CALMOS™

CA85C30

SERIAL COMMUNICATIONS CONTROLLER

- **Low power CMOS**
- **Pin compatible to NMOS versions**
- **High speed - 2.5M bit/sec channels**
- **Two independent, full-duplex channels, each with separate crystal oscillator, baud rate generator and Digital PLL**
- **Multi-protocol operation, programmable for NRZ, NRZI or FM data encoding**
- **Asynchronous mode with programmable clock factor**
- **Break detection and generation**
- **Parity, overrun, and framing error detection**
- **Synchronous mode with internal or external character synchronization**
- **Local Loopback and auto echo modes**
- **Supports T1 digital trunk**
- **Enhanced DMA support**

The CA85C30 CMOS SCC Serial Communications Controller is an enhanced CMOS version of the industry standard NMOS SCC. It is a dual channel, multi-protocol data communications peripheral that easily interfaces to CPUs with non-multiplexed address/data buses. The advanced CMOS process offers lower power consumption, higher performance, and superior noise immunity. The programming flexibility of the internal registers allows the SCC to be configured to satisfy a wide variety of serial communications applications. The many on-chip features, like baud rate generators, digital phase locked loops, and crystal oscillators dramatically reduce the need for external logic. Additional features including a 10 x 19-bit status FIFO and 14-bit byte counter were added to support high speed SDLC transers using DMA controllers.

The SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives etc.).

The SCC can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. It has facilities for modem controls in both channels, and where these controls are not needed by an application, they can be used for general-purpose I/O. The daisy-chain interrupt hierarchy is also supported.

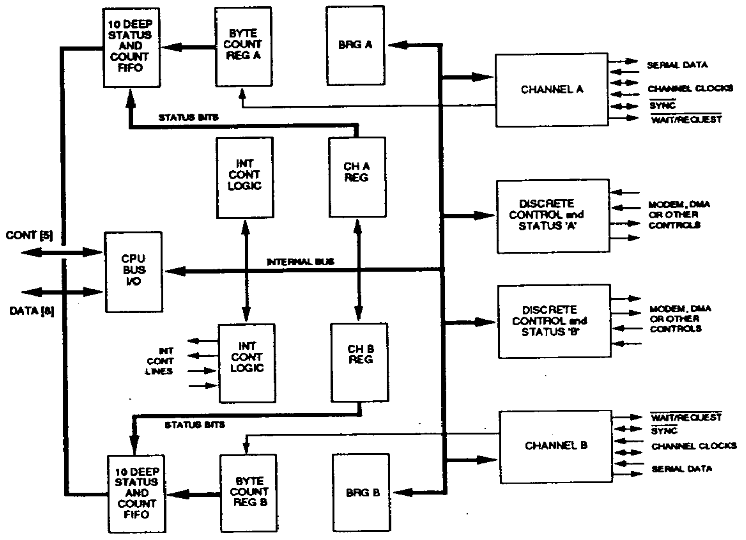


Figure 1 : CA85C30 BLOCK DIAGRAM

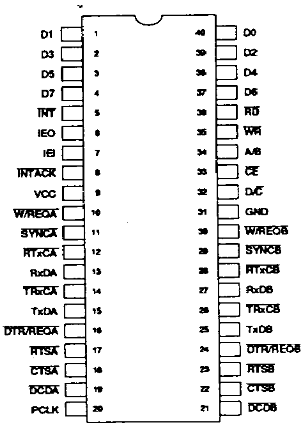


Figure 2 : PIN CONFIGURATION

6

Table 1 : PIN DESCRIPTIONS

Symbol	Pins (PDIP)	Type	Name and Function
A/$\overline{\text{B}}$	34	I	**Channel A/Channel B :** This signal selects the channel in which the read or write operation occurs.
$\overline{\text{CE}}$	33	I	**Chip Enable:** This signal selects the SCC for a read or write operation.
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$	18, 22	I	**Clear To Send:** If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.
D_0 - D_7	1-4, 37-40	I/O	**Data Bus (3-state):** These lines carry data and commands to and from the SCC.
D/$\overline{\text{C}}$	32	I	**Data/Control Select:** This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.
$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$	19, 21	I	**Data Carrier Detect:** These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise-time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.
$\overline{\text{DTR/REQA}}$ $\overline{\text{DTR/REQB}}$	16, 24	O	**Data Terminal Ready/Request:** These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.
IEI	7	I	**Interrupt Enable In:** IEI is used with IEO to form an interrupt daisy-chain when there is more than one interrupt driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.
IEO	6	O	**Interrupt Enable Out:** IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and this inhibits interrupts from lower priority devices.
$\overline{\text{INT}}$	5	O	**Interrupt Request:** This signal is activated when the SCC requests an interrupt.
$\overline{\text{INTACK}}$	8	I	**Interrupt Acknowledge:** This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{\text{RD}}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{\text{INTACK}}$ is latched by the rising edge of PCLK.
PCLK	20	I	**Clock:** This is the master SCC clock used to synchronize internal signals. PCLK is a TTL level signal. PCLK is not required to have any phase relationship with the bit rate clocks except when $^1/_4$ PCLK.
$\overline{\text{RD}}$	36	I	**Read:** This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.
$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$	17, 23	O	**Request to Send:** When the Request To Send (RTS) bit in Write Register 5 (Figure 15) is set, the $\overline{\text{RTS}}$ signal goes Low. When the RTS bit is reset in Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

Table 1 : PIN DESCRIPTIONS cont

Symbol	Pins (PDIP)	Type	Name and Function
$\overline{RTxCA}$, $\overline{RTxCB}$	12, 28	I	**Receive/Transmit Clocks:** These pins can be programmed in several different modes of operation. In each channel $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in asynchronous modes.
RxDA, RxDB	13, 27	I	**Receive Data:** These inputs signals receive serial data at standard TTL levels.
$\overline{SYNCA}$, $\overline{SYNCB}$	11, 29	I/O	**Synchronization:** These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 14) but have no other function. In external Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven *Low* two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$. In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.
$\overline{TRxCA}$, $\overline{TRxCB}$	14, 26	I/O	**Transmit/Receive Clocks:** These pins can be programmed in several different modes of operation. TRxC may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.
TxDA, TxDB	15, 25	O	**Transmit Data:** These output signals transmit serial data at standard TTL levels.
V_{DD}	9		**Power:** 5V ± 5% DC Supply
V_{SS}	31		**Ground:** 0V
$\overline{WR}$	35	I	**Write:** When the SCC is selected, this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.
$\overline{W/REQA}$ $\overline{W/REQB}$	10, 30	O	**Wait/Request:** Open-drain when programmed for a Wait function, driven *High* or *Low* when programmed for a Request function. These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

6

Table 2 : AC CHARACTERISTICS (T_A = 0° to 70°C, V_{DD} = 5V ± 5%)

Symbol	Parameter	Test Conditions	Limits						Units
			6 MHz		8MHz		10MHz		
			Min	Max	Min	Max	Min	Max	
TcPC	PCLK Cycle Time		165	2000	125	2000	100	2000	ns
TdA(DR)	Address Required Valid to Read Data Valid Delay			280		220		180	ns
TdIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Delay	Note 5	200		150		125		ns
TdIEI(IEO)	IEI to IEO Delay Time			100		95		90	ns
TdPC(IEO)	PCLK ↑ to IEO Delay			250		200		175	ns
TdPC(INT)	PCLK ↓ to $\overline{INT}$ Valid Delay	Note 4		500		500		500	ns
TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay			180		140		120	ns
TdRDA(INT)	$\overline{RD}$ ↓ to $\overline{INT}$ Inactive Delay	Note 4		500		500		500	ns
TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay		0		0		0		ns
TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay	Note 2		45		40		35	ns
TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay			180		140		120	ns
TdRDf(REQ)	$\overline{RD}$ ↓ to W/$\overline{REQ}$ Not Valid Delay			200		170		160	ns
TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay		0		0		0		ns
TdRDr(REQ)	$\overline{RD}$ ↑ to $\overline{DTR/REQ}$ Not Valid Delay			4TcPC		4TcPC		4TcPC	ns
TdRD(W)	$\overline{RD}$ ↓ Wait Valid Delay	Note 4		200		170		160	ns
TdRD(WRQ)	$\overline{RD}$ ↑ to $\overline{WR}$ ↓ Delay for No Reset		15		15		15		ns
TdWR(REQ)	$\overline{WR}$ ↓ to W/$\overline{REQ}$ Not Valid Delay			200		170		160	ns
TdWRQ(RD)	$\overline{WR}$ ↑ to $\overline{RD}$ ↓ Delay for No Reset		30		15		15		ns
TdWRr(REQ)	$\overline{WR}$ ↓ $\overline{DTR/REQ}$ Not Valid Delay			4TcPC		4TcPC		4TcPC	ns
TdWR(W)	$\overline{WR}$ ↓ to Wait Valid Delay	Note 4		200		170		160	ns
TfPC	PCLK Fall Time			10		10		10	ns
ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time		0		0		0		ns
ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time		0		0		0		ns
ThCE(RD)	CE to $\overline{RD}$ ↑ Hold Time	Note 1	0		0		0		ns
ThCE(WR)	CE to WR ↑ Hold Time		0		0		0		ns
ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time		20		20		20		ns
ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time		100		40		30		ns
ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time		0		0		0		ns
ThIA(WR)	$\overline{INTACK}$ to WR ↑ Hold Time		0		0		0		ns
ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time		0		0		0		ns
Trc	Valid Access Recovery Time	Note 3	4TcPC		4TcPC		4TcPC		ns
TrPC	PCLK Rise Time			10		10		10	ns
TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time		80		70		50		ns
TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time		80		70		50		ns
TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	Note 1	70		60		50		ns
TsCEh(WR)	$\overline{CE}$ High to WR ↓ Setup Time		70		60		50		ns

