Application Note

***AN2667**/D*
Rev. 0. 12/2003

Multi-Controller Hardware Development for the MPC5xx Family Application Note

Steven McQuade
TECD Applications

ABSTRACT

This application note details how to configure the [1]MPC5xx family of microcontrollers to satisfy the requirements of a multi-controller system. It also describes the hardware for such a system by using an example of the dual MPC561-4 evaluation board (EVB) that was developed to demonstrate multi-controller functionality.

In a multi MPC5xx design that shares the same external bus, each controller can access any memory or peripheral in the system even if it is located within another MPC5xx. A multi-controller configuration increases the I/O capability and overall performance of the system.

Note

The organisation of this document is as follows:

1. **Introduction** provides the motivation behind building a multi-controller system.
2. **General Block Diagram** presents a general block diagram and signal connectivity.
3. **System Configuration Issues** addresses issues from inter-communication to system debug.
4. **Types of Accesses and Interrupt Handling** highlights considerations for types of accesses that can be performed and an interrupt handling scheme.
5. **Operation Mode Switching** deals with operation mode switching and data coherency.
6. **MPC555 / MPC56x Differences** details the differences between multi-MPC555 and multi-MPC56x systems.
7. **Dual Controller Performance** looks at how device choice can affect the overall system performance.
8. **EVB Overview** details the board capabilities.
9. **Conclusion** summarises the capability of the MPC5xx family.

1 MPC5xx refers to all members of the MPC5xx family except the MPC505/9 which are discontinued products.

1. Introduction

In a multi-MPC5xx system, each controller shares the same external bus where each MPC5xx has its own internal space (4 Mbytes) and each controller can access any memory or peripheral even if it is located within another controller in that system. Such a system may be desired in the following cases:

1. Systems that require additional peripherals can use an additional MPC5xx in peripheral mode, and extend serial/timers, and available memories.

2. Systems that require additional peripherals and computing power can use an additional MPC5xx in slave mode, and allow the master to control the slave parts through the external bus. In this case, each controller can run a separate application, which shares the data with another and interacts with it.

Typical applications could be calibration applications, where the external master device accesses systems under development to get information during run time. Other examples would be engine control plus CAN gateway or Steer-By-Wire plus CAN gateway.

There are several reasons for using two MPC5xx devices, namely:

1. Bus compatibility
2. Code compatibility
3. Little re-development cost to the customer as they only have to add another MPC5xx footprint to their PCB.
4. No requirement for further device qualification for the customer which would incurr additional project cost and time.
5. Capability for future expansion when transitioning from MPC561 to MPC563 which is pin compatible but provides 512K of UC3F flash memory.
6. Both controllers provide the same peripherals and are designed specifically for multi-master operation using the external bus.
7. Finally the other interfaces such as CAN and SPI operate at frequencies lower than the external bus which can function up to 66MHz. Therefore using one of these interfaces would reduce communication throughput between the devices.

