MOTOROLA SEMICONDUCTOR APPLICATION NOTE

A Serial Bootloader for Reprogramming the MC68HC912B32 Flash EEPROM

By Gordon Doughman, Field Applications Engineer Software Specialist

1 Introduction

The MC68HC912B32 is a member of the M68HC12 family of 16-bit microcontrollers. It contains 32,768 bytes of bulk-erasable, byte- or word-programmable Flash EEPROM memory. Including Flash EE-PROM, rather than EPROM or ROM, memory on a microcontroller has significant advantages for both the OEM and the end customer.

For the OEM, placing system firmware in Flash EEPROM memory provides numerous benefits. First, firmware development can be extended late into the product development cycle by eliminating the ROM lead times. Second, when an OEM has several products based on the same microcontroller, it can help reduce the inventory problems associated with ROM-based microcontrollers. Finally, if a severe bug is found in the product's firmware during the manufacturing process, the in-circuit reprogrammability of Flash EEPROM memory prevents the OEM from having to scrap any of the work-in-process.

The ability of Flash EEPROM memory to be electrically erased and reprogrammed also provides benefits for the OEM's end customers. The customers' products can be updated or enhanced with new features and capabilities without having to replace any components or return the product to the factory.

Unlike the M68HC11 family, the MC68HC912B32 does not have a Bootstrap ROM containing firmware that allows initial programming of the Flash EEPROM directly through the on-chip Serial Communications Interface (SCI) port. Initial on-chip Flash EEPROM programming requires either special test and handling equipment to program the device before it is placed in the target system or a programming tool such as the SDI12 or the M68EVB912B32, available from Motorola, that is capable of programming the Flash EEPROM through the Real Time Background Debug interface.

The M68EVB912B32 on-chip Flash EEPROM, however, does contain a 2k-byte erase-protected bootblock. The bootblock may be used to contain a special bootloader program that allows erasure and programming of the remaining 30k of the on-chip Flash. In addition to implementing the Flash programming and erase algorithms, the serial bootloader firmware may contain a simple serial communications protocol that allows the use of the on-chip SCI port for obtaining the data to be programmed into the Flash.

Programming and erasing the on-chip Flash EEPROM memory of the MC68HC912B32 presents some unique challenges. Even though the on-chip Flash EEPROM memory has an erase-protected bootblock to contain the firmware implementing the programming and erase algorithms, the code cannot be run directly out of the Flash EEPROM bootblock while the remainder of the Flash array is being erased or programmed. Consequently, during the erase and reprogram process, the code must reside in other on-chip memory or in external memory. In addition, because the erase protected bootblock resides in the top 2k of the memory map (\$F800—\$FFFF), the reset and interrupt vectors cannot be changed without erasing the entire bootblock. This necessitates that a secondary reset/interrupt vector table be placed outside of the 2k bootblock.

The remainder of this application note will explore the requirements of a serial bootloader and the implementation of the programming algorithm for the MC68HC912B32 Flash EEPROM.



2 Overview of the MC68HC912B32's Flash EEPROM

The MC68HC912B32 Flash EEPROM module is arranged as a 16,384 x 16-bit module and may be read as bytes or aligned or misaligned words. Programming is accomplished only by writing bytes or aligned words. The Flash module requires an externally applied program/erase voltage (V_{FP}) to program or erase the array. The program/erase voltage is applied statically to the V_{FP} pin, however, the V_{FP} pin must always be kept at greater-than-or-equal to V_{DD} -0.5 volts to prevent damage to the Flash array. To prevent the accidental erasure or programming of the Flash array, the V_{FP} should only be applied during the program/erase procedure.

Like most external Flash memory devices, the MC68HC912B32 Flash EEPROM module does not provide any automatic timing sequences during the erase or programming cycles. Programming or erasure is accomplished by a sequence of timed writes to the Flash control registers and a byte or aligned word write to the Flash array itself. The programming firmware is entirely responsible for the implementation of the erase and programming algorithms.

2.1 Erasure of the Flash EEPROM Array

Erasure of the MC68HC912B32 Flash EEPROM involves a procedure that can be divided into two parts. Erase pulses to the Flash array are applied by manipulating bits in the FEECTL register. After a pulse is applied, each location of the Flash array is checked for an erased state. When all locations in the Flash array are found to be in the erased state, or the maximum number of erase pulses have been applied, the same number of erase pulses required to erase the array are applied again. This procedure provides a 100% erase margin to the Flash array. After the margin pulses are applied, the Flash array should again be checked to ensure that it was properly erased. The simplified flowchart shown in Figure 1 describes these steps. Detailed descriptions and flowcharts, including timing requirements, describing the Flash erase procedure can be found in the *MC68HC912B32 Technical Summary* (document number MC68HC912B32TS/D).



Figure 1 Simplified Flash Erase Algorithm Flowchart

2.2 Flash Array Programming

Programming the Flash array involves a procedure similar to the erase procedure. As mentioned previously, the MC68HC912B32 Flash may be programmed as either bytes or aligned words. Attempting to program a misaligned word of Flash memory will result in only the high byte (lower address) of the word being programmed into the Flash memory array. As with the erase procedure, programming the Flash involves applying a series of programming pulses to the Flash array by manipulating bits in the FEECTL register. After each pulse is applied, the programmed location is checked to ensure that it contains the proper data. After the location reaches the proper value, or the maximum number of programming pulses have been applied, the same number of pulses required to program the array are applied again. The second set of programming pulses provides a 100% programming margin to the Flash memory location and ensures the integrity of the programmed data. The simplified flowchart shown in Figure 2 describes these steps. Detailed descriptions and flowcharts, including timing requirements, describing the Flash programming procedure can be found in the *MC68HC912B32 Technical Summary* (document number MC68HC912B32TS/D).





3 General Flash Serial Bootloader Requirements

Two of the most important requirements for a program such as the Flash serial bootloader are that it have minimal impact on the final product's software performance and add little or nothing to the hard-ware costs. The Flash serial bootloader described in this application note meets both of these requirements.

Because the MC68HC912B32 includes an on-chip SCI, no additional external hardware is required to communicate with a host computer with the possible exception of an RS-232 level translator chip. In many systems, this may already be a part of the system design as the SCI is often used as a diagnostic port. If an RS-232 level translator is not included as part of the basic system design, a small adapter board could be constructed containing the level translator and RS-232 connector. This board could then be used by service personnel when updating the system firmware so that the cost of the level translator would not have to be added to each system. In addition to the SCI port, a single input pin is required to inform the serial bootloader startup code whether to execute the Flash serial bootloader code or jump to the system application program.

As mentioned previously, because the MC68HC912B32 interrupt and reset vectors reside in the 2k-byte bootblock, they cannot be changed without erasing the bootblock itself. Even though it is possible to erase and reprogram the bootblock from within the bootbloader program, it is inadvisable to do so. If anything were to go wrong during the process of reprogramming the bootblock, it would be impossible to recover from the situation without the use of special programming hardware. For this reason, the serial bootbloader includes a jump table that uses a secondary interrupt and reset vector table located just below the 2k bootblock. Each entry in the secondary interrupt table consists of a 2-byte address that mirrors the primary interrupt and reset vector table located in the erase-protected bootblock. Table 1 shows the correspondence between the primary and secondary interrupt vector tables.

Making use of the CPU12's indexed-indirect program counter relative addressing, each jump table entry consists of a single 4-byte JMP instruction. This form of the JMP instruction requires only six CPU clock cycles to execute, adding only 750 ns to the interrupt latency for a system operating at 8.0 MHz. In most applications this small amount of additional time will not affect the overall performance of the system.

Interrupt Vector Addre	ss Interrupt Source	Secondary Vector Address
\$FFC0 - \$FFCF	Reserved	\$F7C0 - \$F7CF
\$FFD0 - \$FFD1	BDLC (J1850)	\$F7D0
\$FFD2 - \$FFD3	ATD	\$F7D2
\$FFD4 - \$FFD5	Reserved	\$F7D4
\$FFD6 - \$FFD7	SCI 0	\$F7D6
\$FFD8 - \$FFD9	SPI	\$F7D8
\$FFDA - \$FFDB	Pulse Acc. Input Edge	\$ F 7 D A
\$FFDC - \$FFDD	Pulse Acc. Overflow	\$F7DC
\$FFDE - \$FFDF	Timer Overflow	\$F7DE
\$FFE0 - \$FFE1	Timer Channel 7	\$F7E0
\$FFE2 - \$FFE3	Timer Channel 6	\$F7E2
\$FFE4 - \$FFE5	Timer Channel 5	\$F7E4
\$FFE6 - \$FFE7	Timer Channel 4	\$F7E6
\$FFE8 - \$FFE9	Timer Channel 3	\$F7E8
\$FFEA - \$FFEB	Timer Channel 2	\$F7EA
\$FFEC - \$FFED	Timer Channel 1	\$F7EC
\$FFEE - \$FFEF	Timer Channel 0	\$F7EE
\$FFF0 - \$FFF1	Real Time Interrupt	\$F7F0
\$FFF2 - \$FFF3	IRQ	\$F7F2
\$FFF4 - \$FFF5	XIRQ	\$F7F4
\$FFF6 - \$FFF7	SWI	\$F7F6
\$FFF8 - \$FFF9	Illegal Opcode Trap	\$F7F8
\$FFFA - \$FFFB	COP Failure Reset	\$F7FA
\$FFFC - \$FFFD	Clock Mon. Fail Reset	\$F7FC
\$FFFE - \$FFFF	Reset	\$F7FE

Table 1 Primary/Secondary Interrupt Vector Addresses

4 Using The S-Record Bootloader

The S-Record bootloader utilizes the on-chip SCI for communications and does not require any special programming software for the host computer. The only host software required is a simple terminal program that is capable of communicating at 9600 baud and is able to wait for a prompt string before sending a line of text to the MC68HC912B32. The serial bootloader presents a simple command line interface to the user and accepts Motorola S-Record object files. The communications rate of 9600 baud was chosen simply because it is the most common baud rate available on a wide range of computing devices. However, the communication baud rate is the limiting factor in the length of time required to program the Flash. At 9600 baud, an S-record file containing 30k of object code requires approximately 90 seconds to be programmed into the Flash. If the communication rate were doubled to 19,200 baud or quadrupled it to 38,400 would cut the programming time by approximately one half or one quarter respectively.

Execution of the serial bootloader is selected by connecting port pin PDLC0 to a logic '0' level. Applying power to the target system or pressing the reset switch causes the bootloader to display the following prompt on the host terminal's screen:

```
(E)rase or (P)rogram:
```

Before selecting the Erase or Program function, V_{FP} must be applied to the V_{FP} pin of the MC68HC912B32.

4.1 Flash Erasure

Selecting the Erase function by typing an upper or lower case 'E' on the terminal will cause a bulk-erase of the Flash EEPROM array except for the 2k bootblock where the S-Record bootloader program resides. After the erase operation, a verify operation is performed to ensure that all locations are properly erased. If the erase operation was successful, the message 'Erased' is displayed on the screen and the bootloader's prompt is redisplayed.

If any locations were found to contain a value other than \$FF, the message 'Not Erased' is displayed on the terminal screen and the bootloader prompt is redisplayed. If the MC68HC912B32 device will not erase after one or two attempts, check the V_{FP} connection and measure the value of V_{FP} to ensure that it complies with the value published in the Technical Supplement *MC68HC912B32 Electrical Characteristics*. A V_{FP} voltage lower than that specified may cause the erase operation to fail. Applying a V_{FP} voltage higher than that specified may cause permanent damage to the device.

4.2 Flash Programming

The programming algorithm used for the on-chip FLASH memory is such that the time required to program each byte or word can vary from as little as 60 µs to as long as 3.5 ms. However the programming time for each byte or word will typically take no more than 120–180 µs. Because of this variability, the S-Record bootloader uses a software handshaking protocol to control the flow of S-Record data from the host computer. When the S-Record bootloader is ready to receive an S-Record, an ASCII asterisk character (*) is sent to the host computer. The host computer should respond by sending a single S-Record. The S-Record may include a carriage return and/or line feed character(s). Most commercial terminal programs capable of sending ASCII text files have the ability to wait for a specific character or string before sending a line of text.

Typing an upper or lower case 'P' on the terminal causes the bootloader to enter programming mode and wait for S-Records to be sent from the host computer. The host computer should begin by sending a single S-Record and then waiting for the bootloader to return an ASCII asterisk character (*) before sending subsequent S-Records.

The programming operation is terminated when the bootloader receives an 'S9' end-of-file record. If the S-Record object file being sent to the bootloader does not contain an 'S9' record, the bootloader will not return its prompt and will continue to wait for the end of file record. Pressing the target's reset switch, will cause the bootloader to return to its prompt.

If a Flash memory location will not program properly, the message 'Not Programmed' is displayed on the terminal screen and the bootloader's prompt is redisplayed. If problems are encountered when programming the Flash memory, check the V_{FP} connection to the target MCU and measure the value of V_{FP} to ensure that it complies with the value published in the MC68HC912B32 data sheet. A V_{FP} voltage lower than that specified may cause the programming operation to fail. Applying a V_{FP} voltage higher than that specified may cause permanent damage to the device.

If the V_{FP} connection is okay and V_{FP} is within the specified range, the problem may be caused by an S-Record containing data that is outside the range of the available on-chip Flash. The S-Record data must be within the range \$8000—\$F800.

Note: The S-Record bootloader should not be used with S-Records containing a code/data field longer than 64 bytes (S-Record length field greater than 67 (0 x 43) bytes). Sending an S-Record with a code/ data field longer than 64 bytes (S-Record length field greater than 67 (0 x 43) bytes) will cause the bootloader to crash and/or program incorrect data into the Flash.

5 Bootloader Software

The software implementing the serial Flash bootloader, shown in Listing 1, consists of five basic parts: Startup code, secondary interrupt vector jump table, bootloader control loop, programming code and erase code.

5.1 Startup Code

At power up or reset, CPU control is transferred to the routine beginning at the label BootStart. This routine checks the state of PORTDLC bit number 6. If PORTDLC bit number 6 is equal to a logic '0', the code between the labels BootLoad and BootLoadEnd are copied from Flash into the on-chip RAM and CPU control is passed to the bootloader code in RAM. If PORTDLC bit number 6 is equal to a logic '1', CPU control is transferred to the program defined by the address in the secondary reset vector.

5.2 Bootloader Control Loop

The bootloader control loop begins by initializing the SCI and timer system. The SCI is initialized to 9600 baud, 8 data bits, 1 start bit, 1 stop bit and no parity. The timer system is enabled with the fast flag clear option and configures channel 0 for use as an output compare. The output compare function is used to produce accurate timing delays for both the programming and erase routines. Enabling the fast flag clear option allows the timer interrupt flag bit for channel 0 to be cleared simply by writing a new value to the channel 0 timer register.

