CML Microcircuits

COMMUNICATION SEMICONDUCTORS

CMX910
AIS Baseband Processor

D/910/3 April 2006 Provisional Issue

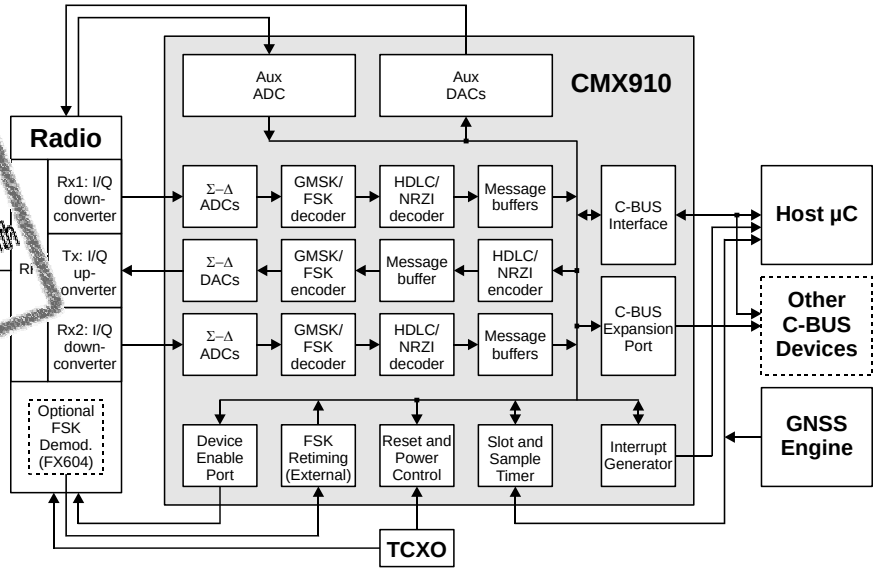
Features:

- Half-Duplex GM(F)SK, FSK and DSC Capabilities
- Slot/Sample Counter with UTC Timing Interface
- Optimum Co-channel and Adjacent-channel Performance
- Flexible Signal Channels
  - Two Simultaneous Rx
  - One Tx
  - Optional FSK Interface
- AIS Data Formatted and Raw Data Modes
- Supports Carrier-Sensing Channel Access (CSTDMA) Operation
- RF Device-Enable Facilities
- C-BUS Serial Interface with Expansion Port
- I and Q Radio Interface
- Low-Power (3.0 to 3.6V) Operation
- Low Profile, 64-lead LQFP (L9) and Leadless VQFN (Q1) Packages
- Auxiliary ADC and DAC Functions
  - 5 x (10-bit) DACs
  - 5-Input MUX (10-bit) ADC

Applications:

- Automatic Identification System (AIS) for Marine Safety
- Class A or B AIS Transponders
- AIS Rx-only Modules

The CMX910 Initialisation Procedure File should be downloaded, from the MyCML portal, for use with this product



1. Brief Description

A highly integrated Baseband Signalling Processor IC, the CMX910 fulfils the requirements of the class A and class B marine Automatic Identification System (AIS) transponder market. The CMX910 is half duplex in operation, comprising two parallel I+Q Rx paths and one Tx path. These are configurable for AIS or DSC operation. The device performs channel filtering and signal modulation/demodulation with associated AIS functions, such as training sequence detection, NRZI conversion and HDLC processing (flags, bit stuffing/de-stuffing, CRC generate/check). An external 1200bps FSK demodulator interface provides a third parallel decode path for DSC, as required by the class A market. Integrated Rx/Tx data buffers and a flexible slot/sample timer are also provided, all of which greatly reduce the processing requirements of the host µC. Provision of a C-BUS expansion port, an RF device enable port and a number of auxiliary ADCs and DACs further simplifies the system hardware design, reducing the overall equipment cost and size.

© 2006 CML Microsystems Plc

CONTENTS

It is always recommended that you check for the latest product datasheet version from the Datasheets page of the CML website: [www.cmlmicro.com].

2. Block Diagram

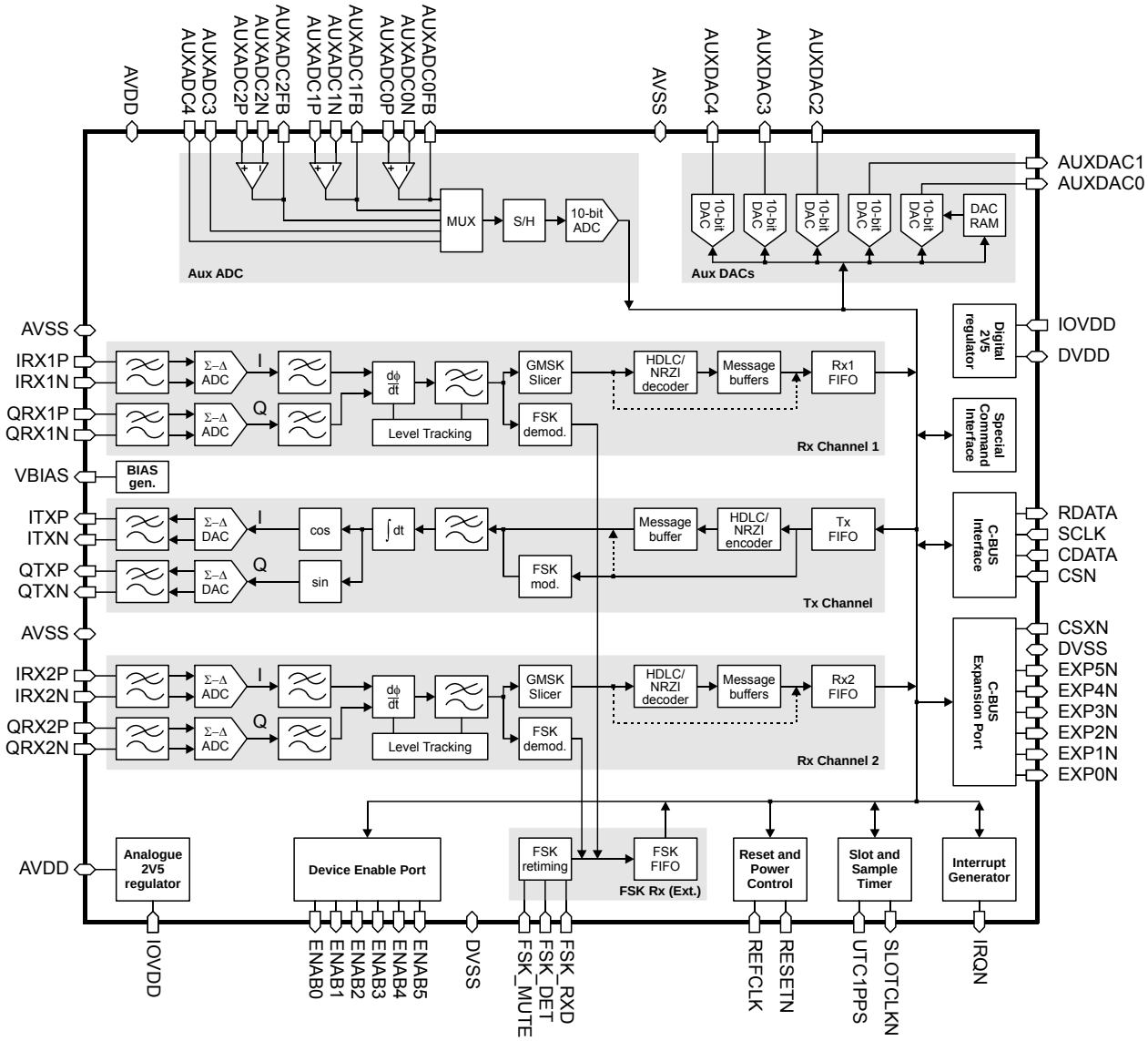


Figure 1 CMX910 Block Diagram

3. Signal List

Package Q1 or L9	Signal		Description
Pin No.	**Name**	**Type**	
1	AV_{SS}	Power	Analogue negative supply rail (ground)
2	IRX1P	I/P	Receive “I” channel 1, positive input
3	IRX1N	I/P	Receive “I” channel 1, negative input
4	QRX1P	I/P	Receive “Q” channel 1, positive input
5	QRX1N	I/P	Receive “Q” channel 1, negative input
6	VBIAS	O/P	A bias line for the internal circuitry, held at ½ AV_{DD}. This pin must be decoupled to AV_{SS} by a capacitor mounted close to the device pins
7	ITXP	O/P	Transmit “I” channel, positive output
8	ITXN	O/P	Transmit “I” channel, negative output
9	QTXP	O/P	Transmit “Q” channel, positive output
10	QTXN	O/P	Transmit “Q” channel, negative output
11	AV_{SS}	Power	Analogue negative supply rail (ground)
12	IRX2P	I/P	Receive “I” channel 2, positive input
13	IRX2N	I/P	Receive “I” channel 2, negative input
14	QRX2P	I/P	Receive “Q” channel 2, positive input
15	QRX2N	I/P	Receive “Q” channel 2, negative input
16	AV_{DD}	Power	Analogue positive supply rail. Decouple to AV_{SS}
17	IOV_{DD}	Power	Digital I/O positive supply rail. Decouple to DV_{SS}
18	ENAB0	O/P	Enable output 0
19	ENAB1	O/P	Enable output 1
20	ENAB2	O/P	Enable output 2
21	ENAB3	O/P	Enable output 3
22	ENAB4	O/P	Enable output 4
23	ENAB5	O/P	Enable output 5
24	DV_{SS}	Power	Digital negative supply rail (ground)
25	FSK_MUTE	I/P	FSK RF squelch indicator
26	FSK_DET	I/P	FSK baseband energy detect indicator
27	FSK_RXD	I/P	Raw FSK demodulator input data
28	REFCLK	I/P	Master input clock (multiple of 2.4MHz)
29	RESETN	I/P	Active low chip reset
30	UTC1PPS	I/P	1Hz UTC sync pulse, typically from GNSS receiver
31	SLOTCLKN	O/P	Slot clock output (active low), pulses at the start of each AIS slot. Configurable as a ‘wire-ORable’ output, requiring an external pullup resistor, or as an active pullup/pulldown.
32	IRQN	O/P	A ‘wire-ORable’ output for connection to the host µC's Interrupt Request input. This output has a low impedance pull down to DV_{SS} when active and is high impedance when inactive. An external pullup resistor is required.

Package Q1 or L9	Signal		Description
Pin No.	**Name**	**Type**	
33	EXP0N	O/P	Chip Select expansion output 0 (active low)
34	EXP1N	O/P	Chip Select expansion output 1 (active low)
35	EXP2N	O/P	Chip Select expansion output 2 (active low)
36	EXP3N	O/P	Chip Select expansion output 3 (active low)
37	EXP4N	O/P	Chip Select expansion output 4 (active low)
38	EXP5N	O/P	Chip Select expansion output 5 (active low)
39	DV_{SS}	Power	Digital negative supply rail (ground)
40	CSXN	I/P	Chip Select expansion input (active low)
41	CSN	I/P	Chip Select input (active low), used to enable a C-BUS data read or write operation on the chip.
42	CDATA	I/P	The C-BUS serial data input from the µC.
43	SCLK	I/P	The C-BUS serial clock input from the µC.
44	RDATA	T/S	A 3-state C-BUS serial data output to the µC. This output is high impedance when not sending data to the µC.
45	DV_{DD}	Power	Digital core positive supply rail. Decouple to DV_{SS}
46	IOV_{DD}	Power	Digital I/O positive supply rail. Decouple to DV_{SS}
47	AUXDAC0	O/P	Auxiliary D-to-A converter output 0 (with ramp)
48	AUXDAC1	O/P	Auxiliary D-to-A converter output 1
49	AUXDAC2	O/P	Auxiliary D-to-A converter output 2
50	AUXDAC3	O/P	Auxiliary D-to-A converter output 3
51	AUXDAC4	O/P	Auxiliary D-to-A converter output 4
52	AV_{SS}	Power	Analogue negative supply rail (ground)
53	AUXADC0FB	O/P	Auxiliary A-to-D converter 0, amplifier feedback
54	AUXADC0N	I/P	Auxiliary A-to-D converter 0, amplifier -ve input
55	AUXADC0P	I/P	Auxiliary A-to-D converter 0, amplifier +ve input
56	AUXADC1FB	O/P	Auxiliary A-to-D converter 1, amplifier feedback
57	AUXADC1N	I/P	Auxiliary A-to-D converter 1, amplifier -ve input
58	AUXADC1P	I/P	Auxiliary A-to-D converter 1, amplifier +ve input
59	AUXADC2FB	O/P	Auxiliary A-to-D converter 2, amplifier feedback
60	AUXADC2N	I/P	Auxiliary A-to-D converter 2, amplifier -ve input
61	AUXADC2P	I/P	Auxiliary A-to-D converter 2, amplifier +ve input
62	AUXADC3	I/P	Auxiliary A-to-D converter input 3
63	AUXADC4	I/P	Auxiliary A-to-D converter input 4
64	AV_{DD}	Power	Analogue positive supply rail. Decouple to AV_{SS}

Notes:
I/P = Input
O/P = Output
BI = Bidirectional
T/S = 3-state Output
NC = No Connection

4. External Components

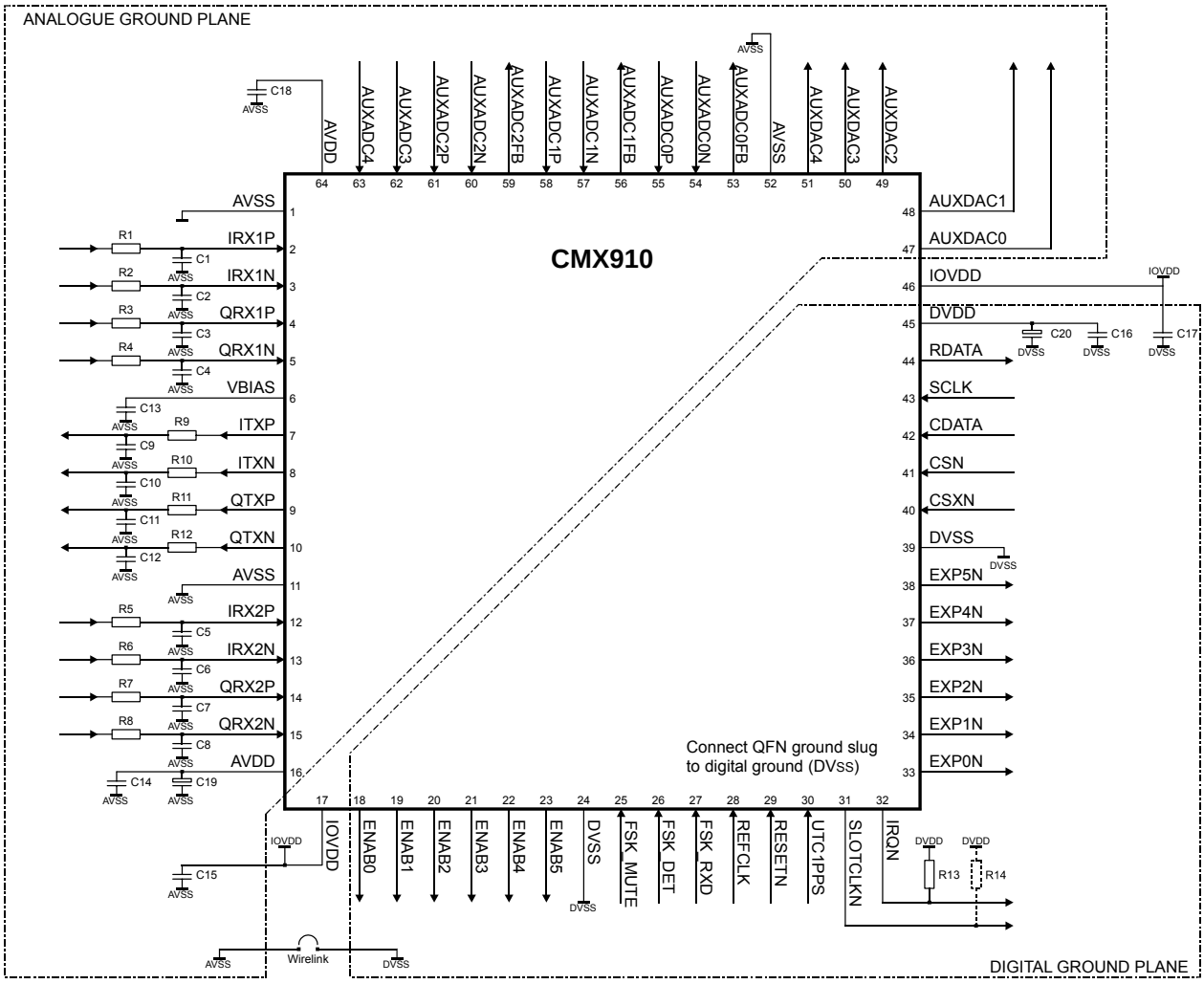


Figure 2 Recommended External Components

C1-C8	1nF	C14-C18	10nF	R1-R8	1kΩ
C9-C12	1nF	C19	10µF	R9-R12	1kΩ
C13	1µF	C20	22µF	R13-R14	10kΩ

A regulated 3.3V supply must be connected to the CMX910's IOV_{DD} pins in order to supply the digital I/O pad circuitry. This 3.3V supply is also used by two on-chip 2.5V regulators which are used to generate the separate AV_{DD} and DV_{DD} supplies (used by the CMX910's on-chip analogue and digital circuitry), the only external components needed are the decoupling capacitors. The AV_{DD} and DV_{DD} supplies are *not* intended to provide current to any external circuits, but may be buffered if required for use as a reference.

To achieve good noise performance, AV_{DD}/DV_{DD} and VBIAS decoupling and protection of the receive path from extraneous in-band signals are very important. It is recommended that the printed circuit board is laid out with separate analogue and digital ground planes in the CMX910 area to provide a low impedance connection between AV_{SS} and the AV_{DD}/VBIAS decoupling capacitors, and between DV_{SS} and the DV_{DD} decoupling capacitors.

5. General Description

5.1 Overview of CMX910 Operation

The CMX910 IC has two main receiver circuits that support simultaneous reception of two AIS channels, or one AIS and one DSC channel. When the two main receive channels are configured for AIS reception, a supplementary DSC-only receiver can be supported by using a separate external FSK demodulator (such as the FX604) interfaced to the CMX910. Data bits received on this FSK interface get packed into bytes and passed through to the host µC. Transmission on a single channel (AIS or DSC) is supported, during which all reception ceases. The main receive and transmit channels use differential I + Q signalling, and digital filtering and signal processing techniques are used to obtain a high level of performance.

The CMX910 also supports the proposed carrier-sensing channel access scheme (CSTDMA) for the AIS class B standard.

Overall timing synchronisation centres around two counters, one counting the number of samples per slot, the other the number of slots in a minute. These allow the µC to retrieve slot and sample timing information for any received message as well as specifying transmit timing accurately. Counters can be synchronised to an external 1 pulse per second UTC reference signal or allowed to run free, in which case they must be kept in alignment by the µC.

The CMX910 offers full frame-formatting (HDLC-type) support for AIS receive and transmit, including NRZI coding, bit stuffing and de-stuffing, training sequence, start/stop flag insertion and deletion, and CRC generation and checking. A raw mode is also provided allowing greater flexibility. DSC transmission and reception is only supported in the equivalent of the AIS raw mode.

The transmit channel and three receive channels interface to the µC through individual 32-deep, byte wide first-in first-out data buffers. These alleviate latency problems and allow higher data rates between the µC and the CMX910. To allow system performance to be further enhanced, the CMX910 can be configured to cause an interrupt if the transmit FIFO fill level drops below a user-defined threshold or any of the receive FIFOs' fill levels exceed a user-defined threshold.

The CMX910 also assists with power saving and control of critical external RF circuits by providing a flexible device enable port. Further monitoring and control functions can be implemented using the integrated auxiliary ADC and DAC circuits.

Communication between the CMX910 and the host µC is done through CML's standard C-BUS interface and interrupt pin. The C-BUS interface is SPI compatible, and can be driven by an SPI master controller. A C-BUS expansion port is also provided which allows the connection of up to six further C-BUS or SPI compatible devices to the µC.

© 2006 CML Microsystems Plc

5.2 C-BUS Interface

This block provides for the transfer of data and control or status information between the CMX910's internal registers and the host µC over the C-BUS serial bus. Each transaction consists of a single Register Address byte sent from the µC which may be followed by a data word sent from the µC to be written into one of the CMX910's Write Only Registers, or a data word read out from one of the CMX910's Read Only Registers; all C-BUS data words are a multiple of 8 bits wide, the width depending on the source or destination register. Note that certain C-BUS transactions require only an address byte to be sent from the µC, no data transfer being required. The operation of the C-BUS is illustrated in Figure 3.

