EGR 270 Fundamentals of Computer Engineering File: Wookie .doc

Example: Mini IDE Assembler and Wookie Simulator

MiniIDE

MiniIDE is a freeware assembler for the 68HC11 and 68HC12 that can be used to create an executable machine language program (.S19 file) and a listing (.LST file) from an assembly language program (.ASM file).

- 1) Install MiniIDE
 - MiniIDE can be downloaded from the instructor's web page or from the following URL: www.mgtek.com/miniide/
 - The installation program should install MiniIDE in a new folder named C:\ProgramFiles\MGTEK\MiniIDE\ as shown below:



2) Launch MiniIDE

- Launch MiniIDE using Start <u>All Programs MGTEK MiniIDE</u>
- 3) Set Options in MiniIDE
 - Select **Build Options**

Select	Build -	– Optio	ns						
🚍 N	linill	DE							
File	Edit	View	Build	Terminal	Help		_		
] 🗅 🚅 🖬 🍜		월급 Build Current 월급 Build Project		Ctrl+F7 F7		B. B. K .			
		<u>S</u> e <u>R</u> e	t Project Fi Ilease Proje	le :ct File					
			Qr	tions		Alt+F7			

- Select the <u>Tools</u> tab in the <u>Options</u> window (see screen on left below) Select the assembler <u>asm11.exe</u> that is located in the MiniIDE folder and select OK.
- Select the <u>Assembler</u> tab in the <u>Options</u> window (see screen on right below) Select <u>Generate listing file</u> and <u>Warning level 4</u> (the highest level so all warnings are displayed).
- Select <u>OK</u>.

Options 🔀	Options 🔀
General Terminal Tools Assembler Lools You can specify or browse for the name of the tools you want to use. Assembler: C:\Program Files\MGTEK\MinilDE\\asm11.exe Use output window Close window on exiting	General Terminal Tools Assembler Settings Specify the settings that you whish to pass to the assembler. Disable warnings Warning level: 4 Compatibility mode Generate jisting file Show cycle counts Show cycle counts Mo Symbols All Used only Options: -w/4 -1 -w/4 -1
OK Cancel Apply	OK Cancel Apply

4) Open or create an assembly language program

- To create a new assembly language program, select <u>File New</u> and enter your program. Then select <u>File Save</u> and save your program using an <u>asm</u> extension.
- To open an existing assembly language program, select <u>File Open</u> and select your assembly language program (Ex2a.asm was opened in the example shown below)

🛢 MinilDE - [Ex2a.asm]	
🛕 File Edit View Build Teri	minal Window Help
] D 🚅 🖬 🍜 X 🖻 🛍 •	
* Program Ex2a.asm * Program to store three * M[\$01], M[\$02], and M[ORG \$100 LDAA #\$0A STAA \$01 LDAA #\$14 STAA \$02 LDAA #\$1E STAA \$03 ADDA \$01 ADDA \$02 STAA \$04 END	e numbers (decimal 10, 20, and 30) into memory locations \$03] and to store the sum in memory location M[\$04] ;Store program at memory location M[\$100] ;Load hexadecimal \$0A into register A ;Store in memory location M[\$01] ;Add the contents of M[\$01] to A ;Add the contents of M[\$02] to A ;Store the sum in M[\$04]

5) Assemble the program

• Select <u>**Build – Build**</u> (YourFileName.asm).

HinilDE - [Ex2a.asm]							
A File Edit View	Build Terminal Window Help						
🗅 🚅 🖬 🎒 X	Build C:\asmtest\Ex2a.asm	Ctrl+F7	B. D. M.				
* Program Ex2a.	별길 Build Project	F7					
<pre>* Program to st * M[\$01], M[\$02</pre>	<u>S</u> et Project File		and 30) into memory locations in memory location M[\$04]				
ORG \$10	<u>R</u> elease Project File		location M[\$100]				
LDAA #\$ STAA \$0	Options	Alt+F7	o register A .M[\$01]				

- Assembling the program results in the creation of:
 - 1. Error messages if the assembler detects any errors (note 0 errors in the example below)
 - 2. Object code or machine code (.S19 file)
 - 3. Listing file (.LST) which shows op codes generated, offsets for branch statements, etc.



• Note that the .S19 and .LST files are created in the same folder where the .ASM file is stored as shown below.

asmtest 🗧							
File Edit View Favorites Tools Help							
🚱 Back 🔹 🛞 🕤 🏂 Search 🔊 Folders 🛄 🔹							
Address 🗁 C:\asmtest							
Folders	Name	Size	Туре				
 Ø Desktop 	EX2A.S19 Ex2a.lst EX2a.asm	1 KB 2 KB 1 KB	Wookie Document LST File ASM File				

- Listing File. You can open the listing file using MiniIDE or with any word processor, such as Notepad. The listing file includes:
 - o original assembly language instructions (mnemonics and operands)
 - o machine code (op codes and operands)
 - the starting memory location
 - program counter



• <u>S19 File</u> The S19 file isn't a file that we normally open, but let's look at it here just to see that it mainly contains the starting memory address and machine code. See if you can spot the starting address and the machine code by comparing it to the LST file above. The S19 file is downloaded into ROM in the 68HC11.



Wookie 68HC11 Simulator

The Wookie simulator is an excellent freeware program that can be used to simulate 68HC11 programs before actually downloading them into a microprocessor in lab. Wookie, short for "Wireless Object-Oriented Kindly Interfaced Emulator," is a Win32 emulator for 68HC11-based software development that was developed by a senior design team at the Milwaukee School of Engineering (MSOE). The software can be downloaded from the instructor's web page or from various websites, including: http://www.msoe.edu/eecs/ce/ceb/resources/

Before running Wookie, let's take a closer look at the program Ex2a.asm shown above and see what it does.

