Micro-IDE

Integrated Development Environment

6811 Simulator

Quick Start Guide

Document Revision: 1.03 Date: 30 May 2004



16301 Blue Ridge Road, Missouri City, Texas 77489 USA Telephone: 1-713-283-990 Fax: 1-281-416-2806 E-mail: info@bipom.com Web: www.bipom.com

© 2001 by BiPOM Electronics, Inc. All rights reserved. MM51E WEB server Quick Start Guide. No part of this work may be reproduced in any manner without written permission of BiPOM Electronics.

All trademarked names in this manual are the property of respective owners.

1. Introduction

The 6811 Simulator simulates the operation of the Motorola 68HC11 microcontroller with up to 64K program memory and 1024 bytes internal data memory.

Powerful, yet easy to use and affordable 68HC11 simulator for Micro-IDE, which is a Windows-based Integrated Development Environment for micro-controllers, the 68HC11 Simulator simplifies code development with Micro C and Micro-IDE. Errors in user programs can be found and fixed quickly in simulation mode by avoiding time consuming downloads to the target board.

68HC11 Simulator is part of the 68HC11 Development System that can be downloaded from http://www.bipom.com/6811dev.htm

68HC11 Simulator supports both assembly and C source level simulation (using Micro C compiler).

2. Preparation for simulation

Some example projects that come with Micro-IDE have settings to enable the simulator.

New projects are also created with the simulator/debugger options enabled. For every Micro-IDE project you can specify individual simulator/debugger settings using "Settings" command under "Project" menu. Simulator/debugger options are on the debugger tab:

Project S	ettings							×
General	Debugger	Environment	Compiler	Linker	Output	Pre-Build	Post-Build	
	ulate Interrup	tsi						
🔲 Sim	ulate Timers							
🗖 Sim	ulate Serial P	orts						
🖵 Use	Hardware Po	orts in Simulatio	n					
		OK		Cancel		Apply	Help	

Default simulator/debugger settings can be set using Toolkit Configurator (which can be started from Programs->Micro-IDE->Toolkit Configurator under the Start menu):

🏂 Toolkit Configurator 🛛 🔀
Toolkits Debuggers Loaders
Debugger Name : 6811 Simulator
Debugger Type : Simulator
Debugger DLL : sim6811
Target Board :
Set as Default Debugger/Simulator
Generic Debugger Options
Simulate Interrupts
Simulate Timers
Simulate Serial Ports
Use hardware ports in simulation
OK Cancel Apply Help

If you have older projects that do not have the simulator/debugger options, you can enable simulator/debugger support by adding command line options. These options are tool-specific.

For example, for Micro C compiler, use the following:

Under Project->Settings:

- On the compiler command line, add -C -S -L
- On the Output Generator command line, add -F -S

These settings apply to Micro C compiler only. These settings will generate a listing file (.LST) which contains the information needed for the simulator/debugger.

Press "Build All" icon button on toolbar of Micro-IDE and check Output\Build window.



3. Starting and Stopping the 6811 simulator

To start simulator, use "Go" button (F5), "Step into" button (F11), "Step Over" button (F10) or "Step out" button. To stop the simulator, you can use "Stop Debugging" button (Shift+F5).

4. 6811 simulator windows

Simulator uses several windows to display the debug information: Source Window, Output Window, Registers Window, Memory Window, Variables Window and Terminal Window.

4.1. Source Window

Source window is the window where you can create and edit your programs. Source window is also used by the simulator to set/remove/display breakpoints and current execution line.



In debugging mode, the line that is about to be executed is colored green and contains a small yellow arrow on the left side.

You can toggle a breakpoint using F9 key. The line with an active breakpoint is colored red.

When debugging, you can view a variable value by simply placing mouse cursor over variable name.

4.2. Output Window

Output Window displays messages about the current debugging session. To view debug messages, select the Debug tab on the Output Window. Any errors and information messages encountered during debugging are displayed here.



4.3. Registers Window

Register window displays the contents of the registers of the processor being debugged.

Registers —			- 🔺 🗡
Name	Value	Extended	
A	0C		
В	42		
D	0C42		
X	16		
Y	00		
SP	03FA	8042	
PC	8029	39	
CCR	D4	SX_I_Z	
TCNT	3866		
TMSK1	80		
TMSK2	CO		
TFLG1	80		
TFLG2	00		
PACTL	00		
TOC1	3851		
TOC2	00	N	
TOC3	42A3	1	
TOC4	00	-	
TOC5	00		
BAUD	30		
SCCR1	00		
SCCR2	00		
SCSR	CO		
SCDR	00		_

Register Window is updated as you single-step through your program. If your program is running continuously (for example, using the Go command), then the Register Window is updated when the simulated program stops due to a breakpoint or other cause.

You can edit register contents in this window by simply typing new values in the window cells.

4.4. Memory Window

Memory window displays the contents of the memory of the target processor. Memory Window is updated as you single-step through your program. If your program is running continuously (for example, using the Go command), then the Memory Window is updated when the simulated program stops due to a breakpoint or other cause.