Data sent from the µC on the CDATA (command data) line is clocked into the CMX910 on the rising edge of the SCLK input. Data sent from the CMX910 to the µC on the RDATA (reply data) line is valid when SCLK is high. The CSN line must be held low during a data transfer and kept high between transfers. The C-BUS interface is compatible with most common µC serial interfaces and may also be easily implemented with general purpose µC I/O pins controlled by a simple software routine. Figure 14 gives detailed C-BUS timing requirements.

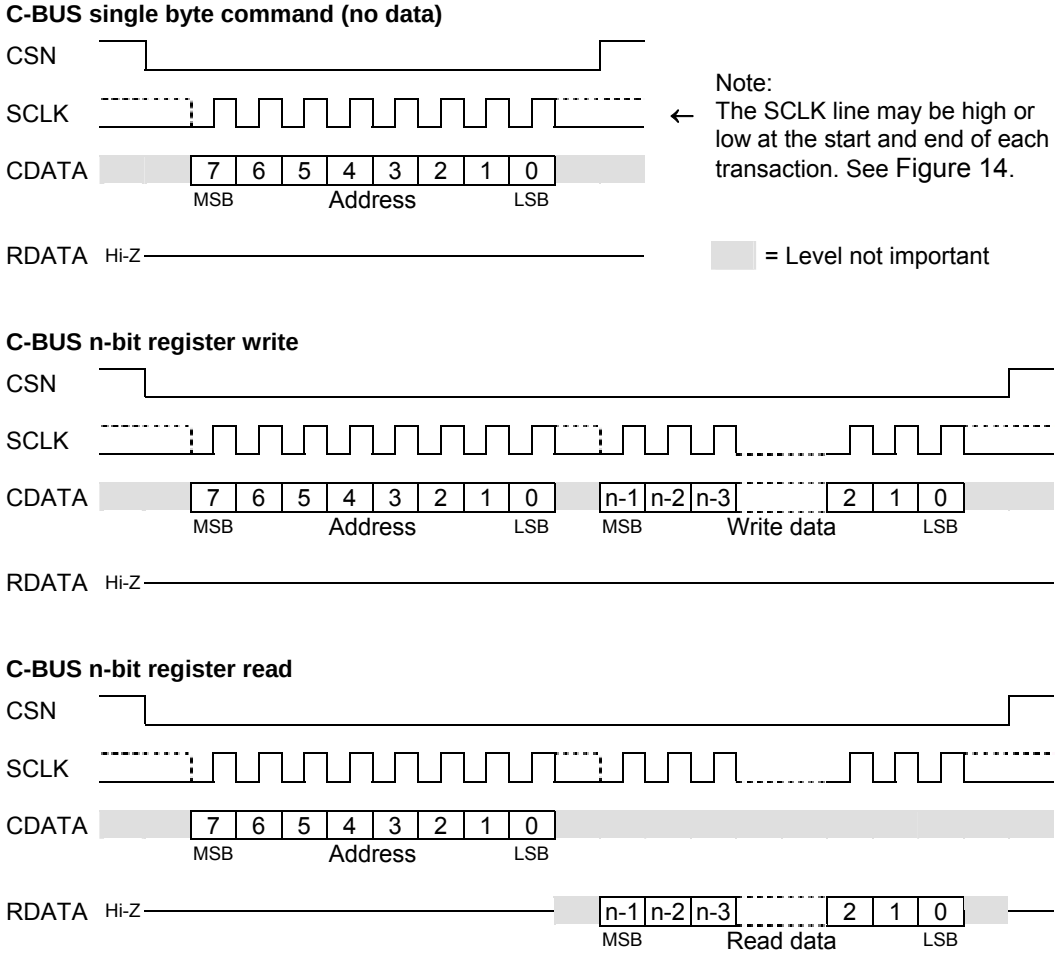


Figure 3 Basic C-BUS Transactions

To increase the data bandwidth between the µC and the CMX910, certain of the C-BUS read and write registers are capable of data-streaming operation. This allows a single address byte to be followed by the transfer of multiple read or write data words, all within the same C-BUS transaction. This can significantly increase the transfer rate of large data blocks, as shown in Figure 4.

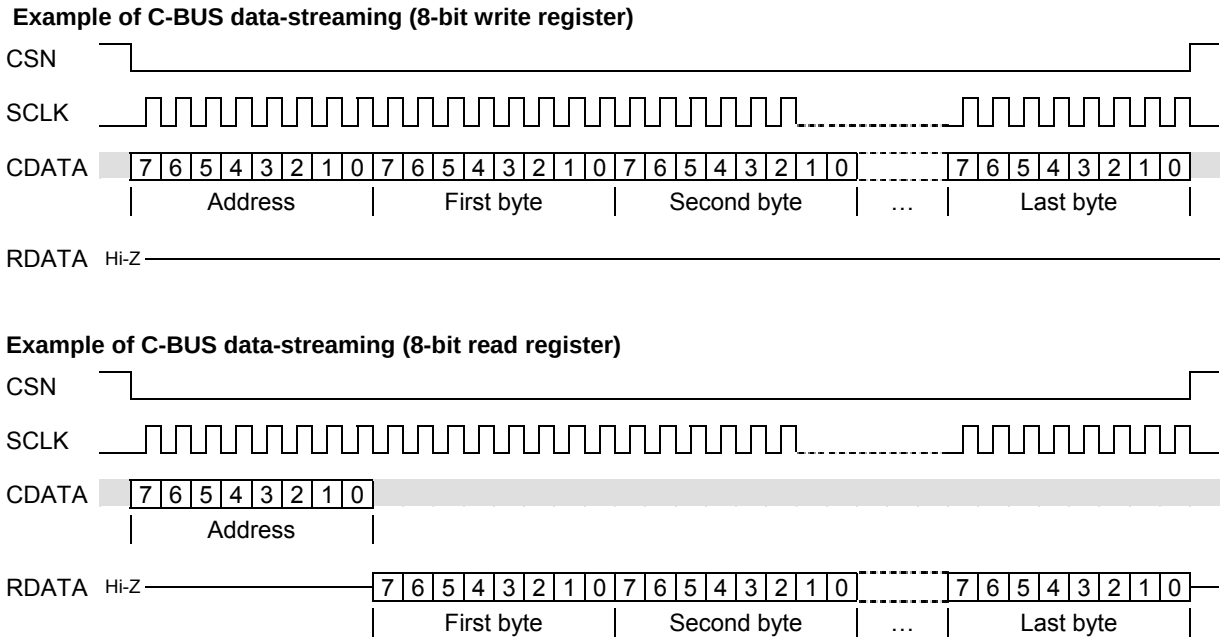


Figure 4 C-BUS Data-Streaming Operation

A summary of the CMX910's C-BUS addresses and registers are shown in Table 1. Note: the CMX910's internal clock must be running before any C-BUS access is attempted, with the exception of the General_Reset command and the Clock_Control and CBUS_Expand registers.

C-BUS register name	Addr	R/W/ Cmd	Size	C-BUS register name	Addr	R/W/ Cmd	Size
Reset and power control				*FSK Interface*			
General_Reset	$01	Cmd	-	FSK_FIFO (DS)	$50	R	8
Clock_Control	$02	W	8	FSK_FIFO_Threshold	$51	W	8
Slot and Sample Timer				FSK_Status	$52	R	16
Slot_Sample_Control	$10	W	8	FSK_Control	$53	W	8
Slot_Sample_Count	$11	R	32	*Auxiliary ADC*			
Sleep_Sample	$12	W	16	ADC0	$60	R	16
Wakeup_Sample	$13	W	16	ADC1	$61	R	16
Slot_Sample_UTC1PPS	$14	R	32	ADC2	$62	R	16
Slot_Nudge	$15	W	16	ADC3	$63	R	16
Sample_Nudge	$16	W	16	ADC4	$64	R	16
Nudge_Trigger	$17	W	16	ADC_Control1	$65	W	8
Max_Auto_Nudge	$18	W	16	ADC_Control2	$66	W	8
Transmit Channel				ADC_Status	$67	R	8
Tx_FIFO (DS)	$20	W	8	ADC_Convert	$68	Cmd	-
Tx_FIFO_Threshold	$21	W	8	*Auxiliary DACs*			
Tx_Status	$22	R	16	DAC0	$70	W	16
Tx_Slot	$23	W	16	DAC1	$71	W	16
Tx_Bits	$24	W	16	DAC2	$72	W	16
Tx_Control	$25	W	16	DAC2	$73	W	16
CSTDMA_Threshold	$26	W	16	DAC4	$74	W	16
Receive Channel 1				DAC_Control	$75	W	8
Rx1_FIFO (DS)	$30	R	8	DAC0_Rampup	$76	Cmd	-
Rx1_FIFO_Threshold	$31	W	8	DAC0_Rampdown	$77	Cmd	-
Rx1_Status	$32	R	16	DAC0_Timestep	$78	W	8
Rx1_Slot	$33	R	16	DAC_RAM_Load (DS)	$79	W	16
Rx1_Sample	$34	R	16	*Interrupts*			
Rx1_Bytes	$35	R	16	Interrupt	$80	R	16
Rx1_Control	$36	W	8	Interrupt_Enable	$81	W	16
Rx1_FreqErr	$37	R	16	*Device Enable Port*			
Rx1_RSSI	$38	R	16	ENAB	$90	W	8
Receive Channel 2				ENAB_Mask	$91	W	8
Rx2_FIFO (DS)	$40	R	8	ENAB_Invert	$92	W	8
Rx2_FIFO_Threshold	$41	W	8	*C-BUS Expansion Port*			
Rx2_Status	$42	R	16	CBUS_Expand	$A0	W	8
Rx2_Slot	$43	R	16	*Special Command Interface*			
Rx2_Sample	$44	R	16	SPC_In0	$B0	W	16
Rx2_Bytes	$45	R	16	SPC_In1	$B1	W	16
Rx2_Control	$46	W	8	SPC_Out0	$B2	R	16
Rx2_FreqErr	$47	R	16	Special_Command	$B4	W	8
Rx2_RSSI	$48	R	16				

(DS) - These registers are capable of data-streaming transactions.
Note: C-BUS addresses $F0 to $FE are allocated for production testing and should not be accessed in normal operation.

Table 1 Summary of C-BUS Registers

5.3 Reset and Power Control

5.3.1 RESETN pin

The CMX910 is reset by taking RESETN low, which causes all internal clocks and bias currents to be disabled and all C-BUS registers to be reset. To bring the CMX910 out of this quiescent state after RESETN is pulled high, a stable clock signal must first be applied to the REFCLK input pin (any multiple of 2.4MHz up to a maximum of 24MHz), then the Clock_Control register must be programmed with the frequency of the applied REFCLK. A period of 10ms must then elapse to allow the CMX910 to initialise, after which time the device is ready for operation. During operation the main Rx and Tx channel analogue circuits and auxiliary ADC and DAC circuits will be powered up as required, depending on how the host µC sets various C-BUS control and configuration registers.

5.3.2 General_Reset Command

General_Reset command (no data) C-BUS Address $01

This command disables all internal bias currents and resets all C-BUS registers *except* for CBUS_Expand and Clock_Control. This means that if the CMX910's internal clocks are running, they will remain running when General_Reset is applied. After a General_Reset command, a period of 10ms must elapse to allow the CMX910 to initialise before any further C-BUS operations are attempted.

5.3.3 Clock Control

The CMX910 can be put back into a low power state at any time by writing $00 to the Clock_Control register. This will disable all internal clocks and bias currents and reset all internal C-BUS registers *except* for CBUS_Expand. To subsequently bring the CMX910 out of this low power state requires the same sequence of operations as if a RESETN pulse had been applied.

Clock_Control register: 8-bit write only. C-BUS Address $02
All bits cleared to 0 when RESETN pin asserted. Register contents are not affected by a General_Reset command. This register can be written while the CMX910's internal clocks are disabled.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 0000				REFCLK mult. factor			

Clock_Control register b3-0: REFCLK Multiplication Factor

b3	b2	b1	b0	
0	0	0	0	Internal clocks disabled, device held in low power mode
0	0	0	1	REFCLK = 2.4MHz
0	0	1	0	REFCLK = 4.8MHz
0	0	1	1	REFCLK = 7.2MHz
0	1	0	0	REFCLK = 9.6MHz
0	1	0	1	REFCLK = 12.0MHz
0	1	1	0	REFCLK = 14.4MHz
0	1	1	1	REFCLK = 16.8MHz
1	0	0	0	REFCLK = 19.2MHz
1	0	0	1	REFCLK = 21.6MHz
1	0	1	0	REFCLK = 24.0MHz
Codes 1011_2 to 1111_2 are reserved, do not use				

5.4 Slot and Sample Timer

The Slot and Sample Timer circuit contains two counters that are used to control and sequence operations in the three main channels (Rx1, Rx2, Tx).

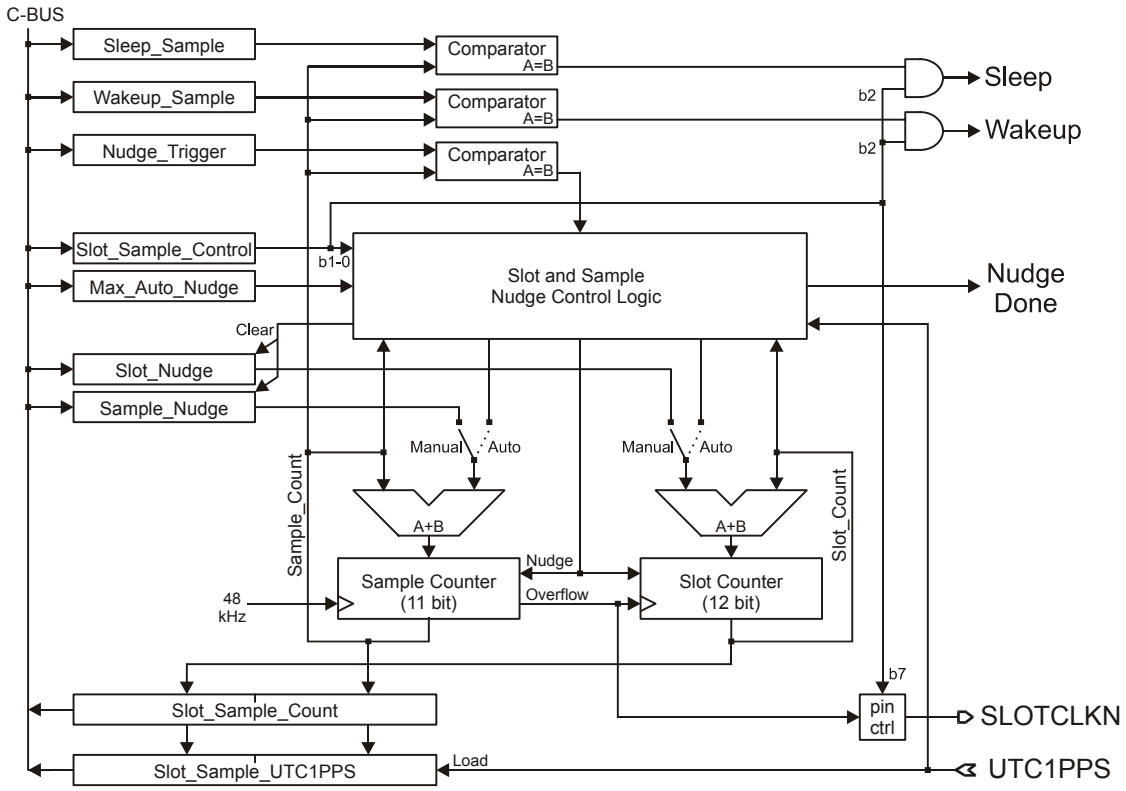


Figure 5 Slot and Sample Timer Circuit

The clock for the slot and sample counters is derived from the REFCLK input pin. The *sample* counter is an 11 bit counter which increments at 48kHz, i.e. five times per AIS data bit, and is used to time various Rx and Tx operations within a slot period. Since there are 256 bit periods per AIS slot, the sample counter increments from 0 to 1279 before rolling over to 0. The *slot* counter is a 12 bit counter and is used to count the slot number in an AIS frame, which lasts for a minute. It is incremented at the beginning of each AIS slot period, i.e. when the sample counter rolls over. There are 37½ slots per second, resulting in 2250 slots per minute. Therefore the slot counter increments from 0 to 2249 before rolling over to 0. When operating correctly, the slot counter rollover should be aligned to the start of the UTC minute. The current value of the slot and sample counters are available to the µC by reading the Slot_Sample_Count register.

The CMX910 produces a pulse on its SLOTCLKN output pin during the first sample period within each slot, this can be used as general timing reference by the µC. Each pulse is active low and lasts for approximately 20.83 µs, and the pulses repeat at 37.5Hz. The signal appearing on the SLOTCLKN pin can be configured to be open-drain pull-down or have active pull-up and pull-down drivers.

When the CMX910 comes out of reset the slot and sample counters will be free running but not synchronised to anything. The µC must synchronise them to an appropriate timing source, either UTC (direct or indirect) or to an appropriate base station as required by Recommendation ITU-R M1371-1. Once initial synchronisation has been established, occasional minor adjustments, or “nudges”, to the sample counter must be made to keep it locked to the chosen timing source – this compensates for any slight drift caused by inaccuracy in the REFCLK frequency. Nudge values can be calculated and applied directly by the µC in a software control loop (“manual nudge”, section 5.4.1). Alternatively, the CMX910 can be configured into certain “auto nudge” modes to establish initial synchronisation and subsequent

tracking of the slot and sample counters with minimal µC intervention, using the UTC1PPS signal as a timing reference (section 5.4.2).

A “Sleep Control” feature is provided which can reduce power consumption significantly when the CMX910 is enabled for AIS reception. This operates by automatically turning off the internal receiver circuits during inactive slots. Sleep Control is described in more detail in section 5.4.3.

The Slot and Sample Timer circuit is configured and controlled through nine C-BUS registers:

Slot_Sample_Control register: 8-bit write only. C-BUS Address $10
Register reset to $80.

Bit:

7	6	5	4	3	2	1	0
Slot clock ctrl	Reserved, set to 0000				En sleep mode	Nudge mode	

Slot_Sample_Control register b7: SLOTCLKN Pin Control
With b7 = 0 the SLOTCLKN pin will be configured to have active pull-up and pull-down drivers. If b7 = 1 the pin will have an open-drain pull-down, requiring an external pull-up resistor.

Slot_Sample_Control register b2: Enable Sleep Mode
Setting b2 = 1 enables AIS sleep mode on receive channels Rx1 and Rx2.

Slot_Sample_Control register b1-0: Nudge Mode
The Nudge Mode bits control how the CMX910 achieves and maintains synchronisation of the slot and sample counters with the UTC timing reference.

b1	b0	
0	0	Manual nudge (auto nudge disabled)
0	1	Auto nudge acquire
1	0	Auto nudge track
1	1	Reserved, do not use

Slot_Sample_Count register: 32-bit read only. C-BUS Address $11
All bits cleared to 0 on reset.

Bit:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	Slot count											

Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	Sample count										

The Slot_Sample_Count register holds the current value of the slot and sample counters.

Sleep_Sample register: 16-bit write only. **C-BUS Address $12**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 00000					Sleep sample value										

The Sleep_Sample register holds the sample value at which the CMX910's Rx1 or Rx2 circuits enter sleep mode during an inactive slot.

Wakeup_Sample register: 16-bit write only. **C-BUS Address $13**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 00000					Wakeup sample value										

The Wakeup_Sample register holds the sample value at which the CMX910's Rx1 or Rx2 circuits leave sleep mode after an inactive slot, in time to detect a training sequence in the next slot.

Slot_Sample_UTC1PPS register: 32-bit read only. **C-BUS Address $14**
All bits cleared to 0 on reset.

Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	Slot count at last rising edge of UTC1PPS											

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	Sample count at last rising edge of UTC1PPS										

The Slot_Sample_UTC1PPS register indicates the value that the slot and sample counters held at the last rising edge of the UTC1PPS pin.

Slot_Nudge register: 16-bit write only. **C-BUS Address $15**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Slot nudge value (two's complement)															

The Slot_Nudge register is written with the amount that the slot counter is to be adjusted at the next nudge trigger point.

Sample_Nudge register: 16-bit write only. **C-BUS Address $16**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Sample nudge value (two's complement)															

The Sample_Nudge register is written with the amount that the sample counter is to be adjusted at the next nudge trigger point (only in manual nudge mode, the Sample_Nudge register is ignored when in either of the auto nudge modes).

Nudge_Trigger register: 16-bit write only. **C-BUS Address $17**

All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 00000					Sample count at which to add nudge values										

The Nudge_Trigger register holds the sample count at which the slot and sample counter nudge values get added.

Max_Auto_Nudge register: 16-bit write only. C-BUS Address $18
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 00000					Maximum auto nudge value										

The Max_Auto_Nudge register is used to set the magnitude of the maximum sample counter nudge in auto nudge track mode.

