Freescale Semiconductor
Application Note

Document Number: AN2293
Rev. 2, 09/2006

Compact Flash Interface for the MPC8245

by *Gary Milliorn*
CPD Application
Freescale Semiconductor, Inc.
Austin, TX

This application note describes the connections necessary to attach a compact flash device to the MPC8245 embedded microprocessor. The compact flash (CF) standard maintained by the CompactFlash Organization describes a standard way to connect and communicate with compact memory, I/O, and disk drive modules. These devices are often used in embedded systems to provide low-cost mass storage or upgradable OS installations (simply by replacing a CF device and rebooting). CF devices can easily be upgraded with socketed devices or permanently attached with a ‘caged’ connector. CF memory cards are available with up to 512 Mbytes of nonvolatile memory storage and CF miniature hard-disk cards that feature up to 1 GB of storage. There are three classes of CF cards:

- PC card memory
- PC card I/O
- True IDE

The PC card memory and true IDE interfaces are popular with embedded systems because of the predictable nature of the devices—raw storage. PC memory cards implement both the memory and true IDE interfaces, and disk drives implement only true IDE interfaces. This application note discusses the design issues for a true IDE device.

Contents

All three types are generally described, with differences noted, to assist designers who want only PC memory or PC I/O devices. The connections of the CF card in PC memory and PC I/O modes are similar to those of the IDE mode, but with a different register interface and slight changes of interface signals.

Finally, CF devices feature the ability to be hot-plugged (that is, inserted and removed while power is applied). This is not required for most embedded applications, and requires special power sequencing and dynamic signal reconfiguration logic that is far beyond the purpose of this application note.

NOTE

The VHDL in this application note is compiled and verified with a software test bench but has not been verified in hardware. There may be significant errors in this application note, or the CF test bench may not have revealed latent errors in the example code. Therefore, this application note should be treated as general design information for creating a CF interface and not as a drop-in component.

All the software contained in this application note is copyrighted 2002 by Freescale, Inc. and may be used freely by Freescale customers for use on MPC8245-based systems, as long as the copyright notice remains present in each literal or derived module or source file. The code can be freely modified to suit customized applications. You are not obligated to make your own proprietary modifications to the code available to the public, nor to maintain copyrights of your own modifications.

1 Conventions

The CF specification uses a ‘-’ prefix to indicate an active-low signal (-XX), while this application note uses an overbar ($\overline{\text{XX}}$). For VHDL equations, the suffix ‘_B’ indicates an active-low signal, otherwise active-high is assumed.

2 Connections

Table 1 summarizes all three signal types of the CF devices.

Table 1. CF Device Connections

Signal	Description	Type	Pins	Interface			Notes
				PC Card Memory	PC Card I/O	True IDE	
Common Signals							
A[10:0]	Address	I	8, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20	Common			1
$\overline{\text{CD}}$[1:2]	Card detect	O	26, 25	Common			
$\overline{\text{CS}}$[0:1]/$\overline{\text{CE}}$[1:2]	Chip selects	I	7, 32	Common			2
$\overline{\text{CSEL}}$	Cable select	I	39	Common			

Table 1. CF Device Connections (continued)

Signal	Description	Type	Pins	Interface			Notes
				PC Card Memory	PC Card I/O	True IDE	
D[15:0]	Data bus	I/O	31, 30, 29, 28, 27, 49, 48, 47, 6, 5, 4, 3, 2, 23, 22, 21	Common			3
$\overline{\mathrm{INPACK}}$	Input acknowledge	O	43	Common			
$\overline{\mathrm{IORD}}$	I/O read strobe	I	34	Common			
$\overline{\mathrm{IOWR}}$	I/O write strobe	I	35	Common			
VS[1:2]	Voltage sense	O	33, 40	Common			
$\overline{\mathrm{WAIT}}$/IORDY	Wait/ready	O	42	Common			
$\overline{\mathrm{WE}}$	Write enable	I	36	Common			
Unique Signals							
$\overline{\mathrm{ATASEL}}$	IDE mode enable	I	9			✓	
$\overline{\mathrm{OE}}$	Output enable			✓	✓		
$\overline{\mathrm{REG}}$	Register select	I	44	✓	✓		
RESET	Reset (active-high)	I	41	✓	✓		
$\overline{\mathrm{RESET}}$	Reset (active-low)					✓	
$\overline{\mathrm{IOIS16}}$/$\overline{\mathrm{IOCS16}}$	16-bit access	O	24	✓			
WP	Write protect				✓	✓	
RDY/$\overline{\mathrm{BSY}}$	Ready/busy	O	37	✓			
$\overline{\mathrm{IREQ}}$	Interrupt request				✓		
INTRQ	Interrupt request					✓	
BVD1	Bus voltage detect	I/O	46	✓			
$\overline{\mathrm{STSCHG}}$	Status changed				✓		
$\overline{\mathrm{PDIAG}}$	Pass diag.					✓	
BVD2	Bus voltage detect	I/O	45	✓			
SPKR	Speaker				✓		
$\overline{\mathrm{DASP}}$	Disk active/Slv. Pr.					✓	
Power							
V_{CC}	Power (3.3 V)		13, 38	Common			

Table 1. CF Device Connections (continued)

Signal	Description	Type	Pins	Interface			Notes
				PC Card Memory	PC Card I/O	True IDE	
Ground	Ground		1, 50	Common			

Notes:

1. A3–A10 must be ground for IDE mode.
2. $\overline{\text{CS}}$/$\overline{\text{CE}}$ are all chip selects, they just have slightly different names.
3. D15 is the MSB for the CF, while MDH0 is the MSB for the MPC8245.

