Application Note, V1.0, Mar. 2007

AP32113

Connecting Memory to TC1130 External Bus Unit (EBU)

Microcontrollers

Never stop thinking

Edition 2006-02

Published by
Infineon Technologies AG
81726 München, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

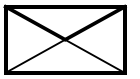
Connecting Memory to TC1130 External Bus Unit (EBU)

Revision History: V1.0, 2006-02

Previous Version(s):
none

Page	Subjects (major changes since last revision)

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com



1 Introduction

TC1130 micro controller offers an interface called External Bus Unit (EBU) to access various external memory devices. Programmable access parameters from asynchronous memory devices to Burst Flash and SDRAM devices makes this interface very flexible in interacting with memory devices. The EBU controls the transactions between external memories or peripheral units, and the internal memories and peripheral units. To connect another external memory device to EBU when a SDRAM device is already connected an intermediate buffer is required to synchronize the clock between Tricore EBU and device.

Note: TC1130 Triboard has 74ALVCH16245 buffer devices as it has 2 SDRAM and 2 Burst Flash devices on it.

This application note describes the way to access Burst Flash memory devices and SDRAM memory devices, which are widely used for code and data storage in high end industrial application. Below is the block diagram of the EBU and its signals used to connect external device.

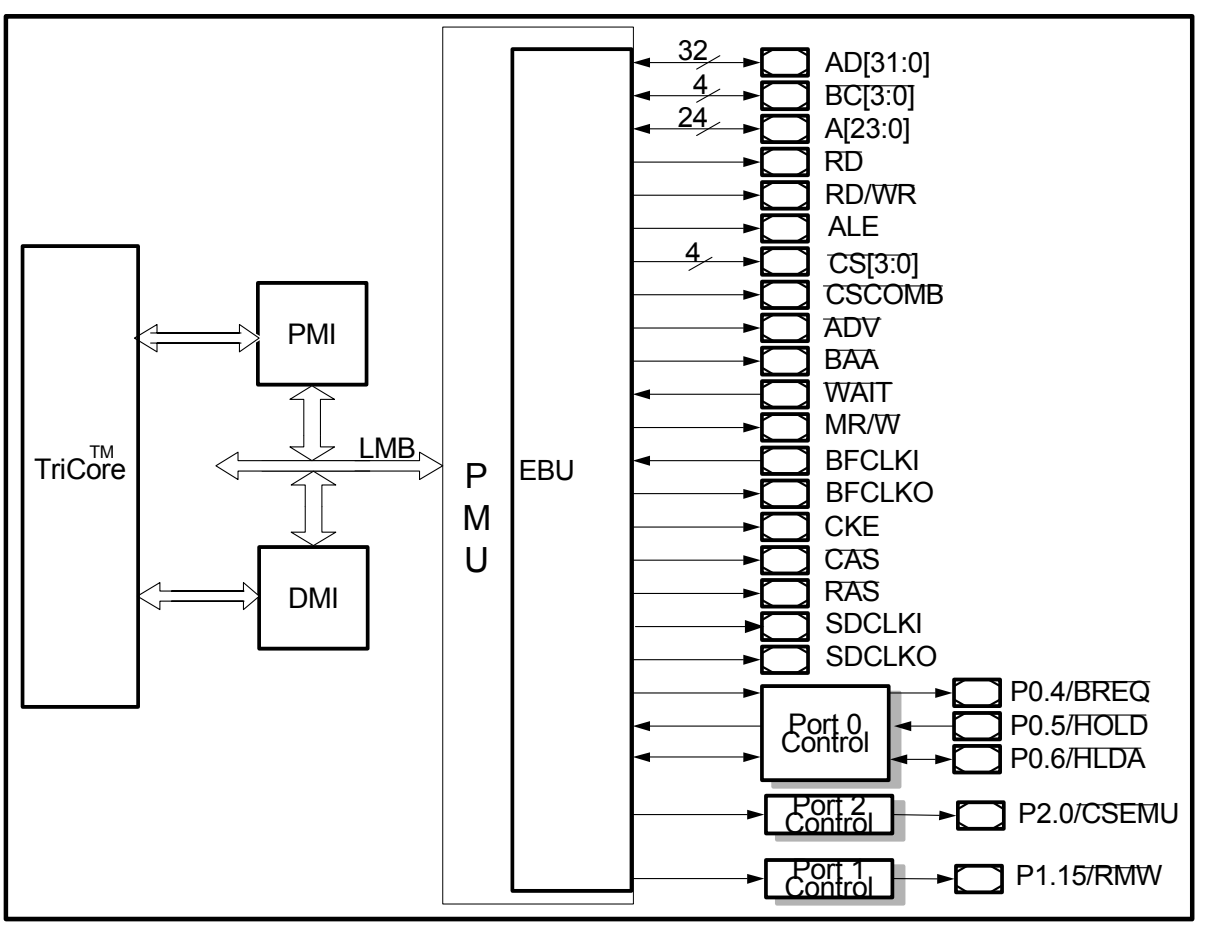


Figure 1 Interface Between EBU and External World

1.1 Device Selection Parameter

Address generation control [AGEN] parameter in Bus Configuration Register [BUSCON] has to be configured to select the type of external device connected over the TC1130 EBU. Below table lists various options available.

Table 1 BUSCON Register AGEN parameter

Registers	Setting	Function
BUSCON	0x00B0 0000	Demultiplexed access (default after reset)
	0x10B0 0000	Multiplexed access.
	0x20B0 0000	Burst FLASH type 0 access
	0x30B0 0000	SDRAM access type 0
	0x40B0 0000	SDRAM access type 1
	0x50B0 0000	Burst FLASH type 1 access.

Note: Refer to TC1130 user manual for register parameters in detail.

2 Burst Flash Memory Device

In this section, the way to program the corresponding registers for Burst Flash memory is described to achieve optimized performance and how to map corresponding parameters with values defined in respective memory device data sheet is given.If the flash device connected to the EBU needs to be programmed than it should be configured as an external peripheral, because the flash device is write protected when it is configured as burst flash in BUSCON.agen[30:28] parameter. In case of Burst Flash device the programming sequence should be followed strictly in order to access, this sequence will be specified in respective datasheets.Below diagram shows the interface diagram between EBU and Burst Flash.

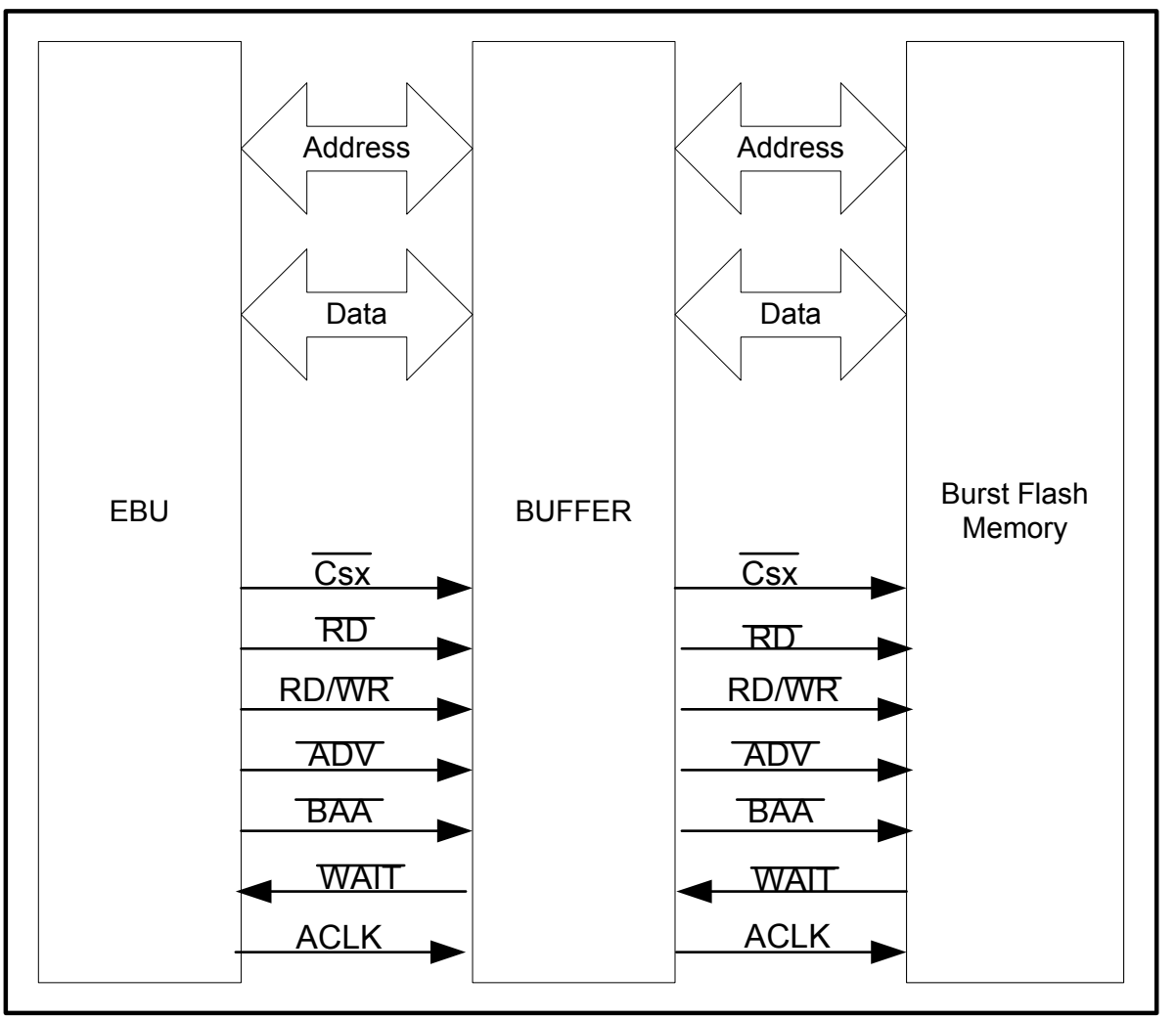


Figure 2 Interface Between EBU and External Burst Flash device

2.1 Supported Burst Flash Devices

In TC1130 the EBU supports the following burst mode FLASH devices:

- AMD Am29BL162 Burst Flash
- Intel 28F128K "Trumbull" Burst Flash
- Intel 28F256K "Trumbull" Burst Flash
- Intel 28F640K "Trumbull" Burst Flash
- Intel 28F800F3 and 28F160F3.
- STM M58BW016 Burst Flash
- STM M58LW064 Burst Flash

Note: This device list is based on data sheets available as of 10/2001. Some of these data sheets were in 'preliminary' status and as such are subject to change by the manufacturer. Such changes may lead to incompatibility with the EBU.

