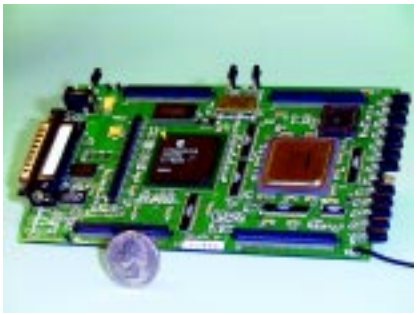
Debugging with the Siemens C167 E2 Bondout Microcontroller

Introduction

There are several possible methods to emulate a microcontroller and each method has its benefits and drawbacks. Siemens chose the bondout controller method for its C166 family and Enhanced Hooks for its C500. This article will focus on the E2 bondout that emulates all C166 family members except the C166 itself. The C166 has its own bondout.

Nearly all emulators made for the C166 family utilize the E2 bondout although there are a few that use standard production chips. The E2 replaces the



older E (or E1). The Nohau EMUL166 is shown without its case. The E2 bondout is on the right side and has a silver top.

Internal or External Mode

The C166 can operate in either single-chip or external mode. In External mode, address and data buses are used to access external memory via ports P0, P1 and P4.

In single-chip, the instructions are fetched and data is accessed from on-board ROM or FLASH and RAM. There are no data and address buses to the outside world that an emulator can use. P0, P1 and P4 can then be used as general purpose I/O.

External mode allows an emulator to see operations since the buses are visible. The emulation must be interrupted in order to read internal registers and RAM. Real-time operation is therefore lost. A bondout or Enhanced Hooks fixes this.

What the E2 Bondout Offers

The E2 bondout operates in both external and single-chip modes. It has extra external buses to provide the emulator access to internal resources and information even when the address and data buses are not available as in single-chip mode. These buses are called IA, ID, OA, OD and XBUS. Most emulators use these buses to obtain detailed internal information and they are also displayed in the Nohau trace memory. Nohau displays the signals it derives from these buses.

You can trigger on and record in trace memory internal events in real-time. These internal events are impossible to see without a special emulation chip such as a bondout or Enhanced Hooks. For more information see www.nohau.com under Technical Publications.

The E2 offers internal triggering, ROM simulation, and detailed trace recording all without intrusion into the emulation.

Special Bondout Buses

Five buses available on the bondout are optionally visible in the trace display as shown in the screen shots below. The descriptions of the buses are general in nature and are not true for all cycles.

IA Bus: Instruction Address

ROM Instruction Address Bus. This 16 bit bus shows the address of instruction transfers on the internal ROM bus. This is visible only in single-chip mode.

ID Bus: Instruction Data

ROM Instruction Data bus. This 32 bit bus displays instruction and data over the ROM bus. Effective only for single-chip.

OA: Operand Address

This 18 bit bus displays read and write addresses. This and the OD bus are shown in the second screen shot. The two * characters shown in the OA field signify valid write cycles that were executed. This field can be used to distinguish between executed opcodes and pipeline flushes.

OD Bus: Opcode Data

This 16 bit bus displays the data word of an opcode. Opcodes of injected instructions (into the pipeline) are visible here. The OD and OA buses are very useful.

XBUS

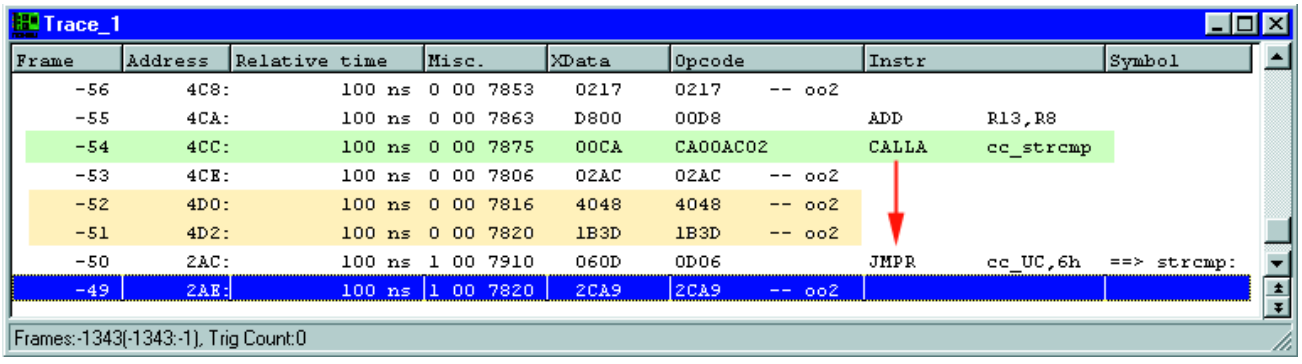
The XBUS is an internal representation of the external bus. This 16 bit bus displays the data on the internal XBUS as well as data writes to external memory.

Misc.

This field displays bit flags that Nohau derives from the bondout controller pins. These signals are useful to determine certain situations the controller is in. Examples include jump taken, jump cache hit, R/W, trigger event, operand size, and instruction or data. Eight I/O signals and the pipeline instruction counter are also displayed in this field.

Instruction Pipeline

The C166 family has a 4 stage instruction pipeline that can be demonstrated



in the trace memory. The four stages are fetch, decode, execute and write back. If an instruction is fetched, but not executed because of a change in program flow (usually from a jump or call), this instruction(s) are flushed from the pipeline and new ones loaded.

These flushed instructions are tracked in order that there is no false triggering on them. This tracking is displayed in the misc. field. Unexecuted instructions must not be used as trigger qualifiers.

It is important to be able to detect pipeline effects since there is a definite delay from when an instruction is fetched to when it performs its operation. The E2 provides enough information to the emulator in order to decode this data.

Pipeline Prefetch & Flushes

The screen shot below resulted from this code sequence being fetched or executed:

```
0004CA: 00D8          ADD     R13,R8
0004CC: CA00AC02      CALLA   cc_strcmp
0004D0: 4840          CMP     R4,#0h
0004D2: 3D1B          JMPR    cc_NZ,1Bh
0004D4: E6FC2050      MOV     R12,#5020h
```

The ADD at 4CA is executed and the CALLA is taken. Program flow continues at *strcmp* at address 2AE.

Frame -54 (green) is the fetch of the CALLA opcode. It is obviously executed at frame -50 with the appearance of JMPR instruction, 400 nsec later. Since each frame records one word, Frame -54 actually contains the first word of the CALLA, namely CA00. The AC02 is put there for readability. Frame -53 contains the second word 02AC. The oo2 means a 2 byte movement. Remember, the C166 is Little Endian so the data came off the bus in this order and Nohau swaps it for convenience and readability for the user.

Frame -52 and -51 are prefetches of the CMP (4840) at 4D0 and the JMPR (3D1B) at 4D2 ! These are flushed instructions that were prefetched, but not used as the program flow changed as a result of the CALLA instruction. Note all this information is present in the Nohau trace plus a lot more. Interestingly, Frame -49 is also a flushed instruction.

Jump Cache

The C166 family has a jump cache mechanism to speed conditional loops. The branch target instruction is stored in the jump cache and each time it is needed, it is fetched from here and not from slower program memory. This branch target instruction is injected directly into the decode stage of the pipeline. The trace screen shown below was derived from this code segment:

```
000206:   A55AA5A5    DISWDT
00020A:   E080        MOV   R0,#8h
00020C:   26F00100    SUB   R0,#1h
000210:   EAE00C02    JMPA  cc_UGT,20Ch
000214:   CC00        NOP
```

The Branch Target Instruction is the SUB at 20C. The DISWDT is the disable watchdog timer. R0 is initialized with 8 and is decremented by 1 by SUB. JMPA tests R0 and jumps back to SUB until R0 equals zero. In this case, a trigger was set for R0 to equal 5, then halt emulation. This fact is not clear in this trace window.

Jump Cache Hit

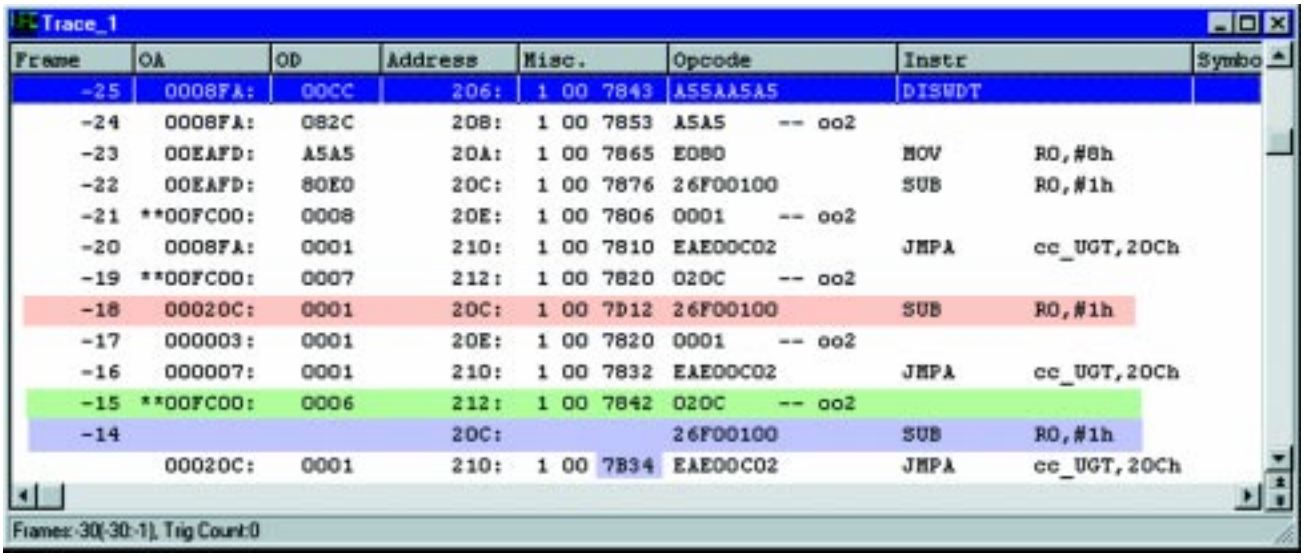
Siemens states that when the cache jump instruction passes through the decode stage of the pipeline for the first time, the branch target instruction is fetched and stored in the jump cache. This is visible in the trace window as described here:

