Serial Interfacing Using RS-232C, Ethernet, and SPI

Reference:

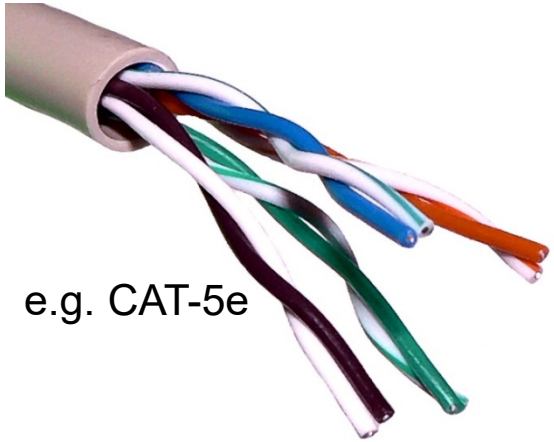
- Freescale Semiconductor, Inc., “MCF5235 Reference Manual”, Doc. No. MCF5235RM, Rev. 2, July 2006.

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.

Digital Signaling Modes

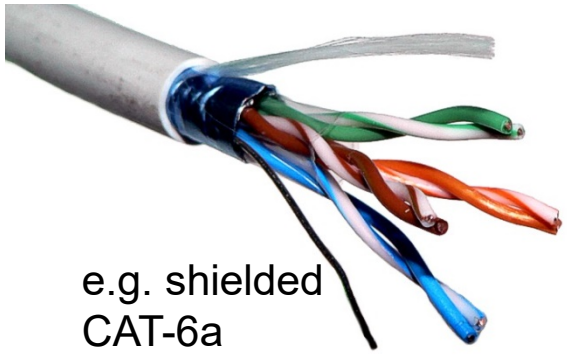
- ***Unbalanced* or *Single-ended* Mode**
  - One wire carries the signal voltage that is defined with respect to a ground (0 volt) reference potential on a second wire or the shield.
  - *Advantages*: One ground wire can be shared for multiple signals.
  - *Disadvantages*: Noise that is added to either the signal or the ground reference (e.g., ground loop noise) can cause bit errors at the receiver.
- ***Balanced* or *Differential* Mode**
  - Two wires carry the signal V^+ (e.g., 0010) and the complemented signal V^- (e.g., 1101). A reference or ground wire is not required.
  - The receiver subtracts the two signals (e.g., $V^+ - V^-$), to recover the single-ended digital data waveform (roughly $2 \times V^+$).
  - If V+ and V- are corrupted by common mode additive noise, then the noise cancels out: $(V^+ + V^{noise}) - (V^- + V^{noise}) = V^+ - V^- = 2\,V^+$
  - Typical sources of additive noise: nearby signals, electric & magnetic fields produced by power supplies, motors, transformers, etc.
  - The noise rejection is even more effective if the wires are twisted.

Some Common Twisted Pair Cable Types



e.g. CAT-5e

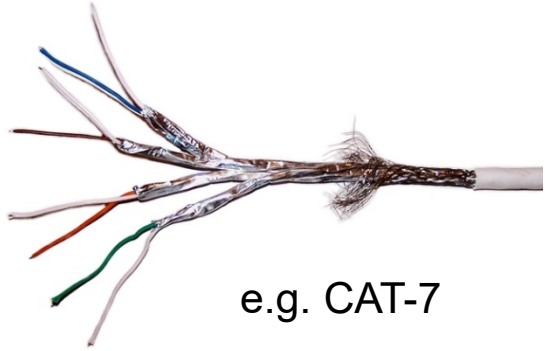
Unshielded Twisted Pair (UTP)



e.g. shielded CAT-6a

Foiled/Shielded Twisted Pair (F/UTP)

- Cable has foil shield.
- TPs are unshielded.



e.g. CAT-7

Fully Shielded Twisted Pair (S/FTP)

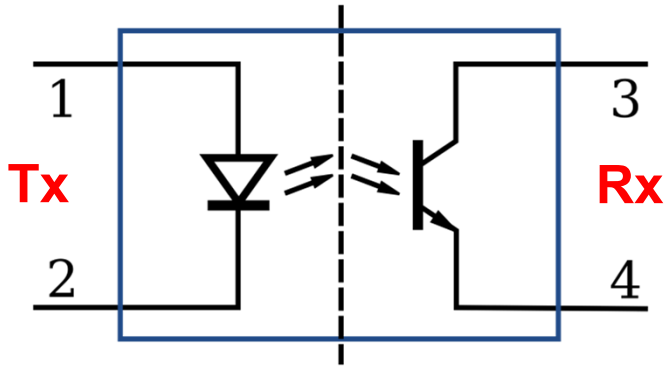
- Cable is shielded.
- Foil for each TP.

Note: The effectiveness of the noise rejection is increased if the twist rate per unit distance is different for each twisted pair (TP).

Two Common Isolation Techniques

Opto-isolator

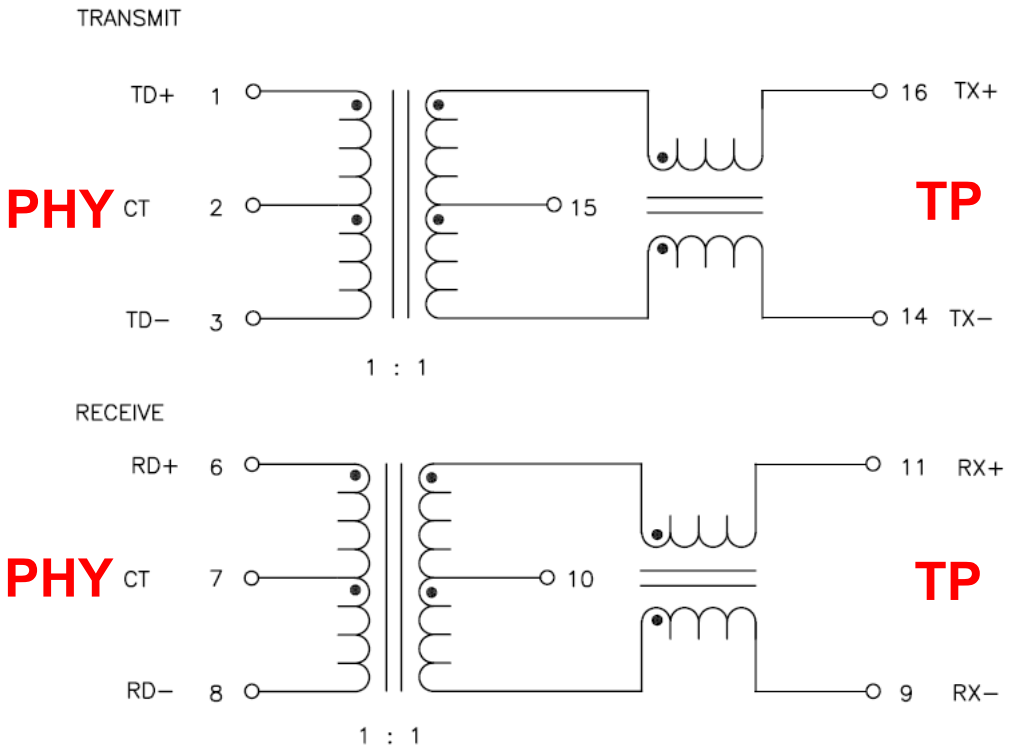
- Protects receiver from high voltage signals and noise
- Ground potentials can differ
- Relatively slow
- Audio applications
- Biomedical sensors



Courtesy Wikimedia.org

Pulse Isolation Transformer

- Blocks DC offsets in signals
- Widely used for LAN signals
- E.g., 10/100Base-TX Ethernet



From Pulse Engineering H1102FNL Datasheet

The RS-232C Serial Transmission Standard

- RS-232C was intended to be a ***low-cost asynchronous serial interface standard*** for connecting large central computers and remote terminal equipment to modems for the purposes of setting up data communication links over the public switched telephone network (PSTN).
- A minimal RS-232C connection requires only ***three wires***: (1) transmit data, (2) receive data, and a (3) ground return.
- RS-232C assumes that the data will be sent as ***7-bit or 8-bit character codes***, plus other "***overhead bits***" that provide timing, minimum character spacing, and (optionally) single-bit error detection using parity bits.
- The data signals are ***single-ended, unmodulated signals***.

A Brief History of RS-232C

- Recommended Standard 232 Version C (RS-232C) was published by the Electronic Industry Association (EIA) in the U.S.A. in 1969 for use in modem connections. It has since become a ***de facto*** standard for connecting computer peripheral equipment.

- The EIA published a slightly revised specification, called “EIA 232D”, in 1987.

- The EIA, together with the Telecommunications Industry Association (TIA), produced yet another updated standard “EIA/TIA-232-E” in 1991.

- Most engineers continue to use the term “RS-232”.

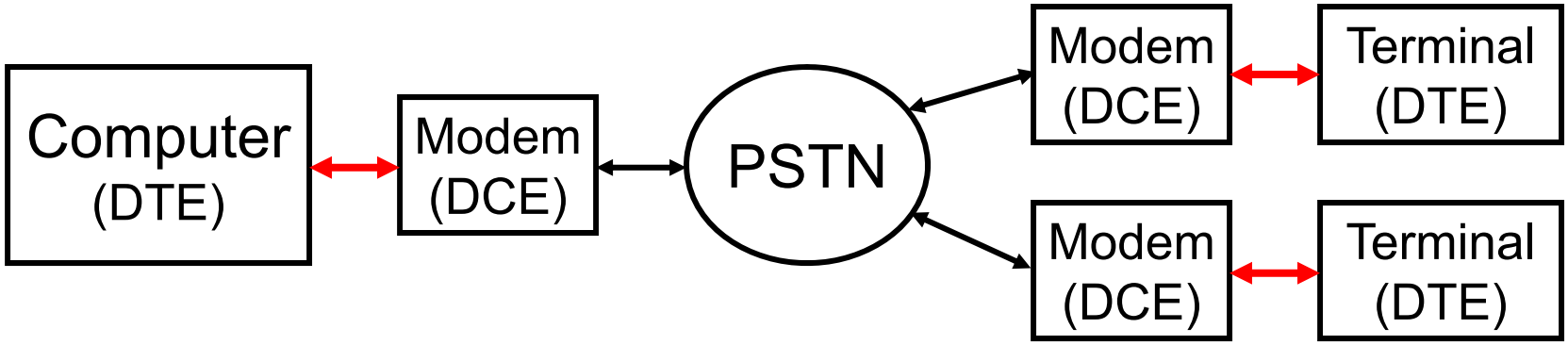
Two Successor Standards to RS-232C

- The official RS-232C specification has serious limitations:
  - maximum transmission distance is only 15 m (50 ft.)
  - maximum bit/baud rate is 20,000 bits per second. (up to 200 Kbps is possible, but outside of the spec.)

- RS-422 is a balanced (differential mode) serial transmission standard with higher maximum bit rates.
  - transmits +/- 6 V down to +/- 2 V differential outputs
  - input signals can be attenuated down to +/- 0.2 V
  - differential signals reject common mode additive noise
  - transmits up to 90 Kbps over up to 1.2 Km (3,900 ft)

- RS-485 is a balanced (differential mode) serial transmission standard with higher maximum bit rates. E.g. 2 Mb/s @ 50 m
  - Supports multi-drop busses using tri-stateable drivers

Intended RS-232C Connections



Data Terminal Equipment (DTE)

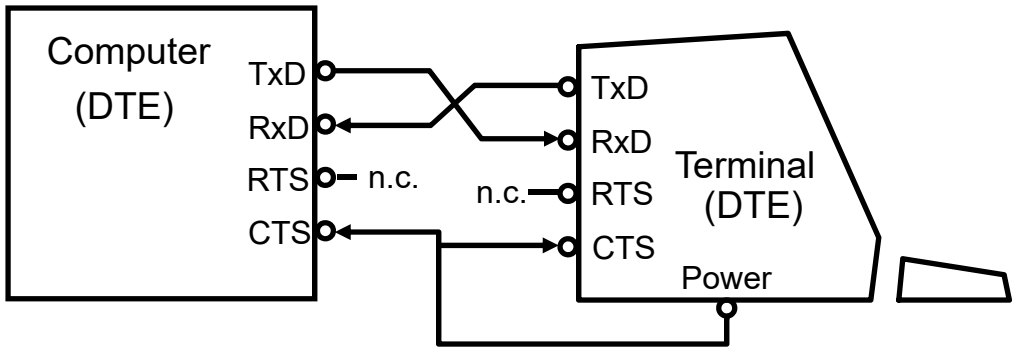
- Computers, terminals

Data Communications Equipment (DCE)

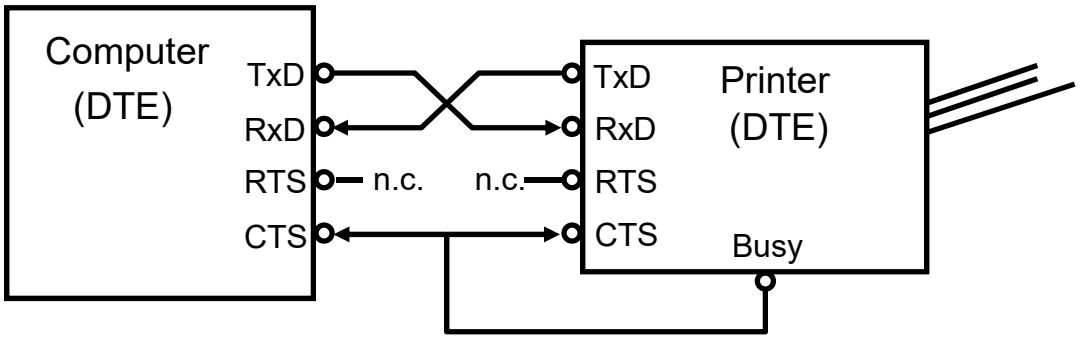
- Modem for interfacing RS-232C binary signals to analog signals that lie within the standard 100 Hz to 3400 Hz telephone passband.

Typical RS-232C Connections

Connection from a computer to a terminal



Connection from a computer to a printer



The Scope of RS-232C

Mechanical Aspects

- 25-pin D-type connectors (a pre-existing standard)
- DB-25-P plug (male) for Data Terminal Equipment (DTE).
- DB-25-S socket (female) for Data Communications Equipment (DCE).

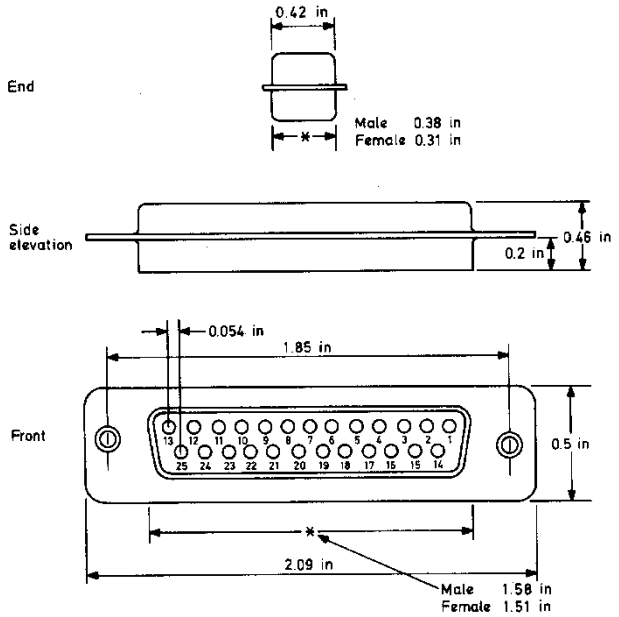
Electrical Aspects

- “Space” (“0”) signal is a voltage in the range +3 V to +15 V.
- “Mark” (“1”) signal is a voltage in the range -3 V to -15 V.
- An idle line is in the “mark” state.
- A sustained “space” voltage is called a “break” signal.