Note: Example settings for devices Intel 28F128K3C and STM M58BW016 *are provided in section* ***2.7****.*

2.2 Modes of Access

EBU supports five modes of Burst FLASH access cycle. These modes differ in the way in which the address is issued to the FLASH device. Selection of the appropriate Burst FLASH configuration is performed by programming the "ctype" and "portw" fields in BUSCON register as follows (note these settings only apply when the "BUSCON.agen" field specifies that the device connected to the appropriate chip select is a Burst FLASH device)

Table 2 Modes of Burst FLASH access cycle

BUSCON.ctype value	BUSCON.portw = 01 (16-bit)	BUSCON.portw = 10 (32-bit)
00	16-bit Non-Multiplexed	32-bit Non-Multiplexed
01	16-bit Multiplexed	32-bit Multiplexed
10	reserved	Twin 16-bit Multiplexed
11	reserved	reserved

2.3 Main Configuration Register - BFCON

Burst Flash Control Register (BFCON) is the main register to be configured when a Burst flash device is connected to the EBU.

BFCON
Ebu Burst Flash Control Register [Reset value: 0010 01D0H]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					DBA1	EBSE1	0			WAITFUNC1	FBBMSEL1	FETBLEN1			
r					rw	rw	r			rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTALTNCY				FDBKEN	DBA0	EBSE0	BFCMSEL	EXTCLOCK		WAITFUNC0	FBBMSEL0	FETBLEN0			
rw				rw	rw	rw	rw	rw		rw	rw	rw			

Note: Individual bit field descriptions are available in TC1130 user manual

Note: Detailed usage of each of the bit fields is mentioned in below sections

2.4 Features Common to All Modes

2.4.1 Configuration Register Settings

After device power-up or reset, Burst Flash memory defaults to asynchronous read configuration. Writing to the device's configuration register enables synchronous burst mode read operation. Configuration register sets the burst order, frequency configuration, burst length and other parameters. Below table lists an overview of configuration register.

Table 3 Configuration Register Definition

Bit Name (Intel)	Description	Intel F	AMD 29BDD	ST M58BW	ST M58LW	Functional Description
RM	Read mode	Bit 15	Bit 15	Bit 15	Bit 15	0 Synchronous Burst 1 Asynchronous
LC	First data latency	Bit [13:11]	Bit [13:10]	Bit [13:11]	Bit [13:11]	Clock cycles from address valid to first data[1)]
WT	Wait Polarity	n/a	n/a	n/a	n/a	0 Active low 1 Active high
	Internal Clock Divider	n/a	n/a	n/a	Bit 10	1 Divides internal clock

Table 3 Configuration Register Definition

Bit Name (Intel)	Description	Intel F	AMD 29BDD	ST M58BW	ST M58LW	Functional Description
DOC	Data Output Configuration	Bit 9	Bit 9	Bit 9	Bit 9	0 1 clock cycle 1 Data valid for 2 clock cycles
WC	Wait Configuration	Bit 8	Bit 8	Bit 8	Bit 8	0 Normal wait 1 Wait advance by 1 clock
BS	Burst Sequence	Bit 7	Bit 7	Bit 7	Bit 7	0 Interleaved 1 Sequential
CC	Clock Configuration	Bit 6	Bit 6	Bit 6	Bit 6	0 Falling edge clock 1 Rising edge clock
BW	Wrap Control	n/a	n/a	Bit 3	n/a	0 Wrapping Burst 1 No wrapping
BL2-0	Burst Length	Bit [2:0]	Bit [2:0]	Bit [2:0]	Bit [2:0]	Number of words per burst.

1) The definition is device dependent, details please refer to corresponding device data sheet.

For writing to configuration register of Burst Flash, different waveforms are required. The following registers need to be set:

- BUSCON.AGEN = 000_B, Asynchronous device type is selected.
- BUSCON.CTYPE may need to be set for multiplexed device configuration.
- BUSCON.PORTW need to be set according to external device port width.
- BUSAP.WAITWRC (write pulse width) and BUSAP.DATAC (data hold time) are set according to corresponding definition in device data sheet.

Normally the definition of parameters in BUSAP are based on the number of LMB clock cycles. So the following formula is applied in most cases:

The number of clock cycles = Specified value time / One clock cycle time

Note: If the result contains fractions, it is counted as a whole number.

For example, chip is running at 120MHz (8.33 ns/period), Intel 28F640K device is connected. According to data sheet of Intel 28F640K, the WE# write pulse width low is minimum 60ns. Based on the formula above, the number of clock cycles for write pulse width is 60/8.33 = 8 clock cycles. So the following setting should be done:

- BUSCON.MULTMAP = $1xxxxxx_B$
- BUSCON.CMULT = 001_B
- BUSAP.WAITWRC = 010_B

2.4.2 Programming Common Parameters

2.4.2.1 Burst Mode and Burst Length Setting

To match the page characteristics of a particular device, The burst flash controller can be programmed using the BFCON.FBBMSEL and BFCON.FETBLEN bit fields. BFCON.FETBLEN should be set to match the page length of the device if it cannot operate in continuous burst mode. If the device does support continuous bursts, best performance will be achieved by clearing BFCON.FBBMSEL to operate the controller in continuous mode.

If continuous mode is not used, the controller will interrupt the transaction when the programmed burst length is reached and issue a new address to the device. This will prevent devices that wrap accesses at the end of each page from returning incorrect data.

Note that, if the page length is correctly programmed into the controller and the device only asserts $\overline{\text{WAIT}}$ in the last data phase of a page, the controller can be operated without enabling $\overline{\text{WAIT}}$ monitoring with most devices. This only applies provided that BFCON.DBA is not used to disable the address alignment of transactions on the external bus and works because the controller will terminate the access at the end of each page to issue a new address to access the next page, rendering $\overline{\text{WAIT}}$ redundant.

2.4.2.2 Burst Address Wrapping Control

The BFCON.DBA bit is set to disable transaction alignment (disable burst address alignment). In normal use, the EBU aligns burst transactions so that the lowest addressed word is retrieved first. This is intended to eliminate unintended interaction between wrapping of bursts on the internal bus and burst wrapping implemented in burst flash devices. Transaction alignment can be disabled for reads by setting the BFCON.DBA bit. This passes the unaligned address directly onto the external bus. This can be used only in case where the device can be configured to match the wrapping expected by the internal bus for all possible opcodes.

2.4.2.3 $\overline{\text{BAA}}$ and $\overline{\text{ADV}}$ Signals Control

The controller will, by default, delay the two control outputs $\overline{\text{BAA}}$ and $\overline{\text{ADV}}$/ALE by a half clock period of the system clock. This is done because many devices use these signals synchronously and changing them on the active clock edge can cause violations of the device timing requirements. For a 100 MHz system clock, these signals will be delayed 5 ns from the rising clock edge of BFCLK. The control bit BFCON.EBSE can be set to disable this delay if it is not required.

2.4.2.4 Clock Feedback Enable

The sampling stages for flash read data (and $\overline{\text{WAIT}}$) can be set to use a clock fed back from the flash clock pin. This intended to increase operating speed by allowing a full period of BFCLK for data to propagate from the flash to the data inputs of the first sampling flip-flops. This does not allow use of an externally generated clock with burst flash devices. This additional operating margin comes at the expense of increased latency as the extra stages needed add two BFCLK cycles to the time required to access data.

Clock feedback is enabled by setting BFCON.FDBKEN and the necessary latency is programmed into the BFCON.DTALTNCY field. This must be 2*(n+1), where n is the value in BFCON.EXTCLOCK.

2.4.2.5 Device Type Selection

BUSCON.CTYPE can be used to support different ways of connecting address information to the burst flash. BUSCON.CTYPE=00_B is for standard device configurations with a separate address bus. When set to 01_B, the address is driven onto the data bus during the address and address hold phases of the access for those devices with a multiplexed address and data bus. If BUSCON.PORTW=10_B and BUSCON.CTYPE=10_B, the same 16 bit address is driven onto the upper and lower halves of the data bus during the address and address hold phases. This is to enable the use of 2, 16 bit wide, multiplexed devices to make a 32 bit wide memory region. If BUSCON.PORTW=01_B and BUSCON.CTYPE=11_B, then the 16 LSBs of the address are driven onto the upper 16 bits of the data bus for all phases of the access. This is a pin saving mode intended to allow the use of a 16 bit standard burst flash with 32 bit multiplexed devices without having to provide all the bits of a separate address bus. Note that in all the multiplexed addressing modes, the address is still driven onto the address bus pins for all phases of the access.

2.4.2.6 $\overline{\text{WAIT}}$ Signal Control

Without Wait

Operation without wait is configured by setting BUSCON.WAIT=00_B or BFCON.WAITFUNC set to “terminate burst” (1) and BUSCON.WAIT to 01_B or 10_B. Note that no devices have been found that operate successfully in “terminate burst” mode unless BUSCON.WAIT is set to 10_B. Also “terminate burst” is incompatible with clock feedback.

In this mode, the burst flash controller will generate the exact number of burst phases necessary to complete the requested access. To do this, it uses the opcode of the transaction and the port width of the memory region being accesses (defined by

BUSCON.PORTW). If necessary, it will split the transaction into multiple accesses as dictated by the BFCON.FBBMSEL bit and the amount of data required.

If “terminate burst” mode is enabled and $\overline{\text{WAIT}}$ is asserted during a data phase, the access will be terminated and a new address passed to the device. The new address will be aligned to the start of the next 32 word page in the device. This is intended to restart the access and fetch the remaining data from the next page in the device and is compatible with the AM29BL162C and other functionally similar devices.

With Wait

Operation with wait is configured by setting BFCON.WAITFUNC set to “extend access” (0) and BUSCON.WAIT to 01_B or 10_B. If $\overline{\text{WAIT}}$ is asserted by the device simultaneously with invalid data use BUSCON.WAIT=10_B. If $\overline{\text{WAIT}}$ is asserted one BFCLK before invalid data, use BUSCON.WAIT=01_B to take advantage of the additional synchronization stage.

In this mode the functions of the controller are split in two. On one side, it continues to generate the access phases but on the other hand it monitors the validity of the returned data samples at the outputs of the sampling stages. If the data is valid, the sample counter is decremented and the updating of the read buffer is enabled. The access is terminated when the sample counter reaches zero. A new address phase is issued when the number of valid samples retrieved is a n integer multiple of the page length programmed in BFCON.FETBLEN (if the controller is not configured for continuous operation). The sampling stages impose a resynchronization delay on the data returned by the flash device so the controller will run one or more additional data phases on the bus before the access can be terminated.