- It stores the program at memory location \$100, so the program counter will start here.
- It loads \$0A (decimal 10) into accumulator A and then stores it at memory location [\$01].
- It loads \$14 (decimal 20) into accumulator A and then stores it at memory location [\$02].
- It loads \$1E (decimal 30) into accumulator A and then stores it at memory location [\$03].
- It adds the contents of M[\$01] to accumulator A (so A will now contain decimal 30+10=40 or \$28)
- It adds the contents of M[\$02] to accumulator A (so A will now contain decimal 40+20=60 or \$3C)
- It stores the value in accumulator A (\$3C or decimal 60) at memory location [\$04].

🚍 MinilDE	- [Ex2a.lst]		
A File Edit	View Build Terminal Window	Help	
🗅 🖻 📕	🎒 X 🖻 🛍 🕫 🖓 🖻]] 11: 11: 11: 11: 11: 11: 11: 11: 11: 11:	
C:\asmtest	t\Ex2a.lst - generated by	MGTEK Assembler ASM11 V1.	26 Build 144 for WIN32 (x86)
1: 2: 3: 4:		* Program Ex2a.asm * Program to store th * M[\$01], M[\$02], and	nree numbers (decimal 10, 20, 1 M[\$03] and to store the sum
5: 6: 7:	=00000100 0100 86 0A 0102 97 01	ORG \$100 LDAA #\$0A STAA \$01	Store program at memory; Load hexadecimal \$0A int; Store in memory location;
8: 9: 10:	0104 86 14 0106 97 02 0108 86 1E	LDAA #\$14 STAA \$02 LDAA #\$1E	
11: 12: 13: 14: 15:	010A 97 03 010C 98 01 010E 98 02 0110 97 04	STAA \$03 ADDA \$01 ADDA \$02 STAA \$04 END	:Add the contents of M[\$0 :Add the contents of M[\$0 :Store the sum in M[\$04]

Now we will load the S19 file into Wookie and watch it change the accumulator contents and memory address contents as we step through the program.

1) Install Wookie

- Wookie can be downloaded from the instructor's web page or from various websites, including: <u>http://www.msoe.edu/eecs/ce/ceb/resources/</u>
- Extract the zip file Wookie171.zip into the location of your choice (C:\Wookie is used below).



2) Launch the Wookie simulator

• Locate Wookie171.exe using Windows Explorer or MyComputer and launch the program.

🚟 Untitled - 6811 Wookie
File Simulator Memory Tools Window Help
IRQ Port A M68HC11 CPU XIRQ XIRQ Port B Erowse Mem. Reset Port C
Memory Watch Registers Port D
View Code Display Port E
Break Keypad Pin Scope

3) Loading the S19 file

• Select <u>File – Load S19 File</u> and select the .S19 file (Ex2A.S19 in this example).

We Untitled - 6811 Wookie		
File Simulator Memory Tools Window Help		
IRQ IRQ Port A NIRQ NIRQ Port B Browse Mem. Reset Port C Memory Watch Registers Port D View Code Display Port E Break Keypad Pin Scope	Look in: Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest Image: samtest	Open Cancel

• After loading the S19 file above, the <u>Set HC11 Mode</u> window will automatically appear. Leave the mode as <u>Brief Case</u>, but change the starting address to the address in the ORG statement in your .asm file). In the example above, the statement ORG \$100 gives the starting address.

Untitled - 6811 Wookie	
File Simulator Memory Tools Window Help	
IRQ IRQ Port A M68HC11 CPU XIRQ Port B Browse Mem. Reset Port C Memory Watch Registers Port D View Code Display Port E Break Keypad Pin Scope	Set HC11 Mode Mode: Brief Case Start Address: 0100 OK
Chang addre (OR	ge the address to match the ss in your ORG statement CG \$100 in this example)

• The <u>Choose LST file format</u> window will now automatically appear. Change the <u>File Type</u> to <u>AS6811/ASM11 (addr code)</u> and leave the <u>Address Offset</u> at <u>0</u> (default).

躍 EX2A.S19 - 6811 Wookie								
File Simulator Memory Tools Window Help								
M68HC11 CPU XIRQ Browse Men. Memory Watch Register	Port A Port B Port C File Type Address Offset (for GCC files)							
View Code Display								
Break Keypad	Pin Scope							

Wookie should now display your code in the <u>Code View</u> window. You might need to drag the window to the right and resize it. The <u>View Code</u> button turns this window off and on (leave it on).
 <u>EX2A.519 - 6811 Wookie</u>

File Simulator Memory Tools Window Help						
M68HC11 CPU	Code View C:\asmtes	t\Ex2a.lst - generated]	by MGTEK Assembler ASM11 VI.	26 Build 144 for WIN32 (x86) - Sat j		
Browse Mem. Reset Reset Port C	1:		* Program Ex2a.asm			
	2:		* Program to store th	ree numbers (decimal 10, 20, and 30)		
Memory Watch Registers Port D	3:		* M[\$01], M[\$02], and	M[\$03] and to store the sum in memo		
	4:					
View Code Display Port E	5:	=00000100	ORG \$100	;Store program at memory location		
	6:	0100 86 OA	LDAA #\$OA	;Load hexadecimal \$0Å into regist		
Break Keyned Din Scone	7:	0102 97 01	STAA \$01	Store in memory location M[\$01];		
Point Reypid	8:	0104 86 14	LDAA #\$14			
	9:	0106 97 02	STAA \$02			
	10:	0108 86 1E	LDAA #\$1E			
	11:	010A 97 03	STAA \$03			
	12:	010C 9B 01	ADDA \$01	;Add the contents of M[\$01] to A		
	13:	010E 9B 02	ADDA \$02	;Add the contents of M[\$02] to A		
	14:	0110 97 04	STAA \$04	;Store the sum in M[\$04]		
	15:		END			
	Load LST File		Clear	1		

