SIEMENS

Data Communication ICs

Enhanced Serial Communication Controller
ESCC2, ESCC8

SAB 82532
SAB 82538

Application Note 10.94

ESCC2, ESCC8 Revision History:	Original Version: 10.94
Previous Releases:	
Page	Subjects (changes since last revision)

Data Classification

Maximum Ratings

Maximum ratings are absolute ratings; exceeding only one of these values may cause irreversible damage to the integrated circuit.

Characteristics

The listed characteristics are ensured over the operating range of the integrated circuit. Typical characteristics specify mean values expected over the production spread. If not otherwise specified, typical characteristics apply at T_A = 25 ˚C and the given supply voltage.

Operating Range

In the operating range the functions given in the circuit description are fulfilled.

For detailed technical information about **"Processing Guidelines"** and **"Quality Assurance"** for ICs, see our **"Product Overview"**.

Edition 10.94
This edition was realized using the software system FrameMaker®

Published by Siemens AG, Bereich Halbleiter, Marketing-Kommunikation, Balanstraße 73, D-81541 München.

Packing

Please use the recycling operators known to you. We can also help you - get in touch with your nearest sales office. By agreement we will take packing material back, if it is sorted. You must bear the costs of transport.

For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

1 Clock Modes

1.1 Summary

The ESCC clock modes offer a high amount of flexibility for generating and/or recovering bit clocks for synchronous and asynchronous serial data transfers.

Chapter 1 describes and illustrates the signal flow between the different internal functional units for each clock mode separately.

Wolfram Kiesecker
Karl Singendonk
December 1993

1.2 Clock Generation and Recovery

The ESCC's clock sources for bit-clock generation are:

1.2.1 Internal Clock Sources

ESCC Oscillator (OSC)
There is one internal oscillator (OSC) per device. The OSC may be used for one or several serial channels. The oscillator's frequency is determined by an external crystal connected to the XTAL1/XTAL2 pin.

Baud Rate Generator (BRG)
The Baud Rate Generator (BRG) is a programmable clock-divider for variable bit-clock rate generation. There is one BRG per serial channel. The BRG's input clock is supplied by the OSC or by the RxCLK pin.

Digital Phase Lock Loop Circuitry (DPLL)
The DPLL performs bit-clock recovery and phase adjustment from the incoming data-stream. The DPLL is supplied with a reference clock from the BRG, which has to be 16 times faster than the nominal data clock rate. There is one DPLL per serial channel.

1.2.2 External Clock Sources

RxCLK Pins
The RxCLK pins may be used to supply the bit-clock either directly or in conjunction with the DPLL and/or BRG. There is one RxCLK pin per serial channel.

TxCLK Pins
The TxCLK pins may be used to supply the transmit clock directly (no phase adjustment or clock scaling). There is one TxCLK pin per serial channel.

1.3 Clock Modes

The Clock Mode determines how the ESCC's Receive and Transmit Clocks are generated. Thirteen different clock modes are programmable through CCR1.CM0…2 and CCR2.SSEL. Each channel of the ESCC may be programmed independently.

Note: "Receive Clock" and "Transmit Clock" are names for internal signals. They are not identical with the signal on the RxCLK or TxCLK pin!

1.3.1 Basic Clock Modes

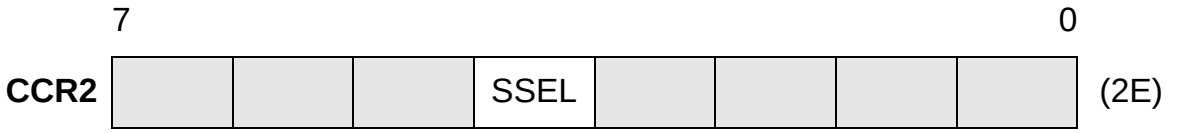
The eight basic clock modes 0-7 are programmable in the ESCC's CCR1 register.

	7							0	
CCR1*						CM2	CM1	CM0	(2D)
(R/W)	**Clock Mode CM2 … CM0**								
						0	0	0	Clock Mode 0
						0	0	1	Clock Mode 1
						0	1	0	Clock Mode 2
						0	1	1	Clock Mode 3
						1	0	0	Clock Mode 4
						1	0	1	Clock Mode 5
						1	1	0	Clock Mode 6
						1	1	1	Clock Mode 7

* Throughout this document: **The unshaded bit-fields are the most relevant for the examined function.**

1.3.2 Clock Mode Extension

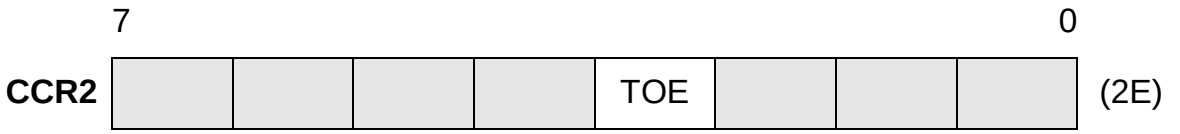
The functionality of clock modes 0, 2, 3, 6, and 7 further depends on the clock mode extension (a or b), which is selected with CCR2.SSEL:



(R/W) **Clk Source Select SSEL**
0 clock mode extension a
1 clock mode extension b

1.3.3 Transmit Clock Output Signal

For clock modes that do not use the TxCLK as a clock source, this pin may be programmed as an output for the internal transmitter clock. Setting CCR2.TOE enables the output function:



(R/W) **TxCLK Output Enable TOE**
0 TxCLK is input
1 TxCLK output (only clock modes 0b, 2b, 3a, 3b, 4, 6b, 7a, 7b)
Special function in clock mode 5

1.3.4 Clock Mode Overview

The following table summarizes all clock modes:

Table 1
ESCC Clock Modes

Channel Configuration		**Clock Mode**	**Clock Sources**				**TxCLK Output**
CCR1. CM2…0	**CCR2. SSEL**		**BRG**	**DPLL**	**REC**	**TRM**	**if CCR2. TOE = 1**
000	0	0a	–	–	RxCLK	TxCLK	–
000	1	0b	OSC	–	RxCLK	BRG	BRG
001	x	1	–	–	RxCLK	RxCLK	–
010	0	2a	RxCLK	BRG	DPLL	TxCLK	–

Table 1
ESCC Clock Modes (cont'd)

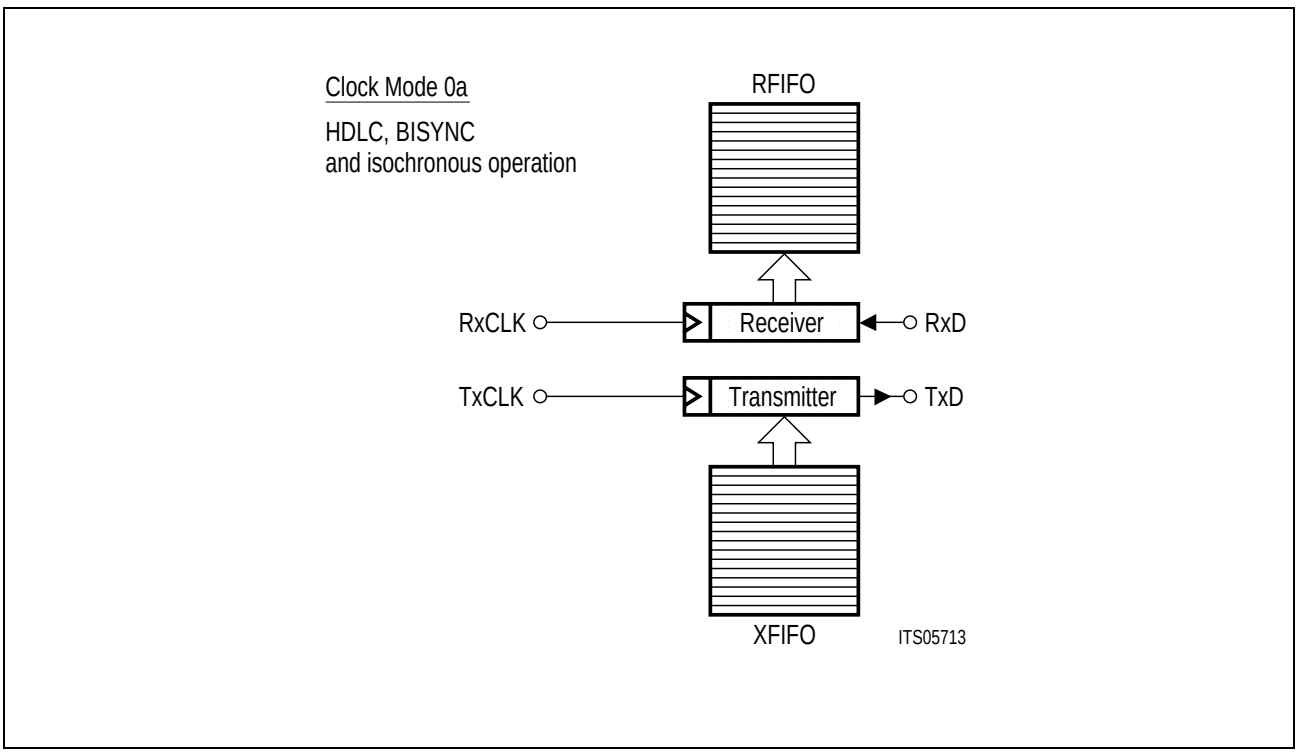
Channel Configuration		Clock Mode	Clock Sources				TxCLK Output
CCR1. CM2…0	**CCR2. SSEL**		**BRG**	**DPLL**	**REC**	**TRM**	**if CCR2. TOE = 1**
010	1	2b	RxCLK	BRG	DPLL	BRG/16	BRG/16
011	0	3a	RxCLK	BRG	DPLL	DPLL	DPLL
011	1	3b	RxCLK	–	BRG	BRG	BRG
100	x	4	–	–	OSC	OSC	OSC
101	x	5	–	–	RxCLK	RxCLK	TS-Ctrl
110	0	6a	OSC	BRG	DPLL	TxCLK	–
110	1	6b	OSC	BRG	DPLL	BRG/16	BRG/16
111	0	7a	OSC	BRG	DPLL	DPLL	DPLL
111	1	7b	OSC	–	BRG	BRG	BRG

Note: In ASYNC mode the baud rate depends on the value of CCR1.BCR. If BCR is set, receive data over-sampling with majority decision is performed. BCR is only valid in clock modes 0a, 0b, 1, 3b, 4, and 7b (without DPLL). It has no effect in clock modes 2a, 2b, 3a, 6a, 6b, and 7a). CCR1.BCR is defined in ASYNC mode only!

1.3.5 Clock Modes in Detail

1.3.5.1 Clock Mode 0a (External Clocks)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



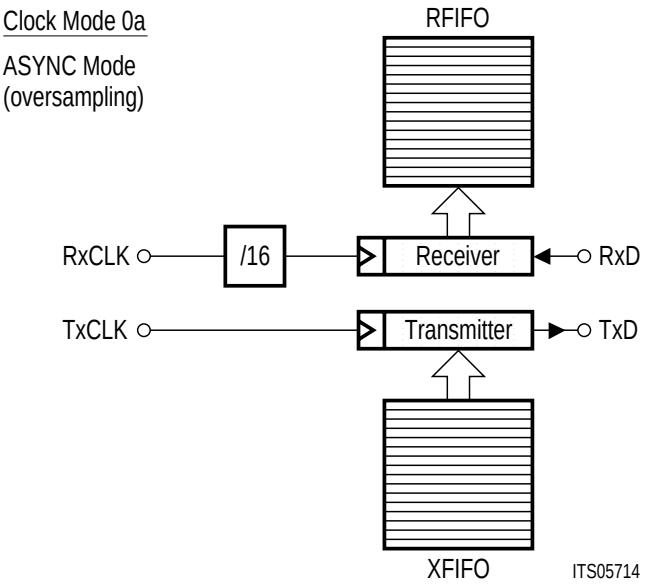
CM2…CM0	SSEL	BCR*
0 0 0	0	0

* ASYNC only!

Separate, externally generated receive and transmit clocks are supplied to the ESCC via their RxCLK and TxCLK pins.

Only in ASYNC Mode
If CCR1.BCR = 0, isochronous operation is selected and RxCLK is identical with the bit clock rate (no over-sampling).

ASYNC Mode (Standard Asynchronous Operation)



CM2…CM0	SSEL	BCR*
0 0 0	0	1

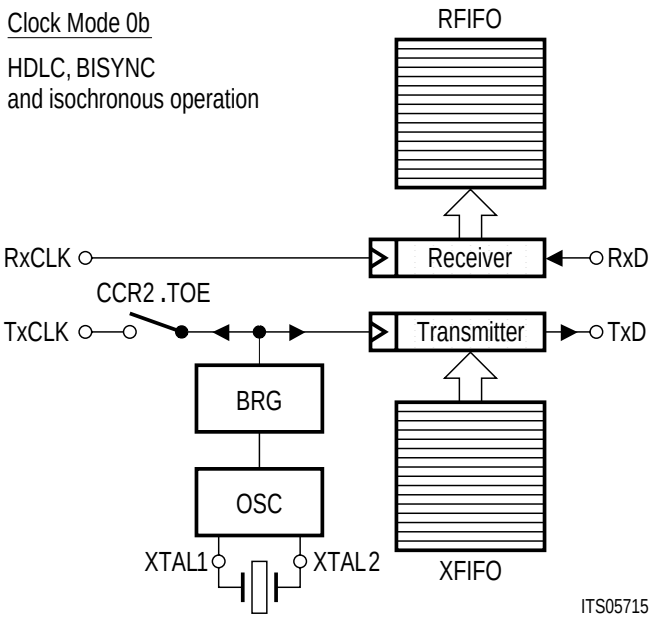
* ASYNC only!

Only in ASYNC Mode

If CCR1.BCR is set, standard asynchronous operation is selected with 3 times over-sampling. RxCLK delivers the sample clock, and has to be 16 times faster than the bit clock rate.

1.3.5.2 Clock Mode 0b (External/Internal Clocks)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



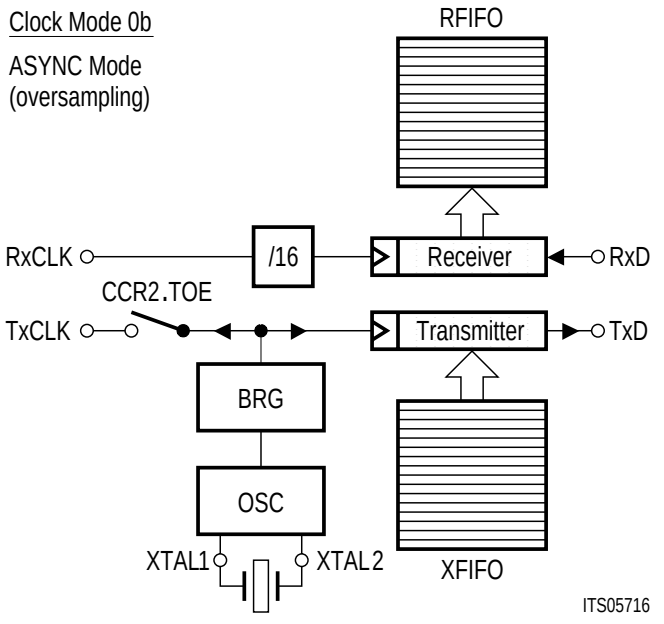
CM2…CM0	SSEL	BCR*
0 0 0	1	0

* ASYNC only!

The ESCC's receive clock is directly supplied via the RxCLK pin. The transmit clock is generated internally by the baud rate generator and the clock signal on XTAL1. The transmit clock is output at pin TxCLK if CCR2.TOE is set.

Only in ASYNC Mode
If CCR1.BCR = 0, isochronous operation is selected and RxCLK is identical with the bit clock rate (no over-sampling).

ASYNC Mode (Standard Asynchronous Operation)



CM2…CM0	SSEL	BCR*
0 0 0	1	1

* ASYNC only!

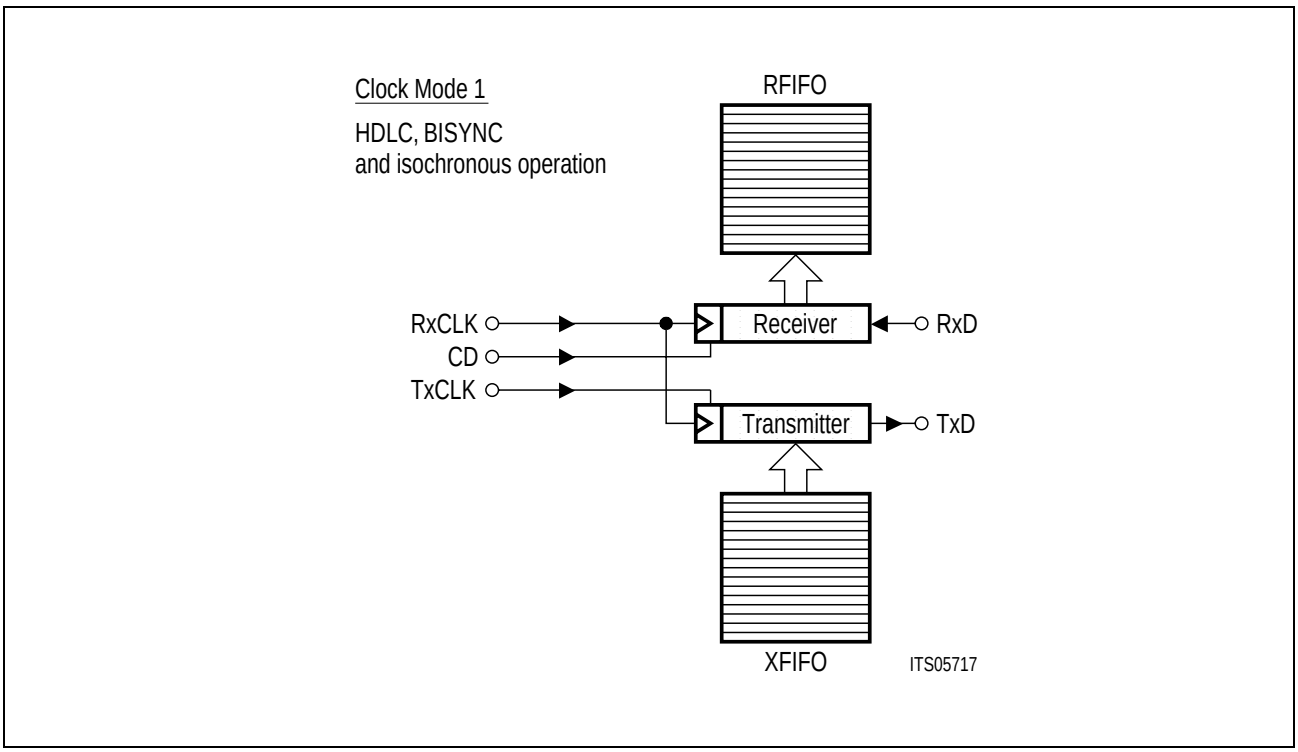
Only in ASYNC Mode

The RxCLK pin supplies the externally generated sample clock for the receiver. The sample clock must be 16 times faster than the baud rate of the incoming data stream (due to over-sampling).

The transmit clock is generated internally by the baud rate generator and the oscillator. The transmit clock is output via pin TxCLK if CCR2.TOE is set.

1.3.5.3 Clock Mode 1 (Receiver and Transmitter Strobes)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



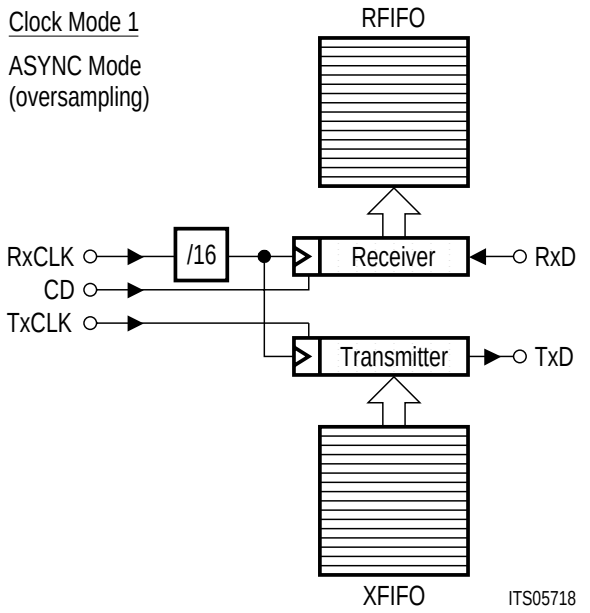
CM2…CM0	SSEL	BCR*
0 0 1	x	0

* ASYNC only!

RxCLK supplies an externally generated clock signal for the receiver and the transmitter. In addition, a receive strobe can be connected via CD and a transmit strobe via TxCLK. These strobe signals work on a per bit basis. This operating mode can be applied in time division multiplex applications or for adjusting disparate transmit and receive data rates.

Note: In Extended Transparent Mode (HDLC/SDLC), the above strobe signals provide byte synchronization (byte alignment).

ASYNC Mode (Standard Asynchronous Operation)



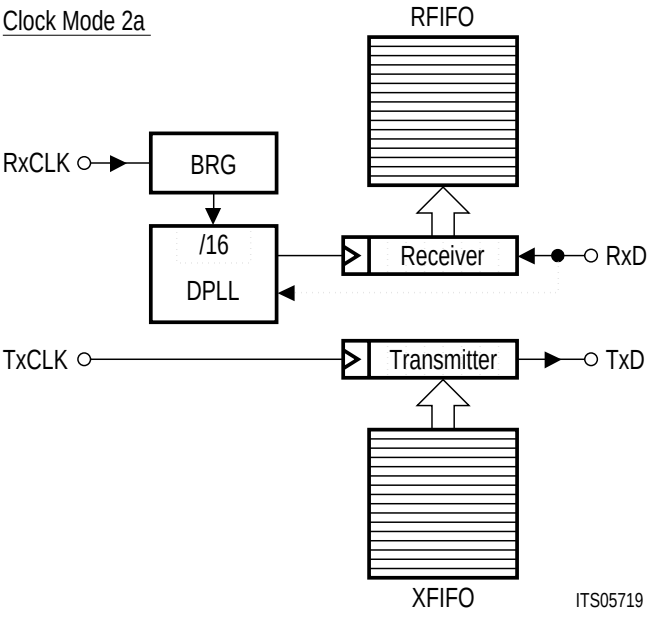
CM2…CM0	SSEL	BCR*
0 0 1	x	1

* ASYNC only!

Only in ASYNC Mode

RxCLK supplies an externally generated clock signal for the receiver and the transmitter. This clock has to be 16 times faster than the bit clock (for receive data over-sampling). The CD and the TxCLK pins supply strobe signals, which work on a per bit basis.

1.3.5.4 Clock Mode 2a (Rec. Clock from DPLL, Trm. Clock from TxCLK)



CM2…CM0	SSEL	BCR*
0 1 0	0	x

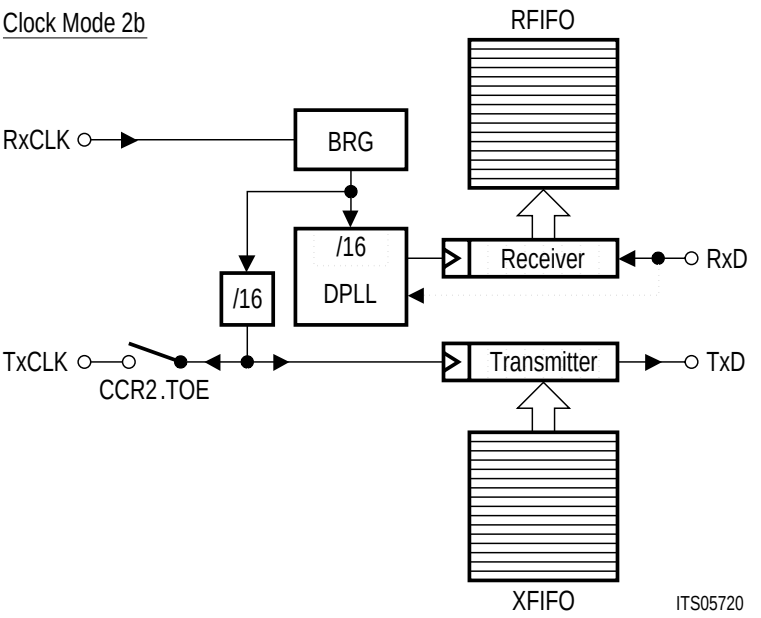
* ASYNC only!

RxCLK supplies an external clock for the BRG. The BRG delivers a reference clock for the DPLL.

The DPLL generates the bit-clock for the receiver. The frequency of the DPLL's reference clock must be 16 times faster than its nominal bit rate.

TxCLK supplies the transmitter's bit-clock directly.

1.3.5.5 Clock Mode 2b (Rec. Clock from DPLL, Trm. Clock from BRG/16)



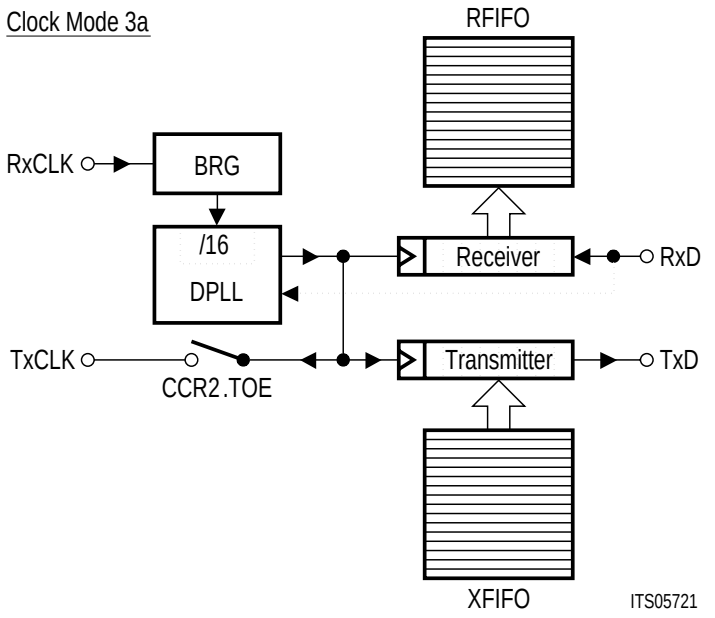
CM2…CM0	SSEL	BCR*
0 1 0	1	x

* ASYNC only!

RxCLK supplies an external clock for the BRG. The receive clock is generated by the DPLL (same as in clock mode 2a).

The bit-clock for the transmitter is the BRG's output signal divided by 16. The transmit clock is output via the TxCLK pin if CCR2.TOE is set.

1.3.5.6 Clock Mode 3a (Rec. and Trm. Clock from RxCLK, BRG and DPLL)



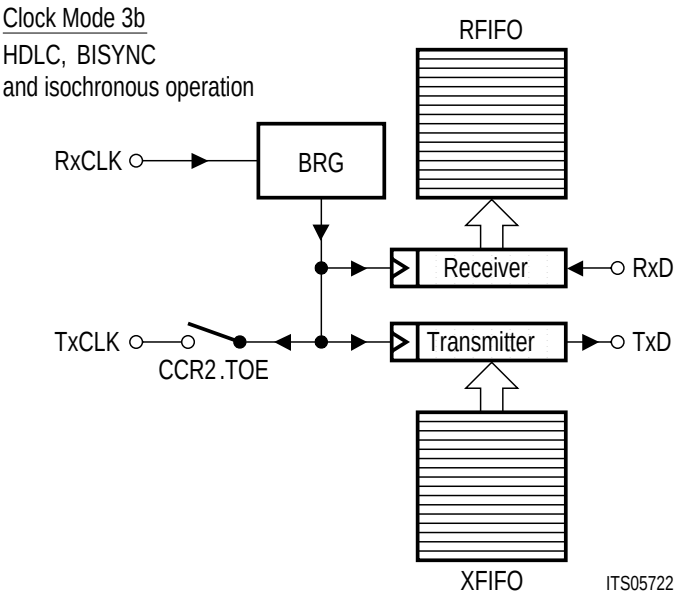
CM2…CM0	SSEL	BCR*
0 1 1	0	x

* ASYNC only!

The BRG receives its input clock from the RxCLK pin. The BRG supplies the reference clock for the DPLL, which must be 16 times faster than the nominal bit rate. The DPLL generates both the receive and the transmit clock. This clock is output via TxCLK if CCR2.TOE is set.

1.3.5.7 Clock Mode 3b (Rec. and Trm. Clock from RxCLK and BRG)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



CM2…CM0	SSEL	BCR*
0 1 1	1	0

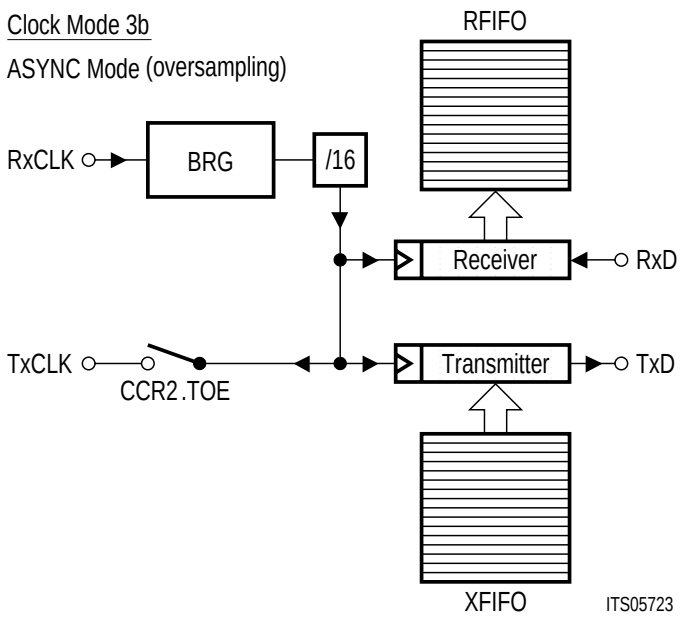
* ASYNC only!

The BRG receives its input clock from the RxCLK pin. The BRG supplies the receive and transmit clocks directly. This clock is output at pin TxCLK, if CCR2.TOE is set.

Only in ASYNC Mode
If CCR1.BCR = 0, isochronous operation is selected and the BRG supplies the receiver's bit clock directly (no over-sampling).

Clock Mode 3b (cont'd)

ASYNC Mode (Standard Asynchronous Operation)



CM2…CM0	SSEL	BCR*
0 1 1	1	1

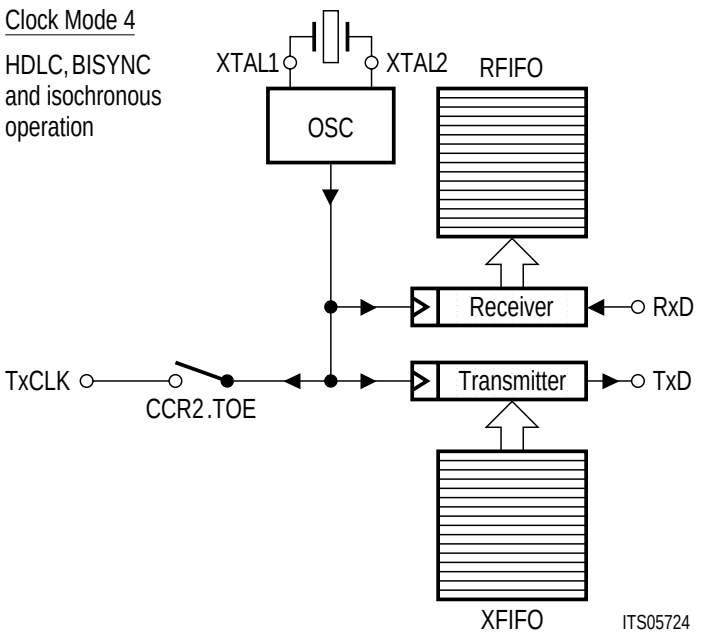
* ASYNC only!

Only in ASYNC Mode

If CCR1.BCR is set, standard asynchronous operation is selected with 3-times over-sampling. RxCLK supplies an external clock for the BRG. The BRG generates the receiver's sample clock (16 times faster than the baud rate). An internal divider (/16) generates the bit clock rate for the receiver and the transmitter. The bit clock is output at the TxCLK pin, if CCR2.TOE is set.

1.3.5.8 Clock Mode 4 (OSC-Direct)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



CM2…CM0	SSEL	BCR*
1 0 0	x	0

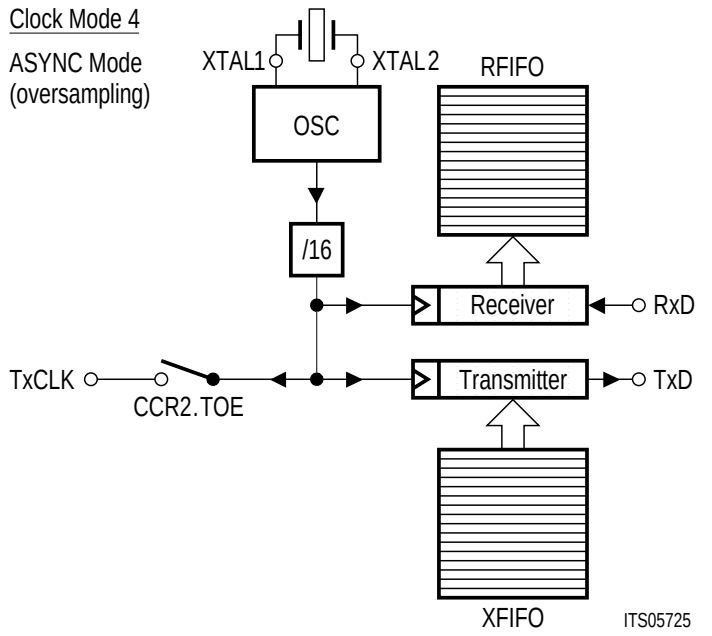
* ASYNC only!

The receive and the transmit clocks are directly supplied by the OSC. The OSC clock is output at pin TxCLK, if CCR2.TOE is set.

Only in ASYNC Mode
If CCR1.BCR = 0, isochronous operation is selected and the OSC supplies the receiver's bit clock directly (no over-sampling).

Clock Mode 4 (cont'd)

ASYNC Mode (Standard Asynchronous Operation)



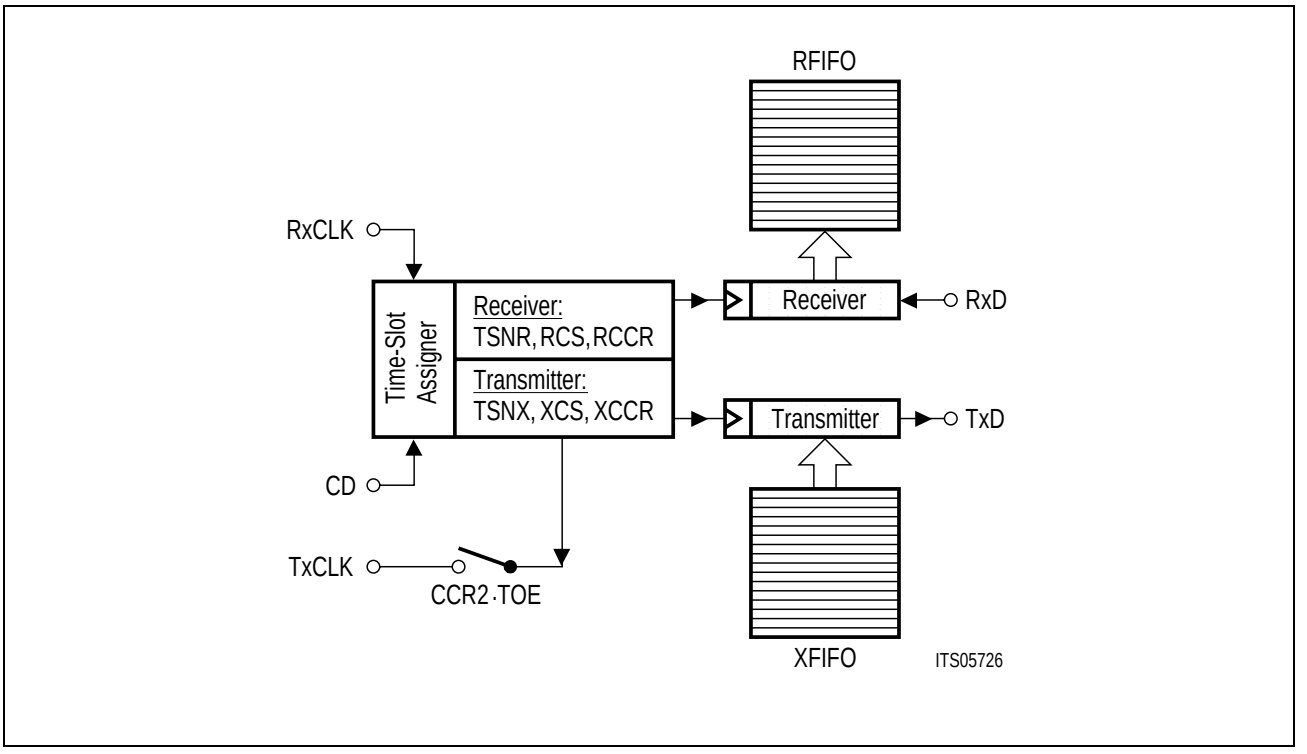
CM2…CM0	SSEL	BCR*
1 0 0	x	1

* ASYNC only!

Only in ASYNC Mode

If CCR1.BCR is set, standard asynchronous operation is selected with 3-times over-sampling. The OSC generates the receiver's sample clock (has to be 16 times faster than the baud rate). An internal divider (/16) generates the bit clock rate for the receiver and the transmitter. The bit clock is output at the TxCLK pin, if CCR2.TOE is set.

1.3.5.9 Clock Mode 5 (Time-Slots)



CM2…CM0	SSEL	BCR*
1 0 1	x	x

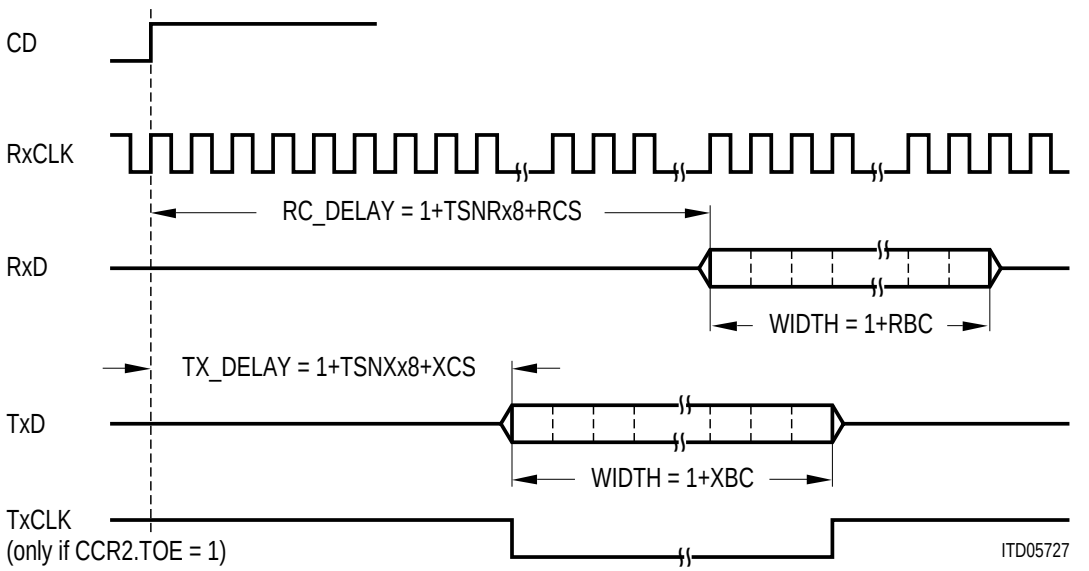
* ASYNC only!

This clock mode has been designed for application in time-slot oriented PCM systems.

The receive and transmit clocks are identical for each channel and must be supplied externally via the RxCLK pin. The ESCC receives and transmits only during a selected time-slot. The width and the position of the time-slot is programmable. The transmit time-slot is indicated with an active-low signal at the TxCLK pin, if CCR2.TOE is set.

Clock Mode 5 (cont'd)

Time-Slot Position and Width



Clock Mode 5 (cont'd)

How to Program the Time-Slot Position

The location of a time-slot is determined relative to the frame synchronization signal at the CD pin. The time-slot position for the transmitter and for the receiver is programmed independently.

The beginning of the transmit time-slot is calculated by the formula:

$$TX\_DELAY = 1 + TSNX \times 8 + XCS$$

TSNX and XCS are bit fields in the TSAX and CCR2 registers:

	7							0	
TSAX	TSNX5						TSNX0	XCS2	XCS1	(30)
(W)	**TSNX5…0:** transmit time-slot number (0x00-0x3F)									

	7							0	
TSAX	TSNX5						TSNX0	XCS2	XCS1	(30)
(W)										

	7							0	
CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV	(2E)
(R/W)									

TSAX. XCS2	TSAX. XCS1	CCR2. XCS0	XCS2…0: transmit clock shift (0-7)

Clock Mode 5 (cont'd)

The beginning of the receive time-slot is calculated by the formula:

RC_DELAY = 1 + TSNR x 8 + RCS

TSNR and RCS are bit fields in the TSAR and CCR2 registers:

	7							0	
TSAR	TSNR5						TSNR0	RCS2	RCS1	(31)
(W)	**TSNR5…0:** transmit time-slot number (0x00-0x3F)									

	7							0	
TSAR	TSNR5						TSNR0	RCS2	RCS1	(30)
(W)										

	7							0	
CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV	(2E)
(R/W)									

TSAR. RCS2	TSAR. RCS1	CCR2. RCS0	RCS2…0: receive clock shift (0-7)

How to Program the Time-Slot Width

The number of bits per time-slot (width) is independently programmed in the transmit and receive channel capacity registers (XCCR, RCCR):

	7		0	
XCCR	XBC7		XBC0	(32)

(W) **XBC7…0:** Transmit bit count: Number of bits in time-slot = *1 + XBC (1…256)*

	7		0	
RCCR	RBC7		RBC0	(33)

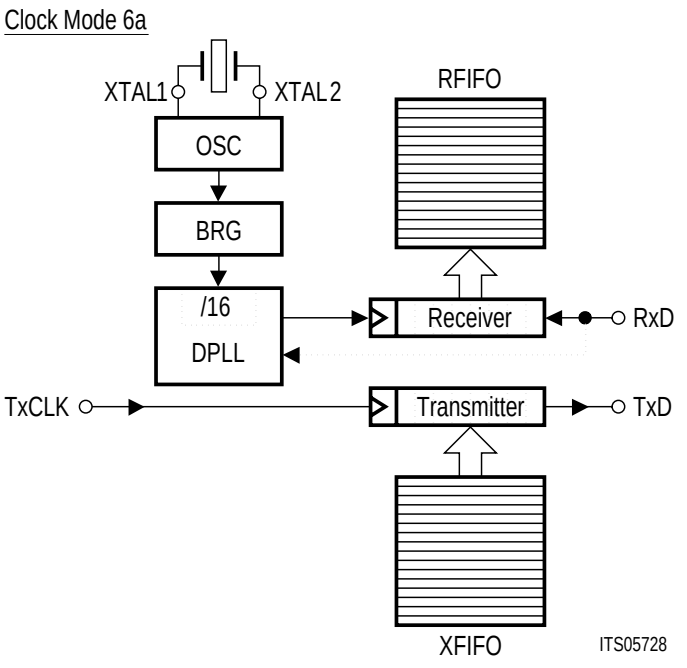
(W) **RBC7…0:** Receive bit count: Number of bits in time-slot = *1 + RBC (1…256)*

Note: Clock Mode 5 is only specified for the 8053x N-10 version.

For correct operation, clock mode 5 should only be used in conjunction with NRZ coding.

In HDLC/SDLC Extended Transparent Mode the time-slot windows provide character synchronization (byte alignment) even if set to one bit length. In all other modes the window size can range from one bit to 256 bits.

1.3.5.10 Clock Mode 6a (Rec. Clock from DPLL, Trm. Clock from TxCLK)



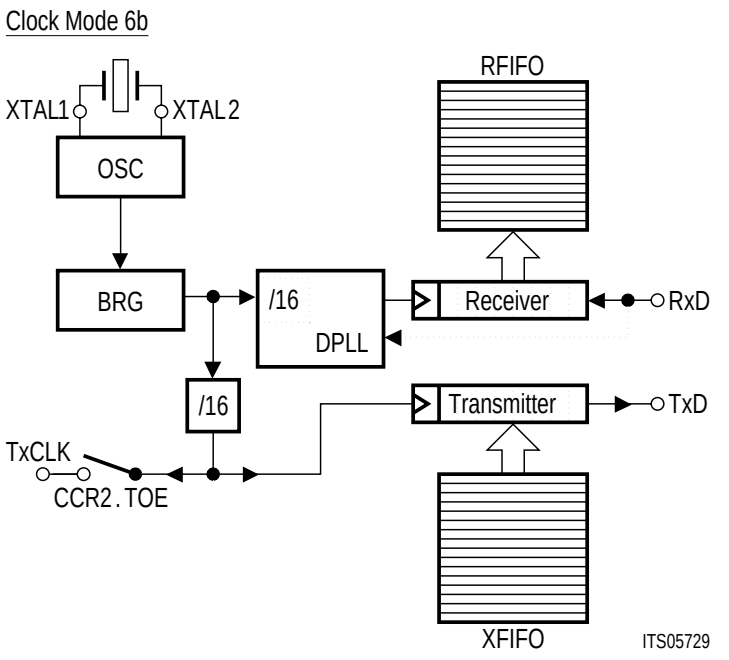
CM2…CM0	SSEL	BCR*
1 1 0	0	x

* ASYNC only!

