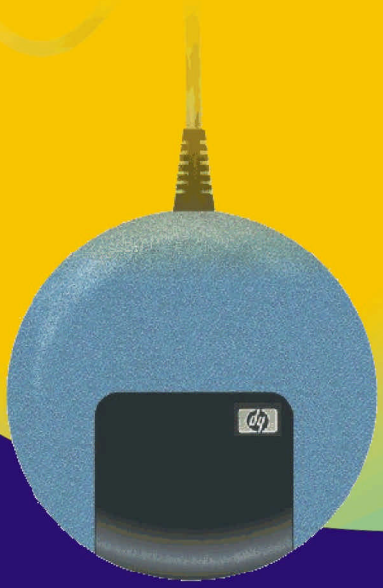
IR Adapter for HP49G and PC

Drawings are intended as impression only

Drawn by Marcel Flipse

MIND DESIGN

Extending Opportunities



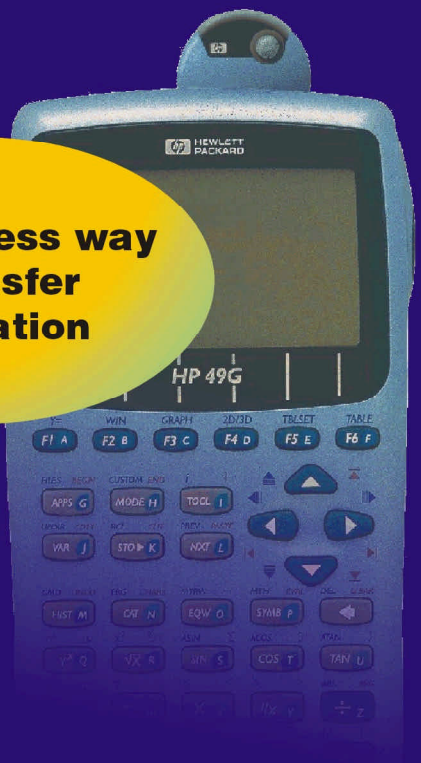
Graphing Calculator

IR Adapter Kit

For HP49G

- HP48G series compatible
- HP82240B IR Printer support
- Hi speed 9600 baud mode

The wIReless way to transfer information



HP49G IR adapter V1.2

Well... here it is. The IR adapter document. I promised many people to finish it, but it took quite a while after all. I hope there are not a lot of errors in it, but if you find any, let me know. Improvements are also welcome. The images shown, are just for fun. I'm sorry HP, for using your box layout.

One of the future 'improvements' I intend to implement (may take another year :-) is a discrete LED and photodiode, instead of the IRDA transceiver. The IRDA transceiver I used has become obsolete, but an IRMS6118 can be used instead. The IR printer format is not described in the document yet. (V1.3 ?)

I can send gerberfiles and sourcecode in TXT format, to those who are interested. Feel free to modify the code, schematics or layout.

Don't fry your calculator. If rumors are true, the HP49G will soon be obsolete as well. So take care....

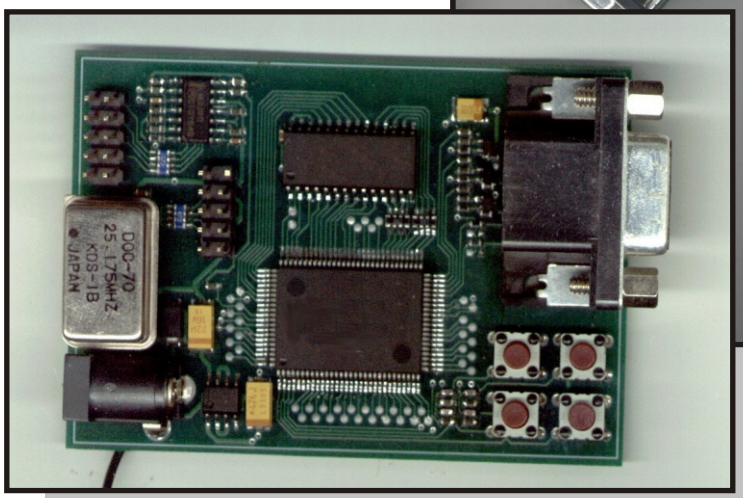
Good luck,

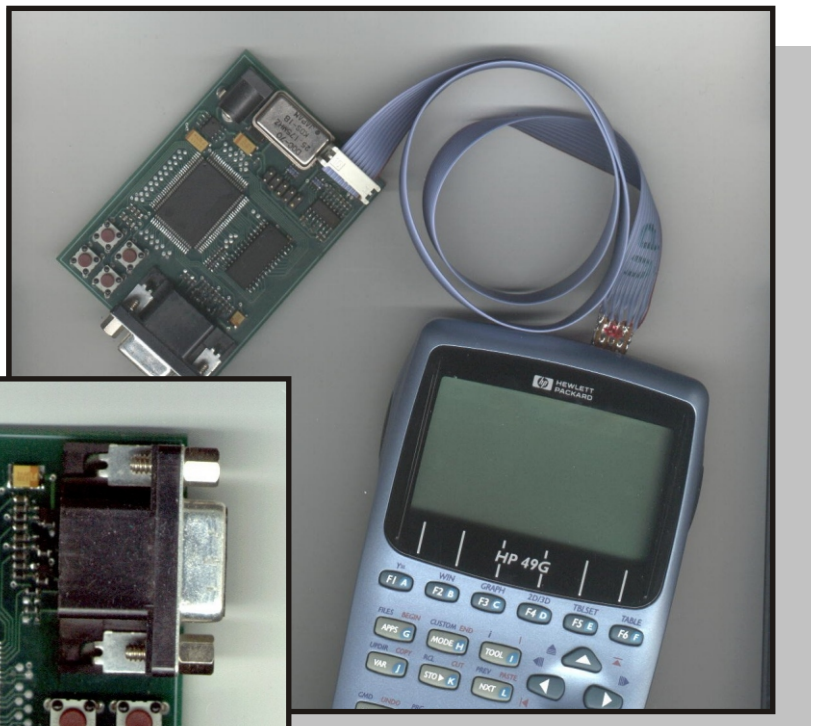
Marcel Flipse

P.S.: Never mind the handsome man on the box...
P.P.S. Oh, and Steve (my technical backup), please don't be to hard on me, when you find a bug. Even HP is known to make errors.

marcel@minddesign.nl
mflipse@zonnet.nl

Next project: a HP VGA card.....





Specifications HP49G IR adapter

HP48G series compatible IR mode at 2400 baud
9600 baud mode for HP49G to HP49G or PC with IR adapter
HP82240B infrared mode
Automatic powerup and powerdown
Non volatile memory to store last used mode

Description
The HP49G adapter converts serial data to infrared data.Four different modes can be selected, to comply with the HP48G series IR format, the HP82240B printer, inter HP49G communications or direct control operation. The adapter will start up with the mode that is last used. Therefore selecting a mode is only necessary if it is different from the last one used.

Communication mode
HP48 or HP49 mode does not require additional software on the calculator. Every transmitted or received byte is sent through the adapter with a maximum delay of a half bit time. The appropriate speed of 2400 or 9600 baud must be selected on the HP49G as well.

Printer mode
Printer mode requires a little additional software on the calculator. Between transmitted bytes a little pause is needed and after every printed line the software should wait for the printer to reposition the thermal head. The speed on the HP49G must be set at 9600 baud.

Direct control mode
In direct control mode the state of the TX pin is fed straight to the IR led. This mode can be used to implement remote control operation. The state of the TX pin can be controlled by the RCS register of the Yorke. Maximum pulse width is limited to 70us for protection.

Power consumption
After powering up the HP49G, the adapter consumes virtually no power. Opening the serial port by typing OPENIO or by accessing the serial port, will power up the adapter. Closing the port by typing CLOSEIO or turning the HP49G off, will shutdown the adapter.