When the second nibble of the Misc. word contains a D (i.e. 7D12), this means the instruction is placed into the jump cache. When this nibble equals B, this signals a jump hit. This instruction is then taken from the cache and injected directly into the pipeline.

Frame -20 contains the first jump and Frame -14 contains the cached SUB. The 7D12 and the 7B34 indicate this. Frame -14 actually has the 7B34 on the following line. Note there is no data in either the OA or OD fields. This is because no fetch occurred.

Two for the Price of One:

The pipeline can allow the CPU to perform two operations at the same time and this is also visible in the Nohau trace. Frames -21, -18 and -15 have two events at the same time.



Note the ** in the OA column which indicates a valid write cycle. Where is this write ? It is to memory FC00 which is equal to R0 because the CP register (Context Pointer) points to FC00. This is the base of the register bank. Recall R0 is being decremented by SUB and you can see this in the OD column on Frames -19 (R0=7) and -15 (R0=6). Frame -21 (R0=8) is the initialization by the MOV at address 20A. This is the first event.

During this time there is also a fetch of the data part of an opcode. In each of Frames -21, -18 and -15, under the Opcode column is the second half of the opcode fetch of the frame before it.

The Opcode value for Frame -19 contains the data or second word for the JMPA at Frame -20. It is 020C swapped to 0C02.

Triggering on Internal Data

The E2 bondout allows triggering on internal memory areas such as RAM, SFRs, ROM and XRAM. Programs can be executed in the internal RAM, and these instruction fetches can be cause triggers. Triggering can start or stop the trace recording, provide an external output to trigger an oscilloscope or logic analyzer or halt emulation. This is an important feature. The E2 bondout provides a way to peek inside the controller to increase your debugging power.

In the example below, the trace recording was halted when R0 was changed to 5. This is impossible to without a bondout chip or by replicating the CPU in the emulator hardware (which is not easy).

A trigger can have up to 50 addresses ANDed with 50 data values or ranges: and there are three triggers in the Nohau EMUL166 ! Plus a Filter to specify what cycles are recorded in the trace memory.

Two Chip Emulation

The E2 bondout contains two CAN modules (hence can support the C164) and 3 XRAM modules of 2K, 4K and 6K. Peripherals that are not on the E2 such as the C163 SSP are supported with a C163 on a daughterboard in Emulation mode. This turns off the 163 CPU but the XBUS peripherals are still active and available to the E2 through the external bus. New derivatives can be supported in this manner. Other peripherals such as the RTC, timers, CAPCOM and PWM are included in the E2 and are accessed normally.

Single Chip Mode

The E2 bondout supports internal ROM or FLASH devices from 32K to 512K. Remapping of this ROM to segment 1 is fully supported. The ROM is simulated with fast RAM in the emulator. There are menu items in the emulator software to inform the E2 it is in ROM mode, how much ROM it must simulate and if remapping is needed.

The user program must set the ROMEN bit and perhaps the ROMS1 bit in the SYSCON register during the initialization period. This is easy to do. These values can be manually poked in with the emulator for small experiments.

The data path for the ROM is 32 bits and this will be reflected accurately in the trace memory in the Opcode column. All ports are available for the user: none are used for address and data buses. The appropriate information is obtained from the IA and ID buses.

Emulation Accuracy

The E2 directly supports the C167, C164 and C161. It can further support the C165 and C163 and other derivatives except for the SAB 80C166 itself.

The E2 is not an exact copy of any of these controllers so some differences will occur. The CPU core is exactly the same so this reduces differences substantially.

Conclusion

Experience has shown the E2 to accurately emulate C166 family members up to its design speed of 33 MHz. A well designed In-Circuit Emulator can substantially reduce your debugging time and grief by using the power of the E2 bondout from Siemens. Nohau is committed to providing the user with all E2 features as technically feasible.

Robert Boys
VP Technical marketing
Nohau Corporation
Campbell, California
(888) 886-6428
rboys@nohau.com