Memory										Ľ
Memory			•	Add	ess:	0×0	0000	0000		
0000000	00	00	48	30	00	00	00	00	H0	
00000008	00	00	00	00	00	00	00	00		
00000010	01	60	00	00	00	12	00	00	.`	
00000018	00	00	00	00	53	00	00	00	s	
00000020	00	00	00	00	00	00	00	00		
00000028	00	00	00	00	00	00	00	00		
00000030	00	00	55	00	00	00	00	00	U	
00000038	00	00	00	00	00	00	00	00		
00000040	00	00	00	00	00	00	00	00		
00000048	00	00	00	33	00	00	00	00	3	
00000050	00	00	00	00	00	00	00	00		
00000058	00	00	00	00	00	00	00	00		
00000060	00	00	00	00	00	00	00	00		
00000068	00	00	ЗF	FF	25	11	23	33	?.%.#3	
00000070	00	00	00	00	00	00	00	00		
00000078	00	00	00	00	00	00	00	00		
00000080	00	00	00	00	00	00	φ	00		
00000088	00	00	00	00	00	00	¢	00		
00000090	00	OD	33	ЗD	00	00	00	00	3=	
00000098	00	00	00	00	00	00	00	00		
000000A0	00	00	00	00	00	00	00	00		
000000A8	00	00	00	00	00	00	00	00		-

You can edit memory contents in this window by simply typing new values in the window cells.

4.5. Variables Window

Variables window displays contents of the global and local variables during debugging.

<u>×</u>	Name	Values	Address	Memory Type	
	 Local Variables 			N	$\overline{\boldsymbol{\zeta}}$
	– Key	0x80	0x3fe	Memory 너국	
	 Global Variables 				
	- Mode	0x0	0x102		\leq
SS L					
iable					
Var	•				\sim
Rea	ady				

You can edit variables in this window by simply typing new values in the window cells.

Right-click on a variable and click "Extend" to edit variable in binary mode.

Right-click in this window and click "Hexadecimal display" to view variable contents in decimal or hexadecimal format.

4.6. Terminal Window

Terminal window is used for exchanging information with RS232 serial port of the simulated processor. Terminal window displays information that is sent to the serial port (for example, using C functions such as **printf**). Terminal window also accepts information from the keyboard and sends this information to the serial port of the simulated processor (for example, using C functions such as **getc** or **getch**).



5. Examples for use Smi6811 Simulator.

With these small examples you can quickly get acquainted with the Sim6811 simulator and ways to use it for the program execution tracing, timer system tracing and interrupts.

5.1. Run Loader/Buffalo.

Sim6811 not only simulates your projects but executes everything as real MINI-MAX/11-A board. Because the real board has pre-programmed BUFFALO and LOADER programs the simulator downloads and executes both these programs too. You can start these pre-programmed programs using the RunBuff program.

5.1.1. Step 1 – Open the project.

Open the project "RunBuff .prj" using "Open Project" command under "Project" menu.



5.1.2. Step 2 – Build the project.

Press "Build All" icon button on toolbar of Micro-IDE and check Output\Build window.

I Micro-IDE - [runBuff]		
<u>File E</u> dit <u>V</u> iew <u>B</u> uild <u>P</u> roject	<u>D</u> ebug <u>T</u> ools <u>W</u> indow <u>H</u> elp	<u>_ 8 ×</u>
D 🚅 🖬 🕼 🐇 🖬 🛍 🖆	2 🗠 🎒 🕸 🎬 🗈 🕒 😰 🗛 🔏	¥ 🙀
	EL EL EK 🖱 Buidair († 1945 🕷 🗣) 🐩 🖆 💻 🗖
Workspace	;	A
unBuff' Project Files	ORG R_VEC	;Position to reset vector
🛄 runBuff.asm	FDB BEGIN	;Set start address
	ORG \$8000	;Our code starts here
	BEGIN	
	LDAA #\$FE	; clear LSB
	STAA \$100A	; clear PE.O
	JMP \$FE60	
	; NOTE.	
	; To run Loader/Buffalo	on the real Mini-Max/11-A board
	; to put a jumper on th	e J5 connector.
	END	
■t ² : Files		
		<u> </u>
Assembling C:\bipom\devtools\Micr	pC\Examples\6811\BunBuffalo\runBuff.asm	1
First pass Opt Second pass 0	error(s).	
		F
Build Debug Find in Files 1 Fi	nd in Files 2 Loader	
	<u></u>	
Build all the files		Ln 1, Col 1

5.2.3. Step 3 – Start simulator.

Open 'RunBuff.lst' file and trace step by step executing the program using "Open ..." command under "File" menu.

Hicro-IDE - [Runbuff.lst]		
<u>File Edit View Build Project De</u>	bug <u>T</u> ools <u>W</u> indow <u>H</u> elp	_ <u>_</u>
] D 😅 🖬 🕼 🖇 🛍 😫 🖄	a 😂 🕸 🛍 🗈 🕒 🔽 🗛 🖧 🙀	
	u 🗈 🗱 🖑 (4) (7) (7) (7) (7) (7) (7) (7) (7) (7) (7	≞ □
Workspace 🛋 🗙	12 ;	A
- G 'runBuff' Pr	13 ORG R_VEC	;Position to reset vector
I runBuff.a⊧	14 FDB BEGIN	;Set start address
	15 ;	
	16 ORG \$8000	;Our code starts here
	17 BEGIN	
	18 LDAA #\$FE	; clear LSB
	19 STAA \$100A	; clear PE.0
	20 JMP \$FE60	
	21 ; NOTE.	
	22 ; To run Loader/Buffalo on t	he real Mini-Max/11-A boa
	23 ; to remove a jumper from J5	connector.
	24	
	25	*
×		
Interrupts, timers and serial port are simu	ulated.	
Please use the Toolkit Configurator to c	hange these settings if needed. Note that simulation of interrupts, tim	ers and/or serial port may slow down simulation.
Chip Type: 6811 Manufacturer: Generic		
Connecting terminal to debugger simula	tion	_
¥		
Build Debug Find in Files 1 Find i	n Files 2 Loader	
Ready		Ln 1, Col 1

Start the simulating process by clicking "Step Into" command under "Debug" menu (or F11) a few times. Please note, simulator should step through some startup code (a few instructions of loader/buffalo) on the entry. Then, one of the lines will be colored green. Simulator is stopped at this line. It will be executed when you press one of the "Step Over", "Step Into", etc. buttons.



