Cologne Chip

HFC - E1

ISDN HDLC FIFO controller
with
Primary Rate Interface (E1)

Data Sheet ✦ October 2007

Revision History of HFC-E1 Data Sheet

Date	Remarks
October 2007	Minor changes were made in this data sheet revision: Information added to Section 2.2.4 concerning asynchronous register read accesses.
August 2006	Main changes in this data sheet revision: Usage of VDD_E1 pins #129, #147 and #164 changed (description in pin list and schematic 5.6), schematic of E1 receiver circuitry corrected, details to PCM data flow description and to PCM timing diagrams added.
March 2006	Information added to data flow programming concerning DTMF function, external SRAM characteristics and FIFO loop implementation.
March 2005	Main changes in this data sheet revision: Information added to SPI interface desciption, clock and interrupt sections. PCM chapter revised. GPI[20] does no longer exist.
March 2004	Main changes in this data sheet revision: Some register and bitmap names have changed (see Appendix B for details), PCM-to-PCM data flow example added, E1 interface chapter completely revised and strongly expanded, restrictions are described on external RAM access.

Cologne Chip AG
Eintrachtstrasse 113
D - 50668 Köln

Germany
Tel.: +49 (0) 221 / 91 24-0
Fax: +49 (0) 221 / 91 24-100

http://www.CologneChip.com
http://www.CologneChip.de
support@CologneChip.com

Contents

List of Figures

List of Tables

List of Registers

Please note !

Register addresses are assigned independently for write and read access; i.e. in many cases there are different registers for write and read access with the same address. Only registers with the same meaning and bitmap structure in both write and read directions are declared to be read / write.

It is important to distinguish between *registers*, *array registers* and *multi-registers*.

Array registers have multiple instances and are indexed by a number. This index is either the FIFO number (R_FIFO with 13 indexed registers) or the PCM time slot number (R_SLOT with 2 indexed registers). Array registers have equal name, bitmap structure and meaning for every instance.

Multi-registers have multiple instances, too, but they are selected by a bitmap value. With this value, different registers can be selected with the same address. Multi-register addresses are 0x15 (14 instances selected by R_PCM_MD0) and 0x0F (2 instances selected by R_FIFO_MD) for HFC-E1. Multi-registers have different names, bitmap structure and meaning for each instance.

The first letter of array register names is 'A_ ...' whereas all other registers begin with 'R_ ...'. The index of array registers and multi-registers has to be specified in the appropriate register.

Registers sorted by name

Please note !

See explanation of register types on page 19.

Write only registers:

Address	Name	Reset group	Page
0xF4	A_CH_MSK[FIFO]	0, 1	170
0xFC	A_CHANNEL[FIFO]	0, 1	174
0xFA	A_CON_HDLC[FIFO]	0, 1	171
0xD1	A_CONF[SLOT]	–	269
0x84	A_FIFO_DATA0_NOINC[FIFO]		169
0x84	A_FIFO_DATA1_NOINC[FIFO]		169
0x84	A_FIFO_DATA2_NOINC[FIFO]		169
0xFD	A_FIFO_SEQ[FIFO]	0, 1	175
0x0E	A_INC_RES_FIFO[FIFO]	–	167
0xFF	A_IRQ_MSK[FIFO]	0, 1	306
0xD0	A_SL_CFG[SLOT]	0, 3	254
0xFB	A_SUBCH_CFG[FIFO]	0, 1	173
0x1B	R_BERT_WD_MD	0, 1	286
0x02	R_BRG_PCM_CFG	H	302
0x00	R_CIRM	H	99
0x18	R_CONF_EN	0, 2	268
0x01	R_CTRL	H	100
0x1D	R_DTMF_N	0	280
0x1C	R_DTMF	0	279
0x20	R_E1_WR_STA	0, 1, 3	202
0x0D	R_FIFO_MD	H	166
0x0F	R_FIFO	0, 1	168
0x0B	R_FIRST_FIFO	0, 1	165
0x0F	R_FSM_IDX	0, 1	168
0x42	R_GPIO_EN0	0	325
0x43	R_GPIO_EN1	0	326
0x40	R_GPIO_OUT0	0	324
0x41	R_GPIO_OUT1	0	325
0x44	R_GPIO_SEL	0	327
0x13	R_IRQ_CTRL	0	304
0x11	R_IRQMSK_MISC	H	303
0x2F	R_JATT_CFG	3	213
0x22	R_LOS0	0, 1, 3	202
0x23	R_LOS1	0, 1, 3	203
0x14	R_PCM_MD0	0, 2	245
0x15	R_PCM_MD1	0, 2	251
0x15	R_PCM_MD2	0, 2	252
0x46	R_PWM_MD	0	260
0x38	R_PWM0	0, 1, 3	259
0x39	R_PWM1	0, 1, 3	259
0x08	R_RAM_ADDR0	0	100
0x09	R_RAM_ADDR1	0	101
0x0A	R_RAM_ADDR2	0	101
0x0C	R_RAM_MISC	H	102
0x30	R_RX_OFFS	0, 1, 3	214
0x25	R_RX_SL0_CFG0	0, 1, 3	205
0x26	R_RX_SL0_CFG1	0, 1, 3	207
0x24	R_RX0	0, 1, 3	204
0x15	R_SH0H	0, 2	253
0x15	R_SH0L	0, 2	253
0x15	R_SH1H	0, 2	254
0x15	R_SH1L	0, 2	253
0x15	R_SL_SEL0	0, 2	246
0x15	R_SL_SEL1	0, 2	247
0x15	R_SL_SEL2	0, 2	247
0x15	R_SL_SEL3	0, 2	248
0x15	R_SL_SEL4	0, 2	248
0x15	R_SL_SEL5	0, 2	249
0x15	R_SL_SEL6	0, 2	249
0x15	R_SL_SEL7	0, 2	250
0x10	R_SLOT	0, 2	244
0x35	R_SYNC_CTRL	0, 1, 3	217
0x31	R_SYNC_OUT	0, 1, 3	215
0x1A	R_TI_WD	0, 1	305
0x34	R_TX_OFFS	0, 1, 4	216
0x2C	R_TX_SL0_CFG0	0, 1, 3	210
0x2E	R_TX_SL0_CFG1	0, 1, 3	212
0x2D	R_TX_SL0	0, 1, 3	211
0x28	R_TX0	0, 1, 3	208
0x29	R_TX1	0, 1, 3	209

Read only registers:

Address	Name	Reset group	Page
0x0C	A_F1[FIFO]	0, 1	178
0x0C	A_F12[FIFO]	0, 1	178
0x0D	A_F2[FIFO]	0, 1	179
0x04	A_Z1[FIFO]	0, 1	176
0x04	A_Z12[FIFO]	0, 1	176
0x05	A_Z1H[FIFO]	0, 1	177
0x04	A_Z1L[FIFO]	0, 1	176
0x06	A_Z2[FIFO]	0, 1	177
0x07	A_Z2H[FIFO]	0, 1	178
0x06	A_Z2L[FIFO]	0, 1	177
0x1B	R_BERT_ECH	0, 1	288
0x1A	R_BERT_ECL	0, 1	287
0x17	R_BERT_STA	0, 1	287
0x16	R_CHIP_ID	H	103
0x1F	R_CHIP_RV	H	103
0x14	R_CONF_OFLOW	0, 1	270
0x35	R_CRC_ECH	0, 3	225
0x34	R_CRC_ECL	0, 3	225
0x37	R_E_ECH	0, 3	226
0x36	R_E_ECL	0, 3	226
0x20	R_E1_RD_STA	0, 3	218
0x19	R_F0_CNTH	0, 1	255
0x18	R_F0_CNTL	0, 1	255
0x31	R_FAS_ECH	0, 3	224
0x30	R_FAS_ECL	0, 3	223
0x44	R_GPI_IN0		329
0x45	R_GPI_IN1		330
0x46	R_GPI_IN2		330
0x47	R_GPI_IN3		331
0x40	R_GPIO_IN0		328
0x41	R_GPIO_IN1		329
0x88	R_INT_DATA	–	179
0xC8	R_IRQ_FIFO_BL0	0, 1	310
0xC9	R_IRQ_FIFO_BL1	0, 1	311
0xCA	R_IRQ_FIFO_BL2	0, 1	312
0xCB	R_IRQ_FIFO_BL3	0, 1	313
0xCC	R_IRQ_FIFO_BL4	0, 1	314
0xCD	R_IRQ_FIFO_BL5	0, 1	315
0xCE	R_IRQ_FIFO_BL6	0, 1	316
0xCF	R_IRQ_FIFO_BL7	0, 1	317
0x11	R_IRQ_MISC	0, 1	308
0x10	R_IRQ_OVIEW	0, 1	307
0x2B	R_JATT_STA	3	222
0x15	R_RAM_USE	0, 1	103
0x25	R_RX_SL0_0	0, 3	220
0x26	R_RX_SL0_1	0, 3	221
0x27	R_RX_SL0_2	0, 3	222
0x39	R_SA6_VAL13_ECH	0, 3	227
0x38	R_SA6_VAL13_ECL	0, 3	226
0x3B	R_SA6_VAL23_ECH	0, 3	228
0x3A	R_SA6_VAL23_ECL	0, 3	227
0x2C	R_SLIP	0, 3	223
0x1C	R_STATUS	–	309
0x24	R_SYNC_STA	0, 3	219
0x33	R_VIO_ECH	0, 3	225
0x32	R_VIO_ECL	0, 3	224

Read / Write registers:

Address	Name	Reset group	Page
0x80	A_FIFO_DATA0[FIFO]	–	180
0x80	A_FIFO_DATA1[FIFO]	–	180
0x80	A_FIFO_DATA2[FIFO]	–	180
0xC0	R_RAM_DATA	–	104

Note: Reset group 0 = soft reset, 1 = HFC reset, 2 = PCM reset, 3 = line interface reset, H = hardware reset. See Table 12.3 on page 295 for ‘Reset group’ explanation.

Registers sorted by address

Please note !

See explanation of register types on page 19.

Write only registers:

Address	Name	Reset group	Page
0x00	R_CIRM	H	99
0x01	R_CTRL	H	100
0x02	R_BRG_PCM_CFG	H	302
0x08	R_RAM_ADDR0	0	100
0x09	R_RAM_ADDR1	0	101
0x0A	R_RAM_ADDR2	0	101
0x0B	R_FIRST_FIFO	0, 1	165
0x0C	R_RAM_MISC	H	102
0x0D	R_FIFO_MD	H	166
0x0E	A_INC_RES_FIFO[FIFO]	–	167
0x0F	R_FSM_IDX	0, 1	168
0x0F	R_FIFO	0, 1	168
0x10	R_SLOT	0, 2	244
0x11	R_IRQMSK_MISC	H	303
0x13	R_IRQ_CTRL	0	304
0x14	R_PCM_MD0	0, 2	245
0x15	R_PCM_MD1	0, 2	251
0x15	R_PCM_MD2	0, 2	252
0x15	R_SH0H	0, 2	253
0x15	R_SH1H	0, 2	254
0x15	R_SH0L	0, 2	253
0x15	R_SH1L	0, 2	253
0x15	R_SL_SEL0	0, 2	246
0x15	R_SL_SEL1	0, 2	247
0x15	R_SL_SEL2	0, 2	247
0x15	R_SL_SEL3	0, 2	248
0x15	R_SL_SEL4	0, 2	248
0x15	R_SL_SEL5	0, 2	249
0x15	R_SL_SEL6	0, 2	249
0x15	R_SL_SEL7	0, 2	250
0x18	R_CONF_EN	0, 2	268
0x1A	R_TI_WD	0, 1	305
0x1B	R_BERT_WD_MD	0, 1	286
0x1C	R_DTMF	0	279
0x1D	R_DTMF_N	0	280

Address	Name	Reset group	Page
0x20	R_E1_WR_STA	0, 1, 3	202
0x22	R_LOS0	0, 1, 3	202
0x23	R_LOS1	0, 1, 3	203
0x24	R_RX0	0, 1, 3	204
0x25	R_RX_SL0_CFG0	0, 1, 3	205
0x26	R_RX_SL0_CFG1	0, 1, 3	207
0x28	R_TX0	0, 1, 3	208
0x29	R_TX1	0, 1, 3	209
0x2C	R_TX_SL0_CFG0	0, 1, 3	210
0x2D	R_TX_SL0	0, 1, 3	211
0x2E	R_TX_SL0_CFG1	0, 1, 3	212
0x2F	R_JATT_CFG	3	213
0x30	R_RX_OFFS	0, 1, 3	214
0x31	R_SYNC_OUT	0, 1, 3	215
0x34	R_TX_OFFS	0, 1, 4	216
0x35	R_SYNC_CTRL	0, 1, 3	217
0x38	R_PWM0	0, 1, 3	259
0x39	R_PWM1	0, 1, 3	259
0x40	R_GPIO_OUT0	0	324
0x41	R_GPIO_OUT1	0	325
0x42	R_GPIO_EN0	0	325
0x43	R_GPIO_EN1	0	326
0x44	R_GPIO_SEL	0	327
0x46	R_PWM_MD	0	260
0x84	A_FIFO_DATA2_NOINC[FIFO]		169
0x84	A_FIFO_DATA0_NOINC[FIFO]		169
0x84	A_FIFO_DATA1_NOINC[FIFO]		169
0xD0	A_SL_CFG[SLOT]	0, 3	254
0xD1	A_CONF[SLOT]	–	269
0xF4	A_CH_MSK[FIFO]	0, 1	170
0xFA	A_CON_HDLC[FIFO]	0, 1	171
0xFB	A_SUBCH_CFG[FIFO]	0, 1	173
0xFC	A_CHANNEL[FIFO]	0, 1	174
0xFD	A_FIFO_SEQ[FIFO]	0, 1	175
0xFF	A_IRQ_MSK[FIFO]	0, 1	306

Read only registers:

Address	Name	Reset group	Page
0x04	A_Z12[FIFO]	0, 1	176
0x04	A_Z1L[FIFO]	0, 1	176
0x04	A_Z1[FIFO]	0, 1	176
0x05	A_Z1H[FIFO]	0, 1	177
0x06	A_Z2L[FIFO]	0, 1	177
0x06	A_Z2[FIFO]	0, 1	177
0x07	A_Z2H[FIFO]	0, 1	178
0x0C	A_F1[FIFO]	0, 1	178
0x0C	A_F12[FIFO]	0, 1	178
0x0D	A_F2[FIFO]	0, 1	179
0x10	R_IRQ_OVIEW	0, 1	307
0x11	R_IRQ_MISC	0, 1	308
0x14	R_CONF_OFLOW	0, 1	270
0x15	R_RAM_USE	0, 1	103
0x16	R_CHIP_ID	H	103
0x17	R_BERT_STA	0, 1	287
0x18	R_F0_CNTL	0, 1	255
0x19	R_F0_CNTH	0, 1	255
0x1A	R_BERT_ECL	0, 1	287
0x1B	R_BERT_ECH	0, 1	288
0x1C	R_STATUS	–	309
0x1F	R_CHIP_RV	H	103
0x20	R_E1_RD_STA	0, 3	218
0x24	R_SYNC_STA	0, 3	219
0x25	R_RX_SL0_0	0, 3	220
0x26	R_RX_SL0_1	0, 3	221
0x27	R_RX_SL0_2	0, 3	222
0x2B	R_JATT_STA	3	222
0x2C	R_SLIP	0, 3	223
0x30	R_FAS_ECL	0, 3	223
0x31	R_FAS_ECH	0, 3	224
0x32	R_VIO_ECL	0, 3	224
0x33	R_VIO_ECH	0, 3	225
0x34	R_CRC_ECL	0, 3	225
0x35	R_CRC_ECH	0, 3	225
0x36	R_E_ECL	0, 3	226
0x37	R_E_ECH	0, 3	226
0x38	R_SA6_VAL13_ECL	0, 3	226
0x39	R_SA6_VAL13_ECH	0, 3	227
0x3A	R_SA6_VAL23_ECL	0, 3	227
0x3B	R_SA6_VAL23_ECH	0, 3	228
0x40	R_GPIO_IN0		328
0x41	R_GPIO_IN1		329
0x44	R_GPI_IN0		329
0x45	R_GPI_IN1		330
0x46	R_GPI_IN2		330
0x47	R_GPI_IN3		331
0x88	R_INT_DATA	–	179
0xC8	R_IRQ_FIFO_BL0	0, 1	310
0xC9	R_IRQ_FIFO_BL1	0, 1	311
0xCA	R_IRQ_FIFO_BL2	0, 1	312
0xCB	R_IRQ_FIFO_BL3	0, 1	313
0xCC	R_IRQ_FIFO_BL4	0, 1	314
0xCD	R_IRQ_FIFO_BL5	0, 1	315
0xCE	R_IRQ_FIFO_BL6	0, 1	316
0xCF	R_IRQ_FIFO_BL7	0, 1	317

Read / Write registers:

Address	Name	Reset group	Page
0x80	A_FIFO_DATA2[FIFO]	–	180
0x80	A_FIFO_DATA0[FIFO]	–	180
0x80	A_FIFO_DATA1[FIFO]	–	180
0xC0	R_RAM_DATA	–	104

Note: Reset group 0 = soft reset, 1 = HFC reset, 2 = PCM reset, 3 = line interface reset, H = hardware reset. See Table 12.3 on page 295 for ‘Reset group’ explanation.

