Microcomputer Busses and Direct Memory Access (DMA)

References:

- Freescale Semiconductor, Inc., “MCF5235 Reference Manual”.
- Freescale Semiconductor, Inc., “MCF5441X Reference Manual”.

Figures and tables from the above documents have been included in these course notes for educational purposes in ECE 315 only. The original documentation should be consulted to ensure accuracy.

Freescale™ and ColdFire® are registered trademarks of Freescale Semiconductor, Inc.

Definitions

Definition from "Microcomputer Busses", by R.M. Cram, (Academic Press, 1991):

"A *bus* is a tool designed to interconnect the functional blocks of a microcomputer in a systematic manner. It provides for standardization in mechanical form, electrical specifications, and communication protocols between board-level devices."

***Processor-specific bus*:** a bus that is intended for use with only one processor or with members of one family of compatible processors.

Ex: Freescale FlexBus, ARM AMBA bus, Altera Avalon bus, IBM CoreConnect, Barco External Bus Interface (EBI)

***Standardized processor-independent bus*:** a bus that is intended to promote interchangeability among a class of board-level products based on possibly different processors.

Ex: PCI, PCIe

Historical busses: S-100, Unibus, Std bus, VME bus, Multibus, etc.

A Functional Classification of Busses

1) Local Processor-Memory & Graphics Busses

--- short, synchronous, high-speed
--- overriding priority: maximize the data throughput

e.g. Rambus, DDR, DDR2, DDR3, VESA, AGP, PCIe

2) Input/Output (I/O), Peripheral and Instrument Busses

--- maximum flexibility is the priority
--- must accommodate a variety of data rates and latencies
--- open standards are used to maximize the potential market

e.g. SCSI, GPIB (IEEE-Std-488), USB, Firewire

3) Expansion Busses (formerly called Backplane Busses)

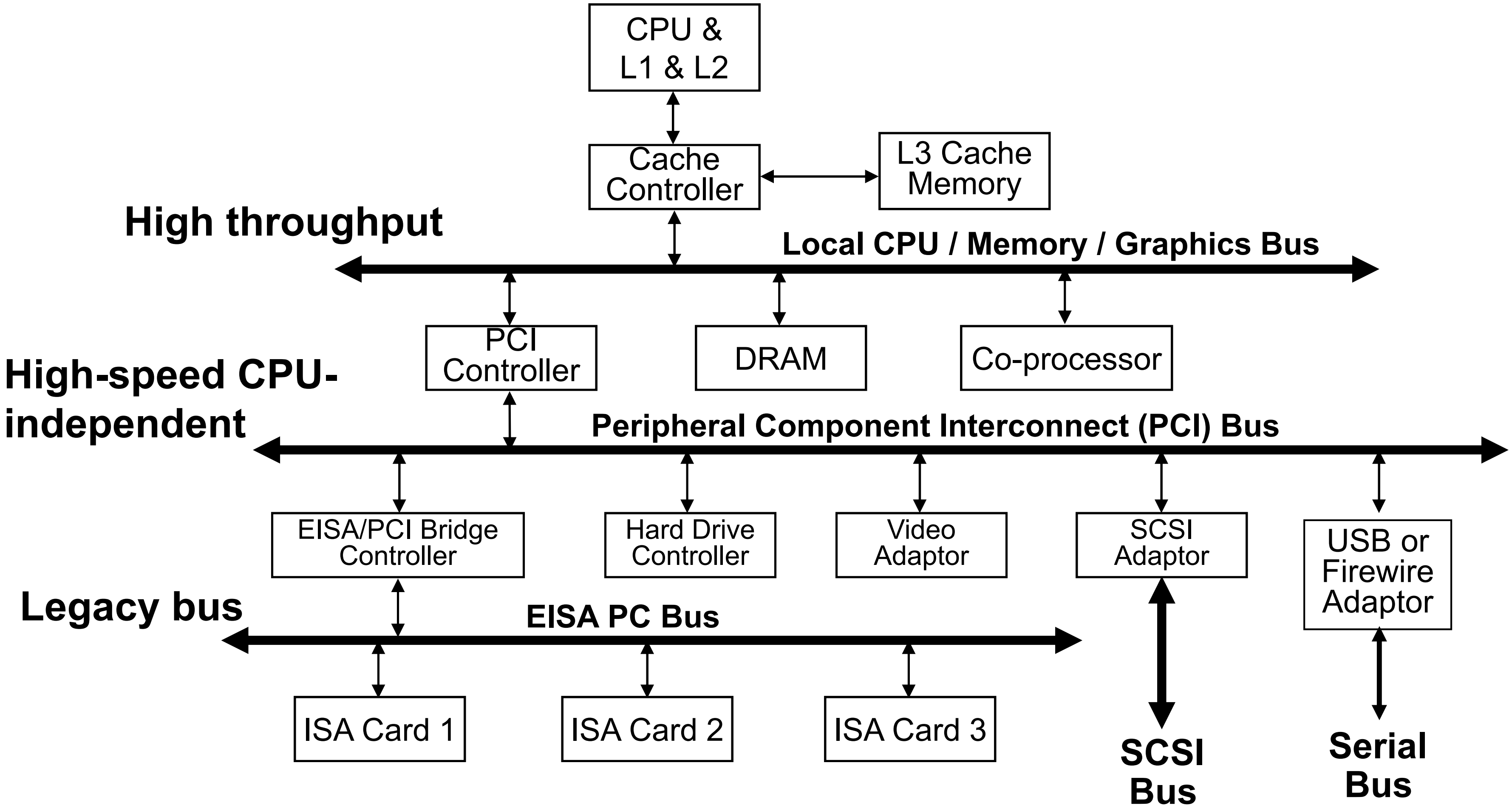
--- often midway in performance between processor-memory busses and I/O, Peripheral & Instrument busses
--- expansion busses are used to reduce system design cost and to reduce the time-to-market for new computer systems

e.g. S100, Std bus, VME, Multibus, PCI, CompactPCI

Hierarchy of Busses (early 1990s)

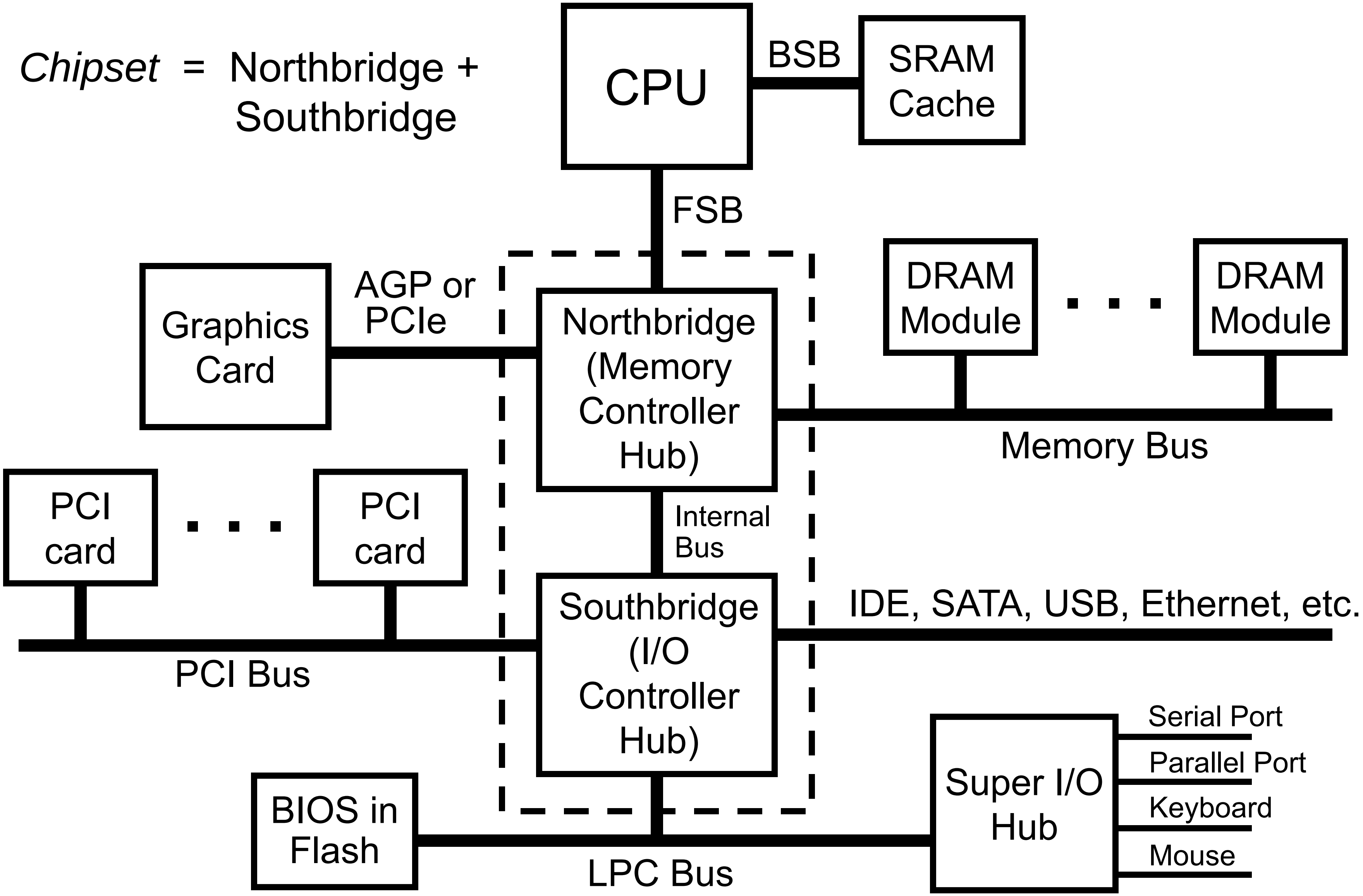
- To exploit the strengths (and avoid the weaknesses) of different kinds of busses, high-performance systems often use a hierarchy of busses.

Example: Personal Computer Bus Architecture (old-fashioned)



Chipset-based PC Bus Architecture (late 1990s)

Chipset = Northbridge + Southbridge



Some PC Bus Acronyms

BSB = Back Side Bus (connects CPU to an external cache)

FSB = Front-Side Bus

SRAM = Static Random-Access Memory (fast & expensive)

DRAM = Dynamic Random-Access Memory (slower & cheap)

AGP = Accelerated Graphics Port

PCI = Peripheral Component Interconnect (bus-based)

PCIe = PCI Express (faster point-to-point version of PCI)

IDE = Integrated Drive Electronics (for connecting hard disks)

SCSI = Small Computer System Interface

SATA = Serial Advanced Technology Attachment

USB = Universal Serial Bus

BIOS = Basic Input/Output System (self-test, autoconfiguration)

LPC = Low Pin Count (for low-speed interfaces to humans)

Intel X299 High-end Chipset (June 2017)

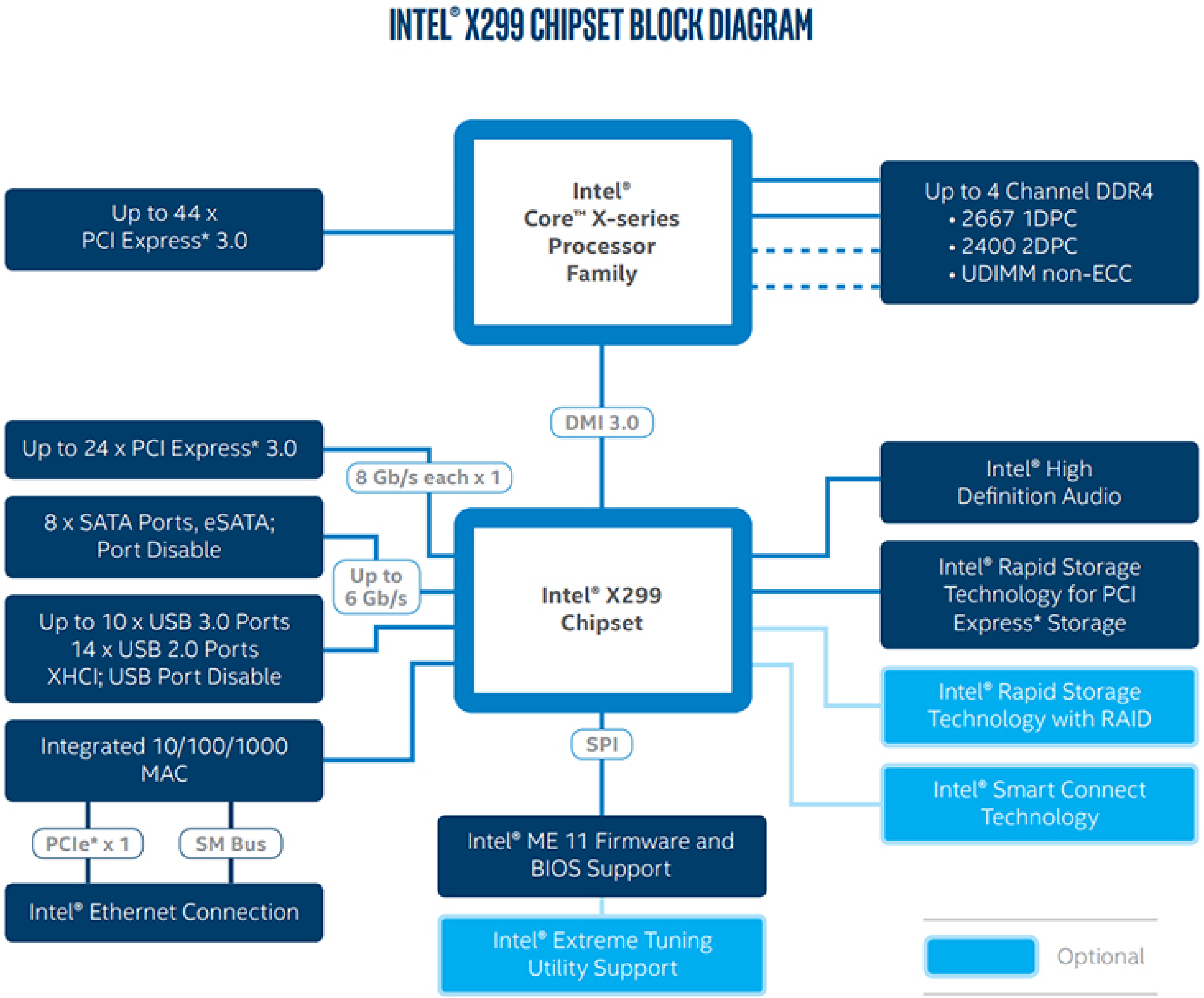

INTEL® X299 CHIPSET BLOCK DIAGRAM
Up to 44 x PCI Express* 3.0
Intel® Core™ X-series Processor Family
Up to 4 Channel DDR4
• 2667 1DPC
• 2400 2DPC
• UDIMM non-ECC
DMI 3.0
Up to 24 x PCI Express* 3.0
8 Gb/s each x 1
8 x SATA Ports, eSATA; Port Disable
Up to 6 Gb/s
Up to 10 x USB 3.0 Ports
14 x USB 2.0 Ports
XHCI; USB Port Disable
Integrated 10/100/1000 MAC
PCIe* x 1
SM Bus
Intel® Ethernet Connection
Intel® X299 Chipset
SPI
Intel® ME 11 Firmware and BIOS Support
Intel® Extreme Tuning Utility Support
Intel® High Definition Audio
Intel® Rapid Storage Technology for PCI Express* Storage
Intel® Rapid Storage Technology with RAID
Intel® Smart Connect Technology
Optional

Diagram from Intel Corp.

Note: Direct Media Interface 3.0 (DMI 3.0) was originally a Northbridge-Southbridge interface bus at Intel.

Some Bus Terminology

Bus protocol **:** A set of allowed bus signal transition sequences and required timing constraints.

Bus operation / transaction **:** A data transfer or control transfer operation that takes place using bus signals according to a bus protocol.

Bus master **:** A subsystem connected to the bus that can determine the bus operations. More than one bus master can be present on the same bus, but only one bus master can have control (i.e., be active) at a time.

e.g. CPUs, DMACs, Graphics Accelerator

Bus slave **:** A subsystem connected to the bus that responds to bus operations initiated by the currently active bus master.

e.g. RAM, ROM, Peripheral / Interface Chips

Bus arbitration **:** The process of determining which one of two or more contending bus masters will be awarded control of the bus (and thereby become the active bus master).

Arbiter **:** A circuit (possibly in the CPU or MCU) that performs arbitration.

Basic Bus Functions

1. **Data transfer**
   - --- master-to-slave(s), slave-to-master, slave-to-slave
   - --- bytes, words, longwords, larger blocks (e.g. cache lines)
2. **Interrupt handling**
   - --- communication of active interrupt signals to the bus master
   - --- arbitration between multiple sources of interrupts
   - --- communication of interrupt vector information (e.g. IACK cycle)
3. **Arbitration among multiple bus masters**
   - --- arbitration protocol to choose one bus master from among two or more possible contending bus masters
   - --- procedure for transferring bus control from the current bus master to a new bus master
4. **Utility functions**
   - --- power (including the voltages used as the reference for digital signals)
   - --- clocks
   - --- system reset & test signals (e.g., JTAG port)
   - --- power failure detection & battery back-up

Signal Groups Within a Typical Bus

1. **Data signals**

   -- encode the data that is passed between the bus master and bus slaves/targets

   -- number of data signals determines the “bit width” of the system

   -- parity bits or other error correction and control (ECC) bits may be transmitted with each data word so that errors can possibly be detected and corrected at the destination

2. **Address signals**

   -- used to identify locations in memory and memory-mapped registers in peripheral & interface chips

   -- the number of address signals determines the maximum size of the (virtual) memory space

Note: Some or all of the data and address signals may be time-multiplexed on the same bus lines to reduce the pins and wiring.

Signal Groups Within a Typical Bus (cont'd)

3. Control signals

--- used to co-ordinate bus transactions

--- used to arbitrate among:

-- multiple possible bus masters

-- multiple sources of interrupts

--- power failure handling

--- entry into and exit from test modes; distribute test data

--- access to system state (e.g. CPU registers and memory) to support software debugging

4. Power signals

--- typical: +5 VDC, +3.3 VDC, +1.8 VDC, +1.2 VDC, etc.

--- optional: +12 VDC, -12 VDC, -5 VDC

Sharing a Bus Among Multiple Bus Masters

Coarsest granularity

1. **Exclusive Control (Burst Mode):** Each bus master retains exclusive control of the bus for several bus transactions.

   e.g. CPU, CPU, DMAC1, DMAC1, DMAC1, CPU, CPU, . . .

2. **Cycle Stealing:** Bus transactions from different bus masters are interleaved on an *ad hoc* basis or on a strictly round-robin basis.

   e.g. CPU, DMAC1, DMAC2, CPU, DMAC1, DMAC2, . . .

3. **Split Transactions (Pipelined Bus)**

   Read transactions are split into two transactions:

   1) master sends read command & target address

   2) slave sends a return block containing data

   The bus is available to be used for other transactions (possibly by other bus masters) during the memory read access time.

   e.g. RAMBUS, Synchronous DRAMs (DDR, DDR2, DDR3, etc.)

Finest granularity

	PROS	CONS
Exclusive Control	-- simplicity -- software method -- no special hardware required	-- coarse granularity -- bus time may not be shared fairly or efficiently
Cycle Stealing	-- fairer sharing of the bus	-- requires hardware support -- however this support is available in most CPU’s
Split Transactions	-- high-speed buses do not have to wait for slowly responding devices	-- requires hardware support in the bus and all connected devices

Arbitration Among Contending Bus Masters

1. **Fixed Priority**

--- Each bus master is assigned a unique priority.

--- The contending bus master with the highest priority is awarded control of the bus.

2. **Rotating Priority**

--- At any one time, each bus master has a unique priority.

--- The priority assignments are periodically rotated so that each bus master takes a turn at having each of the available priorities.

3. **Pseudo-random Selection**

--- The winning bus master is selected "randomly" from among the currently contending masters.

--- The selection algorithm is not truly random because it is an entirely predictable digital algorithm that mimics the statistics of a truly random process. Such an algorithm is called "pseudo-random".

Synchronous, Asynchronous, Semi-synchronous

Synchronous Busses:

--- All subsystems coordinate data transfers with respect to the edges of one common (possibly multi-phase) system clock.

--- All peripheral chips must be able to respond to bus transactions (reads and writes) within the same time constraints.

Asynchronous Busses:

--- Responses to bus transactions can arrive at any time.

--- Need special handshake lines in the control bus to ensure that data transfers take place correctly.

--- Bus masters can work with different peripheral devices with a variety of response times.

Semi-synchronous Busses:

--- A compromise between purely synchronous and purely asynchronous busses.

--- Bus transactions can take a variable number of system clock cycles.

--- System can accommodate a variety of response times from the peripherals.

--- Need a handshake/acknowledge signal to end each bus transaction.

Data Transfer Protocols

Unilateral :

--- The bus master initiates the bus operation and then assumes that the corresponding bus slave(s) will respond within predetermined time constraints.

--- The duration of the bus operation is determined entirely according to the operation selected by the bus master and timing provided by a shared system clock signal.

Bilateral:

--- The bus master initiates the bus operation, but the duration of the subsequent bus operation depends on the responses of both the bus slave and the bus master.

--- The bus master and the bus slave use handshake lines to communicate timing information to each other.

Multilateral:

--- Extension of the bilateral transfer protocol to allow for more than one active bus slave.

Bus Classification

Underlying Timing Data Transfer Protocol	Shared System Clock	No Clock Present
Unilateral	Synchronous e.g. 6800	
Bilateral	Semi-synchronous e.g. 68000, Pentium, MCF523x	Asynchronous e.g. Unibus
Multilateral		Asynchronous GPIB (IEEE-488)

Bus Trade-Offs

Source: J. Hennessy and D. Patterson, "Computer Architecture: A Quantitative Approach", (Morgan Kaufmann, 1990)

Option	Higher Performance	Lower Cost
1) Bus Width	Separate data & address busses	Multiplexed data & address
2) Data Width	Wider is faster e.g. 8, 16, 32, 64	Narrower is < $ e.g. 16, 8
3) Transfer Size	Block transfers using DMA	Single word using CPU
4) Bus Masters	Multiple masters (requires arbitration)	One master, the CPU, no arbit.
5) Split Transactions?	Yes, to get more pipelining	No, too complex
6) Clocking	Synchronous with matched elements	Asynchronous or semisynchronous

Timing Terminology

Caution: Terminology may vary slightly between vendors.

Double check by checking the data sheets

Set-up time , t_{su}: the minimum length of time that a signal must be valid at a circuit input *before* a second triggering signal arrives at a second input.

Usually a clock

Delay time , t_{co}: the length of time that a circuit requires for its output(s) to begin to change in response to a triggering signal arriving at a second input.

