

(230330d) okay, It seems that Dev-Cpp has issues, as in Segger IDE, the debug is okay without any seg faults ... one issue is, I have not yet mapped my keybindings to my unified way as in other IDEs according to table below .... time to teach segger to use this too

.....

## Unifying MIXWORX keyboard shortcuts for debugging

=====

	Atmel Studio (C-HW)	VisualStudio (C#)	VSCoDe (REFACTOR)
DevCpp (C-DEBUG)			

=====

=====

Tools.Options	= !! Ctrl+Alt+K	!! Ctrl+Alt+K	Keyboard Shortcuts
Ctrl+Alt+K			
Project.Properties	= !! Ctrl+Alt+Enter	!! Ctrl+Alt+Enter	!! Ctrl+Alt+Enter
Ctrl+Alt+Enter			
Window.OpenView	= ???	???	!! Shift+Alt+Enter ---
Full Screen	= F11	= F11	= F11 = F11
Command Pallete	=	=	= F1 =

Edit.Redo	= !! Shift+Ctrl+Y	!! Shift+Ctrl+Y	!! Shift+Ctrl+Y	Shift+Ctrl+Y
Edit.Undo	' = !! Shift+Ctrl+\	!! Shift+Ctrl+\	!! Shift+Ctrl+\	
Shift+Ctrl+\				
Window.CloseDocumentWindow	= !! Ctrl+W	?? Ctrl+W	!! Ctrl+W	
Ctrl+W				
Edit.ToggleLineComment	= !! (Shift+)Ctrl+Q	!! Ctrl+Q	!! Ctrl+Q	Ctrl+Q
Edit.BlockLineComment	= ?? ---	!! Shift+Ctrl+Q	!! Shift+Ctrl+Q	
Ctrl+Q				
Edit.SelectCurrentWord	= !! Ctrl+Shift+W	?? Ctrl+Shift+W	---	--
-				
Edit.Find	= Ctrl+F	Ctrl+F	Ctrl+F	Ctrl+F
Edit.FindInFiles	= Ctrl+Shift+F	Ctrl+Shift+F	Ctrl+Shift+F	
Ctrl+Shift+F				
Edit.Replace	† = !! Ctrl+R	!! Ctrl+R	!! Ctrl+R	Ctrl+R
Edit.ReplaceInFiles	= !! Ctrl+Shift+R	!! Ctrl+Shift+R	!! Ctrl+Shift+R	
Ctrl+Shift+R				

Edit.FindNext	= F3	F3	F3	F3
Edit.FindPrevious	= Shift+F3	Shift+F3	Shift+F3	Shift+F3
Edit.GotoDeclaration	= !! Ctrl+F6	!! Ctrl+F6	!! Ctrl+F6	Ctrl+F6
Edit.GotoDefinition	= !! Ctrl+F7	!! Ctrl+F7	!! Ctrl+F6	Ctrl+F7
Edit.VAssistX.FindSymbolDialog	= !! Ctrl+G	!! Ctrl+G	!! Ctrl+G	Ctrl+G
Edit.Goto	= !! Shift+Ctrl+G	?? ---	!! Shift+Ctrl+G	
Shift+Ctrl+G				
Edit.IncrementalSearch	= Ctrl+I	Ctrl+I	?? ---	Ctrl+I
Edit.CompleteWord	= !! Ctrl+Alt+Right	?? Ctrl+Alt+Right	?? ---	!!
Ctrl+Alt+Right				
Edit.ToggleBookmark	= !! Ctrl+Alt+B	!! Ctrl+Alt+B		
Edit.NextBookmark	= !! Ctrl+B	!! Ctrl+B		
Edit.Show(In)Bookmarks	= !! Ctrl+Alt+Shift+B			
Edit.CollapseAllOutlining	= !! Ctrl+/	!! Ctrl+/		
Edit.ExpandAllOutlining	= !! Ctrl+*	!! Ctrl+*		
Build.BuildSolution	= !! F9	!! F9	Arduino:Verify	F9
Build.Compile	= !! Ctrl+F9	!! Ctrl+F9		Ctrl+F9 (syntax check)
Build.RebuildSolution	= !! Ctrl+Alt+F9	!! Ctrl+Alt+F9	Arduino:Intellisense	
Ctrl+Alt+F9				
Build.CleanSolution	= !! Ctrl+Alt+C	!! Ctrl+Alt+C		
Ctrl+Alt+C				
Debug.StartWithoutDebugging	= !! F10	!! F10	Arduino:Upload	
F10				
Debug.AttachTarget	= Ctrl+Alt+F5	Ctrl+Alt+F5		
Ctrl+Alt+F5				
Debug.StartDebuggingAndBreak	= Alt+F5			
Debug.ToggleBreakpoint	= !! F4	!! F4		F4
Debug.Start	= F5	F5	F5	
Debug.StepOver	= !! F6	!! F6		F6
Debug.StepInto	= !! F7	!! F7		F7