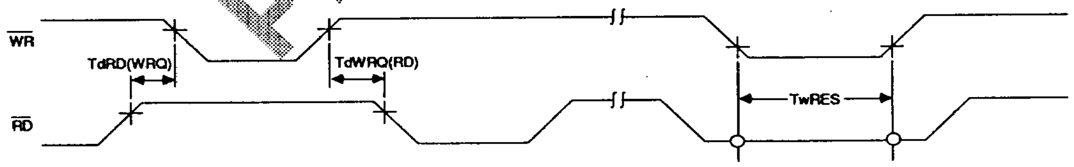
Table 2 : AC CHARACTERISTICS con't

Symbol	Parameter	Test Conditions	Limits 6 MHz Min	Limits 6 MHz Max	Limits 8MHz Min	Limits 8MHz Max	Limits 10MHz Min	Limits 10MHz Max	Units
TsCEI(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	Note 1	0		0		0		ns
TsCEI(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time		0		0		0		ns
TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time		10		10		10		ns
TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	Note 1	160		145		130		ns
TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	Note 1	160		145		130		ns
TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time		20		20		20		ns
TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time		100		95		95		ns
TwPCh	PCLK High Width		70	1000	50	1000	40	1000	ns
TwPCl	PCLK Low Width		70	1000	50	1000	40	1000	ns
TwRDA	$\overline{RD}$ (Acknowledge) Width		200		150		125		ns
TwRDl	$\overline{RD}$ Low Width	Note 1	200		150		125		ns
TwRES	$\overline{WR}$ and $\overline{RD}$ Coincident Low for Reset		200		150		100		ns
TwWRl	$\overline{WR}$ Low Width		200		150		125		ns

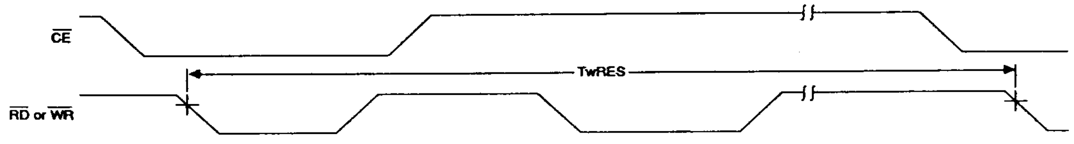
Notes: 1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ± 0.5V change at the output with a maximum DC load and minimum AC load.
3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEI(IEO) for each device separating them in the daisy chain.

Figure 3 : TIMING DIAGRAMS

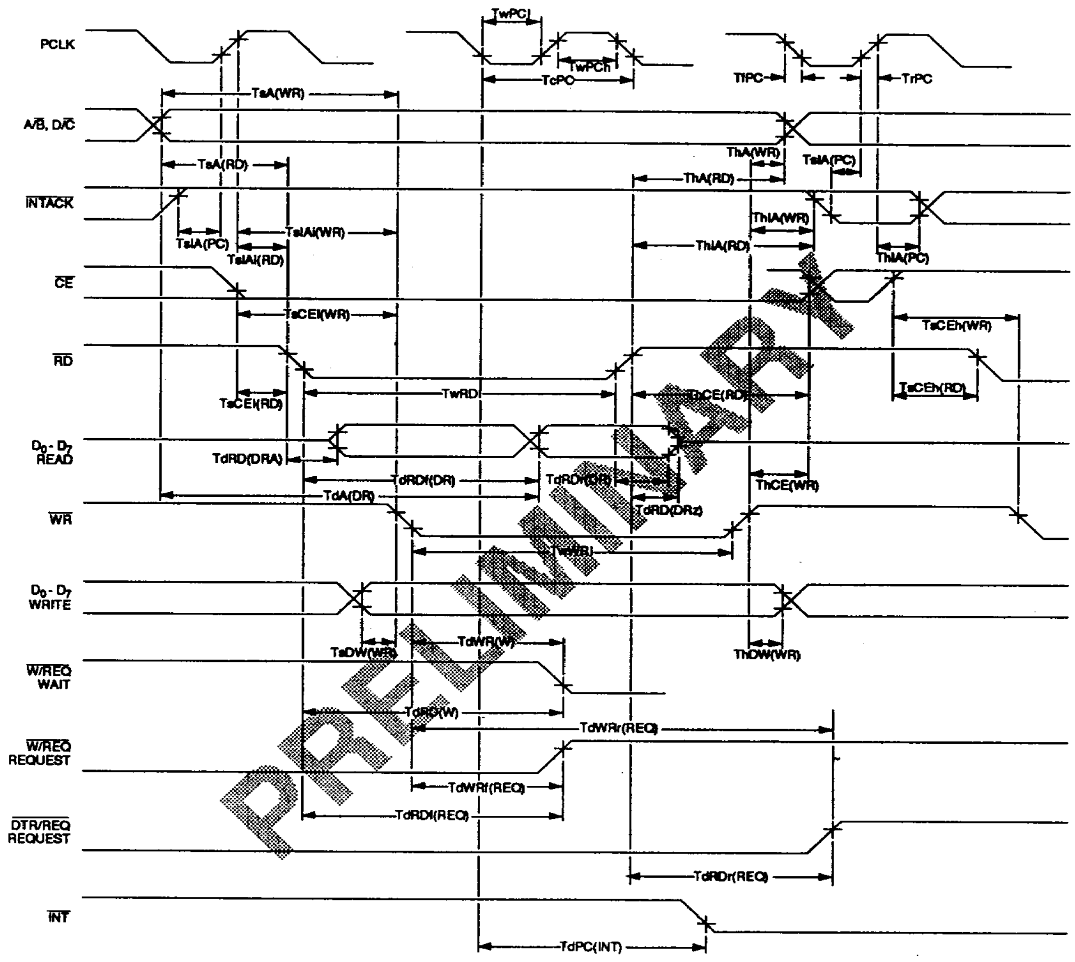
a) Reset Timing



b) Cycle Timing

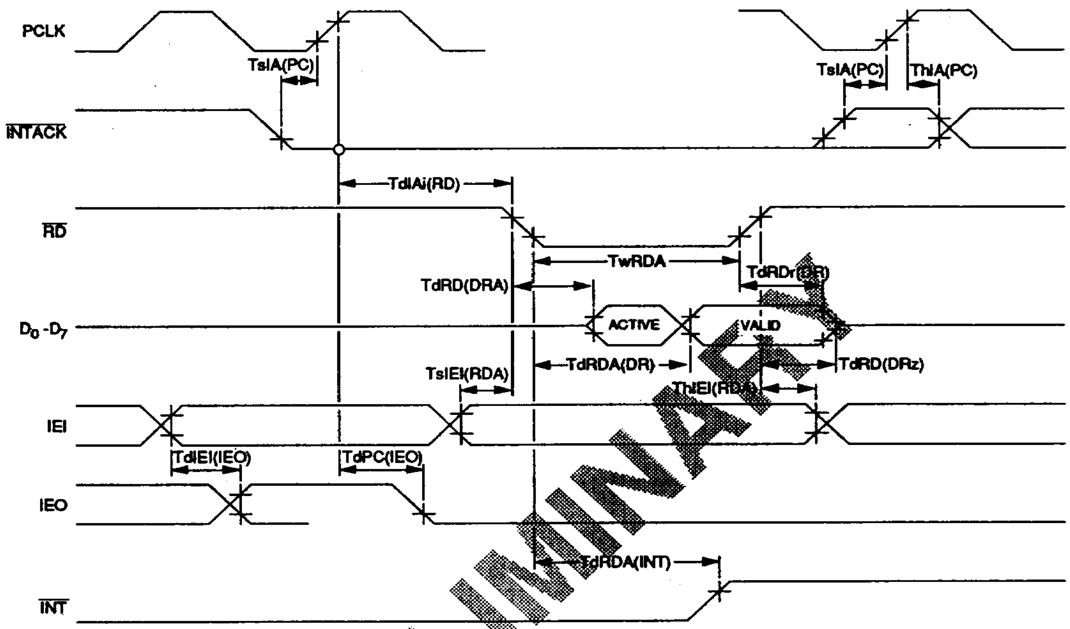


c) Read and Write Timing



Note: Addresses on A/B̄ and D/C̄ and the status on INTACK must remain stable throughout the cycle. If CE falls after RD or WR falls, or if it rises before RD or WR rises, the effective RD or WR is shortened. Data must be valid before the falling edge of WR.

d) Interrupt Acknowledge Timing



Note: Between the time $\overline{\text{INTACK}}$ goes Low and the falling edge of $\overline{\text{RD}}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{\text{RD}}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{\text{RD}}$ Low by placing its interrupt vector on D_0-D_7 and it then sets the appropriate Interrupt-Under-Service latch internally.

