Serial Load Techniques for the HP3 Series

Application Note AN-00155

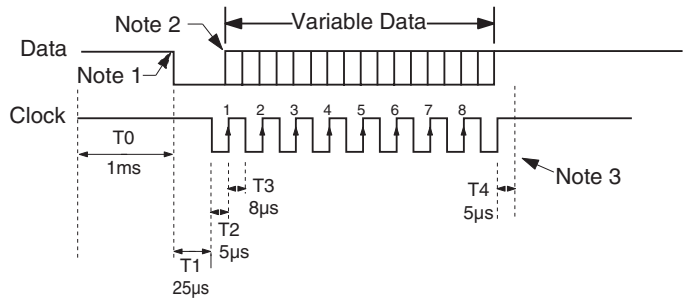
Introduction

The HP3 is the third generation of the popular HP Series transmitters and receivers. All of the HP3 Series modules continue to offer eight parallel selectable channels, but versions are also available that add serial selection of 100 channels. This application note describes how to send the serial data to the HP3 in order to select the desired channel. It does not detail a final product, but should provide a starting point for development.

This application note details the process of sending a channel number to the HP3 modules using a serial link from a microprocessor. The code for this is shown and then it is used in an example. The software in the example will load the HP3 with channel 0 and then increment the channel number by one each time a button is pressed.

Sending Data to the HP3

The timing for serial loading the HP3 is shown in Figure 1.



(T0) Minimum time between packets or prior to data startup	1mS min.
(T1) Data-LO/Clock-HI to Data-LO/Clock-LO	25µS min.
(T2) Clock-LO to Clock-HI	5µS min.
(T3) Clock-HI to Clock-LO	8µS min.
(T4) Data-HI/Clock-HI	5µS min.
Total Packet Time	157µS min.

Figure 1: Serial Data Timing Table

Data is sent synchronously with the LSB sent first. The packet consists of a start period (T1), eight data bits, and a stop period (T4). Data is sent on the Data Line, while being clocked by the Clock Line. When the Data Line and the Clock Line are both HIGH there will be no loading. As soon as the Data Line goes LOW while the Clock Line is HIGH (T1), loading begins. When the Clock Line goes LOW with the Data Line (T2), clocking begins. Data is then sent out on the Data Line, after which the Clock Line is then

Revised 8/20/12

pulsed (T3). Data is recorded by the rising edge of the clock. Clocking continues for eight bits, then the Clock and Data lines both go HIGH (T4). After the minimum required latch time (5µS), the packet is latched. The HP3 requires 1mS between packets (T0), so after this time, the next packet can be started. The total minimum time required for transmission is 157µS.

The Example

For this example a PIC16F630 processor from Microchip was used. Other PICs could be used with minor changes to the code. Only the three Port A pins were used and all of the other data I/O pins were tied to ground. The schematic is shown in Figure 2.

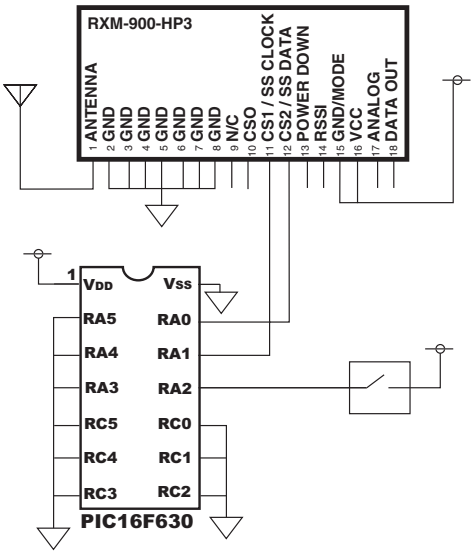


Figure 2: Serial Load Example Schematic

The PIC was configured as follows:

All Protection	OFF
Brown-Out Detection	OFF
Watch-Dog Timer	OFF
Power-Up Timer	OFF
MCLR	Internal
Oscillator	Internal

I/O Lines Used:

Port A, line 0 = Data Line	(Output)
Port A, line 1 = Clock Line	(Output)
Port A, line 2 = Ext. Interrupt	(Input)

The HP3 is put into serial mode when the mode pin (pin 15) is left open or held high. In this condition CS1(pin 11) becomes the serial clock line and CS2 (pin 12) becomes the serial data line.