This mode of operation is designed to be compatible with clock feedback as it can cope with a variable transport delay through the sampling stages.

2.5 Clock Guidelines

The frequency of the clock used to drive burst flash devices can be configured using the BFCON.EXTCLOCK bit fields. Use of an externally generated clock is not supported.

The burst flash clock is derived from the system clock of the burst flash controller. Four modes are available, 1:1, 2:1, 3:1 & 4:1.

Clock frequency should be selected so that the device maximum clock frequency is not violated and so that data clocked out of the device (including $\overline{\text{WAIT}}$) has sufficient time to be captured by the EBU on the next rising clock edge. Burst flash clock feedback can be used to increase margin by clocking the first capture stage with a clock fed back from that at the burst flash clock pin. This eliminates the pad output delay from the timing arc.

Some devices, can be programmed to hold each data word valid for two clock cycles, this can enable a higher clock frequency to be used but does not necessarily increase data bandwidth. Be aware that the maximum clock frequency specified for some devices

requires data to be read every second clock cycle and this can seriously skew estimates of available bus bandwidth.

The controller also implements a power saving mode which disables the burst flash clock between accesses. This is also needed for booting from devices which automatically switch to synchronous mode when a clock edge is detected. Therefore the power saving mode defaults to on after reset and can be disabled by clearing BFCON.BFCMSEL. This will give an increase in performance by eliminating the clock cycles required to enable the clock at the start of each access.

2.6 Phases of Access

In current EBU, burst read accesses to Burst Flash devices are composed of a number of “Standard Access Phases”.

2.6.1 Address Phase

The length of this phase is set by BUSAP.ADDRC and can be multiplied by the BUSCON.CMULT field value by setting bit 0 of BUSCON.MULTMAP. During the address phase the $\overline{ADV}$ and $\overline{ALE}$ signals are active and the access address is driven onto the appropriate pins. This is a mandatory phase of any access and will therefore take place even if BUSAP.ADDRC is set to 00_B. The start of the Address phase is always synchronized to a rising edge of the BFCLK output.

2.6.2 Address Hold Phase

The length of this phase is set by BUSAP.AHOLDC and can be multiplied by the BUSCON.CMULT field value by setting bit 1 of BUSCON.MULTMAP. It has no active control signals but, for multiplexed devices, maintains the address on the AD bus. This is used to avoid hold violations when latching the address into the targeted device. The address is always driven onto dedicated address pins.

2.6.3 Command Delay Phase

The length of this phase is set by BUSAP.CMDDELAY and can be multiplied by the BUSCON.CMULT field value by setting bit 2 of BUSCON.MULTMAP. The phase is optional and is primarily intended for use with multiplexed devices as it allows a delay between the removal of the address from the AD bus and the assertion of $\overline{RD}$ in the Command Wait phase which will enable the drivers of the burst flash device. It can also be used to resynchronize the start of the Command Wait phase to BFCLK, if necessary, as it causes no action in the targeted device.

2.6.4 Command Wait Phase

The length of this phase is set by BUSAP.WAITRDC and is always multiplied by the BUSCON.CMULT field value. This is a mandatory phase and will always be at least one clock cycle long. During this phase $\overline{RD}$ will be driven. The transition to the next phase (Burst phase) will always occur on a rising edge of BFCLK.

For devices which assert $\overline{WAIT}$ during the initial access delay, the value for this phase should be set so that at least one Burst phase occurs before the $\overline{WAIT}$ input is cleared. This eliminates the possibility of missing the first data sample. Otherwise, this phase should end on the same clock edge as the flash device uses to clock the first read data onto the data bus.

2.6.5 Burst Phase

The length of this phase is set by BUSAP.BURSTC and can be multiplied by the BUSCON.CMULT field value by setting bit 3 of BUSCON.MULTMAP. This phase must programmed to be an integer number of BFCLK cycles long. Read data sampling occurs at the rising clock edge at the end of this phase. For most devices, this phase will be set to one BFCLK cycle. However, those devices which can return data every second clock cycle, will require a phase length of two BFCLK cycles if programmed in this way. This phase maintains the $\overline{RD}$ output low and also sets $\overline{BAA}$. $\overline{BAA}$ (burst address advance) is for those devices which use a control signal to determine when to update read data.

2.6.6 Recovery Phase

The length of this phase is set by BUSAP.RDRECOVC and can be multiplied by the BUSCON.CMULT field value by setting bit 5 of BUSCON.MULTMAP. It is also controlled by BUSAP.DTACS and BUSAP.DTARDWR. These fields are always multiplied by BUSCON.CMULT. The delay values are loaded into three timers and the Recovery phase ends when all three timers count down to zero. The DTACS timer is forced to zero immediately if the next transaction is to the same flash device. The DTARDWR timer is forced to zero immediately if the next transaction is a read.

This cycle is intended to enforce a delay between accesses on the external bus and all control signals are inactive.

2.6.7 Unused Phases

2.6.7.1 Data Hold Phase

The data hold phase applies to writes only and is therefore not used by the Burst Flash Controller.

2.6.8 Unused Parameters

BUSAP.WAITWRC, BUSAP.DATAC and BUSAP.WRRECOVC are not used by the Burst Flash Controller.

BUSCON.WRITE has no effect as regions with the BUSCON.AGEN field set to one of the burst flash values will always be write protected.

2.7 Register settings for various Burst Flash Devices

2.7.1 STM M58BW016 Burst Flash Device

Size of this device is 16Mbit (512kbx32). Max device freq is 56 MHz.

Table 4 Register settings

Register name	Value to be set	Description
BUSAPx[1)]	0xC1C41800	96Mhz LMB clock external memory wait cycle mode = 1
	0xC2C41800	96Mhz LMB clock external memory wait cycle mode = 3
BUSCONx	0x20900000	Flash type 0 32 bit, Demux, No multiplier for all phases
BFCON	0x00004883	150Mhz LMB clock Latency Cycle Control: 2*4 No Delay in ADV BFClk = 1/3 of LMBClock
	0x00000853	96Mhz LMB clock Latency Cycle Control: 2*4 No Delay in ADV BFClk = 1/3 of LMBClock

1) This Particular settings are valid for Burst Flash Register configuration value 0x29C2 [150Mhz LMB clock] or 0x2382 [96Mhz LMB clock]

Note: Values specified in the above table are verified on the simulation environment

2.7.2 Intel 28F128K3C Burst Flash Device

Size of this device is 16Mbit (1 M x 16-bit). Max device freq is 40 MHz.

Table 5 Register settings

Register name	Value to be set	Description
BUSCONx	0x2090207F	Flash type 0, 32 bit, Demux 4 * multiplier for all phases
BUSAPx[1)]	0x84410000	96Mhz LMB clock external memory wait cycle mode = 1
	0x82410000	96Mhz LMB clock external memory wait cycle mode = 3
BFCON	0x00C308C3	150Mhz LMB clock Latency Cycle Control: 2*4 No Delay in ADV BFClk = 1/4 of LMBClock

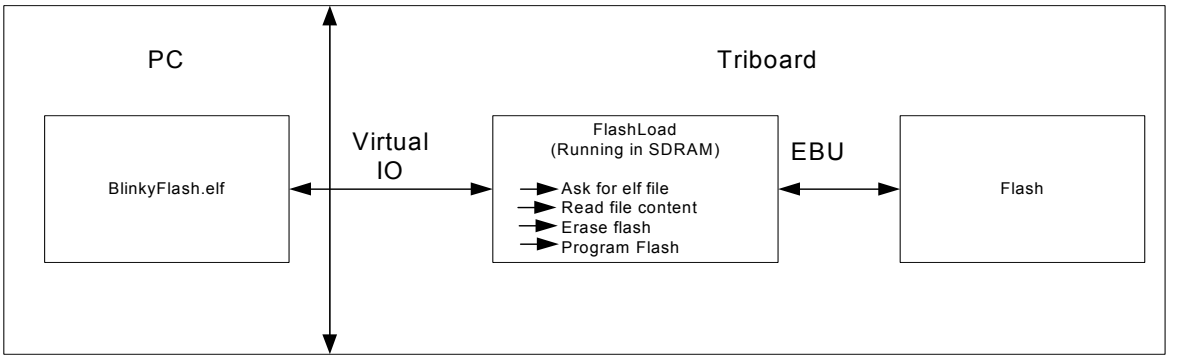
1) This Particular settings are valid for Burst Flash Register configuration value 0x19C7

Note: Values specified in the above table are verified on the simulation environment

2.8 Example to Program the Burst Flash device present on TC1130 Triboard

This section shows an example to program the 28F128K3C Burst Flash device present on the TC1130 Triboard with a simple LED blinking program. Specific flash programming algorithm is required to program any particular BF device, this algorithm has to be generated based on the information provided in the device respective data sheet. Information like how to erase, program and verify the BF device plays very crucial role. There are 2 aspects in interfacing a BF device to the EBU, first one is load a Flash algorithm in to SDRAM for programming the BF device and the second is loading the application [LED blinking program in this case] program in to BF device. How ever flash programing algorithm should be generated in accordance with data sheet of flash device.Here we concentrate on the program to program the flash device available on TC1130 Triboard with the Blinky application program

The TC1130 Triboard has 2 BF devices [28F128K3C] and 2 SDRAM devices [HYB39S256160-8] on it, in this example we will program in such way that the code will be stored in to BF and Data in the SDRAM.Below block diagram gives an idea on Burst flash programming concept.



