

EECE416 :Microcomputer Fundamentals and Design

source: www.mwftr.com

Easy68K

– Editor, Assembler, and Simulator

TUTORIAL

Charles Kim

Easy68K and Download

EASy68K Home, Free 68000 Assembler, 68000 Simulator, 68000 Assembly Language - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.easy68k.com/

Most Visited Free Hotmail Customize Links Windows Marketplace Windows Media Windows

[Prof Kelly](#) [EASy68K](#) [Examples](#) [Forum](#)

EASy68K Editor/Assembler/Simulator for the 68000



Welcome to the EASy68K home page. EASy68K is a 68000 Structured Assembly Language edit, assemble and run 68000 programs on a Windows PC. No additional hardware is required. project distributed under the GNU general public use license.

EASy68K, the #1 [68000 Assembler](#) and [68000 Simulator](#) according to Google.

Download

Check the [Forum](#) for latest version information.

[SetupEASy68K.exe](#) Executable with installer

Ads by Google

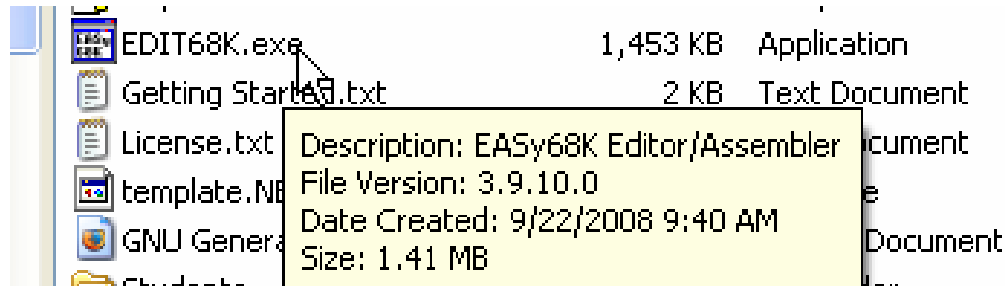
Prototype PCB Assembly
24 hour turn, Full-service, RoHS New customer discount
www.screamingcircuits.com

Download and Install this!

EDIT68K

⌘ EDIT68K.EXE

📁 For Coding and Assembling



⌘ Coding

⌘ Save

```
EASy68K Editor/Assembler v3.9.10
File Edit Project Options Window Help
TrapExample1.X68
-----
* Program      : TrapExample.X68
* Written by   : CK
* Date        : 01OCT08
* Description  : Trap examples under Easy68k Environment
-----
;DATA PART
BS      EQU    $08    Backspace
HT      EQU    $09    Tab (horizontal 5 characters)
LF      EQU    $0A    New line (line feed)
VT      EQU    $0B    Vertical tab (4 lines)
FF      EQU    $0C    Form Feed (Always end printing with a Form Feed.)
CR      EQU    $0D    Carriage Return

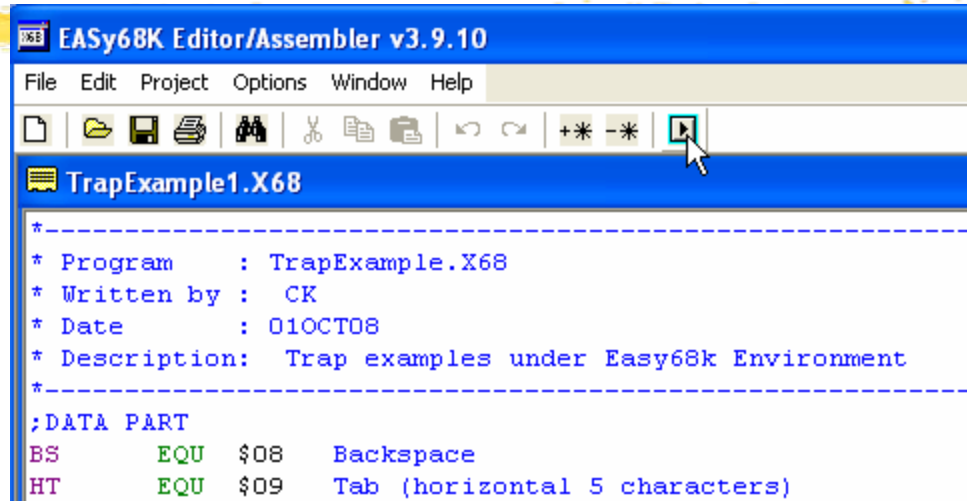
                ORG    $100
msg      DC.B    CR,LF,'good guess!',CR,LF
                DC.B    0                ;string must be ended with 0
wmsg     DC.B    CR,LF,'guess again',CR,LF
                DC.B    0                ;string ended with 0
inqr     DC.B    'Guess a character [a - z]',CR,LF
                DC.B    0                ;ended with 0

                ORG    $200
Store    DS.B    30                ;Allocate 4 Bytes starting @Store. MEMORY

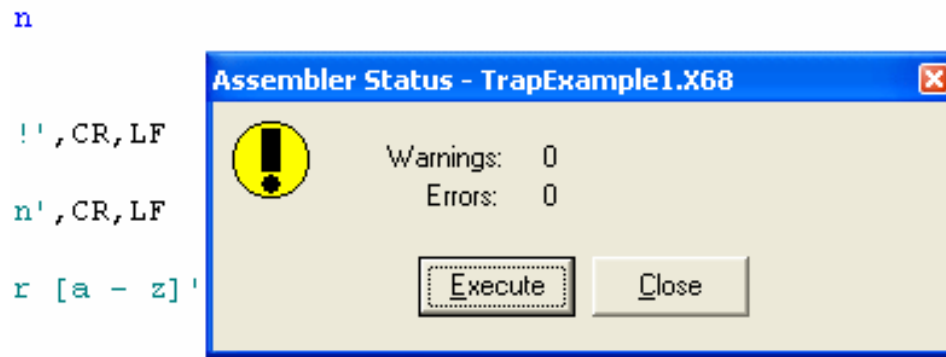
;TRAP # in EASY68K
; (Put Task number into DO)
; then TRAP #15
```