5.4.1 Manual Nudge

Manual nudge mode is enabled by setting Slot_Sample_Control b1-0 to 00 (auto nudge disabled). It is then the responsibility of the µC to keep the slot and sample counters aligned to the relevant timing reference. To assist with manual nudge mode in the case where a UTC time reference is available, the CMX910 copies the value of the Slot_Sample_Count register into the Slot_Sample_UTC1PPS register on each rising edge of the UTC1PPS input pin, from where it can be read by the µC; the UTC1PPS pin should be connected to a 1Hz signal whose rising edge is accurately aligned to the UTC second. Any error in the slot and sample counter values can then be easily determined. If the accurate 1Hz signal is lost or not available, the same information must be derived from timing information received on the AIS channels; this is made available in the RX1_Slot / RX1_Sample and RX2_Slot / Rx2_Sample registers (see section 5.6.1). Note: since there are 37½ slots per second, “even” seconds correspond to a slot boundary and “odd” seconds correspond to the middle of a slot.

In manual nudge mode, the µC initially synchronises the slot and sample counters, and can subsequently make minor adjustments to the sample counter, using the same mechanism in each case: the µC loads the Slot_Nudge and Sample_Nudge registers with two’s-complement values indicating by how much the slot and sample counters should be adjusted, and the Nudge_Trigger register is loaded with the exact sample time within a slot that these nudge values should be added to the counters – typically, the Nudge_Trigger value will need to be initialised only once. As soon as the nudge has been done, a “Nudge_Done” interrupt will be generated by the CMX910 and the Slot_Nudge and Sample_Nudge registers will be cleared to $0000, ready for new values to be written.

The slot counter usually needs adjusting only after a device reset, or if slot alignment has been lost for some reason, e.g. a GNSS timing signal has been lost for some time and has just been re-acquired. If the slot counter needs adjusting, the µC should write to the Slot_Nudge register first, then to the Sample_Nudge register. The act of writing to the Sample_Nudge register indicates to the CMX910 that both nudge values are ready, and they get applied simultaneously at the next nudge trigger point. If, however, only the sample counter needs adjusting then the µC need only write to the Sample_Nudge register, since Slot_Nudge will have been previously auto-cleared. Depending on the accuracy of the REFCLK input signal, it may be necessary to make several adjustments to the sample counter every minute. For instance, a 5ppm error in REFCLK will cause the sample counter to drift by 14.4 counts (nearly 3 bit periods) per minute.

Note that the slot counter “wraps” properly when it is nudged forwards past 2249 or backwards past 0, but the same does not apply to the sample counter – it can get it into an illegal state by nudging forward past 1279 or backwards past 0. Avoid this by ensuring that $0 \leq$ (Nudge_Trigger + Sample_Nudge) ≤ 1279.

5.4.2 Auto Nudge

Two auto nudge modes are provided which assist the µC with the initial synchronisation of the slot and sample counters, and allow the CMX910 to subsequently keep the sample counter aligned without further intervention. This requires an accurate UTC1PPS signal to be applied to the CMX910:

Auto nudge acquire (Slot_Sample_Control b1-0 = 01). This mode can be used for initial counter synchronisation after a device reset or in the case where the slot and sample counters have become grossly misaligned for some reason. Auto nudge acquire should be enabled during an "odd" UTC second, which means that the next UTC1PPS rising edge will be an "even" second. When the next UTC1PPS rising edge occurs, the CMX910 calculates the error in the sample counter (the sample counter should be 0 on an even second), and applies the required correction to the sample counter at the next Nudge_Trigger point. A Nudge_Done interrupt is then generated, indicating that sample counter alignment has been achieved.

Auto nudge track (Slot_Sample_Control b1-0 = 10). In this mode, the CMX910 calculates the correction needed for the sample counter once per second (on the rising edge of UTC1PPS). "Odd" and "even" UTC seconds are treated differently: if 320 ≤ Sample_Count < 960 when the rising edge of UTC1PPS occurs, the CMX910 assumes it to be an "odd" UTC second and calculates a sample nudge value of (640 - Sample_Count). Otherwise, the sample nudge value is calculated as (-1 × Sample_Count). This calculated value (or ±Max_Auto_Nudge, whichever is smaller in magnitude) is then added to the sample counter at the next Nudge_Trigger point. Note: if the µC writes a non-zero value to the Slot_Nudge register when in auto nudge track mode, this will be added to the slot counter at the same time that the sample counter value is updated. The Slot_Nudge register gets auto-cleared after being used which causes a Nudge_Done interrupt, otherwise Nudge Done interrupts are not generated in auto nudge track mode.

The Max_Auto_Nudge register is used to limit the magnitude of the allowed nudge in order to avoid potential timing problems. The Max_Auto_Nudge register is ignored in manual nudge and auto nudge acquire mode.

The typical sequence of events that the µC must perform to achieve and retain slot and sample counter synchronisation (using auto nudge) is shown below:

a) If the device has just come out of reset, initialise Nudge_Trigger (see section 5.4.4) and Max_Auto_Nudge registers.
b) Wait for an odd UTC second to occur, then put the CMX910 into auto nudge acquire mode.
c) Wait for a Nudge Done interrupt, then put the CMX910 into auto nudge track mode.
d) Wait for the next UTC1PPS rising edge, then read the Slot_Sample_UTC1PPS register, and use this to determine the error in the slot counter (the sample counter should be correctly aligned at this point). Write the necessary correction to the Slot_Nudge register.
e) Wait for a Nudge Done interrupt. Both slot and sample counters will now be correctly aligned, and the µC can proceed with AIS Rx and Tx operations, as required. No further Nudge Done interrupts will be generated while in auto nudge track mode unless Slot_Nudge is written to again.
f) Continue monitoring the UTC time signal. If the CMX910 slot and sample counters become misaligned for any reason (for instance, when a UTC leap second occurs), the µC must perform another synchronisation sequence. If a direct UTC time signal becomes lost for any reason, then the µC must switch the CMX910 to manual nudge mode and maintain synchronisation to a UTC indirect source or an appropriate base station.

5.4.3 Sleep Mode

The Rx1 and Rx2 channels are individually configurable using bits in the receiver control registers Rx1_Control and Rx2_Control. When enabled for AIS reception, it is possible to reduce power consumption significantly by configuring the CMX910 to automatically turn off its internal receiver circuits and negate the ENAB1 or ENAB2 pins during inactive slots. This will happen when sleep mode is enabled

and a valid training sequence and start flag has not been detected at the beginning of a slot, i.e. there is no data to demodulate. The Rx1 and Rx2 circuits will automatically power up again at the end of an inactive slot, ready to search for another training sequence in the next slot. Note that when sleep mode is enabled, the Rx1 and Rx2 channels power down independently; it is possible for either, or both, channels to be powered down in any particular slot, depending on the activity in the channels.

The sleep mode feature is enabled by setting bit 2 of the Slot_Sample_Control register. The period within an inactive slot that the Rx circuits are to be disabled must be programmed by the µC into the Sleep_Sample and Wakeup_Sample registers: sleep mode starts within an inactive slot when the sample counter equals the value in the Sleep_Sample register and finishes when the sample counter subsequently reaches the value in the Wakeup_Sample register.

The value in the Sleep_Sample register should be loaded with a sample number just beyond the latest point in a slot that the training sequence and start flag could occur. Account must be taken of the maximum remote transmitter timing error and distance delay, as well as the local receiver timing error and filter delays.

The value in the Wakeup_Sample register can be chosen to be towards the end of the inactive slot, or shortly after the beginning of the next slot. When determining the value to write to Wakeup_Sample, account must be taken of the maximum remote transmitter timing error, as well as the local receiver timing error and receive circuit start-up time.

5.4.4 Selecting the Nudge_Trigger Value

Whether the sample counter tracking is performed using manual nudge mode or auto nudge mode, the value written to the Nudge_Trigger register needs to be chosen carefully to avoid confusing the transmit or receive event timing.

All transmission events are timed relative to a point before the start of the slot in which a message is to be transmitted. This point is defined by the sample value loaded into the Tx_Start parameter in the Tx event sequence table (described in section 5.5.5). At this point the transmission is deemed to have started. All subsequent transmit events within the slot are timed relative to this Tx_Start point. This means that once transmission starts, all subsequent events (e.g. PA ramping, start of modulation) occur with the correct relative timing until the whole slot has been transmitted, irrespective of any change to Sample_Count. Therefore the only transmit problem that may occur is if a sample counter nudge causes the value in Sample_Count to skip past the point defined by Tx_Start, which would cause the event to be missed. This can be prevented by limiting the maximum allowed nudge value and ensuring that the Nudge_Trigger is far enough away from Tx_Start that Sample_Count can never skip past the Tx_Start event.

Similarly, all receive events are timed relative to the point that a start flag is detected after a valid training sequence, so once reception of a data packet begins, changes to Sample_Count will not affect the received data. The µC should be aware, however, that any values reported in the Rx1_Sample, Rx1_Slot, Rx2_Sample and Rx2_Slot registers are the values that were in the slot and sample counters at the time that the Rx1 and Rx2 start flags were detected. To avoid confusion it is therefore advisable to ensure that Nudge_Trigger is sufficiently far away from the likely position of a start flag.

Also, if the receive channels are configured to *sleep* during inactive slots (i.e. Slot_Sample_Control b2 = 1), the Nudge_Trigger value must be far enough away from Sleep_Sample and Wakeup_Sample values that the Sample_Count can never skip past these events (this would cause intermittent receive channel malfunction).

A further restriction is that the value added to the sample counter must not cause an overflow. This means that in manual nudge mode the µC should ensure that $0 \leq$ (Nudge_Trigger + Sample_Nudge) ≤ 1279. In auto nudge track mode, ensure that $0 \leq$ (Nudge_Trigger $\pm$ Max_Auto_Nudge) ≤ 1279.

© 2006 CML Microsystems Plc

5.5 Transmit Operation

The CMX910 is capable of transmitting AIS data in either raw mode or burst mode, and can also be configured for DSC transmission (FSK 1200 baud). AIS Carrier Sensing (CSTDMA) for Class B systems is supported, as is a mechanism to allow two or more messages to be chained into consecutive slots. A block diagram of the transmit data path is shown in Figure 6.

In AIS raw mode and DSC mode, data is passed directly from the Tx FIFO to the G(M)FSK/FSK modulator, so the µC will be responsible for sending any necessary training sequence and performing HDLC processing and NRZI coding for AIS, or other data coding for DSC. When configured in AIS burst mode, the CMX910 uses a secondary internal message buffer to assemble an entire message (up to 5 slot) to which it automatically adds the training sequence, start/stop flags, CRC, bit stuffing and NRZI coding prior to transmission. In either case, the µC must indicate how many data bits the message contains in the Tx_Bits register, and in which slot to power up the external Tx circuits in the Tx_Slot register. After setting up the appropriate registers, transmission is initiated by writing to a bit in the Tx_Control register.

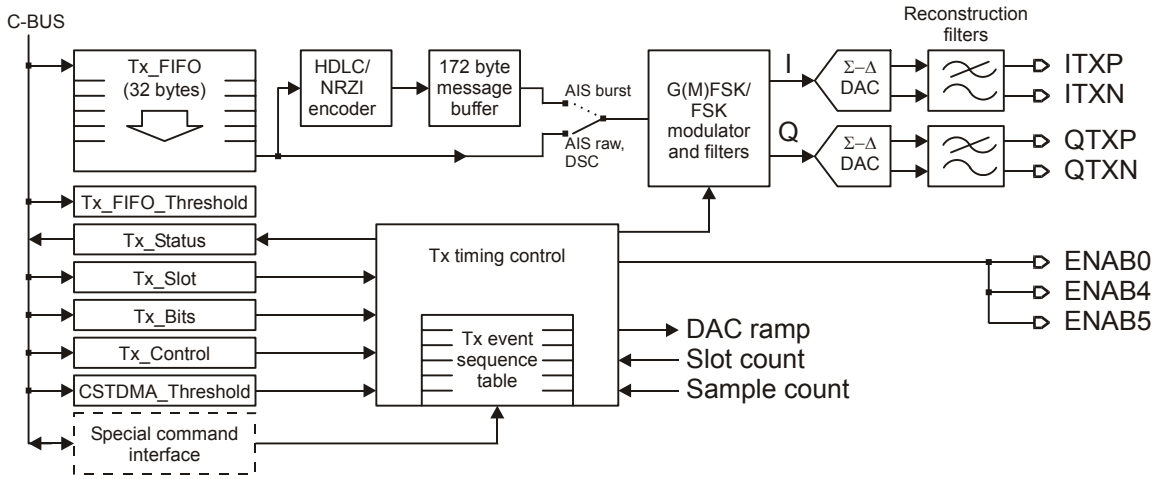


Figure 6 Transmit Channel

5.5.1 Transmitter Registers

Tx_FIFO register: 8-bit write only (data-streaming). C-BUS Address $20
32 byte Tx channel FIFO, emptied on reset. Supports C-BUS data streaming.

Bit:	7	6	5	4	3	2	1	0
	Tx channel data byte							

Tx_FIFO_Threshold register: 8-bit write only. C-BUS Address $21
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
	Reserved, set to 000			Tx FIFO threshold level				

The transmit FIFO threshold register is used to set the level at which a “FIFO nearly empty” warning is triggered. If the number of bytes in Tx_FIFO is less than or equal to the value in bits 4-0 of threshold register then the FIFO Trigger flag (bit 7 of Tx_Status) will be set to 1. This can also be used to generate an interrupt. Bits 7-5 of TX_FIFO_Threshold should be set to 0.

Tx_Status register: 16-bit read only. **C-BUS Address $22**
Register gets set to $0080 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Tx Over-flow	Tx Under-flow	Tx FIFO Fill Level						Tx FIFO Trigger	0	0	0	Tx State			

Tx_Status register b15: Tx Overflow
This bit gets set high if the µC attempts to write to the Tx FIFO when it is already full, indicating that data has been lost. A Tx Overflow does not automatically cause the transmission to be aborted, this must be done separately by the the µC if necessary. The Tx Overflow bit gets cleared as soon as the Tx_Status register has been read.

Tx_Status register b14: Tx Underflow
This bit gets set high if the µC does not send data to the CMX910 quickly enough during transmission, causing a data famine in the Tx channel (this does not happen in AIS burst mode since an entire message must be conveyed to the CMX910 *before* transmission starts). A Tx Underflow does not automatically cause the transmission to be aborted, this must be done by the the µC. Failure to do this will result in erroneous data being transmitted. The Tx Underflow bit gets cleared as soon as it has been read.

Note: Tx_Status b15 and b14 are ORed together for the purpose of generating an interrupt.

Tx_Status register b13-8: Tx FIFO Fill Level
This shows how many bytes are in the Tx FIFO. The number will be in the range 0 to 32.

Tx_Status register b7: Tx FIFO Trigger
This bit will be high if the Tx FIFO fill level is less than or equal to the Tx threshold level, i.e. Tx_Status[13-8] ≤ Tx_FIFO_Threshold[4-0]. This bit can generate an interrupt.

Tx_Status register b3-0: Tx State
Indicates the current transmitter state. If a transmission has not been requested the Tx state will be *Idle*, otherwise the Tx State bits change to reflect the progress of the transmission. After a transmission has completed the Tx State bits will either indicate that the transmission was successful, or will indicate the nature of any problem encountered.

b3	b2	b1	b0	
0	0	0	0	Idle
0	0	0	1	Building message buffer (burst mode)
0	0	1	0	Message buffer ready (burst mode)
0	0	1	1	Tx pending
0	1	0	0	Tx in progress
0	1	0	1	Tx aborted – carrier sensed (CSTDMA)
0	1	1	0	Tx aborted – buffer not ready (burst mode)
0	1	1	1	Tx aborted – message too long (burst mode)
1	0	0	0	Chained message in progress

Tx_Slot register: 16-bit write only. **C-BUS Address $23**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 0000				Slot number in which to begin a transmission sequence											

The TX_Slot register must be loaded with the slot number in which a transmit sequence begins. Typically this will be one or two slots *before* the slot in which data is to be transmitted, allowing time for the external RF circuits to power up and stabilise. Further details about transmit timings are provided in section 5.5.5.

Tx_Bits register: 16-bit write only. **C-BUS Address $24**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Number of bits to transmit															

In AIS burst mode, the Tx_Bits register must be programmed with the total number of data bits in the message *excluding* all of the training sequence, start/end flags, CRC and bit stuffing bits added by the CMX910. In AIS burst mode the number loaded into Tx_Bits should always be a multiple of 8 since the AIS specification requires that the data payload (prior to HDLC coding) be a whole number of bytes.

In AIS raw mode or DSC mode, the Tx_Bits register must be programmed with the total number of data bits in the message *including* all of the training sequence, start/end flags, CRC and bit stuffing bits (AIS mode) or other data coding bits (DSC mode). This number will generally not be a multiple of 8, in which case the last byte sent by the µC through the Tx FIFO must be padded with trailing zeroes.

Tx_Control register: 16-bit write only. **C-BUS Address $25**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved, set to 0000000							Tx Start	CS-TDMA enable	CS-TDMA chan	Tx Mode				Tx FIFO Clear	Tx State Reset

Tx_Control register b8: Tx Start
Setting b8 = 1 causes a transmission to be triggered in the slot specified in the Tx_Slot register. This bit will be automatically cleared as soon as the transmission is complete.

Tx_Control register b7: CSTDMA Enable
Set b7 = 1 for Carrier Sensing TDMA operation, b7 = 0 for normal operation.

Tx_Control register b6: CSTDMA Channel Select
Determines which receive channel is examined for presence of a carrier before the transmit operation starts. Set b6 = 1 to select CSTDMA operation on “Channel 2”, or b6 = 0 for operation on “Channel 1”.

CSTDMA_Threshold register: 16-bit write only. C-BUS Address $26
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CSTDMA signal strength threshold															

The value in the CSTDMA_threshold register is used when attempting a transmission with CSTDMA enabled. At the beginning of the slot in which data is to be transmitted, typically just before the PA is ramped up, the received signal strength in the selected CSTDMA channel (Rx1 or Rx2) is measured. If this exceeds the value in the CSTDMA_threshold register, the transmission is aborted and a *TxDone* interrupt is generated. Otherwise, transmission proceeds normally. Note: the value written to the CSTDMA_Threshold register must be between 0 and 32767 ($7FFF).

The received signal strength is calculated by accumulating the I/Q vector magnitudes at each sample point (i.e. every 20.833µs) between the CSTDMA_START and CSTDMA_END points, as defined in the Tx Event Sequence table (see section 5.5.5). Before being added to the running total, each vector magnitude is multiplied by a fixed gain of 0.4117 and a user-programmable gain "CSTDMA_gain" of between 0 and 1 (this is set up using a special command function, see section 5.12). In the case of overflow, the accumulated signal strength value saturates at 32767 ($7FFF). The total accumulated signal strength value is given by:

$$CSTDMA\_signal\_strength = \sum_{n=CSTDMA\_START+1}^{CSTDMA\_END} \left(\sqrt{I_n^{\,2} + Q_n^{\,2}} \times 0.4117 \times CSTDMA\_gain \right)$$

For the purpose of calculating the CSTDMA signal strength, the signed 16-bit I and Q vector values are taken directly from the output of the selected channel's Σ-Δ ADCs, prior to passing through the channel filters. With a DC voltage of ±1.7V (differential) applied to the input of the I or Q ADC, the corresponding I or Q ADC output will be approximately ±15000.

Tx_Control register b5-2: Tx Mode

b5	b4	b3	b2	
0	0	0	0	AIS raw mode, 25kHz channel
0	0	0	1	AIS raw mode, 12.5kHz channel
0	0	1	0	DSC mode
0	0	1	1	Reserved, do not use
0	1	0	0	AIS burst mode, 25kHz channel
0	1	0	1	AIS burst mode, 12.5kHz channel
0	1	1	0	Reserved, do not use
0	1	1	1	Reserved, do not use
1	0	0	0	AIS test mode, 25kHz channel (transmit data supplied by µC)
1	0	0	1	AIS test mode, 12.5kHz channel (transmit data supplied by µC)
1	0	1	0	DSC test mode (transmit data supplied by µC)
1	0	1	1	Reserved, do not use
1	1	0	0	AIS test mode, 25kHz channel (internal PRBS transmitted)
1	1	0	1	AIS test mode, 12.5kHz channel (internal PRBS transmitted)
1	1	1	0	DSC test mode (internal PRBS transmitted)
1	1	1	1	Reserved, do not use

The Tx Mode bits select the modulation scheme and (AIS) channel spacing to use. The CMX910 automatically configures its internal modulators and channel filters for whichever transmit mode is selected:

- AIS (25KHz channel) = GMSK with a BT-product of 0.4 and a modulation index of 0.5.
- AIS (12.5KHz channel) = GFSK with a BT-product of 0.3 and a modulation index of 0.25.
- DSC = 1200 baud FSK, with frequency modulation of a 1700Hz sub-carrier and a pre-emphasis of 6dB/octave. The frequency shift is between 1300Hz (logic 1) and 2100 Hz (logic 0).

