

LOC OBJECT CODE LINE SOURCE TEXT
VALUE

```

00001 ;*****
00002 ;   BitScope Byte Code Interpreter
00003 ;*****
00004
00005         list      p=16F84,      n=92
00006
00007         include <p16f84.inc>          ; include for this chip
00001 LIST
00002 ; P16F84.INC Standard Header File, Version 2.00      Microchip Technology, Inc.
00136 LIST
00008
00009 ;*****
00010 ;   Set Parameters
00011 ;*****
00012
00000015 00013 BIT_CELL      equ      15          ; set cell width (19k2)
00000024 00014 START_CELL     equ      24          ; start cell to middle of bit0
00000005 00015 SER_0           equ      5           ; tx pin is RB5
00000006 00016 SER_I          equ      6           ; rx pin is RB6
00000003 00017 JMP_PAGE       equ      3           ; jump page 0x380->0x3ff
00018
00000000 00019 RA0              equ      0           ; port bit 0
00000001 00020 RA1              equ      1           ; port bit 1
00000002 00021 RA2              equ      2           ; port bit 2
00000003 00022 RA3              equ      3           ; port bit 3
00000004 00023 RA4              equ      4           ; port bit 4
00024
00000000 00025 RB0              equ      0           ; port bit 0
00000001 00026 RB1              equ      1           ; port bit 1
00000002 00027 RB2              equ      2           ; port bit 2
00000003 00028 RB3              equ      3           ; port bit 3
00000004 00029 RB4              equ      4           ; port bit 4
00000005 00030 RB5              equ      5           ; port bit 5
00000006 00031 RB6              equ      6           ; port bit 6
00000007 00032 RB7              equ      7           ; port bit 7
00033
00000001 00034 SEL0              equ      1           ; RB1
00000002 00035 SEL1              equ      2           ; RB2
00000003 00036 SEL2              equ      3           ; RB3
00037
00038
00039
00040

```

LOC	OBJECT CODE	LINE	SOURCE	TEXT
	VALUE			
		00041	page	
		00042	;	*****
		00043	;	Register definition
		00044	;	*****
		00045		
00000020		00046	tx_data	equ 0x20 ; transmit register
00000021		00047	rx_data	equ 0x21 ; receive register
		00048		
00000022		00049	timer_0	equ 0x22 ; timer low
00000023		00050	timer_1	equ 0x23 ; timer high
		00051		
00000024		00052	count_0	equ 0x24 ; counter
00000025		00053	count_1	equ 0x25 ; counter
		00054		
00000026		00055	gen_0	equ 0x26 ; general purpose
00000027		00056	gen_1	equ 0x27 ;
00000028		00057	gen_2	equ 0x28 ;
00000029		00058	gen_3	equ 0x29 ;
		00059		
0000002A		00060	hex2asc	equ 0x2a ; use for building ascii
0000002B		00061	ascii_l	equ 0x2b ;
0000002C		00062	ascii_h	equ 0x2c ;
		00063		
		00064		
		00065	;	*****
		00066	;	Set initial condition for Ports
		00067	;	*****
		00068		
0000000F		00069	PORTA_INIT	equ 0x0f ; clk '1', CHA, Rng 3 all '1'
00000010		00070	TRISA_INIT	equ 0x10 ; RA0..RA3 OUTs, TOCI is INPUT
		00071		
0000003F		00072	PORTB_INIT	equ 0x3f ; Shift Read
00000043		00073	TRISB_INIT	equ 0x43 ; SerI, SEL0, A-DATA are inputs
		00074		
000000EF		00075	PORTB_TRACE	equ 0xef ; count store trig7
00000041		00076	TRISB_TRACE	equ 0x41 ; serial in, A_DATA in
		00077		
000000EF		00078	OPTION_INI	equ 0xef ;
00000008		00079	INTCON_INI	equ 0x08 ;
		00080		
		00081		
		00082		

LOC	OBJECT CODE	LINE	SOURCE	TEXT
	VALUE			
		00083	page	
		00084	;	*****
		00085	;	Set up block of registers that form part of the Byte Code Machine
		00086	;	*****
		00087		
		00088		
00000030		00089	BCR_BASE	equ 0x30
		00090		
00000030		00091	REG_INP	equ BCR_BASE + 0 ; main data entry register
00000031		00092	REG_ADD	equ BCR_BASE + 1 ; points to a byte code register
00000032		00093	REG_SRC	equ BCR_BASE + 2 ; Register source for load R0
		00094		
00000033		00095	REG_COUNT_L	equ BCR_BASE + 3 ; low byte of sample count
00000034		00096	REG_COUNT_H	equ BCR_BASE + 4 ; high byte
		00097		
00000035		00098	REG_LOGIC	equ BCR_BASE + 5 ; logic trigger byte
00000036		00099	REG_MASK	equ BCR_BASE + 6 ; mask trigger byte
		00100		
00000037		00101	REG_CONTROL	equ BCR_BASE + 7 ; logic control
		00102		; 0 - PG0 bit, BNC (0) POD (1)
		00103		; 1 - TRIG7 src (2,1) as below
		00104		; 2 - 00,DD7 01,CNT-1 10,CNT-2 11,TRG
		00105		; 3 - DIG TRIG (0), ANALOG TRIG (1)
		00106		
00000038		00107	REG_OPTION	equ BCR_BASE + 8 ; operation control reegister
		00108		
00000039		00109	REG_SHIFT_L	equ BCR_BASE + 9 ; value of SPOCK counter
00000040		00110	REG_SHIFT_H	equ BCR_BASE + 10 ; shifted out of chip
		00111		
00000041		00112	REG_DELAY_L	equ BCR_BASE + 11 ; low byte of post TRIG delay
00000042		00113	REG_DELAY_H	equ BCR_BASE + 12 ; high byte of post TRIG delay
		00114		
00000043		00115	REG_TBASE	equ BCR_BASE + 13 ; timebase expander byte
00000044		00116	REG_CHOP	equ BCR_BASE + 14 ; Channel swap settings
		00117		
00000045		00118	REG_DUMP	equ BCR_BASE + 15 ; Number of samples dumped on request
		00119		
00000046		00120	REG_EEDATA	equ BCR_BASE + 16 ; eeeprom data
00000047		00121	REG_EEADDR	equ BCR_BASE + 17 ; eeeprom address
		00122		
00000048		00123	REG_PODTX	equ BCR_BASE + 18 ; POD Tx byte
00000049		00124	REG_PODRX	equ BCR_BASE + 19 ; POD Rx byte
		00125		
		00126		
		00127		

LOC	OBJECT CODE	LINE	SOURCE	TEXT
VALUE				
		00128	page	
		00129	;	*****
		00130	;	Start of code and interrupt vectors
		00131	;	*****
		00132		
0000		00133	org	0x000
0000		00134	reset	
0000 201B		00135	call	init_0 ; initialize the PIC
0001 2028		00136	call	init_bc ; initialize the ByteCode machine
0002		00137	sign_on	
0002 2028		00138	call	bc_id ; print the ID Banner then enable rx
0003		00139	idle	
0003 2803		00140	goto	idle ; wait till we get a prompt
		00141		
		00142		
		00143		
		00144		
		00145		

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00146	page
		00147	*****
		00148	; the serial interrupt edge will get us here - SOFTWARE UART
		00149	*****
0004		00150	
		00151	org 0x004
		00152	
0004		00153	int_vec ; GIE will be clear
		00154	; ? HALT the sample clock ?
		00155	
0004		00156	rx_char
		00157	
0004 3008		00158	movlw 8 ; bits/byte into
0005 00A4		00159	movwf count_0 ; count register
		00160	
0006 3024		00161	movlw START_CELL ; 1.5 serial bit width delay
0007 2051		00162	call t_256 ; - delay to centre of bit 0
0008		00163	rx_1
0008 1B06		00164	btfsc PORTB,SER_I ; if rx 1, then
0009 1403		00165	setc ; set carry bit
000A 1F06		00166	btfss PORTB,SER_I ; if rx 0, then
000B 1003		00167	clrc ; clear carry bit
		00168	
000C 0CA1		00169	rrf rx_data,F ; shift C right into rx_data
		00170	
000D 3013		00171	movlw (BIT_CELL-2) ;
000E 2051		00172	call t_256 ; delay to centre of next bit cell
		00173	
			Message[305]: Using default destination of 1 (file).
000F 0BA4		00174	decfsz count_0 ; loop 8 times
0010 2808		00175	goto rx_1 ;
0011		00176	rx_2
0011 1F06		00177	btfss PORTB,SER_I ; the serial in must be 1 now
0012 2811		00178	goto rx_2 ; or we'll wait!
		00179	
0013 0821		00180	movf rx_data,W ; put the new byte in W
0014 00A0		00181	movwf tx_data ; then in tx_data for echo
		00182	
0015 100B		00183	bcf INTCON,0 ; clear the RB Port Int flag
		00184	
		00185	*****
		00186	; fall thru to Byte Code Interpreter
		00187	*****
		00188	
0016		00189	atomic
0016 3003		00190	movlw JMP_PAGE ; high byte of table
0017 008A		00191	movwf PCLATH ; put it in the latch
		00192	
0018 0821		00193	movf rx_data,W ; get the byte code vector
0019 3880		00194	iorlw 0x80 ; jam the received byte to 80-ff
001A 0082		00195	movwf PCL ; - Geronimo....!
		00196	
		00197	; we just went over niagra falls
		00198	; in 5 program words
		00199	
		00200	
		00201	
		00202	
		00203	
		00204	