After initialization of the hardware, the bootloader displays its prompt and waits for the erase or program command to be entered. If a letter other than 'E' or 'P' is entered, the bootloader prompt is simply redisplayed on the next line. After returning from the execution of either the erase or program command, a message is displayed indicating either success or failure. Execution is then transferred back to the top of the control loop where the command prompt is redisplayed.

5.3 Erase Command Code

The code implementing the erase command consists of a single subroutine beginning at the label FErase in Listing 1. The subroutine implements the Flash erase algorithm as described in the *MC68HC912B32 Technical Summary* (document number MC68HC912B32TS/D). Basically the algorithm involves applying successive 100 ms erase pulses to the Flash array until the array is erased or a maximum of five erase pulses have been applied. Once the array is erased, the same number of erase pulses that were required to erase the Flash array is applied once again to provide a 100% erase margin. Figure 3 contains a detailed flow chart of the *FErase* subroutine.



Figure 3 FErase Subroutine Flowchart

5.4 Program Command Code

The software required to implement the program command is more complex that the Flash erase routine and requires three major subroutines and several simple supporting subroutines. The main subroutine implementing the program command begins at the label FProg in Listing 1. This small subroutine, shown in the flowchart in Figure 4, simply coordinates the reception of S-Records, the programming of the S-Record data into the Flash and sending the 'pace' character to the host computer requesting that the host send the next S-Record. In addition, it checks the type of each S-Record received from the host, ignoring 'S0' records and terminating the command when an 'S9' record is received. Each time a valid 'S1' record is received, the ProgFBlock subroutine is called to program the received data into Flash. If an error occurs during the reception of an S-Record or during the Flash programming process, the program command is terminated.





5.5 GetSRecord Subroutine

The GetSRecord subroutine is called by FProg to receive a single S-Record from the host computer. GetSRecord begins by allocating space on the stack for two local variables, SRecBytes and Check-Sum. The SRecBytes variable is used to hold the converted value of the S-Record length field. This value includes the number of bytes contained in the load address field, the length code/data field and the checksum field. The variable CheckSum is used to contain the calculated checksum value as the S-Record is received.

Next, the subroutine begins to receive characters from the host searching for the character pairs 'S0', 'S1' or 'S9' which indicate the start of a valid S-Record. Once a start of record is found, the S-Record length byte is received and saved in the local variable SRecBytes. Three is subtracted from the S-Record length byte and saved in the global variable DataBytes. This value represents the length of the code/data field and is used by the ProgFBlock subroutine when programming the S-Record data into the Flash. Finally, the load address, code/data field and the checksum field are received and placed in a global data buffer.

During the process of receiving the load address, code/data field and the checksum field each received byte is added to the CheckSum local variable. Because the received checksum is actually the ones compliment of what the calculated checksum should be, adding the two values should produce a result of \$FF. The increment of the variable CheckSum at the end of the receive loop should produce a result of zero if the checksum and all the S-Record fields were received properly. This will result in a 'equal' condition being returned if the S-Record was properly received and a 'not equal' condition being returned if there was a problem receiving the S-Record. Figure 5 contains the flowchart for the GetSRecord subroutine.





5.6 ProgFBlock Subroutine

The ProgFBlock subroutine programs the data received by the GetSRecord subroutine into the onchip Flash. The subroutine implements the Flash programming algorithm as described in the *MC68HC912B32 Technical Summary* (document number MC68HC912B32TS/D). Essentially the algorithm involves applying successive $20-25 \ \mu$ s programming pulses to a byte or aligned word until the memory location is properly programmed or a maximum of 50 programming pulses. Once the memory location is programmed, the same number of pulses that were required to program the location are applied again to provide a 100% programming margin.

To simplify the implementation of the programming algorithm and to keep the bootloader code as small as possible, the ProgFBlock routine only programs a single byte of the Flash at a time. This may seem to impose a severe time penalty when programming 30k of Flash. However, the actual time saved would be extremely small in relation to the amount of time required to send an S-Record file containing 30k of object code. Consider, for example, that most Flash locations are able to be programmed with the application of three programming and three margin pulses. Therefore using a total time of 33 μ s per byte, 22 μ s programming time and 11 μ s read/recovery time, would require 33 μ s x 6 * 30720 or approximately 6.1 seconds to program 30K bytes a byte at a time. If words were programmed instead, the time would be cut approximately in half.

As mentioned previously, the communication baud rate is the limiting factor in the length of time required to program the Flash. Consider an S-Record file containing 30k of object code. If each S–Record contained 32 bytes in the code/data field, each S-Record would be comprised of 74 ASCII characters and the file would contain 960 S-Records for a total file size of 71040 bytes not counting carriage return and/or line feeds. Just transmitting this much ASCII data at 9600 baud would require approximately 74 seconds. This is more than an order of magnitude greater than the three seconds that would be saved by programming a word at a time. Even at a baud rate of 38,400 it would require approximately 19 seconds to transmit 71040 bytes.

The ProgFBlock routine begins by allocating space on the stack for two variables, ProgPulses and PMarginFlag. During programming, the ProgPulses variable is used to maintain a count of the number of programming pulses applied to each programmed byte. When applying the margin pulses, this value is decremented until it reaches zero. The PMarginFlag variable is used as a boolean flag to indicate that the programming margin pulses are being applied. When set to non-zero, it modifies the program flow so that the contents of the Flash memory is not compared to the S-Record data after the application of each margin pulse.

Like the FErase subroutine, channel 0 of the on-chip timer is used to produce the timing delays required for the programming pulses and the read/recovery period. However, because of the need to produce short, accurate time delays, the timer is used in a slightly different manner. Before each program and read/recovery cycle begins, the timer subsystem is disabled by clearing the Timer ENable (TEN) bit in the Timer Status and Control Register (TSCR). When the timer is disabled, the contents of all timer registers, including the value of the Timer CouNTer Register (TCNT), are maintained. This allows the software to read the static value of the TCNT register, add to it a value that will produce a delay of 22 μ s and write the resulting value to the TC0 register without having to compensate for the intervening instruction execution time. The programming voltage is then applied to the array by setting the ENable Programming/Erase bit (ENPE) in the Flash EEPROM ConTroL register (FEECTL) and the timer system enabled. When the programming time period has expired, the programming voltage is removed from the Flash array. The timer is then setup to produce a delay of approximately 11 μ s for the read/ recovery period. Because this delay does not have to be as accurate as the programming pulse, the specification states a minimum of 10 μ s, the timer system is not disabled when setting up the output compare register.

At the end of each program and read/recovery cycle, the Flash data is compared to the received S–Record data. If the two do not match, the program and read/recovery cycle is repeated until the data matches or the maximum number of programming pulses have been applied. Next, an equal number of program and read/recovery cycles are once again applied to the Flash memory location to provide a 100% programming margin. Finally, the Flash data is once again compared to the received S–Record data. If the two do not match, the ProgFBlock routine terminates returning a 'not equal' condition indicating that the programming operation failed.

Figure 6 contains a detailed flow chart of the ProgFBlock subroutine.



Figure 6 ProgFBlock Subroutine Flowchart

5.7 Support Routines

Several additional support subroutines are required by the program and erase functions of the bootloader. The getchar and putchar subroutines provide SCI character I/O. The GetHexByte, CvtHex, and IsHex subroutines provide ASCII hexadecimal-to-binary conversion. The OutStr subroutine is used to send a null (0) terminated ASCII string to the on-chip SCI. It is called by the bootloader main loop to display the its prompt, error messages and command results. Because of the simplicity of these subroutines, no flowcharts are provided.

5.8 Secondary Reset/Interrupt Table

As noted previously, the bootloader supports a secondary reset/interrupt vector table that resides just below the 2k erase-protected bootblock. The jump table, located near the beginning of Listing 1, utilizes a form of indexed addressing that may not be supported by all assemblers. This addressing mode is a form of indexed indirect addressing that uses the program counter as an index register. The per mne-

monic used in place of an index register name stands for *Program Counter Relative* addressing. In reality, the CPU12 does not support an addressing mode known as Program Counter Relative or pcr. Instead, the CPU supports constant offsets from the value of the PC at the first byte of the next instruction. The PCR mnemonic is used to instruct the assembler to calculate an offset to the address specified by the expression preceding the ', pcr' index specification. The offset is calculated by subtracting the value of the PC at the address of the first object code byte of the next instruction from the value supplied in the index offset field. When the JMP instruction is executed, just the opposite occurs. The CPU12 adds the value of the PC at the first object code byte of the next instruction to the offset embedded in the instruction object code. The indirect addressing, indicated by the square brackets, specifies that the address calculated as the sum of the index register (in this case the PC) and the 16-bit offset contains a pointer to the destination of the JMP.

If an assembler does not support Program Counter Relative addressing the following substitution may be made. Replace the text between the square brackets of each JMP instruction with:

[(<InterruptVectorName> - \$800) - (* + 4),pc]

Where <InterruptVectorName> represents the name of the interrupt vector as shown in each JMP instruction, the (* + 4) represents the value of the program counter at the beginning of the next instruction and \$800 is the offset from real interrupt vector to the secondary interrupt vector. This entire expression allows the assembler to calculate the proper offset to the interrupt relative to the value of the program counter.

5.9 Stack Space Allocation

Several of the subroutines in Listing 1 allocate storage space on the stack for temporary variables. These variables are accessed using indexed addressing with the stack pointer as the index register. The offsets to these variables are calculated using the facilities of the assembler and may not be available in all assemblers. As an example, the assembler source sequence that appears just before the FProgBlock subroutine is shown below.

CurrentPC	set	*	; save the current value of the PC
	org	0	; set PC to zero so we can use assembler to
			; generate an offset into the stack.
;			
ProgPulses:	ds	1	; local variable to hold the number of
			; programming pulses.
PMarginFlag:	ds	1	; local variable to indicate we're applying the
			; margin pulses
;			
ora Cu	rrentD	C i	restore the original value of the PC

In this example the set assembler directive is used to assign a value to the label CurrentPC. In this case it is assigning or saving the current value of the Program Counter. In this regards the set directive is similar to the equ directive. However, the set directive may be used to reassign a new value to a label. So the label CurrentPC may be used to save the current value of the program counter each time labels are declared for accessing local storage. Next, the program counter is then set to zero with the use of the org directive. The ds directive, normally used to reserve global variable storage, is simply used to advance the program counter, assigning 'offset' values for the labels ProgPulses and PMar-ginFlag that may be used to access the actual variables on the stack. Finally, the assembler's program counter is restored to its previous value through the use of the org assembler directive.