About this data sheet and Cologne Chip technical support

This data sheet covers all the features of HFC-E1. The reader who absorbs the information in this data sheet will gain a deep and broad understanding of the HFC-E1 microchip.

However, the hurried reader needs not to read the complete data sheet. Every chapter comprises just one topic. What's not needed in the focus of a target application can be skipped over while reading this data sheet.

Organization of this data sheet

Chapters start with a short overview. They typically contain both the electrical description and the programming features of the corresponding subject. Finally, chapters end with a register description.

Links between chapters are mentioned in the text.

Development tools

Driver software plays an important role in all ISDN projects. For this reason we offer more than the hardware:

- An evaluation board of HFC-E1 is available. This can be used in a common PC environment (using PCI bus), e.g.
- A demo layer 1 driver as source code is available.
- There are also header files with all registers and their bitmaps available for programming language C. Please ask the Cologne Chip support team for more information and file delivery.

Visit our web site

Our web site (`http://www.colognechip.com`) contains a download area for all Cologne Chip data sheets. Additional information is given concerning transformers, drivers etc. on the website, too.

By having broad knowledge about ISDN applications, Cologne Chip supports any project individually. Please contact our support team.

Chapter overview

Chapter "General description" (1) begins with an overview to HFC-E1, especially a general block diagram and a feature list. Pinout diagrams for the different processor interfaces and a detailed list of all pins complete this chapter.

Chapter "Universal external bus interface" (2): HFC-E1 supports several processor interfaces which are explained in this chapter. This includes signal and timing characteristics as well as register access and typical connection circuitries. *(Separated inferface modes, read only the section which deals with the interface mode of your interest, prerequisite knowledge of the chosen interface mode is strongly recommended.)*

Chapter "HFC-E1 data flow" (3) starts with the data processing explanation. This chapter deals with the data flow concept which connects all the data interfaces that are explained in the following chapters. *(It is recommended to have at least a basic comprehension to this topic, as it connects several important parts of HFC-E1.)*

Chapter "FIFO handling and HDLC controller" (4) covers the host side of the data flow. This includes both the HDLC controller and the FIFOs. *(Should be read, because FIFOs and the HDLC controller is typically used in every application.)*

Chapter "E1 interface" (5): HFC-E1 has an E1 line interface. This chapter explains the data structures, clock synchronization and external circuitries. *(The most important interface, should be read, prerequisite knowledge of ISDN protocol is strongly recommended.)*

Chapter "PCM interface" (6): The last interface which deals with the data flow described in chapter 3 is the PCM interface. Beneath other, an important topic of this chapter are synchronization features of HFC-E1. *(Read only when used, but don't skip the overview in this chapter even if the PCM interface is not used!)*

Chapter "Pulse width modulation (PWM) outputs" (7): This chapter can be skipped if the PWM interface is not used.

Chapter "Multiparty audio conferences" (8): This chapter can be skipped if the multiparty audio conference is not used.

Chapter "DTMF controller" (9): This chapter can be skipped if the DTMF controller is not used.

Chapter "Bit Error Rate Test (BERT)" (10): This chapter can be skipped if the BERT functionality is not used.

Chapter "Clock, reset, interrupt, timer and watchdog" (12) explains clock generation and distribution, reset functions and interrupt capabilities. *(Must be read.)*

Chapter "General purpose I/O pins (GPIO) and input pins (GPI)" (13): This chapter can be skipped if no GPIO or GPI pins are used.

Chapter "Electrical characteristics" (14): Some information about the electrical characteristics of HFC-E1 are given in this chapter. *(Information for hardware design.)*

Chapter "HFC-E1 package dimensions" (15) shows the HFC-E1 package dimensions. *(Information for hardware design.)*

General remarks to notations

1. The decimal point is written as a point (e.g. 1.23). Thousands separators are written with thin space.
2. Numerical values have different notations for various number systems; e.g. the hexadecimal value 0xC9 is '11001001' in binary and 201 in decimal notation.
3. The prefix 'kilo' is written k for the meaning of 1000 and it is written K for the meaning of 1024.
4. The first letter of register names indicates the type: 'R_ ...' is a register or multi-register, while 'A_ ...' is an array register.

Chapter 1

General description

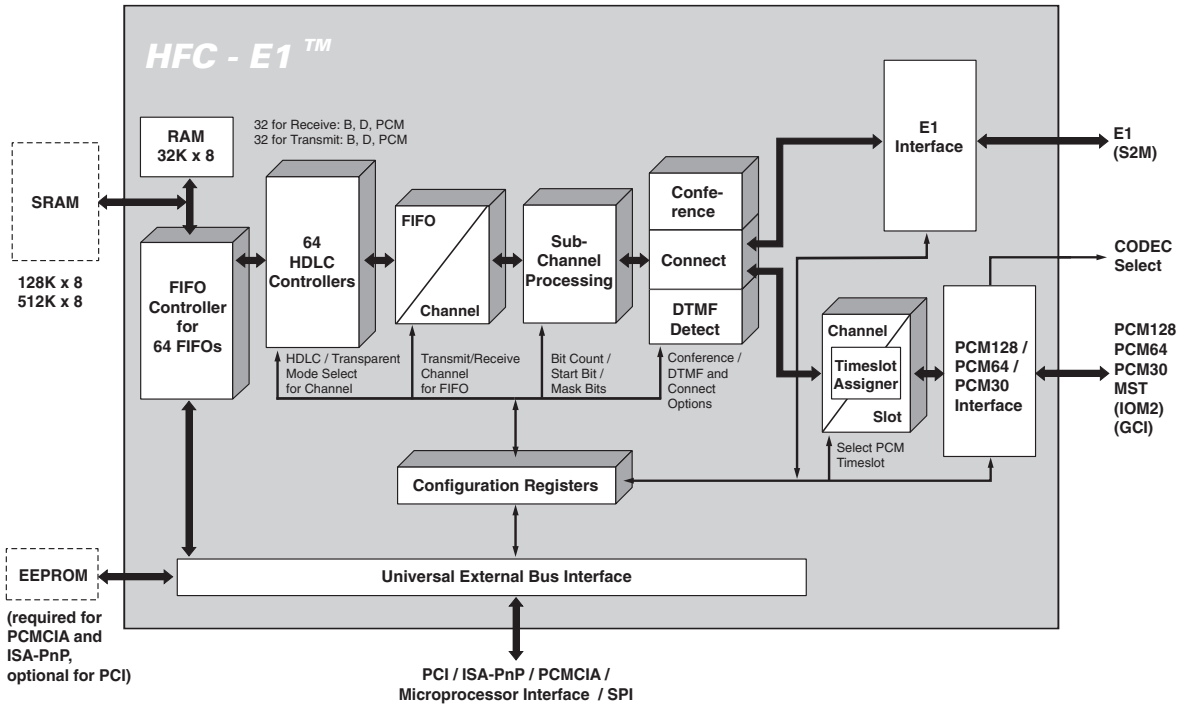


Figure 1.1: *HFC-E1 block diagram*

1.1 System overview

HFC-E1 is an ISDN E1 HDLC Primary Rate Interface controller for all kinds of PRI equipment, such as

- High performance ISDN PC cards
- ISDN PRI terminal adapters
- ISDN PABX for PRI
- VoIP gateways / VoIP routers
- Integrated Access Devices (IAD)
- ISDN LAN routers for PRI
- ISDN least cost routers for PRI
- ISDN test equipment for PRI

The integrated universal bus interface of HFC-E1 can be configured to PCI, ISA Plug and Play, PCMCIA, parallel microprocessor interface or SPI. A PCM128 / PCM64 / PCM30 interface for CODEC or inter chip connection is also integrated. The very deep FIFOs of HFC-E1 are realized with an internal or external SRAM.

1.2 Features

Line interface

- Integrated E1 interface
- ITU-T I.431 ISDN support in TE, NT and LT mode [3]

HDLC controller and FIFO controller

- HDLC controller with support for 31 E1-channels (e.g. 30 ISDN B-channels, 1 ISDN D-channel)
- Transparent mode and data rate independently selectable for all FIFOs
- Up to 32 FIFOs for transmit and receive data each, FIFO sizes configurable
- Maximum 31 HDLC frames (with 128 KByte or 512 KByte external RAM) or 15 HDLC frames (with 32 KByte build-in RAM) per FIFO
- Bit Error Rate Test (BERT) with transmitter and receiver
- Programmable data flow to connect FIFOs, line interface and PCM time slots with each other

PCM interface

- PCM128 / PCM64 / PCM30 interface configurable to MST (MVIP) [1] or Siemens IOM™-2 and Motorola GCI (no monitor and C/I-channel support) for interchip connection or external CODECs
- Programmable PCM time slot assigner for 32 channels in transmit and receive direction each (switch matrix for PCM)
- H.100 data rate supported

Special telecommunication features

- Multiparty audio conferences switchable
- DTMF detection on all 32 channels

Microprocessor bus interface

- Integrated PCI bus interface (Spec. 2.2) for 3.3 V and 5 V signal environment
- Integrated ISA Plug and Play interface with buffers for ISA-databus
- Integrated PCMCIA interface
- Parallel microprocessor interface compatible to Motorola bus and Siemens / Intel bus
- Serial processor interface (SPI)

Miscelleanous features

- Interrupt controller
- Timer and watchdog with interrupt capability
- Two general purpose pulse width modulators (PWM) with dedicated output pins

Technology features

- Single 3.3 V power supply
- CMOS 3.3 V technology, 5 V tolerant on nearly all inputs
- LQFP 208 package
- RoHS compliant (week code dated later than 19-2004)