The CF signals are either similar or ‘sideband’ signals implementing unusual controls that can be ignored in many cases. Only a handful must change to control CF memory, I/O, or IDE cards. Designers implementing PC memory or PC I/O modes should be able to adapt to those devices with only slight modifications to the logic in this application note.

3 Interface Considerations

The CF specification uses standard CMOS signaling levels; at V_{CC} = 3.3 V, a standard CF device and the MPC8245 can be directly connected with no level translation or high-strength drivers needed. However, on many systems, the MPC8245 PortX interface signals used to control the CF device are also needed to provide address and parity to the SDRAM and boot flash memory, and possibly also control auxiliary flash storage and other devices. Because CF devices are allowed to present a 100-pF load on each signal to ensure the ability to operate at high speeds, a small amount of isolation buffering probably results in a more robust and faster system (refer to Figure 1).

Therefore, this application note shows a small buffer on critical signals, but this buffer is only for the purpose noted previously. The buffer can be eliminated if a system timing analysis allows—this is strictly up to the system designer.

Because the memory parity signals are used only to implement the higher address lines for the PortX interface, buffering the parity lines is not needed if only the CF IDE interface is used (which uses only A[2:0]). Other signals, such as $\overline{\text{FWE}}$/$\overline{\text{FOE}}$, can be lightly loaded and not need additional buffering.

Buffers for unidirectional signals such as address and $\overline{\text{FWE}}$ are easily implemented because the buffer can be permanently enabled. For bidirectional data signals, the standard buffer control signals DIR (A/B) and $\overline{\text{OE}}$ should be connected as shown in Figure 1, with logic to control the buffer dependent on how chip selects are implemented.

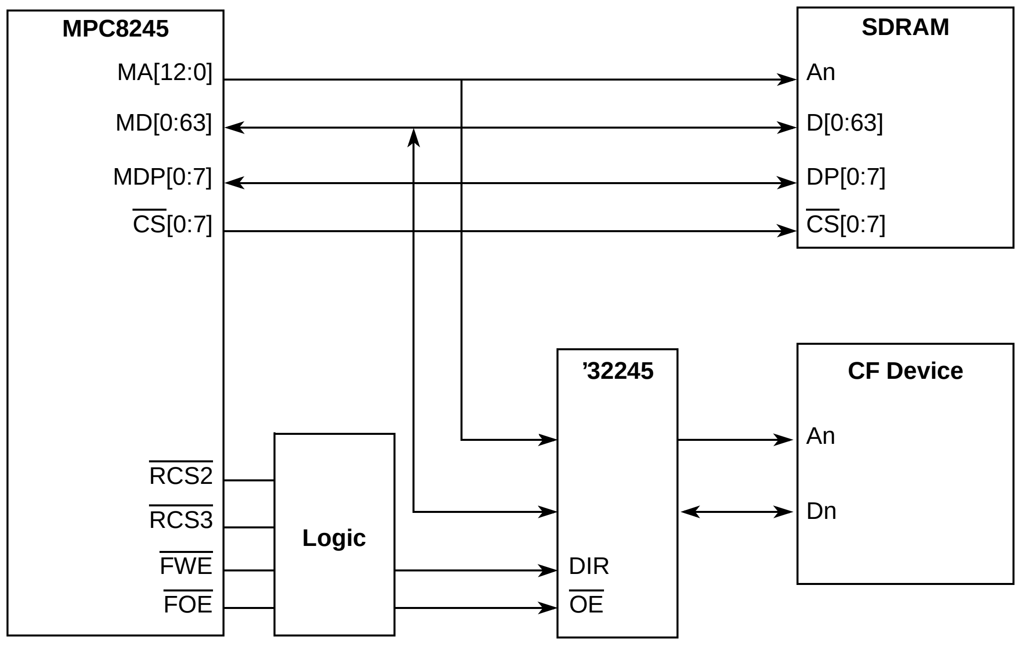

Figure 1. MPC8245 Buffering

4 Chip Selects

The CF devices require two separate chip selects to implement access to data versus control registers or to emulate the two chip selects for IDE status and control. Since the address space requirements of a CF device are less than that provided by the PortX interface, there are two possible ways to provide these chip selects: use $\overline{\text{RCS2}}$ and $\overline{\text{RCS3}}$ as a direct connection or use just one of $\overline{\text{RCS}}$[2:3] with address decode logic to partition the chip select into two. The choice may depend on other system requirements; each has benefits and costs (as shown in Table 2).

Table 2. CF True IDE Mode Read Timing

Type	Mapping	Benefits	Costs
Direct	$\overline{\text{CS0}}$ = $\overline{\text{RCS2}}$ $\overline{\text{CS1}}$ = $\overline{\text{RCS3}}$	Simple $\overline{\text{CS}}$ connections	Needs logic to detect $\overline{\text{RCS2}}$ + $\overline{\text{RCS3}}$ for address/data buffering
Decoded	$\overline{\text{CS0}}$ = $\overline{\text{RCS2}}$ and MA0 $\overline{\text{CS1}}$ = $\overline{\text{RCS2}}$ and!MA0	Logic needed for $\overline{\text{CS0}}$/$\overline{\text{CS1}}$ mapping	Direct use of $\overline{\text{RCS}}$ for buffer enable

For direct $\overline{\text{CS}}$ connection, typical connections for the chip selects and buffer controls are shown in Figure 2.

