DE$IGNING FOR DOLLAR$

Entry Form for

Wireless and Remote Control Applications

Name: Berni Joss

Company: Logitech

Address: Rte d'Eguechaudens 1

City: Bussigny **State/Region:** VD

Country: Switzerland

Zip/Postal Code: CH-1030

Phone: +41-21-863-5216

Email: Berni.Joss@urbanet.ch

Please submit your entry by hardcopy or electronically. If by hardcopy, we recommend that you print out this file. Completed forms may be faxed to the attention of Carol Popovich at (602) 917-4150 or sent to the address provided in the Program Rules section. If electronically, please send completed forms to designing.dollars@microchip.com. Please supply or attach all files in either .TXT or .RTF format. Entries may be submitted on a diskette and mailed to the address provided in the Program Rules section. By submitting this work to Microchip Technology Incorporated, I affirm that this is my original, unpublished work and that I own all rights to such work. By submitting this work to Microchip Technology Incorporated, I grant to Microchip Technology Incorporated a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. I understand that all property rights, such as patents, copyrights and trademarks remain with me. I understand that I am responsible for all applicable taxes associated with the prizes as awarded.

Your Idea:

HP-48 IR – Serial (RS232) interface

This project provides a IR transceiver performing signaling format conversion between the Hewlett-Packard proprietary IR format used on e.g. HP-48 calculators, and the standard RS232 signaling used on any serial port.
This IR transceiver and the format converter are completely transparent to the communication protocol used by the HP-48 and the host: Kermit, X-Modem or custom serial protocols are all supported.

The small size, low-cost of the result are enabled by the following features of the PIC12C508:
1. very small device, available is SO-8 at a very attractive cost
2. internal, calibrated, oscillator requiring no external parts
3. powerful, yet simple, architecture of the CPU gives short development cycle and good performance
4. easy availability (web) of high quality development tools at affordable cost

The full-duplex interfacing and format conversion required here can be implemented as two independent and concurrent state machines, coded as two separate and pseudo-concurrent processes.
One process receives HP-48 IR signals and, after conversion, sends RS232 signals to the wired connector.
The other process accepts RS232 signals and converts them to HP-48 IR pulses.
This elegant and compact approach is encouraged by the PIC's RISC architecture:
- the extremely regular instruction timing, makes implementing real-time concurrent processes easy;
- the "*computed goto"* instruction (ADDWF PCL, F) allows very efficient state machine implementations.

The following resources of the 12C508's have been used for this application:
- 2 bytes of RAM, out of 25 bytes available
- 114 instruction words, of which about 10 for debugging and verification; out of the 512 available
- the internal 4 MHz RC oscillator
- 4 general purpose I/O pins; out of 6 available.

Alternative implementations considered include:
1. discrete solution using e.g. 555 timers; would have resulted in larger and probably more expensive solution
2. HW state machine implemented in PAL/GAL; in addition to a larger size an external power supply would have been required
3. HW state machine implemented in custom ASIC based solution; would give same size solution, but much more expensive for low to medium volumes

Microcontrollers from other manufacturers have been considered, but none where found as cost-effective in cost/performance of the devices as well as of the development tools.

The HP-48 IR format has been described in: *The HP 48SX Calculator Input / Output System, June 1991, Hewlett-Packard Journal, pp.35-40.*

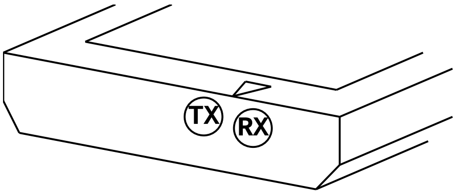
Application operation: (Minimum 1 paragraph on how the hardware/software functions. Additional paragraphs as necessary.) Explain any tricky or clever functions.

Operation is extremely simple:

1. Plug the DB9 connector of the *"HP-48 IR <-> Serial"* cable into the RS232 port on the host, e.g. PC.

2. Place the IR end of the *"HP-48 IR <-> Serial"* cable in front of the HP-48
   - at about 1 to 2 inches distance and roughly aligned such that
   - the LT1061 IR emitting LED is in front of the HP-48's IR receiver (RX), and
   - the LT1032 photodiode faces the HP-48's IR LED (TX).

   The picture below shows a HP-48 on the topside, where the infrared window is located, and illustrates the placement of HP-48's IR transmitter (TX) and receiver (RX):



3. Start your favorite communications application on the host and on the HP-48. All communication applications, working with the HP-48's wired connector, will work with this IR converter if configured for 2400 baud, the only baud rate supported by the HP-48 over IR.

Flow Chart:
Graphical program representation (Flowchart/Structure chart/ State transition diagram, etc. as appropriate).

