Application Note, V1.0, Aug. 2008

AP32126

TriCore

Using Data Access Overlay Functionality with TC1766

Microcontrollers

Never stop thinking

Edition 2008-09-01

Published by
Infineon Technologies AG
81726 München, Germany

AP32126

Revision History:	2008-08	V1.0
Previous Version:	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

Table of Contents **Page**

1 Introduction

This document provides a method for overlaying code memory onto data memory with an example code. This method is primarily intended for the AUDO-NG Family with Overlay memory. An overview of this functionality is shown in Figure 1. An example is shown in Figure 2.

This application note assumes that the reader is already familiar with the TriCore architecture, the Tasking Compiler, the PLS debugger and the TC1766 TriBoard / StarterKit. The reader should also have read the section, Data Access Overlay functionality in the TC1766 User Manual.

Please note that the code delivered with this application note is intended for demonstration purpose only. Neither is its quality nor its robustness guaranteed.

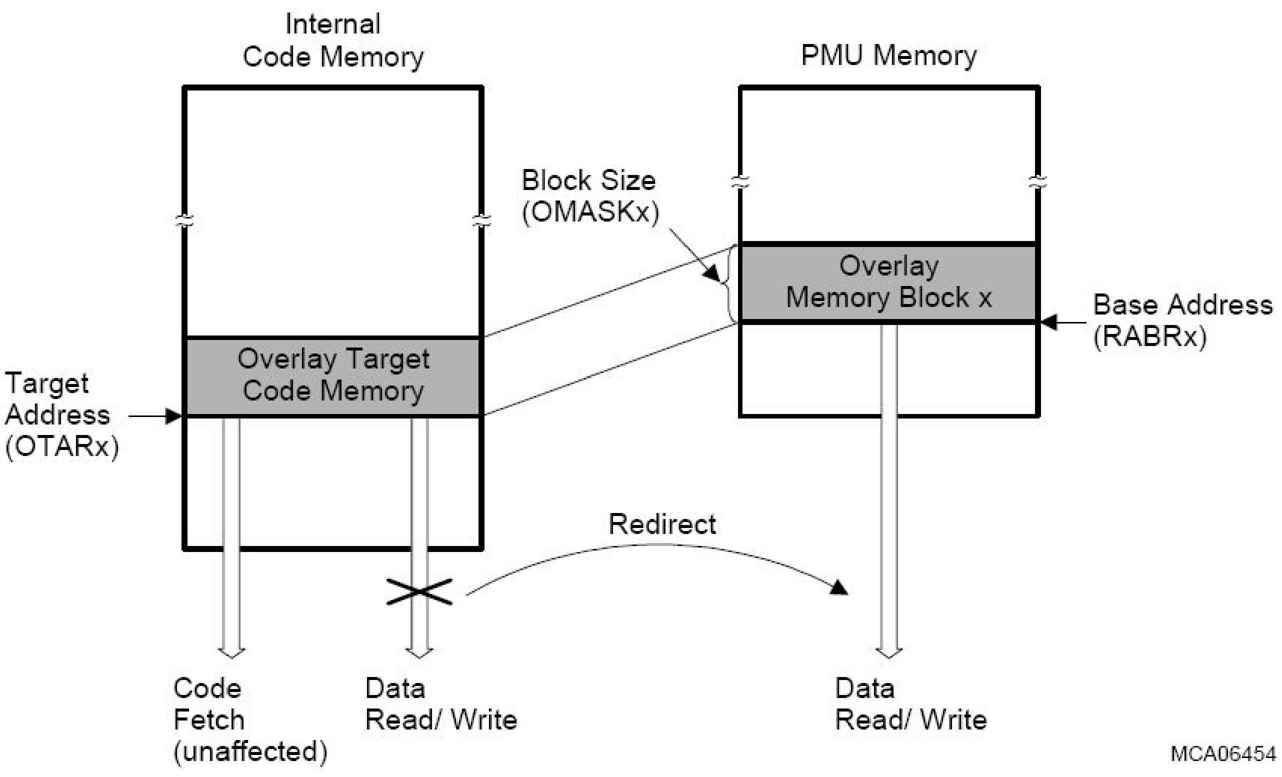


Figure 1 Redirection of Data Read Accesses from Code Memory to PMU SRAM (Internal Data) Memory