4) **Opening Useful Windows in Wookie**

- After loading the .S19 file, the <u>Code View</u> window should be open.
- Also click on the <u>M68HC11 CPU</u> button to open the MCU window which will show the contents of some key registers.
- Also click on the <u>Memory Watch</u> button. Select <u>Add</u> and enter the memory addresses \$01, \$02, \$03, and \$04 since they are used in this example. The <u>Add Memory Watch</u> window below shows memory address \$04 being added.



5) <u>Single Step Through the Simulation.</u> Each time you click on the icon of the person walking (or select <u>Simulator – Simulator Step</u> or press the space bar), the Program Counter (PC) should advance to the address of the next instruction. You can check the contents of memory addresses and registers as you advance.





After 8 steps: (after executing ADDA \$02)	After 9 steps: (after executing STAA \$04)			
PC = \$0110	PC = \$0102			
ACCA = \$3C	ACCA =			
M[\$01] = \$0A	M[\$01] = \$0A			
M[\$02] = \$14	M[\$02] = \$14			
M[\$03] = \$1E	M[\$03] = \$1E			
M[\$04] = \$78 (junk)	M[\$04] = \$3C			
MCU Memory Watch	MCU Memory Watch			
Name Value ACCA \$3C ACCB \$00 ACCD \$3C00 ACCD \$3C00 IX \$0000 PC \$0110 CCR 0101000 SX H IN Z V C Add	Name Value ACCA \$3C ACCB \$00 ACCD \$3C00 ACCD \$3C00 IX \$0000 SP \$0000 PC \$0112 CCR 0101 000 0 SX HINZYC Add			
Modify Update Close Delete Base Close	Modiry Update Liose Delete Base Close			

- Note that the simulation was correct.
- 6) **<u>Run the Entire Simulation</u>** You can also run the entire simulation with one command as described below.
 - <u>Reset the PC</u>. Before rerunning a program, set the PC back to the starting address (\$0100 in this case) using the <u>Modify</u> button in the <u>Memory Watch</u> window (see below). Note that you can also reset other registers and memory addresses.



• <u>**Run the simulation**</u>. Run the full simulation by pressing the button that looks like a traffic light twice (turns green then red) (or select Simulation – Simulations Start/Stop)

PC reset to \$0100		After running the entire	simulation:
ACCA and M[\$01] - M	[[\$04] reset to \$00		
MCU	Memory Watch	MCU	Memory Watch
Name Value ACCA \$00 ACCB \$00 ACCD \$0000 IX \$0000 IY \$0000 SP \$0000 PC \$0100 CCR 0101000	name address hex value \$1 \$00 \$2 \$00 \$3 \$00 \$4 \$00	Name Value ACCA \$3C ACCB \$00 ACCD \$3C00 IX \$0000 IY \$0000 SP \$0000 PC \$0003 CCR 0101000	name address hex value \$1 \$0A \$2 \$14 \$3 \$1E \$4 \$3C
SXHINZVC	Add Modify Update	SXHINZVC	Add Modify Update
Modify Update Close	BaseClose	Modify Update Close	Delete Base Close

Simulating Programs that will be used on the MicroStamp11

The example above was generic and was not geared for a specific version of the 68HC11. However, in lab we use the MicroStamp11, a microcontroller board based on the MC68HC11D0. In general the 68HC11 can be configured to operate in three modes: single-chip mode, bootstrap mode, and expanded chip mode. The MicroStamp11 operates in **expanded-chip mode**.

Example 2: Simulate the program PA6blink.asm

- This program is used in Lab 8.
- The program is designed to make an LED connected to output PA6 blink ON for ½ second and then OFF for ½ second indefinitely.
- Let's use Wookie to simulate it before downloading it into the MicroStamp11 to insure that the program is correct.
- Before beginning the simulation, use Mini IDE to assemble PA6blink.asm in order to create the machine code, PA6blink.s19.

1) Launch the Wookie simulator

• Locate Wookie171.exe using Windows Explorer or MyComputer and launch the program.

entrate a			
File Simulator N	∕lemory Tools V	Vindow Help	
M68HC11 CPU		Port A	
	XIRQ	Port B	
Browse Men.	Reset 🚺	Port C]
Memory Watch	Registers	Port D	
View Code	Display	Port E]
Break Point	Keypad	Pin Scope	

2) Loading the S19 file

• Select <u>File – Load S19 File</u> and select the .S19 file (PA6blink.S19 in this example).

🛱 Untitled - 6811 Wookie		
File Simulator Memory Tools Window Help N68HC11 CPU IRQ Port A Port B Port B Port C Nemory Watch Registers Port D View Code Display Port B Break Keypad Pin Scope	Open Look in: Image: asmtest Image: My Recent Documents Image: BNE.S19 Image: Desktop Image: EX2A.S19 My Documents Image: EX2A.S19 Image: My Network File name: EX2A.S19 Image: EX2A.S19	(2) (X)
	Files of type: Machine Code Files (*.s19)	Cancel

After loading the S19 file above, the <u>Set HC11 Mode</u> window will automatically appear. Recall that the MicroStamp11 operates in expanded-chip mode, so change the mode to <u>Rug Warrior</u> <u>Expanded</u> and change the starting address to the address in the ORG statement in the asm file). In PA6blink.asm, the statement ORG \$8000 gives the starting address.