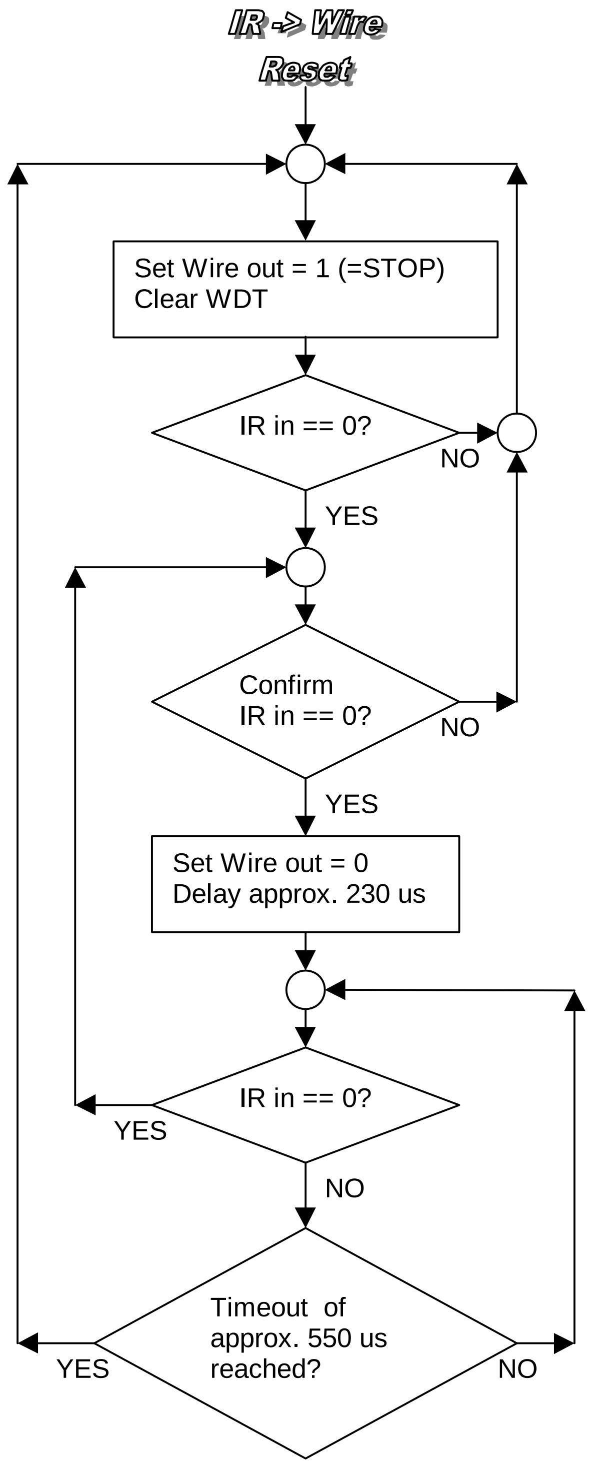
Two independent and pseudo-concurrent processes are implemented:
IR->Wire: implementing the format conversion from HP-48 IR output to wired RS232 (host).
Wire->IR: implementing the format conversion from Wired RS232 (host) to HP-48 IR input.
See flow chart on next page.

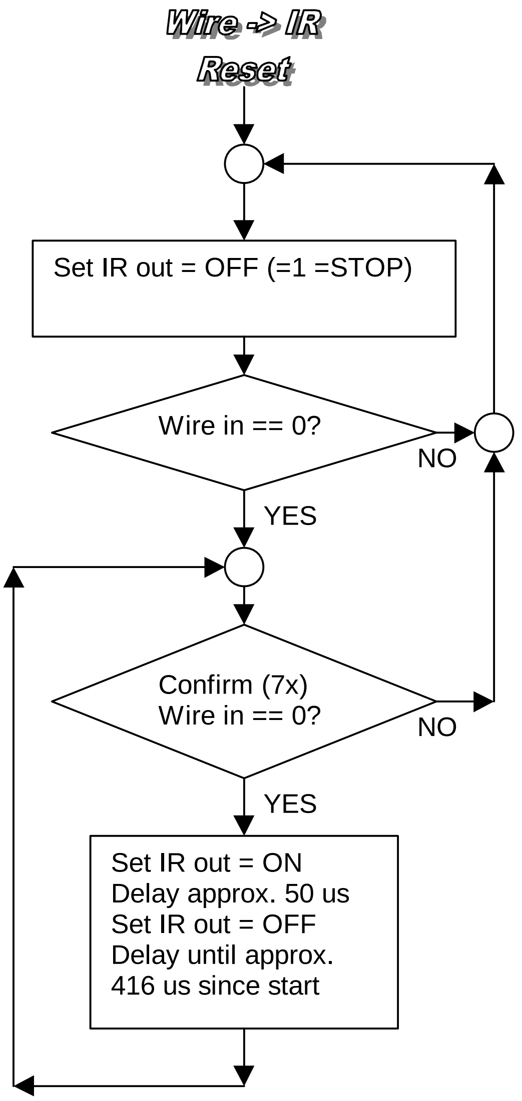
Each process has been decomposed into several sub-tasks. Each of these subtasks completes in exactly 13 CPU cycles. Execution of the various subtasks is controlled by a state variable and a jump table based dispatcher.

Concurrency is achieved by alternatively executing a subtask of process ***IR->Wire*** followed by a subtask of ***Wire->IR*** followed by a subtask of ***IR->Wire*** followed … and so on forever.
Because each subtask takes a fixed number of cycles, the two processes can run completely asynchronously of each other, under control of independent external events.