Data Frame Format

- Start bit & 7 or 8 data bits & optional parity bit & stop bit(s).
- No higher-level in-band protocol is included in the standard.

Mechanical Aspects of RS-232C



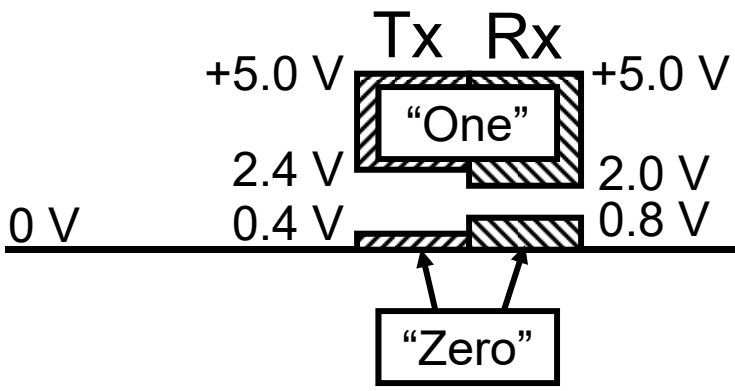
- A 25-pin DB-25 connector (from Cannon Co., 1952) was specified in an appendix to the original RS-232C standard.
- Most modern serial standards use a much smaller connector. For a time, the smaller 9-pin DE-9 connector replaced the DB-25 for RS-232C connections.
- Ethernet, USB & WiFi have mostly replaced RS-232C.

RS-232C Signal Definitions

1. ***Protective ground (shield)***
2. ***Transmitted data (TxD)***
3. ***Received data (RxD)***
4. Request to send (RTS)
5. Clear to send (CTS)
6. Data set ready (DSR)
7. ***Signal ground (GND)***
8. Received signal detected
9. reserved for testing
10. reserved for testing
11. unassigned
12. Secondary signal detected
13. Secondary CTS
14. Secondary TxD
15. Tx Signal Timing for DCE
16. Secondary RxD
17. Rx Signal Element Timing
18. unassigned
19. Secondary RTS
20. Data terminal ready (DTR)
21. Signal quality detector
22. Ringing indicator (RI)
23. Data signal rate selector
24. Tx Signal Timing for DTE
25. unassigned

Logic Levels for TTL (left) and RS-232C (right)

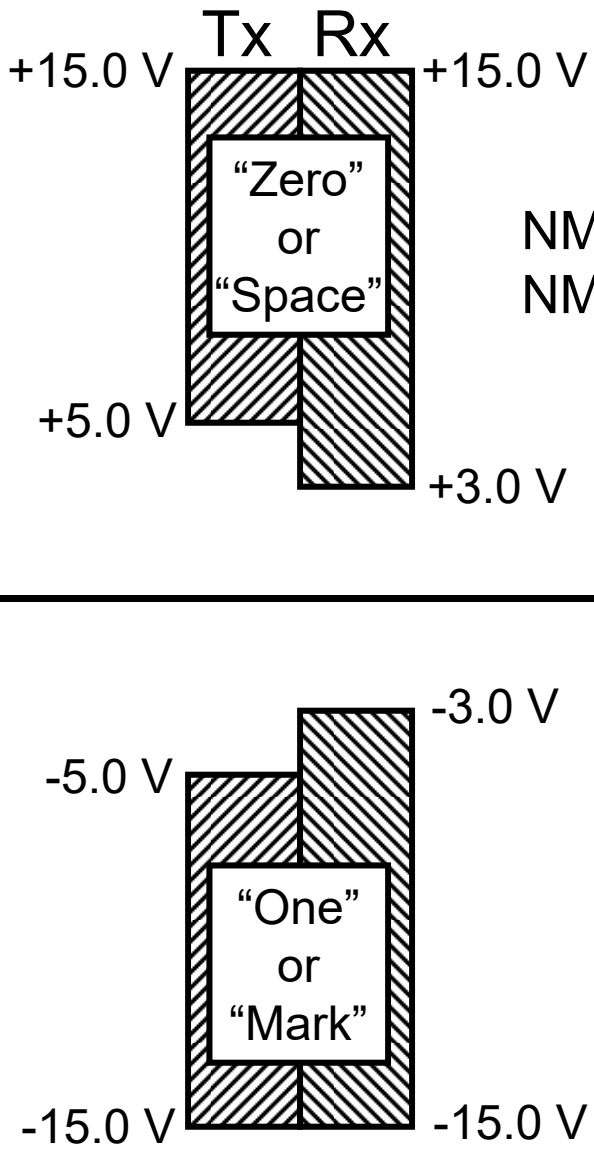
TTL = Transistor-Transistor Logic



Noise Margins in TTL:

$NM_H = 2.4 - 2.0 = 0.4$ V

$NM_L = 0.8 - 0.4 = 0.4$ V



$NM_H = 2.0$ V

$NM_L = 2.0$ V

0 V

More on Logic Signal Ranges

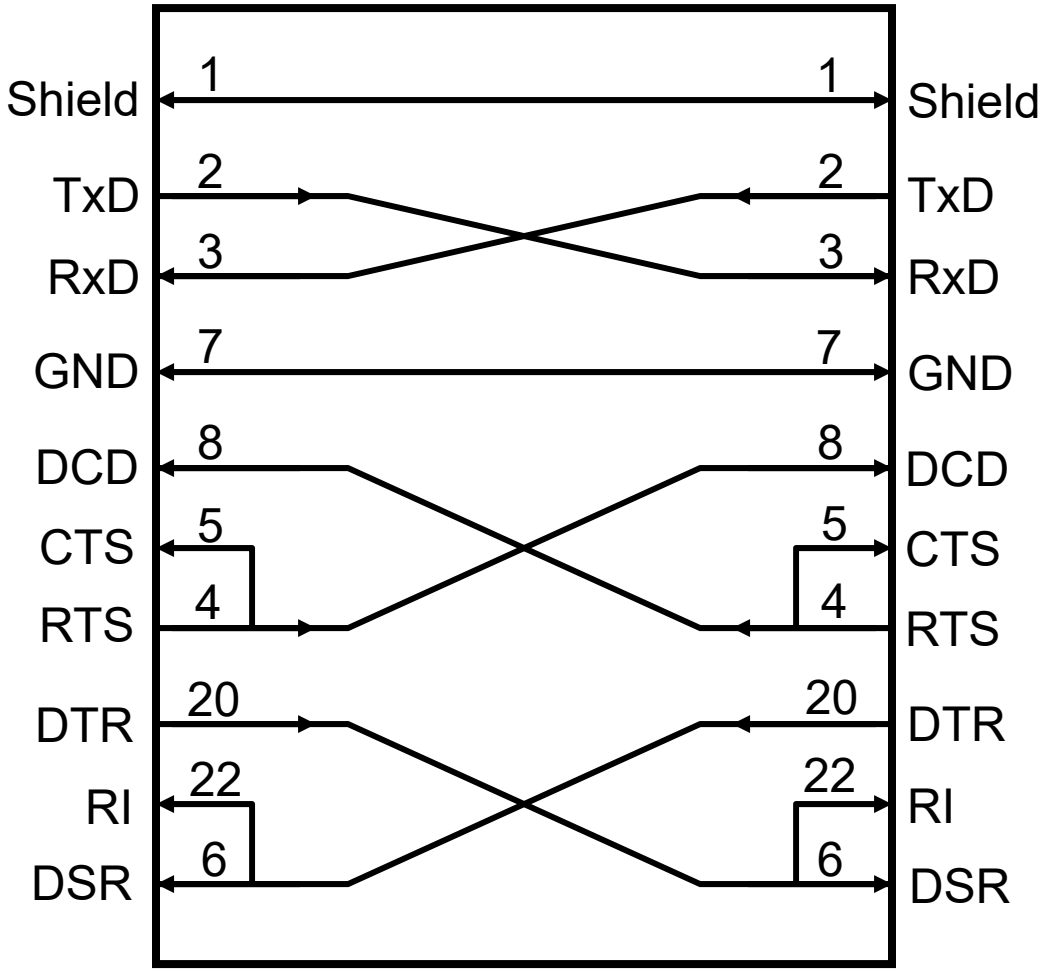
- A digital transmitter/driver circuit must produce “high” and “low” voltage signals. Acceptable output “high” and “low” signals lie in two different analog ranges, which are separated by a range of undefined voltages. For example, in TTL a valid **output “high”** signal is a voltage V_{OH} in the range 2.4V to 5.0 V, and a valid **output “low”** signal is a voltage V_{OL} in the range 0V to 0.4V. Voltages between 2.4V and 0.4V occur when the signal is changing between high and low; these midrange voltages have no meaning.

- A digital receiver/input circuit must detect “high” and “low” signals from received analog voltages. For example, a valid **input “high”** signal is a voltage V_{IH} in the range 2.0V to 5.0V, and a valid **input “low”** signal is a voltage V_{IL} in the range 0V to 0.8V.

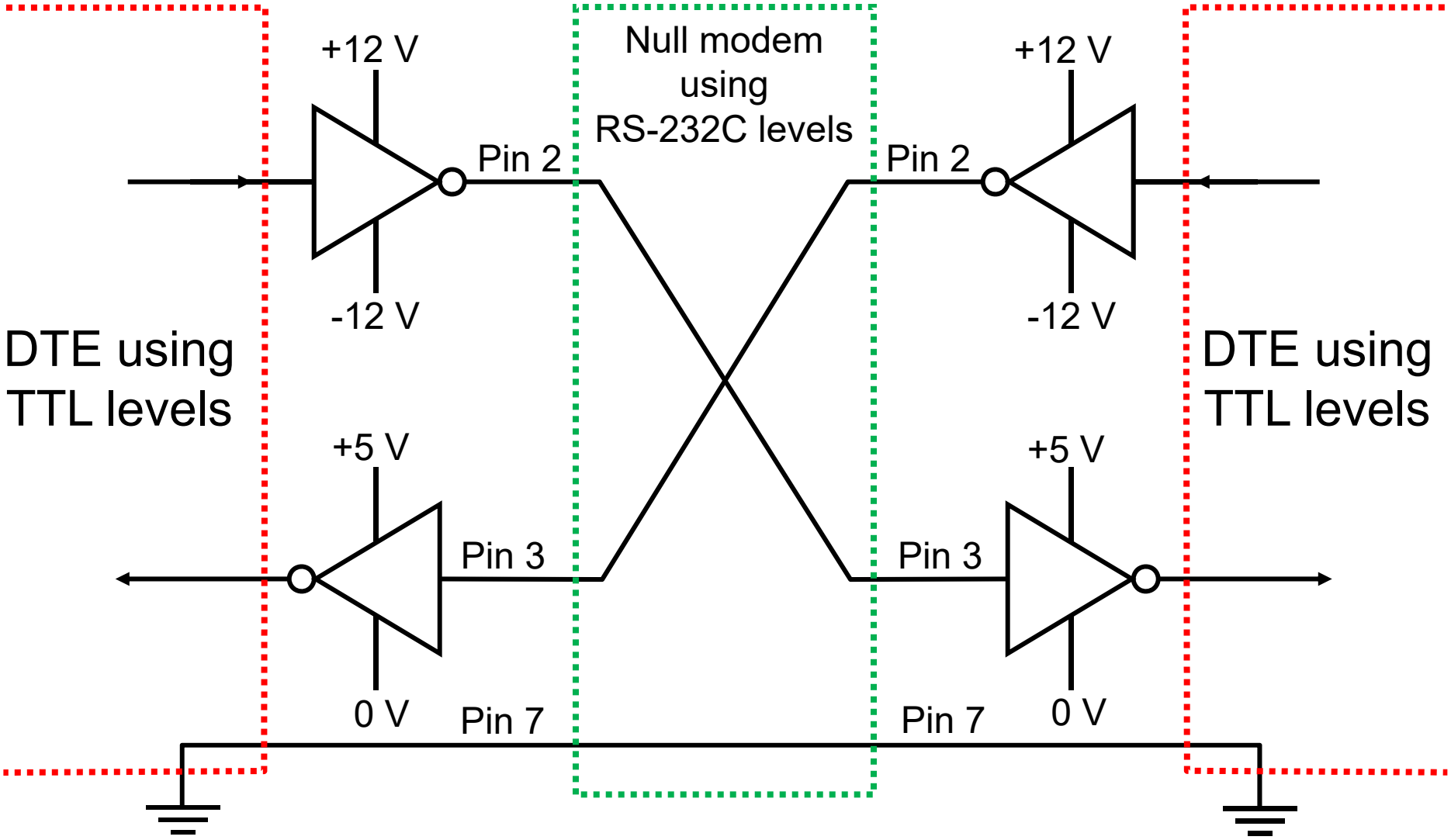
Noise Margins

- During transmission between a digital transmitter/driver and a digital receiver/input, analog impairments can degrade the quality of the signal. Noise voltages (or currents) can be picked up by the signal by electromagnetic coupling. Offsets can also get added to the signal for various reasons, such as differences in the power supply voltages.
- The ***high noise margin*** NM_H is the difference between the lowest driven output high voltage V_{OH} and the lowest acceptable input high voltage V_{IH}. The ***low noise margin*** NM_L is the difference between the highest acceptable input low voltage V_{IL} and the highest driven output low voltage V_{OL}. The noise margins ensure that the digital signals are detected correctly at the receiver despite the presence of analog impairments (noise + offset) that are smaller in amplitude than the noise margins. Cheap noise immunity!

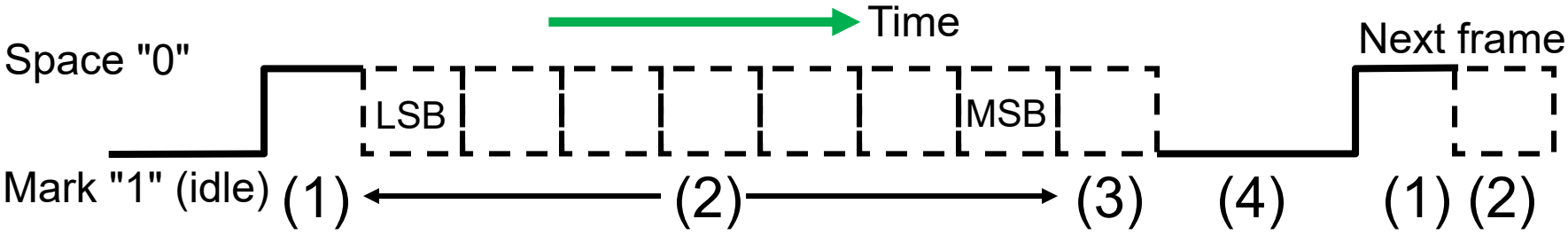
“Null Modem” for DTE-DTE Connections



TTL to RS-232C to TTL Duplex Connection



RS-232C Frame Format



(1) ***Start bit*** for indicating the starting time (mark level to space level transition) of a new character "frame".