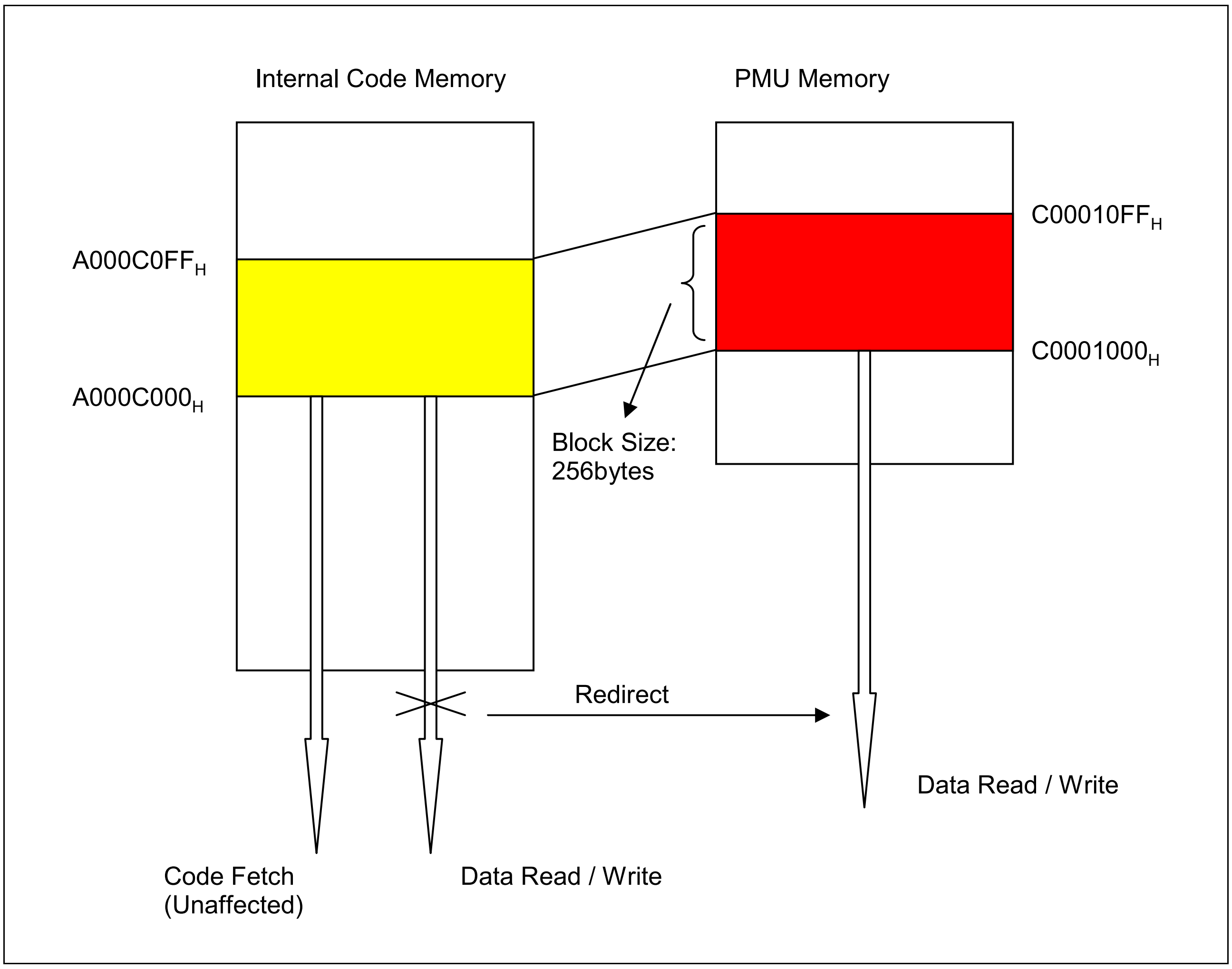


Figure 2 An Example on Redirection of Data Read Accesses from Code Memory to PMU Memory

2 Data Access Overlay Functionality

The data overlay functionality provides the capability to redirect data read accesses from internal code memory to data accesses from the PMU SRAM. This functionality makes it possible, for example, to modify program parameters (which are typically stored in the code memory) during run time of a program. This is commonly referred to as a calibration sequence. Instruction fetches are not affected by the PMU data read access redirection capability. Note that read and write data accesses from/to code memory are redirected.

During the calibration sequence (cycle), calibration memory (usually an internal SRAM) is temporarily utilized by the system. The sequence begins by the system copying its current parameter tables from non-volatile memory (usually Flash) into volatile memory (SRAM). Once copied, an overlay mechanism is instituted (configure required PMU Overlay Control registers). The system now runs as it normally would. The user program thinks it is reading its parameters in their normal address locations in nonvolatile memory. But in reality, they are now being accessed from the SRAM locations. Once the calibration cycle has been completed, the modified parameters that are in the volatile memory need to be stored back to the non-volatile memory. Then the calibration memory is removed from the system and the sequence is complete.

2.1 Program Memory Unit (PMU) Overlay Control Registers

In the TC1766, the 8 Kbyte PMU SRAM memory can be divided into a maximum of sixteen overlay memory blocks, with programmable base address and block sizes, which can be individually enabled for overlay functionality. The block size of each overlay memory block can be in the range of 2 bytes up to 512 bytes.

Users need to configure three PMU Overlay Control Registers to enable the overlay function.

- The overlay target address in the Overlay Target Address Register OTARx (x = 0-15), which determines the base address of the code memory data block x to be redirected (i.e. 0xA000C000, as shown in Figure 2)

- The base address of the overlay memory block in the PMU SRAM in the Redirected Address Base Register RABRx (i.e. 0xC0001000, as shown in Figure 2)

- A mask in the Overlay Mask Register OMASKx, defining the size of the block, the address bits to check for an address match, and which bits are used from the target address and which from the redirected address base (i.e. 0x0FFFFF00, 256bytes, as shown in Figure 2)

3 Code Example

The code example provided, demonstrates how to perform an overlay of 256 bytes of code memory. The memory size is arbitrary, but user needs to be aware that all load/store operations to code memory will be redirected when the overlay control registers are configured / enabled.

The TC1766 TriBoard / StarterKit was used to verify the operation of the code were correct. The software environment used was Tasking EDE (TriCore-PCP Vx-toolset) along with their CrossView Pro debugger or pls UDE / UAD2 debugger. Two Tasking EDE project option files are included to run the example code either from the internal SPRAM (0xD4000000) or from the internal program flash (0xA0000000).

The example code is generated using Digital Application virtual Engineer (DAvE) for TC1766. It is compiled using Tasking EDE v2.5r2. The code is located inside the program flash (0xA0000000). When running the code on the TC1766 TriBoard / StarterKit, the onboard LED will also be toggling periodically. With the overlay control registers configured / enabled, although the code is writing data (0x00 – 0xFF) to program flash address of 0xA000C000 – 0xA000C0FF, the actual data are written to SRAM address of 0xC0001000 – 0xC00010FF (refer to Figure 2).

In addition, the code will do a verification of the data written to the Overlay SRAM location. If the value written is not correct, the code will stop running and the onboard LED will not be toggling.

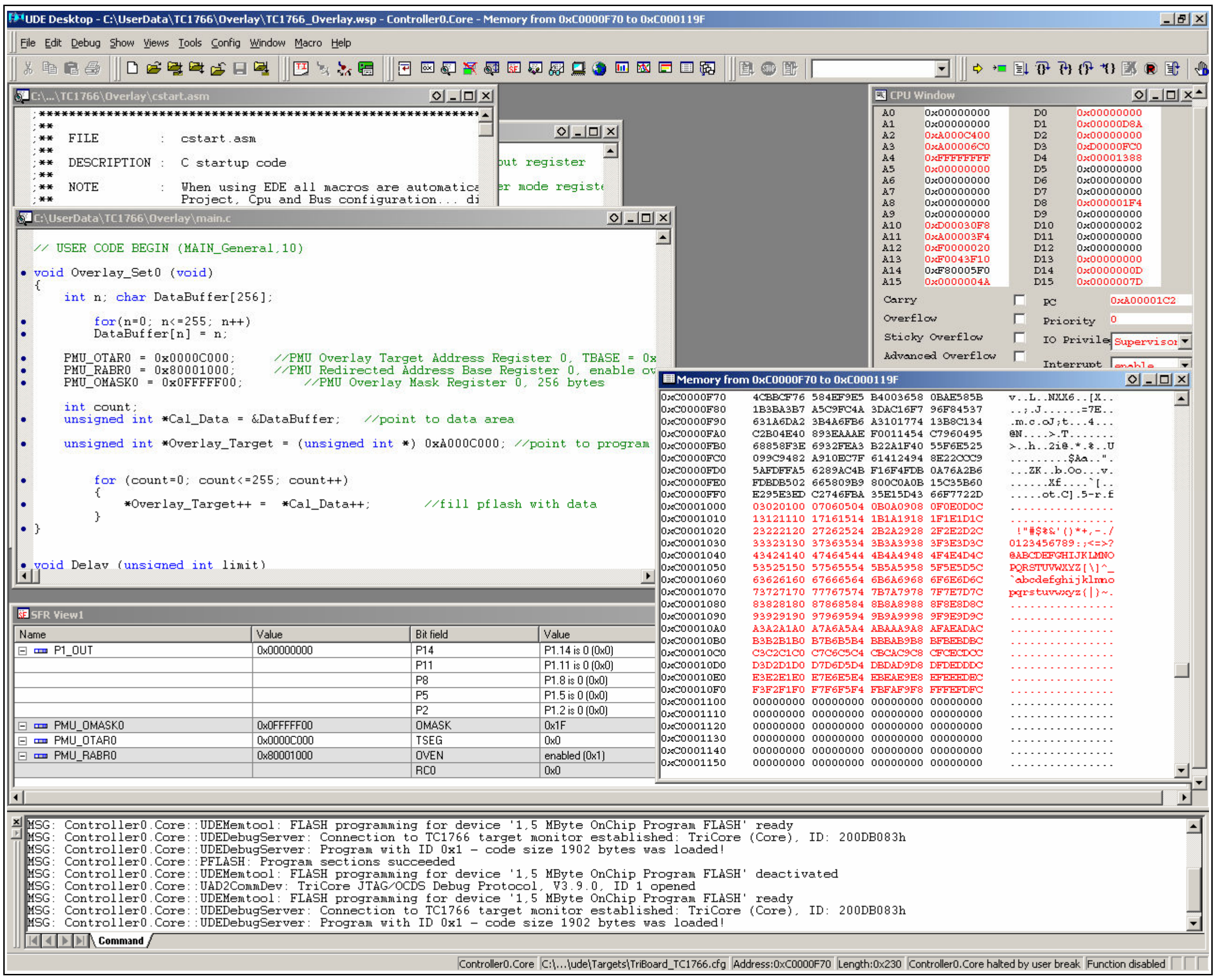
3.1 Running the Code

1. Unzip the Tasking Project attached with this document in the folder C:\Infineon\.
2. Start a debugger session (e.g. pls UDE) and load the executable “overlay.elf”.
3. Run the application.

By activating the memory window of the debugger, users can see that data is written to the SRAM memory (0xC0001000 – 0xC00010FF) although we are writing to the program flash as shown in Figure 3.

As seen in Figure 3, the three overlay control registers are configured / enabled as follows.

- PMU_OTAR0 = 0x0000C000 (PMU Overlay Target Address Register 0, TBASE = 0x000C000, PFlash = 0xA000C000)
- PMU_RABR0 = 0x80001000 (PMU Redirected Address Base Register 0, enable overlay, OBASE = 0xC0001000, from 0x00 to 0xFF)
- PMU_OMASK0 = 0x0FFFFF00 (PMU Overlay Mask Register 0, 256 bytes)


UDE Desktop - C:\UserData\TC1766\Overlay\TC1766_Overlay.wsp - Controller0.Core - Memory from 0xC0000F70 to 0xC000119F
File Edit Debug Show Views Tools Config Window Macro Help
C:\...\TC1766\Overlay\cstart.asm
C:\UserData\TC1766\Overlay\main.c
CPU Window
Memory from 0xC0000F70 to 0xC000119F
SFR View1
Command

```
;*****************************************************************
;**
;**  FILE          :  cstart.asm
;**
;**  DESCRIPTION   :  C startup code
;**
;**  NOTE          :  When using EDE all macros are automatica
;**                   Project, Cpu and Bus configuration... di
```

```
put register
er mode registe
```

```
// USER CODE BEGIN (MAIN_General,10)

void Overlay_Set0 (void)
{
    int n; char DataBuffer[256];

        for(n=0; n<=255; n++)
        DataBuffer[n] = n;

    PMU_OTAR0 = 0x0000C000;     //PMU Overlay Target Address Register 0, TBASE = 0x
    PMU_RABR0 = 0x80001000;     //PMU Redirected Address Base Register 0, enable ov
    PMU_OMASK0 = 0x0FFFFF00;        //PMU Overlay Mask Register 0, 256 bytes

    int count;
    unsigned int *Cal_Data = &DataBuffer;   //point to data area

    unsigned int *Overlay_Target = (unsigned int *) 0xA000C000; //point to program


        for (count=0; count<=255; count++)
        {
            *Overlay_Target++ =  *Cal_Data++;       //fill pflash with data
        }
}


void Delay (unsigned int limit)
```

```
A0    0x00000000      D0    0x00000000
A1    0x00000000      D1    0x00000D8A
A2    0xA000C400      D2    0x00000000
A3    0xA00006C0      D3    0xD0000FC0
A4    0xFFFFFFFF      D4    0x00001388
A5    0x00000000      D5    0x00000000
A6    0x00000000      D6    0x00000000
A7    0x00000000      D7    0x00000000
A8    0x00000000      D8    0x000001F4
A9    0x00000000      D9    0x00000000
A10   0xD00030F8      D10   0x00000002
A11   0xA00003F4      D11   0x00000000
A12   0xF0000020      D12   0x00000000
A13   0xF0043F10      D13   0x00000000
A14   0xF80005F0      D14   0x0000000D
A15   0x0000004A      D15   0x0000007D

Carry                 ☐   PC           0xA00001C2
Overflow              ☐   Priority     0
Sticky Overflow       ☐   IO Privile   Supervisor
Advanced Overflow     ☐   Interrupt    enable
```

```
0xC0000F70     4CBBCF76 584EF9E5 B4003658 0BAE585B     v..L..NXX6..[X..
0xC0000F80     1B3BA3B7 A5C9FC4A 3DAC16F7 96F84537     ..;.J......=7E..
0xC0000F90     631A6DA2 3B4A6FB6 A3101774 13B8C134     .m.c.oJ;t...4...
0xC0000FA0     C2B04E40 893EAAAE F0011454 C7960495     @N....>.T.......
0xC0000FB0     68858F3E 6932FEA3 B22A1F40 55F6E525     >..h..2i@.*.%..U
0xC0000FC0     099C9482 A910EC7F 61412494 8E22CCC9     .........$Aa..".
0xC0000FD0     5AFDFFA5 6289AC4B F16F4FDB 0A76A2B6     ...ZK..b.Oo...v.
0xC0000FE0     FDBDB502 665809B9 800C0A0B 15C35B60     ......Xf....`[..
0xC0000FF0     E295E3ED C2746FBA 35E15D43 66F7722D     .....ot.C].5-r.f
0xC0001000     03020100 07060504 0B0A0908 0F0E0D0C     ................
0xC0001010     13121110 17161514 1B1A1918 1F1E1D1C     ................
0xC0001020     23222120 27262524 2B2A2928 2F2E2D2C      !"#$%&'()*+,-./
0xC0001030     33323130 37363534 3B3A3938 3F3E3D3C     0123456789:;<=>?
0xC0001040     43424140 47464544 4B4A4948 4F4E4D4C     @ABCDEFGHIJKLMNO
0xC0001050     53525150 57565554 5B5A5958 5F5E5D5C     PQRSTUVWXYZ[\]^_
0xC0001060     63626160 67666564 6B6A6968 6F6E6D6C     `abcdefghijklmno
0xC0001070     73727170 77767574 7B7A7978 7F7E7D7C     pqrstuvwxyz{|}~.
0xC0001080     83828180 87868584 8B8A8988 8F8E8D8C     ................
0xC0001090     93929190 97969594 9B9A9998 9F9E9D9C     ................
0xC00010A0     A3A2A1A0 A7A6A5A4 ABAAA9A8 AFAEADAC     ................
0xC00010B0     B3B2B1B0 B7B6B5B4 BBBAB9B8 BFBEBDBC     ................
0xC00010C0     C3C2C1C0 C7C6C5C4 CBCAC9C8 CFCECDCC     ................
0xC00010D0     D3D2D1D0 D7D6D5D4 DBDAD9D8 DFDEDDDC     ................
0xC00010E0     E3E2E1E0 E7E6E5E4 EBEAE9E8 EFEEEDEC     ................
0xC00010F0     F3F2F1F0 F7F6F5F4 FBFAF9F8 FFFEFDFC     ................
0xC0001100     00000000 00000000 00000000 00000000     ................
0xC0001110     00000000 00000000 00000000 00000000     ................
0xC0001120     00000000 00000000 00000000 00000000     ................
0xC0001130     00000000 00000000 00000000 00000000     ................
0xC0001140     00000000 00000000 00000000 00000000     ................
0xC0001150     00000000 00000000 00000000 00000000     ................
```

Name	Value	Bit field	Value
⊟ P1_OUT	0x00000000	P14	P1.14 is 0 (0x0)
		P11	P1.11 is 0 (0x0)
		P8	P1.8 is 0 (0x0)
		P5	P1.5 is 0 (0x0)
		P2	P1.2 is 0 (0x0)
⊟ PMU_OMASK0	0x0FFFFF00	OMASK	0x1F
⊟ PMU_OTAR0	0x0000C000	TSEG	0x0
⊟ PMU_RABR0	0x80001000	OVEN	enabled (0x1)
		RC0	0x0

```
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '1,5 MByte OnChip Program FLASH' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1766 target monitor established: TriCore (Core), ID: 200DB083h
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 1902 bytes was loaded!
MSG: Controller0.Core::PFLASH: Program sections succeeded
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '1,5 MByte OnChip Program FLASH' deactivated
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.9.0, ID 1 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '1,5 MByte OnChip Program FLASH' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1766 target monitor established: TriCore (Core), ID: 200DB083h
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 1902 bytes was loaded!
```

```
Controller0.Core | C:\...\ude\Targets\TriBoard_TC1766.cfg | Address:0xC0000F70 | Length:0x230 | Controller0.Core halted by user break | Function disabled
```

Figure 3 Memory view and registers view of PMU SRAM using the pls debugger