Table 3 : AC CHARACTERISTICS, GENERAL TIMING (T_A = 0° to 70°C, V_{DD} = 5V ± 5%)

			Limits						
		Test	6 MHz		8MHz		10MHz		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Units
TcRTX	$\overline{\text{RTxC}}$ Cycle Time (RxD, TxD)	Notes 6,7	640		500		400		ns
TcRTXX	Crystal Oscillator Period	Note 3	165	1000	125	1000	100	1000	ns
TcTRX	$\overline{\text{TRxC}}$ Cycle Time	Notes 6, 7	640		500		400		ns
TdPC(REQ)	PCLK ↓ to $\overline{\text{W/REQ}}$ Valid Delay			250		250		250	ns
TdPC(W)	PCLK ↓ to Wait Inactive Delay			350		350		350	ns
TdTXCf(TXD)	$\overline{\text{TxC}}$ ↓ to TxD Delay (X1 Mode)	Note 2		230		200		150	ns
TdTxCr(TXD)	$\overline{\text{TxC}}$ ↑ to TxD Delay (X1 Mode)	Notes 2, 5		230		200		150	ns
TdTXD(TRX)	TxD to $\overline{\text{TRxC}}$ Delay (Send Clock Echo)			200		200		200	ns
ThRXD(RXCf)	RxD to $\overline{\text{RxC}}$ ↓ Hold Time (X1 Mode)	Notes 1, 5	150		150		150		ns
ThRXD(RXCr)	RxD to $\overline{\text{RxC}}$ ↑ Hold Time (X1 Mode)	Note 1	150		150		150		ns
ThSY(RXC)	$\overline{\text{SYNC}}$ to $\overline{\text{RxC}}$ ↑ Hold Time		5TcPC		5TcPC		5TcPC		ns
TsRXC(PC)	$\overline{\text{RxC}}$ ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	Notes 1, 4	70	TwPCL	60	TwPCL	40	TwPCL	ns
TsRXD(RXCf)	RxD to $\overline{\text{RxC}}$ ↓ Setup Time (X1 Mode)	Notes 1, 5	0		0		0		ns
TsRXD(RXCr)	RxD to $\overline{\text{RxC}}$ ↑ Setup Time (X1 Mode)	Note 1	0		0		0		ns
TsSY(RXC)	$\overline{\text{SYNC}}$ to $\overline{\text{RxC}}$ ↑ Setup Time	Note 1	–200		–200		–200		ns
TsTXC(PC)	$\overline{\text{TxC}}$ ↓ to PCLK ↑ Setup Time	Notes 2, 4	0		0		0		ns
TwEXT	$\overline{\text{DCD}}$ or $\overline{\text{CTS}}$ Pulse Width		200		200		200		ns
TwRTXh	$\overline{\text{RTxC}}$ High Width	Note 6	180		150		150		ns
TwRTXl	$\overline{\text{RTxC}}$ Low Width	Note 6	180		150		150		ns
TwSY	$\overline{\text{SYNC}}$ Pulse Width		200		200		200		ns
TwTRXh	$\overline{\text{TRxC}}$ High Width	Note 6	180		150		150		ns
TwTRXl	$\overline{\text{TRxC}}$ Low Width	Note 6	180		150		150		ns

Notes:
1. $\overline{\text{RxC}}$ is $\overline{\text{RTxC}}$ or $\overline{\text{TRxC}}$, whichever is supplying the receive clock.
2. $\overline{\text{TxC}}$ is $\overline{\text{TRxC}}$ or $\overline{\text{RTxC}}$, whichever is supplying the transmit clock.
3. Both $\overline{\text{RTxC}}$ and $\overline{\text{SYNC}}$ have 30pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between RxC and PCLK or $\overline{\text{TxC}}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to case PCLK requirements.
7. The maximum receive or transmit data is $^1/_4$ PCLK.

Figure 4 : GENERAL TIMING

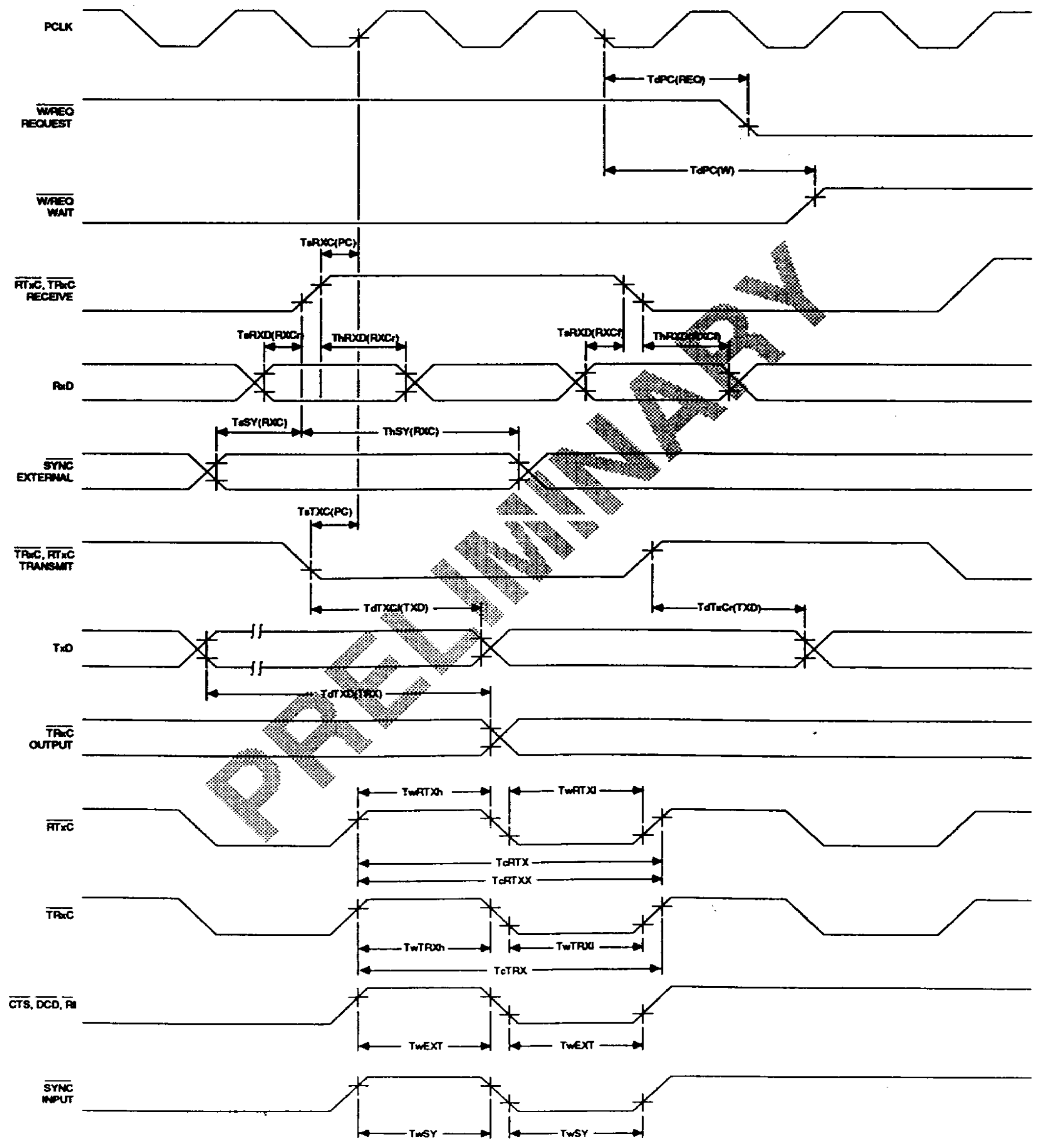
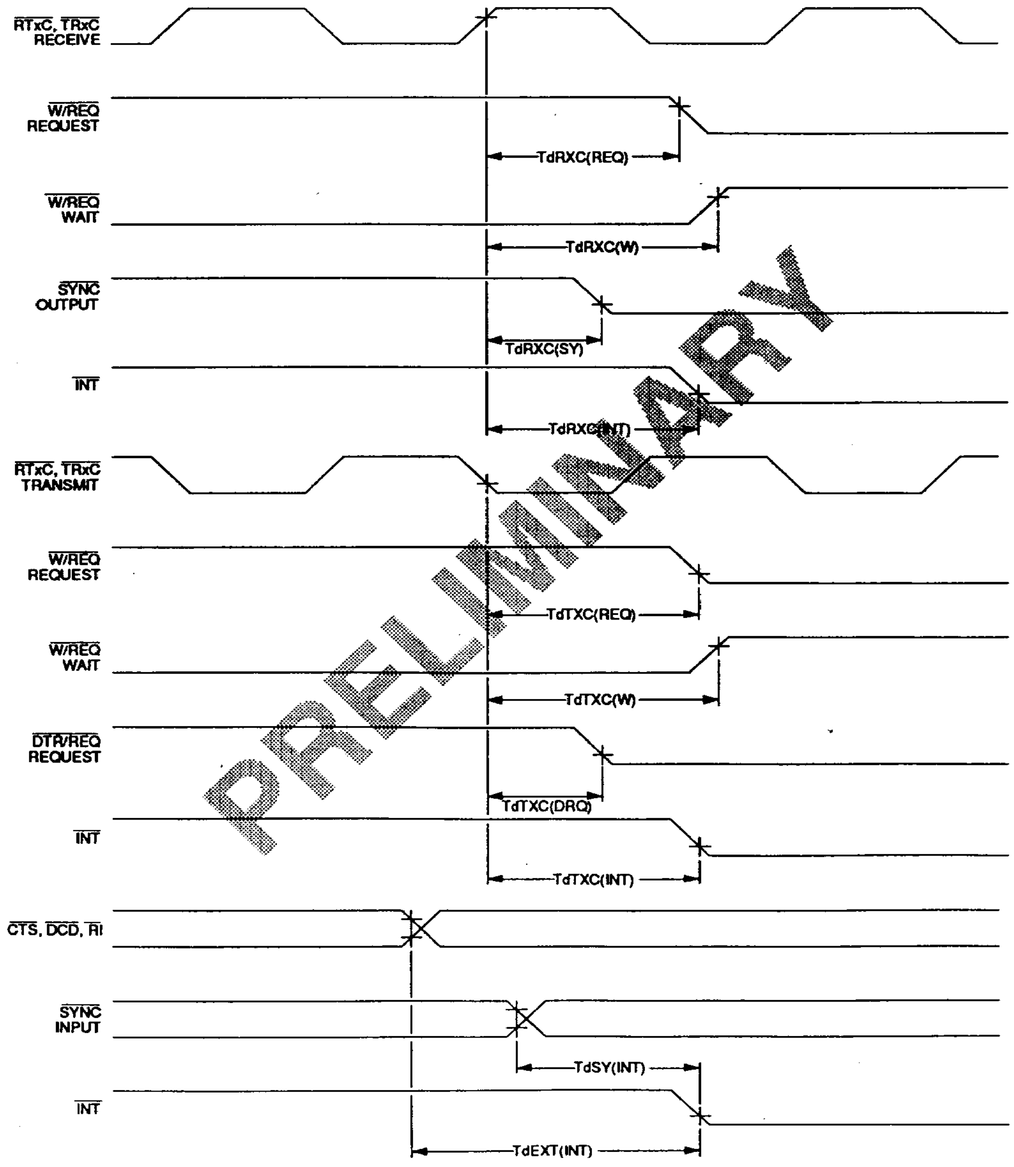