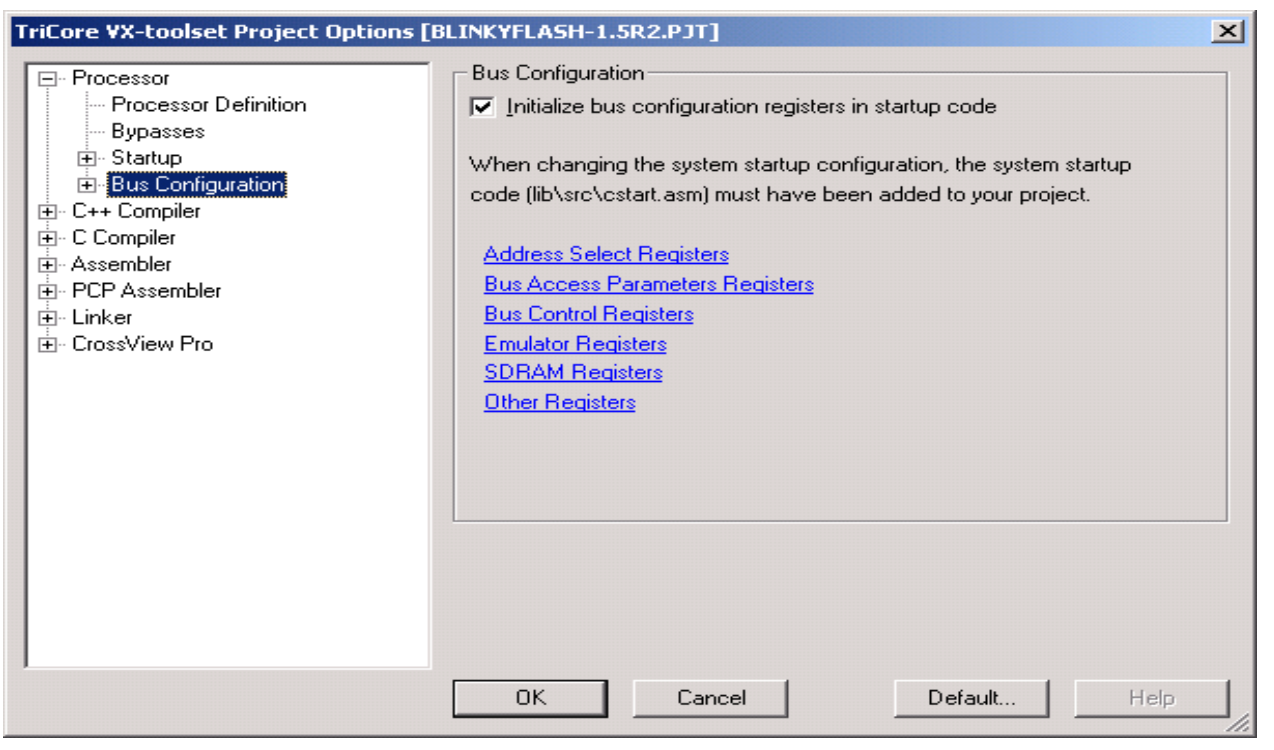
Flash Load should perform the following tasks [as shown in the diagram above]:

- Configure the flash device & ask for the application elf file, in this case it is 'BlinkyFlash.elf'
- Read the content of file
- Erase the flash
- Program the flash

Steps to program the BF device

1. Blinky Program remains same as discussed section 3.8
2. Linker settings needs to be modified to store the code in BF memory area

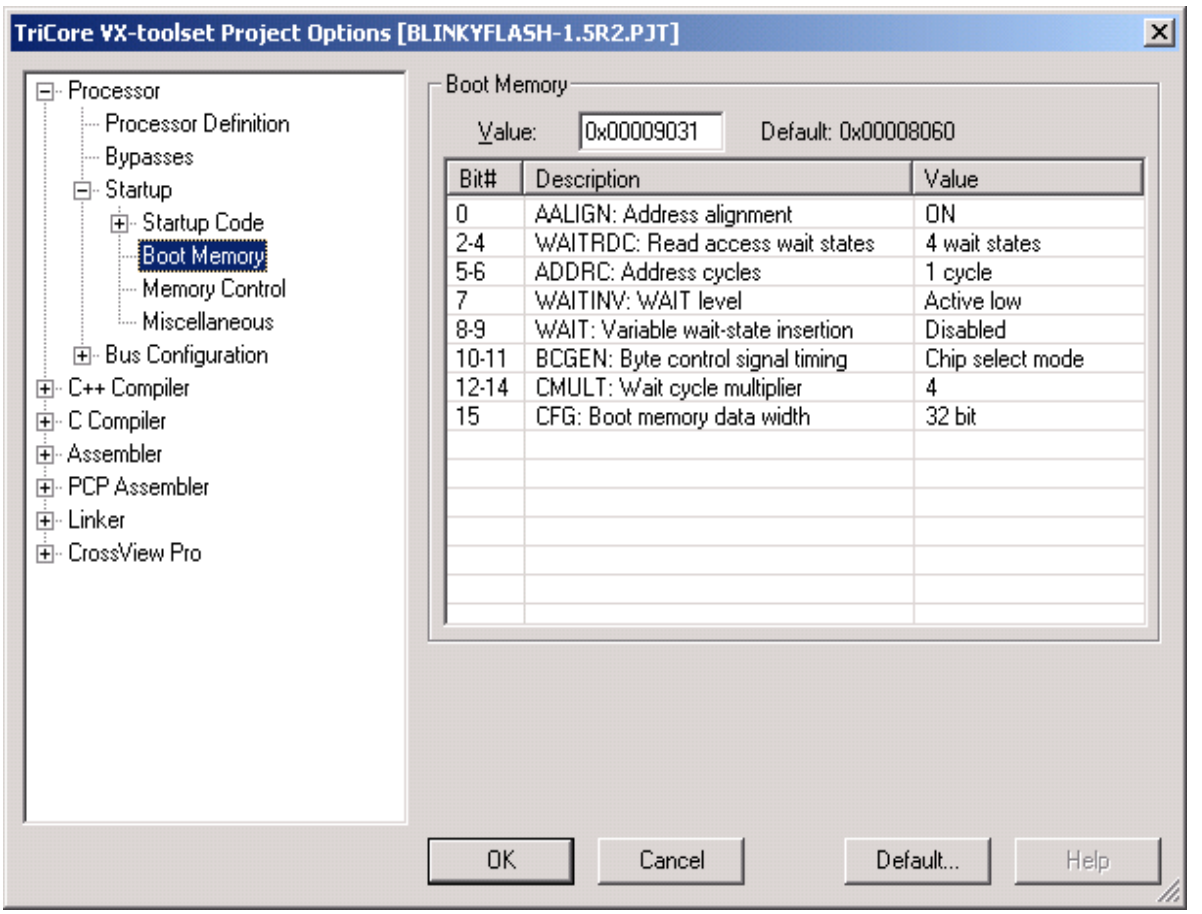
- Change the execution environment to Simulator from 'Triboard (TC1130) with SDRAM'
- Enable the Bus Configuration



- Modify the BootMemory value:

BootMemory = 0x00009031

This value is very important as it is the first word the CPU will read after a reset, it determines which configuration the EBU will use for starting-up the code.



- Modify the ADDRSEL registers:

ADDRSEL0 = 0xA0000021 ADDRSEL1 = 0xA4000011

- Modify the BUSAP registers:

BUSAP0 =0xC7230000 BUSAP1 = 0x42000000

- Modify the BUSCON registers:

BUSCON0 = 0x00922300 BUSCON1 = 0x30B20000

- Modify the SDRAM registers:

SDRMREF0 = 0x000000D7 SDRMCON0 = 0x019E2075

SDRMMOD0 = 0x00000023

- Modify the EBU registers:

EBUCON = 0x00F9FF68

3. Now the code is ready to compile.
4. Compile the BlinkyFlash program and copy the elf file to appropriate directory (obj) in Flash load Program. This FlashLoad Program is used to load the blinky application in to Burst flash device.
5. Load the FlashLoad program using the debugger and run it.

Note: This should run the FlashLoad program from SDRAM, starting at address 0xA4000000. Indeed, the Flash needs to be located at 0xA0000000.

6. Launch the CrossViewPro debugger and run, It will open a Terminal window: FSS0. This is the link between the Tricore and your hard-drive where the 'BlinkyFlash.elf' file is stored.

Note: you might have to change the execution environment configuration file if you have installed Tasking somewhere else than the default location: 'c:\program files\tasking\ctc v2.0\etc\infineon_triboard_tc1130.cfg'

Note: FlashLoad program should be written in such way to ask for the application elf file as described above

7. In the 'Terminal' window, enter 'obj\BlinkyFlash.elf' (without quotes), If the File and your Flash are recognized, the program first erases the Flash and then programs it.
8. Put the OCDS Jumper (close to the serial port) to Position 2

Note: Refer to the Triboard Manual for details

9. Change the Jumper Settings to 'OFF-OFF-ON'

Note: This is the normal setting for external boot from 0xA0000000

10. You can press the Reset Button

Note: Once again you can see that the LED is blinking slower than when it was running from SDRAM. The Flash is indeed a slower device than an SDRAM. The EBU parameters used here are really basic. Those parameters need to be optimized by looking deeper in the Flash device datasheet.

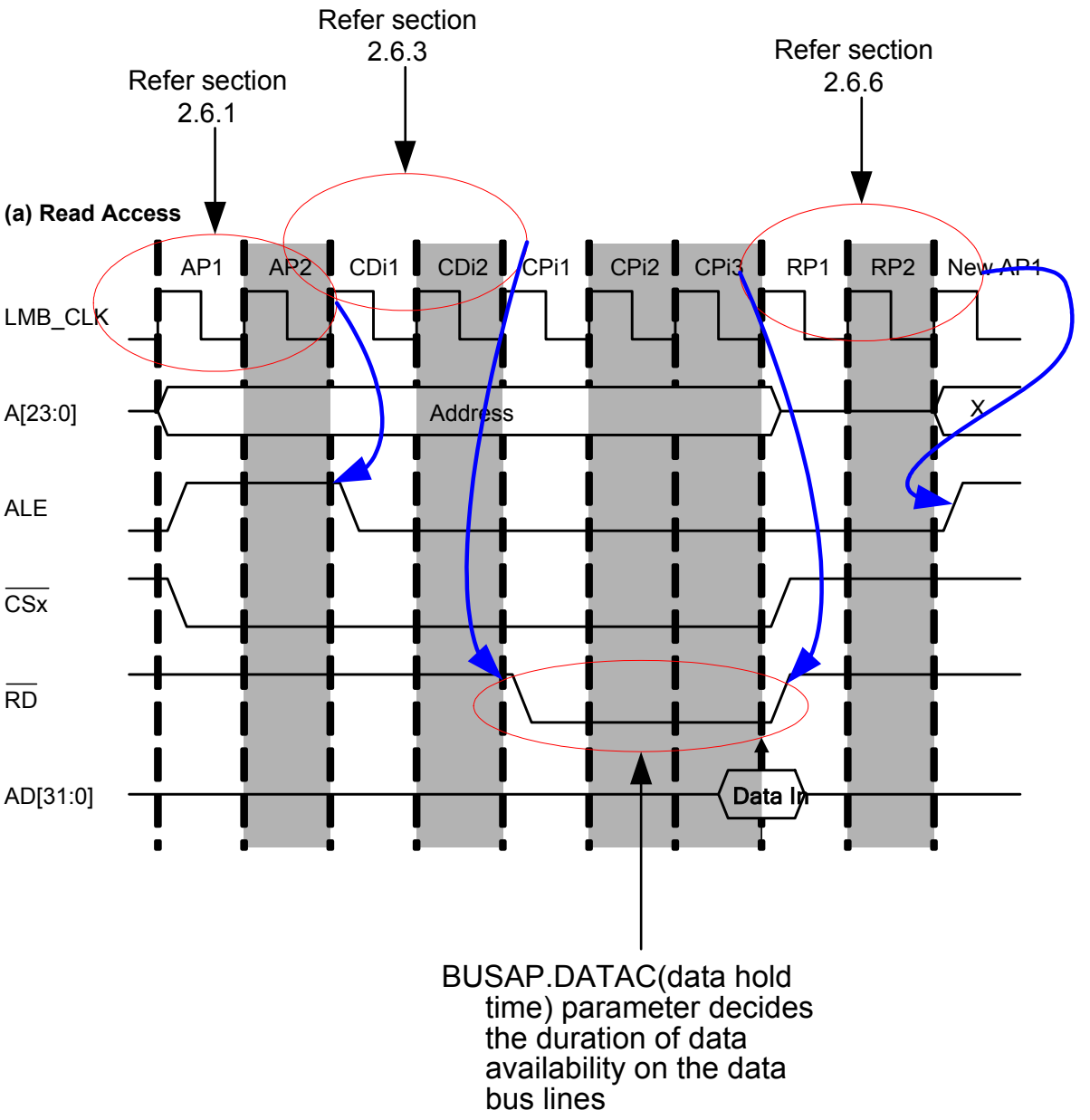
2.8.1 Bus Access Timings for Interfacing Intel BF device to TC1130 EBU