Image: Simulator Memory Tools Window Help Image: Simulator Memory Tools Window He	Set HC11 Mode	Г	
Memory Watch Registers Port D View Code Display Port E Break Keypad Pin Scope	Mode: Rug Warrior Expanded V Start Address: 8000 OK		Note that the MicroStamp11 operates in <u>expanded mode</u> .
Chang addres	e the address to match the ss in your ORG statement G \$8000 in this example).		

• The <u>Choose LST file format</u> window will now automatically appear. Change the <u>File Type</u> to <u>MGTEK ASM11 [line: addr code]</u> and leave the <u>Address Offset</u> at <u>0</u> (default).

🕮 PA6blink.s19 - 6811 Wookie						
File Simulator Memory Tools V	Window Help					
M68HC11 CPU	Port A Port B	Choose LST file format. File Type				
Browse Mem. Reset	Port C	MGTEK ASM11 [line: addr.co.				
Memory Watch Registers	Port D	Address Offset (for GCC files)				
View Code Display	Port E	ОК				
Break Keypad	Pin Scope					

• Wookie should now display your code in the <u>Code View</u> window. You might need to drag the window to the right and resize it. The <u>View Code</u> button turns this window off and on (leave it on).

🛱 PA6blink.s19 - 6811 Wookie	Code View				
File Simulator Memory Tools Window Help					
File Simulator Memory Tools Window Help H66HCll CPU IRQ IRQ Port A Port B Browse Hem. Reset Image: Constraint of the section of the sec	1: 2: 3: 4: 5: 6: 7: 8: 9:	=00008000 8000 86 04 8002 97 3F 8004 8E 00FF	* Filena * Sample * This p Begin: L S L	me: PA6blink.asm program from Mic rogram will cause RG \$8000 DA #\$04 TAA \$3F DS #\$FF	<pre>*roStampl1 Start * an LED connect ;Store pr ;Disable ; Check ;Initiali ; we wil</pre>
View Code Display Port E Break Keypad Pin Scope	10: 11: 12: 13: 14: 15: 16:	8007 86 40 8009 97 00 800B 8D 08 800D 86 00 800F 97 00 8011 8D 02 8013 20 F2	Loop: Li S B Li S B B B	DAA #\$40 TAA \$00 SR Delay DAA #\$00 TAA \$00 SR Delay RA Loop	;Write a ; Check ;Branch t ;Write a ;Create a ;Create a
	17: 18: 19: 20: 21: 22: 23: 24: 25: 26: 27:	8015 8015 18CE FFFF 8019 1809 801B 1808 801D 1809 801F 26 F8 8021 39 =0000FFFE FFFE 8000	Delay: Li Dl: D Bi R 0 Fi	DY #\$FFFF EY NY NE D1 TS RG \$FFFE DE Begin	<pre>;Subrouti ;Load the ;Decremen ;Decremen ;Branch 1 ;Return t ;Define t ; code (</pre>

3) Opening Useful Windows in Wookie

- After loading the .S19 file, the <u>Code View</u> window should be open (if not, click <u>View Code</u>).
- Also click on the <u>M68HC11 CPU</u> button to open the MCU window which will show the contents of some key registers.
- The instructor noted a problem in selecting the Registers window and the PortA window: The addresses aren't correct for the MicroStamp11. Table 4-1 (see class notes) indicates that the address for PORTA is \$0000, not \$1000. Similarly, the address for PORTD is \$0008, not \$1008. In fact, all of the table 4-1 addresses have the form \$00xx, not \$10xx as shown in Wookie. It doesn't seem that there is any way to change them in Wookie either. If anyone can solve this problem, let the instructor know! It is disappointing that we can't use the PortA window as it has a clever let of 8 LED's that show the values on PortA. The 1's and 0's above the LED's indicate whether each pin is configured as an output (1) or an input (0).



Even though the addresses in the Register Watch window are incorrect, we can still use the <u>Memory Watch</u> window to check any addresses we wish (and name them). Select <u>Add</u> and enter the name PortA and the address \$00. Similarly, we can check the address for disabling the watchdog timer (\$FE) and the address for the Reset interrupt (\$FFFE). Also open the <u>Browse</u> <u>Memory</u> window. It is interesting to scroll to the memory address \$8000 and see the machine code for the program that are stored here.