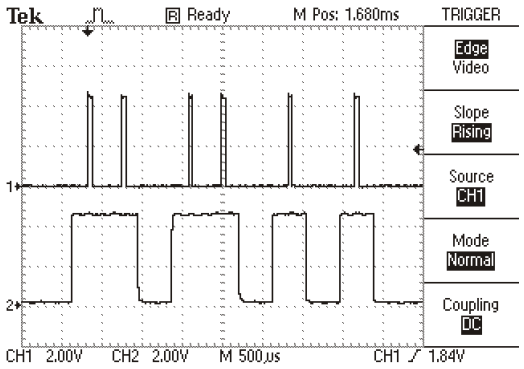
Selecting the mode

The mode is selected by holding down the button, while opening the serial port with OPENIO. The adapter switches to programming mode.
Sending a "0" over the serial port will put the adapter in 2400 baud mode. The adapter echoes the "0" and stores the mode in non-volatile memory. 9600 Baud, printer mode or direct mode are selected the same way by sending a "1", "2" or a "3" respectively.
To request the currently selected mode, send a "?" and the adapter will respond with "0", "1", "2" or "3".
Sending a "R" (reset) or closing the port and re-opening it will put the adapter back in operating mode.
Any other character sent, will be replied by a "X".

Speed	HP48 mode	2400	baud
Speed	HP49 mode	9600	baud
HP82240B	IR side	1170	baud
	HP49 side	9600	baud
Pulse length	2400 baud	52	us
	9600 baud	13	us
	modulation printer IR	32	kHz
Wavelength peak		870	nm
Range		0.5	m
Receiver recovery time max.		800	us

Specifications HP49G IR adapter

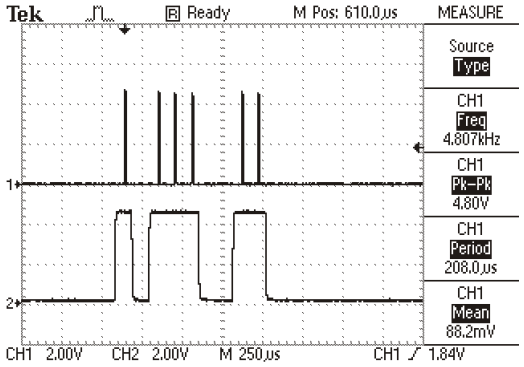
2400 Baud (mode 0)
The scope image below shows the 2400 baud mode.



Mode 0

The upper trace is the output pin to the IRDA transceiver. The lower trace is the HP49G transmit pin. Zeros (high level on trace!) are replaced by infrared pulses of 1/8th bit length.

9600 Baud (mode 1)
This mode only differs in speed.

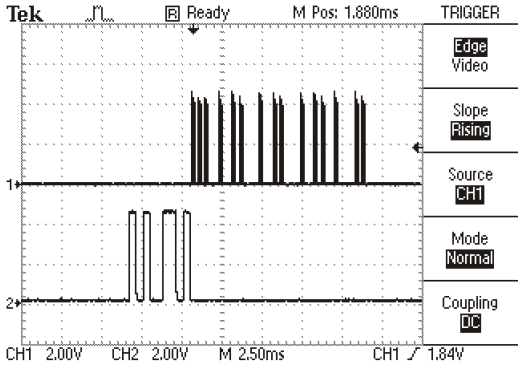


Mode 1

In this picture a different character is sent then the one shown in 2400 baud mode.

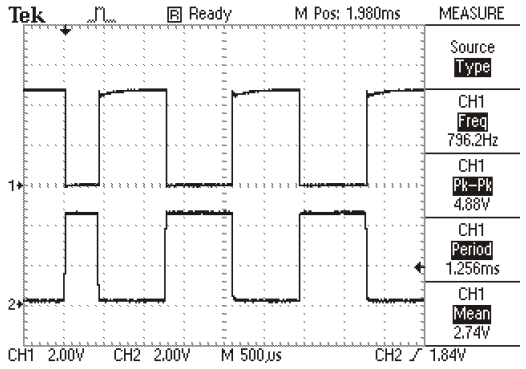
Printer (mode 2)
Printer mode differs a lot from 'normal' communication mode. Each byte sent by the HP49G has to be read completely by the adapter first, before it can be converted to the IR mode used by the printer. This can be seen in the captured scope image. When writing software that uses the printer mode, make use you leave 14 ms between sent bytes. When the adapter is sending the IR bit stream, it has to time to receive a byte at the same time.

Information on the IR format can be found on the web



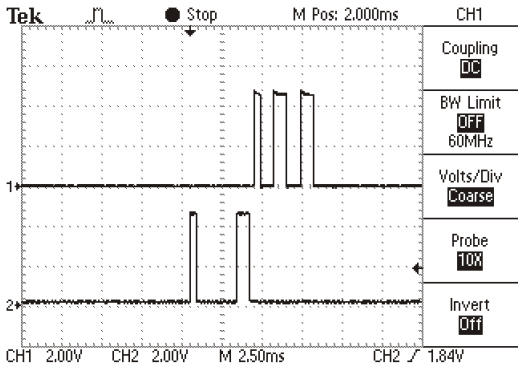
Mode 2

Direct (mode 3)
In this mode the status of the transmit pin is copied to the IRDA transceiver. Note that the traces are each others inverse. This mode may be used by remote control software.

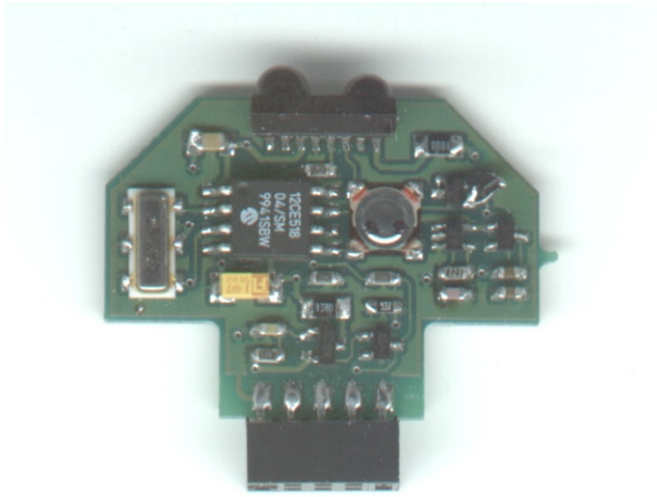


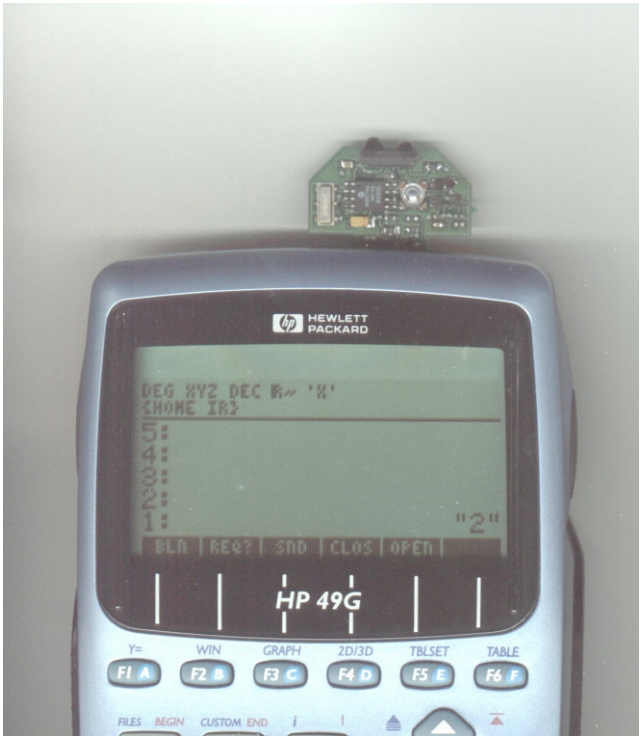
Mode 3

Changing the mode
Here you can see a question from the HP49G << "?" XMIT >> and the reply from the adapter.