The internal OSC supplies the input clock for the BRG. The BRG delivers a reference clock for the DPLL. The DPLL generates the bit-clock for the receiver. The frequency of the DPLL's reference clock must be 16 times faster than its nominal bit rate.

TxCLK supplies the transmitter's bit-clock directly.

1.3.5.11 Clock Mode 6b (Rec. Clock from DPLL, Trm. Clock from BRG/16)



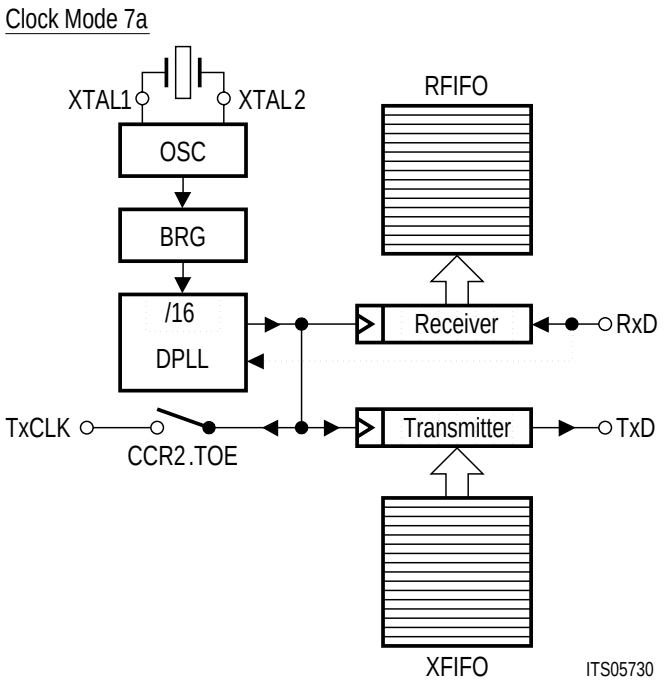
CM2…CM0	SSEL	BCR*
1 1 0	1	x

* ASYNC only!

The internal OSC supplies an input clock for the BRG. The receive clock is generated by the DPLL (same as in clock mode 6a).

The bit-clock for the transmitter is the BRG's output signal divided by 16. If CCR2.TOE is set, the transmit clock is output via the TxCLK pin.

1.3.5.12 Clock Mode 7a (Rec. and Trm. Clock from OSC, BRG and DPLL)



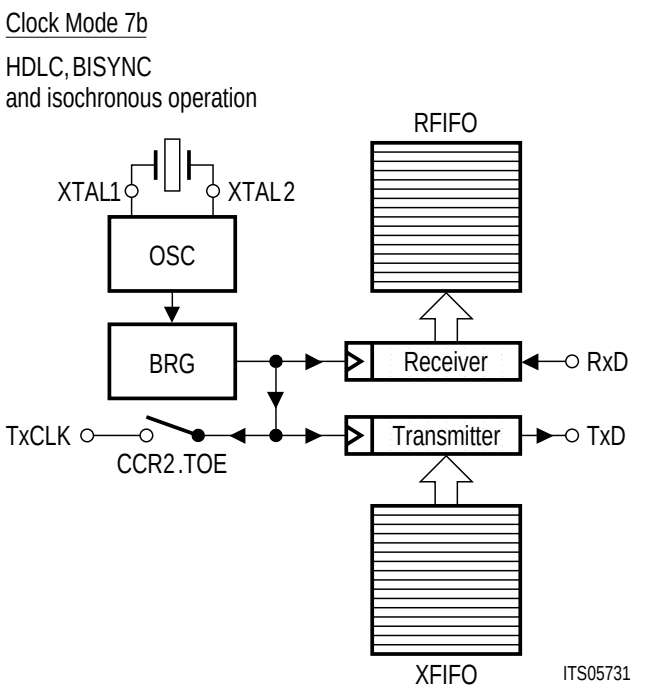
CM2…CM0	SSEL	BCR*
1 1 1	0	x

* ASYNC only!

The BRG receives its input clock from the internal OSC. The BRG supplies the reference clock for the DPLL, which must be 16 times faster than the nominal bit rate. The DPLL generates both the receive and the transmit clock. If CCR2.TOE is set, this clock is output at the TxCLK pin.

1.3.5.13 Clock Mode 7b (Rec. and Trm. Clock from OSC and BRG)

HDLC, BISYNC Mode, Isochronous Operation in ASYNC Mode



CM2…CM0	SSEL	BCR*
1 1 1	1	0

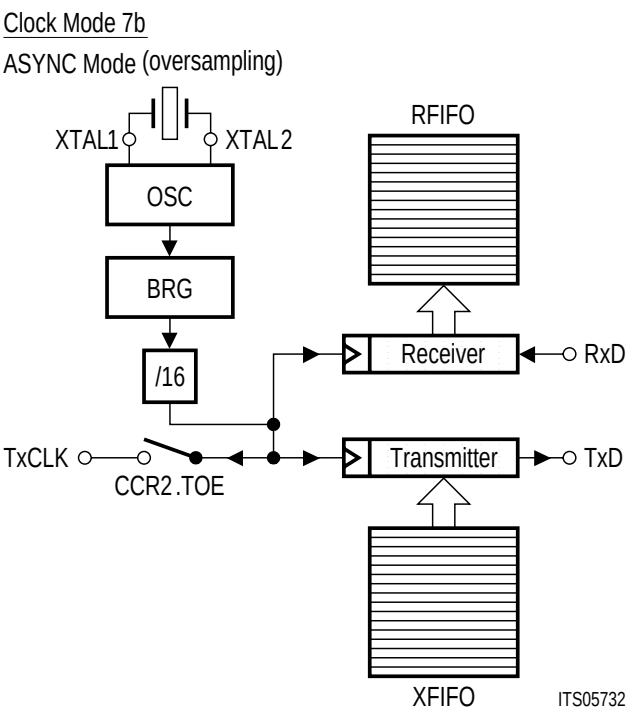
* ASYNC only!

The BRG receives its input clock from the internal OSC. The BRG supplies the receive and transmit clocks directly. This clock is output at pin TxCLK, if CCR2.TOE is set.

Only in ASYNC Mode
If CCR1.BCR = 0, isochronous operation is selected and the BRG supplies the receiver's bit clock directly (no over-sampling).

Clock Mode 7b (cont'd)

ASYNC Mode (Standard Asynchronous Operation)



CM2…CM0	SSEL	BCR*
1 1 1	1	1

* ASYNC only!

Only in ASYNC Mode

If CCR1.BCR is set, standard asynchronous operation is selected with 3-times over-sampling. The internal OSC supplies an input clock for the BRG. The BRG generates the receiver's sample clock (16 times faster than the baud rate). An internal divider (/16) generates the bit clock rate for the receiver and the transmitter. The bit clock is output at the TxCLK pin, if CCR2.TOE is set.

1.4 ESCC System Clock

The ESCC derives its system clock from the transmit clock or from the internal oscillator. The system clock source is determined by CCR0.MCE.

	7						0	
CCR0	PU	MCE	0	SC2	SC0	SM1	SM0	(2C)

(R/W)

Master Clock Enable

0 System clock derived from transmit clock

1 System clock derived from XTAL1

With the master clock feature enabled, the functionality of the microprocessor interface (access to all status and control registers and FIFOs, DMA and interrupt support) is independent from the receive and transmit clocks. The master clock is supplied via pin XTAL1 (or a crystal connected to XTAL1-XTAL2).

The maximum master clock frequency (XTAL1 frequency) depends on the version:

SAB/F 82532 N-10	10 MHz
SAB/F 82538 N-10	10 MHz
SAB 82532 N	2 MHz
SAB 82538 N	2 MHz

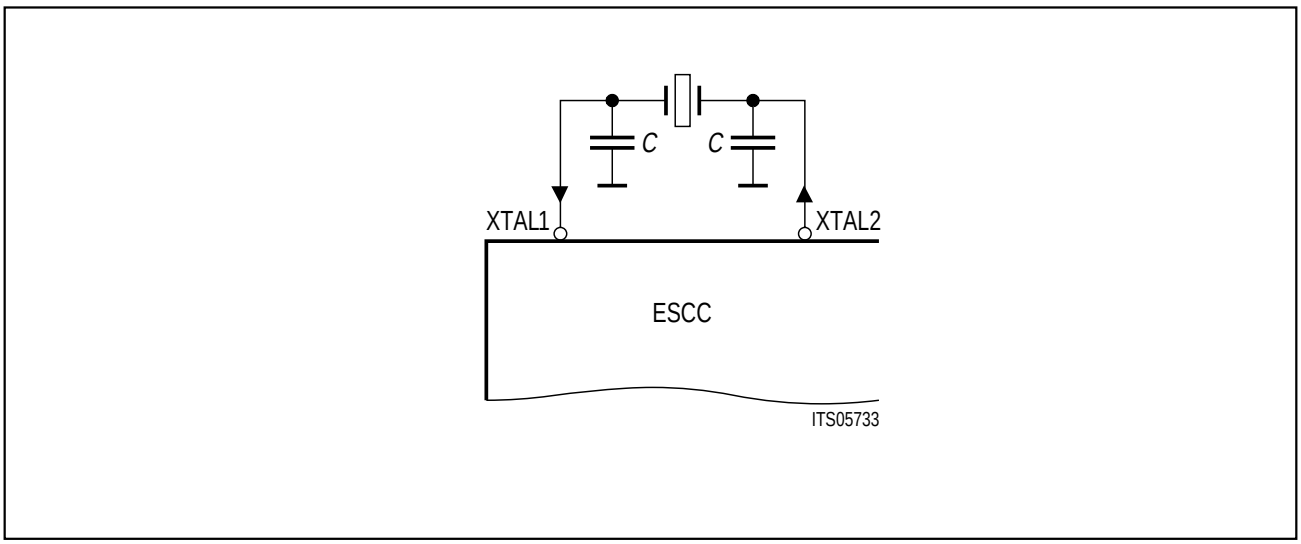
Master Clock Mode Restrictions

- Not applicable in clock mode 5.
- Not applicable in SDLC Loop mode.
- If bus configuration is selected in HDLC/SDLC mode the One-Insertion cannot be used in conjunction with the Master Clock feature.
- fm/fx > 2.5, fm/fr > 3.
- The internal timers run with the master clock.

1.5 Applications

1.5.1 The Crystal Oscillator (OSC)

If one of the clock modes 0b, 4, 6, 7 or the master clock is selected, the internal oscillator (OSC) is enabled. An external crystal can be connected to pins XTAL1-XTAL2. The output signal of the OSC can be used for one or for multiple serial channels (one OSC per device, one baud rate generator and DPLL per serial channel!).



Capacity Values for *C*

f_{XTAL1}	C
< 10 MHz	33 pF
> 10 MHz	22 pF

1.5.2 Crystal Characterization for the ESCC

Mode of Oscillation	Parallel Resonance
Frequency calibration tolerance	50 ppm
Frequency shift during lifetime	10 ppm
Temperature coefficient / frequency shift	50 ppm within the temperature range
Motional capacitance	15 $f_F \pm 20$ %
Effective serial resistance	≤ 50 Ω for 19.2 MHz
Shunt capacitance	≤ 7 pF
Drive level	1 mW
Recommended type	HC-49/U (ANSI standard)

Instead of connecting a crystal, an externally generated clock may be supplied as an input signal for XTAL1.

If the OSC is not used, and no signal is supplied to the XTAL1 input pin, XTAL1 should be connected to V_{DD} (+ 5 V).

1.5.3 Digital Phase Lock Loop (DPLL) for Clock Recovery

The ESCC's on-chip DPLL allows synchronous data reception without transferring a sync clock signal over the serial interface. The DPLL recovers the data clock phase from the incoming data stream, thus optimizing the receiver's bit sampling point. The DPLL's reference clock has to be 16 times faster than the nominal data rate, and is always supplied by the BRG (clock mode 2, 3a, 6, 7a).

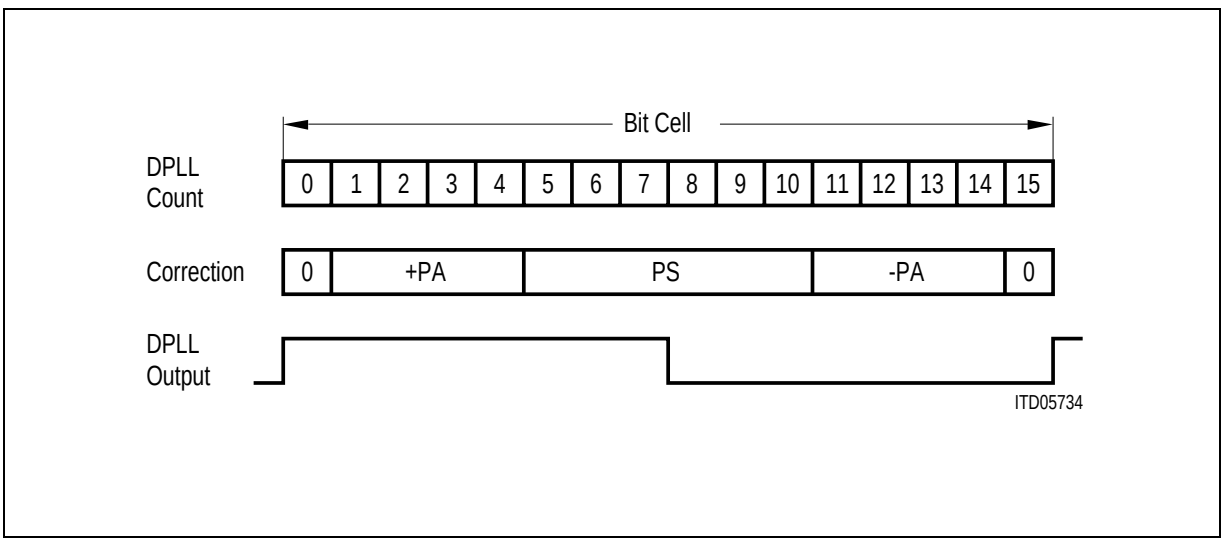
The transmit clock may be obtained from the DPLL (clock modes 3a, 7a), or from the BRG divided by 16 (clock modes 2b, 6b) or is an external clock at the TxCLK pin (clock modes 2a, 6a).

1.5.4 Clock Recovery Algorithms

– **Phase Adjustment**
In the case where an edge appears in the data stream within the PA fields of the time window, the phase will be adjusted by 1/16 of the data clock.

– **Interference Rejection and Spike Filtering**
In the case where two or more edges appear in the data stream within a time period of 16 reference clocks, these are considered as interference and consequently no additional clock adjustment is performed.

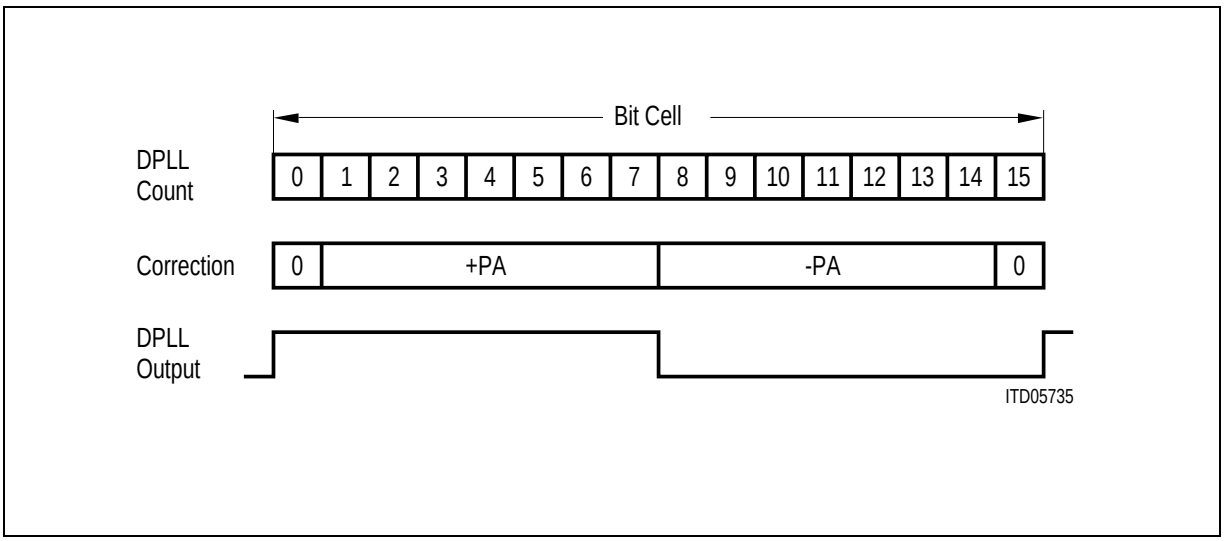
– **Phase Shift (NRZ, NRZI only)**
If NRZ or NRZI coding is selected in CCR0.SC0…2, and the Phase Shift feature is enabled (CCR3.PSD = 0), the DPLL recognizes an edge in the data stream within the PS fields of the time window, takes a second sample of the bit, and shifts the phase by 180 degrees. Edges in all other parts of the time window will be ignored.



DPLL Algorithm for NRZ and NRZI Coding with Phase Shift Enabled

This operation facilitates a **fast** and reliable synchronization for most common applications. One edge on the received data stream is enough for the DPLL to synchronize, thereby eliminating the need for synchronization patterns or preamble characters. However, in case of **extremely** high jitters of the incoming data stream reliable clock recovery cannot be guaranteed.

The Phase Shift function can be **disabled** by setting bit CCR3.PSD. In this case, PA fields are extended as shown below:

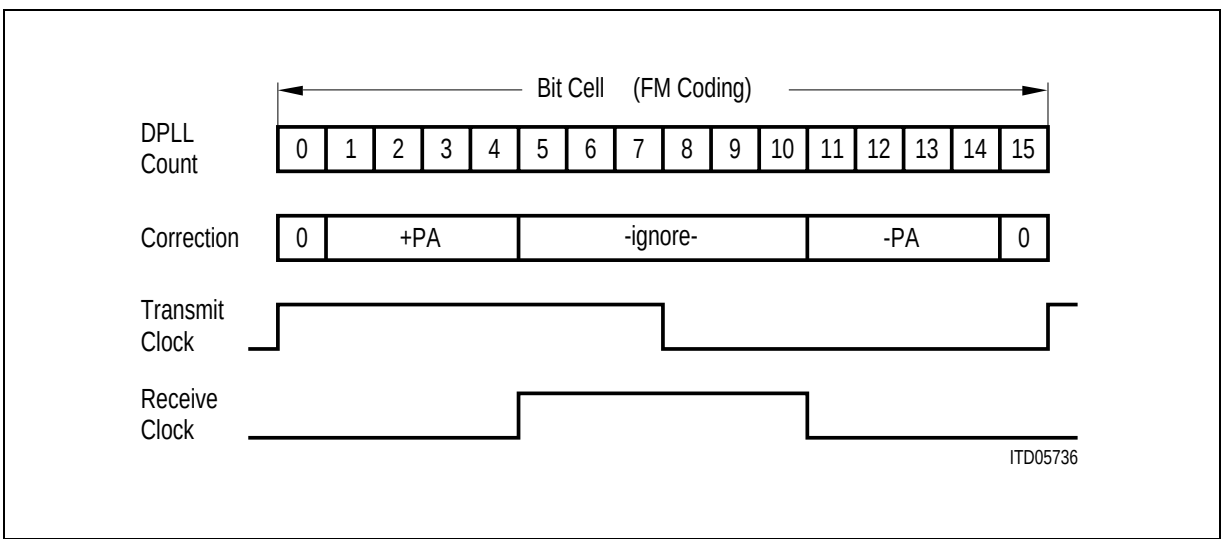


DPLL Algorithm for NRZ and NRZI Coding with Phase Shift Disabled

Now, the DPLL is less sensitive to high jitter amplitudes, but needs **more time** to determine the optimum sampling point. To ensure correct data sampling, preambles should precede the data information.

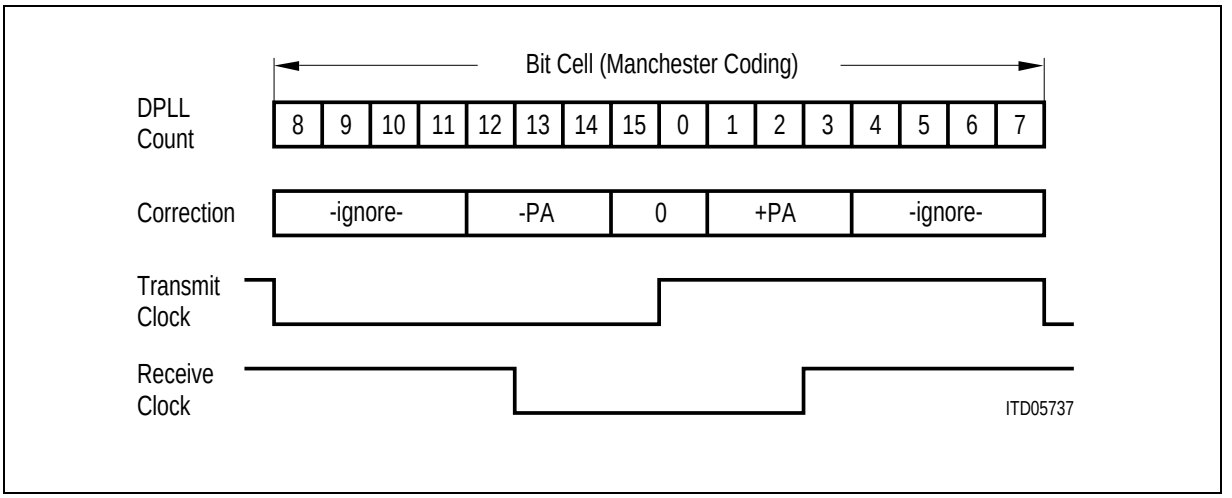
– **DPLL Algorithm for FM0, FM1 and Manchester Coding**

When using bi-phase coding (FM0, FM1, Manchester), a status flag and a maskable interrupt indicate the state of the DPLL (synchronous/asynchronous).



DPLL Algorithm for FM0 and FM1 Coding

If an edge appears in the data stream within the PA fields of the time window, the phase will be adjusted by 1/16 of the data clock.



DPLL Algorithm for Manchester Coding

If an edge appears in the data stream within the PA fields of the time window, the phase will be adjusted by 1/16 of the data clock.

1.5.5 DPLL Sync State Indication

The DPLL state (synchronous or asynchronous) is indicated in the VSTR register. In addition, an interrupt is generated if synchronization is lost and IMR0.PLLA is unmasked (0).

	7						0	
VSTR	CD	DPLA	0	0	VN3		VN0	(34)

(R)

DPLL Asynchronous
This bit is set, if the DPLL lost synchronization.
(Valid only in clock modes 2a, 2b, 3a, 6a, 6b, 7a and if FM0, FM1 or Manchester coding is selected.)

	7							0	
ISR0/ [IMR0]	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF	(3A)

(R/[W])

DPLL Asynchronous
This bit is set, and an interrupt is generated, if the DPLL lost synchronization.
(Valid only in clock modes 2a, 2b, 3a, 6a, 6b, 7a, if FM0, FM1 or Manchester coding is selected, and if IMR0.PLLA = 0.)

1.5.6 The Baud Rate Generator (BRG)

The baud rate generator is used in clock modes 0b, 2a, 2b, 3a, 3b, 6a, 6b, 7a, and 7b. The bit structure of the CCR2 register for these clock modes is shown below:

	7							0	
CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV	(2E)
(R/W)	The bit structure of CCR2 in clock modes 0b, 2, 3, 6, 7								

If CCR2.BDF = 0, the BRG is by-passed (division factor is “1”).

	7							0	
CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV	(2E)

(R/W) **Baud Rate Division Factor**
0: BRG division factor is set to 1 (constant)
1: BRG division factor is determined by BR9…BR0

If CCR2.BDF = 1, the division factor is programmed in CCR2.BR9…8 and BGR.BR7…0:

	7							0	
CCR2	BR9	BR8	BDF=1	SSEL	TOE	RWX	0	DIV	(2E)

(R/W)

									0	
BGR	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	(34)
(W)										
	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0

Baud Rate Division Factor: **k = (N + 1) x 2,** *(N = BR9…0 = 0…1023)*

Value N (BR9…0) for Different BRG Input Frequencies
(supplied by the internal OSC or the RxCLK pin):

In HDLC and BISYNC Mode,
and for isochronous operation in ASYNC Mode (CCR1.BCR = 0):

	6.144 MHz	**7.3728 MHz**	**7.68 MHz**	**9.216 MHz**	**12.288 MHz**	**18.432 MHz**
9.600 Bd	0x13F	0x17F	0x18F	0x1DF	0x27F	0x3BF
19.200 Bd	0x09F	0x0BF	0x0C7	0x0EF	0x13F	0x1DF
38.4 kbit/s	0x04F	0x05F	0x063	0x077	0x09F	0x0EF
115.2 kbit/s	–	0x01F	–	0x027	–	0x04F
153.6 kbit/s	0x013	0x017	0x018	0x01D	0x027	0x03B
256 kbit/s	0x00B	–	0x00E	0x011	0x017	0x023
512 kbit/s	0x005	–	–	0x008	0x00B	0x011
768 kbit/s	0x003	–	0x004	0x005	0x007	0x00B
1.536 Mbit/s	0x001	–	–	0x002	0x003	0x005
2.048 Mbit/s	–	–	–	–	0x002	–

In ASYNC Mode (Standard Asynchronous Operation, CCR1.BCR = 1):

	1.8432 MHz	**2.4576 MHz**	**3.6864 MHz**	**4.9152 MHz**	**6.144 MHz**	**7.3728 MHz**	**9.8304 MHz**	**14.7456 MHz**	**16.0 MHz**	**18.432 MHz**	**24.576 MHz**
300 Bd	0x0BF	0x0FF	0x17F	0x1FF	0x27F	0x2FF	0x3FF	–	–	–	–
1200 Bd	0x02F	0x03F	0x05F	0x07F	0x09F	0x0BF	0x0FF	0x17F	0x1A0	0x1DF	0x27F
9600 Bd	0x005	0x007	0x00B	0x00F	0x013	0x017	0x01F	0x02F	0x033	0x03B	0x04F
19.2 kbit/s	0x002	0x003	0x005	0x007	0x009	0x00B	0x00F	0x017	0x019	0x01D	0x027
38.4 kbit/s	–	0x001	0x002	0x003	0x004	0x005	0x007	0x00B	0x00C	0x00E	0x013
115.2 kbit/s	BDF=1	–	0x0	–	–	0x001	–	0x003	–	0x004	–
153.2 kbit/s	–	BDF=1	–	0x0	–	–	0x001	0x002	–	–	0x004

2 Register Map of the ESCC2

2.1 Summary

The ESCC2 supports a variety of synchronous and asynchronous protocols and data formats with four basic serial modes and several sub-modes. The interpretation of the ESCC2's registers depends on the serial mode. Chapter 2 shows the register map and the bit structure of the ESCC2 for each serial mode.

Wolfram Kiesecker
January 1994

2.2 Four Basic Serial Modes

Serial Mode Selection

The ESCC2 offers four basic serial modes:

- HDLC/SDLC Mode
- SDLC Loop Mode
- ASYNC Mode
- BISYNC Mode

The basic serial mode is selected in the CCR0 register:

	7							0	
CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0	(2C)

Serial Mode SM1…0:	**HDLC/SDLC**	**0 0**
	SDLC Loop	**0 1**
	BISYNC	**1 0**
	ASYNC	**1 1**

Important: The interpretation of the ESCC2 registers depends on the serial mode. Therefore, CCR0.SM1…0 have to be initialized before writing to any other register in the ESCC!

The following eight pages show the **ESCC2** register map and the bit structure for each serial mode:

2.2.1 ESCC2 Register Map in HDLC/SDLC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	RSTA	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	----								
27	----								
28	RAL1	RAL1 (in transparent and extended transparent mode only)							
29	RHCR	RHCR							
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	NRM	CAS	OV	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	ISR1	0	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	RAH1	RAH1						CRI	0
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	1	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

2.2.2 ESCC2 Register Map in SDLC-Loop Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	RSTA	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	MODE	MDS1	MDS0	ADM=0	TMD=0	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (command)							
25	XAD2	XAD2 (response)							
26	----								
27	----								
28	RAL1	RAL1 (in transparent and extended transparent mode only)							
29	RHCR	RHCR							
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	NRM=1	CAS	OV	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2…SC0 = 000 or 010				**SM1…SM0 = 01**
2D	CCR1	SFLG	GALP	GLP	ODS	ITF=1	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	ISR1	EOP	OLP	AOLP	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM=0	TMD=0	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (command)							
25	XAD2	XAD2 (response)							
26	RAH1	RAH1 = 0x00							
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2…SC0 = 000 or 010			**SM1…SM0 = 01**	
2D	CCR1	SFLG	GALP	GLP	ODS	ITF=1	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	EOP	OLP	AOLP	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

2.2.3 ESCC2 Register Map in BISYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	RFNE	SYNC	0	CEC	CTS	0
21	**----**								
22	**MODE**	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**SYNL**	SYNL							
25	**SYNH**	SYNH							
26	**TCR**	TCR7							TCR0
27	**DAFO**	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	**RFC**	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	**----**								
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	NRM	CAS	0	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 10**	
2D	**CCR1**	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	**CCR3**	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	**IPC**	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	**ISR1**	0	0	ALLS	XDU	TIN	CSC	XMR	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**----**								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES
21	PRE	PR7							PR0
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 10**	
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	----								
36	----								
37	----								
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	1	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	IMR1	1	1	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

2.2.4 ESCC2 Register Map in ASYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	RFNE	FCS	TEC	CEC	CTS	0
21	**----**								
22	**MODE**	0	0	0	FLON	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**XON**	XON7							XON0
25	**XOFF**	XOF7							XOF0
26	**TCR**	TCR7							TCR0
27	**DAFO**	0	XBRK	STOP	PAR1	PAR0	PARE	CHL1	CHL0
28	**RFC**	0	DPS	0	RFDF	RFTH1	RFTH0	0	TCDE
29	**----**								
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	0	CAS	0	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 11**	
2D	**CCR1**	0	0	0	ODS	BCR	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	**CCR3**	0	0	0	0	0	0	0	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	**IPC**	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	RPF
3B	**ISR1**	BRK	BRKT	ALLS	XOFF	TIN	CSC	XON	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**----**								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	0	0	XRES
21	----								
22	MODE	0	0	0	FLON	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XON	XON7							XON0
25	XOFF	XOF7							XOF0
26	TCR	TCR7							TCR0
27	DAFO	0	XBRK	STOP	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	0	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 11**	
2D	CCR1	0	0	0	ODS	BCR	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	0	0	0	0	0	0	0	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	TIC	TIC7							TIC0
36	MXN	MXN							MXN0
37	MXF	MXF							MXF
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	RPF
3B	IMR1	BRK	BRKT	ALLS	XOFF	TIN	CSC	XON	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

3 Register Map of the ESCC8

3.1 Summary

The interpretation of the ESCC8's registers is very similar to the ESCC2's. It also depends on the serial mode, as described in the previous chapter. However, the ESCC8 uses a different structure for some registers, which is taken into consideration in Chapter 3.

Wolfram Kiesecker
January 1994

3.2 Four Basic Serial Modes

Serial Mode Selection

Like the ESCC2, the ESCC8 offers four basic serial modes:

- HDLC/SDLC Mode
- SDLC Loop Mode
- ASYNC Mode
- BISYNC Mode

The basic serial mode is selected in the CCR0 register:

	7							0	
CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0	(2C)

Serial Mode SM1…0:	**SM1**	**SM0**
HDLC/SDLC	**0**	**0**
SDLC Loop	**0**	**1**
BISYNC	**1**	**0**
ASYNC	**1**	**1**

Important: The interpretation of the ESCC registers depends on the serial mode. Therefore, CCR0.SM1…0 have to be initialized before writing to any other register in the ESCC!

The following eight pages show the **ESCC8** register map and the bit structure for each serial mode:

3.2.1 ESCC8 Register Map in HDLC/SDLC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	RSTA	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	----								
27	----								
28	RAL1	RAL1 (in transparent and extended transparent mode only)							
29	RHCR	RHCR							
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	NRM	CAS	OV	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	ISR1	0	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	RAH1	RAH1						CRI	0
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	1	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

3.2.2 ESCC8 Register Map in SDLC-Loop Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	RSTA	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	MODE	MDS1	MDS0	ADM=0	TMD=0	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (command)							
25	XAD2	XAD2 (response)							
26	----								
27	----								
28	RAL1	RAL1 (in transparent and extended transparent mode only)							
29	RHCR	RHCR							
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	NRM=1	CAS	OV	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2…SC0 = 000 or 010				**SM1…SM0 = 01**
2D	CCR1	SFLG	GALP	GLP	ODS	ITF=1	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	ISR1	EOP	OLP	AOLP	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM=0	TMD=0	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (command)							
25	XAD2	XAD2 (response)							
26	RAH1	RAH1 = 0x00							
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2…SC0 = 000 or 010			**SM1…SM0 = 01**	
2D	CCR1	SFLG	GALP	GLP	ODS	ITF=1	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	EOP	OLP	AOLP	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

3.2.3 ESCC8 Register Map in BISYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	RFNE	SYNC	0	CEC	CTS	0
21	**----**								
22	**MODE**	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**SYNL**	SYNL							
25	**SYNH**	SYNH							
26	**TCR**	TCR7							TCR0
27	**DAFO**	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	**RFC**	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	**----**								
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	NRM	CAS	0	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 10**	
2D	**CCR1**	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	**CCR3**	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	**IPC**	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	**ISR1**	0	0	ALLS	XDU	TIN	CSC	XMR	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**----**								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES
21	PRE	PR7							PR0
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 10**	
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	----								
36	----								
37	----								
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	1	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	IMR1	1	1	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

3.2.4 ESCC8 Register Map in ASYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	RFNE	FCS	TEC	CEC	CTS	0
21	**----**								
22	**MODE**	0	0	0	FLON	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**XON**	XON7							XON0
25	**XOFF**	XOF7							XON7
26	**TCR**	TCR7							TCR0
27	**DAFO**	0	XBRK	STOP	PAR1	PAR0	PARE	CHL1	CHL0
28	**RFC**	0	DPS	0	RFDF	RFTH1	RFTH0	0	TCDE
29	**----**								
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	0	CAS	0	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 11**	
2D	**CCR1**	0	0	0	ODS	BCR	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	**CCR3**	0	0	0	0	0	0	0	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	**IPC**	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	RPF
3B	**ISR1**	BRK	BRKT	ALLS	XOFF	TIN	CSC	XON	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**----**								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	0	0	XRES
21	----								
22	MODE	0	0	0	FLON	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XON	XON7							XON0
25	XOFF	XOF7							XOF0
26	TCR	TCR7							TCR0
27	DAFO	0	XBRK	STOP	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	0	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 11**	
2D	CCR1	0	0	0	ODS	BCR	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	0	0	0	0	0	0	0	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	TIC	TIC7							TIC0
36	MXN	MXN							MXN0
37	MXF	MXF							MXF
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	RPF
3B	IMR1	BRK	BRKT	ALLS	XOFF	TIN	CSC	XON	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

4 HDLC/SDLC Cook-Book

4.1 Summary

HDLC/SDLC Mode is subdivided into several operating modes, with different levels frame processing and recognition. Chapter 4 provides a number of programming examples for each of the operating modes. The examples are shown as track files for the EASY532 development system.

Chapter 4 has been derived from the track files and application notes developed by Frank Sauber at Siemens in Santa Clara in 1992/93.

Wolfram Kiesecker
1994

4.2 HDLC/SDLC Mode

4.2.1 Overview

HDLC/SDLC mode is a bit-synchronous data transmission mode. It is selected in the CCR0-register with SM0 and SM1:

	7							0	
CCR0	PU	MCE	0	SC2	SC1	SC0	SM1*	SM0*	(2C)

Serial Mode SM1…0:	**HDLC/SDLC**	**0**	**0**
	SDLC Loop	0	1
	MONO-/ BISYNC	1	0
	ASYNC	1	1

* Throughout this document: **The unshaded bit-fields are the most relevant for the examined function.**

HDLC/ SDLC mode offers four sub-modes:

- auto mode
- non-auto mode
- transparent mode
- extended transparent mode

The sub-mode is selected in the MODE-register with MDS0 and MDS1:

	7							0	
MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP	(22)

0	0	auto mode
0	1	non-auto mode
1	0	transparent mode
1	1	extended transparent mode

The sub-modes are differentiated in the amount of protocol support given by the ESCC. For example, in auto mode, I frames and S frames (RR, RNR) are generated by the ESCC autonomously. In non auto mode, the framing is done by the ESCC, but no auto hand-shaking is performed. In transparent mode, only the flags are appended; optionally, the ESCC can do partial address recognition and CRC error protection. Finally, in the extended transparent mode, transmission and reception of any bit-pattern is possible without modification by the ESCC. In this sub-mode, it is the software's responsibility to screen the bit stream in the FIFO and perform any kind of frame- or byte-

synchronize between the transmitter and the receiver. The extended transparent mode offers the fullest flexibility for implementing proprietary protocols, or process non-HDLC data (like voice or video).

We will start our HDLC-mode description with an example for the extended transparent mode, and then work our way up to the more sophisticated HDLC-protocol implementations of the ESCC.

4.3 Extended Transparent Mode

In the extended transparent mode, fully transparent data transmission/reception without HDLC framing is performed, i.e. without flag and address generation/recognition, CRC generation/check, or bit-stuffing. This allows user specific protocol variations.

Pre-requisites for extended transparent mode:
MODE.RAC = 0
XAD1 = 0xFF
XAD2 = 0xFF
RAH2 = 0xFF

Extended transparent mode either uses the RAL1 register for data reception (extended transparent mode 0) or the RFIFO (extended transparent mode 1):

	7							0	
MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP	(22)

1	1	0	1	extended transparent mode 0
1	1	1	1	extended transparent mode 1

Data transmission is always performed out of the XFIFO. In extended transparent mode 0 (MODE.ADM = 0) data reception is done via the RAL1 register. In extended transparent mode 1 (MODE.ADM = 1) the received bit-stream is shifted into the RFIFO. Our example will run the ESCC2 in the extended transparent mode 1.

Data transmission is only possible if the CTS input is active (low)! The state of the CTS-pin is reflected by the CTS-bit in the STAR-register:

	7							0	
STAR	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA	(20)

/CTS is inactive (high):	0
/CTS is active (low):	1

Example

- Extended transparent mode 1
- Repeated data transmission out of the XFIFO (this feature exists in extended transparent mode only)

	7							0	
CMDR	RMC	RHR	XREP	STI	XTF	XIF	XME	XRES	(20)
	0	0	1	0	1	0	1	0	

repeated transmission in extended transparent mode

- Test loop (Transmitter and receiver of the same channel are internally connected.)

4.3.1 Track File HEXTRP01

```
#: ************************************
#: File name: HEXTRP01.TRK
#: Hdlc EXtended TRansParent mode 1
#: clockmode 7b
#: Testloop
#: Transmission Repeat
#:
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
#           The following pins have to be
#           connected on the UCI:
#           CTSA (33) <- lo
#           CTSB (23) <- lo
#:
#: ************************************
#:
#:    HDLC mode:
WRB:  00 => CCR0
#:
#:    Extended transparent mode 1
#:    Test Loop
WRB:  E1 => MODE
#:
#:    Clock mode 7b
#:    Baud Rate Generator enabled
WRB:  07 => CCR1
WRB:  30 => CCR2
#:
#:    Baud Rate = 1Mbit/s
#:    OSC = 16 MHz; => k=16
#:    BR9…0 = 7   (k= [7+1]*2)
WRB:  07 => BRG
#:
#:    Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:    Power up in HDLC mode
WRB:  80 => CCR0
#:
#:    Write data into XFIFO
```

```
WRB:  55 => XFIFO
#:
#:    Check CTS state in STAR
#:    then issue
#:    Transmission Repeat
#:    XREP, XTF, XME
RDB:  4A <= STAR  ! 48
WRB:  2A => CMDR
#:
#:    Receiver Reset &
#:    Transmission Repeat
WRB:  60 => CMDR
#:
#:    Enable RPF-Interrupt
WRB:  FE => IMR0
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
RDD:  32 Bytes    <= RFIFO
#:
#:    Acknowledge RFIFO &
#:    Transmission Repeat
WRB:  A0 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
RDD:  32 Bytes    <= RFIFO
WRB:  A0 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
```

```
INT:  Interrupt(s) acknowledged
#:
#:    Disable RPF-Interrupt
WRB:  FF => IMR0
#:
#:Read RFIFO
RDD:  Data Block  <= RFIFO
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
      AA AA AA AA AA AA AA AA
RDD:  32 Bytes    <= RFIFO
WRB:  A0 => CMDR
#:
#:    Transmitter Reset
#:    Power Down
WRB:  01 => CMDR
WRB:  00 => CCR0
#:
#:    End of Track File
```

Note: Whenever a new command byte is written to the CMDR register (like RMC, RRES), the XREP bit has to be set, too. Otherwise, the ESCC will stop the repeated transmission.

Since no frame or byte synchronization is performed in extended transparent mode, the bit-position of the receiver is random.

Transmitter:

…→	0x55	0x55	0x55	→…

Serial bit stream (LSB first):

→	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	→

Receiver:

…	0xAA	0xAA	0xAA	…

The RFIFO content is a snap shot of 32 x 8 bits on the RxD pin. Frame or byte boundaries cannot be determined.

Example

- Extended transparent mode 1, test loop
- Repeated transmission of the bit pattern 0x11, 0x33 (→0011001100010001→)

4.3.2 Track File HEXTRP02

```
#: ************************************
#: File name: HEXTRP02.TRK
#: Hdlc EXtended TRansParent mode 1
#: clockmode 7b
#: Testloop
#: Transmission Repeat
#:
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
#          The following pins have to be
#          connected on the UCI:
#          CTSA (33) <- lo
#          CTSB (23) <- lo
#:
#: ************************************
#:    HDLC mode:
WRB:  00 => CCR0
#:    Extended transparent mode 1
#:    Test Loop
WRB:  E1 => MODE
#:
#:    Clock mode 7b
#:    Baud Rate Generator enabled
WRB:  07 => CCR1
WRB:  30 => CCR2
#:
#:    Baud Rate = 1Mbit/s
#:    OSC = 16 MHz;  => k=16
#:    BR9…0 = 7   (k= [7+1]*2)
WRB:  07 => BRG
#:
#:    Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:    Power up in HDLC mode
WRB:  80 => CCR0
#:
#:    Write data into XFIFO
WRD:  Data Block  => XFIFO
      11 33
```

```
WRD:   2 Bytes    => XFIFO
#:
#:    Check CTS state in STAR
#:    then issue
#:    Transmission Repeat
#:    XREP, XTF, XME
RDB:  4A <= STAR
WRB:  2A => CMDR
#:
#:    Receiver Reset &
#:    Transmission Repeat
WRB:  60 => CMDR
#:
#:    Enable RPF-Interrupt
WRB:  FE => IMR0
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#:    Disable RPF-Interrupt
WRB:  FF => IMR0
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      89 98 89 98 89 98 89 98
      89 98 89 98 89 98 89 98
      89 98 89 98 89 98 89 98
      89 98 89 98 89 98 89 98
RDD:  32 Bytes    <= RFIFO
WRB:  A0 => CMDR
#:
#:    Transmitter Reset
#:    Power Down
WRB:  01 => CMDR
WRB:  00 => CCR0
#:
#:    End of Track File
```

Transmitter

…→	0x11	0x33	0x11	→…

Serial bit stream (LSB first)

→	1	1	0	0	0	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	→

Receiver

…	0x89	0x98	0x89	

Example

- Extended transparent mode 1.
- Repeated transmission from channel A to channel B.
  (The data and handshake pins of the User Connector Interface (UCI) on the EASY532 board have to be connected as follows:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.3.3 Track File HEXTRP03

```
#: ************************************
#: File name: HEXTRP03.TRK
#: Hdlc EXtended TRansParent mode 1
#: Channel A -> B
#: clockmode 7b -> 6b (DPLL)
#: Transmission Repeat
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Channel A Initialization
#:       HDLC, extended transparent mode 1,
#:       FM0 encoding:
WRB:  10 => CCR0
WRB:  E0 => MODE
#:
#:       Clock mode 7b
#:       Baud Rate Generator enabled
#:       TxD pin is a push-pull output
WRB:  17 => CCR1
WRB:  30 => CCR2
#:
#:       Baud Rate = 1Mbit/s
#:       OSC = 16 MHz;  => k=16
#:       BR9…0 = 7   (k= [7+1]*2)
WRB:  07 => BRG
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:
```

```
#:       Write data into XFIFO
WRD:  Data Block  => XFIFO
      12 34 56
WRD:   3 Bytes    => XFIFO
#:
#:       Transmission Repeat
#:       XREP, XTF, XME
WRB:  2A => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, extended transparent mode 1,
#:       FM0 encoding,
#:       activate RTS output:
WRB:  10 => CCR0
WRB:  E4 => MODE
#:
#:       clock mode 6b,
#:       BRG enabled,
#:       division factor = 1 (16*DPLL):
WRB:  16 => CCR1
WRB:  10 => CCR2
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:
#:
#:       Receiver Reset
WRB:  40 => CMDR
#:
#:       Enable PLLA, RPF-Interrupt
WRB:  F6 => IMR0
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      A0 B1 92 A0 B1 92 A0 B1
      92 A0 B1 92 A0 B1 92 A0
```

```
      B1 92 A0 B1 92 A0 B1 92
      A0 B1 92 A0 B1 92 A0 B1
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      92 A0 B1 92 A0 B1 92 A0
      B1 92 A0 B1 92 A0 B1 92
      A0 B1 92 A0 B1 92 A0 B1
      92 A0 B1 92 A0 B1 92 A0
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#:       Disable RPF-Interrupt
WRB:  F7 => IMR0
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      B1 92 A0 B1 92 A0 B1 92
      A0 B1 92 A0 B1 92 A0 B1
      92 A0 B1 92 A0 B1 92 A0
      B1 92 A0 B1 92 A0 B1 92
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Transmitter Reset
#:       Power Down
WRB:  01 => CMDR
WRB:  10 => CCR0
#:
#:       End of Track File
```

Transmitter

…→	0x56	0x34	0x12	→…

Serial bit stream (LSB first)

→	1	0	0	1	0	1	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	→

Receiver

…	0xB1	0xA0	0x92

4.3.4 Hardware Handshaking

Since EASY532-Chip Level is an interactive tool, it is very difficult to perform real-time tasks. Here is a way how to loosely synchronize data transmission and reception of a 32 byte block. The ESCC channel A transmitter is enabled by an inactive-to-active transition on the CTSA input simultaneously to the channel B receiver reset command. This is achieved by filling channel B's XFIFO before issuing a combined RRES and XTF/XME command. In this configuration, the RTSB output, which has to be connected to the CTSA input, becomes active almost simultaneously with the receiver.