Open the Terminal window, using "Terminal" command under "View" menu.

III Micro-IDE - [Runbuff.lst]	
<u> </u>	
D 😅 🖬 🕼 X ha 🖻 🛆 🗠 🎒 👘 🔶 😨	🗛 😤 🙀
00000	
Workspace — X ;	Terminal
GRG R_VEC	A
FDB BEGIN	
,	
BEGIN	
DDJIII → LDAA #\$FE	
STAA \$100A	
JMP \$FE60	
; NOTE.	
; To run Loader/Buffalo	
; to remove a jumper ir	
Files	<u>×</u>
×	× 1
Interrupts, timers and serial port are simulated.	
Please use the Toolkit Configurator to change these settings if needed. Note th	at simulation of interrupts, timers and/or serial port may slow down simulatio
Chip Type: 6811 Manufacturer: Generic	
Connecting terminal to debugger sinulation	Ľ
Build Dahua Find in Files 1 Find in Files 2 Loader	
	2
Show or hide Terminal	Ln 20, Col 1

The first two lines after BEGIN (LDAA #\$FE and STAA \$100A) are used to clear PORTE.0. This corresponds putting a jumper on J5 (1-3) connector on real board. The next line (JMP \$FE60) gives control to Loader/Buffalo.

III Micro-IDE - [Runbuff.lst]	_ 8 ×
<u> </u>	<u> - 8 ×</u>
D 😅 🖬 🕼 🕺 🗠 🚔 🗇 🕮 ■D 🔶 😰 AA 🖓 🖓	
Workspace IN ORG \$8000	
EGIN BEGIN	A
IDAA #\$FE *MINI-MAX/11-A*	
STAA \$100A CHECKSUM [16EA] OK	
STEDU	
; NOTE. BUFFALO 3.4 (ext) - Bit User Fast	
to remove a jumper fr	
, co remove a jamper if	
f	
	7
Interrupts, timers and serial port are simulated.	
Please use the Toolkit Configurator to change these settings if needed. Note that simulation of interrupts, timers and/or serial port may slow down simulation	°-11
Chip Type: 6811 Manufacturer: Generic	
Lonnecting terminal to debugger simulation	٦IJ
	Me
Step into next statement In 22, Col 1	

Now you can send any command to Buffalo. For example, you can write '?' (HELP) and confirm with <ENTER> button in Terminal Window.



NOTE: Because Loader and Buffalo are not included in your project, you can't trace both them step-by-step executing.

5.2. Counter.

The example increments a variable in a loop. You will learn how to watch variable increment using Micro-IDE windows.

5.2.1. Step 1 – Open the project.

Open the project "Counter.prj", using "Open Project" command under "Project" menu.



5.2.2. Step 2 – Build the project.

Press "Build All " icon button on toolbar of Micro-IDE and check Output\Build window.



5.2.3. Step 3 – Start simulator.

Start the simulating process by clicking "Step Into" command under "Debug" menu (or F11). One of the lines will be colored green. This is the line the simulator stopped at. It will be executed when you press one of the "Step Over", "Step Into", etc. buttons.

5.2.4. Step 4 – Watch variable.

Open Variables Window using "Variables" command in "View" menu. In Variables Window you will see one global variable "count". Run the program by clicking "Step Into" command (or F11) and you will see how "count" variable value is changed.

Micro-IDE - [Counter.c]		
<u>File Edit View Build Project Debu</u>	ug <u>T</u> ools <u>W</u> indow <u>H</u> elp	_ 8 ×
🗅 😅 🖬 🕼 🗴 🖬 🛍 🕰 🛇	2 🎒 🍪 🛍 🗈 🕘 🔽 🗛 🖧 锅 📙 🖪 🖂 🐺	R 🔜 🖂
🖫 🙀 😓 🕷 😭 🔜 🗖 📔	1 🗄 🕱 🐵 (P) (P) (P)	
<pre>#include <6811io.h></pre>		4
<pre>#include <6811reg.h></pre>	•	
unsigned char count	;;	
void main()		
{		
count=0;	// Initialize counter	
while(1)	// Loop forever	
{		
count++;	// Increment counter	
}		
}		
		7
Name Values	Address Memory Type	
Local Variables		
Global Variables	0.100	
Count 2		
able		
Aari		•
Ready	Ln 9, Col 1	

Please note that "Address" column contains "0x0102" value. It is the address of the variable in memory. On the next steps you will see that you can watch the variable in the memory window directly.

5.2.5. Step 5 – Change variable value.

You can change value of the variable editing value in Values column. You can also view and change bits of the variable value. Double-click on the variable row in the Variables Window or select "Extend" command from context menu. A popup window will appear:

III Micro-IDE - [Counter.c]	_ 🗆 ×
🚍 <u>F</u> ile <u>E</u> dit <u>V</u> iew <u>B</u> uild <u>P</u> roject <u>D</u> ebug <u>T</u> ools <u>W</u> indow <u>H</u> elp	_ 티 ×
D 😅 🖬 🕼 X 🖻 🖻 2 🗢 🎒 🏶 🕮 🗈 🔶 😰 M X X 🐂 E 🔊	
] 💱 🏕 🗶 🖫 🔚 🔲 📄 🖼 🖓 🖓 🚱 🚱 🔂	
<pre>#include <6811io.h></pre>	*
<pre>#include <6811reg.h></pre>	
unsigned char count;	
void main()	
count=0; // Initialize counter	
while(1) // Loop forever	
{	
count++; // Increment counter	
}	
}	
	_
a	
Variable: count	
Name Values Add HEX: 2 DEC: 2 Cancel 00K	
Local Variables	
Global Variables	
count 2 0x102	
ples	
	F
Readu	

You can edit value in binary, HEX or DEC here.