~	1CU			Me	nory W	atch						Co	de View								
	Name Va ACCA \$0 ACCD \$0 ACCD \$0 IX \$0 IX \$0 PC \$3 CCR 0 Modify Up	ilue 0 0000 000 000 000 000 000 000 000 00) Cose		name PortA WatchDo Reset Add Delete	ac \$0 9 \$3 \$F	ddress) 3F FFFF Modify Base	hex \$00 \$00 \$00	value Upd Clo	late se			1: 2: 3: 4: 5: 6: 7: 8: 9: 10: 11: 12:	8000 8002 8004 8007 8009 8009 8008	=00008000 86 04 97 3F 8E 00FF 86 40 97 00 8D 08	* * * Be	File Samp This ogin:	name: le prog progra UDA STAA LDS LDAA STAA STAA BSR	PA6blink.s ram from M will cau \$8000 #\$04 \$3F #\$FF #\$40 \$00 Delay	asm MicroStam use an LE	pll Start. D connect. ;Store pr. ; Disable . ; Check . ;Initiali. ; we wil. ;Write a . ; Check . ;Branch t.
М													13: 14:	800D 800F	86 00 97 00			LDAA STAA	#\$00 \$00		;Write a .
Γ	address 00	01 02 F1 F2	03 04 F3 F4	05 E5	06 0)7 ad	Idress	08 F8	09 F9	QA C)B		15: 16: 17:	8011 8013	8D 02 20 F2			BSR BRA	Delay Loop		;Create a. ;Create a.
	8000 86 8010 00 8020 F8 8030 6C 8040 61 8050 73 8060 2E 8070 72	F1 F2 04 97 8D 02 39 20 69 6E 74 65 65 6D 32 36 20 57	3F 8E 20 F2 45 78 6B 2E 64 20 62 6C 20 42 49 4E	61 62 65 75 33	FF 8 CE F 6D 7 73 7 79 2 72 2 69 6 32 2	77 75 36 80 77 80 70 80 74 80 20 80 20 80 20 80 20 80 20 80	100 110 120 130 140 150 160	40 FF 6C 20 4D 41 64 28	97 18 65 2D 47 53 20 78	00 8 09 1 73 5 20 6 54 4 4D 3 31 3 38 3	BD 18 5C 57 45 31 34 36		17: 18: 19: 20: 21: 22: 23: 24: 25:	8015 8015 8019 801B 801D 801F 8021	18CE FFFF 1809 1808 1809 26 F8 39	De	lay: .:	LDY DEY INY DEY BNE RTS	#\$FFFF Dl		;Subrouti. ;Load the. ;Decremen. ;Incremen. ;Decremen. ;Branch i. ;Return t.
	8080 53	75 6E	20 41	70	72 2	20 80	180	31	35	20 3	31		25:		7777000=			ORG	N 7 7 7 2		:Define t

4) <u>Single Step Through the Simulation.</u> Each time you click on the icon of the person walking (or select <u>Simulator – Simulator Step</u> or press the space bar), the Program Counter (PC) should advance to the address of the next instruction. Note that the Program Counter should begin at \$8000.

A) Before starting:

 $\overline{PC} = \$8000$ (starting address) – corresponding line in Code View is shaded

MCU	Memory Watch	Code View	
Name Value ACCA \$00 ACCB \$000 ACCD \$0000 IX \$0000 IY \$0000 SP \$00FF PC \$8000 CCR 0101000 SXHINZVC	name address hex value PortA \$0 \$10 WatchDog \$3F \$00 Reset \$FFFF \$00 Add Modify Update	1: 2: 3: 4: 5: =00008000 6: 8000 86 04 7: 8002 97 3F 8: 8004 8E 00FF 9: 10: 8007 86 40	 * Filename: PA6blink.asm * Sample program from MicroSt * This program will cause an ORC \$8000 Begin: LDAA #f04 STAA \$3F LDS #FFF Loop: LDAA #540
Modify Update Close	Delete Base Close	11: 8009 97 00 12: 800B 8D 08	STAA \$00 BSR Delay
		13: 800D 86 00	LDAA #\$00
		14: 800F 97 00	STAA \$00
		15: 8011 8D 02	BSR Delay
address 00 01 02 03 04	05 06 07 address 08 09 0A 1)B 16: 8013 20 F2	BRA Loop
7FF0 F0 F1 F2 F3 F4 8000 86 04 97 3F 8E 8010 00 8D 02 20 F2	F5 F6 F7 7FF0 F8 F9 FA 00 FF 86 8000 40 97 00 18 CE FF 8010 FF 18 09	B 17: 3D 18: 8015 18 19: 8015 18CE FFFF	Delay: LDY #\$FFFF

B) After 1 step:

PC = \$8002

Note that ACCA now contains \$04

MCU Memory Watch		Memory Watch	Code View	
	Name Value ACCA \$04 ACCB \$00 ACCD \$1400 IX \$0000	name address hex value PortA \$0 \$00 WatchDog \$3F \$00 Reset \$FFFF \$00	1: 2: 3: 4:	* Filename: PASblink.asm * Sample program from MicroSt * This program will cause an
	IY \$0000 SP \$00FF PC \$8002 CCB 01010000		5: =00008000 6: 8000 86 04 7: 8002 97 3F	ORG \$8000 Begin: LDAA #\$04 STAA \$3F
	SXHINZVC Modify Update Close	Add Modify Update Delete Base Close	8: 8004 8E 00FF 9: 10: 8007 86 40 11: 8009 97 00	LDS #\$FF Loop: LDAA #\$40 STAA \$00
	Memory		12: 800B 8D 08 13: 800D 86 00 14: 800F 97 00	BSR Delay LDAA #\$00 STAA \$00 D22 D
	address 00 01 02 03 04 7FF0 F0 F1 F2 F3 F4 9000 95 04 97 25 95	05 06 07 address 08 09 04 F5 F6 F7 7FF0 F8 F9 FA 00 F5 F6 95 9000 40 97 00	15: 8011 80 02 16: 8013 20 F2 17: 19: 8015	ESR Delay BRA Loop
	8010 00 8D 02 20 F2	18 CE FF 8010 FF 18 09	19: 8015 18CE FFFF	LDY #\$FFFF

C) After 2 steps:

PC = \$8004

Note that WatchDog (address \$3F) now contains \$04 disabling the watchdog timer.