Pin 12 of the PIC is connected to a button that will trigger the interrupt.

The Program

Once the PIC is initialized in code, it will go to sleep until a button press causes an interrupt. At this point it will wake-up, load the first channel value of channel 0 into the module, and then return to sleep. Every time the PIC is interrupted, the channel value being loaded is incremented by one until channel 103 is reached, at which point the channel selection will start over at channel 0.

Since channel 100 is the maximum channel, loading channels 101, 102, and 103 will cause an error. If an error occurs, the HP3 defaults to channel 0. Loading these channels ensures that the default Load Error is working by switching to channel 0. This is a convenient debugging tool since it will verify that there was serial port activity, but a problem with the transmission.

Errors are caused when data is corrupted during transmission, or when a channel value greater than the HP3 allows is sent into the module.

The Code

The code for loading a channel into the HP3 and for the example described above is listed on the following pages.

The Code for Loading a Channel

The code listed below will load a channel number into the HP3 module. The channel number is loaded into a register by the user's software. This function will then send the value in that register to the HP3.

```
title "PIC16F630 Synchronous-Serial-Load program"
        list p=16f630,f=inhx32
        #include <p16f630.inc>
        errorlevel -302 ;Keeps bank-select errors from showing on builds
        __CONFIG _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF &
_PWRTE_OFF & _INTRC_OSC_NOCLKOUT

;PORT ALLOCATIONS
;*****************************************************************************
;PORTA
data_ln                 =       d'0'
clock_ln                =       d'1'
interrupt               =       d'2'

;TEMPORARY REGISTER EQUATES
;*****************************************************************************
ch_num_temp             =       40h                     ;Storage register for temp. channel
ch_num                  =       41h                     ;Stores selected channel
count1                  =       42h                     ;General purpose temp counter register
count2                  =       43h                     ;General purpose temp counter register
temp_count              =       44h                     ;General purpose temp counter register

;START OF PROGRAM ROUTINES
;*****************************************************************************

                ;Reset vector for start-up and BOD resets
                org             0x000
                goto            Start

                ;Interrupt vector contains interrupt routine.
                org             0x004
                bcf             intcon,gie
                goto            send_serial

Start           ;configure internal oscillator
                bsf             status,rp0              ;Set file register bank to 1
                call            0x3FF                   ;Retrieve factory calibration value
                movwf           osccal                  ;Update register with factory calibr.

                ;initialize control registers
                movlw           b'00000000'             ;Sets portC to all outputs
                movwf           trisC
                movlw           b'00000100'             ;Sets portA to all outputs except RA2
                movwf           trisA
                movlw           b'01000000'             ;Bit-7 = 0 to enable portA pull-up resistors, &
```

```
                movwf           option_reg              ;---bit-6 = 1 to set RA2/Int on rising edge
                movlw           b'00000100'             ;Enables weak pull-up resistor on RA2 only
                movwf           wpua

                bcf             status,rp0              ;Set file register bank to 0
                movlw           07h                     ;Sets up comparitor to digital outputs