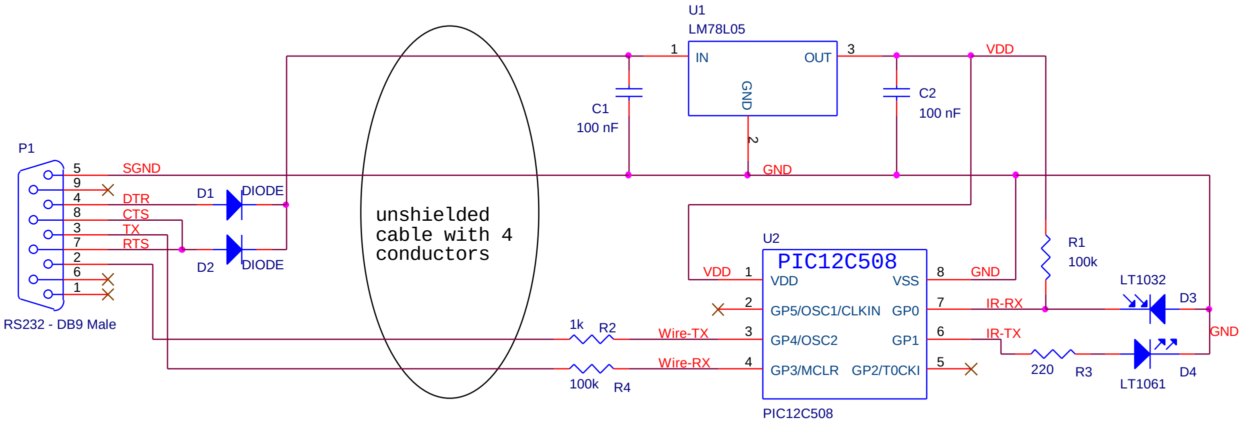
Please find additional descriptions, such as RS232 and the HP-48 IR signaling format, as well as implementation details in the code listing later on.

Author: Berni.Joss@urbanet.ch


IR -> Wire
Reset
Set Wire out = 1 (=STOP)
Clear WDT
IR in == 0?
NO
YES
Confirm
IR in == 0?
NO
YES
Set Wire out = 0
Delay approx. 230 us
IR in == 0?
YES
NO
Timeout of
approx. 550 us
reached?
YES
NO


Wire -> IR
Reset
Set IR out = OFF (=1 =STOP)
Wire in == 0?
NO
YES
Confirm (7x)
Wire in == 0?
NO
YES
Set IR out = ON
Delay approx. 50 us
Set IR out = OFF
Delay until approx.
416 us since start

Graphical hardware representation:
(Schematic, PCB layout etc.).



The layout is not critical, due to the few components and low frequencies involved.
The magic is in the software!
Several units have been built by using insulating pre-punched epoxy boards to provide mechanical stability.

The IR transmitter and receiver used here have been optimized for cost, size and power consumption, rather than for transmission range. The operating range obtained is about 2 inches.
The HP-48's IR port has itself a very limited range of 2 to 3 inches.

The power is "*stolen*" from the host's RS232 port.
As some RS232 ports deliver up to +15 V some sort of voltage regulator is required. Here a classic 78L05 has been preferred to a Zener diode based solution.
The current consumption of about 3 mA is well within the capabilities of even the weakest RS232 drivers used today.

Bill of Materials (BOM):

Part#	**Manufacture**	**Estimated Costs**	
PIC12C508	Microchip	$1.88	
78L05	National	$0.70	
LT1032, photodiode	Lite-ON	$0.72	
LT1061, infrared emitting diode	Lite-ON	$0.32	
1N4001, diode	Lite-ON	$0.12	(Qty 2)
100 nF, 20%, capacitor	Panasonic	$0.42	(Qty 2)
220 Ohm, 10%, resistor	Panasonic	$0.03	
1 kOhm, 10%, resistor	Panasonic	$0.03	
100 kOhm, 10%, resistor	Panasonic	$0.06	(Qty 2)
4 conductor, 3 ft, unshielded, cable	Alpha	$0.75	
9 pin, female, D-Sub connector	NorComp	$0.52	
9 pin, plastic hood	NorComp	$0.84	(Qty 2)
1"x1", pre-punched epoxy board	Vector	$0.10	
	TOTAL US$	$6.49	

Estimated cost is given as offered by Digi-Key, catalog 975Q, for minimum quantities.
Better prices can be obtained by buying larger quantities. Additional cost reduction can be achieved by implementing all components in one housing and not providing a cable.

The IR transmit & receive diodes are not critical, as long as similar type is used; in particular phototransistors are too slow, if used without changes to the provided schematic.
The following alternate parts, instead of the Lite-ON parts, have been successfully tested:
photodiode: SFH205 (Siemens)
IR emitting diode: LD271 (Siemens)

Please Identify Tools used, if applicable

Microchip Hardware Development Tools Used:
(Please include part number and revision)

None

Assembler/Compiler version:
(Should be latest version of MPASM or MPLAB-C).