Table 4 : AC CHARACTERISTICS, SYSTEM TIMING (T_A = 0° to 70°C, V_{DD} = 5V ± 5%)

Symbol	Parameter	Test Conditions	Limits						Units
			6 MHz		8MHz		10MHz		
			Min	Max	Min	Max	Min	Max	
TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	Note 1	2	6	2	6	2	6	TcPC
TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	Notes 1, 2	10	16	10	16	10	16	TcPC
TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	Note 2	8	12	8	12	8	12	TcPC
TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	Note 2	4	7	4	7	4	7	TcPC
TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	Notes 1, 2	8	14	8	14	8	14	TcPC
TdSY(INT)	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	Note 1	2	6	2	6	2	6	TcPC
TdTXC(DRQ)	TxC ↓ to DTR/REQ Valid Delay	Note 3	4	7	4	7	4	7	TcPC
TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	Notes 1, 3	6	10	6	10	6	10	TcPC
TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	Note 3	5	8	5	8	5	8	TcPC
TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	Notes 1, 3	5	11	5	11	5	11	TcPC

Notes: 1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.

Figure 5 : SYSTEM TIMING



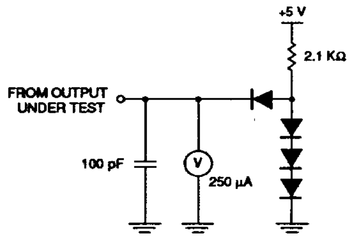


Figure 6 : STANDARD TEST LOAD

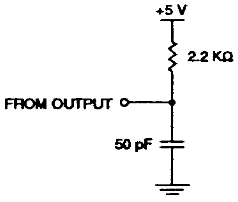


Figure 7 : OPEN-DRAIN TEST LOAD

Table 5 : DC CHARACTERISTICS (T_A = 0° to 70°C, V_{DD} = 5V ± 5%)

Symbol	Parameter	Test Conditions	Limits Min	Limits Max	Units
I_{CC1}	V_{CC} Supply Current	CLK = 10 MHz		30	mA
I_{IL}	Input Leakage	$0.4 \leq V_{IN} \leq +2.4V$		±10.0	μA
I_{OL}	Output Leakage	$0.4 \leq V_{OUT} \leq +2.4V$		±10.0	μA
V_{IH}	Input High Voltage		2.2	V_{CC} + 0.3	V
V_{IL}	Input Low Voltage		–0.3	0.8	V
V_{OH1}	Output High Voltage	I_{OH} = –1.6mA	2.4		V
V_{OH2}	Output High Voltage	I_{OH} = –250μA	V_{CC} – 0.8		V
V_{OL}	Output Low Voltage	I_{OL} = +2.0mA		0.4	V

Table 6 : CAPACITANCE (T_A = 25 °C , V_{DD} = 5V ± 5%, V_{SS} = 0V)

Symbol	Parameter	Test Conditions	Limits Min	Limits Max	Units
C_{IN}	Input capacitance	Unmeasured pins		10	pF
C_{OUT}	Output Capacitance	Returned to V_{SS}		15	pF
C_{IO}	Bidirectional Capacitance			20	pF

Frequency = 1 MHz, over specified temperature range.

Table 7 : RECOMMENDED OPERATING CONDITIONS

DC Supply Voltage		+4.75V to +5.25V
Operating Temperature Range	Commercial	0°C to 70°C
	Industrial	–40°C to +85°C

Table 8 : ABSOLUTE MAXIMUM RATINGS

Voltages on all pins with respect to V_{DD}	–0.3V to +7.0V
Power Dissipation (PD_{MAX})	165 mW
Operating Temperature (T_{OPT})	0°C to 70°C
Storage Temperature (T_{STG})	–65°C TO +150°C

Stresses beyond those listed above may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

OPERATIONAL DESCRIPTION

The functional capabilities of the SCC can be described from two different points of view; as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the SCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities

The SCC provides two independent full-duplex channels programmable for use in any common asynchronous or synchronous data communication protocol. Figure 8 and the following description briefly detail these protocols.

Asynchronous Modes

Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, 1$^1/_2$, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU at both start and end of a received break. Reception is protected from spikes by transient spike-rejection mechanism that checks the signal one-half a bit time after Low level is detected on the receive data input. If the Low does not persist (as in the case of transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with partial character on which they occur. Vectored interrupts allow error conditions to be quickly serviced using dedicated routines. In addition, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

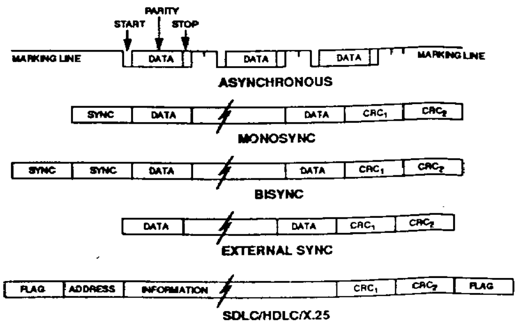


Figure 8 : SOME SCC PROTOCOLS

The SCC does not require symmetric transmit and receive clock signals, thus allowing use of a wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, $^1/_{16}$, $^1/_{32}$, or $^1/_{64}$ of the clock inputs. In asynchronous modes, $\overline{\text{SYNC}}$ may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes

The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6- or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or an external synchronous signal. Leading sync characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 9.

CRC checking for synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bysync.

Both CRC-16 ($X^{16}+X^{15}+X^2+1$) and CCITT ($X^{16}+X^{12}+X^5+1$) error checking polynomials are supported. Either polynomial may be selected in all synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports synchronous bit-oriented protocols like SDLC and HDLC, by performing automatic flag sends, zero insertion, and CRC generation. A special command is used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

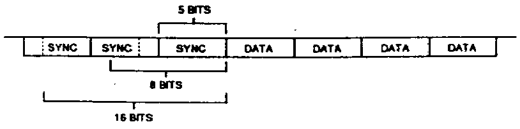


Figure 9 : DETECTING 5- or 7-BIT SYNCHRONOUS CHARACTERS

6

If a transmit underrun occurs in the middle of a message, as external/status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{SYNC}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or 4 bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s. The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The SCC can be conveniently used under DMA control to provide high speed reception or transmission. In reception, for example the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode

The SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 10).

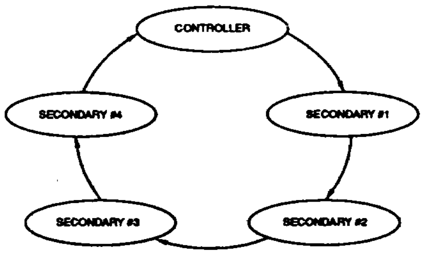


Figure 10 : AN SDLC LOOP

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by re-transmitting them with one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending the special character, EOP (End of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC, and NRZ, NRZI, and FM coding can all be used in this mode.