5.2.6. Step 6 – Variable address in .LST file.

Open the listing file "Counter.Ist" using "Open ..." command under "File" menu.

Open		? ×
Look jn: 🔂	Counter 💽 🖻 💋	-
Counter.lst		
File <u>n</u> ame:	Counter	<u>O</u> pen
Files of <u>t</u> ype:	Listing Files (*.lst)	Cancel

In the end of the listing file find 'SYMBOL TABLE :'. Here you can see address in RAM of the 'count' variable (for this example, 0102). It is the location Micro-C placed this variable to.

I Micro-IDE	- [Counter.	lst]							_ 🗆 ×
<u> </u>	<u>⊻</u> iew <u>B</u> uild	l <u>P</u> roject <u>D</u> ebug	g <u>T</u> ools <u>W</u> indow	<u>H</u> elp					_ 8 ×
🗅 🍰 日	🕼 🐰 🛙	6 6 2 2	🛛 🕹 🗳 🖽 🗉	0 🔴 😨	M & 7	s 🐂 🖪 .	🔉 🗉 🖾 🧔	FA 🔜 🖂	
🙀 🛛	9 🕷 🖻		🗈 🕱 🖉 🗄	ት ት ት	12				
0102			139	* and	want all	l variabl	es alloca	ated in i	Intern
0102			140	×					
0102			141	* OF	G \$4000	Ex	ternal RA	AM begins	here
0102			142	*10432					
0102			143	count	RMB 1				
0103			144	*					
0103			145	* DDS	MICRO-C	6811 Run	time lib:	rary - Su	ıffix
0103			146	×					
0103			147	?heap	EQU *	He	ap memor	y goes he	ere
0103			148	* For	stand-al	lone ROM	system, 1	uncomment	t the
FFFE			149	OF	G SFFFE	Po	sition to	o reset v	vector
FFFE	80 00		150	FL)B ?begin	n Se	t start a	address	
0000			151	*10432					
DUNF:	IELD 68	11 ASSEME	BLER: Coun	ter					PAGE:
SYMBO	L TABLE	:							
		_							
?AB1	-802	2 ?AB2	-802C	?BEGIN	1 -8000	?EQ	-802D	?EXIT1	-801
?HEAP	-010	3 ?NE	-8035	?PARM	-00FE	?REGS	-1000	?RETO	-803
?RET1	-803	A ?TEME	-0100	COUNT	-0102	EXIT	-8013	MAIN	-801
NARGS	-803	4							_
•									
Ready							Ln 1, Col 1		

Close the Listing File, using the <Close> button. You will see C-source view again.

5.2.7. Step 7 – Variable in the Memory Window

Open the Memory Window, using "Memory" command under "View" menu.

III Micro-IDE - [Counter.c]	_ 🗆 ×
<u>File Edit View Build Project D</u> ebug Iools <u>Wi</u> ndow <u>H</u> elp	_ 🗉 🗵
] D 😅 🖬 🕼 🕹 😂 😂 😂 📾 🗈 🕒 🕼 🖓 🖓 🔚 📼 🖾 🖗 🛛 🖼 🖓	
9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
#include <6811io.h>	
<pre>#include <6811reg.h></pre>	
unsigned char count;	
void main()	
count=0: // Initialize counter	
while (1) // Loop forever	
Countil: // Ingroment counter	
Counter, // increment counter	
3	
	-
	Þ
Memory Address UX0000000	
	<u> </u>
00000020 00 00 00 00 00 00 00 00	
0000028 00 00 00 00 00 00 00 00	
	-
Ready Ln 11. Col 2	
· · · · · · · · · · · · · · · · · · ·	,,

In field "Address:" enter 0x00000100 value and press Enter. This will scroll Memory Window to address 0x0100.

×	Memory Address: 0x00000100														
T	000000F0	00	00	00	00	00	00	00	00						
	00000058	00	00	00	00	00	00	00	00						
	00000100	00	00	06	00	00	00	00	00						
	00000108	00	00	00	00	00	00	00	00						
	00000110	00	00	00	00	00	00	00	00						
	00000118	00	00	00	00	00	00	00	00						
	00000120	00	00	00	00	00	00	00	00						
÷.	00000128	00	00	00	00	00	00	00	00						
Đ.	00000130	00	00	00	00	00	00	00	00						
Me															
Rea	ady														

Now you can see 0102 memory location which is address of the "counter" variable. Use F11 button to step through the program and you will see that value at 0102 address is changed.

Micro-IDE - [Counter.c]	- 🗆 ×
	_ 8 ×
] D 😅 🖬 🖉 X № € 2 C 😂 🕸 🛍 🕪 🕒 🔽 🛛 🗍 🗛 X X 👫] 🖪 🗖 🖾 🖓 👰 🗮 🗉 👘	
) 🖫 疑 🤧 🕼 🔳 📕 🖽 🕙 🔁 🕀 🕀	
<pre>#include <6811io.h></pre>	4
<pre>#include <6811reg.h></pre>	
unsigned char count;	
void main()	
count=0; // Initialize counter	
while(1) // Loop forever	
count++; // Increment counter	
	V
<u> </u>	►
Memory Address: 0x0000100	
000000000 00 00 00 00 00 00 00 00	
00000118 00 00 00 00 00 00 00 00	
00000120 00 00 00 00 00 00 00 00	
	-
neauy [Lh 3, Col 1]]	

Also, you can change the 'count' variable value directly in memory. For this purpose, you can mark and enter new value (for example 45).