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00205	page
		00206	*****
		00207	Utility Code
		00208	*****
		00209	
		00210	
		00211	;
		00212	set up port initial conditions
		00213	;
		00214	;
		00215	
001B		00216	init_0
001B 300F		00217	movlw PORTA_INIT ; initial value
001C 0085		00218	movwf PORTA ; into porta
		00219	
001D 3085		00220	movlw TRISA ;
001E 0084		00221	movwf FSR ; point to TRISA
001F 3010		00222	movlw TRISA_INIT ; initial value tristate
0020 0080		00223	movwf INDF ; into tris register
		00224	
0021 303F		00225	movlw PORTB_INIT ; initial value
0022 0086		00226	movwf PORTB ; into portb
		00227	
0023 3086		00228	movlw TRISB ;
0024 0084		00229	movwf FSR ; point to TRISA
0025 3043		00230	movlw TRISB_INIT ; initial value tristate
0026 0080		00231	movwf INDF ; into tris register
		00232	
0027 0008		00233	return ;
		00234	
		00235	
		00236	
0028		00237	init_bc
		00238	
0028		00239	bc_id
		00240	
0028 3081		00241	movlw OPTION_REG ;
0029 0084		00242	movwf FSR ; point to OPTION
002A 30EF		00243	movlw OPTION_INI ; initial value
002B 0080		00244	movwf INDF ; into OPTION register
		00245	
002C 3008		00246	movlw INTCON_INI ; initial value
002D 008B		00247	movwf INTCON ; into INTCON
		00248	
		00249	
002E 236F		00250	call id_str ;
		00251	
002F 300D		00252	movlw 0x0d ; "CR"
0030 2059		00253	call prn_char ; send it
		00254	
0031 0008		00255	return ;
		00256	
		00257	
0032		00258	init_trace
		00259	
0032 3044		00260	movlw REG_CHOP ; initial channel value
0033 0085		00261	movwf PORTA ; into porta
		00262	
0034 3085		00263	movlw TRISA ;
0035 0084		00264	movwf FSR ; point to TRISA
0036 3010		00265	movlw TRISA_INIT ; initial value tristate
0037 0080		00266	movwf INDF ; into tris register
		00267	
0038 30EF		00268	movlw PORTB_TRACE ; initial value
0039 0086		00269	movwf PORTB ; into portb
		00270	
003A 3086		00271	movlw TRISB ;
003B 0084		00272	movwf FSR ; point to TRISA
003C 3041		00273	movlw TRISB_TRACE ; initial value tristate
003D 0080		00274	movwf INDF ; into tris register
		00275	
003E 0008		00276	return
		00277	
		00278	
		00279	

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00280	page
		00281	*****
		00282	; transmit the byte in tx_data
		00283	*****
		00284	
003F		00285	tx_char
		00286	
003F 3008		00287	movlw 8 ; bits/byte into
0040 00A4		00288	movwf count_0 ; count register
		00289	
0041 1286		00290	bcf PORTB,SER_0 ; set tx low for start bit
0042 3015		00291	movlw BIT_CELL ; serial bit width delay
0043 2051		00292	call t_256 ; delay
		00293	
0044		00294	tx_1
0044 0CA0		00295	rrf tx_data,F ; tx data from lsb
0045 1803		00296	btfsc STATUS,C ;
0046 1686		00297	bsf PORTB,SER_0 ; set tx bit high
0047 1C03		00298	btfss STATUS,C ;
0048 1286		00299	bcf PORTB,SER_0 ; set tx bit low
		00300	
0049 3013		00301	movlw (BIT_CELL-2) ;
004A 2051		00302	call t_256 ; delay
		00303	
004B 0BA4		00304	decfsz count_0,F ;
004C 2844		00305	goto tx_1 ; do the other bits
		00306	
004D 1686		00307	bsf PORTB,SER_0 ; set tx high for stop bit
004E 3015		00308	movlw BIT_CELL ; serial bit width delay
004F 2051		00309	call t_256 ; delay
		00310	
		00311	
0050 0008		00312	return ;
		00313	
		00314	
		00315	
		00316	
		00317	*****
		00318	; timer counts 1..255 (2uS..510uS)
		00319	*****
		00320	
0051		00321	t_256 ; entry here assumes count in W
		00322	
0051 00A2		00323	movwf timer_0 ; set timer
0052		00324	t_1
0052 0BA2		00325	decfsz timer_0,F ; start counting down
0053 2852		00326	goto t_1 ; loop til 0
		00327	
0054 0008		00328	return ; that is it
		00329	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
		00330	page	
		00331	;	*****
		00332	;	set int enable - a call here sets GIE
		00333	;	*****
		00334		
0055		00335	echo_null	
0055		00336	set_int	
0055 0009		00337	retfie	; resume code with GIE = 1
		00338		
		00339		
		00340	;	*****
		00341	;	echo byte back and restore int
		00342	;	*****
		00343		
		00344		
0056		00345	echo_char	
		00346		; test for echo flag here
		00347		
0056 203F		00348	call tx_char	; what is in tx_data is sent
0057 2055		00349	call set_int	; re-enable interrupts
0058 0008		00350	return	;
		00351		
		00352		
		00353	;	*****
		00354	;	print char in W register
		00355	;	*****
		00356		
0059		00357	prn_char	
		00358		
0059 00A0		00359	movwf tx_data	; put char in tx buffer
005A 203F		00360	call tx_char	; send it
005B 0008		00361	return	;
		00362		

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00363	page
		00364	*****
		00365	; convert byte in hex2asc -> ascii in ascii_l,ascii_h
		00366	*****
005C		00367	
		00368	hex_asc
		00369	
005C 082A		00370	movf hex2asc,W ; get the hex value
005D 390F		00371	andlw 0x0f ; mask off low nibble
005E 3E30		00372	addlw 0x30 ; add 30 (0..9)
005F 00AB		00373	movwf ascii_l ; put it in ascii_l
		00374	
0060 3C39		00375	sublw 0x39 ; is it greater than 9?
0061 1803		00376	skpnc ; no carry means neg result
0062 2865		00377	goto hex_1 ;
0063 3027		00378	movlw 0x27 ; convert to a..f
0064 07AB		00379	addwf ascii_l,F ; and save it
		00380	
0065		00381	hex_1
0065 082A		00382	movf hex2asc,W ; get the hex value
0066 39F0		00383	andlw 0xf0 ; mask off high nibble
0067 00AC		00384	movwf ascii_h ;
0068 0E2C		00385	swapf ascii_h,W ; swap nibbles
		00386	
0069 3E30		00387	addlw 0x30 ; add 30 (0..9)
006A 00AC		00388	movwf ascii_h ; put it in ascii_h
		00389	
006B 3C39		00390	sublw 0x39 ; is it greater than 9?
006C 1803		00391	skpnc ; no carry means neg result
006D 0008		00392	return ; so add 30 was OK
006E 3027		00393	movlw 0x27 ; else convert to a..f
006F 07AC		00394	addwf ascii_h,F ; and save it
		00395	
0070 0008		00396	return
		00397	
		00398	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
		00399	page	
		00400	;	*****
		00401	;	shift a byte into Spock and get the one shifted out
		00402	;	*****
		00403		
		00404		
0071		00405	shft_byte	
0071 3008		00406	movlw 8	; bit count
0072 00A4		00407	movwf count_0	;
		00408		
0073		00409	shft_0	; put bit 7 of gen_0 out to Spock
0073 1BA6		00410	btfsc gen_0,7	; is msb 0?
0074 1406		00411	bsf PORTB,RB0	; no - set Spock shift in
0075 1FA6		00412	btfss gen_0,7	; else
0076 1006		00413	bcf PORTB,RB0	; yes - clear Spock shift in
		00414		
		00415		; now put Spock Shift Out into Carry
0077 1886		00416	btfsc PORTB,RB1	; is 0?
0078 1403		00417	setc	; no - set carry
0079 1C86		00418	btfss PORTB,RB1	; else
007A 1003		00419	clrc	; yes - clear carry
		00420		
007B 1185		00421	bcf PORTA,RA3	; put zz-clk low
007C 0000		00422	nop	; heed the manufacturers warning!
007D 1585		00423	bsf PORTA,RA3	; then restore it
		00424		
007E 0DA6		00425	rlf gen_0,F	; shift the victim left
		00426		
007F 0BA4		00427	decfsz count_0,F	; loop counter
0080 2873		00428	goto shft_0	;
		00429		
0081 0008		00430	return	; gen_0 will be replaced with Spock data
		00431		
		00432		