Example

- Extended transparent mode 1.
- Single transmission from channel A to channel B.
- Transmitter controlled with CTS pin.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.3.5 Track File HEXTRP04

```
#: ************************************
#: File name: HEXTRP04.TRK
#: Hdlc EXtended TRansParent mode 1
#: Channel A (7b) -> B (6b)
#: CTSA - RTSB transmission control
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Channel A Initialization
#:       HDLC, extended transparent mode 1,
#:       FM0 encoding,
#:       activate RTSA output:
WRB:  10 => CCR0
WRB:  E4 => MODE
#:
#:       Clock mode 7b
#:       Baud Rate = 1Mbit/s
#:       TxD pin is a push-pull output
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up
WRB:  90 => CCR0
#:
#:       Write data into XFIFO
WRD:  Data Block  => XFIFO
```

```
      00 01 02 03 04 05 06 07
      08 09 0A 0B 0C 0D 0E 0F
      7F 7E 7D 7C 7B 7A 79 78
      77 76 75 74 73 72 71 70
WRD:  32 Bytes    => XFIFO
#:
#:       Make sure CTSA is not active yet!
RDB:  48 <= STAR
#:       XTF, XME command
WRB:  0A => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, extended transparent mode 1,
WRB:  10 => CCR0
WRB:  E0 => MODE
#:
#:       Clock mode 6b
#:       Div. Factor =1
#:       TxD pin is a push-pull output
WRB:  16 => CCR1
WRB:  10 => CCR2
#:
#:    Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up
WRB:  90 => CCR0
#:       Dummy character block
#:       to force RTSB low
WRD:  Data Block  => XFIFO
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
WRD:  32 Bytes    => XFIFO
#:
#:
#:       Receiver Reset
#:       XTF, XME command
WRB:  4A => CMDR
#:
#:       Enable PLLA, RPF-Interrupt
WRB:  F6 => IMR0
```

```
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      FF FF FF FF 0F 10 20 30
      40 50 60 70 80 90 A0 B0
      C0 D0 E0 F0 F0 E7 D7 C7
      B7 A7 97 87 77 67 57 47
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#:       Disable RPF-Interrupt
WRB:  F7 => IMR0
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      37 27 17 07 F7 FF FF FF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Power Down
WRB:  10 => CCR0
#:
#:       End of Track File
```

The 32 x 8 bits from channel A are found in the two RFIFO halves of channel B. The data line is in IDLE state before and after the transmission. (“1” = inactive line):

Transmitter (channel A)

…→	0x70	0x71, 72 ... 03, 02, 01	0x00	→…

Serial bit stream (LSB first)

idle			data																						idle			
1	1	1	0	1	1	1	0	0	0	0	0	1	1	1		0	1	0	0	0	0	0	0	0	0	1	1	1	1

Receiver (channel B)

0xF7	0x07	0x17, 27 ... 30, 20, 10	0x0F

4.4 Transmission in Transparent Mode

If the transmitter is in Transparent Mode, it generates opening and closing flags (0x7E) autonomously. These flags are found in the receiver's RFIFO, if it is in Extended Transparent Mode. For transmitter/receiver synchronization the same CTS handshaking mechanism is used as in the previous example.

Example

- Channel A (transmitter) in transparent mode 0.
- Channel B (receiver) in extended transparent mode 1.
- Transmitter controlled with CTS pin.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.4.1 Track File HATRBEX1

```
#: ************************************
#: File name: HATRBEX1.TRK
#: Hdlc channel A TRansparent mode 0
#: channel B EXtended transparent md 1
#: Channel A (7b) -> B (6b)
#: CTSA - RTSB transmission control
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Channel A Initialization
#:       HDLC, FM0, transparent mode 0,
#:       activate RTSA output:
WRB:  10 => CCR0
WRB:  84 => MODE
#:
#:       Clock mode 7b
#:       Baud Rate Generator enabled
#:       TxD pin is a push-pull output
WRB:  17 => CCR1
WRB:  30 => CCR2
#:
#:       Disable CRC generation:
WRB:  02 => CCR3
#:
#:       Baud Rate = 1Mbit/s
WRB:  07 => BRG
#:
#:       Power up
WRB:  90 => CCR0
#:
#:       Write data into XFIFO
WRD:  Data Block  => XFIFO
```

```
      01 80
WRD:   2 Bytes    => XFIFO
#:
#:       Make sure CTSA is not active yet!
RDB:  48 <= STAR
#:      XTF, XME command
WRB:  0A => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, FM0,
#:       extended transparent mode 1:
WRB:  10 => CCR0
WRB:  E0 => MODE
#:
#:       Clock mode 6b
#:       Baud Rate Div. Factor = 1
#:       TxD pin is a push-pull output
WRB:  16 => CCR1
WRB:  10 => CCR2
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:       Dummy character block
#:       to force RTSB low
WRD:  Data Block  => XFIFO
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
WRD:  32 Bytes    => XFIFO
#:
#:
#:       Receiver Reset
#:       XTF, XME command
WRB:  4A => CMDR
#:
#:       Enable PLLA, RPF-Interrupt
WRB:  F6 => IMR0
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
```

```
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#:       Disable RPF-Interrupt
WRB:  F7 => IMR0
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      FF FF FF FB 05 00 FA FD
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Power Down
WRB:  00 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Power Down
WRB:  00 => CCR0
#:
#:       End of Track File
```

Interpretation

The RFIFO content of channel B is:

```
FF FF FF FB 05 00 FA FD FF FF FF, …,
```

which correlates with the following bit-stream:

```
-> |__FF__||__FD__||__FA__||__00__||__05__||__FB__||__FF__| ->
-> 11111111111111011111101000000000000001011111101111111111 ->
```

The beginning and end of a frame is marked by flags (0x7E). The data bytes from channel A's XFIFO reside in between (0x01, 0x80):

```
-> 11111111111111011111101000000000000001011111101111111111 ->
              -> |__7E__||__80__||__01__||__7E__| ->
              ->   end    data2   data1    start  ->
```

4.4.2 Zero-Insertion in Transparent Mode

If more than five consecutive 1s occur within the data pattern, a 0-bit is automatically inserted by the transmitter in transparent mode:

Example

- Channel A (transparent mode 0), transmitting XFIFO content 0x10, 0x7E, 0x80.
- Channel B (extended transparent mode 1), receiving.
- Transmitter controlled with CTS pin.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.4.3 Track File HATRBEX2

```
#: ************************************
#: Frank Sauber ICD Santa Clara USA
#: file name: HATRBEX2.TRK
#: Hdlc channel A TRansparent mode 0
#: channel B EXtended transparent md 1
#: Channel A (7b) -> B (6b)
#: CTSA - RTSB transmission control
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Channel A Initialization:
#:       HDLC, FM0, transparent mode 0,
#:       RTSA active, TxDA push-pull,
#:       CM 7b, Baud Rate = 1 MBit/s,
#:       CRC disabled:
WRB:  10 => CCR0
WRB:  84 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  02 => CCR3
WRB:  07 => BRG
#:
#:       Power up
WRB:  90 => CCR0
#:
#:       Write data into XFIFO
WRD:  Data Block  => XFIFO
      01 7E 80
WRD:   3 Bytes    => XFIFO
#:
#:       Check CTSA in-active!
#:       Issue XTF/XME
```

```
RDB:  48 <= STAR
WRB:  0A => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, FM0,
#:       extended transparent mode 1,
#:       clock mode 6b (DPLL),
#:       Baud rate division factor = 1:
WRB:  10 => CCR0
WRB:  E0 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up
WRB:  90 => CCR0
#:       Dummy character block
#:       to force RTSB low
WRD:  Data Block  => XFIFO
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
WRD:  32 Bytes    => XFIFO
#:
#:
#:       Receiver Reset
#:       XTF, XME command
WRB:  4A => CMDR
#:
#:       Enable PLLA, RPF-Interrupt
WRB:  F6 => IMR0
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
#:
#:       Disable RPF-Interrupt
WRB:  F7 => IMR0
#:
#: Read RFIFO
```

```
RDD:  Data Block  <= RFIFO
      FF FF DF 2F C0 17 A0 DF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Power Down
WRB:  10 => CCR0
#:
#:       End of Track File
```

Zero-Bit Insertion

The character 0x7E in the data field of the frame was modified by the transmitter

```
       ->   |__DF__||__A0__||__17__||__C0__||__2F__||__DF__| ->
       -> 1111011111101000000001011111000000001011111011111 ->
           -> |__7E__||__80__||_^_7E__||__01__||__7E__| ->
           ->   end    data3    data2    data1   start  ->
```

4.4.4 CRC Generation in Transparent Mode

If enabled with CCR3.XCRC = 0, a checksum is automatically calculated and appended to the data field of the frame:

Example

- Channel A (transparent mode 0), transmitting with CRC generation enabled
- Channel B (extended transparent mode 1), receiving.
- Transmitter controlled with CTS pin.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.4.5 Track File HATRBEX3

```
#: ************************************
#: File name: HATRBEX3.TRK
#: Hdlc channel A TRansparent mode 0
#: channel B EXtended transparent md 1
#: clockmode 7b -> 6b
#: Channel A -> B
#: CTSA - RTSB transmission control
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Channel A Initialization
#:       HDLC, transparent mode 0,
#:       FM0 data encoding
#:                activate RTSA output
WRB:  10 => CCR0
WRB:  84 => MODE
#:
#:       Clock mode 7b
#:       Baud Rate Generator enabled
#:       TxD pin is a push-pull output
WRB:  17 => CCR1
WRB:  30 => CCR2
#:
#:       Baud Rate = 1Mbit/s
#:       OSC = 16 MHz;  => k=16
#:       BR9…0 = 7      (k= [7+1]*2)
WRB:  07 => BRG
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:
#:       Write data into XFIFO
```

```
WRB:  55 => XFIFO
#:
#:       Make sure CTSA is not active yet!
RDB:  48 <= STAR
#:       XTF, XME command
WRB:  0A => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, extended transparent mode 1,
#:       FM0 coding
WRB:  10 => CCR0
WRB:  E0 => MODE
#:
#:       Clock mode 6b
#:       Baud Rate Generator enabled
#:       division factor =1 (DPLL*16)
#:       TxD pin is a push-pull output
WRB:  16 => CCR1
WRB:  10 => CCR2
#:
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:       Dummy character block
#:       to force RTSB low
WRD:  Data Block  => XFIFO
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
      00 00 00 00 00 00 00 00
WRD:  32 Bytes    => XFIFO
#:
#:       Receiver Reset
#:       XTF, XME command
WRB:  4A => CMDR
#:
#:       Enable PLLA, RPF-Interrupt
WRB:  F6 => IMR0
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
```

```
INT:  Interrupt(s) acknowledged
#:
#:       Disable PLLA, RPF-Interrupt
WRB:  FF => IMR0
#:
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      FF FF BF 5F 15 54 BD DF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
      FF FF FF FF FF FF FF FF
RDD:  32 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Power Down
WRB:  10 => CCR0
#:
#:       End of Track File
```

CRC Generation

```
       ->   |__DF__||__BD__||__54__||__15__||__5F__||__BF__| ->
       -> 1111011111101111010101010000010101010111111010111111 ->
           -> |__7E__||__F5__||__50__||__55__||__7E__| ->
           ->   end     crc2    crc1    data1   start  ->
```

The 16-bit CRC for 0x55 was calculated as 0xF550.

4.5 Receiver in Transparent Mode 0

Transparent Mode 0 recognizes incoming frames and stores their content in the RFIFO. A CRC checksum for the received data is calculated and compared with the received CRC value. The result of this comparison is indicated by the RSTA.CRC bit.

RSTA.CRC = 0 : CRC check failed

RSTA.CRC = 1 : CRC check ok

The next example uses channel A for transmission in extended transparent mode 1, and channel B as transparent mode 0 receiver. This allows the creation of frames with valid and invalid CRC octets in channel A's XFIFO. The RSTA of channel B is examined for both cases:

Example

- Channel A (ext. transp. mode 1), transmitting frames with correct and wrong CRC.
- Channel B (transparent mode 0), receiving.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.5.1 Track File HEXTRAM1

```
#: ************************************
#: File name: HEXTRAM1.TRK
#: Hdlc channel A EXt. transp. mode 1
#: channel B TRAnsparent Mode 0
#: clockmode 7b -> 7a
#: Channel A -> B
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:    Channel A Initialization
#:    HDLC, extended transparent mode 1,
#:       FM0, Clock mode 7b
#:       Baud Rate Generator enabled
#:       Baud Rate = 1Mbit/s
#:       TxD pin is a push-pull output
WRB:  10 => CCR0
WRB:  E0 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
#:       Pre-requisits:
WRB:  FF => XAD1
WRB:  FF => XAD2
WRB:  FF => RAH2
#:
#:       Power up in HDLC mode
WRB:  90 => CCR0
#:
#:       Channel B Initialization
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       HDLC, transparent mode 0
```

```
#:    FM0, clock mode 6b,
#:    Receiver active,
#:       activate RTSA output
#:       Baud rate div. factor = 1
#:       TxD push-pull output, store CRC
WRB:  10 => CCR0
WRB:  8C => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  04 => CCR3
#:    Enable RME, PLLA-interrupt,
#:       Power up in HDLC mode
WRB:  77 => IMR0
WRB:  90 => CCR0
#:       Receiver Reset
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:    send frame with correct CRC
WRD:  Data Block  => XFIFO
      7E 55 50 F5 7E
WRD:   5 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
#:
RDB:  04 <= RBCL
RDD:  Data Block<= RFIFO
      55 50 F5 A2
RDD:   4 Bytes    <= RFIFO
RDB:  A2 <= RSTA
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  01 => CMDR
#:
#:    send frame with wrong CRC
WRD:  Data Block  => XFIFO
      7E 55 51 F5 7E
WRD:   5 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
```

```
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
#:
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      55 51 F5 82
RDD:   4 Bytes    <= RFIFO
RDB:  82 <= RSTA
WRB:  80 => CMDR
#:    done
WRB:  FF => IMR0
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Power Down
WRB:  10 => CCR0
#:
#:       End of Track File
```

The result of the CRC check is indicated in the RSTA register

Received Frame	RSTA	Data	CRC	RSTA.CRC
0x55, 0x50, 0xF5	0xA2	0x55	0xF550	0
0x55, 0x51, 0xF5	0x82	0x55	0xF551	1

4.6 Transmitter and Receiver in Transparent Mode

If the transmitter and the receiver are both in Transparent Mode, HDLC framing (flags), zero insertion/extraction and CRC generation/check are performed automatically. From a received frame only the data is stored in the RFIFO, and a status byte (copy of the RSTA register) is appended. The CRC octets may be stored in the RFIFO as an option (CCR3.RCRC = 1):

Example

- Channel A (transparent mode 0), transmitting
- Channel B (transparent mode 0), receiving, CRC not stored in the RFIFO
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.6.1 Track File HTRPMD01

```
#: ************************************
#: File name : HTRPMD01.TRK
#: Hdlc TRansParent MoDe 0
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Channel A Initialization
#:       HDLC, FM0, transparent mode 0,
#:       Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  80 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
#:       Channel B Initialization
#:       HDLC, FM0, transparent mode 0,
#:       DPLL, baud rate div. factor = 1:
WRB:  10 => CCR0
WRB:  84 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  90 => CCR0
#:
#:       Enable PLLA, RME-Interrupt
WRB:  77 => IMR0
#:
#:    Enable receiver and reset:
WRB:  8C => MODE
```

```
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Write data into XFIFO
WRD:  Data Block  => XFIFO
      01 7E 80
WRD:   3 Bytes    => XFIFO
#:
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
#:
#:       Channel B interrupt
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
#:
RDB:  04 <= RBCL
#: Read RFIFO
RDD:  Data Block  <= RFIFO
      01 7E 80 A2
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       Disable RME-Interrupt
WRB:  F7 => IMR0
#:
#:       Power Down
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Power Down
WRB:  10 => CCR0
#:
#:       End of Track File
```

The RFIFO contains the three received bytes (0x01, 0x7E, 0x80) plus a copy of the RSTA register (0xA2 = valid frame, CRC OK).

In Transparent Mode 0, the CRC bytes are stored in the RFIFO if CCR3.RCRC = 1:

Example

- Channel A (transparent mode 0), transmitting
- Channel B (transparent mode 0), receiving, CRC is stored in the RFIFO
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.6.2 Track File HTRPMD02

```
#: ************************************
#: File name : HTRPMD02.TRK
#: Hdlc TRansParent MoDe 0
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Channel A Initialization
#:       HDLC, FM0, transparent mode 0,
#:       Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  80 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       Channel B Initialization
#:       HDLC, FM0, transparent mode 0,
#:       DPLL, baud rate div. factor = 1,
#:       store CRC to RFIFO:
WRB:  10 => CCR0
WRB:  84 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  04 => CCR3
WRB:  90 => CCR0
#:
#:       Enable PLLA, RME-Interrupt,
#:    Init receiver:
WRB:  77 => IMR0
WRB:  8C => MODE
WRB:  40 => CMDR
```

```
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      01 7E 80
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
#:    Channel B interrupt
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
#:
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      01 7E 80 CC 72 A2
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Transmit another frame
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  55 => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      55 50 F5 A2
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:    End of Track File
```

Two frames have been received:

1st frame:	Data bytes:	0x01, 0x7E, 0x80
	CRC bytes:	0xCC, 0x72
	(Status:	0xA2, → valid frame, CRC OK)
2nd frame:	Data bytes:	0x55
	CRC bytes:	0x50, 0xF5 (same as in HATRBEX3.TRK)
	(Status:	0xA2, → valid frame, CRC OK)

The CRC generation in the transmitter may be switched off. In this case, the correct CRC value has to be transmitted from the XFIFO:

Example

- Channel A (transparent mode 0), transmitting without automatic CRC generation.
- Channel B (transparent mode 0), receiving, CRC is stored in the RFIFO.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.6.3 Track File HTRPMD03

```
#: ************************************
#: File name : HTRPMD03.TRK
#: Hdlc TRansParent MoDe 0
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Channel A Initialization
#:       HDLC, FM0, transparent mode 0,
#:       CRC generation off,
#:       Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  80 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  02 => CCR3
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       Channel B Initialization
#:       HDLC, FM0, transparent mode 0,
#:       DPLL, baud rate div. factor = 1:
WRB:  10 => CCR0
WRB:  84 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  00 => CCR3
WRB:  90 => CCR0
#:
WRB:  77 => IMR0
WRB:  8C => MODE
WRB:  40 => CMDR
#:
```

```
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Send data + correct CRC
WRD:  Data Block  => XFIFO
      55 50 F5
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      55 A2
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Send data + wrong CRC
WRD:  Data Block  => XFIFO
      55 51 F5
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
#:       Channel B interrupt
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      55 82
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

One frame was transmitted with the correct CRC value (0x55, 0x50, 0xF5), a second frame had the CRC modified (0x55, 0x**51**, 0xF5). The receiver status for the first frame is 0xA2 (valid frame, CRC OK), but is 0x82 (valid frame, CRC check **failed**) for the second frame.

4.6.4 Address Recognition

The ESCC performs high byte address recognition in transparent mode, if MODE.ADM is set (Transparent Mode 1). The receiver compares the first byte after the opening flag with the content of its RAH1 and RAH2 registers. Only if one of them matches, the frame will be stored into the RFIFO.

Example

- Channel A (transparent mode 1), transmitting frames with different addresses.
- Channel B (transparent mode 1), receiving, if address is recognized.
  (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.6.5 Track File HTRPMD04

```
#: ************************************
#: File name : HTRPMD04.TRK
#: Hdlc TRansParent MoDe 1,
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Channel A Initialization
#:       HDLC, FM0, transparent mode 1,
#:       Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  A0 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       Channel B Initialization
#:       HDLC, FM0, transparent mode 1,
#:       DPLL, baud rate div. factor = 1:
WRB:  10 => CCR0
WRB:  A4 => MODE
WRB:  30 => RAH1
WRB:  40 => RAH2
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  90 => CCR0
#:
WRB:  77 => IMR0
WRB:  AC => MODE
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
```

```
#:
#:       Send addr. 0x30 + data byte
WRD:  Data Block  => XFIFO
      30 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      55 A8
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Send addr. 0x40 + data byte
WRD:  Data Block  => XFIFO
      40 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
#:       Channel B interrupt
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      55 A0
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       Send unknown addr. + data byte
WRD:  Data Block  => XFIFO
      50 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
```

```
#:       No RME interrupt here !
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

The receiver recognized the frames with address byte 0x30 and 0x40 and loaded them into the RFIFO. It ignored the frame with address byte 0x50 (or any address different from RAH1 or RAH2). The status byte was 0xA8 for the first frame (valid frame, CRC OK, RAH1 recognized) and 0xA0 for the second (valid frame, CRC OK, RAH2 recognized).

The received address byte is stored into the RFIFO if enabled with CCR3.RADD = 1:

Example

- Channel A (transparent mode 1), transmitting frames with different addresses.
- Channel B (transparent mode 1), rcv., address recognition and stored in RFIFO. (The following pin connections on the User Connector Interface (UCI) have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.6.6 Track File HTRPMD05

```
#: ************************************
#: File name : HTRPMD05.TRK
#: Hdlc TRansParent MoDe 1,
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, transparent mode 1,
#:       7b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  A0 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, transparent mode 1,
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  A4 => MODE
WRB:  30 => RAH1
WRB:  40 => RAH2
WRB:  16 => CCR1
WRB:  10 => CCR2
#:       Store address in RFIFO
WRB:  10 => CCR3
WRB:  90 => CCR0
WRB:  77 => IMR0
WRB:  AC => MODE
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
```

```
      30 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      30 55 A8
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      40 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      40 55 A0
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

4.7 Non-Auto Mode

In Non-Auto Mode, the ESCC receiver performs 8- or 16-bit address recognition (MODE.ADM).

4.7.1 8-Bit Address Recognition in Non-Auto Mode

MODE.ADM = 0 selects the *8-bit address mode*. The address byte of the received frame is compared with the RAL1 and RAL2. The frame is only stored in the RFIFO if one of them matches.

Example

- Channel A (non-auto mode (8)), transmitting frames with different addresses.
- Channel B (non-auto mode (8)), receiving, if address is recognized.
  (The following UCI pin connections have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.7.2 Track File HNONAM01

```
#: ************************************
#: File name : HNONAM01.TRK
#: Hdlc NON-Auto-Mode (8),
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (8),
#:       7b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  40 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (8),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  44 => MODE
WRB:  30 => RAL1
WRB:  40 => RAL2
WRB:  16 => CCR1
WRB:  10 => CCR2
#:       Store address and CRC in RFIFO
WRB:  14 => CCR3
WRB:  90 => CCR0
WRB:  77 => IMR0
WRB:  4C => MODE
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
```

```
      30 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      30 55 CD BC A3
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      40 55
WRD:   2 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      40 55 09 4C A2
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:    Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

The receiver status byte for the first frame is 0xA3 (valid frame, CRC OK, RAL1 recognized). For the second frame, it is 0xA2 (valid frame, CRC OK, RAL2 recognized). In X.25 LAP-B applications RSTA.LA may be used as a command/response indication:

8-Bit Addr. Matches	RSTA.LA	C/R Type
RAL1	1	Command
RAL2	0	Response

4.7.3 16-Bit Address Recognition in Non-Auto Mode

MODE.ADM = 1 selects the *16-bit address mode*. The first of two address bytes of the received frame are compared with the RAH1 and RAH2 registers, and with the fixed broadcast values 0xFE and 0xFC. The second byte is compared with the RAL1 and RAL2 register. The frame is only stored in the RFIFO if both bytes are recognized.

Example

- Channel A (non-auto mode (16)), transmitting frames with different addresses.
- Channel B (non-auto mode (16)), receiving, if address is recognized.
  (The following UCI pin connections have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.7.4 Track File HNONAM02

```
#: ************************************
#: File name : HNONAM02.TRK
#: Hdlc NON-Auto-Mode (16),
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (16),
#:       7b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  60 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (16),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  64 => MODE
WRB:  30 => RAL1
WRB:  18 => RAH1
WRB:  40 => RAL2
WRB:  1C => RAH2
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  14 => CCR3
WRB:  90 => CCR0
WRB:  77 => IMR0
WRB:  6C => MODE
WRB:  40 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
```

```
WRD:  Data Block  => XFIFO
      18 30 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      18 30 55 11 36 A9
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      1C 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      1C 40 55 B4 A5 A0
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      18 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
```

```
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      18 40 55 D5 C6 A8
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      1C 30 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      1C 30 55 70 55 A1
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      FC 30 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      FC 30 55 D1 5C A5
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      FE 30 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
```

```
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      FE 30 55 69 E9 A7
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      FC 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      FC 40 55 15 AC A4
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      FE 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  06 <= RBCL
RDD:  Data Block  <= RFIFO
      FE 40 55 AD 19 A6
```

```
RDD:   6 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      FE 50 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
#:       No interrupts here !
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
WRD:  Data Block  => XFIFO
      18 50 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
#:       No interrupts here !
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

The receiver recognized frames with several combinations of high and low address bytes (see table below). The receiver status byte (copy of the RSTA register) indicates which address combination was recognized:

RAH1	RAH2	RAL1	RAL2	High Addr	Low Addr	RSTA	RSTA. HA1…0	RSTA. C/R	RSTA. LA
0x18	0x1C	0x30	0x40	0x18	0x30	0xA9	10	0	1
				0x18	0x40	0xA8	10	0	0
				0x1C	0x30	0xA1	00	0	1
				0x1C	0x40	0xA0	00	0	0
				0x1A	0x30	0xAB	10	1	1
				0x1A	0x40	0xAA	10	1	0
				0x1E	0x30	0xA3	00	1	1
				0x1E	0x40	0xA2	00	1	0
				0xFC	0x30	0xA5	01	0	1
				0xFC	0x40	0xA4	01	0	0
				0xFE	0x30	0xA7	01	1	1
				0xFE	0x40	0xA6	01	1	0
				0xFE	0x50	not recognized			

4.7.5 The Effect of RAH1.CRI in Non-Auto Mode

Example

- Channel A (non-auto mode (16)), transmitting command and response frames.
- Channel B (non-auto mode (16)), receiving with RAH1.CRI = 1.
  (The following UCI pin connections have to be made:

  TxDA (36) → (25) RxDB
  CTSA (33) ← (24) RTSB)

4.7.6 Track File HNONAM03

```
#: ************************************
#: File name : HNONAM03.TRK
#: Hdlc NON-Auto-Mode (16),
#: Channel A (7b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        CTSA (33) <- (24) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (16),
#:       7b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  60 => MODE
WRB:  17 => CCR1
WRB:  30 => CCR2
WRB:  07 => BRG
WRB:  90 => CCR0
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:    HDLC, FM0, non-auto mode (16),
#:    6b, DPLL, BR div. factor = 1,
#:    CRC not stored in RFIFO
#:    RAH1.CRI = 0
WRB:  10 => CCR0
WRB:  64 => MODE
WRB:  30 => RAL1
WRB:  18 => RAH1
WRB:  40 => RAL2
WRB:  1C => RAH2
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  90 => CCR0
WRB:  77 => IMR0
WRB:  6C => MODE
WRB:  40 => CMDR
```

```
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       RAH1, RAL1
WRD:  Data Block  => XFIFO
      18 30 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      18 30 55 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       RAH2, C/R, RAL2
WRD:  Data Block  => XFIFO
      1E 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      1E 40 55 A2
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    RAH1.CRI = 1, Rcv. reset
WRB:  1A => RAH1
WRB:  40 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       RAH1, RAL1
WRD:  Data Block  => XFIFO
      18 30 55
```

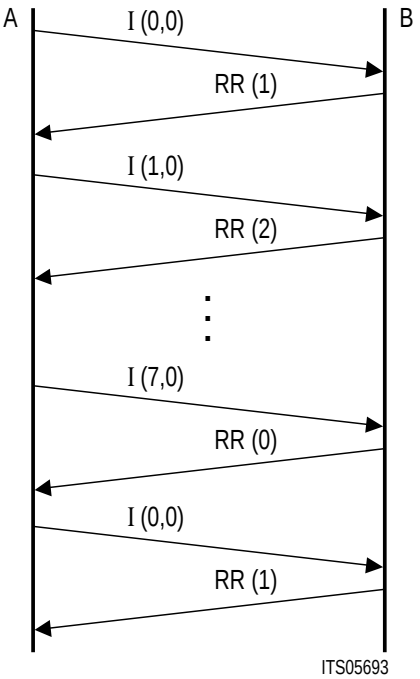
```
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      18 30 55 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:
#:       RAH2, C/R, RAL2
WRD:  Data Block  => XFIFO
      1E 40 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      1E 40 55 A2
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

RAH1	CRI	RAH2	RAL1	RAL2	High Addr	Low Addr	RSTA	RSTA. HA1…0	RSTA. C/R	RSTA. LA
0x18	0	0x1C	0x30	0x40	0x18	0x30	0xA9	10	0	1
0x18	0	0x1C	0x30	0x40	0x1E	0x30	0xA2	00	1	0
0x1A	1	0x1C	0x30	0x40	0x18	0x30	0xA9	10	0	1
0x1A	1	0x1C	0x30	0x40	0x1E	0x30	0xA2	00	1	0

The C/R bit in the RSTA register reflects the state of bit 1 in the high address byte of the received frame. RAH1.CRI has no effect in Non-Auto Mode!

4.8 Auto Mode

In Auto Mode, the ESCC receiver performs 8- or 16- bit address recognition like in Non-Auto Mode. The transmitter can either send transparent frames (CMDR.XTF) or I frames (CMDR.XIF). The ESCC autonomously processes I-, RR-, and RNR frames of the HDLC protocol.



4.8.1 HDLC Protocol Example

Station *A* transmits eight I frames to station *B*. *B* responds to each one with an RR frame. Station *A* is the ESCC2 channel A in **Non-Auto Mode**, B is channel B in **Auto Mode**.

Command and response addresses:

Commands from A to B	0x10
Responses from A to B	0x11
Commands from B to A	0x11
Responses from B to A	0x10

Since channel A is in Non-Auto Mode the I frame has to be assembled in the XFIFO and has to be sent as a transparent frame:

HDLC I-frame structure (8-bit address field)

	flag	ADDR	CTRL	INFO Field	CRC		flag
	(7E)						(7E)
Source:	auto	XTF: XFIFO			auto		auto

The opening flag is generated automatically.

I frames have the address of a command (in this example: 0x10).

Control field bit 0 is 0 (I-frame identification)

Control field bits 3…1 contain the *send sequence number* N(S). It starts at zero and is incremented with every new frame that is sent.

Control field bit 4 is the *poll bit*. The poll bit is zero in this example.

Control field bits 7…5 contain the *receive sequence number* N(R). In this example no I frames are received from station B, therefore, N(R) remains at zero.

The info field carries user data with flexible length.

The CRC field is generated automatically.

The closing flag is generated automatically.

Channel B is in Auto Mode. It will receive the I frames and automatically acknowledge with RR frames. The structure of an RR frame is:

HDLC RR-frame structure (8-bit address field):

	flag	ADDR	CTRL	CRC		flag
	(7E)					(7E)
Source:	auto	XAD2	auto	auto		auto

The RR frames are found in the RFIFO of channel A (Non-Auto Mode):

The opening flag is stripped of by the receiver.

Channel A is receiving RR frames from B as a response:
The address is 0x10 (Channel B's XAD2).

Control field bits 3…0 are "0001" (RR-frame identification).

Control field bit 4 is the *poll bit*. The poll bit is zero in this example.

Control field bits 7…5 contain the *receive sequence number* N(R). N(R) shows the sequence number if the next expected I frame.

The CRC can optionally be stored in the RFIFO (CCR3.RCRC).

The closing flag is stripped of by the receiver.

Example

- Channel A (non-auto mode (8)), xmit I frames, rcv. RR frames.
- Channel B (auto mode (8)), rcv. I frames, auto-response (RR frame).
  (The following UCI pin connections have to be made:

  TxDA (36) → (25) RxDB
  RxDA (35) ← (26) TxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.8.2 Track File HAUTOM01

```
#: ************************************
#: File name : HAUTOM01.TRK
#: Hdlc non-auto -> AUTO-Mode (8),
#: Channel A (6b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, non-auto mode (8),
#:       6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  44 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  14 => CCR3
WRB:  11 => RAL1
WRB:  10 => RAL2
WRB:  90 => CCR0
WRB:  4C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, auto mode (8),
#:    6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  25 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
#: Command/response addresses
WRB:  10 => RAL1
WRB:  11 => RAL2
```

```
WRB:  00 => RAH1
WRB:  11 => XAD1
WRB:  10 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 00 12
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block<= RFIFO
      10 00 12 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      10 21 5D AA A0
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 02 34
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
```

```
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 02 34 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      10 41 5B C9 A2
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 04 56
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 04 56 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      10 61 59 E8 A2
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
```

```
WRD:  Data Block  => XFIFO
      10 06 78
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 06 78 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      10 81 57 0F A2
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:    switch off ADDR/CRC storage
WRB:  00 => CCR3
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 08 9A
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block< = RFIFO
      10 08 9A A1
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:       switch off ADDR storage
WRB:  00 => CCR3
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
```

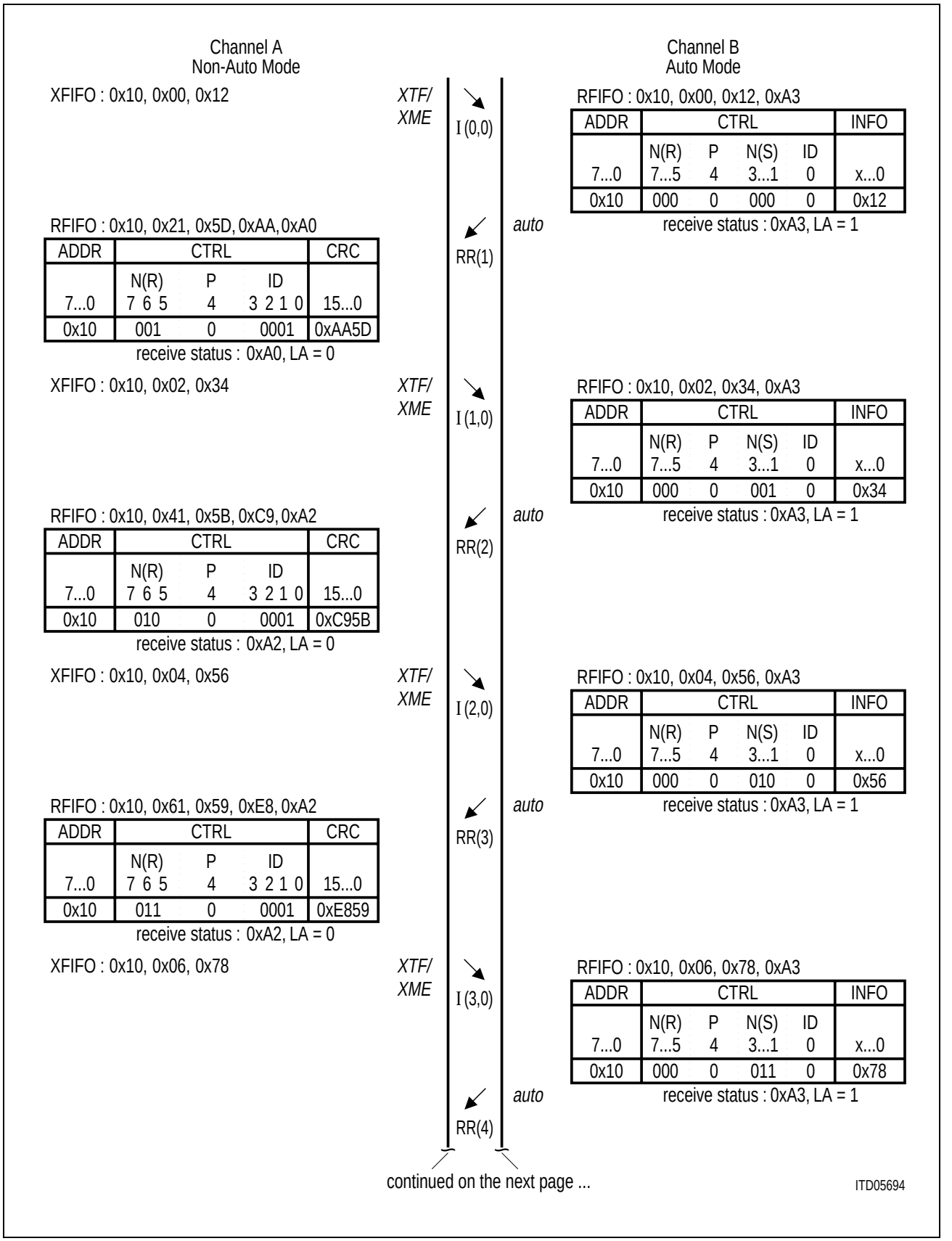
```
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      A1 A0
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 0A BC
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      0A BC A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      C1 A2
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 0C DE
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
```

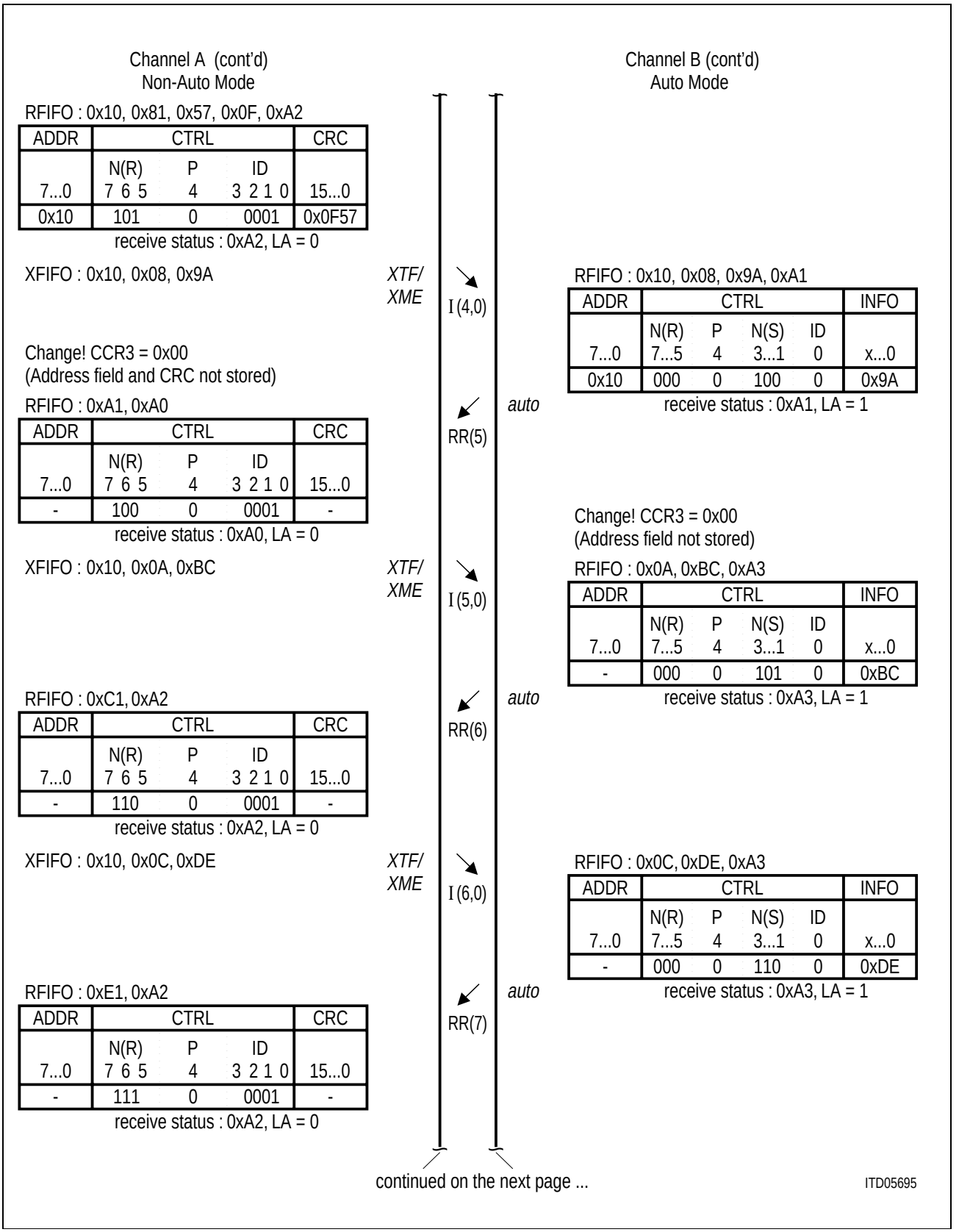
```
      0C DE A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      E1 A2
RDD:   2 Bytes<= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 0E F0
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      0E F0 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      01 A2
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    New I-frame
WRD:  Data Block  => XFIFO
      10 00 55
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
```

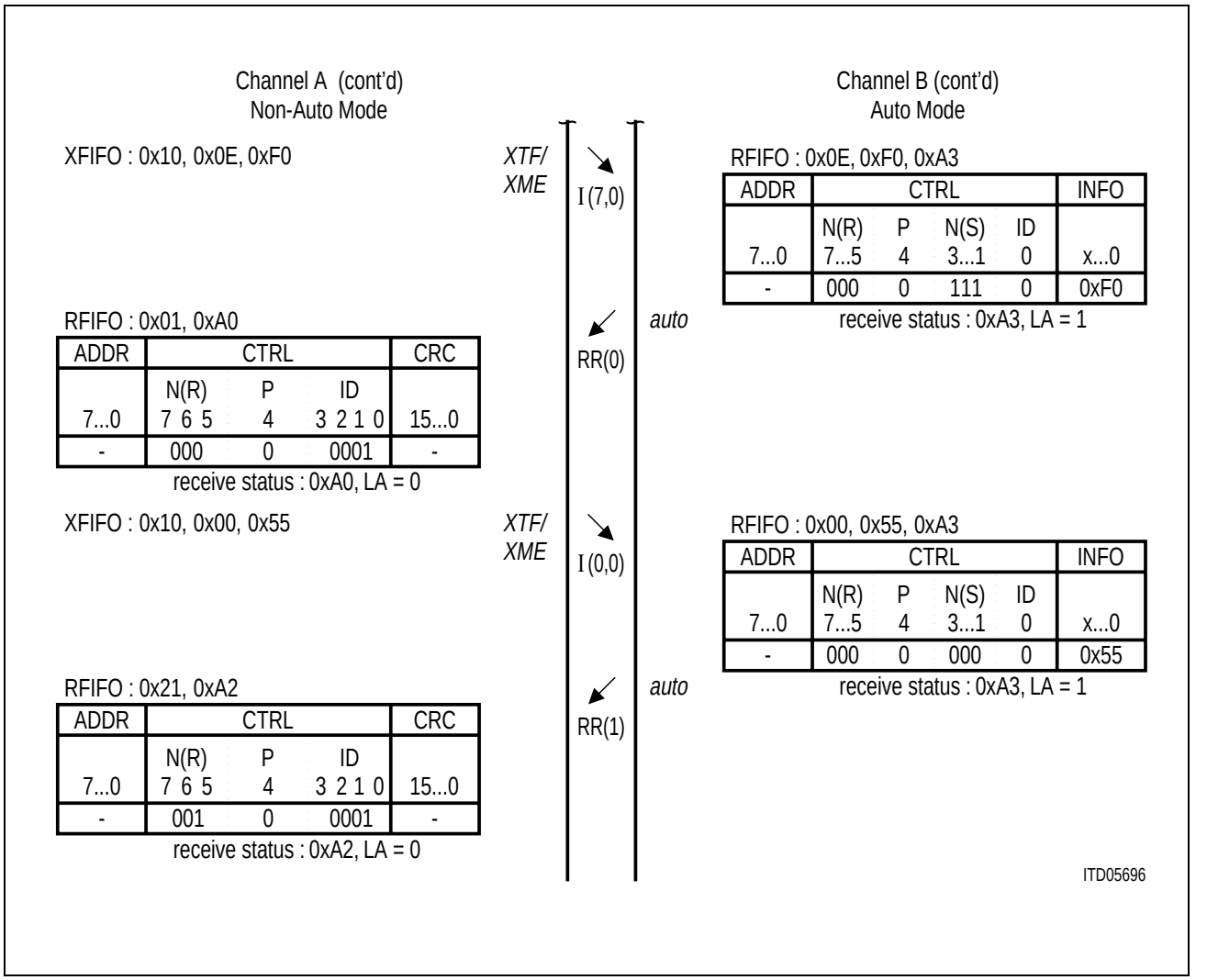
```
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      00 55 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RBCL
RDD:  Data Block  <= RFIFO
      21 A2
RDD:   2 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Disable Interrupts, power down
WRB:  FF => IMR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:       End of Track File
```

When *B* receives an I frame from *A*, it generates an RME interrupt and sends an RR frame as a response. *A* is in Non-Auto Mode and therefore does not interpret the RR frame. Instead it is moved into the RFIFO and an RME interrupt is generated.