×	Memory			•	<u>A</u> ddr	ess:	0×0			
T	000000 F 0	00	00	00	00	00	00	00	00	
	000000 F 8	00	00	00	00	00	00	00	00	
	00000100	00	00	45	00	00	00	00	00	E
	00000108	00	00	00	00	00	00	00	00	
	00000110	00	00	00	00	00	00	00	00	
	00000118	00	00	00	00	00	00	00	00	
	00000120	00	00	00	00	00	00	00	00	
2	00000128	00	00	00	00	00	00	00	00	
loui	00000130	00	00	00	00	00	00	00	00	
Ξ										
Rea	ady									

Of course, you can continue program executing, after changing the variable value in memory, use the F11 button.

×	Memory			•	Add	ess:	0×0	0000	0100	
T	000000 F 0	00	00	00	00	00	00	00	00	
	00000058	00	00	00	00	00	00	00	00	
	00000100	00	00	46	00	00	00	00	00	F
	00000108	00	00	00	00	00	00	00	00	
	00000110	00	00	00	00	00	00	00	00	
	00000118	00	00	00	00	00	00	00	00	
	00000120	00	00	00	00	00	00	00	00	
ž.	00000128	00	00	00	00	00	00	00	00	
Ū	00000130	00	00	00	00	00	00	00	00	
Ξe										
Rea	ady									

5.2.8. Step 8 – Disassembly.

In this step we'll trace how CPU changes the variable value using the registers. First, you have to open disassembly window by clicking "Disassembly" command under "View" menu.

In this window, you can see and trace the program in the assembly and code view. After that, you have to open registers window, using "Registers" command under "View" menu.

_ Ele Edi View Build Project Debug Iools Window Help □ 22 日 27 日 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
▋▋▆▋▋▕▖▆▋∠∠ॖॖॖॖॖॖॖॎऄॖॕॿ▋●▋】ॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖॖ
] 21 22 22 20 20 20 20 20 20 20 20 20 20 20
801F F7 01 02 57 STAB count
8022 58 *7: while(1) Name Value Extend
DDUNFIELD 6811 ASSEMBLER: Counter
8022 59 ?AB1 EQU *
8022 60 *8: {
8022 61 *9: count++; PC 8022 F6
➡ 8022 F6 01 02 62 LDAB count CCR 00 SXL.
8025 5C 63 INCB
8026 F7 01 02 64 STAB count TMSK2 00
8029 65 *10: }
8029 7E 80 22 66 JMP 7AB1 PACTL 00
Memory Address: 0x0000100
000000F0 00 00 00 00 00 00 00 00
000000F8 00 00 00 00 00 00 00
00000118 00 00 00 00 00 00 00 00
00000120 00 00 00 00 00 00 00 00
Readu In SE Col 1

In disassembly view you can see how CPU loads the accumulator 'B' from 'count' ("LDAB count" instruction), increments the accumulator 'B' ("INCB" instruction) and stores the accumulator 'B' in 'count' ("STAB count" instruction). You can step through these instructions using F11 button.

In Registers Window you also can change the accumulator 'B' value (for example, to 72). To do it you have to click on the Value column, enter new value, and press 'ENTER' button.

(NOTE: you have to change accumulator 'B' value after "INCB" instruction and before "STAB count" instruction.)

Micro-IDE - [Counter.Ist]	
Eile Edit ⊻iew Build Project Debug Iools Window Help	_ 뭔 ×
🗅 🚅 🖬 🕼 🕴 🗠 🗠 🎒 🦃 🛍 📭 🕒 😨 🛛 🗛 🖓 🙀 🖪 🗖 🗖	I 🐺 🔂 🔜 🖂
9 9 9 9 🖉 📃 🔲 🖹 🐨 🗶 9 9 9 9	
801F F7 01 02 57 STAB count	Registers
8022 58 *7: while(1)	Name Value Extend.
DDUNFIELD 6811 ASSEMBLER: Counter	B 72
	D 0C72
8022 59 ?AB1 EQU *	
8022 60 *8: {	SP 03FD 8013
8022 61 *9: count++;	PC 8026 F7
8022 F6 01 02 62 LDAB count	TCNT 011F
8025 5C 63 INCB	TMSK1 00
₽8026 F7 01 02 64 STAB count	TFLG1 00
8029 65 *10: }	TFLG2 00
8029 7E 80 22 66 JMP ?AB1	TOC1 FFFF
	<u> </u>
00000100 00 00 46 00 00 00 00 00	
00000108 00 00 00 00 00 00 00	
00000110 00 00 00 00 00 00 00	
00000118 00 00 00 00 00 00 00	
	-
Ready In 68	,Col 1

If you want to see accumulator 'B' (or other register) contents in binary mode, you can click the right mouse button, after positioning the cursor over the accumulator (or other register) name (for example, 'B').