Assembling

- ⌘ Click the “Assemble Source Code” button OR F9 Key

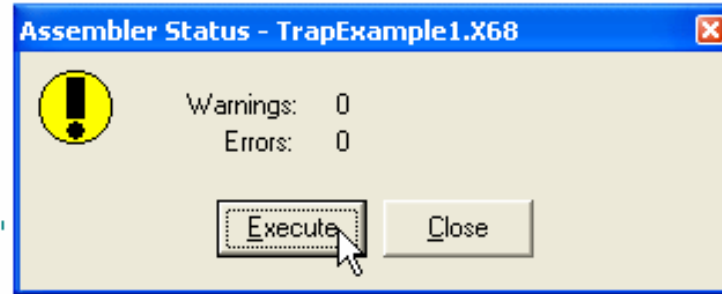


- ⌘ When there is no Error, the following window pops up.



Simulation

⌘ Click, "Execute" button.



⌘ Initial Simulation Screen

⌘

C:\Easy68k\TrapExample1.S68

File Run View Options Help

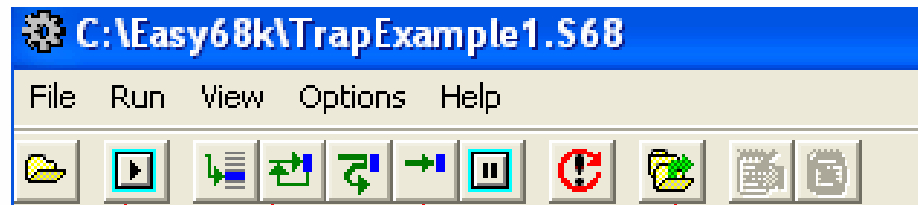
Registers

D0=	00000000	D4=	00000000	A0=	00000000	A4=	00000000	T	S	INT	XNZVC	Cycles
D1=	00000000	D5=	00000000	A1=	00000000	A5=	00000000	SR=	0010000000000000			0
D2=	00000000	D6=	00000000	A2=	00000000	A6=	00000000	US=	00FF0000			Clear Cycles
D3=	00000000	D7=	00000000	A3=	00000000	A7=	01000000	SS=	01000000	PC=	00000000	