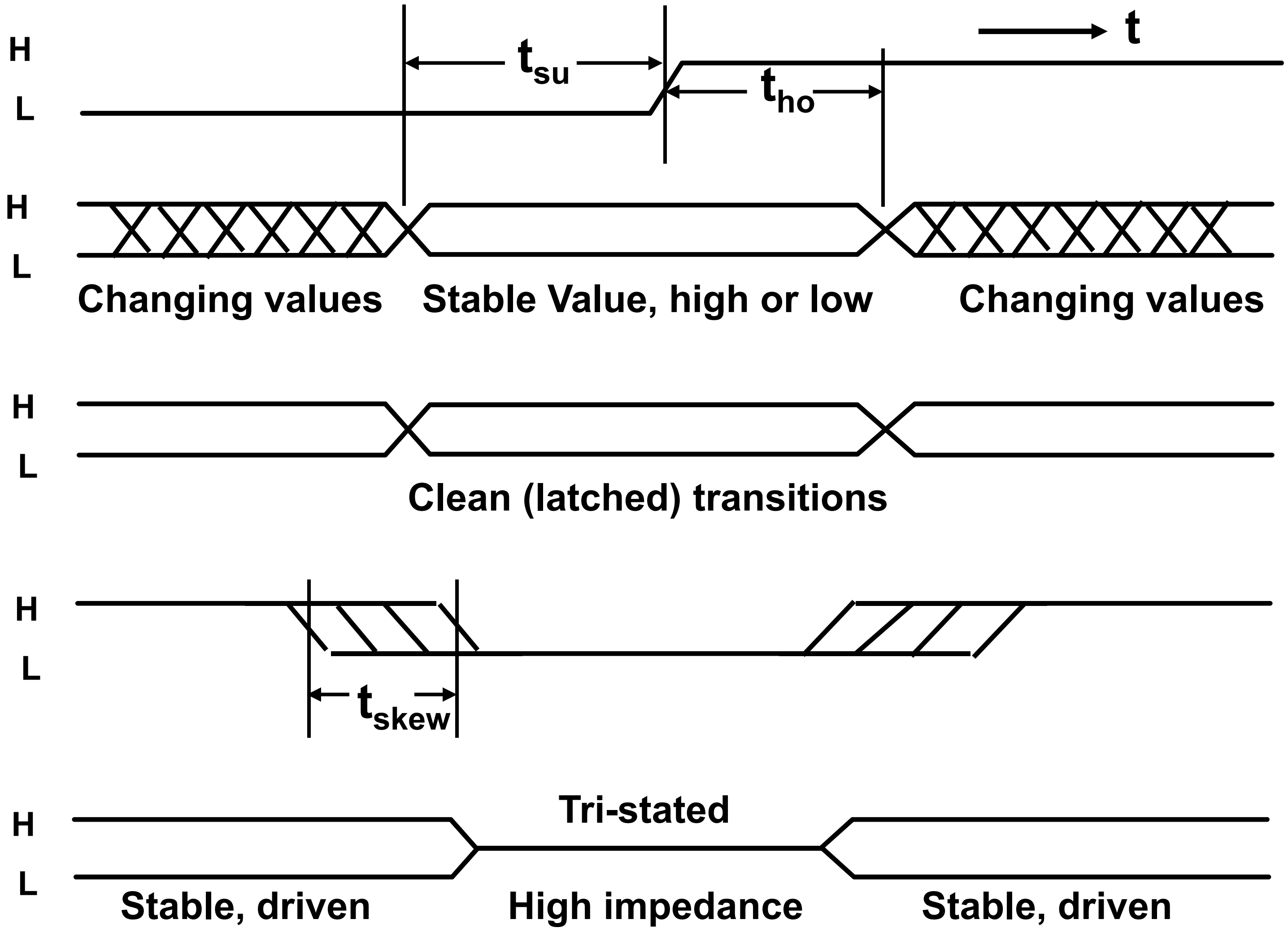
Hold time , t_{ho}: the minimum length of time that a signal must be kept valid at a circuit input *after* a triggering signal has been received at a second input.

Timing skew , t_{skew}: the maximum range of times over which a particular signal transition (L -> H or H -> L) can occur.

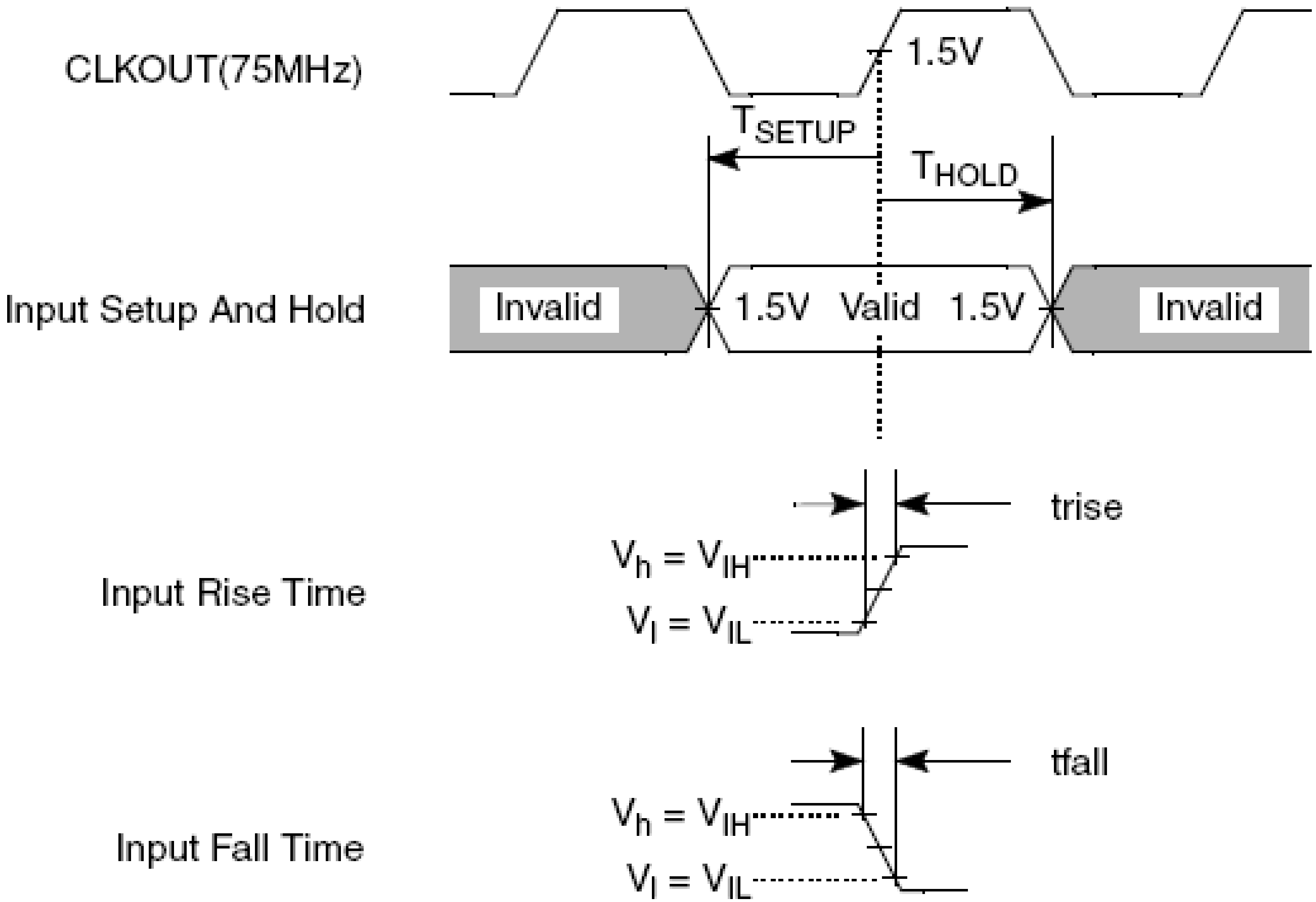
-- Due to variations in driver output resistance

-- Combinational logic outputs take a while to stabilize

Timing Diagram Notation



Ex: Setup and Hold Time Def'ns for the MCF523x



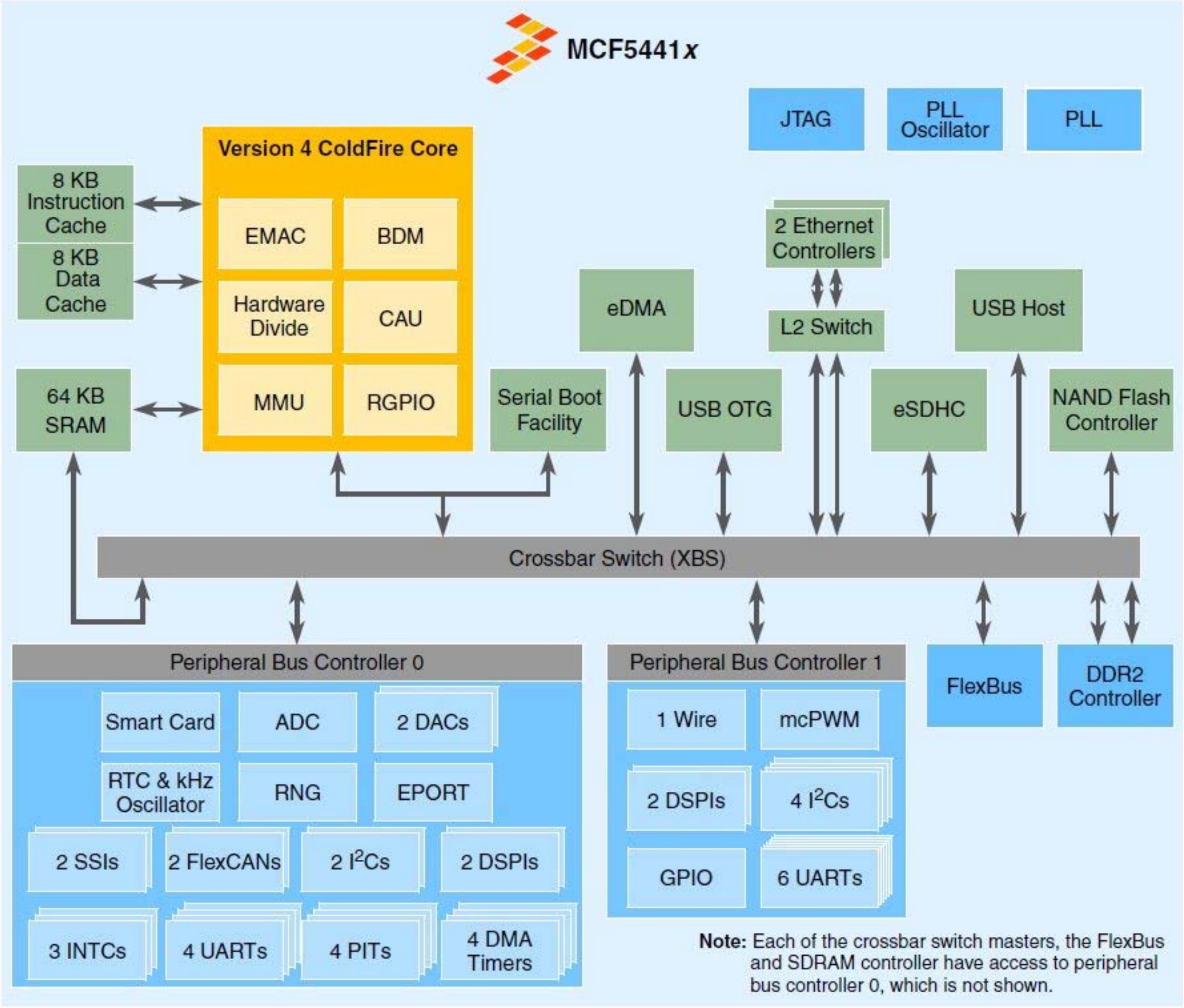
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

The M68000 Bus

- **Processor-specific** (Note: the 68000 bus later became the basis for the VME backplane bus standard).
- **Semisynchronous:** Bus transactions take a variable number of CPU clock cycles. This provides timing flexibility.
- **Bilateral:** The bus master (CPU) initiates the transaction, but the slave uses data acknowledge (DTACK) to fix the number of cycles that are required for the transaction.
- **16 or 32-bit Data Bus:** Width depends on the processor.
- **24-bit Address Bus:** A0 may be decoded on some CPUs. e.g. 68000 uses Upper and Lower Data Strobes, no A0.

  Only $\overline{\text{UDS}}$ is active when A0 = 0 and one byte is being transferred
  Only $\overline{\text{LDS}}$ is active when A0 = 1 and one byte is being transferred.
  Both $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ are active for word and long word transfers.

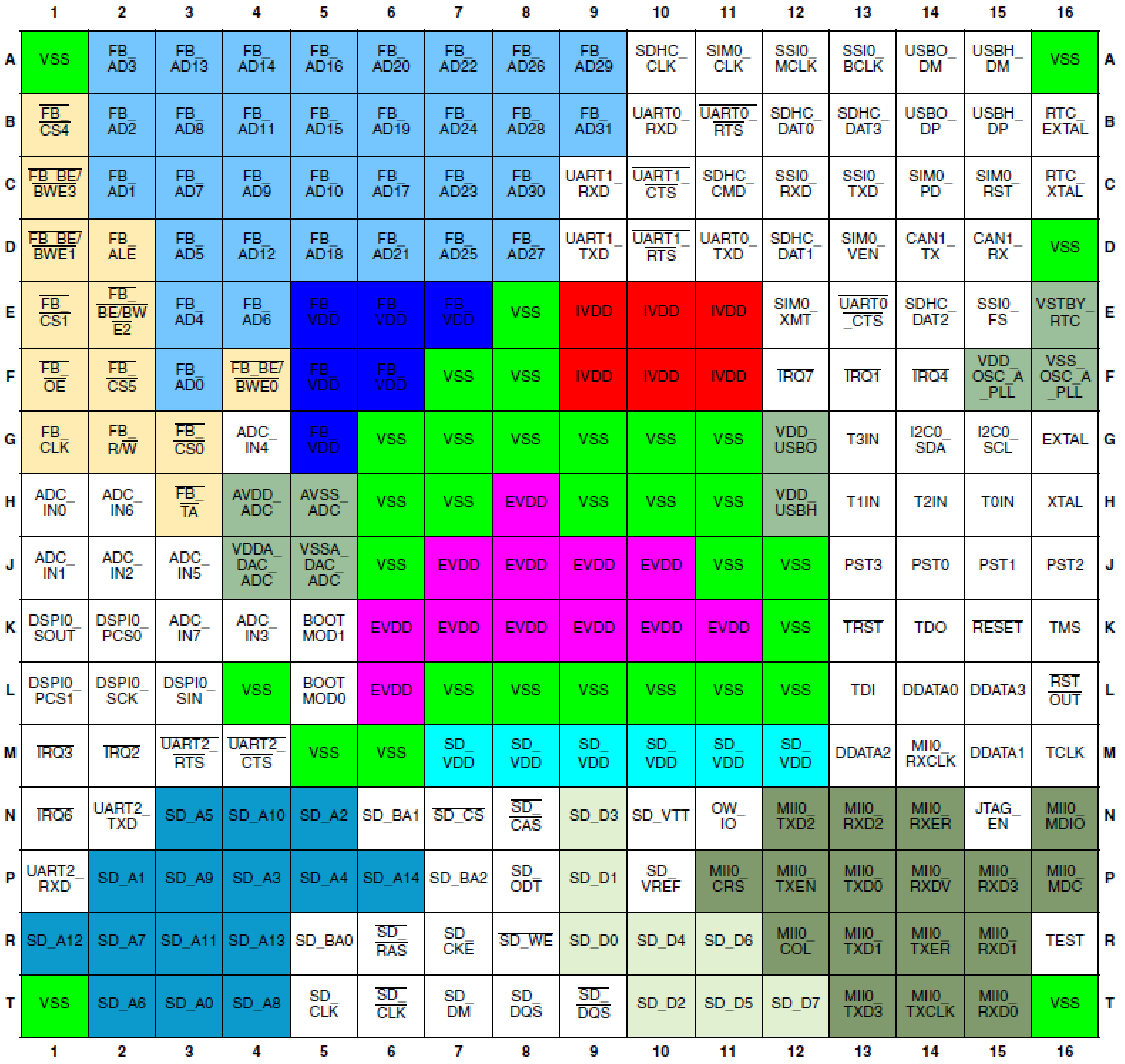
MCF5441X Architecture



Note: Each of the crossbar switch masters, the FlexBus and SDRAM controller have access to peripheral bus controller 0, which is not shown.

Copyright of Freescale Semiconductor, Inc. 2012.

MCF5441X Pinout Diagram



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	VSS	FB_AD3	FB_AD13	FB_AD14	FB_AD16	FB_AD20	FB_AD22	FB_AD26	FB_AD29	SDHC_CLK	SIM0_CLK	SSI0_MCLK	SSI0_BCLK	USBO_DM	USBH_DM	VSS	A
B	FB_CS4	FB_AD2	FB_AD8	FB_AD11	FB_AD15	FB_AD19	FB_AD24	FB_AD28	FB_AD31	UART0_RXD	UART0_RTS	SDHC_DAT0	SDHC_DAT3	USBO_DP	USBH_DP	RTC_EXTAL	B
C	FB_BE/BWE3	FB_AD1	FB_AD7	FB_AD9	FB_AD10	FB_AD17	FB_AD23	FB_AD30	UART1_RXD	UART1_CTS	SDHC_CMD	SSI0_RXD	SSI0_TXD	SIM0_PD	SIM0_RST	RTC_XTAL	C
D	FB_BE/BWE1	FB_ALE	FB_AD5	FB_AD12	FB_AD18	FB_AD21	FB_AD25	FB_AD27	UART1_TXD	UART1_RTS	UART0_TXD	SDHC_DAT1	SIM0_VEN	CAN1_TX	CAN1_RX	VSS	D
E	FB_CS1	FB_BE/BWE2	FB_AD4	FB_AD6	FB_VDD	FB_VDD	FB_VDD	VSS	IVDD	IVDD	IVDD	SIM0_XMT	UART0_CTS	SDHC_DAT2	SSI0_FS	VSTBY_RTC	E
F	FB_OE	FB_CS5	FB_AD0	FB_BE/BWE0	FB_VDD	FB_VDD	VSS	VSS	IVDD	IVDD	IVDD	IRQ7	IRQ1	IRQ4	VDD_OSC_A_PLL	VSS_OSC_A_PLL	F
G	FB_CLK	FB_R/W	FB_CS0	ADC_IN4	FB_VDD	VSS	VSS	VSS	VSS	VSS	VSS	VDD_USBO	T3IN	I2C0_SDA	I2C0_SCL	EXTAL	G
H	ADC_IN0	ADC_IN6	FB_TA	AVDD_ADC	AVSS_ADC	VSS	VSS	EVDD	VSS	VSS	VSS	VDD_USBH	T1IN	T2IN	T0IN	XTAL	H
J	ADC_IN1	ADC_IN2	ADC_IN5	VDDA_DAC_ADC	VSSA_DAC_ADC	VSS	EVDD	EVDD	EVDD	EVDD	VSS	VSS	PST3	PST0	PST1	PST2	J
K	DSPI0_SOUT	DSPI0_PCS0	ADC_IN7	ADC_IN3	BOOTMOD1	EVDD	EVDD	EVDD	EVDD	EVDD	EVDD	VSS	TRST	TDO	RESET	TMS	K
L	DSPI0_PCS1	DSPI0_SCK	DSPI0_SIN	VSS	BOOTMOD0	EVDD	VSS	VSS	VSS	VSS	VSS	VSS	TDI	DDATA0	DDATA3	RSTOUT	L
M	IRQ3	IRQ2	UART2_RTS	UART2_CTS	VSS	VSS	SD_VDD	SD_VDD	SD_VDD	SD_VDD	SD_VDD	SD_VDD	DDATA2	MII0_RXCLK	DDATA1	TCLK	M
N	IRQ6	UART2_TXD	SD_A5	SD_A10	SD_A2	SD_BA1	SD_CS	SD_CAS	SD_D3	SD_VTT	OW_IO	MII0_TXD2	MII0_RXD2	MII0_RXER	JTAG_EN	MII0_MDIO	N
P	UART2_RXD	SD_A1	SD_A9	SD_A3	SD_A4	SD_A14	SD_BA2	SD_ODT	SD_D1	SD_VREF	MII0_CRS	MII0_TXEN	MII0_TXD0	MII0_RXDV	MII0_RXD3	MII0_MDC	P
R	SD_A12	SD_A7	SD_A11	SD_A13	SD_BA0	SD_RAS	SD_CKE	SD_WE	SD_D0	SD_D4	SD_D6	MII0_COL	MII0_TXD1	MII0_TXER	MII0_RXD1	TEST	R
T	VSS	SD_A6	SD_A0	SD_A8	SD_CLK	SD_CLK	SD_DM	SD_DQS	SD_DQS	SD_D2	SD_D5	SD_D7	MII0_TXD3	MII0_TXCLK	MII0_RXD0	VSS	T
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	

Figure 8. MCF54415, MCF54416, MCF54417, and MCF54418 Pinout (256 MAPBGA)

MCF5441X FlexBus

- The FlexBus interface allows the MCF5441X to connect with a wide variety of external devices (e.g., boot ROM, program code in flash memory, FPGAs, simple peripherals) without the need for additional “glue logic”.
- There is a *time-multiplexed* 32-bit address/data bus, which can be configured to pass data to 8-bit, 16-bit or 32-bit external devices. In the first clock cycle, the multiplexed bus carries the address; in the second (and more) clock cycle(s), the data is passed on the selected number of bytes.
- Four byte strobes are provided for the address/data bus.
- Six programmable chip selects (CSs) are available that are asserted for different address ranges, data bit widths (8, 16 or 32 bits), different address set and hold times w.r.t. CS assertion, and different numbers of wait states.