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		00433	page
		00434	*****
		00435	; read a byte from the digital RAM
		00436	*****
		00437	
0082		00438	dig_rd
0082 1086		00439	bcf PORTB,SEL0 ;
0083 1106		00440	bcf PORTB,SEL1 ;
0084 1186		00441	bcf PORTB,SEL2 ; point to bit 0
		00442	
0085 1206		00443	bcf PORTB,RB4 ; we must be count
		00444	
0086 3008		00445	movlw 8 ; loop counter
0087 00A7		00446	movwf gen_1 ; don't kill other loop counts
0088		00447	dig_1
		00448	
0088 1A05		00449	btfsc PORTA,RA4 ; is 0?
0089 1403		00450	setc ; no - set carry
008A 1E05		00451	btfss PORTA,RA4 ; else
008B 1003		00452	clrc ; yes - clear carry
		00453	
008C 0CA6		00454	rff gen_0,F ; shift carry into gen_0
		00455	
008D 0A86		00456	incf PORTB,F ;
008E 0A86		00457	incf PORTB,F ; inc three SEL bits
		00458	
008F 0BA7		00459	decfsz gen_1,F ; loop
0090 2888		00460	goto dig_1 ; get next bit
		00461	
0091 1206		00462	bcf PORTB,RB4 ; we must be count
0092 0008		00463	return ; RAM byte is in gen_0
		00464	
		00465	
		00466	*****
		00467	; read a byte from tha analog RAM
		00468	*****
		00469	
0093		00470	ana_rd
0093 1086		00471	bcf PORTB,SEL0 ;
0094 1106		00472	bcf PORTB,SEL1 ;
0095 1186		00473	bcf PORTB,SEL2 ; point to bit 0
		00474	
0096 1206		00475	bcf PORTB,RB4 ; we must be count
		00476	
0097 3008		00477	movlw 8 ; loop counter
0098 00A7		00478	movwf gen_1 ; don't kill other loop counts
0099		00479	ana_1
		00480	
0099 1806		00481	btfsc PORTB,RB0 ; is 0?
009A 1403		00482	setc ; no - set carry
009B 1C06		00483	btfss PORTB,RB0 ; else
009C 1003		00484	clrc ; yes - clear carry
		00485	
009D 0CA6		00486	rff gen_0,F ; shift carry into gen_0
		00487	
009E 0A86		00488	incf PORTB,F ;
009F 0A86		00489	incf PORTB,F ; inc three SEL bits
		00490	
00A0 0BA7		00491	decfsz gen_1,F ; loop
00A1 2899		00492	goto ana_1 ; get next bit
		00493	
00A2 1206		00494	bcf PORTB,RB4 ; we must be count
00A3 0008		00495	return ; RAM byte is in gen_0
		00496	
		00497	
		00498	
		00499	

LOC	OBJECT CODE	VALUE	LINE	SOURCE	TEXT
			00500	page	
			00501	;	*****
			00502	;	Byte Code action routines - All serial chars will initiate one of these!
			00503	;	*****
0100			00504	org 0x100	; move up here
			00505		
			00506	;	the first group of Byte Code Ops are general ones
			00507	;	these will be followed by machine specific ones
			00508		
			00509	;	*****
			00510	;	no_op - don't respond
			00511	;	*****
			00512		
0100			00513	no_op	
0100 2055			00514	call	echo_null ; turn UART back on
0101 2803			00515	goto	idle ; then go back to waiting
			00516		
			00517		
			00518	;	*****
			00519	;	echo - respond with char received
			00520	;	*****
			00521		
0102			00522	echo	
0102 201B			00523	call	init_0 ; make sure the ports are OK
0103 2056			00524	call	echo_char ; send back what got us here
0104 2803			00525	goto	idle ; then just hang around for next char
			00526		
			00527		

LOC	OBJECT CODE	LINE	SOURCE	TEXT
		00528	page	
		00529	;	*****
		00530	;	[- clear the input register
		00531	;	*****
		00532		
0105		00533	clr_inp	
0105 01B0		00534	clrf	REG_INP ; input register
0106 2902		00535	goto	echo ; then echo it
		00536		
		00537		
		00538		
		00539	;	*****
		00540	;	0 -> f inc the register[0..3] to the value required
		00541	;	*****
		00542		
0107 0AB0		00543	nib_f incf	REG_INP,F ; input register value depends
0108 0AB0		00544	nib_e incf	REG_INP,F ; on entry point
0109 0AB0		00545	nib_d incf	REG_INP,F ;
010A 0AB0		00546	nib_c incf	REG_INP,F ;
010B 0AB0		00547	nib_b incf	REG_INP,F ;
010C 0AB0		00548	nib_a incf	REG_INP,F ;
010D 0AB0		00549	nib_9 incf	REG_INP,F ;
010E 0AB0		00550	nib_8 incf	REG_INP,F ;
010F 0AB0		00551	nib_7 incf	REG_INP,F ;
0110 0AB0		00552	nib_6 incf	REG_INP,F ;
0111 0AB0		00553	nib_5 incf	REG_INP,F ;
0112 0AB0		00554	nib_4 incf	REG_INP,F ;
0113 0AB0		00555	nib_3 incf	REG_INP,F ;
0114 0AB0		00556	nib_2 incf	REG_INP,F ;
0115 0AB0		00557	nib_1 incf	REG_INP,F ;
0116		00558	nib_0	
0116 0EB0		00559	swapf	REG_INP,F ; swap the nibbles
0117 2902		00560	goto	echo ; send back the byte - thats it!
		00561		

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		00562	page
		00563	*****
		00564	; @ - load the input register into the address pointer
		00565	*****
		00566	
0118		00567	addr_ld
0118 0830		00568	movf REG_INP,W ; txfer the input register to
0119 00B1		00569	movwf REG_ADD ; the address register
		00570	
011A 2902		00571	goto echo ;
		00572	
		00573	*****
		00574	; # - load the input register into the source pointer
		00575	*****
		00576	
011B		00577	src_ld
011B 0830		00578	movf REG_INP,W ; txfer the input register to
011C 00B2		00579	movwf REG_SRC ; the source register
		00580	
011D 2902		00581	goto echo ;
		00582	
		00583	*****
		00584	; L - load input register from register pointed to
		00585	*****
		00586	
011E		00587	load_reg
011E 0832		00588	movf REG_SRC,W ; get the SRC register
011F 3E30		00589	addlw BCR_BASE ; base address of bc regs
0120 0084		00590	movwf FSR ; put in index
		00591	
0121 0800		00592	movf INDF,W ; put REG contents in W
0122 00B0		00593	movwf REG_INP ; and shove it in input reg
		00594	
0123 2902		00595	goto echo ;
		00596	
		00597	
		00598	*****
		00599	; S - store input register to register pointed to
		00600	*****
		00601	
0124		00602	stor_reg
0124 0831		00603	movf REG_ADD,W ; get the ADDR register
0125 3E30		00604	addlw BCR_BASE ; base address of bc regs
0126 0084		00605	movwf FSR ; put in index
		00606	
0127 0830		00607	movf REG_INP,W ; get the contents of input
0128 00B0		00608	movwf INDF ; put it in the register
		00609	
0129 2902		00610	goto echo ;
		00611	
		00612	
		00613	*****
		00614	; 'n' - increment ADDR register
		00615	*****
		00616	
012A		00617	inc_add
012A 0AB1		00618	incf REG_ADD,F ; increment the ADDR register
		00619	
012B 2902		00620	goto echo ;
		00621	
		00622	
		00623	*****
		00624	; '+' - increment ADDR-> register
		00625	*****
		00626	
012C		00627	inc_reg
012C 0831		00628	movf REG_ADD,W ; get the ADDR register
012D 3E30		00629	addlw BCR_BASE ; base address of bc regs
012E 0084		00630	movwf FSR ; put in index
		00631	
012F 0A80		00632	incf INDF,F ; increment the ADDR-> register
		00633	
0130 2902		00634	goto echo ;
		00635	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
	VALUE	00636	page	
		00637	;	*****
		00638	;	'-' - decrement ADDR-> register
		00639	;	*****
		00640		
0131		00641	dec_reg	
0131 0831		00642	movf	REG_ADD,W ; get the ADDR register
0132 3E30		00643	addlw	BCR_BASE ; base address of bc regs
0133 0084		00644	movwf	FSR ; put in index
		00645		
0134 0380		00646	decf	INDF,F ; decrement the ADDR-> register
		00647		
0135 2902		00648	goto	echo ;
		00649		
		00650		

