SIGNUM SYSTEMS

© 2000 All rights reserved

USP-Z8 In-Circuit Emulator

Working with Banked Program Memory

SWII-Z8-Bank 1.00-11.10.00.11.43

PURPOSE: This document describes the Signum USP-Z8 in-circuit emulator's banked program memory mode of operation.

1. Hardware Requirements

To be able for a Signum Systems USP-Z8 ICE to take advantage of banked program memory, the emulator should be equipped with optional 128 KB or 256 KB of overlay External Program Memory. Note that in the standard configuration, the ICE is equipped with only 64 KB of overlay external program memory.

Additionally, the target's hardware must provide the emulator's POD with two more address bus lines (A16 and A17) as well as three bank switching signals (B0, B1 and B2).

The USP-Z8 supports 32 KB and 64 KB program memory bank schemes.

1.1 64 KB Program Memory Banks

This memory expansion requires two additional address bus lines: A16 and A17. These lines must be generated by the user's target hardware and connected, by means of jumpers, to the pins labeled "B0" (line A16) and "B1" (line A17) on the POD. Pin B2 must be grounded (Vss) and B3 pulled-up to +5V (Vcc), or left unconnected to enable this configuration.

The Program memory address space is configured as follows:

A16	A17			
B0	**B1**	**B2**	**B3**	**Program memory locations**
0	0	0	1	0 – 0x0FFFF
1	0	0	1	0x10000 – 0x1FFFF
0	1	0	1	0x20000 – 0x2FFFF
1	1	0	1	0x30000 – 0x3FFFF

If only 128 KB of the address space is used, the address line A17 (B1) must be connected to ground (Vss).

1.2 32 KB Program Memory Banks

This scheme allows up to 8 banks of memory, each consisting of 32 KB. To enable this memory model the pin labeled B3 (on the POD) must be shorted to ground (Vss).

To switch between the banks, three signal lines must be generated by user's target hardware, and connected to the pins labeled B0, B1, and B2 on the POD. These lines are defined to be active low. The first 32 KB (also called the root bank, root segment, or common segment code) is always present, and occupies locations 0 – 0x7FFF. Banks 0 through 6 (32 KB each) will be switched according to the bank select lines, and will occupy processor address space locations 0x8000H – 0xFFFF.

The Program memory address space is selected as follows:

A16	A17			
B0	**B1**	**B2**	**B3**	**Bank number**
0	0	0	0	Common code
1	0	0	0	Bank 0
0	1	0	0	Bank 1
1	1	0	0	Bank 2
0	0	1	0	Bank 3
1	0	1	0	Bank 4
0	1	1	0	Bank 5
1	1	1	0	Bank 6

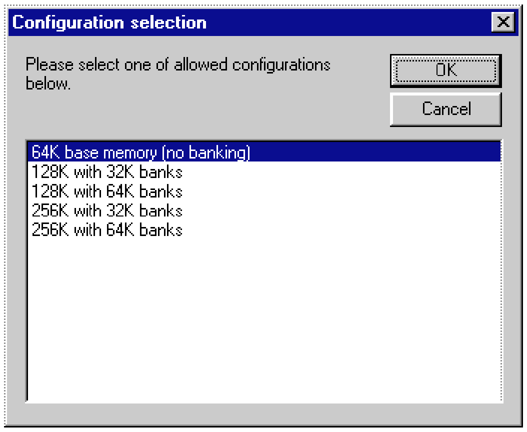
If only 128k of program memory is installed, pin B2 must be grounded.

2. Software Requirements

Chameleon debugger ver 2.65 for the Signum Systems USP-Z8 in-circuit emulator supports banked program memory models with memory sizes of 256 KB and 128 KB as well as bank sizes of 64 KB and 32 KB.

2.1 Banked Memory Model Selection

Selection of the banked memory model (64 KB or 32 KB program memory banks) is performed during the debugger’s **Add Target** action which creates a new debug target. The action can be launched from the **System Configuration** dialog box with the **Target Selection** property page selected by pressing the **Add Target** button of the dialog box. (The dialog box, in turn, can be opened by selecting **System Configuration** from the **View** menu). In the target creation phase, the user is required to select several target parameters, such as the processor to be emulated, physical interface to be used to connect the emulator to the host PC, or target configuration properties. The dialog box for setting the target configuration is similar to that in the illustration:



Please select one of allowed configurations below.