MCF5441X FlexBus Signals

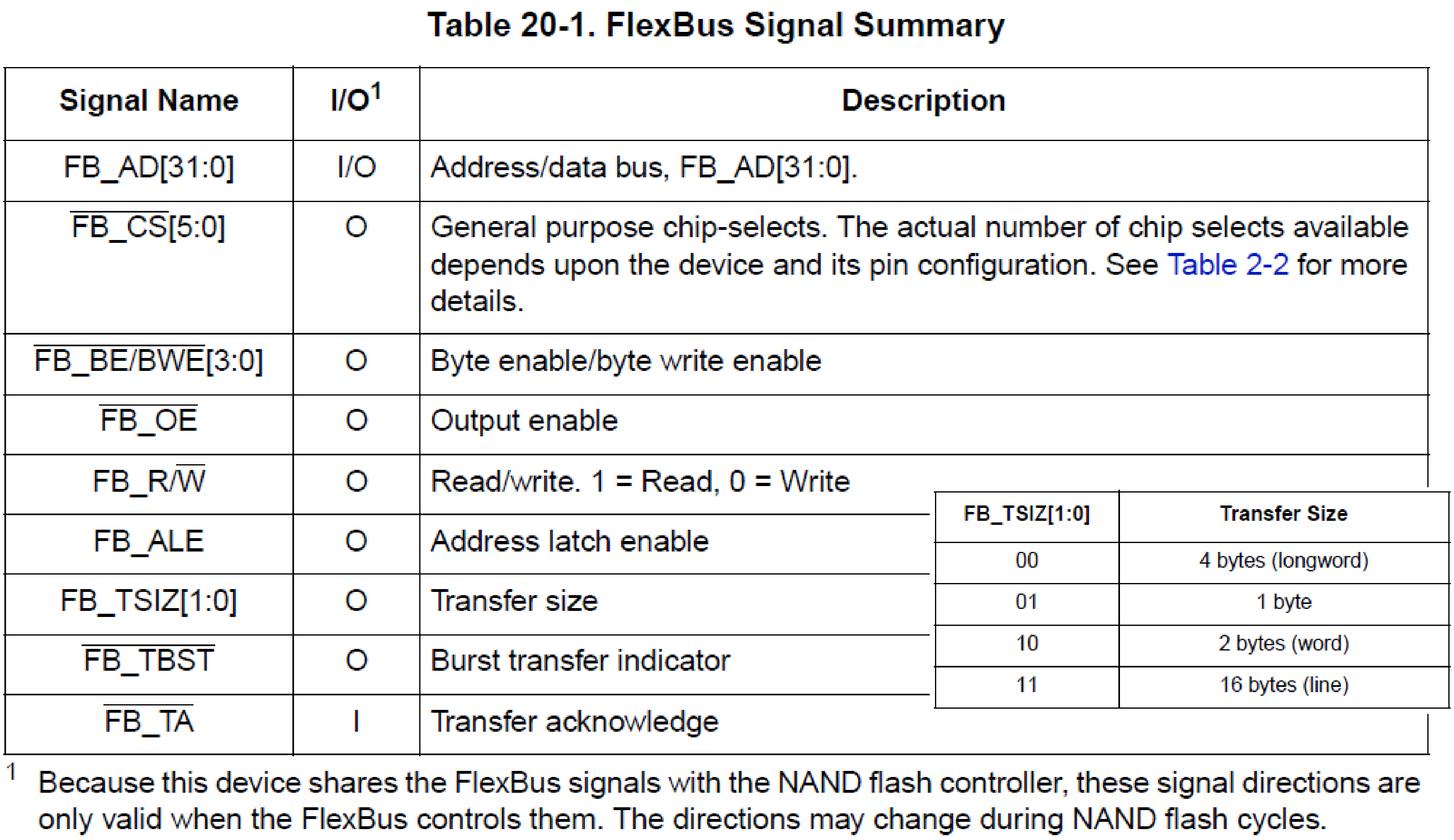


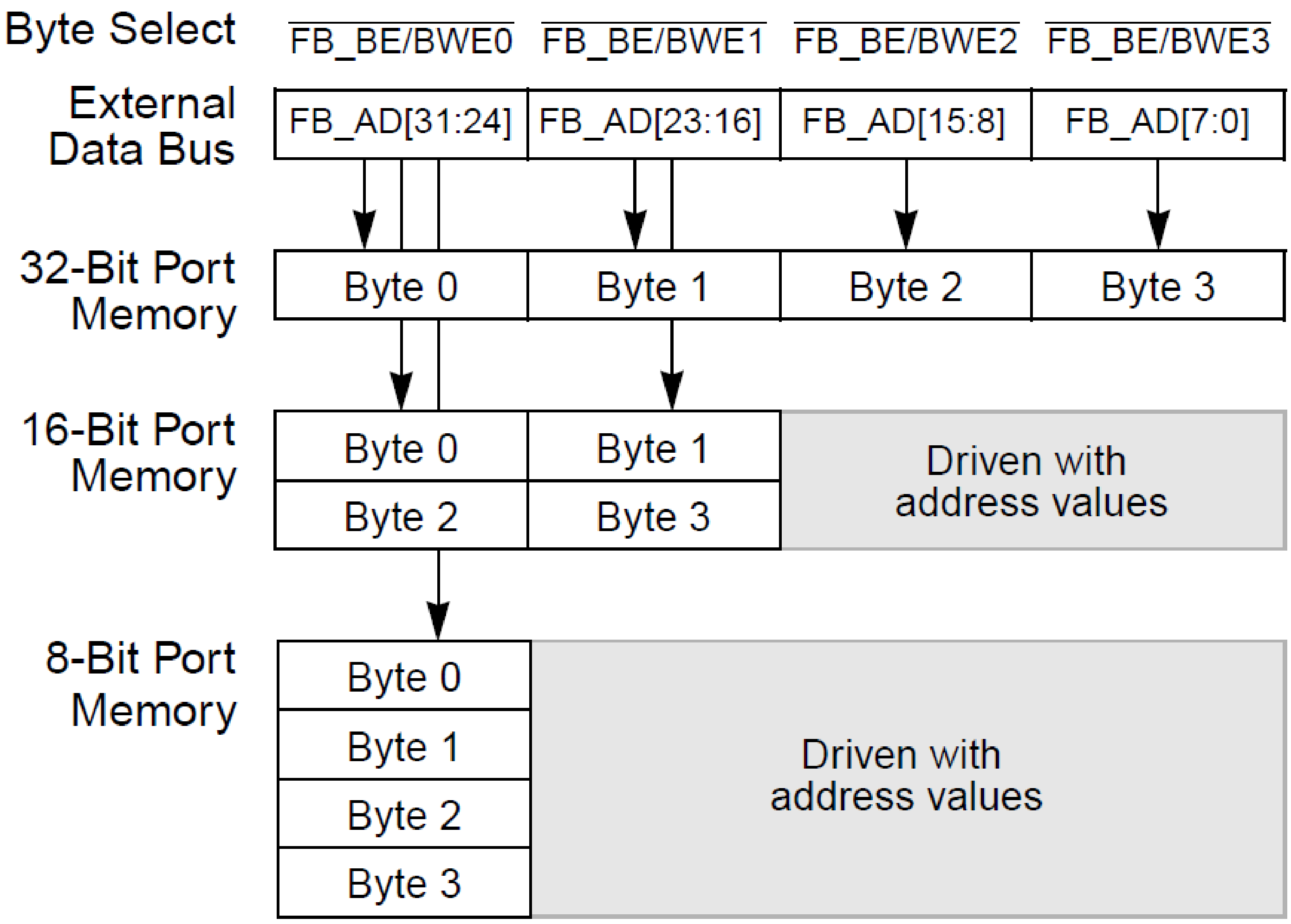
Table 20-1. FlexBus Signal Summary

Signal Name	I/O[1]	Description
FB_AD[31:0]	I/O	Address/data bus, FB_AD[31:0].
$\overline{\text{FB\_CS}}$[5:0]	O	General purpose chip-selects. The actual number of chip selects available depends upon the device and its pin configuration. See Table 2-2 for more details.
$\overline{\text{FB\_BE/BWE}}$[3:0]	O	Byte enable/byte write enable
$\overline{\text{FB\_OE}}$	O	Output enable
FB_R/$\overline{\text{W}}$	O	Read/write. 1 = Read, 0 = Write
FB_ALE	O	Address latch enable
FB_TSIZ[1:0]	O	Transfer size
$\overline{\text{FB\_TBST}}$	O	Burst transfer indicator
$\overline{\text{FB\_TA}}$	I	Transfer acknowledge

[1] Because this device shares the FlexBus signals with the NAND flash controller, these signal directions are only valid when the FlexBus controls them. The directions may change during NAND flash cycles.

FB_TSIZ[1:0]	Transfer Size
00	4 bytes (longword)
01	1 byte
10	2 bytes (word)
11	16 bytes (line)

MCF5441X FlexBus Signals



Byte Select

$\overline{\text{FB\_BE/BWE0}}$ $\overline{\text{FB\_BE/BWE1}}$ $\overline{\text{FB\_BE/BWE2}}$ $\overline{\text{FB\_BE/BWE3}}$

External Data Bus

FB_AD[31:24] FB_AD[23:16] FB_AD[15:8] FB_AD[7:0]

32-Bit Port Memory

Byte 0 Byte 1 Byte 2 Byte 3

16-Bit Port Memory

Byte 0 Byte 1

Byte 2 Byte 3

Driven with address values

8-Bit Port Memory

Byte 0

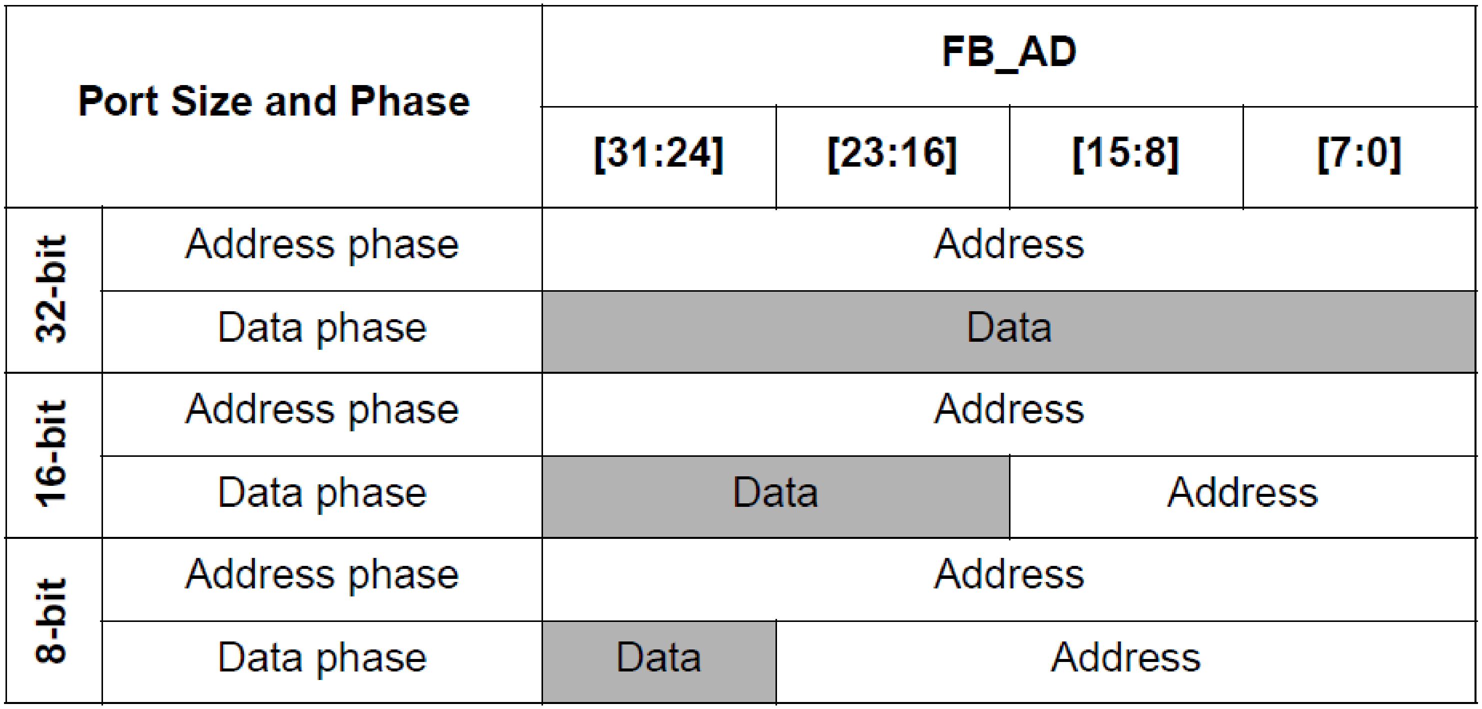
Byte 1

Byte 2

Byte 3

Driven with address values

Multiplexed Address/Data Bus Patterns



Port Size and Phase		FB_AD			
		[31:24]	[23:16]	[15:8]	[7:0]
32-bit	Address phase	Address			
	Data phase	Data			
16-bit	Address phase	Address			
	Data phase	Data		Address	
8-bit	Address phase	Address			
	Data phase	Data	Address		

Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Timing Specs

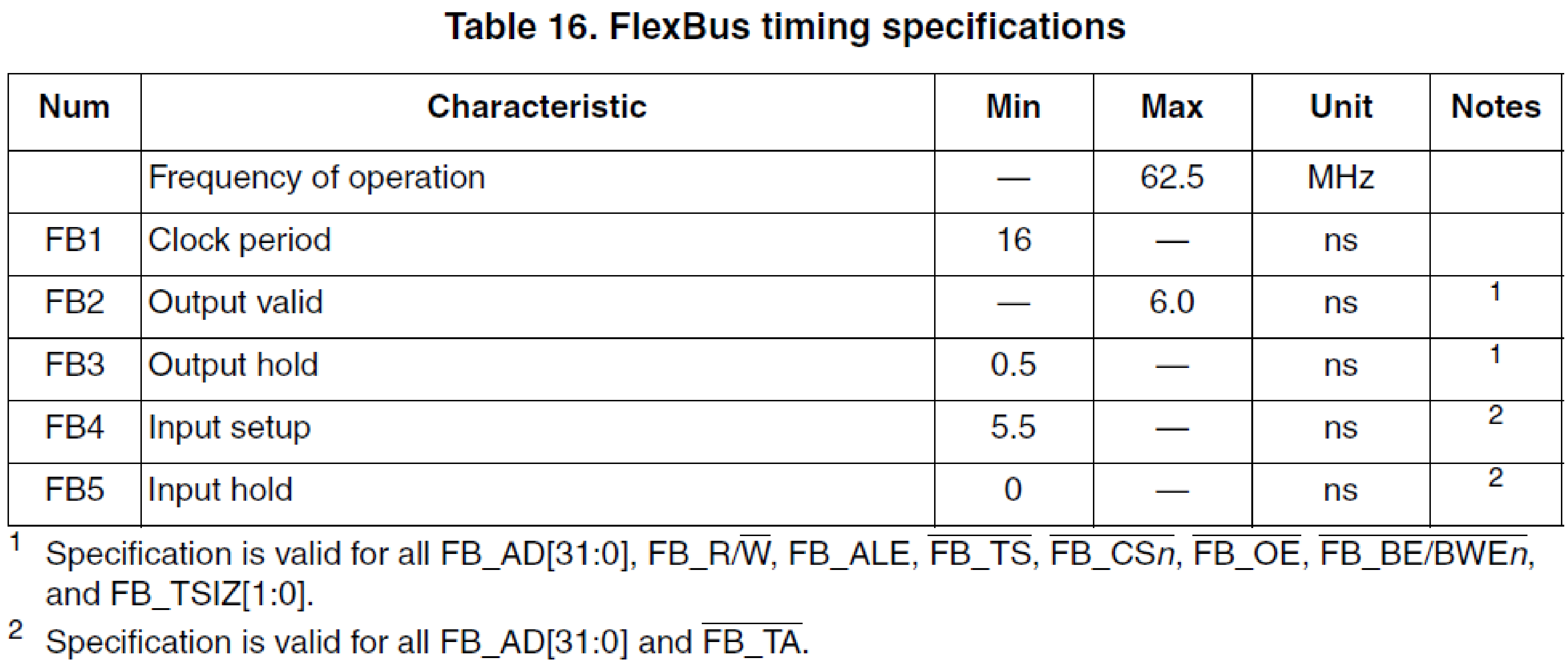


Table 16. FlexBus timing specifications

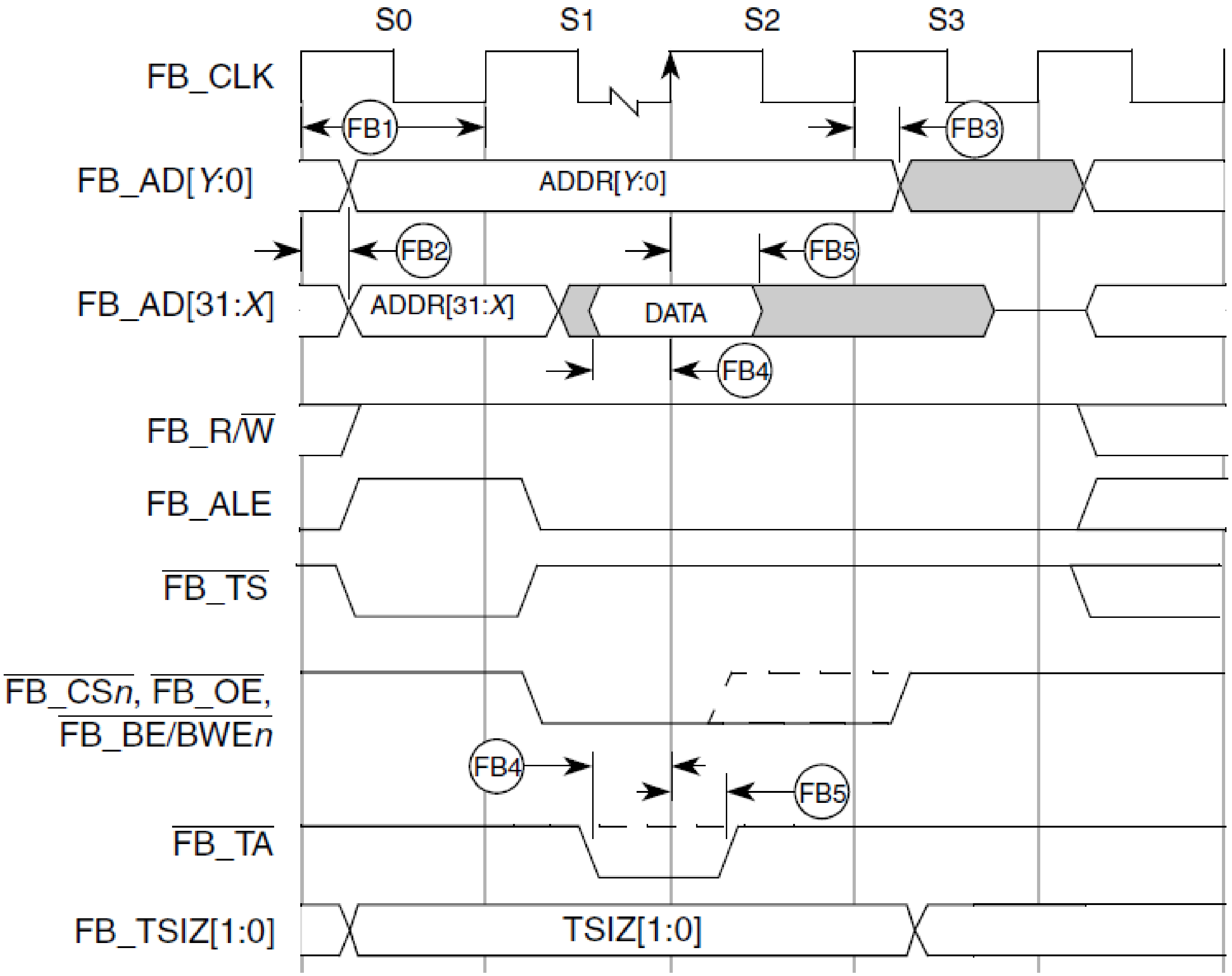
Num	Characteristic	Min	Max	Unit	Notes
	Frequency of operation	—	62.5	MHz	
FB1	Clock period	16	—	ns	
FB2	Output valid	—	6.0	ns	[1]
FB3	Output hold	0.5	—	ns	[1]
FB4	Input setup	5.5	—	ns	[2]
FB5	Input hold	0	—	ns	[2]

[1] Specification is valid for all FB_AD[31:0], FB_R/$\overline{\text{W}}$, FB_ALE, $\overline{\text{FB\_TS}}$, $\overline{\text{FB\_CS}n}$, $\overline{\text{FB\_OE}}$, $\overline{\text{FB\_BE/BWE}n}$, and FB_TSIZ[1:0].

[2] Specification is valid for all FB_AD[31:0] and $\overline{\text{FB\_TA}}$.

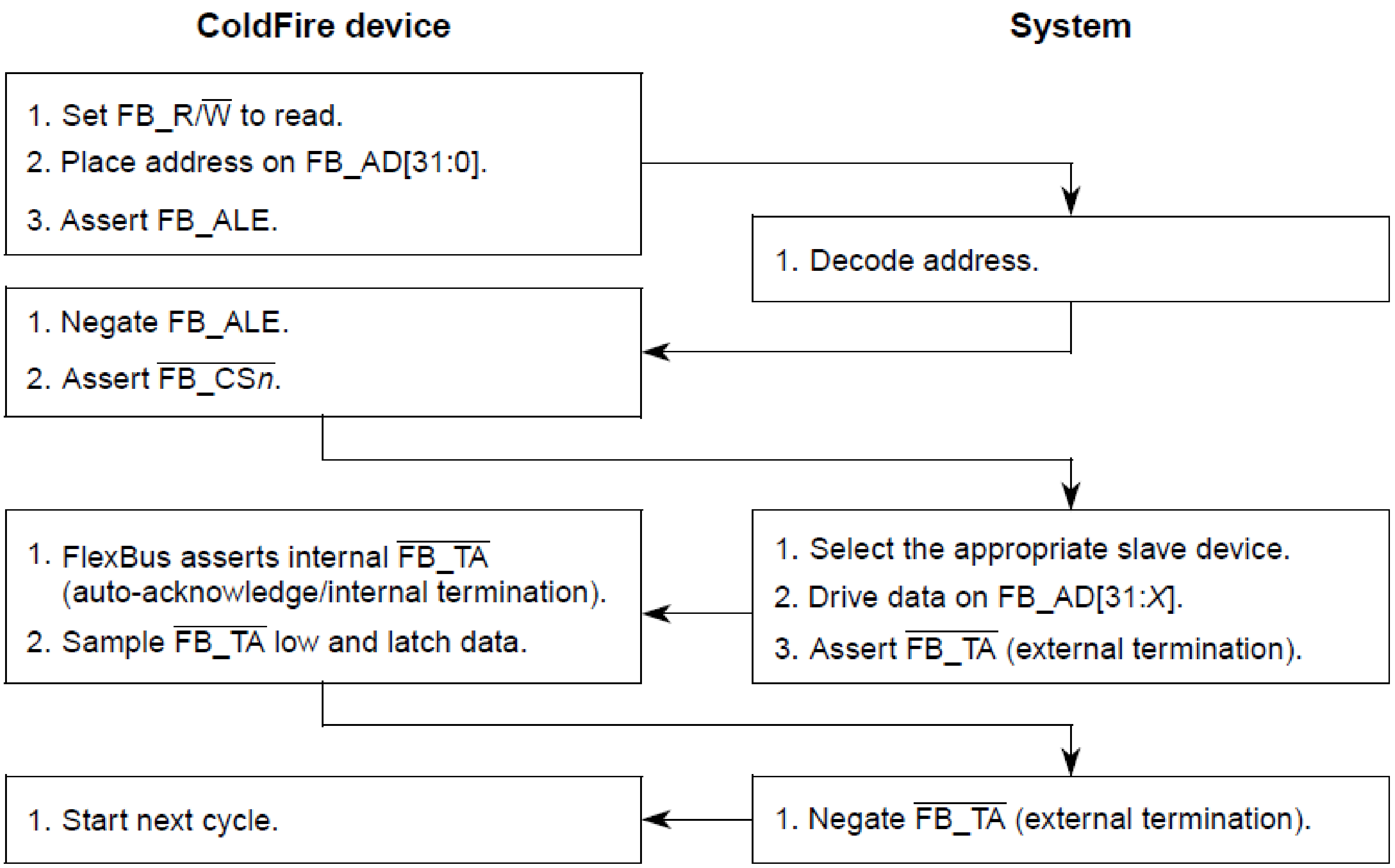
Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Read Cycle (1)



Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Read Cycle (2)



ColdFire device

System

1. Set FB_R/$\overline{\text{W}}$ to read.
2. Place address on FB_AD[31:0].
3. Assert FB_ALE.

1. Decode address.

1. Negate FB_ALE.
2. Assert $\overline{\text{FB\_CS}n}$.

1. Select the appropriate slave device.
2. Drive data on FB_AD[31:*X*].
3. Assert $\overline{\text{FB\_TA}}$ (external termination).