                movwf           cmcon

                ;Initialize values and wait to send data
                clrf            portA
                clrf            portC
                clrf            ch_num_temp
                clrf            intcon
                bsf             portA,data_ln
                bsf             portA,clock_ln

; You must first fill "ch_num_temp" with an 8-bit value
;               movlw           d'?'
;               movwf           ch_num_temp

send_serial
                movf            ch_num_temp,w           ;For testing only
                movwf           ch_num                  ;Register for rotating bits
                movlw           d'8'                    ;Sets temporary conter for 8 bits
                movwf           temp_count
                bcf             portA,data_ln           ;Data LOW and Clock HIGH = packet ready (T1)
                bsf             portA,clock_ln
                call            start_delay             ;Time delay for packet ready (T1)
                bcf             portA,clock_ln          ;Clock goes LOW for start bit (T2)
                call            data_delay
bit_move        btfsc           ch_num,0                ;Is bit to be sent a '1' or '0'
                bsf             portA,data_ln           ;If '1', then set modules data pin to '1'
                call            data_delay
                rrf             ch_num,f                ;Moves in next bit. Loads LSB first
                bsf             portA,clock_ln          ;Set modules clock pin '1': (rising edge)
                call            clock_delay             ;Settling delay: may be longer or shorter (T3)
                bcf             portA,clock_ln          ;Clears modules clock pin for clock pulse
                bcf             portA,data_ln           ;Initialize modules data pin to Low
                decfsz          temp_count,f            ;Have all 8 bits been sent?
                goto            bit_move                ;No? Then continue sending
done            bsf             portA,data_ln           ;Yes? Then time to latch packet
                bsf             portA,clock_ln          ;Both set = Data Latched (T4)
wait            btfsc           portA,interrupt         ;Wait until button is released
                goto            wait
sleep_loop
                movlw           b'10010000'             ;Sets GIE & RA2/Int, and clears INTF
                movwf           intcon
                sleep                                   ;Takes 2uS to wake up
                nop                                     ;Allows interrupt vector to be used (0x004)
```

```
;-------------------------------------------------------------------------------
;********Delay = 47uS********
start_delay
               clrf           count1
               movlw          05h                    ;Change this value to adjust delay time
               movwf          count1
loop_1         decfsz         count1,f
               goto           loop_1
               return
;-------------------------------------------------------------------------------
;********Delay = 9uS*********
clock_delay
               clrf           count2
               movlw          01h                    ;Change this value to adjust delay time
               movwf          count2
loop_3         decfsz         count2,f
               goto           loop_3
               return
;-------------------------------------------------------------------------------
;********Delay = 9uS**********
data_delay
               clrf           count2
               movlw          01h                    ;Change this value to adjust delay time
               movwf          count2
loop_2         decfsz         count2,f
               goto           loop_2
               return
;-------------------------------------------------------------------------------

               end
```