[1] Mitel Serial Telecom bus

1.3 Pin description

1.3.1 Pinout diagram

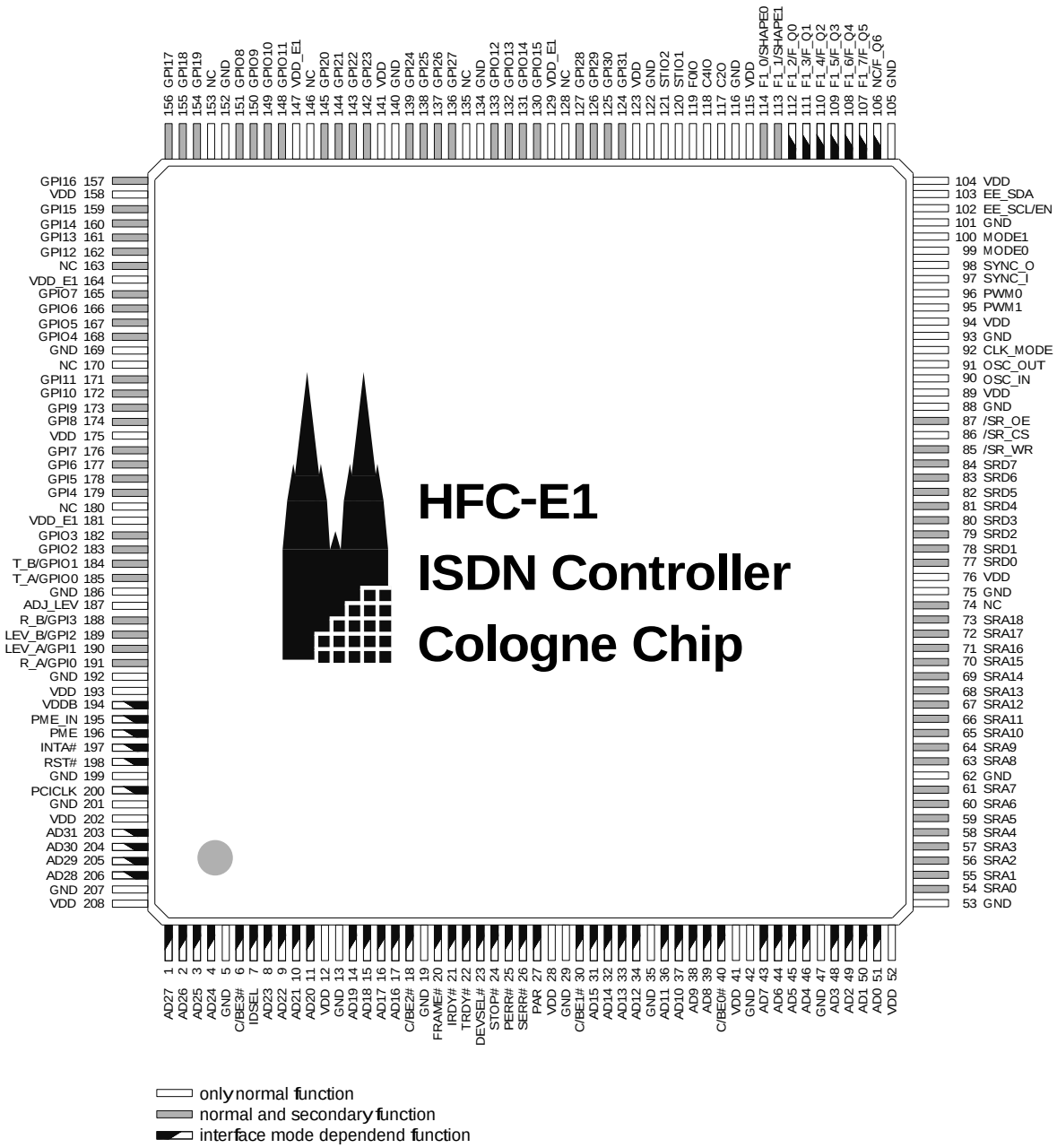


Figure 1.2: *HFC-E1 pinout in PCI mode*

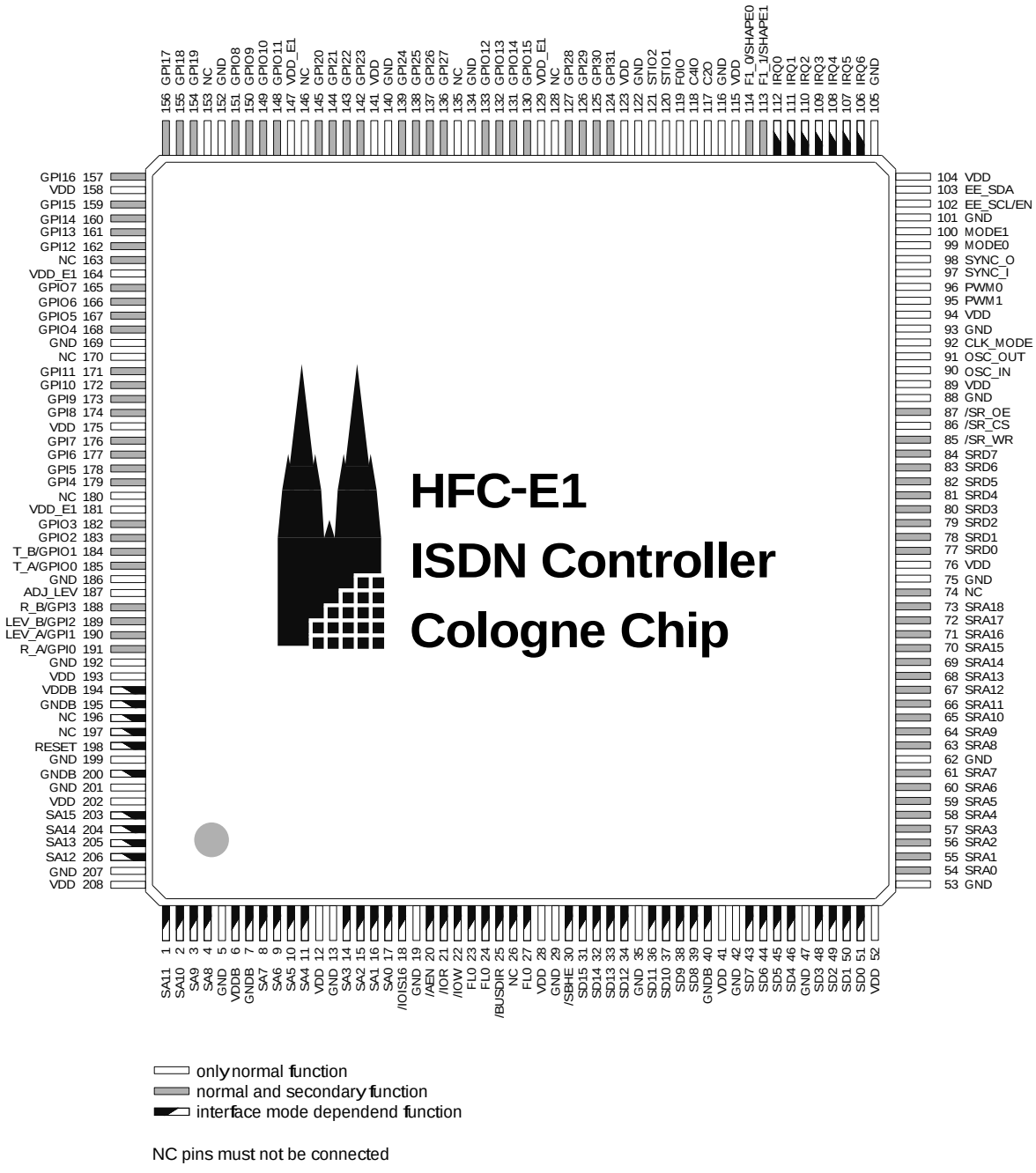


Figure 1.3: *HFC-E1 pinout in ISA PnP mode*

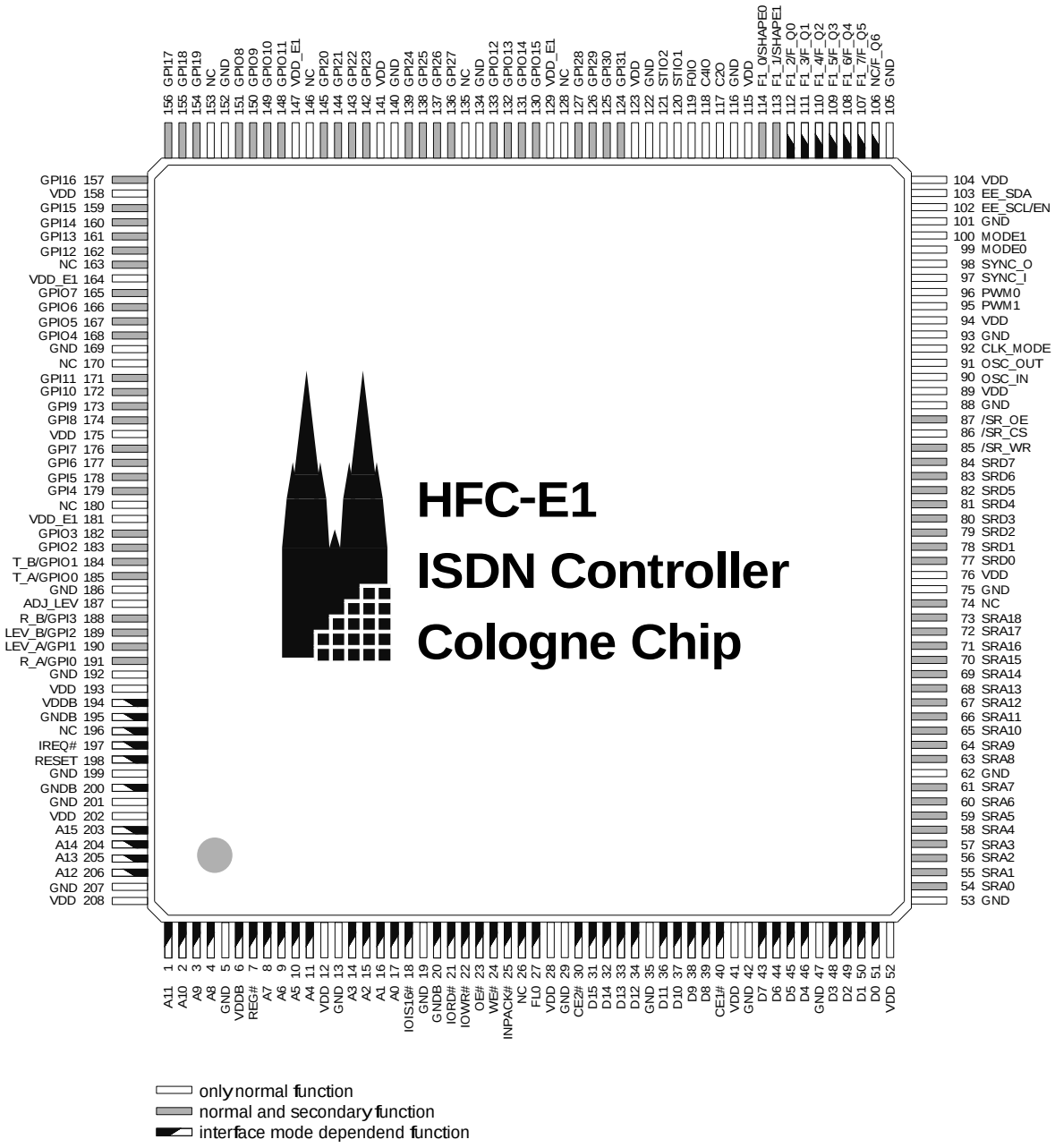


Figure 1.4: *HFC-E1 pinout in PCMCIA mode*

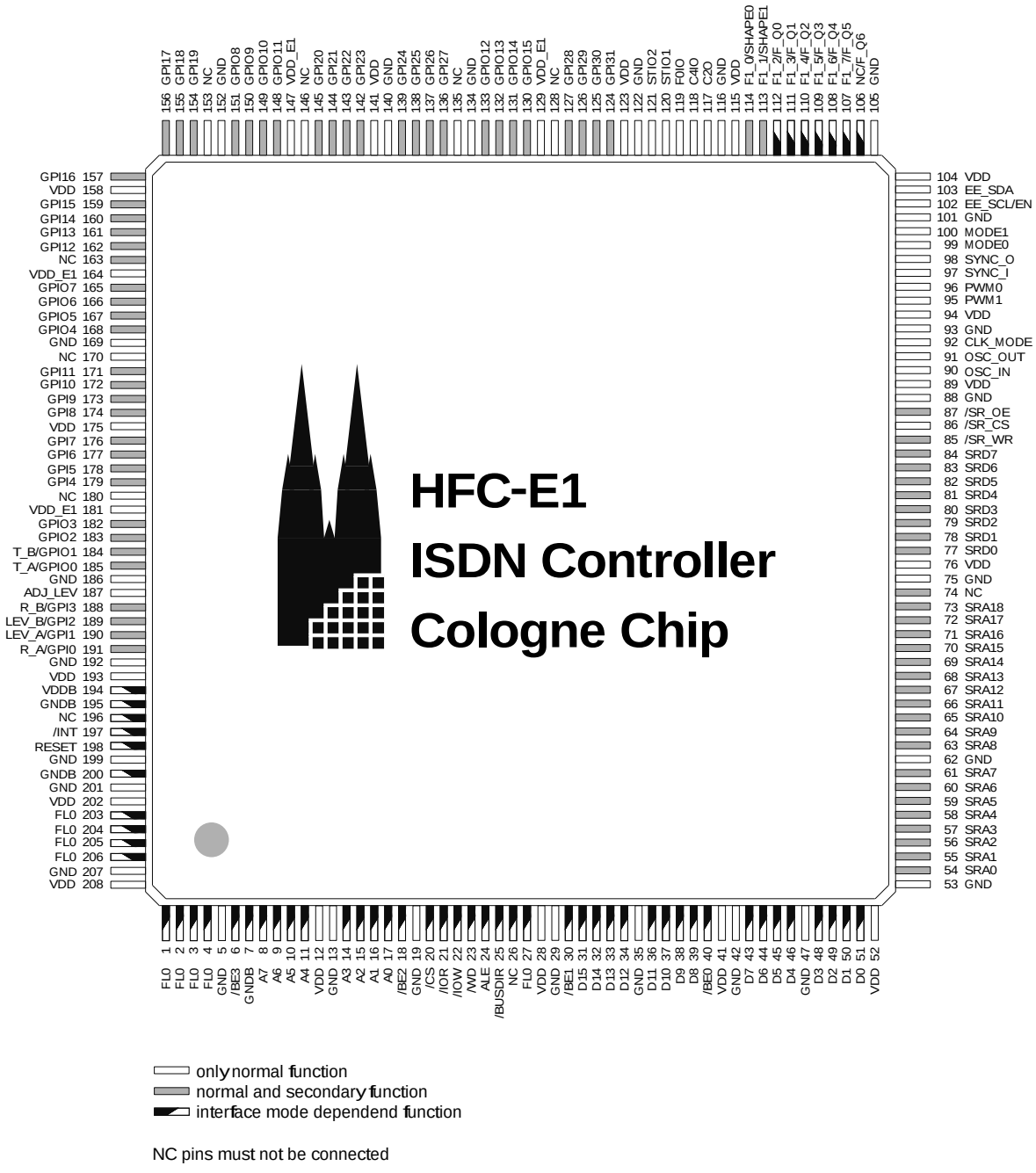


Figure 1.5: *HFC-E1 pinout in parallel processor mode*

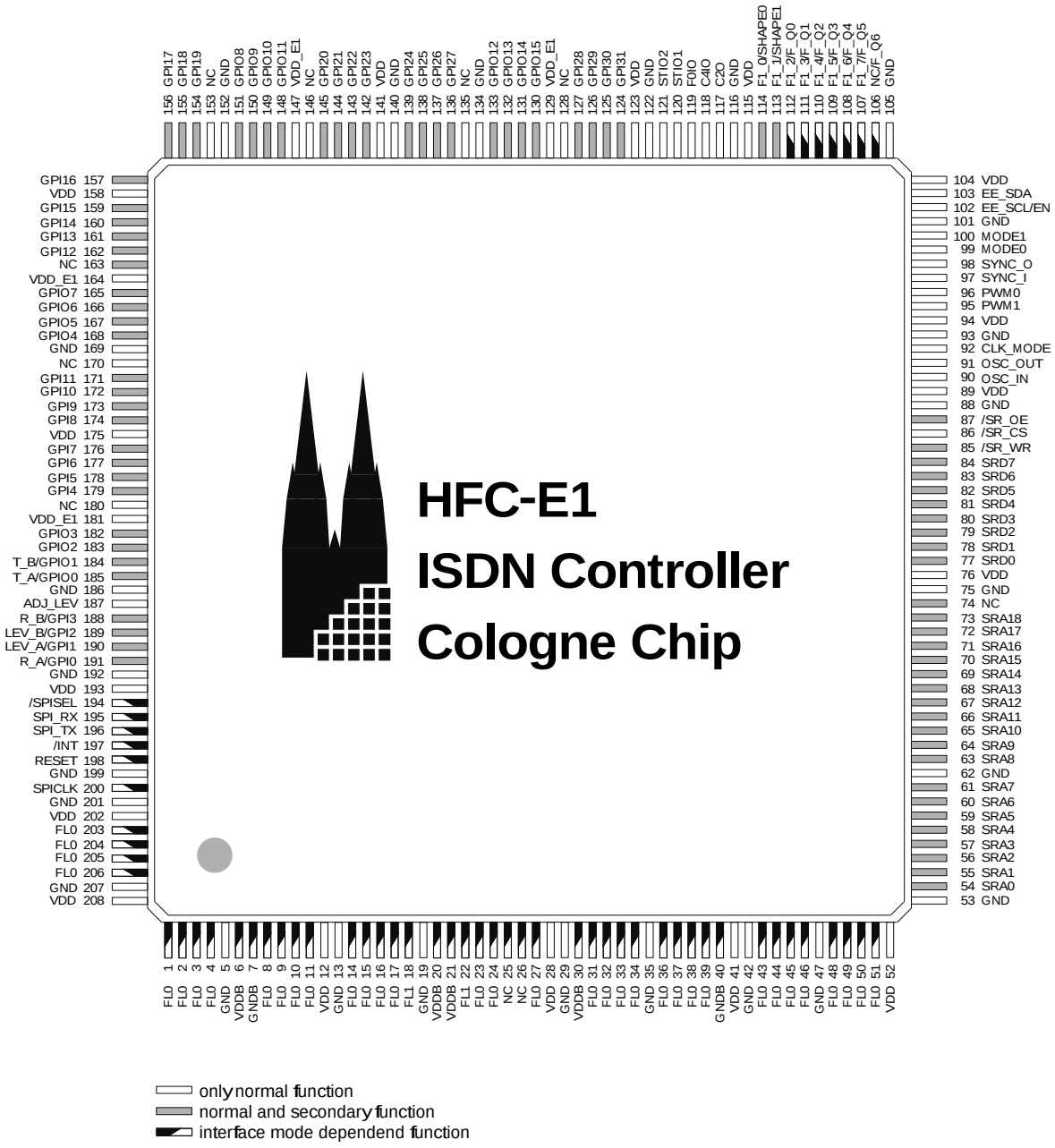


Figure 1.6: *HFC-E1 pinout in SPI mode*

1.3.2 Pin list

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
				Universal bus interface		
1	PCI	AD27	IO	Address / Data bit 27	LVCMOS	8
	ISA PnP	SA11	I	Address bit 11	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
2	PCI	AD26	IO	Address / Data bit 26	LVCMOS	8
	ISA PnP	SA10	I	Address bit 10	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
3	PCI	AD25	IO	Address / Data bit 25	LVCMOS	8
	ISA PnP	SA9	I	Address bit 9	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
4	PCI	AD24	IO	Address / Data bit 24	LVCMOS	8
	ISA PnP	SA8	I	Address bit 8	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
5		GND		Ground		
6	PCI	C/BE3#	I	Bus command and Byte Enable 3	LVCMOS	
	ISA PnP	VDDB		+3.3 V power supply		
	Processor	/BE3	I	Byte Enable 3	LVCMOS	
	SPI	VDDB		+3.3 V power supply		
7	PCI	IDSEL	I	Initialisation Device Select	LVCMOS	
	ISA PnP	GNDB	I	Ground	LVCMOS	
	Processor	GNDB	I	Ground	LVCMOS	
	SPI	GNDB	I	Ground	LVCMOS	
8	PCI	AD23	IO	Address / Data bit 23	LVCMOS	8
	ISA PnP	SA7	I	Address bit 7	LVCMOS	
	Processor	A7	I	Address bit 7	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
9	PCI	AD22	IO	Address / Data bit 22	LVCMOS	8
	ISA PnP	SA6	I	Address bit 6	LVCMOS	
	Processor	A6	I	Address bit 6	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
10	PCI	AD21	IO	Address / Data bit 21	LVCMOS	8
	ISA PnP	SA5	I	Address bit 5	LVCMOS	
	Processor	A5	I	Address bit 5	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
11	PCI	AD20	IO	Address / Data bit 20	LVCMOS	8
	ISA PnP	SA4	I	Address bit 4	LVCMOS	
	Processor	A4	I	Address bit 4	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
12		VDD		+3.3 V power supply		
13		GND		Ground		
14	PCI	AD19	IO	Address / Data bit 19	LVCMOS	8
	ISA PnP	SA3	I	Address bit 3	LVCMOS	
	Processor	A3	I	Address bit 3	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
15	PCI	AD18	IO	Address / Data bit 18	LVCMOS	8
	ISA PnP	SA2	I	Address bit 2	LVCMOS	
	Processor	A2	I	Address bit 2	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
16	PCI	AD17	IO	Address / Data bit 17	LVCMOS	8
	ISA PnP	SA1	I	Address bit 1	LVCMOS	
	Processor	A1	I	Address bit 1	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
17	PCI	AD16	IO	Address / Data bit 16	LVCMOS	8
	ISA PnP	SA0	I	Address bit 0	LVCMOS	
	Processor	A0	I	Address bit 0	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
18	PCI	C/BE2#	I	Bus command and Byte Enable 2	LVCMOS	
	ISA PnP	/IOIS16	Ood	16 bit access enable		8
	Processor	/BE2	I	Byte Enable 2	LVCMOS	
	SPI	FL1	I	Fixed level (high), connect to power supply via ext. pull-up	LVCMOS	
19		GND		Ground		
20	PCI	FRAME#	I	Cycle Frame	LVCMOS	
	ISA PnP	/AEN	I	Address Enable	LVCMOS	
	Processor	/CS	I	Chip Select	LVCMOS	
	SPI	VDDB		+3.3 V power supply		
21	PCI	IRDY#	I	Initiator Ready	LVCMOS	
	ISA PnP	/IOR	I	Read Enable	LVCMOS	
	Processor	/IOR	I	Read Enable	LVCMOS	
	SPI	VDDB		+3.3 V power supply		