For both AIS and FSK operation, setting b5=1 puts the CMX910 into transmit test mode: this causes data to be transmitted immediately, without waiting for the next transmit trigger point as in normal transmit modes. Transmitted test data can be configured to come from the µC through the Tx FIFO, or from an internally generated pseudo-random bit sequence. No data coding or insertion of training sequences will be carried out, and the CMX910 will not attempt to perform RF control using the ENAB pins and DAC0 ramping; transmission will continue until transmit test mode is cleared by the µC. Note that when any test mode is enabled, it is essential that Tx_Control b8 = 0.

Tx_Control register b1: Tx FIFO Clear
Data written to this bit does not get stored; instead, writing a 1 to this bit generates a reset pulse which empties the Tx FIFO and resets the Tx FIFO fill level (Tx_Status b13-8) to zero.

Tx_Control register b0: Tx State Reset
Immediately after power up, the Tx channel must be initialised by writing a 1 to the Tx State Reset bit of Tx_Control. Writing a 1 to the Tx State Reset bit can also be done at any other time in order to cause any pending or active transmission to be terminated – this causes the PA and transmit hardware to be switched off, any internal states related to Tx to be cleared and the internal message buffers (AIS burst mode) to be wiped. The Tx FIFO will not be cleared by writing a 1 to this bit, that can be done if necessary by writing a 1 to Tx_Control bit 1. Note: when a 1 is written to this bit, a delay of at least 250 µs is required for the CMX910 to reset the transmit channel before the Tx_Control register is written to again.

5.5.2 AIS Raw Mode Transmit

In AIS raw mode, transmit data is passed directly from the Tx FIFO to the G(M)FSK modulator. Apart from the Tx FIFO, no buffering is performed inside the CMX910. The µC must calculate the entire transmitted message including the training sequence, HDLC processing (start/stop flags, bit stuffing, and CRC insertion) and NRZI coding. Note: In AIS raw mode, data written to Tx_FIFO is transmitted *most significant bit first*. The AIS message structure, however, requires each message byte to be output *least significant bit first*. The µC must therefore ensure that during the process of HDLC processing and NRZI coding that the resulting data bytes are correctly reversed.

The µC is expected to perform the following sequence of operations in order to transmit a data packet in raw mode:

- Initialise the transmitter timing registers as described in section 5.5.5. (This only needs to be done once after the device has come out of reset).
- Check that the Tx State flags in the Tx_Status register indicate that the transmitter is in the *Idle* state and that the Tx FIFO does not contain data from an earlier aborted transmission. If necessary, the transmitter state machine can be reset and Tx_FIFO can be cleared by writing 1 to Tx_Control register b1-0.
- Write the total number of bits to be transmitted into the Tx_Bits register (not necessarily a multiple of 8, because of bit stuffing).
- Write the timing reference slot number to Tx_Slot (this will most likely be a one or two slots before the slot in which to transmit data, to allow time for the external Tx circuits to power up).
- Prime the Tx FIFO with at least one byte of transmit data, up to the entire message if it fits. Unused bits in the last byte should be padded with zeroes – note that data written to the Tx_FIFO in AIS raw mode is transmitted most significant bit first.
- Request a transmission by writing to the Tx Start bit in the Tx_Control register.
- Feed any remaining data bytes to the Tx FIFO as they are required, then wait for a Tx Done interrupt and check the transmitter state in the Tx_Status register to determine if the transmission was successful.

The actions of the CMX910 are:

- If the Tx State is cleared by the µC, the external RF circuits are turned off if necessary (DAC0 ramped down and then ENAB0, 4 and 5 negated, assuming the CMX910 has control of these functions). The Tx State then becomes *Idle*.
- Upon receiving a Tx Start request, the CMX910 notes the Tx_Slot and Tx_Bits values and sets the Tx State to *Tx pending*.
- When the requested transmit point arrives, the Tx State changes to *Tx in progress* and the external transmit circuits are enabled according to how the transmit timing is configured.
- At the end of a transmitted message the PA ramps down. If the Tx_FIFO is empty at this point the Tx State changes to *Idle*, a Tx Done interrupt is generated, and the external RF circuits get disabled. If the Tx_FIFO is not empty, the CMX910 assumes a chained message is being sent; it notes the length of this new message (in Tx_Bits), and leaves the external RF circuits enabled. The Tx State then changes to *Chained message*, then to *Tx in progress* when transmission of the new message begins in the following slot.

Note that if CSTDMA mode is active and a carrier is sensed in the selected channel at the beginning of the requested transmit slot, the transmission is aborted (Tx State changes to *Tx aborted, carrier sensed*) – this causes a Tx Done interrupt to be generated. Data in Tx_FIFO is retained, so the µC can choose to issue a Tx State Reset and clear Tx_FIFO, or reschedule the transmission in another slot.

In AIS raw mode, if data is fed to Tx_FIFO too slowly then the Tx Underflow bit in the Tx_Status register gets set high, or if more data is written than Tx_FIFO can accommodate then the Tx Overflow bit gets set high. In either case, a Tx FIFO Error interrupt gets generated, but transmission continues. The response of the µC should be to reset the Tx State and clear Tx_FIFO by setting Tx_Control b0 and b1 high. This prevents erroneous data from being transmitted.

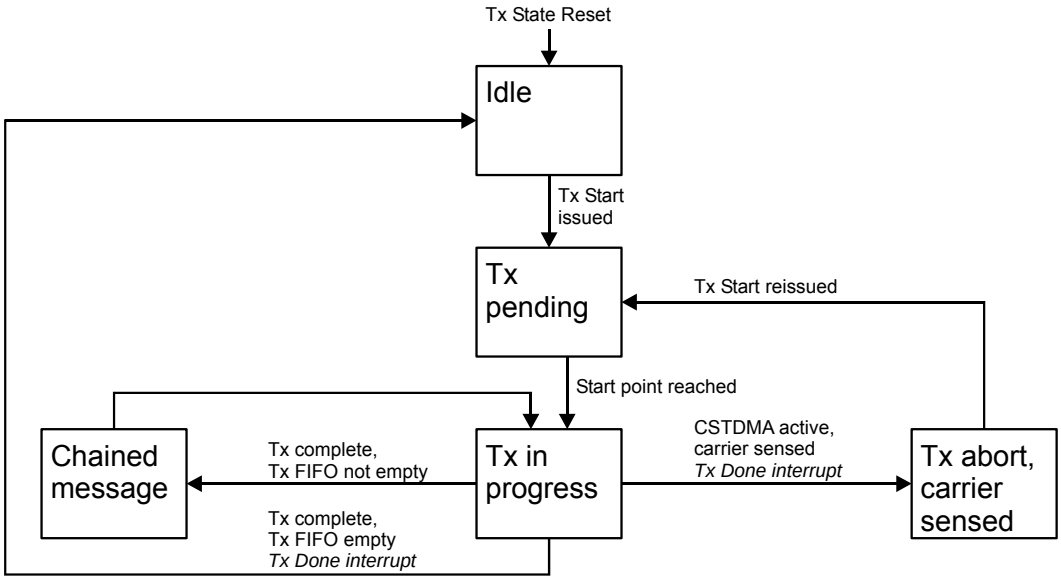


Figure 7 Tx (AIS raw mode) state transitions

5.5.3 AIS Burst Mode Transmit

In AIS burst mode, an entire message is transferred to the CMX910 prior to transmission. The CMX910 then processes the data by performing bit stuffing, NRZI encoding and addition of training sequence, start/stop flags and CRC checksum as required by AIS. The resulting bit stream is held in a secondary message buffer within the CMX910. This message processing must be complete before transmission is allowed to begin, so it is not possible to get a Tx underflow error in AIS burst mode. Note: in AIS burst mode, the data bytes are automatically transmitted *least significant bit first* as required by the AIS specification.

The µC is expected to perform the following sequence of operations in order to transmit an AIS data packet in burst mode:

- Initialise the transmitter timing registers as described in section 5.5.5. (This only needs to be done once after the device has come out of reset).
- Check that the Tx State flags in the Tx_Status register indicate that the transmitter is in the *Idle* state and that Tx_FIFO does not contain data from an earlier aborted transmission. If necessary, the transmitter state machine can be reset and Tx_FIFO can be cleared by writing 1 to Tx_Control register b1-0.
- Write the total number of data bits to be transmitted into the Tx_Bits register. This should be the number of data bits in the packet *excluding* any bit stuffing bits, flags, CRC checksum or training sequence. The AIS specification requires this number to be a multiple of eight.
- Load the whole (up to 5 slot) message into the CMX910 through Tx_FIFO.
- Write the timing reference slot number to Tx_Slot (this will most likely be a one or two slots before the slot in which to transmit data, to allow time for the external Tx circuits to power up).
- Request a transmission by writing to the Tx Start bit in the Tx_Control register.
- Wait for a Tx Done interrupt, then check the transmitter state in the Tx_Status register to determine if the transmission was successful.

The actions of the CMX910 during a successful transmission will be:

- If a Tx State Reset is performed by the µC, the external RF circuits are turned off if necessary (DAC0 ramped down and ENAB0, 4, 5 negated, assuming the CMX910 has control of these functions). The Tx State bits in the Tx_Status register then indicate *Idle*.
- As soon as data is written to the Tx_FIFO, the CMX910 will begin assembling a packet for transmission in its internal message buffer. During this operation, the Tx State indicates *Building message buffer*.
- When the CMX910 has assembled a complete transmit packet (including training sequence and HDLC coding), if a Tx Start has already been issued the Tx State changes to *Tx pending*, otherwise the Tx State changes to *Message buffer ready* and waits for Tx Start to be issued before going to *Tx pending*.
- When the requested transmit point arrives, the Tx State changes to *Tx in progress* and the external transmit circuits are enabled according to how the transmit timing is configured.
- At the end of a transmitted message the PA ramps down. If the Tx_FIFO is empty at this point the Tx State changes to *Idle*, a Tx Done interrupt is generated, and the external RF circuits get disabled. Otherwise the CMX910 assumes a chained message is being sent; it notes the length of this new message (in Tx_Bits), leaves the external RF circuits enabled, and builds a new message (during which time Tx State first changes to *Chained message* for approximately 20µs then to *Building message buffer*, then to *Tx pending*). This message is transmitted in the following slot, during which time Tx State indicates *Tx in progress*.

A number of error conditions are checked for during AIS burst mode transmit, each of which causes transmission to be aborted and a Tx Done interrupt to be generated. The associated Tx States are:

1. *Tx aborted, message too long*: This happens if the internal message buffer is not big enough for the HDLC coded data (should not happen in normal operation, as the message buffer is big enough for a 5-slot message). This condition requires the µC to issue a Tx State Reset.
2. *Tx aborted, buffer not ready*: This happens if the requested Tx start point arrives before the message buffer is ready. The µC can then choose to issue a Tx State Reset or to carry on building the message and reschedule the transmission in another slot.
3. *Tx aborted, carrier sensed*: This happens if CSTDMA mode is active and a carrier is sensed in the selected channel at the beginning of the requested transmit slot. Data in the message buffer is retained, so that the µC can choose to issue a Tx State Reset or to reschedule the transmission in another slot.

It is not possible to get a Tx_FIFO underflow in AIS burst mode, but writing too quickly could cause an overflow (this sets the Tx Overflow bit in the Tx_Status register), causing a Tx FIFO Error interrupt to be generated. The contents of the message buffer will become corrupted by the overflow but the transmit operation will continue, so the µC should abort the transmission (i.e. reset the Tx State and clear Tx_FIFO by setting Tx_Control b0 and b1 high).

There is a possibility that the CMX910 will add enough stuffing bits to a message in AIS burst mode to cause the message to overrun into the next slot, although there is enough padding in the AIS slot structure to ensure that this occurrence is extremely rare. A minor slot overrun is not normally considered to be a problem in the AIS system, but if this happens when the CMX910 is attempting to chain a message in the next slot, it is possible that the CMX910 will miss the chained message start event. This will cause the chained message to be transmitted one slot too late. To enable the µC to recognise if this error condition is about to happen, the special command “read_message_length” (section 5.12) is provided that allows the µC to determine the total number of message bits that have been created in the CMX910’s internal message buffer. When this special command is issued, the CMX910 waits until the message buffer is complete then generates a “Special Command Done” interrupt. The µC can then read the total number of message bits from the SPC_Out0 register; which includes the training sequence, start/stop flags, CRC checksum and stuffing bits, as well as the actual message bits. The µC should also take into account the magnitude of any nudge that may be performed during the transmission to determine whether the message is too long to allow chaining to be attempted.

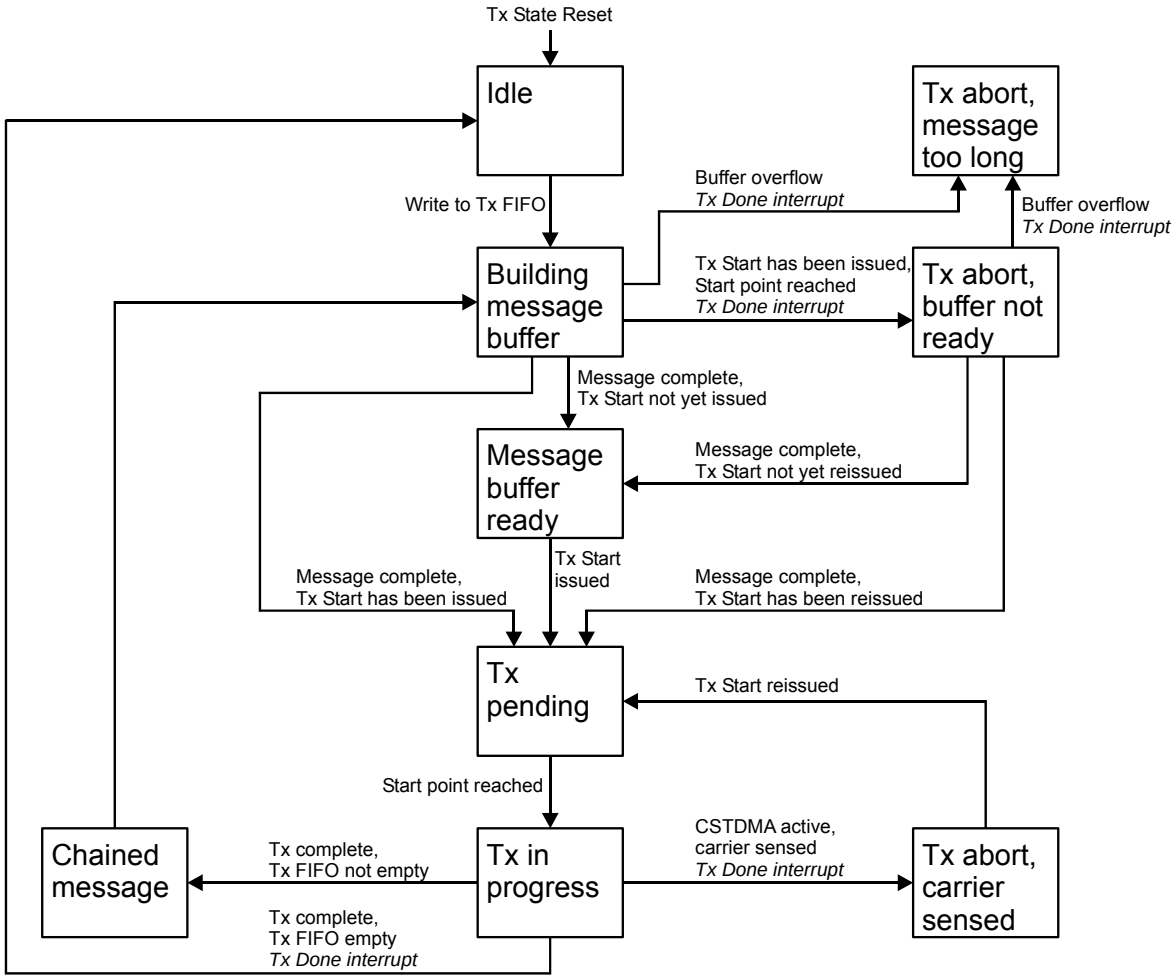


Figure 8 Tx (AIS burst mode) state transitions

5.5.4 DSC Transmit

DSC transmission operates in a similar way to AIS raw mode: transmit data is passed directly from the Tx FIFO to the FSK modulator and filters. Apart from the Tx FIFO, no buffering is performed inside the CMX910. The µC must calculate the entire DSC transmit message including the dot pattern and phasing sequence, and all subsequent message and checksum words (both DX and RX characters). Note: In DSC mode, data written to Tx_FIFO is transmitted *least significant bit first*.

The µC is expected to perform the following sequence of operations in order to transmit a DSC message:

- Initialise the transmitter timing registers as described in section 5.5.5. (This only needs to be done once after the device has come out of reset).
- Check that the Tx State flags in the Tx_Status register indicate that the transmitter is in the *Idle* state and that the Tx FIFO does not contain data from an earlier aborted transmission. If necessary, the transmitter state machine can be reset and Tx_FIFO can be cleared by writing 1 to Tx_Control register b1-0.
- Write the total number of bits to be transmitted into the Tx_Bits register.
- Write the timing reference slot number to Tx_Slot (this will most likely be a one or two slots before the slot in which to transmit data, to allow time for the external Tx circuits to power up).

- Prime the Tx FIFO with at least one byte of transmit data, up to the entire message if it fits. Unused bits in the last byte should be padded with zeroes – note that data written to the Tx_FIFO in DSC mode is transmitted least significant bit first.
- Request a transmission by writing to the Tx Start bit in the Tx_Control register.
- Feed any remaining data bytes to the Tx FIFO as they are required, then wait for a Tx Done interrupt and check the transmitter state in the Tx_Status register to determine if the transmission was successful.

The actions of the CMX910 are:

- If the Tx State is cleared by the µC, the external RF circuits are turned off if necessary (DAC0 ramped down and then ENAB0, 4 and 5 negated, assuming the CMX910 has control of these functions). The Tx State then becomes *Idle*.
- Upon receiving a Tx Start request, the CMX910 notes the Tx_Slot and Tx_Bits values and sets the Tx State to *Tx pending*.
- When the requested transmit point arrives, the Tx State changes to *Tx in progress* and the external transmit circuits are enabled according to how the transmit timing is configured.
- At the end of a transmitted message the PA ramps down, the Tx State changes to *Idle*, a Tx Done interrupt is generated, and the external RF circuits get disabled.

In DSC mode, if data is fed to Tx_FIFO too slowly then the Tx Underflow bit in the Tx_Status register gets set high, or if more data is written than Tx_FIFO can accommodate then the Tx Overflow bit gets set high. In either case, a Tx FIFO Error interrupt gets generated, but transmission continues. The response of the µC should be to reset the Tx State and clear Tx_FIFO by setting Tx_Control b0 and b1 high. This prevents erroneous data from being transmitted.

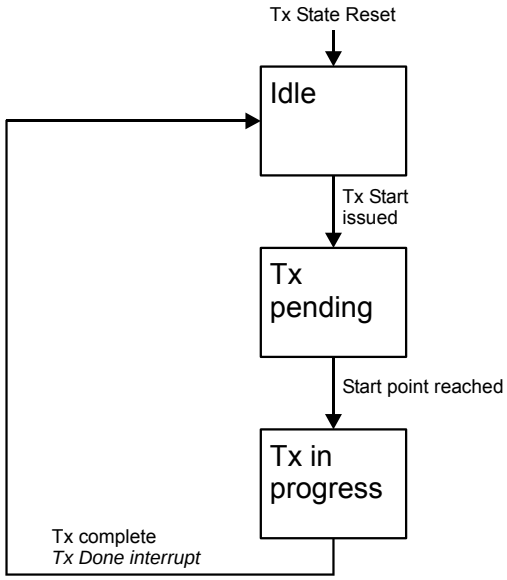


Figure 9 Tx (DSC mode) state transitions

5.5.5 Transmitter Timing Control

The CMX910 can be easily configured to control the timing of transmission events. This includes the enabling of external RF circuits (e.g. synthesisers and power amplifier), as well as the time at which internal data modulation begins. The flexibility of this timing control allows the CMX910 to be straightforwardly adapted to the characteristics of the RF transmit circuits, such as the power up time or synthesiser lock time. The control of the external RF transmit circuits is effected through three of the Device Enable Port digital output pins (ENAB0, 4, 5) along with the DAC0 ramping function.

Figure 10 Typical AIS Transmission

A typical AIS transmission is shown in Figure 10. The CMX910 starts timing transmit “power-up” actions relative to point A, which will typically be in a slot prior to the one in which transmission is to occur – this allows external RF transmit circuits time to power up and stabilise. At the end of a transmission, a sequence of “power-down” actions is performed. The CMX910 times these “power-down” actions relative to the last message bit having been modulated, shown as point B in Figure 10. In that way differences in message length due to bit stuffing, and multiple-slot messages, are automatically accommodated.