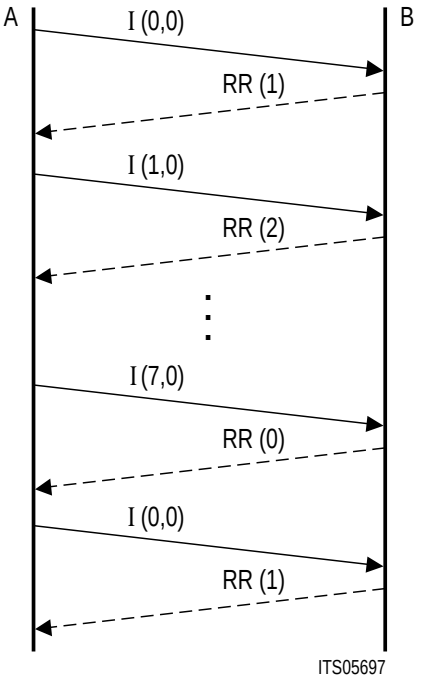
4.8.3 Communication Chart with Frame Structures




Channel A (cont'd)
Non-Auto Mode
RFIFO : 0x10, 0x81, 0x57, 0x0F, 0xA2
ADDR | CTRL (N(R) 7 6 5 | P 4 | ID 3 2 1 0) | CRC
7...0 | | | | 15...0
0x10 | 101 | 0 | 0001 | 0x0F57
receive status : 0xA2, LA = 0
XFIFO : 0x10, 0x08, 0x9A
XTF/ XME
I (4,0)
Channel B (cont'd)
Auto Mode
RFIFO : 0x10, 0x08, 0x9A, 0xA1
ADDR | CTRL (N(R) 7...5 | P 4 | N(S) 3...1 | ID 0) | INFO
7...0 | | | | | x...0
0x10 | 000 | 0 | 100 | 0 | 0x9A
receive status : 0xA1, LA = 1
Change! CCR3 = 0x00
(Address field and CRC not stored)
RFIFO : 0xA1, 0xA0
ADDR | CTRL (N(R) 7 6 5 | P 4 | ID 3 2 1 0) | CRC
7...0 | | | | 15...0
- | 100 | 0 | 0001 | -
receive status : 0xA0, LA = 0
auto
RR(5)
XFIFO : 0x10, 0x0A, 0xBC
XTF/ XME
I (5,0)
Change! CCR3 = 0x00
(Address field not stored)
RFIFO : 0x0A, 0xBC, 0xA3
ADDR | CTRL (N(R) 7...5 | P 4 | N(S) 3...1 | ID 0) | INFO
7...0 | | | | | x...0
- | 000 | 0 | 101 | 0 | 0xBC
receive status : 0xA3, LA = 1
RFIFO : 0xC1, 0xA2
ADDR | CTRL (N(R) 7 6 5 | P 4 | ID 3 2 1 0) | CRC
7...0 | | | | 15...0
- | 110 | 0 | 0001 | -
receive status : 0xA2, LA = 0
auto
RR(6)
XFIFO : 0x10, 0x0C, 0xDE
XTF/ XME
I (6,0)
RFIFO : 0x0C, 0xDE, 0xA3
ADDR | CTRL (N(R) 7...5 | P 4 | N(S) 3...1 | ID 0) | INFO
7...0 | | | | | x...0
- | 000 | 0 | 110 | 0 | 0xDE
receive status : 0xA3, LA = 1
RFIFO : 0xE1, 0xA2
ADDR | CTRL (N(R) 7 6 5 | P 4 | ID 3 2 1 0) | CRC
7...0 | | | | 15...0
- | 111 | 0 | 0001 | -
receive status : 0xA2, LA = 0
auto
RR(7)
continued on the next page ...
ITD05695


Channel A (cont'd)
Non-Auto Mode
Channel B (cont'd)
Auto Mode
XFIFO : 0x10, 0x0E, 0xF0
XTF/ XME
I (7,0)
RFIFO : 0x0E, 0xF0, 0xA3
ADDR | CTRL | INFO
7...0 | N(R) 7...5 | P 4 | N(S) 3...1 | ID 0 | x...0
- | 000 | 0 | 111 | 0 | 0xF0
receive status : 0xA3, LA = 1
RFIFO : 0x01, 0xA0
ADDR | CTRL | CRC
7...0 | N(R) 7 6 5 | P 4 | ID 3 2 1 0 | 15...0
- | 000 | 0 | 0001 | -
receive status : 0xA0, LA = 0
auto
RR(0)
XFIFO : 0x10, 0x00, 0x55
XTF/ XME
I (0,0)
RFIFO : 0x00, 0x55, 0xA3
ADDR | CTRL | INFO
7...0 | N(R) 7...5 | P 4 | N(S) 3...1 | ID 0 | x...0
- | 000 | 0 | 000 | 0 | 0x55
receive status : 0xA3, LA = 1
RFIFO : 0x21, 0xA2
ADDR | CTRL | CRC
7...0 | N(R) 7 6 5 | P 4 | ID 3 2 1 0 | 15...0
- | 001 | 0 | 0001 | -
receive status : 0xA2, LA = 0
auto
RR(1)
ITD05696

If both channels operate in Auto Mode, the RR frames are generated and interpreted autonomously. Only the I frames will be found in the RFIFO:



Example

- Channel A (auto mode (8)),
  transmitting I frames (XIF!).
- Channel B (auto mode (8)),
  auto-responding (RR frames).
  (The following UCI pin connections have to be made:

  TxDA (36) → (25) RxDB
  RxDA (35) ← (26) TxDB
  CTSA (33) ← (24) RTSB
  RTSA (34) → (23) CTSB)

4.8.4 Track File HAUTOM02

```
#: ************************************
#: File name : HAUTOM02.TRK
#: Hdlc AUTO-Mode -> auto-mode (8),
#: Channel A (6b) -> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, auto-mode (8),
#:       6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  25 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  11 => RAL1
WRB:  10 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  10 => XAD1
WRB:  11 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, auto mode (8),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  25 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
```

```
WRB:  10 => RAL1
WRB:  11 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  11 => XAD1
WRB:  10 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#:
#:    Send I-frame (XIF)
WRB:  12 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      00 12 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  34 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
```

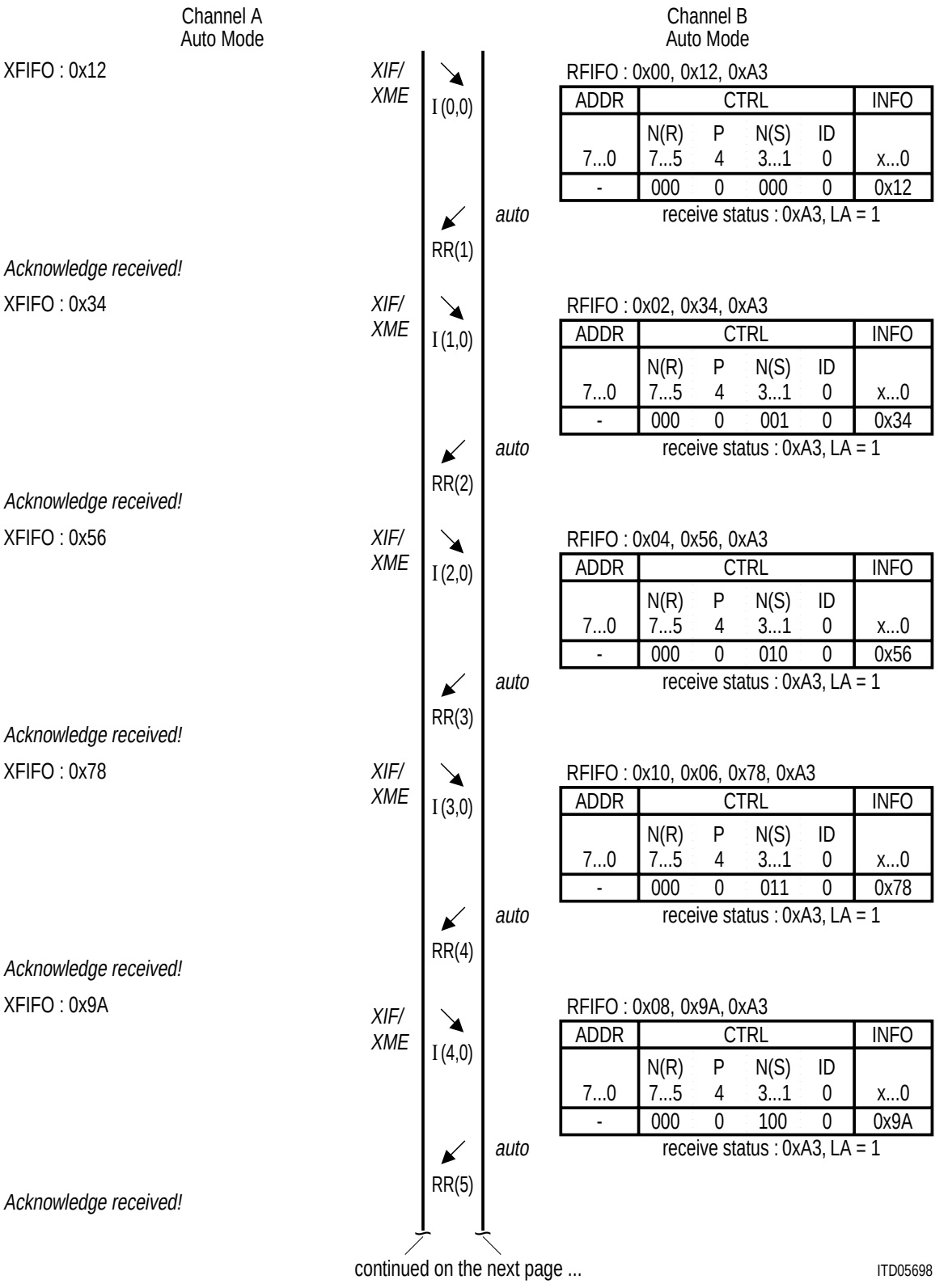
```
      02 34 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  56 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      04 56 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  78 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      06 78 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  9A => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
```

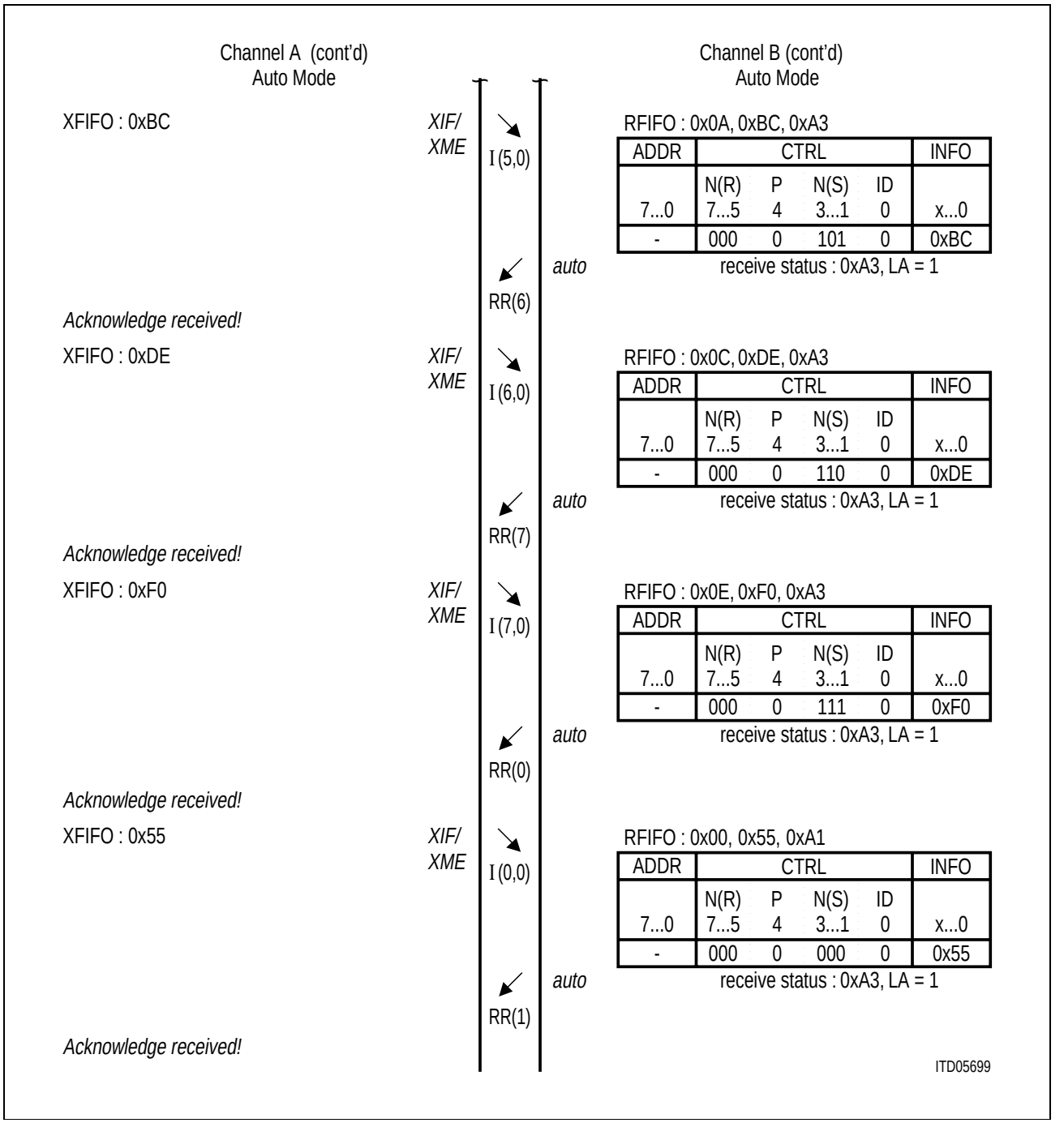
```
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      08 9A A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  BC => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      0A BC A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  DE => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      0C DE A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  F0 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
```

```
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      0E F0 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame (XIF)
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  55 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      00 55 A1
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Disable Interrupts, power down
WRB:  FF => IMR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  FF => IMR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

If the XIF/XME command is used, channel A autonomously generates the complete frame with address field, control field and CRC around the XFIFO content. These I frames are found in channel B's RFIFO. They are identical with the “transparent I frames” from the previous example. The RR frames are no longer stored in the RFIFO.

4.8.5 Communication Chart with Frame Structures




Channel A (cont'd)
Auto Mode
Channel B (cont'd)
Auto Mode
XFIFO : 0xBC
XIF/ XME
I (5,0)
RFIFO : 0x0A, 0xBC, 0xA3
ADDR
CTRL
INFO
N(R) P N(S) ID
7...0 7...5 4 3...1 0 x...0
- 000 0 101 0 0xBC
auto
receive status : 0xA3, LA = 1
RR(6)
Acknowledge received!
XFIFO : 0xDE
XIF/ XME
I (6,0)
RFIFO : 0x0C, 0xDE, 0xA3
ADDR
CTRL
INFO
N(R) P N(S) ID
7...0 7...5 4 3...1 0 x...0
- 000 0 110 0 0xDE
auto
receive status : 0xA3, LA = 1
RR(7)
Acknowledge received!
XFIFO : 0xF0
XIF/ XME
I (7,0)
RFIFO : 0x0E, 0xF0, 0xA3
ADDR
CTRL
INFO
N(R) P N(S) ID
7...0 7...5 4 3...1 0 x...0
- 000 0 111 0 0xF0
auto
receive status : 0xA3, LA = 1
RR(0)
Acknowledge received!
XFIFO : 0x55
XIF/ XME
I (0,0)
RFIFO : 0x00, 0x55, 0xA1
ADDR
CTRL
INFO
N(R) P N(S) ID
7...0 7...5 4 3...1 0 x...0
- 000 0 000 0 0x55
auto
receive status : 0xA3, LA = 1
RR(1)
Acknowledge received!
ITD05699

When I frames are transferred in both directions between A and B, the N(R) and N(S) sequence numbers are incremented:

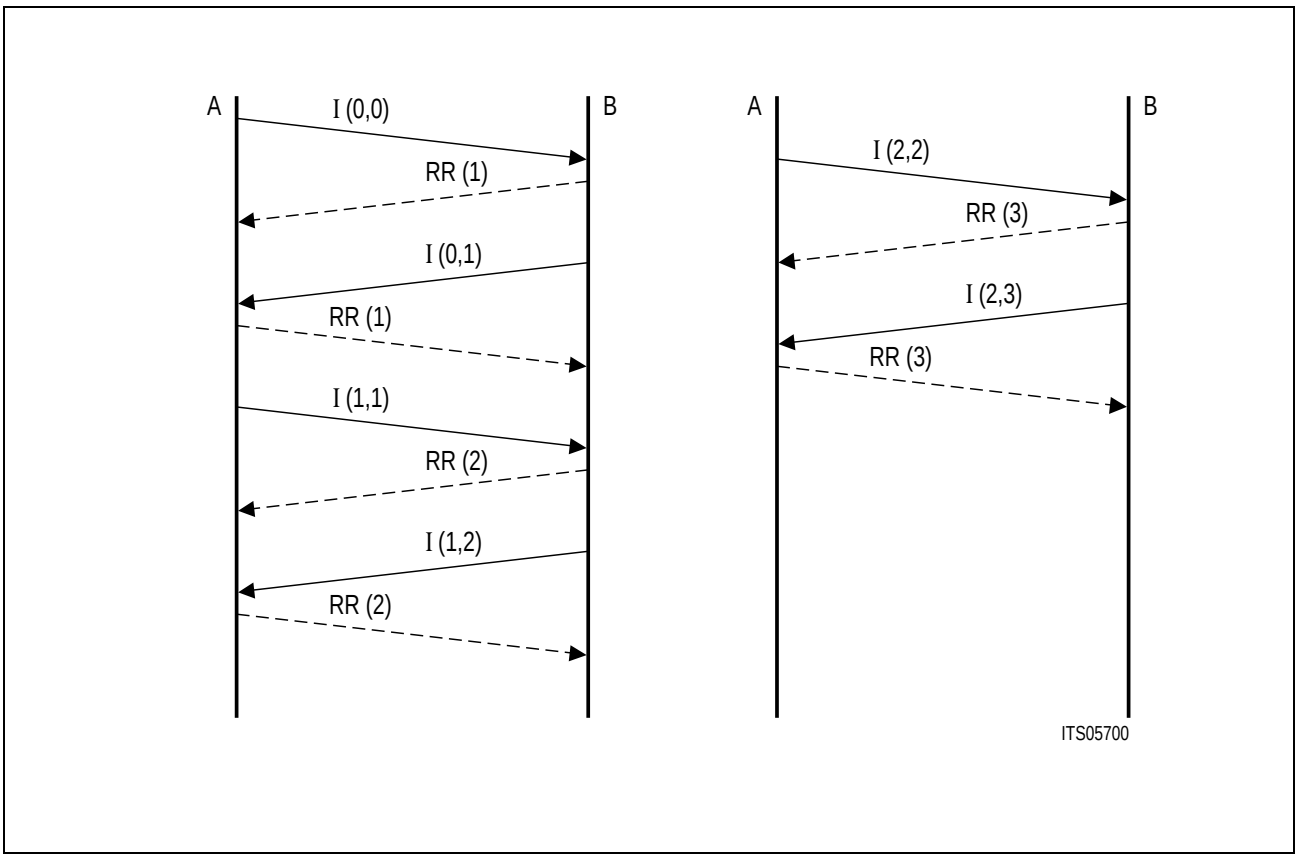


Figure 1
I Frames in Both Directions (and RR), between A and B

Example

- Channel A (auto mode (8)), xmit & rcv. I frames (auto-response).
- Channel B (auto mode (8)), xmit & rcv. I frames (auto-response).
  (The following UCI pin connections have to be made:

TxDA (36)	→	(25) RxDB
RxDA (35)	←	(26) TxDB
CTSA (33)	←	(24) RTSB
RTSA (34)	→	(23) CTSB)

4.8.6 Track File HAUTOM03

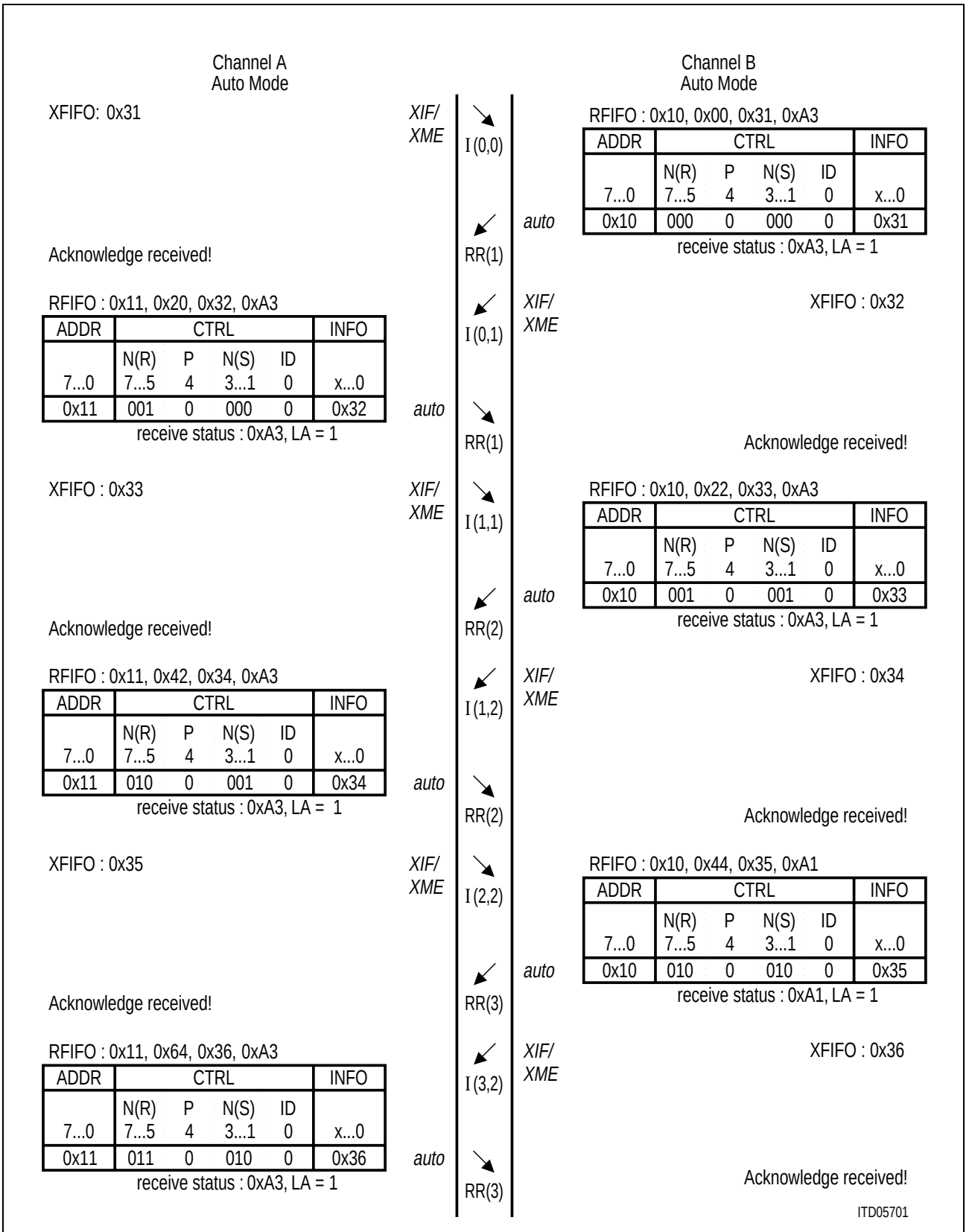
```
#: ************************************
#: File name : HAUTOM03.TRK
#: Hdlc AUTO-Mode <-> auto-mode (8),
#: Channel A (6b) <-> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) RTSB
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, auto-mode (8),
#:       6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  25 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  11 => RAL1
WRB:  10 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  10 => XAD1
WRB:  11 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, auto mode (8),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  25 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
```

```
WRB:  10 => CCR3
WRB:  10 => RAL1
WRB:  11 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  11 => XAD1
WRB:  10 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#:
#:    Send I-frame A -> B
WRB:  31 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 00 31 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame B -> A
WRB:  32 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
```

```
      11 20 32 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame A -> B
WRB:  33 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 22 33 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame B -> A
WRB:  34 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      11 42 34 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame A -> B
WRB:  35 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
```

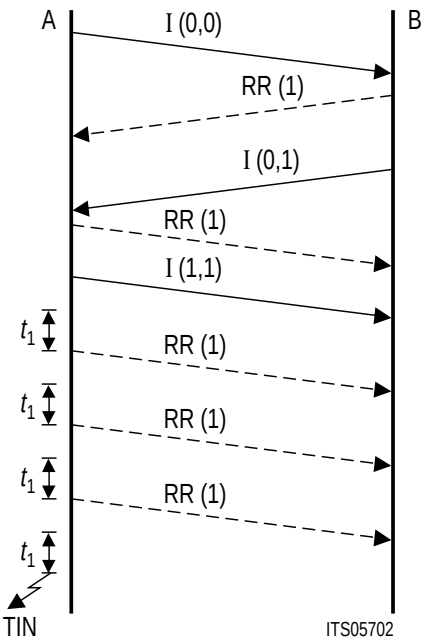
```
RDD:  Data Block  <= RFIFO
      10 44 35 A1
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame B -> A
WRB:  36 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      11 64 36 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Disable Interrupts, power down
WRB:  FF => IMR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:       End of Track File
```

4.8.7 Communication Chart with Frame Structures



4.9 Polling

If one station does not acknowledge an I frame, it is polled by the originator of the I frame. The time-out value and the retry-count is programmable in the TIMR register. When the retry-count is exceeded, the originator of the I frame generates a TIN-interrupt:



Example

- Channel A (auto mode (8)), xmit & rcv. I frames (auto-response).
- Channel B (auto mode (8)), xmit & rcv. I frames (auto-response).
  (The following UCI pin connections have to be made:

TxDA (36)	→	(25) RxDB
RxDA (35)	←	(26) TxDB
CTSA (33)	←	(24) RTSB
RTSA (34)	→	(23) CTSB)

4.9.1 Track File HAUTOM04

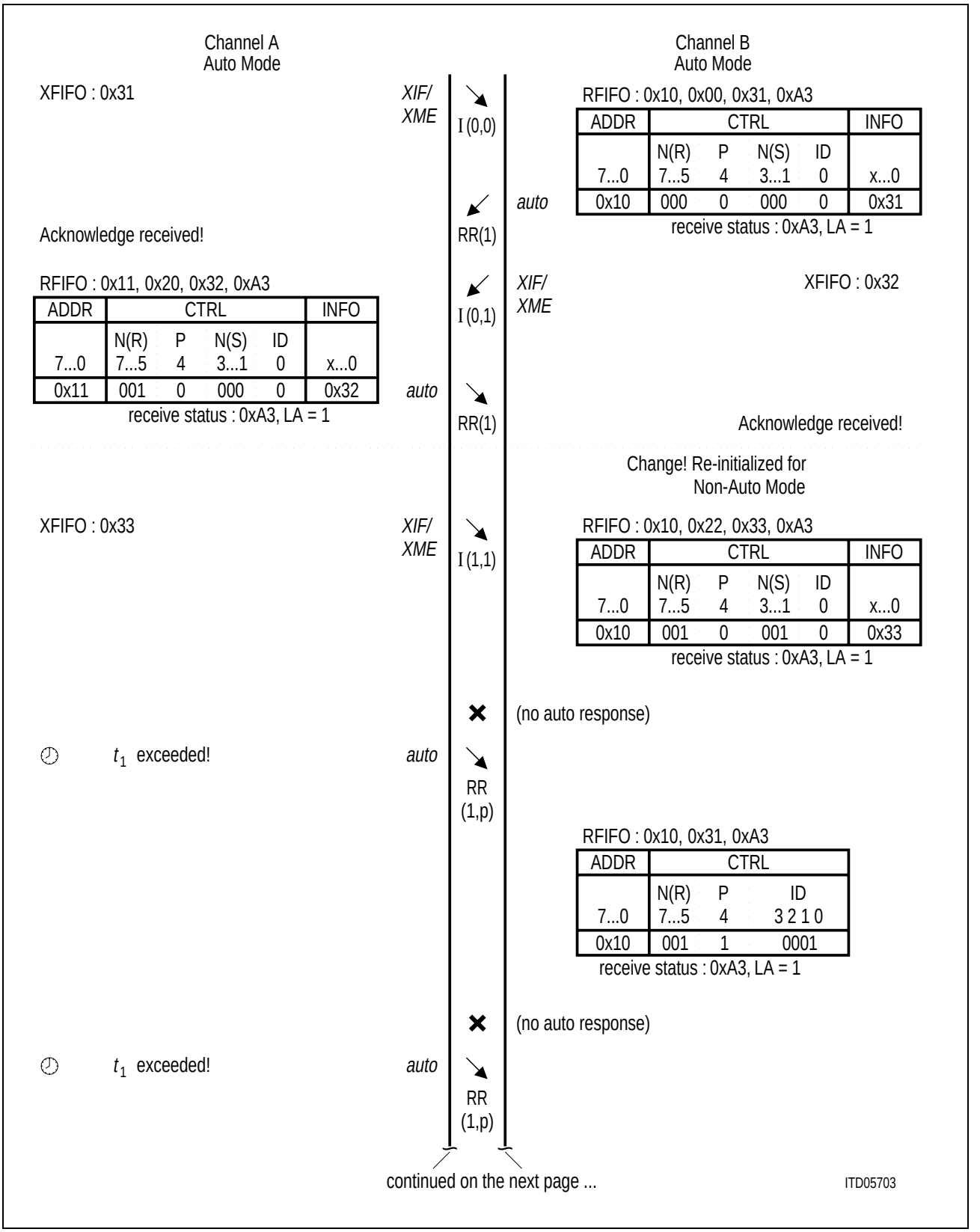
```
#: ************************************
#: File name : HAUTOM04.TRK
#: Hdlc AUTO-Mode <-> auto-mode (8),
#: Error case !
#: Channel A (6b) <-> B (6b)
#:
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) RTSB
#:
#: ************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, auto-mode (8),
#:       6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  14 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  11 => RAL1
WRB:  10 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  10 => XAD1
WRB:  11 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, auto mode (8),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  14 => MODE
```

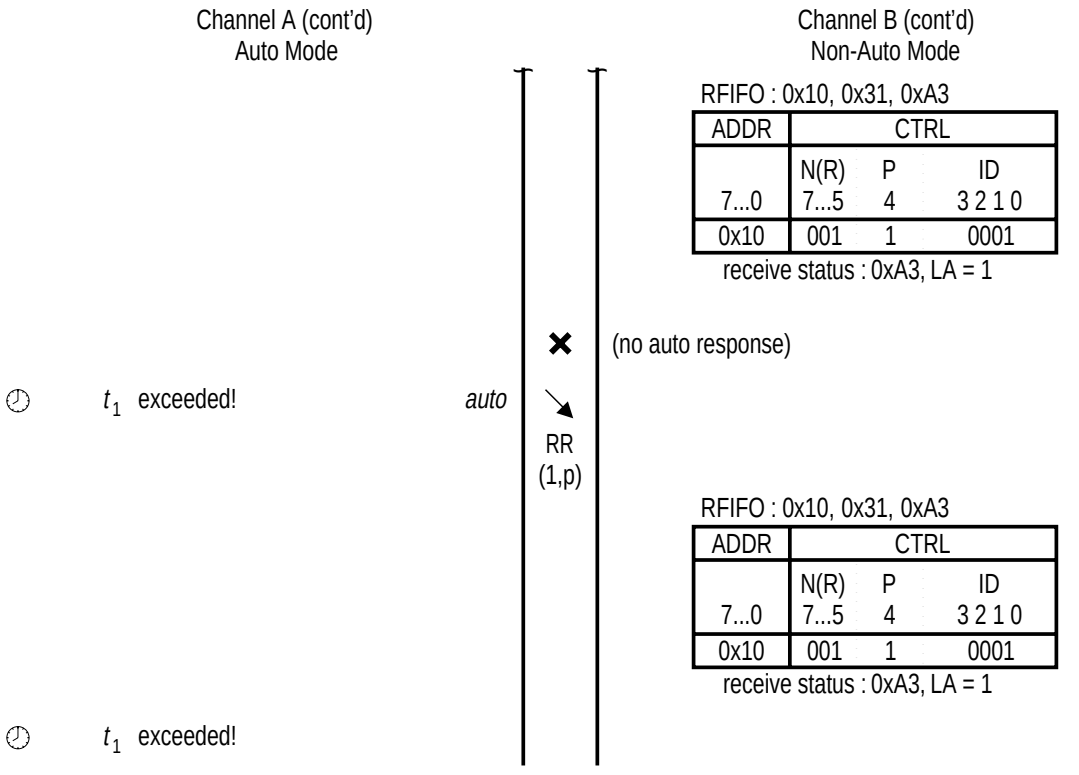
```
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  10 => RAL1
WRB:  11 => RAL2
WRB:  00 => RAH1
WRB:  00 => RAH2
WRB:  11 => XAD1
WRB:  10 => XAD2
WRB:  90 => CCR0
WRB:  1C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#     Enable TIN interrupt
WRB:  F7 => IMR1
#:
#:    Send I-frame A -> B
WRB:  31 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 00 31 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:    Send I-frame B -> A
WRB:  32 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
```

```
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      11 20 32 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     Re-initialize channel B for
#     NON-auto mode (8),
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  4C => MODE
WRB:  41 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
#:    Send I-frame A -> B
WRB:  33 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  09 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      10 22 33 A3
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  09 <= GIS
#     another ch.B interrupt !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      10 31 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  09 <= GIS
#     another ch.B interrupt !
RDB:  80 <= ISR0
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      10 31 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  09 <= GIS
#     another ch.B interrupt !
RDB:  80 <= ISR0
```

```
RDB:  03 <= RBCL
RDD:  Data Block  <= RFIFO
      10 31 A3
RDD:   3 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  08 <= GIS
#     Channel A interrupt pending
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  08 <= ISR1
INT:  Interrupt(s) acknowledged
#     TIN interrupt
#     -> link down !
#     Disable Interrupts, power down
WRB:  FF => IMR0
WRB:  FF => IMR1
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#     End of Track File
```

4.9.2 Communication Chart with Frame Structures





Note : The poll frames (RR-frames) are sent as commands (Addr. 0x10).
The poll-bit is set.

ITD05704

4.9.3 Two-Byte Address Mode

HDLC frames with a 2-byte address field are used in LAP-D applications.

The high-address byte is composed of the *service access point identifier* (SAPI), the *command/response bit* (C/R) and the *extended address bit* (EA).

Hi-Address Byte	SAPI	C/R	EA=0

- The SAPI for user data on a D-channel is typically 0x01.
- The C/R bit differentiates between commands and responses:
  Commands from the network side to the user side: C/R bit = 1.
  Responses from the user side to the network side: C/R bit = 1.
  Commands from the user side to the network side: C/R bit = 0.
  Responses from the network side to the user side: C/R bit = 0.
- EA = 0 indicates that another address byte is following.

The low-address byte is composed of the *terminal endpoint identifier* (TEI) and the EA-bit.

Lo-Address Byte	TEI	EA=1

- EA = 1 indicates the end of the address field.
- TEI can have any value between 0 and 0x7F. 0x7F is reserved for broadcast frames.

MODE.ADM = 1 selects the 2-byte address recognition mode of the ESCC. Auto-processing of I frames is performed only if SAPI of the received I frame matches RAH1 (bits 7…2 of the RAH1 register) and TEI&EA matches the RAL1 register content. Frames that match RAH2/RAL2 are received into the RFIFO but not automatically acknowledged. The ESCC also handles the C/R bit autonomously. For proper operation, RAH1.CRI has to be set in accordance to the following table:

ESCC on the user side:	RAH1.CRI = 0
ESCC on the network side:	RAH1.CRI = 1

The following example operates channel A in Non-Auto Mode and channel B in Auto Mode. Channel B is assumed to be the network side (RAH1.CRI = 1). Since channel A is in non-auto mode, the RR frames from channel B are found in A's RFIFO for further analysis:

Example

- Channel A (non-auto md. (16)), xmit I frames (XTF/XME), rcv. RR frames.
- Channel B (auto mode (16)), CRI = 1, rcv. I frames, auto-generate RR frames.
  (The following UCI pin connections have to be made:

TxDA (36)	→	(25) RxDB
RxDA (35)	←	(26) TxDB
CTSA (33)	←	(24) RTSB
RTSA (34)	→	(23) CTSB)

4.9.4 Track File HAUTOM05

```
#**************************************
#     File name : HAUTOM05.TRK
#     Hdlc non-auto -> AUTO-Mode (16),
#     Channel A (6b) -> B (6b, CRI=1)
#
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#
#**************************************
#
NDC:  ESCC2.Channel A.HDLC/SDLC
#     HDLC, FM0, non-auto mode (16),
#     6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  64 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
#:    16-bit addresses
WRB:  04 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAL2
WRB:  00 => RAH2
WRB:  90 => CCR0
WRB:  6C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#     HDLC, FM0, auto mode (16),
#     6b, DPLL, BR div.factor = 1:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  34 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
```

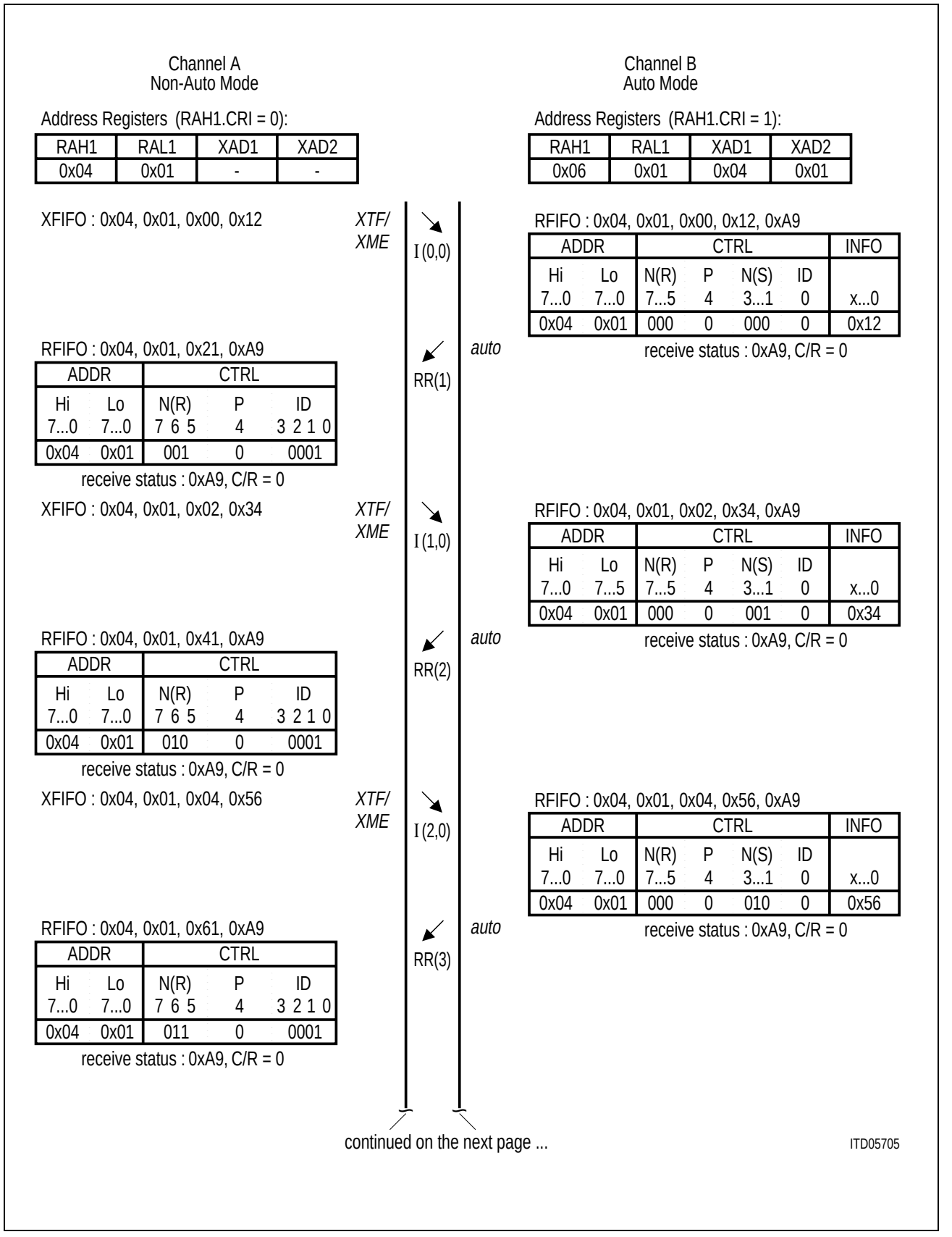
```
#:    16-bit addresses
#:    CRI=1 !
WRB:  06 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAH2
WRB:  00 => RAL2
WRB:  04 => XAD1
WRB:  01 => XAD2
WRB:  90 => CCR0
WRB:  3C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
WRB:  F7 => IMR1
#
NDC:  ESCC2.Channel A.HDLC/SDLC
#     Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 00 12
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 00 12 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 21 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
```

```
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 02 34
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 02 34 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 41 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 04 56
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 04 56 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
```

```
RDD:  Data Block  <= RFIFO
      04 01 61 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  AA => XFIFO
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  06 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 60 AA AB
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  06 <= GIS
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 71 AB
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  06 <= GIS
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 71 AB
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  06 <= GIS
RDB:  80 <= ISR0
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 71 AB
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
RDB:  02 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  08 <= ISR1
INT:  Interrupt(s) acknowledged
WRB:  FF => IMR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  FF => IMR0
```

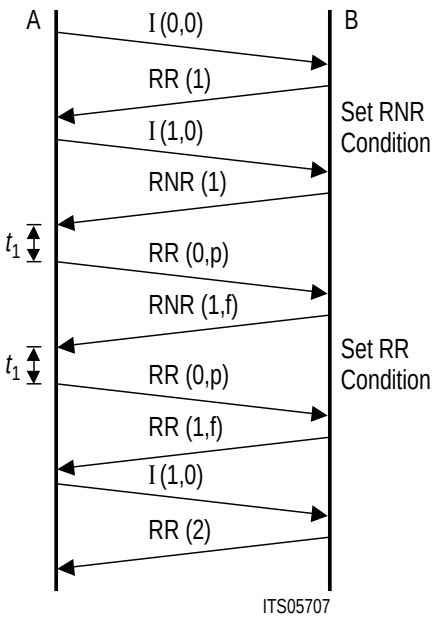
```
WRB:  10 => CCR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

4.9.5 Communication Chart with Frame Structures



4.10 RNR Flow Control

In Auto Mode, the ESCC can be switched into the *Not Ready*-Mode with CMDR.RNR. Instead of sending an RR frame, the ESCC will then respond with an RNR frame and will **not** store the I frame in the RFIFO.