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
22	PCI	TRDY#	O	Target Ready		8
	ISA PnP	/IOW	I	Write Enable	LVCMOS	
	Processor	/IOW	I	Write Enable	LVCMOS	
	SPI	FL1	I	Fixed level (high), connect to power supply via ext. pull-up	LVCMOS	
23	PCI	DEVSEL#	O	Device Select		8
	ISA PnP	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	Processor	/WD	Ood	Watch Dog Output		8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
24	PCI	STOP#	O	Stop		8
	ISA PnP	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	Processor	ALE	I	Address Latch Enable	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
25	PCI	PERR#	IO	Parity Error	LVCMOS	8
	ISA PnP	/BUSDIR	O	Bus Direction		8
	Processor	/BUSDIR	O	Bus Direction		8
	SPI	NC				
26	PCI	SERR#	Ood	System Error		8
	ISA PnP	NC				
	Processor	NC				
	SPI	NC				
27	PCI	PAR	IO	Parity Bit	LVCMOS	8
	ISA PnP	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
28		VDD		+3.3 V power supply		
29		GND		Ground		
30	PCI	C/BE1#	I	Bus command and Byte Enable 1	LVCMOS	
	ISA PnP	/SBHE	I	High byte enable	LVCMOS	
	Processor	/BE1	I	Byte Enable 1	LVCMOS	
	SPI	VDDB		+3.3 V power supply		
31	PCI	AD15	IO	Address / Data bit 15	LVCMOS	8
	ISA PnP	SD15	IO	ISA Data Bus Bit 15	LVCMOS	8
	Processor	D15	IO	Data bit 15	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
32	PCI	AD14	IO	Address / Data bit 14	LVCMOS	8
	ISA PnP	SD14	IO	ISA Data Bus Bit 14	LVCMOS	8
	Processor	D14	IO	Data bit 14	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
33	PCI	AD13	IO	Address / Data bit 13	LVCMOS	8
	ISA PnP	SD13	IO	ISA Data Bus Bit 13	LVCMOS	8
	Processor	D13	IO	Data bit 13	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
34	PCI	AD12	IO	Address / Data bit 12	LVCMOS	8
	ISA PnP	SD12	IO	ISA Data Bus Bit 12	LVCMOS	8
	Processor	D12	IO	Data bit 12	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
35		GND		Ground		
36	PCI	AD11	IO	Address / Data bit 11	LVCMOS	8
	ISA PnP	SD11	IO	ISA Data Bus Bit 11	LVCMOS	8
	Processor	D11	IO	Data bit 11	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
37	PCI	AD10	IO	Address / Data bit 10	LVCMOS	8
	ISA PnP	SD10	IO	ISA Data Bus Bit 10	LVCMOS	8
	Processor	D10	IO	Data bit 10	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
38	PCI	AD9	IO	Address / Data bit 9	LVCMOS	8
	ISA PnP	SD9	IO	ISA Data Bus Bit 9	LVCMOS	8
	Processor	D9	IO	Data bit 9	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
39	PCI	AD8	IO	Address / Data bit 8	LVCMOS	8
	ISA PnP	SD8	IO	ISA Data Bus Bit 8	LVCMOS	8
	Processor	D8	IO	Data bit 8	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
40	PCI	C/BE0#	I	Bus command and Byte Enable 0	LVCMOS	
	ISA PnP	GNDB		Ground		
	Processor	/BE0	I	Byte Enable 0	LVCMOS	
	SPI	GNDB		Ground		
41		VDD		+3.3 V power supply		
42		GND		Ground		
43	PCI	AD7	IO	Address / Data bit 7	LVCMOS	8
	ISA PnP	SD7	IO	ISA Data Bus Bit 7	LVCMOS	8
	Processor	D7	IO	Data bit 7	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
44	PCI	AD6	IO	Address / Data bit 6	LVCMOS	8
	ISA PnP	SD6	IO	ISA Data Bus Bit 6	LVCMOS	8
	Processor	D6	IO	Data bit 6	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
45	PCI	AD5	IO	Address / Data bit 5	LVCMOS	8
	ISA PnP	SD5	IO	ISA Data Bus Bit 5	LVCMOS	8
	Processor	D5	IO	Data bit 5	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
46	PCI	AD4	IO	Address / Data bit 4	LVCMOS	8
	ISA PnP	SD4	IO	ISA Data Bus Bit 4	LVCMOS	8
	Processor	D4	IO	Data bit 4	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
47		GND		Ground		
48	PCI	AD3	IO	Address / Data bit 3	LVCMOS	8
	ISA PnP	SD3	IO	ISA Data Bus Bit 3	LVCMOS	8
	Processor	D3	IO	Data bit 3	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
49	PCI	AD2	IO	Address / Data bit 2	LVCMOS	8
	ISA PnP	SD2	IO	ISA Data Bus Bit 2	LVCMOS	8
	Processor	D2	IO	Data bit 2	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
50	PCI	AD1	IO	Address / Data bit 1	LVCMOS	8
	ISA PnP	SD1	IO	ISA Data Bus Bit 1	LVCMOS	8
	Processor	D1	IO	Data bit 1	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
51	PCI	AD0	IO	Address / Data bit 0	LVCMOS	8
	ISA PnP	SD0	IO	ISA Data Bus Bit 0	LVCMOS	8
	Processor	D0	IO	Data bit 0	LVCMOS	8
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down	LVCMOS	
52		VDD		+3.3 V power supply		
53		GND		Ground		
		SRAM / Auxiliary interface				
54		SRA0	O	Address bit 0 for external SRAM		2
55		SRA1	O	Address bit 1 for external SRAM		2
56		SRA2	O	Address bit 2 for external SRAM		2
57		SRA3	O	Address bit 3 for external SRAM		2
58		SRA4	O	Address bit 4 for external SRAM		2
59		SRA5	O	Address bit 5 for external SRAM		2
60		SRA6	O	Address bit 6 for external SRAM		2
61		SRA7	O	Address bit 7 for external SRAM		2

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
62		GND		Ground		
63		SRA8	O	Address bit 8 for external SRAM		2
64		SRA9	O	Address bit 9 for external SRAM		2
65		SRA10	O	Address bit 10 for external SRAM		2
66		SRA11	O	Address bit 11 for external SRAM		2
67		SRA12	O	Address bit 12 for external SRAM		2
68		SRA13	O	Address bit 13 for external SRAM		2
69		SRA14	O	Address bit 14 for external SRAM		2
70		SRA15	O	Address bit 15 for external SRAM		2
71		SRA16	O	Address bit 16 for external SRAM		2
72		SRA17	O	Address bit 17 for external SRAM		2
73		SRA18	O	Address bit 18 for external SRAM		2
74		NC				
75		GND		Ground		
76		VDD		+3.3 V power supply		
77		SRD0	IO	Data bit 0 for external SRAM	LVCMOS	8
78		SRD1	IO	Data bit 1 for external SRAM	LVCMOS	8
79		SRD2	IO	Data bit 2 for external SRAM	LVCMOS	8
80		SRD3	IO	Data bit 3 for external SRAM	LVCMOS	8
81		SRD4	IO	Data bit 4 for external SRAM	LVCMOS	8
82		SRD5	IO	Data bit 5 for external SRAM	LVCMOS	8
83		SRD6	IO	Data bit 6 for external SRAM	LVCMOS	8
84		SRD7	IO	Data bit 7 for external SRAM	LVCMOS	8
85		/SR_WR	O	Write enable for external SRAM		4
86		/SR_CS	O	Chip Select for external SRAM		4
87		/SR_OE	O	Output enable for external SRAM		4
88		GND		Ground		
89		VDD		+3.3 V power supply		
				Clock		
90		OSC_IN	I	Oscillator Input Signal		
91		OSC_OUT	O	Oscillator Output Signal		
92		CLK_MODE	I	Clock Mode	LVCMOS	
93		GND		Ground		

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
94		VDD		+3.3 V power supply		
				Miscellaneous		
95		PWM1	O	Pulse Width Modulator Output 1		8
96		PWM0	O	Pulse Width Modulator Output 0		8
97		SYNC_I	I	Synchronization Input	LVCMOS	
98		SYNC_O	O	Synchronization Output		4
99		MODE0	I	Interface Mode pin 0	LVCMOS	
100		MODE1	I	Interface Mode pin 1	LVCMOS	
101		GND		Ground		
				EEPROM		
102		EE_SCL/EN	IO	EEPROM clock / EEPROM enable	LVCMOS	1
103		EE_SDA	IO	EEPROM data I/O	LVCMOS	1
104		VDD		+3.3 V power supply		
105		GND		Ground		
				PCM		
106	1st function	NC				
	2nd function	F_Q6	O	PCM time slot count 6		6
	ISA PnP	IRQ6	O	ISA Interrupt Request 6		6
107	1st function	F1_7	O	PCM CODEC enable 7		6
	2nd function	F_Q5	O	PCM time slot count 5		6
	ISA PnP	IRQ5	O	ISA Interrupt Request 5		6
108	1st function	F1_6	O	PCM CODEC enable 6		6
	2nd function	F_Q4	O	PCM time slot count 4		6
	ISA PnP	IRQ4	O	ISA Interrupt Request 4		6
109	1st function	F1_5	O	PCM CODEC enable 5		6
	2nd function	F_Q3	O	PCM time slot count 3		6
	ISA PnP	IRQ3	O	ISA Interrupt Request 3		6
110	1st function	F1_4	O	PCM CODEC enable 4		6
	2nd function	F_Q2	O	PCM time slot count 2		6
	ISA PnP	IRQ2	O	ISA Interrupt Request 2		6
111	1st function	F1_3	O	PCM CODEC enable 3		6
	2nd function	F_Q1	O	PCM time slot count 1		6
	ISA PnP	IRQ1	O	ISA Interrupt Request 1		6
112	1st function	F1_2	O	PCM CODEC enable 2		6
	2nd function	F_Q0	O	PCM time slot count 0		6
	ISA PnP	IRQ0	O	ISA Interrupt Request 0		6
113	1st function	F1_1	O	PCM CODEC enable 1		6
	2nd function	SHAPE1	O	PCM CODEC enable shape signal 1		6

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
114	1st function 2nd function	F1_0 SHAPE0	O O	PCM CODEC enable 0 PCM CODEC enable shape signal 0		6 6
115		VDD		+3.3 V power supply		
116		GND		Ground		
117		C2O	O	PCM bit clock output		8
118		C4IO	IOpu	PCM double bit clock I/O	LVCMOS	8
119		F0IO	IOpu	PCM frame clock I/O (8 kHz)	LVCMOS	8
120		STIO1	IOpu	PCM data bus 1, I or O per time slot	LVCMOS	8
121		STIO2	IOpu	PCM data bus 2, I or O per time slot	LVCMOS	8
122		GND		Ground		
123		VDD		+3.3 V power supply		
				GPIO		
124		GPI31	I	General Purpose Input pin 31	LVCMOS	
125		GPI30	I	General Purpose Input pin 30	LVCMOS	
126		GPI29	I	General Purpose Input pin 29	LVCMOS	
127		GPI28	I	General Purpose Input pin 28	LVCMOS	
128		NC				
129		VDD_E1		Power supply for GPIO 12 . . 15 output drivers (should be connected to VDD)		
130		GPIO15	IO	General Purpose I/O pin 15	LVCMOS	16
131		GPIO14	IO	General Purpose I/O pin 14	LVCMOS	16
132		GPIO13	IO	General Purpose I/O pin 13	LVCMOS	16
133		GPIO12	IO	General Purpose I/O pin 12	LVCMOS	16
134		GND		Ground		
135		NC				
136		GPI27	I	General Purpose Input pin 27	LVCMOS	
137		GPI26	I	General Purpose Input pin 26	LVCMOS	
138		GPI25	I	General Purpose Input pin 25	LVCMOS	
139		GPI24	I	General Purpose Input pin 24	LVCMOS	
140		GND		Ground		
141		VDD		+3.3 V power supply		
142		GPI23	I	General Purpose Input pin 23	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
143		GPI22	I	General Purpose Input pin 22	LVCMOS	
144		GPI21	I	General Purpose Input pin 21	LVCMOS	
145		GPI20	I	General Purpose Input pin 20	LVCMOS	
146		NC				
147		VDD_E1		Power supply for GPIO 8 . . 11 output drivers (should be connected to VDD)		
148		GPIO11	IO	General Purpose I/O pin 11	LVCMOS	16
149		GPIO10	IO	General Purpose I/O pin 10	LVCMOS	16
150		GPIO9	IO	General Purpose I/O pin 9	LVCMOS	16
151		GPIO8	IO	General Purpose I/O pin 8	LVCMOS	16
152		GND		Ground		
153		NC				
154		GPI19	I	General Purpose Input pin 19	LVCMOS	
155		GPI18	I	General Purpose Input pin 18	LVCMOS	
156		GPI17	I	General Purpose Input pin 17	LVCMOS	
157		GPI16	I	General Purpose Input pin 16	LVCMOS	
158		VDD		+3.3 V power supply		
159		GPI15	I	General Purpose Input pin 15	LVCMOS	
160		GPI14	I	General Purpose Input pin 14	LVCMOS	
161		GPI13	I	General Purpose Input pin 13	LVCMOS	
162		GPI12	I	General Purpose Input pin 12	LVCMOS	
163		NC				
164		VDD_E1		Power supply for GPIO 4 . . 7 output drivers (should be connected to VDD)		
165		GPIO7	IO	General Purpose I/O pin 7	LVCMOS	16
166		GPIO6	IO	General Purpose I/O pin 6	LVCMOS	16
167		GPIO5	IO	General Purpose I/O pin 5	LVCMOS	16
168		GPIO4	IO	General Purpose I/O pin 4	LVCMOS	16
169		GND		Ground		
170		NC				
171		GPI11	I	General Purpose Input pin 11	LVCMOS	
172		GPI10	I	General Purpose Input pin 10	LVCMOS	
173		GPI9	I	General Purpose Input pin 9	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
174		GPI8	I	General Purpose Input pin 8	LVCMOS	
175		VDD		+3.3 V power supply		
176		GPI7	I	General Purpose Input pin 7	LVCMOS	
177		GPI6	I	General Purpose Input pin 6	LVCMOS	
178		GPI5	I	General Purpose Input pin 5	LVCMOS	
179		GPI4	I	General Purpose Input pin 4	LVCMOS	
180		NC				
181		VDD_E1		app. +2.8 V power supply (depends on the E1 transmit amplitude)		
182		GPIO3	IO	General Purpose I/O pin 3	LVCMOS	16
183		GPIO2	IO	General Purpose I/O pin 2	LVCMOS	16
				E1 interface		
184	1st function	T_B	O	E1 interface transmit data B		16
	2nd function	GPIO1	IO	General Purpose I/O pin 1	LVCMOS	16
185	1st function	T_A	O	E1 interface transmit data A		16
	2nd function	GPIO0	IO	General Purpose I/O pin 0	LVCMOS	16
186		GND		Ground		
187		ADJ_LEV	Ood	E1 interface level generator		
188	1st function	R_B	I	E1 interface receive input B	E1	
	2nd function	GPI3	I	General Purpose Input pin 3	LVCMOS	
189	1st function	LEV_B	I	E1 interface level detect B	E1	
	2nd function	GPI2	I	General Purpose Input pin 2	LVCMOS	
190	1st function	LEV_A	I	E1 interface level detect A	E1	
	2nd function	GPI1	I	General Purpose Input pin 1	LVCMOS	
191	1st function	R_A	I	E1 interface receive input A	E1	
	2nd function	GPI0	I	General Purpose Input pin 0	LVCMOS	
192		GND		Ground		
193		VDD		+3.3 V power supply		
				Universal bus interface (continued)		
194	PCI	VDDB	I	+3.3 V power supply	LVCMOS	
	ISA PnP	VDDB	I	+3.3 V power supply	LVCMOS	
	Processor	VDDB	I	+3.3 V power supply	LVCMOS	
	SPI	/SPISEL	I	SPI device select low active	LVCMOS	
195	PCI	PME_IN	I	Power Management Event Input	LVCMOS	
	ISA PnP	GNDB		Ground		
	Processor	GNDB		Ground		
	SPI	SPI_RX	I	SPI receive data input	LVCMOS	

(continued on next page)

(continued from previous page)

Pin	Interface	Name	I/O	Description	U_{in} / V	I_{out} / mA
196	PCI	PME	O	Power Management Event output		4
	ISA PnP	NC				
	Processor	NC				
	SPI	SPI_TX	O	SPI transmit data output		4
197	PCI	INTA#	Ood	Interrupt request		4
	ISA PnP	NC				
	Processor	/INT	Ood	Interrupt request		4
	SPI	/INT	Ood	Interrupt request		4
198	PCI	RST#	I	Reset low active	LVCMOS	
	ISA PnP	RESET	I	Reset high active	LVCMOS	
	Processor	RESET	I	Reset high active	LVCMOS	
	SPI	RESET	I	Reset high active	LVCMOS	
199		GND		Ground		
200	PCI	PCICLK	I	PCI Clock Input	LVCMOS	
	ISA PnP	GNDB		Ground		
	Processor	GNDB		Ground		
	SPI	SPICLK	I	SPI clock input	LVCMOS	
201		GND		Ground		
202		VDD		+3.3 V power supply		
203	PCI	AD31	IO	Address / Data bit 31	LVCMOS	8
	ISA PnP	SA15	I	Address bit 15	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
204	PCI	AD30	IO	Address / Data bit 30	LVCMOS	8
	ISA PnP	SA14	I	Address bit 14	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
205	PCI	AD29	IO	Address / Data bit 29	LVCMOS	8
	ISA PnP	SA13	I	Address bit 13	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
206	PCI	AD28	IO	Address / Data bit 28	LVCMOS	8
	ISA PnP	SA12	I	Address bit 12	LVCMOS	
	Processor	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
	SPI	FL0	I	Fixed level (low), connect to ground via ext. pull-down		
207		GND		Ground		
208		VDD		+3.3 V power supply		

Legend:	I	Input pin
	O	Output pin
	IO	Bidirectional pin
	Ood	Output pin with open drain
	IOpu	Bidirectional pin with internal pull-up resistor of app. 100 kΩ to VDD
	NC	Not connected
	FL0	Fixed level (low), must be connected to ground via external pull-down (e.g. 100 kΩ)
	FL1	Fixed level (high), must be connected to power supply via external pull-up (e.g. 100 kΩ)
	GNDB	Ground pin only if it has no other meaning, depends on the selected bus interface mode.
	VDDB	Power supply pin only if it has no other meaning, depends on the selected bus interface mode.

Unused input pins should be connected to ground. Unused I/O pins should be connected via a 100 kΩ resistor to ground.

Important !

FL0 and FL1 pins might be driven as chip output during power-on. To prevent a short circuit these pins must either be connected via a resistor (e.g. 100 kΩ) to ground or power supply (VDD) respectively, or they can be directly connected to ground or power supply, if RESET is always active during power-on.

Chapter 2

Universal external bus interface

(Overview tables of the HFC-E1 bus interface pins can be found at the beginning of sections 2.2 . . 2.7.)

Table 2.1: *Overview of the HFC-E1 bus interface registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x00	R_CIRM	99	0x15	R_RAM_USE	103
0x01	R_CTRL	100	0x16	R_CHIP_ID	103
0x08	R_RAM_ADDR0	100	0x1C	R_STATUS	309
0x09	R_RAM_ADDR1	101	0x1F	R_CHIP_RV	103
0x0A	R_RAM_ADDR2	101			
0x0C	R_RAM_MISC	102			

2.1 Mode selection

HFC-E1 has an integrated universal external bus interface which can be configured as PCI, ISA PnP, PCMCIA, parallel microprocessor interface and SPI. Table 2.2 shows how to select the bus mode via the two pins MODE0 and MODE1 .