64K base memory (no banking)
128K with 32K banks
128K with 64K banks
256K with 32K banks
256K with 64K banks

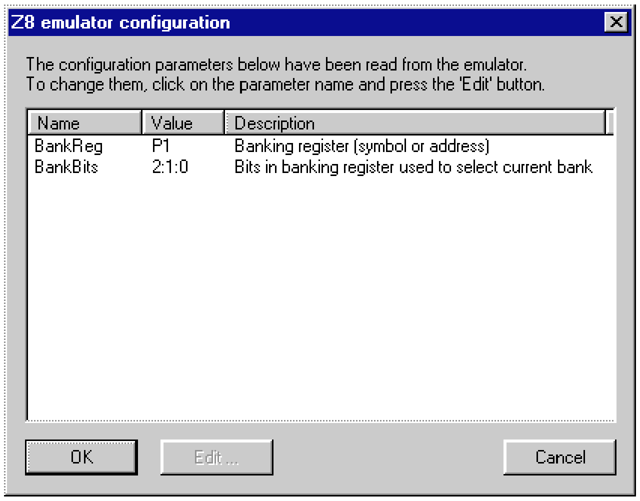
Additionally, then the banked memory model configuration is selected, the user must specify the parameters for that memory model.

2.2 Banked Memory Model Parameters

To allow the debugger to track the bank selection made by the program being debugged, two parameters must be specified for the debugger, BankReg and BankBits.

- **BankReg** tells the debugger which processor register, port, or memory location contains the current bank selection.
- **BankBits** tells the debugger which BankReg bits are used by the program for the current bank selection.

The values of the parameters are specified during the target creation phase after a banked memory model configuration was selected. The dialog box showing these parameters and their values may look as shown below:



The configuration parameters below have been read from the emulator.
To change them, click on the parameter name and press the 'Edit' button.

Name	Value	Description
BankReg	P1	Banking register (symbol or address)
BankBits	2:1:0	Bits in banking register used to select current bank

The value of a parameter can be edited by selecting the parameter name and clicking on the EDIT button.

The value of the **BankReg** parameter can be specified as the name of a processor register or port, as the name of a symbol, or as a plain address, e.g., (XDATA)0x4800. After that the debugger converts the value of this parameter to the plain address expression in the form “(memory-type)value” and stores the address in the internal variable **BankRegAddr**. The available memory types are Program, XDATA, GPR, and _SFR.

The value of the **BankBits** parameter is specified as two or three numbers between 0 and 7, separated by the colon “:”. The numbers are handled as numbers of bits in the memory location specified by the BankReg and BankRegAddr. These bits are used to select the current program memory bank. The numbers are evaluated from left to right with the leftmost number representing the most significant bit and the rightmost number representing the least significant bit. For example, the parameter value “1:2:0” designates bit 1 as the most significant bit for the bank selection and bit 0 as the least significant bit. When the parameter value is “2:1:0”, the same bits of the banking register are used to select the current bank, however, they are evaluated differently, bit 2 of the register being used as the most significant bit for the bank selection.

2.3 Preparing a Program for a Banked Memory Model Configuration

The program to be executed on a target equipped with banked program memory is responsible for correct bank switching. This program must ensure that the banking register specified by the **BankReg** parameter and the register bits specified for the debugger by the **BankBits** parameter are used. Therefore, the application program must be compiled and linked for the same banked memory model as that specified during the target creation process. If your development tools do not support banked memory, then the following procedure should be followed:

1. Grouped the program modules into sets which will be loaded into separate 32 KB or 64 KB memory areas (memory banks), depending on the selected banking model.

2. Individual program module sets must be compiled and linked into separate executable modules.

3. The special bank definition file needs to be written for the OBJCOMP converter utility. The name of the definition file will be used by the converter as the resulting user application name. The contents of the definition file will be used to allocate bank numbers to the appropriate modules created by the linker. The definition file is a plain text file. Each line of this file specifies the memory banks to which individual linked modules will be allocated. For example, for the 2500AD tools, the file BAPP.TXT may look like this:

   ```
   b0.sym     0
   b1.sym     1
   b2.sym     2
   ```

   In this example, module b0.sym will be loaded into bank 0, module b1.sym into bank 1 and module b2.sym into bank 2. The resulting application file will be named BAPP.IDB by the OBJCOMP converter.