MUU	Memory watch	Code view	
Name Value ACCA \$04 ACCD \$0400 ACCD \$0400 IX \$0000 IY \$0000 SP \$00FF PC \$8004	name address hex value PortA \$0 \$00 WatchDog \$3F \$04 Reset \$FFFF \$00	1: 2: 3: 4: 5: =00008000 6: 8000 86 04	* Filename: PA6blink.asm * Sample program from MicroSt * This program will cause an ORG \$8000 Begin: LDAA \$\$04
CCR 01010000		7: 8002 97 3F 8: 8004 8E 00FF	STAA \$3F LDS #\$FF
Modify Update Close	Add Modify Update Delete Base Close	9: 10: 8007 86 40 11: 8009 97 00 12: 8008 8D 08	Loop: LDAA #\$40 STAA \$00 ESR Delay
Memory		13: 800D 86 00 14: 800F 97 00	LDAA #\$00 STAA \$00
		15: 8011 8D 02	BSR Delay
address 00 01 02 03 04	05 06 07 address 08 09 0A 0E	16: 8013 20 F2	BRA Loop
7FF0 F0 F1 F2 F3 F4 8000 86 04 97 3F 8E 8010 00 8D 02 20 F2	F5 F6 F7 7FF0 F8 F9 FA F8 00 FF 86 8000 40 97 00 80 18 CE FF 8010 FF 18 09 18	17: 18: 8015 19: 8015 18CK FFFF	Delay: LDY #\$FFFF

D) After 3 steps:

PC = \$8004

Note that the Stack Pointer (SP) has been initialized to \$00FF.

MCU	Memory Watch	Code View
Name Value ACCA \$04 ACCB \$00 ACCD \$4000 IX \$0000 IY \$0000 SP \$00FF PC \$8007 CCR 0.101000 SXHINZYC	name address hex value PortA \$0 \$00 WatchDog \$3F \$04 Reset \$FFFF \$00	1: 2: 3: 4: 5: =00 6: 8000 86 7: 8002 97 8: 8004 8E 9:
Modify Update Close	Delete Base Close	10: 8007 86 11: 8009 97 12: 800B 8D
Memory		13: 800D 86 14: 800F 97
address 00 01 02 03 04 7FF0 F0 F1 F2 F3 F4	05 06 07 address 08 09 0A 0B F5 F6 F7 7FF0 F8 F9 FA FB	16: 8013 20 17:
8000 86 04 97 3F 8E 8010 00 8D 02 20 F2	00 FF 86 8000 40 97 00 8D 18 CE FF 8010 FF 18 09 18	18: 8015 19: 8015 180

1.		* Filename: DàGhlink asm
2:		* Sample program from MicroS
3:		 * This program will cause an
4:		
5:	=00008000	ORG \$8000
6:	8000 86 04	Begin: LDAA #\$04
7:	8002 97 3F	STAA \$3F
8:	8004 SE 00FF	LDS #\$FF
9:		
10:	8007 86 40	Loop: LDAA #\$40
11:	8009 97 00	STAA \$00
12:	800B 8D 08	BSR Delay
13:	800D 86 00	LDAA #\$00
14:	800F 97 00	STAA \$00
15:	8011 8D 02	BSR Delay
16:	8013 20 F2	BRA Loop
17:		
18:	8015	Delay:
19:	8015 18CE FFFF	LDY #\$FFFF

E) After 4 steps:

PC = \$8006

Note that ACCA now contains \$40.

MCU	Memory Watch	Code View	
Name Value ACCA \$40 ACCB \$00 ACCD \$4000 IX \$0000 IX \$0000 SP \$00FF PC \$8009 CCR 01010000 SXHINZVC	name address hex value PortA \$0 \$00 WatchDog \$3F \$04 Reset \$FFFF \$00	1: 2: 3: 4: 5: =00008000 6: 8000 86 04 7: 8002 97 3F 8: 8004 8E 00FF 9:	 Filename: PASblink.asm Sample program from MicroSt This program will cause an DRG \$8000 Begin: LDAA \$\$04 STAA \$\$F LDS \$\$FF
Modify Update Close	Delete Base Close	10: 8007 86 40 11: 8009 97 00	Loop: LDAA #\$40 STAA \$00
	(<u>P</u>	12: 800B 8D 08	BSR Delay
Memory		13: 800D 86 00	LDAA #\$UU
address 00 01 02 03 04 7EE0 E0 E1 E2 E3 E4	05 06 07 address 08 09 0A 0E E5 E6 E7 7E0 E8 E9 E4 E6	14: 800F 97 00 15: 8011 8D 02 16: 8013 20 F2 8 17:	BSR Delay BRA Loop
8000 86 04 97 3F 8E 8010 00 8D 02 20 F2	00 FF 86 8000 40 97 00 8E 18 CF FF 8010 FF 18 09 18) 18: 8015 3 19- 8015 18CR FFFF	Delay: LDY #SFFFF

F) After 5 steps:

PC = \$8008

Note that PortA now contains \$40 (or binary 01000000) so PA6 = 1 and the LED turns ON.

100		Code men				
Name Value ACCA \$40 ACCB \$00 ACCD \$4000 IX \$0000 IY \$0000 SP \$00FF PC \$800B CCR 01010000 SXHINZVC Modify Update	name address hex value PortA \$0 \$40 WatchDog \$3F \$04 Reset \$FFFF \$00	1: 2: 3: 4: 5: 6: 7: 8: 9: 9: 10: 11:	=00008000 8000 86 04 8002 97 3F 8004 8E 00FF 8007 86 40 8009 97 00	* Filer * Samp] * This Begin: Loop:	name: PA le program program LDAA STAA LDAA STAA	6blink.asm m from MicroSts will cause an I \$8000 #\$04 \$37 #\$FF #\$40 \$00
		12.	8000 86 00		LDAA -	#<00
Memory		14-	8008 97 00		STAA	#700 \$00
		15:	8011 8D 02		BSR 1	Delav
address 00 01 02 03 04	05 06 07 address 08 09 0A 08	16:	8013 20 F2		BRA	Loop
7FF0 F0 F1 F2 F3 F4	F5 F6 F7 7FF0 F8 F9 FA FB	17:				-
8000 86 04 97 3F 8E	00 FF 86 8000 40 97 00 8D	18:	8015	Delay:		
8010 00 8D 02 20 F2	18 CE FF 8010 FF 18 09 18	19:	8015 18CE FFFF		LDY	#\$FFFF