Table 2.2: *Access types*

Bus mode	MODE1	MODE0	8 bit	16 bit	32 bit	Page
PCI	0	0				54
PCI memory mapped mode			✓	✓	✓	
PCI I/O mapped mode			✓	✓	✓	
ISA Plug and Play	1	0	✓	✓	✗	61
PCMCIA	1	1	✓	✓	✗	67
Parallel processor interface	0	1				70
Mode 2: Motorola			✓	✓	✗	
Mode 3: Intel, non-multiplexed			✓	✓	✗	
Mode 4: Intel, multiplexed			✓	✓	✓	
SPI *	0	1	✓	✗	✗	90

*: SPI mode is selected by using parallel processor interface mode and connecting pin 200 to SPI clock.

The external bus interface supports 8 bit, 16 bit and 32 bit accesses. The access types available depend on the selected bus mode like shown in Table 2.2.

Sections 2.3 to 2.7 explain how to use HFC-E1 in the different bus modes.

2.2 Common features of all interface modes

Table 2.3: *Overview of common bus interface pins* *

Number	Name	Description
99	MODE0	Interface Mode pin 0
100	MODE1	Interface Mode pin 1
102	EE_SCL/EN	EEPROM clock / EEPROM enable
103	EE_SDA	EEPROM data I/O

*: See sections 2.3 to 2.7 for overview tables of the interface specific pins.

2.2.1 EEPROM programming

The ISA PnP and PCMCIA interfaces require an external EEPROM. For the PCI and the parallel processor interface mode, this EEPROM is optional. The EEPROM is highly recommended in PCI mode. The EEPROM programming specification is only available on special request from Cologne Chip to avoid destruction of configuration information by unauthorized programs or software viruses.

PCI mode: 128 bytes of the EEPROM are copied into the PCI configuration space after every hardware reset.

ISA PnP mode: EPPROM data is directly accessible by the ISA PnP interface. A maximum of 512 bytes are used to store the configuration.

PCMCIA mode: 512 bytes of the EEPROM are copied into the SRAM after every hardware reset. It is used as PCMCIA attribute memory. Table 2.4 show the SRAM start addresses for different RAM sizes.

Parallel processor mode: The EEPROM is not used in the parallel processor mode and pins EE_SCL/EN and EE_SDA must be deactivated as shown in Figure 2.2.

SPI mode: The EEPROM is not used in the SPI mode and pins EE_SCL/EN and EE_SDA must be deactivated as shown in Figure 2.2.

Table 2.4: *SRAM start address*

SRAM size	Start address in SRAM
32k x 8	0x1A00
128k x 8	0x2A00
512k x 8	0x2A00

2.2.2 EEPROM circuitry

Figure 2.1 shows the connection of an EEPROM (e.g. 24C04 type) to the HFC-E1 pins EE_SCL/EN and EE_SDA .

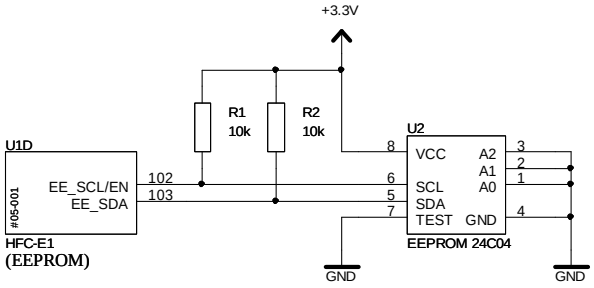


Figure 2.1: *EEPROM connection circuitry*

If no EEPROM is used, pin EE_SCL/EN must be connected to ground while EE_SDA must remain open as shown in Figure 2.2.

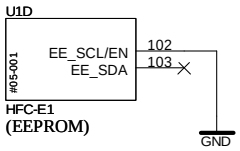


Figure 2.2: *EE_SCL/EN and EE_SDA connection without EEPROM*

2.2.3 RAM access

The SRAM of HFC-E1 can be accessed by the host. To do so, the desired RAM address has to be written into registers R_RAM_ADDR0 . . R_RAM_ADDR2 first. Then data can be read / written by reading / writing register R_RAM_DATA. An automatic increment function can be set in register R_RAM_ADDR2 if the internal SRAM is used.

2.2.4 Register access

In PCI I/O mapped mode, ISA PnP, PCMCIA mode and SPI mode all registers are selected by writing the register address into the *Control Internal Pointer* (CIP) register. This is done by writing the CIP on the higher I/O addresses (AD2 , SA2 , A2 , $A/\overline{D}$ = '1'). The CIP register can also be read with AD2 , SA2 , A2 , $A/\overline{D}$ = '1'.

All consecutive read or write data accesses (AD2 , SA2 , A2 , $A/\overline{D}$ = '0') are done with the selected register until the CIP register is changed.

In parallel processor interface mode all internal registers can be directly accessed. The registers are selected by A0 . . A7 .

In PCI mode internal A0 and A1 are generated from the byte enable lines.

Please note !

Every asynchronous register read access should be done multiple times until two consecutive read accesses result in the same value. Only this way it is ensured that the read bits are stable.

This information applies to the following registers:

A_Z1 (0x04)	A_Z1L (0x04)
A_Z1H (0x05)	A_Z2 (0x06)
A_Z2L (0x06)	A_Z2H (0x07)
A_Z12 (0x04)	A_F1 (0x0C)
A_F2 (0x0D)	A_F12 (0x0C)
R_IRQ_OVIEW (0x10)	R_RAM_USE (0x15)
R_BERT_STA (0x17)	R_F0_CNTL (0x18)
R_FAS_ECL (0x30)	R_VIO_ECL (0x32)
R_CRC_ECL (0x34)	R_E_ECL (0x36)
R_SA6_VAL13_ECL (0x38)	R_SA6_VAL23_ECL (0x3A)

R_IRQ_FIFO_BL0 (0xC8) . . R_IRQ_FIFO_BL7 (0xCF)

2.3 PCI interface

Table 2.5: *Overview of the PCI interface pins*

Number	Name	Description
203 . . 206, 1 . . 4	AD31 . . AD24	Address / Data byte 3
8 . . 17	AD23 . . AD16	Address / Data byte 2
31 . . 39	AD15 . . AD8	Address / Data byte 1
43 . . 51	AD7 . . AD0	Address / Data byte 0
6, 18, 30, 40	C/BE3# . . C/BE0#	Bus command and Byte Enable 3 . . 0
7	IDSEL	Initialisation Device Select
20	FRAME#	Cycle Frame
21	IRDY#	Initiator Ready
22	TRDY#	Target Ready
23	DEVSEL#	Device Select
24	STOP#	Stop
25	PERR#	Parity Error
26	SERR#	System Error
27	PAR	Parity Bit
195	PME_IN	Power Management Event Input
196	PME	Power Management Event output
197	INTA#	Interrupt request
198	RST#	Reset low active
200	PCICLK	PCI Clock Input

The PCI mode is selected by MODE0 = '0' and MODE1 = '0'. Only PCI target mode accesses are supported by HFC-E1.

5 V PCI bus signaling environment is supported with 3.3 V supply voltage of HFC-E1. Never connect the power supply of HFC-E1 to 5 V!

The PCI interface is build according to the PCI Specification 2.2.

2.3.1 PCI command types

Table 2.6 shows the supported PCI commands of HFC-E1.

Memory Read Line and Memory Read Multiple commands are aliased to Memory Read. Memory Write and Invalidate is aliased to Memory Write.

Table 2.6: *PCI command types*

C/BE3#	C/BE2#	C/BE1#	C/BE0#	nibble value	Command type
0	0	1	0	2	I/O Read
0	1	1	0	6	Memory Read
1	1	0	0	0xC	Memory Read Multiple
1	1	1	0	0xE	Memory Read Line
1	0	1	0	0xA	Configuration Read
0	0	1	1	3	I/O Write
0	1	1	1	7	Memory Write
1	1	1	1	0xF	Memory Write and Invalidate
1	0	1	1	0xB	Configuration Write

2.3.2 PCI access description

HFC-E1 uses only PCI target accesses. A PCI master function is not implemented.

Two modes exist for register access:

1. If HFC-E1 is used in *PCI memory mapped mode* all registers can directly be accessed by adding their CIP address to the configured Memory Base Address.
2. In *PCI I/O mapped mode*, HFC-E1 only occupies 8 bytes in the I/O address space.

In PCI I/O mapped mode all registers are selected by writing the register address into the *Control Internal Pointer* (CIP) register. This is done by writing HFC-E1 on the higher I/O addresses (AD2 = '1'). If the auxiliary interface is used (see Chapter 11) the CIP write access must have a width of 16 bit.

All consecutive read or write data accesses (AD2 = '0') use the selected register until the CIP register is changed.

2.3.3 PCI configuration registers

The PCI configuration space is defined by the configuration register set which is illustrated in Figure 2.3. In the configuration address space 0x00..0x47 the PCI configuration register values are either

- set by the HFC-E1 default settings of the configuration values or
- they can be written to upper configuration register addresses or
- they are read from the external EEPROM.

The external EEPROM is optional. If no EEPROM is available, pin EE_SCL/EN has to be connected to GND and pin EE_SDA has to be left open. Without EEPROM the PCI configuration registers will be loaded with the default values shown in Table 2.7.

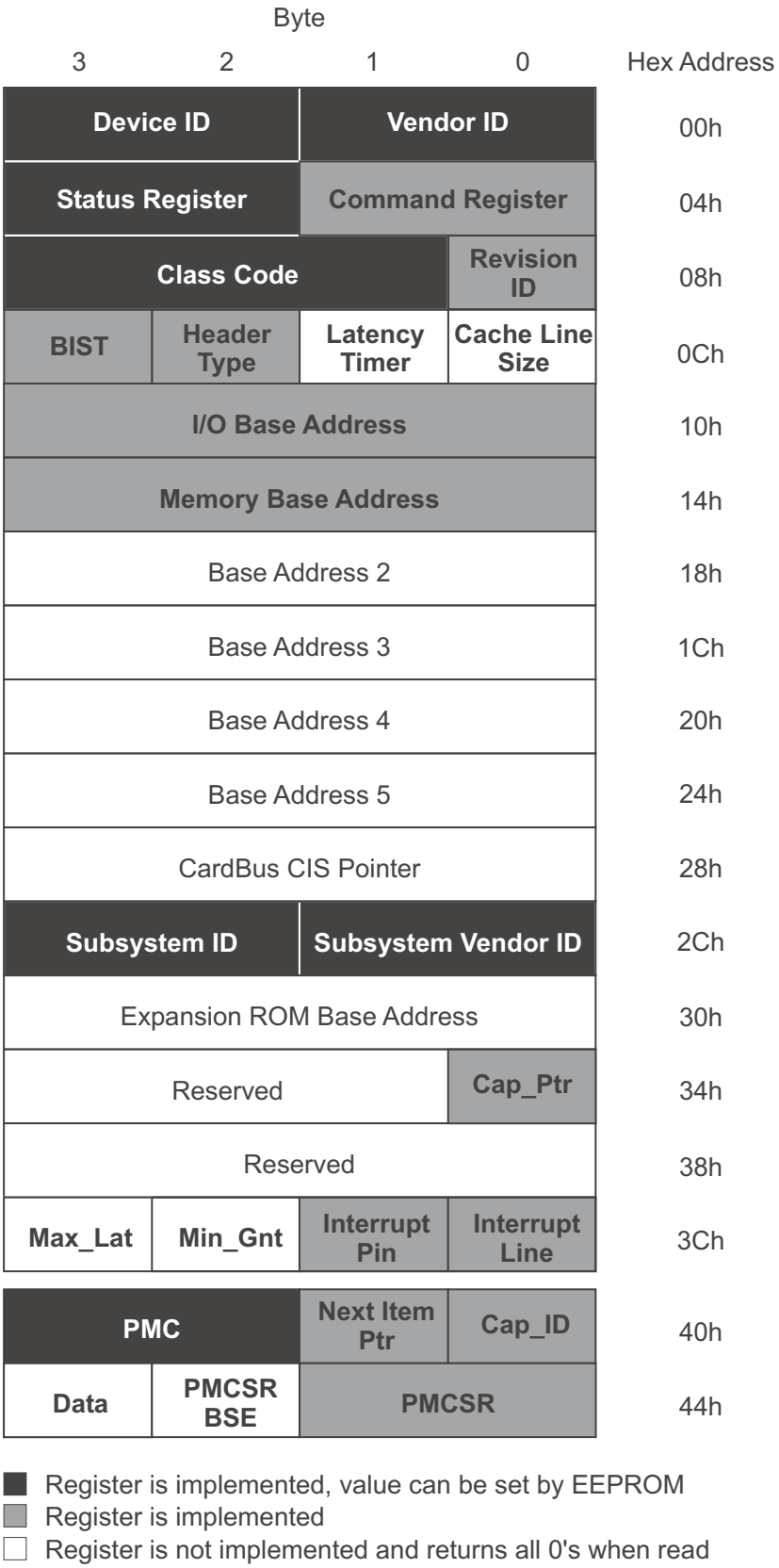


Figure 2.3: *PCI configuration registers*

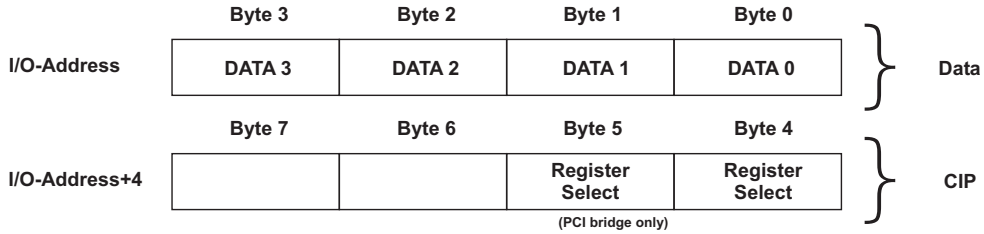


Figure 2.4: *PCI access in PCI I/O mapped mode*

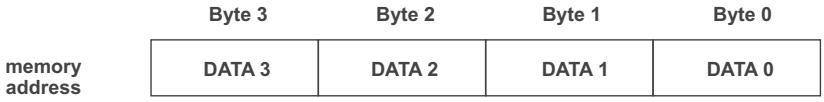


Figure 2.5: *PCI access in PCI memory mapped mode*

All configuration registers which can be set by the EEPROM can also be written by configuration write accesses to the upper addresses of the configuration register space (from 0xC0 upwards). The addresses for configuration writes are shown in Table 2.7. Unimplemented registers return all '0's when read.

Table 2.7: *PCI configuration registers*

Register Name	Address	Width	Default Value	Remarks
Vendor ID	0x00	Word	0x1397	Value can be set by EEPROM. Base address for configuration write is 0xC0.
Device ID	0x02	Word	0x30B1	Value can be set by EEPROM. Base address for configuration write is 0xC0.
Command Register	0x04	Word	0x0000	Bits Function 0 Enables / disables I/O space accesses 1 Enables / disables memory space accesses 5..2 fixed to 0 6 PERR# enable / disable 7 fixed to '0' 8 SERR# enable / disable 15..9 fixed to 0

(continued on next page)

Table 2.7: *PCI configuration registers*

(continued from previous page)

Register Name	Address	Width	Default Value	Remarks
Status Register	0x06	Word	0x0210	Bits 0 . . 7 can be set by EEPROM. Base address for configuration write is 0xC4. Bits Function 3..0 reserved 4 '1' = *Capabilities List* exists, fixed to '1' 5 '0' = 33 MHz capable (default) '1' = 66 MHz capable 6 reserved 7 '0' = fast Back-to-Back not capable (default) '1' = fast Back-to-Back capable 8 fixed to '0' 10..9 fixed to '01': timing of DEVSEL# is medium 13..11 fixed to '000' 14 system error (address parity error) 15 any detected data or system parity error
Revision ID	0x08	Byte	0x01	HFC-E1 Revision 01
Class Code	0x09	3 Bytes	0x020400	Class code for 'ISDN controller'.Value can be set by EEPROM. Base address for configuration write is 0xC8.
Header Type	0x0E	Byte	0x00	Header type 0
BIST	0x0F	Byte	0x00	No build in self test supported.
I/O Base Address	0x10	DWord		Bits 2 . . 0 are fixed to '001' Bits 31 . . 3 are r/w by configuration accesses. 8 Byte address space is used.
Memory Base Address	0x14	DWord		Bits 11 . . 0 are all '0' Bits 31 . . 12 are r/w by configuration accesses. 4 KByte address space is used.
Subsystem Vendor ID	0x2C	Word	0x1397	Value can be set by EEPROM. Base address for configuration write is 0xEC.
Subsystem ID	0x2E	Word	0x30B1	Value can be set by EEPROM. Base address for configuration write is 0xEC.
Cap_Ptr	0x34	Byte	0x40	Offset to Power Management register block.
Interrupt Line	0x3C	Byte	0xFF	This register must be configured by configuration write.
Interrupt Pin	0x3D	Byte	0x01	INTA# supported
Cap_ID	0x40	Byte	0x01	Capability ID. 0x01 identifies the linked list item as PCI Power Management registers.