Point A in Figure 10 occurs in the slot defined in the C-BUS Tx_Slot register. The absolute sample number at which point A occurs within the slot, along with the relative timings of all of the subsequent transmit events, are configured as a table of values that are communicated to the CMX910 using a Special Command Interface operation (section 5.12) – this operation must be performed before any transmissions are attempted. Typically, this will only need to be done once as part of an initialisation routine. All timings are measured in units of “sample times”, each of which lasts for one fifth of an AIS bit period ($1/5 \times 1/9600 \cong 20.833\mu s$).

The transmit sequence consists of several different types of event. These are:

- Changes to the external hardware, done through the Tx device enable pins ENAB0, 4, 5 (typically used to turn the RF on/off, Tx on/off, and PA on/off) and the triggering of the DAC0 ramp up/down.
- Trigger for the start/end of the CSTDMA period (only has an effect if CSTDMA is enabled).
- Timing trigger for the start and end of the data modulation.
- A chained message start and end event to allow for AIS transmission in consecutive slots without a full switch off in-between.
- A dummy event in case any of the above are not required in the application.

The transmit event sequence is programmed as follows:

1. Apply special command “load_tx_sequence” ($10). This readies the CMX910’s Tx event sequence table to accept data.
2. Write the Tx_Start sample value into the SPC_In0 register and the ENAB0-5 pin reset state into SPC_In1, then apply special command “poke2_inc” ($08). This copies the data into the internal Tx event sequence table.
3. For the remaining fourteen transmit events, load the event number into SPC_In0 and the delay (relative to the previous event) into SPC_In1, and apply special command “poke2_inc” ($08). The

events must be loaded in chronological order, with a minimum delay of 1 sample period between consecutive events (maximum delay = 32767). All fourteen of these event slots must be filled; if certain transmit events are not required, they should be replaced with the DUMMY event.

4. Apply special command “setup_tx_sequence” ($0F). This causes the CMX910 to process the table of Tx events and configure its internal transmit logic accordingly.

Note that when the start sample (Tx_Start) is reached within Tx_Slot, the CMX910 inserts a fixed delay of four samples – this is in *addition* to the delay specified for the first event in the sequence table. Also note that the last six events specified in the sequence table are timed from the end of the last message bit being fed to the internal modulator.

The transmit events and their code numbers are as follows:

RF_ON	1	Pin ENAB0 is asserted.
TX_ON	2	Pin ENAB4 is asserted.
PA_ON	3	Pin ENAB5 is asserted.
DAC0_RAMPUP	4	DAC0 begins ramping up (used for PA ramp up control).
CSTDMA_START	5	Defines the start of the CSTDMA sensing window.
CSTDMA_END	6	Defines the end of the CSTDMA sensing window.
MODULATE_START	7	Data modulation begins. (Tx data fed to internal filters).
MODULATE_END	8	Data modulation ends; reference for power down events.
DAC0_RAMPDOWN	9	DAC0 begins ramping down.
PA_OFF	10	Pin ENAB5 is negated.
TX_OFF	11	Pin ENAB4 is negated.
RF_OFF	12	Pin ENAB0 is negated.
DUMMY	13	No action.
CHAINED_MESSAGE_START	14	The start of the sub-sequence repeated for chained messages.
CHAINED_MESSAGE_END	15	The end of the sub-sequence repeated for chained messages.

For the duration that ENAB5 is asserted (i.e. between the PA_ON and PA_OFF events) the CMX910 turns off the three Rx control lines (ENAB1-3), and disables its internal Rx1/Rx2 circuits. It is intended that during this period the external Tx/Rx switch for the antenna switches to select Tx.

When calculating the MODULATE_START timing value, the delay through the CMX910’s internal transmit filters and any external components must be taken into account to ensure that data bits appear on-air at the correct time (the filter delays are specified in section 7). The MODULATE_END event has an in-built delay of 46 sample times to allow the last bit to make its way out of the transmit filter. Allowance must be made for this built-in delay, as well as for the delay through any external components, when calculating the timing of the transmit power down events.

Because all actions subsequent to Tx_Start effectively use *relative* timings, they will not be disturbed by any sample counter “nudge” that may occur during transmission. It is important, however, to ensure that any “nudge” that occurs does not cause the sample counter to skip past the Tx_Start point, which would cause the transmission to be missed.

A working example of how to set up a transmit event sequence is shown in Table 2 (the order of events and delay timings shown are for illustrative purposes only):

Parameter	SPC _In0	SPC _In1	Explanation
Tx_Start sample value and ENAB0-5 pin reset state	500	0	The transmit sequence starts at sample number 500 in a slot. The ENAB0-5 pins will be set to 0 on a reset or a command to start the Tx sequence, i.e. they are all active high.
TX_ON	2	5	After the initial 4 sample delay, insert another 5 sample delay then assert the TX control line (ENAB4).
RF_ON	1	790	Insert 790 sample delay then assert the RF control line (ENAB0).
CSTDMA_START[1,3]	5	20	Insert 20 sample delay then start monitoring the chosen Rx input for a signal which will cause an abort (if CSTDMA enabled).
CSTDMA_END[1,3]	6	32	Insert 32 sample delay then stop CSTDMA monitoring.
PA_ON	3	15	Insert 15 sample delay then assert the PA control line (ENAB5). At this point, the three Rx control lines (ENAB1-3) are negated.
CHAINED_MESSAGE_START[1,2]	14	1	Insert 1 sample delay then place a marker for the point at which the sequence will restart if a consecutive Tx message is needed.
DAC0_RAMPUP	4	6	Insert 6 sample delay then initiate the DAC0 ramp-up (for AIS, the transmitted signal will be carrier only at this point)
MODULATE_START[1,2]	7	8	Insert 8 sample delay then start feeding data to the transmit modulator and filters.
At this point during a transmission the CMX910 feeds the entire message to the transmit modulator bit-by-bit. All subsequent transmit events are timed relative to the end of the last message bit, indicated by the MODULATE_END event.			
DAC0_RAMPDOWN[4]	9	1	Insert 1 sample delay then initiate the DAC0 ramp-down.
MODULATE_END[1,2]	8	10	Insert 10 sample delay then define the “end-of-modulation” point (“B” in Figure 10). Between MODULATE_START and MODULATE_END there are (message bits × 5) +46 samples for AIS, or (message bits × 40) + 46 samples for FSK.
CHAINED_MESSAGE_END[1,2]	15	30	Insert 30 sample delay. If more data is present in Tx_FIFO, chain a second (or third…) message on.
PA_OFF	10	6	Insert 6 sample delay then negate the PA control line. At the same time, the three Rx control lines (ENAB1-3) are reasserted.
TX_OFF	11	7	Insert 7 sample delay then negate the TX control line.
RF_OFF	12	8	Insert 8 sample delay then negate the RF control line.

Notes:
1. It is essential that the CSTDMA, CHAINED_MESSAGE and MODULATE *START* events precede their associated *END* events, otherwise undesirable results will be obtained.
2. MODULATE_START must come after CHAINED_MESSAGE_START and MODULATE_END must come before CHAINED_MESSAGE_END for consecutive messages to work. MODULATE_START and CHAINED_MESSAGE_START must appear in the first group of eight timed events, MODULATE_END and CHAINED_MESSAGE_END must appear in the final group of six.
3. It is intended that CSTDMA only operates in the period prior to the actual on-air transmission of the first in a sequence of chained messages. Both CSTDMA_START and CSTDMA_END should come before PA_ON and DAC0_RAMPUP, otherwise CSTDMA will not operate correctly. Also, the delay between CSTDMA_START and CSTDMA_END must be ≥ 7.
4. In this example DAC0_RAMPDOWN is specified before MODULATE_END in the last group of six events, so the DAC0 ramp-down begins prior to the MODULATE_END point – with the values shown, the DAC0 ramp-down starts 10 samples before MODULATE_END.

Table 2 Example Tx Event Sequence Setup

5.6 Receive Operation

The CMX910 has two main receive channels (Rx1 and Rx2) which are capable of receiving AIS data in either *raw* mode or *burst* mode, and either of which may be configured for DSC reception (FSK 1200 baud). Alternatively, an external FSK demodulator may be interfaced to the CMX910, allowing simultaneous reception of two AIS channels and one DSC channel. The Rx1 and Rx2 channels can be configured and operated independently.

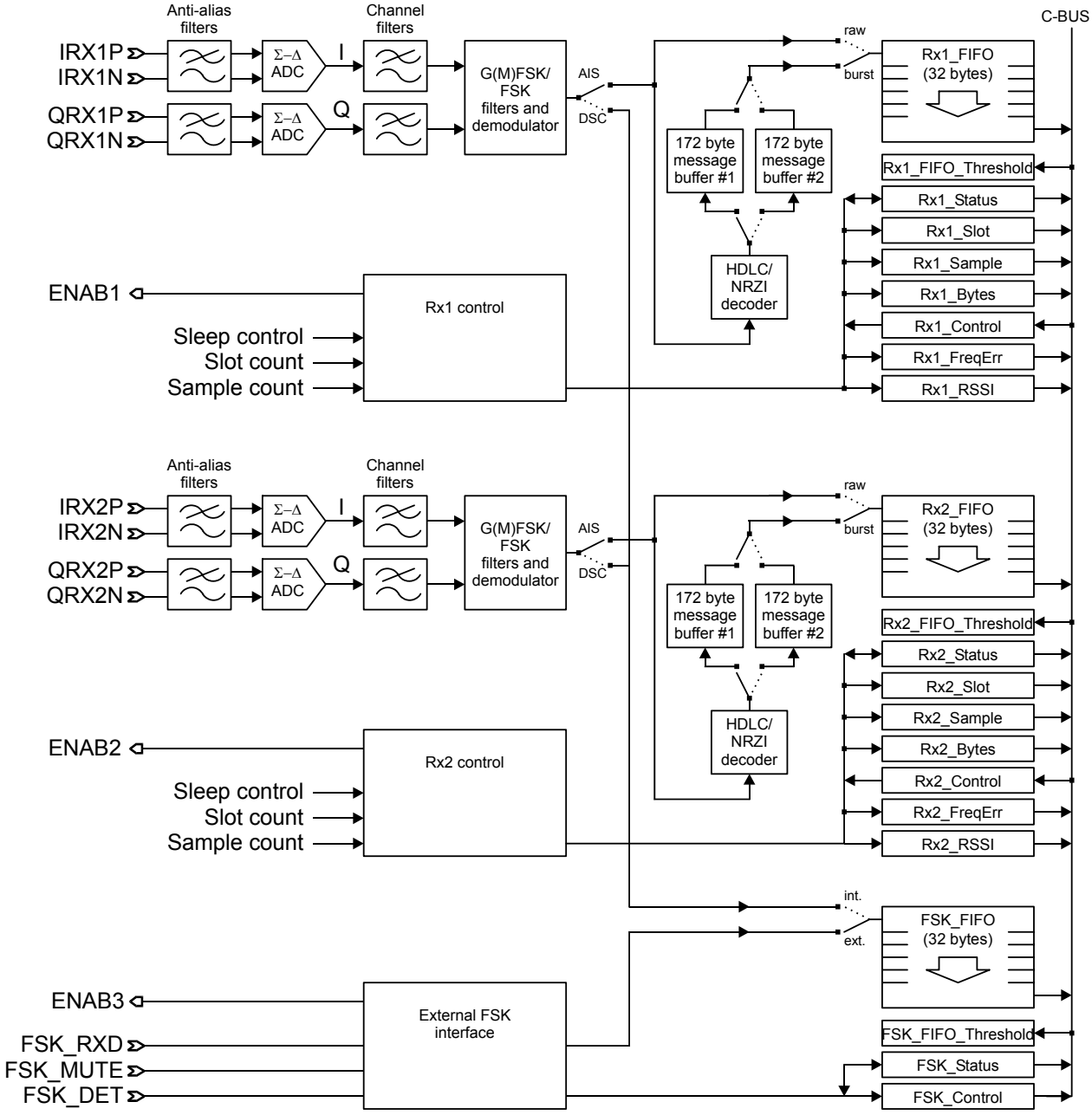


Figure 11 Receive Channel

5.6.1 Receiver Registers

Rx1_FIFO register: 8-bit read only (data-streaming). C-BUS Address $30
Rx2_FIFO register: 8-bit read only (data-streaming). C-BUS Address $40
FSK_FIFO register: 8-bit read only (data-streaming). C-BUS Address $50
Independent 32 byte receive channel FIFOs, emptied on reset. Support C-BUS data streaming.

Bit:	7	6	5	4	3	2	1	0
$30:	Rx1 channel data byte							
$40:	Rx2 channel data byte							
$50:	FSK channel data byte							

Rx1_FIFO and Rx2_FIFO are used to transfer AIS data to the µC from the Rx1 and Rx2 channels respectively. FSK_FIFO is used to transfer FSK data to the µC from whichever channel is enabled for DSC reception (either the Rx1 channel, Rx2 channel or the external FSK interface).

Rx1_FIFO_Threshold register: 8-bit write only. C-BUS Address $31
Rx2_FIFO_Threshold register: 8-bit write only. C-BUS Address $41
FSK_FIFO_Threshold register: 8-bit write only. C-BUS Address $51
All registers set to $1F on reset.

Bit:	7	6	5	4	3	2	1	0
$31:	Reserved, set to 000			Rx1 FIFO threshold level				
$41:	Reserved, set to 000			Rx2 FIFO threshold level				
$51:	Reserved, set to 000			FSK FIFO threshold level				

The receive FIFO threshold registers are used to set the level at which a “FIFO nearly full” warning is triggered for each of the three receive channel FIFOs. If the number of bytes in a FIFO is greater than the value in bits 4-0 of the associated threshold register then the FIFO Trigger flag (bit 7 of the associated status register) will be set high. These flags can also be used to generate interrupts. Bits 7-5 of the threshold registers should be set to 0.

Rx1_Status register: 16-bit read only. C-BUS Address $32
Rx2_Status register: 16-bit read only. C-BUS Address $42
FSK_Status register: 16-bit read only. C-BUS Address $52
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$32:	Rx1 Over-flow	Rx1 Under-flow	Rx1 FIFO Fill Level						Rx1 FIFO Trigger	0	0	0	0	Rx1 State		
$42:	Rx2 Over-flow	Rx2 Under-flow	Rx2 FIFO Fill Level						Rx2 FIFO Trigger	0	0	0	0	Rx2 State		
$52:	FSK Over-flow	FSK Under-flow	FSK FIFO Fill Level						FSK FIFO Trigger	0	0	0	0	0	0	0

Rx1/Rx2/FSK_Status register b15: Overflow
These bits get set high if the µC does not read the associated FIFO quickly enough and allows it to overflow, causing received data to be lost. Each overflow bit gets cleared as soon as the associated status register is read.

Rx1/Rx2/FSK_Status register b14: Underflow
These bits get set high if the µC attempts to read from the associated FIFO when it is already empty. The Rx1/Rx2/FSK underflow bits get cleared as soon as the associated status register is read.

Note: Each receive channel status register has b15 and b14 ORed together for the purpose of generating an interrupt.

Rx1/Rx2/FSK_Status register b13-8: FIFO Fill Level
This shows how many bytes are in the associated FIFO. The number will be in the range 0 to 32.

Rx1/Rx2/FSK_Status register b7: FIFO Trigger
These bits will be high if the associated FIFO fill level is greater than the threshold level, i.e. These bits can generate an interrupt.

- Rx1 FIFO Trigger is set high if (Rx1_Status[13-8] > Rx1_FIFO_Threshold[4-0])
- Rx2 FIFO Trigger is set high if (Rx2_Status[13-8] > Rx2_FIFO_Threshold[4-0])
- FSK FIFO Trigger is set high if (FSK_Status[13-8] > FSK_FIFO_Threshold[4-0])

Rx1/Rx2_Status register b2-0: AIS State
Indicates the current state of the Rx1 and Rx2 channel (AIS only).

b2	b1	b0	
0	0	0	Idle
0	0	1	Receiving
0	1	0	Error: Message too long or missing end flag (burst mode)
0	1	1	Error: CRC mismatch (burst mode)
1	0	0	Error: New frame header found but both message buffers full (burst mode)
1	0	1	Error: End flag not on byte boundary (burst mode)

Rx1_Slot register: 16-bit read only. **C-BUS Address $33**
Rx2_Slot register: 16-bit read only. **C-BUS Address $43**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$33:	Slot number in which a valid training sequence + start flag was detected in channel Rx1															
$43:	Slot number in which a valid training sequence + start flag was detected in channel Rx2															

Rx1_Sample register: 16-bit read only. **C-BUS Address $34**
Rx2_Sample register: 16-bit read only. **C-BUS Address $44**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$34:	Sample number in which a valid training sequence + start flag was detected in channel Rx1															
$44:	Sample number in which a valid training sequence + start flag was detected in channel Rx2															

When receiving an AIS message on the Rx1 channel, the CMX910 writes the time at which the last bit of the start flag is detected into the Rx1_Slot and Rx1_Sample registers (this is only done if a the start flag is preceded by a valid training sequence). The Rx2_Slot and Rx2_Sample registers perform the same function for the Rx2 channel. Note: the delay through the CMX910's internal filters mean that the reported Slot/Sample number will be approximately 52 sample periods (10.4 bit periods) later than the arrival time of the last bit of the start flag at the device pins.

Rx1_Bytes register: 16-bit read only. **C-BUS Address $35**
Rx2_Bytes register: 16-bit read only. **C-BUS Address $45**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$35:	Number of message bytes received in channel Rx1 (AIS burst mode only)															
$45:	Number of message bytes received in channel Rx2 (AIS burst mode only)															

Registers Rx1_Bytes and Rx2_Bytes are only used in AIS burst mode and indicate the number of message bytes that are available for the µC to read out.

Rx1_Control register: 8-bit write only. **C-BUS Address $36**
Rx2_Control register: 8-bit write only. **C-BUS Address $46**
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
$36:	Rsvd, set to 0	Rx1 DC offset correction		Rx1/2 Burst Enab	Rx1 mode		Rx1 FIFO Clear	Rx1 State Reset
$46:	Rsvd, set to 0	Rx2 DC offset correction		Rsvd, set to 0	Rx2 mode		Rx2 FIFO Clear	Rx2 State Reset

Rx1/Rx2_Control register b6-5: DC offset correction
These bits are used to independently configure the I/Q DC offset correction mode for the Rx1 and Rx2 channels.

b6	b5	
0	0	I/Q DC offset correction: Reset and hold
0	1	I/Q DC offset correction: Hold
1	0	I/Q DC offset correction: Run slowly
1	1	I/Q DC offset correction: Run quickly

Rx1_Control register b4: Rx1 and Rx2 Burst Mode Enable
Bit 4 of the Rx1_Control register is used to select between burst and raw mode operation for *both* the Rx1 and Rx2 channel (AIS only): b4 = 1 for burst mode, b4 = 0 for raw mode.

Rx1/Rx2_Control register b3-2: Rx1 and Rx2 Mode
These bits are used to independently configure the modulation type for the Rx1 and Rx2 channels.

b3	b2	
0	0	AIS, 25kHz channel
0	1	AIS, 12.5kHz channel
1	0	DSC
1	1	Reserved, do not use

Rx1/Rx2_Control register b1: FIFO Clear
Data written to these bits do not get stored; instead, writing a 1 to either of these bits generates a reset pulse which clears the Rx1 or Rx2 FIFO and resets the FIFO fill level (Rx1_Status or Rx2_Status b13-8) to zero.

Rx1/Rx2_Control register b0: Rx State Reset

Immediately after power up, the Rx1 and Rx2 channels must be initialised by writing a 1 to bit 0 (Rx State Reset) of the Rx1_Control or Rx2_Control registers. Writing a 1 to the Rx State Reset bits can also be done at any other time in order to immediately terminate an active reception in that channel and to cause the Rx state (AIS raw or burst mode) to revert to "Idle". Any internal states related to receive will be cleared and the internal message buffers (AIS burst mode) will be wiped. The receive FIFOs are not cleared by writing a 1 to the Rx State Reset bits, that can be done if necessary by writing a 1 to Rx1_Control bit 1 or Rx2_Control bit 1 (AIS raw/burst mode) or FSK_Control bit 1 (DSC mode). Note: after a 1 is written to bit 0 of the Rx1_Control (or Rx2_Control), it will take up to 250µs before that channel is reset properly and data stops being written to the associated FIFO (Rx1_FIFO, Rx2_FIFO, or FSK_FIFO). Only after that time should the FIFO be cleared or the Rx1_Control (or Rx2_Control) register be written to again.

Rx1_FreqErr register: 16-bit read only. **C-BUS Address $37**
Rx2_FreqErr register: 16-bit read only. **C-BUS Address $47**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$37:	Rx1 channel frequency error (Hz)															
$47:	Rx2 channel frequency error (Hz)															

Registers Rx1_FreqErr and Rx2_FreqErr are only valid for AIS burst mode reception, and indicate the frequency error of the received carrier (relative to the local oscillator) in units of Hertz. The values are calculated from the received training sequence and start flag and are reported in 2's complement format.