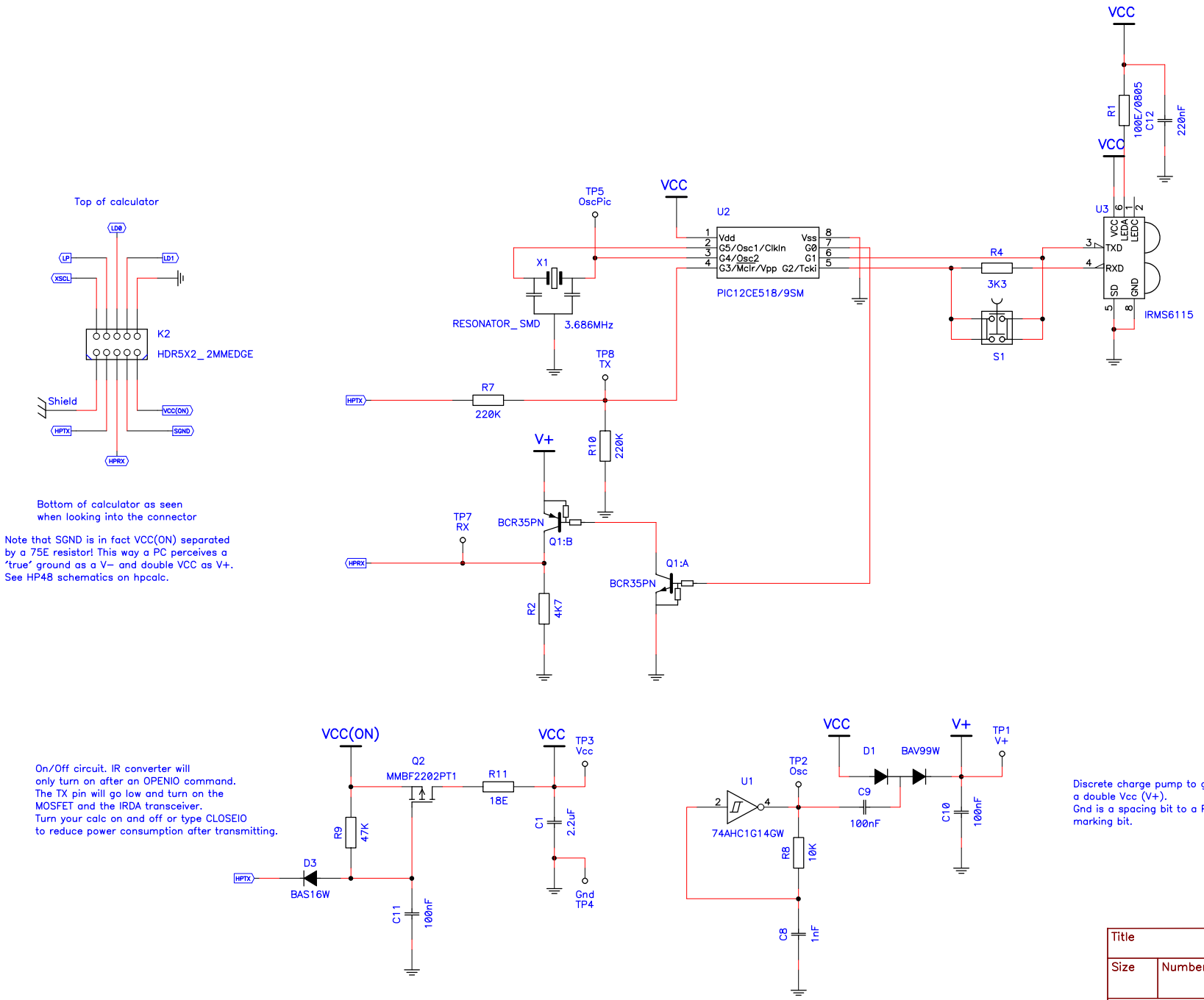


Some piccies

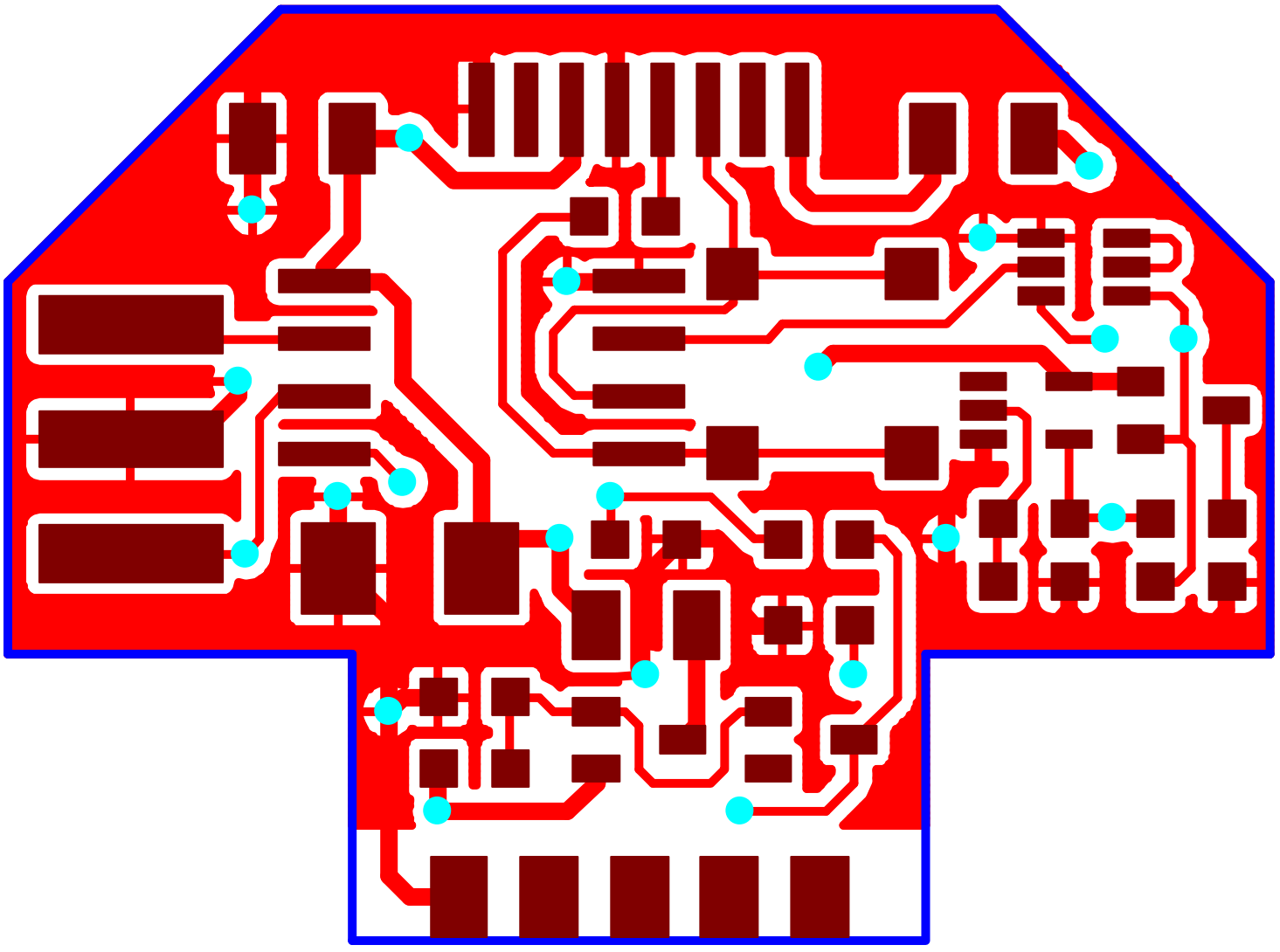