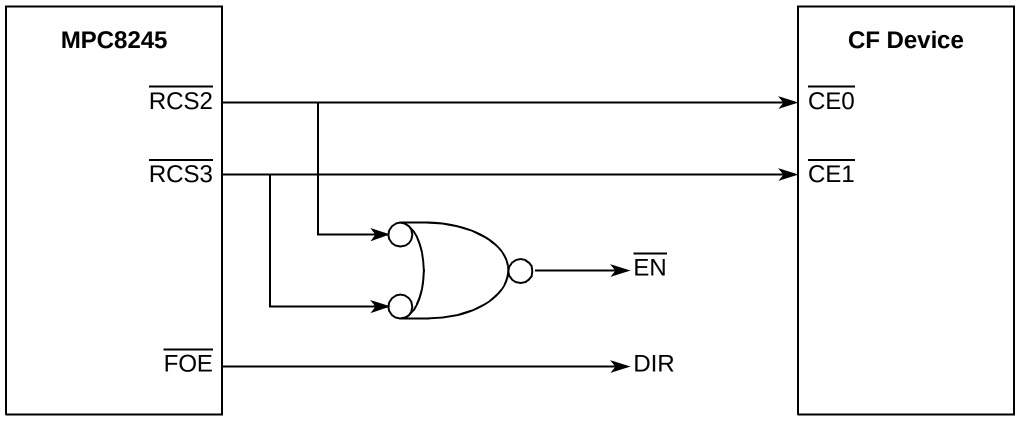

Figure 2. Dual $\overline{\text{RCS}}$ Connections

Dual $\overline{\text{RCS}}$ connections can be implemented with the following VHDL code or equivalent:

```
EN_B <= '0' WHEN (RCS2_B = '0' AND RCS3_B = '0')
          ELSE '1';
```

Single $\overline{\text{RCS}}$ controls are shown in Figure 3.

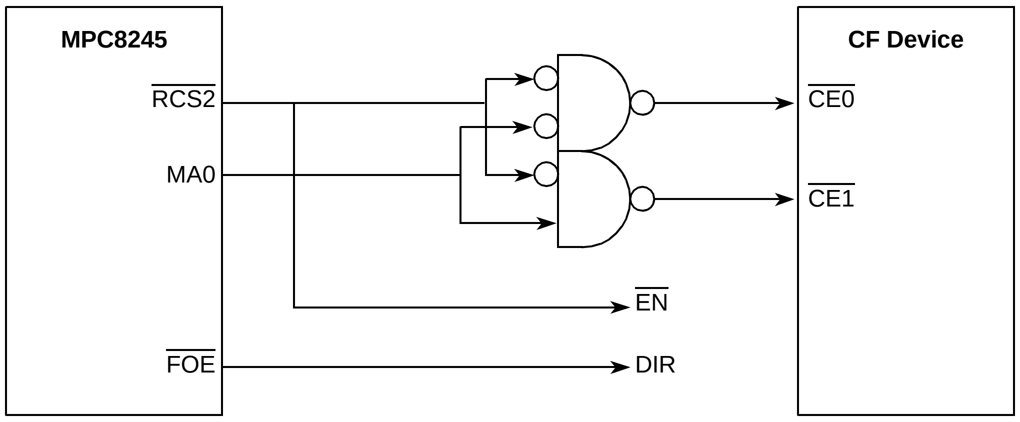

Figure 3. Single $\overline{\text{RCS}}$ Connections

```
CE0_B <= '0' WHEN (RCS2_B = '0' AND MA0 = '0')
             ELSE '1';

CE1_B <= '0' WHEN (RCS2_B = '0' AND MA0 = '1')
             ELSE '1';
```

Since either selection typically requires a small amount of logic, the choice is usually driven by other system design factors.

5 Timing Issues

An important consideration of the CF interface is the AC timing, both signal-to-signal relationships, and the overall duration of signals. Signal-to-signal relationships are maintained by proper programming of the PortX controller, and will be covered in the following sections.

The overall duration of signals becomes an issue when considering the high-speed of the MPC8245 PortX interface (possibly 8 ns clocks with a 133 MHz bus) and the relatively slow speeds allowed by the CF standard. Referring to the overall access times discussed in Section 5.1, “Read Timing” and Section 5.2, “Write Timing,” cycles may take up to 255 ns, or approximately 31 clocks at 133 MHz. Since this is just beyond the limits of the standard mode of the MPC8245 PortX interface, the ‘handshake’ mode is needed. The handshake mode allows PortX cycles to stretch out indefinitely until the $\overline{\text{DRDY}}$ signal is asserted. This mode is not discussed here, as a thorough familiarity with the MPC8245 PortX interface is assumed; refer to Sections 6.3.5 and 6.3.6 of the *MPC8245 Integrated Processor User’s Manual*.

A slight complication of the CF interface is that the IORDY ($\overline{\text{WAIT}}$) signal, which would be a natural means of controlling the MPC8245 PortX handshake mode, is optional and not mandatory. Bit 11 of the IDE capabilities word indicates whether it is supported or not. Unless only IORDY-compatible devices are used, IORDY is not usable as a sole means of timing control for a general-purpose CF IDE interface. Since it is not clear what percentage of CF devices support IORDY, the control logic incorporates both.

5.1 Read Timing

The read access timing waveform is shown in Figure 4, while the parameters and values are shown in Table 3.