Rx1_RSSI register: 16-bit read only. **C-BUS Address $38**
Rx2_RSSI register: 16-bit read only. **C-BUS Address $48**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$38:	Rx1 channel RSSI															
$48:	Rx2 channel RSSI															

Registers Rx1_RSSI and Rx2_RSSI are valid for AIS burst and raw mode reception, and indicate the signal strength of the received carrier. The RSSI values are calculated over a specified window within each slot, as set up by the "rssi_window" special command function (see section 5.12).

The RSSI values are calculated by accumulating the I/Q vector magnitudes from the associated channel filter outputs at each sample point (i.e. every 20.833µs) over the specified window. Before being added to the running total, each vector magnitude is multiplied by a fixed gain of 0.4117 and a user-programmable gain "RSSI_gain" of between 0 and 1 (this is also set up using a special command function). In the case of overflow, the accumulated RSSI value saturates at 32767 ($7FFF). The total accumulated signal strength value is given by:

$$RSSI = \sum_{n=RSSI\_start}^{RSSI\_start+length-2} \left(\sqrt{I_n^{\,2} + Q_n^{\,2}} \times 0.4117 \times RSSI\_gain \right)$$

With a DC voltage of ±1.7V (differential) applied to the input of the I or Q ADC, the corresponding I or Q channel filter output will be approximately ±30000.

Rx1_RSSI register: 8-bit read only. **C-BUS Address $38**
Rx2_RSSI register: 8-bit read only. **C-BUS Address $48**
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
$38:	Rx1 channel RSSI							
$48:	Rx2 channel RSSI							

Registers Rx1_RSSI and Rx2_RSSI are only valid for AIS burst mode reception, and indicate the signal strength of the received carrier in units of dB (an I/Q vector magnitude of 0.5V at the device pins will give an RSSI value of approximately 112). The values are calculated from the received training sequence and start flag.

FSK_Control register: 8-bit write only. **C-BUS Address $53**
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
	Reserved, set to 000000						FSK FIFO Clear	En FSK I/F

FSK_Control register b1: FSK FIFO Clear
Data written to this bit does not get stored; instead, writing a 1 to this bit generates a reset pulse which clears the FSK FIFO and resets the FSK FIFO fill level (FSK_Status b13-8) to zero.

FSK_Control register b0: Enable FSK Interface
Setting b0 = 1 enables the external FSK interface circuit, allowing serial data from the FSK_RXD pin to be packed into bytes and transferred to the µC via the FSK_FIFO (subject to the FSK_DET and FSK_MUTE pins being in the correct state). If b0 = 0, data will instead be loaded into the FSK_FIFO from whichever, if either, of the main Rx1 or Rx2 receive channels is configured for DSC reception.

5.6.2 AIS Raw Mode Receive

The operation of receive channel Rx1 in AIS raw mode is described below (the operation of receive channel Rx2 in AIS raw mode is essentially identical to that of Rx1, but is controlled through its own individual C-BUS registers).

In AIS raw mode the CMX910 searches the Rx1 channel for a header (training + start flag sequence) to detect the start of a message, then transfers the received data (starting with the three training bytes and the start flag, then all subsequent demodulated bytes) directly to Rx1_FIFO as soon as available. This byte stream continues even after the end of a message and in the absence of a received signal (the data will then be indeterminate), but will cease while the Rx1 channel is *sleeping* (section 5.4.3). Resynchronisation of the Rx1 data stream occurs each time the CMX910 detects a valid training sequence and start flag on the channel (at which point the Rx1_Slot and Rx1_Sample registers are updated) but no other indication is given that valid messages are being received; it is the responsibility of the µC to detect the training and start flag bytes in the received data stream, and to perform all HDLC/NRZI decoding, CRC checking and end flag detection.

Bit ordering of the received bytes in AIS raw mode is the same as in Tx AIS raw mode, i.e. the received bits are packed into bytes *most significant bit first*. As the AIS message structure requires message bytes to be transmitted least significant bit first, the µC must ensure that during the process of HDLC/NRZI decoding that the resulting data bytes are correctly reversed. Depending on the configuration of the remote transmitter, one of four different types of NRZI encoded training bytes may be received – this

situation arises because the AIS specification allows a transmitter's NRZI encoder to start in either of its two quiescent states, and the pre-NRZI encoded training bytes can also be one of two different types ($55 or $AA). Therefore, for any particular message, the three received training bytes in AIS raw mode will all be either $33, $66, $99 or $CC, although the first few bits may be corrupted depending on the power-up characteristics of the remote transmitter and local receiver circuits.

In AIS raw mode, whenever an Rx1 state reset is performed (by setting Rx1_Control b0 = 1) the channel state (Rx1_Status b2-0) becomes *Idle.* This changes to *Receiving* when the first valid training sequence and start flag have been detected, where it remains until another Rx1 state reset occurs.

5.6.3 AIS Burst Mode Receive

The operation of receive channel Rx1 in AIS burst mode is described below (the operation of receive channel Rx2 in AIS burst mode is essentially identical to that of Rx1).

In AIS burst mode the Rx1 channel state (Rx1_Status b2-0) changes to *Receiving* when a valid training sequence and start flag are detected. The CMX910 then performs NRZI decoding and bit destuffing on the received data stream, and calculates the CRC checksum. At the end of the message the receive channel state changes from *Receiving* to either *Idle* or one of four error states (below). At the same time, an "Rx1 State Alert" interrupt is flagged.

The four error conditions that the CMX910 can detect in a received message (in burst mode) are:

1. *Message too long or missing end flag*. This indicates that the received message, after bit destuffing, is too long to fit into an internal 172 byte message buffer. This condition could be caused by a missing or corrupted end flag.
2. *CRC mismatch*. This indicates that the received frame checksum does not match that calculated by the CMX910, most probably as the result of one or more message bits being corrupted.
3. *New frame header found but both message buffers full*. This happens if both internal message buffers are in use when another message arrives. This is caused by a failure of the µC to read the received messages out quickly enough.
4. *End flag not on byte boundary*. This indicates that the received message, after bit destuffing, is not a multiple of 8 bits. Assuming that the message was transmitted correctly, the most probable cause of this error is an end flag being missed due to noise, and a subsequent message's start flag being misidentified as the expected end flag.

If one of these four error conditions is detected in a received message the CMX910 discards the message data and, after flagging the "Rx1 State Alert" interrupt, continues searching for the next training sequence and start flag.

If a message with no error is found the Rx1 channel state changes from *Receiving* to *Idle* (causing an "Rx1 State Alert" interrupt); the decoded message, comprising the three training sequence bytes, start flag, message payload, end flag and CRC bytes, is then copied to one of the CMX910's internal message buffers. When its turn comes around to be read out, an "Rx1 Burst Available" interrupt is generated. At this point the CMX910 updates the registers Rx1_Slot, Rx1_Sample, Rx1_Bytes, Rx1_FreqErr and Rx1_RSSI with the values calculated for that message, then begins transferring the data from the internal message buffer to Rx1_FIFO. Note: a new message will only generate an "Rx1 Burst Available" interrupt when any previous message has been read out from Rx1_FIFO in its entirety.

For any particular message, the three received (NRZI-decoded) training bytes in AIS burst mode will all be either $55 or $AA depending on the configuration of the remote transmitter, although the first few bits may be corrupted depending on the power-up characteristics of the remote transmitter and local receiver circuits.

5.6.4 DSC Receive (Main Channel)

Either the Rx1 or Rx2 channel can be configured for DSC reception. The CMX910 first applies 6dB/octave de-emphasis to the received signal, then demodulates the resulting 1200 baud NRZ FSK data. Only one of the channels at a time must be configured for DSC reception. The received data is packed into 8-bit bytes for forward transmission to the µC. The CMX910 makes no attempt to align data bits within the bytes or perform dot pattern or data phasing detection, those functions must be performed by the host µC. No attempt is made to correctly align data, it is simply packed into bytes (least significant bit first) as it arrives.

Note: when the Rx1 or Rx2 channel is configured for DSC operation, the received data is forwarded to the µC through FSK_FIFO, *not* Rx1_FIFO or Rx2_FIFO. This prevents the DSC reception from corrupting any AIS data that may still be present in Rx1_FIFO or Rx2_FIFO.

5.6.5 DSC Receive (External FSK Interface)

The CMX910's external FSK interface retimes asynchronous NRZ FSK data from an external 1200 baud demodulator such as the FX604, and is intended for use as a third parallel receive channel for DSC reception in the case that both of the main receive channels are assigned to AIS operation. The data from the external FSK demodulator is applied to the FSK_RXD pin and gets packed into 8-bit bytes (least significant bit first) before being loaded into FSK_FIFO for forward transmission to the µC. The CMX910's FSK interface circuit does not align data bits within the bytes or perform dot pattern or data phasing detection, those functions must be performed by the host µC.

Two further input pins are provided to prevent data from being decoded in the absence of a valid FSK signal: the FSK_MUTE input is intended for connection to the DSC radio sub-system, and should go high to indicate no received signal; the FSK_DET input is intended for connection to the FSK demodulator and indicates that valid FSK data is being received. Data will only be written to the FSK FIFO if FSK_MUTE = 0 and FSK_DET = 1.

The CMX910 samples data on the FSK_RXD pin half a bit period after each transition (i.e. in the middle of a received bit), and every bit period thereafter until another transition occurs. A transition on the FSK_RXD pin should occur at least once in every ten bit periods in order to maintain reliable synchronisation with the incoming bit stream. The CMX910's data retiming circuit can tolerate an error of 1.5% in the input data baud rate.

© 2006 CML Microsystems Plc

5.7 Auxiliary A-to-D Converter

A 10-bit ADC is provided to assist in a variety of measurement and control functions. The ADC includes an internal sample-and-hold circuit and is designed to produce a digital output proportional to the analogue supply (AV_{DD}), full scale being the positive supply. An input multiplexer allows the input to be selected from one of five sources. Three of these inputs (ADCs 2, 1 and 0) are provided with an uncommitted op-amp, each of which can be disabled if required. There are five ADC data output registers (ADC4-0), one for each ADC channel. Control and digital data output is via the C-BUS.

The auxiliary ADC can be operated in either single-shot mode, where the µC can initiate a single conversion of each enabled ADC channel, or in auto convert mode, where the enabled ADC channels are converted in a continuous loop. By default, the time taken to convert *each* ADC channel is (529 ÷ 48)µs ≈ 11.021µs. In addition to this, at the beginning of a single-shot conversion or when auto convert mode is first enabled, two dummy conversion cycles (≈ 22.042µs) are performed before any actual conversion begins, allowing time for the ADC's internal bias circuits to power up and settle. The conversion rate can be altered using one of the CMX910's Special Commands (section 5.12).

The ADC and its sample-and-hold automatically power down when not in use. The three uncommitted op-amps are only powered down when they are disabled, using bits in register ADC_Control2.

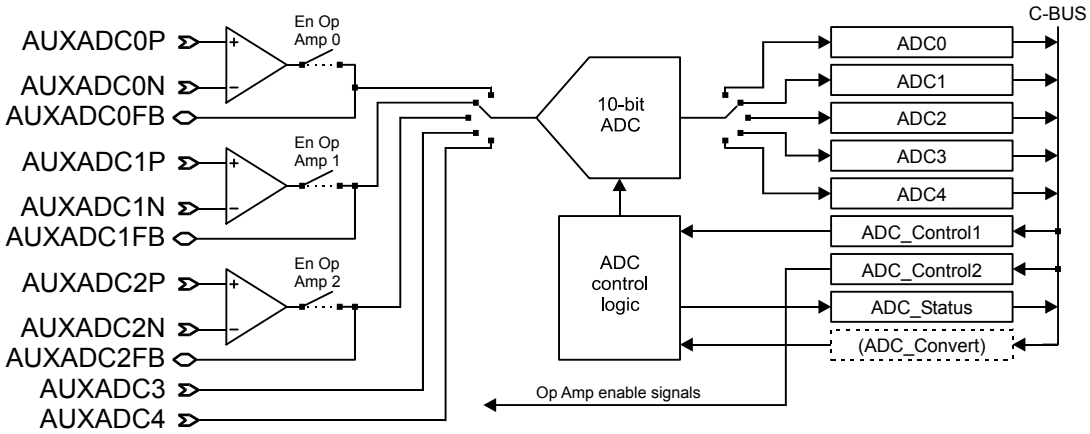


Figure 12 Auxiliary ADC

ADC0 data register: 16-bit read only. C-BUS Address $60
ADC1 data register: 16-bit read only. C-BUS Address $61
ADC2 data register: 16-bit read only. C-BUS Address $62
ADC3 data register: 16-bit read only. C-BUS Address $63
ADC4 data register: 16-bit read only. C-BUS Address $64
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$60:	0	0	0	0	0	0	ADC0 data									
$61:	0	0	0	0	0	0	ADC1 data									
$62:	0	0	0	0	0	0	ADC2 data									
$63:	0	0	0	0	0	0	ADC3 data									
$64:	0	0	0	0	0	0	ADC4 data									

ADC_Control1 register: 8-bit write only. **C-BUS Address $65**
All bits cleared to 0 on reset.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00		Auto Conv.	En ADC4	En ADC3	En ADC2	En ADC1	En ADC0

ADC_Control1 register b5: ADC Auto Convert
Setting b5 = 1 causes the ADC to continuously convert the enabled channels in ascending order. While in auto convert mode, the channel enable bits (ADC_Control1 b4-0) can be changed at any time, and will update the ADC conversion scheduler immediately. Auto convert is terminated by setting b5 = 0.

ADC_Control1 register b4-0: Enable ADC4-0
Writing a 1 to these bits selects the corresponding ADC channels for conversion.

ADC_Control2 register: 8-bit write only. **C-BUS Address $66**
All bits cleared to 0 on reset.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00000					En Op Amp2	En Op Amp1	En Op Amp0

ADC_Control2 register b2-0: Enable Op Amp 2-0
The three uncommitted op-amps are independently controllable using these bits: setting a bit to 1 enables the uncommitted op-amp in the corresponding ADC channel; setting the bit to 0 powers down the corresponding op-amp and puts its output into a high impedance state. When an op-amp is powered down, its “output” pin (AUXADC2FB, AUXADC1FB, and AUXADC0FB) may be used as an input to the ADC.

ADC_Status register: 8-bit read only. **C-BUS Address $67**
Reset state is $01.

Bit:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	ADC Ready

ADC_Status register b0: ADC Ready
This bit goes low when a single-shot conversion is started using the ADC_Convert command, and remains low until conversions on all enabled ADC channels have completed. This bit can be polled by the µC to find out when the conversion sequence has completed; it is also connected to the interrupt generator block. This bit does *not* go low when auto convert mode is enabled, or if an ADC_Convert command is issued while auto convert mode is enabled.

ADC_Convert command (no data) **C-BUS Address $68**
Issuing this C-BUS command causes a single-shot conversion to be initiated in which a single conversion is done on each of the enabled ADC channels in ascending order. This command is ignored if it is issued while the ADC is in auto convert mode.

5.8 Auxiliary D-to-A Converters

The CMX910 is provided with five general purpose 10-bit digital-to-analogue converters (DAC0–4) to assist in a variety of control functions. These DACs are independent of each other, and can be individually enabled or powered down. The DACs are designed to provide an output as a proportion of the analogue supply voltage, depending on the DAC's data register setting: a value of 0 drives that DAC's output to AV_{SS}; a value of 1023 ($3FF_{16}$) drives the output to AV_{DD}.

DAC0 has an additional *ramp mode* feature. With this mode enabled, the contents of an internal 64 word × 10 bit DAC RAM can be transferred in ascending order to the DAC0 data register (a *ramp-up* sequence), or in descending order (a *ramp-down* sequence). The rate at which the DAC RAM contents are copied to DAC0 can be programmed through the C-BUS register DAC0_Timestep. The DAC0 ramp-up and ramp-down facility is particularly useful for controlling the profile of the transmitter power at the beginning and end of a transmit slot, in order to minimise adjacent-channel splatter. The DAC0 ramp-up and ramp-down can be initiated automatically by the CMX910 as part of its transmit event sequencer, or can be configured through the DAC_Control register to operate via the CBUS commands DAC0_Rampup and DAC0_Rampdown.

The default contents of the DAC RAM can be changed by first putting DAC0 into *DAC RAM load mode*. This resets the internal DAC RAM address pointer to 0. The RAM contents can then be modified by repeatedly writing to C-BUS location DAC_RAM_Load – the address pointer is automatically incremented after each word that is written.

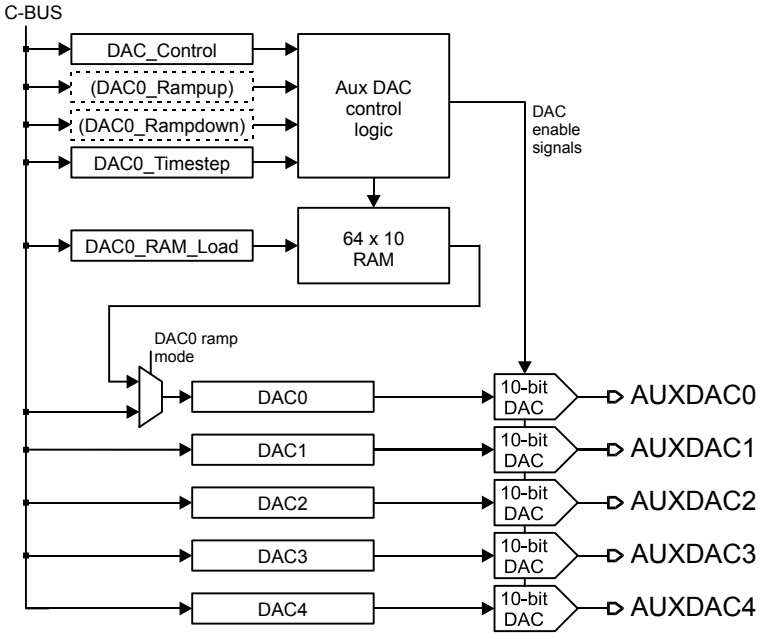


Figure 13 Auxiliary DACs

DAC0 data register: 16-bit write only. **C-BUS Address $70**
DAC1 data register: 16-bit write only. **C-BUS Address $71**
DAC2 data register: 16-bit write only. **C-BUS Address $72**
DAC3 data register: 16-bit write only. **C-BUS Address $73**
DAC4 data register: 16-bit write only. **C-BUS Address $74**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$70:	Reserved, set to 000000						DAC0 data (written by C-BUS, or from DAC RAM in ramp mode)									
$71:	Reserved, set to 000000						DAC1 data									
$72:	Reserved, set to 000000						DAC2 data									
$73:	Reserved, set to 000000						DAC3 data									
$74:	Reserved, set to 000000						DAC4 data									

DAC_Control register: 8-bit write only. **C-BUS Address $75**
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
	DAC RAM Load	C-BUS Ramp Control	DAC0 Ramp Mode	En DAC4	En DAC3	En DAC2	En DAC1	En DAC0

DAC_Control register b7: DAC RAM Load
Setting b7 = 1 to this bit causes DAC0 to immediately enter DAC RAM load mode, causing the internal DAC RAM address pointer to be reset to 0. The DAC RAM can then be loaded by repeatedly writing data to the C-BUS DAC_RAM_Load register. While in DAC RAM load mode, any attempt to write directly to the DAC0 data register will be ignored, as will any attempt to initiate a ramp-up or ramp-down sequence (whether issued automatically by the CMX910 or from the C-BUS). As soon as the µC has finished loading the DAC RAM, DAC_Control register b7 can be taken low again.

DAC_Control register b6: C-BUS Ramp Control
If DAC0 is in ramp mode, the state of this register bit determines how the ramp-up and ramp-down functions are controlled: b6 = 0 for automatic control by the CMX910, b6 = 1 for control via the C-BUS commands DAC0_Rampup and DAC0_Rampdown.

DAC_Control register b5: DAC0 Ramp Mode
Setting DAC_Control b5 = 1 at the same time as DAC_Control b7 = 0 puts DAC0 into ramp mode. The internal DAC RAM address pointer immediately gets reset to 0, an initial read of the DAC RAM is performed and the resulting data is transferred to the DAC0 data register. The circuit then responds to any subsequent ramp-up and ramp-down commands. While in ramp mode, any attempt to write directly to the DAC0 data register will be ignored, as will any attempt to write to the DAC RAM using the DAC_RAM_Load register.

Note: if both DAC_Control b7 = 0 and b5 = 0, the µC is able to write directly to the DAC0 data register, and any ramp-up or ramp-down commands or attempts to write to the DAC RAM will be ignored:

b7	b6	b5	
0	x	0	DAC0 data register can be written directly by µC
0	0	1	DAC0 in ramp mode, automatically controlled by the CMX910
0	1	1	DAC0 in ramp mode, controlled by the µC
1	x	x	DAC0 in DAC RAM load mode

DAC_Control register b4-0: Enable DAC4-0

Writing a 1 to these bits powers up the corresponding DAC analogue circuit, writing a 0 powers down the DAC and puts the DAC output pin into a high impedance state.