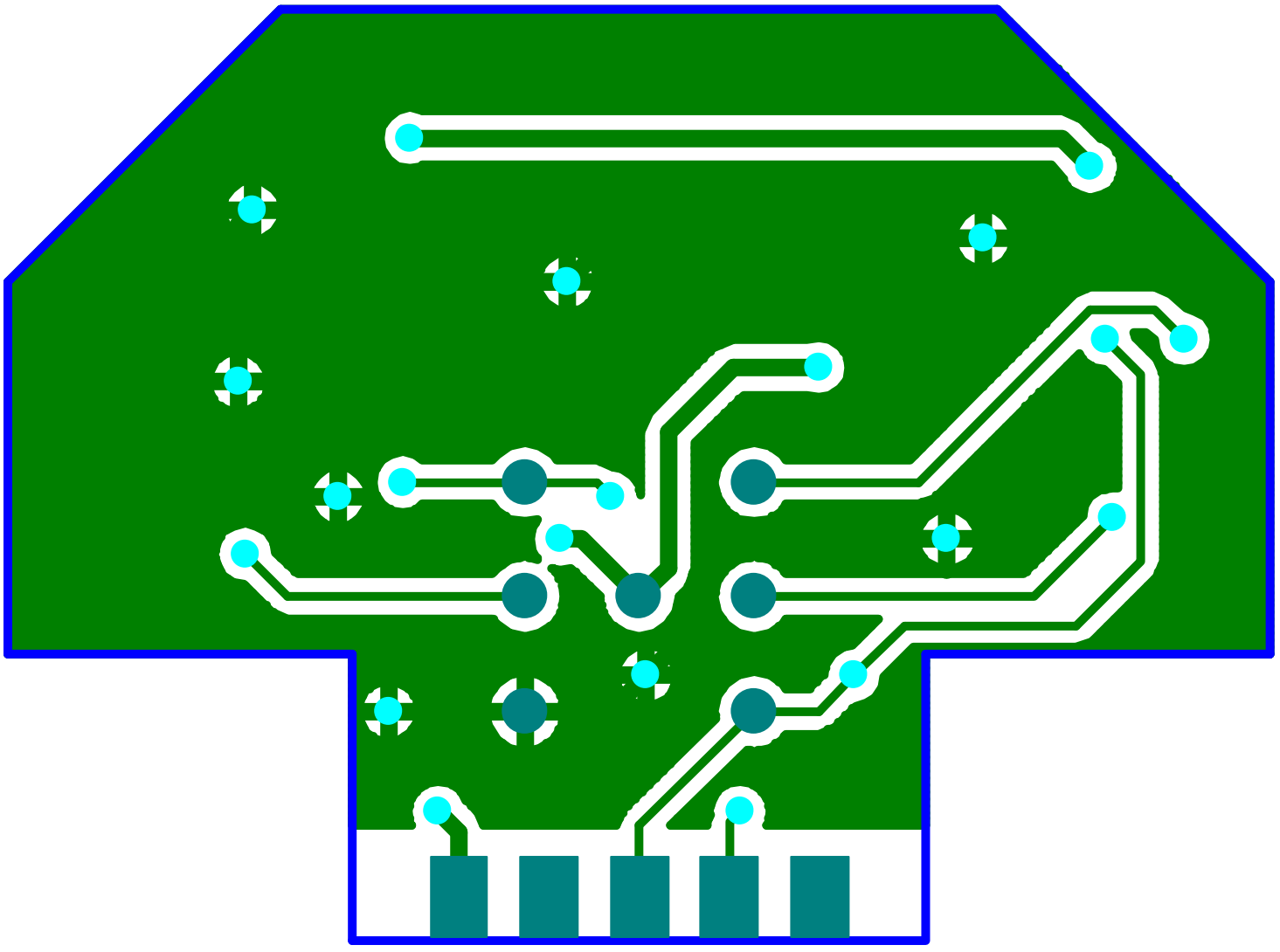


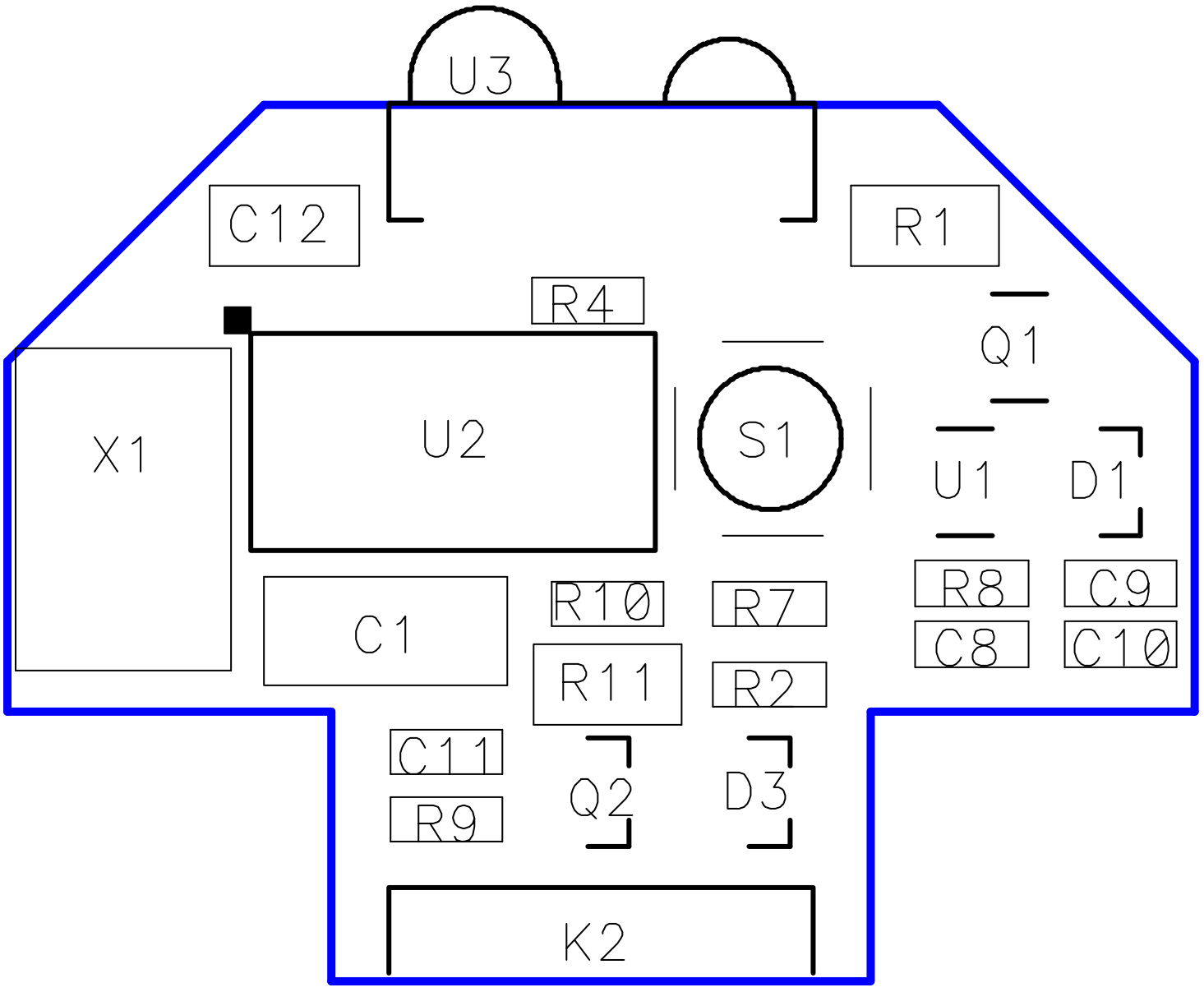
Finally on ‘speaking terms’