2. System Overview

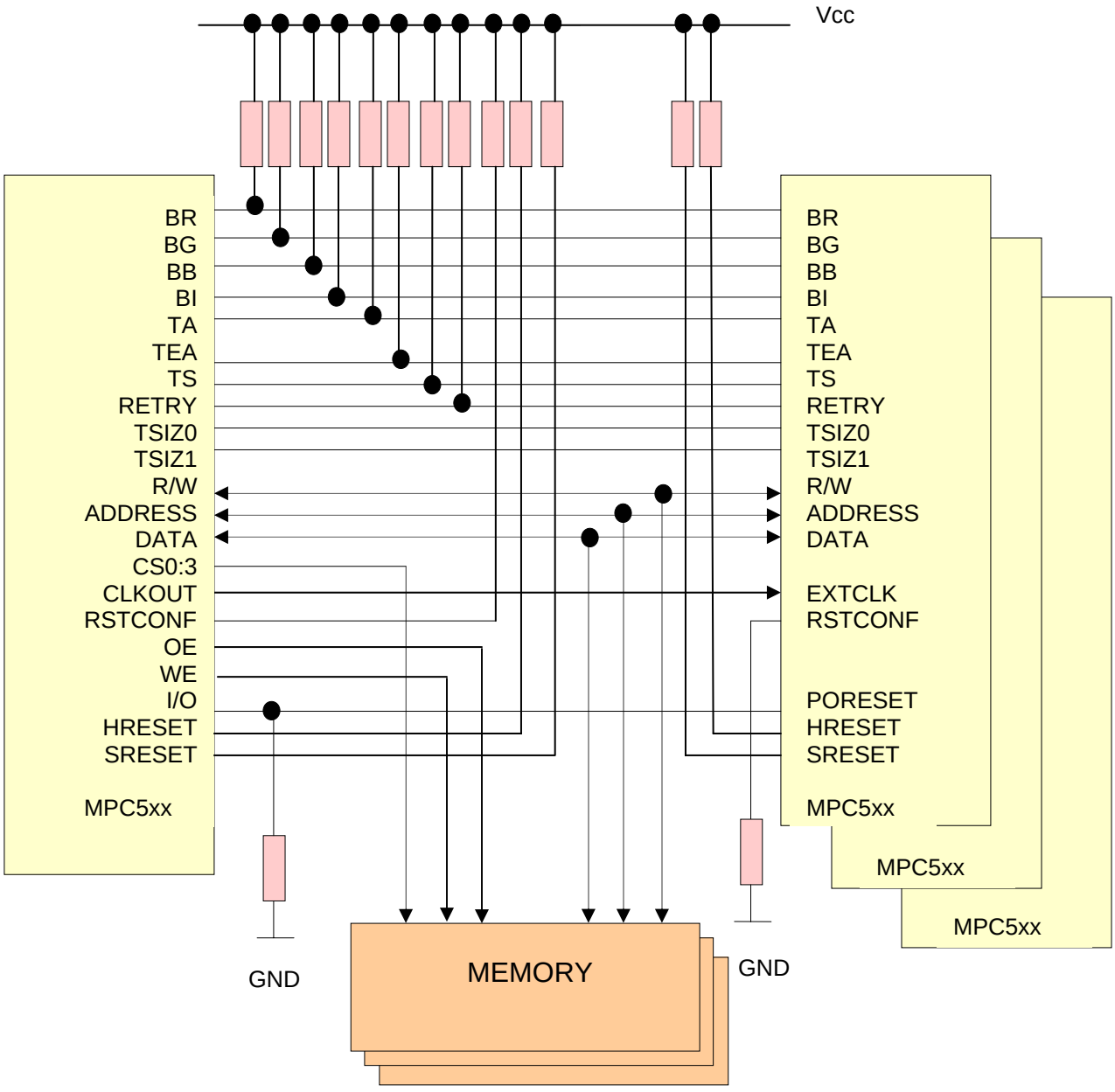

Fig.2

Note

Communication between devices does not require the chip selects.If more than 2 devices are required in the system then an external bus arbiter is necessary. Refer to the MPC561-4 Dual Controller EVB schematics for further detail.

3. System Configuration Issues

The following is a list of questions commonly asked when designing a multi-MPC5xx system:

1. How do the MPC5xx devices communicate with one another?
2. How is each device differentiated?
3. How is Master/Slave/Peripheral mode selected?
4. How is reset controlled on both devices?
5. What is the clocking strategy?
6. How is each device initialised?
7. How is debug handled?

The remainder of this document addresses these questions and provides the answers which ultimately allow customers to successfully design their own system.

3.1 How do the MPC5xx devices communicate with one another?

The devices within a multi-MPC5xx system communicate with one another across the External Bus Interface. As shown below in Figure 3.1 the transfer is initiated with the assertion of Transfer Start. The internal address of the other controller is also presented at this point along with R / #W. The slave device latches the address within the same clock cycle. The slave device fetches the data and presents it on the bus. The master device then latches the data within the same clock that the slave presents the data. The cycle is terminated by the TA (successful transfer), TEA(unsuccessful transfer) or RETRY.

RETRY is provided to solve a deadlock that may occur when the core of device [1]A tries to perform an external access but the external bus is busy because of an access of another device [1]B to some internal address of device A. Once this situation is encountered, RETRY is asserted to give device B a signal to release the external bus and to retry its access later giving device A the opportunity to complete its access.

The Burst Inhibit (BI) signal may be required when an external master cannot perform burst accesses to internal resources because the slave interface does not support bursts. Generally speaking the BI line is optional, because the MCU requests bursts only for instruction fetches. Thus if the multiprocessor system is designed in a way that one processor does not fetch instructions from internal memories of another one, the BI line is not required.

If there are only two MPC5xx devices within the system the bus arbitration can be handled by the arbiter in either device. The arbiter can be selected by setting or clearing the SIUMCR [EARB].If this bit is set then external arbitration on the external bus is assumed in which case internal arbitration should be selected on the second device. This would mean SIUMCR[EARB] should be programmed as “1” and “0” on each respective device. If there are more than two chips in the system, the only choice is to use some other external arbiter.On the device where SIUMCR [EARB] = “0”, there is a

need to program an SIUMCR [EARP] field as well. The SIUMCR [EARP] field determines the priority of external arbitration request.