Baud Rate Generator

Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggle upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the TRxC pin, the output of the baud rate generator may be echoed out via the TRxC pin.

The following formula relates the time constant to the baud rate where PCLK or RTxC is the baud rate generator input frequency in Hz. The clock mode is 1, 16, 32, or 64 as selected in Write Register 4, bits D6 and D7. Synchronous operation modes should select 1 and asynchronous should select 16, 32, or 64.

$$\text{Time Constant} = \frac{\text{PCLK or RTxC Frequency}}{2(\text{Baud Rate})(\text{Clock Mode})} - 2$$

Digital Phase-Locked Loop

The SCC contains a Digital Phase-Locked Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI encoding, DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the incoming data stream for edges (either 1 to 0 or 0 to 1). When an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), to produce a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 and 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding

The SCC may be programmed to encode and decode the serial data in four different ways (Figure 11). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

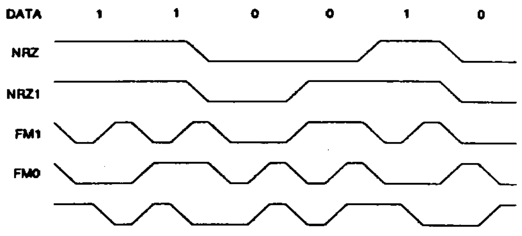


Figure 11 : DATA ENCODING METHODS

Auto Echo and Local Loopback

The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before re-transmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT}}$/$\overline{\text{REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in asynchronous, synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities

The SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling

All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in DSLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for date to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts

When an SCC responds to an Interrupt Acknowledge signal ($\overline{\text{INTACK}}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 14 and 15).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source; Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 12). As a microprocessor peripheral, the SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{\text{INT}}$. The CPU then responds with $\overline{\text{INTACK}}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{\text{INT}}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in, and external to, the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled *low* and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts; Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty, (implying that the transmitter must have had a data character written into it, so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following; receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector

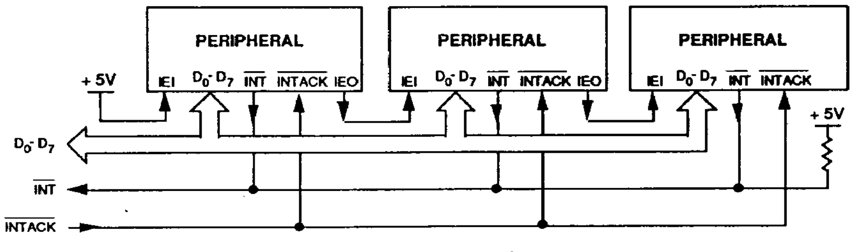


Figure 12 : INTERRUPT SCHEDULE

during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions at any time following the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the CTs, DCD, and SYNC pins. An External/Status interrupt is also caused by a Transmit Underrun condition, a zero count in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a feature that allows the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This ensures the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the primary station request to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer

The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output can be defined under software control as a WAIT line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC $\overline{\text{REQUEST}}$ output indicates that the SCC is not ready to transfer data to or from memory. To the CPU, the WAIT line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}/\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

ARCHITECTURE

The SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, plus the logic necessary to interface to modems or other external devices (Figure 1).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature.

The register set for each channel includes 10 control (write) registers, 2 sync-character (write) registers, and 4 status (read) registers. In addition, each baud rate generator has 2 (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write only Master Interrupt Control register and 3 read registers: one containing the vector with status information (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WRO-WR15 — Write Registers 0 through 15.

RR0-RR3, RR10, RR12, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 9 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path

The transmit and receive data path illustrated in Figure 13 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20 bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

Table 9 : READ AND WRITE REGISTER FUNCTIONS

Register	Function
Read	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified Interrupt vector (Channel B only) Unmodified Interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive Buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status Interrupt information
Write	
WR0	CRC initialize, initialization commands for the various modes, Register Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync characters or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

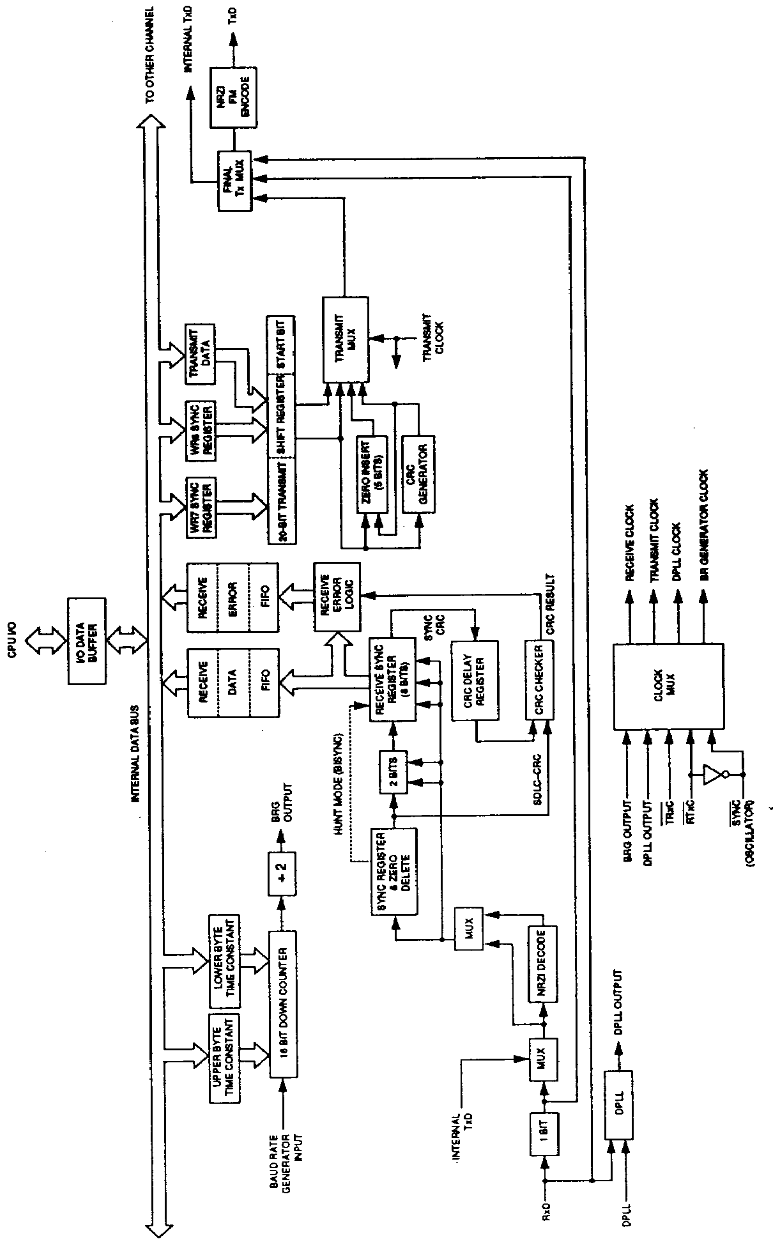


Figure 13 : DATA PATH

PROGRAMMING

The SCC contains write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

In the SCC, register addressing is direct for the data registers only, which are selected by a High on the D/C pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed.

All of the registers in the SCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Read Registers

The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10 and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 14 shows the format of each read register.

The status bits of RR0 and RR1 are grouped to simplify status monitoring; that is, when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

Write Registers

The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 15 shows the format of each write register.