Top of calculator
LD0
LP
LD1
XSCL
K2
HDR5X2_2MMEDGE
Shield
VCC(ON)
HPTX
SGND
HPRX
Bottom of calculator as seen
when looking into the connector
Note that SGND is in fact VCC(ON) separated
by a 75E resistor! This way a PC perceives a
'true' ground as a V– and double VCC as V+.
See HP48 schematics on hpcalc.
TP5
OscPic
VCC
U2
Vdd
G5/Osc1/Clkln
G4/Osc2
G3/Mclr/Vpp
Vss
G0
G1
G2/Tcki
PIC12CE518/9SM
X1
RESONATOR_SMD
3.686MHz
TP8
TX
R7
220K
HPTX
R10
220K
V+
TP7
RX
BCR35PN
Q1:B
HPRX
Q1:A
BCR35PN
R2
4K7
R4
3K7
S1
VCC
R1
100E/0805
C12
220nF
VCC
U3
VCC
LEDA
LEDC
TXD
RXD
SD
GND
IRMS6115
On/Off circuit. IR converter will
only turn on after an OPENIO command.
The TX pin will go low and turn on the
MOSFET and the IRDA transceiver.
Turn your calc on and off or type CLOSEIO
to reduce power consumption after transmitting.
VCC(ON)
Q2
MMBF2202PT1
R11
18E
VCC
TP3
Vcc
R9
47K
C1
2.2uF
HPTX
D3
BAS16W
C11
100nF
Gnd
TP4
VCC
TP2
Osc
U1
74AHC1G14GW
D1
BAV99W
V+
TP1
V+
C9
100nF
R8
10K
C10
100nF
C8
1nF
Discrete charge pump to generate
a double Vcc (V+).
Gnd is a spacing bit to a PC and V+ a
marking bit.
Title
HP49G IR adapter
Size
Number
---hob---
Rev
1.1
Date
Wed May 30, 2001
Drawn by
M. Flipse
Filename
Hp49ir3.sch
Sheet
1
of
1

Count	ComponentName	RefDes	PatternName	Value	Description
1	74AHC1G14GW	U1	P_SOT353		Single Gate Schm. Inverter
1	B3B_SPNO	S1	P_B3B_SPNO		Switch
1	BAS16W	D3	P_SOT323		Single Diode
1	BAV99W	D1	P_SOT323		Dual Diode
1	BCR35PN	Q1	P_SOT363		Dual Digital NPN/PNP Transistor
1	HDR5X2_2MMEDGE	K2	P_5X2_2MMEDGE		Double Row Socket 5x2 (2mm pitch)
1	IRMS6115	U3	P_IRMS6115		IRDA transceiver (Obsolete, use IRMS6118)
1	MMBF2202PT1	Q2	P_SOT323		Mosfet
1	NPCAP_0603	C8	P_0603	1nF	MLC Capacitor
3	NPCAP_0603	C9 C10 C11	P_0603	100nF	MLC Capacitor
1	NPCAP_0805	C12	P_0805	220nF	MLC Capacitor
1	NPCAP_1206	C1	P_1206	2.2uF	MLC Capacitor or Size A Tantalum Capacitor
1	PIC12CE518/9SM	U2	P_SOL8		Microchip Microcontroller
1	RESONATOR_SMD	X1	P_RESONATOR	3.686MHz	CSTCC Resonator
1	RES_0603	R4	P_0603	3K3	Resistor
1	RES_0603	R2	P_0603	4K7	Resistor
1	RES_0603	R8	P_0603	10K	Resistor
1	RES_0603	R9	P_0603	47K	Resistor
2	RES_0603	R7 R10	P_0603	330K	Resistor
1	RES_0805	R11	P_0805	18E	Resistor
1	RES_0805	R1	P_0805	100E	Resistor
1	TSTPOINT	TP4	P_TSTPOINT	Gnd	Testpoint
1	TSTPOINT	TP2	P_TSTPOINT	Osc	Testpoint
1	TSTPOINT	TP5	P_TSTPOINT	OscPic	Testpoint
1	TSTPOINT	TP7	P_TSTPOINT	RX	Testpoint
1	TSTPOINT	TP8	P_TSTPOINT	TX	Testpoint
1	TSTPOINT	TP1	P_TSTPOINT	V+	Testpoint
1	TSTPOINT	TP3	P_TSTPOINT	Vcc	Testpoint