1 The letters refer to 2 different devices in a system. It does not matter which device is master or slave.

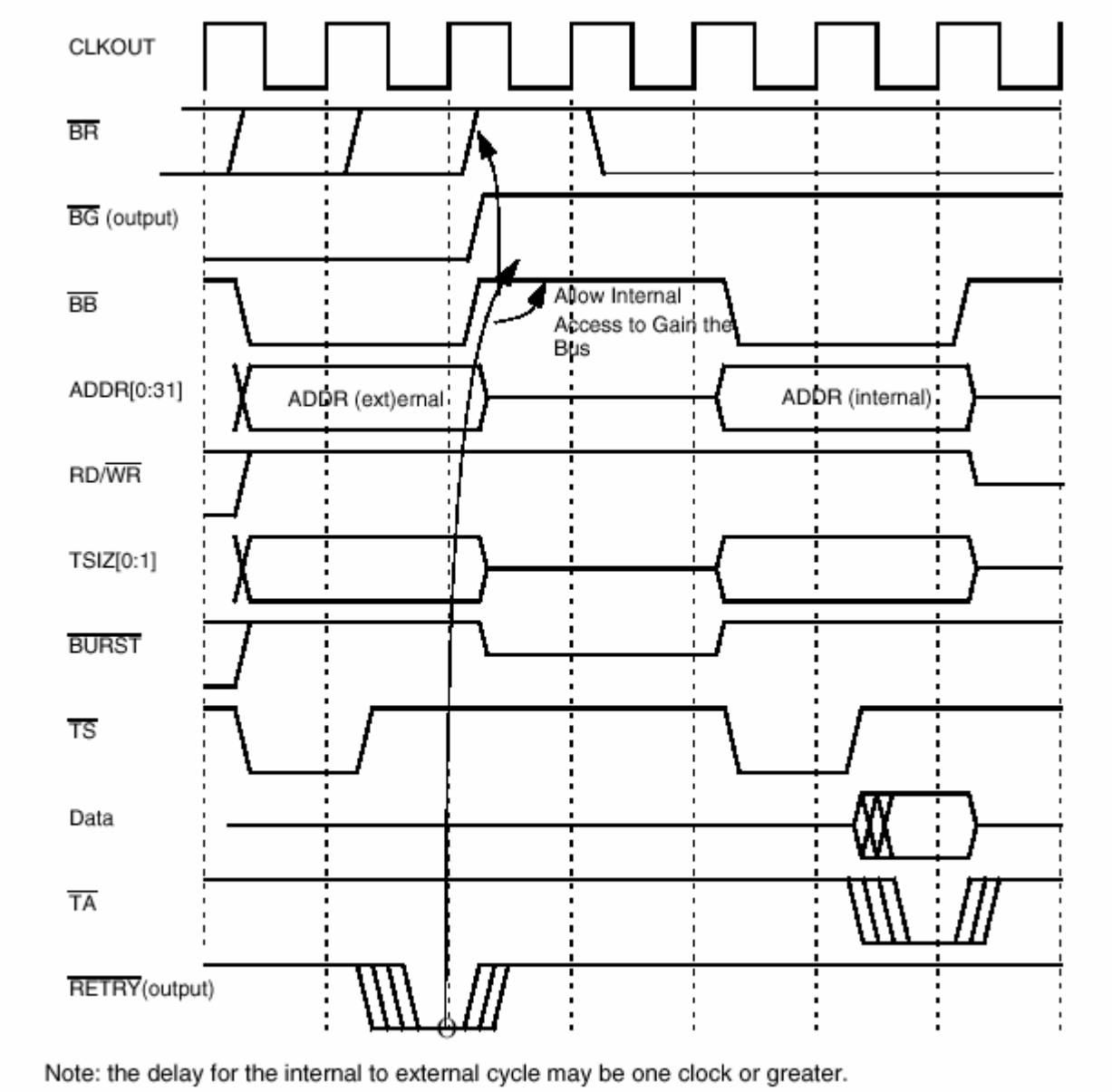

Figure 3.1 Retry of External Master Access (Internal Arbiter)

3.2 How is each device differentiated?

Each device within the system has to have its own unique identity to eliminate bus contention. The MPC5xx family has a register specifically for setting the address of the internal memory map. This register is called the Internal Memory Map Register (IMMR).

The IMMR[ISB0:2] field has three bits for setting the address thus allowing eight different locations however it must be noted that ISB[0] is not compared on internal

accesses therefore only 3 slaves are allowed in the system. The only possible way to use more than four devices (1 Master & 3 Slaves) in the system is to configure up to an additional four devices as masters that do not allow any other master to access their internal address space. Another reason for having different internal memory maps is that each controller is allowed to have its own interrupt handlers and reset routines.

In any case, the maximum capacitance of bus and clock signals should not exceed the value stated in the electrical specification for this family of devices.