6 Program Listings

6.1 Listing 1 — Serial Flash Bootloader

-- Micro Dialects, Inc. uASM-HC12 Assembler Tue, Mar 18, 1997 3:10 PM -- Page 1

2		;			
3	OOFE	, PORTDLC:	equ	\$00fe	; BDLC Port data register
4		;			
5	0016	COPCTL:	equ	\$0016	; COP timer control register.
6		;	-		
7	00C4	SR1:	equ	\$00c4	; SCIO status register #1.
8	00C7	DRL:	equ	\$00c7	; SCIO data register (low byte).
9	00C0	Baud:	equ	\$00c0	; SCIO baud rate register (16-bits).
10	00C3	CR2:	equ	\$00c3	; SCIO control register.
11		;			
12	0086	TSCR:	equ	\$0086	; timer status & control register.
13	0080	TIOS:	equ	\$0080	i
14	0084	TCNT:	equ	\$0084	; timer/counter register (16-bits).
15	008D	TMSK2:	equ	\$008d	; timer interrupt mask/prescaler control register.
16	008E	TFLG1:	equ	\$008e	; timer interrupt flag register.
17	0090	TC0:	equ	\$0090	; timer capture/compare register (16-bits).
18		;			
19	00F4	FEELCK:	equ	\$00£4	; Flash bootblock lock register.
20	00F5	FEEMCR:	equ	\$00±5	; Flash module configuration register.
21	00F.1	FEECIL:	equ	\$00£7	; Flash erase/programming control register.
22		;		*	
23	0080	TDRE	equ	\$80	; transmit data register empty bit.
24	0020	RDRF:	equ	\$20	; receive data register full bit.
25	0010	;		410	- pla bla practice de trame a la bla la popomi a data
26	0008	FEESWAL:	equ	ςυδ 2TΛ	, Disable FLASH array in WAIT mode bit in FEECTL register
27	0008	SVFP.	equ	\$U8 ,	Flash programming voltage present bit in FRECIL register
28	0004	ERAS ·	equ	\$U4 600	, Flash Brase blu in FEBCLE register
29	0002	LAI ·	equ	\$UZ	, Address/Data latch enable bit in FEECH register
0 C 1 C	0001	ENFE.	eyu	γuı	, riash programming vorcage enable bit in FEECIL register
31	0.080	/	0.001	690	. Timer enable bit
22	0080	1 EIN •	equ	200	, iimer enable bit.
33		'			
.****	*****	*****	******	****	*****
, 32					
36		Constants			
37		;			
38	1200	EClock:	eau	800000	; E-clock frequency in Hz
39	0034	Baud9600:	eau	8000000/16/9600	; value for baud register, based on clock frequency.
40	61A8 T	nS100: eon	i ECl	ock/320 ; ti	mer delay constant for 100 mS delay based on /32 prescaler.
41	00FA	mS1: eq	ru EC	clock/32000 ; t	timer delay constant for 1 mS delay based on /32 prescaler.
42	00B0 us	322: equ	((EC]	ock/10000)*22)/100 :	timer delay constant for 22 yS delay based on /1 program
			((= = =	0011/10000/22//100/	ciller detay constant for 22 us detay based on /1 prescarer.
43	0058 us	311: eau	((EC1	ock/10000)*11)/100;	timer delay constant for 11 uS delay based on /1 prescaler.
43 44	0058 us	311: equ	((EC1	ock/10000)*11)/100;	timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler.
43 44 45	0058 us 8000	S11: equ ; FlashStart:	((ECl	ock/10000)*11)/100; \$8000	<pre>timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip).</pre>
43 44 45 46	0058 us 8000 8000	S11: equ ; FlashStart: FlashSize:	((ECl equ equ	\$8000 32768	<pre>timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32.</pre>
43 44 45 46 47	0058 us 8000 8000 0800	311: equ ; FlashStart: FlashSize: BootBlkSize:	((ECl equ equ equ	<pre>sck/10000) *11)/100 ; \$8000 32768 2048</pre>	<pre>timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size.</pre>
43 44 45 46 47 48	0058 us 8000 8000 0800 0032	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses:</pre>	((ECl equ equ equ equ	sck/10000)*11)/100 ; \$8000 32768 2048 50	<pre>inter delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses.</pre>
43 44 45 46 47 48 49	0058 uS 8000 8000 0800 0032 0005	<pre>311: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses:</pre>	((ECl equ equ equ equ equ equ	<pre>sck/10000/11/100 ; s8000 32768 2048 50 5</pre>	<pre>inter delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses.</pre>
43 44 45 46 47 48 49 50	0058 uS 8000 8000 0800 0032 0005	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ;</pre>	((ECl equ equ equ equ equ	ock/10000)*11)/100 ; \$8000 32768 2048 50 5	<pre>inter delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses.</pre>
43 44 45 46 47 48 49 50 51	0058 us 8000 8000 0800 0032 0005 0800	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart:</pre>	((ECl equ equ equ equ equ equ	<pre>sck/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800</pre>	<pre>timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM.</pre>
43 44 45 46 47 48 49 50 51 52	0058 us 8000 8000 0032 0005 0800 0400	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxFrasePulses: ; RAMStart: RAMStart: RAMSize:</pre>	((ECl equ equ equ equ equ equ equ equ	\$8000 \$2068/10000)*11)/100 ; \$8000 2048 50 5 \$800 \$400	<pre>timer delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM.</pre>
43 44 45 46 47 48 49 50 51 52 53	0058 uS 8000 8000 0032 0005 0800 0400 0000	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop:</pre>	((ECl equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer.</pre>
43 44 45 46 47 48 49 50 51 52 53 54	0058 uS 8000 8000 0032 0005 0800 0400 0C00	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ;</pre>	((ECl equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55	0058 us 8000 8000 0032 0005 0800 0400 0000 0030	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; MaxPragPulses: ; RAMSize: RAMSize: StackTop: ; SORecType:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cck/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0'</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56	0058 us 8000 8000 0032 0005 08800 0400 0000 0030 0031	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxPrasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SlRecType:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cok/10000)*11)/100 ; s8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1'</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57	0058 uS 8000 8000 0032 0005 0800 0400 0C00 0031 0031 0039	<pre>S11: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SORecType: SORecType:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9'</pre>	<pre>chief delay constant for 22 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash size for 912B32. ; Flash size for 912B32. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58	0058 us 8000 8000 0032 0005 0800 0400 0000 0030 0030 0031 0039	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SJRecType: SSRecType:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cck/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9'</pre>	<pre>chief delay constant for 22 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 466 47 48 49 50 51 52 53 54 55 56 57 58 59	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039	<pre>sll: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxFrasePulses: ; RAMStart: RAMStart: RAMSize: StackTop: ; SORecType: SJRecType: ; ; ;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cok/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9'</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 466 47 48 49 50 51 52 53 54 55 56 57 58 59 60	0058 uS 8000 8000 0032 0005 0800 0400 0400 0200 0030 0031 0039	<pre>Sil: equ ; FlashStart: FlashSize: BootBlKSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SJRecType: ;; ;</pre>	((ECl equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize .0. '1' '9'</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 47 48 49 50 51 52 53 54 55 56 57 58 59 60 60	0058 uS 8000 8000 0032 0005 0800 0400 0C00 0031 0031 0039	<pre>311: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SORecType: ; ; ; </pre>	((ECl equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9'</pre>	<pre>chief delay constant for 22 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;****	0058 uS 8000 8000 0032 0005 0800 0400 0C00 0030 0031 0039	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxFragPulses: ; RAMSize: StackTop: ; SORecType: SIRecType: ; ; </pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9' </pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;****	0058 uS 8000 8000 0032 0005 0880 0400 0000 0030 0031 0039	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProgPulses: ; RAMStart: RAMStart: RAMStart: StackTop: ; SORecType: SSRecType: ; ; ; </pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cok/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 455 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 62 63	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0031	<pre>il: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SORecType: ;; ; ; ; ; ;;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMSize: StackTop: ; SORecType: SIRecType: ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Sck/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9' \$fc00</pre>	<pre>chiler delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProsePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SPRecType: ; ; uASM-HC12 Assembl</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; MASCII '9' used as</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;****2 63 64 	0058 uS 8000 8000 0032 0005 0800 0400 0400 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlKSize: MaxProgPulses: MaxErasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SPRecType: ; ; uASM-HC12 Assembl</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>sklood, 12000, 121)/100 ; sklood 200k/10000)*11)/100 ; sklood 2018 50 5 \$kloo \$400 RAMStart+RAMSize 10 11 99 ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 57 58 59 60 61 ;**** 63 64 	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: MaxProgPulses: ; MaxProgPulses: ; RAMSize: StackTop: ; SORecType: SORecType: SSRecType: ; ; uASM-HC12 Assembl ;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Sock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 22 us delay based on /1 prescaler. timer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashStze: BootBlkSize: MaxProgPulses: MaxFrasePulses: ; RAMStart: RAMStart: RAMStart: StackTop: ; SORecType: SJRecType: ; ; uASM-HC12 Assembl ; .</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 455 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 65 65 64	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxPrasePulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SIRecType: ; ; uASM-HC12 Assembl ; ; BootStart:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$400 RAMStart+RAMSize '0' '1' '9' \$fc00 , Mar 18, 1997 3:10</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 455 46 47 48 49 50 51 52 53 54 57 58 59 60 61 ;**** 63 64 65 66 67 67	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HC12 Assembl ; ; BootStart:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>sock/10000)*11)/100 ; s8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 455 466 47 488 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 65 666 67 68 67 68	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProgPulses: ; RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HC12 Assembl ; ; BootStart: ;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; MACIII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; MACIII '9' used as S9 record type indicator. ************************************</pre>
43 44 455 466 47 488 49 50 51 52 53 54 55 566 57 588 59 60 61 ;**** 62 63 64 67 65 66 67 68 69 97	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>sll: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. i mer delay constant for 11 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type ind</pre>
43 44 45 46 47 48 49 50 51 52 53 54 57 58 59 60 61 ;**** 63 64 65 66 67 68 69 70	0058 us 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc.	<pre>ill: equ ; FlashStart: FlashStart: FlashSize: MaxProgPulses: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$4000 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 65 66 67 68 69 70 70 71 72	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 Micro Dialects, Inc. FC00 FC03 4FFE4004 FC07 05FBFEF3 FC0B 790016	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HCl2 Assembl ; ; BootStart: jmg ; RootCopy:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. if lash EEPROM start address (Single chip). if lash size for 912B32. Erase protected bootblock size. maximum number of programming pulses. i maximum number of erase pulses. i start address of on-chip RAM. i size of on-chip RAM. i address to initialize the stack pointer. i ASCII '0' used as S0 record type indicator. i ASCII '1' used as S1 record type indicator. i ASCII '1' used as S1 record type indicator. i ASCII '9' used as S9 record type indicator. MACCII '9' used '9' used as S9 record type indicator. MACCII '9' used '9' us</pre>
43 44 45 46 47 48 49 50 51 52 53 55 56 57 58 59 60 61 ;**** 62 63 64 65 66 67 68 69 70 71 72 72	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC0B 790016 FC0E CFC7C	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SJRecType: SJRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>\$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$ \$</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. i feash estart address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; initialize the stack pointer ; 'reset' vector. ; disable watchdog ; point to the start of the Flash bootloader in Flash</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 63 64 65 66 67 68 69 70 71 72 73 74	0058 us 8000 8000 0032 0005 0800 0400 0C00 0030 0031 0039 FC00 CF0C00 FC00 Dialects, Inc. FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC0B 790016 FC0E CEFC7C FC11 CD0800	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SIRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 63 64 65 66 67 68 69 70 71 72 73 74 75	0058 uS 8000 8000 0032 0005 0800 0400 0c00 0030 0031 0039 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC0E CEFC7C FC11 CD0800 FC14 CCFECF	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SORecType: SORecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '9' used as S9 record type indicator. ; initialize the stack pointer ; 'reset' vector. ; disable watchdog ; point to the start of the Flash bootloader in Flash. ; point to the start of the bootloader code.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76	0058 uS 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC0E CEFC7C FC11 CD0800 FC14 CCFECF FC17 85F7C	<pre>Sil: equ ; FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxFrasePulses: ; RAMStart: RAMStart: RAMStart: SORecType: SORecType: SORecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. if Flash EEPROM start address (Single chip). if lash size for 912B32. if Erase protected bootblock size. maximum number of programming pulses. if address of on-chip RAM. if address to initialize the stack pointer. if ASCII '0' used as S0 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '9' used as S9 record type indicator. Model '9' used '9' used as S9' type indicator. Model '9' used '9' used as S9' type indicator. Model '9' used '9' u</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 63 64 65 666 67 68 69 70 71 72 73 74 75 76 77	0058 us 8000 8000 0032 005 0800 0400 0000 0030 0031 0039 FC00 CF0C00 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC08 CFEC7C FC11 CD0800 FC14 CCFECF FC17 83FC7C FC14 R0A3070	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SIRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy: NoveMore:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>COPCTL #BootLoad #RAMStart #BootLoad #BootLoad #JotLoad #BootLoad #Start #BootLoad #BootLoad #Start #BootLoad #BootLoad #Start #BootLoad #Bot</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. timer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ; initialize the stack pointer ; 'reset' vector. ; disable watchdog ; point to the start of the Flash bootloader in Flash. ; calculate the size of the bootloader code. ; move a byte of the bootloader into RAM.</pre>
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 63 64 65 66 66 67 68 69 70 71 72 73 74 75 76 77 78	0058 us 8000 8000 0032 0005 0800 0400 0000 0030 0031 0039 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC05 CEFC7C FC11 CD0800 FC14 CCFECF FC17 83FC7C FC18 387C7C FC18 180A3070 FC1E 0434F9	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SORecType: SORecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy: NoveMore:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$800 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 uS delay based on /1 prescaler. if prescaler. if Plash EEPROM start address (Single chip). if Plash size for 912B32. Erase protected bootblock size. maximum number of programming pulses. if maximum number of erase pulses. if start address of on-chip RAM. if address to initialize the stack pointer. if ASCII '0' used as S0 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '9' used as S9 record type indicator. if y PortDLC bit #6 == 0? if unitialize the stack pointer if v PortDLC bit #6 == 0? if usable watchdog if point to the start of the Flash bootloader in Flash. if calculate the size of the bootloader code. if move a byte of the bootloader into RAM. if dec byte count, move till done. if onc hip RAM. if address and the size of the colloader code. if move a byte of the bootloader into RAM. if dec byte count, move till done. if address and the size of the colloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM. if address and the size of the bootloader into RAM.</pre>
43 44 455 466 47 488 49 50 51 52 53 54 55 56 57 58 59 60 61 ;**** 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 8 79 70 77 77 77 78 79	0058 uS 8000 8000 0032 0005 0800 0400 0030 0031 0039 FC00 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC0E CEFC7C FC11 CD0800 FC14 CCFECF FC17 83FC7C FC18 180A3070 FC12 0434F9 FC20 06800	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxFrasePulses: ; RAMStart: RAMStart: RAMStart: SORecType: SORecType: SORecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy: NoveMore:</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Cock/10000)*11)/100 ; \$8000 32768 2048 50 5 \$8000 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. if Flash EEPROM start address (Single chip). if Flash size for 912B32. if Erase protected bootblock size. maximum number of programming pulses. if maximum number of erase pulses. if start address of on-chip RAM. if size of on-chip RAM. if address to initialize the stack pointer. if ASCII '0' used as S0 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '1' used as S1 record type indicator. if ASCII '9' used as S9 record type indicator. if address the stack pointer if y if you the start of type indicator. if disable watchdog if point to the start of the Flash bootloader in Flash. if point to the start of on-chip RAM. if calculate the size of the bootloader code. if move a byte of the bootloader into RAM. if dec byte count, move till done. if execute the bootloader code. if execute the bootloader code.</pre>
43 44 455 466 477 48 49 50 51 52 53 54 55 56 57 58 59 60 57 58 64 65 666 67 68 69 70 71 72 73 74 75 76 77 78 79 80 80 80 80 80 80 80 80 80 80 80 80 80	0058 us 8000 8000 0032 0005 0800 0005 0800 0030 0031 0039 FC00 FC00 CF0C00 FC03 4FFE4004 FC07 05FBFBF3 FC08 790016 FC07 05FBFBF3 FC08 790016 FC07 CF0C00 FC14 CCFECF FC17 83FC7C FC14 180A3070 FC18 0434F9 FC21 060800	<pre>Sil: equ ; FlashStart: FlashStart: FlashSize: BootBlkSize: MaxProgPulses: MaxProgPulses: ; RAMStart: RAMSize: StackTop: ; SORecType: SIRecType: SSRecType: ; ; uASM-HC12 Assembl ; ; BootStart: jmg ; BootCopy: MoveMore: ;;</pre>	((ECl equ equ equ equ equ equ equ equ equ equ	<pre>Sock/10000)*11)/100 ; sk000 32768 2048 50 5 \$400 RAMStart+RAMSize '0' '1' '9' ******************************</pre>	<pre>chief delay constant for 12 us delay based on /1 prescaler. i mer delay constant for 11 us delay based on /1 prescaler. ; Flash EEPROM start address (Single chip). ; Flash size for 912B32. ; Erase protected bootblock size. ; maximum number of programming pulses. ; maximum number of erase pulses. ; start address of on-chip RAM. ; size of on-chip RAM. ; address to initialize the stack pointer. ; ASCII '0' used as S0 record type indicator. ; ASCII '1' used as S1 record type indicator. ; ASCII '1' used as S9 record type indicator. ; ASCII '9' used as S9 record type indicator. ************************************</pre>