U3
C12
R1
R4
Q1
X1
U2
S1
U1
D1
R8
C9
C1
R10
R7
C8
C10
R11
R2
C11
Q2
D3
R9
K2

```
;*******************************************************
;* file: hpir.asm                                      *
;* auth: m. flipse                                     *
;*                                                     *
;*            "USE THE SOURCE, LUKE"                   *
;*                                                     *
;* date: 21-11-1999 ver1.0                             *
;*        initial release                              *
;* date: 19-01-2000 ver 1.1                            *
;*        added 9600 baud support for 49 <-> 49 only   *
;* date: 05-02-2000                                    *
;*        added printer functionality                  *
;* date: 25-10-2000                                    *
;*        adjusted timing for 3.6864 MHz resonator     *
;*        instead of 2.45 MHz (SMD availability problem)*
;*                                                     *
;* serial format                                       *
;*    ___ ___ ___ ___ ___ ___ ___ ___ ___              *
;* __/   x___x___x___x___x___x___x___x___.___.____     *
;*                                                     *
;*     s   0   1   2   3   4   5   6   7   p   p       *
;*                                                     *
;* s    one startbit                                   *
;* 0..7 8 databits, lsb first, a '0' is a positive     *
;*      voltage and a '1' negative                     *
;* p    stopbits, at least 2                           *
;*                                                     *
;* At 2400 baud, each bit is 1/2400 = 416.6 us long,   *
;*   which is 384 instructions                         *
;* At 9600 baud, each bit is 104.2 us long: 96 instr.  *
;*                                                     *
;* ir format                                           *
;*                                                     *
;*                                                     *
;* ___|___|___|___|___|___|___|___|___|___.___._______ *
;*                                                     *
;*    s   0   1   2   3   4   5   6   7   p   p        *
;*                                                     *
;* s    one startbit; '0'                              *
;* 0..7 8 databits, lsb first                          *
;*      | = 52us lightpulse if databit is zero         *
;* p    at least 2 stopbits; '1' (no light)            *
;*                                                     *
;* time between bits is 416.6 us like the serial       *
;* transmission, but bit length is 52 us (1/8 of       *
;* wire bit length and dito for 9600 baud)             *
;*                                                     *
;* external osc = 3.6864MHz: 921600 Instr/sec: (Wooow!) *
;*         Tinstr = 1.085us                            *
;*                                                     *
;* when transmitting at 2400:                          *
;* IR lighpulse   = 48 instructions                    *
;* total bit time = 384 instructions                   *
;*                                                     *
;* when transmitting at 9600 :                         *
;* IR lighpulse   = 12 instructions                    *
;* total bit time = 96 instructions                    *
;*                                                     *
;* and the printer format... that is a long story      *
;* ask Master Yoda if you want to know                 *
;*******************************************************

        list    p=12ce518       ; also used in the Millenium Falcon as
        radix   dec             ; hyperdrive controller

        ; remove this line if you are compiling for a 'true' CE518
        ;#define emulated        ; code has been developed with an icepic
                                 ; (C54 daughter board) and a piggy board
                                 ; with an 24LC01 connected to port A

        #include "p12ce518.inc" ; included with MPLAB
        #include "hp49vars.inc" ; what's in a name...

        __config _mclre_off & _cp_off & _wdt_off & _xt_osc ; hmmm, why should I code protect
anyway?
                                           ; you can read the source now...

        org     0               ; hyperdrive reset vector

reset
        goto    init            ; start your engines!

;*******************************************************
;* jumptables                                          *
;*******************************************************

jump_goto_mode
```

```
        goto    main_2400         ; mode 0: 2400 baud, hp48 compat.
        goto    main_9k6          ;      1: 9600 baud, 49<->49 only
        goto    main_printer      ;      2: printer mode
        goto    main_direct       ;      3: direct mode (for TV remote?)
        goto    main_progmode     ;      4: programming mode

jump_change_mode
        goto    set_lo_speed      ; well, low meaning 2400 baud
        goto    set_hi_speed      ; sub-light 9600 baud speed
        goto    set_printer_mode  ; wookie style printer mode
        goto    set_direct_mode   ; implemented this one, just in case

jump_i2c
        goto    init_address      ; pc offset=0, write control done, send address
        goto    init_write_data   ; pc offset=1, write address done, send data
        goto    stop_bit          ; pc offset=2, write done, send stop bit
        goto    init_address      ; pc offset=3, write control done, send address
        goto    init_read_control ; pc offset=4, send read control
        goto    read_bit_counter  ; pc offset=5, set counter and read byte
        goto    stop_bit          ; pc offset=6, random read done, send stop

;*******************************************************
;* subroutine delay                                    *
;*******************************************************

delay
        movwf   waitawhile      ; store 'wait' value

        decfsz  waitawhile      ; tombtiedomtiedom
        goto    $-1             ; yawn.......

        retlw   .0              ; ah, finished!

; end delay (3 * waitawhile + 4, including call and retlw)

;*******************************************************
;* subroutine ir_rcv, at 2400/9600 baud                *
;*******************************************************

ir_rcv
        movlw   .9              ; here's what we will do next:
        movwf   counter         ; receive 9 bits (incl. startbit)

        movf    delay_ir_start,w ; now wait until we are halfway the
        call    delay           ; startbit before sending anything

nextirbit
        movf    tmr0,f          ; copy onto itself
        skpnz                   ; to see if it is zero
        goto    ah_no_ir_bit    ; jump, nothing happened :-(

        nop                     ; little timing adjustment
        bsf     gpio,hptx
        goto    well_go_on

ah_no_ir_bit
        bcf     gpio,hptx
        dnop

well_go_on
        clrf    tmr0            ; clear the timer, so we can detect
                                ; a new IR light pulse

        movf    delay_ir_bit,w  ; wait until we are are halfway
        call    delay           ; the next bit

        dnop

        decfsz  counter         ; all 9 bits received and transmitted?
        goto    nextirbit       ; damn, next one then

        dnop
        dnop
        nop

        bcf     gpio,hptx       ; stop bit

        retlw   .0              ; done

; end ir_rcv

;*******************************************************
;* subroutine ser_rcv, at 2400/9600 baud               *
;*******************************************************

ser_rcv
        movlw   .9              ; same as ir_rc:
```

```
        movwf   counter         ; receive 9 bits (incl. startbit)

        movf    delay_ser_start,w ; now wait until we are halfway the
        call    delay           ; startbit before doing anything

nexttxbit
        btfss   gpio,hprx       ; rx high?
        goto    ah_no_tx_bit    ; jump, nothing happened :-(

        bsf     gpio,irtx       ; yep, set hpir high, i.e. light on
        movf    delay_ser_pulse,w
        call    delay           ; wait 1/8 bit

        bcf     gpio,irtx       ; and end the pulse
        goto    well_get_next

ah_no_tx_bit
        movf    delay_ser_nopulse,w
        call    delay           ; do nothing

well_get_next
        movf    delay_ser_bit,w ; wait until we are are halfway
        call    delay           ; the next bit
        dnop

        decfsz  counter         ; all 9 bits received and transmitted?
        goto    nexttxbit       ; damn, next one then

        movlw   .29             ; receiver gets 'blinded' during transmitting
        call    delay           ; wait 100 us for full sensitivity

        retlw   .0              ; done

; end ser_rcvlo

;*****************************************************
;* subroutine receive @ 2400 baud                    *
;*****************************************************

receive
        btfss   gpio,hprx       ; was it really a startbit?
        retlw   .0              ; nope, false alarm
        dnop

        movlw   .8
        movwf   counter         ; 8 bits to receive

        movlw   .187            ; wait until halfway 1 bit
rxagain
        call    delay

        rrf     rxreg
        bcf     rxreg,7

        btfss   gpio,hprx
        bsf     rxreg,7

        movlw   .124

        decfsz  counter
        goto    rxagain

        retlw   .0

; end receive

;*****************************************************
;* subroutine xmit @ 2400 baud                       *
;*****************************************************

xmit
        bsf     gpio,hptx
        movwf   txreg           ; store the byte

        dnop                    ; startbit
        dnop

        movlw   8               ; 8 bits to send
        movwf   counter

xmitnext
        movlw   .122
        call    delay
        rrf     txreg
        btfsc   status,c
        goto    sendone
        goto    sendzero
```

```
sendone
        nop
        bcf     gpio,hptx       ; TX will mark

        goto    nextbit

sendzero
        bsf     gpio,hptx
        dnop

nextbit
        dnop
        decfsz  counter
        goto    xmitnext

        movlw   .124
        call    delay