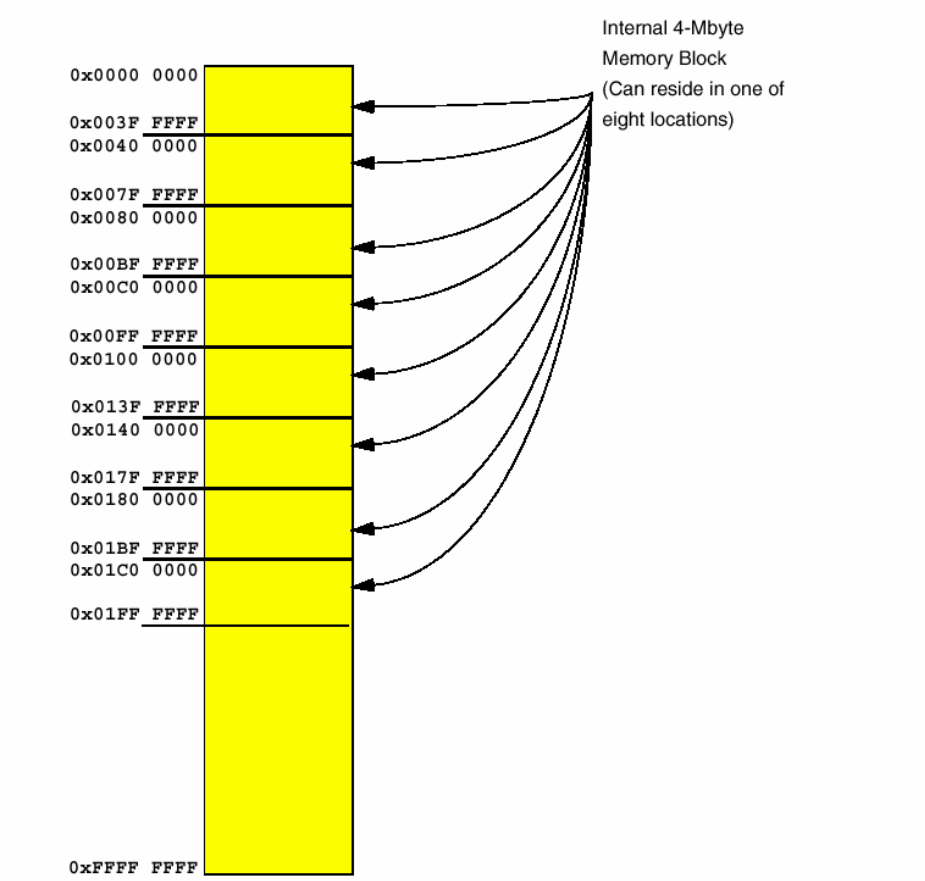

Figure 3.2 MPC5xx Internal Memory Map

e.g. In a dual controller system the master device could have its base address at 0x0 by setting IMMR[ISB0:2] = 0b000 and the slave device could have its base address at 0x400000 by setting IMMR[ISB] = 0b001.

3.3 How is Master/Peripheral/Slave Mode selected?

Each controller can be programmed as a Master, Slave or Peripheral device by configuring the EMCR[PRPM] (peripheral mode bit) and EMCR[SLVM] (slave mode bit) in the following manner:

- Master Mode (Default)
  From reset master mode is selected by default on all members of the MPC5xx family. The pertinent bits which control this functionality are EMCR[PRPM] = 0 &

EMCR[SLVM] = 0. When in master mode no external master can access the MPC5XX internal address space.

- Slave Mode (RCPU Core Is Active)
  To select Slave mode EMCR[PRPM] = 0 & EMCR[SLVM] = 1
  Application code can be executed in this mode. This mode also allows the slave access to the internal address space of the master..

  This mode is only valid if peripheral mode is not selected i.e. peripheral mode will override slave selection.

  For the devices that are configured to operate in slave mode, the MLRC[0:1] field should be set to “01” or “11” in order to operate the pin IRQ3/KR/RETRY/SGPIOC3 as RETRY. MLRC[0:1] bits are located in the SIUMCR register of the USIU and their function is to determine the operation of pin multiplexing.

- Peripheral Mode (RCPU Core Is Shut Down)
  If the EMCR[PRPM] bit is set, the core of the device is shut off and an alternative master on the external bus can access any internal module. At least one MPC5xx in the system must have this bit cleared as there must be at least one master.

  This mode is selectable from reset by setting D16(EMCR[PRPM]) of the Reset Configuration Word(RCW). In this mode the peripherals are accessed in the same manner as an external memory.Also in this mode there can be no access to the master as the core is shut down. This mode is also selectable by EMCR[PRPM] = 1.