Debug.StepOut	= !! F8	!! F8	F8
Debug.StopDebugging	= !! Ctrl+F5	!! Ctrl+F5	Ctrl+F5
Debug.RunToCursor/Continue	= !! Shift+F5	!! Shift+F5	
Shift+F5 (continue)			
Debug.BreakAll	= Ctrl+Alt+Break	Ctrl+Alt+Break	
	Shift+Ctrl+Alt+F5		
Debug.QuickWatch	= !! Ctrl+Shift+F4	!! Ctrl+Shift+F4	
Ctrl+Shift+F4 (modify)			
Debug.AddWatch	= !! Shift+F2	!! Shift+F2	
Shift+F2			
Debug.DeleteWatch	= ?? Ctrl+F2	?? Ctrl+F2	Ctrl+F2
Debug.ClearAllWatches	= ?? Ctrl+Alt+F2	?? Ctrl+Alt+F2	
Ctrl+Alt+F2			
Debug.Disassembly	= !! Ctrl+F11	!! Ctrl+F11	Ctrl+F11
Debug.Breakpoints	= Alt+F9		
Debug.EnableBreakpoint	= Ctrl+F9		
Debug.DeleteAllBreakpoints	= Ctrl+Shift+F9		
Debug.FunctionBreakpoint	= Ctrl+B		
Debug.Reset	= Shift+F5, Ctrl+Alt+F5		
Debug.Restart	=		
Debug.RunToCursor	= Ctrl+F10		
Debug.SetNextStatement	= Ctrl+Shift+F10		
File.Rename	= F2		
Tools.DeviceProgramming	= Ctrl+Shift+P		
VAssistX.RefactorRename	= Shift+Alt+R		
View.NextError	= Ctrl+Shift+F12		



It looks like you have a table of unified keyboard shortcuts for debugging across different IDEs, including Atmel Studio, Visual Studio, VSCode, and DevCpp. To set up similar keybindings in SEGGER Embedded Studio, follow these steps:

1. Open SEGGER Embedded Studio.
2. Navigate to the menu bar and click on "Tools."
3. In the "Tools" menu, select "Options..."
4. In the "Options" window, select "Keyboard" under the "Environment" section.
5. You will see a list of commands and their current keyboard shortcuts. To change a command's shortcut, click on the command, then click the "Press shortcut keys" textbox and press the desired key combination.
6. Click "Assign" to set the new shortcut.
7. Repeat this process for all commands you want to update to match your unified keybindings.
8. Click "OK" to save your changes and close the "Options" window.

Remember that the commands and their names might be slightly different in SEGGER Embedded Studio compared to other IDEs, so you might need to search for equivalent commands when setting up your keybindings.



how is in segger named the Project Options for keybinding?

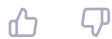


In SEGGER Embedded Studio, the command for opening the Project Options is called "Project Edit Options...". To set a keybinding for it, follow these steps:

1. Open SEGGER Embedded Studio.
2. Navigate to the menu bar and click on "Tools."
3. In the "Tools" menu, select "Options..."
4. In the "Options" window, select "Keyboard" under the "Environment" section.
5. In the "Show commands containing" search bar, type "Project Edit Options" to filter the list of commands.
6. Click on "Project.EditOptions" in the list.

7. Click the "Press shortcut keys" textbox and press the desired key combination (e.g., Ctrl+Alt+Enter, according to your table).
8. Click "Assign" to set the new shortcut.
9. Click "OK" to save your changes and close the "Options" window.

Now you have successfully set a keybinding for the Project Options in SEGGER Embedded Studio.