DAC0_Rampup command (no data) **C-BUS Address $76**
DAC0_Rampdown command (no data) **C-BUS Address $77**

These two commands are enabled only if DAC_Control register b7-5 = 011. In that case, issuing a DAC0_Rampup command causes DAC0 to begin ramping up (RAM[0→63] copied to DAC0 data register), and DAC0_Rampdown causes DAC0 to begin ramping down (RAM [63→0] copied to DAC0 data register). If a DAC0_Rampup command is issued while DAC0 is in the process of ramping down, or vice-versa, the ramp process immediately reverses direction.

The CMX910 ignores any DAC0_Rampup commands issued when DAC0 is already ramped up, or any DAC0_Rampdown commands issued when DAC0 is already ramped down.

DAC0_Timestep register: 8-bit write only. **C-BUS Address $78**

All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
	DAC0 ramp timestep							

The contents of the DAC0_Timestep register determine the rate at which the DAC RAM data is transferred to the DAC0 data register during a ramp-up or ramp-down sequence:

$$\text{Time between each data transfer} = (\text{DAC0\_Timestep} + 1) \times 0.25\mu s$$

The time taken for the entire ramp-up or ramp-down process to complete is therefore:

$$t_{RAMP} = 63 \times (\text{DAC0\_Timestep} + 1) \times 0.25\mu s$$

DAC_RAM_Load register: 16-bit write only (data-streaming). C-BUS Address $79

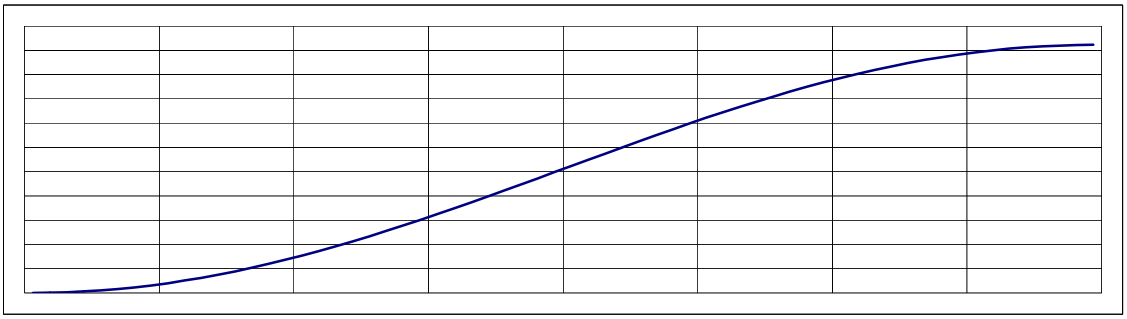
Bit:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved, set to 000000						DAC RAM data									

The contents of the DAC RAM can be modified by writing to the DAC_RAM_Load C-BUS register (only the least significant 10 bits are used). To do this, the device must first be put into DAC RAM load mode by setting DAC_Control b7 = 1 (this resets the internal RAM address pointer to 0). After each 10-bit word is written to the DAC RAM the internal RAM address pointer is automatically incremented, ready for the next word to be entered. A total of 64 data writes to DAC_RAM_Load are necessary to modify the entire RAM contents, beginning at RAM address 0 and finishing at address 63. Afterwards, the µC must set DAC_Control b7 = 0 to exit DAC RAM load mode. The DAC_RAM_Load register supports C-BUS data-streaming.

The DAC RAM contents default to a raised cosine profile: $y_n = \frac{1023}{2} \times \left(1 - \cos\left(\frac{n\pi}{63}\right)\right)$

Default DAC RAM contents after reset (hexadecimal)															
0 000	1 001	2 003	3 006	4 00A	5 010	6 017	7 01F	8 028	9 033	10 03E	11 04B	12 059	13 068	14 078	15 089
16 09A	17 0AD	18 0C1	19 0D5	20 0EA	21 100	22 116	23 12D	24 145	25 15D	26 175	27 18E	28 1A7	29 1C0	30 1D9	31 1F3
32 20C	33 226	34 23F	35 258	36 271	37 28A	38 2A2	39 2BA	40 2D2	41 2E9	42 2FF	43 315	44 32A	45 33E	46 352	47 365
48 376	49 387	50 397	51 3A6	52 3B4	53 3C1	54 3CC	55 3D7	56 3E0	57 3E8	58 3EF	59 3F5	60 3F9	61 3FC	62 3FE	63 3FF



5.9 Interrupt Generator

The CMX910 has sixteen internal interrupt sources which provide status information to the µC – these are accessible through a C-BUS read register. The interrupt flags may also be enabled to drive the open-drain IRQN output pin.

Interrupt register: 16-bit read only. **C-BUS Address $80**
Register gets set to $0080 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Spcl Cmnd Done	Aux ADC Done	Rx2 State Alert	Rx1 State Alert	Tx FIFO Error	FSK FIFO Error	Rx2 FIFO Error	Rx1 FIFO Error	Tx FIFO Trigger	FSK FIFO Trigger	Rx2 FIFO Trigger	Rx1 FIFO Trigger	Tx Done	Rx2 Burst Avail.	Rx1 Burst Avail.	Nudge Done

Interrupt register b15: Special Command Done
This bit goes high to indicate that a command issued to the Special Command Interface has completed. This bit gets cleared automatically after being read.

Interrupt register b14: Aux ADC Done
This bit goes high to indicate that the Auxiliary ADC has finished doing a sequence of conversions after an ADC_Convert command was issued. This bit gets cleared automatically after being read.

Interrupt register b13: Rx2 State Alert
This bit goes high in AIS burst mode only, and indicates that the Rx2 channel has finished receiving a message – the Rx2_Status register can be read to determine whether a valid message was received or an error occurred. This bit gets cleared automatically after being read.

Interrupt register b12: Rx1 State Alert
This bit goes high in AIS burst mode only, and indicates that the Rx1 channel has finished receiving a message – the Rx1_Status register can be read to determine whether a valid message was received or an error occurred. This bit gets cleared automatically after being read.

Interrupt register b11: Tx FIFO Error
This bit goes high to indicate that an overflow or an underflow error has occurred in the Tx channel FIFO. This can be determined by reading the Tx_Status register (bits 15 and 14). This bit gets cleared automatically after being read.

Interrupt register b10: FSK FIFO Error
This bit goes high to indicate that an overflow or an underflow error has occurred in the FSK channel FIFO. This can be determined by reading the FSK_Status register (bits 15 and 14). This bit gets cleared automatically after being read.

Interrupt register b9: Rx2 FIFO Error
This bit goes high to indicate that an overflow or an underflow error has occurred in the Rx2 channel FIFO. This can be determined by reading the Rx2_Status register (bits 15 and 14). This bit gets cleared automatically after being read.

Interrupt register b8: Rx1 FIFO Error
This bit goes high to indicate that an overflow or an underflow error has occurred in the Rx1 channel FIFO. This can be determined by reading the Rx1_Status register (bits 15 and 14). This bit gets cleared automatically after being read.

Interrupt register b7: Tx FIFO Trigger
This bit is connected directly to Tx_Status b7, and goes high to indicate that the Tx FIFO fill level has dropped below a user-defined threshold (Tx_Status[13-8] ≤ Tx_FIFO_Threshold[4-0]). Note that this bit is level triggered, it does *not* get cleared by being read.

Interrupt register b6: FSK FIFO Trigger
This bit is connected directly to FSK_Status b7, and goes high to indicate that the FSK FIFO fill level has exceeded a user-defined threshold (FSK_Status[13-8] > FSK_FIFO_Threshold[4-0]). Note that this bit is level triggered, it does *not* get cleared by being read.

Interrupt register b5: Rx2 FIFO Trigger
This bit is connected directly to Rx2_Status b7, and goes high to indicate that the Rx2 FIFO fill level has exceeded a user-defined threshold (Rx2_Status[13-8] > Rx2_FIFO_Threshold[4-0]). Note that this bit is level triggered, it does *not* get cleared by being read.

Interrupt register b4: Rx1 FIFO Trigger
This bit is connected directly to Rx1_Status b7, and goes high to indicate that the Rx1 FIFO fill level has exceeded a user-defined threshold (Rx1_Status[13-8] > Rx1_FIFO_Threshold[4-0]). Note that this bit is level triggered, it does *not* get cleared by being read.

Interrupt register b3: Tx Done
This bit goes high to indicate that a transmit operation has completed or that an error condition has occurred. This bit gets cleared automatically after being read.

Interrupt register b2: Rx2 Burst Available
This bit goes high to indicate that an AIS packet is available in channel Rx2 (AIS burst mode only). This bit gets cleared automatically after being read.

Interrupt register b1: Rx1 Burst Available
This bit goes high to indicate that an AIS packet is available in channel Rx1 (AIS burst mode only). This bit gets cleared automatically after being read.

Interrupt register b0: Nudge Done
This bit goes high to indicate that a requested slot/sample nudge operation has been completed. This bit gets cleared automatically after being read.

Interrupt_Enable register: 16-bit write only. **C-BUS Address $81**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IRQN En 15	IRQN En 14	IRQN En 13	IRQN En 12	IRQN En 11	IRQN En 10	IRQN En 9	IRQN En 8	IRQN En 7	IRQN En 6	IRQN En 5	IRQN En 4	IRQN En 3	IRQN En 2	IRQN En 1	IRQN En 0

Interrupt_Enable register b15-0: Interrupt Enable Bits
The setting in this register determines which of the bits in the Interrupt register can cause a host µC interrupt – setting a bit high in the Interrupt_Enable register allows the associated bit in the Interrupt register to drive the IRQN pin low. The Interrupt_Enable register can be written at any time, and can be used to enable or disable any combination of the sixteen interrupt sources. Note: the value written to the Interrupt_Enable register does not affect the contents of the Interrupt register.
IMPORTANT NOTICE: If an interrupt occurs whilst the Interrupt register is being read by the host controller, there is a possibility that the new interrupt may be lost. To minimise the chances of this happening, it is recommended that the Interrupt register should only be read when the IRQN pin goes active (the interrupt register should NOT be polled). It is further recommended that when the host microcontroller is waiting for an interrupt, a timeout routine is implemented.

5.10 Device Enable Port

The device enable port (pins ENAB5 – ENAB0) provides for the timed control of peripheral RF circuits that is required for TDMA operation. The ENAB5 – ENAB0 pins are digital outputs and will typically be used as enabling signals for the external receiver/transmitter circuits and power amplifier. By default, the CMX910 will automatically control all six device enable pins. In conjunction with the automatic PA ramping feature of DAC0, this greatly simplifies the control of the RF circuits. If desired, individual pins of the device enable port may be configured to be under µC control via the C-BUS.

ENAB register: 8-bit write only. **C-BUS Address $90**
All bits cleared to 0 on reset.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00		ENAB 5	ENAB 4	ENAB 3	ENAB 2	ENAB 1	ENAB 0

ENAB register b5-0: ENAB5 – ENAB0 Data
By writing to the bits in the ENAB register, the µC can directly control the logic state of the corresponding ENAB5 – ENAB0 pins. This only happens for those pins which have been configured to be under C-BUS control, i.e. those whose corresponding bit in the ENAB_Mask register is set high. Note: if the corresponding bit in the ENAB_Invert register is also set high, the logic level appearing at the device pin will be the *inverse* of the data in the ENAB register bit.

ENAB_Mask register: 8-bit write only. **C-BUS Address $91**
All bits cleared to 0 on reset.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00		ENAB Mask 5	ENAB Mask 4	ENAB Mask 3	ENAB Mask 2	ENAB Mask 1	ENAB Mask 0

ENAB_Mask register b5-0: ENAB5 – ENAB0 Mask
Each bit that is set high in the ENAB_Mask register causes the corresponding device enable pin to be under direct control of the µC. Those bits in the ENAB_Mask register that are low cause the corresponding pin to be under the automatic control of the CMX910.

ENAB_Invert register: 8-bit write only. **C-BUS Address $92**
All bits cleared to 0 on reset.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00		ENAB Invert 5	ENAB Invert 4	ENAB Invert 3	ENAB Invert 2	ENAB Invert 1	ENAB Invert 0

ENAB_Invert register b5-0: ENAB5 – ENAB0 Inversion Control
The polarity of the ENAB5 – ENAB0 pins are individually controlled through this register. By setting a bit high in the ENAB_Invert register, the logic level appearing on the corresponding device enable pin will be inverted. This inversion is applied whether the pin is under µC control or automatic CMX910 control.

5.11 C-BUS Expansion Port

The C-BUS expansion port facilitates the connection of the host µC to as many as six additional C-BUS compatible devices, while only requiring one additional C-BUS chip select pin. To operate the expansion port, the µC writes to the CBUS_Expand register through the CMX910's normal C-BUS port (SCLK, CDATA, CSN) – this can be done even when the CMX910's internal clocks are disabled. Subsequently, for each bit set in bits 5-0 of the CBUS_Expand register, taking the chip select expansion input (CSXN) low will cause the associated EXP5N – EXP0N output pin to go low. Each of the EXP5N – EXP0N pins can be connected to the (active low) chip-select input of another C-BUS compatible device. Usually only one bit in the CBUS_Expand register would be set high at a time. Setting more than one bit high may cause simultaneous selection of a number of C-BUS devices; in that case, care must be taken to avoid contention if a common RDATA line is used.

If expansion of the C-BUS is not required, the outputs EXP5N – EXP0N can be used as general purpose digital outputs by holding CSXN low and writing the required data pattern to the CBUS_Expand register. The data will then appear on the EXP5N-EXP0N pins in inverted form.

CBUS_Expand register: 8-bit write only. **C-BUS Address $A0**
All bits cleared to 0 on reset.
This register can be written while the CMX910's internal clocks are disabled.

Bit:

7	6	5	4	3	2	1	0
Reserved, set to 00		EXP5N enab	EXP4N enab	EXP3N enab	EXP2N enab	EXP1N enab	EXP0N enab

CBUS_Expand register b5-0: EXP5N – EXP0N Expansion Bus Enable
Each bit that is set to 1 causes the corresponding EXP5N – EXP0N output pin to be driven low when the CSXN input pin is driven low. Bits that are set to 0 cause the corresponding EXP5N – EXP0N output pin to be held high.

5.12 Special Command Interface

The Special Command Interface allows the CMX910 to perform the special tasks defined below. The interface comprises two 16-bit write registers and a 16-bit read register for data transfers and an 8-bit write register which is used to instruct the CMX910 which special command to perform.

When executing a special command that requires data to be transferred to the CMX910, the data must first be written to the SPC_In0/1 registers, then the command code should be written to the Special_Command register. The act of writing to the Special_Command register causes the CMX910 to begin processing the command, during which time the data in SPC_In0, SPC_In1 and Special_Command should not be changed. When the special command has completed, the "Special Command Done" bit in the Interrupt register goes high and the Special_Command register gets cleared, and the returned data (if any) will be available in SPC_Out0. The CMX910 is then ready to accept another special command.

SPC_In0 register: 16-bit write only. **C-BUS Address $B0**
SPC_In1 register: 16-bit write only. **C-BUS Address $B1**
SPC_Out0 register: 16-bit read only. **C-BUS Address $B2**
All bits cleared to 0 on reset.

Bit:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$B0:	Special data in 0 (least significant data word used by special command)															
$B1:	Special data in 1 (most significant data word used by special command)															
$B2:	Special data out 0 (data word returned by special command)															

Special_Command register: 8-bit write only. **C-BUS Address $B4**
All bits cleared to 0 on reset.

Bit:	7	6	5	4	3	2	1	0
	Special command code							

The following special commands are available for use by the host µC. Unspecified commands are reserved for production test and should not be used.

Special command code	SPC_ In0	SPC_ In1	SPC_ Out0	Notes
$04 (poke_inc)	•			Copies SPC_In0 into an internal table then increments the internal address pointer.
$08 (poke2_inc)	•	•		Copies SPC_In0 and SPC_In1 into two consecutive locations in an internal table then increments the internal address by 2.
$0D (set_aux_adc_speed)	•			Sets Aux ADC convert time: $t_{CONVERT}$ = ((22 × SPC_In0) + 1) ÷ 48 µs (per channel). Note: 24 ≤ SPC_In0 ≤ 256.
$0F (setup_tx_sequence)				Causes the CMX910 to process the previously loaded table of Tx events and configure its internal Tx logic accordingly.
$10 (load_tx_sequence)				Sets an internal address pointer to the base of the CMX910's Tx event sequence table, readying it to accept data.
$12 (rssi_window)	•	•		Sets the window within a slot over which Rx1 and Rx2 RSSI values are calculated. SPC_In0 defines the *RSSI_start* sample number, SPC_In1 defines the *RSSI_length* number, i.e. the number of samples to accumulate over.

Special command code	SPC_ In0	SPC_ In1	SPC_ Out0	Notes
$14 (setup_gain)	•	•		Sets the gain factor to be applied to each sample when calculating RSSI or CSTDMA signal strength: RSSI_gain = SPC_In0 ÷ 32768 CSTDMA_gain = SPC_In1 ÷ 32768 Note: The value written into SPC_In0/1 must be between 0 and 32767 ($7FFF).
$17 (read_message_length)			•	Used in AIS burst mode transmit only. Waits until the message buffer is ready then reports back (in SPC_Out0) the total number of message bits created.
$1F (eye_monitor)	•			Allows the Rx1 and Rx2 “eye diagrams” to be monitored for diagnostic purposes. When enabled, this function drives the demodulated Rx1 and Rx2 channel signals onto the Tx I and Q output pins respectively. SPC_In0 = $0000: Disable eye_monitor SPC_In0 = $0001: Enable eye_monitor

6. Supplementary Information

6.1 Glossary of Terms

ADC	Analogue-to-Digital Converter.
AIS	Automatic Identification System, an identification, location and communication system using the maritime VHF radio band.
baud	The number of symbols transmitted per second in a modulated signal. Not necessarily the same as bits per second.
bps	Bits Per Second, the speed at which bits are transmitted over a data channel.
BT	Bandwidth-Time product, equal to filter –3dB bandwidth (B) × symbol time (T).
C-BUS	CML's serial communication interface for peripheral devices.
CRC	Cyclic Redundancy Check, a type of error detecting code.
CSTDMA	Carrier Sense TDMA, the channel access scheme used by AIS Class B.
DAC	Digital-to-Analogue Converter
DC	Direct Current, often used to refer to the static or unchanging part of a signal.
DSC	Digital Selective Calling, an early maritime communication system.
FIFO	First In First Out, a queue-based data buffer where data is output in the same order as it arrives. Also known as an “elastic buffer”.
FSK	Frequency Shift Keying, a modulation scheme in which data symbols are represented by a shift in output signal frequency.
GALILEO	A satellite navigation system developed by the European Union.
GFSK	Gaussian Frequency Shift Keying, an FSK modulation scheme with a Gaussian filtered output signal.
GLONASS	Global Orbiting Navigation Satellite System, a satellite navigation system developed by the Commonwealth of Independent States (former Soviet republics).
GMSK	Gaussian Minimum Shift Keying, an FSK modulation scheme with a modulation index of 0.5 and a Gaussian filtered output signal.
GNSS	Global Navigation Satellite System, a generic term for the GPS, GLONASS and GALILEO satellite navigation systems.
GPS	Global Positioning System, a satellite navigation system developed by the U.S. Department of Defense.
HDLC	High-level Data Link Control, a synchronous data link protocol adopted by the ITU for use in AIS.
IC	Integrated Circuit.
IMO	International Maritime Organisation.
ITU	International Telecommunication Union, an organisation established to standardise and regulate international radio and and telecommunications.
µC	Microcontroller
NRZ	Non Return to Zero, a binary data coding scheme where 1s and 0s are represented by distinct signal levels.
NRZI	Non Return to Zero Inverted, a binary data coding scheme where 1s cause no change in a transmitted signal level, and 0s cause a change (this is NRZI “change on 0” as used in AIS, as opposed to NRZI “change on 1”).
PA	Power Amplifier.
PER	Packet Error Rate.
PRBS	Pseudo-Random Bit Sequence, an apparently random stream of bits typically generated by a linear-feedback shift register.

RAM	Random Access Memory.
RF	Radio Frequency.
RSSI	Received Signal Strength Indicator.
Rx	Receive or Receiver
SPI	Serial Peripheral Interface, a common inter-chip serial communications interface.
SOTDMA	Self Organised TDMA, the channel access scheme used by AIS Class A.
TCXO	Temperature Compensated Crystal Oscillator.
TDMA	Time Division Multiple Access, a technique for sharing access to a radio channel by dividing it into different time slots.
Tx	Transmit or Transmitter
UTC	Universal Time (Coordinated), the official measure of time in the world. Kept in synchronisation with the earth's rotation by the introduction of occasional leap seconds.
VHF	Very High Frequency, ITU band 8 (30 – 300MHz) that includes the maritime mobile band.
VQFN	Very thin profile Quad Flat No lead, an IC package type.