Microchip MPLAB 3.31, MPASM 2.01

Include files:
(Should use latest header files)

P12C508.INC, as provided by Microchip with MPLAB 3.31

Software listing:
(hard copy and electronic form)

Author: Berni.Joss@urbanet.ch

```
MPASM 02.01 Released          HP48-IR.ASM   12-6-1997  17:49:36         PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00001 ;***********************************************************************
                    00002         title           "HP48 IR <-> RS232 converter"
                    00003         subtitle        "2400 Baud, using internal 4 MHz oscillator"
                    00004         #define Revision        h'0101'
                    00005
                    00006 ;***********************************************************************
                    00007 ;
                    00008 ;        HP48 IR (2400 baud) TO RS232 TRANSLATOR
                    00009 ;        ---------------------------------------
                    00010 ;
                    00011 ; Converts the 50us RZ pulse signaling used by HP48 IR link to "regular"
                    00012 ; 416us NRZ signaling used by RS232.
                    00013 ;
                    00014 ; Via this interface it is possible to upload and download to a PC without
                    00015 ; connecting a cable to the HP48.
                    00016 ; The whole interface circuit can be powered off RS232 wires from the PC,
                    00017 ; no battery of power supply is needed.
                    00018 ;
                    00019 ;
                    00020 ; Author: Berni Joss, berni.joss@urbanet.ch
                    00021 ; Dec, 06, 1997.
                    00022 ;
                    00023 ; This code may be freely used and distributed, provided the copyright notice
                    00024 ;
                    00025 ;
                    00026
                    00027 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00028
                    00029         list    c=132, n=0
                    00030
                    00031         processor       12C508
                    00032
                    00033         radix           dec
                    00034         expand
                    00035
                    00036         include "c:\progra~1\mplab\P12C508.INC" ;provided by MicroChip
                    00001         LIST
                    00002 ; P12C508.INC  Standard Header File, Version 1.02    Microchip Technology, Inc.
                    00105         LIST
                    00037         cblock          h'07'           ;first free user RAM location
                    00038         endc
                    00039
                    00040
0200 0000 0001 0000 00041         __idlocs        Revision        ; ID
     0001
0FFF 0FEE           00042         __config        _IntRC_OSC & _WDT_ON & _CP_OFF & _MCLRE_OFF
                    00043                         ;Internal 4 MHz RC oscillator
                    00044                         ;Watch-Dog enabled
                    00045                         ;Code protection disabled
                    00046                         ;use MCLR pin as GPIO
                    00047
                    00048 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00049 ; utility macros:
                    00050
  FFFFFFFF          00051 tris_INIT       =       h'FFFFFFFF'     ;undeclared pins are inputs
  00000000          00052 tris_INIT       =       h'00000000'     ;undeclared pins are inputs
                    00053 #define pin_number(port, pin)   pin
                    00054 #define input(port, pin)        tris_INIT |= (1 << ((port-GPIO)*8 + pin))
                    00055 #define output(port, pin)       tris_INIT &= ~(1 << ((port-GPIO)*8 + pin))
                    00056 #define tris_init(port)         (h'FF' & (tris_INIT >> ((port-GPIO)*8)))
                    00057
                    00058
                    00059 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00060 ; I/O pins used for IR -> Wire
                    00061 #define pin_IR_in       GPIO, 0 ;connected to IR receiver
  00000001          00062 input(  pin_IR_in)
                    00063
                    00064 #define pin_Wire_out    GPIO, 4 ;connected to RS232 transmitter
  00000001          00065 output( pin_Wire_out)
                    00066
                    00067 ; I/O pins used for IR <- Wire
                    00068 #define pin_Wire_in     GPIO, 3 ;connected to RS232 receiver
  00000009          00069 input(  pin_Wire_in)
                    00070
                    00071 #define pin_IR_out      GPIO, 1 ;connected to IR transmitter
  00000009          00072 output( pin_IR_out)
                    00073
                    00074 ; signaling polarity
```

```
                  #define skipIf_IR0      btfsc   pin_IR_in       ;skip if IR light is seen
                  #define skipIf_IR1      btfss   pin_IR_in       ;skip if no light is seen

                  #define set_IR0         bsf     pin_IR_out      ;generates IR light
                  #define set_IR1         bcf     pin_IR_out      ;does not generate light

                  #define skipIf_Wire0    btfss   pin_Wire_in     ;skip if RS232 = START, data=0
                  #define skipIf_Wire1    btfsc   pin_Wire_in     ;skip if RS232 = data 1, STOP

                  #define set_Wire0       bsf     pin_Wire_out    ;output START or data 0
                  #define set_Wire1       bcf     pin_Wire_out    ;output data 1 or STOP

                  ; Debug outputs:
                  #define Reset_Pulse     GPIO, 7         ; NOTE this port does not exist!
00000009          output( Reset_Pulse)
                  #define Wire_RX_window  GPIO, 5
00000009          output( Wire_RX_window)
                  #define IR_RX_window    GPIO, 2
00000009          output( IR_RX_window)

                  ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                  ;
                  ;       OVERVIEW OF OPERATION
                  ;       ---------------------
                  ;
                  ; This code runs two independent processes:
                  ; 1. Wire2IR  : receiving from wired RS232, sending to IR (HP48 format)
                  ; 2. IR2Wire  : receiving from IR (HP48 format), sending to wired RS232
                  ;
                  ; Each of the two processes is implemented like a synchronous hardware
