURN WITH PULL UP
; CRYSTAL 6MHZ
; RESET CONTROLLED BY FET BS110, DIODES FROM KEYBOARD.
; PIN 12 GOES TO GROUND WHEN IN POWER OFF

            ;------------------
            ; Receiver
            ;------------------
Led         BIT   0B5h   ; P3.5
IrInput     BIT   0B3h   ; P3.3
Scope       BIT   0B4h   ; P3.4
RxTx        BIT   0B7h   ; P3.7 ; Grounded if Receiver/High if TX
            ;------------------
            ; Transmitter
            ;------------------
TXLED       EQU   P3     ; FF = Off, 00 = On
KO1         Bit   097h   ; P1.7 Keyboard Output
KO2         Bit   096h   ; P1.6
KO3         Bit   095h   ; P1.5
KO4         Bit   094h   ; P1.4
KI1         Bit   093h   ; P1.3 Keyboard Input
KI2         Bit   092h   ; P1.2

IRDATA     DATA   010H   ; Memory to store IR code received

;=====================================;
;
; P H I L I P S     R E M O T E
; CODE "10" AT AV6 UNIVERSAL PROGRAMMABLE REMOTE CONTROL
;
;=====================================;
ORG 00H
MAIN:       Mov  SP,#SPVALUE          ;

PH0:        Mov  P1,#0FFh             ; Reset Decoder Output High
            Setb LED                  ; Turn off IR Indicator
            ;                         ;
PH1:        Mov  4,#022               ; 2MHz
            Mov  5,#000h              ; 2MHz Time loop
PH2:        Jnb  IrInput,PH3          ; Wait for Incomming IR (first low)
            Djnz 5,PH2                ; This time loop is to keep the last
            Djnz 4,PH2                ; ..pressed key available at P1 for few
            Jmp  PH0                  ; ..time in case you press it again
                                      ; ..and avoid glitches at P1.
            ;-------------------------;
PH3:        Clr  A                    ; IR Receiver First Low level.
            Clr  Led                  ; Lite IR Indicator
            ;                         ;
            Mov  2,#110               ; Time Loop (3/4 bit time)
            Djnz 2,$                  ; Waste Time to sync second bit
```