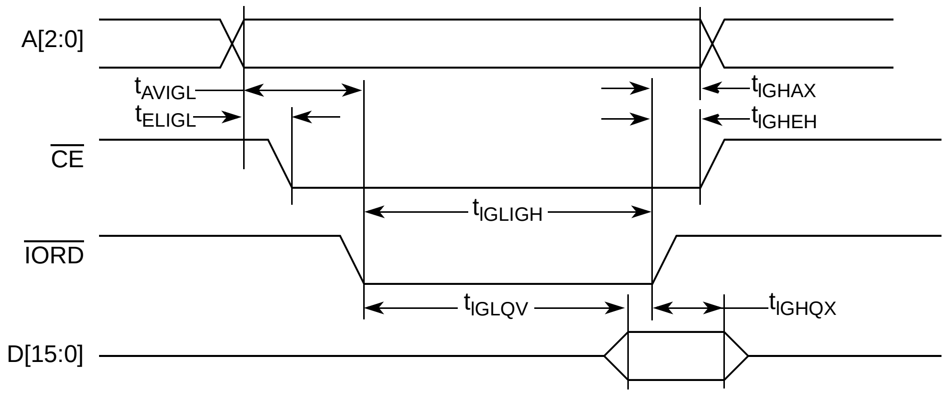

Figure 4. CF True IDE Mode Read Cycle

Table 3. CF True IDE Mode Read Timing

Timing Parameter	Description	Min	Max	Unit
t_{IGLQV}	Data delay after $\overline{\text{IORD}}$	—	100	ns
t_{IGHQX}	Data hold following $\overline{\text{IORD}}$	0	—	ns
t_{IGLIGH}	$\overline{\text{IORD}}$ width time	165	—	ns
t_{AVIGL}	Address setup before $\overline{\text{IORD}}$	70	—	ns
t_{IGHAX}	Address hold following $\overline{\text{IORD}}$	20	—	ns
t_{ELIGL}	$\overline{\text{CE}}$ setup before $\overline{\text{IORD}}$	5	—	ns
t_{IGHEH}	$\overline{\text{CE}}$ hold following $\overline{\text{IORD}}$	20	—	ns

First, notice is that the overall access time can be as high as 255 ns (t_{AVIGL} + t_{IGLIGH} + t_{IGHAX}). This is just beyond the capability of the MPC8245 Flash/PortX interface operating at 133 MHz, and even then the setup/hold time relationships are not exactly as required. When this is the case, $\overline{\text{DRDY}}$ handshake mode is typically used, as previously detailed.

Further examining the waveform and timing values, it is clear there is no required relationship between address and chip select; the MPC8245 PortX interface provides one clock of address setup prior to the chip select, so the address and $\overline{\text{CE}}$ signals can be mapped to the standard PortX MA[20:0] and $\overline{\text{RCS}}$[2:3] signals. However, since a minimum of 70 ns is required before $\overline{\text{IORD}}$ can be asserted, additional logic is required to implement $\overline{\text{IORD}}$ other than a simple logical function of $\overline{\text{RCS}}$[2:3] and $\overline{\text{FWE}}$.

To implement the delayed $\overline{\text{IORD}}$, the ASFALL facility of the PortX interface is used. The signal $\overline{\text{AS}}$ can be programmed to be asserted low, 1 to 15 clocks after the start of any cycle (read or write). By setting ASFALL to (70 ns/bus clock + 1), the $\overline{\text{AS}}$ signal will be asserted low >= 70 ns after the start of any cycle. Additional logic combines $\overline{\text{AS}}$ with $\overline{\text{RCS}}$[2:3] and $\overline{\text{FWE}}$ to produce the $\overline{\text{IORD}}$ signal.

Since $\overline{\text{AS}}$ is typically much shorter in duration than the handshake-controlled $\overline{\text{RCS}n}$ signal, it needs to be latched to extend the duration. Also, $\overline{\text{RCS}n}$ can be delayed, but that takes as much logic as a simple set-reset flip-flop.

Similarly, the address signal needs to be maintained after $\overline{\text{IORD}}$, though data does not. Since the MPC8245 may change the address after releasing $\overline{\text{RCS}}$[2:3], the ASRISE setting would normally be used to remove the signal one or two clocks early. In handshake mode, the $\overline{\text{DRDY}}$ signal can be used to disable ASRISE early, but if $\overline{\text{DRDY}}$ is not asserted early, the normal ASRISE timing values are in effect. The effect is that $\overline{\text{AS}}$ assertion is usually much shorter than the overall access time, and $\overline{\text{AS}}$ cannot be used by itself (unless the targeted card is guaranteed to work at that speed — this is an optimizable option). The hold time is maintained by the inherent disable time provided by the MPC8245—four to five clocks.

With these restrictions considered, the interface logic necessary to perform reads is shown in Figure 5.