As explained earlier all major timings parameters of EBU can be programmed as to meet the external device specifications, BUSCON.MULTMAP parameter is used to decide for which time component a multiplier is used.

Below figures show an example of accessing an Intel-Style demultiplexed device for both read and write accesses. This shows the insertion of delay cycles (shown shaded) to adjust the access cycle to the device’s timing requirements. Both read and write accesses begin with a two cycle Address Phase followed by a two cycle Command Delay Phase.

For the read access, the Command Delay Phase is followed by a three cycle Command Phase. At the end of the Command Phase, the data is read (latched) by the EBU. A one cycle Recovery Phase is inserted at the end of the cycle. At the start of this Recovery Phase all control signals return to their non-active levels.

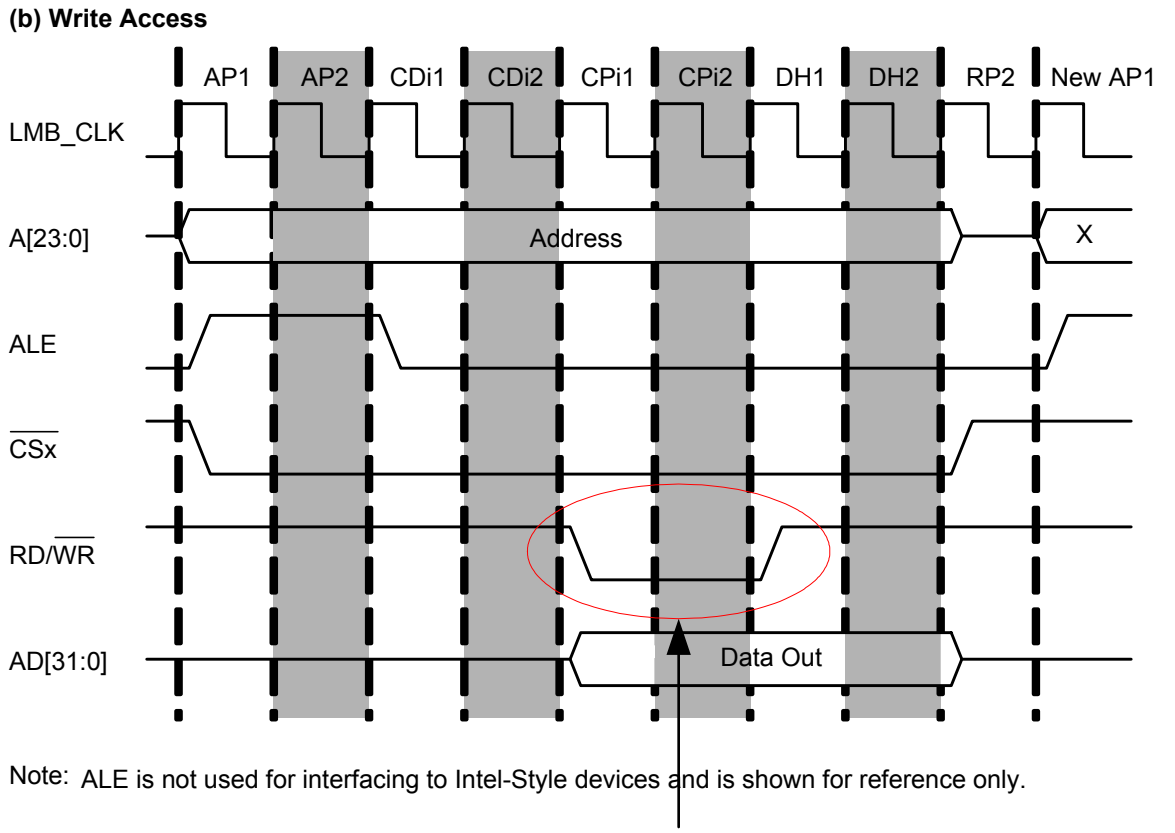
Figure 3 Example of an Intel-style Demultiplexed Device Read Access



For the write access, the Command Delay Phase is followed by a two-cycle Command Phase. During a write access it is possible to insert a Data Hold Phase to satisfy the data hold time requirements of the device. In the example the Data Hold Phase consists of

two cycles. During the Data Hold Phase, the RD/WR control signal is driven to the non-active state but the data and address are still driven on the bus.

Figure 4 Example of an Intel-style Demultiplexed Device Write Access



3 SDRAM Memory Device

In this section, the way to program the corresponding registers for SDRAM memory is described and the detailed settings for TC1130 TriBoard are given. EBU can simultaneously support two types of SDRAM, each having different access/refresh parameters. An additional buffer is required for slower external devices when SDRAM is connected. Below diagram shows the simple interface diagram for EBU and SDRAM.

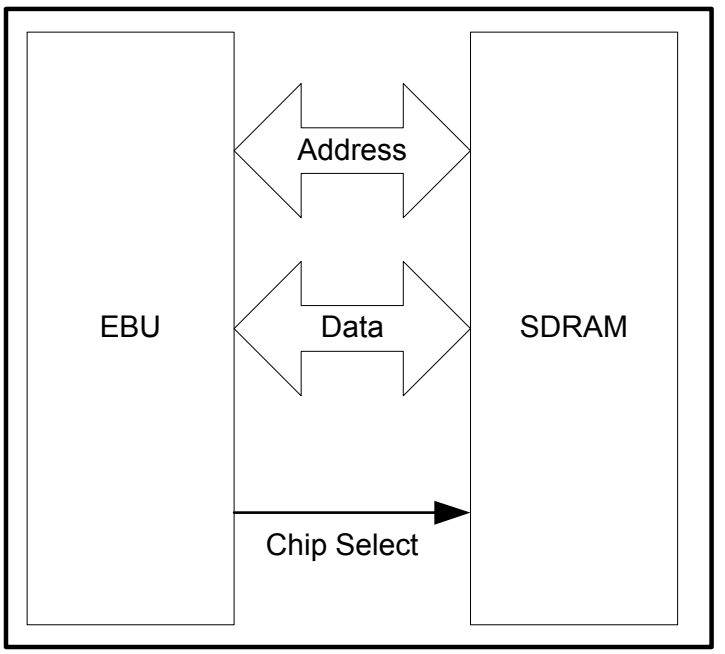


Figure 5 Interface Between EBU and External SDRAM device

3.1 Supported SDRAM Devices

The supported SDRAM devices include (but not limited to) the following:

- Infineon, HYB39S16160, HYB39S256160
- Samsung, KM416S1020
- Micron, MT48LC2M32, MT48LCM4M16, MT48LC16M16
- Hyundai, HY57V161610, HY57V651620

Note: This device list is based on data sheets available as of 10/2001. Some of these data sheets were in 'preliminary' status and as such are subject to change by the manufacturer. Such changes may lead to incompatibility with the EBU

3.2 Supported SDRAM Commands

The following commands are issued by the EBU while talking to SDRAM devices.

- Device deselected (**DSEL**)

- No Operation (**NOP**)
- Bank Activate (**ACT**)
- Read without Auto precharge (**READ**)
- Write without Auto precharge (**WRITE**)
- Precharge select bank (**PRE**)
- Precharge all banks (**PALL**)
- Auto Refresh (**AR**)
- Self Refresh Entry (**SLFRSH**)
- Self Refresh Exit (**SLFRSHX**)
- Mode Register Set (**MRS**)
- Extended Mode Register Set (**EMRS**) (for DDR SDRAM only)

3.3 Initialization Sequence

Proper Initializing sequence must be followed in order to communicate with SDRAM. Following is the Initialization sequence for any type of SDRAM connected over TC1130 EBU:

11. At least one NOP cycle must be issued after 1 ms of CS inactive (device deselect).
12. First step must be followed by 200 µs pause by software and a Precharge All Banks command.
13. After second step, the device must go through Auto Refresh Cycles (the number of refresh commands is programmable through CRFSH in SDRMCON[1:0] registers and the number of NOP cycles in between is programmable through CRC).
14. At the end of third step, the Mode Register must be programmed through the address lines. Following that some number of NOP cycles programmable through CRSC in SDRMCON[1:0] registers.

This sequence must be carried out for each type of SDRAM.The sequence is triggered by writing to the SDRAM mode register SDRMOD0 or SDRMOD1. All the regions having AGEN in BUSCON[3:0] set to ‘011’ will be configured with the mode from SDRMOD0, while regions with AGEN equals ‘100’ configured from SDRMOD1. While this sequence is being executed, SDRMBUSY flag in the SDRSTAT[1:0] status register will be set accordingly.

