February 2, 1993

Application Note

Interfacing the MC68020 to a Slave MC68302

Overview

The MC68302 serial channels maybe used in systems where the combined rate on all three serial channels total more than 2 Mbps fullduplex. In such a case, if a true protocol is being executed on the data streams, and the data flow is continuous, the M68000 core may be thelimiting factor in the system performance. Thus, it may be advantageous to disable the M68000 CPU on the MC68302, and use a faster processor instead. This application note discusses an example of how to disable the M68000 core processoron the MC68302, and use instead, a higher performance MC68020 processor.

The natural choice for this application is the MC68020 or MC68030.

The EC020 is about 2.5 times faster than the M68000 at the same clock speed. Since the MC68020, or more specifically thelower cost 68EC020 is available in generally the same general speed range as the M68302, we willconsider a 68EC020 to MC68302 interface at the same clock rate. Note, however, that it would also be possible to takea 33 MHz MC68020 and interface it to a 16.67 MHz MC68302, and therefore obtain over 4 times the processing performanceof the M68000 core.

Interface

The MC68EC020 to MC68302 interfaceis shown in figure 1. The following paragraphs provide explanation for the diagram.

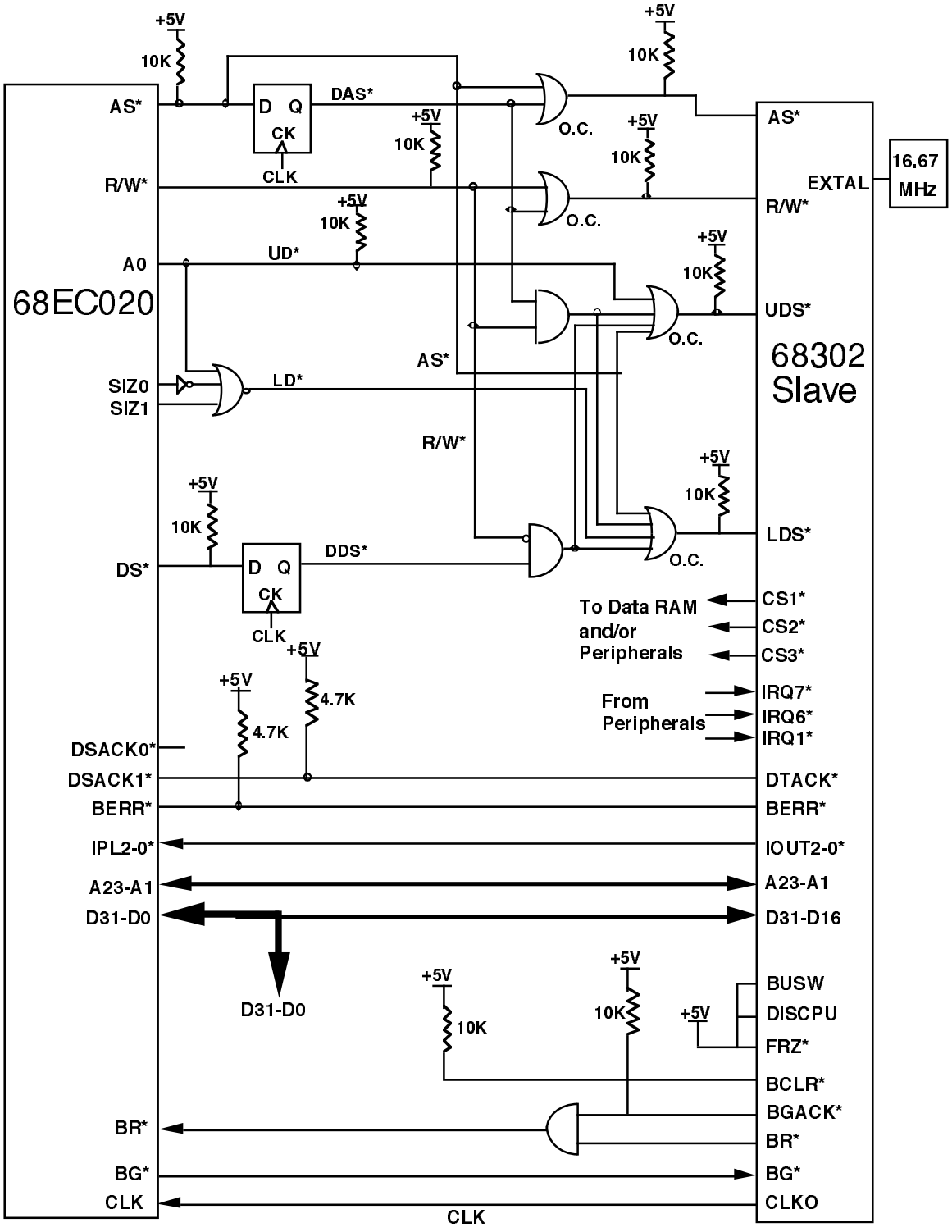

Figure 1: MC68EC020 to MC68302 Interface

The clocking for thesystem is accomplished with the help of the MC68302. The 16.67 MHz oscillator is connected to the EXTAL pin of the MC68302, and the CLKO linof the MC68302 becomes the system clock, and is connected to the CLK pin of the EC020. The reason for this

choice is that the MC68302 references all its timings to the CLKO output pin, not the EXTAL input pin, while the EC020 references all timings to its CLK input pin. Thus, in this configuration, the potential clock skew between the two devices is minimized. It would also be possible to clock both the MC68302 and the MC68EC020 with the same clock source, but the MC68302 CLKO skew of 2-11 ns would have to be considered in all the timings. This becomes increasingly critical as the system clock speed increases. The design shown would be easily scaleable to 20 MHz using a 20 MHz MC68302, and a 25 MHz MC68EC020 (the next higher speed EC020 version available).

The data bus is easily configured. The MC68302 is placed on the upper 16 bits of the EC020 data bus (D31-D16) as required for EC020 peripherals that are to be treated as 16-bit ports. The EC020 performs all accesses to the MC68302 as a 16-bit peripheral. Thus, the DSACK1* pin on the EC020 is the only pin that is required to be asserted during the data acknowledgement phase of the cycle. This selects a 16-bit port. The DTACK* line of the MC68302 is an output during all accesses to the MC68302, and is therefore connected to the EC020 DSACK1* line. Both devices three-state their data buses when they are not the bus master, thus the buses may be directly connected.