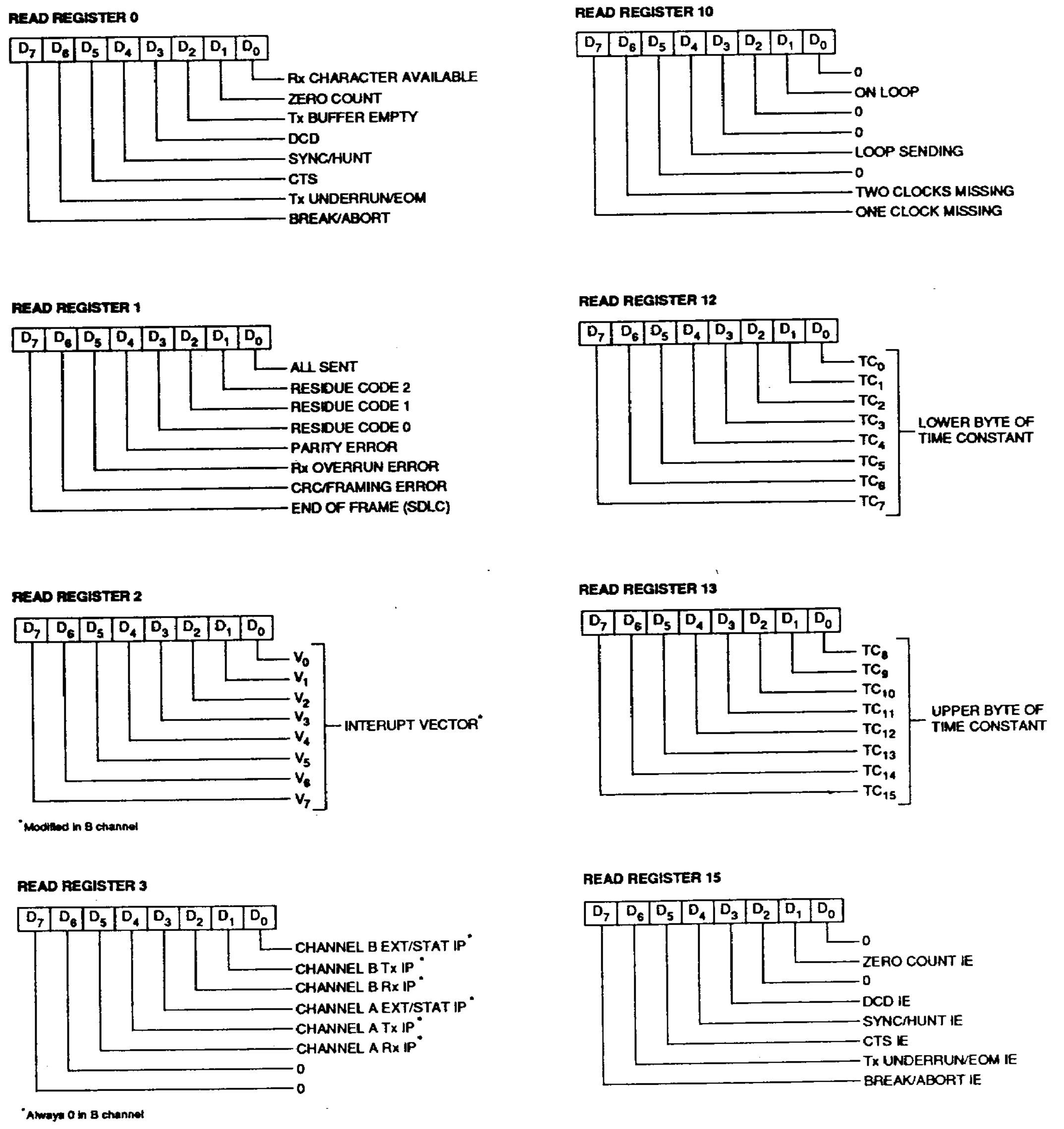
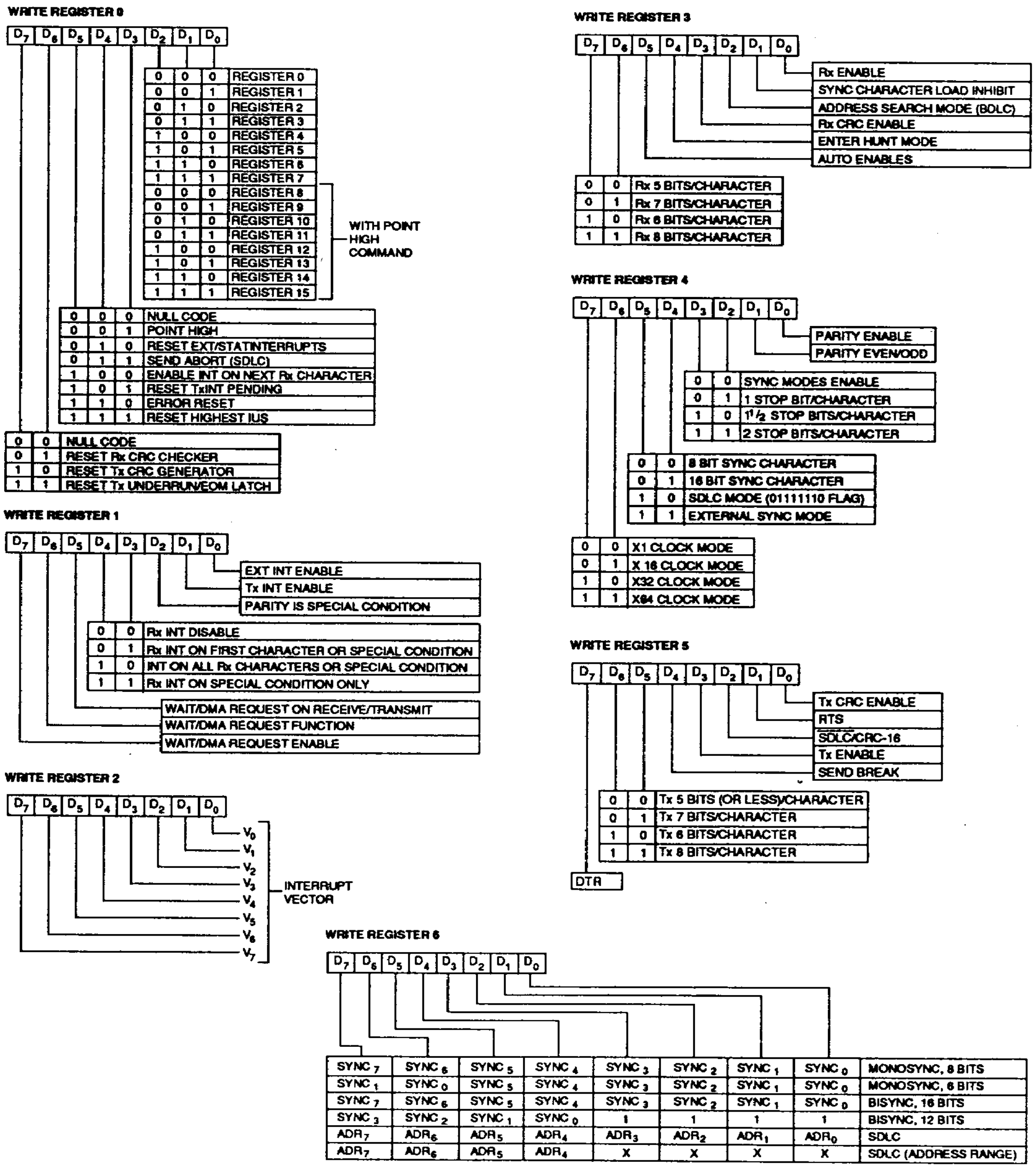


Figure 14 : READ REGISTER BIT FUNCTIONS

6



WRITE REGISTER 0

D2	D1	D0	
0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7
0	0	0	REGISTER 8
0	0	1	REGISTER 9
0	1	0	REGISTER 10
0	1	1	REGISTER 11
1	0	0	REGISTER 12
1	0	1	REGISTER 13
1	1	0	REGISTER 14
1	1	1	REGISTER 15

WITH POINT HIGH COMMAND

D5	D4	D3	
0	0	0	NULL CODE
0	0	1	POINT HIGH
0	1	0	RESET EXT/STAT INTERRUPTS
0	1	1	SEND ABORT (SDLC)
1	0	0	ENABLE INT ON NEXT Rx CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RESET HIGHEST IUS

D7	D6	
0	0	NULL CODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 1

Bit	
D0	EXT INT ENABLE
D1	Tx INT ENABLE
D2	PARITY IS SPECIAL CONDITION
D5	WAIT/DMA REQUEST ON RECEIVE/TRANSMIT
D6	WAIT/DMA REQUEST FUNCTION
D7	WAIT/DMA REQUEST ENABLE

D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER OR SPECIAL CONDITION
1	0	INT ON ALL Rx CHARACTERS OR SPECIAL CONDITION
1	1	Rx INT ON SPECIAL CONDITION ONLY

WRITE REGISTER 2

Bit	
D0	V_0
D1	V_1
D2	V_2
D3	V_3
D4	V_4
D5	V_5
D6	V_6
D7	V_7

INTERRUPT VECTOR

WRITE REGISTER 3

Bit	
D0	Rx ENABLE
D1	SYNC CHARACTER LOAD INHIBIT
D2	ADDRESS SEARCH MODE (SDLC)
D3	Rx CRC ENABLE
D4	ENTER HUNT MODE
D5	AUTO ENABLES

D7	D6	
0	0	Rx 5 BITS/CHARACTER
0	1	Rx 7 BITS/CHARACTER
1	0	Rx 6 BITS/CHARACTER
1	1	Rx 8 BITS/CHARACTER

WRITE REGISTER 4

Bit	
D0	PARITY ENABLE
D1	PARITY EVEN/ODD

D3	D2	
0	0	SYNC MODES ENABLE
0	1	1 STOP BIT/CHARACTER
1	0	1½ STOP BITS/CHARACTER
1	1	2 STOP BITS/CHARACTER

D5	D4	
0	0	8 BIT SYNC CHARACTER
0	1	16 BIT SYNC CHARACTER
1	0	SDLC MODE (01111110 FLAG)
1	1	EXTERNAL SYNC MODE

D7	D6	
0	0	X1 CLOCK MODE
0	1	X 16 CLOCK MODE
1	0	X32 CLOCK MODE
1	1	X64 CLOCK MODE

WRITE REGISTER 5

Bit	
D0	Tx CRC ENABLE
D1	RTS
D2	SDLC/CRC-16
D3	Tx ENABLE
D4	SEND BREAK
D7	DTR

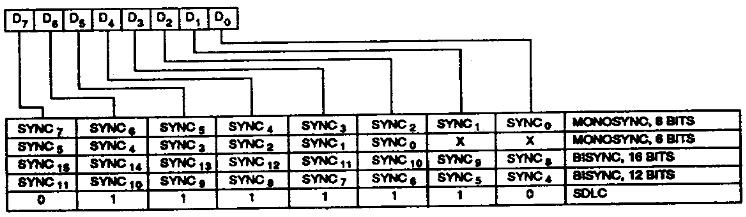
D6	D5	
0	0	Tx 5 BITS (OR LESS)/CHARACTER
0	1	Tx 7 BITS/CHARACTER
1	0	Tx 6 BITS/CHARACTER
1	1	Tx 8 BITS/CHARACTER