3.4 RESET Configuration

In a single controller system the microcontroller is configured from reset using the Reset Configuration Word(RCW). Since the system has a common data bus, it is necessary to program RESET configuration in different ways in a multi-controller system. All MPC5xx devices in the multi-controller system have to be configured from reset using the RCW, however these words have to be taken from different places as each device will be configured differently.

There are three places the RCW can be selected from. Those being:

- Data bus
- Internal Flash
- Default(all 0s)

E.g. One possibility is to program one device with the external RCW (RSTCONF=0) and the other with the internal RCW (from internal flash memory, RSTCONF=1).

- Master = MPC563 RCW from internal flash
- Slave = MPC561 Data bus

Another possibility is that every MPC5xx in the system will take the RCW from its own internal flash (RSTCONF=1 for all devices). This way may be preferable, since there is no requirement to drive the external data bus with the RCW.

Once the system is built, but before any internal flash memories have been programmed, the RESET configuration word should be taken as default (all the bits are 0). This gives the user the opportunity to program the internal flash memories with their RCW. The system must be reset once more in order for the new RCW to be taken. The following bits of a RCW are of importance when configuring a multi-MPC5xx environment:

- PRPM
- EARB
- ISB0:2
- IP
- ETRE
- OERC
- FLEN

If reset is released simultaneously for all devices there will be an issue as the master will have no capability of programming the slaves before they boot. This means that by default the slaves will be configured as masters unless PRPM has been set in their RCW.

The connection of HRESET and/or SRESET pins between the different devices in the system is not required. Such a connection would mean simultaneous RESET for all MPC5xx devices in the system. However, it may be useful to connect HRESET or SRESET to an interrupt request(IRQx) input of the master in order to inform it about the reset event.

3.5 Clocking Strategy

A problem can arise if the CLKOUT signal from the master feeds the EXTCLK of the slaves. The problem arises when the master changes its operation frequency to system frequency. When this happens the PLL loses lock which means the slave devices have no clock to execute their own code. An option used to overcome this would be to use an oscillator module to feed all MPC5xx devices' EXTCLK pin. The MODCK[1:3] pins for the slave devices are set to 0x100 which selects extclk pin as a 1:1 clock source for the slave. This enables the clock on the slave devices to synchronise to the clock on the master device.

To overcome the above issues with reset and clocking an I/O port on the master should be connected to PORESET of all the slave devices. Once the master PLL has locked this port can be driven to release PORESET on all the slaves.

3.6 Initialisation

Note: The text in the following section considers a dual controller system such as the MPC563(master) / MPC561(slave) EVB.

On system boot up the master device may boot from the internal flash memory while holding the slave device in reset. The master device uses the internal flash RCW on boot up.

The slave can be initialised in 2 ways:

- The device is put into peripheral mode from reset using the RCW. The slave code is then copied from the master's internal or external flash memory to the slave's internal CALRAM or SRAM. (For this to work the exception vector table has to be relocated to the internal CALRAM.) Also the RCW has to select this relocation by setting D1(IP), D19(ETRE), D24,D25(OERC) to 1 in the RCW.

- The initialisation code is programmed into internal or external flash and the slave boots from this then switches itself to slave mode.

The master releases the slave from reset, the slave then samples the RCW from the external data bus. The RCW for the slave sets up the device in peripheral mode. In peripheral mode the master has access to the internal memory map of the slave. The master configures the slave and downloads the relocatable code into the slave's CALRAM.

The master releases the slave from peripheral mode into slave mode and the slave executes the downloaded code. In this configuration the slave does not have access to the master device.

The master then executes it own code from internal or external flash memory.

The compile and link tools(in this case DIAB from Windriver) used to link the application code provides the ability to partition the code between the slave and master devices. This provides the ability to relocate the slave code into the slave device from the master's flash memory. Refer to Application Note titled " Dual Controller Software Development For The Dual MPC561/3 EVB".

3.7 System Debug

The tool used for debugging this system was Lauterbach TRACE32. This tool has the capability for debugging multiple microcontrollers simultaneously.