                  ; state machine.  The equivalent of one "state machine clock cycle"
                  ; takes 13 PIC instruction cycles.
                  ; The two processes alternate execution:
                  ; - 13 cyc for IR2Wire
                  ; - 13 cyc for Wire2IR
                  ; - 13 cyc for IR2Wire
                  ; - 13 cyc for Wire2IR
                  ; - ... and so on forever ...
                  ;
                  ; This allows for full duplex operation.
                  ;
                  ; Neither of the two processes "understands" the data it handles.
                  ; The conversion is performed bit by bit, that is without reconstructing
                  ; the transmitted bytes. Start and stop bits are handled the same as 0 and 1
                  ; bits.
                  ;
                  ;
                  ;       SUMMARY OF HP48 IR FORMAT
                  ;       -------------------------
                  ;
                  ; As described in the HP48 user's manual, with IR mode selected, flag -33
                  ; set, transmission is always 2400 baud (416us bit duration) and no parity;
                  ; ignoring IOPAR settings.
                  ; The data format is identical to RS232, but coding of the bits is
                  ; not directly compatible:
                  ; - "0" and "START" bits are sent as IR light pulses 50us to 70us long
                  ; - "1" and "STOP" bits do not generate any IR light
                  ;
                  ; As an example, the character '#' hex 23 is sent as a sequence of 6 IR
                  ; pulses ['I' = IR light, '_' = no light]
                  ;
                  ; HP48:
                  ;
                  ;     I                    I      I      I             I      I
                  ; ____I____________________I______I______I_____________I______I____________
                  ;
                  ;     START  LSB    .      .      .      .      .      .      MSB    STOP
                  ;
                  ; whereas the same character '#' is code as follows on RS232:
                  ; ['+' : 3 Volts or more positive, '_' : -3 Volts or more negative]
                  ;
                  ; RS232:
                  ;
                  ;     ++++++++             +++++++++++++++++++++       ++++++++++++++
                  ; ____|      |_____________|                   |_______|            |______
                  ;
                  ;     START  LSB    .      .      .      .      .      .      MSB    STOP
                  ;

                  ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                  ; RAM:

                          cblock
00000007                  state_IR2Wire           ; state of process #1
```

```
  00000008          00159         state_Wire2IR           ; state of process #2
                    00160         endc
                    00161
                    00162
                    00163 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00164 ; RESET:
                    00165
0000                00166         org     0               ; 1st instruction for 12C5xx
0000 0025           00167         movwf   OSCCAL          ; we are using internal RC osc
0001 0A65           00168         goto    Start
                    00169
                    00170
                    00171 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00172 ; the following nop codes are used to "balance" execution times of
                    00173 ; various tasks in the state machine.
                    00174                                         ;since Main_Wire2IR
0002                00175 done__9_cyc_____Wire2IR                 ;       9 cyc
0002 0000           00176         nop                             ;       10 cyc
0003                00177 done_10_cyc_____Wire2IR                 ;       10 cyc
0003 0000           00178         nop                             ;       11 cyc
0004                00179 done_11_cyc_____Wire2IR                 ;       11 cyc
0004 0000           00180         nop                             ;       12 cyc
0005                00181 done_12_cyc_____Wire2IR                 ;       12 cyc
0005 0000           00182         nop                             ;       13 cyc
0006                00183 done_13_cyc_____Wire2IR                 ;       13 cyc
                    00184
                    00185 ;........................................................................
                    00186 ; IR2Wire state machine dispatcher:
                    00187 ; We reach here exactly every 26 cyc (instruction cycles).
                    00188 ; Execution of the current state takes exactly 13 cyc.
                    00189 ; Timing tolerance is about +/- 25%.
                    00190
0006                00191 Main_IR2Wire:                           ;since Main_IR2Wire     since Start of bit
0006 0207           00192         movf    state_IR2Wire, W        ;       1 cyc
0007 02A7           00193         incf    state_IR2Wire, F        ;       2 cyc
0008 01E2           00194         addwf   PCL, F                  ;       4 cyc
0009                00195 s_base_IR2Wire:
                    00196
                    00197         if ($ % 512) > 255
                    00198         ERROR jump table must stay within first 256 bytes of page
                    00199         endif
                    00200
  00000000          00201 s_init_IR2Wire          equ     $ - s_base_IR2Wire
  00000000          00202 s_waitForBit0___IR2Wire equ     $ - s_base_IR2Wire
0009 0A37           00203         goto    waitForBit0___IR2Wire   ;       6 cyc             0 cyc
  00000001          00204 s_confirmBit0___IR2Wire equ     $ - s_base_IR2Wire
000A 0A3D           00205         goto    confirmBit0___IR2Wire   ;       6 cyc            26 cyc
000B 0A42           00206         goto    sendBit0______IR2Wire   ;       6 cyc            52 cyc
000C 0A42           00207         goto    sendBit0______IR2Wire   ;       6 cyc            78 cyc
000D 0A42           00208         goto    sendBit0______IR2Wire   ;       6 cyc           104 cyc
000E 0A42           00209         goto    sendBit0______IR2Wire   ;       6 cyc           130 cyc
000F 0A42           00210         goto    sendBit0______IR2Wire   ;       6 cyc           156 cyc
0010 0A42           00211         goto    sendBit0______IR2Wire   ;       6 cyc           182 cyc
0011 0A42           00212         goto    sendBit0______IR2Wire   ;       6 cyc           208 cyc