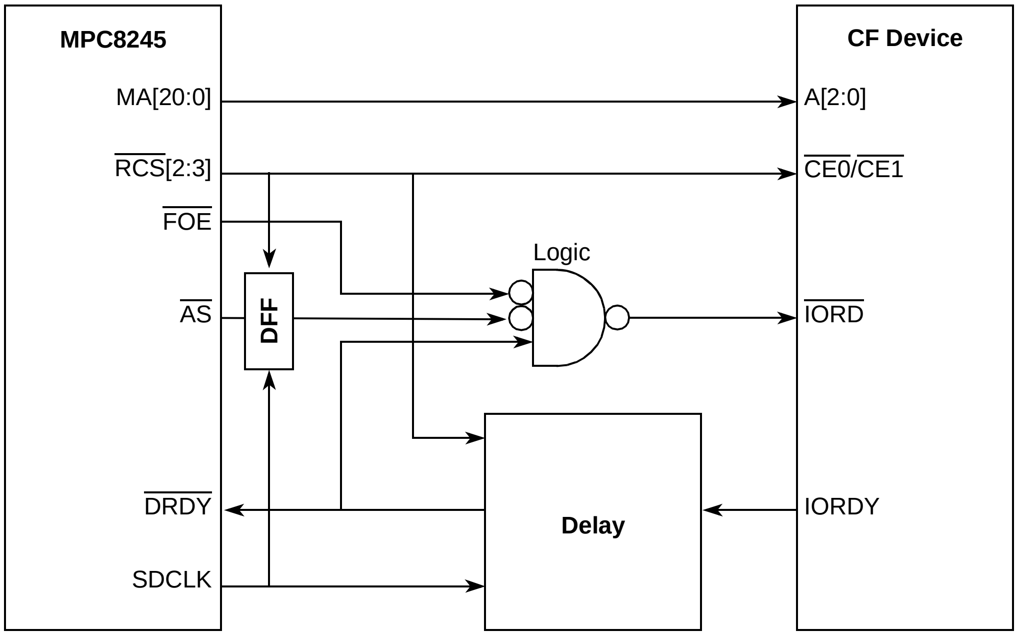

Figure 5. MPC8245 PortX to CF Read Interface

The VHDL equations (portions) to implement these functions are:

```
-- CF Read Logic
SIGNAL TIMER : std_logic_vector(4 downto 0);
SIGNAL ASL_B : std_logic;
BEGIN
            DELAY_RST_B = RCS2_B AND RCS3_B;
            IORD_B <= ‘0’ WHEN (    RCS2_B = ‘0’  AND  ASL_B  = ’0’
                                AND FOE_B  = ‘0’  AND  DRDY_B = ‘1’)
                                                          ELSE ‘1’;
            ASLL : PROCESS( SDCLK, AS_B, DELAY_RST_B )
            BEGIN
                IF (DELAY_RST_B = ‘0’) THEN-- reset ASL_B on RCSx clear
                                                      ASL_B <= ‘1’;
                       ELSIF (AS_B = ‘0’) THEN-- latch ASL_B on AS_B
                                                      ASL_B <= ‘0’;
                                                            END IF;
            END PROCESS;
            DELAY : PROCESS( SDCLK, IORDY, DELAY_RST_B )
            BEGIN
                            IF (DELAY_RST_B = '0') THEN-- idle timer
                                           TIMER <= (others => '0');
```

```
                                    ELSIF (IORDY = '0') THEN-- halt timer if not ready
                                                                     TIMER <= TIMER;
                              ELSIF (SDCLK'EVENT AND SDCLK = '1') THEN-- else increment
                                                           TIMER <= TIMER + ''00001'';
                                                                             END IF;
               END PROCESS DELAY;
               DRDY <= '0' WHEN (TIMER >= ''10000'')
                                                                           ELSE '1';
```

Note that $\overline{\text{DRDY}}$ must be asserted low continuously until the $\overline{\text{RCS}}$[2:3] signals are de-asserted. This is correctly implemented in the preceding equations.

For IORDY, when available, if the signal is deasserted (implying it is available), it prevents the timer from incrementing, and thereby triggering $\overline{\text{DRDY}}$ to end the cycle. When IORDY is released, the timer resumes at this point. This allows both IORDY-extensions to the cycle and timer-controlled extensions, with the latter controlling in cases where IORDY is not supported. The timing diagram in Figure 6 shows a CF IDE read cycle as implemented by the MPC8245 PortX interface with accompanying logic.

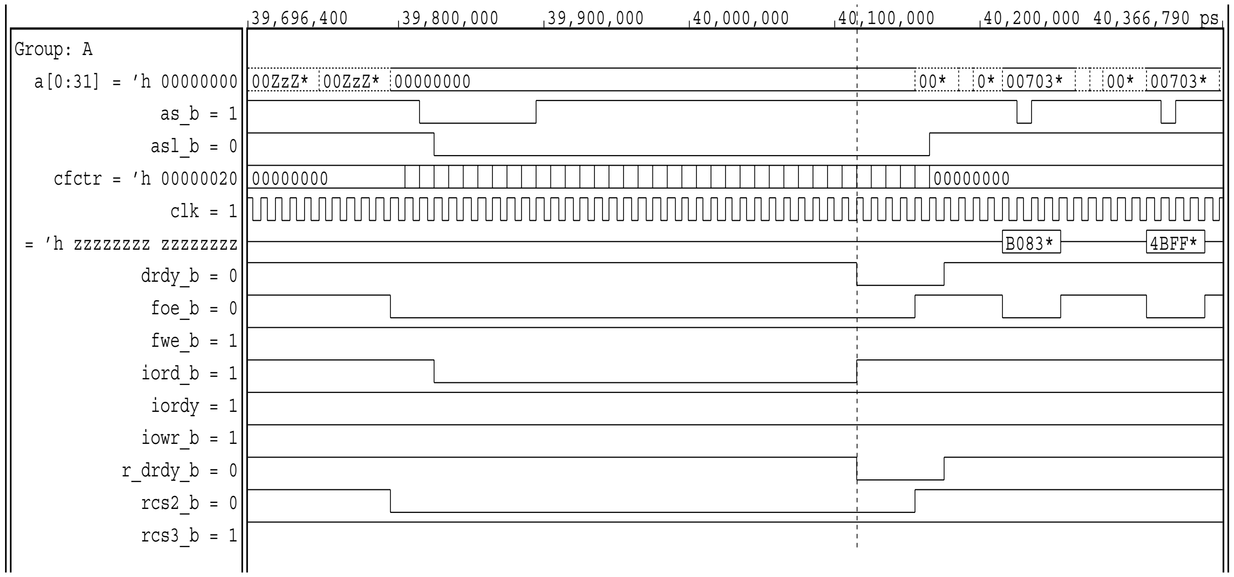

Figure 6. MPC8245 CF Read Cycle

As expected, the $\overline{\text{DRDY}}$ facility extended the access time well beyond the capabilities of the ROMFAL/ROMNAL settings and allowed the access to take as many cycles as necessary (64 in this example).