WRITE REGISTER 6

D7	D6	D5	D4	D3	D2	D1	D0	
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 8 BITS
$SYNC_1$	$SYNC_0$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 6 BITS
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	BISYNC, 16 BITS
$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	1	1	1	1	BISYNC, 12 BITS
ADR_7	ADR_6	ADR_5	ADR_4	ADR_3	ADR_2	ADR_1	ADR_0	SDLC
ADR_7	ADR_6	ADR_5	ADR_4	X	X	X	X	SDLC (ADDRESS RANGE)

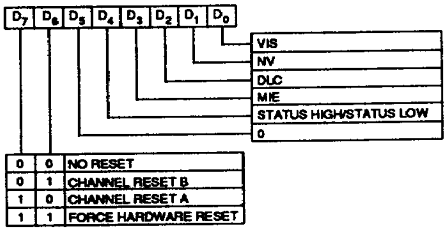
Figure 15 : WRITE REGISTER BIT FUNCTIONS

WRITE REGISTER 7



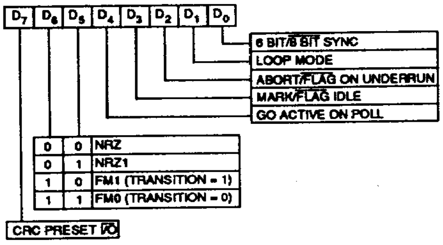
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	MONOSYNC, 8 BITS
$SYNC_5$	$SYNC_4$	$SYNC_3$	$SYNC_2$	$SYNC_1$	$SYNC_0$	X	X	MONOSYNC, 6 BITS
$SYNC_{15}$	$SYNC_{14}$	$SYNC_{13}$	$SYNC_{12}$	$SYNC_{11}$	$SYNC_{10}$	$SYNC_9$	$SYNC_8$	BISYNC, 16 BITS
$SYNC_{11}$	$SYNC_{10}$	$SYNC_9$	$SYNC_8$	$SYNC_7$	$SYNC_6$	$SYNC_5$	$SYNC_4$	BISYNC, 12 BITS
0	1	1	1	1	1	1	0	SDLC

WRITE REGISTER 9



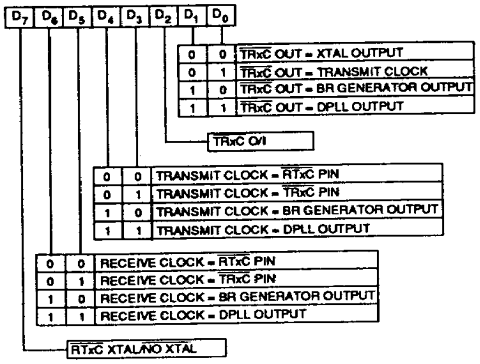
D_7	D_6	
0	0	NO RESET
0	1	CHANNEL RESET B
1	0	CHANNEL RESET A
1	1	FORCE HARDWARE RESET

WRITE REGISTER 10


D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
6 BIT/$\overline{8}$ BIT SYNC
LOOP MODE
ABORT/$\overline{\text{FLAG}}$ ON UNDERRUN
MARK/$\overline{\text{FLAG}}$ IDLE
GO ACTIVE ON POLL
CRC PRESET I/$\overline{\text{O}}$

D_6	D_5	
0	0	NRZ
0	1	NRZ1
1	0	FM1 (TRANSITION = 1)
1	1	FM0 (TRANSITION = 0)

WRITE REGISTER 11


D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
$\overline{\text{TRxC}}$ O/I
$\overline{\text{RTxC}}$ XTAL/NO XTAL

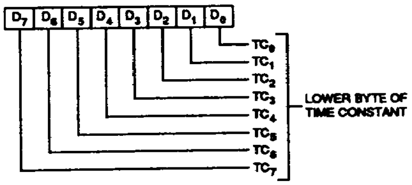
D_1	D_0	
0	0	$\overline{\text{TRxC}}$ OUT = XTAL OUTPUT
0	1	$\overline{\text{TRxC}}$ OUT = TRANSMIT CLOCK
1	0	$\overline{\text{TRxC}}$ OUT = BR GENERATOR OUTPUT
1	1	$\overline{\text{TRxC}}$ OUT = DPLL OUTPUT

D_4	D_3	
0	0	TRANSMIT CLOCK = $\overline{\text{RTxC}}$ PIN
0	1	TRANSMIT CLOCK = $\overline{\text{TRxC}}$ PIN
1	0	TRANSMIT CLOCK = BR GENERATOR OUTPUT
1	1	TRANSMIT CLOCK = DPLL OUTPUT

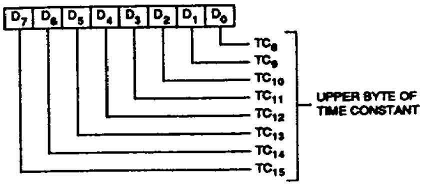
D_6	D_5	
0	0	RECEIVE CLOCK = $\overline{\text{RTxC}}$ PIN
0	1	RECEIVE CLOCK = $\overline{\text{TRxC}}$ PIN
1	0	RECEIVE CLOCK = BR GENERATOR OUTPUT
1	1	RECEIVE CLOCK = DPLL OUTPUT

Figure 15 : WRITE REGISTER BIT FUNCTIONS con't

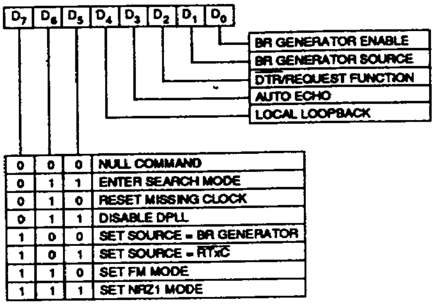
WRITE REGISTER 12


D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
TC_0
TC_1
TC_2
TC_3
TC_4
TC_5
TC_6
TC_7
LOWER BYTE OF TIME CONSTANT

WRITE REGISTER 13

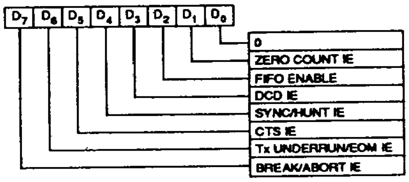

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
TC_8
TC_9
TC_{10}
TC_{11}
TC_{12}
TC_{13}
TC_{14}
TC_{15}
UPPER BYTE OF TIME CONSTANT

WRITE REGISTER 14



D_7	D_6	D_5	
0	0	0	NULL COMMAND
0	1	1	ENTER SEARCH MODE
0	1	0	RESET MISSING CLOCK
0	1	1	DISABLE DPLL
1	0	0	SET SOURCE = BR GENERATOR
1	0	1	SET SOURCE = $\overline{\text{RTxC}}$
1	1	0	SET FM MODE
1	1	1	SET NRZ1 MODE

WRITE REGISTER 15



FIFO

FIFO Enhancements

Used with a DMA controller, the enhanced CA85C30 FIFO maximizes the ability of the SCC to receive high speed back-to-back SDLC messages while minimizing frame overruns due to CPU latencies inresponding to interrupts.

Additional logic was added to the industry standard NMOS SCC, including a 10 deep by 19 bit status FIFO, 14-bit receive byte counter, and control logic. Note that the 10 x 19 bit status FIFO is separate from the existing three byte receive data FIFO.

When the enhancement is enabled, the status in read register 1 (RR1) and byte count for the SDLC frame will be stored in the 10 x 19 bit status FIFO. This allows the DMA controller to transfer the next frame into memory while the CPU verifies that the message was properly received.

Summarizing the operation, data is received, assembled, loaded into the three byte receive FIFO before being transferred to memory by the DMA controller. When a flag is received at the end of an SDLC frame, the frame byte count from the 14-bit counter and five status bits are loaded into the status FIFO for verification by the CPU. The CRC checker is automatically reset in preparation for the next frame which can begin immediately. Since the byte count and status are saved for each frame, the message integrity can be verified at a later time. Status information for up to 10 frames can be stored before a status FIFO overrun could occur. Refer to the block diagram in Figure 16.