        bcf     gpio,hptx
        movlw   .126            ; and a single stopbit
        call    delay           ; should be enough as it is a single byte reply

        retlw   .0

; end xmit

;*****************************************************
;* subroutine printer_tx                             *
;*****************************************************

printer_tx
        movlw   b'00000111'
        movwf   pr_start        ; load startbits

        clrf    parity
        movlw   b'01111000'
        andwf   rxreg,w         ; correction bit H1
        call    calc_parity
        clrc
        rlf     parity

        movlw   b'11100110'
        andwf   rxreg,w         ; correction bit H2
        call    calc_parity
        clrc
        rlf     parity

        movlw   b'11010101'
        andwf   rxreg,w         ; correction bit H3
        call    calc_parity
        clrc
        rlf     parity

        movlw   b'10001011'
        andwf   rxreg,w         ; correction bit H4
        call    calc_parity

        movlw   .4
        movwf   counter
        clrf    temp
stretch_par                     ; change 4 correction bits into
        clrc                    ; 8 half bit times
        rrf     temp
        rrf     temp
        movlw   b'10000000'
        iorwf   temp
        movlw   b'11000000'
        btfss   parity,0
        xorwf   temp
        rrf     parity
        decfsz  counter
        goto    stretch_par

        movf    temp,w
        movwf   pr_corrbits

        movlw   .8
        movwf   counter
stretch_data                    ; change 8 data bits into
        clrc                    ; 16 half bit times
        rrf     pr_data1
        rrf     pr_data2
        clrc
        rrf     pr_data1
        rrf     pr_data2
        movlw   b'01000000'
```

```
        iorwf   pr_data1
        movlw   b'11000000'
        btfsc   rxreg,0
        xorwf   pr_data1
        rrf     rxreg
        decfsz  counter
        goto    stretch_data

        movlw   .4
        movwf   counter
align_prdata
        rlf     pr_data2        ; align all 27 bits to the left
        rlf     pr_data1        ; so they can be left-shifted out
        rlf     pr_corrbits
        rlf     pr_start
        decfsz  counter
        goto    align_prdata

        movlw   .27             ; 27 half bit times to transmit
        movwf   counter         ; 3 half start bits
send_irbits                     ; 8 halfcorrection bits
        rlf     pr_data2        ; 16 half data bits
        rlf     pr_data1
        rlf     pr_corrbits
        rlf     pr_start
        btfss   pr_start,7
        goto    no_burst
        call    ir_burst
        goto    next_irbit

no_burst
        nop
        movlw   .126
        call    delay

next_irbit
        decfsz  counter
        goto    send_irbits
        retlw   .0

calc_parity
        movwf   temp            ; save rxreg with mask to temp
        swapf   temp,w
        xorwf   temp            ; xor upper nibble with lower nibble

        rrf     temp,w
        movwf   counter         ; use counter as a temp var
        rrf     counter,w
        xorwf   temp

        rrf     temp,w
        xorwf   temp

        btfsc   temp,0
        bsf     parity,0

        retlw   .0

; end printer_tx

;*****************************************************
;* ir_burst                                          *
;*****************************************************

ir_burst
        movlw   .8              ; 8 pulses to send at 32768 Hz
        movwf   temp            ; (well as close as possible)

pulse_again
        bsf     gpio,irtx       ; glow in the dark....

        movlw   4               ; can't use delay subroutine here
        movwf   waitawhile      ; we're already using 2 stack levels :-(
        decfsz  waitawhile
        goto    $-1

        bcf     gpio,irtx       ; and.... light off

        movlw   3
        movwf   waitawhile
        decfsz  waitawhile
        goto    $-1

        decfsz  temp
        goto    pulse_again

        movlw   51
```

```
        movwf   waitawhile      ; store 'wait' value
        decfsz  waitawhile      ; tombtiedomtiedom
        goto    $-1             ; yawn.......

        retlw   .0

; end ir_burst

;*****************************************************
;* subroutine check_mode                             *
;*****************************************************

check_mode
        movlw   .4              ; going to check if irtx and
        movwf   counter         ; irrx are connected (prog. mode)
                                ; and repeat the test 4 times
checkagain
        clrf    tmr0
        bsf     gpio,irtx
        bcf     gpio,irtx       ; pulse irtx

        nop

        tstf    tmr0            ; did timer change value?
        skpnz
        goto    end_check       ; they're not connected

        decfsz  counter         ; repeat the test a few times
        goto    checkagain

        movlw   .4              ; they are connected allright
        movwf   mode            ; so programming mode it is!
        retlw   .0

end_check
        clrf    ee_addr         ; read first byte in eeprom
        call    ee_read_byte    ; go get it
        movf    ee_data,w
        andlw   b'00000011'     ; mask any unwanted bits
        movwf   mode            ; and store in mode

        retlw   .0

; end check_mode

;*****************************************************
;* i2c routined from microchip website               *
;* I don't like to reinvent the wheel :-)            *
;* read_current has been removed                     *
;*****************************************************

        org     0x0FD           ; !!!ENTRIES JUST BELOW 0X100!!!!

ee_write_byte
        movlw   b'10000000'     ; pc offset for write byte.  ee_ok: bit7 = '1'
        goto    init_write_control

ee_read_byte
        movlw   b'10000011'     ; pc offset for read random.  ee_ok: bit7 = '1'

init_write_control
        movwf   pc_offset       ; load pc offset register, value preset in w
        movlw   b'10100000'     ; control byte with write bit, bit 0 = '0'

start_bit
        bcf     i2c_port,sda    ; start bit, sda and scl preset to '1'

;*****************************************************

prep_transfer_byte
        movwf   ee_byte         ; byte to transfer to eeprom already in w
        movlw   .8              ; counter to transfer 8 bits
        movwf   counter
#ifdef  emulated
        movlw   0x00            ; make sure both are outputs
        tris    i2c_port
#endif


;************  clock out data (control, address, or data) byte  ************
output_byte
        bcf     i2c_port,scl    ; set clock low during data set-up
        rlf     ee_byte, f      ; rotate left, high order bit into carry bit
        bcf     i2c_port,sda    ; set data low, if rotated carry bit is
        skpnc                   ;   a '1', then:
        bsf     i2c_port,sda    ; reset data pin to a one, otherwise leave low
        nop                     ; delay, allow pullup to raise sda
```

```
        bsf     i2c_port,scl    ; clock data into eeprom
        decfsz  counter, f      ; repeat until entire byte is sent
        goto    output_byte
        nop                     ; needed to meet timing (thigh=4000ns)

;************************  acknowledge check ****************************
;************************************************************************
;
        bcf     i2c_port,scl    ; set scl low, 0.5us < ack valid < 3us
        nop                     ; needed to meet timing (tlow= 4700ns)
        bsf     i2c_port,sda
#ifdef  emulated
        movlw   (0x01 << sda)   ; make sda an input
        tris    i2c_port
#endif
        goto    $+1             ; tlow=4700ns
        bsf     i2c_port,scl    ; raise scl, eeprom acknowledge still valid
        btfsc   i2c_port,sda    ; check sda for acknowledge (low)
        bcf     pc_offset,ee_ok ; if sda not low (no ack), set error flag
        bcf     i2c_port,scl    ; lower scl, eeprom release bus
        btfss   pc_offset,ee_ok ; if no error continue, else stop bit
        goto    stop_bit
#ifdef  emulated
        movlw   0x00            ; sda back to an output
        tris    i2c_port
#endif

;*****  set up program counter offset, based on eeprom operating mode  *****
;************************************************************************
;
        movf    pc_offset,w
        andlw   b'00001111'
        movwf   temp
        movlw   jump_i2c
        addwf   temp,w
        movwf   pcl

        goto    init_address      ;pc offset=0, write control done, send address
        goto    init_write_data   ;pc offset=1, write address done, send data
        goto    stop_bit          ;pc offset=2, write done, send stop bit
        goto    init_address      ;pc offset=3, write control done, send address
        goto    init_read_control ;pc offset=4, send read control
        goto    read_bit_counter  ;pc offset=5, set counter and read byte
        goto    stop_bit          ;pc offset=6, random read done, send stop


;**********  initalize eeprom data (address, data, or control) bytes  ******
;************************************************************************
;
init_address
        incf    pc_offset, f    ; increment pc offset to 2 (write) or to 4 (read)
        movf    ee_addr,w       ; put eeprom address in w, ready to send to eeprom
        goto    prep_transfer_byte

init_write_data
        incf    pc_offset, f    ; increment pc offset to go to stop_bit next
        movf    ee_data,w       ; put eeprom data in w, ready to send to eeprom
        goto    prep_transfer_byte

init_read_control
        bsf     i2c_port,scl    ; raise scl
        nop
        bsf     i2c_port,sda    ; raise sda
        incf    pc_offset, f    ; increment pc offset to go to read_bit_counter next
        movlw   b'10100001'     ; set up read control byte, ready to send to eeprom
        goto    start_bit       ;   bit 0 = '1' for read operation

;************************  read eeprom data  ****************************
;************************************************************************
;
read_bit_counter
        bsf     i2c_port,sda    ; set data bit to 1 so we're not pulling bus down.
        nop
        bsf     i2c_port,scl