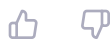
no, its ViewProperties ... very hard to find :-)



Apologies for the confusion. In SEGGER Embedded Studio, the command for opening the Project Properties is indeed called "ViewProperties." To set a keybinding for it, follow these steps:

1. Open SEGGER Embedded Studio.
2. Navigate to the menu bar and click on "Tools."
3. In the "Tools" menu, select "Options..."
4. In the "Options" window, select "Keyboard" under the "Environment" section.
5. In the "Show commands containing" search bar, type "ViewProperties" to filter the list of commands.
6. Click on "View.Properties" in the list.
7. Click the "Press shortcut keys" textbox and press the desired key combination (e.g., Ctrl+Alt+Enter, according to your table).
8. Click "Assign" to set the new shortcut.
9. Click "OK" to save your changes and close the "Options" window.

Now you have successfully set a keybinding for the Project Properties in SEGGER Embedded Studio.





how is in segger identified Find... for keybindings?



In SEGGER Embedded Studio, the command for opening the "Find..." dialog is called "Edit.Find". To set a keybinding for it, follow these steps:

1. Open SEGGER Embedded Studio.
2. Navigate to the menu bar and click on "Tools."
3. In the "Tools" menu, select "Options..."
4. In the "Options" window, select "Keyboard" under the "Environment" section.
5. In the "Show commands containing" search bar, type "Edit.Find" to filter the list of commands.
6. Click on "Edit.Find" in the list.
7. Click the "Press shortcut keys" textbox and press the desired key combination (e.g., Ctrl+F, according to your table).
8. Click "Assign" to set the new shortcut.
9. Click "OK" to save your changes and close the "Options" window.

Now you have successfully set a keybinding for the "Find..." command in SEGGER Embedded Studio.



I am unable to map Find... it's okay to be selected from menu but the keybinding doesn't work



I apologize for the inconvenience you're experiencing. It seems there might be an issue with the keybinding in SEGGER Embedded Studio. Here are a few suggestions to try and resolve the issue:

1. Make sure there are no conflicting keybindings for the same key combination (Ctrl+F). You can do this by searching for the keybinding in the "Keyboard" section of the "Options" window. If you find a conflict, remove or change the conflicting keybinding.

2. After setting the keybinding, ensure that you've clicked the "Assign" button before closing the "Options" window. This step confirms the new keybinding.
3. Try using a different key combination to see if the issue is specific to the Ctrl+F keybinding. For example, try setting the keybinding to Ctrl+Shift+F or another combination that doesn't conflict with existing keybindings.
4. If none of the above steps work, you may want to try restarting SEGGER Embedded Studio to see if the keybinding takes effect after relaunching the application.

If the problem persists, it's possible that there's a bug or limitation within SEGGER Embedded Studio. In this case, you can report the issue to SEGGER's support team, providing details of the problem and the steps you've taken to try to resolve it. They may be able to provide a solution or fix the issue in a future update.

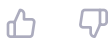


well, segger ide has now mapped keybindings to mixworx standard, okay



I'm glad to hear that you were able to successfully map the keybindings in SEGGER Embedded Studio to your MIXWORX standard. Now you should be able to use the same familiar key combinations across different IDEs, making it easier and more efficient to work with various development environments.

If you have any further questions or need assistance with any other aspect of SEGGER Embedded Studio or other development tools, please don't hesitate to ask.



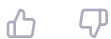
now, I am loudly thinking about possible features of VMEX in its tiniest version for 8kB flash; it seems that we can fit there only the interpreter core with some support for serial program load and flashing over text hex format or so... so only THIS SINGLE of full monitor, which will be available in larger versions of VMEX (regular monitor commands, inclding disassembly and line assembly of full ASLIX) ... so now I am targeting the 8kB flash, having now interpreter compiled into cca 2,5KB in size optimized C, cca 4,5kB for speed optimized C and I am working hard on best/simplest source code in C which can handle as source for manual coversion of code to target assemblers too...