The setting sequence of corresponding registers is critical. Especially the SDRMOD[1:0] registers, they must be the last one to set because this write will trigger the initialization sequence. The recommended sequence of setting registers is as follows:

1. EBUCON
2. All other EBU registers except SDRAM specific registers
3. SDRMCON0
4. SDRMMOD0
5. SDRMREF0

6. SDRMCON1
7. SDRMMOD1
8. SDRMREF1

The EBU uses the CAS latency value and burst length to adjust the burst read timing. All other fields have no influence on the EBU, which means only single value is accepted for those fields.

The entire initialization sequence will be issued only on the first write (since reset) to the relevant SDRMOD register. On subsequent writes, the SDRAM device does not need to be initialized, so a simple mode register set command can be issued. A precharge-all command needs to be issued to the SDRAM before this can happen.

3.4 Main SDRAM Configuration Registers

3.4.1 SDRMREFx

SDRAM REFRESH REGISTER **[Reset value: 0000'0000$_H$]**

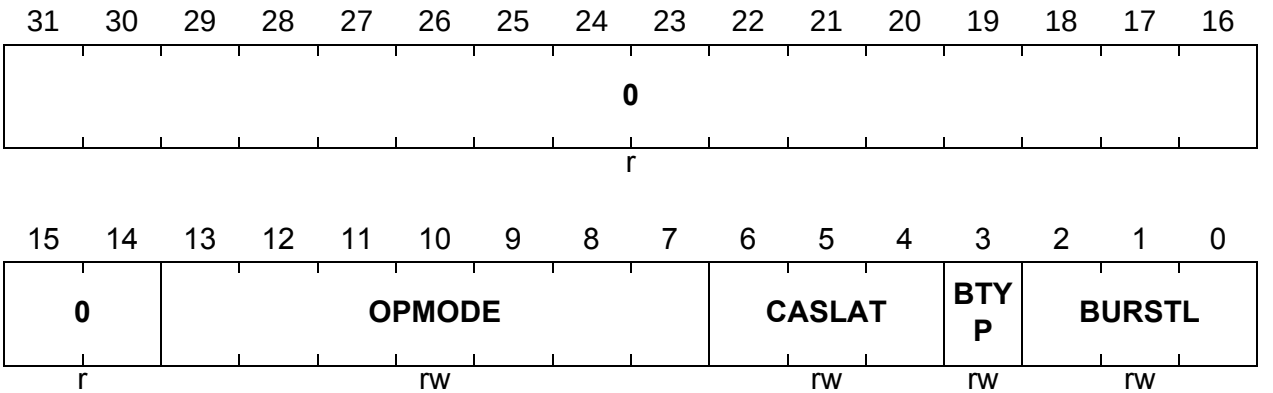
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		AUTOSELFR	SELFREN	SELFRNST	SELFREX	SELFREXST	REFRESHR			REFREHC					
r		rw	rw	r	rw	w	rw			rw					

*Note: Required bit fields are described In detail in section **3.5***

3.4.2 SDRMODx

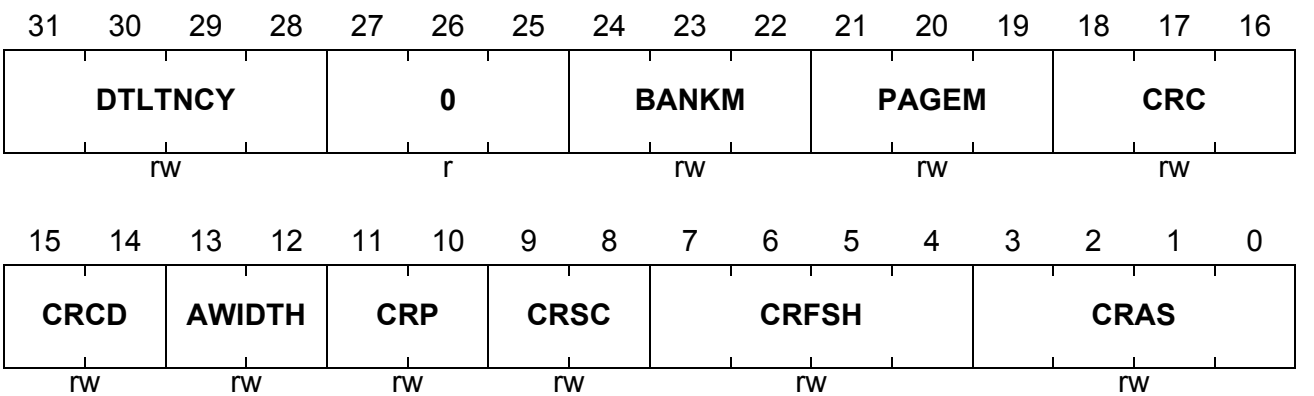
SDRAM MODE REGISTER **[Reset value: 0000'0000$_H$]**



*Note: Required bit fields are described In detail in section **3.5***

3.4.3 SDRMCONx

SDRAM CONTROL REGISTER **[Reset value: 0000'0000$_H$]**



*Note: Required bit fields are described In detail in section **3.5***

3.5 Features Common to All Modes

3.5.1 Refresh Cycles

SDRAM Refresh register is mainly used for configuring the various refresh cycle parameters for regular SDRAM refreshing. Number of clock cycles between refresh operations will be set by using SDRMREF.REFRESHC field. SDRMREF.SELFREN and SDRMREF.SELFREXST parameters will be used to determine whether self refresh entry and exit commands should be issued to all the connected devices, when set to 1 self refresh entry and exit commands are issued to all types devices. When a refresh is due the number of additional refresh commands should be issued is configured using

SDRMREF.REFRESHR parameter. When SDRAM gives up control on the external bus automatic self refresh entry command will be issued if SDRMREF.AUTOSELFC bit is set and this bit also enables SDRAM to automatically exit self refresh mode when it regains control on external bus.

3.5.2 Row Access Configurations

Number of clock cycles between row activate and the following precharge command is set by SDRMCON.CRAS parameter. To set the number of NOP cycles inserted after a precharge command SDRMCON.CRP field is used. Due to CAS latency and burst length the actual number performed can be greater. SDRMCON.CRCD parameter sets number of NOP cycles between a row address and a column address in order to insert delay between row to column access. For setting number of NOP cycles between refresh commands to be issued in the power-up initialization sequence use SDRMCON.CRC field.

3.5.3 Column Access Configurations

Based on the PORTW field in BUSCON register the number of address bits that has to be used for column address from bit 0 is set using SDRMCON.AWIDTH field.

3.5.4 Mode Register set up

SDRMCON.CRSC field sets the number of NOP cycles to be executed after a mode register set command. Mode register set command is issued to configure SDRMOD register during initialization.

3.5.5 Bank & Page memory Masking

To address the SDRAM memory in right manner, proper configuration should be done on the EBU side based on the type of external RAM connected to it, for this purpose SDRMCON.PAGEM and SDRMCON.BANKM fields are used to define number of address bits from bit 26 to be used for comparing page and bank tags.

3.5.6 Burst Access & Operation mode

EBU can be configured to generate burst lengths of either one or eight via the SDRMOD[1:0].BURSTL bit fields. When configured for a burst length of eight the interface will use data masking to support shorter write accesses. However, when configured for a burst length of one data masking is not used. Shorter write access should be used when faster execution speed is required. EBU supports only standard burst write operation mode this is set using SDRMOD.OPMODE field.

3.5.7 Latency settings

Latency is the delay/wait time between the EBU and SDRAM accesses in order to sync the data or code transfer. When EBU is sampling the inputs from SDRAM, number of LMB clock cycles delay to be inserted is configured using SDRMCON.DTALTNCY parameter, this should be based on the clock settings supported by external device. CAS latency is one of the important feature that configures or tells the EBU that how long it has to wait from the issue of read command to availability of data, this feature is set using SDRMOD.CASLAT parameter.

3.6 Access SDRAM Memory

Accessing SDRAM device is command-based, from initialization to normal read/write and refresh control. The important points are to follow the sequence and set correct parameters. In EBU, the following registers are related to SDRAM device access control.

- SDRMCON
- SDRMOD
- SDRMREF
- SDRSTAT

In which, SDRMCON and SDRMREF registers set the parameters related to timing. The **Table 6** gives guideline how to set these parameters. The minimum number of clock cycles required can be calculated by dividing t_{xxx} with clock cycle time and rounding up to the next higher integer.

Table 6 The relationship between register definition and parameters

Register	Bit Field	Function Description	Corresponding to[1)]
SDRMCON	CRAS	Minimum Row Active Delay Counter (clock cycles)	T_{ras} in device data sheet
	CRFSH	Initialization Refresh Command Counter	Refer to Initialization Section of device data sheet for exact number
	CRSC	Mode Register Setup Time (clock cycles)	T_{mrd} in device data sheet
	CRP	Row Precharge Time Counter (clock cycles)	T_{rp} in device data sheet
	CRCD	Row to Column Delay Counter (clock cycles)	T_{rcd} in device data sheet
	CRC	Row Cycle Time Counter (clock cycles)	T_{rc} in device data sheet

Table 6 The relationship between register definition and parameters

Register	Bit Field	Function Description	Corresponding to[1)]
SDRMREF	REFRESHC	Number of clock cycles between refresh operations	T_{ref} in device data sheet
	REFRESHR	Number of refresh command	Refer to device data sheet

1) The terms used in this column are based on the Intel PC SDRAM Specification, Revision 1.7 except T_{ref}.

3.7 Register settings for various SDRAM Devices

3.7.1 Registers Independent of Device Type

3.7.1.1 EBUCON - EBU Configuration Register

Table 7 EBUCON Register Settings

Registers	Setting	Function
EBU_CON	0x04F9 FF68	1:1 mode, clock gating on
	0x00F9 FF68[1)]	1:1 mode, clock gating off
	0x05F9 FF68	1:2 mode, clock gating on
	0x01F9 FF68	1:2 mode, clock gating off

1) The option is fastest setting and verified on TC1130 Triboard

3.7.1.2 BUSCONx - EBU Bus Configuration Register

Table 8 BUSCON Register Settings

Registers	Setting	Function
BUSCON	0x30700000	SDRAM Type 0 & 16 bit port width
	0x30B00000	Prefetch Off, SDRAM Type 0 & 32 bit port width
	0x40700000	SDRAM Type 1 & 16 bit port width
	0x40B00000	SDRAM Type 1 & 32 bit port width
	0x30B40000[1)]	Prefetch On, SDRAM Type 0 & 32 bit port width

1) The option is fastest setting and verified on TC1130 Triboard

3.7.2 HY57V651620 SDRAM 100MHz device

This configuration assumes an LMB clock of 100MHz, the SDRAM clock should also run at 100MHz. This configuration is a 16-bit device. 64Mb = 8MB.