7. Performance Specification

7.1 Electrical Performance

7.1.1 Absolute Maximum Ratings

Exceeding these maximum ratings can result in damage to the device.

	Min.	Max.	Unit
Supply (IOV_{DD} - AV_{SS} or DV_{SS})	-0.3	4.5	V
Voltage on any digital pin to DV_{SS}	-0.3	IOV_{DD} + 0.3	V
Voltage on any analogue pin to AV_{SS}	-0.3	AV_{DD} + 0.3	
Current into or out of IOV_{DD}, AV_{SS} and DV_{SS} pins	-30	+30	mA
Current into or out of any other pin	-20	+20	mA

	Min.	Max.	Unit
Total Allowable Power Dissipation at Tamb = 25°C		500	mW
... Derating		5	mW/°C
Storage Temperature	-55	+125	°C

7.1.2 Operating Limits

Correct operation of the device outside these limits is not implied.

	Notes	Min.	Max.	Unit
Supply (IOV_{DD} – AV_{SS} or DV_{SS})		3.00	3.60	V
Operating Temperature		-40	+85	°C

7.1.3 Operating Characteristics

Details in this section represent design target values and are not currently guaranteed.

For the following conditions unless otherwise specified:
Min./Max. figures: IOV_{DD} = 3.0V to 3.6V, Tamb = -40C to +85°C.
Typ. figures: IOV_{DD} = 3.3V, Tamb = 25°C.
Load capacitance for digital outputs = 30pF.

DC Parameters	Notes	Min.	Typ.	Max.	Unit
I_{IOVDD} (powersaved)	1	-	20	100	µA
I_{IOVDD} (fully operational)	1	-	35	60	mA
AV_{DD} supply voltage		2.25	2.5	2.75	V
DV_{DD} supply voltage		2.25	2.5	2.75	V
VBIAS reference voltage			$AV_{DD}/2$		V
Current into VBIAS pin		-0.1	-	0.1	µA
Input logic '1' level		70%	-	-	IOV_{DD}
Input logic '0' level		-	-	30%	IOV_{DD}
Digital input leakage current (Vin = 0 to IOV_{DD})		-5.0	-	5.0	µA
Input/Output pin capacitance		-	-	15	pF
Output logic '1' level @ I_{OH} = -2mA	2	80%	-	-	IOV_{DD}
Output logic '0' level @ I_{OL} = 3mA	2	-	-	0.4	V
'Off' state leakage current (Vout = IOV_{DD})		-	-	10	µA

Clock and Timing Parameters		Notes	Min.	Typ.	Max.	Unit
REFCLK tolerance			-	-	±TBD	ppm
REFCLK mark:space ratio			35%	-	65%	
UTC1PPS rising edge tolerance (wrt UTC second)			-	-	±5	µs
UTC1PPS 'high' pulse width			100	-	-	ns
UTC1PPS 'low' pulse width			100	-	-	ns
SLOTCLKN pulse width			-	20.833	-	µs
CSXN to EXP[0-5]N propagation delay			0	-	25	ns
C-BUS Timings (Figure 14)						
t_{CSE}	CSN-Enable to Clock-High time		100	-	-	ns
t_{CSH}	Last Clock-High to CSN-High time		100	-	-	ns
t_{LOZ}	Clock-Low to Reply Output enable time		0.0	-	-	ns
t_{HIZ}	CSN-High to Reply Output 3-state time		-	-	1.0	µs
t_{CSOFF}	CSN-High time between transactions		1.0	-	-	µs
t_{NXT}	Inter-Byte time		200	-	-	ns
t_{CK}	Clock-Cycle time		200	-	-	ns
t_{CH}	Serial Clock-High time		100	-	-	ns
t_{CL}	Serial Clock-Low time		100	-	-	ns
t_{CDS}	Command Data Set-Up time		75	-	-	ns
t_{CDH}	Command Data Hold time		25	-	-	ns
t_{RDS}	Reply Data Set-Up time		50	-	-	ns
t_{RDH}	Reply Data Hold time		0	-	-	ns

Transmit Parameters	Notes	Min.	Typ.	Max.	Unit
AIS (GFSK 9600bps), 12.5kHz channel					
Bit rate accuracy		-	-	±50	ppm
BT		-	0.3	-	
Modulation index		-	0.25	-	
Storage time	3	-	TBD	-	symbol
AIS (GMSK 9600bps), 25kHz channel					
Bit rate accuracy		-	-	±50	ppm
BT		-	0.4	-	
Modulation index		-	0. 5	-	
Storage time	3	-	TBD	-	symbol
DSC (FSK 1200bps, 6dB/octave pre-emphasis)					
Bit rate accuracy		-	-	±50	ppm
Sub-carrier		-	1700	-	Hz
Tx mark frequency		-	1300	-	Hz
Tx space frequency		-	2100	-	Hz
Modulation index		-	2	-	
Storage time	3	-	TBD	-	symbol
Tx DAC					
Resolution		-	14	-	bits
Integral accuracy		-	-	±2	LSB
Differential accuracy		-	-	±2	LSB
SINAD	4	-65	-70	-	dB
Offset		-	-	±20	mV
Gain matching, I to Q		-	-	±0.25	dB
Phase matching, I to Q		-	-	±0.5	degree
I, Q output level	5	2.0	2.1	2.2	V

Receive Parameters	Notes	Min.	Typ.	Max.	Unit
AIS (GFSK 9600bps), 12.5kHz channel					
Bit rate accuracy		-	-	±50	ppm
Storage time	6	-	TBD	-	symbol
Packet error rate (PER) limit				20%	
PER with –18dB co-channel interference				20%	
PER with –98dBm static interference				20%	
PER due to intermodulation and blocking				TBD	
Adjacent channel rejection		-50			dB
Spurious response		TBD			dB
AIS (GMSK 9600bps), 25kHz channel					
Bit rate accuracy		-	-	±50	ppm
Storage time	6	-	TBD	-	symbol
Packet error rate (PER) limit				20%	
PER with –10dB co-channel interference				20%	
PER with –107dBm static interference				20%	
PER due to intermodulation and blocking				20%	
Adjacent channel rejection		-70			dB
Spurious response		-70			dB
DSC (FSK 1200bps, 6dB/octave de-emphasis)					
Bit rate accuracy		-	-	±TBD	ppm
Sub-carrier		-	1700	-	Hz
Tx mark frequency		1290	1300	1310	Hz
Tx space frequency		2090	2100	2110	Hz
Storage time	6	-	TBD	-	symbol
Bit error rate (BER) at a sensitivity of –107dBm				1%	
Rx ADC					
Resolution		-	16	-	bits
Signal to noise	4	80	85	-	dB
SINAD	4	75	80	-	dB
Input impedance at 100Hz		100	-	-	kΩ
Differential input voltage	8	-	1.7	1.9	V pk-pk

Auxiliary ADC	Notes	Min.	Typ.	Max.	Unit
Resolution		-	10	-	bits
Conversion time (per input)	9	11.021	-	117.35	µs
Integral non-linearity	11	-	-	±4	bits
Differential non-linearity	11	-	-	±3	bits
Zero error (offset)		-	-	±10	mV
Uncommitted op-amps:					
Gain (no load)		-	60	-	dB
Input offset		-	-	±5	mV
Common mode input voltage		0	-	AV_{DD}	
Output current		-	-	±125	µA
Output voltage		0.5	-	AV_{DD}-0.5	V
Capacitive load (including pin capacitance)		-	-	30	pF
Input noise voltage in 100Hz – 10kHz bandwidth		-	20	-	µV rms
Unity-gain bandwidth (no load)		-	2.5	-	MHz

Auxiliary DACs	Notes	Min.	Typ.	Max.	Unit
Resolution		-	10	-	bits
Settling time to 0.5 LSB	10			10	µs
Output resistance		-	-	250	Ω
Integral non-linearity		-	-	±4	bits
Differential non-linearity		-	-	±1	bits
Zero error (offset)		-	-	±10	mV
Resistive load		5	-	-	kΩ
Output noise voltage in 30kHz bandwidth		-	5	-	µV rms

Notes:
1. Not including any current drawn from the device pins by external circuitry.
2. Maximum total I_{OL} for all outputs is 50mA, maximum total I_{OH} for all outputs is 50mA.
3. Through G(M)FSK/FSK transmit filter, DAC, reconstruction filter and external RC filter.
4. Measured with a 2.25kHz test signal in a 9kHz bandwidth.
5. Peak to peak differential, assuming AV_{DD} = 2.5V. Level is proportional to AV_{DD}.
6. Through external RC filter, anti-alias filter, ADC, channel filter, and G(M)FSK/FSK receive filters.
7. Extrapolated from third harmonic distortion at maximum signal.
8. This means ±0.425V (typ.) on each input of the differential pair.
9. Programmable through the Special Command Interface.
10. Worst case large signal transition.
11. For signal levels between 0.5% and 99.5% of AV_{DD}.

7.1.3 Operating Characteristics (continued)

Timing Diagrams

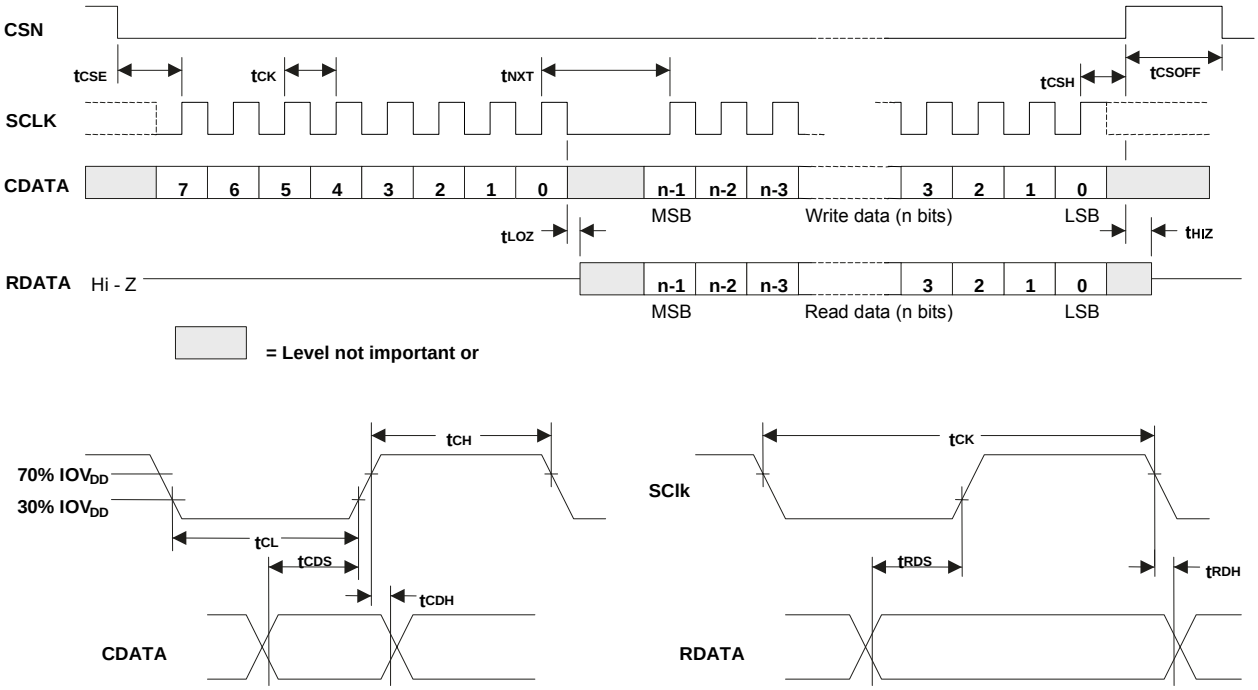
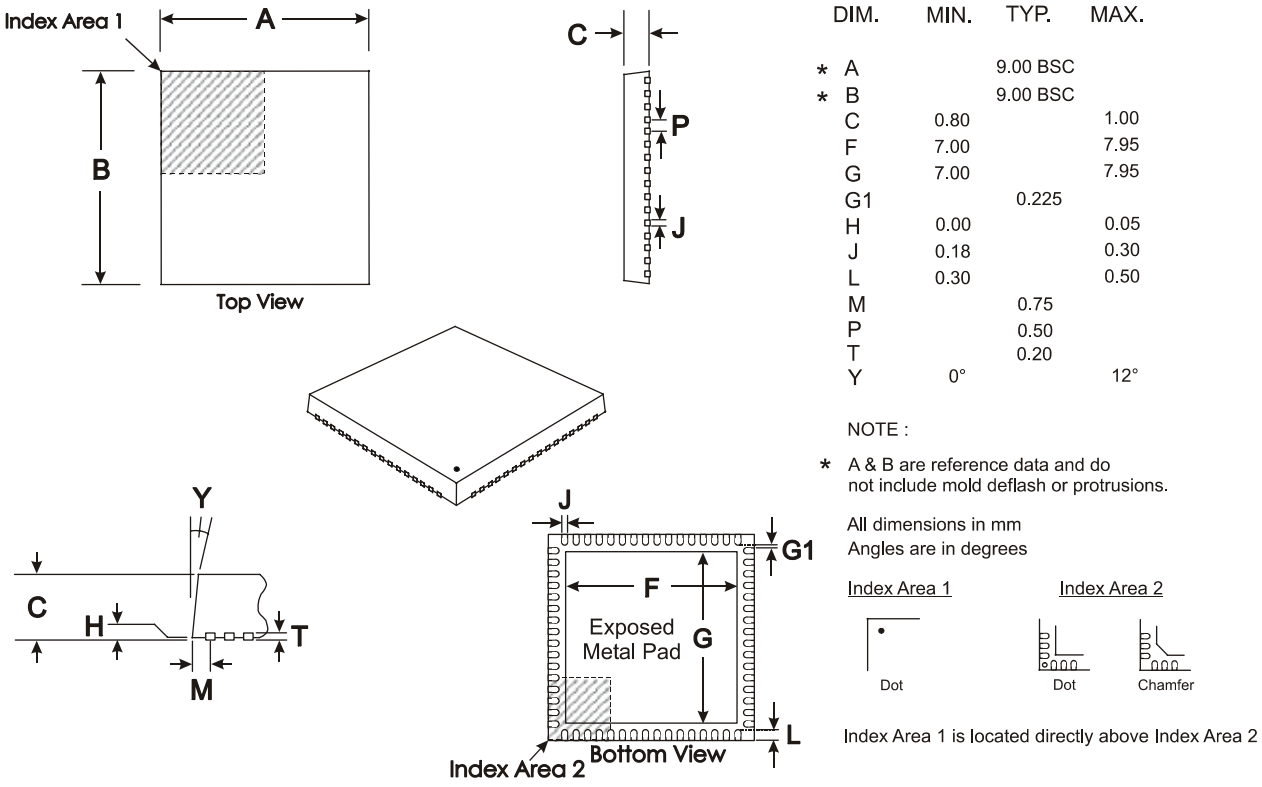


Figure 14 C-BUS Timing

7.2 Packaging



	DIM.	MIN.	TYP.	MAX.
*	A		9.00 BSC	
*	B		9.00 BSC	
	C	0.80		1.00
	F	7.00		7.95
	G	7.00		7.95
	G1		0.225	
	H	0.00		0.05
	J	0.18		0.30
	L	0.30		0.50
	M		0.75	
	P		0.50	
	T		0.20	
	Y	0°		12°

NOTE :

* A & B are reference data and do not include mold deflash or protrusions.

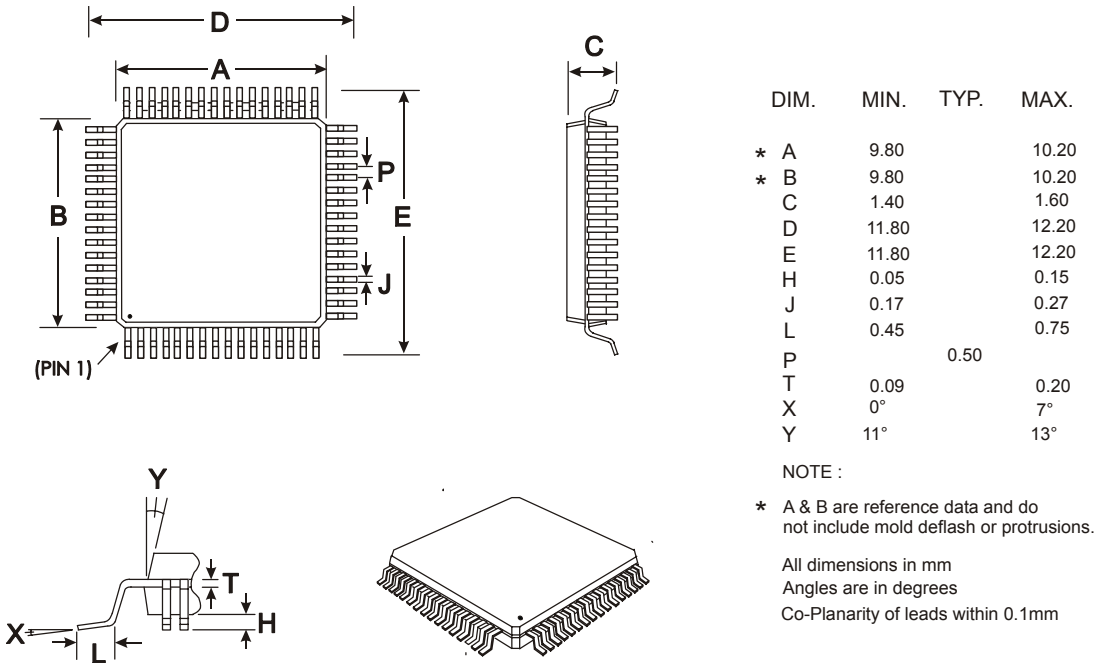
All dimensions in mm
Angles are in degrees

Index Area 1 is located directly above Index Area 2

Note:

The underside of the Q1 package is conductive and should be electrically connected to the digital ground. The circuit board should be designed so that no unwanted short circuits can occur.

Figure 15 Q1 Mechanical Outline: *Order as part no. CMX910Q1*



DIM.	MIN.	TYP.	MAX.
* A	9.80		10.20
* B	9.80		10.20
C	1.40		1.60
D	11.80		12.20
E	11.80		12.20
H	0.05		0.15
J	0.17		0.27
L	0.45		0.75
P		0.50	
T	0.09		0.20
X	0°		7°
Y	11°		13°

NOTE :

* A & B are reference data and do not include mold deflash or protrusions.

All dimensions in mm
Angles are in degrees
Co-Planarity of leads within 0.1mm

Figure 16 L9 Mechanical Outline: *Order as part no. CMX910L9*

Handling precautions: This product includes input protection, however, precautions should be taken to prevent device damage from electro-static discharge. CML does not assume any responsibility for the use of any circuitry described. No IPR or circuit patent licences are implied. CML reserves the right at any time without notice to change the said circuitry and this product specification. CML has a policy of testing every product shipped using calibrated test equipment to ensure compliance with this product specification. Specific testing of all circuit parameters is not necessarily performed.

www.cmlmicro.com

For FAQs see: www.cmlmicro.com/products/faqs/

For a full data sheet listing see: www.cmlmicro.com/products/datasheets/download.htm

For detailed application notes: www.cmlmicro.com/products/applications/

CML Microcircuits (UK) Ltd COMMUNICATION SEMICONDUCTORS	CML Microcircuits (USA) Inc. COMMUNICATION SEMICONDUCTORS	CML Microcircuits (Singapore) Pte Ltd COMMUNICATION SEMICONDUCTORS	
Oval Park, Langford, Maldon, Essex, CM9 6WG - England.	4800 Bethania Station Road, Winston-Salem, NC 27105 - USA.	No 2 Kallang Pudding Road, #09 - 05/06 Mactech Industrial Building, Singapore 349307	No. 218, Tian Mu Road West, Tower 1, Unit 1008, Shanghai Kerry Everbright City, Zhabei, Shanghai 200070, China.
Tel: +44 (0)1621 875500 Fax: +44 (0)1621 875600	Tel: +1 336 744 5050, 800 638 5577 Fax: +1 336 744 5054	Tel: +65 6745 0426 Fax: +65 6745 2917	Tel: +86 21 6317 4107 +86 21 6317 8916 Fax: +86 21 6317 0243
Sales: sales@cmlmicro.com	Sales: us.sales@cmlmicro.com	Sales: sg.sales@cmlmicro.com	Sales: cn.sales@cmlmicro.com.cn
Technical Support: techsupport@cmlmicro.com	Technical Support: us.techsupport@cmlmicro.com	Technical Support: sg.techsupport@cmlmicro.com	Technical Support: sg.techsupport@cmlmicro.com