;****					*******	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
82			;			
83			; This is	the jump	b table that is used t	o access the secondary interrupt vector table. Each one
84			; of the ad	ctual in	terrupt vectors, begin	ing at \$ffd0, points to an entry in this table. Each jmp
85			; instruct	lon uses	s indexed indirect pro	gram counter relative (pcr) addressing to access the
00			, secondar	ry interi	rupt vector table that	is localed just below the 2k bootblock.
89			'			
;****	****	*****	******	*******	*****	******
. 89			;			
90			;			
91	FC24	05FBFBA8	JBDLC:	jmp	[BDLC-\$800,pcr]	
92	FC28	05FBFBA6	JATD:	jmp	[ATD-\$800,pcr]	
93	FC2C	05FBFBA6	JSCI0:	jmp	[SCI0-\$800,pcr]	
94	FC30	05FBFBA4	JSPI:	jmp	[SPI-\$800,pcr]	
95	FC34	05FBFBA2	JPACCIE:	jmp	[PACCIE-\$800,pcr]	
96	FC38	05FBFBA0	JPACCOV:	jmp	[PACCOv-\$800,pcr]	
97	FC3C	05FBFB9E	JTimerOv:	jmp	[TimerOv-\$800,pcr]	
98	FC40	05FBFB9C	JTimerCh7:	jmp	[TimerCh7-\$800,pcr]	
100	FC44	05FBFB9A	JTimerCh6:	Jmp	[TimerCh6-\$800,pcr]	
100	FC48	05FBFB98	JTImerCh5:	Jmp	[TimerCh5-\$800,pcr]	
101	FC4C	05FBFB96	JIIMerCH4.	Juip	[TimerCh4-\$800,per]	
102	FC50	05000000	JTimerCh2:	jilip imp	[TimerCh2_\$800,pcr]	
104	FC54	05FBFB90	JTimerCh1:	jiiip imp	[TimerCh1-\$800 pcr]	
105	FC5C	05FBFB8E	JTimerCh0:	imp	[TimerCh0-\$800.pcr]	
106	FC60	05FBFB8C	JRTI:	imp	[RTI-\$800.pcr]	
107	FC64	05FBFB8A	JIRO:	imp	[IRO-\$800,pcr]	
108	FC68	05FBFB88	JXIRQ	jmp	[XIRQ-\$800,pcr]	
109	FC6C	05FBFB86	JSWI:	jmp	[SWI-\$800,pcr]	
110	FC70	05FBFB84	JIllop:	jmp	[Illop-\$800,pcr]	
111	FC74	05FBFB82	JCOPFail:	jmp	[COPFail-\$800,pcr]	
112	FC78	05FBFB80	JClockFail:	jmp	[ClockFail-\$800,pcr]	
113			;			
114			;			
115						
;****	****	* * * * * * * * * * * * * * * * * * * *	*******	*******	*****	***************************************
116			;			
110			; The code	residir	ig between the labels	BootLoad and BootLoadEnd comprises the Dootloader code
110			, that is	copied i	nco RAM. The bootload	er must execute from the on-chip RAM because the Flash
120		;	written i	n a posi	tion independent manne	erng programmed of erased. The bootroader code was
121		,	;	n a post	cion independente manne	i bo chae ie will excedee property when copied into RAM.
122						
;****	****	*****	* * * * * * * * * * * * * * * *	*******	******	******
123			;			
124			;			
125	FC7C		BootLoad:	equ	*	
126	E070					
100	FC/C	CC0034		ldd	#Baud9600 ;	set SCI to 9600 baud @ 8.0 MHz
127	FC7C	5CC0		ldd std	#Baud9600 ; Baud	set SCI to 9600 baud @ 8.0 MHz
127 128	FC7F FC81	CC0034 5CC0 C60C		ldd std ldab	#Baud9600 ; Baud #\$0c ;	set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver.
127 128 	FC7F FC81 Micro	CC0034 5CC0 C60C Dialects, Inc. uA	SM-HC12 Assemble	ldd std ldab er Tue,	#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM	set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3
127 128 	FC7F FC81 Micro	CC0034 5CC0 C60C Dialects, Inc. uA	SM-HC12 Assemble	ldd std ldab er Tue,	#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM	set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3
127 128	FC7F FC81 Micro	5CC0 66CC D Dialects, Inc. uA	SM-HCl2 Assemble	ldd std ldab er Tue,	#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM	set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3
127 128 129	FC7F FC81 Micro FC83 FC83	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601	SM-HC12 Assemble	ldd std ldab er Tue, stab ldab	#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ;	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver Page 3 disable the erasure or programming of the 2k bootblock</pre>
127 128 129 130 131	FC7F FC81 Micro FC83 FC85 FC85	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4	.SM-HCl2 Assemble	ldd std ldab er Tue, stab ldab stab	<pre>#Baud9600 ; Baud #\$0c ; . Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver Page 3 disable the erasure or programming of the 2k bootblock.</pre>
127 128 129 130 131 132	FC83 FC85 FC87 FC87 FC89	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0	.SM-HCl2 Assemble	ldd std ldab er Tue, stab ldab stab ldab	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears.</pre>
129 127 128 129 130 131 132 133	FC7F FC81 Micro FC83 FC85 FC85 FC87 FC89 FC88	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86	.SM-HC12 Assemble	ldd std ldab er Tue, stab ldab stab ldab stab	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 CR2 #\$01 ; FEELCK #\$b0 ; TSCR</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears.</pre>
127 128 129 130 131 132 133 134	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC80	CC0034 5CC0 C60C 5 Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601	.SM-HCl2 Assemble	ldd std ldab er Tue, stab ldab stab ldab stab ldab	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare.</pre>
129 127 128 130 131 132 133 134 135	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC80 FC8F	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80	.SM-HC12 Assemble	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab	#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; TSCR #\$01 ; TIOS	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare.</pre>
129 129 130 131 132 133 134 135 136	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC80 FC87 FC87 FC87 FC87 FC87	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C	.SM-HCl2 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab ldab	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; </pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt.</pre>
129 129 130 131 132 133 134 135 136	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC80 FC88 FC80 FC87 FC91 FC95	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C	.SM-HCl2 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab leax jsr	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; </pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it.</pre>
129 129 130 131 132 133 134 135 136 137	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC80 FC87 FC91 FC95 FC99	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C 15FA01F6	.SM-HCl2 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user.</pre>
129 129 130 131 132 133 134 135 136 137 138 139	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC88 FC81 FC95 FC99 FC90	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C 15FA01F6 15FA01F9	.SM-HC12 Assemble BLLoop:	ldd std ldab rr Tue, stab ldab stab ldab stab ldab stab ldab stab jear jsr jsr	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR # #\$01 ; TSCR # #\$01 ; TSCR 5 BLPrompt,pcr ; OutStr,pcr ; putchar,pcr ; putchar,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it.</pre>
129 129 130 131 132 133 134 135 136 137 138 139 140	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC87 FC89 FC87 FC91 FC95 FC99 FC90 FC90	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C600 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F9 37 JAED0058	.SM-HC12 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr pshb	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ; putchar,pcr ; ; OutStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. save it.</pre>
127 128 129 130 131 132 134 135 136 137 138 139 140 141	FC7C FC781 Micro FC83 FC85 FC85 FC87 FC88 FC88 FC88 FC88 FC91 FC91 FC99 FC90 FC90 FC24	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C 15FA01F6 15FA01F9 37 1AFA0058 15FA021B	.SM-HCl2 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab	<pre>#Baud9600 ; Baud #\$0c ; .Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; putchar,pcr ; ; CrLfStr,pcr ; OutStr,pcr ; 0utStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line.</pre>
127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142	FC7C FC7F1 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC89 FC99 FC90 FC90 FC91 FC92 FC92 FC41	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA02CC 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33	.SM-HCl2 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr pshb leax jsr pulb	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ; putchar,pcr ; CrLfStr,pcr ; OutStr,pcr ; OutStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character.</pre>
127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143	FC7F FC7F FC81 Micro FC85 FC87 FC89 FC88 FC88 FC88 FC89 FC99 FC99 FC99	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F6 15FA01F6 15FA01F8 37 1AFA0058 15FA021B 33 C40F	.SM-HC12 Assemble BLLoop:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr pshb leax jsr pulb	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; putchar,pcr ; putchar,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; outStr,pcr ; Suttr,pcr ; CrLfStr,pcr ; OutStr,pcr ; Suttr,pcr ; Suttr,</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters)</pre>
127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC89 FC91 FC91 FC92 FC90 FC92 FC92 FC92 FC46 FC46 FC46 FC46 FC46 FC46 FC46 FC46	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF	.SM-HCl2 Assemble BLLoop: andb	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr pshb leax jsr pulb , #\$df	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ; putchar,pcr ; CrLfStr,pcr ; OutStr,pcr ; OutStr,pcr ; Simple</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters).</pre>
1277128 1277128 1277128 130 131 1323134 1351313 136 137 138 136 137 138 139 140 141 142 143 144 144	FC7F FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC88 FC89 FC90 FC90 FC90 FC90 FC90 FC91 FC42 FC42 FC42 FC44 FC44 FC44 FC44 FC44	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C600 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145	.SM-HCl2 Assemble BLLoop: andb CheckFErase:	ldd std ldab er Tue, stab ldab stab leax jsr jsr jsr jsr jsr jsr jsr jbub stab leax jsr jsr jbub stab	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; Strifstr,pcr ; OutStr,pcr ; Simple #'E' ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered?</pre>
127 127 128 130 131 132 133 134 135 137 138 139 140 141 142 143 144 145 147	FC7E FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC89 FC88 FC91 FC91 FC91 FC91 FC92 FC99 FC90 FC41 FC42 FC46 FC48 FC48 FC48	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F6 15FA01F6 15FA021B 33 C4DF C145 261A	.SM-HCl2 Assemble BLLoop: andb CheckFErase:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab stab ldab stab blab stab stab blab stab blab stab stab stab stab stab stab stab st	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command.</pre>
1271128 127128 127128 1301131 132131 132131 1332 1331 1331 1332 1331 1344 1355 1366 1371 1342 1344 1432 1442 1442 1442 1442 144	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC87 FC95 FC91 FC95 FC91 FC95 FC91 FC92 FC92 FC92 FC92 FC92 FC92 FC92 FC92	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 55B80 1AFA006C 15FA01F6 15FA01F6 15FA01F6 15FA01F6 13FA0058 15FA021B 33 C4DF C145 261A 15FA003A	.SM-HC12 Assemble BLLoop: andb CheckFErase:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab leax jsr jsr jsr pshb leax jsr pulb (mpb bne jsr	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ; getchar,pcr ; crLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CheckVfp,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present.</pre>
1271 1272 128 127 128 127 128 127 128 130 132 133 134 133 134 133 134 133 134 135 136 137 138 136 147 144 144 144 144 146 146 146 146 146 146	FC7F FC81 Micro FC83 FC85 FC87 FC89 FC88 FC87 FC99 FC99 FC99 FC99 FC99 FC99 FC42 FC46 FC4A FC4A FC4A FC4A FC4A FC4A	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA003A 26DA	.SM-HCl2 Assemble BLLoop: andb CheckFErase:	<pre>ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr pshb leax jsr jsr pulb 0 #\$df cmpb bne</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; getchar,pcr ; getchar,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; BLPrompt ; BLPOOP ; CheckVfp,pcr ; BLLOOP ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present.</pre>
1271 1272 128 127 128 127 128 127 128 127 128 127 128 127 128 137 138 133 133 133 133 133 133 133 133 133	FC7F FC81 Micrr FC85 FC87 FC87 FC87 FC89 FC99 FC99 FC99 FC99 FC99 FC97 FC97 FC9	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C600 5B86 C601 5B80 1AFA006C 15FA01F6 15FA01F6 15FA022C 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA03A 26DA 15FA018	.SM-HCl2 Assemble BLLoop: andb CheckFErase:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr pshb leax jsr pshb leax jsr pshb bleax jsr pshb bleax jsr pshb	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; futchar,pcr ; CutStr,pcr ; OutStr,pcr ; GutStr,pcr ; CutStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; FErase,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash.</pre>
1271 1271 128 1271 128 128 130 131 132 133 132 133 132 133 137 137 138 139 140 141 142 143 144 144 145 144 147 148 149 155 151	FC7F FC81 Micr FC83 FC85 FC87 FC87 FC87 FC87 FC99 FC95 FC99 FC95 FC99 FC95 FC92 FC94 FC94 FC94 FC94 FC95 FC96 FC96 FC96 FC96 FC96 FC96 FC96 FC96	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5F800 1AFA006C 15FA02C 15FA01F6 15FA022C 15FA01F6 15FA021B 33 C4DF C145 261A 15FA003A 26DA 15FA018 1AFA005A	.SM-HC12 Assemble BLLoop: andb CheckFErase:	<pre>ldd std ldab er Tue, stab ldab stab leax jsr jsr jsr jsr jsr jsr jsr jer bhe leax jsr jsr jer jer leax jsr jer leax jsr jer leax jer jer leax jer leax jer leax jer leax jer leax jer leax jer jer jer leax jer jer jer jer jer jer jer jer jer jer</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CLfStr,pcr ; OutStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ClfStr,pcr ; ElLoop ; FErase,pcr ; ENot,pcr ; ENot,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string.</pre>
1271 1271 128 129 130 131 132 133 134 135 138 138 139 136 137 138 138 139 136 141 142 144 144 144 144 144 144 144 145 155 152	FC7F FC81 Micr. FC83 FC85 FC87 FC89 FC87 FC89 FC87 FC91 FC95 FC91 FC95 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 SC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C6B0 SB86 C601 SB80 LAFA006C 1SFA012C 1SFA01F6 1SFA01F9 37 LAFA0058 1SFA021B 33 C4DF C145 261A 1SFA018 LAFA005A 2604 	.SM-HC12 Assemble BLLoop: andb CheckFErase:	ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr bleax jsr cmpb bne jsr bne jsr leax bne	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$b0 ; TSCR #\$01 ; TSCR #\$01 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futchar,pcr ; chtfStr,pcr ; 0utStr,pcr ; ChtfStr,pcr ; ChtfStr,pcr ; ChteckVfp,pcr ; BLLoop ; FETASe,pcr ; BNLoop ; FENAS,pcr ; BALENASE ; BALENA</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly.</pre>
1277 128 127 128 127 128 127 128 127 128 127 128 139 133 133 133 133 133 133 133 133 133	FC7F FC81 Micrr FC85 FC85 FC87 FC89 FC88 FC95 FC99 FC91 FC95 FC99 FC91 FC47 FC48 FC47 FC48 FC47 FC48 FC47 FC47 FC47 FC47 FC47 FC47 FC47 FC47	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA022C 15FA01F6 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA03A 26DA 15FA018 1AFA005A 2604 1AFA005A	.SM-HCl2 Assemble BLLoop: andb CheckFErase:	ldd stab ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr pshb leax jsr jsr jsr jsr bue jsr jsr jsr bne jsr jer bue bne jsr leax jsr jer	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; crLfStr,pcr ; CutStr,pcr ; cutStr,pcr ; GutStr,pcr ; CutStr,pcr ; CutStr,pcr ; CutStr,pcr ; BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futchar,pcr ; CutStr,pcr ; CutStr,pcr ; BLPrompt,pcr ; getchar,pcr ; CutStr,pcr ; CutStr,pcr ; Simple #'E' ; ChekVfp,pcr ; FErase,pcr ; ENot,pcr ; Fassed,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 131\\ 131\\ 132\\ 133\\ 134\\ 135\\ 136\\ 137\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138$	FC7F FC81 Micrr FC85 FC87 FC87 FC87 FC99 FC99 FC99 FC95 FC97 FC95 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C600 5B86 C601 5B80 1AFA006C 15FA012C 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA03A 26DA 15FA018 1AFA005A 2604 1AFA0058 15FA01FC 0000	.SM-HCl2 Assemble BLLoop: andb CheckFErase: BadErase:	<pre>ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr pulb cmpb bne jsr bne jsr bne jsr leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futStr,pcr ; CutStr,pcr ; GutStr,pcr ; GutStr,pcr ; ChekVfp,pcr ; BLLoop ; FErase,pcr ; BadErase ; Erased,pcr ; GutStr,pcr ; BadErase ; Erased,pcr ; CutStr,pcr ; BadErase ; Erased,pcr ; GutStr,pcr ; BadErase ; Erased,pcr ; CutStr,pcr ; BadErase ; Erased,pcr ; CutStr,pcr ; BadErase ; Erased,pcr ; CutStr,pcr ; C</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string </pre>
127 127 128 130 131 132 133 134 133 134 133 134 133 134 133 134 133 134 133 134 133 134 135 136 137 137 140 142 143 144 145 144 145 155 155 155 155 155	FC7E FC81 Micro FC83 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC91 FC95 FC91 FC95 FC91 FC95 FC92 FC42 FC44 FC44 FC45 FC45 FC55 FC57 FC85 FC65 FC65 FC65 FC65 FC65 FC65 FC65 FC6	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 15FA026 15FA02C 15FA01F6 15FA022C 15FA01F6 15FA021B 33 C40F C145 261A 15FA003A 26DA 15FA018 1AFA005A 2604 1AFA0058 15FA01FC 20C6	.SM-HC12 Assemble BLLoop: andb CheckFErase: BadErase:	<pre>ldd std ldab ar Tue, stab ldab stab leax jsr jsr jsr jsr bnb leax jsr jsr jsr bnb leax jsr jsr bnb leax jsr jsr bnb leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; Simple #'E' ; ChckVfp,pcr ; ELLoop ; FErase,pcr ; Erased,pcr ; BLLoop ; </pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for Vfp present. yes. check for Vfp present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again.</pre>
127 127 128 129 130 131 132 133 133 133 133 133 133 133 133	FC7F FC81 FC85 FC85 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC95 FC91 FC95 FC91 FC92 FC92 FC92 FC94 FC95 FC94 FC95 FC94 FC95 FC94 FC95 FC96 FC96 FC96 FC96 FC96 FC96 FC96 FC96	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5FA022C 15FA016C 15FA02CC 15FA01F9 37 1AFA005C 15FA01F9 33 C4DF C145 261A 15FA021B 33 C4DF C145 261A 15FA003A 26DA 15FA018 1AFA005A 2604 1AFA005A 2604 1AFA005A 2604 C150 C150 C150 C150 C150 C150 C150 C150	.SM-HCl2 Assemble BLLoop: andb CheckFErase: BadErase: ;	<pre>ldd std ldab rr Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr pshb leax jsr pulb o #\$df cmpb bne jsr leax jsr leax jsr bne leax jsr stab bne leax jsr </pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; FETASe,pcr ; ENOt,pcr ; BadBrase ; Enased,pcr ; OutStr,pcr ; BLLoop ; HID; HID; HID; SLLOOP ; HID; CheckVfp,pcr ; BadBrase ; BadBrase ; BaLCOP ; HID; HID; HID; HID; HID; HID; HID; HID</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. present accommand entered?</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 131\\ 131\\ 132\\ 133\\ 134\\ 135\\ 138\\ 133\\ 134\\ 135\\ 138\\ 133\\ 134\\ 135\\ 138\\ 133\\ 134\\ 142\\ 143\\ 144\\ 144\\ 145\\ 152\\ 153\\ 154\\ 155\\ 155\\ 155\\ 155\\ 155\\ 155\\ 155$	FC7F FC81 Micrr FC85 FC85 FC87 FC89 FC88 FC87 FC99 FC99 FC99 FC99 FC99 FC99 FC99 FC9	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5BF4 C6B0 1AFA006C 15FA012C 15FA01F6 15FA01F6 15FA021B 33 C4DF C145 261A 15FA03A 26DA 15FA018 1AFA005A 2604 1AFA005B 15FA01FC 20C6 C150 26C2	.SM-HC12 Assemble BLLoop: andb CheckFErase: BadErase: ; ChkProg:	ldd stab ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr pshb leax jsr jsr jsr leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futfstr,pcr ; CutStr,pcr ; GutStr,pcr ; GutStr,pcr ; butchar,pcr ; butchar,pcr ; BLForg ; CheckVfp,pcr ; BLLoop ; Erased,pcr ; Butcop ; UtStr,pcr ; Butcop ; H'P' ; BULCOP ; BULCOP ; H'P' ; BULCOP ; BULCOP</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered?</pre>
127 127 128 127 128 128 128 128 138 139 132 133 134 133 134 133 134 133 134 133 134 135 136 137 138 139 142 143 144 145 144 145 155 155 155 155 155 155	FC7F FC81 Micrr FC85 FC87 FC87 FC87 FC99 FC99 FC99 FC95 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA012C 15FA01F6 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA03A 26DA 15FA0118 1AFA005A 2604 1AFA0058 15FA01FC 20C6 C150 26C2 15FA01C	.SM-HC12 Assemble BLLoop: CheckFErase: BadErase: ; ChkProg:	<pre>ldd std ldab er Tue, stab ldab stab leax jsr jsr jsr bne jsr leax jsr bne jsr leax jsr bne jsr bne jsr bne jsr bne jsr bne jsr bne jsr jsr bne jsr jsr jsr bne jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; FEELCK #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; ChifStr,pcr ; ChifStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; Erased,pcr ; BLLoop ; #'P' ; BLLoop ; #'P' ; BLLoop ; #'P' ; BLLoop ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. ves check for Vfp present</pre>
$\begin{array}{c} 127\\ 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 133\\ 134\\ 133\\ 134\\ 133\\ 139\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138$	FC7F FC81 Micr. FC83 FC85 FC87 FC89 FC88 FC87 FC95 FC91 FC95 FC91 FC95 FC91 FC95 FC97 FC92 FC94 FC94 FC40 FC40 FC40 FC40 FC40 FC40 FC40 FC4	CC0034 SCC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C600 SB86 C601 SB80 LAFA006C 1SFA012C 1SFA01F6 ISFA01F6 ISFA01F9 37 LAFA0058 1SFA021B 33 C4DF C145 261A 1SFA018 LAFA005A 26DA 1SFA018 LAFA005A 26DA SFA01FC 20C6 C150 26C2 SFA01C 26BC	.SM-HC12 Assemble BLLoop: and CheckFErase: BadErase: ; ChkProg:	<pre>ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr bleax jsr pulb bne jsr bne leax bne leax bne jsr bne jsr bne bne jsr bne bne jsr bne bne jsr bne bne bne bne bne</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; FEELCK #\$01 ; TSCR #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; FErase,pcr ; Ends,pcr ; BLLoop ; CheckVfp,pcr ; CheckVfp,pcr ; BLLoop ; CheckVfp,pcr ; BLLoop ; CheckVfp,pcr ; CheckVfp</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present</pre>
127 127 128 129 130 131 132 133 133 133 133 133 133 133 133	FC7F FC81 FC85 FC85 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC88	CC0034 5CC0 C60C Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA012C 15FA01F6 15FA01F9 37 1AFA0058 15FA021B 33 C4DF C145 261A 15FA018 1AFA005A 26DA 15FA018 1AFA005A 2604 1AFA005A 2604 2604 15FA01FC 20C6 C150 26C2 15FA001C 26BC 15FA006C	.SM-HCl2 Assemble BLLoop: andb CheckFErase: BadErase: ; ChkProg:	<pre>ldd ldab std ldab er Tue, stab ldab stab ldab stab ldab stab leax jsr jsr jsr pshb leax jsr jsr bne jsr leax jsr bne leax jsr bra cmpb bne jsr bra cmpb bne jsr bra jsr bra jsr bra bne jsr bra bne jsr bra bne jsr bne js bne</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futfstr,pcr ; CrLfStr,pcr ; CutStr,pcr ; CutStr,pcr ; BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; fetchar,pcr ; CutStr,pcr ; CutStr,pcr ; BLLoop ; FErase,pcr ; ENC,pcr ; BLLoop ; BLLoop ; BLLoop ; BLLoop ; BLLoop ; BLLoop ; BLLoop ; CutStr,pcr ; BLLoop ; BLLoop ; Ferase,pcr ; BLLoop ; FProg,pcr ; BLLoop ; FProg,pcr ; BLLoop ; CutStr,pcr ; BLLoop ; FProg,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. check for Vfp present. go print prompt if not present. yes. check for Vfp present. go print prompt if not present. yes. check for Vfp present. go print prompt if not present. yes. op orgent met Flash.</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 131\\ 131\\ 132\\ 133\\ 134\\ 135\\ 136\\ 137\\ 138\\ 139\\ 132\\ 134\\ 135\\ 136\\ 137\\ 138\\ 139\\ 139\\ 140\\ 141\\ 142\\ 143\\ 144\\ 147\\ 148\\ 147\\ 148\\ 147\\ 148\\ 147\\ 148\\ 147\\ 155\\ 155\\ 155\\ 155\\ 156\\ 155\\ 156\\ 155\\ 156\\ 155\\ 156\\ 156$	FC7F FC81 Micrr FC85 FC87 FC89 FC85 FC95 FC99 FC95 FC95 FC95 FC95 FC95 FC9	CC0034 SCC0 C60C D Dialects, Inc. uA SBC3 C601 SBF4 C680 SB86 C601 SB80 1AFA006C 1SFA012C 1SFA01F6 1SFA01F6 1SFA01F9 37 1AFA0058 1SFA021B 33 C4DF C145 261A 1SFA018 1AFA005A 26DA 1SFA018 1AFA005B 1SFA018 1AFA005B 1SFA01FC 20C6 C150 26C2 1SFA001C 26BC 1SFA006C 39	.SM-HC12 Assemble BLLoop: and CheckFErase: BadErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr pulb cmpb bne jsr leax jsr bne jsr leax jsr bne jsr cmpb bne jsr bra bne jsr jsr pshb</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$01 ; FEELCK #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; Erased,pcr ; BLLoop ; Erased,pcr ; BLLoop ; ElLoop ; ElLoop ; BLLoop ; BLLoop ; BLLoop ; ElLoop ; BLLoop ; FProg,pcr ; BLLoop ; FProg,pcr ; BLLoop ; FProg,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. deck for Vfp present. go print prompt if not present. yes. deck for Vfp present. go print prompt if not present. yes. deck for Vfp present. go print prompt if not present. yes. deck for Vfp present. go print prompt if not present. yes. deck for Vfp present. go print prompt if not present. yes. go program the Flash.</pre>
$\begin{array}{c} 127\\ 127\\ 128\\ 127\\ 128\\ 128\\ 130\\ 131\\ 132\\ 133\\ 134\\ 133\\ 134\\ 133\\ 134\\ 133\\ 134\\ 133\\ 138\\ 136\\ 137\\ 138\\ 142\\ 143\\ 144\\ 144\\ 144\\ 144\\ 144\\ 144\\ 144$	FC7F FC81 Micro FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC88	CC0034 5CC0 C60C D Dialects, Inc. uA 5BC3 C601 5BF4 C6B0 5B86 C601 5B80 1AFA006C 15FA02C 15FA01F6 15FA01F6 15FA01F6 15FA021B 33 C4DF C145 261A 15FA018 1AFA0058 15FA018 1AFA005A 26DA 15FA018 1AFA005A 26DA 15FA017C 20C6 C150 26C2 15FA001C 26BC 15FA001C 26BC 15FA001C 26BC 15FA0020	.SM-HC12 Assemble BLLoop: CheckFErase: BadErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab er Tue, stab ldab stab leax jsr jsr pub b ne jsr leax jsr leax jsr bne jsr leax jsr bne jsr leax jsr bne jsr leax jsr jsr leax jsr jsr jsr leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; FEELCK #\$01 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CLfStr,pcr ; OutStr,pcr ; ElLoop ; FErase,pcr ; ELLoop ; ELLoop ; FProg,pcr ; CheckVfp,pcr ; BLLoop ; FProg,pcr ; CLfStr,pcr ; CLfStr,pcr ; ELLoop ; FProg,pcr ; CLfStr,pcr ; CLeckVfp,pcr ; ELLoop ; FProg,pcr ; CLfStr,pcr ; CLeckVfp,pcr ; CLeckVfp,pcr ; ELLoop ; FProg,pcr ; CLIFStr,pcr ; CLIFStr,pcr ; CLIFStr,pcr ; CLIFStr,pcr ; CLIFStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line.</pre>
$\begin{array}{c} 127\\ 127\\ 128\\ 127\\ 128\\ 127\\ 128\\ 128\\ 131\\ 132\\ 133\\ 133\\ 133\\ 133\\ 133\\ 133$	FC7F FC81 FC85 FC85 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC88	CC0034 SCC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C6B0 SB86 C601 SB80 1AFA006C 1SFA012C 1SFA01F6 ISFA01F9 37 1AFA0058 1SFA01F9 33 C4DF C145 261A 1SFA018 1AFA005A 2604 1AFA005A 2604 1AFA005A 2604 1SFA01C2 20C6 C150 26C2 1SFA00CC 39 1AFA0020 1SFA02E3 SFA0020 SFA002E3 SFA002C3 S	SM-HC12 Assemble BLLoop: andb CheckFErase: BadErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab rr Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr pshb leax jsr jsr jsr leax jsr jsr jsr bne jsr leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futfstr,pcr ; CLfStr,pcr ; CLfStr,pcr ; ChekvOp,pcr ; BLLoop ; FETASe,pcr ; ENot,pcr ; BLLoop ; FETASe,pcr ; CutStr,pcr ; BadBrase ; Erased,pcr ; CutStr,pcr ; BLLoop ; FELLOOP ; FELLOOP ; FELLOOP ; FELLOOP ; FELLOOP ; CutStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line.</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 131\\ 132\\ 133\\ 134\\ 135\\ 138\\ 133\\ 134\\ 135\\ 138\\ 133\\ 134\\ 142\\ 143\\ 144\\ 142\\ 143\\ 144\\ 144\\ 142\\ 155\\ 156\\ 155\\ 156\\ 155\\ 156\\ 155\\ 156\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 163\\ 164\\ 165\\ 162\\ 162\\ 162\\ 163\\ 164\\ 165\\ 162\\ 162\\ 162\\ 162\\ 162\\ 162\\ 162\\ 162$	FC7F FC81 Micrr FC85 FC85 FC87 FC89 FC88 FC87 FC95 FC91 FC95 FC91 FC95 FC92 FC92 FC92 FC92 FC94 FC94 FC94 FC94 FC94 FC94 FC95 FC97 FC95 FC97 FC92 FC94 FC96 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 SCC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C600 SB86 C601 SB80 IAFA006C ISFA022C ISFA01F6 ISFA01F6 ISFA01F9 37 IAFA0058 ISFA021B 33 C4DF C145 261A ISFA021B 33 C4DF C145 261A ISFA003A 260A ISFA003A 260A ISFA018 IAFA0058 ISFA0058 ISFA0058 ISFA0058 ISFA017C 20C6 C150 26C2 ISFA001C 26C2 ISFA006C 39 IAFA0020 ISFA0123 38	.SM-HC12 Assemble BLLoop: andb CheckFErase: BadErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab std ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr pshb leax jsr bne jsr leax jsr bne leax jsr bne jsr bne</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$01 ; FEELCK #\$01 ; TIOS BLFrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; Simple #'E' ; CheckVfp,pcr ; BLLoop ; FETABE,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; CutStr,pcr ; ButStr,pcr ; CutStr,pcr ; ButStr,pcr ; CutStr,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; ButStr,pcr ; CheckVfp,pcr ; ButStr,pcr ; ButStr,pcr ; CheckVfp,pcr ; ButStr,pcr ; CheckVfp,pcr ; ButStr,pcr ; CheckVfp,pcr ; CheckVfp,pcr ; CutfStr,pcr ; OutStr,pcr ; CutfStr,pcr ; OutStr,pcr ; CutfStr,pcr ; OutStr,pcr ; CutfStr,pcr ; OutStr,pcr ; CutfStr,pcr ; OutStr,pcr ; CutfStr,pcr ; OutStr,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line. restore the returned success/fail condition.</pre>
127 127 128 129 130 131 132 133 134 139 136 137 138 139 130 142 143 144 145 144 145 144 145 155 156 157 158 155 156 161 162 163	FC7F FC81 Micrr FC85 FC85 FC87 FC89 FC89 FC95 FC99 FC95 FC99 FC95 FC95 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 SCC0 C60C D Dialects, Inc. uA SBC3 C601 SBF4 C600 SB86 C601 SB80 1AFA006C 1SFA022C 1SFA01F6 1SFA01F6 1SFA028 SB87 C145 261A 1SFA021B 33 C4DF C145 261A 1SFA021B 33 C4DF C145 261A 1SFA018 1AFA0058 1SFA018 1AFA0058 1SFA01FC 20C6 C150 26C2 1SFA001C 26BC 1SFA001C 26BC 1SFA002C 1SFA002C 1SFA002C 1SFA001C 26BC 1AFA002C 1SFA01E3 38 1AFA003D	.SM-HC12 Assemble BLLoop: andb CheckFErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab ar Tue, stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$01 ; FEELCK #\$01 ; TIOS BLPrompt,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; OutStr,pcr ; ChefStr,pcr ; BLLoop ; FErase,pcr ; ENot,pcr ; BLLoop ; Erased,pcr ; BLLoop ; Erased,pcr ; BLLoop ; Erased,pcr ; DutStr,pcr ; CheckVfp,pcr ; BLLoop ; FProg,pcr ; ELLoop ; FProg,pcr ; CheckVfp,pcr ; BLLoop ; FProg,pcr ; CheckVfp,pcr ; CheckVfp,pcr ; BLLoop ; FProg,pcr ; CheckTp,pcr ; CheckT</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. doeck for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line. restore the returned success/fail condition. point to the 'not programmed' string.</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 131\\ 132\\ 133\\ 134\\ 133\\ 139\\ 136\\ 137\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138\\ 138$	FC7F FC81 Micr. FC85 FC85 FC87 FC89 FC88 FC80 FC87 FC91 FC95 FC91 FC95 FC91 FC95 FC91 FC95 FC97 FC97 FC97 FC97 FC97 FC97 FC97 FC97	CC0034 SCC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C600 SB86 C601 SB80 LAFA006C 15FA012C 15FA0176 15FA0179 37 LAFA0058 15FA021B 33 C4DF C145 261A 15FA018 LAFA005A 260A 15FA018 LAFA005A 2604 LAFA005A 2604 2605 C150 26C2 15FA01FC 2006 C150 26C2 15FA001C 26BC 15FA0020 15FA0020 15FA0020 15FA0020 15FA0020 26DC	SM-HC12 Assemble BLLoop: andb CheckFErase: ; ChkProg: EEProgStat:	<pre>ldd std ldab ar Tue, stab ldab stab ldab stab ldab stab ldab stab ldab stab leax jsr pulb leax jsr pulb jsr bne jsr bne jsr bne leax jsr jsr jsr jsr jsr jsr jsr jsr jsr jsr</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TSCR #\$01 ; TSCR #\$01 ; getchar,pcr ; outStr,pcr ; OutStr,pcr ; OutStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; ChckVfp,pcr ; BLLoop ; FErase,pcr ; FErase,pcr ; EndErase ; EndErase ; FProg,pcr ; BLLoop ; ChckVfp,pcr ; BLLoop ; ChckVfp,pcr ; BLLoop ; ChckVfp,pcr ; BLLoop ; FProg,pcr ; CrLfStr,pcr ; BLLoop ; FProg,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; BLLoop ; FProg,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; CrLfStr,pcr ; DutStr,pcr ; DutStr,pcr ; DutStr,pcr ; CrLfStr,pcr ; DutStr,pcr ; DutS</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if it didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line. restore the returned success/fail condition. point to the 'not programmed' string. go display the string if programming failed.</pre>
$\begin{array}{c} 127\\ 128\\ 127\\ 128\\ 128\\ 128\\ 128\\ 131\\ 132\\ 133\\ 133\\ 133\\ 133\\ 133\\ 133$	FC7F FC81 Micr. FC85 FC85 FC87 FC89 FC88 FC88 FC88 FC88 FC88 FC88 FC88	CC0034 SCC0 C60C Dialects, Inc. uA SBC3 C601 SBF4 C680 SB86 C601 SB80 1AFA006C 1SFA022C 1SFA01F6 ISFA01F9 37 1AFA0058 1SFA021B 33 C4DF C145 261A 1SFA021B 33 C4DF C145 261A 1SFA003A 260A 1SFA018 1AFA005A 2604 1AFA005A 2604 1SFA01FC 20C6 C150 2662 1SFA006C 39 1AFA0020 1SFA01C 266C 1SFA001C 266C 1SFA001C 266C 1SFA001C 266C 1SFA0020 1SFA01E3 38 1AFA003B	SM-HC12 Assemble BLLoop: and CheckFErase: BadErase: ; ChkProg: EEProgStat:	<pre>ldd ldab std ldab er Tue, stab ldab stab ldab stab ldab stab ldab stab leax jsr jsr jsr jsr pulb ne jsr leax jsr bne leax jsr bne jsr bne jsr bne leax jsr pulc leax pulc leax pulc leax leax</pre>	<pre>#Baud9600 ; Baud #\$0c ; Mar 18, 1997 3:10 PM CR2 #\$01 ; FEELCK #\$00 ; TSCR #\$01 ; TIOS BLPrompt,pcr ; getchar,pcr ; getchar,pcr ; getchar,pcr ; futfstr,pcr ; CutStr,pcr ; CutStr,pcr ; GutStr,pcr ; Simple #'E' ; ChekProg ; FErase,pcr ; ELLoop ; FErase,pcr ; Butopp ; BLLoop ; FErase,pcr ; Butopp ; BLLoop ; FErase,pcr ; Butopp ; BLLoop ; FErase,pcr ; Butopp ; FICAG,pcr ; BLLoop ; CutStr,pcr ; BLLoop ; FFrase,pcr ; BLLoop ; CutStr,pcr ; BLLoop ; CutStr,pcr ; BLLoop ; FFrase,pcr ; BLLoop ; FProg,pcr ; PNot,pcr ; BadBrase ; Programmed,pcr ; FOGT, pcr ; BadBrase ; Programmed,pcr ;</pre>	<pre>set SCI to 9600 baud @ 8.0 MHz enable the transmitter & receiver. Page 3 disable the erasure or programming of the 2k bootblock. enable the timer system. set for fast flag clears. enable timer channel 0 as an output compare. point to the bootloader prompt. display it. get the command from the user. echo it. save it. go to the next line. restore the entered character. convert to upper case (only works for alpha characters). erase command entered? no. go check for the program command. yes. check for Vfp present. go print prompt if not present. yes. go erase the Flash. point to the 'not erased' string. branch if t didn't erase properly. if it did, point to the 'erased' string go back & print the prompt again. program command entered? no. go redisplay the command prompt. yes. check for Vfp present. go print prompt if not present. yes. go program the Flash. save the returned success/fail condition. go to the next line. restore the returned success/fail condition. point to the 'not programmed' string. go display the string if programmed' string. go display the string if programmed' string.</pre>