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		00651	page
		00652	;*****
		00653	; P - print value of register pointed to (ASCII)
		00654	;*****
		00655	
0136		00656	prn_reg
		00657	
0136 0831		00658	movf REG_ADD,W ; get the ADDR register
0137 3E30		00659	addlw BCR_BASE ; offset by base address of bc regs
0138 0084		00660	movwf FSR ; put in index
0139 0800		00661	movf INDF,W ; get the contents
013A 00AA		00662	movwf hex2asc ; and put it on the operating table
013B 205C		00663	call hex_asc ; 2 chars will be in ascii_l, ascii_h
		00664	
013C 2056		00665	call echo_char ; send the byte that got us here
		00666	
013D 300D		00667	movlw 0x0d ; "CR"
013E 2059		00668	call prn_char ; send it
013F 082C		00669	movf ascii_h,W ; get the high char
0140 2059		00670	call prn_char ; send it
0141 082B		00671	movf ascii_l,W ; get the low char
0142 2059		00672	call prn_char ; send it
0143 300D		00673	movlw 0x0d ; "CR"
0144 2059		00674	call prn_char ; send it
		00675	
0145 2803		00676	goto idle ; thats it
		00677	
		00678	
		00679	;*****
		00680	; ? - print ID of this Byte Code machine
		00681	;*****
		00682	
		00683	
0146		00684	prn_id
0146 2056		00685	call echo_char ; send back the '?'
		00686	
0147 300D		00687	movlw 0x0d ; "CR"
0148 2059		00688	call prn_char ; send it
0149 236F		00689	call id_str ; 8 ascii string eg 'BC000100'
014A 300D		00690	movlw 0x0d ; "CR"
014B 2059		00691	call prn_char ; send it
		00692	
014C 2803		00693	goto idle ; thats it
		00694	
		00695	
		00696	;*****
		00697	; u - update counter word from counter preload
		00698	;*****
		00699	
014D		00700	reg_up
014D 0833		00701	movf REG_COUNT_L,W ; count preload -> count value
014E 00B9		00702	movwf REG_SHIFT_L ;
		00703	
014F 0834		00704	movf REG_COUNT_H,W ;
0150 00C0		00705	movwf REG_SHIFT_H ;
		00706	
0151 2902		00707	goto echo ;
		00708	
		00709	

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00710	page
		00711	;*****
		00712	; > - load registers into Spock PLD
		00713	;*****
		00714	
0152		00715	
		00716	pld_ld
		00717	
0152 3085		00718	movlw TRISA ;
0153 0084		00719	movwf FSR ; point to TRISA
0154 1180		00720	bcf INDF,RA3 ; make sure clk is frozen
		00721	
0155 3086		00722	movlw TRISB ;
0156 0084		00723	movwf FSR ; point to TRISB
0157 1480		00724	bsf INDF,RB1 ; SEL0 is input
		00725	
		00726	
0158 1585		00727	bsf PORTA,RA3 ; clk is high
0159 1606		00728	bsf PORTB,RB4 ; set Spock to shift
015A 1386		00729	bcf PORTB,RB7 ; turn store off
		00730	
015B 1000		00731	bcf INDF,RB0 ; RB0 (A-DATA) output
		00732	
		00733	
015C 0837		00734	movf REG_CONTROL,W ; control byte
015D 00A6		00735	movwf gen_0 ; shift out reg
015E 2071		00736	call shft_byte ;
		00737	
015F 0836		00738	movf REG_MASK,W ; control byte
0160 00A6		00739	movwf gen_0 ; shift out reg
0161 2071		00740	call shft_byte ;
		00741	
0162 0835		00742	movf REG_LOGIC,W ; control byte
0163 00A6		00743	movwf gen_0 ; shift out reg
0164 2071		00744	call shft_byte ;
		00745	
0165 0834		00746	movf REG_COUNT_H,W ; control byte
0166 00A6		00747	movwf gen_0 ; shift out reg
0167 2071		00748	call shft_byte ;
		00749	
0168 0833		00750	movf REG_COUNT_L,W ; control byte
0169 00A6		00751	movwf gen_0 ; shift out reg
016A 2071		00752	call shft_byte ;
		00753	
		00754	
016B 1400		00755	bsf INDF,RB0 ; RB0 (A-DATA) input
		00756	
		00757	
016C 294D		00758	goto reg_up ; update count regs, echo return
		00759	

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00760	page
		00761	*****
		00762	< - read count registers from Spock PLD
		00763	*****
		00764	
		00765	
016D		00766	pld_rd
		00767	
016D 216F		00768	call spock_out ; call as a subroutine
016E 2902		00769	goto echo ; echo return
		00770	
016F		00771	spock_out
016F 3085		00772	movlw TRISA ;
0170 0084		00773	movwf FSR ; point to TRISA
0171 1180		00774	bcf INDF,RA3 ; make sure clk is frozen
		00775	
0172 3086		00776	movlw TRISB ;
0173 0084		00777	movwf FSR ; point to TRISB
0174 1480		00778	bsf INDF,RB1 ; SEL0 is input
		00779	
		00780	
0175 1585		00781	bsf PORTA,RA3 ; clk is high
0176 1606		00782	bsf PORTB,RB4 ; set Spock to shift
0177 1386		00783	bcf PORTB,RB7 ; turn store off
		00784	
0178 1000		00785	bcf INDF,RB0 ; RB0 (A-DATA) output
		00786	
		00787	
0179 0837		00788	movf REG_CONTROL,W ; control byte
017A 00A6		00789	movwf gen_0 ; shift out reg
017B 2071		00790	call shft_byte ;
017C 0826		00791	movf gen_0,W ; get byte
017D 00C0		00792	movwf REG_SHIFT_H ; put it away
		00793	
017E 0836		00794	movf REG_MASK,W ; mask byte
017F 00A6		00795	movwf gen_0 ; shift out reg
0180 2071		00796	call shft_byte ;
0181 0826		00797	movf gen_0,W ; get byte
0182 00B9		00798	movwf REG_SHIFT_L ; put it away
		00799	
0183 0835		00800	movf REG_LOGIC,W ; logic byte
0184 00A6		00801	movwf gen_0 ; shift out reg
0185 2071		00802	call shft_byte ;
		00803	
0186 0840		00804	movf REG_SHIFT_H,W ; shift byte
0187 00A6		00805	movwf gen_0 ; shift out reg
0188 2071		00806	call shft_byte ;
		00807	
0189 0839		00808	movf REG_SHIFT_L,W ; shift byte
018A 00A6		00809	movwf gen_0 ; shift out reg
018B 2071		00810	call shft_byte ;
		00811	
018C 1400		00812	bsf INDF,RB0 ; RB0 (A-DATA) input
		00813	
018D 0008		00814	return
		00815	
		00816	
		00817	
		00818	