It sounds like you are working on developing a minimal version of VMEX for an 8kB flash memory constraint. Given the limited memory, focusing on the interpreter core and support for serial program load and flashing over a text hex format is a good approach. Here are a few suggestions to help you optimize your implementation:

1. Optimize your code: Keep the code as minimal and efficient as possible. Use simple data structures and algorithms to reduce memory overhead. Opt for smaller variable types where possible, and avoid using global variables if not necessary.
2. Remove unnecessary features: Since you are targeting a tiny version of VMEX, consider removing any features that are not essential to the core functionality of the interpreter.
3. Use compiler optimizations: Make use of compiler optimization flags when compiling your code to ensure that the compiled binary is as small as possible. For example, with GCC, you can use the `-Os` flag to optimize for size. Be aware that optimization levels can affect the balance between code size and performance.
4. Consider compression techniques: If you still need to reduce the memory footprint of your implementation, you might want to explore using compression techniques for the text hex format. This could help you save some space in the limited memory.
5. Manual conversion to target assemblers: When converting the C source code to target assemblers manually, make sure to optimize the assembly code for the target platform. Focus on minimizing instruction count and using registers efficiently.

Keep in mind that you may need to make trade-offs between code size and performance in order to fit the core functionality of VMEX into the 8kB flash memory constraint. Good luck with your project, and if you have any questions or need further guidance, please don



please, continue



Of course! Here are a few more suggestions to help you further optimize your VMEX implementation for an 8kB flash memory constraint:

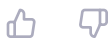
6. Reuse code where possible: Look for opportunities to reuse existing code instead of writing new functions for similar tasks. Code reuse can help you reduce the overall size



of your implementation.

7. Minimize library dependencies: Only include the libraries that are absolutely necessary for your VMEX implementation. Each additional library will likely increase the size of your compiled binary.
8. Use inline functions: When appropriate, use inline functions to reduce the overhead of function calls. This can help reduce the memory footprint while maintaining code readability.
9. Consider using a custom memory allocator: If your VMEX implementation requires dynamic memory allocation, you might want to consider using a custom memory allocator tailored to your specific use case. This can help reduce memory overhead and potentially improve performance.
10. Test and iterate: Continuously test your VMEX implementation on the target platform with the 8kB flash memory constraint. This will help you identify potential bottlenecks, areas for optimization, and any issues that may arise due to memory limitations. Iterate on your design and make adjustments as needed to ensure that the core functionality of VMEX is maintained.

Remember that achieving an optimal balance between code size, performance, and functionality is an iterative process that may require multiple design revisions. Don't hesitate to reevaluate your approach and make changes as necessary to meet the requirements of your target platform. If you need further guidance or have any questions, please feel free to ask.



okay, can you help me to solve one annoying problem in MPLAB X IDE? its is based on java runtime, and it seems there is some flaw in OpenFileDialog access, as there is large few seconds delay after launching the command in IDE, simply every new access to filesystem from java ide has this flaw, and it must be some general issue of java ....



I understand that you're experiencing a delay when opening the OpenFileDialog in MPLAB X IDE, which is based on the Java runtime. While I can't provide a direct fix for this issue, I can suggest a few general steps to help troubleshoot the problem:

1. Update your Java Runtime Environment (JRE): Ensure that you're using the latest version of the JRE compatible with MPLAB X IDE. Outdated versions of the JRE might have performance issues or bugs that have been addressed in newer releases.
2. Update MPLAB X IDE: Make sure you're using the latest version of MPLAB X IDE, as newer versions might contain performance improvements or bug fixes that can help resolve this issue.
3. Check your system's resources: Verify that your system has sufficient resources (RAM, CPU, and disk space) to run MPLAB X IDE and other applications smoothly. If your system is low on resources, it can cause performance issues in the IDE.
4. Disable antivirus or security software temporarily: Some antivirus or security software may interfere with the OpenFileDialog or file system access, causing delays. Try disabling any such software temporarily to see if the problem persists.
5. Check for network drives or slow devices: If your system is connected to network drives or slow storage devices, it may cause delays when accessing the file system. Disconnect any unnecessary network drives or devices and test the OpenFileDialog again.
6. Adjust Java runtime settings: You can try adjusting Java's runtime settings, such as the heap size, to improve performance. To do this, locate the `mplab_ide.conf` file in the MPLAB X IDE installation folder (usually in the `etc` subfolder) and modify the `-J-Xmx` parameter to allocate more memory to the Java runtime, for example, `-J-Xmx1024m` for 1 GB of memory.
7. Reinstall MPLAB X IDE: If none of the above steps resolve the issue, consider reinstalling MPLAB X IDE to ensure that all components are correctly installed and configured.