(2) Seven or eight ***data bits***. These are usually used to encode 7-bit ASCII or ISO characters, or 8-bit EBCDIC characters. Binary data can also be transmitted.

(3) Optional even or odd ***parity bit*** for detecting bit errors.

(4) ***Stop bit(s)*** spacer time after a character. Does not have to be a whole number of bit times long.

Hardware Flow Control Signals in RS-232C

Several parallel control signals are provided in RS-232C to allow out-of-band (separate from the data communication path) signaling between the DTC and DCE. For example:

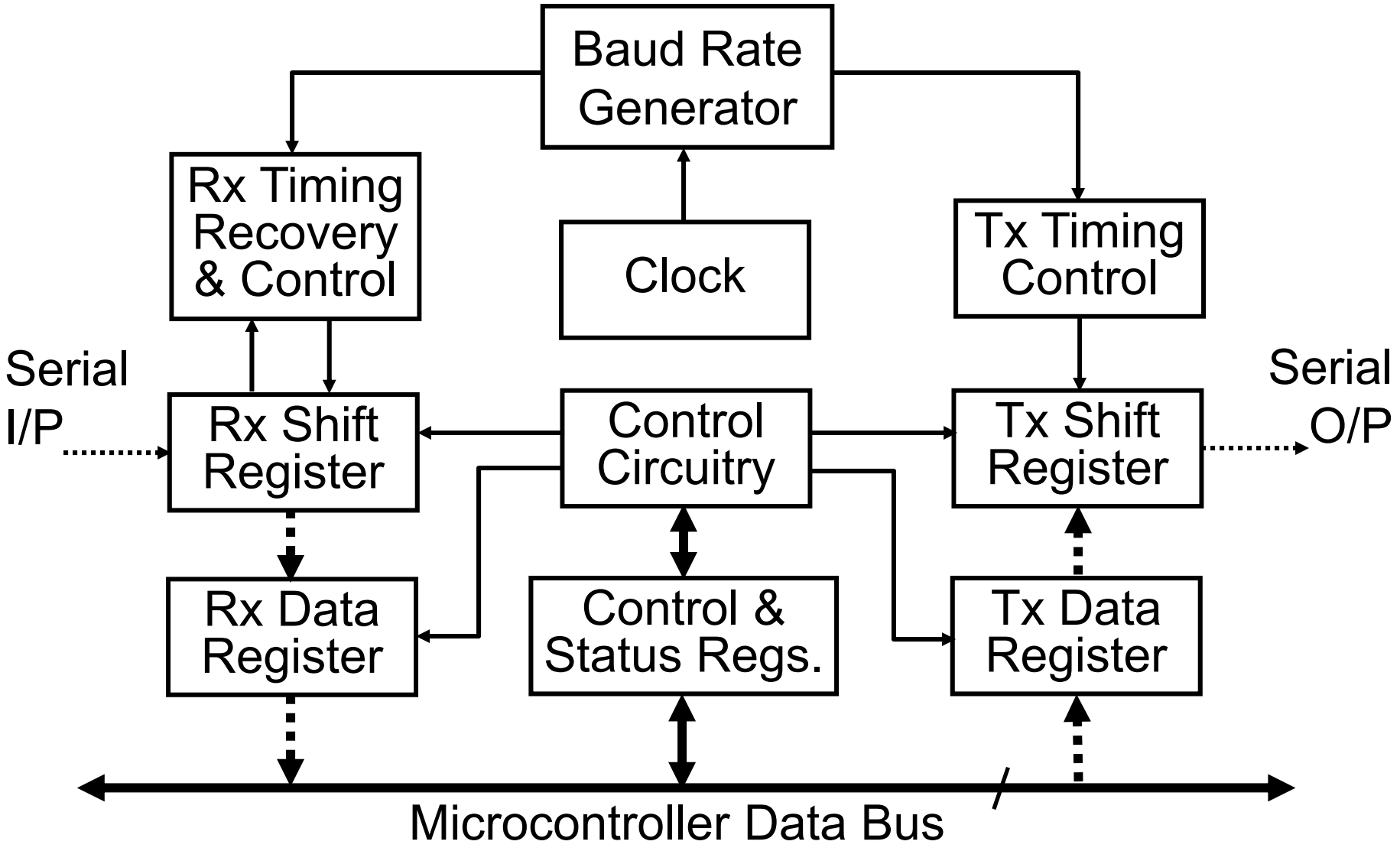
Request to Send (RTS) is asserted to "mark" by the DTE to tell the DCE that the DTE wishes to transmit data.

Clear to Send (CTS) is asserted to "mark" by the DCE to tell the DTE that the DCE is ready to receive data.

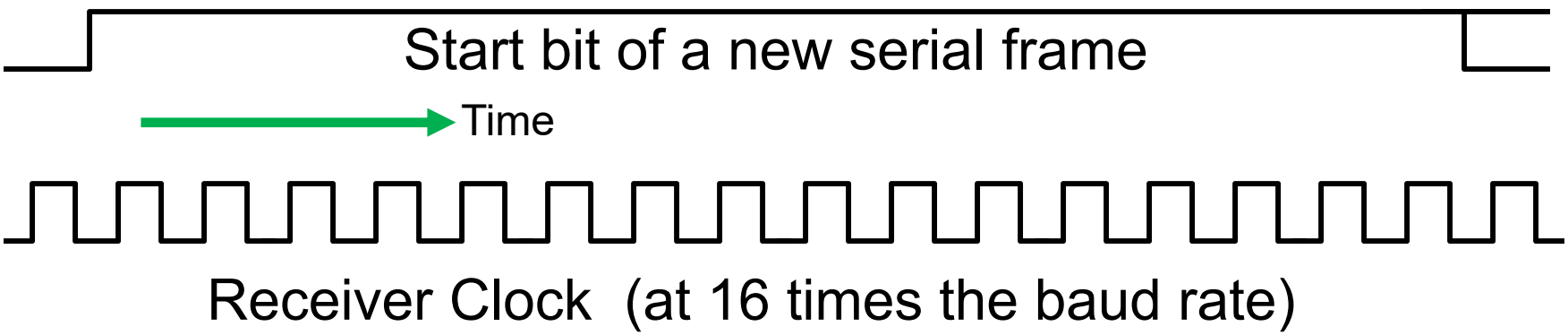
Data Set Ready (DSR) is asserted to "mark" by the DCE to tell the DTE that it is switched on and is not in a test mode.

Data Terminal Ready (DTR) is asserted to "mark" by the DTE as long as it wishes to keep the connection active.

Architecture of a Generic Serial Interface



Timing Recovery in an Asynchronous Serial Receiver



Procedure for finding the bit/baud center times:

1) Look out for the leading edge of the start bit.
2) Count off 8 clock periods of the x16 receiver clock.
3) Now count 16 clock periods to get to the first data bit.
4) The remaining bits (data, parity, stop) should lie at further multiples of 16 receiver clock periods.

Peripheral Interface Registers

- Computer interface hardware is accessed through registers, which are *write-only*, *read-only*, or *readable and writeable* locations in the memory address space.
- ***Control registers*** are writeable locations that allow the CPU to select interface operational modes and options.
- ***Status registers*** are readable locations that allow the CPU to determine interface status information, such as the presence of newly received data, active interrupt conditions, error conditions, etc.
- ***Data registers*** are used as ports to convey data between the CPU and the interface.

Data Transfer Control Methods

- There are four basic methods in which input / output data is transferred within a microcomputer between locations in main memory and I/O interfaces:
  1) ***Unconditional data transfer*** by CPU writes to a data register, or CPU reads from a data register.
  2) ***Busy-waiting***, i.e., the CPU must wait for some status condition to go true before writing or reading (also called “conditional CPU-controlled data transfer”).
  3) ***Interrupt-driven data transfer***.
  4) ***Direct memory access (DMA)***.
- All four methods are ultimately under the control of the CPU. However, in the DMA method, the CPU temporarily assigns some control for the data transfer to a “DMA controller”.

Motorola/Freescale Interfaces that Support Asynchronous Serial Communications

MC6850 - *Asynchronous Communications Interface Adaptor* (ACIA)

- Introduced in the mid 1970s to support the 6800 8-bit μP
- Provides one bi-directional serial port

MC68681 - *Dual Universal Receiver Transmitter* (DUART) chip

- Introduced in the mid 1980s to support the M68000 μP family
- Provides two bi-directional serial ports

68HC05 - *Serial Communications Interface* (SCI) subsystem

- Introduced in the early 1980s as part of the 68HC05 8-bit μC

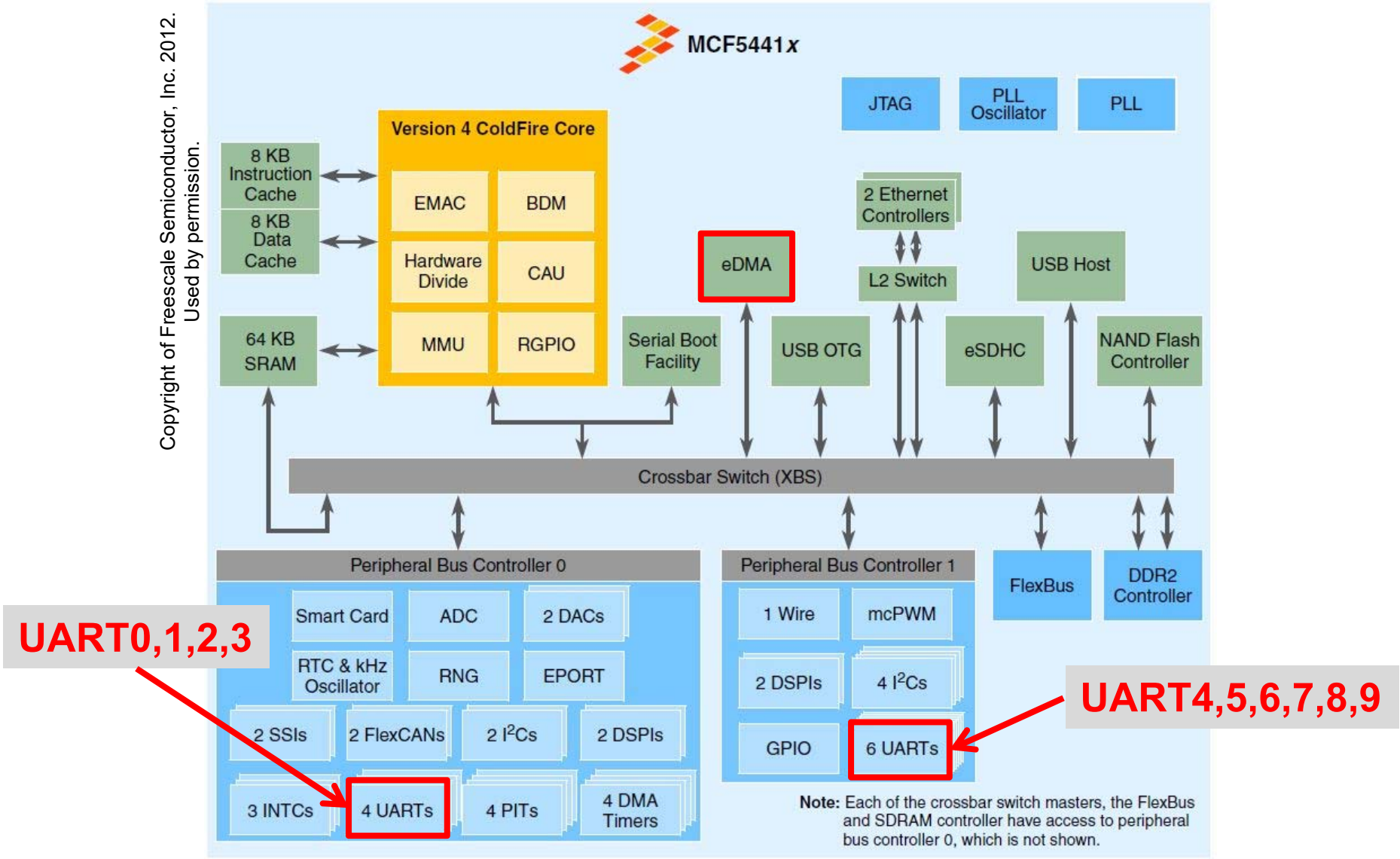
68HC11 - *Serial Communications Interface* (SCI) subsystem

- Introduced in the mid 1980s as part of the 68HC11 8-bit μC

683xx - *Serial Communications Interface* (SCI) subsystem

- Introduced in the late 1980s as part of the 683xx 32-bit μCs

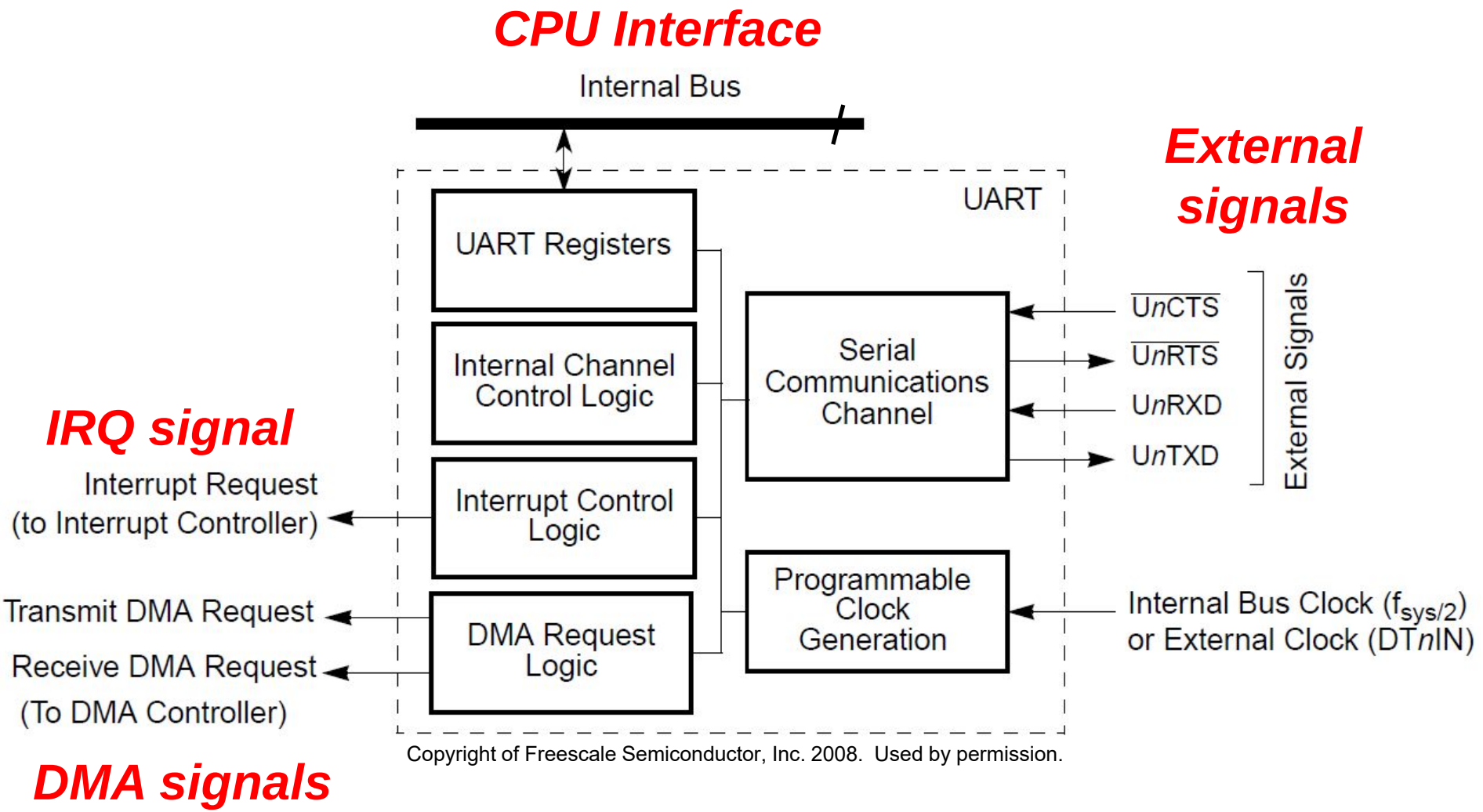
The Ten UART Interfaces in the MCF54415



Universal Asynchronous Receiver-Transmitter

- The MCF54415 contains ten ***Universal Asynchronous Receiver-Transmitter*** (UART) interfaces. Ten is a rather generous number!
- Each UART provides an independent, fully-programmable, ***full-duplex serial interface*** for the CPU.
- ***Clock generation*** can be programmed to derive different baud rates using either the MCF54415 clock or (only for UARTs #0-3) externally provided clocks.
- A wide ***variety of data formats*** is supported: 5, 6, 7 or 8-bit data; odd, even, no parity, or force constant parity; 1.0, 1.5 or 2.0 stop bits.
- ***Four interrupting conditions*** can be programmed for each UART.
- Serial communications can use ***polling***, ***interrupts*** or ***DMA***.
- ***Three error conditions*** are detected by each receiver: (1) parity error; (2) framing error; and (3) overrun error.
- Receiver direction data is ***quadruple-buffered***, and the transmitter direction data is ***double-buffered***.