Table 9 Various SDRAM Register Settings for HY57V651620 device

Register Name	Value to be set	Description
SDRMCON	0x008D9A85	
SDRMREF	0x00000018	
SDRMOD	0x00000020	Burst Length - 0 CAS Latency - 2
	0x00000023[1)]	Burst Length - 8 CAS Latency - 2
	0x00000030	Burst Length - 0 CAS Latency - 3
	0x00000033	Burst Length - 8 CAS Latency - 3

1) The option is fastest setting and verified on TC1130 Triboard

3.7.3 HYB39S256160-8 SDRAM 125MHz device.

This configuration assumes an LMB clock of 100MHz, the SDRAM clock should also run at 100MHz. This configuration is 1 x 16-bit device. So, 256Mb = 32MB.

Table 10 Various SDRAM Register Settings for HYB39S256160-8 device

Register Name	Value to be set	Description
SDRMCON	0x0115AA85	
	0x01161075[1)]	For clock 1:1 mode
SDRMREF	0x000000D7	
SDRMOD	0x00000020	Burst Length - 0 & CAS Latency - 2
	0x00000023[1)]	Burst Length - 8 & CAS Latency - 2
	0x00000030	Burst Length - 0 & CAS Latency - 3
	0x00000033	Burst Length - 8 & CAS Latency - 3

1) The option is fastest setting and verified on TC1130 Triboard.

3.8 Example to Program the SDRAM device present on TC1130 Triboard

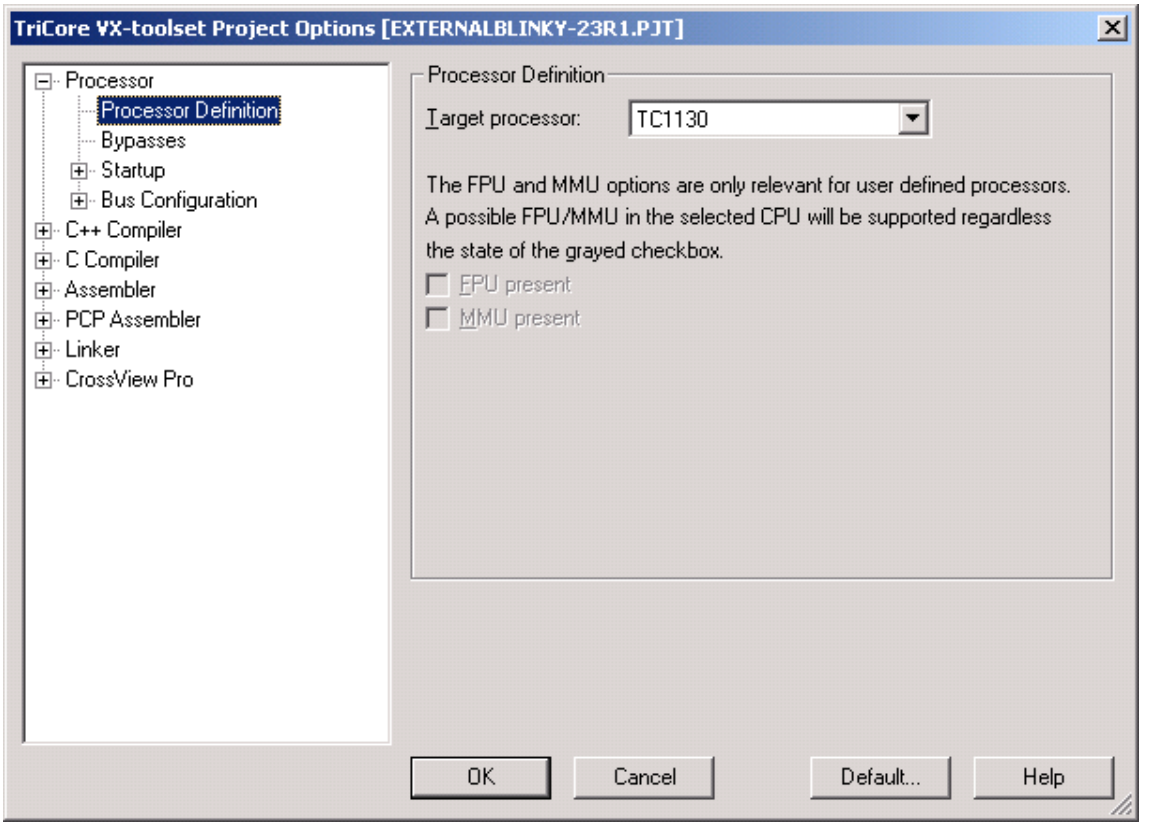
This section shows an example to program the HYB39S256160-8 SDRAM device present on the TC1130 Triboard with a simple LED blinking program.The aim of this exercise is to Blink a LED connected to the Triboard (P0.7) using only SDRAM. No interrupts will be used here, only a ‘for loop‘ for delay purposes

1. Write a simple program to blink the LED on P0.7The main program is fairly easy, the port pin P0.7 is initialized as output and switched on and off. The ‘tc1130regs.h‘ file declares all the TC1130 registers (i.e. Ports, Peripherals, …)

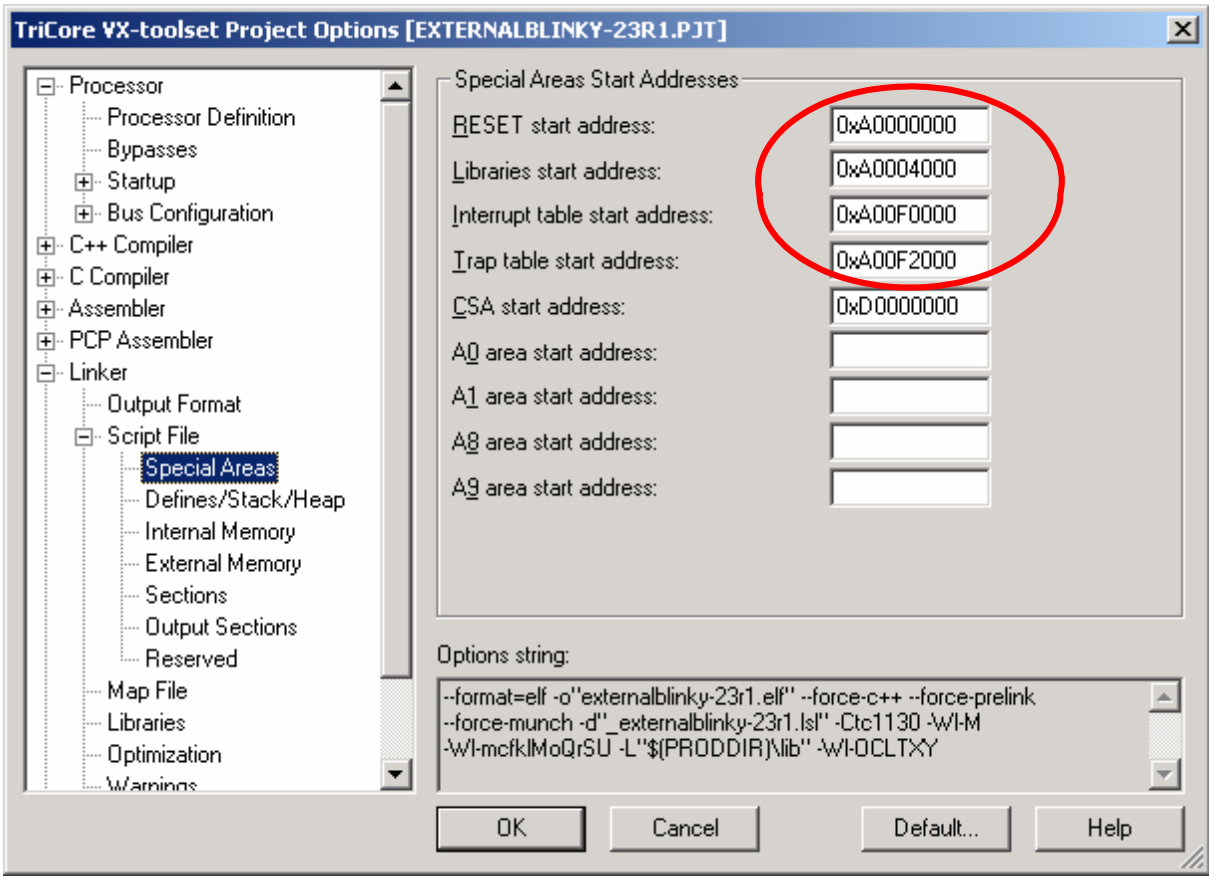
```
/*
* blinky.c
* Example for using TC1130 board.
*/
#include <tc1130regs.h>
void delay(void)
{
volatile int i,j,k;
for (i=0;i<5000;i++)
for (j=0;j<40;j++)
k = i+j;
}
int main(void)
{
// set P0.7 as output
P0_DIR_P7 = 1;
// main loop
while (1)
{
// set led on
P0_OUT_P7 = 1;
// small pause
delay();
// set led off
P0_OUT_P7 = 0;
// small pause
delay();
}
return 0;
}
```

2. Create a project work space in tasking tool chain using TC1130 as target and include the blinky.c which is shown above
3. Now most important task is setting the Project options in Tasking, Linker options has to be set to run the code from external memory