0012 0A42           00213         goto    sendBit0______IR2Wire   ;       6 cyc           234 cyc
0013 0A42           00214         goto    sendBit0______IR2Wire   ;       6 cyc           260 cyc
0014 0A45           00215         goto    checkNewBit0__IR2Wire   ;       6 cyc           286 cyc
0015 0A45           00216         goto    checkNewBit0__IR2Wire   ;       6 cyc           312 cyc
0016 0A45           00217         goto    checkNewBit0__IR2Wire   ;       6 cyc           338 cyc
0017 0A45           00218         goto    checkNewBit0__IR2Wire   ;       6 cyc           364 cyc
0018 0A45           00219         goto    checkNewBit0__IR2Wire   ;       6 cyc           390 cyc
0019 0A45           00220         goto    checkNewBit0__IR2Wire   ;       6 cyc           416 cyc
001A 0A45           00221         goto    checkNewBit0__IR2Wire   ;       6 cyc           442 cyc
001B 0A45           00222         goto    checkNewBit0__IR2Wire   ;       6 cyc           468 cyc
001C 0A45           00223         goto    checkNewBit0__IR2Wire   ;       6 cyc           494 cyc
001D 0A45           00224         goto    checkNewBit0__IR2Wire   ;       6 cyc           520 cyc
001E 0A4A           00225         goto    lstChkNewBit0_IR2Wire   ;       6 cyc           546 cyc
                    00226
                    00227         if ($ % 512) > 255
                    00228         ERROR jump table must stay within first 256 bytes of page
                    00229         endif
                    00230
                    00231 ;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00232 ; the following nop codes are used to "balance" execution times of
                    00233 ; various tasks in the state machine.
                    00234                                         ;since Main_IR2Wire
001F                00235 done__9_cyc_____IR2Wire:                ;       9 cyc
001F 0000           00236         nop                             ;       10 cyc
0020                00237 done_10_cyc_____IR2Wire                 ;       10 cyc
0020 0000           00238         nop                             ;       11 cyc
0021                00239 done_11_cyc_____IR2Wire                 ;       11 cyc
0021 0000           00240         nop                             ;       12 cyc
0022                00241 done_12_cyc_____IR2Wire                 ;       12 cyc
0022 0000           00242         nop                             ;       13 cyc
```

```
0023                00243 done_13_cyc_____IR2Wire                 ;       13 cyc
                    00244
                    00245 ;.........................................................................
                    00246 ; Wire2IR state machine dispatcher:
                    00247 ; We reach here exactly every 26 cyc (instruction cycles).
                    00248 ; Execution of the current state takes exactly 13 cyc.
                    00249
0023                00250 Main_Wire2IR:                           ;since Main_Wire2IR     since Start of bit
0023 0208           00251         movf    state_Wire2IR, W        ;       1 cyc
0024 02A8           00252         incf    state_Wire2IR, F        ;       2 cyc
0025 01E2           00253         addwf   PCL, F                  ;       4 cyc
0026                00254 s_base_Wire2IR:
                    00255
                    00256         if ($ % 512) > 255
                    00257         ERROR jump table must stay within first 256 bytes of page
                    00258         endif
                    00259
  00000000          00260 s_init_Wire2IR          equ     $ - s_base_Wire2IR
  00000000          00261 s_waitForBit0___Wire2IR equ     $ - s_base_Wire2IR
0026 0A4F           00262         goto    waitForBit0___Wire2IR   ;       6 cyc             0 cyc
  00000001          00263 s_confirmBit0___Wire2IR equ     $ - s_base_Wire2IR
0027 0A55           00264         goto    confirmBit0___Wire2IR   ;       6 cyc            26 cyc
0028 0A55           00265         goto    confirmBit0___Wire2IR   ;       6 cyc            52 cyc
0029 0A55           00266         goto    confirmBit0___Wire2IR   ;       6 cyc            78 cyc
002A 0A55           00267         goto    confirmBit0___Wire2IR   ;       6 cyc           104 cyc
002B 0A55           00268         goto    confirmBit0___Wire2IR   ;       6 cyc           130 cyc
002C 0A55           00269         goto    confirmBit0___Wire2IR   ;       6 cyc           156 cyc
002D 0A55           00270         goto    confirmBit0___Wire2IR   ;       6 cyc           182 cyc
002E 0A5A           00271         goto    pulseBit0_____Wire2IR   ;       6 cyc           208 cyc
002F 0A5A           00272         goto    pulseBit0_____Wire2IR   ;       6 cyc           234 cyc
0030 0A5D           00273         goto    endPulse______Wire2IR   ;       6 cyc           260 cyc
0031 0A5D           00274         goto    endPulse______Wire2IR   ;       6 cyc           286 cyc
0032 0A5D           00275         goto    endPulse______Wire2IR   ;       6 cyc           312 cyc
0033 0A5D           00276         goto    endPulse______Wire2IR   ;       6 cyc           338 cyc
0034 0A5D           00277         goto    endPulse______Wire2IR   ;       6 cyc           364 cyc
0035 0A5D           00278         goto    endPulse______Wire2IR   ;       6 cyc           390 cyc
0036 0A5F           00279         goto    lstChkNewBit0_Wire2IR   ;       6 cyc           416 cyc
                    00280
                    00281         if ($ % 512) > 255
                    00282         ERROR jump table must stay within first 256 bytes of page
                    00283         endif
                    00284
                    00285 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00286
                    00287 ;.........................................................................
0037                00288 waitForBit0___IR2Wire:                  ; 6 cyc since Main
0037 0486           00289         set_Wire1               ; 7 cyc
0038 0C00           00290         movlw   s_waitForBit0___IR2Wire ; 8 cyc
0039 0606           00291         skipIf_IR0              ; 9 cyc
003A 0027           00292         movwf   state_IR2Wire           ;10 cyc
003B 0004           00293         clrwdt                          ;11 cyc
003C 0A23           00294         goto    done_13_cyc_____IR2Wire ;13 cyc
                    00295
                    00296 ;.........................................................................