4. The OBJCOMP converter utility has to be executed with the @<definition-file> parameter instead of a plain <input-module-name>. For example, for 2500AD tools, the invocation may look like this:

   ```
   OBJCOMP –F2500AD @BAPP.TXT
   ```

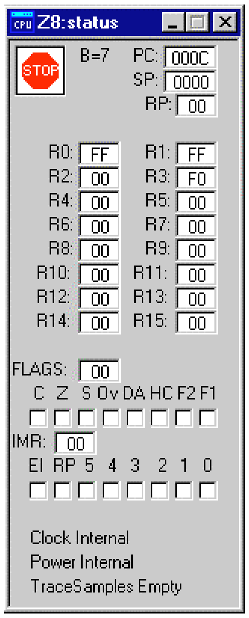
   The OBJCOMP converter creates a *.IDB file which contains the code as well as debug information necessary to load the application code into proper memory banks and show code symbols in the appropriate memory banks.

Thus prepared IDB file can be loaded to emulator’s banked memory from the LOAD window or using the LOAD command in the COMMAND window.

2.4 Debugging with Banked Memory Model Configuration

After the target creation phase is finished, the new target is added to the set of targets managed by the debugger. Some windows and commands behave somewhat differently when a banked memory configuration is active. The deviations include:

The STATUS window



```
B=7      PC: 000C
         SP: 0000
         RP: 00

R0:  FF   R1:  FF
R2:  00   R3:  F0
R4:  00   R5:  00
R6:  00   R7:  00
R8:  00   R9:  00
R10: 00   R11: 00
R12: 00   R13: 00
R14: 00   R15: 00

FLAGS: 00
C Z S Ov DA HC F2 F1
IMR: 00
EI RP 5 4 3 2 1 0

Clock Internal
Power Internal
TraceSamples Empty
```

The STATUS window displays the current bank number, set by the application program as the value of the **B** variable, between the CPU state icon and the PC register label.

Command Address Format

The address format for banked memory is

<bank-number> : <bank-offset>

For example,

0:0x9000	*Offset 0x9000 in bank 0.*
2:0x1234	*Offset 0x1234 in bank 2.*
6:0xA040	*Offset 0xA040 in bank 6.*

This standard address format can be used in any command that is passed a program memory address as its parameter, or in any window that expects a program memory address. For example, the breakpoints in the BREAK SET command, address expressions in the WATCH window, or the address range in the ADDRESS MATCH dialog box (Complex Events settings) can be specified in the banked memory address format.

Program Memory Mapping

Use the **MAP** command to map the appropriate areas of the program memory to the emulator's overlay memory (**ICE**) or user target board (**USER**). Consider the following examples.

MAP PROGRAM ALL ICE	*Maps the entire program memory to the emulator.*
MAP PROGRAM 1:0x8000 TO 1:0x8FFF USER	*Maps the memory area between 0x8000 and 0x8FFF in memory bank 1 to the user target board.*
MAP	*Displays the current memory mapping.*

Checking which Target Configuration was Selected

To display the set of so-called system variables, use the SYSVAR command. The command lists the variables associated with the selected configuration (system variable __CONFIGURATION__) as well as the target name, selected processor, PC interface, software version, and so on. An example of the SYSVAR command's listing is shown below.

```
  __VERSION__       = 265
  __TARGET_NAME__   = "Z8"
  __CPU_FAMILY__    = "Z8"
  __CPU_MODEL__     = "Z86C93"
  __EMULATOR__      = "USPZ8"
  __INTERFACE__     = "Serial"
  __CPU_RUNNING__   = 0
  __CPU_STATE__     = "Stopped"
  __CONFIGURATION__ = "256K_BNK32"
```

The value "256K_BNK32" of the __CONFIGURATION__ variable tells us that the target "Z8" is working with 256 KB program memory and 32 KB memory bank.