Simplified UART Architecture Diagram, *n* = 0, ..., 9



External UART Signals

Signal	Description
Transmitter Serial Data Output (U*n*TXD)	U*n*TXD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on U*n*TXD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
Receiver Serial Data Input (U*n*RXD)	Data received on U*n*RXD is sampled on the rising edge of the clock source, with the lsb received first.
Clear-to- Send (U*n*CTS)	This input can generate an interrupt on a change of state.
Request-to-Send ($\overline{\text{U}n\text{RTS}}$)	This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter's $\overline{\text{U}n\text{CTS}}$, $\overline{\text{U}n\text{RTS}}$ can control serial data flow.

Note: The four DT*n*IN pins (*n* = 0, 1, 2, 3) can also be used as an external clock input to drive the UART clock generation circuit in place of the $f_{sys/2}$ signal, which is the MCF54415 internal system bus clock. DT0IN is shared by UARTs 0, 4 & 8. DT1IN is shared by UARTs 1, 5, 9. DT2IN is shared by UARTs 2 & 6. DT3IN is shared by UARTs 3 & 7.

Different UART*n* Pin Connections, *n* = 0, ..., 9

n	UART*n*TXD	UART*n*RXD	UART*n*RTS	UART*n*CTS	External Clk
0	D11	B10	B11	E13	H15 (DT0IN)
1	D9	C9	D10	C10	H13 (DT1IN)
2	N2	P1	M3	M4	H14 (DT2IN)
3	K1	L3	*N/A*	*N/A*	G13 (DT3IN)
4	E13	B11	*N/A*	*N/A*	H15 (DT0IN)
5	C10	D10	*N/A*	*N/A*	H13 (DT1IN)
6	M4	M3	*N/A*	*N/A*	H14 (DT2IN)
7	E15	A13	*N/A*	*N/A*	G13 (DT3IN)
8	G15	G14	*N/A*	*N/A*	H15 (DT0IN)
9	D14	D15	*N/A*	*N/A*	H13 (DT1IN)

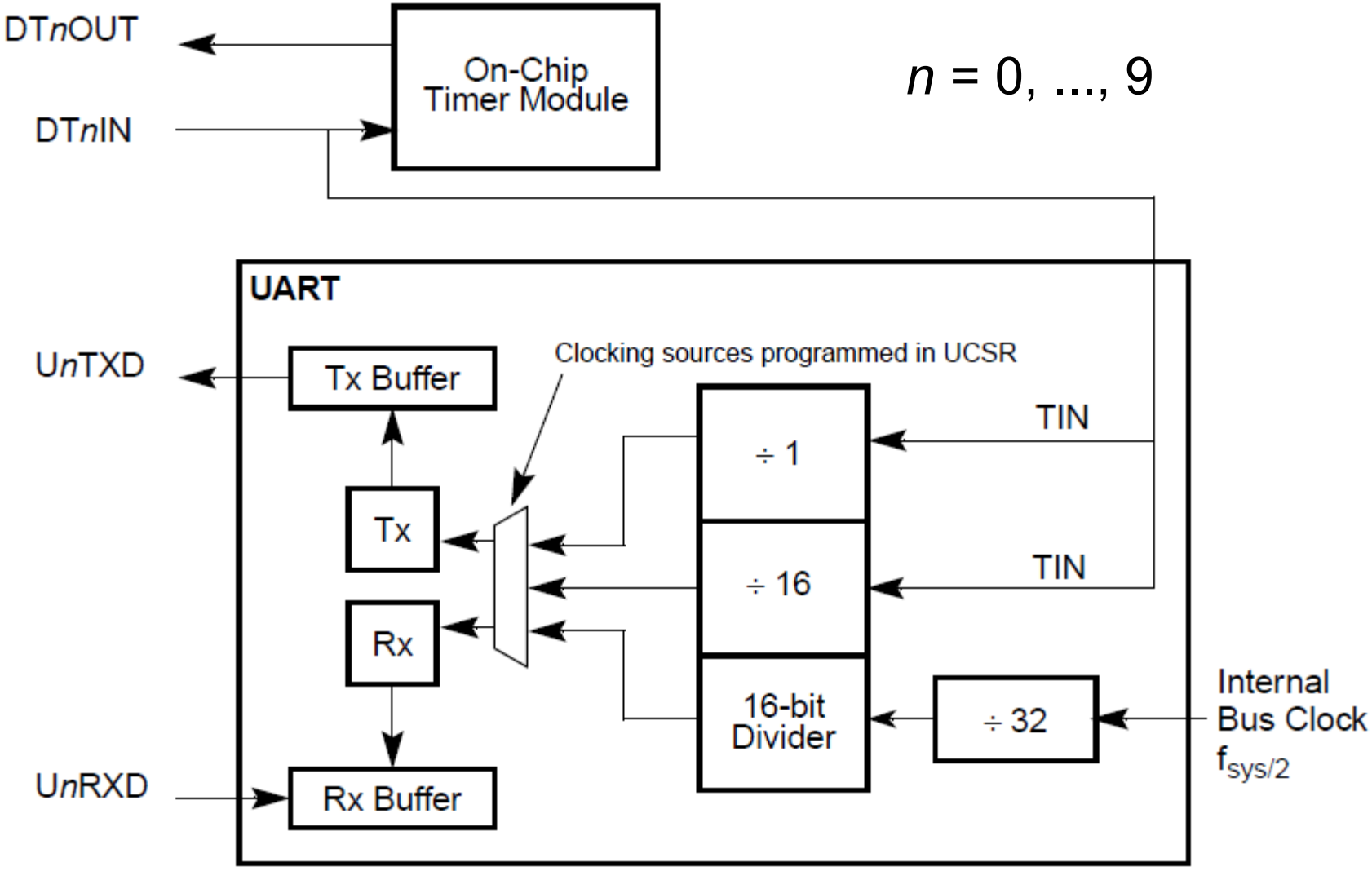
Note: The Request-To-Send (RTS) and Clear-To-Send (CTS) handshake lines are only provided for UART0, UART1 & UART2.

MCF54415 Pinout for the 256-MAPBGA Package

	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	BOOT MOD1	EVDD	EVDD	EVDD	EVDD	EVDD	EVDD	VSS	TRST	TDO	RESET	TMS	K
L	DSPI0_PCS1	DSPI0_SCK	DSPI0_SIN	VSS	BOOT MOD0	EVDD	VSS	VSS	VSS	VSS	VSS	VSS	TDI	DDATA0	DDATA3	RST_OUT	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	

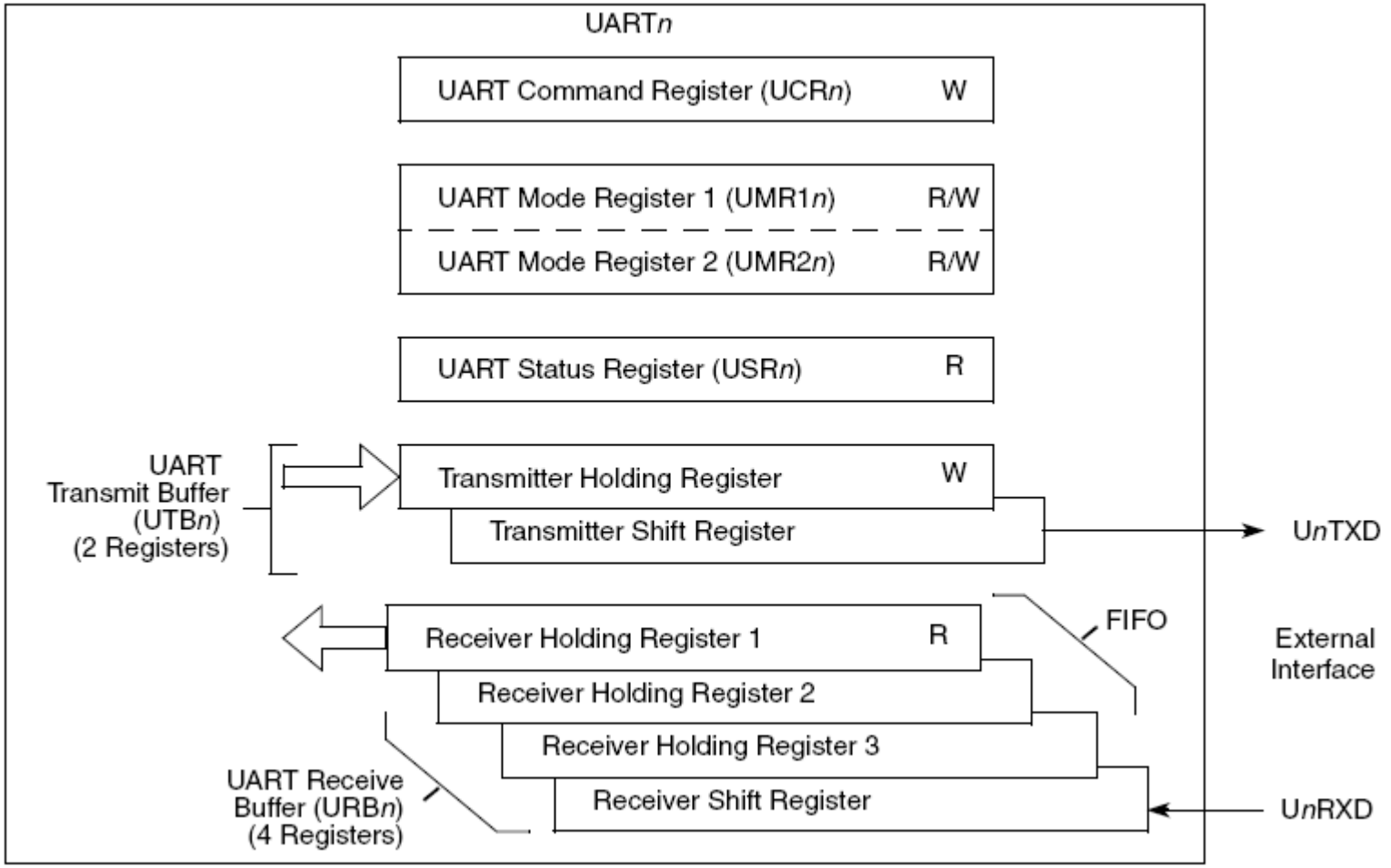
Figure 8. MCF54415, MCF54416, MCF54417, and MCF54418 Pinout (256 MAPBGA)

UART*n* Clock Generation Sub-Block



UART Transmitter/Receiver Registers

***n* = 0, ..., 9**

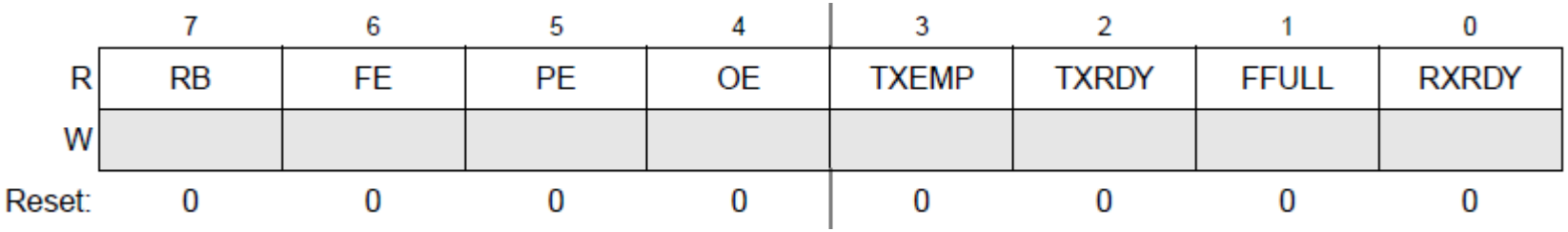


UART*n* Status Register, *n* = 0, ..., 9

The CPU-accessible registers of the ten UARTs are mapped into 16-Kbyte regions in the CPU's memory as follows:

	Base Address
UART0	0xFC06_0000
UART1	0xFC06_4000
UART2	0xFC06_8000
UART3	0xFC06_C000
UART4	0xEC06_0000
UART5	0xEC06_4000
UART6	0xEC06_8000
UART7	0xEC06_C000
UART8	0xEC07_0000
UART9	0xEC07_4000

UART*n* Status Register, *n* = 0, ..., 9



	7	6	5	4	3	2	1	0
R	RB	FE	PE	OE	TXEMP	TXRDY	FFULL	RXRDY
W								
Reset:	0	0	0	0	0	0	0	0

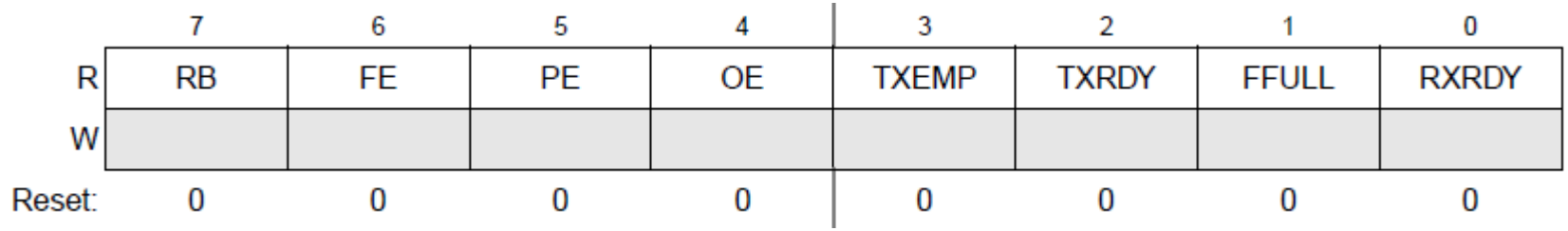
RB = 1 (0) : a ***break*** signal has been (has not been) *received.*

FE = 1 (0) : a ***framing error***: a stop bit was not received (was received) at the expected time for the last received character.