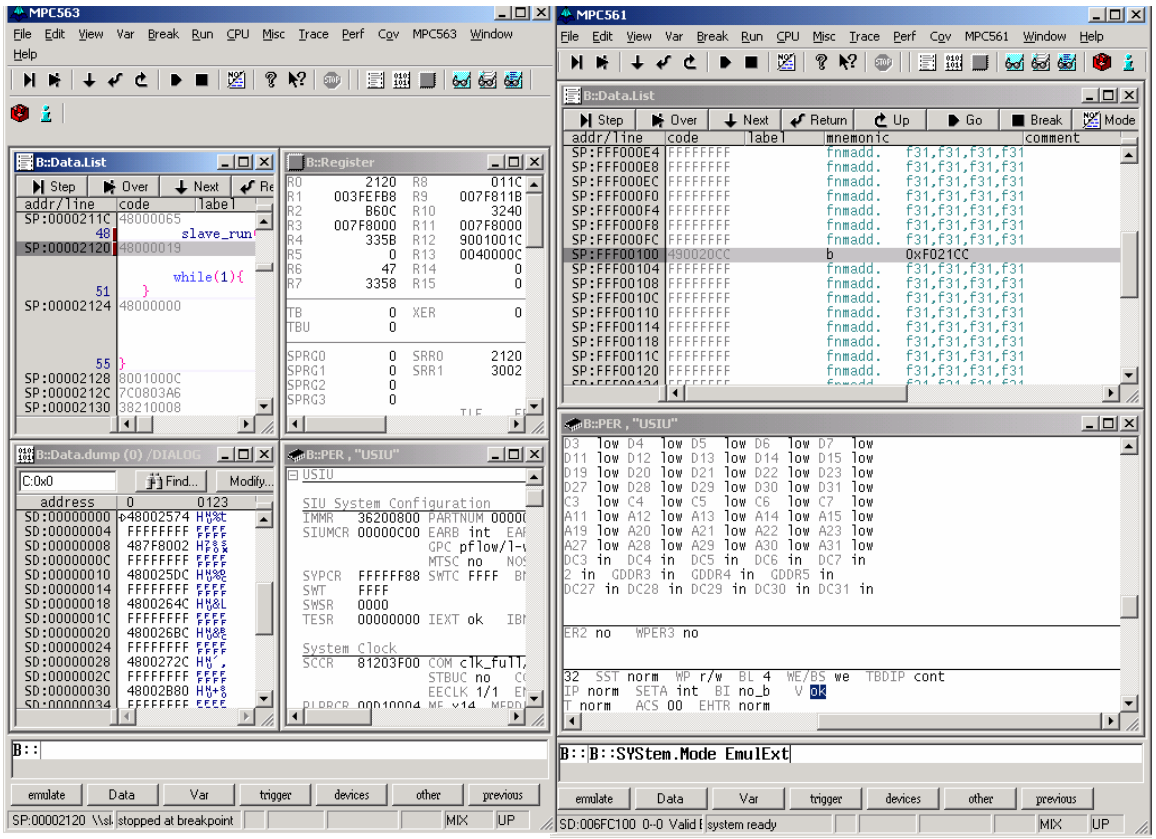

Figure 3.3

It can be seen from the figure above that with this tool there can be a window for each device displaying memory, code listing, registers and modules within the device. All debugging capabilities such as single step, run to breakpoint etc. are provided. TRACE32 also has the ability for NEXUS class 3 program trace.

This tool also allows the setting of breakpoints in one device which when reached will halt all other devices.

The figure below shows an example of the hardware setup used to debug the dual controller system.