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		00819	page
		00820	*****
		00821	; S dump Sample RAM contents
		00822	*****
		00823	
018E		00824	data_dmp
		00825	
018E 1585		00826	bsf PORTA,RA3 ; clk is high
		00827	
018F 3085		00828	movlw TRISA ;
0190 0084		00829	movwf FSR ; point to TRISA
0191 1180		00830	bcf INDF,RA3 ; make sure clk is frozen
		00831	
		00832	
0192 3086		00833	movlw TRISB ;
0193 0084		00834	movwf FSR ; point to TRISB
0194 1080		00835	bcf INDF,RB1 ; SEL0 is output
		00836	
		00837	
0195 1206		00838	bcf PORTB,RB4 ; set Spock to count
0196 1386		00839	bcf PORTB,RB7 ; turn store off - we want to read
		00840	
0197 2056		00841	call echo_char ; reply
		00842	
0198 300D		00843	movlw 0x0d ; "CR"
0199 2059		00844	call prn_char ; send it
		00845	
019A 0845		00846	movf REG_DUMP,W ; get the number of samples to send
019B 00A5		00847	movwf count_1 ; this is sample dump counter
		00848	
019C		00849	prn_row
019C 3016		00850	movlw 16 ; no of cols
019D 00A4		00851	movwf count_0 ; loop counter for line
		00852	
019E		00853	prn_col
019E 2082		00854	call dig_rd ; digital sample to gen_0
019F 0826		00855	movf gen_0,W ; -> W
01A0 00AA		00856	movwf hex2asc ; and put it on the operating table
01A1 205C		00857	call hex_asc ; 2 chars will be in ascii_l, ascii_h
01A2 082C		00858	movf ascii_h,W ; get the high char
01A3 2059		00859	call prn_char ; send it
01A4 082B		00860	movf ascii_l,W ; get the low char
01A5 2059		00861	call prn_char ; send it
		00862	
01A6 2093		00863	call ana_rd ; analog sample to gen_0
01A7 0826		00864	movf gen_0,W ; -> W
01A8 00AA		00865	movwf hex2asc ; and put it on the operating table
01A9 205C		00866	call hex_asc ; 2 chars will be in ascii_l, ascii_h
01AA 082C		00867	movf ascii_h,W ; get the high char
01AB 2059		00868	call prn_char ; send it
01AC 082B		00869	movf ascii_l,W ; get the low char
01AD 2059		00870	call prn_char ; send it
		00871	
01AE 302C		00872	movlw 0x2c ; "comma"
01AF 2059		00873	call prn_char ; send it
		00874	
01B0 1185		00875	bcf PORTA,RA3 ; pulse zz-clk low - _/\
01B1 1585		00876	bsf PORTA,RA3 ; inc counter in Spock
		00877	
01B2 0AB9		00878	incf REG_SHIFT_L,F ; inc the register to match
01B3 1903		00879	skpnz ;
01B4 0AC0		00880	incf REG_SHIFT_H,F ; carry to high byte
		00881	
01B5 0BA5		00882	decfsz count_1,F ; have we done enough?
01B6 29BB		00883	goto col_tst ; no - finish the row
		00884	
01B7 300D		00885	movlw 0x0d ; "CR"
01B8 2059		00886	call prn_char ; send it
		00887	
01B9 201B		00888	call init_0 ; shift saves power
		00889	
01BA 2803		00890	goto idle ; thats it
		00891	
01BB		00892	col_tst
01BB 0BA4		00893	decfsz count_0,F ;
01BC 299E		00894	goto prn_col ; keep going with row
		00895	
01BD 300D		00896	movlw 0x0d ; "CR"
01BE 2059		00897	call prn_char ; send it
01BF 299C		00898	goto prn_row ; do another row
		00899	
		00900	

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		00901	page
		00902	*****
		00903	; T Trace until TRIG then Delay and Freeze
		00904	*****
		00905	
01C0		00906	trace_go
01C0 2032		00907	call init_trace ; first set up the machine for sample
		00908	
01C1 2056		00909	call echo_char ; echo command that got us here
		00910	
		00911	
01C2 3003		00912	movlw high trace_opt ; high part of jump table address
01C3 008A		00913	movwf PCLATH ; put it in the latch
01C4 0838		00914	movf REG_OPTION,W ; low nibble is vector to trace loop
01C5 390F		00915	andlw 0x0f ; mask off the upper bits
01C6 3E50		00916	addlw low trace_opt ; patch 5 in - politically correct
01C7 0082		00917	movwf PCL ; vector off to the jump table
		00918	
		00919	; all the streams must end with a
		00920	; "goto rejoin"
		00921	
01C8		00922	rejoin
01C8 216F		00923	call spock_out ; get the current value of the counter
		00924	; we have to print this after a trigger
		00925	
01C9 300D		00926	movlw 0x0d ; "CR"
01CA 2059		00927	call prn_char ; send it
		00928	
01CB 0840		00929	movf REG_SHIFT_H,W ; get the contents
01CC 00AA		00930	movwf hex2asc ; and put it on the operating table
01CD 205C		00931	call hex_asc ; 2 chars will be in ascii_l, ascii_h
		00932	
01CE 2059		00933	call prn_char ; send it
01CF 082C		00934	movf ascii_h,W ; get the high char
01D0 2059		00935	call prn_char ; send it
01D1 082B		00936	movf ascii_l,W ; get the low char
01D2 2059		00937	call prn_char ; send it
		00938	
01D3 0839		00939	movf REG_SHIFT_L,W ; get the contents
01D4 00AA		00940	movwf hex2asc ; and put it on the operating table
01D5 205C		00941	call hex_asc ; 2 chars will be in ascii_l, ascii_h
		00942	
01D6 2059		00943	call prn_char ; send it
01D7 082C		00944	movf ascii_h,W ; get the high char
01D8 2059		00945	call prn_char ; send it
01D9 082B		00946	movf ascii_l,W ; get the low char
01DA 2059		00947	call prn_char ; send it
		00948	
		00949	
01DB 300D		00950	movlw 0x0d ; "CR"
01DC 2059		00951	call prn_char ; send it
		00952	
01DD 2803		00953	goto idle ; thats it
		00954	
		00955	
		00956	
		00957	

LOC	OBJECT CODE	LINE	SOURCE TEXT
		00958	page
		00959	*****
		00960	; POD Pass-Thru serial protocol
		00961	*****
		00962	
01DE		00963	pod_thru
		00964	
		00965	
01DE 1585		00966	bsf PORTA,RA3 ; clk is high
		00967	
01DF 3085		00968	movlw TRISA ;
01E0 0084		00969	movwf FSR ; point to TRISA
01E1 1180		00970	bcf INDF,RA3 ; make sure clk is frozen
01E2 1480		00971	bsf INDF,RA1 ; range 1 is input
		00972	
01E3 3086		00973	movlw TRISB ;
01E4 0084		00974	movwf FSR ; point to TRISB
01E5 1480		00975	bsf INDF,RB1 ; SEL0 is tristate
		00976	
01E6 1606		00977	bsf PORTB,RB4 ; shift mode connects POD
		00978	
01E7 2056		00979	call echo_char ; reply
		00980	
01E8		00981	tx_pod
01E8 0848		00982	movf REG_PODTX,W ; byte to send
01E9 00A0		00983	movwf tx_data ;
		00984	
01EA 3008		00985	movlw 8 ; bits/byte into
01EB 00A4		00986	movwf count_0 ; count register
		00987	
01EC 1005		00988	bcf PORTA,RA0 ; set tx low for start bit
01ED 3015		00989	movlw BIT_CELL ; serial bit width delay
01EE 2051		00990	call t_256 ; delay
		00991	
01EF		00992	txp_1
01EF 0CA0		00993	rxf tx_data,F ; tx data from lsb
01F0 1803		00994	btfsc STATUS,C ;
01F1 1405		00995	bsf PORTA,RA0 ; set tx bit high
01F2 1C03		00996	btfss STATUS,C ;
01F3 1005		00997	bcf PORTA,RA0 ; set tx bit low
		00998	
01F4 3013		00999	movlw (BIT_CELL-2) ;
01F5 2051		01000	call t_256 ; delay
		01001	
01F6 0BA4		01002	decfsz count_0,F ;
01F7 29EF		01003	goto txp_1 ; do the other bits
		01004	
01F8 1405		01005	bsf PORTA,RA0 ; set tx high for stop bit
01F9 3015		01006	movlw BIT_CELL ; serial bit width delay
01FA 2051		01007	call t_256 ; delay
		01008	
		01009	; we have sent the byte - now we go into a loop until a command
		01010	
01FB		01011	pod_1
		01012	
01FB 1885		01013	btfsc PORTA,RA1 ; is 0?
01FC 1686		01014	bsf PORTB,RB5 ; no - set serial out
01FD 1C85		01015	btfss PORTA,RA1 ; else
01FE 1286		01016	bcf PORTB,RB5 ; yes - clear serial out
01FF 29FB		01017	goto pod_1 ; loop until command
		01018	
		01019	
		01020	
		01021	
		01022	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
VALUE				
		01023	page	
		01024	;	*****
		01025	;	x POD byte exchange
		01026	;	*****
		01027		
0200		01028	pod_ex	
		01029		
0200 2902		01030	goto	echo ; not implemented yet
		01031		
		01032		
		01033		
		01034		