PE = 1 (0) : a ***parity error*** occurred (did not occur) for the last received character. Note: the parity can be disabled, even, odd, or fixed.

OE = 1 (0) : an ***overrun error*** has occurred (not occurred). In an overrun error, the CPU or DMA controller was too slow to read data, and so ≥ 1 received characters were lost.

UART*n* Status Register, *n* = 0, ..., 9



	7	6	5	4	3	2	1	0
R	RB	FE	PE	OE	TXEMP	TXRDY	FFULL	RXRDY
W								
Reset:	0	0	0	0	0	0	0	0

TXEMP = 1 (0) : the *transmitter data buffers* are all (are not all) ***empty***. The TXEMP =1 condition is also called “underrun”.

TXRDY = 1 (0) : the *transmitter holding register* has room (does not have room) and is ***ready*** (is not ready) for another character to be loaded by the CPU or DMA controller.

FFULL = 1 (0) : the *receiver **FIFO*** buffer is (is not) ***full*** and cannot accept (can accept) a new character. When FFULL = 1, any more received characters will be lost.

RXRDY = 1 (0) : the *receiver FIFO* contains at least one new character (contains no new characters) ***ready*** to be read.

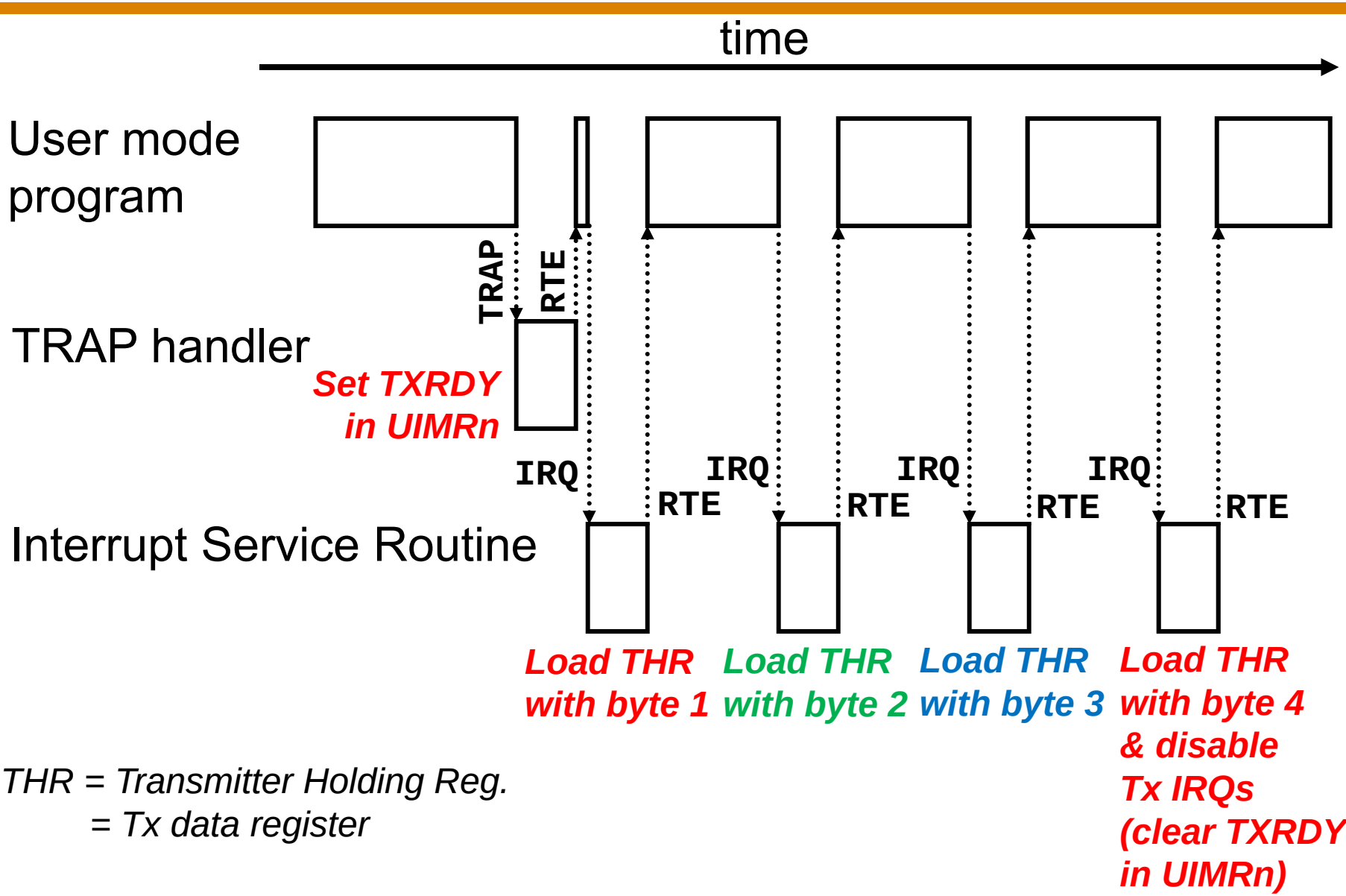
UART Interrupt Status and Mask Registers

	7	6	5	4	3	2	1	0
R (UISR*n*)	COS	0	0	0	0	DB	FFULL/ RXRDY	TXRDY
W (UIMR*n*)	COS	0	0	0	0	DB	FFULL/ RXRDY	TXRDY
Reset	0	0	0	0	0	0	0	0
Address	IPSBAR + 0x0214 (UISR0); IPSBAR + 0x0254 (UISR1); IPSBAR + 0x0294 (UISR2)							

COS = “Change of State”
DB = “Delta Break”
FFULL = “FIFO Full”
RXRDY = “Receiver Ready”
TXRDY = “Transmitter Ready”

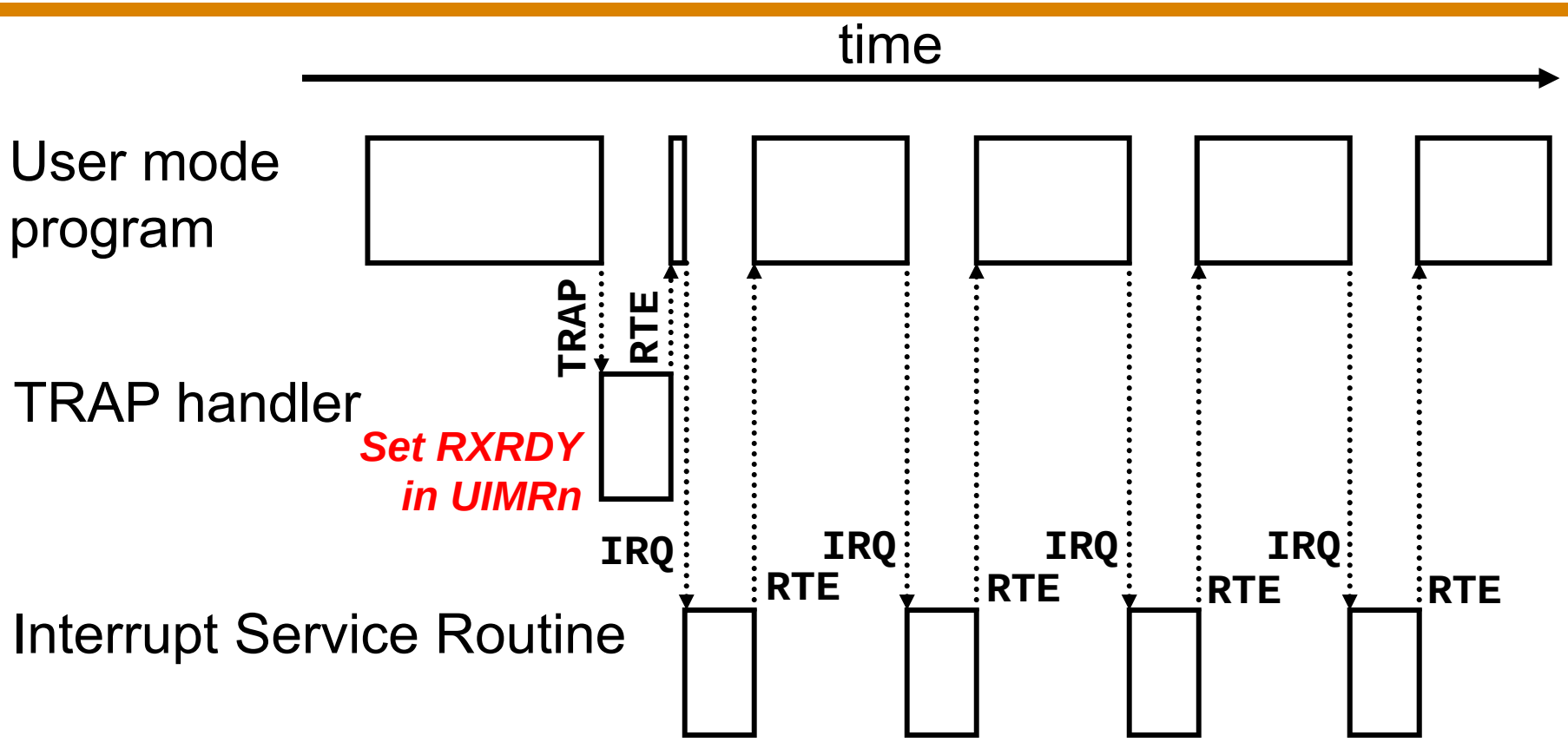
- Each of the ten UARTs has one read-only ***Interrupt Status Register*** (UISR*n*) and one write-only ***Interrupt Mask Register*** (UIMR*n*), which share the same address.
- Only 4 out of the 8 bits are used in each register.
- The 4 bits correspond to potentially interrupt-causing conditions.
- A UART will assert an active interrupt signal only if a status bit and the corresponding mask bit are both 1.
- An interrupt can be stopped by writing the corresponding mask bit to 0.

Interrupt-Driven UART*n* Output for Four Bytes



THR = Transmitter Holding Reg.
= Tx data register

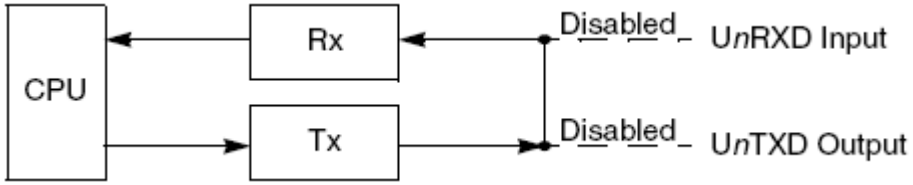
Interrupt-Driven UARTn Input for Four Bytes



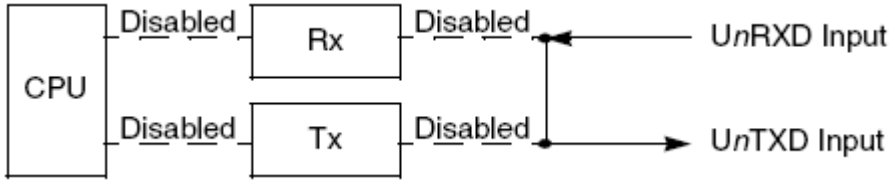
RHR1 = Receiver Holding Reg. #1
= Rx data register

UART Loopback Test Modes

1) *Local Loopback Mode*
   - Allows local UART to verify its Tx and Rx circuits
   - External Tx and Rx connections are not tested



2) *Remote Loopback Mode*
   - Allows remote UART to verify its Tx and Rx circuits
   - External Tx and Rx connections are tested



Simplified UART Initialization Sequence

Register	Setting
UCR*n*	Reset the receiver and transmitter. Reset the mode pointer (MISC[2–0] = 0b001).
UIMR*n*	Enable the preferred interrupt sources.
UACR*n*	Initialize the input enable control (IEC bit).
UCSR*n*	Select the receiver and transmitter clock. Use timer as source if required.
UMR1*n*	If preferred, program operation of receiver ready-to-send (RxRTS bit). Select receiver-ready or FIFO-full notification (RxRDY/FFULL bit). Select character or block error mode (ERR bit). Select parity mode and type (PM and PT bits). Select number of bits per character (B/Cx bits).
UMR2*n*	Select the mode of operation (CMx bits). If preferred, program operation of transmitter ready-to-send (TxRTS). If preferred, program operation of clear-to-send (TxCTS bit). Select stop-bit length (SBx bits).
UCR*n*	Enable transmitter and/or receiver.

Background: Network Topology (1)

- ***Network topology*** is the arrangement of communicating ***nodes*** and the communication ***links*** among those nodes. Typical nodes are computers, servers, switches, and I/O devices (e.g. printers, scanners, cameras).
- A network topology can be modelled as a mathematical ***graph*** $G = \{ V, E \}$, with a finite set V of ***vertices*** (i.e., nodes) and a finite set E of ***edges*** (i.e., links) such that $E \subseteq V \times V$.
- A ***physical network topology*** accurately shows the relative physical positions of the nodes and links. The distances on the topology may be scaled differently for convenience in diagrams, but the relative positions of the nodes should correspond to their physical/geographical positions.
- A ***logical network topology*** shows the data flows through a network. The logical network topology may differ from the underlying physical network topology. For example, multiple parallel physical links in a physical network topology, or a linear chain of connected nodes, may be merged into a single logical link between the two end nodes.

Background: Network Topology (2)

- The communication wire(s) or cable(s) form either (1) a point-to-point link or (2) a shared multidrop bus (often just called a bus).
- For a ***point-to-point link*** there are two communicating nodes, one at either end of the link. If the link only allows communication in one direction at a time it is called a ***simplex*** link. If the link allows simultaneous communication in both directions it is called a ***duplex*** link (e.g., RS-232C).
- A ***shared multidrop bus*** provides connections among two or more nodes that share the bus (e.g. RS-485). Only one node can broadcast on the bus at any one time, creating either a point-to-point link (one receiving node) or a ***point-to-multipoint broadcast*** (two or more receiving nodes).
- When a shared bus is used, a method needs to be provided to deal with the possibility that two nodes may wish to transmit at the same time. E.g.
  - Allow ***collisions*** to occur, but detect them and recover from them.
  - Circulate one "token" on the bus that allows only one node to transmit.
  - Provide a central "arbiter" that selects the one active transmitting node.