Hicro-IDE - [Counter.lst]	<u> </u>
📰 Eile Edit View Build Project Debug Tools Window Help	_ & ×
D 😅 🖬 🕼 3 🖻 🖻 🛆 🗠 🎒 🧇 🕮 🗈 🜒 🛛 🗛 🕉 🗞 🙀 🖪 🗖 📼	幕 🎘 🔜
1 💱 😼 📷 🔜 🔲 🖹 🖹 🖉 🖑 🔁 🖓 🖓	
801F F7 01 02 57 S7 Register: B	Registers 🛋 🗙
8022 58 *7. HEX: 72 DEC: 114 Cancel [0K]	Name Value Extend.
DDUNETELD 6811 ASSEMBLED: Counter 1514131211109 8 7 6 5 4 3 2 1 0	A OC
Ocoococolilicolo	B 72
50 and may 4	D k\$ 0C72
8UZZ 59 ?ABI EQU *	Ŷ 00
8022 60 *8: {	SP 03FD 8013
8022 61 *9: count++;	PC 8026 F7
8022 F6 01 02 62 LDAB count	CCR DO SX_I
8025 5C 63 INCB	TMSK1 00
➡8026 F7 01 02 64 STAB count	TMSK2 00
8020 65 ±10 1	TFLG1 00
0029 00 00 00 00 00 00 00 00 00 00 00 00 00	PACTI 00
8029 /E 80 22 00 JMP /ABI	TOC1 FFFF
Memory Address: 0x0000100	
000000F0 00 00 00 00 00 00 00	
000000F8 00 00 00 00 00 00 00	=
00000100 00 00 46 00 00 00 00F	
> 00000128 00 00 00 00 00 00 00	
2 00000130 00 00 00 00 00 00 00 00	_
Me	<u> </u>
Ready In 68, 0	Col 1

A popup window appears. In this window you can change the register value in binary, decimal or hex format. After you enter needed value you can confirm by clicking 'OK' button or cancel the changes with 'Cancel' button.

5.2.9. Step 9 – Using Buffalo and Loader.

You can break executing your program and start Buffalo and Loader at any time. For this purpose you can open the Terminal window using "Terminal" command under "View" menu.

Micro-IDE - [Counter.lst]		_ 8 ×
<u> </u>	ools <u>W</u> indow <u>H</u> elp	_ <u>-</u>
D 🚅 🖬 🕼 🕺 🖻 🛍 🗠 🗠 4	🗿 🗇 🛅 💼 🕒 😨 🗛 😤 🖓	
1 1 1 1 1 1 1 1	📓 🕙 P P P 💱 🕵 😹 📰 🔳	
8022	60 *8: { Registers	×
8022	61 *9: count++; Name Va	lue Exten. 🔺
⇒ 8022 F6 01 02	62 LDAB count A 00	
8025 5C	63 INCB	03
8026 F7 01 02	64 STAB count X 10	0A
8029	65 *10: }	FD 8013
8029 7E 80 22	66 JMP ?AB1 PC 80	22 F6
8020	67 2AB2 EOU *	
Memory Address: 0x00000100		
000000F0 00 00 00 00 00 00 00 00		_
00000108 00 00 00 00 00 00 00 00 00		
00000110 00 00 00 00 00 00 00 00		
		-
¥)		
치		
leminal		<u>×</u>
Show or hide Terminal	Ln 66, Col 1	

After that you must set PORTE=0xFE and PC=0xFE60 using the Register Window and start the program using <GO> button (F5).

Micro-IDE - [Counter.lst]		_ 8 ×
☐ <u>File</u> <u>E</u> dit <u>V</u> iew <u>B</u> uild <u>P</u> roject <u>D</u> ebug <u>I</u> ools <u>W</u> indow <u>H</u> elp		- 8 ×
] D 😅 🖬 🕼 🖄 🗠 🛍 😂 🥔 🕮 🛍 🕘 😨] 🛤 🖓 🐐		
8022 60 *8: {	Registers	×
8022 61 *9: count++;	Name Value	Exten. 🔺
8022 F6 01 02 62 LDAB count	X 100A	+ _
8025 5C 63 INCB	SP 03FD	8013
➡8026 F7 01 02 64 STAB count	PC FE60	CE SX I
8029 65 *10: }	TCNT C7	04 (2)2
8029 7E 80 22 66 JMP ?AB1	TMSK1 00 TMSK2 00	
	•	╧
		<u> </u>
00000100 00 046 00 00 00 00 0F		
00000120 00 00 00 00 00 00 00		-
		_
☆*MINI-MAX/11-A*		
CHECKSUM [16EA] OK		
BUFFALO 3.4 (ext) - Bit User Fast Friendly Aid to Logical Opera	ation	
Starts or continues the program In 70, Col 42		

Now, for example, you can see memory area in Terminal Window using the Buffalo command: MD 100 120.