LOC	OBJECT CODE	LINE	SOURCE TEXT	VALUE
		01035	page	
		01036	;	
		01037	; All trace options (0..15) are coded here.	
		01038	;	
		01039		
		01040		
0201		01041	trace_0	
0201	30FF	01042	movlw 0xff	
0202	0081	01043	movwf TMR0	
0203	110B	01044	bcf INTCON,2	
		01045		
0204	0841	01046	movf REG_DELAY_L,W	
0205	00A6	01047	movwf gen_0	
0206	0AA6	01048	incf gen_0,F	
0207	0842	01049	movf REG_DELAY_H,W	
0208	00A7	01050	movwf gen_1	
0209	0AA7	01051	incf gen_1,F	
		01052		
020A	3085	01053	movlw TRISA	
020B	0084	01054	movwf FSR	
		01055		
020C	1580	01056	bsf INDF,RA3	
		01057		
020D		01058	tr_00	
020D	1A05	01059	btfsc PORTA,RA4	
020E	2A12	01060	goto tr_01	
020F	190B	01061	btfsc INTCON,2	
0210	2A12	01062	goto tr_01	
0211	2A0D	01063	goto tr_00	
		01064		
0212		01065	tr_01	
0212	0843	01066	movf REG_TBASE,W	
0213	2051	01067	call t_256	
0214	0BA6	01068	decfsz gen_0,F	
0215	2A12	01069	goto tr_01	
0216	0BA7	01070	decfsz gen_1,F	
0217	2A12	01071	goto tr_01	
		01072		
0218	1180	01073	bcf INDF,RA3	
		01074		
0219	29C8	01075	goto rejoin	
		01076		
		01077		

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		01078	page
		01079	
021A		01080	trace_1 ; trace 1 -
		01081	
		01082	
021A 29C8		01083	goto rejoin ;
		01084	
021B		01085	trace_2 ; trace 2 -
		01086	
		01087	
021B 29C8		01088	goto rejoin ;
		01089	
021C		01090	trace_3 ; trace 3 -
		01091	
		01092	
021C 29C8		01093	goto rejoin ;
		01094	
021D		01095	trace_4 ; trace 4 -
		01096	
		01097	
021D 29C8		01098	goto rejoin ;
		01099	
021E		01100	trace_5 ; trace 5 -
		01101	
		01102	
021E 29C8		01103	goto rejoin ;
		01104	
021F		01105	trace_6 ; trace 6 -
		01106	
		01107	
021F 29C8		01108	goto rejoin ;
		01109	
0220		01110	trace_7 ; trace 7 -
		01111	
		01112	
0220 29C8		01113	goto rejoin ;
		01114	
0221		01115	trace_8 ; trace 8 -
		01116	
		01117	
0221 29C8		01118	goto rejoin ;
		01119	
0222		01120	trace_9 ; trace 9 -
		01121	
		01122	
0222 29C8		01123	goto rejoin ;
		01124	
0223		01125	trace_10 ; trace 10 -
		01126	
		01127	
0223 29C8		01128	goto rejoin ;
		01129	
0224		01130	trace_11 ; trace 11 -
		01131	
		01132	
0224 29C8		01133	goto rejoin ;
		01134	
0225		01135	trace_12 ; trace 12 -
		01136	
		01137	
0225 29C8		01138	goto rejoin ;
		01139	
0226		01140	trace_13 ; trace 13 -
		01141	
		01142	
0226 29C8		01143	goto rejoin ;
		01144	
0227		01145	trace_14 ; trace 14 -
		01146	
		01147	
0227 29C8		01148	goto rejoin ;
		01149	
0228		01150	trace_15 ; trace 15 -
		01151	
		01152	
0228 29C8		01153	goto rejoin ;
		01154	
		01155	
		01156	
		01157	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
VALUE				
		01158	page	
		01159	;	*****
		01160	;	Jump vectors to sample options live up here before jump table
		01161	;	*****
		01162		
0350		01163	org	0x350
		01164		
0350		01165	trace_opt	
		01166		
0350 2A01		01167	opt_0 goto	trace_0 ; trace 0 -
		01168		
0351 2A1A		01169	opt_1 goto	trace_1 ; trace 1 -
		01170		
0352 2A1B		01171	opt_2 goto	trace_2 ; trace 2 -
		01172		
0353 2A1C		01173	opt_3 goto	trace_3 ; trace 3 -
		01174		
0354 2A1D		01175	opt_4 goto	trace_4 ; trace 4 -
		01176		
0355 2A1E		01177	opt_5 goto	trace_5 ; trace 5 -
		01178		
0356 2A1F		01179	opt_6 goto	trace_6 ; trace 6 -
		01180		
0357 2A20		01181	opt_7 goto	trace_7 ; trace 7 -
		01182		
0358 2A21		01183	opt_8 goto	trace_8 ; trace 8 -
		01184		
0359 2A22		01185	opt_9 goto	trace_9 ; trace 9 -
		01186		
035A 2A23		01187	opt_10 goto	trace_10 ; trace 10 -
		01188		
035B 2A24		01189	opt_11 goto	trace_11 ; trace 11 -
		01190		
035C 2A25		01191	opt_12 goto	trace_12 ; trace 12 -
		01192		
035D 2A26		01193	opt_13 goto	trace_13 ; trace 13 -
		01194		
035E 2A27		01195	opt_14 goto	trace_14 ; trace 14 -
		01196		
035F 2A28		01197	opt_15 goto	trace_15 ; trace 15 -
		01198		
		01199		
		01200		
		01201		

LOC	OBJECT CODE	LINE	SOURCE	TEXT
		01202	page	
		01203	;	*****
		01204	;	ID code lives up before jump table
		01205	;	*****
		01206		
036F		01207	org	0x36f
036F		01208	id_str	
036F 3042		01209	movlw	'B' ;
0370 2059		01210	call	prn_char ; send it
0371 3043		01211	movlw	'C' ;
0372 2059		01212	call	prn_char ; send it
0373 3030		01213	movlw	'0' ;
0374 2059		01214	call	prn_char ; send it
0375 3030		01215	movlw	'0' ;
0376 2059		01216	call	prn_char ; send it
0377 3030		01217	movlw	'0' ;
0378 2059		01218	call	prn_char ; send it
0379 3031		01219	movlw	'1' ;
037A 2059		01220	call	prn_char ; send it
037B 3030		01221	movlw	'0' ;
037C 2059		01222	call	prn_char ; send it
037D 3030		01223	movlw	'0' ;
037E 2059		01224	call	prn_char ; send it
		01225		
037F 0008		01226	return	;
		01227		
		01228		
		01229		