5.2 Write Timing

The write cycle is similar to the read cycle, except a different control signal is used. The write access timing waveform is shown in Figure 7, while the parameters and values are shown in Table 4.

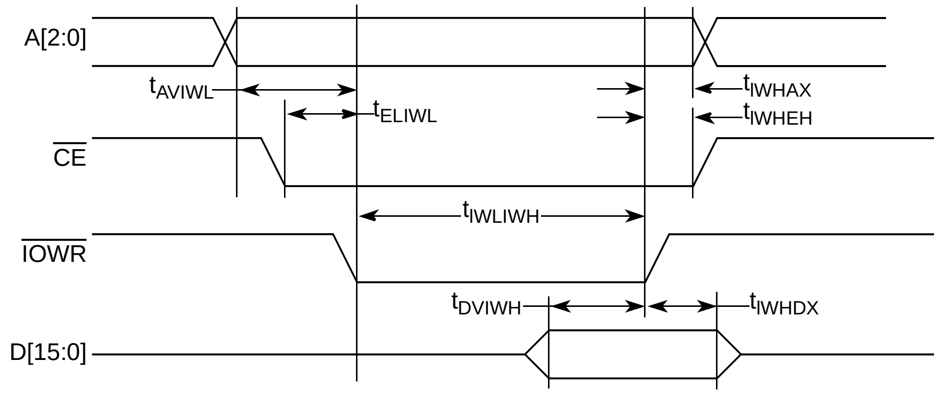

Figure 7. CF True IDE Mode Write Cycle

Table 4. CF True IDE Mode Write Timing

Timing Parameter	Description	Min	Max	Unit
t_{DVIWH}	Data setup before $\overline{IOWR}$	60	—	ns
t_{IWHDX}	Data hold following $\overline{IOWR}$	30	—	ns
t_{IWLIWH}	$\overline{IOWR}$ width time	165	—	ns
t_{AVIWL}	Address setup before $\overline{IOWR}$	70	—	ns
t_{IWHAX}	Address hold following $\overline{IOWR}$	20	—	ns
t_{ELIWL}	$\overline{CE}$ setup before $\overline{IOWR}$	5	—	ns
t_{IWHEH}	$\overline{CE}$ hold following $\overline{IOWR}$	20	—	ns

The restrictions on timing are very similar to that of the read path, except that data must be present before $\overline{IOWR}$ is released. For write cycles, the interface logic is shown in Figure 8.

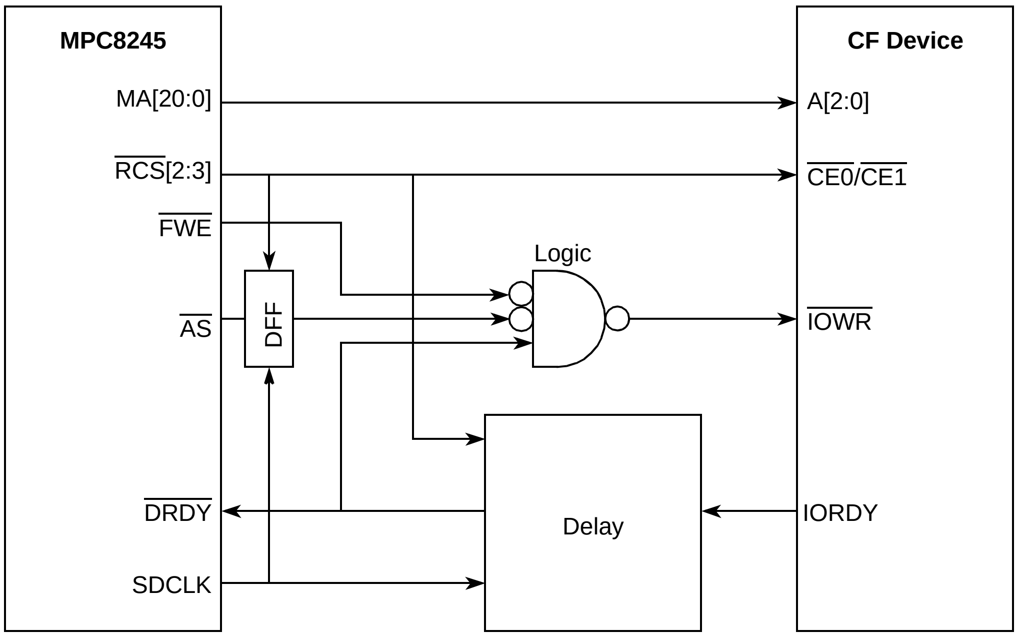

Figure 8. MPC8245 PortX to CF Write Interface

This is so similar to the read path, that the read and write logic can be combined and the following equations is all that is needed to create a read/write controller for the CF device:

```
                IOWR_B <= ‘0’ WHEN (    RCS2_B = ‘0’  AND  ASL_B  = ’0’
                                                AND FWE_B  = ‘0’  AND  DRDY_B = ‘1’)
                                                                           ELSE ‘1’;
```

The timing diagram in Figure 9 shows a CF IDE write cycle as implemented by the MPC8245 PortX interface with accompanying logic.