G) After 6 steps:

 $\overline{PC} = \$8015 -$ so the program has branched to the delay subroutine

MCU	Memory Watch	Code View	
Name Value ACCA \$40 ACCB \$00 ACCD \$4000 K \$0000 N \$0000 SP \$00FD PC \$8015 CCR 0.10.1.0.0.0 SXHINZYC Modify	name address hex value PortA \$0 \$40 WatchDog \$3F \$04 Reset \$FFFF \$00 Add Modify Update Delete Base Close	1: 2: 3: 4: 5: =00008000 6: 8000 86 04 7: 8002 97 3F 8: 8004 8E 00FF 9: 10: 8007 86 40 11: 8009 87 00 12: 8008 8D 08	 * Filename: PA6blink.asm * Sample program from MicroSt. * This program will cause an : ORG \$8000 Begin: LDAA #\$04 STAA \$37 LDS #\$FF Loop: LDAA #\$40 STAA \$00 BSR Delay
Memory address 00 01 02 03 04 7FF0 F0 F1 F2 F3 F4 8000 86 04 97 3F 8E	05 06 07 address 08 09 0A 0E 75 F6 F7 7FF0 F8 F9 FA FE 00 FF 86 8000 40 97 00 8E	13: 800D 86 00 14: 800F 97 00 15: 8011 8D 02 16: 8013 20 F2 17: 18:	LDAA #\$00 STAA \$00 BSR Delay BPA Loop Delay:
8010 00 8D 02 20 F2	18 CE FF 8010 FF 18 09 18	19: 8015 18CE FFFF	LDY #\$FFFF

H) After 7 steps:

PC = \$8019

Note that IY (Index Register Y) now contains FFFF

MCU	Memory Watch	Code View	
Name Value ACCA \$40 ACCB \$00 ACCD \$4000 IX \$5000 IX \$5000 IY \$5000 PC \$8019 CCR 01011000 SXHINZYC	name address hex value PortA \$0 \$40 WatchDog \$3F \$04 Reset \$FFFF \$00	1: * Filename: PASDini 2: * Sample program from 3: * This program will of 4: * 5: =00008000 6: 8000 86 04 7: 8002 97 3F 8: 8004 8E 00FF LDS #\$FF	k.asm m MicroSts cause an I
Memory	Add Modify Update Delete Base Close	9: 10: 8007 86 40 Loop: LDAA #\$40 11: 8009 97 00 STAA \$00 12: 800B 8D 08 BS Pelay 13: 800D 96 00 LDAA #\$00 14: 800F 97 00 STAA \$00	
address 00 01 02 03 04 7FF0 F0 F1 F2 F3 F4 8000 86 04 97 3F 8E 8010 00 8D 02 20 F2 8020 F8 39 20 45 78	05 06 07 address 08 09 04 0 F5 F6 F7 7FF0 F8 F9 FA F1 00 FF 86 8000 40 97 00 8 18 CE FF 8010 FF 18 09 1 11 6D 6D 6D 6D 6C 6C 7.3 5	15: 8011 8D 02 BSR Delay 3 16: 8013 20 F2 BRA Loop 3 17: 18: 8015 Delay: 19: 8015 18CE FFFF LDY #\$FFF LDY #\$FFF	F
8030 6C 69 6E 68 6E 68 22 8040 61 74 65 64 20 8050 73 65 60 62 62 8060 2E 32 36 20 42 8070 72 20 57 49 42 8080 53 75 6E 20 41	6C 73 74 8030 20 20 60 70 66 79 20 8040 4D 47 54 41 53 4D 33 75 69 6C 806 64 20 31 33 32 20 8060 64 20 31 35 22 88 3 70 72 20 8060 31 35 20 30 31 35 20 30 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 31 35 20 33 32 20 308 31 35 20 31 35 20 33 32 30 30 31	201 6012 1005 D11 DE1 7 21: 801B 1809 DEY 5 22: 801D 1809 DEY 4 23: 801F 26 F8 ENE D1 4 24: 8021 39 RTS 25: 2 25: =0000FFFE 0EG SFFFE	

We have a problem! IY is decremented each time through the delay loop until it equals 0. Its initial value is \$FFFF so we have to go through the loop 65,535 times! That is a lot of single stepping! We need a better way – by adding **BreakPoint**. If we specify a breakpoint (address) and click on the stoplight icon, the simulation will run up to that point.

I) Add a BreakPoint of \$8011 and click the traffic light:

\$8011 is the point where the delay subroutine is called the second time, so we have loaded a new value into PortA.

PortA now contains 00 (= 00000000 in binary) so PA6 = 0 and the LED is OFF.