Code for the Example

The code below is the code for the example described earlier. Each time a button is pressed, the PIC will increment the channel number by one, starting with channel 0 and going to channel 103.

```
title "PIC16F630 Synchronous-Serial-Load program"
list p=16f630,f=inhx32
#include <p16f630.inc>
errorlevel -302        ;Keeps bank-select errors from showing on builds
__CONFIG _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF & _PWRTE_OFF &
_INTRC_OSC_NOCLKOUT

;PORT ALLOCATIONS
;*******************************************************************************
;PORTA
data_ln                =       d'0'
clock_ln               =       d'1'
interrupt              =       d'2'
```

```
;TEMPORARY REGISTER EQUATES
;*******************************************************************************
ch_num_temp             =       40h                     ;Storage register for temp. channel
ch_num                  =       41h                     ;Stores selected channel
count1                  =       42h                     ;General purpose temp counter register
count2                  =       43h                     ;General purpose temp counter register
temp_count              =       44h                     ;General purpose temp counter register

;START OF PROGRAM ROUTINES
;*******************************************************************************
;               Timing: (35uS) Start-Up to Send Data Packet
;                              (23uS) T-1
;                              (04uS) T-2
;                              (08uS) T-3
;                              (05uS) T-4
;                              (11uS) Wake-Up to Send Data Packet

                ;Reset vector for start-up and BOD resets
                org             0x000
                goto            Start

                ;Interrupt vector contains interrupt routine.
                org             0x004
                bcf             intcon,gie
                goto            send_serial

Start           ;configure internal oscillator
                bsf             status,rp0              ;Set file register bank to 1
                call            0x3FF                   ;Retrieve factory calibration value
                movwf           osccal                  ;Update register with factory calibr.

                ;initialize control registers
                bsf             pcon,0                  ;This resets the Brown-Out-Detect flag
                bsf             pcon,1                  ;This resets the Power-Up-Timer flag
                movlw           b'00000000'             ;Sets portC to all outputs
                movwf           trisC
                movlw           b'00000100'             ;Sets portA to all outputs except RA2
                movwf           trisA
                movlw           b'01000000'             ;Bit-7 = 0 to enable portA pull-up resistors, &
                movwf           option_reg              ;---bit-6 = 1 to set RA2/Int on rising edge
                movlw           b'00000100'             ;Enables weak pull-up resistor on RA2 only
                movwf           wpua

                bcf             status,rp0              ;Set file register bank to 0
                movlw           07h                     ;Sets up comparitor to digital outputs
                movwf           cmcon

                ;Initialize values and wait to send data
                clrf            portA
                clrf            portC
```

```
                clrf            ch_num_temp
                clrf            intcon
                bsf             portA,data_ln
                bsf             portA,clock_ln
                goto            sleep_loop
send_serial
                movf            ch_num_temp,w           ;For testing only
                movwf           ch_num
                movlw           d'8'                    ;Sets temporary conter for 8 bits
                movwf           temp_count
                bcf             portA,data_ln           ;Data LOW and Clock HIGH = packet ready (T1)
                bsf             portA,clock_ln
                call            start_delay             ;Time delay for packet ready (T1)
                bcf             portA,clock_ln          ;Clock goes LOW for start bit (T2)
bit_move        btfsc           ch_num,0                ;Is bit to be sent a '1' or '0'
                bsf             portA,data_ln           ;If '1', then set modules data pin to '1'
rrf             ch_num,f                                ;Moves in next bit. Loads LSB first
bsf             portA,          clock_ln                ;Set modules clock pin '1': (rising edge)
                call            clock_delay             ;Settling delay: may be longer or shorter (T3)
                bcf             portA,clock_ln          ;Clears modules clock pin for clock pulse
                bcf             portA,data_ln           ;Initialize modules data pin to Low
                decfsz          temp_count,f            ;Have all 8 bits been sent?
                goto            bit_move                ;No? Then continue sending
done            bsf             portA,data_ln           ;Yes? Then time to latch packet
                bsf             portA,clock_ln                  ;Both set = Data Latched (T4)
                incf            ch_num_temp,f           ;Increment channel for next send
                movlw           d'104'                  ;Going to CH.104 allows 3 defaults to CH.0
                subwf           ch_num_temp,w           ;Subtract the current ch. number value from the max
                btfsc           status,2                ;---possible channels to see if it is time to start
                clrf            ch_num_temp             ;---over with channel-0
wait            btfsc           portA,interrupt                 ;Wait until button is released
                goto            wait
                movlw           0FFh                    ;200mS debounce delay for button press
                movwf           count2                  ;Loads count_2 with b'1111 1111'
Cnt2            movlw           0FFh
                movwf           count1                  ;Loads count_1 with b'1111 1111'
Cnt1            decfsz          count1,f                ;Stay here until count1 is zero
                goto            Cnt1
                decfsz          count2,f                ;Count_1 is empty, so decrement count_2
                goto            Cnt2
sleep_loop
                movlw           b'10010000'             ;Sets GIE & RA2/Int, and clears INTF
                movwf           intcon
                sleep                                   ;Takes 2uS to wake up
                nop                                     ;Allows so interrupt vector to be used
;-------------------------------------------------------------------------------
start_delay
                clrf            count1
                movlw           05h
                movwf           count1
```

```
loop_1          decfsz          count1,f
                goto            loop_1
                return
;*******************************************************************************
clock_delay
                clrf            count2
                movlw           01h
                movwf           count2
                return
;*******************************************************************************
                end
```


159 Ort Lane, Merlin, OR, US 97532
Phone: +1 541 471 6256
Fax: +1 541 471 6251
www.linxtechnologies.com