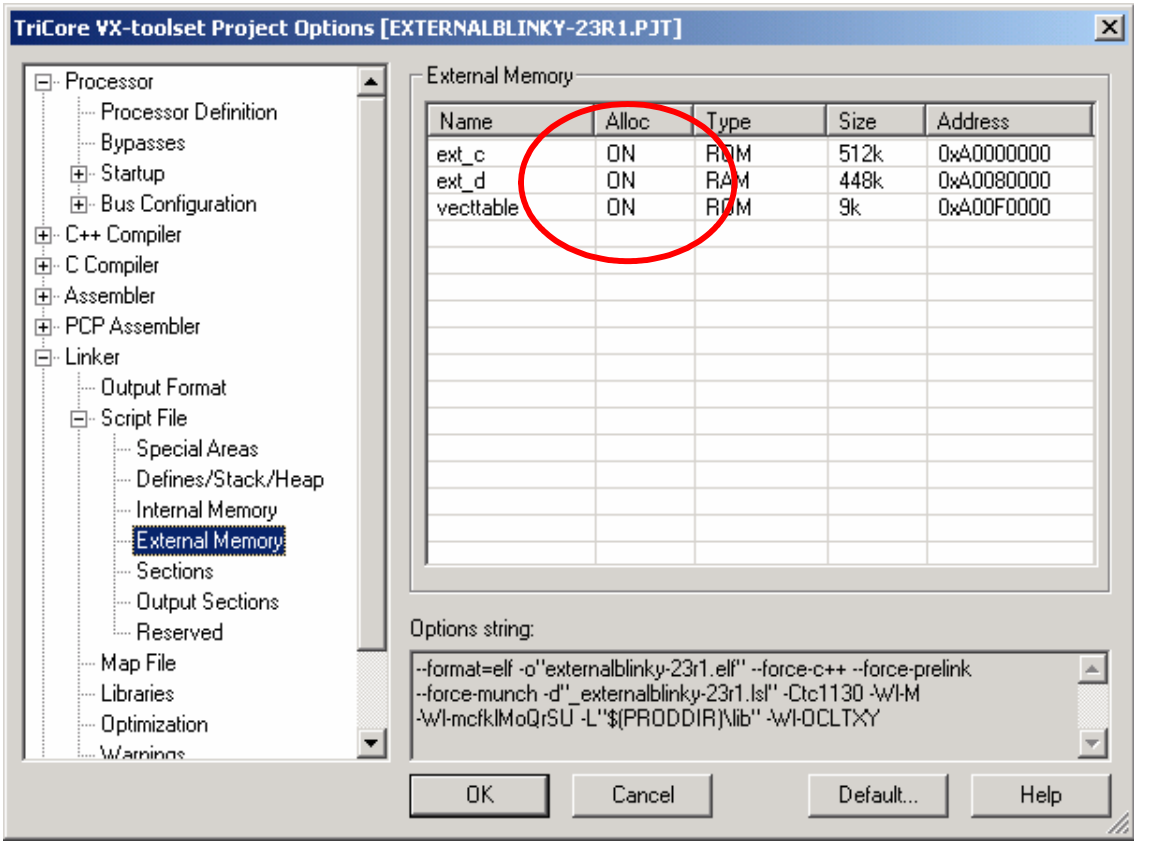
select the target as TC1130



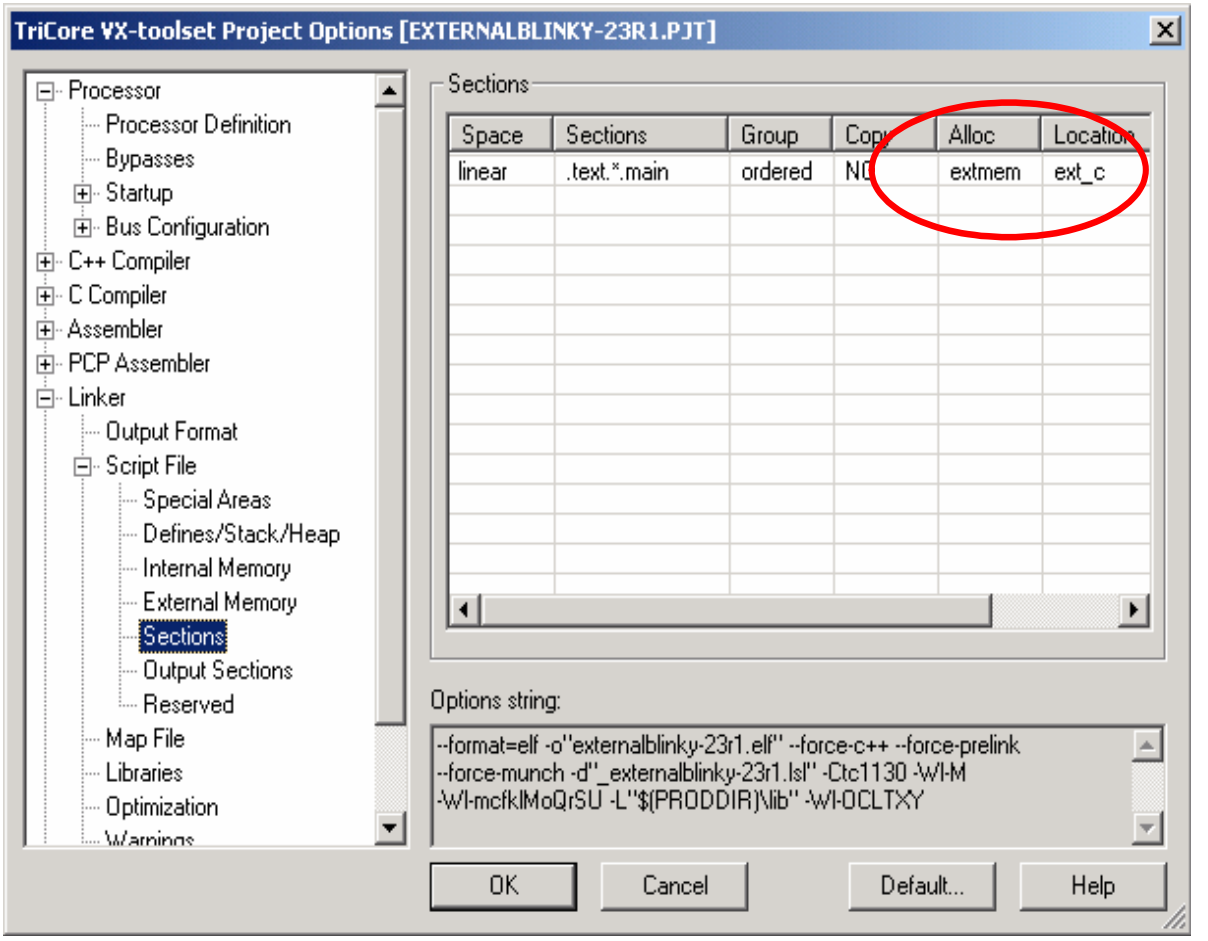
Set the code storing memory as the external memory region, these are the addresses located in SDRAM



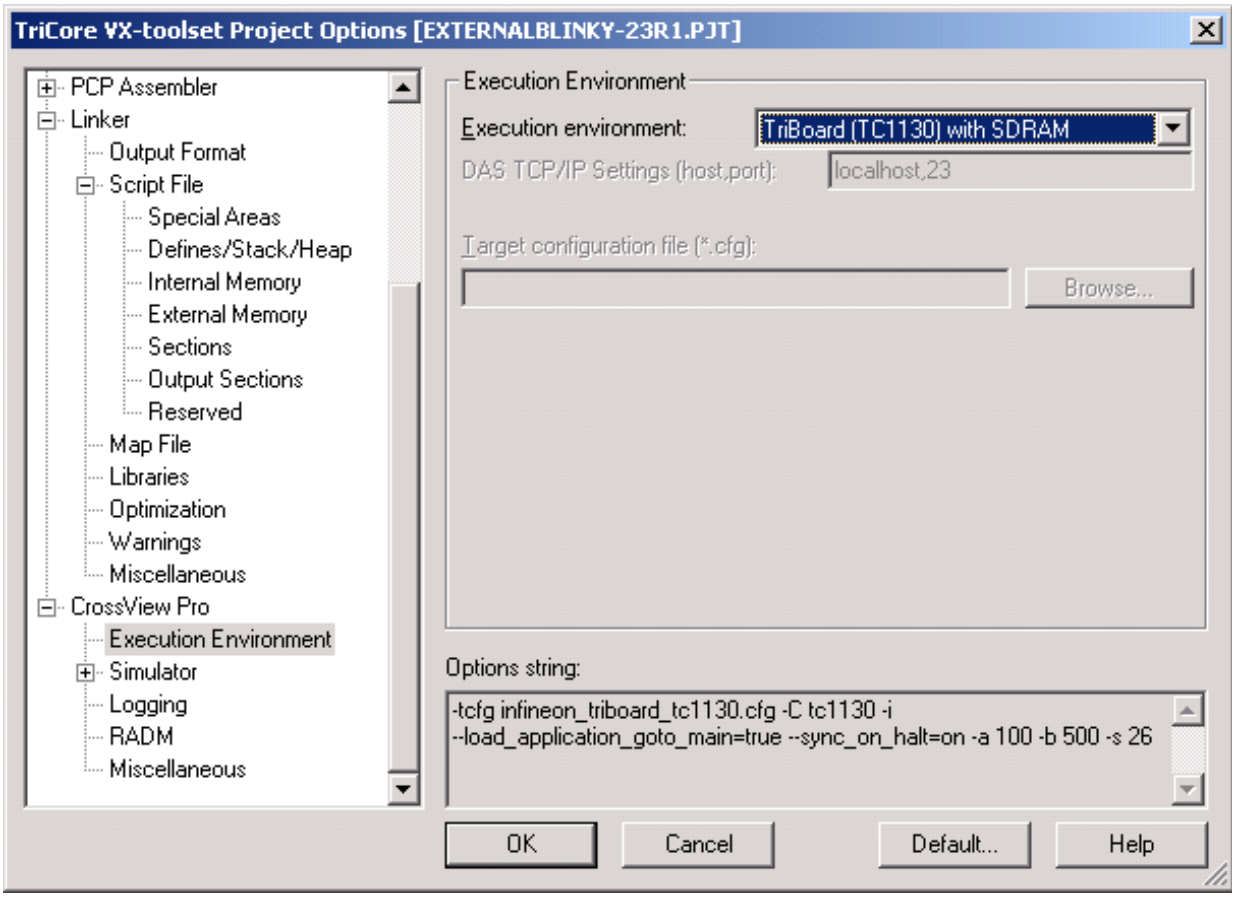
Enable the external memory segments



Select external memory for code



Select a triboard with SDRAM



Following are the SDRAM settings that are done automatically in the debugger with the selection of above said execution environment: These configurations should be done manually If any other debugging environment is been used.

```
EBU_ADDRSEL1 = 0xa0000853; //for cache disabled
EBU_ADDRSEL2 = 0xa1000853; //for cache disabled
EBU_BUSCON1  = 0x30b40000; // sdram access type 0, prefetch on,
//32 bit port width
EBU_CON      = 0x00F9FF68; //1:1 mode, no gating
EBU_SDRMCON0 = 0x01161075; //if 1:2 mode choose 0x21161075
EBU_SDRMOD0  = 0x00000023; // Cas latency 2, burstlength 8
EBU_BUSAP1   = 0xffffffff;
EBU_SDRMREF0 = 0x000000D7;
```

4. Now the code is ready for compilation
5. After compilation launch the crossviewpro debugger and run the program
6. LED on the triboard should start blinking now.

Note: EBU settings given in this example are very basic.

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG