Application Note, V 1.0, Sept. 2002

AP32033

TC1775

Using the $\overline{\text{Code}}$ signal to Overlay External Code Memory onto External Data Memory

Microcontrollers

Never stop thinking.

TC1775

Revision History:	**2002-09**	V 1.0
Previous Version:	2000-06	V 0.1

Page	Subjects (major changes since last revision)
all	Changed to Infineon Template

Controller Area Network (CAN): License of Robert Bosch GmbH

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

Edition 2002-09

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.

Table of Contents

1 Introduction

This document provides a method for overlaying external code memory onto external data memory. This method is primarily intended for the TC1775 A-Step. However, this should be applicable to all future steps of the TC1775. An overview of this functionality is shown in Figure 1.

Beginning with the TC1775 B-Step, there are on-chip resources that can provide similar functionality without requiring any external control circuitry.

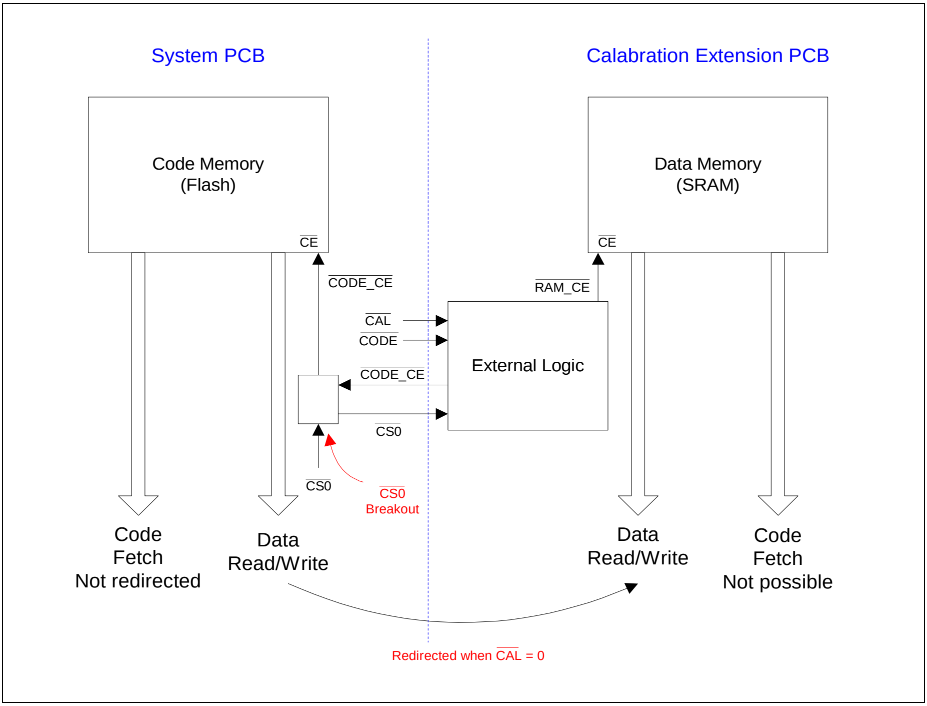

Figure 1 Block Diagram of Overlay Redirection

2 Overlay Functionality

The overlay functionality provides a method to redirect load/store accesses from the code memory to external data (calibration) memory. The purpose for this functionally is primarily used to modify parameters (constants) normally stored in the code memory during runtime without remapping variable locations. This is commonly referred to as a calibration sequence.

During this calibration sequence (cycle), calibration memory (usually an external SRAM or DPRAM) is temporary connected to the system. The sequence begins by the system coping its current parameter tables from non-volatile memory (usually Flash) into volatile memory (SRAM). Once copied, a transparent overlay mechanism is instituted (in this example the signal $\overline{\text{CAL}}$ is used). The system now runs as it normally would. The user program thinks it is reading its parameters in their normal address locations in non-volatile 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.

This document provides details of one method where this can be achieved on the TC1775.

Note: When the $\overline{\text{CAL}}$ signal is active any load/store operation to the Code memory will be redirected to the Calibration memory.

Table 1 Access and Redirection Types

Access Type	Redirection	
	To Internal Data Memory (Segment 13)	**To External Data Memory (Segments 10, 11, 14)**
Data Access from external Code Memory	A-step not possible, B-step possible	A-step not possible, B-step possible
Data Access from external Code Memory (using signals $\overline{\text{CODE}}$ and $\overline{\text{CS0}}$)	Not Possible	Explained in this AppNote

3 Assumptions

There are a number of assumptions that are assumed to correctly implement the overlay functionality. They are listed as follows:

1. The external data memory (Calibration memory) is not normally resident on the user's PCB. It is only connected (mechanically / electrically) during a calibration sequence. There is a breakout connection for pins $\overline{\text{CS0}}$ and $\overline{\text{CE}}$ (code memory chip enable, see Figure 3).
2. The data memory is referred to as the calibration memory. The chip select used for selecting the code memory is the same chip select for the calibration memory. In this example, chip select $\overline{\text{CS0}}$ is used.
3. To guarantee proper access, the overlay memory must meet the same access requirements as the code memory (i.e. bus size, memory size, etc.) The settings for register EBU_BUSCON0 of the External Bus Unit (EBU) will also be used for load/ store accesses to the Calibration memory.
4. The user acounts for the extra propagation delay from the external logic gates concerning bus settings.

4 Signals

4.1 Chip Select Lines, $\overline{\text{CSx}}$

The External Bus Unit (EBU) provides four user chip selects, $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, $\overline{\text{CS2}}$ and $\overline{\text{CS3}}$. The address range for each of the chip selects is generated, per the programmed value of the respective address select registers, EBU_ADDSELx.

The chip select line $\overline{\text{CS0}}$ is the default chip select and is used automatically by the EBU for external memory access after reset. Therefore, non-volatile code memory is usually connected to this chip select.

Note: If overlapping address ranges are programmed in the EBU_ADDSELx registers, only one chip select, the one with the lowest number (highest priority), will be activated on an access within the overlapping range. Since $\overline{\text{CS0}}$ has the highest priority, another chip select cannot be used to overlay another memory region over it (as was the case in the C166 family EBC).

4.2 Instruction Fetch Indication Signal, $\overline{\text{CODE}}$

The EBU provides an additional signal $\overline{\text{CODE}}$ to distinguish between instruction fetch accesses and data load/store accesses on the external bus. A low level (logic zero) on this line indicates an instruction fetch is presently being performed by the Program Memory Unit (PMU). For all other types of accesses the $\overline{\text{CODE}}$ signal will remain at a high level (logic one).

Note: The user can program bit CS0D in register PMU_EIFCON to define whether $\overline{\text{CS0}}$ is generated during a burst mode access or not. It is assumed that bit CS0D is clear (default value) and is generated during code accesses.

4.3 Calibration Indication Signal, $\overline{\text{CAL}}$

The $\overline{\text{CAL}}$ signal is used to control which memory is selected during a load/store access ($\overline{\text{CS0}}$ must also be active). A low level (logic zero) on this line indicates a load/store access will be performed on the calibration memory. A high level (logic one) indicates a load/store access will be performed on the code memory. The state of this input has no effect on code fetches, they will always be accessed from the code memory (see truth table definition in Table 2).

The logic presented in Figure 2, provides the necessary logic to be able to perform the memory overlay. This includes the incorporation a signal called $\overline{\text{CAL}}$. This signal provides the capability to switch data accesses between the code memory and the data memory.

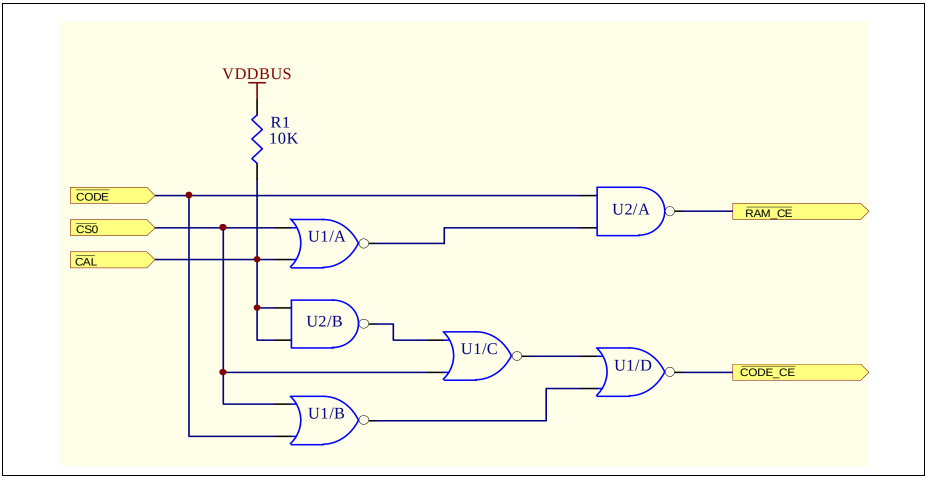

Figure 2 Schematic of the logic required for redirecting data access from external code (Flash) memory to external data (SRAM) memory.

4.4 External Logic Truth Table

Table 2, defines the logic truth table for the overlay functionality. There are three inputs ($\overline{\text{CS0}}$, $\overline{\text{CODE}}$, $\overline{\text{CAL}}$) and two outputs ($\overline{\text{RAM\_CE}}$ and $\overline{\text{CODE\_CE}}$).

Table 2 Truth Table for Logic

$\overline{\text{CS0}}$	$\overline{\text{CODE}}$	$\overline{\text{CAL}}$	$\overline{\text{RAM\_CE}}$	$\overline{\text{CODE\_CE}}$
0	0	0	1	0
0	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

4.5 $\overline{\text{CS0}}$ Breakout description

To connect the external logic to the users board. The $\overline{\text{CS0}}$ line on the users board needs to be modified. The method show in Figure 3, can be implemented using jumper settings. For normal operation the jumper should be set to position **A-to-B**, and during a calibration cycle the jumper should be set to position **A-to-D**.

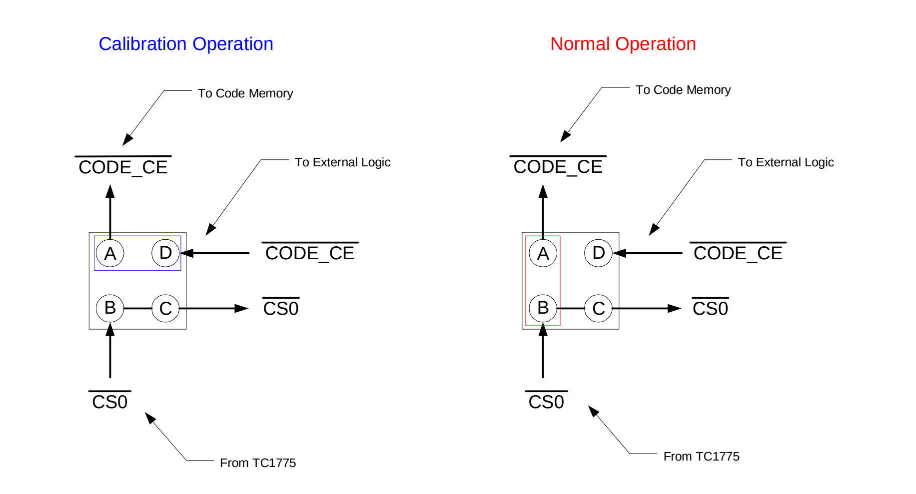

Figure 3 $\overline{\text{CS0}}$ Breakout Jumper Selection settings

5 Code Example

The code example provided, demonstrates how to perform an overlay of 1 K 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 $\overline{\text{CAL}}$ signal is active. The TC1775 TriBoard board was used to verify the operation of the code and logic were correct.

The software environment used was Tasking EDE along with their CrossView JTAG debugger. However, Green Hills environment can also be used.

The external logic was connected as previously described, and the $\overline{\text{CAL}}$ signal was connected to port pin P13.15.

6 Appendix

C-Code source code listing for file “main.c”.

```
#include "utilities.h"
#include "scu.h"
#include "port.h"

/* General defines */
#define ICR             0xFE2C
#define TOGGLE_LED      PX5
#define CAL_ENABLE      PX15

/* Prototype definitions */
void CopyOverlayMemory(unsigned int *src, unsigned int *des, unsigned int
Size);


/* main function */
void main (void)
{
  PORT    *psPort13;
  psPort13 = (PORT *)(P13_BASE);

  /* set P13.5, P13.15 high  */
  psPort13->OUT |= TOGGLE_LED | CAL_ENABLE;

  /* set P13.5, P13.15 are used as general purpose outputs */
  psPort13->ALTSEL0 &= ~TOGGLE_LED & ~CAL_ENABLE;
  psPort13->ALTSEL1 &= ~TOGGLE_LED & ~CAL_ENABLE;

  /* set P13.5 to output and high */
  psPort13->DIR |= TOGGLE_LED | CAL_ENABLE;

  /* set CPU frequency to 16 MHz, assumes 16 MHz crystal connected and
     N = 8 */
  SetCPUClock(5);  /* sets Kfactor to 8 */

  /* set CPU-priority to 0 */
  _mtcr(ICR, (_mfcr(ICR) & 0xFFFFFF00));

  /* enable interrupts  */
  _enable();
```

```
  /*
   *  THIS PROCESS IS RUN BEFORE THE CALIBRATION SEQUENCE
   */

  /* Enable load/store access to external code memory */
  psPort13->OUT &= ~CAL_ENABLE;

  /* Copy 1K external code memory to internal SRAM */
  CopyOverlayMemory((unsigned int *) 0xB0000000,
                    (unsigned int *) 0xD0008000, 256);

  /* Enable load/store access to external SRAM */
  psPort13->OUT &= ~CAL_ENABLE;

  /* Copy 1K internal SRAM to external SRAM */
  CopyOverlayMemory((unsigned int *) 0xD0008000,
                    (unsigned int *) 0xB0000000, 256);



  /*
   *  After the calibration sequence has ended, the modified parameters
   *  need to be written back to code memory. This normally would require
   *  code to write/erase flash memory...
   */

  /* loop forever... */
   while(1)
   {
     psPort13->OUT ^= TOGGLE_LED;
   };
}



/*
 * FUNCTION
 *
 */
void CopyOverlayMemory(unsigned int *src, unsigned int *des, unsigned int
Size)
{
  for ( ; Size; Size--)
     *des++ = *src++;
}
```

Infineon goes for Business Excellence

“Business excellence means intelligent approaches and clearly defined processes, which are both constantly under review and ultimately lead to good operating results.
Better operating results and business excellence mean less idleness and wastefulness for all of us, more professional success, more accurate information, a better overview and, thereby, less frustration and more satisfaction.”

Dr. Ulrich Schumacher

http://www.infineon.com

Published by Infineon Technologies AG