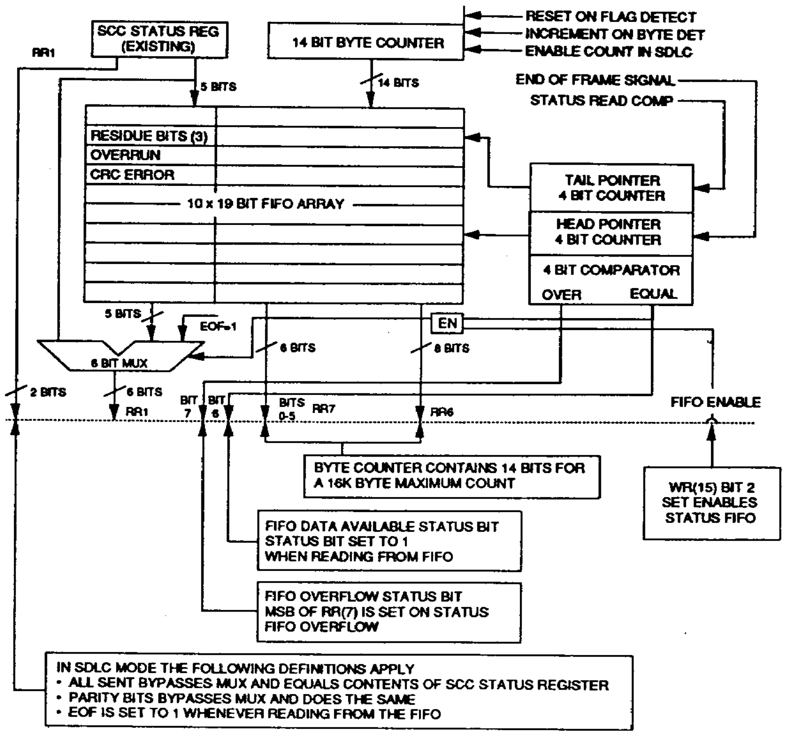

SCC STATUS REG (EXISTING)
RR1
14 BIT BYTE COUNTER
RESET ON FLAG DETECT
INCREMENT ON BYTE DET
ENABLE COUNT IN SDLC
5 BITS
14 BITS
END OF FRAME SIGNAL
STATUS READ COMP
RESIDUE BITS (3)
OVERRUN
CRC ERROR
10 x 19 BIT FIFO ARRAY
TAIL POINTER 4 BIT COUNTER
HEAD POINTER 4 BIT COUNTER
4 BIT COMPARATOR
OVER
EQUAL
5 BITS
EOF=1
EN
6 BITS
8 BITS
6 BIT MUX
2 BITS
6 BITS
RR1
BIT 7
BIT 6
BITS 0-5
RR7
RR6
FIFO ENABLE
BYTE COUNTER CONTAINS 14 BITS FOR A 16K BYTE MAXIMUM COUNT
WR(15) BIT 2 SET ENABLES STATUS FIFO
FIFO DATA AVAILABLE STATUS BIT STATUS BIT SET TO 1 WHEN READING FROM FIFO
FIFO OVERFLOW STATUS BIT MSB OF RR(7) IS SET ON STATUS FIFO OVERFLOW
IN SDLC MODE THE FOLLOWING DEFINITIONS APPLY
• ALL SENT BYPASSES MUX AND EQUALS CONTENTS OF SCC STATUS REGISTER
• PARITY BITS BYPASSES MUX AND DOES THE SAME
• EOF IS SET TO 1 WHENEVER READING FROM THE FIFO

Figure 16 : SCC STATUS REGISTER MODIFICATIONS

Enable/Disable

The FIFO is enabled when WR15 bit 2 is set and the SCC is in SDLC/HDLC mode. Otherwise the status register contents bypass the FIFO and go directly to the bus interface (FIFO pointer logic is reset when disabled or via a channel or power-on reset). When the FIFO mode is disabled, the SCC is completely downwards-compatible with the NMOS 8530. The FIFO mode is disabled on power-up (WR15 bit 2 is set to 0 on reset). The effects of backward compatibility on the register set are that RR4 is an image of RR0, RR5 is an image of RR1, RR6 is an image of RR2 and RR7 is an image of RR3. For details of the added registers, refer to Figure 18. The status of the FIFO Enable signal can be obtained by reading RR15 bit 2. If the FIFO is enabled, the bit will be set to 1, otherwise it will be reset.

Read Operation

When WR15 bit 2 is set and the FIFO is not empty, the next read to any of status register RR1 or the additional registers RR6 and RR7 will actually be from the FIFO. Reading status register RR1 causes one location of the FIFO to be emptied, so status should be read after reading the byte count, otherwise the count will be incorrect. Before the FIFO underflows, it is disabled. In this case, the multiplexer is switched to allow status to read directly from the status register, and reads from RR7 and RR6 will contain bits that are undefined. Bit 6 of RR7 (FIFO Data Available) can be used to determine if status data is coming from the FIFO or directly from the status register, since it is set to 1 whenever the FIFO is not empty.

Since not all status bits must be stored in the FIFO, the All Sent, Parity, and EOF bits will bypass the FIFO. The status bits sent through the FIFO will be Residue Bits (3), Overrun, and CRC Error.

The sequence for proper operation of the byte count and FIFO logic is to read the registers in the following order; RR7, RR6 and RR1 (reading RR6 is optional). Additional logic prevents the FIFO from being emptied by multiple reads from RR1. The read from RR7 latches trhe FIFO empty/full status bit (bit 6) and steers the status multiplexer to read from the SCC megacell instead of the status FIFO (since the status FIFO is empty). The read from RR1 allows an entry to be read from the FIFO (if the FIFO was empty, logic is added to prevent a FIFO underflow condition).

Write Operation

When the end of an SDLC frame (EOF) has been received and the FIFO is enabled, the contents of the status and byte-count registers are loaded into the FIFO. The EOF signal is used to increment the FIFO. If the FIFO overflows, the MSB of RR7 (FIFO Overflow) is set to indicate the overflow. This bit and the FIFO control logic is reset by disabling and re-enabling the FIFO control bit (WR15 bit 2). Details of FIFO control timing during an SDLC frame are shown in Figure 17.

Byte Counter Detail

The 14-bit byte counter allows for packets up to 16K bytes to be received, (see Figures 16 and 17).

Enable

The byte counter is enabled in the SDLC/HDLC mode.

Reset

The byte counter is reset whenever an SDLC flag character is received. The reset is timed so that the contents of the byte counter are successfully written into the FIFO.

Increment

The byte counter is incremented by writes to the data FIFO. The counter represents the number of bytes received by the SCC, rather than the number of bytes transferred from the SCC. (These counts may differ by up to the number of bytes in the receive data FIFO contained in the SCC).

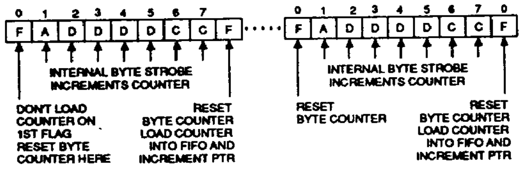

0 1 2 3 4 5 6 7
F A D D D D C C F
0 1 2 3 4 5 6 7 0
F A D D D D C C F
INTERNAL BYTE STROBE INCREMENTS COUNTER
DONT LOAD COUNTER ON 1ST FLAG RESET BYTE COUNTER HERE
RESET BYTE COUNTER LOAD COUNTER INTO FIFO AND INCREMENT PTR
INTERNAL BYTE STROBE INCREMENTS COUNTER
RESET BYTE COUNTER
RESET BYTE COUNTER LOAD COUNTER INTO FIFO AND INCREMENT PTR

Figure 17 : SDLC BYTE COUNTING DETAIL

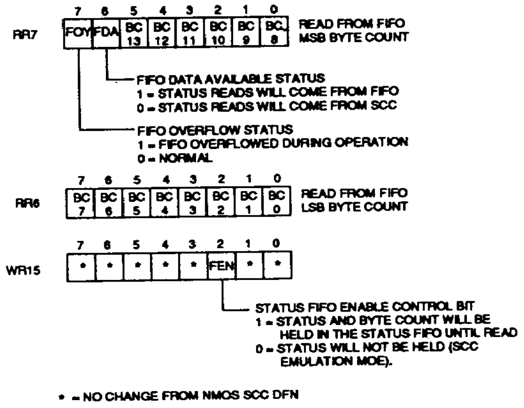

7 6 5 4 3 2 1 0
RR7
FOY FDA BC 13 BC 12 BC 11 BC 10 BC 9 BC 8
READ FROM FIFO MSB BYTE COUNT
FIFO DATA AVAILABLE STATUS
1 = STATUS READS WILL COME FROM FIFO
0 = STATUS READS WILL COME FROM SCC
FIFO OVERFLOW STATUS
1 = FIFO OVERFLOWED DURING OPERATION
0 = NORMAL
RR6
BC 7 BC 6 BC 5 BC 4 BC 3 BC 2 BC 1 BC 0
READ FROM FIFO LSB BYTE COUNT
WR15
* * * * * FEN * *
STATUS FIFO ENABLE CONTROL BIT
1 = STATUS AND BYTE COUNT WILL BE HELD IN THE STATUS FIFO UNTIL READ
0 = STATUS WILL NOT BE HELD (SCC EMULATION MOE).
* = NO CHANGE FROM NMOS SCC DFN

Figure 18 : SCC ADDITIONAL REGISTERS