Break Point 8011 Keypad Pin Sc	ope					
MCU Memory Wat	ch Cod	le View				
Name Value ACCA \$00 ACCB \$00 ACCD \$0000 IX \$0000 IY \$00000 SP \$00FF PC \$8011 CCR 0.10.1.0.1.0.0 SX H IN Z V C Add	address hex value \$0 \$00 \$3F \$04 \$FFFF \$00 Modify Undate	1: 2: 3: 4: 5: 6: 4 7: 4 8: 4	=00008000 8000 86 04 8002 97 3F 8004 88 00FF	* Filer * Samp * This Begin:	name: P le progr program ORG LDAA STAA LDS	A6blink.asm am from MicroSt vill cause an \$8000 \$\$04 \$3F \$\$FF
Modify Update Close Delete	Base Close	9: 10: : 11: : 12: : 13: :	8007 86 40 8009 97 00 8008 8D 08 800D 86 00	Loop:	LDAA STAA BSR LDAA	#\$40 \$00 Delay #\$00
Memory		14: :	300 F 97 00		STAA	\$00
address 00 01 02 02 04 05 06 07		15: 1	3011 8D 02 3013 20 82		BSR	Delay
autress ou or oz oz </td <td>address 06 09 04 08 7FF0 F8 F9 FA FB 8000 40 97 00 8D 8010 FF 18 09 18</td> <td>10: 1 17: 18: 1 19: 1</td> <td>3015 20 72 3015 3015 18CE FFFF</td> <td>Delay:</td> <td>LDY</td> <td>#\$FFFF</td>	address 06 09 04 08 7FF0 F8 F9 FA FB 8000 40 97 00 8D 8010 FF 18 09 18	10: 1 17: 18: 1 19: 1	3015 20 72 3015 3015 18C E FFFF	Delay:	LDY	#\$FFFF

J) Add a BreakPoint of \$800B and click the traffic light:

\$800B is the point where the delay subroutine is called the first time, so we have loaded a new value into PortA.

PortA now contains \$40 (=01000000 in binary) so PA6 = 1 and the LED is ON.

Break 800B Keypad	Pin Scope					
MCU Men	nory Watch	Code View				
Name Value r ACCA \$40	name address hex value PortA \$0 \$40	1:		* File	name: I	PA6blink.asm
ACCB \$00 V ACCD \$4000 R IX \$0000	WatchDog \$3F \$U4 Reset \$FFFF \$00	2: 3: 4:		* Samp * This	le program program	ram from MicroSt n will cause an
IY \$0000 SP \$00FF		5:	=00008000		ORG	\$8000
PC \$800B		6: 7:	8000 86 04 8002 97 3F	Begin:	LDAA STAA	#\$04 \$3F
CCR 0101 000 0 SXHINZVC	Add Modify Update	8: 9:	8004 SE 00FF		LDS	#\$FF
Modify Update Close	Delete Base Close	10:	8007 86 40	Loop:	LDAA	#\$40
		11:	8009 97 00		STAA	\$00 Dalar
<u>د ا</u>		12:	800B 80 08		LDAA	Delay #400
Memory		14-	8007 97 00		STAA	#700 \$00
		15:	8011 8D 02		BSR	Delay
address 00 01 02 03 04 05	06 07 address 08 09 0A 08	16:	8013 20 F2		BRA	Loop
7FF0 F0 F1 F2 F3 F4 F5	F6 F7 7FF0 F8 F9 FA FB	17:				
8000 86 04 97 3F 8E 00	FF 86 8000 40 97 00 8D	18:	8015	Delay:		
N 8010 00 8D 02 20 F2 18	CE FF 8010 FF 18 09 18	19:	8015 18CE FFFF		LDY	#\$FFFF

K) <u>Repeat steps I and J indefinitely to simulate the LED turning OFF and ON</u>

Note that this program contains an infinite loop, so the LED will blink ON and OFF indefinitely. In other words, the simulation never ends! However, we have found out enough to be sure that it will work correctly when we download it into the MicroStamp11.

L) <u>Reset Vector</u>

Recall that the last two lines of the program (ORG \$FFFF and FDB Begin) are used to specify the address of where the program is to be redirected if the reset button is pressed on the MicroStamp11 board. Note that these lines are never reached by single-stepping or by starting/stopping the simulation. The commands are assembler directives and they are executed before the program ever runs. Also note that the Memory Watch named Reset was poorly defined earlier as address \$FFFF which is only the lower byte of the reset vector. It was replaced here using ResetUB (upper byte) and ResetLB (lower byte) and we can see that it contains the address \$8000 which corresponds to the label Begin at the start of the program.

Point Point	Pode View	
MCU Memory Watch		
Name Value ACCA \$40 ACCB \$00 ACCD \$4000 ACCD \$4000 ACCD \$4000 IX \$00000 SP \$00FF PC \$\$00B CCR \$10.10.00.0	1: 2: 3: 4: 5: 8000 36 04 7: 8002 97 3F 8: 8004 8E 00FF	 Filename: PA6blink.asm Sample program from MicroSt This program will cause an ORG \$8000 Begin: LDAA \$\$04 STAA \$37 LDS \$\$FF
SXHINZVC Add Modify Update Modify Update Close Close	9: 10: 8007 86 40 11: 8009 97 00 12: 800B 8D 08 13: 800D 86 00 14: 900F 97 00	Loop: LDAA #\$40 STAA \$00 BSR Delay LDAA #\$00 STAA \$00
Memory	15: 8011 8D 02	BSR Delav
	16: 8013 20 F2	BRA Loop
address 00 01 02 03 04 05 06 07 address 08 09 0A 08 0C 0D 0E 0F FF60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	17: 18: 8015 19: 8015 18CE FFFF 20: 8019 1809	Delay: LDY #\$FFFF D1: DEY
FF30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0	21: 801B 1808 22: 801D 1809 23: 801F 26 F8 24: 8021 39	INY DEY BNE D1 RTS
FFE0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0	25: 26: =0000FFFE 27: FFFE 8000	ORG \$FFFE FDB Begin