1. FlexBus asserts internal $\overline{\text{FB\_TA}}$ (auto-acknowledge/internal termination).
2. Sample $\overline{\text{FB\_TA}}$ low and latch data.

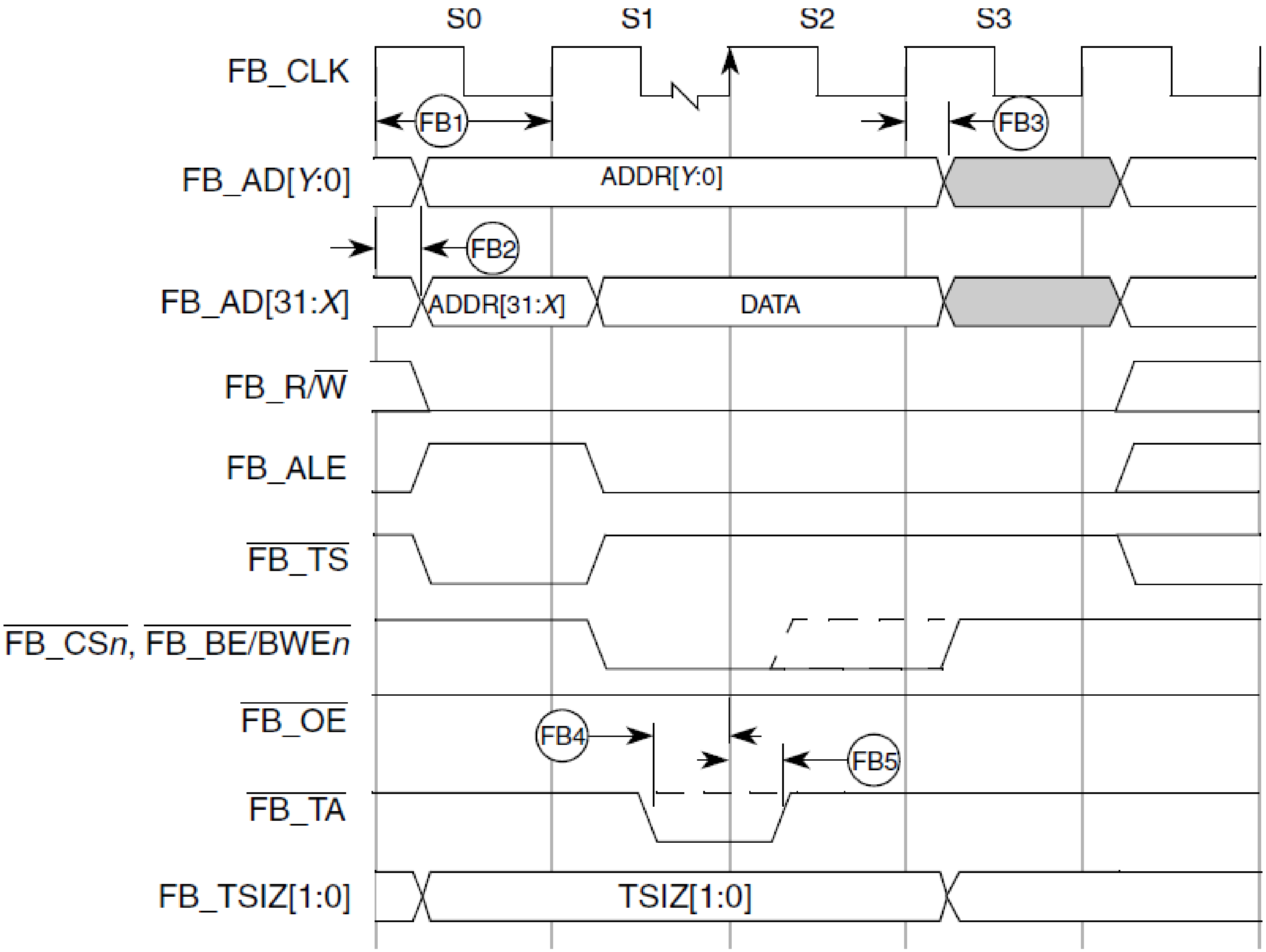
1. Negate $\overline{\text{FB\_TA}}$ (external termination).

1. Start next cycle.

Figure 20-7. Read Cycle Flowchart

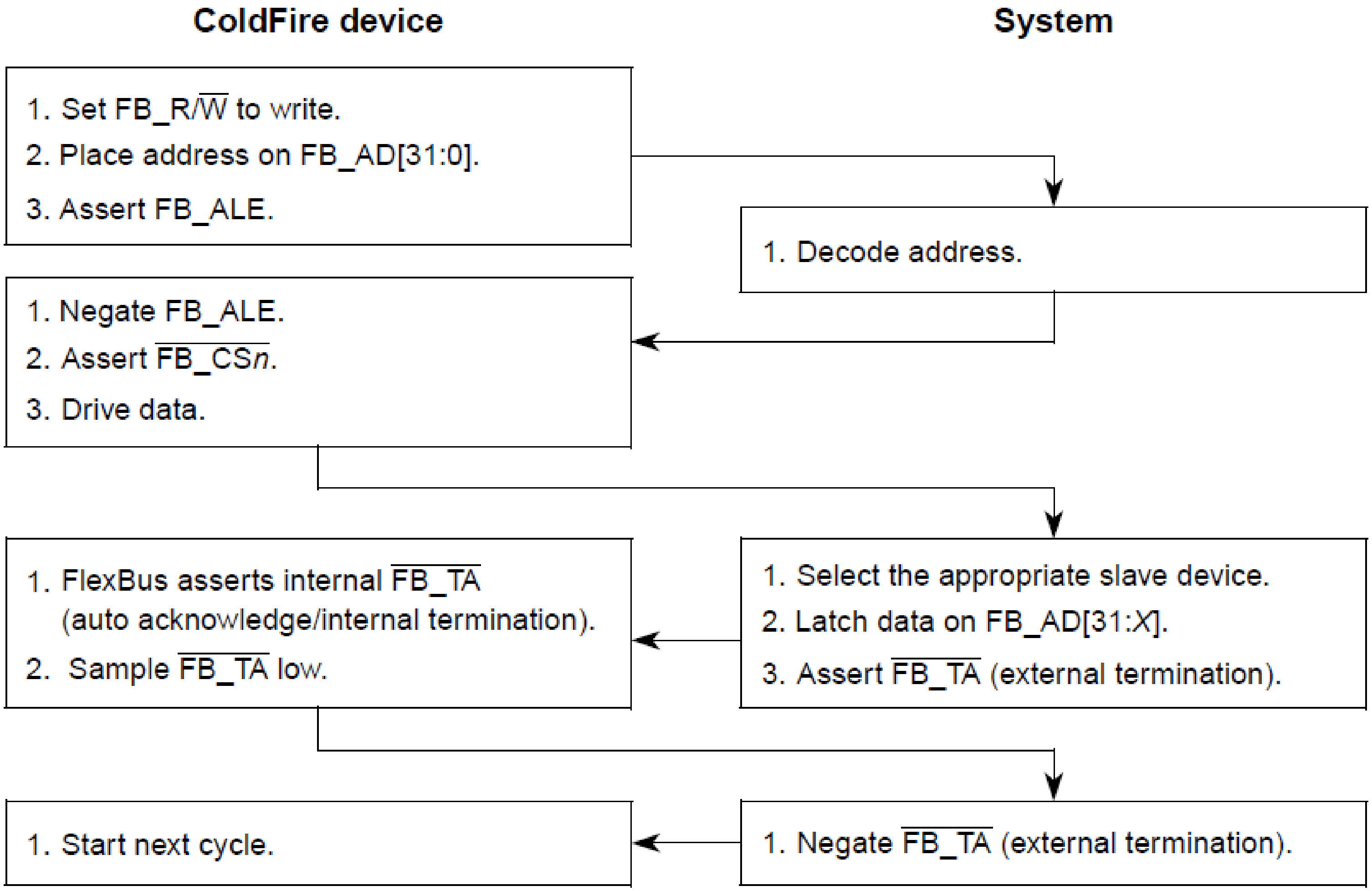
Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Write Cycle (1)



Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Write Cycle (2)



ColdFire device

System

1. Set FB_R/$\overline{\text{W}}$ to write.
2. Place address on FB_AD[31:0].
3. Assert FB_ALE.

1. Decode address.

1. Negate FB_ALE.
2. Assert $\overline{\text{FB\_CS}n}$.
3. Drive data.

1. Select the appropriate slave device.
2. Latch data on FB_AD[31:*X*].
3. Assert $\overline{\text{FB\_TA}}$ (external termination).

1. FlexBus asserts internal $\overline{\text{FB\_TA}}$ (auto acknowledge/internal termination).
2. Sample $\overline{\text{FB\_TA}}$ low.

1. Negate $\overline{\text{FB\_TA}}$ (external termination).

1. Start next cycle.

Figure 20-9. Write-Cycle Flowchart

FlexBus Longword Burst Read from 8-bit Port

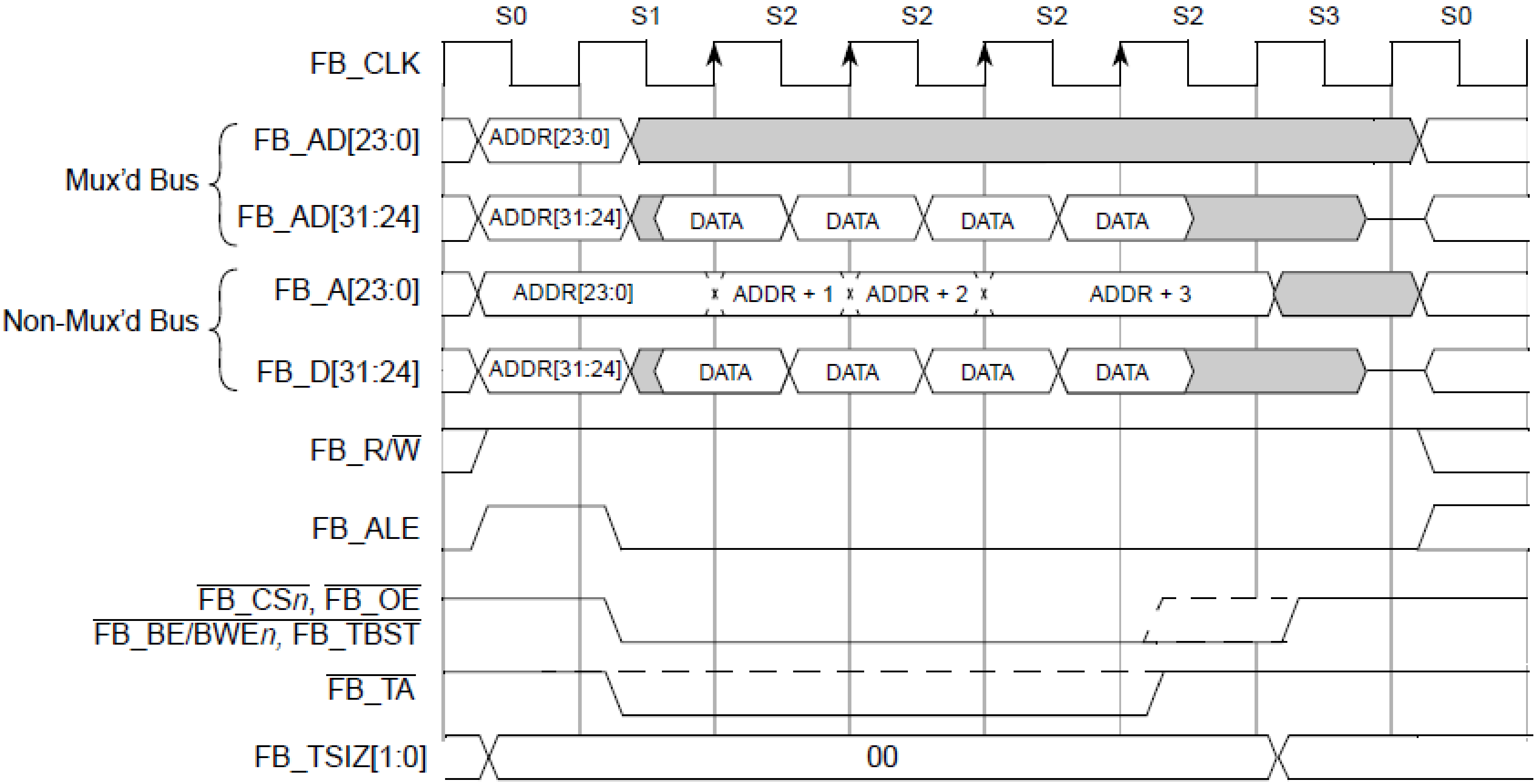


Figure 20-26. Longword-Read Burst from 8-Bit Port 2-1-1-1 (No Wait States)

Copyright of Freescale Semiconductor, Inc. 2012.

FlexBus Longword Burst Write to 8-bit Port

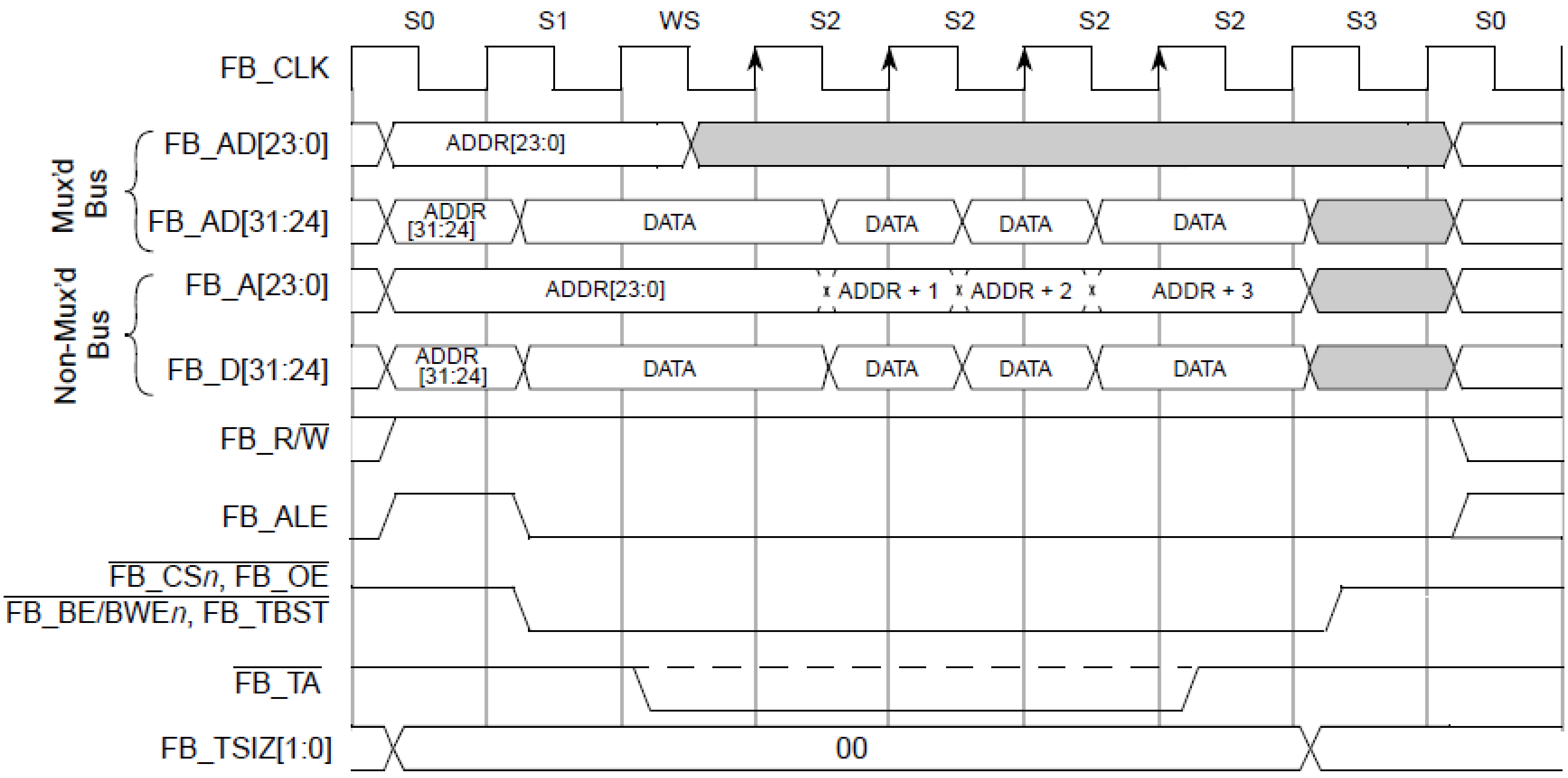
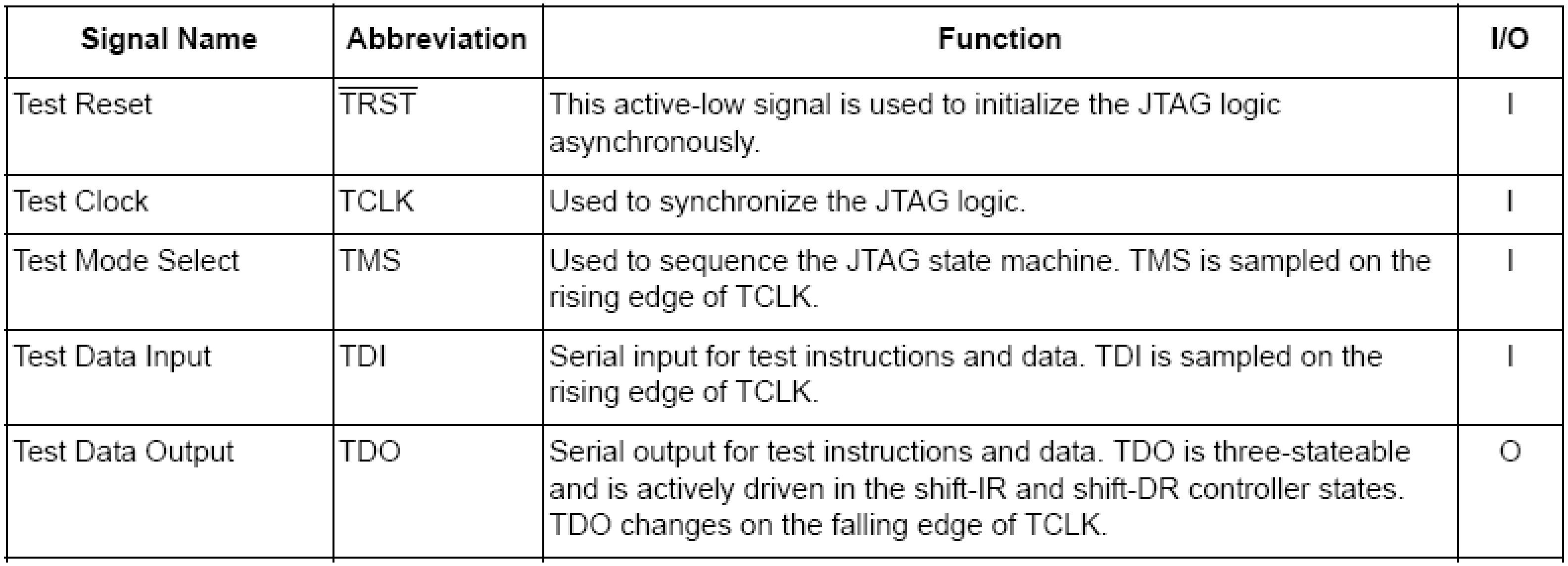


Figure 20-27. Longword-Write Burst to 8-Bit Port 3-1-1-1 (No Wait States)

Copyright of Freescale Semiconductor, Inc. 2012.