(continued on next page)

Table 2.7: *PCI configuration registers*

(continued from previous page)

Register Name	Address	Width	Default Value	Remarks
Next Item Ptr	0x41	Byte	0x00	There are no next items in the linked list.
PMC [*1]	0x42	Word	0x7E22	Power Management Capabilities, see also 'PCI Bus Power Management Interface Specification Rev. 1.1'.This register's value can be set by EEPROM. Base address for configuration write is 0xE0.

Bits	Function
0..2	'010' = PCI Power Management Spec. Version 1.1.
3	'0' = HFC-E1 does not require PCI-clock to generate PME .
4	Fixed to '0'.
5	'1' = Device specific initialisation is required.
8..6	'000' = No report of auxiliary current.
9	'1' = Supports D1 Power Management State [*2].
10	'1' = Supports D2 Power Management State [*2].
15..11	PME can be asserted from D0, D1, D2 and D3_hot. No D3_cold support [*1].

PMCSR	0x44	Word	0x0000	Power Management Control/Status

Bits	Function
1..0	**PowerState**: These bits are used both to determine the current power state of a function and to set the function into a new power state [*2].
	'00': D0
	'01': D1
	'10': D2
	'11': D3_hot
7..2	fixed to '0'
8	**PME_En**:
	'1' enables the function to assert PME .
	'0' = PME assertion is disabled.
14..9	fixed to 0
15	**PME_Status**: This bit is set when the function would normally assert the signal PME independent of the state of bit **PME_En**.
	Writing '1' to this bit will clear it and cause the function to stop asserting a PME (if enabled).
	Writing '0' has no effect.

[*1]: D3_cold support is implemented but must be set in the EEPROM configuration data.

[*2]: Changing the power management does not change the power dissipation. It is only implemented for PCI specification compatibility.

2.3.4 PCI connection circuitry

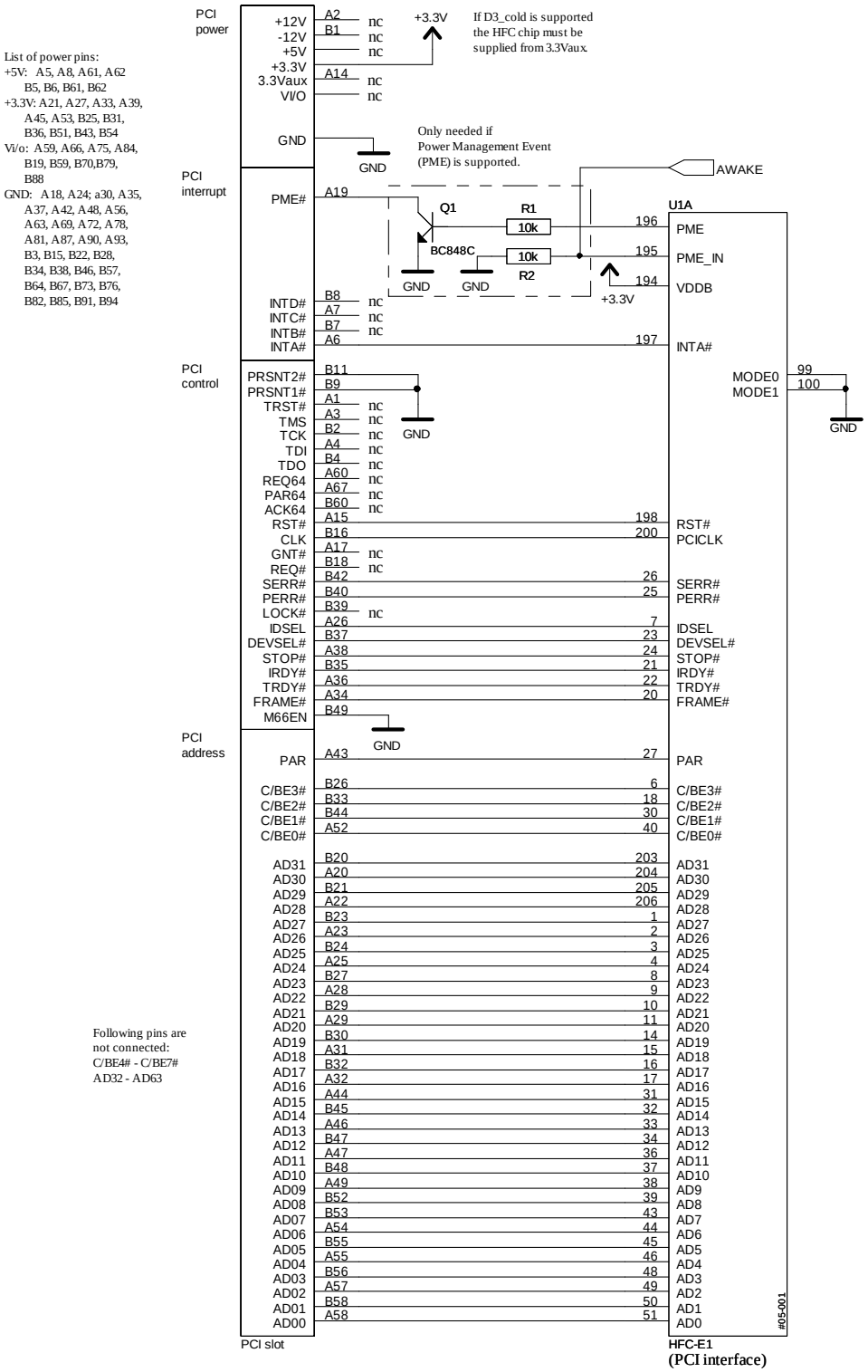


Figure 2.6: *PCI connection circuitry*

2.4 ISA Plug and Play interface

Table 2.8: *Overview of the ISA PnP interface pins*

Number	Name	Description
203 . . 206,1 . . 4	SA15 . . SA8	Address byte 1
8 . . 17	SA7 . . SA0	Address byte 0
31 . . 39	SD15 . . SD8	Data byte 1
43 . . 51	SD7 . . SD0	Data byte 0
106 . . 112	IRQ6 . . IRQ0	ISA Interrupt Request 6 . . 0
18	/IOIS16	16 bit access enable
20	/AEN	Address Enable
21	/IOR	Read Enable
22	/IOW	Write Enable
25	/BUSDIR	Bus Direction
30	/SBHE	High byte enable
198	RESET	Reset high active

ISA Plug and Play mode is selected by MODE0 = '0' and MODE1 = '1'. HFC-E1 needs eight consecutive addresses in the I/O map of a PC for operation. Usually one of several ISA IRQ lines is also used. Section 2.4.1 describes how to configure the interrupt lines of HFC-E1.

The port address is selected by SA0 . . SA15 lines. The address with SA2 = '1' is used for register selection via the CIP (Control Internal Pointer) and the address with SA2 = '0' is used for data read / write like shown in Table 2.9. Bits SA3 . . SA15 are decoded by the address decoder to match the PnP configuration address.

Table 2.9: *ISA address decoding (X = don't care)*

SA2	/IOR	/IOW	/AEN	Operation
X	X	X	1	no access
X	1	1	X	no access
0	0	1	0	read data
0	1	0	0	write data
1	0	1	0	read CIP
1	1	0	0	write CIP

HFC-E1 has no memory or DMA access to any component on the ISA PC bus. Because of its characteristic power drive, no external driver for the ISA PC bus data lines is needed. If necessary (e.g. due to an old ISA specification which requires 24 mA output current), an external bus driver can be added. In this case the output signal /BUSDIR determines the driver direction.

/BUSDIR = '0' means that HFC-E1 is read and data is driven to the external bus.

/BUSDIR = '1' means that data is driven (written) into HFC-E1.

2.4.1 IRQ assignment

Seven different interrupt lines IRQ6 . . IRQ0 are supported by HFC-E1. They are tristated after a hardware reset.

The IRQ assigned by the PnP BIOS can be read from bitmap V_PNP_IRQ in register R_CHIP_ID. Bitmap V_IRQ_SEL in register R_CIRM has to be set according to the IRQ wiring between HFC-E1 and the ISA slot on the PCB. Thus the IRQ number assigned by the PnP BIOS is connected to the right IRQ line on the ISA bus.

2.4.2 ISA Plug and Play registers

Table 2.10: *ISA Plug and Play registers*

Card level control register address	Read / write Mode	Accessable in state	Description
0x00	w	Isolation state, Config state *1	**Set read data port address register**. Bits 0 . . 7 become bits 2 . . 9 of the port's I/O address. Bits 10 and 11 are hardwired to '00' and bits 0 and 1 are hardwired to '11'.
0x01	r	Isolation state	**Serial isolation register**. Used to read the serial identifier during the card isolation process.
0x02	w	Sleep state, Isolation state, Config state	**Configuration control register**. Bits — Function 0 — **Reset Bit**. The value '1' resets all of the card's configuration registers to their default state. The CSN is not affected. 1 — **Return to wait for key state**. When set to '1', all cards return to wait for key state. Their CSNs and configuration registers are not affected. This command is issued after all cards have been configured and activated. 2 — **Reset CSN to '0'**. When set to '1', all cards reset their CSN to '0'. All bits are automatically cleared by the hardware. 7..3 — Reserved, must be '00000'

(continued on next page)

Table 2.10: *ISA Plug and Play registers*

(continued from previous page)

Card level control register address	Read / write Mode	Accessable in state	Description
0x03	w	Sleep state, Isolation state, Config state	**Wake command register.** Writing a CSN to this register has the following effects: • If the value written is 0x00, all cards in the sleep state with a CSN = 0x00 go to the isolation state. All cards in Config state (CSN not 0x00) go to the sleep state. • If the value written is not 0x00, all cards in the sleep state with a matching CSN go to the Config state. All cards in the isolation state go to the sleep state. Every write access to a card's wake command register with a match on its CSN causes the pointer to the serial identifier / resource data to be reset to the first byte of the serial identifier.
0x04	r	Config state	**Resource data register.** This register is used to read the device's recource data. Each time when a read is performed from this register a byte of the resource data is returned and the resource data pointer is incremented. Prior to reading each byte, the programmer must read from the status register to determine if the next byte is available for reading from the resource data register. The card's serial identifier and checksum must be read prior to accessing the resource requirement list via this register.
0x05	r	Config state	**Status register.** Prior to reading the next byte of the device's resource data, the programmer must read from this register and check bit 0 for '1' value. This is the resource data byte available bit. Bits 1 . . 7 are reserved.
0x06	r/w	Isolation state [*2] Config state	**Card select number (CSN) register.** The configuration software uses the CSN register to assign a unique ID to the card. The CSN is then used to wake up the card's configuration logic whenever the configuration program must access its configuration registers.
0x07	r	Config state	**Logical device number register.** The number in this register points to the logical device the next commands will operate on. HFC-E1 only supports one logical device. This register is hardwired to all '0''s.

(continued on next page)

Table 2.10: *ISA Plug and Play registers*

(continued from previous page)

Card level control register address	Read / write Mode	Accessable in state	Description
0x30	r/w	Config state	**Activate register.** Setting bit 0 to '1' activates the card on the ISA bus. When cleared, the card cannot respond to any ISA bus transactions (other than accesses to its Plug and Play configuration ports). Reset clears bit 0. Bits 1 . . 7 are reserved and return 0 when read. HFC-E1 only supports one logical device, so it is not necessary to write the logical device number into the card's logical device number register prior to writing to this register.
0x31	r/w	Config state	**I/O range check register.** Bits Function 0 When set, the logical device returns 0x55 in response to any read from the logical device's assigned I/O space. When cleared, 0xAA is returned. 1 When set to '1', enables I/O range checking and disables it when cleared to '0'. When enabled, bit 0 is used to select a pattern for the logical device to return. **This bit is only valid if the logical device is deactivated (see *Activate register*).** 7..2 Reserved, return '000000' when read
0x60	r/w	Config state	**I/O decoder 0 base address upper byte.** I/O port base address bits 8 . . 15.
0x61	r/w	Config state	**I/O decoder 0 base address lower byte.** I/O port base address bits 0 . . 7.
0x70	r/w	Config state	**IRQ select configuration register 0.** Bits 0 . . 3 specify the selected IRQ number. Bits 4 . . 7 are reserved.
0x71	r/w	Config state	**IRQ type configuration register 0.** Bits 0 and 1 are ignored. Bits 2 . . 7 are reserved.
0x74	r	Config state	**DMA configuration register 0.** Bits Function 2..0 Select which DMA channel (0 . . 7) is used for DMA 0. DMA channel 4, the cascade channel, indicates no DMA channel is active. 7..3 Reserved. Because no DMA is used this register is hardwired to 0x04.

(continued on next page)

Table 2.10: *ISA Plug and Play registers*

(continued from previous page)

Card level control register address	Read / write Mode	Accessable in state	Description
0x75	r	Config state	**DMA configuration register 1.** Bits Function 2..0 Select which DMA channel (0..7) is used for DMA 1. DMA channel 4, the cascade channel, indicates no DMA channel is active. 7..3 Reserved. Because no DMA is used this register is hardwired to 0x04.

*1: This is an extension to the Plug and Play Specification.

*2: Only when the isolation process is finished. The last card remains in isolation state until a CSN is assigned.

Important !

All ISA registers which are not implemented return 0x00 with a read access except the DMA configuration registers at address 0x74 and 0x75. These two registers return 0x04 with a read access. This means no DMA channel has been selected.

2.4.3 ISA connection circuitry

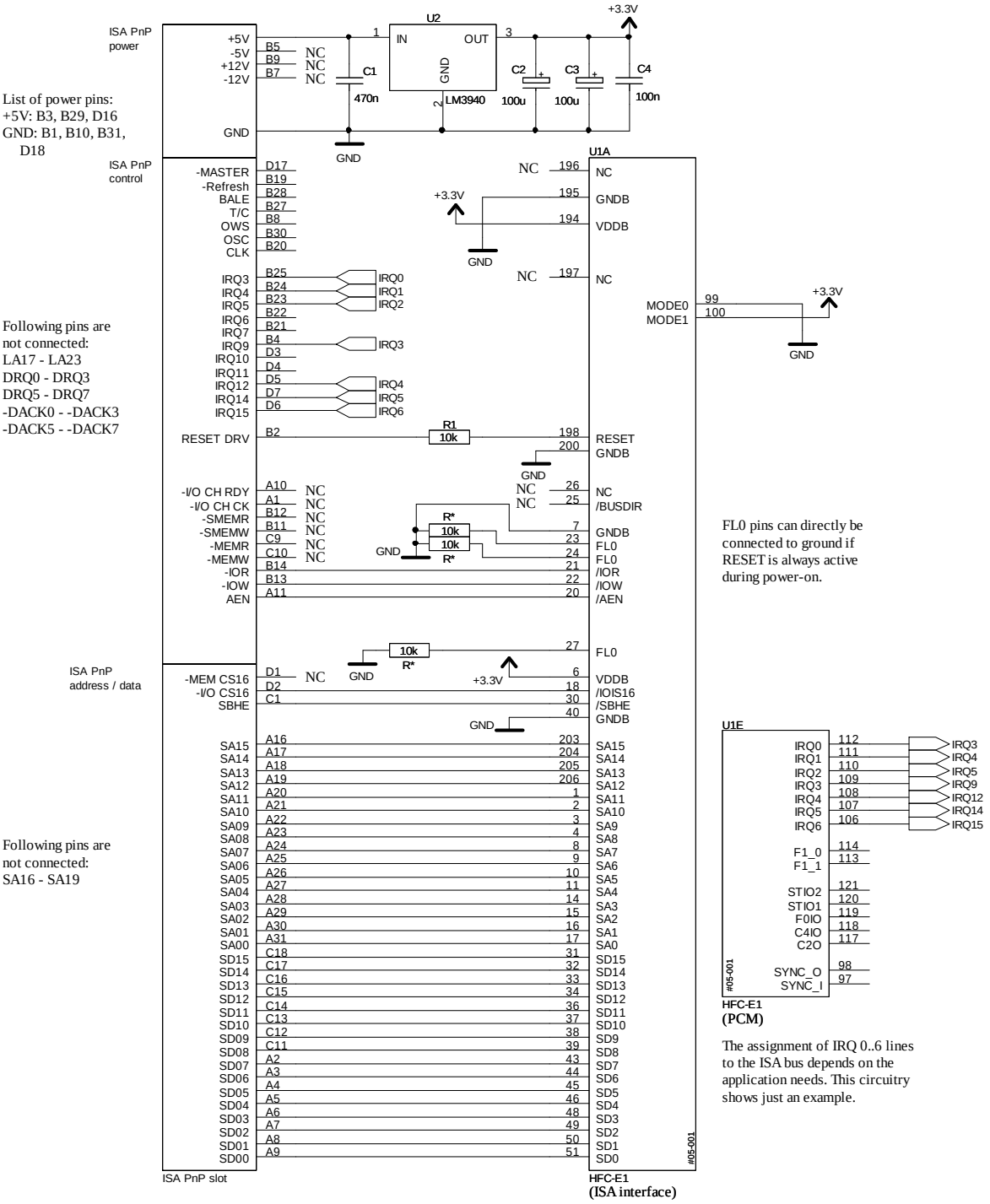


Figure 2.7: *ISA PnP circuitry, see Section 2.4.1 on page 62 for IRQ line assignment*

2.5 PCMCIA interface

Table 2.11: *Overview of the PCMCIA interface pins*

Number	Name	Description
203 . . 206, 1 . . 4	A15 . . A8	Address byte 1
8 . . 17	A7 . . A0	Address byte 0
31 . . 39	D15 . . D8	Data byte 1
43 . . 51	D7 . . D0	Data byte 0
7	REG#	PCMCIA Register and Attr. Mem. Select
18	IOIS16#	16 bit access enable
21	IORD#	Read Enable
22	IOWR#	Write Enable
23	OE#	PCMCIA Output Enable for Attr. Mem. Read
24	WE#	PCMCIA Write Enable for Conf. Reg. Write
25	INPACK#	Read access
30	CE2#	High byte enable
40	CE1#	Low byte enable
197	IREQ#	Interrupt request
198	RESET	Reset high active

The PCMCIA mode is selected by MODE0 = '1' and MODE1 = '1'. HFC-E1 occupies eight consecutive addresses in the I/O map.