The address bus matches well between the EC020 and the MC68302. Both devices support up to address line A23. The EC020 however, supports A0 in combination with the SIZ1-0 pins to determine which byte should be accessed. This has to be mapped to the MC68302's UDS* and LDS* lines, which are the upper and lower byte selects for accesses to and from the MC68302 for each word accessed.

Notice that no chip select to the MC68302 is required! This is because the MC68302 decodes the full address space to determine whether it is being selected. Actually the MC68302 does have a chip select of sorts, but it is an output pin -- IAC (not shown). This pin is often very helpful in debugging. It asserts whenever an internal location of the MC68302 is accessed (either by the MC68302, or another device).

The MC68302 only reserves locations $F0-FF for fixed location registers. This area contains the very important base address register (BAR) that determines where the rest of the MC68302 registers (a 4-K byte address block) will be located. The 4K byte block should be programmed to exist at an area that does not overlap any other memory or peripherals in the EC020 address map. However, if low memory exists, it is likely that the $F0-FF space will overlap the low memory. Perhaps the best solution is to avoid using low memory except for the initial boot ROM, but if this is not possible, three solutions to the low RAM problem exist:

(1) Go ahead and add the chip select decode logic for the MC68302 in the $F0-FF address space, to distinguish between accesses to RAM, and accesses to the MC68302. This is the most complete solution, but also the most costly.

(2) Only allow write accesses to the $F0-FF area. This will prevent bus contention in the F0-FF area, but the DTACK* from the memory and the DTACK* from the MC68302 should be checked for timing compatibility. The memory DTACK* should not generate earlier than the MC68302 DTACK*. This can be avoided by using the IAC line of the MC68302 to disable the memory generated DTACK*. The reason this option is viable, is that many MC68302 applications can avoid needing to read this area of the MC68302. Most of the activity is simply writing the desired MC68302 configurations.

(3) Use an MC68302 chip select to decode the low memory RAM. The MC68302 chip selects have the convenient property, that they will not assert on accesses to any existing register location on the MC68302. (The MC68302 chip selects will be discussed below).

The bus error line may be directly connected between the two devices. The MC68302 will support the "hardware watchdog" function that will terminate any "stuck" EC020 or MC68302-generated bus cycle. The AS* line input to the MC68302 is monitored by the hardware watchdog. If the AS* line stays asserted for too long (a programmable value such as 1024 clock cycles) the BERR* pin will be asserted to terminate the bus cycle. This feature eliminates the need to provide this function externally in a EC020 based system.