The Joint Test Action Group (JTAG) Port

- In 1985 an industry group, called the "Joint Test Action Group" began the development of standard way of accessing chips for production test purposes (to enter test modes, apply tests & retrieve test results).
- The result in 1990 was *IEEE Std. 1149.1-1990 "Standard Test Access Port and Boundary Scan Architecture".*
- The standard specifies a 4 (optionally a 5) pin test bus interface that is commonly called a "JTAG port". The MCF523x has the following:

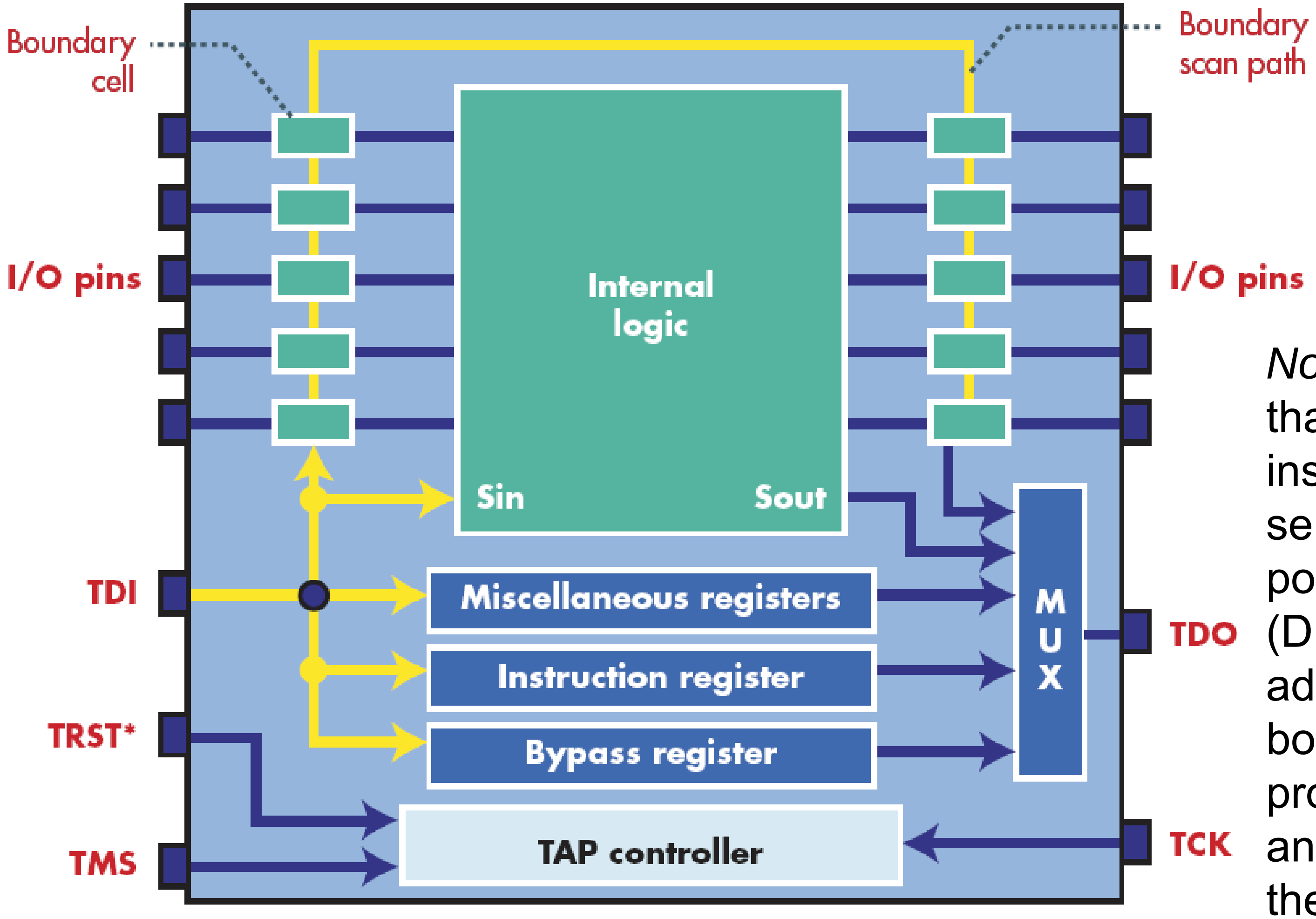


Signal Name	Abbreviation	Function	I/O
Test Reset	$\overline{\text{TRST}}$	This active-low signal is used to initialize the JTAG logic asynchronously.	I
Test Clock	TCLK	Used to synchronize the JTAG logic.	I
Test Mode Select	TMS	Used to sequence the JTAG state machine. TMS is sampled on the rising edge of TCLK.	I
Test Data Input	TDI	Serial input for test instructions and data. TDI is sampled on the rising edge of TCLK.	I
Test Data Output	TDO	Serial output for test instructions and data. TDO is three-stateable and is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCLK.	O

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

JTAG Port (IEEE Std 1149.1) Architecture

Basic IC architecture of IEEE 1149.1.



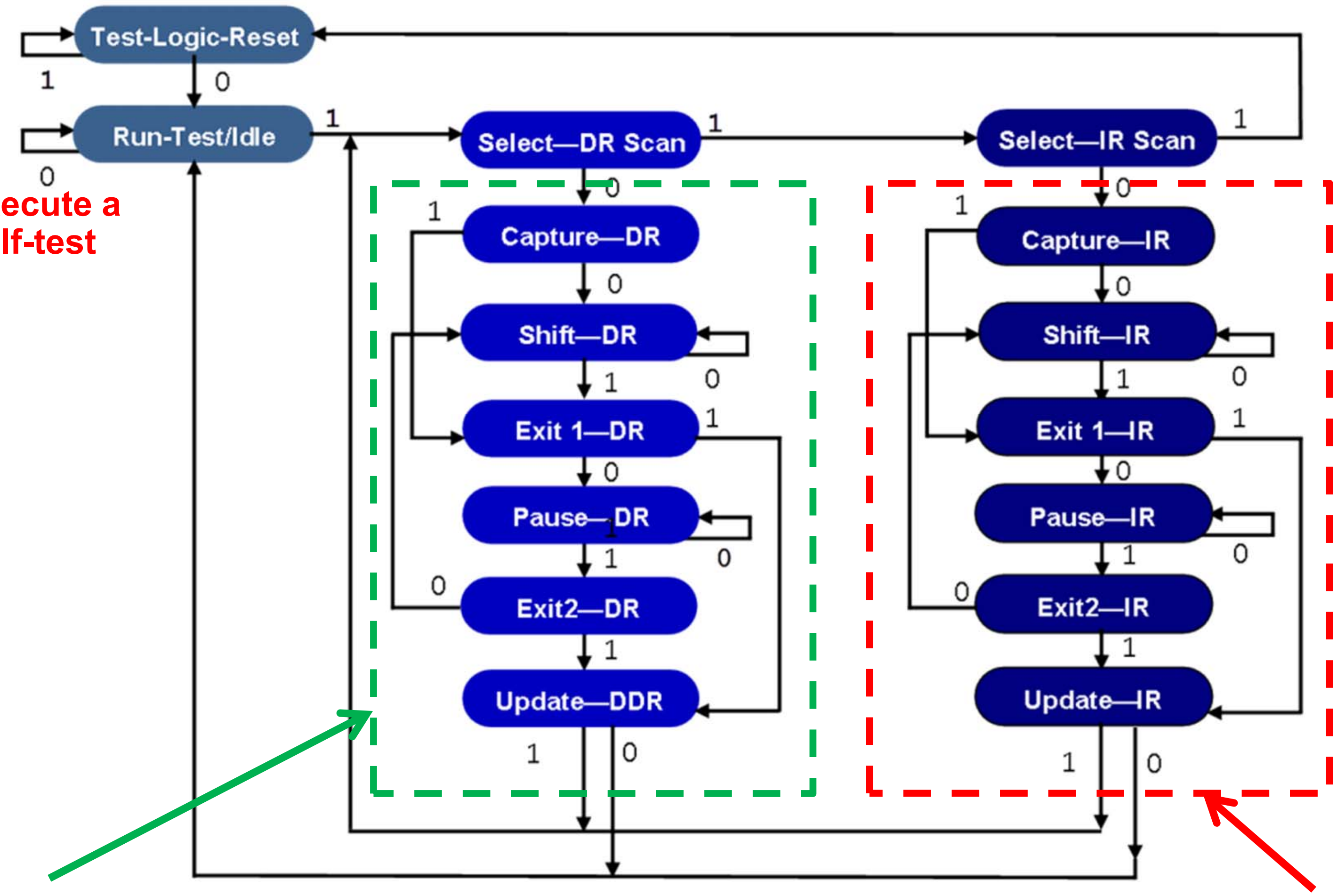
Courtesy of Texas Instruments Inc., 2008.

Note: The instruction that is loaded into the instruction register (IR) selects one of several possible data registers (DRs) to be the one addressed DR. The boundary cells, which provide observability and controllability over the I/O pins, form the boundary scan data register.

The JTAG TAP Controller State Machine

Reset the test logic

Execute a Self-test



Capture status bits for old instruction

Shift out status bits; shift in new Instruction. The instruction specifies one addressed DR.

Update IR with new instruction

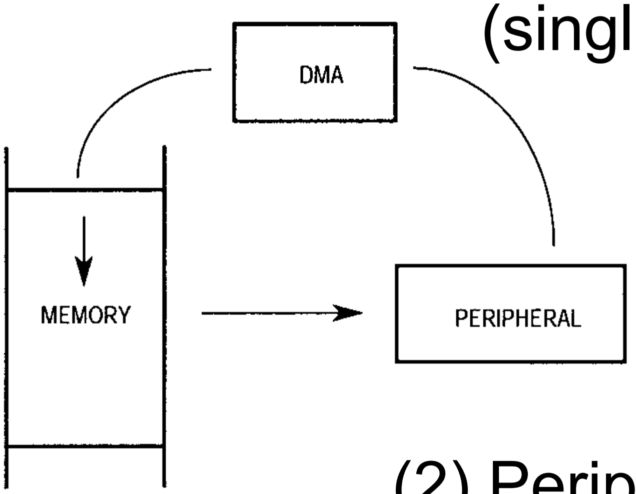
States used to scan out the old contents of the addressed data register (DR), and to load the new contents into the same DR.

Courtesy of Flynn Systems Corp.

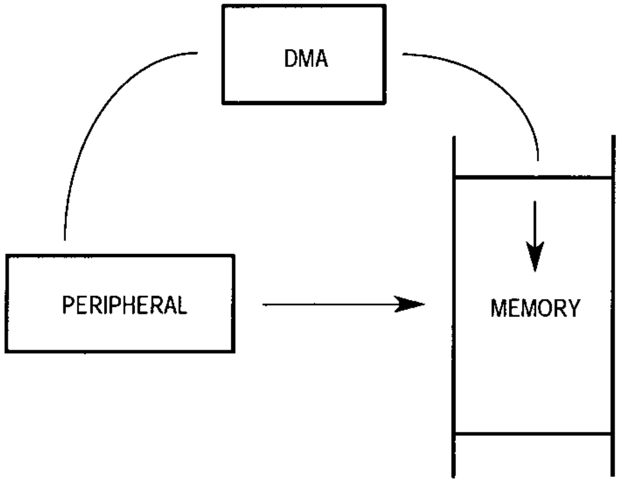
States used to scan out the old instruction, and to load a new instruction into the IR.

Direct Memory Access (DMA) Data Movements

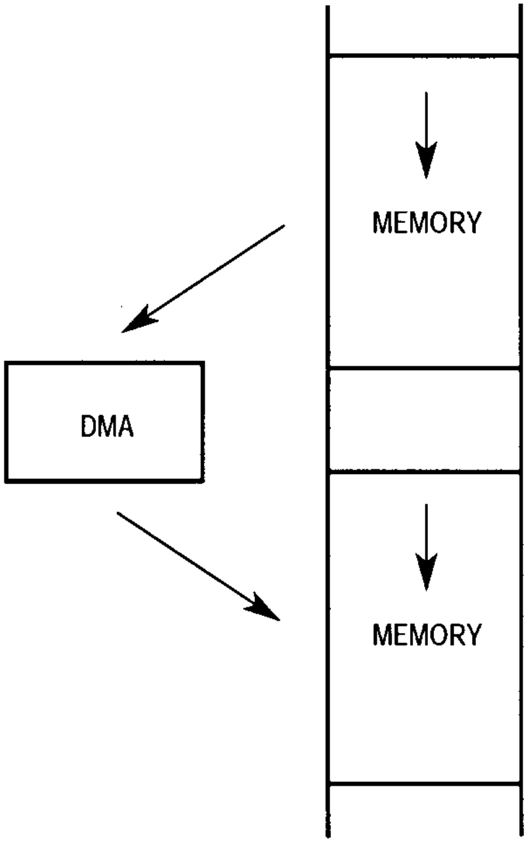
(1) Memory-to-peripheral (single-address)



(2) Peripheral-to-memory (single-address)



(3) Memory-to-memory (dual-address)



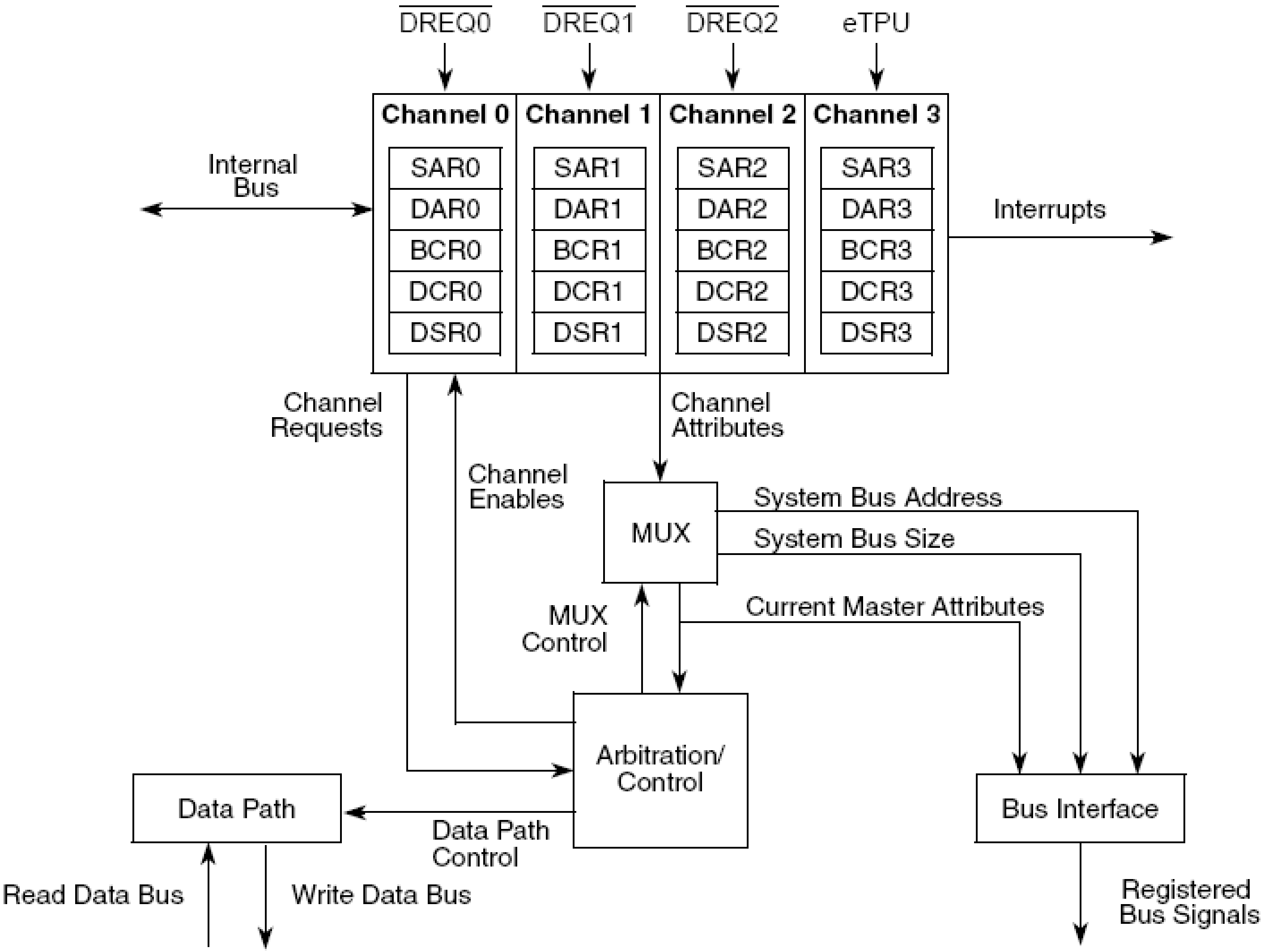
Direct Memory Access in the MCF5441X

- 64 fully programmable channels with independent control descriptors.
- Data movement using dual-address (source & destination) transfers for 8-, 16-, 32- and 128-bit data values.
- Support for nested minor loops (byte counts) and major loops (number of minor loop iterations).
- Can be linked to external request/acknowledge pins.
- Support for channel-to-channel linking for continuous data transfers.
- Support for scatter/gather data patterns.
- Channel transfers can be assigned fixed priorities, or can be assigned round-robin priorities.
- DMA transfers can be triggered by (1) CPU requests, (2) linked channel transfers, and (3) external DMA requests.

Direct Memory Access in the MCF5234

- Four general-purpose Direct Memory Access channels are provided in the MCF5234, which are called: DMA0, DMA1, DMA2 and DMA3.
- Each channel can independently move data over the system bus as bytes, words, longwords, or 16-byte "lines". The source and destinations in each DMA channel can have different data widths.
- Each channel has: (1) a *source address register* (SARn), (2) a *destination address register* (DARn), (3) a *byte count register* (BCRn), (4) a *control register* (DCRn), and (5) a *status register* (DSRn).
- The four DMA channels can receive DMA requests from a variety of possible sources:
  - by the CPU writing the START bit in a DCRn, n = 0, 1, 2 or 3
  - three on-chip UART (serial communications interfaces)
  - four on-chip 32-bit hardware timers
  - the eTPU (treated as if it were an off-chip DMA request)
  - three DMA request signals from off-chip: DREQ0, DREQ1, DREQ2

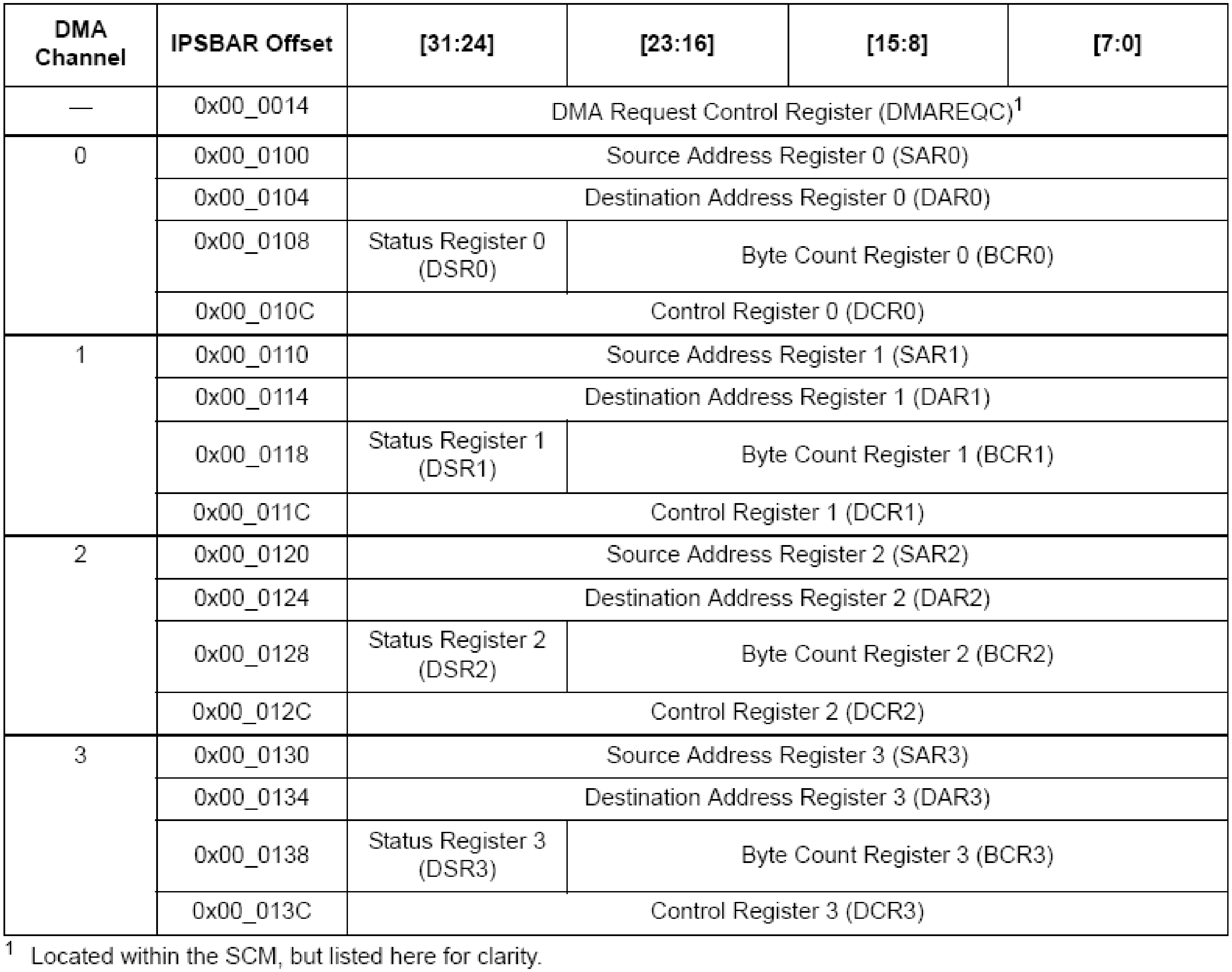
Architecture of the MCF5234 DMA Module



Steps in a DMA Data Transfer

1. **Channel Initialization:** The channel registers are loaded with control information, the initial source address, the initial destination address, and the byte count.
2. **Data Transfer:**
   - Requests for data transfer are received from the CPU, the eTPU, a UART, a hardware timer, or an external peripheral.
   - For each request, data bytes are transferred from the source address to the destination address.
   - One or both addresses are incremented.
   - The data width may be different for the source and destination.
   - The system bus is shared during the transfer using either the cycle-stealing mode or the continuous transfer mode.
   - A 24-bit byte counter is decremented by 1, 2, 4 or 16 to eventually determine when the data transfer has finished.
3. **Channel Termination:** The channel status register is updated. A hardware interrupt may be produced to alert the CPU.