170 171 172 :**** ***** 173 ; The CheckVfp subroutine checks the SVFP bit in the FEECTL register to see if Vfp has been applied 174 ; to the Vfp pin. If Vfp is present, a zero or equal condition is returned. If Vfp is not present, 175 ; 176 a not zero or not equal condition is returned. 177 178 :**** 179 CheckVfp: 180 FCEF 87 clra ; assume that Vfp is present (set Z == 1). 181 FCF0 4EF70809 brset FEECTL, SVFP, VfpOK ; programming voltage present? 182 FCF4 1AFA003B leax NoVfpError,pcr ; no. inform the user. 183 FCF8 15FA01C9 jsr OutStr,pcr 184 FCFC 42 inca ; return Z == 0 (not zero condition) 185 FCFD 3D :XOqtV rts 186 187 188 :**** 189 : 190 FCFE 0D0A00 \$0d,\$0a,0 CrLfStr: fcb 191 FD01 0D0A28452972 BLPrompt: fcb \$0d,\$0a,"(E)rase or (P)rogram:",0 192 FD19 4E6F7420 ENot: fcb "Not. " Micro Dialects, Inc. uASM-HC12 Assembler Tue, Mar 18, 1997 3:10 PM -- Page 4 193 FD1D 457261736564 "Erased",0 Erased: fcb 194 FD24 4E6F7420 195 FD28 50726F677261 PNot: fcb "Not " Programmed: "Programmed".0 fcb \$0d,\$0a,"Vfp Not Present",0 196 FD33 0D0A56667020 NoVfpError: fcb 197 198 199 ****** 200 201 FD45 FProg: eau 202 FD45 C600 203 FD47 5B8D ldab #\$00 ; set the prescaler to /1.stab TMSK2 204 FD49 2006 ; don't send the 'pace' character the first time. bra FSkipFirst FSendPace: 205 FD4B C62A ldab # ' * ' ; the ascii asterisk is the pace character. 206 FD4D 15FA0149 putchar.pcr ; tell the host it's ok to send the next S-Record. isr 207 FD51 15FA00E4 FSkipFirst: jsr GetSRecord,pcr go get the S-Record. 208 FD55 2612 bne ProgDone ; non-zero condition means there was an error 209 FD57 E6FA0174 ; check the record type. ldab RecType.pcr #S9RecType 210 FD5B C139 cmpb was it an S9 record? 211 FD5D 270A beq ProgDone ; yes. we're done.
; no. was it an S0 record? 212 FD5F C130 cmpb #S0RecTvpe 213 FD61 27E8 ; yes. just ignore it. beq FSendPace ; no. that means it was an S1 record. go program the data into Flash. ; zero condition means all went ok. 214 FD63 15FA0003 isr ProgFBlock,pcr 215 FD67 27E2 FSendPace beq 216 FD69 3D ProgDone: rts ; if we fall through, we automatically return a non-zero condition. ; if we get here after detecting an S9 record, we'll return a zero condition. 217 218 219 220 _--;****** 221 222 223 FD6A CurrentPC ; save the current value of the PC set 224 0000 0 ; set PC to zero so we can use assembler to generate an org offset into the stack. 225 226 0000 ProgPulses: ds ; local variable to hold the number of programming pulses. 1 227 0001 PMarginFlag: 1 ; local variable to indicate we're applying the margin pulses ds 228 229 FD6A org CurrentPC 230 231 FD6A ProgFBlock: equ 232 FD6A 3B pshd ; easy way to allocate 2 bytes on the stack. ; get the S-Record (Flash) load address. 233 FD6B EEFA0162 LoadAddr,pcr ldx 234 FD6F 19FA0160 point to the received S-Record data. leay SRecData,pcr 235 FD73 6980 ProgLoop: clr ProgPulses, sp ; initialize the ProgPulses local variable. initialize the PMarginFlag local variable. 236 FD75 6981 clr PMarginFlag, sp 237 FD77 4CF702 FEECTL,LAT turn on the Flash address/data latches. bset 238 FD7A 180A4000 movb 0.v.0.x put the data into the latches. 239 FD7E 4D8680 PPulseLoop: bclr TSCR, TEN stop the timer so we can produce accurate time delays. 240 FD81 6280 ProgPulses, sp ; add 1 to the number of programming pulses we've applied. inc 241 FD83 E680 ldab ProgPulses, sp ; get the new value. 242 FD85 C132 #MaxProgPulses ; have we applied the maximum allowable programming pulses? cmpb PMarginLoop ; no. go apply a programming pulse.
; yes. now try applying 'MaxProgPulses' of margin. 243 FD87 2304 bls 244 FD89 18088101 #1,PMarginFlag,sp movb 245 FD8D CC00B0 PMarginLoop: ldd #us22 get the constant for a 22 uS delay. 246 FD90 D384 247 FD92 5C90 addd TCNT ; add it to the current value of the timer counter register. ; initialize the output compare register with the delay value. std TC0 248 FD94 4CF701 bset FEECTL, ENPE ; turn on Vfp TSCR,TEN TFLG1,\$01,* ; turn on the timer. ; wait here until Vfp has been applied for 22 uS. 249 FD97 4C8680 bset 250 FD9A 4F8E01FC brclr 251 FD9E 4DF701 bclr FEECTL, ENPE ; turn off Vfp. ; get the constant for a 11 uS delay. ; add it to the current value of the timer counter register. 252 FDA1 CC0058 1dd #u\$11 253 FDA4 D384 addd TCNT 254 FDA6 5C90 std TC0 ; initialize the output compare register with the delay value. brclr 255 FDA8 4F8E01FC ; wait here until Vfp has been removed for 11 uS.