Address -----Code----- Line -----Source----->>

```
0000021E          30 ;TASK No.      FUNCTIONS
0000021E          31 ;=====      =====
0000021E          32 ;0            PRINT_MSG (CRLF) (Display Mess
0000021E          33 ;1            PRINT_MSG (Same as 0 but w/o C
0000021E          34 ;5            READ_CHR Read a single charact
0000021E          35 ;6            PRINT_CHR Display a single ch
0000021E          36 ;9            HALT halt the simulator
0000021E          37
0000021E          38
0000021E          39 ;PROGRAM PART
00000000          40 START  ORG    $0
```

Simulation Buttons



Run (F9)

Reload Program (Ctrl + F3)

Run To Cursor (Ctrl + F9)

Rewind Program (Ctrl + F2)

Auto Trace (F10)

Pause (F6)

Step Over (F8)

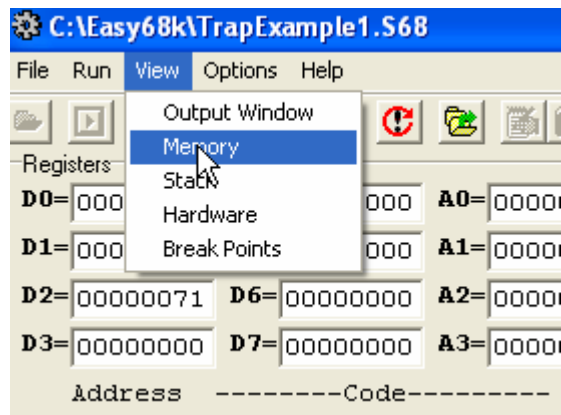
Trace Into (F7)

Memory View

⌘ After Auto Trace or Run (with Cmd screen)

```
Sim68K I/O
Guess a character [a - z]
a
guess again
c
guess again
d
guess again
e
guess again
f
guess again
g
guess again
q
good guess!
```

⌘ Click “View” and Select “Memory”



Memory View

68000 Memory

Address: From: 00000000 To: 00000000 Bytes: 00000000 Copy Fill Save

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF
00000000:	24	7C	00	00	02	00	32	3C	00	60	10	3C	00	00	22	7C	\$ ----2<-`-<--"
00000010:	00	00	01	20	4E	4F	10	3C	00	05	4E	4F	14	01	14	C1	--- NO-<--NO----
00000020:	0C	02	00	71	67	00	00	14	32	3C	00	60	10	3C	00	00	---qg---2<-`-<--
00000030:	22	7C	00	00	01	10	4E	4F	60	DC	32	3C	00	60	10	3C	" ----NO`-2<-`-<--
00000040:	00	00	22	7C	00	00	01	00	4E	4F	FF	FF	FF	FF	FF	FF	--" ----NO-----
00000050:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000060:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000070:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000080:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000090:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000A0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000B0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000C0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000D0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000E0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000000F0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000100:	0D	0A	67	6F	6F	64	20	67	75	65	73	73	21	0D	0A	00	--good guess!---
00000110:	0D	0A	67	75	65	73	73	20	61	67	61	69	6E	0D	0A	00	--guess again---
00000120:	47	75	65	73	73	20	61	20	63	68	61	72	61	63	74	65	Guess a characte
00000130:	72	20	5B	61	20	2D	20	7A	5D	0D	0A	00	FF	FF	FF	FF	r [a - z]-----
00000140:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000150:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000160:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000170:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000180:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000190:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001A0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001B0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001C0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001D0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001E0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
000001F0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----
00000200:	61	63	64	65	66	67	71	FF	FF	FF	FF	FF	FF	FF	FF	FF	acdefgq-----
00000210:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	-----

Row

Page

Live

Page

Live

TRAPs in EASY68K

⌘ TRAP

- ⊞ Interruption of Execution
- ⊞ Respond Programmer's Task Command

⌘ Number of Tasks in Easy68K

- ⊞ 22 Tasks
- ⊞ Examples
 - ⊞ Read Message (from Keyboard)
 - ⊞ Print Message (to Computer Monitor)
 - ⊞ Read a Character
 - ⊞ Print a Character
 - ⊞ Read a Number
 - ⊞ Print a Number
 - ⊞ Etc

⌘ Traditional TRAP tasks

- ⊞ **Print Message (to Computer Monitor) of address at A1**
 - ⊞ --- Task # 0 (with CR.LF)
 - ⊞ --- Task #1 (without CR.LF)
- ⊞ **Read a Character and store it at D1.B --- Task #5**
- ⊞ **Print a Character stored at D1.B --- Task #6**
- ⊞ **Key Echo On on or off --- Task #12 (OFF- D1.B==0, ON --- D1.B== Non zero)**

⌘ How is a TRAP called/executed?