DMA Module Register Map

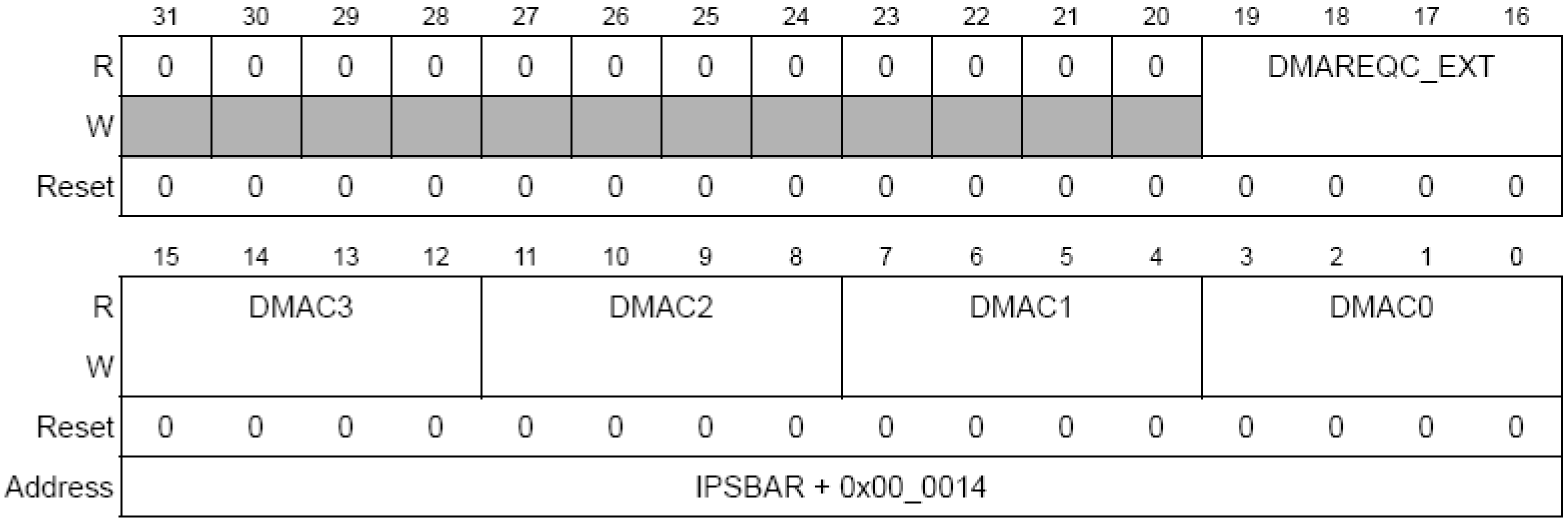


DMA Channel	IPSBAR Offset	[31:24]	[23:16]	[15:8]	[7:0]
—	0x00_0014	DMA Request Control Register (DMAREQC)[1]			
0	0x00_0100	Source Address Register 0 (SAR0)			
	0x00_0104	Destination Address Register 0 (DAR0)			
	0x00_0108	Status Register 0 (DSR0)	Byte Count Register 0 (BCR0)		
	0x00_010C	Control Register 0 (DCR0)			
1	0x00_0110	Source Address Register 1 (SAR1)			
	0x00_0114	Destination Address Register 1 (DAR1)			
	0x00_0118	Status Register 1 (DSR1)	Byte Count Register 1 (BCR1)		
	0x00_011C	Control Register 1 (DCR1)			
2	0x00_0120	Source Address Register 2 (SAR2)			
	0x00_0124	Destination Address Register 2 (DAR2)			
	0x00_0128	Status Register 2 (DSR2)	Byte Count Register 2 (BCR2)		
	0x00_012C	Control Register 2 (DCR2)			
3	0x00_0130	Source Address Register 3 (SAR3)			
	0x00_0134	Destination Address Register 3 (DAR3)			
	0x00_0138	Status Register 3 (DSR3)	Byte Count Register 3 (BCR3)		
	0x00_013C	Control Register 3 (DCR3)			

[1] Located within the SCM, but listed here for clarity.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

DMA Request Control Register (DMAREQC)



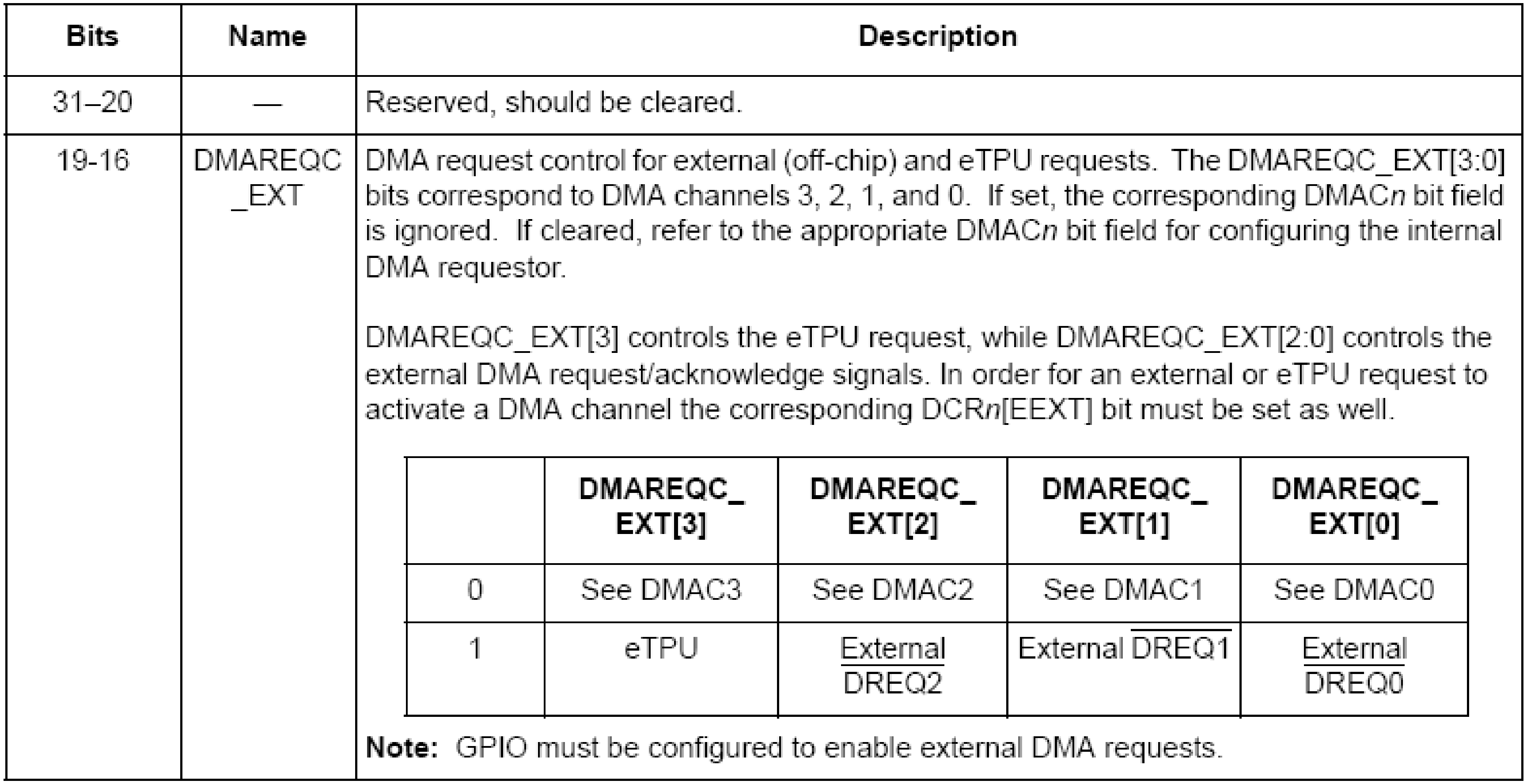
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	DMAREQC_EXT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DMAC3				DMAC2				DMAC1				DMAC0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_0014															

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

- Entries of value 1 in the DMAREQC_EXT field determine which of the eTPU and off-chip signals DREQ2-0 can request DMA transfers using channels DMAC3-0, respectively.
- If a DMARREQC_EXT bit is 0, then the source of DMA requests for that channel is determined by the corresponding DMACn field.

DMAREQC Bits 31-16



<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>31–20</td><td>—</td><td>Reserved, should be cleared.</td></tr>
<tr><td>19-16</td><td>DMAREQC
_EXT</td><td>DMA request control for external (off-chip) and eTPU requests. The DMAREQC_EXT[3:0] bits correspond to DMA channels 3, 2, 1, and 0. If set, the corresponding DMAC<i>n</i> bit field is ignored. If cleared, refer to the appropriate DMAC<i>n</i> bit field for configuring the internal DMA requestor.

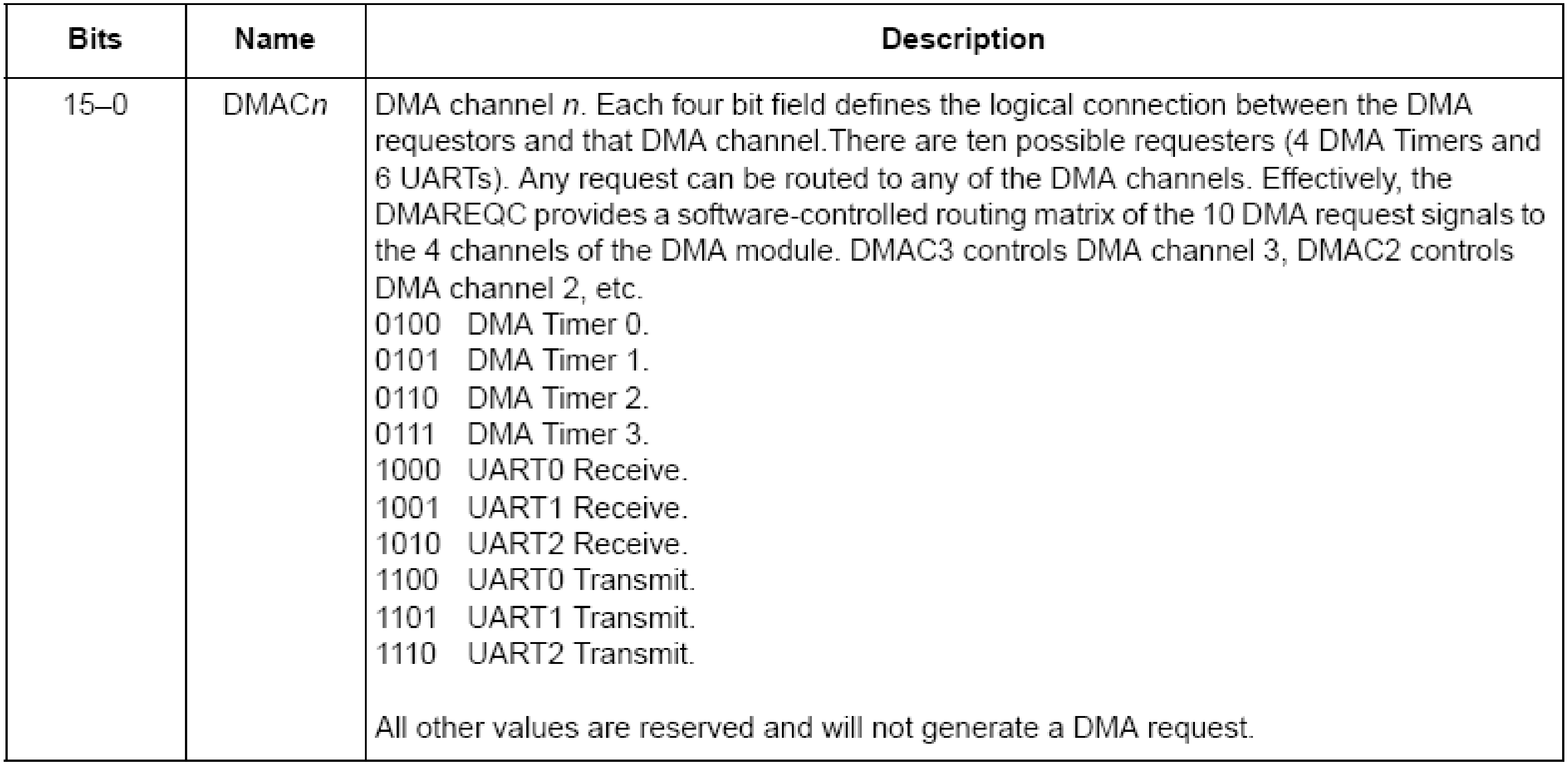
DMAREQC_EXT[3] controls the eTPU request, while DMAREQC_EXT[2:0] controls the external DMA request/acknowledge signals. In order for an external or eTPU request to activate a DMA channel the corresponding DCR<i>n</i>[EEXT] bit must be set as well.

<table>
<tr><th></th><th>DMAREQC_
EXT[3]</th><th>DMAREQC_
EXT[2]</th><th>DMAREQC_
EXT[1]</th><th>DMAREQC_
EXT[0]</th></tr>
<tr><td>0</td><td>See DMAC3</td><td>See DMAC2</td><td>See DMAC1</td><td>See DMAC0</td></tr>
<tr><td>1</td><td>eTPU</td><td>External $\overline{\text{DREQ2}}$</td><td>External $\overline{\text{DREQ1}}$</td><td>External $\overline{\text{DREQ0}}$</td></tr>
</table>

<b>Note:</b> GPIO must be configured to enable external DMA requests.</td></tr>
</table>

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

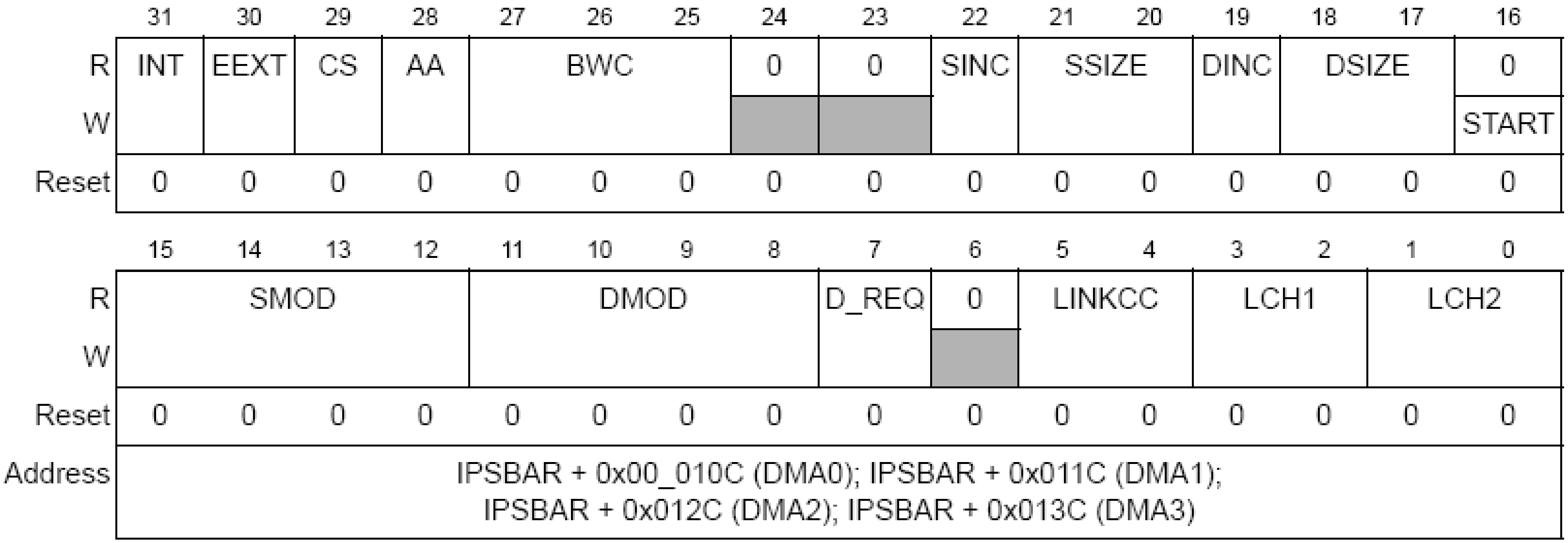
DMAREQC Bits 15-0



Bits	Name	Description
15–0	DMAC*n*	DMA channel *n*. Each four bit field defines the logical connection between the DMA requestors and that DMA channel.There are ten possible requesters (4 DMA Timers and 6 UARTs). Any request can be routed to any of the DMA channels. Effectively, the DMAREQC provides a software-controlled routing matrix of the 10 DMA request signals to the 4 channels of the DMA module. DMAC3 controls DMA channel 3, DMAC2 controls DMA channel 2, etc. 0100 DMA Timer 0. 0101 DMA Timer 1. 0110 DMA Timer 2. 0111 DMA Timer 3. 1000 UART0 Receive. 1001 UART1 Receive. 1010 UART2 Receive. 1100 UART0 Transmit. 1101 UART1 Transmit. 1110 UART2 Transmit. All other values are reserved and will not generate a DMA request.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

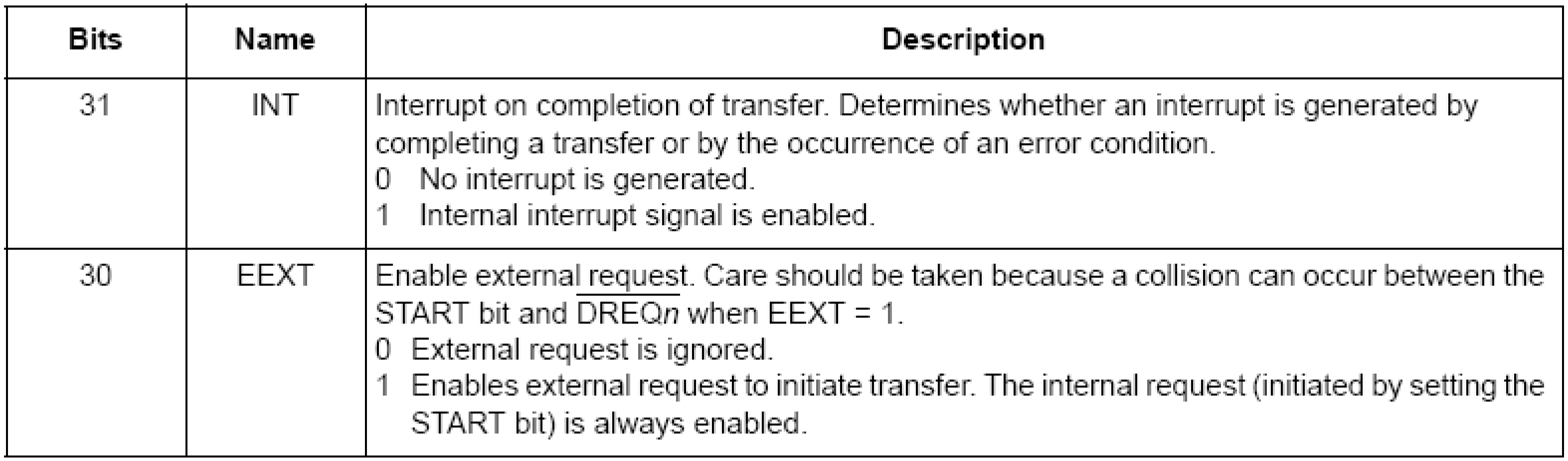
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W								▨	▨							START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W										▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

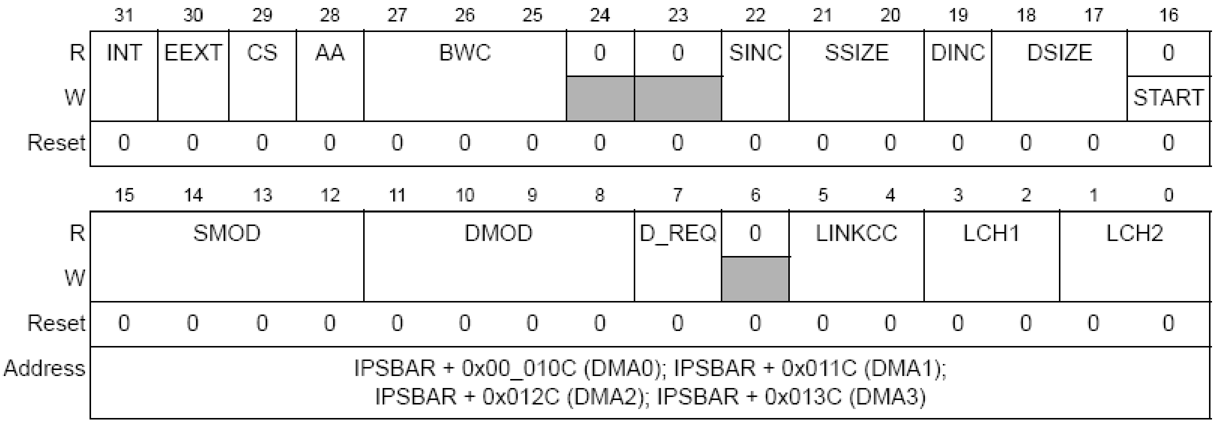
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.



Bits	Name	Description
31	INT	Interrupt on completion of transfer. Determines whether an interrupt is generated by completing a transfer or by the occurrence of an error condition. 0 No interrupt is generated. 1 Internal interrupt signal is enabled.
30	EEXT	Enable external request. Care should be taken because a collision can occur between the START bit and $\overline{\text{DREQ}n}$ when EEXT = 1. 0 External request is ignored. 1 Enables external request to initiate transfer. The internal request (initiated by setting the START bit) is always enabled.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

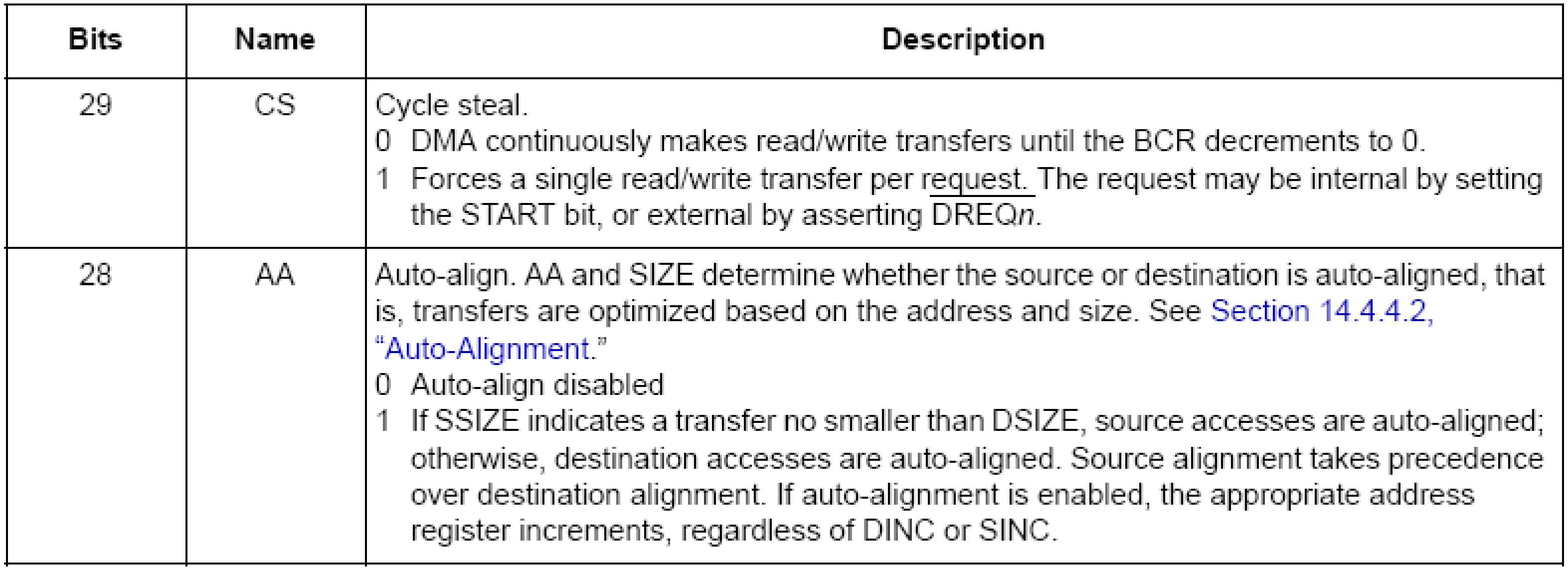
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W								▨	▨							START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W										▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

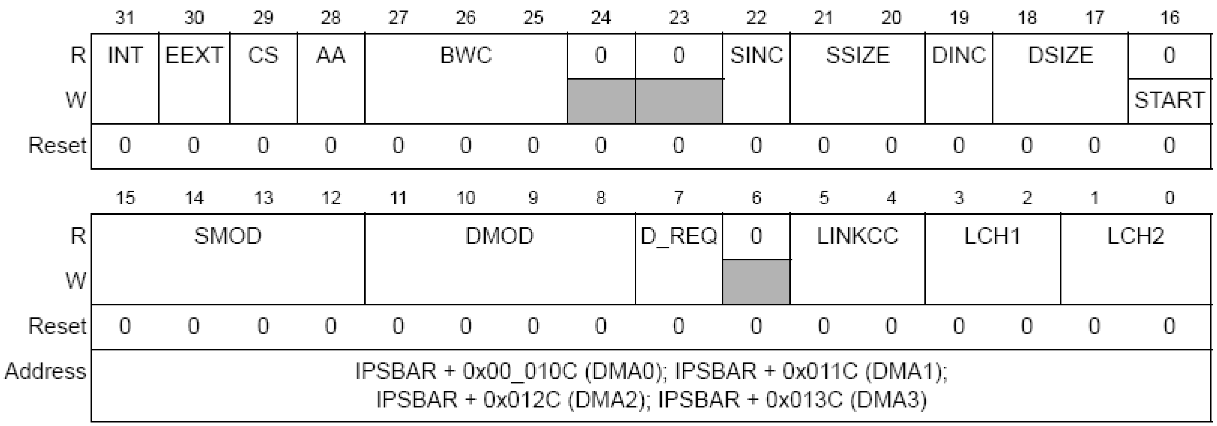
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.



Bits	Name	Description
29	CS	Cycle steal. 0 DMA continuously makes read/write transfers until the BCR decrements to 0. 1 Forces a single read/write transfer per request. The request may be internal by setting the START bit, or external by asserting DREQ*n*.
28	AA	Auto-align. AA and SIZE determine whether the source or destination is auto-aligned, that is, transfers are optimized based on the address and size. See Section 14.4.4.2, "Auto-Alignment." 0 Auto-align disabled 1 If SSIZE indicates a transfer no smaller than DSIZE, source accesses are auto-aligned; otherwise, destination accesses are auto-aligned. Source alignment takes precedence over destination alignment. If auto-alignment is enabled, the appropriate address register increments, regardless of DINC or SINC.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

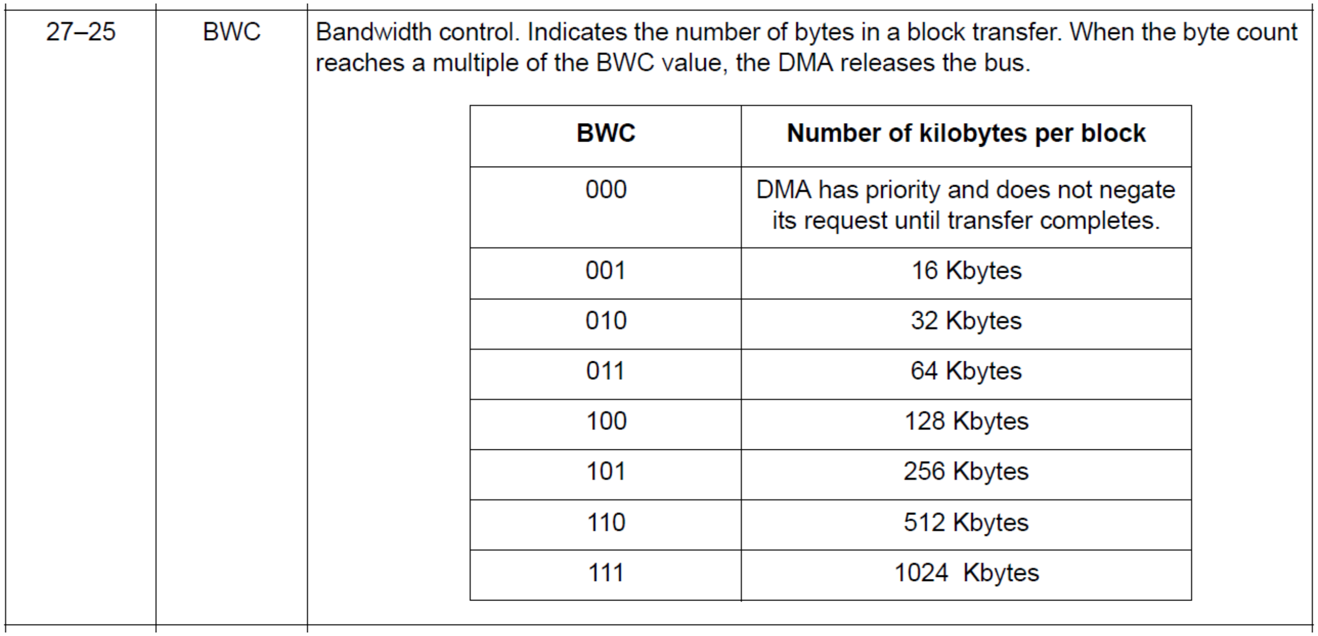
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W								▨	▨							START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W										▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

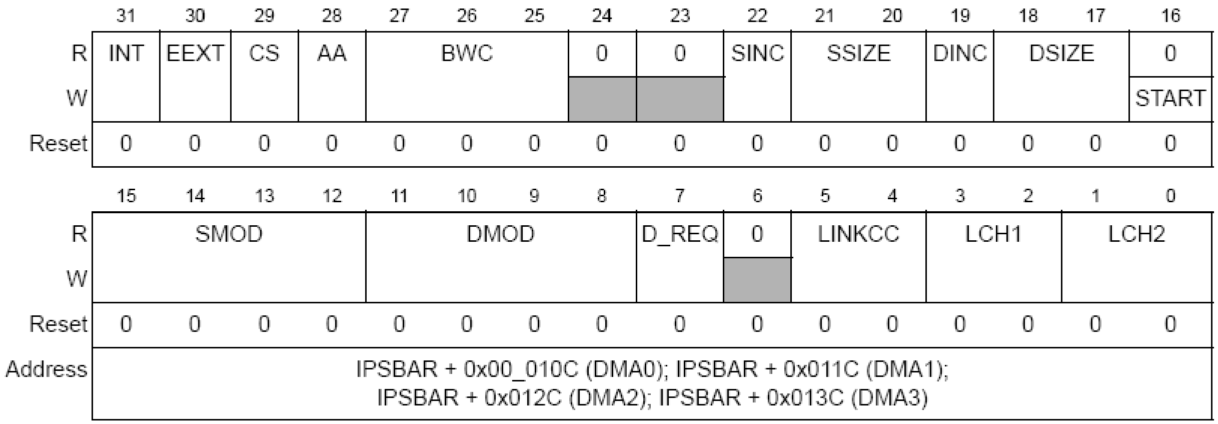


27–25	BWC	Bandwidth control. Indicates the number of bytes in a block transfer. When the byte count reaches a multiple of the BWC value, the DMA releases the bus.

BWC	Number of kilobytes per block
000	DMA has priority and does not negate its request until transfer completes.
001	16 Kbytes
010	32 Kbytes
011	64 Kbytes
100	128 Kbytes
101	256 Kbytes
110	512 Kbytes
111	1024 Kbytes

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

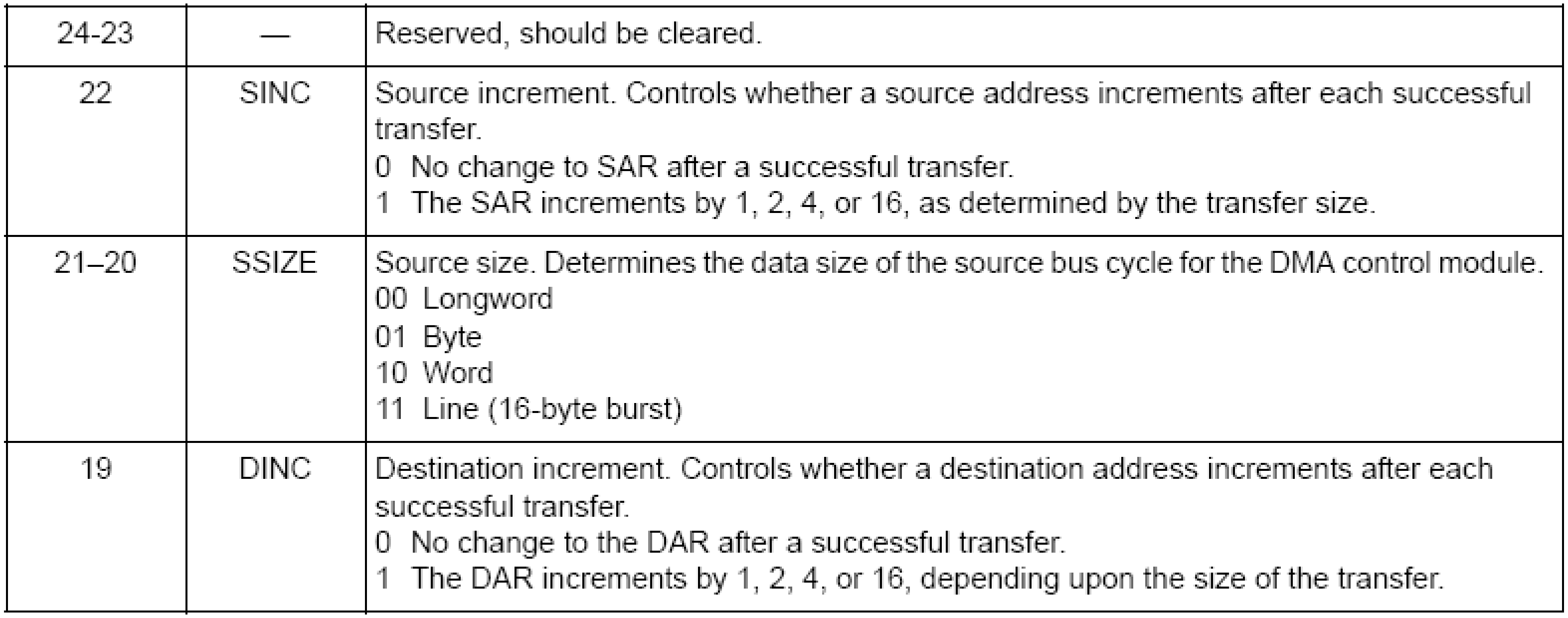
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W								▨	▨							START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W										▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

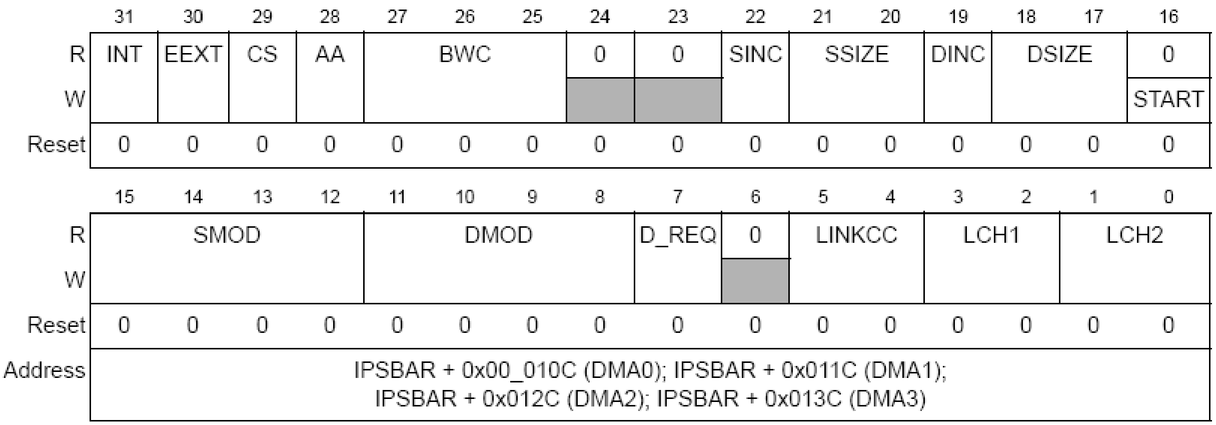
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.



24-23	—	Reserved, should be cleared.
22	SINC	Source increment. Controls whether a source address increments after each successful transfer. 0 No change to SAR after a successful transfer. 1 The SAR increments by 1, 2, 4, or 16, as determined by the transfer size.
21–20	SSIZE	Source size. Determines the data size of the source bus cycle for the DMA control module. 00 Longword 01 Byte 10 Word 11 Line (16-byte burst)
19	DINC	Destination increment. Controls whether a destination address increments after each successful transfer. 0 No change to the DAR after a successful transfer. 1 The DAR increments by 1, 2, 4, or 16, depending upon the size of the transfer.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

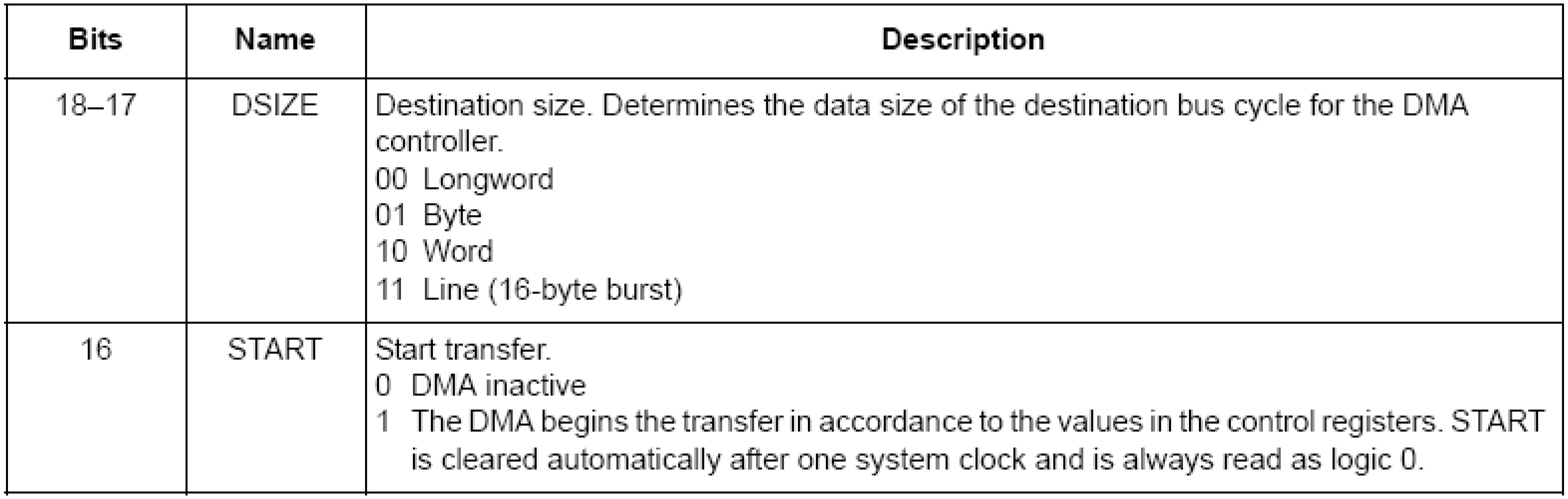
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W																START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

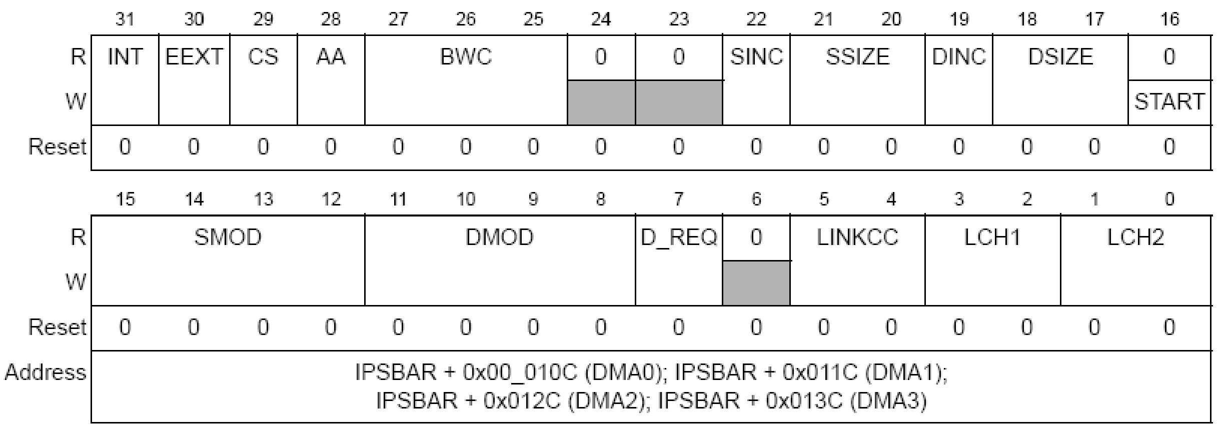
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.



Bits	Name	Description
18–17	DSIZE	Destination size. Determines the data size of the destination bus cycle for the DMA controller. 00 Longword 01 Byte 10 Word 11 Line (16-byte burst)
16	START	Start transfer. 0 DMA inactive 1 The DMA begins the transfer in accordance to the values in the control registers. START is cleared automatically after one system clock and is always read as logic 0.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

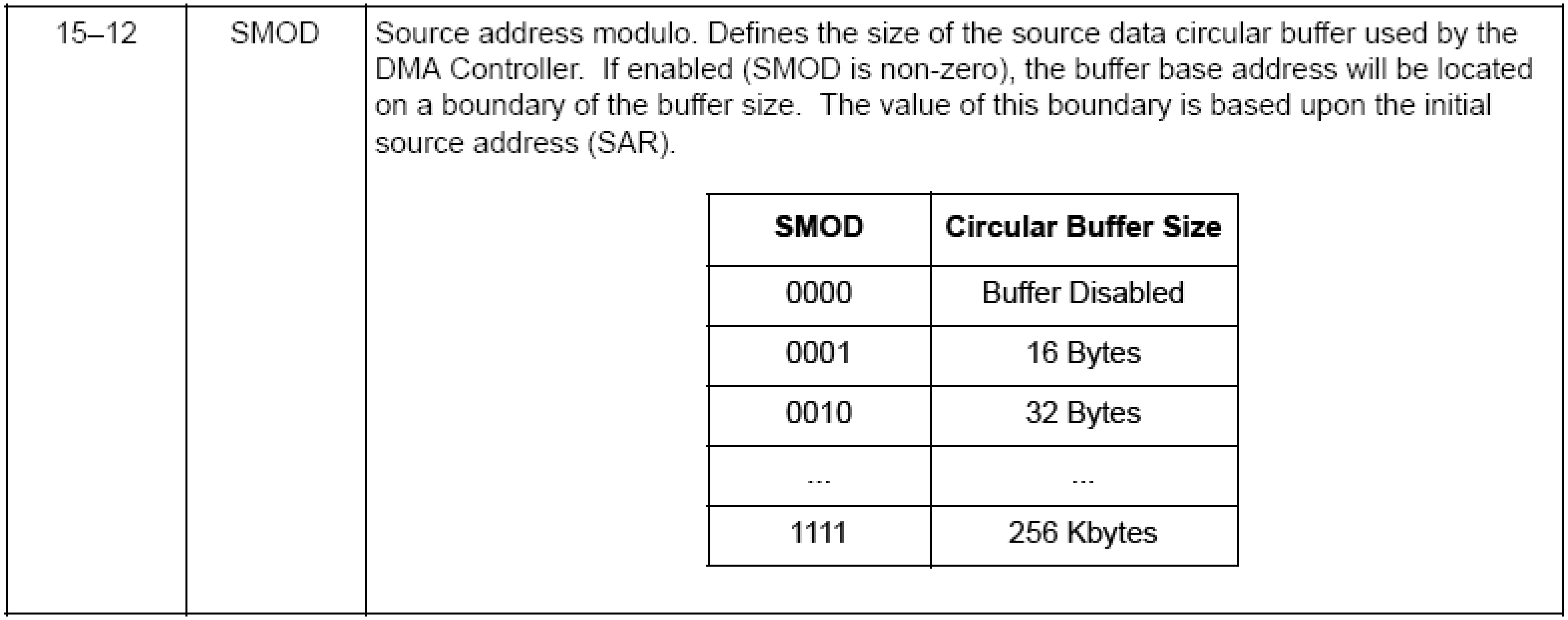
DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W																START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

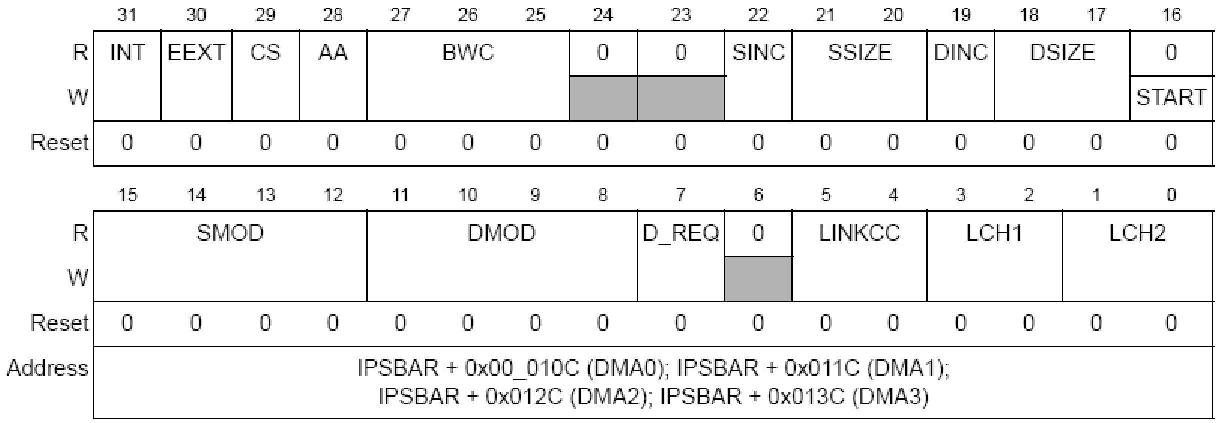


15–12	SMOD	Source address modulo. Defines the size of the source data circular buffer used by the DMA Controller. If enabled (SMOD is non-zero), the buffer base address will be located on a boundary of the buffer size. The value of this boundary is based upon the initial source address (SAR).