The base I/O address must be 8 byte aligned. Lines A3 . . A15 are don't care for I/O accesses.

The address with A2 = '1' is used for register selection via CIP. The address with A2 = '0' is used for data read / write.

2.5.1 Attribute memory

After a hardware reset the card's information structure (CIS) is copied from the EEPROM to the SRAM, starting with the address shown in Table 2.4. The CIS is located on even numbered addresses from 0 to 0x3FE in the attribute memory space. The CIS occupies 512 byte. To avoid accesses in this copy phase, signal IREQ# of HFC-E1 is active. This is interpreted as 'wait' by the PCMCIA host controller after card insertion.

2.5.2 PCMCIA registers

Table 2.12: *PCMCIA registers*

Register Name	Address *	Width	Remarks			
Configuration Option Register (COR)	0x400	Byte	**Bit**	**Name**	**Reset value**	**Function**
			5..0	Configuration Index	0x00	Bit 0 must be set to '1' to enable accesses to HFC-E1.
			6	LevIREQ	1	This bit is not implemented and returns always '1' when read to indicate usage of level mode interrupts.
			7	SRESET		SRESET card. Setting this bit to '1' places the card in the reset state. This bit must be cleared to zero for normal operation.
Card Configuration and Status Register (CSR)	0x402	Byte	**Bit**	**Name**	**Reset value**	**Function**
			0	Rsvd	0	
			1	Intr	0	Internal state of interrupt request (IREQ#).
			2	PwrDwn	0	Unimplemented, returns '0' when read.
			3	Audio	0	Unimplemented, returns '0' when read.
			4	Rsvd	0	Unimplemented, returns '0' when read.
			5	IOis8	0	Returns '0' when read to indicate an 16 bit data path.
			6	SigChg	0	Unimplemented, returns '0' when read.
			7	Changed	0	Unimplemented, returns '0' when read.

*: Register address in attribute memory

2.5.3 PCMCIA connection circuitry

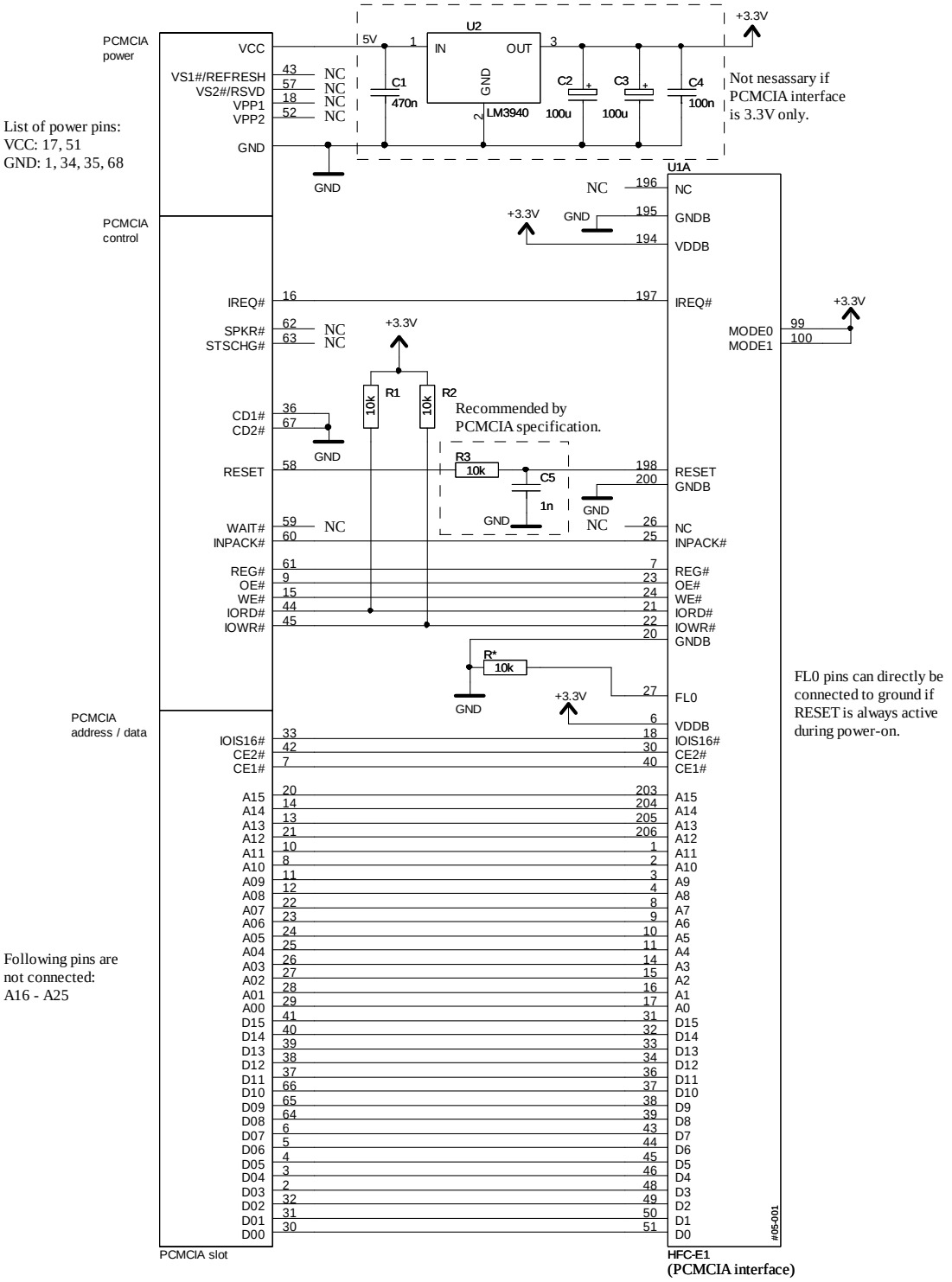


Figure 2.8: *PCMCIA circuitry*

2.6 Parallel processor interface

Table 2.13: *Overview of the parallel processor interface pins in mode 2 and 3*

Number	Name	Description
8 . . 17	A7 . . A0	Address byte
43 . . 51	D7 . . D0	Data byte 0
31 . . 39	D15 . . D8	Data byte 1
6, 18, 30, 40	/BE3 . . /BE0	Byte Enable 3 . . 0
20	/CS	Chip Select
21	/IOR	Read Enable
22	/IOW	Write Enable
23	/WD	Watch Dog Output
24	ALE	Address Latch Enable
25	/BUSDIR	Bus Direction
197	/INT	Interrupt request
198	RESET	Reset high active

Table 2.14: *Overview of the parallel processor interface pins in mode 4*

Number	Name	Description
43 . . 51	AD7 . . AD0	Address / Data byte 0
31 . . 39	AD15 . . AD8	Address / Data byte 1
8 . . 17	AD23 . . AD16	Address / Data byte 2
203 . . 206, 1 . . 4	AD31 . . AD24	Address / Data byte 3
6, 18, 30, 40	/BE3 . . /BE0	Byte Enable 3 . . 0
20	/CS	Chip Select
21	/IOR	Read Enable
22	/IOW	Write Enable
23	/WD	Watch Dog Output
24	ALE	Address Latch Enable
25	/BUSDIR	Bus Direction
197	/INT	Interrupt request
198	RESET	Reset high active

The parallel processor interface mode is selected by MODE0 = '1' and MODE1 = '0'. Then 256 I/O addresses (A0 .. A7) are used for addressing the internal registers of HFC-E1 directly by their address.

In parallel processor interface mode some user data can be stored in the EEPROM (see Section 2.2.1 for details).

2.6.1 Parallel processor interface modes

HFC-E1 has three different parallel processor interface modes. Due to name compatibility with other chips of the HFC series these interface modes are numbered 2 .. 4 like shown in Table 2.15.

Table 2.15: *Pins and signal names in the parallel processor interface modes*

HFC-E1 pins		Signal names		
Number	Name	Mode 2 (Motorola) Non-multiplexed	Mode 3 (Intel) Non-multiplexed	Mode 4 (Intel) Multiplexed
20	/CS	/CS	/CS	/CS
21	/IOR	/DS	/RD	/RD
22	/IOW	R/W	/WR	/WR
24	ALE	'1'	'0'	ALE

The interface modes 2 and 3 use separate lines for address and data. These two modes are selected by ALE . This pin must have a fixed level and should be directly connected to ground or power supply. Mode 4 has multiplexed address / data lines. The address is latched from lines D7 .. D0 with the falling edge of ALE .

The parallel processor interface mode is determined during hardware reset time (pin RESET). For modes 2 and 3, pin ALE must have the appropriate level. Mode 4 is selected after reset with the first rising edge of ALE . HFC-E1 then switches permanently from mode 2 or mode 3 into mode 4. HFC-E1 cannot switch to mode 4 before end of reset time. Rising and falling edges of ALE are ignored during reset time.

ALE must be stable after reset except in interface mode 4.

2.6.2 Signal and timing characteristics

Table 2.16 shows the interface signals for the different parallel processor interface modes. Timing characteristics are shown in Figures 2.9 to 2.12 for mode 2 and mode 3. Figures 2.13 to 2.18 show mode 4 timing characteristics. Please see Table 2.17 for a quick timing and symbol list finding.

In interface mode 4 it is possible to access byte, word or double word on lines AD31 .. AD0 . Due to the multiplexed lines the PCI pin names are used in this case. In interface modes 2 and 3, pins AD31 .. AD24 are not available.

Unused byte enable pins should be connected to power supply via pull-up resistors. In mode 4 unused bus lines AD[31..] should be connected to ground via pull-down resistors to avoid floating inputs.

Table 2.16: *Overview of read and write accesses of the parallel processor interface (X = don't care)*

/CS	/IOR (/DS, /RD)	/IOW (R/W, /WR)	ALE	Operation	Processor interface mode
1	X	X	X	no access	all
X	1	1	X	no access	all
0	0	1	1	read data	mode 2
0	0	0	1	write data	mode 2
0	0	1	0	read data	mode 3
0	1	0	0	write data	mode 3
0	0	1	0 *	read data	mode 4
0	1	0	0 *	write data	mode 4

*: 1-pulse latches register address

Important !

/BE2 and /BE3 must always be '1' in mode 2 and mode 3.

Table 2.17: *Timing diagrams of the parallel processor interface*

Mode	Processor	Access type		Timing		Timing values	
				Figure	on page	table	on page
2 & 3	8 bit	8 bit	read	2.9	73	2.19	77
2 & 3	8 bit	8 bit	write	2.10	75	2.20	79
2 & 3	16 bit	16 bit & 8 bit	read	2.11	76	2.19	77
2 & 3	16 bit	16 bit & 8 bit	write	2.12	78	2.20	79
4	8 bit	8 bit	read	2.13	80	2.22	85
4	8 bit	8 bit	write	2.14	81	2.23	87
4	16 bit	16 bit	read	2.15	82	2.22	85
4	16 bit	16 bit	write	2.16	83	2.23	87
4	32 bit	32 bit	read	2.17	84	2.22	85
4	32 bit	32 bit	write	2.18	86	2.23	87

2.6.2.1 8 bit processors in mode 2 (Motorola) and mode 3 (Intel)

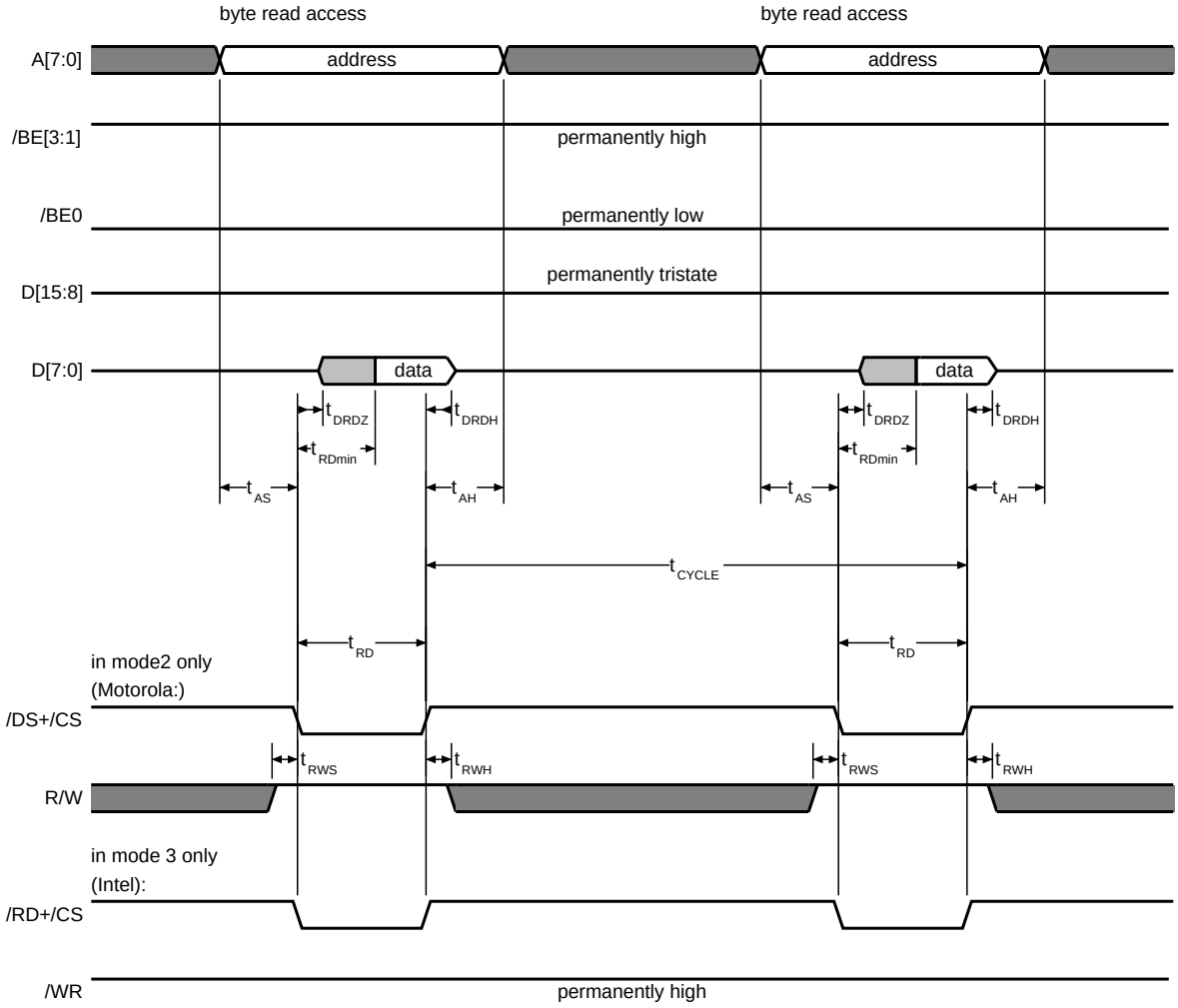


Figure 2.9: *Read access from 8 bit processors in mode 2 (Motorola) and mode 3 (Intel)*

8 bit processors read data like shown in Figure 2.9. Timing values are listed in Table 2.19.

/BE3 . . /BE1 must always be '1'. /BE0 can be fixed to '0' or must be low during access to switch the data bus D7 . . D0 from tristate into data driven state.

Data can be read in mode 2 (Motorola) with [1]

$$/\mathrm{BE0} = \text{'0'} \quad \text{and} \quad (/\mathrm{DS} + /\mathrm{CS}) = \text{'0'} \quad \text{and} \quad \mathrm{R/W} = \text{'1'} \ .$$

In mode 3 (Intel, non-multiplexed) the states

$$/\mathrm{BE0} = \text{'0'} \quad \text{and} \quad (/\mathrm{RD} + /\mathrm{CS}) = \text{'0'} \quad \text{and} \quad /\mathrm{WR} = \text{'1'}$$

must be fulfilled to drive data out. The data bus is stable after t_{RDmin} and returns into tristate after t_{DRDH}.

[1] /DS + /CS means logical OR function of the two signals.

Address and /BE0 (if not fixed to low) require a setup time t_{AS} which starts when all address and byte enable signals are valid. The hold time of these lines is t_{AH}.

Short read method

In some applications it may be difficult to implement a long read access ($t_{RD} \geq 5 \cdot t_{CLKI}$) for only some registers (here called *target register*).