LOC	OBJECT CODE	LINE	SOURCE TEXT
VALUE			
		01230	page
		01231	*****
		01232	; Main Jump Table for ByteCode - RANGE 00 -> 7F
		01233	*****
		01234	
0380		01235	org 0x380 ; locate table up here at 380 -> 3ff
		01236	
0380 2800		01237	bc_00 goto reset ; "NUL" will always restart the machine
0381 2900		01238	bc_01 goto no_op ;
0382 2900		01239	bc_02 goto no_op ;
0383 2900		01240	bc_03 goto no_op ;
0384 2900		01241	bc_04 goto no_op ;
0385 2900		01242	bc_05 goto no_op ;
0386 2900		01243	bc_06 goto no_op ;
0387 2902		01244	bc_07 goto echo ;
0388 2900		01245	bc_08 goto no_op ;
0389 2900		01246	bc_09 goto no_op ;
038A 2902		01247	bc_0a goto echo ;
038B 2900		01248	bc_0b goto no_op ;
038C 2900		01249	bc_0c goto no_op ;
038D 2902		01250	bc_0d goto echo ;
038E 2900		01251	bc_0e goto no_op ;
038F 2900		01252	bc_0f goto no_op ;
		01253	
0390 2900		01254	bc_10 goto no_op ;
0391 2900		01255	bc_11 goto no_op ;
0392 2900		01256	bc_12 goto no_op ;
0393 2900		01257	bc_13 goto no_op ;
0394 2900		01258	bc_14 goto no_op ;
0395 2900		01259	bc_15 goto no_op ;
0396 2900		01260	bc_16 goto no_op ;
0397 2900		01261	bc_17 goto no_op ;
0398 2900		01262	bc_18 goto no_op ;
0399 2900		01263	bc_19 goto no_op ;
039A 2900		01264	bc_1a goto no_op ;
039B 2900		01265	bc_1b goto no_op ;
039C 2900		01266	bc_1c goto no_op ;
039D 2900		01267	bc_1d goto no_op ;
039E 2900		01268	bc_1e goto no_op ;
039F 2900		01269	bc_1f goto no_op ;
		01270	
03A0 2902		01271	bc_20 goto echo ; " " Space
03A1 2902		01272	bc_21 goto echo ; "!"
03A2 2902		01273	bc_22 goto echo ; ""
03A3 291B		01274	bc_23 goto src_ld ; "#" Load R0 into R2
03A4 2902		01275	bc_24 goto echo ; "\$"
03A5 2902		01276	bc_25 goto echo ; "%"
03A6 2902		01277	bc_26 goto echo ; "&"
03A7 2902		01278	bc_27 goto echo ; "+"
03A8 2902		01279	bc_28 goto echo ; "("
03A9 2902		01280	bc_29 goto echo ; ")"
03AA 2902		01281	bc_2a goto echo ; "*"
03AB 292C		01282	bc_2b goto inc_reg ; "+" increment @R1
03AC 2902		01283	bc_2c goto echo ; ","
03AD 2931		01284	bc_2d goto dec_reg ; "-" decrement @R1
03AE 2902		01285	bc_2e goto echo ; "."
03AF 2902		01286	bc_2f goto echo ; "/"
		01287	
03B0 2916		01288	bc_30 goto nib_0 ; "0" enter hex value 0
03B1 2915		01289	bc_31 goto nib_1 ; "1" enter hex value 1
03B2 2914		01290	bc_32 goto nib_2 ; "2" enter hex value 2
03B3 2913		01291	bc_33 goto nib_3 ; "3" enter hex value 3
03B4 2912		01292	bc_34 goto nib_4 ; "4" enter hex value 4
03B5 2911		01293	bc_35 goto nib_5 ; "5" enter hex value 5
03B6 2910		01294	bc_36 goto nib_6 ; "6" enter hex value 6
03B7 290F		01295	bc_37 goto nib_7 ; "7" enter hex value 7
03B8 290E		01296	bc_38 goto nib_8 ; "8" enter hex value 8
03B9 290D		01297	bc_39 goto nib_9 ; "9" enter hex value 9
03BA 2902		01298	bc_3a goto echo ; ":"
03BB 2902		01299	bc_3b goto echo ; ";"
03BC 296D		01300	bc_3c goto pld_rd ; "<" Read counter from Spock
03BD 2902		01301	bc_3d goto echo ; "="
03BE 2952		01302	bc_3e goto pld_ld ; ">" Load presets into Spock
03BF 2946		01303	bc_3f goto prn_id ; "?" print chip ID
		01304	

LOC	OBJECT CODE	LINE	SOURCE	TEXT
		01305	page	
03C0	2918	01306	bc_40	goto addr_ld ; "@" load R0 into R1
03C1	2902	01307	bc_41	goto echo ; "A"
03C2	2902	01308	bc_42	goto echo ; "B"
03C3	2902	01309	bc_43	goto echo ; "C"
03C4	2902	01310	bc_44	goto echo ; "D"
03C5	2902	01311	bc_45	goto echo ; "E"
03C6	2902	01312	bc_46	goto echo ; "F"
03C7	2902	01313	bc_47	goto echo ; "G"
03C8	2902	01314	bc_48	goto echo ; "H"
03C9	2902	01315	bc_49	goto echo ; "I"
03CA	2902	01316	bc_4a	goto echo ; "J"
03CB	2902	01317	bc_4b	goto echo ; "K"
03CC	2902	01318	bc_4c	goto echo ; "L"
03CD	2902	01319	bc_4d	goto echo ; "M"
03CE	2902	01320	bc_4e	goto echo ; "N"
03CF	2902	01321	bc_4f	goto echo ; "O"
		01322		
03D0	2902	01323	bc_50	goto echo ; "P"
03D1	2902	01324	bc_51	goto echo ; "Q"
03D2	2902	01325	bc_52	goto echo ; "R"
03D3	298E	01326	bc_53	goto data_dmp ; "S" Dump Sample RAM csv format
03D4	29C0	01327	bc_54	goto trace_go ; "T" The big Sample with TRIG
03D5	2902	01328	bc_55	goto echo ; "U"
03D6	2902	01329	bc_56	goto echo ; "V"
03D7	2902	01330	bc_57	goto echo ; "W"
03D8	2902	01331	bc_58	goto echo ; "X"
03D9	2902	01332	bc_59	goto echo ; "Y"
03DA	2902	01333	bc_5a	goto echo ; "Z"
03DB	2905	01334	bc_5b	goto clr_inp ; "[" clear INPUT register
03DC	2902	01335	bc_5c	goto echo ; "\"
03DD	2916	01336	bc_5d	goto nib_0 ; "]" swap the input nibbles
03DE	2902	01337	bc_5e	goto echo ; "^"
03DF	2902	01338	bc_5f	goto echo ; "_"
		01339		
03E0	2902	01340	bc_60	goto echo ; "`"
03E1	290C	01341	bc_61	goto nib_a ; "a" enter hex value A
03E2	290B	01342	bc_62	goto nib_b ; "b" enter hex value B
03E3	290A	01343	bc_63	goto nib_c ; "c" enter hex value C
03E4	2909	01344	bc_64	goto nib_d ; "d" enter hex value D
03E5	2908	01345	bc_65	goto nib_e ; "e" enter hex value E
03E6	2907	01346	bc_66	goto nib_f ; "f" enter hex value F
03E7	2902	01347	bc_67	goto echo ; "g"
03E8	2902	01348	bc_68	goto echo ; "h"
03E9	2902	01349	bc_69	goto echo ; "i"
03EA	2902	01350	bc_6a	goto echo ; "j"
03EB	2902	01351	bc_6b	goto echo ; "k"
03EC	291E	01352	bc_6c	goto load_reg ; "l" load input_reg from @R2
03ED	2902	01353	bc_6d	goto echo ; "m"
03EE	292A	01354	bc_6e	goto inc_add ; "n" next address
03EF	2902	01355	bc_6f	goto echo ; "o"
		01356		
03F0	2936	01357	bc_70	goto prn_reg ; "p" print the contents of @R1
03F1	2902	01358	bc_71	goto echo ; "q"
03F2	2902	01359	bc_72	goto echo ; "r"
03F3	2924	01360	bc_73	goto stor_reg ; "s" store input_reg to @R1
03F4	2902	01361	bc_74	goto echo ; "t"
03F5	294D	01362	bc_75	goto reg_up ; "u" update current RAM address
03F6	2902	01363	bc_76	goto echo ; "v"
03F7	2902	01364	bc_77	goto echo ; "w"
03F8	2A00	01365	bc_78	goto pod_ex ; "x"
03F9	2902	01366	bc_79	goto echo ; "y"
03FA	2902	01367	bc_7a	goto echo ; "z"
03FB	2902	01368	bc_7b	goto echo ; "{"
03FC	29DE	01369	bc_7c	goto pod_thru ; " "
03FD	2902	01370	bc_7d	goto echo ; "]"
03FE	2902	01371	bc_7e	goto echo ; "_"
03FF	2900	01372	bc_7f	goto no_op ; "DEL"
		01373		
		01374		END