Background: Network Topology (3)

- Many different network topologies can be created using only point-to-point links. Here are some of the common resulting topologies:

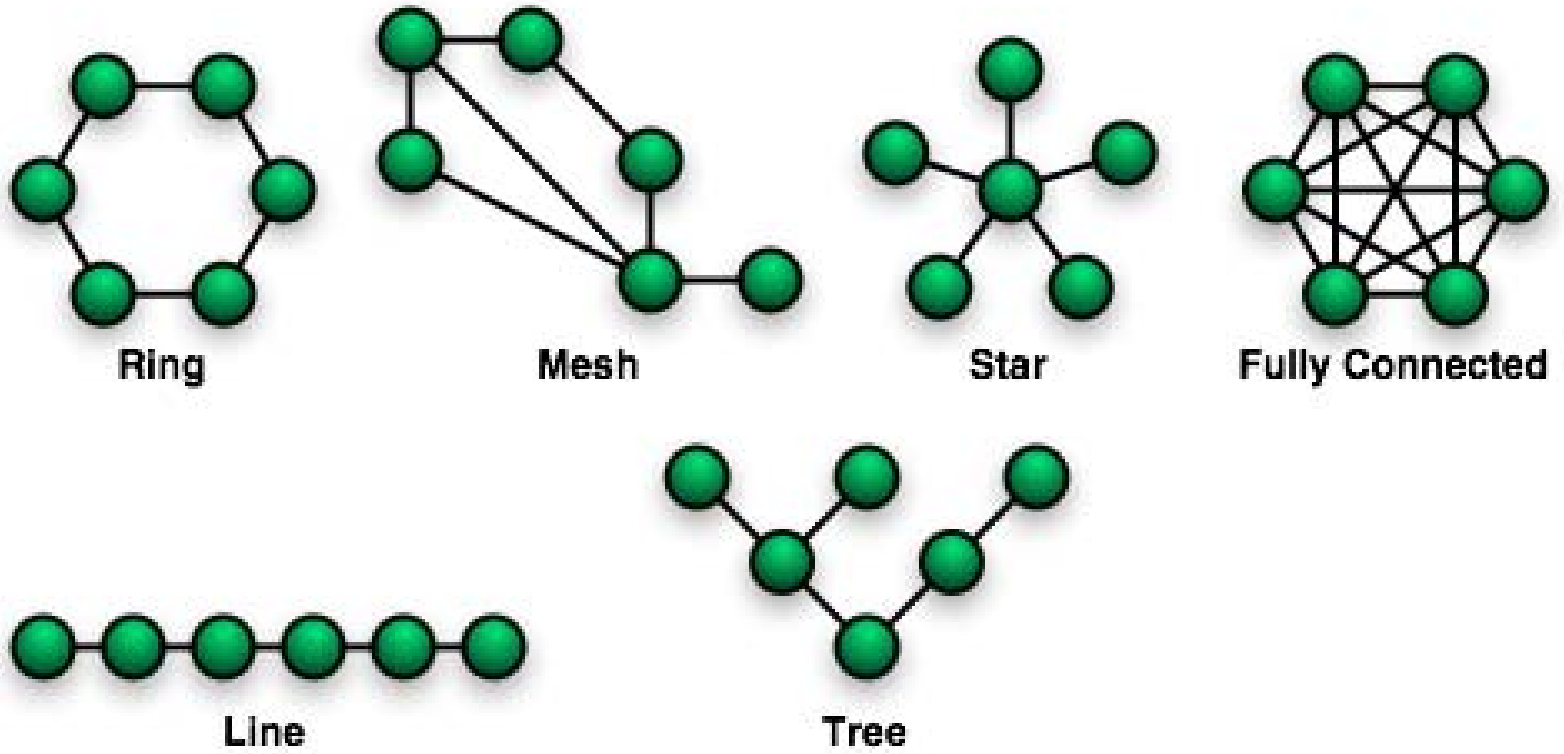


Figure courtesy of Wikimedia Commons

The Ethernet Local Area Network (LAN)

- **Local Area Network** standards were developed to allow computers and other office equipment to communicate within limited areas (e.g., a department) at much higher speeds than the telephone network.
- In 1973, Xerox Corp. released the **Ethernet** LAN standard. This standard was then adopted as IEEE Std 802.3 and is now widely used.
- The original Ethernet standard uses a **shared coaxial cable** as the medium, and has a peak data rate of 10 Mbps. Typical data rates were a few Mbps because of the need for idle time on the cable.
- Only one transmitter station on the cable can transmit at a time.
- A transmitter with data to send will wait until the cable is available before attempting to transmit data onto the cable.
- If two (or more) transmitters happen to start transmitting at the same time, then the resulting "**collision**" is detected by those transmitters.
- After a collision, the active transmitters each wait for a **random back-off time** before trying to transmit again (to hopefully avoid a 2^{nd} collision).

Ethernet Frame Format

The bits in an Ethernet signal are organized into "frames" as follows:

- ***Preamble***: 7 bytes (56 bits) of alternating 0s and 1s for synchronization
- ***Start Frame Delimiter*** (SFD): 1 byte = 0b10101011
- ***Destination Address*** (DA): 6 bytes specifying the destination station

  An example address, given in hex, is: 07-01-02-01-2C-4B
- ***Source Address*** (SA): 6 bytes specifying the source station
- ***Length / Type***: 2 bytes
  - If <1518, then this field specifies the length of the data field
  - If >1536, then this field specifies the upper layer LAN protocol
- ***Data***: Anywhere from 46 to 1500 bytes of data
- ***Cyclic Redundancy Check*** (CRC): 4 bytes computed using CRC-32

  The CRC is computed at the source station as the remainder obtained by dividing the data field (padded with an all-0 field) by a standard 33-bit binary polynomial divisor. At the destination station, the data (padded with the received CRC) is divided by the same divisor. The remainder will be zero if there were no errors in transmission; if nonzero, cause a retransmission.

The CRC-32 Calculation

- ***Polynomial division*** over the finite field GF(2) is easy to implement in hardware. The hardware just requires shift operations and binary XORs.
- For an *n*-bit binary CRC, the ***divisor polynomial*** contains *n*+1 bits. The most significant bit (MSB) of the divisor is always a 1.
- The divisor is shifted past the input bit stream, starting out left justified with the most significant bit (e.g., the first arriving bit).

```
10110110110110111101  <-- Input bitstream
1101                  <-- Divisor for 3-bit CRC
```

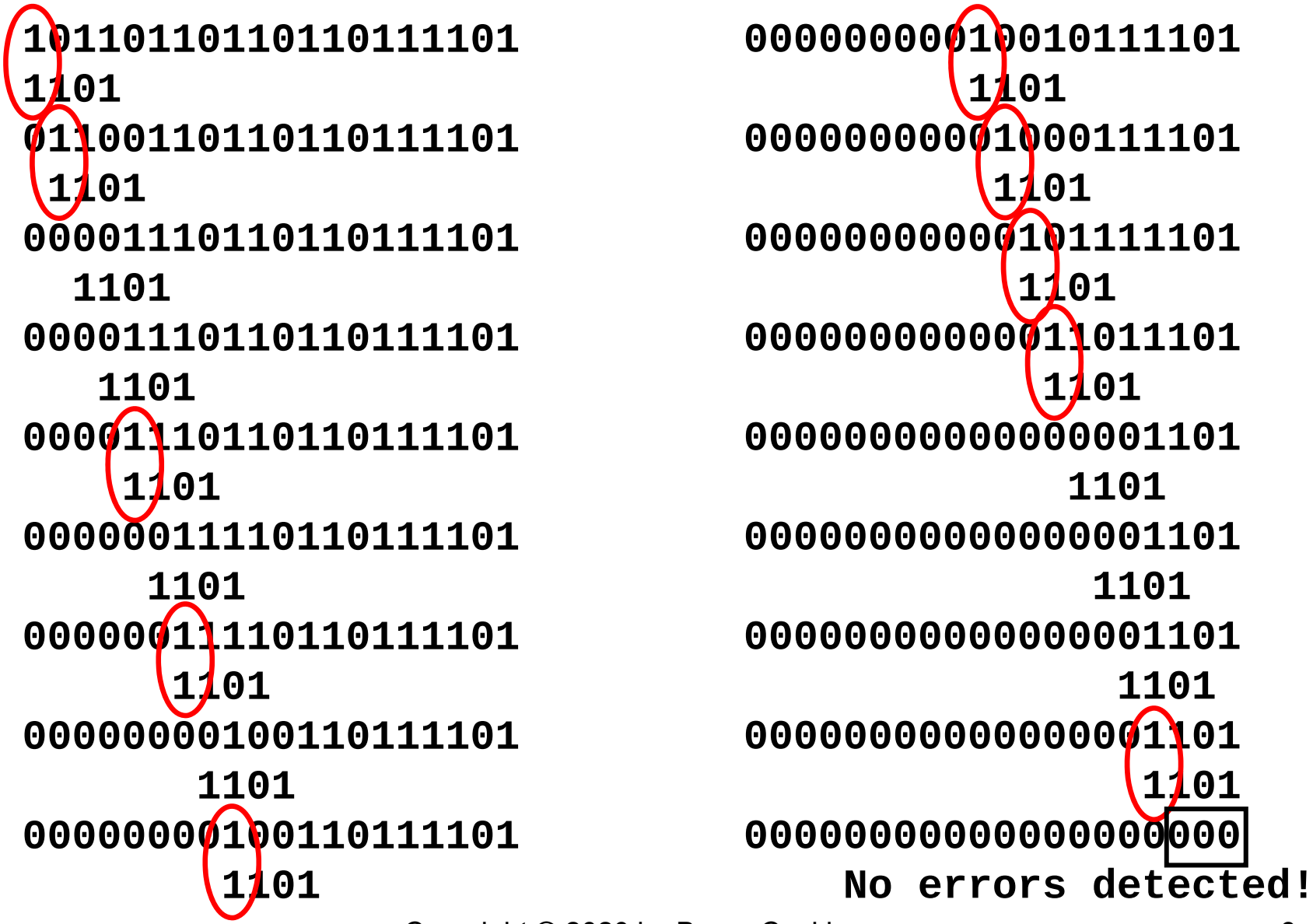
- If the divisor MSB is alongside a 1 in the input stream, then the two vectors are XOR-ed together bitwise. Shift the divisor right, and repeat.

```
01100110110110110101
 1101
```

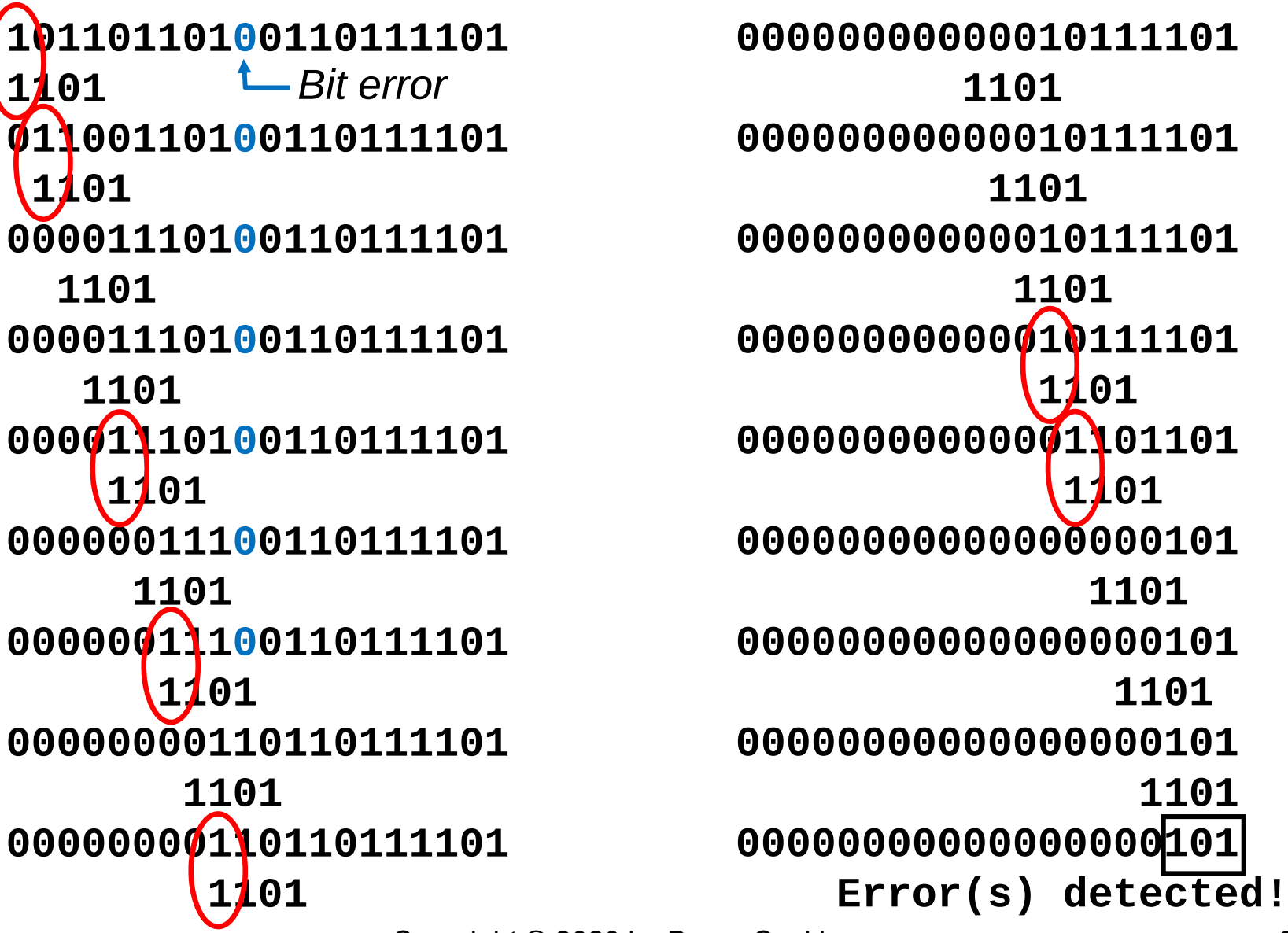
- Once the process is finished, all of the input bits will have been zeroed, except maybe the remainder at the right end. This is the computed CRC.

```
00000000000000000000  <-- CRC (zero, so no errors detected)
                1101
```

Example CRC-3 Calculation (no error)



Example CRC-3 Calculation (with detected error)



Ethernet Data Rates and Transmission Media

- Ethernet is available for several different data rates (10, 100, 1000, 10000 Mbps, etc.) and a variety of media:
  - **RG-8** thick 50-Ω coaxial cable, for 10-Mbps (10BASE5, the original Ethernet standard). Now obsolete.
  - **RG-58** “thin” 50-Ω coaxial cable, for 10-Mbps (10BASE2).