Since the EC020 device in the design is an EC020 device, rather than the standard EC020 device, a two-wire arbitration scheme (BR* and BG*) is used rather than the standard three-wire arbitration scheme (BR*, BG* and BGACK*). Since the MC68302 supports the three-wire scheme, an interface must be provided that keeps the MC68EC020 BR* line asserted during the entire time that the MC68302 wants, or has, the bus. The bus arbitration is only necessary for the MC68302 when its IDMA or SDMA channels need access to the bus. The bus arbitration signals are not required for accesses by the EC020 to the MC68302. The BCLR* function on the MC68302 is not used in this design, meaning that both the SDMA and IDMA functions of the MC68302 will keep the bus as long as needed, once the bus is obtained. (The BCLR* function could be used to suspend the IDMA transfers if some other device needed the bus quickly.)

The MC68302 supports a 3 line encoded interrupt priority output on the IOUT2-0 pins. These lines can be directly connected to the IPL2-0 inputs on the EC020. In this way, the MC68302 can provide a level 4 encoding of its own internal interrupt sources, and can act as an interrupt controller for three other devices in the system at levels 1, 6, and 7. This is shown by the IRQ1*, IRQ6*, and IRQ7* pins which are interrupt sources from other peripherals in the system. It is important to program the IRQ pins to respond to levels rather than edges. By programming them to levels, the interrupt encoding to the EC020 will be cancelled, when the value on the IRQ pin returns to the high (deasserted) state. If more than three external sources are needed, the PB8, PB9, PB10, and PB11 pins on the slave can be used for additional interrupt sources. The PB lines are edge triggered.

If still more external interrupts are needed, the IRQ* lines can be programmed to work as IPL lines, allowing a 3 line encoding input, and thus 6 external interrupts (level 4 is never allowed to be selected externally).

In any case, the MC68302 can provide vectors to the EC020 for all level 4 internal interrupt sources, and levels 1, 6, and 7, by programming the vector generation enable (VGE) bit in the system configuration register (SCR) on the MC68302. This is the recommended action in a MC68302 slave application to the EC020. (Note that in the previous case of a master MC68302 and multiple slave MC68302s, the programming of the VGE bit was not recommended.)

The bus width (BUSW) pin on the MC68302 is pulled high to configure the MC68302 for 16-bit bus, rather than 8-bit bus operation. Also, the FRZ* pin is pulled high to disable the freeze logic on the MC68302.

The only other aspect to cover in the interface from the EC020 to the MC68302, is the generation of the AS*, R/W*, UDS* and LDS* signals for the MC68302. The main reason for the additional logic shown is to deal with the fact that the EC020 generates its AS*, R/W*, and DS* one half clock earlier than what is desired by the MC68302 on synchronous accesses. Figure 2 shows this in the MC68EC020 read access timing.