Micro-I	DE - [C	ount	er. Ist	1				S. P. 1													
	dit ⊻ie ⊡ 4571	₩ <u>B</u>	uild	Projec m	t <u>D</u> el	\sim	_00ls /==	_ <u>W</u> ind ⊲⊳ aa	iow <u>I</u>	<u>t</u> elp	. L G		a .24	-0	67						_8×
		ð					≥						98 89 V C	89	199		_				
] 🗠 🎮		العهد	\$ \$ 8 ,			ļ	30	0	{*}	{ } * {	(}* []	1		8 9.		r 🔜					
8022								1	50	*8	:		{						Registers		
8022				_					51	*9	:			COI	unt	++;			Name	Value 100A	Exten.
8022	Fo	U.	1 U:	2					52	L	DAB	CO	unt					_	Ŷ	00	
8025	50	0.	1 0.	~				1	53 54	1	NCB	~~~	+						SP	03FD	8013 CE
8029	- E /	υ.	L U.	2					55 55	*1	TAD N•	00	unc i						CCR	DO	SX_I_
8029	7 ह	80	1 2	2					56 56	т. П	O. MP	2 A B	, 1						TCNT TMSK1	C7	
8020				_				1	57	7A	в2	EOU	*					-1	TMSK2	00	-
•												- 2 -									
≚ Memor		_	▼ Ar	Idress:	n×nnr	100100)														
	00.00	00	46.0	0 00	00.0	0 00	F														
000001	.08 00	00	00 0	0 00	00 0	0 00															=
000001	10 00	00	00 0	0 00	00 0	0 00	••••														
P 000001	.18 00	00	00 0	0 00	00 0	0 00		• • • • •													-
× BUEE	ALO.	2 /	1 14	-v+)	-	Bit	• 11e	or	Fac	2 + 1	F ri,	and	1.77	vid	to	Log	cel	Oner	etion		_
≥MD	100	120	= (\ 	SAU,		DI	. 0.)CL	rua			sna.	LY	.ra	0	шоg.	LCUI	oper	acton		
	200		-																		
0100	00	00	46	00	00	00	00	00	00	00	00	00	00	00	00	00	F				
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00					
min >																					
Ter																					-
Ready																	Ln 7	'0, Col 42			

Of course, you can go back to your program at any time, but you must break non-stop executing the Buffalo monitor program using "Break" command under "Debug" menu.

Then, you can start your program again from every address but please have in mind the accumulator, registers, Program Counter, and Stack Pointer are random. The correct way is to run your program from address \$8000 or manually set up all registers to correct values.



5.3. Timer.

This example demonstrates simulation of timers and interrupts.

First, this program prepares RAM vectors for interrupts (if you are using Buffalo) and synchronizes the TCNT value using the Timer Overflow interrupt (to provide equally executing on real board and simulator). Then, it sets up timer output compare 1 (TOC1) and timer output compare 2 (TOC2) to generate regular interrupts. When interrupt occurs the corresponding interrupt service routine (ISR) sends 'TOCx' string to the serial port. When program waits for interrupts it sends char '.' to the serial port.

NOTE: When you begin executing this example, make sure that the file 6811int.h contains the following line (Fig.2). Use "Open" command under "File" menu (Fig.1) to view the file.

Open			? X
Look jn: 🚺	MicroC	- 🗈 💆	📸 🔳
BipomLib Docs11 Docs51 Examples Help Lib11	Lib51 Mon51 Projects 10 6811 int. h 6811 io. h 1 6811 reg. h	h) 8051adc.h h) 8051bit.h h) 8051int.h h) 8051io.h h) 8051reg.h h) Assert.h	h buffalo.h h Ctype.h h Debug.h h Limits.h h Setjmp.h h Stdarg.h
•			F
File <u>n</u> ame:	6811int		<u>O</u> pen
Files of <u>type</u> :	C Files (*.c,*.h)	•	Cancel
Fig.1			



Fig.2

5.3.1. Step 1 – Open the project.

Open the project "Tim11.prj" using "Open Project" command under "Project" menu.

III Micro-IDE - [Tim11.c]	
<u>File Edit View Build Project Debug Tools Window</u>	
D 😅 🖬 🕼 X 🖻 🖻 🖄 🗠 😂 🍩 🎬 🛚	D 🗢 🛛 🦛 🕉 🖌 🐂
5 () 14 () 14 () 17 ()) 🗗 (F) 📴 🐝 😓 🐝 🖆 💻 🗖
Workspace X Tim11: Project Files printf (" TFLG2=0x TMSK2=0x	<pre>NnSynchronize TCNT\n"); 80; // clear TOF flag 80; // Enable interrupt from Timer overflow</pre>
asm { CLI }	// global enable maskable interrupts
while (1 { printf }) // waiting timer interrupts (".");
Files	
Build Debug Find in Files 1 Find in Files 2 Loader	
Ready	Ln 1, Col 1

5.3.2. Step 2 – Build the project.

Click "Build All" icon button on the toolbar of Micro-IDE and make sure there are no errors reported in Output\Build window.

Micro-IDE - (Tim11.c)
Eile Edit View Build Project Debug Iools Window Help
🗅 📂 🖬 🕼 🤽 🕾 😂 🥸 🌉 🗈 🕢 😰 🛤 新 🏠 👒
/orkspace I M printf("\nSynchronize TCNT\n");
TFLG2=0x80; // clear TOF flag TMSK2=0x80; // Enable interrupt from Timer overflow
asm { CLI // global enable maskable interrupts }
<pre>while (1) { // waiting timer interrupts printf("."); }</pre>
Compiling C:\bipom\devtools\MicroC\Examples\6811\Timer\Tim11.c Linking Tim11.asm Generating 'Tim11.hex' First pass Opt Second pass 0 error(s). Build Debug Find in Files 1 Find in Files 2 Loader
ild all the files

5.3.3. Step 3 – Start simulator.

Start the simulation using the F11 button.



First of all, the program fills the JMP Table for interrupts (function RamVectors) because the Buffalo monitor redefines interrupt vectors.

VECTOR JUMP TABLE:

00DF:7E 81 15 JMP FUNC_1 (Interrupt from TOC1) 00DC:7E 81 27 JMP FUNC_2 (Interrupt from TOC2) 00D0:7E 81 39 JMP FUNC_3 (Interrupt from TOVF)

Now you can open the Registers and Terminal windows using "Registers" and "Terminal" commands under "View" menu.



5.3.4. Step 4 – Run the sample.

The program begins with synchronizing TCNT using the Timer Overflow interrupt (TOVF). When interrupt TOVF occurs the program disables TOVF interrupt and sets up timer output compare 1 (TOC1) and timer output compare 2 (TOC2) to generate regular interrupts.

Trace the synchronizing TCNT, main program execution and see how TOC1 and TOC2 interrupts are handled. Use F11 button to step.

III Micro-IDE - [Tim11.c]						_ 8 ×
<u>File Edit View Build Project Debug Tools</u>	<u>W</u> indow <u>H</u> elp					_ 8 ×
] 🗅 🚅 🖬 🕼 🗼 🖬 📾 🗠 🗠 🎒	🧇 🛗 💼 🔴	🕐 🕺 🖓 🖓				
🖪 🗖 🔲 🖾 🧔 🛱 🧾 🗉 🖹 🕷	4 6 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0) 💱 🙀 😓 🕷 😰 📃 🗖				
<pre>printf(".");</pre>	Step into (F	Tayminal	- A X	Registers		×
→ → →				Name	Value	Exten. 🔺
}		Synchronize TCNT		A	80	
11		-		В	9000	
INTERRUPT(TOC1) myfunc1	0	Begin		X	815F	
		begin		Y	8161	
		 mog1		SP	03FD	8013
TFLGI=UX80;	// clea	TOCI		CCB	8057 E0	SXH
printf("\nTOCI\n");				TCNT	2003	<u>.</u>
TOC1=TCNT+0xA000;	// New	TOC2		TMSK1	CO	
}				TMSK2	00	
11				TELG2	38	
INTERRUPT (TOC2) myfunc2	ιo Ι			PACTL	00	
				TOC1	49F3	
mm c1 - 0 - 40 -	(/			1002	BBC3	
TFLGI=UX40;	// Clea			7904	FFFF	L _ L
•			v			
×						1
Please use the Toolkit Configurator to change thes	e settinas if needed. No	te that simulation of interrupts, timers and/or :	serial port r	nav slow do	wn simula	tion.
Chip Type: 6811 Manufacturer: Generic						
Lonnecting terminal to debugger simulation						-
						•
Build Debug Find in Files 1 Find in Files 2 L	oader					
Step into next statement		Ln	48, Col 1			

5.3.5. Step 5 – Changing registers values.

Now you can change the TOC1 and TOC2 value (for example, set both them to 0x0000) so that interrupts from TOC1 and TOC2 happen at the same time. To change values of the registers you can use the Registers Window.



Keep on pressing F11 and trace interrupts handling. Because TOC1 has higher priority level, these interrupts are executed first.

III Micro-IDE - [Tim11.c]	_ 8 ×
<u>Fi</u> le <u>E</u> dit <u>V</u> iew <u>B</u> uild <u>P</u> roject <u>D</u> ebug <u>I</u> ools <u>W</u> indow <u>H</u> elp	_ <u>_</u> X
printf(".");	Registers 🛋 🗙
⇒ } TOC1	Name Value Exten.
}	A 80
// TOC2	D 8000
INTERRUPT(_TOC1_) myfunc1()	X 815F
{	1 SP 03ED 8013
TFLG1=0x80; // clea ToC1	PC 8057 7E
<pre>printf("\nTOC1\n");</pre>	CCR EO SXH
TOC1=TCNT+0xA000; // New TOC2	TMSK1 C0
}	TMSK2 00
// TOC1	TELG1 38
INTERRUPT(TOC2) myfunc2()	PACTL 00
{	TOC1 BBE4
TFLG1=0x40; // clea	TOC3 FFFF
Interrupts, timers and serial port are simulated.	
Please use the Toolkit Configurator to change these settings if needed. Note that simulation of interrupts, timers and/or serial p	ort may slow down simulation.
Chip Type: 6811 Manufacturer: Generic	
Connecting terminal to debugger simulation	
Build Debug Find in Files 1 Find in Files 2 Loader	
Ready In 48, Col	1

5.3.5. Step 5 – Running in non-step mode.

You can start the program using the F5 button or by clicking "Go" toolbar button to execute program continuously.

III Micro-IDE - [Tim11.c]		
☐ File Edit View Build Project Debug Tools Window Help		_ 8 ×
] D 📁 🖬 🕼 🕺 🕰 😂 😂 🕮 🗈 🔶 🕲 🛤 🖧 🐐		
🗖 🖾 🐼 😸 📰 🕄 🕒 한 한 한 한 🖉 🖾 📰 🧱 💷		
➡ TFLG1=0x80; // clea	Kegisters	<u>×</u>
<pre>printf("\nTOC1\n");</pre>	▲ Name	Value Exten. 🔺
TOC1=TCNT+0xA000; // New TOC1	A	25
}	D	2E
// TOC2	X	00
INTERRUPT(_TOC2_) myfund2()	SP	0380 811F
{TOC1	PC	806C C6
TFLG1=0x40; // clea	TCNT	4E98
printf("\nTOC2\n"); TOC2	TMSK1	C0
TOC2=TCNT+0xA000; // New	TFLG1	B8
TOC1	TFLG2	C0
	TOC1	4E6D
INTERROPT(_TOVE_) myrunc3() TOC2	TOC2	8F7F
	- 7204	FFFF -
×		
Interrupts, timers and serial port are simulated.		ᅴ
Please use the Toolkit Configurator to change these settings if needed. Note that simulation of interrupts, timers and/or serial	l port may slow do	own simulation.
Chip Type: 6811 Manufacturer: Generic		
Connecting terminal to debugger simulation		<u> </u>
Build Debug Find in Files 1 Find in Files 2 Loader		
Ready Ln 53, 0	Col 1	