        movlw   .8              ; set counter so 8 bits will be read into eedata
        movwf   counter
#ifdef  emulated
        movlw   (0x01 << sda)
        tris    i2c_port
#endif


read_byte
        bsf     i2c_port,scl    ; raise scl, sda valid.  sda still input from ack
        setc                    ; assume bit to be read = 1
        btfss   i2c_port,sda    ; check if sda = 1
        clrc                    ; if sda not = 1 then clear carry bit
        rlf     ee_data, f      ; rotate carry bit (=sda) into eedata;
        bcf     i2c_port,scl    ; lower scl
        bsf     i2c_port,sda    ; reset sda
        decfsz  counter, f      ; decrement counter
```

```
        goto    read_byte       ; read next bit if not finished reading byte

        bsf     i2c_port,scl
        nop
        bcf     i2c_port,scl
;*****************  generate a stop bit and go home  *********************
;**************************************************************************
;
stop_bit
#ifdef  emulated
        movlw   0x00    ; set sda as output
        tris    i2c_port
#endif
        bcf     i2c_port,sda ; sda=0, on tris, to prepare for transition to '1'
        bsf     i2c_port,scl ; scl = 1 to prepare for stop bit
        goto    $+1          ; equivalent 4 nops neccessary for i2c spec tsu:sto = 4.7us
        goto    $+1
        bsf     i2c_port,sda ; stop bit, sda transition to '1' while scl high

        btfss   pc_offset,ee_ok ; check for error
        retlw   no              ; if error, send back no
        retlw   ok              ; if no error, send back ok

;**************************************************************************
;*********************  end stealing eeprom subroutines  ******************

;*****************************************************
;*****************************************************
;**     main loop                                   **
;*****************************************************
;*****************************************************

main
        call    check_mode      ; check what mode the
                                ; converter is in

        movlw   low(jump_goto_mode)
        addwf   mode,w          ; add mode
        movwf   pcl             ; and jump!

main_2400
        movlw   .60
        movwf   delay_ir_start  ; delay until halfway startbit
        movlw   .122
        movwf   delay_ir_bit    ; delay between bits

        movlw   .61
        movwf   delay_ser_start
        movlw   .14
        movwf   delay_ser_pulse ; length of IR pulse
        movlw   .15
        movwf   delay_ser_nopulse
        movlw   .107
        movwf   delay_ser_bit

        goto    mode_0_1

main_9k6
        movlw   .12
        movwf   delay_ir_start
        movlw   .26
        movwf   delay_ir_bit

        movlw   .13
        movwf   delay_ser_start
        movlw   .2
        movwf   delay_ser_pulse
        movlw   .3
        movwf   delay_ser_nopulse
        movlw   .23
        movwf   delay_ser_bit

mode_0_1
        btfsc   tmr0,0          ; external ir light pulse?
        call    ir_rcv          ; quick... do something!

        btfsc   gpio,hprx       ; serial port then?
        call    ser_rcv         ; okay, get the byte!
        goto    mode_0_1        ; hmmmm, lets try again

main_printer
        btfss   gpio,hprx       ; startbit detected?
        goto    main_printer    ; nah... check again

        call    receive         ; receive the byte
        call    printer_tx      ; calculate error corr. bits
                                ; and transmit it to the printer
```

```
        goto    main_printer

main_direct
        btfss   gpio,hprx       ; bit detected?
        bsf     gpio,irtx       ; shine a light on us

        btfsc   gpio,hprx
        bcf     gpio,irtx       ; darkness....

        goto    main_direct

main_progmode
        btfss   gpio,hprx       ; startbit?
        goto    main_progmode   ; no, wait

        call    receive         ; get the byte

        movlw   "?"
        xorwf   rxreg,w         ; was it a mode request?
        skpnz
        goto    req_cur_mode    ; yes, reply then

        movlw   "R"
        xorwf   rxreg,w         ; was it a reset command?
        skpnz
        goto    reset           ; yep, -> rebooooooot!

        movlw   b'11111100'
        andwf   rxreg,w
        xorlw   "0"             ; was it "0".."3"?
        skpz
        goto    send_error

        movlw   b'00000011'     ; mask unwanted bits
        andwf   rxreg

        movlw   low(jump_change_mode)
        addwf   rxreg,w
        movwf   pcl             ; jump to table

set_lo_speed
        movlw   .0
        goto    prog_it

set_hi_speed
        movlw   .1
        goto    prog_it

set_printer_mode
        movlw   .2
        goto    prog_it

set_direct_mode
        movlw   .3

prog_it
        movwf   mode            ; store mode for immediate use
        movwf   ee_data
        clrf    ee_addr
        call    ee_write_byte   ; and store it in eeprom

req_cur_mode
        clrf    ee_addr
        call    ee_read_byte    ; read it back (you never know)
        movf    ee_data,w
        movwf   mode
        movlw   .48
        addwf   mode,w          ; change it to ASCII, 0, 1, 2 etc.
        call    xmit            ; and send to calc
        goto    main_progmode

send_error
        movlw   "X"
        call    xmit
        goto    main_progmode

;*****************************************************
;*****************************************************
;**     end main                                    **
;*****************************************************
;*****************************************************

;*****************************************************
;* subroutine init                                   *
;*****************************************************

init
```

```
        ;       initialise PIC first
        ;                 __ __
        ;        vdd =|  U  |= vss
        ;       osc1 =|     |= G0 -> hptx
        ;       osc2 =|     |= G1 -> irtx
        ; hprx -> G3 =|_____|= G2 <- irrx

        movlw   b'00000000'
                ; ||||||||_______ hptx
                ; |||||||________ irtx
                ; ||||||_________ irrx
                ; |||||__________ hprx
                ; ||||___________ -
                ; |||____________ -
                ; ||_____________ sda
                ; |______________ scl
        movwf   gpio

        movlw   b'00011100'
                ; ||||||||_______ hptx - output
                ; |||||||________ irtx - output
                ; ||||||_________ irrx - input
                ; |||||__________ hprx - input
                ; ||||___________
                ; |||____________
                ; ||_____________ ada
                ; |______________ scl
        tris    gpio

        movlw   b'11101111'
                ; ||||||||_______ PS0
                ; |||||||________ PS1
                ; ||||||_________ PS2  wdt rate 1:128
                ; |||||__________ PSA  prescaler to wdt
                ; ||||___________ T0SE increment on low-hi _/~
                ; |||____________ T0CS transition on T0CKI pin
                ; ||_____________ /GPPU disable internal pull up
                ; |______________ /GPWU disable wakeup on pin change
        option

        clrf    tmr0

        movlw   .0
        call    delay
        movlw   .0
        call    delay
        movlw   .0
        call    delay
        movlw   .0
        call    delay           ; wait a while for power to
        movlw   .0              ; stabilize
        call    delay

        goto    main

; end init

        end
```

```
        list
; hp49vars.inc    include file containing pres_os specific declarations
;                 and macros
;       nolist

;*******************************************************
;*                                                     *
;*                 used registers                      *
;*                                                     *
;*******************************************************

        cblock  0x07

        waitawhile
        counter

        txreg
        rxreg
        parity

        temp
        temp2

        mode

        ee_byte                 ; Byte sent to or received from
                                ; EEPROM (control, address, or data)
        ee_data                 ; EEPROM Data
        ee_addr                 ; EEPROM Address
        pc_offset               ; PC offset register (low order 4 bits),
                                ;  value based on operating mode of EEPROM.
                                ;  Also, bit 7 used for EE_OK flag

        pr_start                ; start bits to send
        pr_corrbits             ; error correction bits
        pr_data1                ; data bits (part 1)
        pr_data2                ;           (part 2)

        delay_ir_start
        delay_ir_bit

        delay_ser_start
        delay_ser_pulse
        delay_ser_nopulse
        delay_ser_bit

        endc

;*******************************************************
;*                 bit definitions                     *
;*******************************************************

hptx        equ     0
irtx        equ     1
irrx        equ     2
hprx        equ     3

;*******************************************************
;*             emulator definitions                    *
;*******************************************************

#ifdef  emulated
i2c_port    equ     5           ; port a control register, used for i2c
scl         equ     01h         ; eeprom clock, scl (i/o bit 7)
sda         equ     00h         ; eeprom data,  sda (i/o bit 6)
#else
i2c_port    equ     gpio        ; port b control register, used for i2c
scl         equ     07h         ; eeprom clock, scl (i/o bit 7)
sda         equ     06h         ; eeprom data,  sda (i/o bit 6)
#endif

;*******************************************************
;*          constants and pseudo code definitions      *
;*******************************************************

ok          equ     01h
no          equ     00h
ee_ok       equ     07h         ; bit 7 in pc_offset used as ok flag for ee

#define dnop    goto $+1        ; pseudo code
#define bank0   bcf status,pa0
#define bank1   bsf status,pa0

        list
```