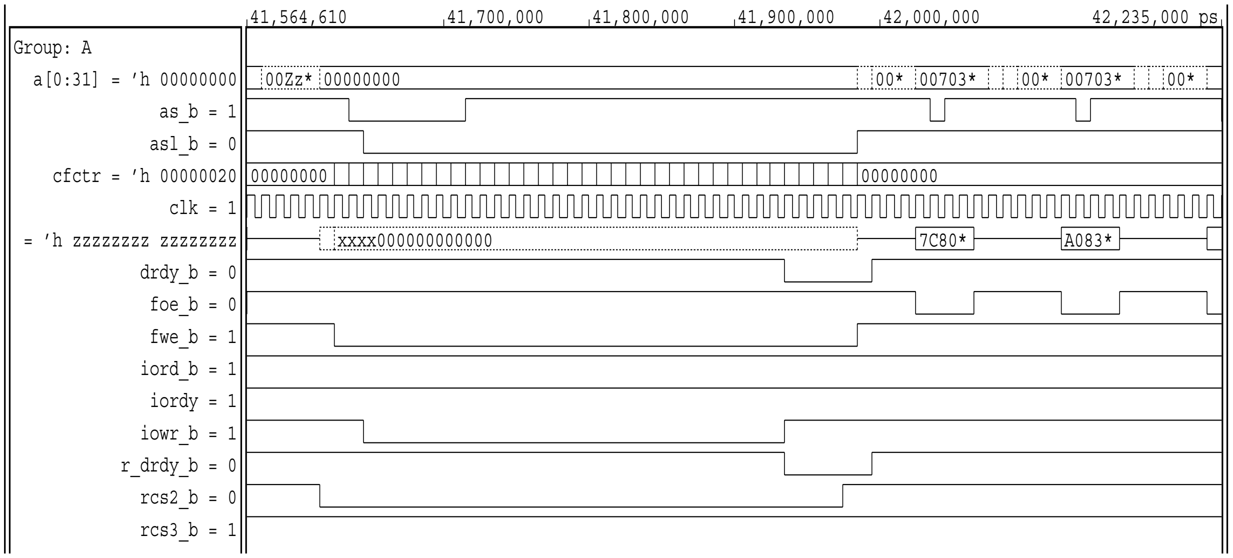

Figure 9. MPC8245 CF Write Cycle

6 Complete IDE Interface Connections

In true IDE mode, the CF device is accessed as if it were an IDE disk drive operating in PIO (non-DMA) modes, with only the corresponding standard complement of eight address lines. Normally, these extra address lines are grounded, but to allow a common interface for true IDE and PC card memory modes, it is sufficient simply to drive the unused address lines to 0x0000 during accesses.

Several signals that are unique to operating multiple IDE disk drives ($\overline{\text{DASP}}$/$\overline{\text{PDIAG}}$) can be ignored in this application. The $\overline{\text{IOIS16}}$ signal, for example, is used to indicate 8-bit versus 16-bit accesses and is not needed. Because software can set the extended ROM banks to a 16-bit mode, only 16-bit accesses occur. With these restrictions, the IDE-mode connections can be summarized as shown in Table 5.

Table 5. CF IDE Device Connections

Signal	Description	Type	Pins	Connections
A[10:3]	Address	I	8, 10, 11, 12, 14, 15, 16, 17	To ground
A[2:0]	Address	I	18, 19, 20	To MA[2:0] in same order, through optional but recommended buffer
$\overline{\text{CD}}$[1:2]	Card detect	O	26, 25	No connect

Table 5. CF IDE Device Connections (continued)

Signal	Description	Type	Pins	Connections
$\overline{\mathrm{CS}}$[0:1]	Chip selects	I	7, 32	To $\overline{\mathrm{RCS2}}$/$\overline{\mathrm{RCS3}}$ or to decoder logic
$\overline{\mathrm{CSEL}}$	Cable select	I	39	To ground
D[15:0]	Data bus	I/O	31, 30, 29, 28, 27, 49, 48, 47, 6, 5, 4, 3, 2, 23, 22, 21	To MDH[0:15] through optional but recommended buffer
$\overline{\mathrm{INPACK}}$	Input acknowledge	O	43	No connect
$\overline{\mathrm{IORD}}$	I/O read strobe	I	34	To FPGA
$\overline{\mathrm{IOWR}}$	I/O write strobe	I	35	To FPGA
VS[1:2]	Voltage sense	O	33, 40	No connect
IORDY	Ready	O	42	To FPGA
$\overline{\mathrm{WE}}$	Write enable	I	36	To V_{CC} (3.3 V)
$\overline{\mathrm{ATASEL}}$	IDE mode enable	I	9	To ground
$\overline{\mathrm{REG}}$	Register select	I	44	To V_{CC} (3.3 V)
$\overline{\mathrm{RESET}}$	Reset (active-low)	I	41	To $\overline{\mathrm{HRESET}}$ or $\overline{\mathrm{PCIRST}}$
$\overline{\mathrm{IOIS16}}$/$\overline{\mathrm{IOCS16}}$	16-bit access	O	24	No connect
INTRQ	Interrupt request	O	37	Through inverter to $\overline{\mathrm{IRQ}}$[0:4]
$\overline{\mathrm{PDIAG}}$	Pass diag.	I/O	46	Pull up, to optional activity LED
$\overline{\mathrm{DASP}}$	Disk active/Slv. Pr.	I/O	45	No connect
V_{CC}	Power (3.3 V)	—	13, 38	To 3.3 V
Ground	Ground	—	1, 50	To ground

These connections produce a diagram as shown in Figure 10.