If the problem persists, it's possible that there's a bug or limitation within the Java runtime or MPLAB X IDE itself. In this case, you can report the issue to Microchip's support team, providing details of the problem and the steps you've taken to try to resolve it. They may be able to provide a solution or fix the issue in a future update.



well, it happened in 6.00 version and now also in newly installed 6.05 version, I hoped it can be solved, but its bound probably more to the used java runtime ...



If the issue persists across different versions of MPLAB X IDE, it could indeed be related to the Java runtime or the specific implementation of OpenFileDialog in the IDE. Here are

some additional steps you can try to address the issue:

1. Update Java Development Kit (JDK): MPLAB X IDE also requires a JDK for some features. Make sure you have the latest JDK version compatible with MPLAB X IDE installed on your system. You can download the latest JDK from Oracle's official website.
2. Check your Java installation: Make sure your Java installation is correctly configured. You can verify your Java installation by running `java -version` in a command prompt or terminal. If you have multiple Java versions installed, ensure that the correct version is being used by MPLAB X IDE.
3. Adjust graphics settings: Some systems with integrated and dedicated graphics cards might face issues with Java applications. Try adjusting your graphics settings to use a specific graphics card for MPLAB X IDE or update your graphics drivers.
4. Disable hardware acceleration: In some cases, disabling hardware acceleration can help with performance issues in Java applications. To disable hardware acceleration, locate the `mplab_ide.conf` file in the MPLAB X IDE installation folder (usually in the `etc` subfolder), and add the following line at the end of the file:

arduino

 Copy code

```
-J-Dsun.java2d.noddraw=true
```

Save the file and restart MPLAB X IDE to see if the issue is resolved.

5. Check for Java bugs or known issues: Search for any known bugs or issues related to the Java runtime or OpenFileDialog component that might be affecting MPLAB X IDE. You can look through the release notes for your Java version or search for reported issues on the Java bug tracker.
6. Consider using a different IDE: If the



well, isnt somewhere knowledge about OpenFileDialog stalling in java? something more generic, not tied to mplab x ide?



Yes, there are some more general issues with Java's `OpenFileDialog` that can cause performance problems or stalling. Here are some possible reasons and potential solutions:

1. **Slow file system access:** `OpenFileDialog` might stall or hang due to slow file system access, particularly when connected to network drives or slow storage devices. To resolve this, disconnect any unnecessary network drives or devices and test the `OpenFileDialog` again.
2. **Java's LookAndFeel settings:** Java applications, including the `OpenFileDialog`, can use different LookAndFeel (L&F) settings for their user interface. Some L&F settings can cause performance issues on certain platforms or systems. You can try changing the L&F setting of your Java application to see if it improves `OpenFileDialog` performance. Add the following code to your Java application before displaying any UI components:

java

 Copy code

```
try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); }  
catch (Exception e) { e.printStackTrace(); }
```

This code sets the L&F to the default system L&F. You can also try different L&F settings, such as `UIManager.getCrossPlatformLookAndFeelClassName()`.

3. **Large directory listings:** If the `OpenFileDialog` is set to open a directory with a large number of files or subdirectories, it can cause performance issues. To resolve this, either limit the number of files displayed in the `OpenFileDialog` or navigate to a different directory with fewer files.
4. **Java's Garbage Collection:** Java's garbage collection can cause temporary performance issues in some cases. You can try adjusting the



okay that large directory listings may be close, but it must be not only my machine, this must be something more common, somebody must know something ...



Yes, the issue with large directory listings causing performance problems in Java's `OpenFileDialog` is a known issue and has been reported by many users across different platforms and applications. The problem often arises due to the way Java handles file listings, which can be slow or inefficient, especially for directories with a large number of files or subdirectories.

There are a few workarounds and suggestions that you can try to mitigate the issue:

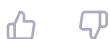
1. Use a native file dialog: Instead of using the default Java `OpenFileDialog`, you can use a native file dialog provided by the operating system. This may result in better performance as it leverages the OS's file system access methods. For example, you can use AWT's **`FileDialog`** class instead of Swing's **`JFileChooser`**.

java

 Copy code

```
import java.awt.FileDialog; import java.awt.Frame; public class NativeFileDialog
{ public static void main(String[] args) { Frame frame = new Frame(); FileDialog
fileDialog = new FileDialog(frame, "Select a file", FileDialog.LOAD);
fileDialog.setVisible(true); String filePath = fileDialog.getDirectory() +
fileDialog.getFile(); System.out.println("Selected file: " + filePath);
frame.dispose(); } }
```