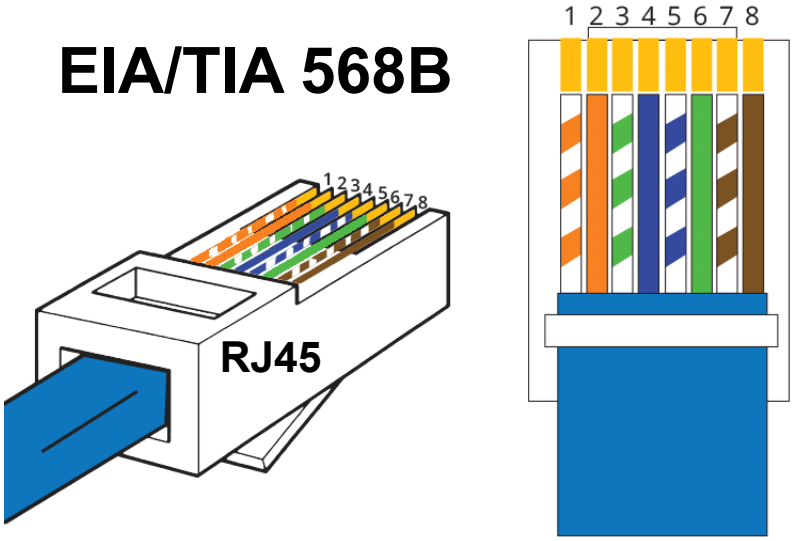
  Coax cables have been replaced by point-to-point twisted pair links.
  - **CAT-3** twisted pair, for 10-Mbps (10BASE-T)
  - **CAT-5e** twisted pair, for both 100-Mbps (100BASE-TX, a.k.a. Fast Ethernet) and 1-Gbps (1000BASE-T)
  - **CAT-6a**, for 10-Gbps (10GBASE-T)
  - **CAT-7**, intended for 10-Gbps, but CAT-6a is now used instead
  - **CAT-8**, for short range (5 to 30m) at 25 or 40 Gbps
  - **Optical fibre**, for 100-Mbps (100BASE-FX), 1-Gbps (1000BASE-SX),10-Gbps (10GBASE-LX4), 40-Gbps

Twisted Pairs within an Ethernet Cable

- Just as in RS-232C, the 10BASE-T and 100BASE-TX standards are asymmetrical interfaces, where each cable has a computer/peripheral end and a communications equipment (e.g., router, hub) end.
- CAT-5e (and better) cables contain four twisted pairs.
- One twisted pair (terminating on pins 1 & 2) is used for the transmit (Tx) direction, and a second twisted pair (terminating on pins 3 & 6) is used for the receive direction.
- The two other twisted pairs (terminating on pins 4 & 5, and on 7 & 8) can be used to carry a second bi-directional 10BASE-T or 100BASE-TX connection.
- Faster Ethernet standards (e.g., 1000BASE-T, 10GBASE-T) use all four twisted pairs, with signals travelling simultaneously in both directions.
- A *crossover cable* was originally required to connect two devices (e.g. computer-to-computer) of the same type. However, most modern Ethernet ports have a circuit that automatically creates a crossover if necessary, using a mechanism defined in the Auto MDI-X specification.

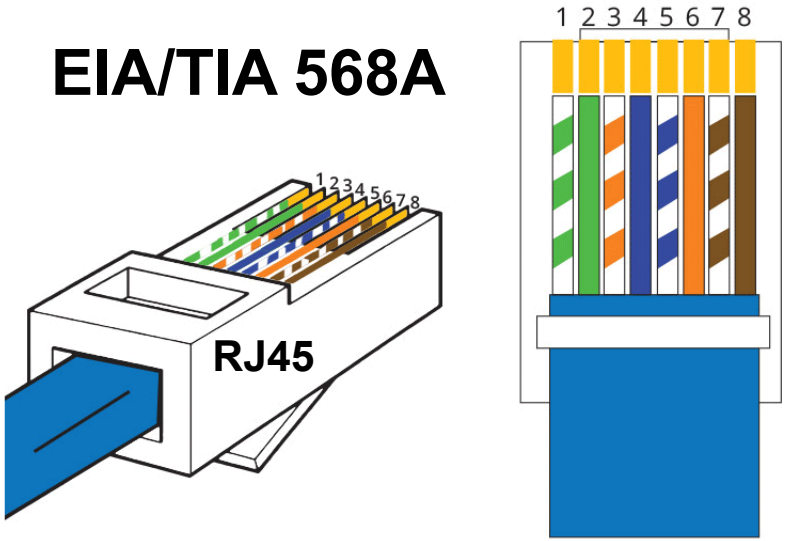
Type A and B Pin-outs at the RJ45 Plug

EIA/TIA 568B



1. White Orange	5. White Blue
2. Orange	6. Green
3. White Green	7. White Brown
4. Blue	8. Brown

EIA/TIA 568A

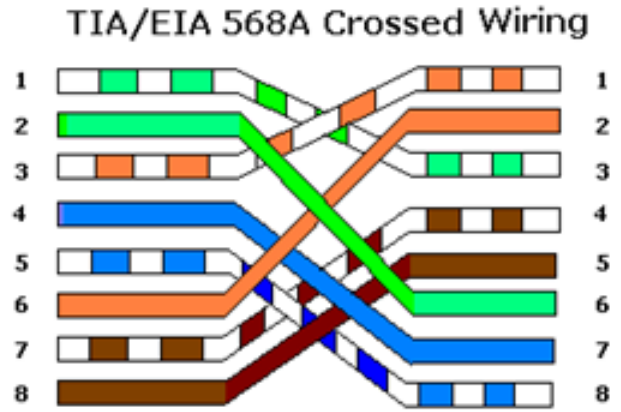


1. White Green	5. White Blue
2. Green	6. Orange
3. White Orange	7. White Brown
4. Blue	8. Brown

Figure courtesy of www.cat6wiringdiageram.com

- Normal (straight-through) patch cables are A↔A or B↔B.
- Ethernet crossover cables are A↔B or B↔A.

Straight-through and Crossover Ethernet Cables



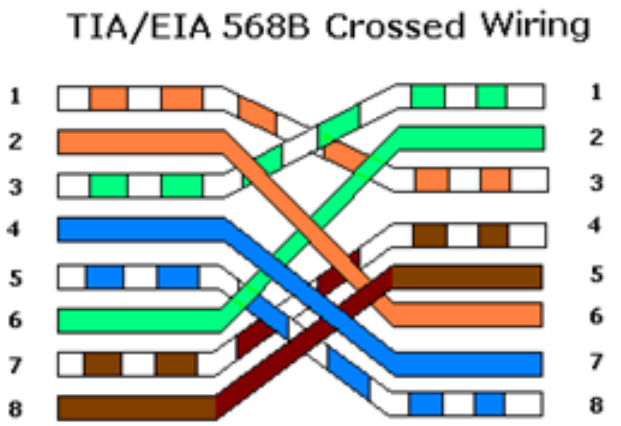


Figure courtesy of electronicsforu.com

Figure A

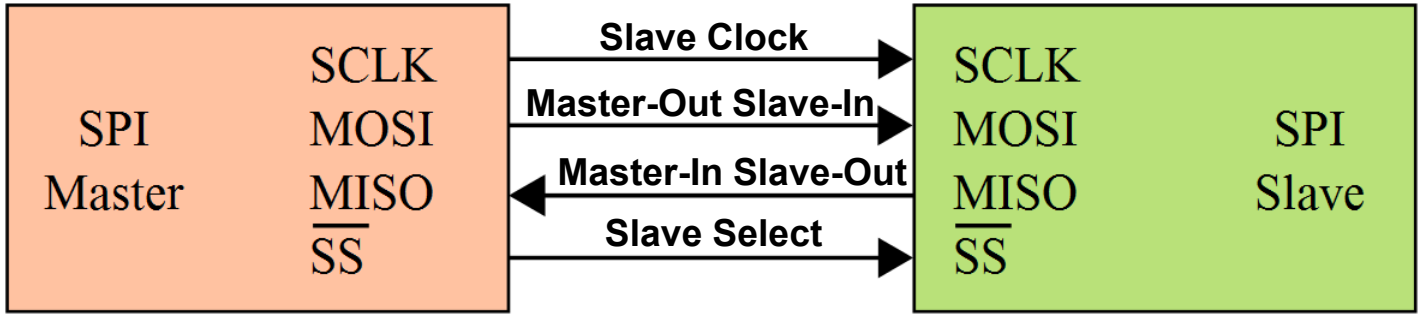
Shows the Pin Out of Straight through Cables

Figure B

Shows the Pin Out of Crossover Cables

The Serial Peripheral Interface (SPI) bus

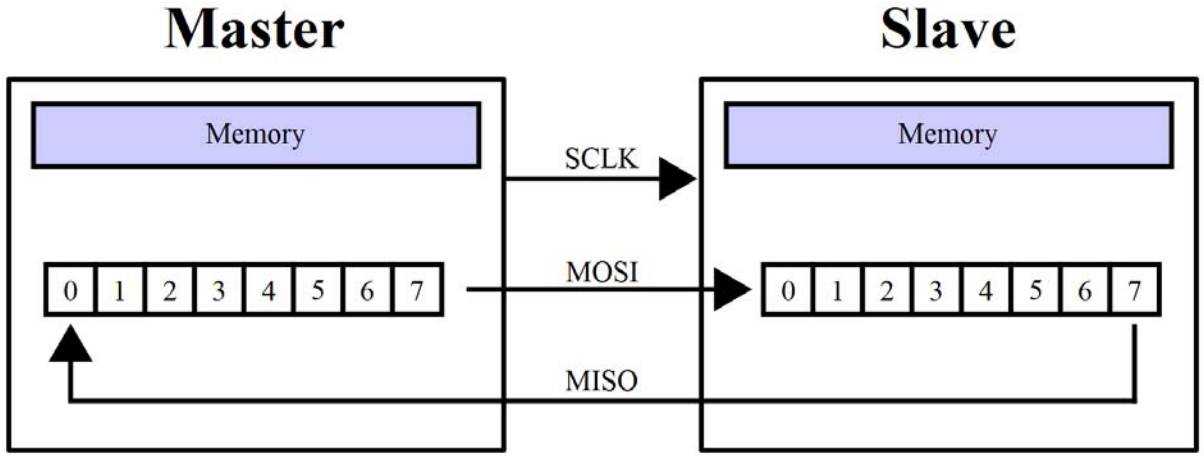
- An *ad hoc* standard that was first proposed by Motorola in the late 1980s. (Motorola spun off its semiconductor division as Freescale Semiconductor in 2004.)
- SPI became, and remains, a popular low-cost interface used by embedded systems to communicate with each other and with peripherals (e.g., LCD displays).
- SPI has a simple master-slave architecture where the master and slave communicate over four wires.



From Wikipedia/Wikimedia

- *Caution*: There is some variation in the spelling of the signal abbreviations. The standard is not well enforced.

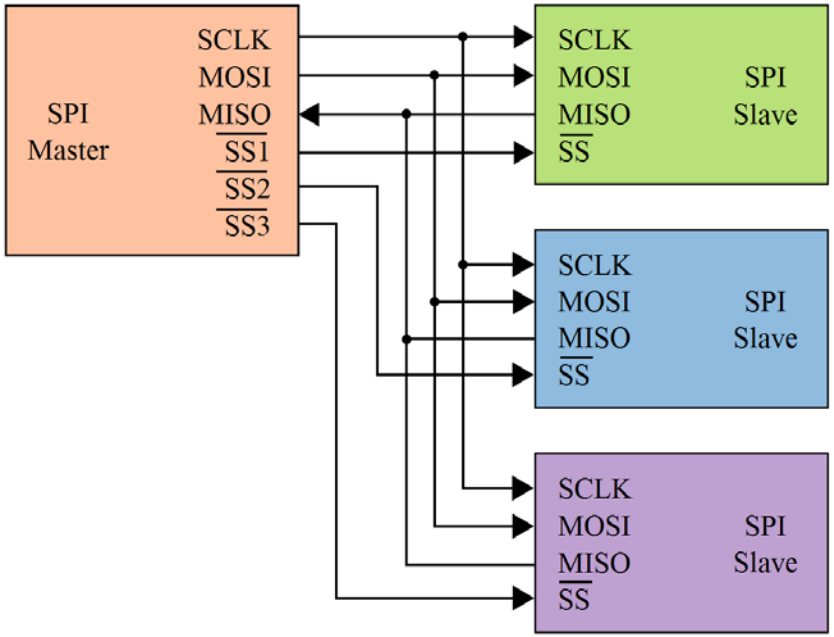
SPI Data Transfer Shift Register



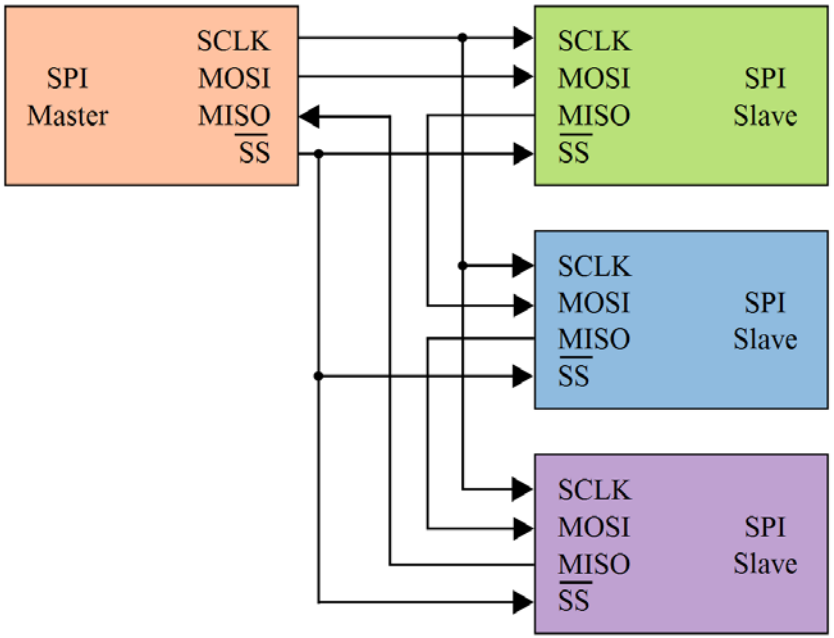
From Wikipedia/Wikimedia

- The standard architecture for transferring data between the master and one slave is to create a large shift register that is split equally between the master and slave.
- New master data shifts over MOSI into the slave while the previous slave data shifts out over MISO to the master. The master effectively reads old data from the slave while writing new data into the slave.
- Note: The MOSI signal is always driven by the master. MISO is actively driven only when the slave is selected; otherwise MISO is tri-stated.

SPI Configurations for Multiple Slaves



From Wikipedia/Wikimedia

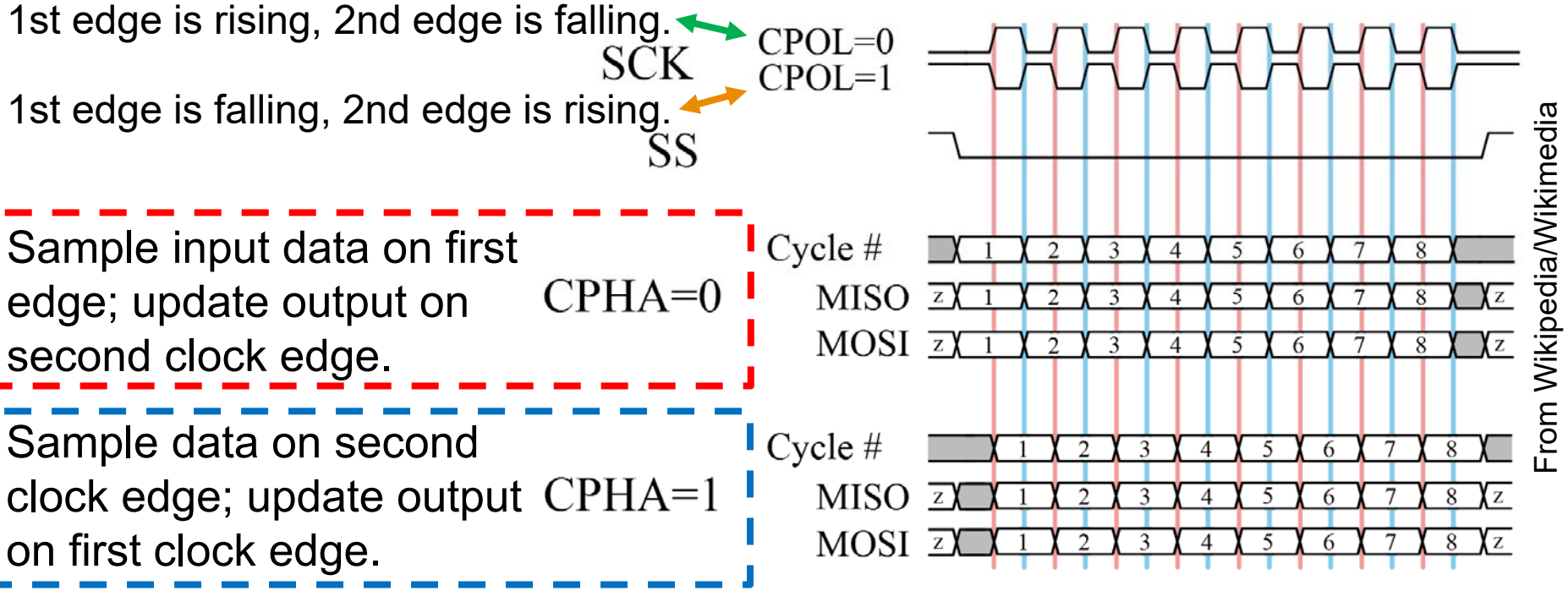


From Wikipedia/Wikimedia

- Separately selected slaves.
- Master must provide one slave select signal (SS*) for each slave, and only one of these can be active at a time (to avoid conflicting signals on MISO).