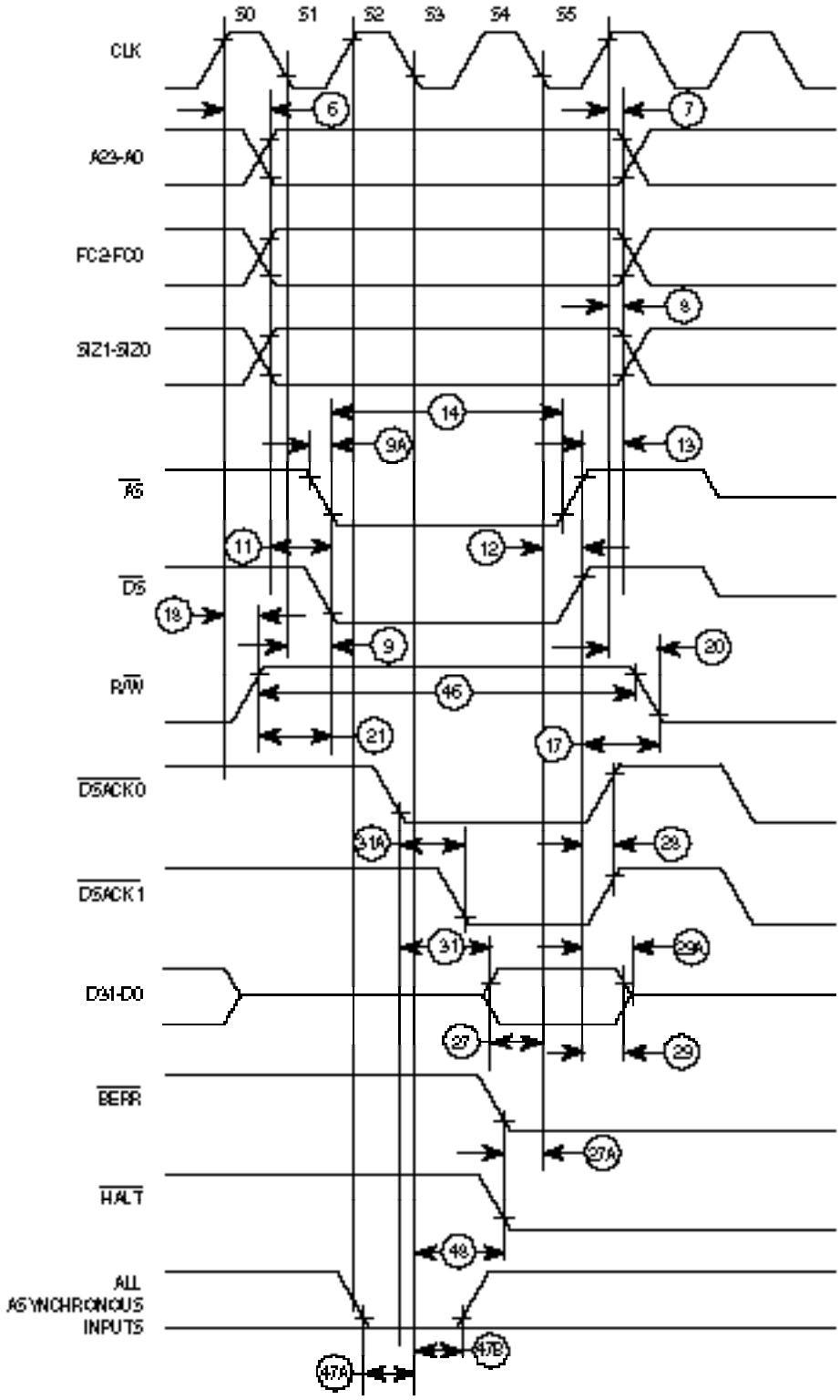

Figure 2: MC68EC020 Read Cycle Timing Diagram

The goal of this design is to allow the EC020 to access the MC68302 synchronously. This means that the synchronous access mode (SAM) bit in the MC68302 should be set on the first access by the EC020. The first access will be a three wait state asynchronous access to the MC68302 (four wait states from the standpoint of the EC020), but the synchronous timings shown is appropriate for either access. Because of the DTACK* timing on a read cycle, it's advantageous to set the external master waitstate (EMWS) bit on the MC68302, giving from the MC68302 standpoint, a one wait state read and a zero wait state write access. From the EC020 standpoint, this results in 2 wait state reads, and one wait state writes.

Now for the specifics. The MC68302 AS* needs to be delayed 1/2 clock from the EC020, but the 020 deassertion time is fine.

The R/W* pin timing is fine for a read (the pin remains high), but must be delayed 1/2 clock from the 020 AS* pin falling edge on a write. The deassertion timing is OK.

The UDS* and LDS* pins are asserted based on the particular byte the EC020 desires to access. Both UDS* and LDS* should be asserted 1/2 clock after the 020 AS* on a read, and 1/2 clock after the 020 DS* on a write. These pins may be deasserted at the same time as the rising edge of the 020 AS* pin.

Note that the final outputs to the AS*, R/W*, UDS* and LDS* pins are with open collector gates. When the MC68302 takes mastership of the system bus, these control lines may be used to control the access of the MC68302 to memory. This makes particular sense if the MC68302 chip selects are used.

In slave mode, three chip selects are left available for use in the system. They are the CS1*, CS2*, and CS3* pins shown. For accesses originated in the MC68302 (i.e. the IDMA or SDMA) they can operate as fast as 4 clocks (a zero wait state access from the standpoint of the MC68302 DMAs). However, they can also be used to select memory accessed by the EC020. The only trick is the EMWS bit mentioned earlier. This bit, when set (and we are setting it in this example), adds one wait state to any access by an external master that uses a MC68302 chip select line. This is because the chip selects can be asserted earlier by an MC68302 on-chip master than by off-chip logic. The EMWS bit helps counter this inevitability. Thus, from the EC020's perspective, the fastest access by a MC68302 chip select is 2 wait states. Also, it should be noted that the three MC68302 chip selects have to be enabled by software -- they do not enable automatically after reset. All these things point to the MC68302 chip selects being used for applications such as slower peripheral chipselects, or RAM chip selects -- if the speed of access by the EC020 to the RAM need not be faster than 2 wait states.