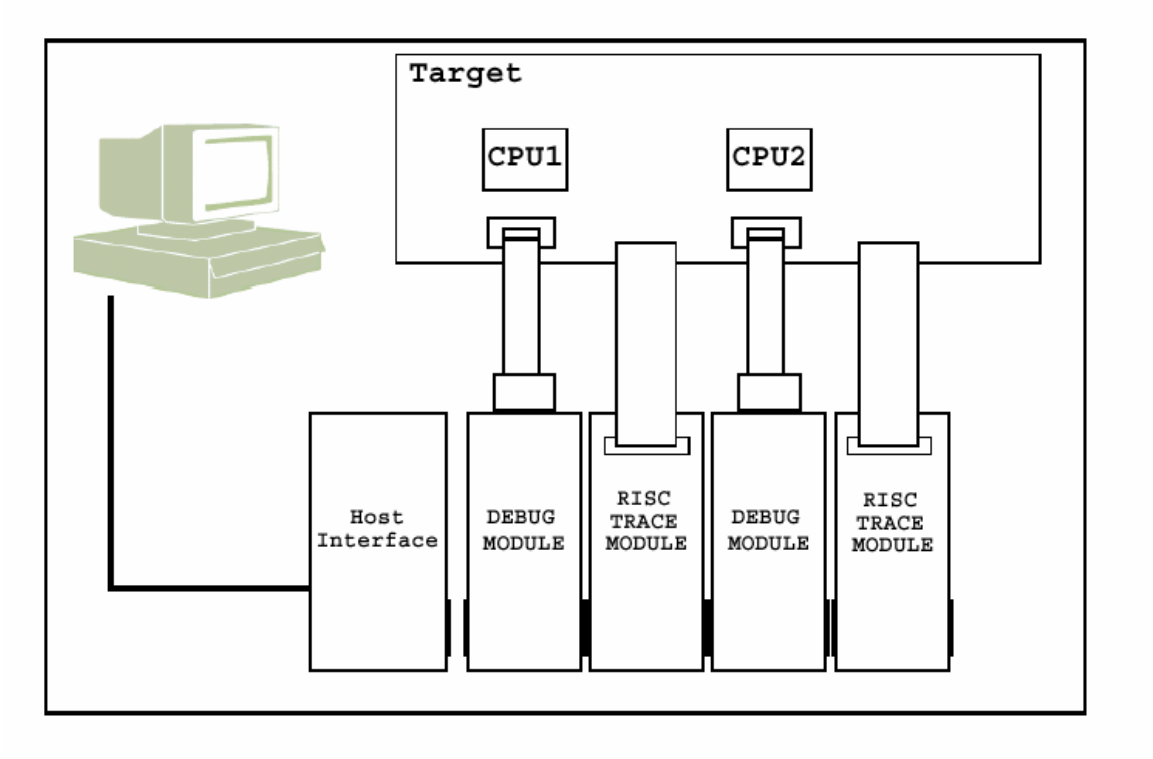

Figure 3.4

Note

The following applies to the MPC56x when using Lauterbach TRACE32. Lauterbach was used for the development of this apps note however any other toolchain could be used.

NEXUS will not work with Risc Trace as per the diagram above. PowerTrace modules (from Lauterbach) are needed in order to take advantage of the NEXUS capabilities. If the user is content using BDM debugging then PowerDebug modules should be used. This gives three possible combinations for a debug system:

1) PowerDebug(Device 1) : PowerDebug(Device 2)
2) PowerDebug(Device 1) : PowerTrace(Device 2)
3) PowerTrace (Device 1) : PowerTrace(Device 2)

With the USE= bitmask in the TRACE32 config files, PowerTrace is 1 bit wide and PowerDebug is two bits wide therefore the USE lines for each configuration above would be:

1) USE=1100, USE=0011
2) USE=110, USE=001 (or USE=100, USE=011)
3) USE=10, USE=01

Units are numbered from the point of connection downwards. Only the top most unit in the stack requires a connection to the host PC and this is the one that must be powered. Applying a power supply to other units in the chain will cause problems on the PODBus.

The debugger software knows which device should be controlled by monitoring the USE bitmask that is used to describe the controlled devices.

- 1 means control the device
- 0 means skip the device

For the EVB development the tool configuration file was amended to add the following definition to the config.32 file (MPC561):

USE=110

Add the following definition to the config.32 file (MPC563):

USE=001

Note

No device can be debugged when in Peripheral Mode as the core of the device is not active. Slave Mode should be selected for debug. Ensure D16(PRPM) of the RCW is not set when trying to communicate with the debugger.

Also ensure SIUMCR[MLRC] = 01 or 11 for RETRY functionality and EMCR[SIZEN] = 1 to allow access to the Module Configuration Registers.

4.0 Types of Accesses and Interrupt Handling

4.1 Types of Accesses

An external master can perform an access to any location within the internal address space, no matter if it accesses flash memory, CALRAM, any peripheral module, or USIU internal register.

There are some access attributes that can be driven on the MPC5xx internal bus when it is accessed by an external master. These are:

- Access size (byte, half-word, and word)
- Supervisor/user
- Instruction/data
- Reservation cycle
- Regular access/PPC register access
- Program trace

In order to drive these attributes on the internal bus, the EMCR register in the USIU must be programmed before the access. This means that if one attribute of an access which is going to be performed differs from current EMCR settings, then EMCR must be reprogrammed. Access size may be driven directly from the external bus if the SIZEN bit in EMCR is cleared, (i.e., the requirement to reprogram EMCR in every attribute change may be bypassed if the only attribute that changes is access size).

4.2 Interrupt Handling

The MPC5xx that is configured to operate in peripheral mode does not handle interrupts that are generated by its timers and peripherals. This task must be performed by a device in master or slave mode. In order to do so, the pin SGPIOC7/IRQOUT/LWP0 must be configured as IRQOUT (SIUMCR[GPC] = 11). In this way, any internal interrupt request will assert this pin..

The pin SGPIOC7/IRQOUT/LWP0 must be connected to one of the interrupt request inputs (IRQ[1:7]). IRQ0 should not be used as it is an NMI and will result in application code jumping to 0x100(reset exception).

NOTE

These pins may have another function due to pin multiplexing. It is the responsibility of the user to program pin multiplexing correctly.

5 Switch Between Operation Modes

The user must be careful when switching between operation modes. Namely, when two cores are active and the mode of one device is switched from slave to master, or slave to peripheral A situation may arise where a slave device is performing some internal accesses that have not yet completed. In this instance when the master tries to place the slave device into peripheral, or master mode it may cause the external access to be terminated by RETRY.

NOTE

If the pin IRQ3/KR/RETRY/SGPIOC3 is not programmed to operate as retry in advance, the system will hang.

6 MPC555/MPC56x Differences

The MPC5xx family consists of the MPC555 and MPC561-6 devices. The differences between the MPC555 and MPC56x are minimal with regard to multi-controller operation. The major difference is that the MPC555 cannot have its vector table relocated to internal SRAM therefore application code cannot be executed from internal RAM from reset.

The other difference is that the bus voltages are different, the MPC555 is 3.3V and the MPC56x is 2.6V therefore it is recommended to use multiple controllers of either the MPC555 or the MPC56x.

7 Performance Impact On a Dual Controller System

The following analysis considers the MPC561/3 as per the EVB. The points highlighted also apply to the MPC555 and MPC565.

2 x MPC561:

There is a performance hit when the master accesses the slave. This is because the master needs the EBI to do this. This means it cannot fetch instructions from external memory at the same time as accessing the slave. Note that if both devices had to share the same external memory for instructions the overall system performance would be less than that of a single device due to the bus arbitration required.

As the MPC561 has no internal flash memory it cannot use the flash RCW. As the master device will be executing instructions from external memory it makes sense for the master to take the default RCWand allow the slave to take its from the data bus.

1x MPC563 & 1x MPC561:

There is still a performance hit if the master accesses the external slave. The data access to the slave freezes the internal UBUS,which prevents instruction fetches from internal flash.

The master can now take its RCW from the internal flash memory. The slave code size is limited to the size of the internal CALRAM. This is a cost effective solution.

2 x MPC563:

As above there is the same performance hit.

There are more options for RCW as each device can have its RCW located in internal flash memory. This is a simpler solution as both RCWs can be stored internally and the code for each device can be stored internally.

If the slave code is bigger than the internal CALRAM then 512K flash memory is available. This solution will be slightly more expensive than the others.

8 EVB Overview

The Dual MPC561-4 Evaluation Board has the following specification:

- MC33394 (PowerOak) Regulator
- 20MHz Xtal
- 2 x MPC561/2/3/4 sockets
- 2 x Mode Config Switches(MODCK settings) + 32 RCW Switches
- 2 x NEXUS + 2 x BDM
- 3 x Mictor Connectors
- 6 x CAN(5 stand alone and 1 integrated with the MC33394)
- 2 x RS232 9 way D type connectors

- 1 x AMD AM29BDD160 (512K x 32) Flash(socketed)
- 1 x ST M58BW016 (512K x 32) Flash
- 1 x IDT IDT71T016(64K x 16) SRAM(No Burst)
- 1 x CYPRESS CY7C1339(128K x 32) SRAM(Burstable)
- 1 x NEC uPD43256(32K x 8) SRAM(No Burst)
- 1 x ALTERA EPM7064B CPLD(68 I/O, 1250 gates)
- 5 x General Purpose LEDs

When designing a multi-MPC5xx board such as this, care should be taken to ensure that both devices are placed as close to one another as possible and routed very tightly. The CLKOUT and bus traces should have 50 ohm impedances.

It is also critical to ensure that the buses are not overloaded and adhere to the electrical specification for the devices used in order to operate up to 66MHz.

Refer to the schematics, board plots and user manual for more detail.

Note

An 8 layer board was used for this design to minimize noise and ground bounce. Each board design is application dependant therefore the board designer must take all signal integrity and EMC requirements into account when designing the PCB(Printed Circuit Board).

9 Conclusion

This application note provides details of how to build a dual processor system based on the Motorola MPC5xx family of microcontrollers. This configuration allows increased I/O capability and performance with a minimum number of components within the embedded system.

When designing the system, performance implications should be considered carefully when partitioning the software. As described in section 7 when the master fetches data or instructions from the internal memories or peripherals of the slave device the UBUS gets stalled. To minimize the code execution performance hit on the master the slave device should collect all the data the master requires and store this in an area of internal CALRAM. This allows the master to fetch multiple data in one time slot (not bursting), reducing the overall impact on the system.

Although the performance impact has been highlighted it must be remembered that the overall system performance has increased. This is due to the fact that there may be more I/O being controlled and seperate applications running simultaneously.

The schematics, layout and user's manual for this board can be found on the external web. Please add link.

HOW TO REACH US:

USA / EUROPE / Locations Not Listed:

Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

JAPAN:

Motorola Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu Minato-ku
Tokyo, 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.
Silicon Harbour Centre
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

HOME PAGE:

http://motorola.com/semiconductors

AN2667/D, **Rev 0, 12/2003**