003D                00297 confirmBit0___IR2Wire:                  ; 6 cyc since Main
003D 0C00           00298         movlw   s_waitForBit0___IR2Wire ; 7 cyc
003E 0606           00299         skipIf_IR0                      ; 8 cyc
003F 0027           00300         movwf   state_IR2Wire           ; 9 cyc
0040 0546           00301         bsf     IR_RX_window            ;10 cyc
0041 0A22           00302         goto    done_12_cyc_____IR2Wire ;12 cyc
                    00303
                    00304 ;.........................................................................
0042                00305 sendBit0______IR2Wire:                  ; 6 cyc since Main
0042 0586           00306         set_Wire0                       ; 7 cyc
0043 0446           00307         bcf     IR_RX_window            ; 8 cyc
0044 0A20           00308         goto    done_10_cyc_____IR2Wire ;10 cyc
                    00309
                    00310 ;.........................................................................
0045                00311 checkNewBit0__IR2Wire:                  ; 6 cyc since Main
0045 0C01           00312         movlw   s_confirmBit0___IR2Wire ; 7 cyc
0046 0706           00313         skipIf_IR1                      ; 8 cyc
0047 0027           00314         movwf   state_IR2Wire           ; 9 cyc
0048 0546           00315         bsf     IR_RX_window            ;10 cyc
0049 0A22           00316         goto    done_12_cyc_____IR2Wire ;12 cyc
                    00317 ;.........................................................................
004A                00318 lstChkNewBit0_IR2Wire:                  ; 6 cyc since Main
004A 0C00           00319         movlw   s_waitForBit0___IR2Wire ; 7 cyc
004B 0706           00320         skipIf_IR1                      ; 8 cyc
004C 0C01           00321         movlw   s_confirmBit0___IR2Wire ; 9 cyc
004D 0027           00322         movwf   state_IR2Wire           ;10 cyc
004E 0A22           00323         goto    done_12_cyc_____IR2Wire ;12 cyc
                    00324
                    00325 ;.........................................................................
                    00326
                    00327 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00328 
                    00329 ;.........................................................................
004F                00330 waitForBit0___Wire2IR:                  ; 6 cyc since Main
004F 0426           00331         set_IR1                         ; 7 cyc
0050 0C00           00332         movlw   s_waitForBit0___Wire2IR ; 8 cyc
0051 0766           00333         skipIf_Wire0                    ; 9 cyc
0052 0028           00334         movwf   state_Wire2IR           ;10 cyc
0053 05A6           00335         bsf     Wire_RX_window          ;11 cyc
0054 0A06           00336         goto    done_13_cyc_____Wire2IR ;13 cyc
                    00337 
                    00338 ;.........................................................................
0055                00339 confirmBit0___Wire2IR:                  ; 6 cyc since Main
0055 0C00           00340         movlw   s_waitForBit0___Wire2IR ; 7 cyc
0056 0766           00341         skipIf_Wire0                    ; 8 cyc
0057 0028           00342         movwf   state_Wire2IR           ; 9 cyc
0058 05A6           00343         bsf     Wire_RX_window          ;10 cyc
0059 0A05           00344         goto    done_12_cyc_____Wire2IR ;12 cyc
                    00345 
                    00346 ;.........................................................................
005A                00347 pulseBit0_____Wire2IR:                  ; 6 cyc since Main
005A 0526           00348         set_IR0                         ; 7 cyc
005B 04A6           00349         bcf     Wire_RX_window          ; 8 cyc
005C 0A03           00350         goto    done_10_cyc_____Wire2IR ;10 cyc
                    00351 
                    00352 ;.........................................................................
005D                00353 endPulse______Wire2IR:                  ; 6 cyc since Main
005D 0426           00354         set_IR1                         ; 7 cyc
005E 0A02           00355         goto    done__9_cyc_____Wire2IR ; 9 cyc
                    00356 
                    00357 ;.........................................................................
005F                00358 lstChkNewBit0_Wire2IR:                  ; 6 cyc since Main
005F 0C00           00359         movlw   s_waitForBit0___Wire2IR ; 7 cyc
0060 0666           00360         skipIf_Wire1                    ; 8 cyc
0061 0C01           00361         movlw   s_confirmBit0___Wire2IR ; 9 cyc
0062 05A6           00362         bsf     Wire_RX_window          ;10 cyc
0063 0028           00363         movwf   state_Wire2IR           ;11 cyc
0064 0A06           00364         goto    done_13_cyc_____Wire2IR ;13 cyc
                    00365 
                    00366 ;.........................................................................
                    00367 
                    00368 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
0065                00369 Start:
                    00370         ;initialize used registers and configuration ...
                    00371 
0065 0C09           00372         movlw   tris_init(GPIO)
0066 0006           00373         tris    GPIO
0067 0066           00374         clrf    GPIO
0068 0C09           00375         movlw   tris_init(GPIO)
0069 0006           00376         tris    GPIO
006A 0066           00377         clrf    GPIO
                    00378 
006B 05E6           00379         bsf     Reset_Pulse     ;should allow to detect watch-dog problems
                    00380 
006C 0C00           00381         movlw   s_init_IR2Wire
006D 0027           00382         movwf   state_IR2Wire
                    00383 
006E 0C00           00384         movlw   s_init_Wire2IR
006F 0028           00385         movwf   state_Wire2IR
                    00386 
0070 04E6           00387         bcf     Reset_Pulse
                    00388         ;done with initializations,
                    00389         ;let's start the two processes:
0071 0A06           00390         goto    Main_IR2Wire
                    00391 
                    00392 ;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
                    00393 
                    00394         end
```