- ⊞ Put task number to A1 or D0
- ⊞ Put address into A1 (for "print message" case)
- ⊞ Then "Trap #15" instruction line

```
MOVE.B #5, D0          Read a single character
TRAP   #15             Store into D1
```

```
└ MOVE.B #0, D0
  MOVEA.L #rmsg, A1
  TRAP   #15
```

2 or 3 steps for TRAP

⌘ Step 1 (for only *Read /Print Message* case)

- ☑ Put the length of the message in number of characters into D1 register

⌘ Step 2

- ☑ Put the task number to D0 Registers

☒ Task Numbers

- 0 ---- Print Message (which is stored at the memory whose starting address is stored in A1 register) ended with LF and CR
- 1 ---- Print Message ended without LF/CR
- 5 ---- Read a single character from Keyboard (the read character is stored to D1 register)
- 6 ---- Print a single character (stored in D1 register) into the computer screen

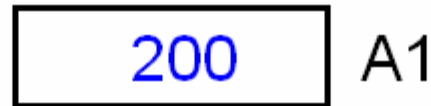
⌘ Step 3

- ☑ Execute the Trap by the instruction of "Trap #15"

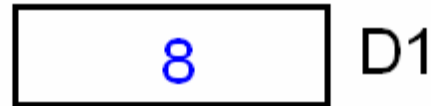
Illustration-1

Print a message of 8 characters stored from the memory address of \$200. * Task # is 1

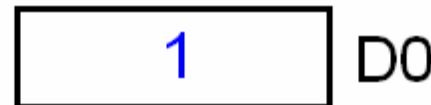
1. Put addr \$200 into A1.



2. Put the length to D1 register.



3. Put the task # to D0



4. 'Trap #15'

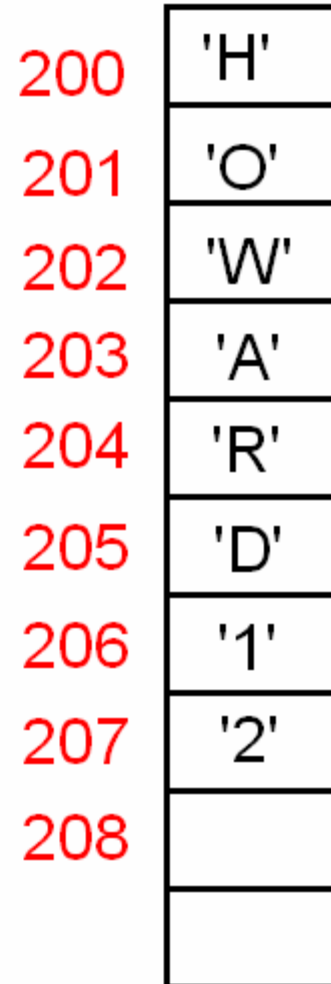


Illustration - 2

Read a character typed in from a keyboard.
(Task # = 5)

1. Put the task # to D0 register

D0

2. Waits for a key (Program does it automatically. Only when key is entered then the next line of code executes.)



D1

3. When a key is entered, its ASCII value is stored in D1 register.

Trap Example Code

```
EASy68K Editor/Assembler v4.7.3 - [TrapEx.X68]
File Edit Project Options Window Help
-----
* Program      : TrapEx.X68
* Written by   : CK
* Description:  Trap examples under Easy68k Environment
*
;DATA PART

LF      EQU  $0A   New line (line feed)
CR      EQU  $0D   Carriage Return

                ORG      $100
rmsg    DC.B    CR,LF,'good guess!',CR,LF           ;string must be ended with 0
                DC.B    0
wmsg    DC.B    CR,LF,'guess again',CR,LF          ;string ended with 0
                DC.B    0
inqr    DC.B    'Guess a character [a - z]',CR,LF
                DC.B    0                           ;ended with 0

;TRAP # in EASY68K
; (Put Task number into D0)
; then TRAP #15
```

- ⌘ Type the first code
- ⌘ Assemble it
- ⌘ Simulation with Step Over (F8) and Trace the D0, D1, and A1 registers