Example

- Channel A (non-auto mode (16)), xmit I frames (XTF/XME), rcv. RR frames.
- Channel B (auto mode (16)), CRI = 1, rcv. I frames, auto-gen. RR and RNR frames. (The following UCI pin connections have to be made:

TxDA (36)	→	(25) RxDB
RxDA (35)	←	(26) TxDB
CTSA (33)	←	(24) RTSB
RTSA (34)	→	(23) CTSB)

4.10.1 Track File HAUTOM06

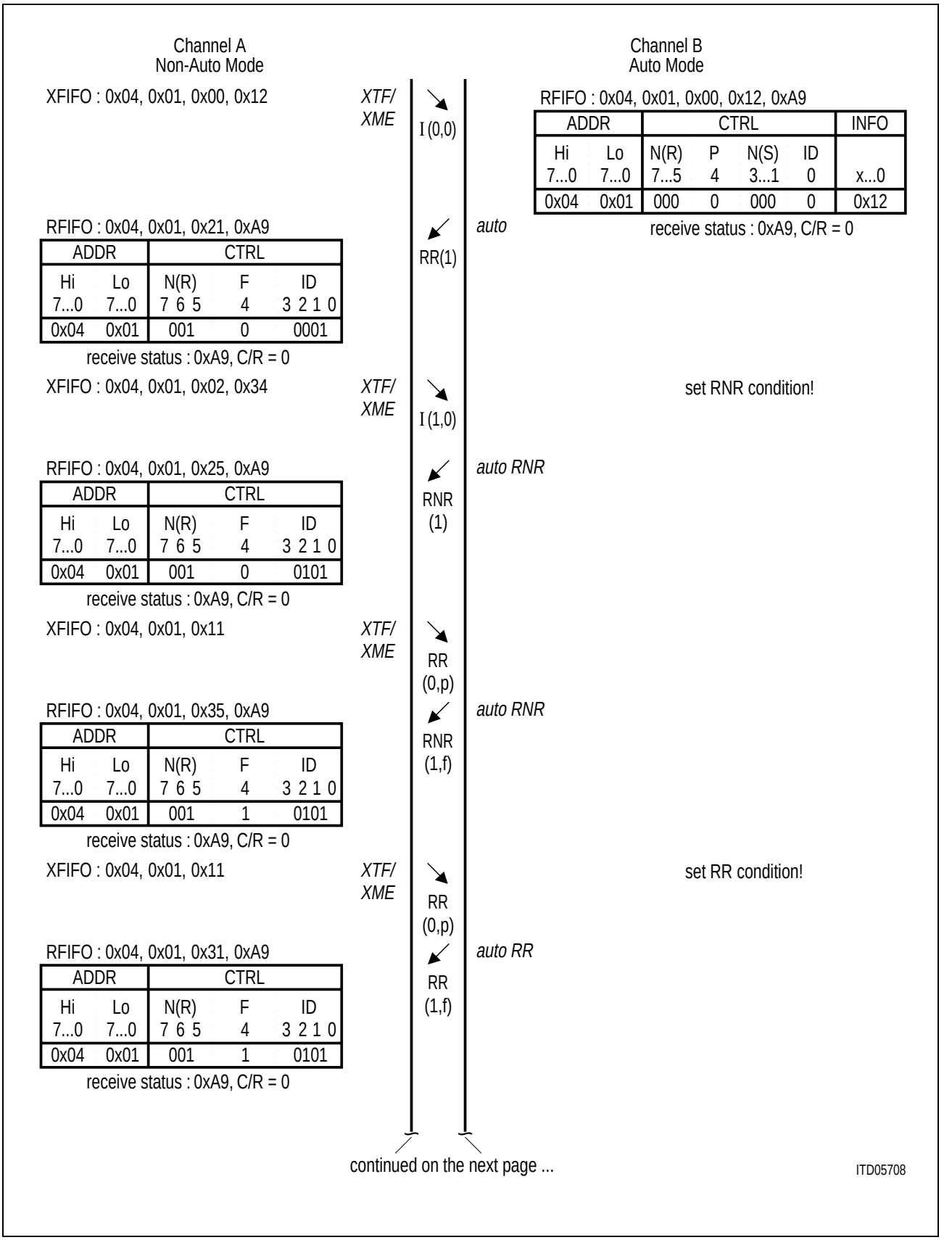
```
#*************************************
#        File name : HAUTOM06.TRK
#        Hdlc non-auto -> AUTO-Mode (16),
#        Channel A (6b) -> B (6b, CRI=1)
#        RNR Flow Control
#
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
#        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#
#*************************************
#
NDC:  ESCC2.Channel A.HDLC/SDLC
#        HDLC, FM0, non-auto mode (16),
#        6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  64 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  04 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAL2
WRB:  00 => RAH2
WRB:  90 => CCR0
WRB:  6C => MODE
WRB:  41 => CMDR
#:
NDC:  ESCC2.Channel B.HDLC/SDLC
#     HDLC, FM0, auto mode (16),
#     6b, DPLL, BR div.factor = 1:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  34 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
```

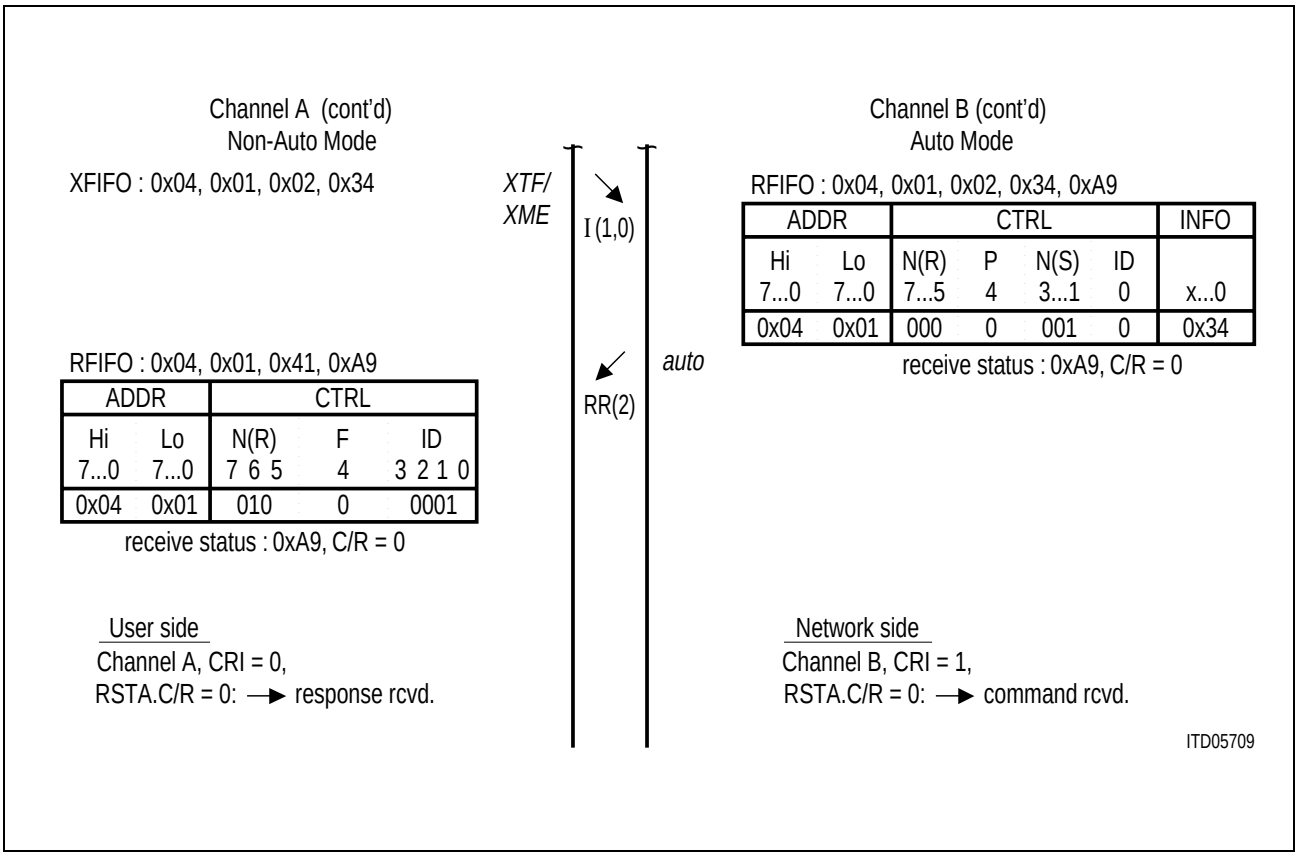
```
WRB:  10 => CCR3
WRB:  06 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAH2
WRB:  00 => RAL2
WRB:  04 => XAD1
WRB:  01 => XAD2
WRB:  90 => CCR0
WRB:  3C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
#
NDC:  ESCC2.Channel A.HDLC/SDLC
#     Clear PLLA flag
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 00 12
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 00 12 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 21 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
NDC:  ESCC2.Channel B.HDLC/SDLC
#     Put ch. B into RNR state
```

```
WRB:  20 => CMDR
RDB:  6A <= STAR
#        -> XRNR
NDC:  ESCC2.Channel A.HDLC/SDLC
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 02 34
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 25 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     Send RR(p)-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 11
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 35 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
NDC:  ESCC2.Channel B.HDLC/SDLC
#     Put ch. B into RR state
WRB:  00 => CMDR
RDB:  4A <= STAR
NDC:  ESCC2.Channel A.HDLC/SDLC
#
#     Send RR(p)-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 11
WRD:   3 Bytes    => XFIFO
RDB:  4A <= STAR
```

```
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 31 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     Send I-frame (XTF)
WRD:  Data Block  => XFIFO
      04 01 02 34
WRD:   4 Bytes    => XFIFO
RDB:  4A <= STAR
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  05 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 02 34 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  04 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 41 A9
RDD:   4 Bytes    <= RFIFO
WRB:  80 => CMDR
WRB:  FF => IMR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

4.10.2 Communication Chart with Frame Structures




Channel A (cont'd)
Non-Auto Mode
XFIFO : 0x04, 0x01, 0x02, 0x34
XTF/
XME
I (1,0)
RR(2)
auto
RFIFO : 0x04, 0x01, 0x41, 0xA9
ADDR
CTRL
Hi
Lo
N(R)
F
ID
7...0
7...0
7 6 5
4
3 2 1 0
0x04
0x01
010
0
0001
receive status : 0xA9, C/R = 0
User side
Channel A, CRI = 0,
RSTA.C/R = 0: → response rcvd.
Channel B (cont'd)
Auto Mode
RFIFO : 0x04, 0x01, 0x02, 0x34, 0xA9
ADDR
CTRL
INFO
Hi
Lo
N(R)
P
N(S)
ID
7...0
7...0
7...5
4
3...1
0
x...0
0x04
0x01
000
0
001
0
0x34
receive status : 0xA9, C/R = 0
Network side
Channel B, CRI = 1,
RSTA.C/R = 0: → command rcvd.
ITD05709

4.11 Auto Mode to Auto Mode

If the ESCC channels operate in Auto Mode on both sides, only the I frames (commands) are stored. Command and response frames are differentiated by the C/R-bit.

The proper setting of the RAH1.CRI bit is very important for automatic error recovery!

Example

- Channel A (auto mode (16)), xmit/rcv I frames (XIF/XME), auto-RR.
- Channel B (auto mode (16)), CRI = 1, xmit/rcv. I frames(XIF/XME), auto-RR.
  (The following UCI pin connections have to be made:

TxDA (36)	→	(25) RxDB
RxDA (35)	←	(26) TxDB
CTSA (33)	←	(24) RTSB
RTSA (34)	→	(23) CTSB)

4.11.1 Track File HAUTOM07

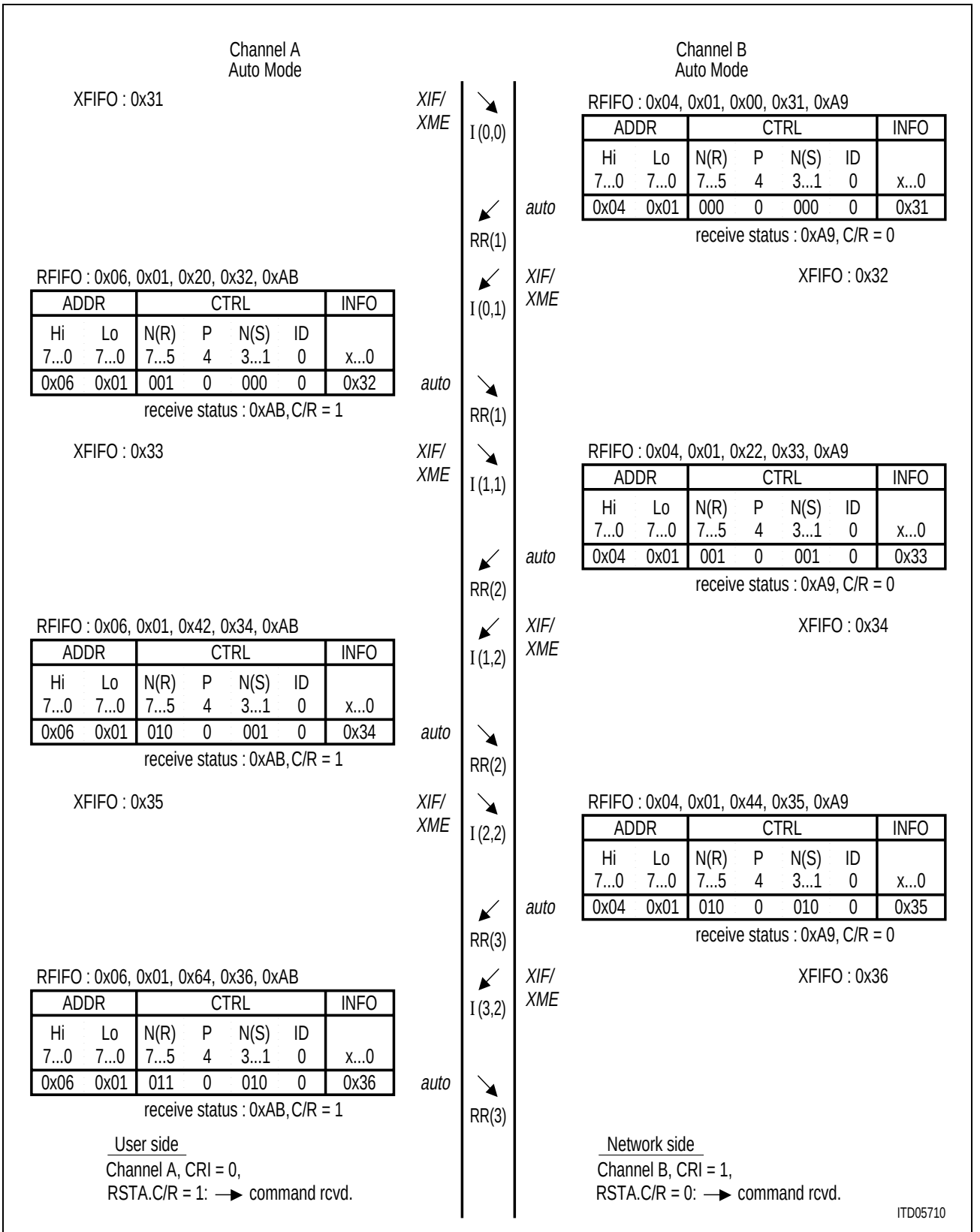
```
#****************************************
#        File name : HAUTOM07.TRK
#        Hdlc AUTO-Mode <-> auto-mode (16),
#        Channel A (6b) <-> B (6b)
#        Channel B RAH1.CRI = 1
#
#        IMPORTANT:
#        Before starting this track file,
#        reset the ESCC2 with the
#        OPTIONS / RESET_PC_BOARD
#        command on the main screen !
#
#        The following pins have to be
*        connected on the UCI:
#        TxDA (36) -> (25) RxDB
#        RxDA (35) <- (26) TxDB
#        CTSA (33) <- (24) RTSB
#        RTSA (34) -> (23) CTSB
#
#****************************************
#:
NDC:  ESCC2.Channel A.HDLC/SDLC
#:       HDLC, FM0, auto-mode (16),
#:       6b, Baud Rate = 1Mbit/s:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  34 => MODE
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  04 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAH2
WRB:  00 => RAL2
WRB:  04 => XAD1
WRB:  01 => XAD2
WRB:  90 => CCR0
WRB:  3C => MODE
WRB:  41 => CMDR
#:
NDC:     ESCC2.Channel B.HDLC/SDLC
#:       HDLC, FM0, auto mode (8),
#:       6b, DPLL, BR div. factor = 1:
WRB:  10 => CCR0
WRB:  65 => TIMR
WRB:  34 => MODE
```

```
WRB:  16 => CCR1
WRB:  10 => CCR2
WRB:  10 => CCR3
WRB:  06 => RAH1
WRB:  01 => RAL1
WRB:  00 => RAH2
WRB:  00 => RAL2
WRB:  04 => XAD1
WRB:  01 => XAD2
WRB:  90 => CCR0
WRB:  3C => MODE
WRB:  41 => CMDR
WRB:  77 => IMR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  80 => IPC
RDB:  08 <= ISR0
WRB:  00 => IPC
WRB:  77 => IMR0
#:
#:       Send I-frame A -> B
WRB:  31 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 00 31 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Send I-frame B -> A
WRB:  32 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
```

```
      06 01 20 32 AB
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Send I-frame A -> B
WRB:  33 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      04 01 22 33 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Send I-frame B -> A
WRB:  34 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 42 34 AB
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Send I-frame A -> B
WRB:  35 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
NDC:  ESCC2.Channel B.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
```

```
RDD:  Data Block  <= RFIFO
      04 01 44 35 A9
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Send I-frame B -> A
WRB:  36 => XFIFO
RDB:  4A <= STAR
WRB:  06 => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
NDC:  ESCC2.Channel A.HDLC/SDLC
INT:  Interrupt(s) pending !
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      06 01 64 36 AB
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#:
#:       Disable Interrupts, power down
WRB:  FF => IMR0
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  FF => IMR0
WRB:  10 => CCR0
NDC:  ESCC2.Channel A.HDLC/SDLC
WRB:  10 => CCR0
#:
#:       End of Track File
```

4.11.2 Communication Chart with Frame Structures



4.12 Appendix

Status and Control Registers in HDLC/SDLC Mode

Register Map for the ESCC2

Register Map for the ESCC8

(The ESCC2 and the ESCC8 use slightly different formats for the GIS, IVA, IPC, PVR, PIS, PIM and PCR register)

4.12.1 ESCC2 Register Map in HDLC/SDLC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	**RSTA**	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	**MODE**	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**XAD1**	XAD1 (LAP-D SAPI, LAP-B command)							
25	**XAD2**	XAD2 (LAP-D TEI, LAP-B response)							
26	**----**								
27	**----**								
28	**RAL1**	RAL1 (in transparent and extended transparent mode only)							
29	**RHCR**	RHCR							
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	NRM	CAS	OV	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	**CCR1**	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	**CCR3**	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	**IPC**	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	**ISR1**	0	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**CCR4**	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	RAH1	RAH1						CRI	0
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	1	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

4.12.2 ESCC8 Register Map in HDLC/SDLC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	**RFIFO**	32 Byte accessible portion of the RFIFO							
20	**STAR**	XDOV	XFW	XRNR	RRNR	RLI	CEC	CTS	WFA
21	**RSTA**	VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA
22	**MODE**	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	**TIMR**	CNT			VALUE				
24	**XAD1**	XAD1 (LAP-D SAPI, LAP-B command)							
25	**XAD2**	XAD2 (LAP-D TEI, LAP-B response)							
26	**----**								
27	**----**								
28	**RAL1**	RAL1 (in transparent and extended transparent mode only)							
29	**RHCR**	RHCR							
2A	**RBCL**	RBC7							RBC0
2B	**RBCH**	DMA	NRM	CAS	OV	RBC11			RBC8
2C	**CCR0**	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	**CCR1**	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7:								
	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	**CCR3**	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	**----**								
31	**----**								
32	**----**								
33	**----**								
34	**VSTR**	CD	DPLA	0	0	VN3			VN0
35	**----**								
36	**----**								
37	**----**								
38	**GIS**	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	**IPC**	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	**ISR0**	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	**ISR1**	0	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	**PVR**	PVR7							PVR0
3D	**PIS**	PIS7							PIS0
3E	**PCR**	PCR7							PCR0
3F	**CCR4**	0	0	0	0	0	0	RFT1	RFT0

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RHR	RNR	STI	XTF	XIF	XME	XRES
21	PRE	PR7							PR0
22	MODE	MDS1	MDS0	ADM	TMD	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	XAD1	XAD1 (LAP-D SAPI, LAP-B command)							
25	XAD2	XAD2 (LAP-D TEI, LAP-B response)							
26	RAH1	RAH1						CRI	0
27	RAH2	RAH2						MCS	0
28	RAL1	RAL1 (in auto and non-auto mode only)							
29	RAL2	RAL2							
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	NRM	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	**SM1…SM0 = 00**	
2D	CCR1	SFLG	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	C32	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	C32	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	C32	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	C32	DIV
2F	CCR3	PRE1	PRE0	EPT	RADD	CRL	RCRC	XCRC	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	RLCR	RC	RL6						RL0
36	AML	AML7							AML0
37	AMH	AMH7							AMH0
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	RME	RFS	RSC	PCE	PLLA	CDSC	RFO	RPF
3B	IMR1	1	RDO	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	CCR4	0	0	0	0	0	0	RFT1	RFT0

5 Monosync/BISYNC Cook-Book

5.1 Summary

Chapter 5 is a step-by-step register programming guide for the Monosync/Bisync Mode of the ESCC. It is based on the operational description given in chapter 2.5 of the ESCC User Manuals.

Wolfram Kiesecker
1994

5.2 Character Oriented Serial Mode (MONOSYNC/BISYNC)

5.2.1 Data Frame

Character oriented protocols achieve synchronization between the transmitting and the receiving station by means of special SYN characters. MONOSYNC and IBM's BISYNC procedures are two examples for character oriented protocols. MONOSYNC uses one SYN character, and BISYNC uses two. **Figure 2** shows an example of the message format.

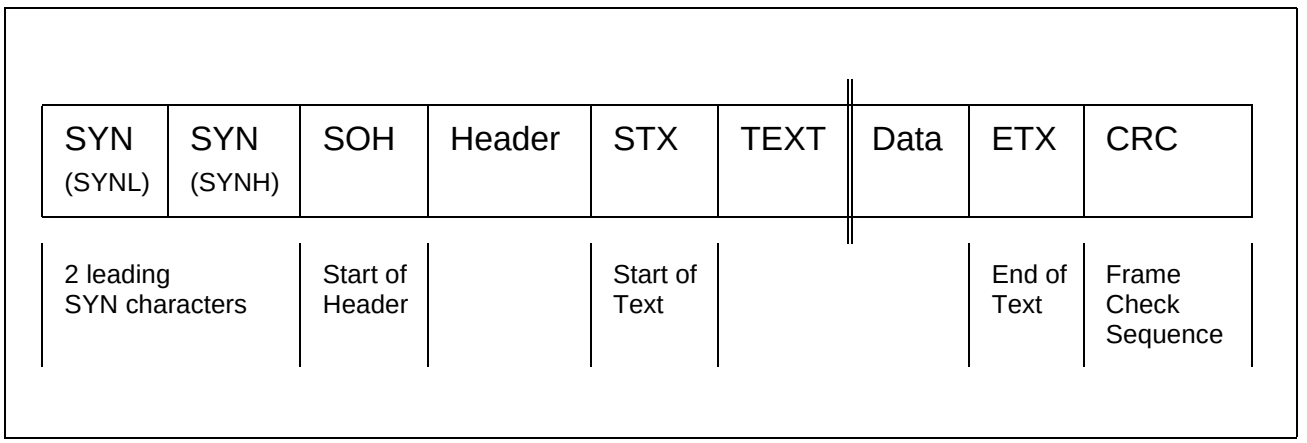


Figure 2
BISYNC Message Format

MONOSYNC / BISYNC serial mode is selected in the CCR0 register:

	7							0	
CCR0	PU	MCE	0	SC2	SC1	SC0	SM1*	SM0*	(2C)

Serial Mode SM1…0	HDLC/SDLC	0	0
	SDLC Loop	0	1
	MONO- / BISYNC	**1**	**0**
	ASYNC	1	1

* Throughout this document: **The unshaded bit-fields are the most relevant for the examined function**.

MONOSYNC/BISYNC operation and the SYN character length is selected in the MODE register. The SYN character is programmable in SYNL for MONOSYNC mode, and in SYNL **and** SYNH for BISYNC mode.

	7							0	
MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP	(22)

SLEN	BISNC	
0	0	6 bit MONOSYNC character
0	1	6 bit BISYNC character
1	0	8 bit MONOSYNC character
1	1	8 bit BISYNC character

- 6 bit MONOSYNC

	7		5					0	
SYNL									(24)

- 6 bit BISYNC (SYNL first, SYNH second)

	7		5					0	
SYNL									(24)
SYNH									(25)

- 8 bit MONOSYNC

	7							0	
SYNL									(24)

- 8 bit BISYNC (SYNL first, SYNH second)

	7							0	
SYNL									(24)
SYNH									(25)

The data character length, parity bit enable and parity format (even / odd / mark / space) is programmable in the DAFO register:

	7							0	
DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0	(22)

Parity format PAR1, PAR0

0	0	SPACE “0”
0	1	odd parity
1	0	even parity
1	1	MARK “1”

Parity Enable PARE

0	parity disabled
1	parity enabled

Data Character Length CHL1, CHL0

0	0	8 bits
0	1	7 bits
1	0	6 bits
1	1	5 bits

5.2.2 Data Reception

A number of prerequisites need to be fulfilled before text or data are received into the RFIFO:

The general receiver functions need to be enabled by setting the RAC bit in the MODE register:

	7							0	
MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP	(22)

Receiver Active RAC
0 Receiver Functions disabled
1 Receiver Functions enabled

Additionally, an external CD signal may be used for data reception control. This feature is called “Carrier Detect Auto Start” and is enabled with the XBCH:CAS bit:

	7					0	
XBCH	DMA	0	CAS	XC	XBC11 XBC8		(2B)

CD Auto Start CAS
0 CD pin is a general input (not in Clock Modes 1&5)
1 Receiver is enabled/disabled with CD pin:
High: receiver enabled
Low: receiver disabled

The receiver starts monitoring the incoming data stream for the presence of the specified SYN character(s) only after a HUNT command (CMDR:HUNT) has been issued:

	7							0	
CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES	(20)

Enter HUNT state HUNT
0 no change
1 Forces the receiver into the HUNT state immediately. Previous synchronization is lost and the search for SYN characters starts.

When SYN is detected, the ESCC synchronizes to the data frame and generates a SCD interrupt (ISR0:SCD):

	7							0	
ISR0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF	(3A)

SYN Character Detect SCD
- 0 no change
- 1 SYN character detected after the HUNT command was issued. Rcv. is now in the synchronized state.

From here on, all data is pushed (LSB first) into the RFIFO, i.e. control sequences, data characters and an optional CRC frame check sequence. If selected with RFC:DPS and RFC:RFDF the character's parity bit and/or the parity status is stored together with each data byte in the RFIFO:

	7						0	
RFC	0	DPS	SLOAD	RFDF	RFTH1 RFTH0	0	TCDE	(28)

Disable Parity Storage
- 0 the parity bit is stored (if DAFO.PARE and < 8 bit/char.)
- 1 the parity bit is **not** stored within the data byte

RFIFO Data Format RFDF
- 0 only data bytes are stored in RFIFO
- 1 status byte for each char. is stored, too.

Generally, the SYN character length and data character length may be programmed independently and differently. However, if the RFC:SLOAD option is selected (store SYN character in RFIFO), the SYN character length and the data character length (including the parity bit) must be the same!

	7						0	
RFC	0	DPS	SLOAD	RFDF	RFTH1 RFTH0	0	TCDE	(28)

SYN Char, Load SLOAD
- 0 SYN characters are **not** stored in the RFIFO
- 1 SYN characters are stored in the RFIFO

Subsequently, the following combinations are allowed if RFC:SLOAD = 1:

MODE SLEN	DAFO CHL1…0	DAFO PARE	RFC:SLOAD len(SYN) = len(data + p)	
0	10	0	6 bit SYN	6 bit data, no parity
0	11	1	6 bit SYN	5 bit data, plus parity
1	00	0	8 bit SYN	8 bit data, no parity
1	01	1	8 bit SYN	7 bit data, plus parity

Data from the RFIFO is transferred into the system memory by a DMA controller or by the CPU in an interrupt controlled fashion:

XBCH

7					0
DMA	0	CAS	XC	XBC11 … XBC8	(2B)

0 Interrupt controlled data transfer (Interrupt Mode)
1 DMA controlled transfer (DMA Mode)

The ESCC generates a DMA request or an interrupt signal when the RFIFO content reaches a certain number of bytes. This threshold is programmable in the RFC register:

RFC

7							0
0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE

(28)

RFIFO Threshold Level RFTH1…0			
	0	0	1 byte
	0	1	4 bytes
	1	0	16 bytes
	1	1	32 bytes

If the ESCC operates in interrupt mode (XBCH:DMA = 0) and the RFIFO threshold level is reached a RPF interrupt is generated and the RPF bit in the ISR0 register is set:

	7							0	
ISR0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF	(3A)
							Receive Pool Full RPF	1	INT!

In order to generate an interrupt signal the RPF-bit needs to be unmasked in the IMR0 register:

	7							0	
IMR0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF	(3A)
							RPF unmasked	0	

The RFIFO stores received data transparently. Software needs to perform de-formatting and data consistency check, such as frame length check, text/data extraction and CRC verification.

5.2.3 Data Reception is Stopped if

1. MODE:RAC is reset

	7							0	
MODE	0	0	SLEN	BISNC	**RAC = 0**	RTS	TRS	TLP	(22)

2. XBCH:CAS is set and the CD signal becomes inactive (low)

	7							0	
XBCH	DMA	0	**CAS = 1**	XC	XBC11			XBC8	(2B)

AND CD pulled low!

3. CMDR:HUNT is issued again

	7							0	
CMDR	RMC	RRES	RFRD	STI	XF	**HUNT = 1**	XME	XRES	(20)

4. CMDR:RRES is issued

	7							0	
CMDR	RMC	**RRES = 1**	RFRD	STI	XF	HUNT	XME	XRES	(20)

5. RFC:TCDE is set and a programmed termination character was detected

	7							0	
RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	**TCDE = 1**	(28)

	7							0	
TCR	**Termination Character Register = last rcvd. character**								(26)

On action 1. (RAC = 0) the reception remains disabled until the receiver is activated again (RAC = 1). The receiver returns into (non-synchronized) HUNT state directly.

After the CD signal became inactive (action 2.) the receiver is re-activated with a hi-level at the CD pin. It returns into (non-synchronized) HUNT state directly.

A HUNT command (action 3.) or RRES command (action 4.) forces the receiver into (non-synchronized) HUNT state directly.

After a termination character the receiver remains inactive until a new HUNT command takes it into (non-synchronized) HUNT state.

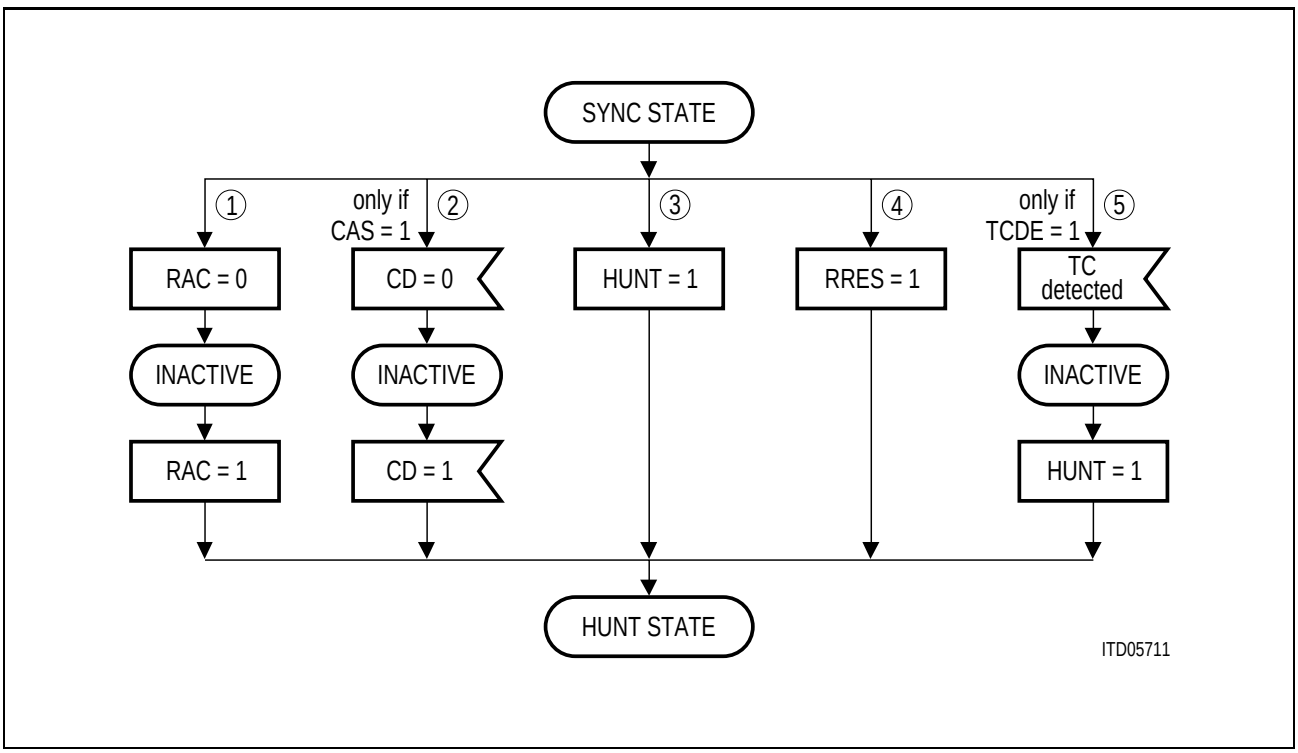


Figure 3
Receiver State Diagram

5.3 Data Transmission

The ESCC provides a deep (2 x 32 bytes) Transmit FIFO (XFIFO) to hold data for MONOSYNC or BISYNC transmission. The XFIFO is filled with up to 32 bytes either by the CPU (interrupt mode) or by the DMA controller (DMA mode). The transmission is initiated (LSB first) with the XF-command (Xmit Frame) in the CMDR register:

	7							0	
CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES	(20)

Transmit Frame XF

0 no change

1 **DMA Mode**: initiate DMA transfer and start serial transmission
INT Mode: start serial transmission

	7					0	
XBCH	DMA	0	CAS	XC	XBC11 XBC8		(2B)

0 INT mode (Interrupt controlled data transfer)

1 DMA mode (DMA controlled data transfer)

The ESCC constructs the MONOSYNC/BISYNC message frame by appending the data bytes from the XFIFO to the SYN character(s). While data is sent out from one half (up to 32 bytes) of the XFIFO, the other half may already be loaded from the system memory via DMA transfer or by the CPU (INT mode).

In DMA mode the transmission is completed when the specified number of characters have been transferred. The number of characters is programmable in the Transmit Byte Count Registers (XBCL, XBCH):

	7							0	
XBCL	XBC7							XBC0	(2A)

Length of the next data block to be transmitted (XBC8…XBC11 in XBCH)

	7					0	
XBCH	DMA	0	CAS	XC	XBC11 XBC8		(2B)

1 (DMA only!) with XBCL (1…4096 byte block)

In INT mode the ESCC generates XPR interrupts to indicate that it is ready to load more data into the XFIFO.

	7							0	
ISR1	0	0	ALLS	XDU	TIN	CSC	XMR	XPR	(3B)

Transmit Pool Ready XPR 1 INT!

(IMR1:XPR needs to be unmasked (0) in order to generate an interrupt signal:)

	7							0	
IMR1	0	0	ALLS	XDU	TIN	CSC	XMR	XPR	(3B)

XPR unmasked 0

The CPU issues a new XF command every time after refilling the XFIFO. The last data block in a frame is marked by a XF/XME command combination:

	7							0	
CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES	(20)

Transmit Message End XF/XME **1** **1**

The data format (character length, parity bit, parity format) is programmable in DAFO:

	7							0	
DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0	(22)

Parity format (PAR1, PAR0)	0	0	SPACE “0”
	0	1	odd parity
	1	0	even parity
	1	1	MARK “1”

Parity Enable (PARE)	0	parity disabled
	1	parity enabled

Data Character Length (CHL1, CHL0)	0	0	8 bits
	0	1	7 bits
	1	0	6 bits
	1	1	5 bits

DAFO effects only data character, not SYN, CRC or preamble bytes.

The ESCC can automatically calculate a 16 bit CRC sequence and append it to the message frame. The CRC generator is programmable in register CCR3:

	7							0	
CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD	(2F)

CRC on (CON) 0 exclude next byte/word from CRC calc.
1 include next byte/word into CRC calcul.

CRC Reset Level (CRL) 0 initialized to FFFF
1 initialized to 0000

CRC Append (CAPP) 0 off
1 on

Select CRC Algorithm (CRCM) 0 CRC-16
1 CRC-CCITT

CCR3 gives a choice between the CRC-16 and the CRC-CCITT algorithm.

CCR3:CON allows to specifically include (CON = 1) or exclude (CON = 0) characters during the CRC calculation. In other words: CON determines whether the **current byte** (**or word**) written to the XFIFO has to be included into the CRC calculation or not. It has to be programmed **before** the affected byte/word is written to the XFIFO. In case of a word access both characters are either included or excluded. It is not necessary to reprogram CON for every character, when consecutive characters are all either included into or excluded from the CRC calculation.

Internally generated SYN characters are always excluded from the CRC calculation.

The CRC generator is automatically initialized before the transmission of a new frame starts. The initialization value is selected with CCR3:CRL.

The CRC sequence (2 bytes) is always sent without parity bits.

The ESCC sends automatically interframe timefill upon completion of a message frame. The interframe timefill format can be either SYN characters or IDLE (continuous hi level). The selection is made in CCR1:

	7							0	
CCR1	0	0	0	ODS	ITF	CM2		CM0	(2D)

Interframe Timefill Format 0 continuous “1”
1 send SYN characters

5.4 Special Functions

5.4.1 Preamble Transmission

The preamble bit pattern (8-bit) is programmable in the PRE register. Preamble transmission enable and the number of pattern repetitions is selected in CCR3:

	7							0	
PRE	PR7							PR0	(21)

8-bit preamble bit pattern, which is sent out during preamble transmission

	7							0	
CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD	(2F)

x	x	0	preamble transmission disabled
0	0	1	1 x preamble pattern
0	1	1	2 x preamble pattern
1	0	1	4 x preamble pattern
1	1	1	8 x preamble pattern

Preamble transmission starts immediately after the interframe timefill transmission and precedes a new message frame. If the preamble pattern equals the SYN pattern, the receiving side will trigger with the preamble.

5.4.2 Continuous Transmission (DMA mode only!)

Normal DMA mode operation (XBCH:DMA = 1) requests the specified number of bytes (XBC0…XBC11) from the system memory and sends them out as one message frame. However, in Continuous Transmission Mode (XBCH:DMA = 1, XBCH:XC = 1) the transmit byte count is ignored and the ESCC will re-initiate 32-byte DMA transfers every time new data can be entered into the XFIFO:

	7							0	
XBCH	DMA	0	CAS	XC	XBC11			XBC8	(2B)

1	0	“normal” DMA Mode (read XBC0…XBC11)
1	**1**	Continuous Transmission

This feature allows transmission of frames with more than 4095 bytes (maximum transmit byte count in XBC0…XBC11). Continuous Transmission goes on until the XC bit is reset. When XC is reset during continuous transmission XBC0…XBC11 becomes valid again and the ESCC will complete the operation with the transmission of the specified number of bytes. If a data underrun condition occurs in the XFIFO during continuous transmission the ESCC suspends its DMA requests and the data transmission. Instead of a CRC sequence the message frame is terminated with continuous “1” (IDLE).

5.4.3 CRC Parity Inhibit

For applications that do not make use of the internal CRC generator (CCR3:CAPP = 0), an externally (software) calculated 16-bit checksum may be excluded from the automatic data formatting (parity, etc.). Characters are formatted if:

	7							0	
CCR3	PRE1	PRE0	EPT	**CON = 0**	CRL	**CAPP = 0**	CRCM	PSD	(2F)

CON must be set for the two bytes which contain the checksum and a Transmit Message end command needs to issued after writing these two bytes:

2. Set CON

	7							0	
CCR3	PRE1	PRE0	EPT	**CON = 1**	CRL	CAPP = 0	CRCM	PSD	(2F)

3. Write two bytes into XFIFO
4. Issue Transmit Message End command

	7							0	
CMDR	RMC	RRES	RFRD	STI	**XF = 1**	HUNT	**XME = 1**	XRES	(20)

5.5 Appendix A

Status and Control Registers in BISYNC Mode

Register Map for the ESCC2

Register Map for the ESCC8

(The ESCC2 and the ESCC8 use slightly different formats for the GIS, IVA, IPC, PVR, PIS, PIM and PCR register)

5.5.1 ESCC2 Register Map in BISYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	RFNE	SYNC	0	CEC	CTS	0
21	----								
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	0	CAS	0	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PI	0	0	0	ISA1	ISA0	ISB1	ISB0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	ISR1	0	0	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	----								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES
21	PRE	PR7							PR0
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	----								
36	----								
37	----								
38	IVA	T7				T3	0	0	0
39	IPC	VIS	0	0	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	1	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	IMR1	1	1	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

5.5.2 ESCC8 Register Map in BISYNC Mode

Offset	Read Registers (*read-back* registers are shaded)								
00… 1F	RFIFO	32 Byte accessible portion of the RFIFO							
20	STAR	XDOV	XFW	RFNE	SYNC	0	CEC	CTS	0
21	----								
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	RBCL	RBC7							RBC0
2B	RBCH	DMA	0	CAS	0	RBC11			RBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
2E	clock mode 0a, 1: CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7: CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4: CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5: CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	----								
31	----								
32	----								
33	----								
34	VSTR	CD	DPLA	0	0	VN3			VN0
35	----								
36	----								
37	----								
38	GIS	PIA	PIB	PIC	PID	CII	CN2	CN1	CN0
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	ISR0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	ISR1	0	0	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIS	PIS7							PIS0
3E	PCR	PCR7							PCR0
3F	----								

Offset	Write Registers								
00… 1F	XFIFO	32 Byte accessible portion of the XFIFO							
20	CMDR	RMC	RRES	RFRD	STI	XF	HUNT	XME	XRES
21	PRE	PR7							PR0
22	MODE	0	0	SLEN	BISNC	RAC	RTS	TRS	TLP
23	TIMR	CNT			VALUE				
24	SYNL	SYNL							
25	SYNH	SYNH							
26	TCR	TCR7							TCR0
27	DAFO	0	0	0	PAR1	PAR0	PARE	CHL1	CHL0
28	RFC	0	DPS	SLOAD	RFDF	RFTH1	RFTH0	0	TCDE
29	----								
2A	XBCL	XBC7							XBC0
2B	XBCH	DMA	0	CAS	XC	XBC11			XBC8
2C	CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0
2D	CCR1	0	0	0	ODS	ITF	CM2		CM0
	clock mode 0a, 1:								
	CCR2	SOC1	SOC0	0	0	0	RWX	0	DIV
	clock mode 0b, 2, 3, 6, 7:								
2E	CCR2	BR9	BR8	BDF	SSEL	TOE	RWX	0	DIV
	clock mode 4:								
	CCR2	SOC1	SOC0	0	0	TOE	RWX	0	DIV
	clock mode 5:								
	CCR2	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV
2F	CCR3	PRE1	PRE0	EPT	CON	CRL	CAPP	CRCM	PSD
30	TSAX	TSNX						XCS2	XCS1
31	TSAR	TSNR						RCS2	RCS1
32	XCCR	XBC7							XBC0
33	RCCR	RBC7							RBC0
34	BGR	BR7							BR0
35	----								
36	----								
37	----								
38	IVA	T7	T6	T5	T4	T3	T2	ROT	EDA
39	IPC	VIS	ROTM	SLA2	SLA1	SLA0	CASM	IC1	IC0
3A	IMR0	TCD	1	PERR	SCD	PLLA	CDSC	RFO	RPF
3B	IMR1	1	1	ALLS	XDU	TIN	CSC	XMR	XPR
3C	PVR	PVR7							PVR0
3D	PIM	PIM7							PIM0
3E	PCR	PCR7							PCR0
3F	----								

5.6 Appendix B

5.6.1 ASCII-Code Table

		Bits 6…4							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Bits 0…3	**0**	NUL	DLE	SP	0	@	P	‘	p
	1	SOH	XON	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	EXT	XOFF	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	’	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	|
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	_	o	DEL

5.7 Appendix C

5.7.1 Track Files for the EASY532 Evaluation System

5.7.2 Track File 1

```
####### EASY532 Track File ########
#
#     BISY_1.TRK
#
###################################
#
#     A simple example for BISYNC mode
#     Channel A Tx -> Channel A Rc
#     internal test loop
#
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
###################################
#
#     Reset state:
#     ============
NDC:  ESCC2.Channel A.HDLC/SDLC
#
#     BISYNC mode:
#     ============
WRB:  02 => CCR0
NDC:  ESCC2.Channel A.BISYNC
#
#     clock mode 7b,
#     SYN characters between frames,
#     baud rate div. factor = 1:
#     ==========================
WRB:  0F => CCR1
WRB:  10 => CCR2
#
#     Power Up in BISYNC mode:
#     ========================
WRB:  82 => CCR0
#
#     8-bit BISYNC characters:
#     ========================
WRB:  30 => MODE
```

```
#
#     SYN char. = ASCII-SYN (0x16):
#     =============================
WRB:  16 => SYNL
WRB:  16 => SYNH
#
#     Unmask RPF interrupt source:
#     ============================
WRB:  FE => IMR0
#
#     8-bit BISYNC characters,
#     Activate the receiver,
#     switch test loop:
#     =================
WRB:  39 => MODE
#
#     Enter hunt state:
#     =================
WRB:  04 => CMDR
#
#     Write data into XFIFO:
#     ======================
WRD:  Data Block  => XFIFO
      00 01 02 03
WRD:   4 Bytes    => XFIFO
#
#     Transmit frame &
#     Transmit message end command:
#     =============================
WRB:  0A => CMDR
#
#     Interrupt !!!
#     =====================
INT:  Interrupt(s) pending !
#
#     Investigate interrupt source:
#     =============================
RDB:  04 <= GIS
#           -> Channel A ISR0 interrupt
RDB:  01 <= ISR0
#           -> RPF (data available in RFIFO)
INT:  Interrupt(s) acknowledged
#
#     Read RFIFO:
#     ===========
RDB:  00 <= RFIFO
#
```