Checking and Modifying Banked Memory Model Parameters

The EMU command can be used to display the banked memory model parameters: BankReg, BankBits and BankRegAddr. EMU also lists other parameters describing the current target, as shown in the following example:

```
Name: Z8, Version: 1.10
Parameters:
  Processor=Z8
  WaitTime=0
  BinErrCnt=0
  BinSpeed=115200 bps
  BankReg=P1
  BankRegAddr=(_SFR)0x1
  BankBits=2:1:0
End of parameters
```

It is also possible to modify the current target parameters with the EMU command. The syntax is as follows.

```
EMU <parameter-name> = <new-value>
```

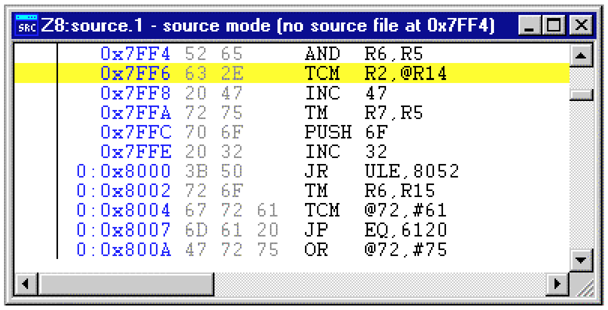
We strongly recommend to use this command solely for modifying the banked memory model parameters, that is, **BankReg** and **BankBits**, and only when the current setting is incorrect. For example:

EMU BankReg = P2	*Sets the banking register as the processor port P2.*
EMU BankBits = 4:3:2	*Sets bits 4 through 2 of the banking registers as the bits used to select the current bank.*

We strongly recommend **not** to modify any other parameters except for BankReg and BankBits. An incorrect setting can produce unpredictable results.

The SOURCE Window

The SOURCE window displays banked addresses in the <bank-number>:<offset> form in Mixed or Disassembly display modes. (The illustration below shows the boundary between a common code segment and bank 0 in the 32 KB banked memory model.) Using vertical scroll bar of the window, or the VIEW AT command, the whole program can be viewed in the SOURCE window.



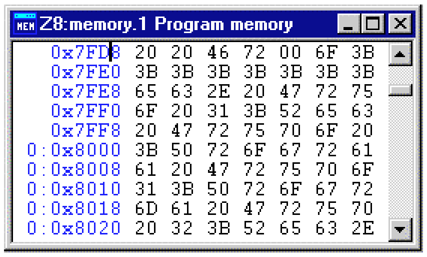
```
  0x7FF4 52 65     AND  R6,R5
  0x7FF6 63 2E     TCM  R2,@R14
  0x7FF8 20 47     INC  47
  0x7FFA 72 75     TM   R7,R5
  0x7FFC 70 6F     PUSH 6F
  0x7FFE 20 32     INC  32
0:0x8000 3B 50     JR   ULE,8052
0:0x8002 72 6F     TM   R6,R15
0:0x8004 67 72 61  TCM  @72,#61
0:0x8007 6D 61 20  JP   EQ,6120
0:0x800A 47 72 75  OR   @72,#75
```

Due to the limitations of the emulator, the memory contents is shown properly only when the program memory is mapped to the emulator's overlay memory (ICE).

When a memory area is mapped to USER only, the current bank area is displayed properly because now the emulator is not able to access other memory than the current memory bank.

The PROGRAM MEMORY Window

The PROGRAM MEMORY window displays banked addresses in the same format as the SOURCE window, i.e., <bank-number>:<offset>.



```
  0x7FD8 20 20 46 72 00 6F 3B
  0x7FE0 3B 3B 3B 3B 3B 3B 3B
  0x7FE8 65 63 2E 20 47 72 75
  0x7FF0 6F 20 31 3B 52 65 63
  0x7FF8 20 47 72 75 70 6F 20
0:0x8000 3B 50 72 6F 67 72 61
0:0x8008 61 20 47 72 75 70 6F
0:0x8010 31 3B 50 72 6F 67 72
0:0x8018 6D 61 20 47 72 75 70
0:0x8020 20 32 3B 52 65 63 2E
```

The PROGRAM MEMORY window is subject to the same limitations as the SOURCE window, thus it displays correctly only those memory areas that are mapped to ICE.

❑