256 FDAC E781 PMarginFlag,sp FDAC E781 tst PMarginFlag,sp ; are we ap Micro Dialects, Inc. uASM-HC12 Assembler Tue, Mar 18, 1997 3:10 PM -- Page 5 ; are we applying the programming margin pulses? ; no. go see if the data programmed properly. 257 FDAE 2706 beq CmpData 258 FDB0 6380 dec ProgPulses.sp ; yes. have we applied margin pulses equal to the numper of programming pulses? 259 FDB2 26D9 bne PMarginLoop ; no. go apply more margin pulses. 260 FDB4 200C bra PMarginDone ; yes. go check the data again. 261 262 FDB6 E600 CmpData: ldab 0,x ; get the data from the Flash memory. 263 FDB8 E140 cmpb 0,y ; same as the S-Record data? ; no. go apply some more programming pulses. 264 FDBA 26C2 PPulseLoop bne 265 FDBC 18088101 movb #1,PMarginFlag,sp ; yes. set the programming margin flag. 266 FDC0 20CB bra PMarginLoop ; go apply the margin programming pulses. 267 bclr 268 FDC2 4DF702 PMarginDone: FEECTL,LAT ; turn off the Flash address/data latches to prepare for programming the next location. 269 FDC5 E630 ldab 1,x+ ; get the data from the Flash memory for a final compare. 1,y+ ; same as the S-Record data? ; no. bad Flash memory (or Vfp not applied). 270 FDC7 E170 cmpb 271 FDC9 2606 PDone bne done with all the S-Record bytes? 272 FDCB 63FA0101 dec DataBytes,pcr ; no. program the next location.
; deallocate the locals. 273 FDCF 26A2 bne ProgLoop 274 FDD1 3A PDone: puld 275 FDD2 3D ; return. rts 276 : 277 278 279 FDD3 ; save the current value of the PC CurrentPC set 280 0000 0 org ; set PC to zero so we can use assembler to generate an offset into the stack. 281 282 0000 NumPulses: ds 1 ; local variable to hold the number of erase pulses. EMarginFlag: ds 283 0001 1 ; local variable to indicate we're applying margin erase pulses. 284 0002 NotErasedFlag: ds 1 ; local variable to indicate thet the Flash array is not erased. 285 ; 286 FDD3 org CurrentPC 287 288 FDD3 FErase: equ 289 FDD3 1B9D -3,sp ; allocate stack space for locals. leas ; set the prescaler to /32. 290 FDD5 C605 ldab #\$05 291 FDD7 5B8D stab TMSK2 292 FDD9 6981 EMarginFlag, sp ; clear the margin pulse flag. clr NumPulses, sp 293 FDDB 6980 ; clear the erase pulse count. clr FEECTL, LAT+ERAS 294 FDDD 4CF706 bset ; turn on the address/data latches & erase bit. 295 FDE0 7C8000 FlashStart ; write to any Flash address (data doesn't matter). std 296 NumPulses, sp 297 FDE3 E680 EraseLoop: ldab ; get the 'pulse' count 298 FDE5 C105 #MaxErasePulses applied the maximum number of erase pulses? cmpb 299 FDE7 2738 DoEMargin yes. go apply the erase margin pulse. beq 300 FDE9 6280 add 1 to the number of 100 mS 'pulses' to apply inc NumPulses, sp FEECTL, ENPE 301 FDEB 4CF701 turn on Vfp. PulseLoop: bset 302 FDEE CC61A8 #mS100 timer constant to produce a 100 mS delay ldd add it to the current value of the timer. initialize the output compare register. 303 FDF1 D384 addd TCNT 304 FDF3 5C90 TC0 std 305 FDF5 4F8E01FC brclr TFLG1,\$01,* check for the output compare flag to be set. 306 FDF9 4DF701 no turn off Vfp bclr FEECTL, ENPE : 307 FDFC CC00FA timer constant to produce a 1 mS delay ldd #mS1 308 FDFF D384 addd TCNT add it to the current value of the timer. 309 FE01 5C90 initialize the output compare register. std TC0 ; 310 FE03 4F8E01FC brclr TFLG1,\$01,* check for the output compare flag to be set. ; are we applying margin erase pulses? ; no. go check to see if the last pulse erased the array. ; yes. have we applied enough margin pulses? 311 FE07 E781 tst EMarginFlag,sp 312 FE09 2704 CheckErase beq 313 FEOB 6380 dec NumPulses.sp 314 FEOD 26DC bne PulseLoop ; no. go apply some more. 315 316 FEOF 6982 ; clear the erased flag CheckErase: clr NotErasedFlag, sp ; point to the start of the flash block. 317 FE11 CE8000 ldx #FlashStart 318 FE14 CD3C00 ldy #(FlashSize-BootBlkSize)/2; get a count of the number of words we're going to check. ldd 319 FE17 CCFFFF #\$FFFF ; the value of an erased word. EraseChkLoop: 320 FE1A AC31 cpd 2.x+ ; this word erased? -- Micro Dialects, Inc. uASM-HC12 Assembler Tue, Mar 18, 1997 3:10 PM -- Page 6 321 FE1C 260B NotErased ; no. go set flag & apply another erase pulse. bne 322 FE1E 0436F9 y,EraseChkLoop ; yes. decrement word count & go check the next word. dbne 323 324 FE21 E781 DoEMargin: tst EMarginFlag, sp ; have we already applied the margin pulse? 325 FE23 260C the NotErasedFlag. ; yes. we're done. the result of the erase function is in bne EraseDone EMarginFlag, sp 326 FE25 6281 ; no. set the 'margin pulse applied' flag. inc 327 FE27 20C2 bra PulseLoop ; go apply the margin erase pulse. 328 329 FE29 6282 NotErased: NotErasedFlag, sp ; array was not erased. flag the condition. inc EMarginFlag, sp 330 FE2B E781 331 FE2D 2602 ; have we already applied the margin pulse? ; yes. we're done. the Flash is bad. tst bne EraseDone 332 FE2F 20B2 bra EraseLoop ; haven't yet applied the margin pulse. go apply another erase pulse. 333 334 FE31 7900F7 FEECTL ; make sure that the LAT & ERAS bit is clear. EraseDone: clr 335 FE34 E682 ; get the erase result. ldab NotErasedFlag, sp 336 FE36 1B83 ; get rid of the locals. leas 3,sp 337 FE38 3D ; return. rts 338 339 ;