SYMBOL TABLE

LABEL	VALUE
BCR_BASE	00000030
BIT_CELL	00000015
C	00000000
DC	00000001
EEADR	00000009
EECON1	00000088
EECON2	00000089
EEDATA	00000008
EEIE	00000006
EEIF	00000004
F	00000001
FSR	00000004
GIE	00000007
INDF	00000000
INTCON	0000000B
INTCON_INI	00000008
INTE	00000004
INTEDG	00000006
INTF	00000001
IRP	00000007
JMP_PAGE	00000003
NOT_PD	00000003
NOT_RBPU	00000007
NOT_TO	00000004
OPTION_INI	000000EF
OPTION_REG	00000081
PCL	00000002
PCLATH	0000000A
PORTA	00000005
PORTA_INIT	0000000F
PORTB	00000006
PORTB_INIT	0000003F
PORTB_TRACE	000000EF
PS0	00000000
PS1	00000001
PS2	00000002
PSA	00000003
RA0	00000000
RA1	00000001
RA2	00000002
RA3	00000003
RA4	00000004
RB0	00000000
RB1	00000001
RB2	00000002
RB3	00000003
RB4	00000004
RB5	00000005
RB6	00000006
RB7	00000007
RBIE	00000003
RBIF	00000000
RD	00000000
REG_ADD	00000031
REG_CHOP	00000044
REG_CONTROL	00000037
REG_COUNT_H	00000034
REG_COUNT_L	00000033
REG_DELAY_H	00000042
REG_DELAY_L	00000041
REG_DUMP	00000045
REG_EEADDR	00000047
REG_EEDATA	00000046
REG_INP	00000030
REG_LOGIC	00000035
REG_MASK	00000036
REG_OPTION	00000038
REG_PODRX	00000049
REG_PODTX	00000048
REG_SHIFT_H	00000040
REG_SHIFT_L	00000039
REG_SRC	00000032
REG_TBASE	00000043
RP0	00000005
RP1	00000006
SEL0	00000001
SEL1	00000002
SEL2	00000003
SER_I	00000006
SER_O	00000005
START_CELL	00000024
STATUS	00000003
T0CS	00000005

T0IE	00000005
T0IF	00000002
T0SE	00000004

SYMBOL TABLE
LABEL

VALUE

TMR0	00000001
TRISA	00000085
TRISA_INIT	00000010
TRISB	00000086
TRISB_INIT	00000043
TRISB_TRACE	00000041
W	00000000
WR	00000001
WREN	00000002
WRERR	00000003
Z	00000002
_CP_OFF	00003FFF
_CP_ON	0000000F
_HS_OSC	00003FFE
_LP_OSC	00003FFC
_PWRTE_OFF	00003FFF
_PWRTE_ON	00003FF7
_RC_OSC	00003FFF
_WDT_OFF	00003FFB
_WDT_ON	00003FFF
_XT_OSC	00003FFD
_16F84	00000001
addr_ld	00000118
ana_1	00000099
ana_rd	00000093
ascii_h	0000002C
ascii_l	0000002B
atomic	00000016
bc_00	00000380
bc_01	00000381
bc_02	00000382
bc_03	00000383
bc_04	00000384
bc_05	00000385
bc_06	00000386
bc_07	00000387
bc_08	00000388
bc_09	00000389
bc_0a	0000038A
bc_0b	0000038B
bc_0c	0000038C
bc_0d	0000038D
bc_0e	0000038E
bc_0f	0000038F
bc_10	00000390
bc_11	00000391
bc_12	00000392
bc_13	00000393
bc_14	00000394
bc_15	00000395
bc_16	00000396
bc_17	00000397
bc_18	00000398
bc_19	00000399
bc_1a	0000039A
bc_1b	0000039B
bc_1c	0000039C
bc_1d	0000039D
bc_1e	0000039E
bc_1f	0000039F
bc_20	000003A0
bc_21	000003A1
bc_22	000003A2
bc_23	000003A3
bc_24	000003A4
bc_25	000003A5
bc_26	000003A6
bc_27	000003A7
bc_28	000003A8
bc_29	000003A9
bc_2a	000003AA
bc_2b	000003AB
bc_2c	000003AC
bc_2d	000003AD
bc_2e	000003AE
bc_2f	000003AF
bc_30	000003B0
bc_31	000003B1
bc_32	000003B2
bc_33	000003B3
bc_34	000003B4
bc_35	000003B5
bc_36	000003B6

bc_37
bc_38
bc_39

000003B7
000003B8
000003B9

SYMBOL TABLE
LABEL

VALUE

bc_3a	000003BA
bc_3b	000003BB
bc_3c	000003BC
bc_3d	000003BD
bc_3e	000003BE
bc_3f	000003BF
bc_40	000003C0
bc_41	000003C1
bc_42	000003C2
bc_43	000003C3
bc_44	000003C4
bc_45	000003C5
bc_46	000003C6
bc_47	000003C7
bc_48	000003C8
bc_49	000003C9
bc_4a	000003CA
bc_4b	000003CB
bc_4c	000003CC
bc_4d	000003CD
bc_4e	000003CE
bc_4f	000003CF
bc_50	000003D0
bc_51	000003D1
bc_52	000003D2
bc_53	000003D3
bc_54	000003D4
bc_55	000003D5
bc_56	000003D6
bc_57	000003D7
bc_58	000003D8
bc_59	000003D9
bc_5a	000003DA
bc_5b	000003DB
bc_5c	000003DC
bc_5d	000003DD
bc_5e	000003DE
bc_5f	000003DF
bc_60	000003E0
bc_61	000003E1
bc_62	000003E2
bc_63	000003E3
bc_64	000003E4
bc_65	000003E5
bc_66	000003E6
bc_67	000003E7
bc_68	000003E8
bc_69	000003E9
bc_6a	000003EA
bc_6b	000003EB
bc_6c	000003EC
bc_6d	000003ED
bc_6e	000003EE
bc_6f	000003EF
bc_70	000003F0
bc_71	000003F1
bc_72	000003F2
bc_73	000003F3
bc_74	000003F4
bc_75	000003F5
bc_76	000003F6
bc_77	000003F7
bc_78	000003F8
bc_79	000003F9
bc_7a	000003FA
bc_7b	000003FB
bc_7c	000003FC
bc_7d	000003FD
bc_7e	000003FE
bc_7f	000003FF
bc_id	00000028
clr_inp	00000105
col_tst	000001BB
count_0	00000024
count_1	00000025
data_dmp	0000018E
dec_reg	00000131
dig_1	00000088
dig_rd	00000082
echo	00000102
echo_char	00000056
echo_null	00000055
gen_0	00000026

gen_1	00000027
gen_2	00000028
gen_3	00000029

SYMBOL TABLE

LABEL	VALUE
hex2asc	0000002A
hex_1	00000065
hex_asc	0000005C
id_str	0000036F
idle	00000003
inc_add	0000012A
inc_reg	0000012C
init_0	0000001B
init_bc	00000028
init_trace	00000032
int_vec	00000004
load_reg	0000011E
nib_0	00000116
nib_1	00000115
nib_2	00000114
nib_3	00000113
nib_4	00000112
nib_5	00000111
nib_6	00000110
nib_7	0000010F
nib_8	0000010E
nib_9	0000010D
nib_a	0000010C
nib_b	0000010B
nib_c	0000010A
nib_d	00000109
nib_e	00000108
nib_f	00000107
no_op	00000100
opt_0	00000350
opt_1	00000351
opt_10	0000035A
opt_11	0000035B
opt_12	0000035C
opt_13	0000035D
opt_14	0000035E
opt_15	0000035F
opt_2	00000352
opt_3	00000353
opt_4	00000354
opt_5	00000355
opt_6	00000356
opt_7	00000357
opt_8	00000358
opt_9	00000359
pld_ld	00000152
pld_rd	0000016D
pod_1	000001FB
pod_ex	00000200
pod_thru	000001DE
prn_char	00000059
prn_col	0000019E
prn_id	00000146
prn_reg	00000136
prn_row	0000019C
reg_up	0000014D
rejoin	000001C8
reset	00000000
rx_1	00000008
rx_2	00000011
rx_char	00000004
rx_data	00000021
set_int	00000055
shft_0	00000073
shft_byte	00000071
sign_on	00000002
spock_out	0000016F
src_ld	0000011B
stor_reg	00000124
t_1	00000052
t_256	00000051
timer_0	00000022
timer_1	00000023
tr_00	0000020D
tr_01	00000212
trace_0	00000201
trace_1	0000021A
trace_10	00000223
trace_11	00000224
trace_12	00000225
trace_13	00000226
trace_14	00000227
trace_15	00000228

trace_2	0000021B
trace_3	0000021C
trace_4	0000021D

SYMBOL TABLE

LABEL	VALUE
trace_5	0000021E
trace_6	0000021F
trace_7	00000220
trace_8	00000221
trace_9	00000222
trace_go	000001C0
trace_opt	00000350
tx_1	00000044
tx_char	0000003F
tx_data	00000020
tx_pod	000001E8
txp_1	000001EF

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

```
0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXX-----
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXX-----
0340 : ----- XXXXXXXXXXXXXXXX -----X XXXXXXXXXXXXXXXX
0380 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
03C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

All other memory blocks unused.

Program Memory Words Used: 622
Program Memory Words Free: 402

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 1 reported, 0 suppressed