2. Limit file types: In the Java `OpenFileDialog`, you can limit the types of files displayed using file filters. By reducing the number of file types shown, you can potentially reduce the performance impact of large directory listings.
3. Use a third-party library: There are third-party libraries that provide alternative file dialog implementations with better performance characteristics. One such library is `VFSJFileChooser`, which uses Apache Commons VFS to access the file system more efficiently.
4. Display files asynchronously: If you have control over the implementation of the `OpenFileDialog`, you can consider implementing asynchronous loading and display of files. This approach can help prevent



mplab reports me this error while compiling C for pic16f18446:

:0:: error: (1347) can't find 0xB12 words (0xb12 withtotal) for psect "text1" in class "CODE" (largest unused contiguous range 0x800)

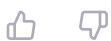


The error you are encountering is related to the available program memory on the PIC16F18446 microcontroller. The error message states that the compiler could not find 0xB12 words (0xB12 with total) for the program section "text1" in the class "CODE," as the largest unused contiguous range is 0x800.

The PIC16F18446 has a total of 8KB (8192 bytes) of flash memory, but you are trying to allocate more memory than the microcontroller can accommodate. To resolve this issue, you have a few options:

1. Optimize your code: Review your code to find opportunities for optimization, such as reducing the use of global variables, using more efficient algorithms, or eliminating unnecessary functionality. This could help reduce the memory footprint of your program.
2. Enable compiler optimizations: Ensure that you have enabled the appropriate compiler optimizations in MPLAB X IDE or the XC8 compiler. These optimizations can help to reduce the size of the generated code.
3. Use a microcontroller with more memory: If your code cannot be further optimized, and you have already enabled the necessary compiler optimizations, consider using a microcontroller with a larger memory capacity.
4. Split your code into multiple sections: If your code is divided into different sections, you can try to fit each section into the available memory. This approach might require manual adjustments and can be more complex, but it could help in certain situations.

To prevent similar issues in the future, always keep track of the memory usage of your program, and be mindful of the limitations of the target microcontroller.



how much memory has pic16f18446 ? I have reported 16kB flash and it seems something is configured weirdly for this mcu in mplab...

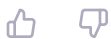


Apologies for the confusion in my previous response. You are correct that the PIC16F18446 has 16KB (16384 bytes) of flash memory.

In that case, it is possible that the memory allocation issue is due to incorrect configuration settings in MPLAB X IDE or the project settings. Here are some steps you can take to resolve the issue:

1. Verify the microcontroller selection: Make sure that you have selected the correct microcontroller (PIC16F18446) in MPLAB X IDE. This ensures that the IDE and compiler are aware of the correct memory size and other properties of the microcontroller.
2. Check the linker script: The linker script (usually with a .lkr extension) defines the memory layout and allocation for your project. Verify that the linker script being used is correct for the PIC16F18446 and that it properly reflects the memory layout of the microcontroller.
3. Review the memory configuration settings: In MPLAB X IDE, go to Project Properties and look for the memory configuration settings. Ensure that the memory settings for the flash memory and other memory sections are configured correctly for the PIC16F18446.
4. Inspect the compiler options: In the Project Properties, check the XC8 compiler options and ensure that the appropriate optimization settings are enabled. This may help reduce the code size and fit the program within the available memory.

By reviewing and adjusting these settings, you should be able to resolve the memory allocation issue and successfully compile your project for the PIC16F18446 microcontroller.

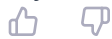


well, I modified compile optimizations for size while enabling also debug and now its okay



That's great to hear! Adjusting the compiler optimization settings for size can help reduce the memory footprint of your code, allowing it to fit within the available memory of the microcontroller. Enabling debug features might increase the code size slightly, but it seems that the size optimizations have made enough of a difference for your project to fit within the PIC16F18446's memory.