```
            ;                       ; 2MHz = 1.296ms
            Mov  B,#7               ;
PH4:        Setb Scope              ; Scope Gate Indicator Pulse UP
            Mov  C,IrInput          ; Shift IR state to Carry bit
            Clr  Scope              ; Scope Gate Indicator Pulse Down
            Rlc  A                  ; Insert LEFT Carry Bit into A Reg
            ;                       ;
            Mov  2,#145             ; Waste time for next BIT
            Djnz 2,$                ; 2MHz clock, 1.728ms
            ;                       ;
            Djnz B,PH4              ; Rotate 7 bits into A
            ;                       ; AGC#2, CHKbit, ADDRESS field (5 bits)
            ;-----------------------;
            Anl  A,#0F7h            ; Cut off CHECK Bit (Flipping bit)
            Cjne A,#40h,PH5         ; 10h means second AGC UP & Address 00
            ;                       ; Not 10h?, wrong sync, go again (PH5)
            ;-----------------------;
            Mov  B,#6               ; Will Get the COMMAND field
PH4A:       Setb Scope              ; Scope Gate Indicator Pulse UP
            Mov  C,IrInput          ; Shift IR state into Carry bit
            Clr  Scope              ; Scope Gate Indicator Pulse Down
            Rlc  A                  ; Insert LEFT carry bit into A Reg
            ;                       ;
            Mov  2,#145             ; Waste time for next BIT
            Djnz 2,$                ; 2MHZ clock = 1.728ms
            ;                       ;
            Djnz B,PH4A             ; Rotate 8 bits into A (Command)
            ;                       ; Here A holds Command bits
            ;-----------------------;
            Mov  IRData,A           ; Save Command at IRData memory
            Cjne A,#25h,$+3         ; Is Command >=25h
            Jnc  PH5                ; Yes? Wrong everything, go again PH5
            ;                       ;
            Call PH6                ; Go get Conversion at Table
            Cjne A,#0FFh,PH4B       ; If conversion <> FF, is valid.
            Jmp  PH5                ; ..Not Expected Command. go again PH5
            ;                       ;
PH4B:       Mov  P1,A               ; Valid Command is post at P1 Port
            Jmp  PH1                ; Job done, go wait next pulse train.
            ;                       ;
            ;-----------------------;
PH5:        Mov  3,#8               ; 2MHz
            Mov  2,#112             ; Wait 15 bit time
PH5A:       Djnz 2,$                ; Wasting time to sync ok next train
            Djnz 3,PH5A             ;
            Jmp  PH0                ; Go back and wait for next train.
            ;-----------------------;
PH6:        Inc  A                  ; Table conversion, Inc skip RET
            Movc A,@A+PC            ; Point to TABLE + Value at A (1-255)
            Ret                     ;
            ;-----------------------;
            ;  The following table bit values at the first column, means
            ;  the translation from remote control key to the value to be
            ;  be post at P1.  The program ignores values of "FF".
            ;  Change those bits according to your needs.
            ;
            ;  Bit #7 is "0" when a valid key is pressed, otherwise is "1"
            ;  It can be used to recognize values as key "zero" pressed,
            ;  when all bits UP means command ZERO.
            ;
            ;  This TV remote uses only Address = "00"
            ;
            ;    PHILIPS TV REMOTE CONTROL DECODING
            ;
            ;    VALUE TO P1   REMOTE KEY   COMMAND CODE FIELD (HEX)
            ;    -----------   ----------   ------------------------
TABLE:      DB   01111111b    ; 0         ; 0
            DB   01111110b    ; 1         ; 1
            DB   01111101b    ; 2         ; 2
            DB   01111100b    ; 3         ; 3
            DB   01111011b    ; 4         ; 4
```

```
                DB   01111010b    ; 5         ; 5
                DB   01111001b    ; 6         ; 6
                DB   01111000b    ; 7         ; 7
                DB   01110111b    ; 8         ; 8
                DB   01110110b    ; 9         ; 9
                DB   11111111b    ;           ; A
                DB   11111111b    ;           ; B
                DB   01110011b    ; ON/OFF    ; C
                DB   01110010b    ; MUTE      ; D
                DB   01110001b    ; PP        ; E
                DB   01110000b    ; OSD       ; F
                DB   01101111b    ; Volume+   ; 10
                DB   01101110b    ; Volume-   ; 11
                DB   01101101b    ; Bright+   ; 12
                DB   01101100b    ; Bright-   ; 13
                DB   01101011b    ; Color+    ; 14
                DB   01111010b    ; Color-    ; 15
                DB   11111111b    ;           ; 16
                DB   11111111b    ;           ; 17
                DB   11111111b    ;           ; 18
                DB   11111111b    ;           ; 19
                DB   11111111b    ;           ; 1A
                DB   11111111b    ;           ; 1B
                DB   01100011b    ; Contrast+ ; 1C
                DB   01100010b    ; Contrast- ; 1D
                DB   11111111b    ;           ; 1E
                DB   11111111b    ;           ; 1F
                DB   01011111b    ; Program+  ; 20
                DB   01011110b    ; Program-  ; 21
                DB   11111111b    ;           ; 22
                DB   11111111b    ;           ; 23
                DB   01011011b    ; Timer     ; 24
                DB   01010111b    ; Special 1 ; 25
                DB   01000001b    ; Special 2 ; 26
                DB   01000111b    ; Special 3 ; 27
                DB   01001110b    ; Special 4 ; 28
                DB   01000101b    ; Special 5 ; 29
                DB   01010010b    ; Special 6 ; 2A
END
```

If you want to post here your hardware/software solution with a PIC, email me.

I am sorry, English was not reviewed, should have zillions of errors. No time to do that this week. Money is more important!!! :)

Good Luck.