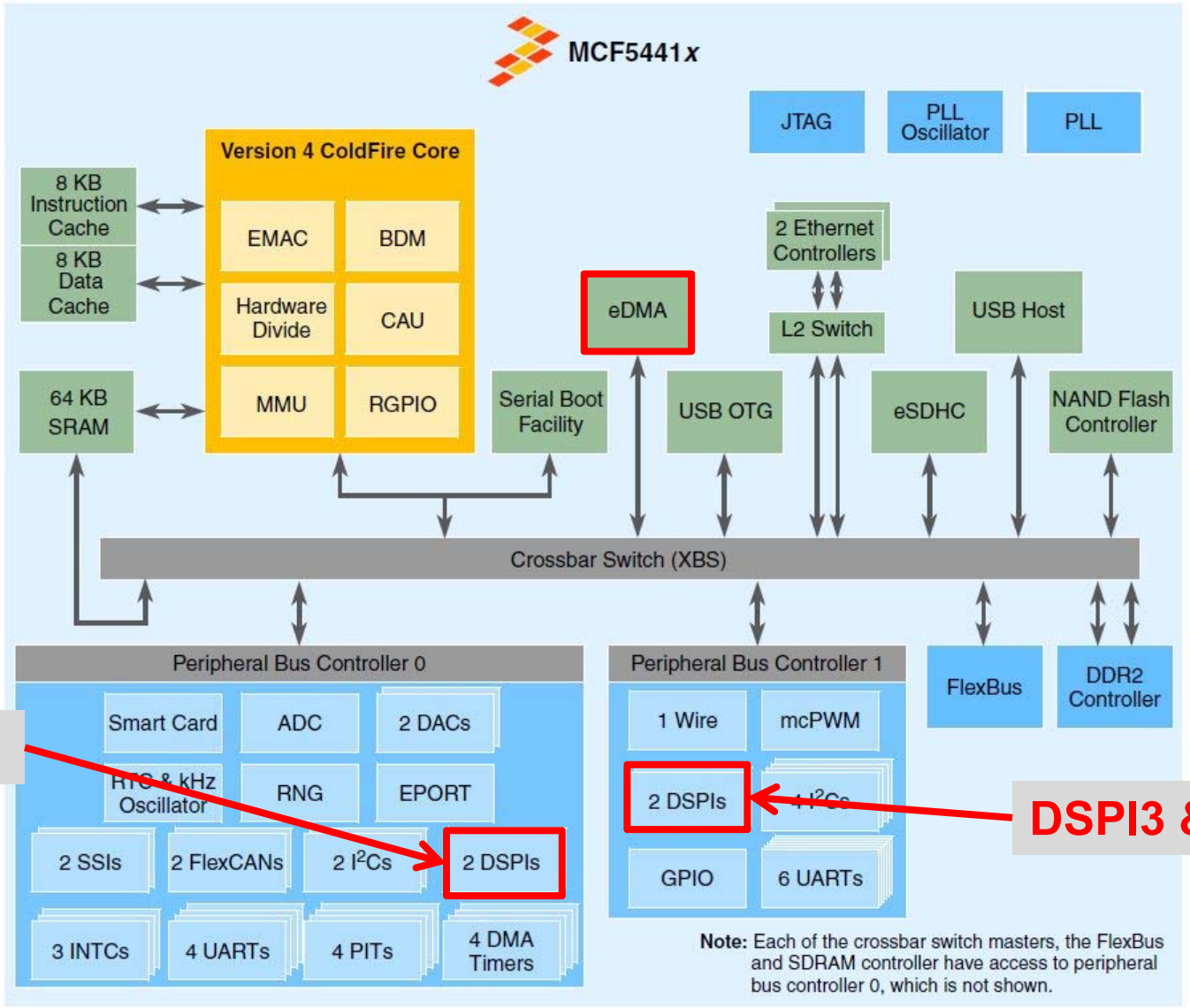
- ***Daisy-chained*** slaves.
- The slaves effectively become one large slave with a shared slave select signal.
- The shift register is the concatenation of all serial registers.

SPI Clock Polarity and Phase Relationships



- Unfortunately, because SPI is an *ad hoc* standard that is not policed by any organization, variations have appeared in the waveforms in “SPI compatible” devices.
- CPOL = 0 (1) means SCK starts off at low (high) voltage.
- CPHA = 0 (1) sample data on first (second) SCK edge.

The Four DMA SPI Interfaces in the MCF54415



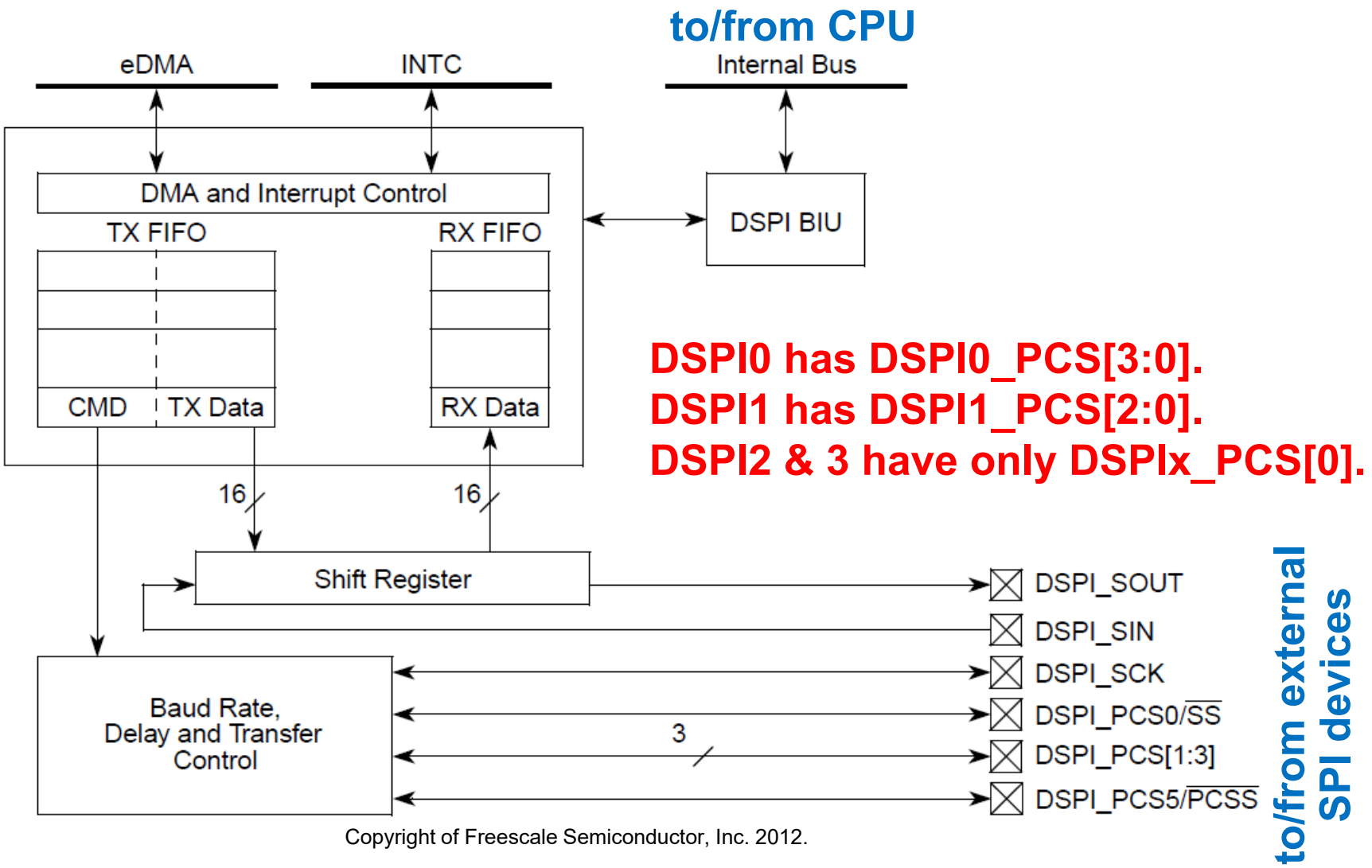
DSPI0 & 1

DSPI3 & 4

The Four DSPI Interfaces

- The MCF54415 has four **D**irect Memory Access **S**erial **P**eripheral **I**nterfaces (DSPIs), labelled DSPI0, DSP1, DSPI2 and DSPI3.
- Each DSPI can connect to multiple external devices (e.g., LCD displays, microcontrollers) that have SPI interfaces.
- Each DSPI block can be in one of four modes: (1) master mode; (2) slave mode; (3) disabled (low power); & (4) debug.
- All four DSP interfaces can handle 16 queued output transfers and 16 queued input transfers. The transfers are controlled, without requiring detailed CPU involvement, by an ***enhanced direct memory access*** (eDMA) controller that is also provided in the MCF54415.

DSPI Interface Architecture on the MCF54415



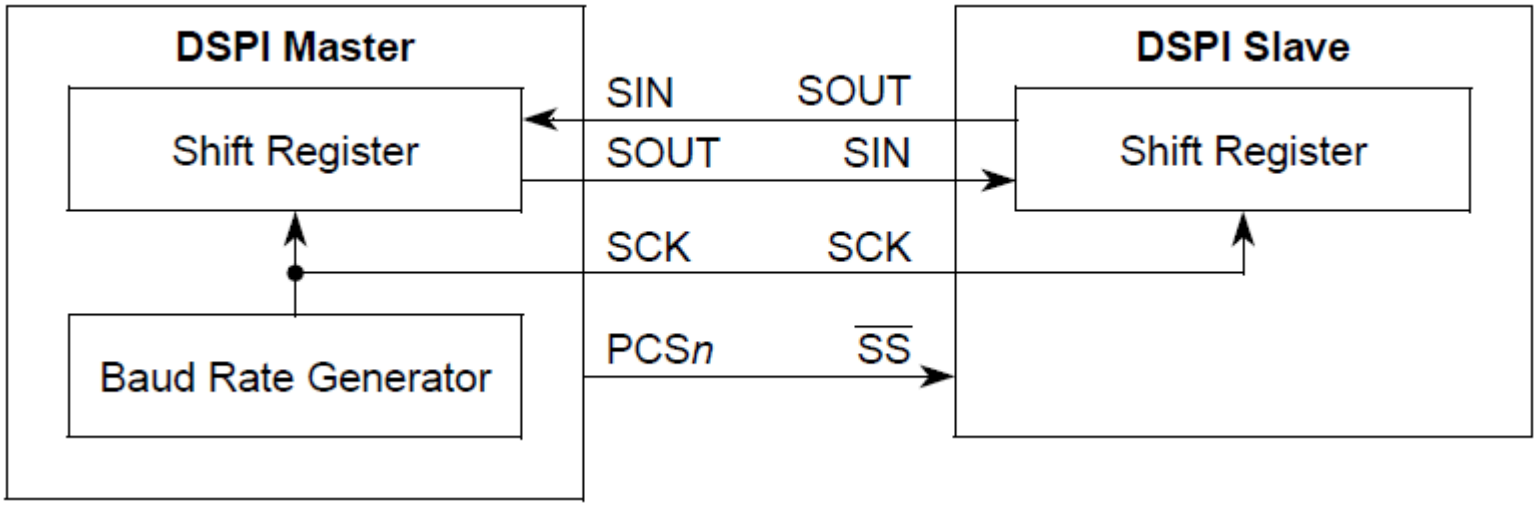
DSPI0 has DSPI0_PCS[3:0].
DSPI1 has DSPI1_PCS[2:0].
DSPI2 & 3 have only DSPIx_PCS[0].

DSPI Interface Architecture on the MCF5441x

Name	Function			
	Master Mode	I/O	Slave Mode	I/O
DSPI_PCS0/$\overline{SS}$	Peripheral chip select 0	Output	Slave select	Input
DSPI_PCS[1:3]	Peripheral chip select 1–3	Output	Unused	—
DSPI_PCS5/$\overline{PCSS}$	Peripheral chip select 5/ Peripheral chip select strobe	Output	Unused	—
DSPI_SIN	Serial data in	Input	Serial data in	Input
DSPI_SOUT	Serial data out	Output	Serial data out	Output
DSPI_SCK	Serial clock	Output	Serial clock	Input

- In ***master mode***, a DSPI block drives the _SCK clock and the peripheral chip selects _PCSn. It also determines the SPI transactions using its Tx FIFO.
- In ***slave mode***, an external SPI master drives the _SCK clock. The DSPI block is enabled when _SS is asserted.

DSPI Interface Architecture on the MCF5441x



- In ***master mode***, a DSPI block drives the _SCK clock and the peripheral chip selects _PCSs. It also determines the SPI transactions using its Tx FIFO.
- In ***slave mode***, an external SPI master drives the _SCK clock. The DSPI block is enabled when _SS is asserted.

Memory-mapped DSPI Registers

- The DPSI blocks are controlled by the CPU through memory-mapped registers. Each DSPI is assigned a 16 K-(16384-) byte large region in the CPU's memory map.

Base Address	Module
0xFC05_C000	DSPI 0
0xFC03_C000	DSPI 1
0xEC03_8000	DSPI 2
0xEC03_C000	DSPI 3

- Each DSPI block has 46 32-bit memory-mapped registers that the CPU can access. 33 of these registers are read-only; the other 13 registers are readable and writeable.

Memory-mapped DSPI Registers

Offset	Register Name and Abbreviation	Size	Mode
0x00	DSPI module configuration reg. (DSPI_MCR)	32-bit	R/W
0x08	DSPI transfer count register (DSPI_TCR)	32-bit	R/W
0x0C	DSPI clock and transfer attributes regs.	32-bit	R/W
...	(DSPI_CTAR0..7)		
0x28			
0x2C	DSPI status register (DSPI_SR)	32-bit	R/W
0x30	DSPI DMA/interrupt request & enable reg.	32-bit	R/W
0x34	DSPI push Tx FIFO register (DSPI_PUSHR)	32-bit	R/W
0x38	DSPI pop Rx FIFO register (DSPI_POPR)	32-bit	R
0x3C	DSPI transmit FIFO registers	32-bit	R
...	(DSPI_TXFR0..15)		
0x78			
0x7C	DSPI receive FIFO registers	32-bit	R
...	(DSPI_RXFR0..15)		
0xB8			