```
#     Acknowledge RFIFO read:
#     =======================
WRB:  80 => CMDR
#
#     Another Interrupt !!!
#     =====================
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
#           -> another RPF
INT:  Interrupt(s) acknowledged
#
#     Read RFIFO and acknowledge:
#     ===========================
RDB:  01 <= RFIFO
WRB:  80 => CMDR
#
#     AGAIN: RPF interrupt handler:
#     =============================
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RFIFO
WRB:  80 => CMDR
#
#     … and one more …
#     ====================
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RFIFO
WRB:  80 => CMDR
#
#     End of track file
#     4 characters transmitted,
#     and 4 characters received.
```

5.7.3 Track File 2

```
####### EASY532 Track File ########
#
#     BISY_2.TRK
#
###################################
#
#     Very similar to BISY_1.TRK
#     except that CRC automatically
#     transmitted and received as two
#     additional bytes.
#
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
###################################
#
#     BISYNC initialization
#     =====================
WRB:  02 => CCR0
NDC:  ESCC2.Channel A.BISYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
#
#     Enable and append CRC-16:
#     =========================
WRB:  14 => CCR3
#
#     Power On in BISYNC mode
WRB:  82 => CCR0
WRB:  30 => MODE
WRB:  16 => SYNL
WRB:  16 => SYNH
WRB:  FE => IMR0
WRB:  39 => MODE
#
#     HUNT !
#     ======
WRB:  04 => CMDR
#
#     the same 4 bytes:
#     =================
WRD:  Data Block  => XFIFO
      00 01 02 03
```

```
WRD:   4 Bytes    => XFIFO
WRB:  0A => CMDR
#
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  00 <= RFIFO
WRB:  80 => CMDR
#           -> 1st data byte
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  01 <= RFIFO
WRB:  80 => CMDR
#           -> 2nd data byte
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  02 <= RFIFO
WRB:  80 => CMDR
#           -> 3rd data byte
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  03 <= RFIFO
WRB:  80 => CMDR
#           -> 4th data byte
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  10 <= RFIFO
WRB:  80 => CMDR
#           -> 1st CRC byte
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  85 <= RFIFO
WRB:  80 => CMDR
#           -> 2nd CRC byte
#
#
#     End of track file
```

5.7.4 Track File 3

```
####### EASY532 Track File ########
#
#     BISY_3.TRK
#
###################################
#
#     This track file uses the ETX
#     termination character for end
#     of frame recognition. The RFIFO
#     threshold is 32 bytes.
#
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
#####################################
#
#     BISYNC initialization
#     =====================
WRB:  02 => CCR0
NDC:  ESCC2.Channel A.BISYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  82 => CCR0
WRB:  30 => MODE
WRB:  16 => SYNL
WRB:  16 => SYNH
#
#     ASCII - ETX termination char.
#     =============================
WRB:  03 => TCR
#
#     RFIFO threshold = 32 bytes
#     termination character
#     recognition enabled:
#     ====================
WRB:  0D => RFC
#
#     TCD and RPF interrupt enabled:
#     ==============================
WRB:  7E => IMR0
#
WRB:  39 => MODE
WRB:  04 => CMDR
```

```
#
#     ETX appended to data characters:
#     ================================
WRD:  Data Block  => XFIFO
      41 42 43 44 03
WRD:   5 Bytes    => XFIFO
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
#           -> TCD Interrupt
INT:  Interrupt(s) acknowledged
#
#     Check received byte count:
#     ==========================
RDB:  05 <= RBCL
#
#     Block read from RFIFO:
#     ======================
RDD:  Data Block  <= RFIFO
      41 42 43 44 03
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     End of track file
########################################
```

5.7.5 Track File 4

```
####### EASY532 Track File ########
#
#     BISY_4.TRK
#
###################################
#
#     The same procedure as in BISY_3.TRK.
#     Two frames are transmitted.
#     The receiver has to be brought back
#     into HUNT state after the first frame.
#
#     IMPORTANT:
#     Before starting this track file,
#     reset the ESCC2 with the
#     OPTIONS / RESET_PC_BOARD
#     command on the main screen !
#
#####################################
#
#     BISYNC initialization
#     =====================
WRB:  02 => CCR0
NDC:  ESCC2.Channel A.BISYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  82 => CCR0
WRB:  30 => MODE
WRB:  16 => SYNL
WRB:  16 => SYNH
WRB:  03 => TCR
WRB:  0D => RFC
WRB:  7E => IMR0
WRB:  39 => MODE
WRB:  04 => CMDR
WRD:  Data Block  => XFIFO
      41 42 43 44 03
WRD:   5 Bytes    => XFIFO
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      41 42 43 44 03
RDD:   5 Bytes    <= RFIFO
```

```
WRB:  80 => CMDR
#
#     new HUNT command required after
#     termination character recognition:
#     ==================================
WRB:  04 => CMDR
WRD:  Data Block  => XFIFO
      45 46 47 48 03
WRD:   5 Bytes    => XFIFO
WRB:  0A => CMDR
INT:  Interrupt(s) pending !
RDB:  04 <= GIS
RDB:  80 <= ISR0
INT:  Interrupt(s) acknowledged
RDB:  05 <= RBCL
RDD:  Data Block  <= RFIFO
      45 46 47 48 03
RDD:   5 Bytes    <= RFIFO
WRB:  80 => CMDR
#
#     End of track file
########################################
```

6 Async Cook-Book

6.1 Summary

This application note is intended to illustrate the aspects of the ESCC that are required for Asynchronous operation.

Noel Giamello
1993

6.2 General Information

This application note is intended to be used with the technical manual of the SAB82532 (ESCC2) or the SAB82538 (ESCC8). The full functional description of the ASYNC mode is found in chapter 2, section 4 of the ESCC User Manuals and the detailed register description is given in chapter 4, section 2.

6.2.1 Introduction

The Enhanced Serial Communication Controllers ESCC2/8 (SAB82532, two channel device and the SAB82538, eight channel device) are multiprotocol data communication controllers with two/eight symmetrical serial channels. They have been designed to implement high-speed communication links and to reduce the hardware and software overhead needed for serial synchronous/asynchronous communications.

Each channel contains an independent clock generator, DPLL, encoder/decoder and programmable protocol hardware. This application note provides information on the asynchronous and isochronous modes of operation. The DPLL is available in isochronous mode. Data communications with asynchronous, synchronous character oriented, and HDLC based protocols with extended support of X.25 LAPB, the ISDN LAPD, and SDLC protocols is implemented. The ESCC2/8 is capable of handling a large set of layer 2 protocol functions independently of the host processor.

The 82532N-10 and 82538N-10 versions of the Enhanced Serial Communication Controller (ESCC2/8) opens a wide area for application which use time division multiplex methods (e.g. time-slot oriented PCM systems, systems designed for packet switching, ISDN applications) by its programmable telecom-specific features. In this special operating mode (clock mode 5), which is applicable to all serial modes (HDLC/SDLC, ASYNC, BISYNC), the ESCC2/8 can transmit or receive data packets in one of up to 64 time-slots of programmable width.

The device is controlled via a parallel 16-bit wide interface which is directly compatible with the most popular 8/16 bit microprocessors (Siemens/Intel or Motorola type). The internal FIFOs (64 bytes per direction and channel) with additional DMA capability provide a powerful interface to the higher layers implemented in a microcontroller. For interrupt controlled systems, the ESCC2/8 supports daisy chaining and interrupt vector generation.

Please refer to the ESCC technical manual for sections 1.1 to 2.2.5. The Asynchronous Serial Mode CookBook will cover specifics for the Asynchronous and Isochronous Modes of operation.

6.3 Asynchronous Serial Mode

6.3.1 Character Frame

Character framing is achieved by special Start and Stop bits. Each data character is preceded by one Start bit and terminated by one or two Stop bits. The character length is selectable from 5 up to 8 bits. Optionally, a parity bit can be added which complements the number of ones to an even or odd quantity (even/odd parity). The parity bit can also be programmed to have a fixed value (Mark or Space). **Figure 4** shows the asynchronous character format.

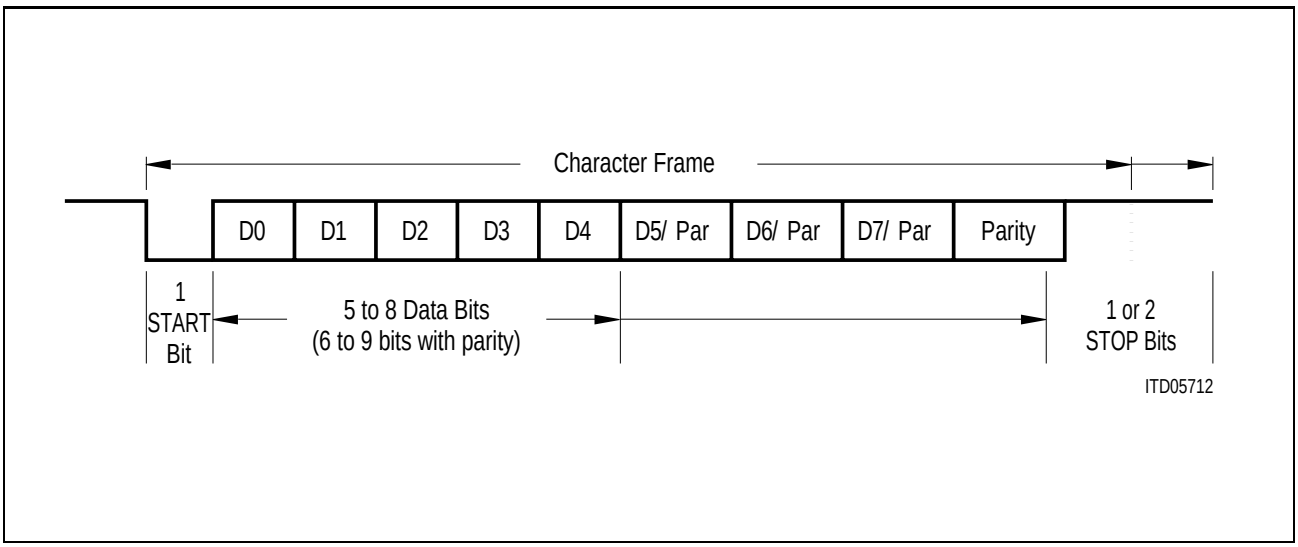


Figure 4
Asynchronous Character Frame

6.3.2 Data Reception

6.3.3 Operating Modes

The ESCC2 offers the flexibility to combine clock modes, data encoding and data sampling in many different ways. However, only definite combinations make sense and are recommended for correct operation in the asynchronous mode. To properly program the ESCC2 (or 8) the following registers have to utilized.

Channel Configuration Register 0 (Read/Write)

The asynchronous serial mode is selected in the Channel Configuration Register 0 (CCR0) register by writing the appropriate value into the SM1 and SM0 bit locations. The data encoding scheme is chosen with bits SC2, SC1 & SC0 (for example NRZ) and power up or power down mode with the PU bit.

First, in setting up **CCR0** for asynchronous mode:

CCR0	PU	MCE	0	SC2	SC1	SC0	SM1*	SM0*	(2C)

SERIAL MODE SM1…0		
HDLC/SDLC	0	0
SDLC loop	0	1
Mono/Bisync	1	0
Async	**1**	**1**

* Throughout this document: **The unshaded bit-fields are the most relevant for the examined functions.**

The **data encoding** scheme (for example NRZ) is chosen by setting the three bits SC2, SC1 and SC0 to the appropriate serial port configuration:

Note: The Isochronous mode requires data encoding for correct operation.

CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0	(2C)

Serial Port Configuration SC2…SC0			
0	0	0	NRZ data encoding
0	0	1	(not recommended mode)
0	1	0	NRZI data encoding
0	1	1	(not recommended)
1	0	0	FM0 data encoding
1	0	1	FM1 data encoding
1	1	0	Manchester data encoding
1	1	1	(not used)

CCR0	PU	MCE	0	SC2	SC1	SC0	SM1	SM0	(2C)

0	power down (standby)	**PU…Switches between power up and power down mode**
1	power up (active)	

Mode Register (Read/Write)

The Mode Register is used to define the state and control of Request To Send (RTS) function:

MODE	0	0	0	FLON	RAC	RTS	TRS	TLP	(22)

Request To Send $\overline{\text{RTS}}$ (active low signal)… 0
The **$\overline{\text{RTS}}$** pin is controlled by the ESCC2 autonomously. **$\overline{\text{RTS}}$** is activated when data transmission starts and deactivated when transmission is completed.

The **$\overline{\text{RTS}}$** pin is controlled by the CPU… 1
If this bit is set, the **$\overline{\text{RTS}}$** pin is activated immediately and remains active until this bit is reset.

Channel Configuration Register 1 (Read/Write)

The Channel Configuration Register 1 (CCR1) is used to select which **clock mode** you want to use. Please note that clock mode 5 is allowed only for the SAB82532N-10 and SAB82538N-10.

CCR1	0	0	0	ODS	BCR	CM2	CM1	CM0	(2D)

Clock Mode CM2…CM0			
clock mode 0	0	0	0
clock mode 1	0	0	1
clock mode 2	0	1	0
clock mode 3	0	1	1
clock mode 4	1	0	0
clock mode 5	1	0	1
clock mode 6	1	1	0
clock mode 7	1	1	1

The extension “a” or “b” for the clock modes (0a, 0b, 2a, 2b, 3a, 3b, 6a, 6b, 7a, & 7b) is set in the CCR2 register which is described on page 5.

The **Output Driver Select Bit** (ODS) defines the function of the transmit data pins (TxDA, TxDB):

CCR1	0	0	0	ODS	BCR	CM2	CM1	CM0	(2D)

Output Driver Select ODS

0	TxD pin is an open-drain output.
1	TxD pin is a push-pull output.

The **Bit Clock Rate** (BCR) selects either asynchronous or isochronous operation. This bit is valid (set) only for clock modes *not* using the DPLL (0, 1, 3b, 4, 7b).

CCR1	0	0	0	ODS	BCR	CM2	CM1	CM0	(2D)

Bit Clock Rate BCR

0	selects isochronous operation with a bit clock rate = 1. Data is sampled once. (**See Isochronous Mode**).
1	selects standard asynchronous operation with a clock rate = 16. Data is sampled 3 times around the nominal bit center. The effective BIT value is determined by majority decision. For correct operation, **NRZ** data encoding has to be selected.

6.3.4 Asynchronous Mode

Prerequisites for this mode:

- Bit clock rate 16 selected (CCR1.BCR = 1)
- Clock mode 0, 1, 3b, 4, or 7b selected
- NRZ data encoding

The receiver, which operates with a clock rate equal to 16 times the nominal data bit rate, synchronizes itself to each character by detecting and verifying the Start Bit. Since character length, parity and Stop Bit length is known, the ensuing valid bits are sampled. Oversampling (3 samples) around the nominal bit center in conjunction with majority decision is provided every received bit (including Start Bit).

The synchronization lasts for one character, the next incoming character causes a new synchronization to be performed. As a result, the demand for high clock accuracy is reduced. Two communication stations using the asynchronous procedure are clocked independently, their clocks need not be in phase or locked to exactly the same frequency but, in fact, may differ from one another within a certain range.

6.3.5 Isochronous Mode

Prerequisites for this mode:

- Bit clock rate 1 selected (CCR1.BCR = 0)
- Clock mode 2, 3a, 6, or 7a (DPLL mode) has to used in conjunction with FM0, FM1 or Manchester encoding

The isochronous mode uses the asynchronous character format. However, each data bit is only sampled once (no oversampling).

In clock modes 0 and 1, the input clock has to be externally phase locked to the data stream. This mode allows much higher transfer rates. Clock modes 3b, 4 and 7b are not recommended due to difficulties with bit synchronization when using the internal baud rate generator.

In clock modes 2, 3a, 6 and 7a, clock recovery is provided by the internal DPLL. Correct synchronization of the DPLL is achieved if there are enough edges within the data stream, which is generally ensured only if Bi-Phase encoding (FM0, FM1 or Manchester, see CCR0 mentioned previously) is used.

Channel Configuration Register 2 (Read/Write)

The CCR2 register allows programming of the Baud rate Generator Register (BGR) baud rate division factor (BDF bit) and selection of clock mode extensions “a” or “b” (Clock Source Select bit SSEL). CCR2 also contains the two MSBs (BRG9 and BRG8) for the BRG division factor. Bits BRG7...BRG0 are in the Baud Rate Generator Register (BGR).

CCR2

clock modes 0a,1	SOC1	SOC1	0	0	0	RWX	0	DIV	(2E)
clock modes 0b, 2, 3, 6, 7	**BR9**	**BR8**	**BDF**	SSEL	TOE	RWX	0	DIV	
clock mode 4	SOC1	SOC0	0	0	TOE	RWX	0	DIV	
clock mode 5	SOC1	SOC0	XCS0	RCS0	TOE	RWX	0	DIV	

Baud Rate Division Factor BDF

0 The division factor of the baud rate generator is set to 1 (constant).

1 The division factor is determined by BR9…BR0 bits in **CCR2** and **BRG** registers.
Note: unused bits have to be set to logical “0”.

As can be seen above the CCR2 register has four different configurations depending on the clock mode. In continuing with the configuration that uses the BDF and baud rate generator.

clock modes 0b, 2, 3, 6, 7	BR9	BR8	BDF	**SSEL**	TOE	RWX	O	DIV

Clock Source Select SSEL

0 selects clock mode extension “a”

1 selects clock mode extension “b” (depends on chosen clock mode)

Next, you need to program the **B**aud Rate **G**enerator **R**egister (**BGR**) if you intend to use a clock mode that utilizes the baud rate generator.

BGR	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	(34)

In the BGR register you program the eight LSBs of the baud rate generator division factor. The resulting factor is k = (N+1) x 2, when enabled with the **B**aud rate **D**ivision

Factor (BDF) bit in the CCR2 register. The value of N is set by the 8 bits of the BGR register plus the 2 bits (BRG8 and BRG9) of the CCR2 register. Therefore, N can be programmed between 0 to 1023 ($2^{10} – 1$).

Interrupt Mask Register 0, 1 (WRITE)

IMR0	TCD	TIME	PERR	PERR	PLLA	CDSC	RFO	RPF	(3A)
IMR1	BRK	BRKT	ALLS	XOFF	TIN	CSC	XON	XPR	(3B)

Note: Unused bits have to be set to logical “0”.

Each interrupt source can generate an interrupt signal at port pin INT (the function of the output stage is defined via the IPC, Interrupt Port Configuration, register). A “1” in a bit position of IMR0 or IMR1 sets the mask active (the bit is “masked”) for the interrupt status in ISR0 or ISR1. Masked interrupt status bits neither generate an interrupt vector or a signal on INT, nor are they visible in the GIS register (Global Interrupt Status register). In addition, the interrupts will not be displayed in the Interrupt Status Register if the VIS (Masked Interrupts Visible) bit is set to “0”. The interrupts will be displayed in the Interrupt Status Register if the VIS bit is set to a “1”. The VIS bit is found in the IPC.

Note: After RESET, all interrupts are disabled.

The bit definitions for **ISR0** and **ISR1** are provided next and match those of the IMR0 and IMR1.

Interrupt Status Register 0 (READ)

ISR0	TCD	TIME	PERR	PERR	PLLA	CDSC	RFO	RPF	(3A)

All bits are reset when ISR0 is read. Additionally, TCD (Termination Character Detected) and RPF (Receive Pool Full) are reset when the corresponding interrupt is output.

Note: If the IPC.VIS bit (VIS bit in the IPC register) is set to “1”, interrupt status bits in ISR0 may be flagged although they are masked via the IMR0 register. However, these masked interrupt status bits do not generate an interrupt vector or a signal on the INT, nor are they visible in the GIS register.

TCD Termination Character Detected
The termination character (TCR) has been received and a data block (of length less than threshold level) is now available in the RFIFO. The actual block length can be determined by reading the RBCL (Receive Byte Count Low) register first.

TIME Time Out
The time out limit has been exceeded.
If the respective mask bit is reset (i.e. TIME interrupt is enabled), the received data stream is monitored for exceeding the fixed time limit after the last character has been received (time limit = 4 x CFL; Character Frame Length which includes the start bit, character length, parity bit and stop bits).

PERR Parity Error
This bit is valid only if the parity check/generation is enabled.
If it is set (1) then a character with parity error has been received. If enabled via RFDF (RFIFO Data Format bit is located in the RFC, RFIFO Control Register), parity error information is stored in RFIFO in the status byte pertaining to that character.

FERR Framing Error
This bit indicates that a character has been received with a framing error, i.e. the receiver has detected a “0” in a stop bit position. If enabled via RFDF (RFIFO Data Format bit in the RFC register), this information is stored in the RFIFO in the status byte pertaining to that character.

PLLA DPLL Asynchronous
This bit is only valid when the receive clock is supplied by the DPLL and FM0, FM1 or Manchester data encoding is selected.

CDSC Carrier Detect Status Change
Indicates that a state transition has occurred on the CD pin. The actual state can be read from the VSTR (Version Status Register-READ only) register.

RFO Receive FIFO Overflow
This interrupt is generated if RFIFO (Receive FIFO) is full and a further character is received. This interrupt can be used for statistical purposes and indicates that the CPU does not respond quickly enough to an RPF or TCD interrupt.

RPF Receive Pool Full
This bit is set if the RFIFO is filled with data (character and optional status information) up to the programmed threshold level.

Interrupt Status Register 1 (READ)

All bits are reset when ISR1 is read. Additionally, XPR (Transmit Pool Ready) is reset when the corresponding interrupt vector is output.

Note: If bit IPC.VIS is set to “1”, interrupt status bits in ISR1 may be flagged although they are masked via the IMR1 register. However, these masked interrupt status bits neither generate an interrupt vector or a signal on the INT pin, nor are they visible in the GIS register.

BRK Break
This bit is set when a Break signal which is a static low level for a time equal to (character length + parity + stop bit(s)) has been detected on RxD.

BRKT Break Terminated
This bit is set when a Break signal on RxD (Receive Data) is terminated.

ALLS All Sent
This bit is set when the XFIFO is empty and the last character is completely sent out on TxD (Transmit Data).

XOFF XOFF Character Detected
This interrupt status indicates that the currently received character matches the value specified via the XOFF register. XOFF is used for flow control along with XON. This function is independent of the programming of bit MODE.FLON (Flow control On bit in the Mode register).

TIN Timer Interrupt
The internal timer has expired (see also description of TIMR, timer register).

CSC Clear To Send Status Change
Indicates that a state transition has occurred on CTS\ (a “\” indicates an active low signal). The actual state can be read from the STAR (status) register (CTS bit).

XON XON Character Detected
This interrupt status indicates that the currently received character matches the value specified via the XON (XON character) register. XON is used for flow control along with XOFF. The function is independent of the programming of bit MODE.FLON (Flow Control On bit in the Mode register).

XPR Transmit Pool Ready
A data block of up to 32 bytes can be written to the XFIFO (transmit FIFO).

6.3.6 Test Loop

You have the capability of setting up a test loop which internally connects the transmitter and receiver of the same channel. The test loop is activated in the MODE register (Test Loop TLP). The receiver also has to be made active by setting the RAC (Receiver Active) bit.

Mode Register (Read/Write)

Switches the receiver to operational or inoperational state:

MODE	0	0	0	FLON	RAC	RTS	TRS	TLP	(22)

Receiver Active RAC

0	receiver inactive
1	receiver active

Test Loop TLP

Inactive	0
The input and output of the ASYNC channels are internally connected.	1

6.3.7 Storage of Data

In addition to enabling the receiver, you have to program an appropriate RFIFO data format. This is done by setting the RFDF (RFIFO Data Format) bit and the RFTH1 & RFTH2 (RFIFO Threshold Level) bits in the RFC (RFIFO Control) register.

The received data is stored in the RFIFO with the LSB received first. The CD input may be used to control data reception. Character length, number of Stop bits and the optional parity bit are checked. Storage of parity bits can be disabled. Errors are indicated through interrupts. Additionally, the character error status (framing and parity) can optionally be stored in the RFIFO. (can refer to chapter 4.2.2 in the ESCC User's Manual).

Filling of the accessible part of RFIFO is controlled by:

- programmable threshold level
- detection of the programmable Termination Character (optional).

Additionally, the Time-Out condition as optional status information indicates that a certain time (refer to register ISR0) has elapsed since the reception of the last character.

RFIFO Control Register (Read/Write)

RFC	0	DPS	0	RFDF	RFTH1	RFTH0	0	TCDE	(28)

RFIFO Data Format RFDF

0 only data bytes (character plus optional parity up to 8 bit) are stored.

1 additionally to every data byte, an attached status byte is stored.

RFIFO Threshold Level

		Threshold Level (bytes) RFDF = 0	RFDF = 1
0	0	1 (1d)	2 (1d + 1s)
0	1	4 (4d)	4 (2d + 2s)
1	0	16 (16d)	16 (8d + 8s)
1	1	32 (32d)	32 (16d + 16s)

d = data byte s = status byte

If the threshold level is reached, the RPF interrupt is generated if enabled. After RPF is generated, the contents of RFIFO (RFTH bytes) can be read by the CPU. To indicate that this RFIFO pool can be released, an RMC (Receive Message Complete bit in the Command Register) command has to issued.

Reading data from RFIFO can be done in 8 bit (byte) or 16 bit (word) accesses depending on the selected bus interface mode. The LSB is received first from the serial interface.

6.3.8 Data Transmission

The selection of asynchronous or isochronous operation has no further influence on the transmitter. The bit clock rate is solely a dividing factor for the selected clock source.

Transmission of the contents of XFIFO starts after the XF command is issued (the LSB is sent out first). Further data is requested by interrupt (XPR) or DMA. The character frame for each character, consisting of Start Bit, the character itself with defined character length, optionally generated bit and Stop Bit(s) is assembled.

After finishing transmission (indicated by the “All Sent” interrupt), IDLE (logical “1”) is transmitted on TxD.

In addition, the $\overline{CTS}$ signal may be used to control data transmission.

6.4 Special Features

6.4.1 Break Detection/Generation

Break generation: On issuing the XBRK command (register DAFO), the TxD pin immediately forced to physical “0” level with the first following clock edge, and released with the first clock edge after this command has been reset.

Break detection: The ESCC recognizes the Break condition upon receiving consecutive (physical) “0”s for the defined character length, the optional parity and the selected number of Stop Bits (“zero” character and framing error). The “zero” character is not pushed to RFIFO. If enabled, the BRK interrupt is generated.

6.4.2 Interrupt Controlled Data Transfer (Interrupt Mode)

This mode is selected if the DMA bit in the XBCH (Transmit Byte Count High) register is set. Up to 32 bytes/16 works of received data can be read from the RFIFO following a RPF (Receive Pool Full) interrupt or a TCD (Termination Character Detected) interrupt depending on the selected RFIFO mode (see RFC register description):

RPF interrupt: A fixed number of bytes/words (programmed threshold level RFTH0,1) has to read by the CPU.

TCD interrupt: Termination character detected. The received data stream is monitored for “termination character” (programmable in the TCR (Termination Character register). The number of valid bytes in RFIFO is determined by reading the RBCL register.

If necessary, the CPU can access the RFIFO by issuing RFIFO Read command (CMDR.RFRD) before threshold level or the termination condition is reached. The number of valid bytes is determined by reading the RBCL register. Additionally, the RFNE bit in the STAR register will indicate whether or not the RFIFO is not empty.

6.4.3 Continuous Transmission (DMA Mode Only)

If data transfer from system memory to the ESCC is done by DMA (DMA bit in XBCH is set) the number of characters to be transmitted is usually defined via the Transmit Byte Count registers (XBCH, XBCL: bits XBC11...XBC0).

Transmit Byte Count Low (WRITE)

	7						0	
XBCL	**XBC7**						**XBC0**	(2A)

Together with XBCH (bits XBC11...XBC8) this register is used in DMA Mode only, to program the length (1...4095 bytes) of the next data block to be transmitted.

This allows the ESCC2 to request the correct amount of DMA cycles after an XF command in CMDR.

Transmit Byte Count High (WRITE)

	7					0	
XBCH	**DMA**	**0**	**CAS**	**XC**	**XBC11**	**XBC8**	(2B)

Note: Unused bits have to be set to logical “0”.

DMA **DMA Mode**
Selects the data transfer mode of ESCC2 to/from System Memory.
0...Interrupt controlled data transfer (Interrupt Mode).
1...DMA controlled data transfer (DMA Mode).

CAS **Carrier Detect Auto Start**
When set, a high on the CD pin enables the corresponding receiver and data reception is started. When not set, if not in Clock Mode 1 or 5, the CD pin can be used as a general input.

XC **Transmit Continuously**
Only valid if DMA Mode is selected.
If the XC bit is set, the ESCC2 continuously requests for transmit data ignoring the transmit byte count programmed via XBCH, XBCL.

XBC11…XBC8 **Transmit Byte Count (most significant bits)**
Valid only if DMA Mode is selected.
Together with XBCL (bits XBC7...XBC0), determine the number of characters to be transmitted.

Please note that if the “Transmit Continuously” (XC) bit in XBCH is set, the byte count value is ignored and the DMA interface of the ESCC will continuously request for transmit data any time 32 characters can be stored in XFIFO.

Note: If the XC bit is reset during continuous transmission, the transmit byte count becomes valid again, and the ESCC will request the amount of DMA transfers programmed via XBC11...XBC0. Otherwise, the continuous transmission is stopped when a data underrun condition occurs in XFIFO, i.e. the DMA controller does not transfer further data to ESCC. In this case continuous “1”s (IDLE) are transmitted.

6.5 Appendix

EASY532 Track Files for Asynchronous Operation

6.5.1 Track Files

Actual examples of code that has been used to program the ESCC are sometimes known as “track files“. The track files were generated using the EASY532 Evaluation Tool. An introduction to the tool and the usage of track files is given in chapter 7 of this application manual.

To ensure that the ESCC is in a defined state before starting the execution of a track file it is recommended that a hardware reset is generated with the “OPTIONS/RESET PC BOARD” (**refer to chapter 7, Hardware Reset section for explanation**).

The first track file shows the ESCC in a TEST mode using the loop-back feature and asynchronously sending data from Channel A transmitter to Channel A receiver. The second track file was used to communicate asynchronously from the ESCC2, EASY532, to another PC via the COM port. The second PC was running the Windows Terminal application.

6.5.2 Loop-Back on Channel A

```
#: ****************************************
#: SIEMENS AG
#: Frank Sauber ICD Santa Clara USA
#: file name : escc2_01
#: ASYNC- Mode
#: clockmode 7b // TxD open drain
#: Testloop // data clock 488 Hz
#: ****************************************
#:
NDC: ESCC2.Channel A.HDLC/SDLC
#: ASYNC-Mode
WRB: 03 => CCR0
NDC: ESCC2.Channel A.ASYNC
#: TxD is open drain
#: clock mode 7
#: Bit Clock Rate 16
WRB: 0F => CCR1
#: enables BRG
#: clock mode extension b
WRB: F0 => CCR2
#: BRG div. factor
WRB: FF => BRG
#: enables interrupt(s)
WRB: FE => IMR0
WRB: CC => IMR1
#: define level of data filled in RFIFO
WRB: 0C => RFC
#: receiver active
#: test loop (channel A -> channel A)
#: escc2 CPU controlled
WRB: 0D => MODE
#: power up
WRB: 83 => CCR0
INT: Interrupt(s) pending !
RDB: 08 <= GIS
RDB: 01 <= ISR1
INT: Interrupt(s) acknowledged
#: write XFIFO
WRD: Data Block => XFIFO
     28 48 68 88
WRD:  4 Bytes => XFIFO
#: transmit data
WRB: 08 => CMDR
INT: Interrupt(s) pending !
RDB: 08 <= GIS
RDB: 21 <= ISR1
```

```
INT: Interrupt(s) acknowledged
#: read-enable RFIFO
WRB: 20 => CMDR
#: read RFIFO
RDD: Data Block <= RFIFO
     28 48 68 88
RDD:  4 Bytes <= RFIFO
#: read RBCL (in RFIFO received bits)
RDB: 04 <= RBCL
WRB: 80 => CMDR
```

6.5.3 PC (ESCC-EASY532) to PC (Windows-Terminal App) Communication via COM Port

```
**********************************
Noel Giamello/ SIEMENS Components Inc.
Clock Mode 7b, Async Mode,
**********************************
WRB: FF => IMR0
WRB: FF => IMR1
WRB: 03 => IPC                    Interrupt Port Configuration
WRB: 00 => XBCH                   Transmit Byte Count High
WRB: 00 => MODE
WRB: 03 => CCR0
NDC: ESCC2.Channel A.ASYNC
WRB: 48 => RFC                    RFIFO Control Register
WRB: 1F => CCR1
WRB: 33 => BRG                    (BGR) Baud Rate Generator Register
WRB: 30 => CCR2
WRB: 0C => MODE
WRB: 83 => CCR0
WRB: 3C => IMR0
INT: Interrupt(s) pending !
RDB: 04 <= GIS                    Global Interrupt Status Register
RDB: 40 <= ISR0                   Interrupt Status Register 0-Interrupt
                                  has occurred due to a time-out.
INT: Interrupt(s) acknowledged
WRB: DE => IMR1                   Interrupt Mask Register 1
RDB: 42 <= STAR                   Status Register
RDB: 42 <= STAR
WRB: 41 => CMDR
INT: Interrupt(s) pending !
RDB: 0C <= GIS
RDB: 01 <= ISR1
RDB: 04 <= GIS
RDB: 40 <= ISR0
INT: Interrupt(s) acknowledged
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 40 <= ISR0
INT: Interrupt(s) acknowledged
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 41 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 10 <= RBCL
RDD: Data Block <= RFIFO          Data received from the remote
                                  terminal (in Windows)
```

```
     30 31 32 33 34 35 36 37
     38 39 30 31 32 33 34 35
RDD: 16 Bytes <= RFIFO            Number of bytes received from the
                                  remote terminal.
RDB: 62 <= STAR                   RFIFO is not empty, XFIFO, Clear To Send
                                  active.
WRB: 80 => CMDR                   Receive Message Complete - see
                                  technical manual for explanation
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 40 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 62 <= STAR
WRB: 20 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 80 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 05 <= RBCL                   There are 5 bytes available in the
                                  RFIFO-sent from remote.
RDD: Data Block <= RFIFO
     36 37 38 39 41
RDD: 5 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 41 <= ISR0                   Time out interrupt
INT: Interrupt(s) acknowledged
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 40 <= ISR0
INT: Interrupt(s) acknowledged
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 42 <= ISR0                   Time out and a RFIFO overflow
                                  (over 32 bytes received)
INT: Interrupt(s) acknowledged
RDB: 62 <= STAR
RDD: Data Block <= RFIFO
     30 31 32 33 34 35 36 37
     38 39 41 42 43 44 45 46
RDD: 16 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
```

```
RDB: 01 <= ISR0
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     30 31 32 33 34 35 36 37
     38 39 41 42 43 44 45 46
RDD: 16 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 01 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 39 <= RFIFO
RDD: Data Block <= RFIFO
     39 30 31 32 33 34 35 36
     37 38 39 41 42 43 44 20
RDD: 16 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
RDB: 42 <= STAR
WRB: 20 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 80 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 00 <= RBCL
RDB: 42 <= STAR
WRB: 80 => CMDR
RDB: 0C <= MODE
WRB: 0D => MODE
WRD: Data Block => XFIFO          Data written into the XFIFO of the
     01 02 03 04 05 06 07 08      ESCC -to be transmitted to the
     09 0A 0B 0C 0D 0E 0F 00      remote terminal.
     01 02 03 04 05
WRD: 21 Bytes => XFIFO
RDB: 42 <= STAR
WRB: 08 => CMDR
INT: Interrupt(s) pending !
RDB: 0C <= GIS
RDB: 21 <= ISR1
RDB: 41 <= ISR0
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     01 02 03 04 05 06 07 08
     09 0A 0B 0C 0D 0E 0F 00
RDD: 16 Bytes    <= RFIFO
RDB: 62 <= STAR
```

```
WRB: 80 => CMDR
WRB: FF => XFIFO
RDB: 62 <= STAR
WRB: 08 => CMDR
INT: Interrupt(s) pending !
RDB: 0C <= GIS
RDB: 21 <= ISR1
RDB: 40 <= ISR0
INT: Interrupt(s) acknowledged
RDB: 62 <= STAR
WRB: 20 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 80 <= ISR0
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     01 02 03 04 05 FF
RDD:  6 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
WRD: Data Block => XFIFO
     00
WRD:  1 Bytes => XFIFO
WRD: Data Block => XFIFO
     01 02 03 04 05 06 07 08
     09 0A 0B 0C 0D 0E 0F 00
     01 02 03 04 05 06 07 08
     09 0A 0B 0C 0D 0E 0F
WRD: 31 Bytes    => XFIFO
RDB: 42 <= STAR
WRB: 08 => CMDR
INT: Interrupt(s) pending !
RDB: 0C <= GIS
RDB: 41 <= ISR0
RDB: 21 <= ISR1
INT: Interrupt(s) acknowledged
WRD: Data Block => XFIFO
     11 22 33 44 55 66 77 88
     99 AA BB CC DD EE FF 00
WRD: 16 Bytes => XFIFO
WRB: 12 => XFIFO
RDB: 62 <= STAR
WRB: 08 => CMDR
INT: Interrupt(s) pending !
RDB: 0C <= GIS
RDB: 21 <= ISR1
RDB: 42 <= ISR0
```

```
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     00 01 02 03 04 05 06 07
     08 09 0A 0B 0C 0D 0E 0F
RDD: 16 Bytes <= RFIFO
RDB: 20 <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 01 <= ISR0
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     00 01 02 03 04 05 06 07
     08 09 0A 0B 0C 0D 0E 0F
RDD: 16 Bytes <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 01 <= ISR0
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
     11 22 33 44 55 66 77 88
     99 AA BB CC DD EE FF 00
RDD: 16 Bytes <= RFIFO
RDB: 20 <= RFIFO
RDB: 62 <= STAR
WRB: 80 => CMDR
RDB: 42 <= STAR
WRB: 20 => CMDR
INT: Interrupt(s) pending !
RDB: 04 <= GIS
RDB: 80 <= ISR
INT: Interrupt(s) acknowledged
RDB: 00 <= RBCL
RDB: 42 <= STAR
WRB: 80 => CMDR
```

7 Bus Configuration

7.1 Summary

The ESCC supports bus configuration by collision detection and resolution. Chapter 7 provides some application hints for building up a bus system based on the ESCC.

Karl Singendonk
July 1994

7.2 General Information

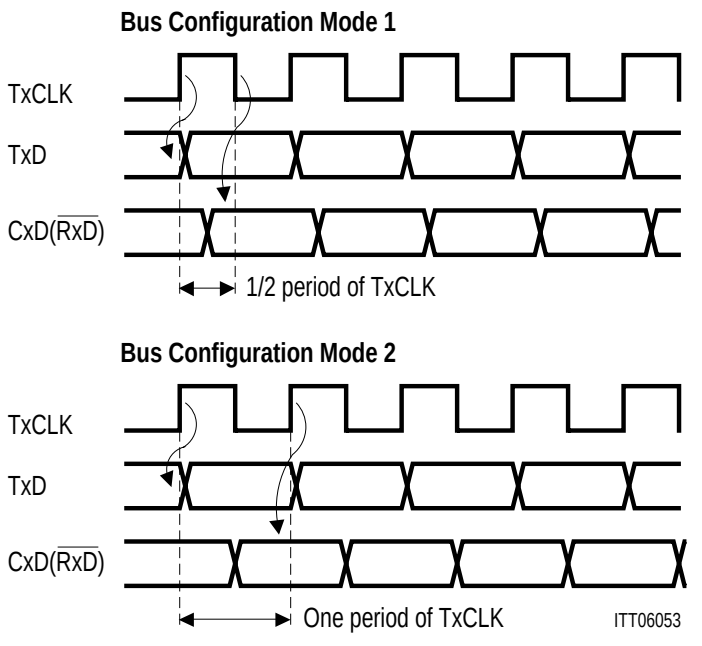
Beside the point-to-point configuration, the ESCC effectively supports point-to-multipoint (pt-mpt, or bus) configurations by means of internal idle and collision detection/collision resolution methods.

Prerequisites for bus operation are:

- NRZ encoding in bus configuration mode (CCR0.SC2 … SC0)

	7							0	
CCR0	**PU**	**MCE**	**0**	**SC2**	**SC1**	**SC0**	**SM1**	**SM0**	(2C)
Bus configuration, timing mode 1, NRZ				**0**	**0**	**1**			
Bus configuration, timing mode 2, NRZ				**0**	**1**	**1**			

The data are transmitted with the rising edge of TxCLK. Depending on the timing mode, that has been selected, the time interval between transmitting data and evaluation of the transmitted data (through pin CxD) for collision detection can be 1/2 period or one period of TxCLK.



- OR'ing of data from every transmitter on the bus
- feedback of bus information (CxD input).

Note, that in bus configuration mode

- the idle state of the transmit data pin TxD is continuous logical '1' implicitly
- 'ONE insertion' can be used as follows

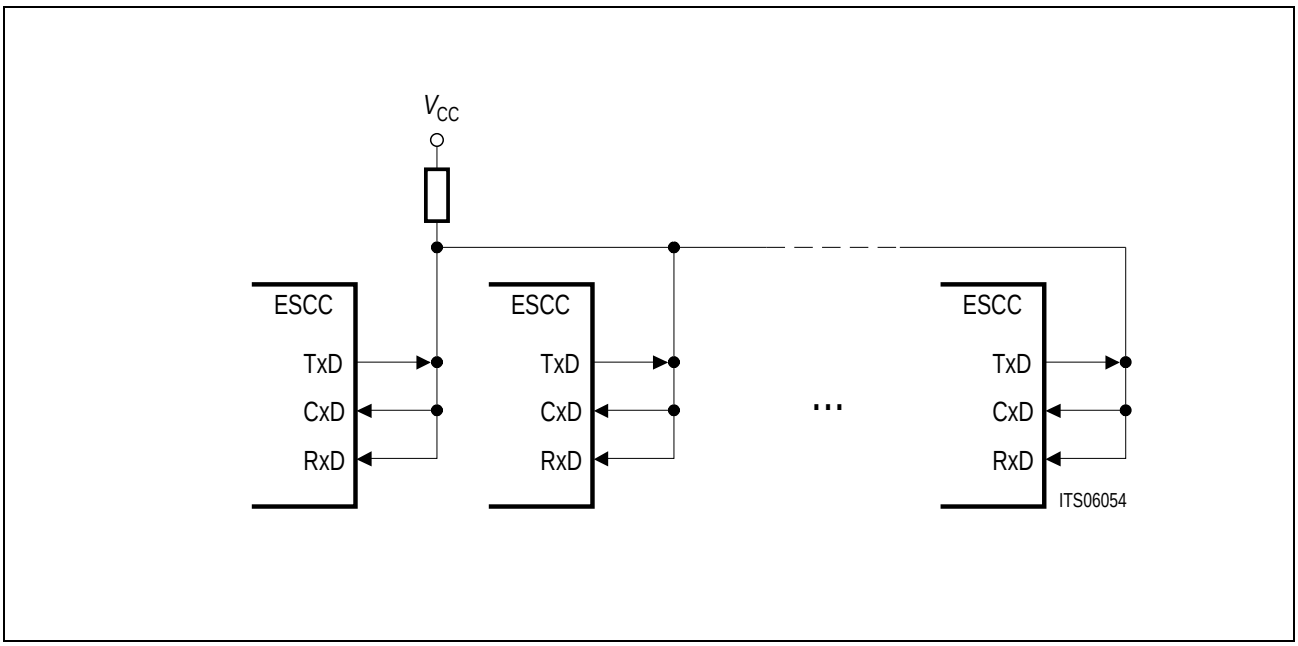
	7							0	
CCR1	**SFLG**	**GALP**	**GLP**	**ODS**	ITF**/OIN**	**CM2**	**CM1**	**CM0**	(2D)

ONE insertion (deletion) mechanism is deactivated	**0**
ONE insertion (deletion) mechanism is activated	**1**

When the ONE insertion (deletion) mechanism is activated, a '1' is inserted after seven consecutive '0's in the transmit data stream and a '1' is deleted after seven consecutive '0's in the receive data stream. This enables clock information to be recovered from the receive data stream by means of a DPLL even in case of NRZ encoding, because a transition at bit cell boundary occurs at least every 7 bits. The 'ONE Insertion' cannot be used in conjunction with the master clock option.

7.3 Application Example

A simple way to OR the data from every transmitter on the bus is a wired-or configuration, using the TxD open drain capability:



	7							0	
CCR1	**SFLG**	**GALP**	**GLP**	**ODS**	**ITF/OIN**	**CM2**	**CM1**	**CM0**	(2D)
TxD pin is an open drain output				**0**					

The initialization can be performed as follows (e.g. for the EASY532-Board, when considering the HW modifications):

```
WRB: 03  => IPC          ; Interrupt: Push/Pull output, active high
WRB: 84  => CCR0         ; Power Up, Bus Configuration Mode 1
WRB: 07  => CCR1         ; Clock Mode 7(a), TxD: open drain
WRB: 88  => MODE         ; transparent mode, receiver active
WRB: 7F  => IMR0         ; enable RME interrupt
WRB: DC  => IMR1         ; enable ALLS, XMR, and XPR interrupt
WRB: 01  => CMDR         ; transmitter RESET
INT: Interrupt(s) pending!
RDB: 01   <= ISR1        ; XPR interrupt
INT: Interrupt(s) acknowledged
```

Since in this specific application example the ESCC is operating in transparent mode 0, all HDLC frames are received. An example for data transmission is shown in the next section of the track file: the channel, that has transmitted a frame, receives the same frame from the bus.