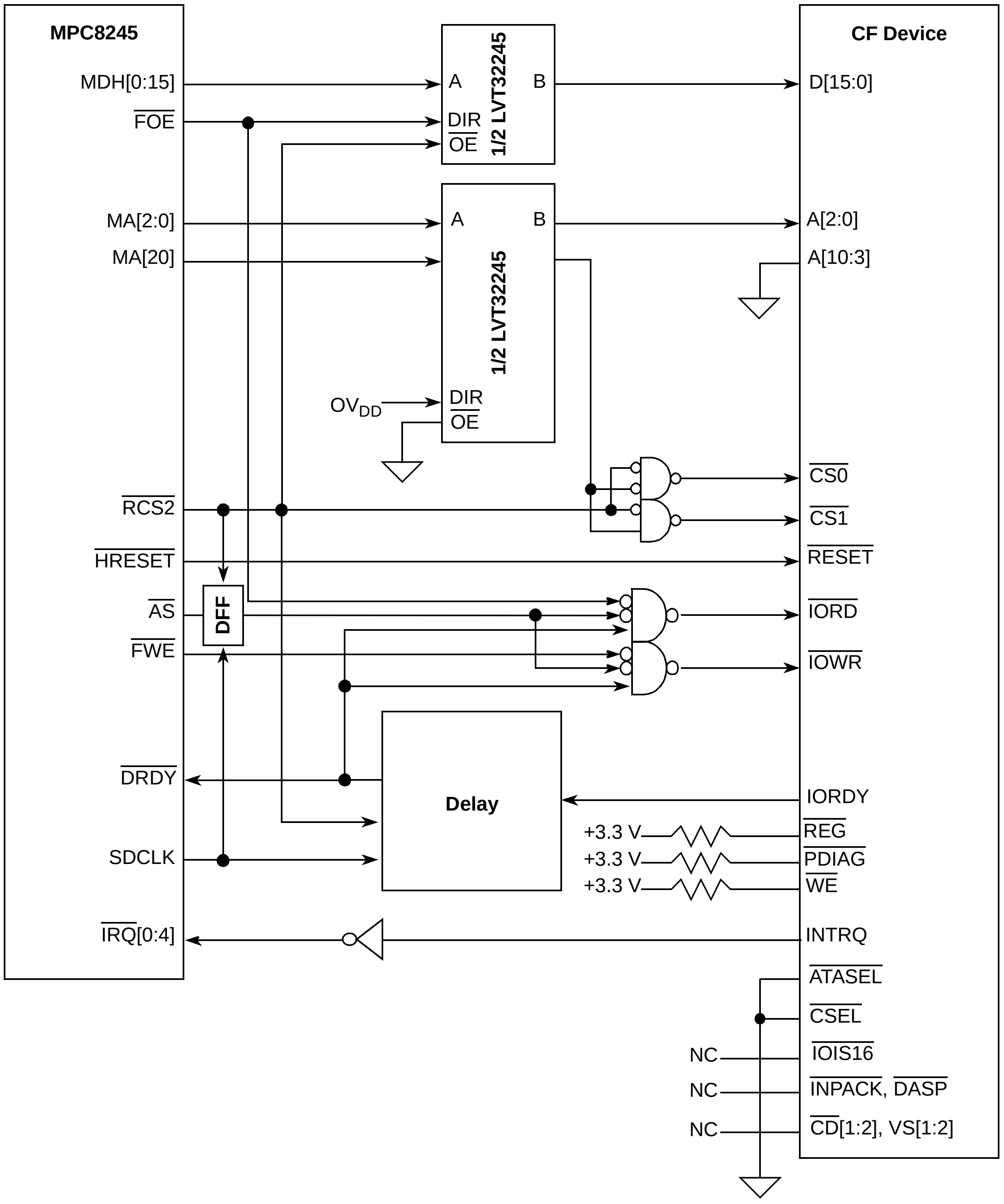

Figure 10. CF IDE Interface

7 Software Setup

The MPC8245 PortX interface must be properly programmed before the CF device can be used. The following register fields need to be set to the values shown:

`ERCR1.RCS2_EN`	`1-- default`
`ERCR1.RCS2_BURST`	`0-- non-burst for CF`
`ERCR1.RCS2_DBW`	`01-- 01=16 bits, though 8 is also possible`
`ERCR1.RCS2_CTL`	`11-- 11=handshake mode, required`
`ERCR1.ROMFAL`	`11111-- not used, set to max`
`ERCR1.ROMNAL`	`11111-- not used, set to max`
`ERCR1.ASFALL`	`00100-- delay 4 clocks`
`ERCR1.ASRISE`	`00110-- rise 6 clocks`
`ERCR1.RCS2_TS_WAIT_TIMER`	`00010-- two cycles between accesses`

Because $\overline{\text{AS}}$ is latched for reasons previously discussed, the ASRISE parameter is not critical. The RCS3 control register, ERCR2, uses the same settings if it is used with the CF interface. Other register fields (such as Flash location, are application-dependent). After this point, card-specific initialization can be performed. For IDE disk interface devices, the usual sequence is:

1. Issue RESET command to control register.
2. Poll status until not busy.
3. Issue RECALIBRATE command to control register.
4. Poll status until not busy.
5. Issue IDENTIFY command to control register.
6. Poll status until not busy.
7. Read 256 words of data from command pointer.

Now application programs can read and write data and commands to the device. Usually the desired 512-byte block of data is indexed using the LBA (Logical Block Address), a 28-bit address.

Note that the MPC8245 cannot boot from the CF device. The EXTROM port is used, and this requires software setup. A small boot-loader can initialize the PortX interface and then jump.

8 References

This section lists additional reference information:

- MPC8245 documentation, refer to www.freescale.com
  — *MPC8245 Integrated Processor User's Manual*
  — *MPC8245 Integrated Processor Hardware Specifications*
- CompactFlash Specifications, refer to www.compactflash.org/
- For reference purposes, many designers may refer to application notes and example/reference designs available on the Freescale website.

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN2293
Rev. 2
09/2006

Preliminary—Subject to Change Without Notice