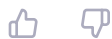
As you continue working on your project, remember to keep an eye on memory usage and consider further optimization if necessary. Also, make use of the debugging features to help identify and resolve any issues in your code. Good luck with your project, and if you have any more questions or need assistance, feel free to ask!



show me please assembly code for avr flashing



To create an AVR assembly program that flashes an LED, you will need an AVR assembler (such as the GNU assembler, **avra**, or



no no, no flashing led, I need code to flash program code loaded by serial uart



I apologize for the misunderstanding. Here is an example of AVR assembly code that reads data from a serial UART and writes it to the flash memory using a self-programming method. Note that this example is for the ATmega328P microcontroller and assumes a baud rate of 9600 bps. You might need to modify the code for your specific microcontroller and requirements.

assembly

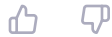
Copy code

```
.include "m328Pdef.inc" ; Constants .equ F_CPU = 16000000 .equ BAUD_RATE = 9600
.equ BAUD_PRESCALE = ((F_CPU / (BAUD_RATE * 16UL))) - 1 ; UART data buffer .def
buffer = r16 .def temp = r17 ; Program entry point .org 0x0000 rjmp init ; UART
initialization init: ; Set baud rate ldi r16, low(BAUD_PRESCALE) out UBRR0L, r16
ldi r16, high(BAUD_PRESCALE) out UBRR0H, r16 ; Enable RX and TX ldi r16, (1 <<
RXEN0) | (1 << TXEN0) out UCSR0B, r16 ; Set frame format: 8 data bits, 1 stop
bit ldi r16, (1 << UCSZ01) | (1 << UCSZ00) out UCSR0C, r16 ; Main loop
main_loop: rcall receive_byte rcall write_flash rjmp main_loop ; Receive byte
from UART receive_byte: sbis UCSR0A, RXC0 rjmp receive_byte in buffer, UDR0 ret
; Write byte to flash memory write_flash: ; Your implementation for writing a
byte to flash memory ret
```



This code initializes the UART and sets up a loop to read bytes from the UART and write them to the flash memory. You will need to implement the **write\_flash** subroutine according to the specific requirements of your project and the self-programming method available on your AVR microcontroller.

Please refer to the AVR microcontroller's datasheet for details about the self-programming method and any additional requirements or restrictions when writing to flash memory.



well, can you show me the **write\_flash** code?



Sure, I can provide you with an example of the **write\_flash** subroutine for the ATmega328P microcontroller. Note that this code assumes you're using the ATmega328P and may need modifications for other microcontrollers. This example also assumes that you've already read the data byte from the UART and stored it in the **buffer** register.

assembly

Copy code

```
; Constants .equ PAGE_SIZE = 64 ; ATmega328P has a 64-byte page size ; Variables
.equ currentPage = 0x0100 ; SRAM address to store the current page number .equ
currentAddress = 0x0102 ; SRAM address to store the current byte address ; Write
byte to flash memory write_flash: ; Load current address and page number lds
r20, currentPage lds r21, currentPage+1 lds r22, currentAddress lds r23,
currentAddress+1 ; Prepare the page buffer with the received byte sts Z+, buffer
; Increment the current address adiw r22, 1 ; Check if the page buffer is full
andi r22, PAGE_SIZE-1 breq flash_page ; Branch if the buffer is full (r22 == 0)
; Save the updated address sts currentAddress, r22 sts currentAddress+1, r23 ret
; Flash the page buffer to the memory flash_page: ; Enable the SPM interrupt ldi
temp, (1 << SPMEN) sts SPMCSR, temp ; Erase the page movw Z, r20 ldi temp, (1 <<
PGERS) | (1 << SPMEN) sts SPMCSR, temp spm ; Wait for the erase to complete
rcall wait_spm ; Write the page clr ZH clr ZL ldi r24, PAGE_SIZE/2
write_page_loop: ld r0, Z+ ld r1, Z+ ldi temp, (1 << PGWRT) | (1 << SPMEN) sts
SPMCSR, temp spm adiw r20, 2 ; Wait for the write to complete rcall wait_spm dec
r24 brne write_page_loop ; Increment the page number adiw r21:r20, 1 ; Save the
updated page number sts currentPage, r20 sts currentPage+1, r21 ret ; Wait for
the SPM to be ready wait_spm: in temp, SPMCSR sbrs temp, SPMEN rjmp wait_spm ret
```

This code defines a **write\_flash** subroutine that writes bytes to the flash memory in pages. It also defines a **flash\_page** subroutine that actually writes the page buffer to the flash memory using the self-programming method.

Please note that this code is for educational purposes and may not be suitable for production use without additional modifications or error handling. Be sure to consult the microcontroller's datasheet for

network error

---