```
WRD: Data Block => XFIFO
    01 02 03 04 05       ; write data into XFIFO
WRD: 5 Bytes => XFIFO
RDB: 48 <= STAR          ; check CEC
WRB: 0A => CMDR          ; set XTF and XME cmd
INT: Interrupt(s) pending!
RDB: 21 <= ISR1          ; ALLS and XPR interrupt
RDB: 80 <= ISR0          ; RME interrupt
INT: Interrupt(s) acknowledged
RDD: Data Block <= RFIFO
   01 02 03 04 05 A2     ; read data out of RFIFO
RDD: 6 Bytes <= RFIFO
RDB: 48 <= STAR          ; check CEC
WRB: 80 => CMDR          ; RMC cmd
```

7.3.1 HDLC/SDLC Bus Configuration Mode

In **HDLC/SDLC bus configuration mode**, the ESCC provides:

- collision avoidance
- collision detection
- collision resolution.

7.3.2 Collision Avoidance

By means of idle detection on the bus, the transmitter does not start transmission before detecting 8 or 10 consecutive 1's. The minimum number of consecutive 1's before starting the transmission depends on the current priority of the corresponding station (please, **refer to Priority**).

7.3.3 Collision Detection

During the transmission, the data transmitted on TxD is compared with the data on CxD. In case of mismatch (1 sent and 0 detected, or vice versa) data transmission is immediately aborted, and idle (logical 1) is transmitted. Depending on the timing mode, that has been selected, the time interval between sending data and evaluation of the transmitted data for collision detection can be 1/2 period or one period of TxCLK.

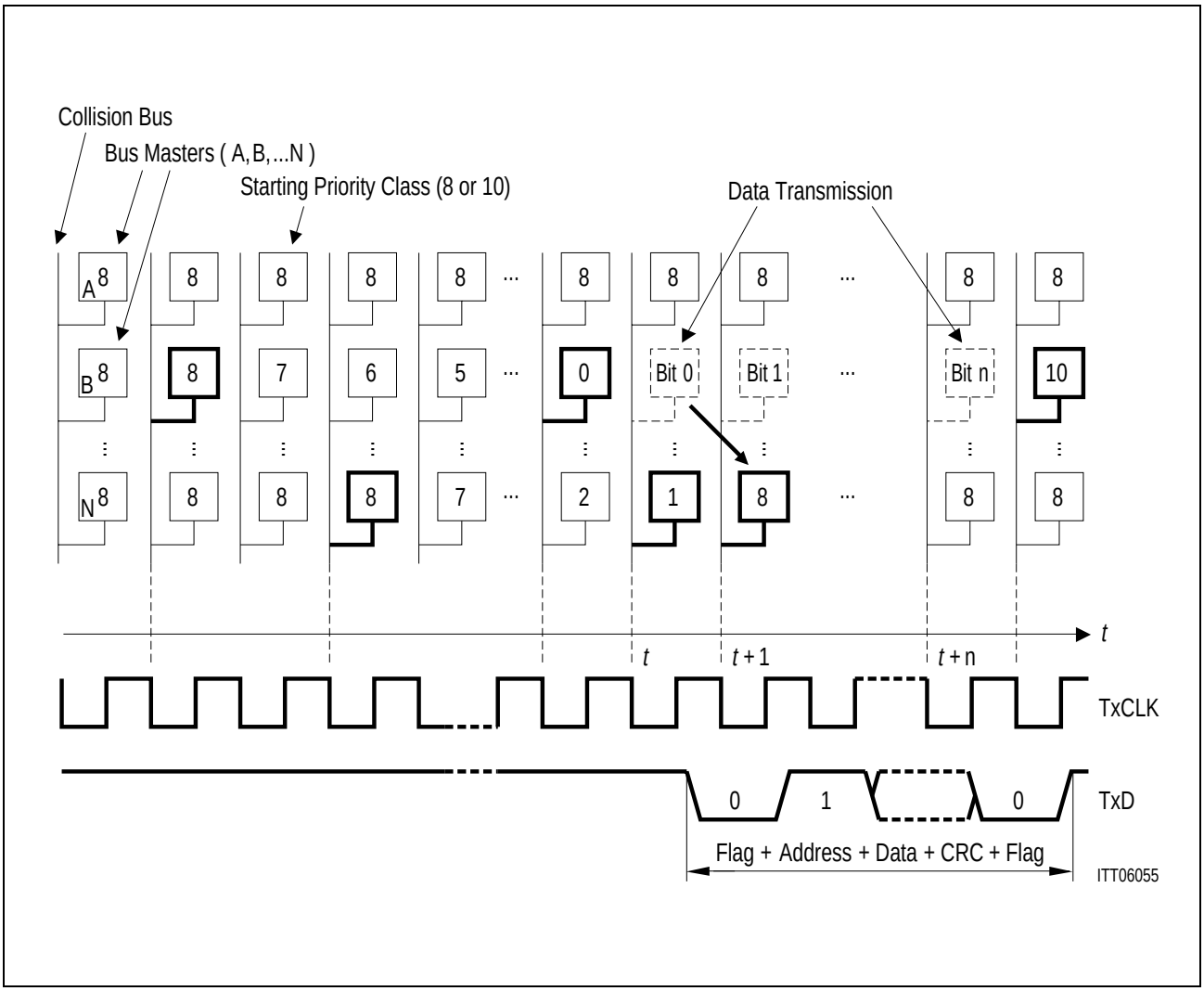
7.3.4 Collision Resolution

Since a zero ('low') on the bus prevails over a 1 ('high impedance'), if a wired-or connection is implemented, and since the address fields of the HDLC frames sent by different stations normally differ from one another, the fact that a collision has occurred will be detected prior to or at the latest within the address field. The frame of the transmitter with the highest temporary priority (determined by the address field) is not affected and is transmitted successfully. All other stations cease transmission immediately and return to bus monitoring state.

Transmission will be initiated again by the ESCC2 as soon as possible if the first part of the frame is still present in the XFIFO. If not, an XMR interrupt is generated.

7.3.5 Priority

At the beginning all stations, that are connected to the bus, are in the same priority class '8': Therefore, each station has to monitor the bus for at least eight consecutive '1's. To ensure, that all competing stations are given a fair access to the transmission medium, once a station has successfully completed the transmission of a frame, it is given the lower level of priority:'10'.



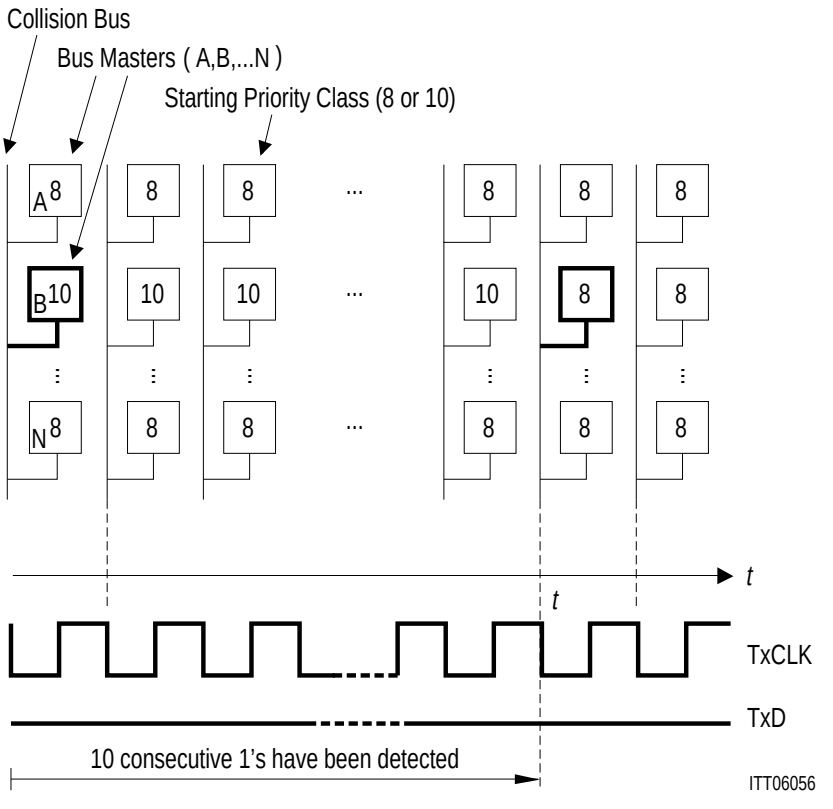
The diagram above shows an example, where several masters *(A, B, … N)* are sharing the same collision bus in a multimaster configuration. At the beginning all stations are in the same priority class ’8’ (high). At first station *B* gets a transmit request and monitors the bus for eight consecutive 1’s before starting transmission. The wait procedure is shown by the counter [*7,6,5,…0*], that is decremented with every subsequent bit, when it is 1 (“high“). In the meantime station *N* gets an transmit request, too, and starts decrementing its internal counter in the same way.

When station *B* has detected eight consecutive 1’s on the bus, it starts transmission by sending the first bit [bit 0] of the opening flag. Station *N* detects the zero (= bit 0 of station *B*), and concludes that the bus is not free; its internal counter is set to ’8’ again.

Since only six (or less) consecutive 1’s can be seen on the bus as long as station *B* is transmitting a frame, no additional station will try to access to the bus. Consequently a collision is avoided.

When an HDLC frame has been transmitted successfully, the internal priority class of the corresponding station is decreased (’8’ -> ’10’). Thus, in order for the same station (station *B*) to be able to transmit another frame, ten consecutive 1’s on the bus must be detected. This guarantees that the transmission requests of other stations (e.g. station *N* with priority class ’8’) are satisfied before a same station (station *B*) is allowed a second bus access.

When ten consecutive 1’s have been detected, transmission is again allowed and the priority class is increased.



Assuming that several stations have the priority class ’10’ and ten consecutive 1’s have been detected, all stations get the priority class ’8’ automatically.

7.3.6 Bus Configuration in BISYNC Mode

In **BISYNC mode**, the collision-resolution is implemented by the microprocessor. After collision detection the transmitter and XFIFO is reset and pin TxD goes to ’1’. The XMR interrupt is provided, which requests the microprocessor to repeat the whole message or block of characters. No priority mechanism is implemented for BISYNC Mode.

7.3.7 Bus Configuration in ASYNC Mode

In **ASYNC mode**, a bus configuration is not recommended.

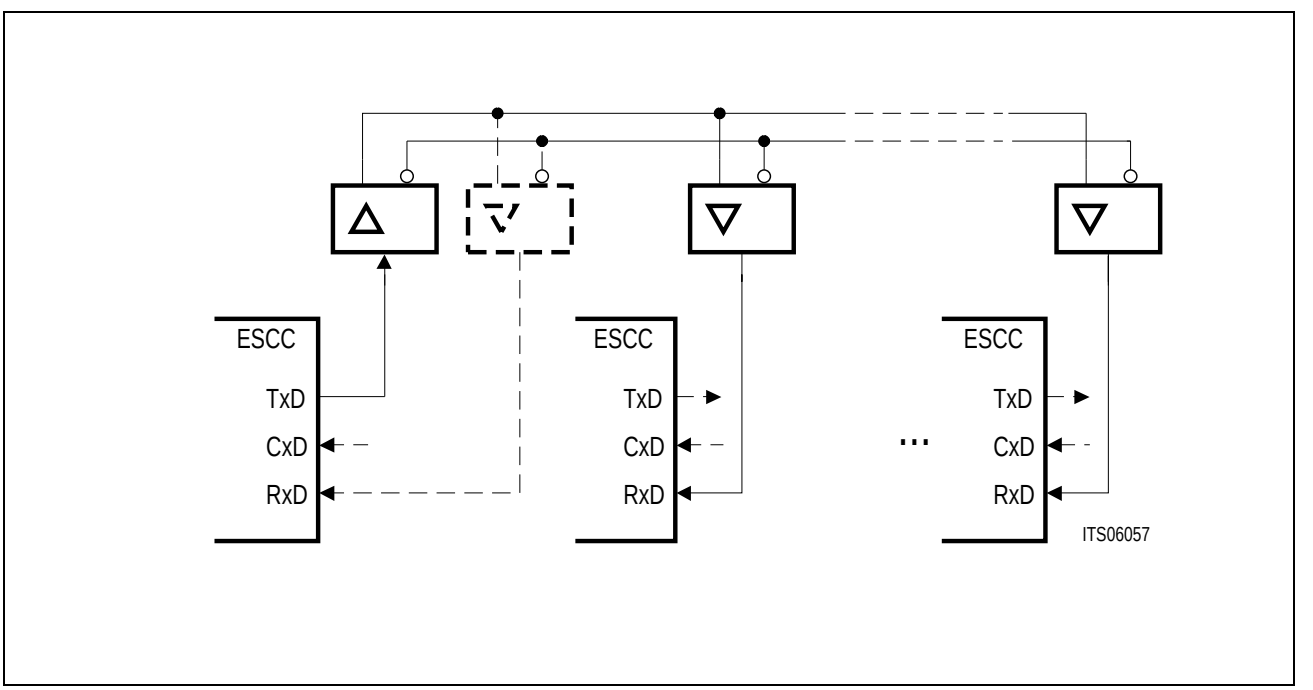
7.3.8 Balanced Transmission Lines

High speed data transmission between computer system components and peripherals over long distances, under high noise conditions, usually proves to be very difficult. RS422 and RS485 are balanced (differential) digital transmission line interfaces developed to incorporate and improve upon the advantages of the current-loop interface and improve on the EIA-232 limitations.

The advantages are:

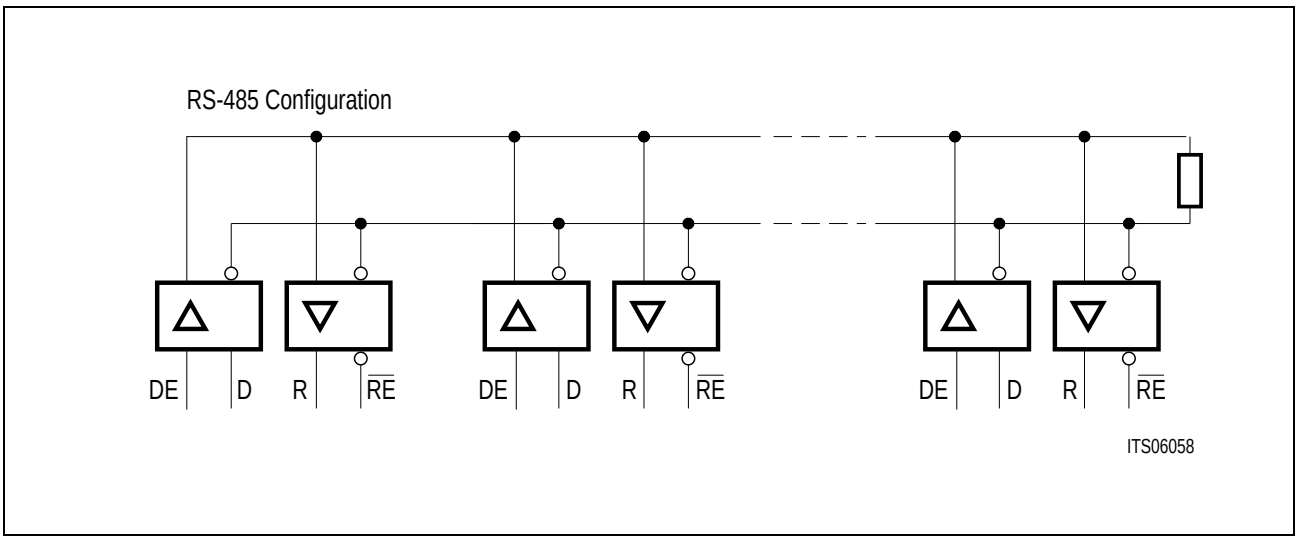
- Data rate – to 10 Mbit/s and beyond
- Longer line length – up to 1200 metres
- Differential transmission – less noise sensitive

RS422 offers a reliable multi-point one way communication. A typical application area is its use in transmitting data from a central computer to multiple remote monitors, printers or stations, such as airport arrival and departure monitors.



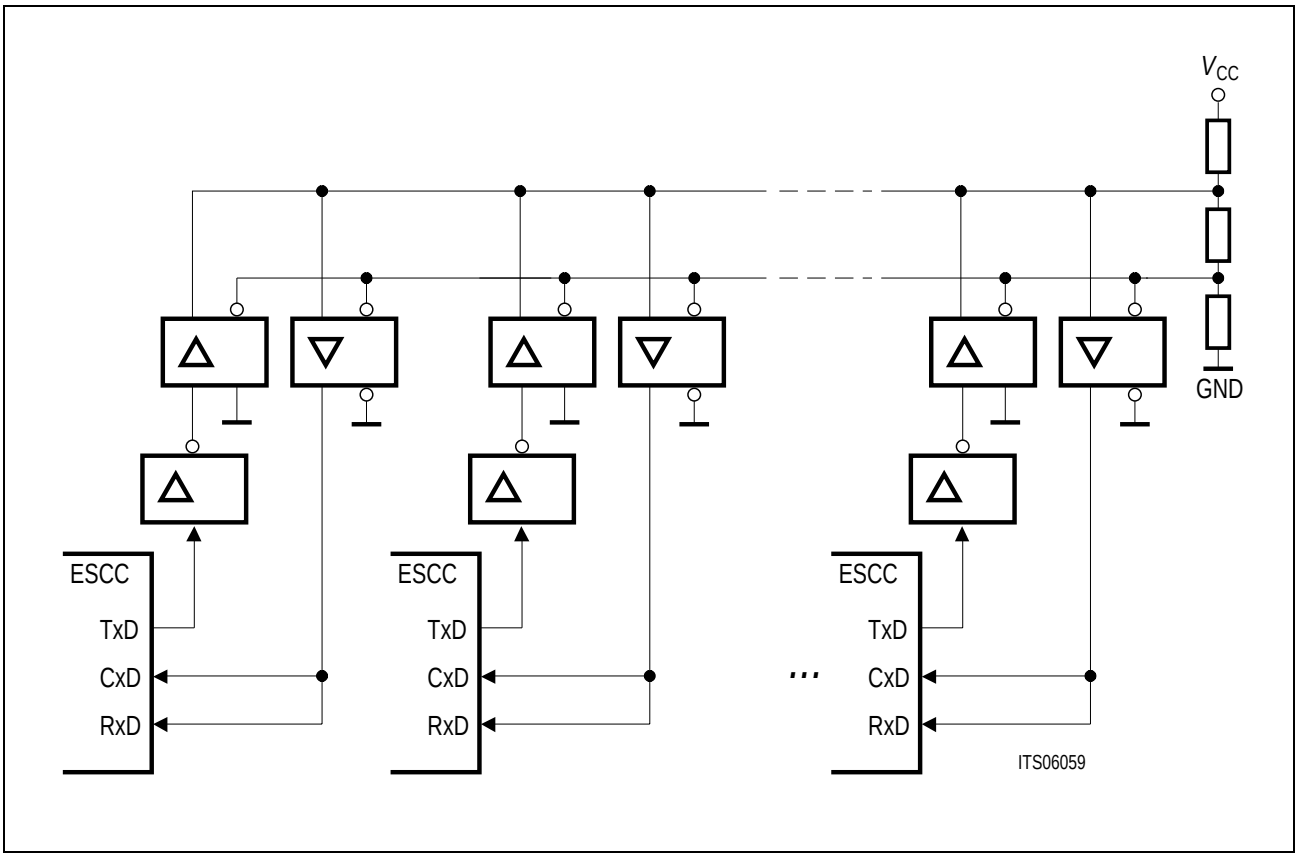
RS485 is an upgraded version of RS422 extending the number of peripherals and terminals that a computer can interface to, particularly where longer line length or increased data rates are called for. Additionally, RS485 allows for **bi-directional multi-point party line communication** and can effectively be used for “mini-LAN“, such as data transmission between a central computer and remote intelligent stations or for backplanes.

The RS485 bus can be used instead of a wired-or connection to provide half duplex communication between several stations.



In any party line interface system, with multiple driver/receivers, there will be long periods of time when the driving devices are in-active. This state known as line idle occurs when the drivers place their outputs into high impedance state. During line idle, the voltage along the line is left floating, i.e. indeterminate – neither logic high nor logic low. As a result the receiver could be falsely triggered into either a logic high or logic low state, depending upon the presence of noise and the polarity of the floating lines. This is obviously undesirable as the circuitry following the receiver could interpret this a valid information. A hard wired fail safe should provide a defined voltage across the receiver’s input. External pull-up and pull-down resistors as well as termination resistors will provide a defined voltage across the line.

Since in some clock modes the ESCC does not deactivate $\overline{\text{RTS}}$ after collision detection, do not use the $\overline{\text{RTS}}$ signal directly to enable drivers for TxD in a balanced bus configuration. Instead use an arrangement of the type shown in the figure below, based on a RS485 bus configuration.



The data input of the TX driver is tied to GND; the driver is enabled by TxD itself:

- [**TxD = 1**] => [bus = “high impedance“/1, determined by pull ups/pull downs; “**recessive**“]
- [**TxD = 0**] => [bus = 0, driven by Tx driver; “**dominant**“].

Since a zero (“low“) as a driven signal is “dominant“ on the bus, it prevails over a “recessive“ 1, that corresponds to the “high impedance“ condition and is not driven.

For more detail about line length, max. bit rate …, please refer to the data books of the specific transmission circuits and line characteristics.

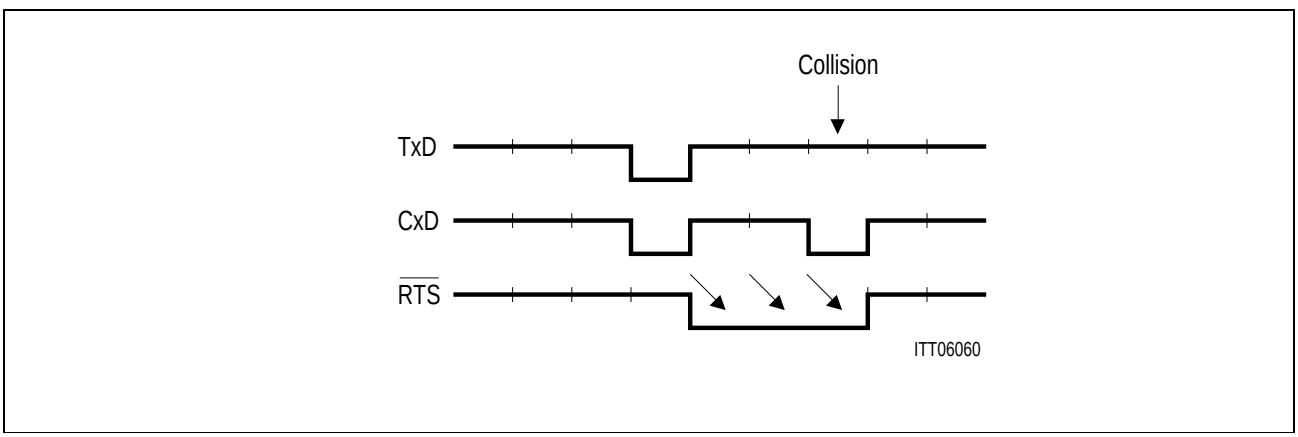
7.3.9 Functions of $\overline{\text{RTS}}$ Output

In clock modes 0, 1 and 4, the $\overline{\text{RTS}}$ output can be programmed via CCR2 to be active when data (frame or character) is being transmitted.

	7							0	
CCR2 for Clock Mode 0a, 1, 4, 5	**SOC1**	**SOC0**	**–**	**–**	**–**	**RWX**	**C32**	**DIV**	(2E)

Bus configuration		
$\overline{\text{RTS}}$ output is activated during transmission of a frame	**0**	**X**
$\overline{\text{RTS}}$ output is always high ($\overline{\text{RTS}}$ disabled)	**1**	**0**
$\overline{\text{RTS}}$ indicates the reception of a data frame (active low)	**1**	**1**

This signal is delayed by one clock period with respect to the data output TxD, and marks all data bits that could be transmitted without collision. In this way a configuration may be implemented in which the bus access is resolved on a local basis (collision bus) and where the data are sent one clock period later on a separate transmission line.



8 ESCC Evaluation Tool

8.1 Summary

This chapter gives a step-by-step introduction to the EASY532 development system for the ESCC products.

Wolfram Kiesecker
1994

8.2 An Introduction to the EASY532 Evaluation Kit

The kit is designed for the use in an MS-DOS PC.

It consists of:

- the ESCC2 Evaluation Board,
- the EASY532 software package
- and a user's manual

8.2.1 Board Installation

The ESCC2 board plugs into one PC-XT/AT extension slot. It communicates with the host PC via four configuration registers and through a dual-port RAM. The base address of the configuration registers (PC I/O address space) and the size and location of the dual-port RAM (PC memory address space) is configurable by software and with on-board DIP switches.

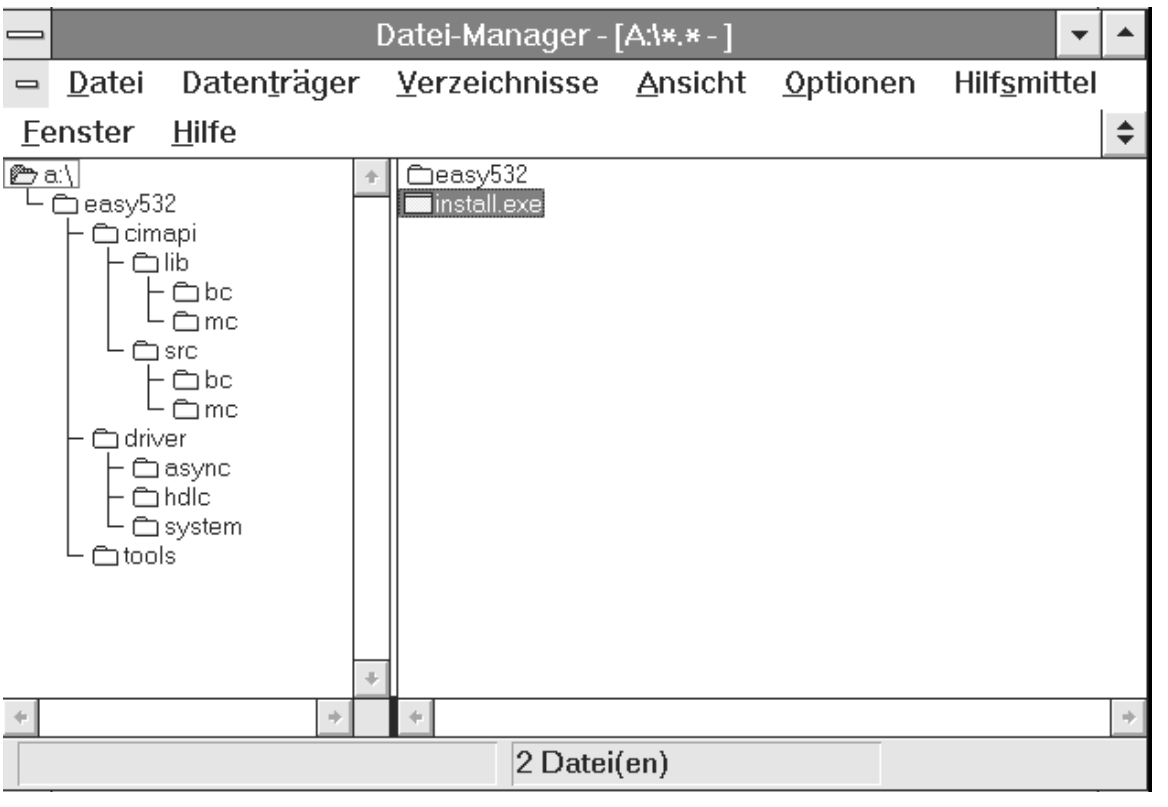
8.2.2 Software Installation

The software package comes on a 3.5" or 5.25" floppy diskette with an installation program (INSTALL.EXE). INSTALL is invoked from the floppy drive's root directory.

Example:

```
A:>install
```

will copy the EASY532 directory structure from the diskette onto the PC's hard disk.

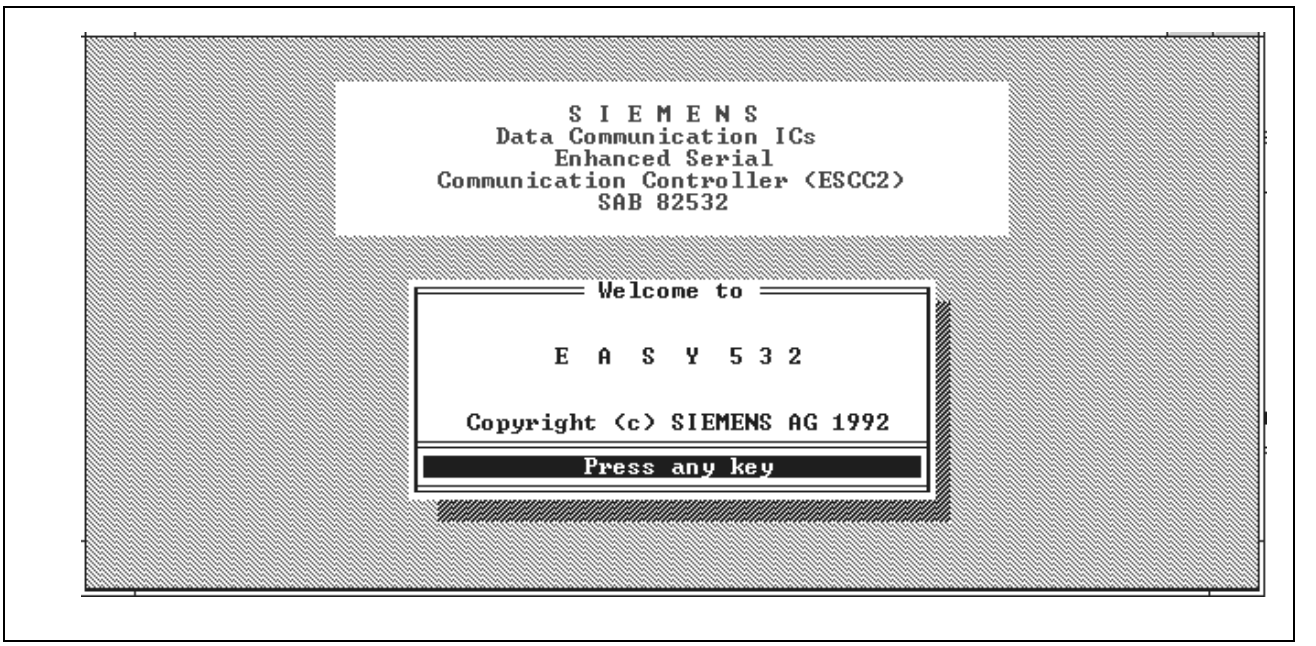


Structure of the EASY532 Software Package

8.2.3 System Configuration

Change directory to the newly created `\easy532\tools` directory on your hard drive. Invoke EASY532.EXE for the initial configuration:

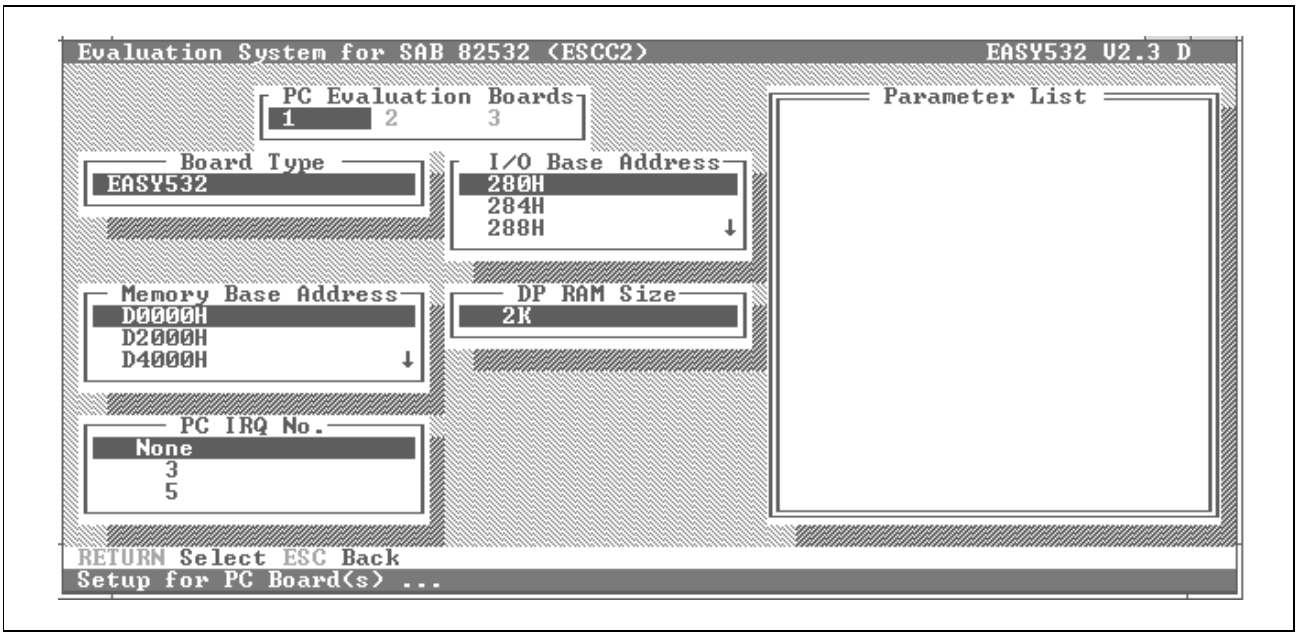
(e.g. `C:\EASY532\TOOLS>EASY532`)



Screen 1
EASY532 Greeting

Press any key. Then select the system parameters step by step:

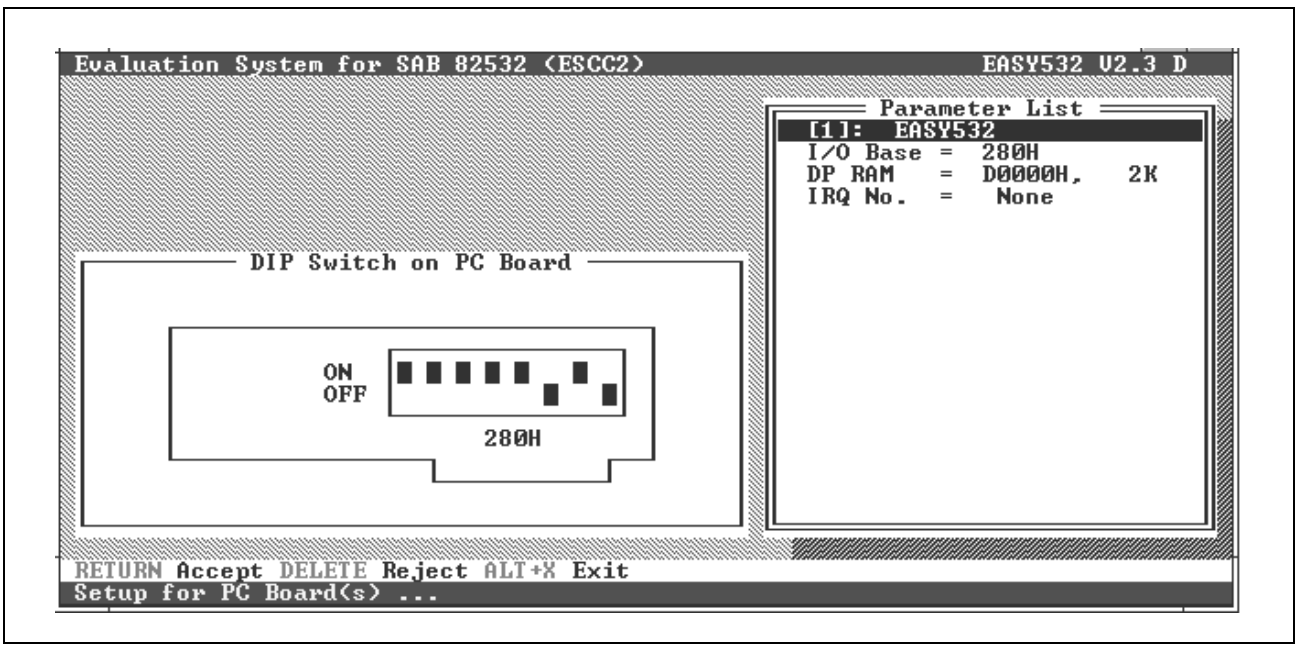
- number of evaluation boards (typically 1)
- Board Type (EASY532)
- I/O Base Address default = 280H)
- Memory Base Address (default = D0000H)
- Dual Port RAM Size typically = 2K)
- PC IRQ No. (default = None)



Setup Screen 1
EASY532 System Parameter Selection

(Screen #2 will come up if EASY532.CFG does not exist in your current directory. For re-configurations EASY532.CFG can be deleted before invoking EASY532.EXE)

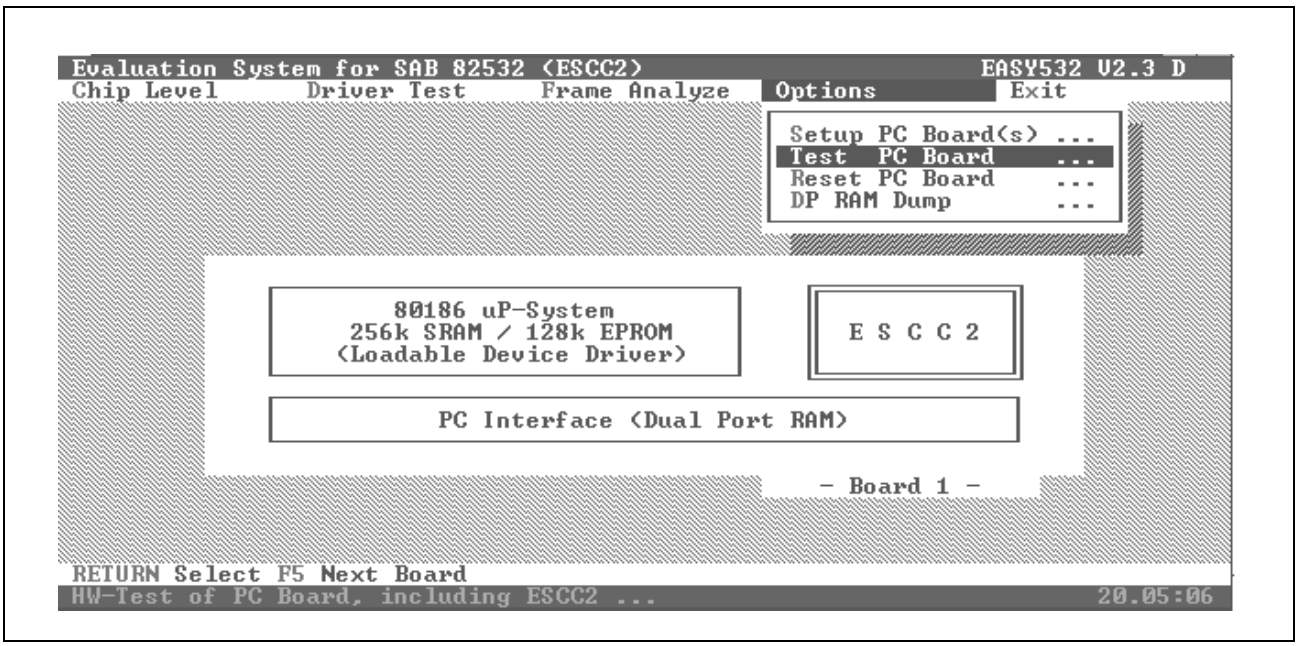
The next Screen (#3) shows the parameter list and the appropriate DIP switch settings on the ESCC2 evaluation board. Please make sure, that your board is configured accordingly.



Setup Screen 2
Parameter List and DIP Switch Settings

This screen completes the system configuration. EASY532.CFG is created in your current directory. Instead of going through the system setup every time, EASY532 will read its configuration file and come up with the main screen directly:

8.2.4 EASY532 Session

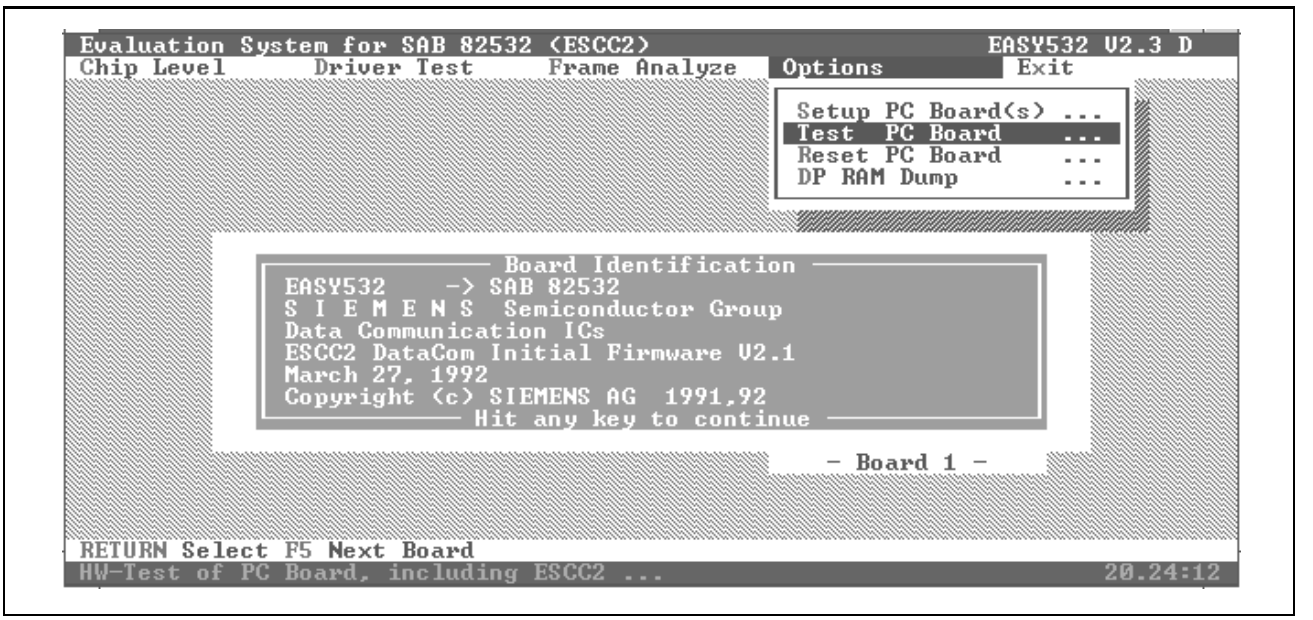


Screen 2
Main Screen of the EASY532 Evaluation Tool

8.2.5 System Test

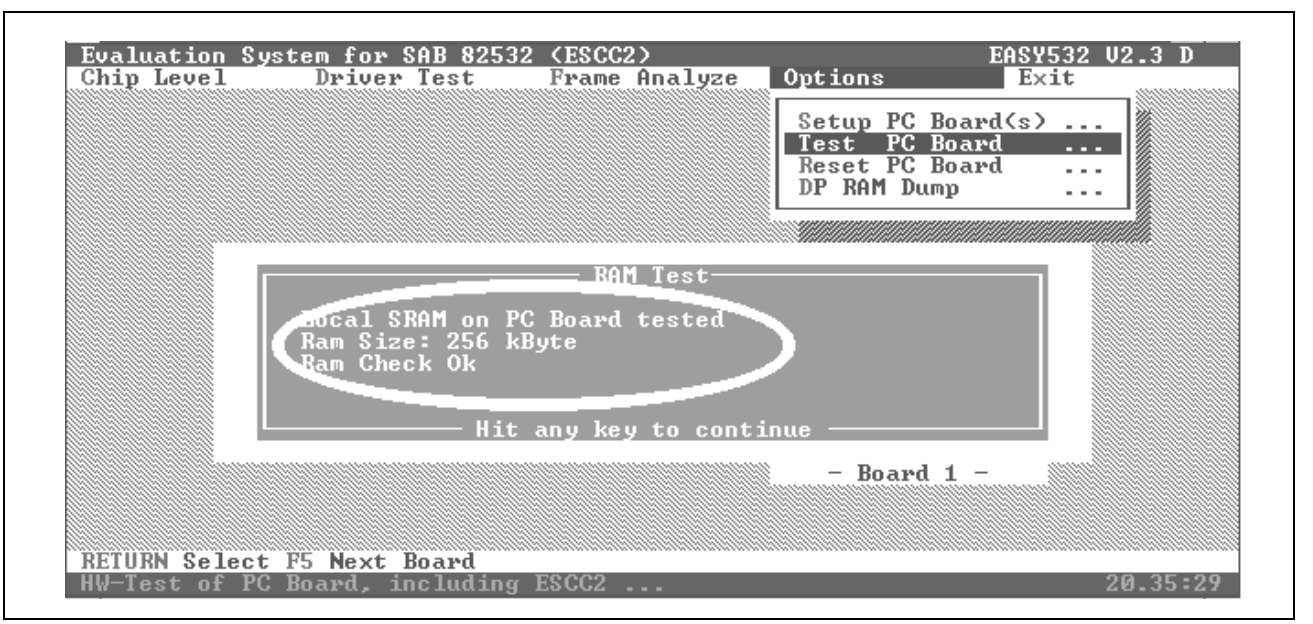
EASY532 offers a self-test feature under the *Options* menu item. If software and hardware are working properly *Test PC Board* displays system information on three screens:

- the firmware version implemented on the ESCC2 board:



Test Screen 5
Firmware Version

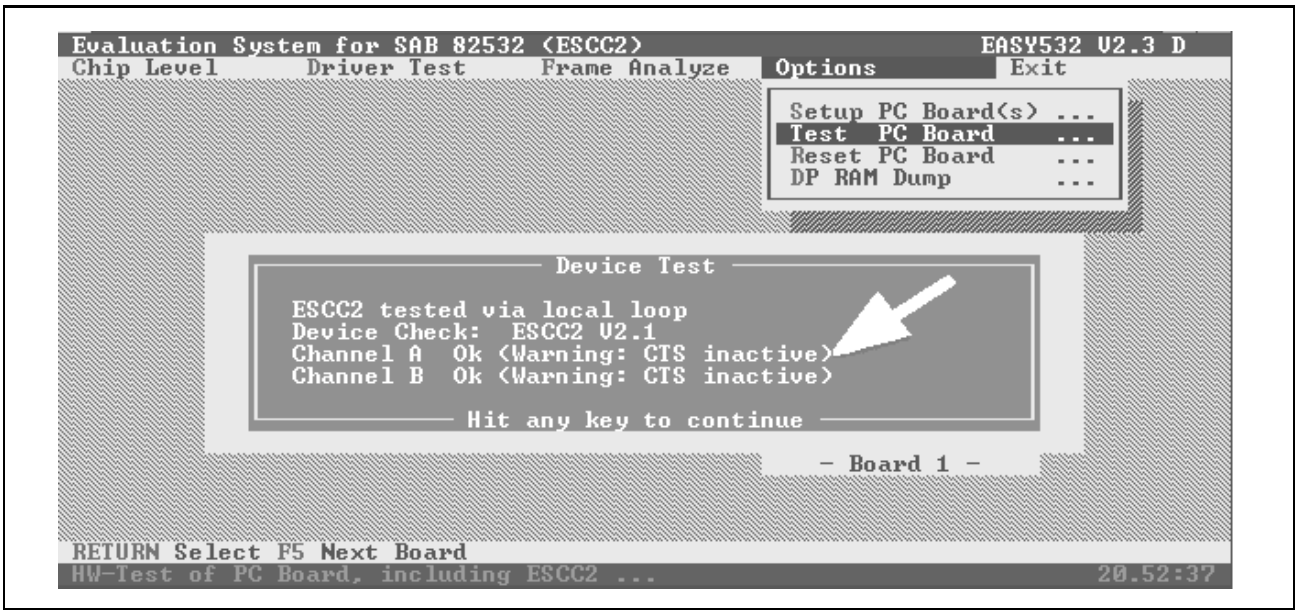
- the size of the on-board SRAM and the RAM check result,



Test Screen 6
SRAM Size and Check Result

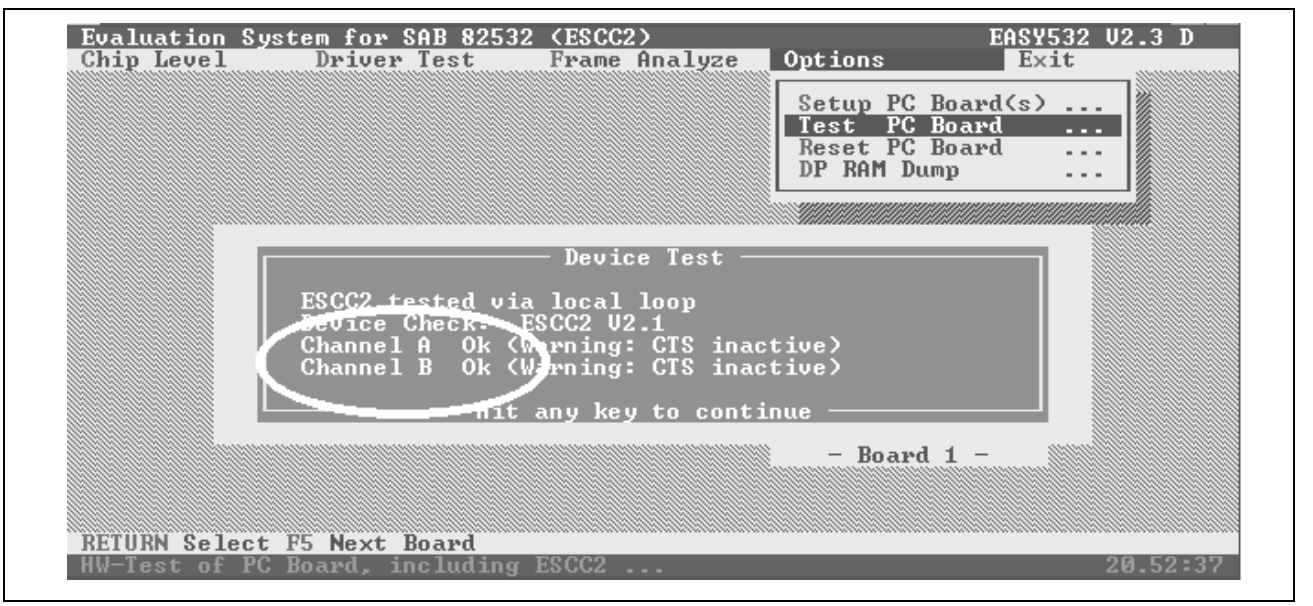
- the version of the ESCC2 and the communication status of the two channels

Note: EASY532 will show “ESCC2 V2.1”, even if the board has the ESCC2 V2.2 installed!



Test Screen 7
ESCC Version and Channel Communication Status, CTS Warning!

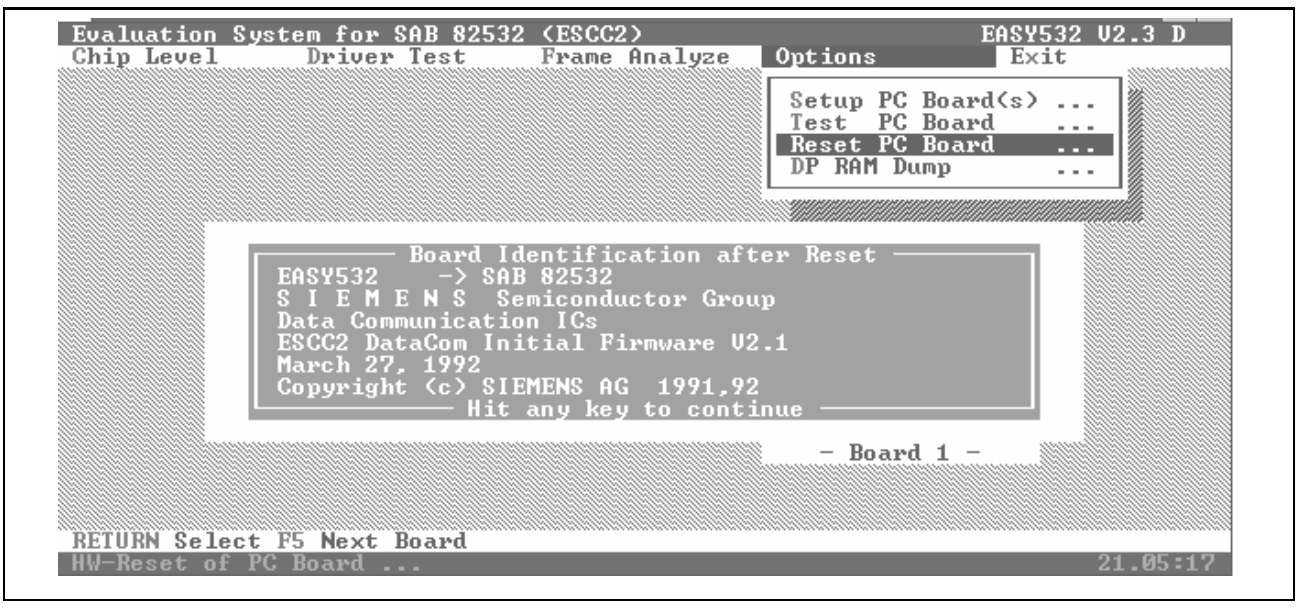
If the CTSA or CTSB input is inactive (high) a warning is displayed. The ESCC2 cannot transmit as long as CTS is inactive (not even in a test loop with MODE.TLP = 1). The warning disappears if the CTSx inputs of the board's communication connector are pulled low (CTS active):



Test Screen 3
Communication Status Okay, No CTS Warning

8.2.6 Hardware Reset

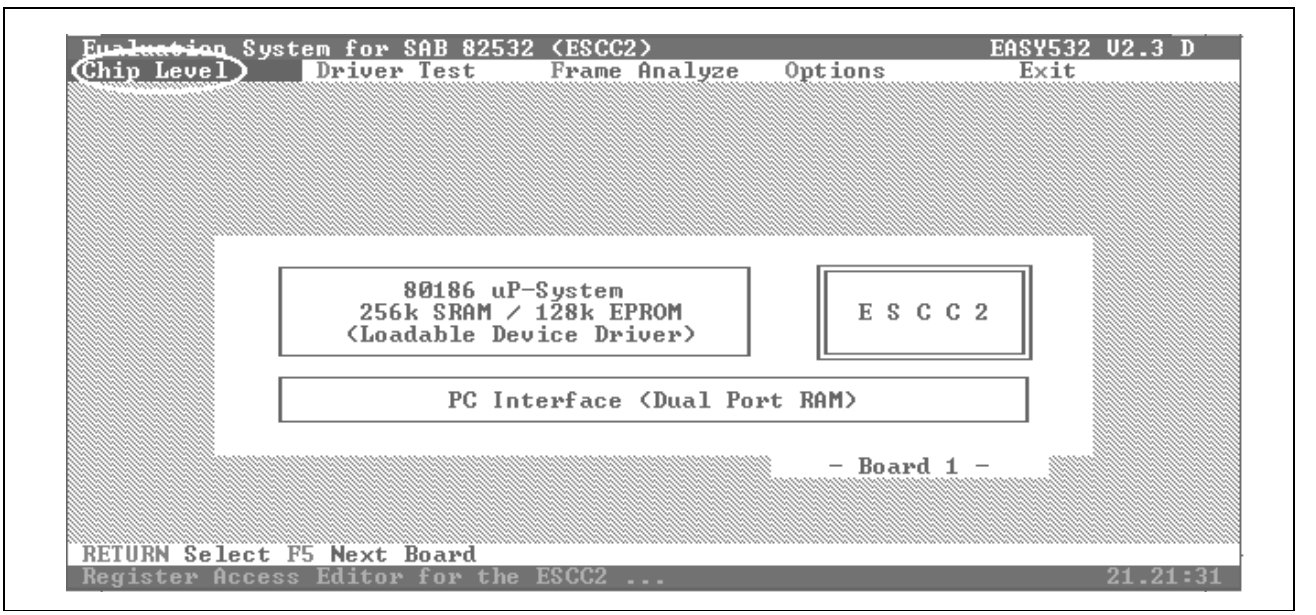
The ESCC2 board's reset button is located above the communication connector. The ESCC2 can also be reset (hardware reset!) from the *Options* menu with *Reset PC Board*. A hardware reset forces all ESCC2 registers into their initial state (reset value).



8.2.7 *Chip Level* Evaluation Session

Register programming of the ESCC2 is very uncomplicated with EASY532. The user can operate the ESCC2 in all basic functions without having to write a single line of code!

From the main screen select *Chip Level*:



Main Screen

The Chip Level screen is divided into three windows:

- the register map of the selected channel,
- the content and the bit structure of the selected register,
- a log window. Every action is saved for later review.

Chip Level Screen

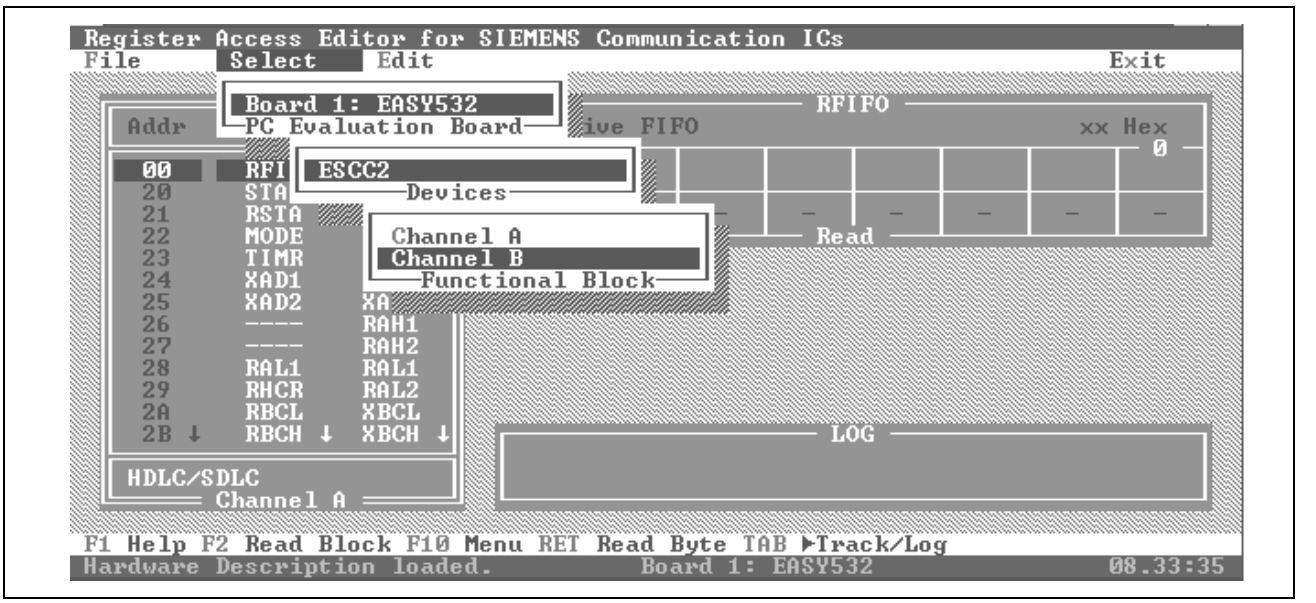
The menu items *File, Select, Edit* and *Exit* can be reached with <F10> and the cursor keys, or with hot-keys directly:

File	<Alt-f>
Select	<Alt-s>
Edit	<Alt-e>
Exit	<Alt-x>

8.2.8 Example Session

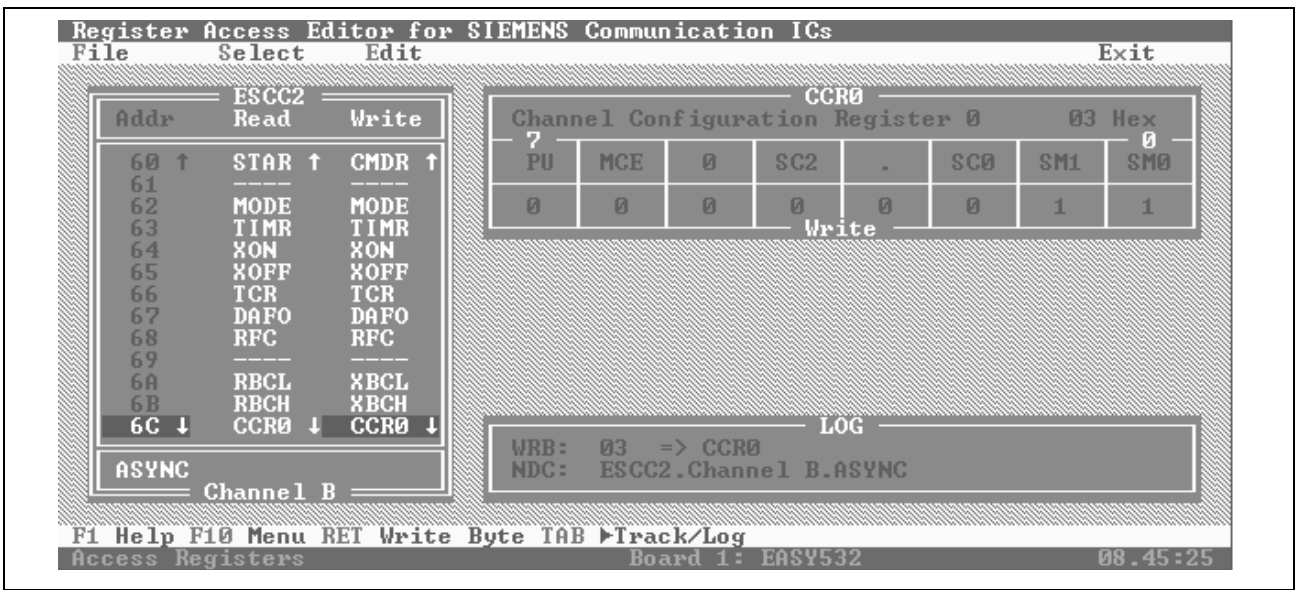
1. Select ESCC2 Channel

Press <Alt-s>, select *Board 1, ESCC2, Channel B*
Hit <Enter>. (watch the log window.)



2. Select the *CCR0* Register for *Write* Access and Program SM1…SM0 = 11 (ASYNC mode)

Move cursor right to the *Write* column and down to the *CCR0* register.
Hit <Enter> (the cursor jumps into the structure window)
Type “03” as hex value, or move cursor to the SM1, SM0 field and toggle the bit with <space>.
Hit <Enter> again. (Watch the log window)



3. Minimum Initialization for Asynchronous Communication in a Test Loop:

Repeat step 2. for CCR1, CCR2, CCR0, MODE, XFIFO and CMDR:

```
03 =>     CCR0     ; select ASYNC mode (step from above)
0F =>     CCR1     ; clock mode 7, async
10 =>     CCR2     ; clock mode extension b, baud rate div. factor = 1
83 =>     CCR0     ; power up, (ASYNC)
09 =>     MODE     ; receiver active, test loop
55 =>     XFIFO    ; one single character in transmit FIFO
08 =>     CMDR     ; transmit command
```

4. Retrieve Data from RFIFO:

The receive FIFO content can be read by selecting *RFIFO* from the register map and reading it with <Enter>

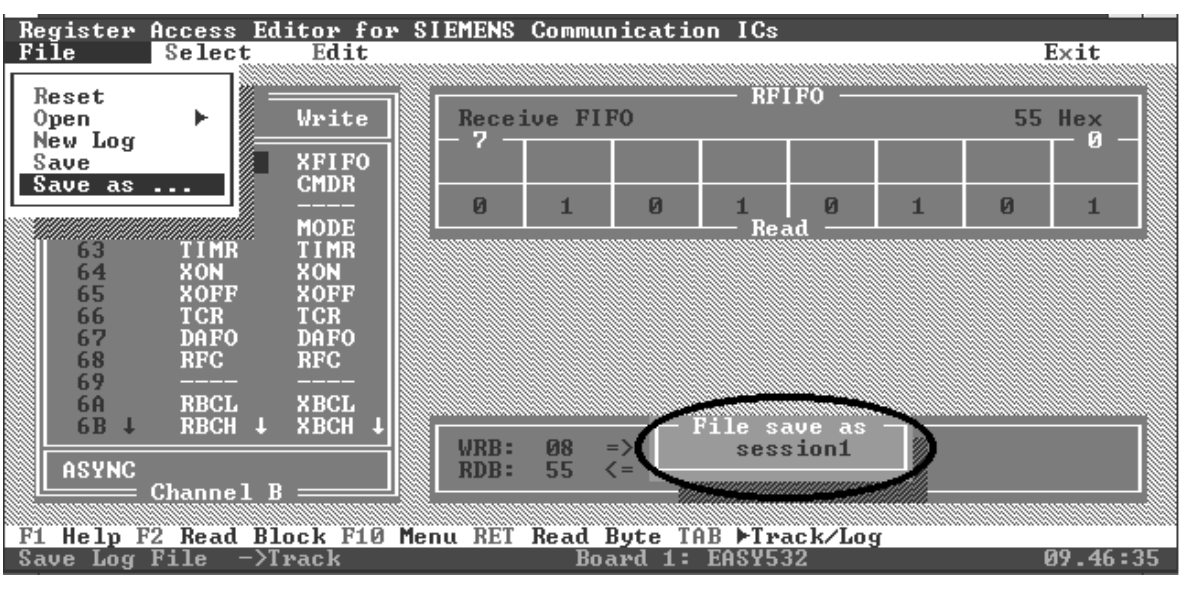
```
55 <=     RFIFO    ; data received via test loop
80 =>     CMDR     ; Acknowledge: Receive Message complete!
```

5. Save the Whole Session Onto the Hard Disk

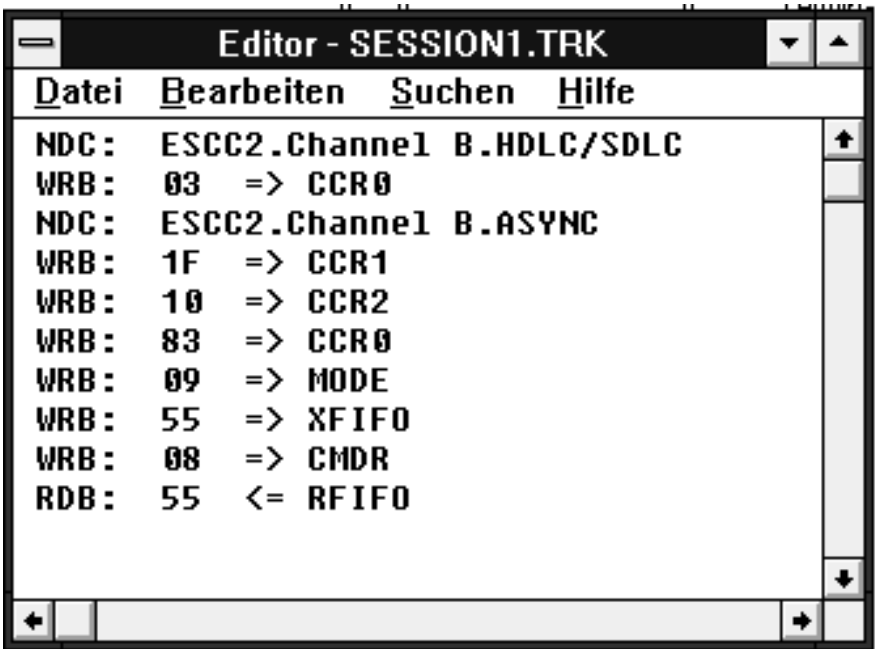
The content of the log window can be saved by selecting *File* (<Alt-f>) *Save_as*. The file format is ASCII.

Example:

Save the log window content as “SESSION1.TRK”:
Press <Alt-f>, select *Save_as*…, <Enter>, type `SESSION1`, hit <Enter>.



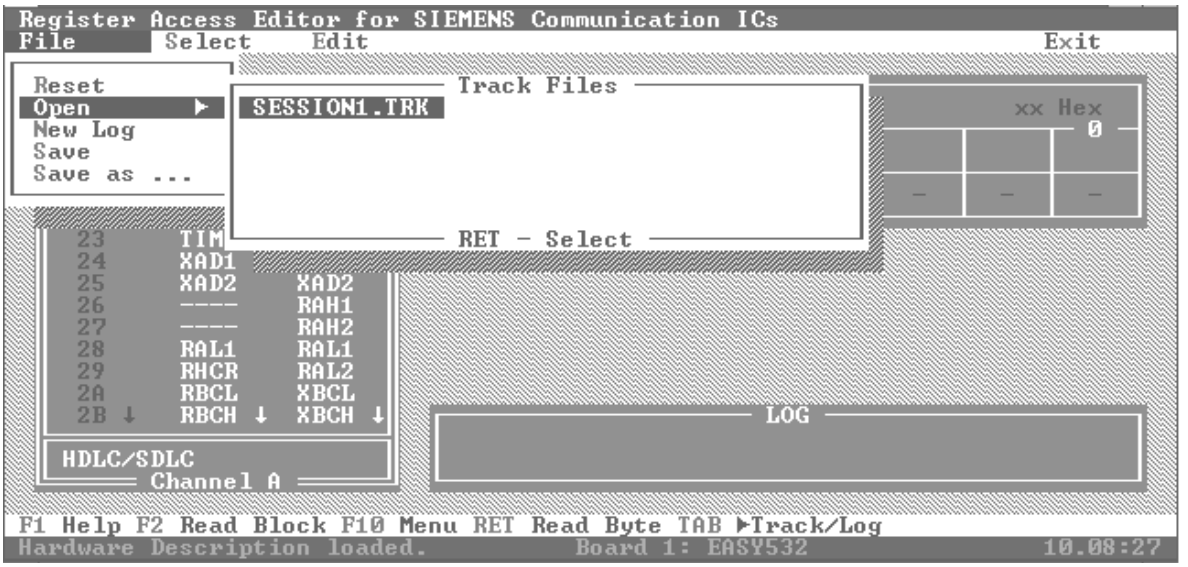
The log window content is saved as a *Track File* with the extension ".TRK". Since track files are saved in ASCII format, they can be viewed and modified easily, with any text editor (like MS Windows Notepad):



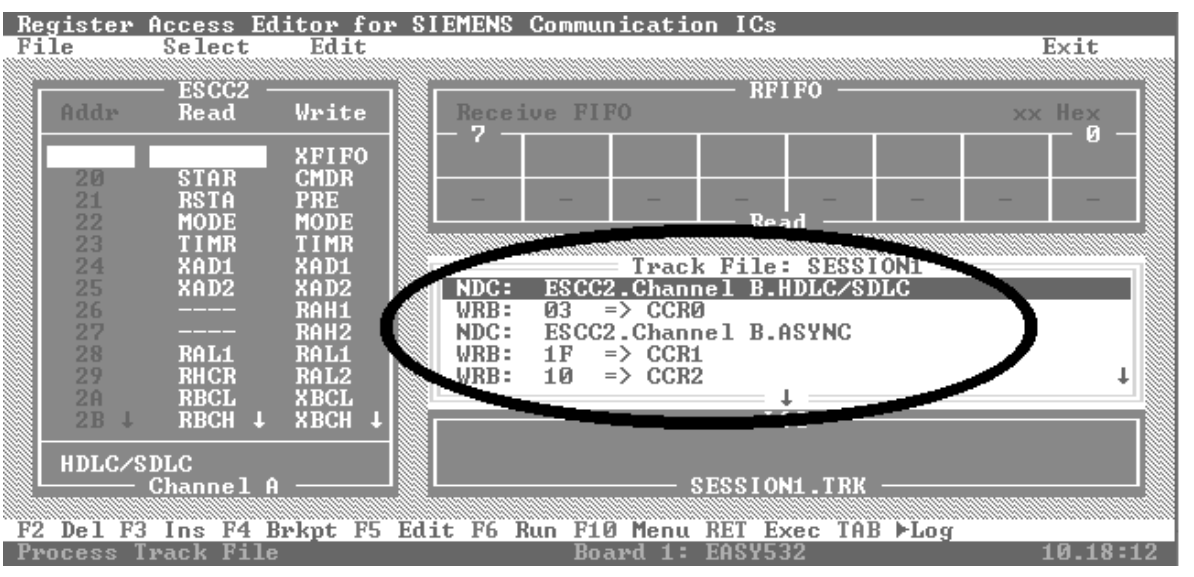
Track File
SESSION1.TRK

EASY532 can load track files and run them again:

Select *File* (<Alt-F>), *Open*, and the track file.



A new *Track File Window* appears on the Chip Level screen.



Within the *Track File* window lines can be selected (Cursor Up/Down), deleted (<F2>), inserted (<F3>), edited (<F5>), and executed (<Enter>). Breakpoints are set at the cursor position with <F4>. <F6> starts the execution of the track file from the cursor position on.

Example

1. Reset the ESCC2 and Re-Load SESSION1.TRK

Exit *Chip Level* (<Alt-x>)
On the main screen, select *Options, Reset PC Board* (<Alt-o>, <Alt-r>, <Enter>)
Go back to the *Chip Level* screen (<any key>, <Alt-c>, <Enter>)
Load track file (<Alt-f>, <Alt-o>, <Enter>, select file name, <Enter>)

2. Insert Comments

Make sure cursor is at the first line, hit <F3> (# identifies a comment line)
Type: `Channel B initialization for ASYNC mode`, <Enter>
Move cursor to line: `WRB: 55 => XFIFO`, hit <F3>
Type: `Transmit data 0x55`, hit <Enter>
Move cursor to line: `RDB: 55 <= RFIFO`, hit <F3>
Type: `Read received data and acknowledge`, hit <Enter>

3. Execute Track File

Clear the log window with <Alt-f>, <Alt-n>, <Enter>
Move cursor to the top of the file.
Hit <F6>

4. Send Another Byte (without re-initializing)

Move cursor to line: `WRB: 08 => CMDR`, set break point with <F4>
Move cursor to line: `# Transmit data 0x55`, execute 2 lines with <F6>

5. Read RFIFO and Acknowledge

Execute the remaining 2 lines with <F6>
Note, that these two lines are appended to the log window content!

6. Save the New Session (log window!) as SESSION2

<Alt-f>, *Save as*, <Enter>, SESSION2, <Enter>

7. Examine SESSION2.TRK

Track File SESSION2.TRK

```
# Channel B initialization for ASYNC mode
NDC:  ESCC2.Channel B.HDLC/SDLC
WRB:  03 => CCR0
NDC:  ESCC2.Channel B.ASYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  83 => CCR0
WRB:  09 => MODE
# Transmit data 0x55
WRB:  55 => XFIFO
WRB:  08 => CMDR
# Read received data and acknowledge
RDB:  55 <= RFIFO
WRB:  80 => CMDR
# Transmit data 0x55
WRB:  55 => XFIFO
WRB:  08 => CMDR
# Read received data and acknowledge
RDB:  55 <= RFIFO
WRB:  80 => CMDR
```

8.2.9 Interrupts

EASY532 indicates ESCC2 interrupts in the lower right corner of the *Chip Level* screen. ESCC2 interrupts are enabled by unmasking (“0”) the interrupt source bit in the interrupt mask registers IMR0 and IMR1.

Example

1. **Reset the ESCC2**

   (from the main screen <Alt-o>, <Alt-r>, <Enter>)

2. **Load SESSION2.TRK**

   (from the *Chip Level* screen <Alt-f>, <o>, <Enter>, select)

3. **Initialize Channel B in ASYNC Mode**

   (e.g.: single step with <Enter> until cursor reaches line:
   `# Transmit data 0x55)`.

4. **Change to the Register Map Window** (with <Tab>, <Tab>)

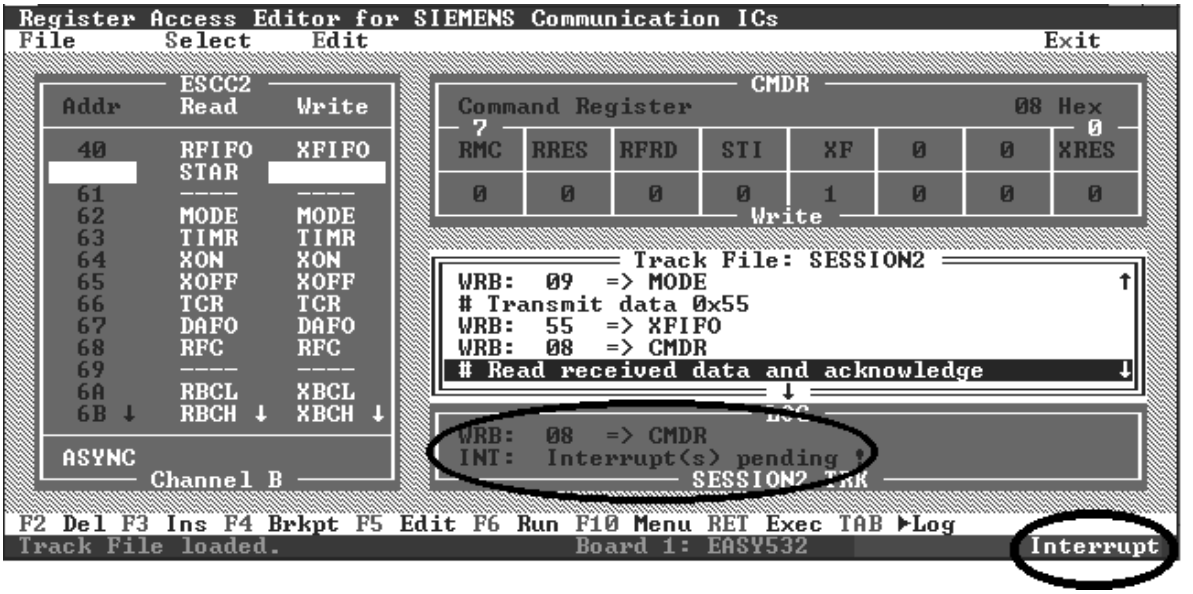
5. **Select IMR0 with the Cursor Keys and Unmask the RPF bit** (FE => IMR0).

6. **Change to the Track File Window** (with <Tab>).

7. **Single Step Through the Next 3 Lines** (with <Enter>):

```
# Transmit data 0x55
WRB:   55 => XFIFO
WRB:   08 => CMDR
```

8. **Watch the Interrupt Flag in the Lower Right Corner and the Indication in the Log Window.**



9. **Change to the Register Map Window** (with <Tab>, <Tab>).

10. **Select GIS (read column) with the Cursor Keys and Read with <Enter>**

11. **The ISB0 Bit is Set, which Points to the ISR0 Register of Channel B.**

12. **Read ISR0, the RPF Bit is Set, Indicating that Data is Available from the RFIFO.**

13. **The `Interrupt` Flag in the Lower Right Corner is Cleared**
    (Interrupt acknowledged).

14. **Change to the Track File Window and Continue Single Stepping:**

```
# Read received data and acknowledge
RDB:   55 <= RFIFO
WRB:   80 => CMDR
```

15. **Repeat Steps 7. through 14.**

16. **Save the Log Window as SESSION3.**

17. Examine SESSION3.TRK.

The manually entered commands – write IMR0, read GIS, ISR0 – are inserted between the commands from SESSION2.TRK

Track File SESSION3.TRK

```
NDC:  ESCC2.Channel B.HDLC/SDLC
# Channel B initialization for ASYNC mode
WRB:  03 => CCR0
NDC:  ESCC2.Channel B.ASYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  83 => CCR0
WRB:  09 => MODE
WRB:  FE => IMR0
# Transmit data 0x55
WRB:  55 => XFIFO
WRB:  08 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  55 <= RFIFO
WRB:  80 => CMDR
# Transmit data 0x55
WRB:  55 => XFIFO
WRB:  08 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  55 <= RFIFO
WRB:  80 => CMDR
```

8.2.10 Block Read/Write

When RFIFO or XFIFO is selected in the register map window, <F2> opens a small data window for up to 32 bytes for block access to the FIFOs.

Example

1. **Reset the ESCC2**

2. **Load SESSION3.TRK**

3. **Single Step to Line: `# Transmit Data 0x55`.**

4. **Select XFIFO in the Register Map Window**
   (Note: Do not press <Enter>!:

5. **Press <F2>**

6. **Enter:**

   11<Tab>22<Tab>33<Tab>44<Tab>55<Enter>.

7. **Go to Track File Window.**

8. **Skip 2 Lines:**

   (use the cursor-down key:)

   ```
   # Transmit data 0x55
   WRB:   55 => XFIFO
   ```

9. **Execute:**

   ```
   WRB:   08 => CMDR
   ```

10. **Watch the Interrupt Flag.**

    ```
    INT:   Interrupt(s) pending !
    ```

11. Acknowledge the Interrupt and Read Data:

(single step through the track file)

```
RDB:   01 <= GIS
RDB:   01 <= ISR0
INT:   Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:   11 <= RFIFO
WRB:   80 => CMDR
```

12. Another Interrupt is Now Pending !

```
INT:   Interrupt(s) pending !
```

13. Acknowledge Interrupt and Read Data:

```
RDB:   01 <= GIS
RDB:   01 <= ISR0
INT:   Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:   22 <= RFIFO, 33, 44, 55
WRB:   80 => CMDR
```

14. Repeat Step 12 for every New Interrupt.

15. Save Log Window Content as SESSION4.

16. Examine the Track File:

Track File: SESSION4.TRK

```
NDC:  ESCC2.Channel B.HDLC/SDLC
# Channel B initialization for ASYNC mode
WRB:  03 => CCR0
NDC:  ESCC2.Channel B.ASYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  83 => CCR0
WRB:  09 => MODE
WRB:  FE => IMR0
WRD:  Data Block  => XFIFO
      11 22 33 44 55
WRD:   5 Bytes    => XFIFO
WRB:  08 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
```

```
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  11 <= RFIFO       ! 55
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  22 <= RFIFO       ! 55
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  33 <= RFIFO       ! 55
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  44 <= RFIFO       ! 55
WRB:  80 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
# Read received data and acknowledge
RDB:  55 <= RFIFO
WRB:  80 => CMDR
```

Currently, the threshold of the RFIFO is 1 byte (reset value of RFC = 0x00). The next example will initialize the RFC register and set the RFIFO threshold to 32 bytes:

Example

1. **Reset the ESCC2**

2. **Load SESSION4.TRK**

3. **Single Step Through:**

```
NDC:   ESCC2.Channel B.HDLC/SDLC
# Channel B initialization for ASYNC mode
WRB:   03 => CCR0
NDC:   ESCC2.Channel B.ASYNC
WRB:   0F => CCR1
WRB:   10 => CCR2
```

4. **Enter from the Register Map Window:**

```
WRB:   0C => RFC
```

5. **Continue Single Stepping (track file window):**

```
WRB:   83 => CCR0
WRB:   09 => MODE
WRB:   FE => IMR0
WRD:    Data Block  => XFIFO
       11 22 33 44 55
WRD:   5 Bytes      => XFIFO
WRB:   08 => CMDR
```

6. **There is No Interrupt !!**

   (no flag, no indication in the log window, - track file window is a recording!)

7. **Block Write 27 Bytes Into the XFIFO (from register map window)**

   Select XFIFO, hit <F2>, enter 27 bytes separated by <Tab> and terminated with <Enter>.

8. **Issue XF Command.**

```
WRB:   08 => CMDR
```

9. **Interrupt!**

10. Execute

```
INT:   Interrupt(s) pending !
RDB:   01 <= GIS
RDB:   01 <= ISR0
INT:   Interrupt(s) acknowledged
```

11. Move the Cursor to RFIFO in the Register Map Window.

12. Press <F2> for a Block Read.

The first 5 bytes 11, 22, 33, 44, 55 and the next 27 bytes are found as one block in the RFIFO

13. Acknowledge RFIFO Read

```
WRB:   80 => CMDR
```

14. Save as SESSION5

15. Examine SESSION5.TRK:

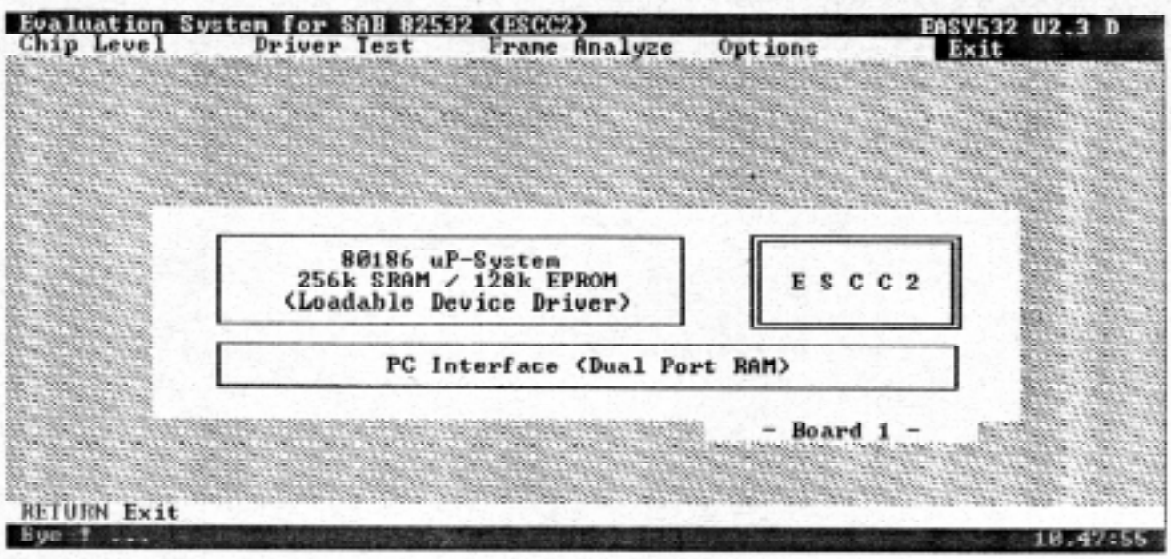
Track File: SESSION5.TRK

```
NDC:  ESCC2.Channel B.HDLC/SDLC
# Channel B initialization for ASYNC mode
WRB:  03 => CCR0
NDC:  ESCC2.Channel B.ASYNC
WRB:  0F => CCR1
WRB:  10 => CCR2
WRB:  0C => RFC
WRB:  83 => CCR0
WRB:  09 => MODE
WRB:  FE => IMR0
WRD:  Data Block  => XFIFO
      11 22 33 44 55
WRD:   5 Bytes    => XFIFO
WRB:  08 => CMDR
WRD:  Data Block  => XFIFO
      01 02 03 04 05 06 07 08
      09 0A 0B 0C 0D 0E 0F 10
      11 12 13 14 15 16 17 18
      19 1A 1B
WRD:  27 Bytes    => XFIFO
WRB:  08 => CMDR
INT:  Interrupt(s) pending !
RDB:  01 <= GIS
```

```
RDB:  01 <= ISR0
INT:  Interrupt(s) acknowledged
RDD:  Data Block  <= RFIFO
      11 22 33 44 55 01 02 03
      04 05 06 07 08 09 0A 0B
      0C 0D 0E 0F 10 11 12 13
      14 15 16 17 18 19 1A 1B
RDD:  32 Bytes    <= RFIFO
WRB;  80 => CMDR
```

8.2.11 Exit EASY532

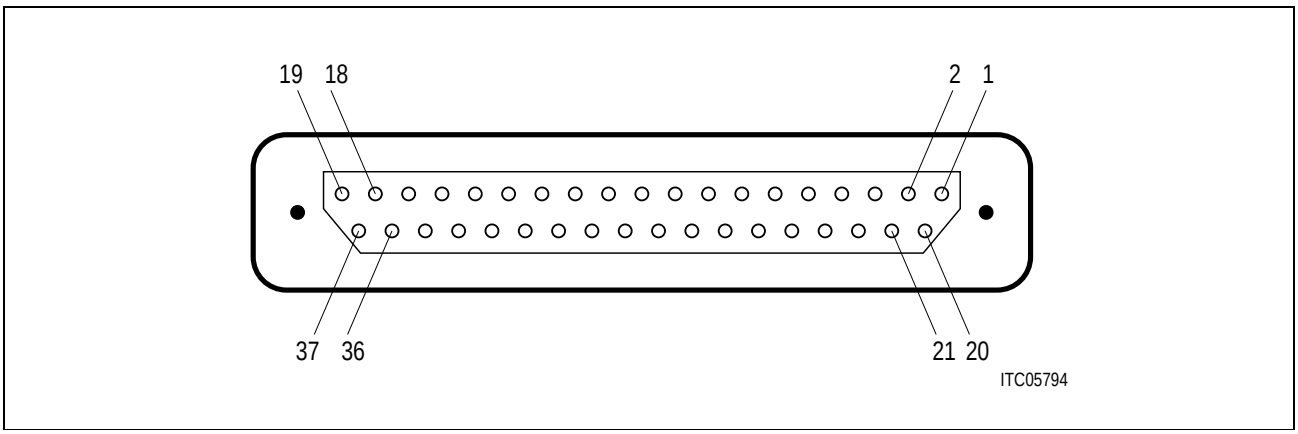
The EASY532 session is terminated from the main screen by selecting the *Exit* menu item or with the <Alt-x> hot-key.



The Last Screen
Exit … Bye!

8.3 Appendix

8.3.1 User Connector Interface of the EASY532 Board



Pin	Signal	Type	Function
1	V_{SS}	GND	
2	V_{DD}	+ 5 V	
3	+ 12 V	+ 12 V	
4	– 12 V	– 12 V	
5	N.C.		
6	N.C.		
7	P_RESET*	I	ESCC2 Reset
8			
9			
10	P7	I/O	Parallel port pin 7
11	P6	I/O	Parallel port pin 6
12	P5	I/O	Parallel port pin 5
13	P4	I/O	Parallel port pin 4
14	P3	I/O	Parallel port pin 3

Pin	Signal	Type	Function
15	P2	I/O	Parallel port pin 2
16	P1	I/O	Parallel port pin 1
17	P0	I/O	Parallel port pin 0
18	V_{SS}	GND	
19	V_{SS}	GND	
20	V_{SS}	GND	
21	RxCLKB	I	Receive clock of channel B
22	CDB	I	Carrier detect of channel B
23	CTSB*/CxDB	I	Clear to send/collision data of channel B
24	RTSB*	O	Request to send of channel B
25	RxDB	I/O	Receive data of channel B
26	TxDB	I/O	Transmit data of channel B
27	TxCLKB	I/O	Transmit clock of channel B
28	P_INT*	O	ESCC2 interrupt
29			
30	N.C.		
31	RxCLKA	I	Receive clock of channel A
32	CDA	I	Carrier detect of channel A
33	CTSA*/CxDA	I	Clear to send/collision data of channel A
34	RTSA*	O	Request to send of channel A
35	RxDA	I/O	Receive data of channel A
36	TxDA	I/O	Transmit data of channel A
37	TxCLKA	I/O	Transmit clock of channel A

8.3.2 ASCII-Code Table

		Bits 6…4							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Bits 0…3	**0**	NUL	DLE	SP	0	@	P	‘	p
	1	SOH	XON	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	XOFF	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	’	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	|
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	_	o	DEL