Author: Berni.Joss@urbanet.ch

```
MPASM 02.01 Released          HP48-IR.ASM   12-6-1997  17:49:36         PAGE  2
HP48 IR <-> RS232 converter
2400 Baud, using internal 4 MHz oscillator
SYMBOL TABLE
  LABEL                             VALUE

C                                 00000000
DC                                00000001
F                                 00000001
FSR                               00000004
GPIO                              00000006
GPWUF                             00000007
INDF                              00000000
IR_RX_window                      GPIO, 2
Main_IR2Wire                      00000006
Main_Wire2IR                      00000023
NOT_GPPU                          00000006
NOT_GPWU                          00000007
NOT_PD                            00000003
NOT_TO                            00000004
OSCCAL                            00000005
PA0                               00000005
PCL                               00000002
PS0                               00000000
PS1                               00000001
PS2                               00000002
PSA                               00000003
Reset_Pulse                       GPIO, 7
Revision                          h'0101'
STATUS                            00000003
Start                             00000065
T0CS                              00000005
T0SE                              00000004
TMR0                              00000001
W                                 00000000
Wire_RX_window                    GPIO, 5
Z                                 00000002
_CP_OFF                           00000FFF
_CP_ON                            00000FF7
_ExtRC_OSC                        00000FFF
_IntRC_OSC                        00000FFE
_LP_OSC                           00000FFC
_MCLRE_OFF                        00000FEF
_MCLRE_ON                         00000FFF
_WDT_OFF                          00000FFB
_WDT_ON                           00000FFF
_XT_OSC                           00000FFD
__12C508                          00000001
checkNewBit0__IR2Wire             00000045
confirmBit0___IR2Wire             0000003D
confirmBit0___Wire2IR             00000055
done_10_cyc_____IR2Wire           00000020
done_10_cyc_____Wire2IR           00000003
done_11_cyc_____IR2Wire           00000021
done_11_cyc_____Wire2IR           00000004
done_12_cyc_____IR2Wire           00000022
done_12_cyc_____Wire2IR           00000005
done_13_cyc_____IR2Wire           00000023
done_13_cyc_____Wire2IR           00000006
done__9_cyc_____IR2Wire           0000001F
done__9_cyc_____Wire2IR           00000002
endPulse______Wire2IR             0000005D
input                             tris_INIT |= (1 << ((port-GPIO)*8 + pin))
lstChkNewBit0_IR2Wire             0000004A
lstChkNewBit0_Wire2IR             0000005F
output                            tris_INIT &= ~(1 << ((port-GPIO)*8 + pin))
pin_IR_in                         GPIO, 0
pin_IR_out                        GPIO, 1
pin_Wire_in                       GPIO, 3
pin_Wire_out                      GPIO, 4
pin_number                        pin
pulseBit0_____Wire2IR             0000005A
s_base_IR2Wire                    00000009
s_base_Wire2IR                    00000026
s_confirmBit0___IR2Wire           00000001
s_confirmBit0___Wire2IR           00000001
s_init_IR2Wire                    00000000
s_init_Wire2IR                    00000000
s_waitForBit0___IR2Wire           00000000
s_waitForBit0___Wire2IR           00000000
sendBit0______IR2Wire             00000042
set_IR0                           bsf     pin_IR_out
set_IR1                           bcf     pin_IR_out
set_Wire0                         bsf     pin_Wire_out
```

```
set_Wire1                         bcf     pin_Wire_out
skipIf_IR0                        btfsc   pin_IR_in
skipIf_IR1                        btfss   pin_IR_in
skipIf_Wire0                      btfss   pin_Wire_in
skipIf_Wire1                      btfsc   pin_Wire_in
state_IR2Wire                     00000007
state_Wire2IR                     00000008
tris_INIT                         00000009
tris_init                         (h'FF' & (tris_INIT >> ((port-GPIO)*8)))
waitForBit0___IR2Wire             00000037
waitForBit0___Wire2IR             0000004F


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XX--------------
0200 : XXXX------------ ---------------- ---------------- ----------------
0FC0 : ---------------- ---------------- ---------------- ---------------X

All other memory blocks unused.

Program Memory Words Used:   114
Program Memory Words Free:   398


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,     0 suppressed
```

Author: Berni.Joss@urbanet.ch