SMOD	Circular Buffer Size
0000	Buffer Disabled
0001	16 Bytes
0010	32 Bytes
...	...
1111	256 Kbytes

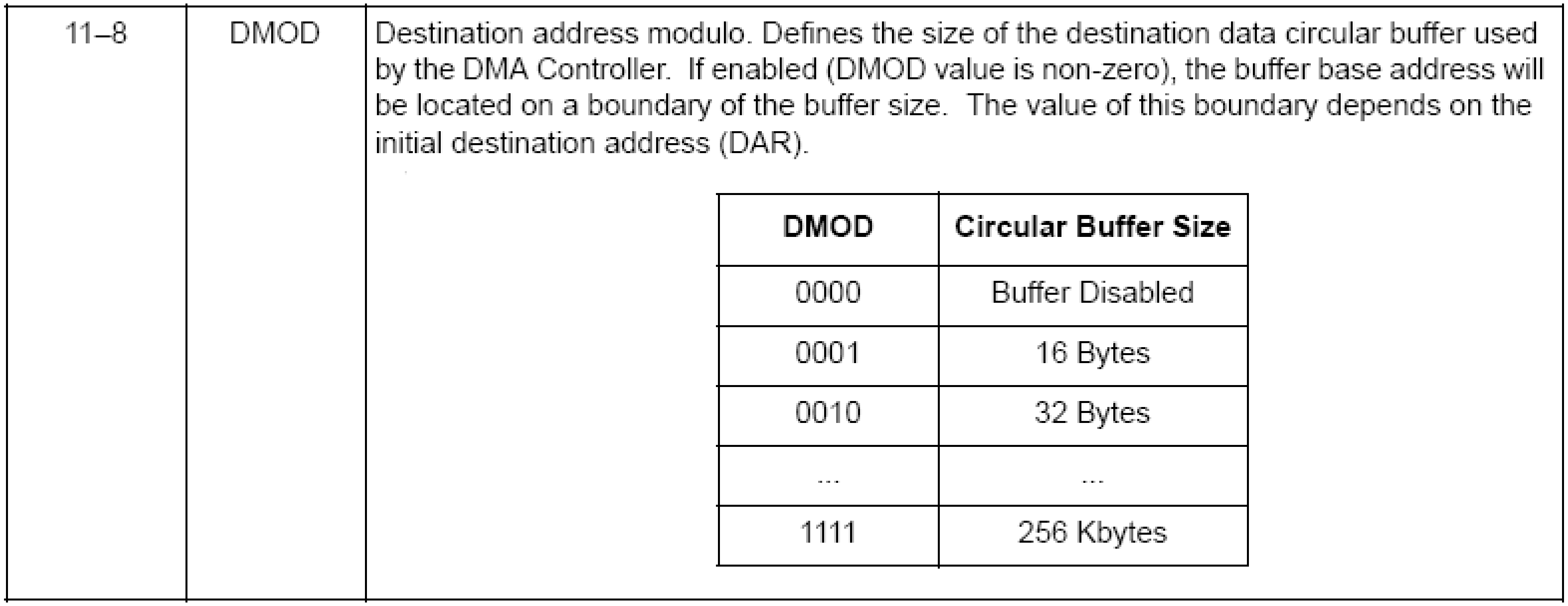
Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

DMA Channel Control Register, DCRn



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT	EEXT	CS	AA	BWC			0	0	SINC	SSIZE		DINC	DSIZE		0
W																START
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

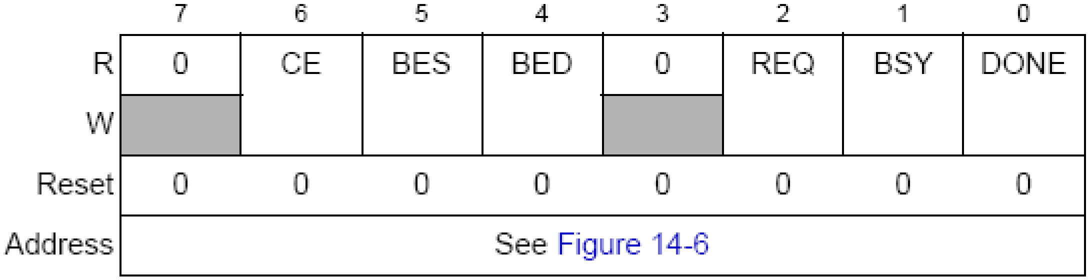
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SMOD				DMOD				D_REQ	0	LINKCC		LCH1		LCH2	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x00_010C (DMA0); IPSBAR + 0x011C (DMA1); IPSBAR + 0x012C (DMA2); IPSBAR + 0x013C (DMA3)															



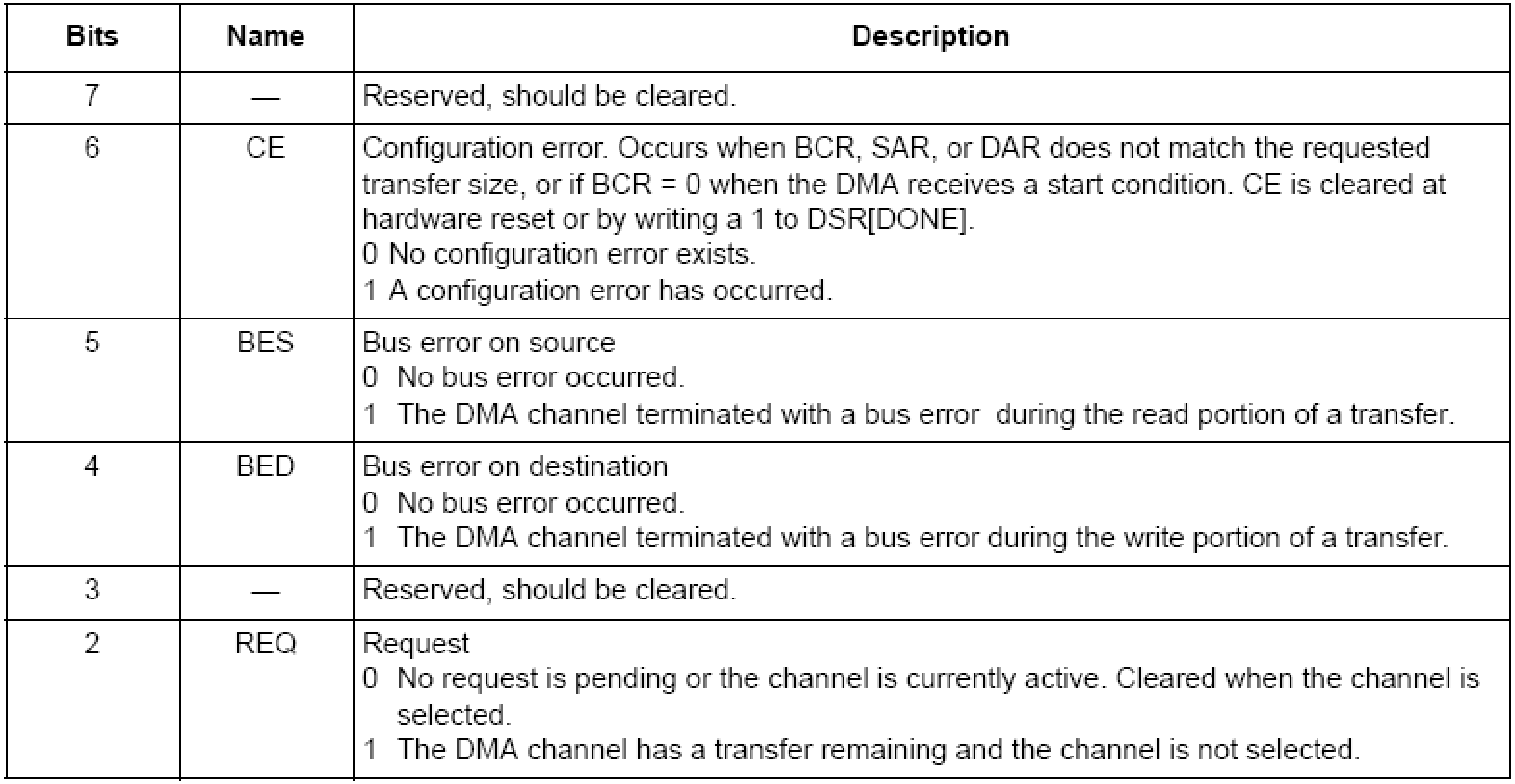
11–8	DMOD	Destination address modulo. Defines the size of the destination data circular buffer used by the DMA Controller. If enabled (DMOD value is non-zero), the buffer base address will be located on a boundary of the buffer size. The value of this boundary depends on the initial destination address (DAR).

DMOD	Circular Buffer Size
0000	Buffer Disabled
0001	16 Bytes
0010	32 Bytes
...	...
1111	256 Kbytes

DMA Channel Status Registers, DSRn

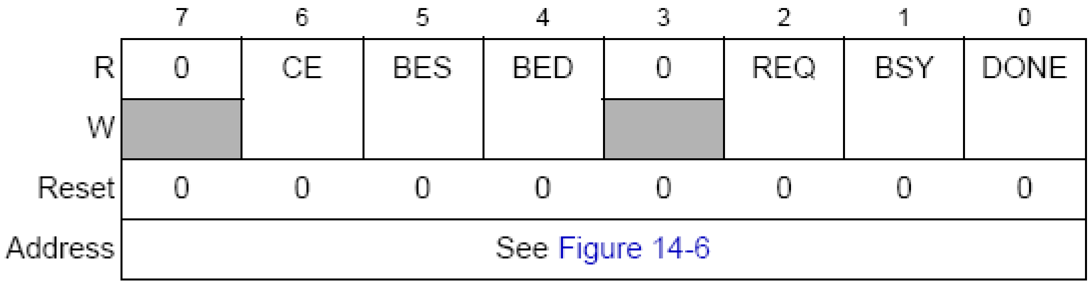


	7	6	5	4	3	2	1	0
R	0	CE	BES	BED	0	REQ	BSY	DONE
W	▨				▨			
Reset	0	0	0	0	0	0	0	0
Address	See Figure 14-6							



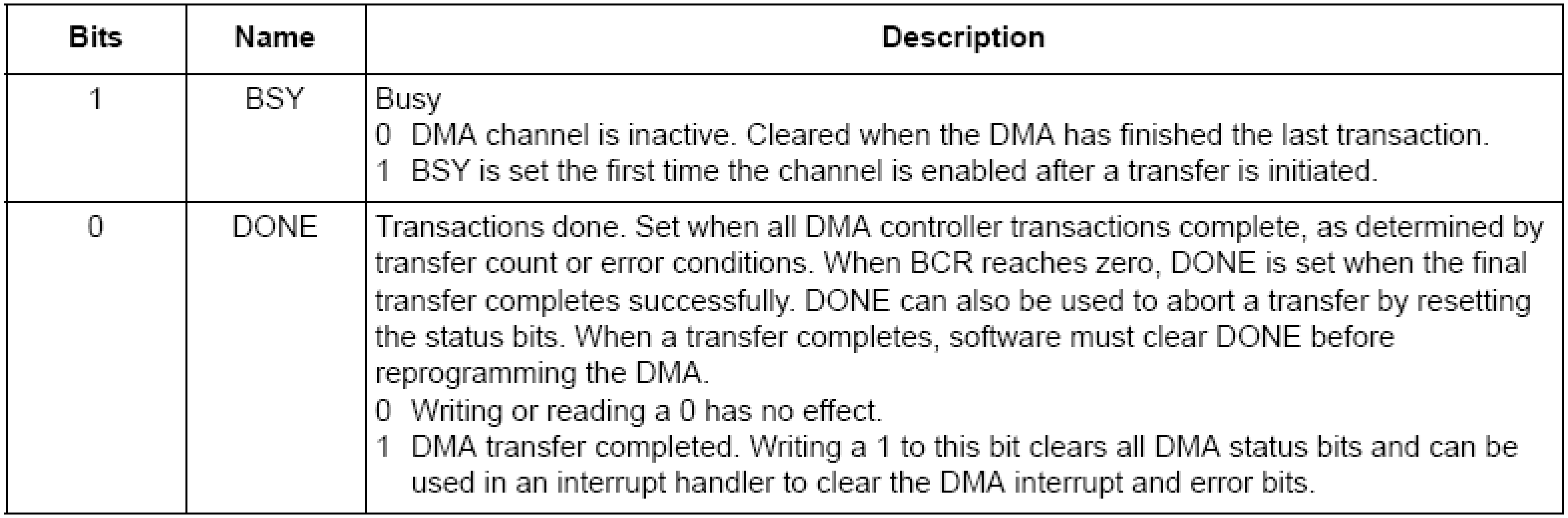
Bits	Name	Description
7	—	Reserved, should be cleared.
6	CE	Configuration error. Occurs when BCR, SAR, or DAR does not match the requested transfer size, or if BCR = 0 when the DMA receives a start condition. CE is cleared at hardware reset or by writing a 1 to DSR[DONE]. 0 No configuration error exists. 1 A configuration error has occurred.
5	BES	Bus error on source 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the read portion of a transfer.
4	BED	Bus error on destination 0 No bus error occurred. 1 The DMA channel terminated with a bus error during the write portion of a transfer.
3	—	Reserved, should be cleared.
2	REQ	Request 0 No request is pending or the channel is currently active. Cleared when the channel is selected. 1 The DMA channel has a transfer remaining and the channel is not selected.

DMA Channel Status Registers, DSRn



	7	6	5	4	3	2	1	0
R	0	CE	BES	BED	0	REQ	BSY	DONE
W	▨				▨			
Reset	0	0	0	0	0	0	0	0
Address	See Figure 14-6							

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.



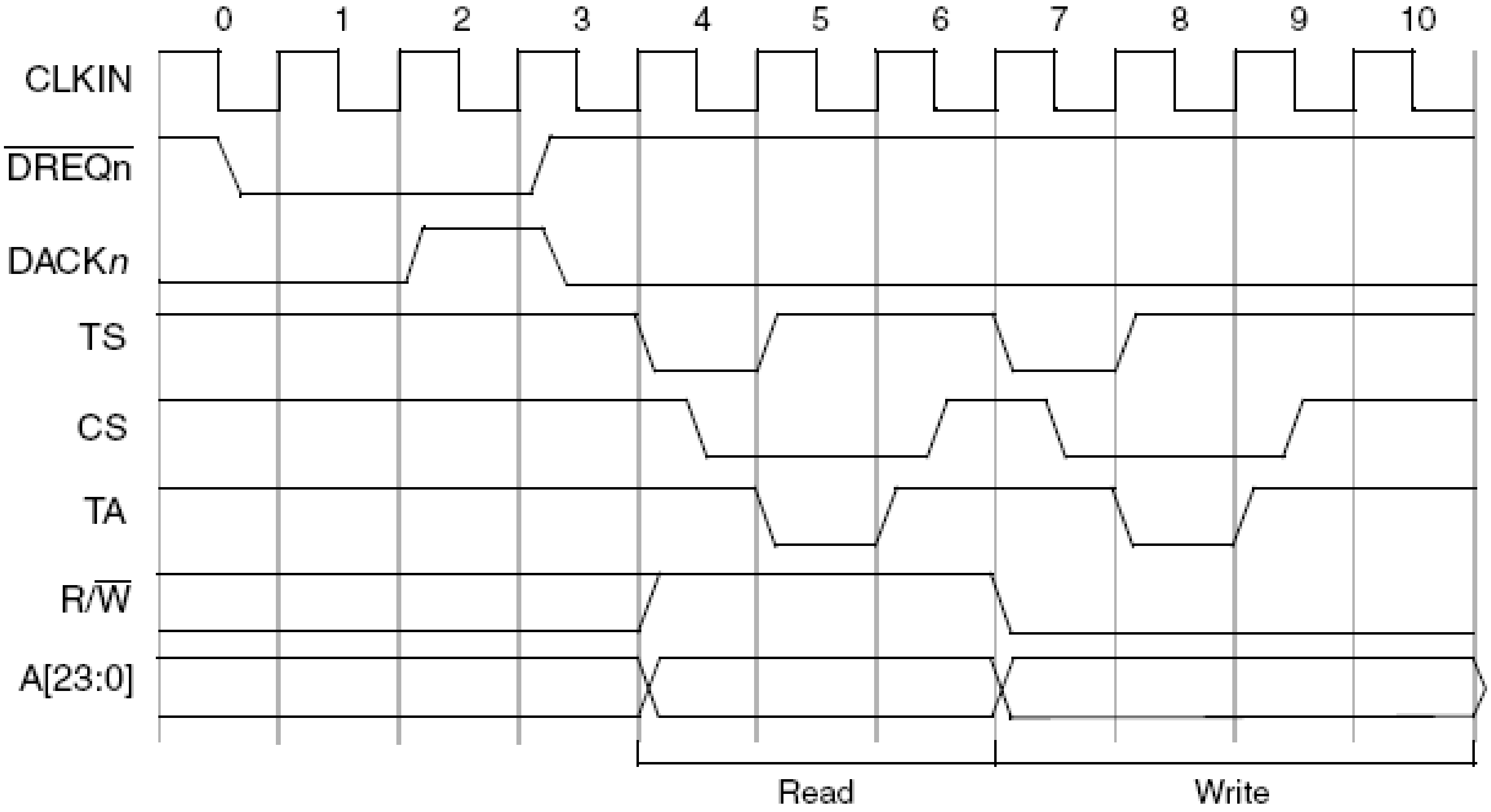
Bits	Name	Description
1	BSY	Busy 0 DMA channel is inactive. Cleared when the DMA has finished the last transaction. 1 BSY is set the first time the channel is enabled after a transfer is initiated.
0	DONE	Transactions done. Set when all DMA controller transactions complete, as determined by transfer count or error conditions. When BCR reaches zero, DONE is set when the final transfer completes successfully. DONE can also be used to abort a transfer by resetting the status bits. When a transfer completes, software must clear DONE before reprogramming the DMA. 0 Writing or reading a 0 has no effect. 1 DMA transfer completed. Writing a 1 to this bit clears all DMA status bits and can be used in an interrupt handler to clear the DMA interrupt and error bits.

Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

Cycle-Stealing Mode vs. Continuous Mode

- DMA transfers must share the system bus with the CPU, and the sharing mechanism must be fast to preserve the advantages of fast DMA data transfer
- Two different bus sharing modes are present in the MCF5234 microcontroller:
- ***Cycle-Stealing Mode*** (DCRn[CS] = 1): Only one complete transfer occurs from the source to destination for each request. If an external DREQn is held asserted (low), then a continuous burst of data transfers will occur.
- ***Continuous Mode*** (DCRn[CS] = 0): Once a request is made, bytes are transferred continuously until either: (1) the BCRn reaches zero or a multiple of DCRn[BWC]; or (2) DSRn[DONE] is written to 1 by the CPU to terminate the DMA.
- The DMA channels are prioritized, with DMA0 having the highest priority and DMA3 having the lowest priority.

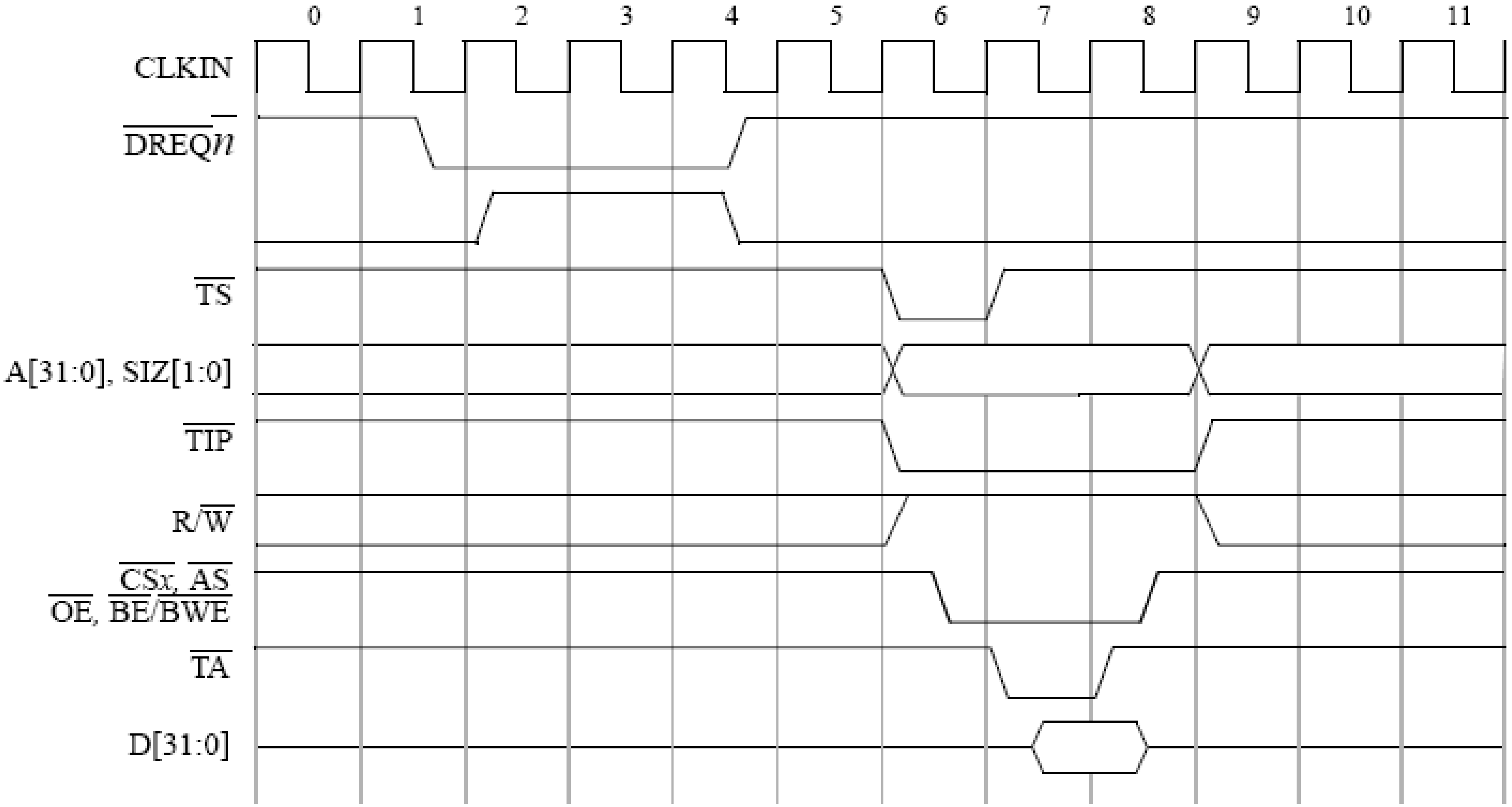
Dual-Address DMA Bus Waveforms



Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

Note: Data is flowing here from one memory buffer to another.

Single-Address DMA Bus Waveforms



Copyright of Freescale Semiconductor, Inc. 2008. Used by permission.

Note: Data is flowing here from a memory buffer to a peripheral.