340 ;****	*****	*****	****	* * * * * * * *	* * * * * * * * * * * * * * * * * * * *	******
341			;			
342 343	FE39 0000		CurrentPC	set orq	* 0	; save the current value of the PC ; set PC to zero so we can use assembler to generate an
offse	t into	o the stack.				
344 345	0000		; SRecBytes:	ds	1	; holds the number of bytes in the received S-Record.
346	0001		CheckSum:	ds	1	; used for calculated checksum.
347			;	0.22	GurrontDO	
349	1035		;	org	Currentere	
350	FE39	1000	GetSRecord:	equ	*	·
351	FE3B	15FA0054	LookForSOR:	jsr	-2,sp getchar,pcr	; get a character from the receiver.
353	FE3F	C153		cmpb	#'S'	; start-of-record character?
354	FE41 FE43	26F8 15FA004C		bne jsr	getchar,pcr	; no. go back & get another character. ; yes. we found the start-of-record character (ASCII 'S')
356	FE47	C130		cmpb	#S0RecType	; found an SO (header) record?
357	FE49	2708	;	beq	SaveRecType	; no. go check for an S9 record.
359	FE4B	C139	CheckForS9:	cmpb	#S9RecType	; found an S9 (end) record?
360	FE4D	2704	;	beq	SaveRecType	; no. go check for an S1 record.
362	FE4F	C131	ChkForS1:	cmpb	#S1RecType	; found an S1 (code/data) record?
363	FE51	26E8	bne SaveRegTupe:	LookFor	SOR ; no. fal	se start-of-record character received. go check for another.
365	FE57	15FA0046	Saveneerype.	jsr	GetHexByte,pcr	; get the S-Record length byte.
366	FE5B	2620 6780	-	bne	BadSRec	; return if there was an error.
368	FE5F	6B81	sta	ab Che	eckSum,sp ; i	nitialize the checksum calculation with the data byte count
369	FE61	0003	subb	#3	; subtract	the load address & checksum field from the data field count.
370	FE63 FE67	1AFA0066		stab leax	DataBytes,pcr LoadAddr,pcr	; save the code/data field size. ; point to the load address/code/data/checksum buffer.
372	FE6B	15FA0032	RcvData:	jsr	GetHexByte,pcr	; get an S-Record data byte.
373	FE6F FE71	260C 6B30		bne stab	BadSRec 1,x+	; return 11 there was an error. ; save the byte in the data buffer.
375	FE73	EB81		addb	CheckSum, sp	; add the byte into the checksum.
376	FE75	6B81 6380		stab dec	CheckSum, sp	; save the result.
378	FE79	26F0		bne	RcvData	; no. go get some more.
379	FE7B	6281 1882	PadSpec.	inc	CheckSum, sp	; if checksum was ok, the result will be zero.
381	FE7F	3D	Bauskee.	rts	2,52	
382	• * * * * *	****	; * * * * * * * * * * * * * * * * * * *	******	*****	****
384	,		;			
	Micro	o Dialects, Inc. u	ASM-HC12 Assemble	er Tue	, Mar 18, 1997 3:10	PM Page 7
385	FE80		IsHex:	equ	*	
385 386	FE80 FE80	C130	IsHex:	equ cmpb	* #'0'	; less than ascii hex zero?
385 386 387 388	FE80 FE80 FE82 FE84	C130 250E C139	IsHex: bl	equ cmpb o No cmpb	* #'0' tHex ; y #'9'	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine?</pre>
385 386 387 388 389	FE80 FE80 FE82 FE84 FE86	C130 250E C139 2308	IsHex: bl	equ cmpb o No cmpb bls	* #'0' EHex ; y #'9' ISHexl	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication.</pre>
385 386 387 388 389 390 391	FE80 FE80 FE82 FE84 FE86 FE88 FE88	C130 250E C139 2308 C141 2506	IsHex: bl	equ cmpb o No cmpb bls cmpb o No	* Hex ; y H'9' ISHex1 #'A' Eltex ; y	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication.</pre>
385 386 387 388 389 390 391 392	FE80 FE82 FE84 FE86 FE88 FE8A FE8A	C130 C139 2308 C141 2506 C146	IsHex: bl	equ cmpb o No cmpb bls cmpb o No cmpb	* #'0' Hex ; y #'9' IsHex1 #'A' Elex ; y #'F'	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'?</pre>
385 386 387 388 389 390 391 392 393 394	FE80 FE82 FE84 FE86 FE88 FE8A FE82 FE82 FE82 FE90	C130 250E C139 2308 C141 2506 C146 2202 1404	IsHex: bl IsHex1:	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NotHex #S04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication.</pre>
385 386 387 388 390 391 392 393 394 395	FE80 FE82 FE84 FE86 FE88 FE88 FE8A FE8C FE8E FE90 FE92	C130 250E 2139 2308 C141 2506 C146 2202 1404 3D	IsHex: bl bl IsHex1: NotHex:	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc rts	* Hex ; y H'9' ISHex1 #'A' Elex ; y H'F' NotHex #\$04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication.</pre>
385 386 387 388 390 391 392 393 394 395 396 395	FE80 FE82 FE84 FE86 FE88 FE8A FE8C FE8E FE90 FE92	C130 250E 2139 2308 C141 2506 C146 2202 1404 3D	IsHex: bl bl IsHex1: NotHex: ;	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc rts	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NOTHEX #\$04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication.</pre>
385 386 387 388 390 391 392 393 394 395 396 397 ;***	FE80 FE82 FE84 FE86 FE88 FE88 FE82 FE82 FE90 FE92	C130 250E 2139 2308 C141 2506 C146 2202 1404 3D	IsHex: bl bl IsHex1: NotHex: ;	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc rts	* #'0' Hex ; y #'9' IsHex1 #'A' Etex ; y #'F' NotHex #\$04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication.</pre>
385 386 387 388 389 391 392 393 394 395 396 397 ;**** 398	FE80 FE82 FE84 FE86 FE88 FE8A FE82 FE90 FE92	C130 250E C139 2308 C141 2506 C146 2202 1404 3D	IsHex: bl bl IsHex1: NotHex: ; getchar:	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc rts ********	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NOTHEX #\$04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication.</pre>
385 386 387 390 391 393 394 395 396 396 397 ;**** 398 399 400	FE80 FE82 FE86 FE86 FE88 FE80 FE82 FE90 FE92 *****	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC	IsHex: bl IsHex1: NotHex: ; getchar:	equ cmpb o No cmpb bls cmpb o No cmpb bhi orcc rts ******** equ brclr	* #'0' Hex ; y #'9' IsHex1 #'A' Etex ; y #'F' NotHex ; y #\$04	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set.</pre>
385 386 387 388 390 391 392 393 394 395 396 397 ;**** 398 399 400 401	FE80 FE82 FE84 FE86 FE88 FE8A FE82 FE90 FE92 FE93 FE93 FE97 FE99	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D	IsHex: bl bl IsHex1: NotHex: ; getchar:	equ cmpb o No cmpb bls cmpb bhi o Co cmpb bhi orcc rts ********* equ brclr ldab	* #'0' Hex ; y #'9' IsHex1 #'A' Etex ; y #'F' NotHex ; y #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; retrieve.</pre>
385 386 387 388 390 391 392 393 394 395 396 395 396 397 ;**** 398 399 400 401 402 403	FE80 FE82 FE84 FE86 FE88 FE87 FE90 FE92 FE93 FE93 FE97 FE99	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D	IsHex: bl bl IsHex1: NotHex: ; getchar: ;	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return.</pre>
385 386 387 388 389 390 391 392 393 394 395 396 396 397 ;**** 398 399 400 401 402 403 403	FE80 FE82 FE84 FE88 FE88 FE88 FE80 FE90 FE92 ***** FE93 FE93 FE93 FE97 FE99	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D	IsHex: bl IsHex1: NotHex: ; getchar: ;	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return.</pre>
385 386 387 388 390 391 392 393 394 395 396 399 400 401 402 403 401 402 403	FE80 FE84 FE86 FE88 FE88 FE87 FE90 FE92 FE93 FE93 FE93 FE93	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D	IsHex: bl IsHex1: NotHex: ; getchar: ;	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NOTHEX #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return.</pre>
385 386 387 390 390 391 392 393 394 395 397 398 399 400 401 402 403 402 403 402 404 405 406 407	FE80 FE82 FE84 FE84 FE86 FE82 FE82 FE92 FE93 FE93 FE93 FE99 ******	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D	IsHex: bl IsHex1: NotHex: ; getchar: ; putchar:	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set.</pre>
385 386 387 390 390 391 392 393 394 395 399 400 401 402 403 402 403 402 403 402 403 402 403 402 404 404 405 406 407 408 409 409 409 409 409 409 409 409 409 409	FE80 FE82 FE84 FE84 FE86 FE82 FE92 FE93 FE93 FE93 FE99 FE99 FE99 FE99	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7	<pre>IsHex: bl IsHex1: NotHex: ; getchar: ; putchar:</pre>	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts *********	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character.</pre>
385 386 387 388 389 390 391 392 393 396 397 398 400 401 402 403 404 405 406 407 405 406 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 407 408 407 407 407 407 407 407 407 407 407 407	FE80 FE80 FE84 FE84 FE88 FE82 FE82 FE82 FE93 FE93 FE93 FE93 FE97 FE99 ****** FE94 FE94 FE94 FE94 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7 3D	<pre>IsHex: bl IsHex1: NotHex: ; getchar: ; putchar: ;</pre>	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts *********	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return.</pre>
385 386 387 388 389 390 391 392 393 394 395 397 400 401 402 403 404 402 403 404 405 406 407 408 409 411	FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE92 FE93 FE93 FE93 FE97 FE99 FE99 FE92 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7 3D	<pre>IsHex:</pre>	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts *********	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return.</pre>
385 3866387 388 389 390 391 392 393 394 403 402 403 402 403 404 405 406 407 405 406 407 408 409 401 402 403 404 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 408 407 407 408 407 407 407 407 407 407 407 407 407 407	FE80 FE80 FE84 FE88 FE88 FE82 FE82 FE82 FE82 FE92 FE93 FE93 FE93 FE97 FE99 ****** FE94 FE94 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7 3D	IsHex: bl bl IsHex1: NotHex: ; getchar: ; putchar: ;	equ cmpb o No cmpb bls o No cmpb bhi orcc rts ********* equ brclr ldab rts *********	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return.</pre>
385 386 387 388 389 390 391 392 393 394 400 401 402 403 404 405 406 407 406 407 408 409 401 411 ;****	FE80 FE80 FE84 FE88 FE88 FE82 FE82 FE82 FE82 FE92 FE92 FE93 FE93 FE97 FE99 FE99 FE94 FE94 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7 3D	<pre>IsHex: bl IsHex1: NotHex: ; getchar: ; putchar: ; GetHexByte:</pre>	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts *********	* #'0' Hex ; y #'9' IsHex1 #'A' Hex ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return.</pre>
385 386 387 388 389 390 391 392 393 394 395 397 400 401 402 403 404 407 405 406 407 406 407 408 409 411 ;**** 413 414 414	FE80 FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE92 FE93 FE93 FE93 FE97 FE99 FE99 FE94 FE94 FE94 FE94 FE94 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC480FC 5BC7 3D	<pre>IsHex:</pre>	equ cmpb o No cmpb bls cmpb bhi o No cmpb bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts	* #'0' EHEX ; y #'9' IsHex1 #'A' EHEX ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; tetrieve the character. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; get the upper nybble from the SCI. ; religi hay observater?</pre>
385 386 387 390 391 392 393 394 395 392 400 401 402 403 404 407 405 406 407 408 409 411 ;**** 412 413 414 415	FE80 FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE92 FE93 FE93 FE97 FE99 FE99 FE99 FE94 FE94 FE94 FE94 FE41 FE43	C130 250E C139 2308 C141 2506 C146 2202 1404 3D ***********************************	<pre>IsHex:</pre>	equ cmpb o No cmpb bls cmpb bhi o Corc rts ********* equ brclr ldab rts ********* equ brclr stab rts *********	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NOTHEX #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary.</pre>
385 386 387 388 389 390 391 392 393 394 400 401 402 403 404 407 405 406 407 408 409 411 ;**** 412 413 414 415 416 417 417 77 27	FE80 FE80 FE82 FE84 FE88 FE82 FE82 FE82 FE82 FE92 FE93 FE93 FE97 FE99 FE99 FE99 FE94 FE94 FE94 FE94 FE41 FE41 FE45 FE47 FE47	C130 250E C139 2308 C141 2506 C146 2202 1404 3D ***********************************	<pre>IsHex: bl bl IsHex1: NotHex: ; ; ; putchar: ; GetHexByte:</pre>	equ cmpb o No cmpb bls cmpb bhi o cmp bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********** equ brclr bcclr bcclr bcclr stab rts	* #'0' EHEX ; y #'9' IsHex1 #'1' EHEX ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; loop waiting for the RDRF bit to be set. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; yet the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; ho. return with a non-zero ccr indication. ; return.</pre>
385 386 387 390 391 392 393 394 395 396 397 400 401 402 403 404 407 405 406 407 408 409 411 ;**** 412 413 414 415 416 417 418	FE80 FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE92 FE93 FE93 FE97 FE99 FE99 FE99 FE94 FE94 FE94 FE94 FE41 FE43 FE45 FE44 FE44	C130 250E C139 2308 C141 2506 C146 2202 1404 3D ***********************************	IsHex: IsHex: bl IsHex1: NotHex: ; ; ; putchar: ; GetHexByte: OK1:	equ cmpb o No cmpb bls cmpb bhi o cmp bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********* equ brclr ldab rts brclr ldab rts its its its its its its its its its i	* #'0' EHEX ; y #'9' ISHEX1 #'A' EHEX ; y #'F' NOTHEX #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no return a zero ccr indication. ; retrieve the character. ; return. ; loop waiting for the RDRF bit to be set. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; return. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; no. return with a non-zero ccr indication. ; convert the ascii-hex character to binary. ; shift it to the upper 4-bits.</pre>
385 386 387 390 391 392 393 394 400 401 402 403 404 407 405 406 407 408 409 410 411 ;**** 412 414 415 416 417 418 419 410 417 418 419 410 417 418 419 410 417 418 419 410 417 418 419 410 417 418 419 410 410 410 410 410 410 410 410 410 410	FE80 FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE93 FE93 FE93 FE97 FE99 FE99 FE94 FE94 FE94 FE94 FE94 FE94	C130 250E C139 2308 C141 2506 C146 2202 1404 3D ***********************************	<pre>IsHex: bl bl IsHex1: NotHex: ; getchar: ; putchar: ; GetHexByte: OK1:</pre>	equ cmpb o No cmpb bls cmpb bhi o cmp bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********** equ brclr ldab rts **********	* #'0' EHEX ; y #'9' IsHex1 #'A' EHEx ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no return a zero ccr indication. ; rot return a zero ccr indication. ; retrieve the character. ; return. ; loop waiting for the TDRE bit to be set. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; return. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; no. return with a non-zero ccr indication. ; convert the ascii-hex character to binary. ; shift it to the upper 4-bits.</pre>
385 386 387 390 391 392 393 394 400 401 402 403 404 407 408 409 410 411 ;**** 412 413 414 415 416 417 418 414 417 418 419 420 421 422	FE80 FE82 FE84 FE88 FE88 FE82 FE82 FE82 FE82 FE93 FE93 FE93 FE93 FE97 FE99 FE99 FE94 FE94 FE94 FE94 FE94 FE45 FE47 FE44 FE44 FE44 FE44 FE44 FE44 FE44	C130 250E C139 2308 C141 2506 C146 2202 1404 3D ***********************************	<pre>IsHex: bl bl IsHex1: NotHex: ; getchar: ; putchar: ; GetHexByte: OK1:</pre>	equ cmpb o No cmpb bls cmpb bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********** equ brclr ldab rts brclr ldab rts brclr ldab rts brclr ldab rts brclr brclr ldab rts	* #'0' EHEX ; y #'9' IsHex1 #'1' EHEX ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no return a zero ccr indication. ; retrieve the character. ; return. ; loop waiting for the RDRF bit to be set. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; et urn. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; no. return with a non-zero ccr indication. ; convert the ascii-hex character to binary. ; shift it to the upper 4-bits. ; save it on the stack. ; get the lower nybble from the SCI.</pre>
385 386 387 390 391 392 393 394 400 401 402 403 404 407 405 406 407 408 409 410 411 ;**** 412 412 414 415 416 417 418 414 415 416 417 418 412 422 423	FE80 FE80 FE82 FE84 FE88 FE82 FE82 FE82 FE82 FE93 FE93 FE93 FE93 FE97 FE99 FE94 FE94 FE94 FE94 FE94 FE94 FE24 FE24 FE24 FE24 FE24 FE24 FE24 FE2	C130 250E C139 2308 C141 2506 C146 2202 1404 3D **********************************	<pre>IsHex: bl bl IsHex1: NotHex: ; getchar: ; putchar: ; GetHexByte: OK1:</pre>	equ cmpb o No cmpb bls cmpb bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********** equ brclr ldab rts brclr ldab rts **********	* #'0' EHEX ; y #'9' IsHex1 #'1' EHEx ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no return a zero ccr indication. ; rot return a zero ccr indication. ; retrieve the character. ; return. ; loop waiting for the RDRF bit to be set. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; no. return with a non-zero ccr indication. ; convert the ascii-hex character to binary. ; shift it to the upper 4-bits. ; save it on the stack. ; get the lower nybble from the SCI. ; valid hex character? ; walid hex character? ; walid hex character? ; solution the stack. ; get the lower nybble from the SCI. ; valid hex character? ; walid hex character? ; walid hex character? ; solution the stack. ; get the lower nybble from the SCI. ; valid hex character? ; valid hex character?</pre>
385 386 387 390 391 392 393 394 400 401 402 403 404 407 405 406 407 408 409 410 411 ;**** 412 412 414 415 416 417 418 419 420 421 422 423 424 424 425	FE80 FE80 FE82 FE84 FE88 FE82 FE82 FE82 FE93 FE93 FE93 FE93 FE93 FE97 FE99 FE94 FE94 FE94 FE94 FE94 FE94 FE44 FE4	C130 250E C139 2308 C141 2506 C146 2202 1404 3D 4FC420FC D6C7 3D 4FC420FC D6C7 3D 7FC 3D 7FC 3D 7FC 3D 77F0 77F0 77F0 77F0 77F0 77F0 77F0 77	<pre>IsHex: bl bl IsHex1: NotHex: ; getchar: ; putchar: ; GetHexByte: OK1:</pre>	equ cmpb o No cmpb bls cmpb bhi orcc rts ********* equ brclr ldab rts ********* equ brclr stab rts ********** equ brclr ldab rts brclr ldab rts brclr ldab rts brclr br	* #'0' EHEx ; y #'9' IsHex1 #'A' EHEx ; y #'F' NotHex #\$04 ************************************	<pre>; less than ascii hex zero? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex nine? ; yes. character is hex. return a zero ccr indication. ; less than ascii hex 'A'? res. character is not hex. return a non-zero ccr indication. ; less than or equal to ascii hex 'F'? ; yes. character is hex. return a non-zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; no. return a zero ccr indication. ; rot return a zero ccr indication. ; retrieve the character. ; return. ; loop waiting for the RDRF bit to be set. ; return. ; loop waiting for the TDRE bit to be set. ; send the character. ; return. ; get the upper nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; no. return with a non-zero ccr indication. ; convert the ascii-hex character to binary. ; shift it to the upper 4-bits. ; save it on the stack. ; get the lower nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; shift it to the upper 4-bits. ; save it on the stack. ; get the lower nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; shift it to the upper 4-bits. ; save it on the stack. ; get the lower nybble from the SCI. ; valid hex character? ; yes. go convert it to binary. ; remover saved upper byte from the stack</pre>

; convert the ascii-hex character to binary. 427 FEB6 0704 OK2: CvtHex bsr 428 FEB8 EBB0 addb ; add it to the upper nybble. 1,sp+ 429 FEBA 87 clra ; simple way to set the Z ccr bit. 430 FEBB 3D rts ; return. 431 ; 432 ;****** 433 #'0' ; subtract ascii '0' from the hex character. 434 FEBC C030 CvtHex: subb ; was it a decimal digit? ; yes. ok as is. #\$09 435 FEBE C109 cmpb 436 FEC0 2302 bls CvtHexRtn ; no. it was an ascii hex letter ('A' - 'F'). 437 FEC2 C007 subb #\$07 438 FEC4 3D CvtHexRtn: rts 439 440 441 . 442 FEC5 * OutStr: equ ; send a null terminated string to the display. 443 FEC5 E630 444 FEC7 2705 ldab 1,x+ ; get a character, advance pointer, null? beq OutStrDone ; yes. return. 445 FEC9 15F9CE putchar,pcr ; no. send it out the SCI. isr 446 FECC 20F7 ; go get the next character. bra OutStr 447 FECE 3D OutStrDone: rts 448 -- Micro Dialects, Inc. uASM-HC12 Assembler Tue, Mar 18, 1997 3:10 PM -- Page 8 449 ; 450 ;**** 451 452 FECF BootLoadEnd: eau 453 454 455 ;Global Variable declarations 456 ; 457 . ; received record type. ascii '0' = S0; ascii '1' = S1; ascii '9' = S9
; number of data bytes in the S-Record.
; load address of the S-Record. 458 FECF RecType: ds 1 DataBytes: LoadAddr: 1 459 FED0 ds 460 FED1 ds 2 ىيە 65 ds 461 FED3 SRecData: ; S-Record data storage. (handle 64-byte S-Records + received checksum) 462 ; 463 464 ; 465 ; 466 ; 467 FFD0 org \$ffd0 ; 468 469 FFD0 FC24 BDLC: JBDLC dw 470 FFD2 FC28 ATD: dw JATD 471 FFD4 FFFF 472 FFD6 FC2C \$ffff dw SCI0: JSCI0 dw 473 FFD8 FC30 SPI: dw JSPI PACCIE: 474 FFDA FC34 JPACCIE. dw 475 FFDC FC38 PACCOv: JPACCOv dw 476 FFDE FC3C 477 FFE0 FC40 TimerOv: dw JTimerOv TimerCh7: JTimerCh7 dw 478 FFE2 FC44 TimerCh6: dw JTimerCh6 479 FFE4 FC48 480 FFE6 FC4C TimerCh5: dw JTimerCh5 TimerCh4: dw JTimerCh4 481 FFE8 FC50 TimerCh3: dw JTimerCh3 482 FFEA FC54 TimerCh2: dw JTimerCh2 483 FFEC FC58 TimerCh1: dw JTimerCh1 484 FFEE FC5C TimerCh0: dw JTimerCh0 485 FFF0 FC60 RTT: dw TRTT 486 FFF2 FC64 IRO: dw JIRO 487 FFF4 FC68 XIRQ: dw JXIRQ 488 FFF6 FC6C SWT: dw TSWT 489 FFF8 FC70 Illop: dw JIllop 490 FFFA FC74 COPFail: dw JCOPFail ClockFail: 491 FFFC FC78 dw JClockFail 492 FFFE FC00 Reset: BootStart dw 493 494 0000 end Errors: None Labels: 155 Last Program Address: \$FFFF Last Storage Address: \$FFFF Program Bytes: \$02FF 767 Storage Bytes: \$004C 76

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

MCUinit, MCUasm, MCUdebug, and RTEK are trademarks of Motorola, Inc. MOTOROLA and ! are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us: USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver Colorado 80217. 1-800-441-2447, (303) 675-2140 Mfax™: RMFAX0@email.sps.mot.com - TOUCHTONE (602) 244-6609, U.S. and Canada Only 1-800-774-1848 INTERNET: http://Design-NET.com JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315 ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298 Mov is a trademark of Matarala. Inc.