For this reason there is an alternative method with two register read accesses with $t_{RD} \geq 20\,\text{ns}$ each:

1. The read access to the target register initiates a data transmission from the RAM to the target register. This job is always done correctly with long and short t_{RD}, but after a short t_{RD} the data is not yet 'arrived' at the target register. Thus the data which is read with a short t_{RD} must be ignored ...
2. ... but the data byte can be read from register R_INT_DATA. This second register read access can also be executed with a short $t_{RD} \geq 20\,\text{ns}$. For the time from the first access to the second one t_{CYCLE} must be met, of course.

The short read method is practical for all read registers in the address range 0xC0 .. 0xFF, these target registers are R_IRQ_FIFO_BL0 .. R_IRQ_FIFO_BL7 and R_RAM_DATA.

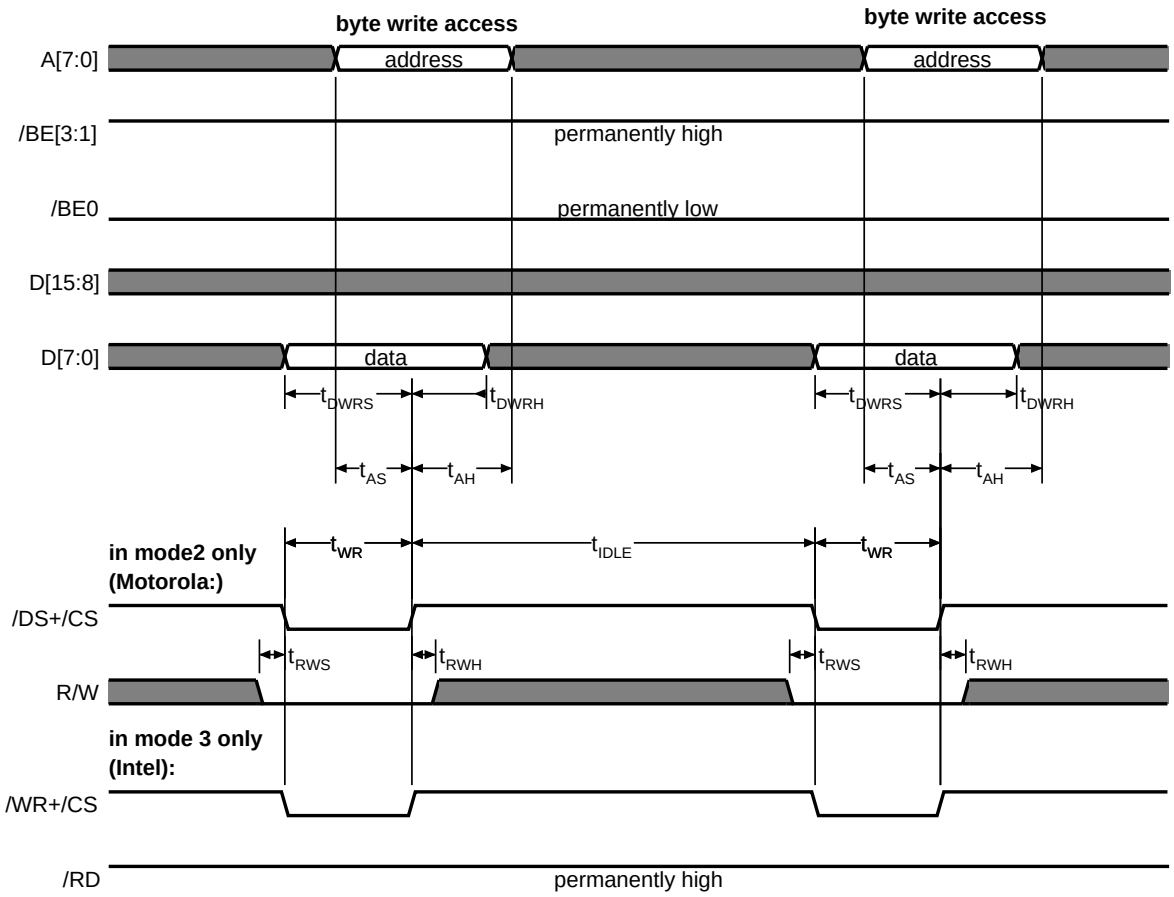


Figure 2.10: *Write access from 8 bit processors in mode 2 (Motorola) and mode 3 (Intel)*

8 bit processors write data like shown in Figure 2.10. Timing values are listed in Table 2.20.

/BE3 . . /BE1 must always be '1'. /BE0 controls the data bus D7 . . D0 and can be fixed to '0'.

Data is written with ⌟ of (/DS + /CS) in mode 2 (Motorola) or with ⌟ of (/WR + /CS) in mode 3 (Intel, non-multiplexed) respectively. HFC-E1 requires a data setup time t_{DWRS} and a data hold time t_{DWRH}.

Address and /BE0 (if not fixed to low) require a setup time t_{AS} which starts when all address and byte enable signals are valid. The hold time of these lines is t_{AH}.

2.6.2.2 16 bit processors in mode 2 (Motorola) and mode 3 (Intel)

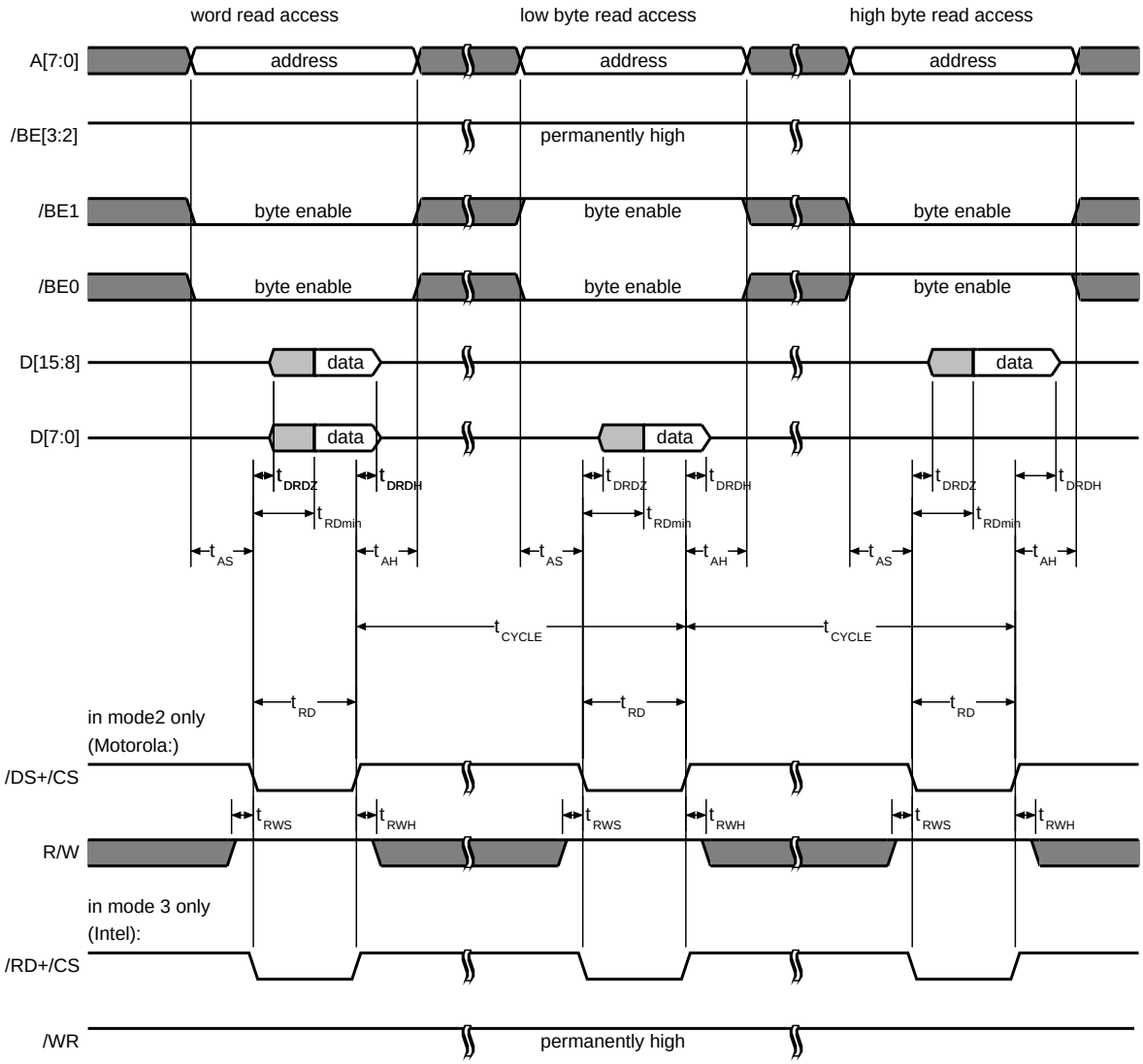


Figure 2.11: *Byte and word read access from 16 bit processors in mode 2 (Motorola) and mode 3 (Intel)*

16 bit processors can either read data with byte or word access like shown in Figure 2.11. FIFO and F-/Z-counter read access have 8 bit or 16 bit width alternatively. The 16 bit processor must support byte access because all other register read accesses must have a width of 8 bit.

/BE2 and /BE3 must always be '1'. /BE0 and /BE1 switch the data bus D15 .. D0 from tristate into data driven state during read access (see Table 2.18).

Data can be read in mode 2 (Motorola) with

$$\text{/BE} = \text{'0'} \quad \text{and} \quad (\text{/DS} + \text{/CS}) = \text{'0'} \quad \text{and} \quad \text{R/W} = \text{'1'} \ .$$

In mode 3 (Intel, non-multiplexed) the states

$$\text{/BE} = \text{'0'} \quad \text{and} \quad (\text{/RD} + \text{/CS}) = \text{'0'} \quad \text{and} \quad \text{/WR} = \text{'1'}$$

Table 2.18: *Data access width in mode 2 and 3*

A[0]	/BE1	/BE0	Data access
'X'	'1'	'1'	no access
'0'	'1'	'0'	byte access on D[7:0]
'1'	'0'	'1'	byte access on D[15:8]
'0'	'0'	'0'	word access

must be fulfilled to drive data out. The data bus is stable after t_{RDmin} and returns into tristate after t_{DRDH}.

Address and /BE require a setup time t_{AS} which starts when all address and byte enable signals are valid. The hold time of these lines is t_{AH}.

Table 2.19: *Symbols of read accesses in Figures 2.9 and 2.11*

Symbol	min / ns	max / ns	Characteristic
t_{AS}	10		Address and /BE valid to /DS+/CS (/RD+/CS) ⌝ setup time
t_{AH}	10		Address and /BE hold time after /DS+/CS (/RD+/CS) ⌟
t_{DRDZ}	2		/DS+/CS (/RD+/CS) ⌝ to data buffer turn on time
t_{DRDH}	2	15	/DS+/CS (/RD+/CS) ⌟ to data buffer turn off time
t_{RWS}	2		R/W setup time to /DS+/CS ⌝
t_{RWH}	2		R/W hold time after /DS+/CS ⌟
t_{RD}			Read time:
	20		A[7] = '0' (address range 0 ... 0x7F: normal register access)
	20		A[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$5 \cdot t_{CLKI}$		A[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access, FIFO interrupt registers) *
t_{CYCLE}			Cycle time between two consecutive /DS+/CS (/RD+/CS) ⌟
	$1.5 \cdot t_{CLKI}$		A[7] = '0' (address range 0 ... 0x7F: normal register access)
			A[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$5.5 \cdot t_{CLKI}$		– after byte access
	$6.5 \cdot t_{CLKI}$		– after word access
	$5.5 \cdot t_{CLKI}$		A[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access, FIFO interrupt registers)

*: See 'Short read method' on page 74.

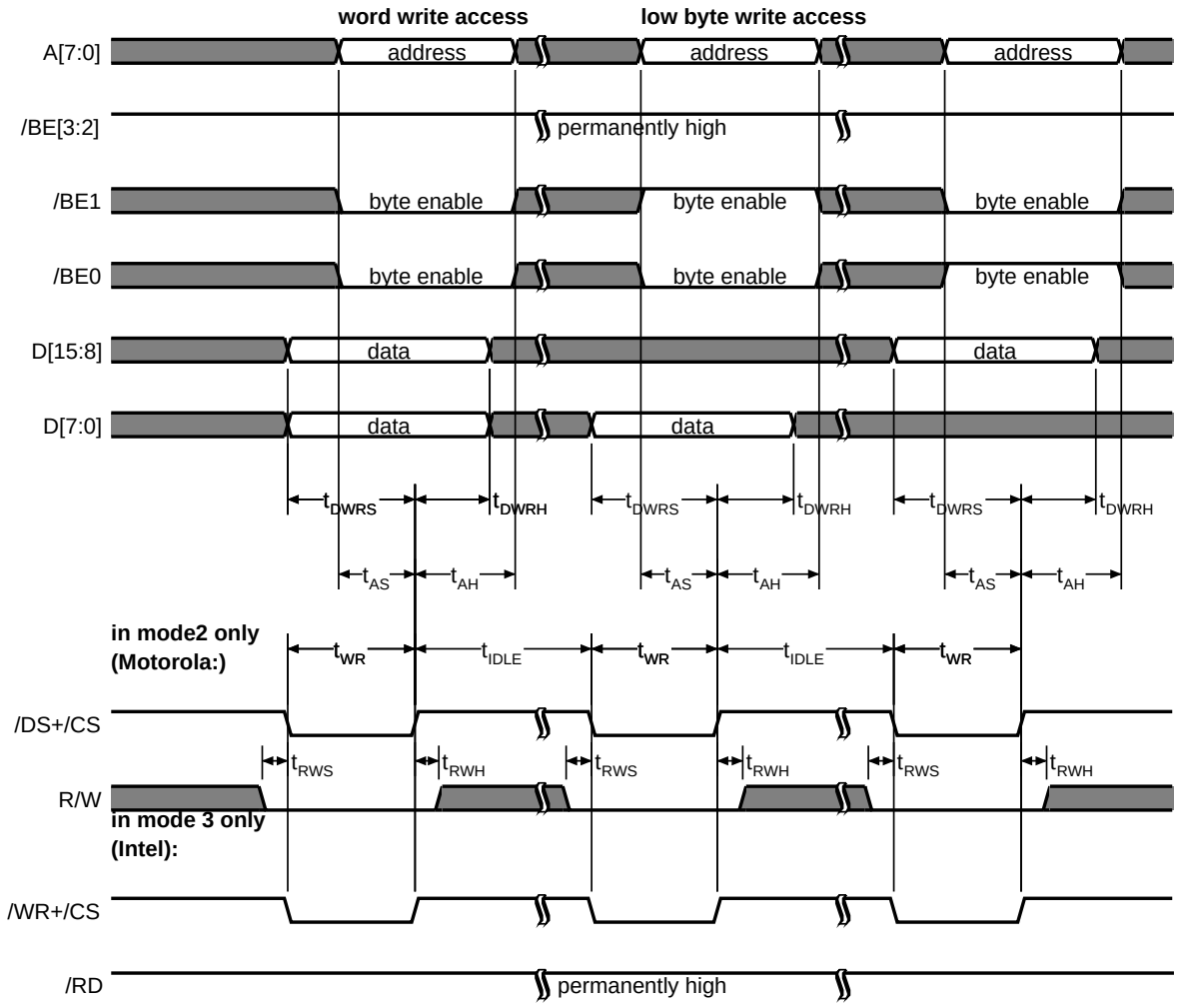


Figure 2.12: *Byte and word write access from 16 bit processors in mode 2 (Motorola) and mode 3 (Intel)*

16 bit processors can either write data with byte or word access like shown in Figure 2.12. FIFO write access have 8 bit or 16 bit width alternatively. The 16 bit processor must support byte access because all other register write accesses must have a width of 8 bit.

/BE2 and /BE3 must always be '1'. /BE0 and /BE1 control the low byte and high byte of the data bus D15 . . D0 (see Table 2.18).

Data is written with ⌟⌜ of (/DS + /CS) in mode 2 (Motorola) respectively with ⌟⌜ of (/WR + /CS) in mode 3 (Intel, non-multiplexed). HFC-E1 requires a data setup time t_{DWRS} and a data hold time t_{DWRH}.

Address and /BE require a setup time t_{AS} which starts when all address and byte enable signals are valid. The hold time of these lines is t_{AH}.

Table 2.20: *Symbols of write accesses in Figures 2.10 and 2.12*

Symbol	min / ns	max / ns	Characteristic
t_{AS}	10		Address and /BE valid to /DS+/CS (/WR+/CS) ⌟⌜ setup time
t_{AH}	10		Address and /BE hold time after /DS+/CS (/WR+/CS) ⌟⌜
t_{DWRS}	20		Write data setup time to /DS+/CS (/WR+/CS) ⌟⌜
t_{DWRH}	10		Write data hold time from /DS+/CS (/WR+/CS) ⌟⌜
t_{RWS}	2		R/W setup time to /DS+/CS (/WR+/CS) ⌝⌞
t_{RWH}	2		R/W hold time after /DS+/CS (/WR+/CS) ⌟⌜
t_{WR}	20		Write time
t_{IDLE}			/DS+/CS (/WR+/CS) high time between two consecutive accesses
	$1.5 \cdot t_{CLKI}$		A[7] = '0' (address range 0 ... 0x7F: normal register access)
			A[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$3.5 \cdot t_{CLKI}$		– after byte access
	$4.5 \cdot t_{CLKI}$		– after word access
	$3.5 \cdot t_{CLKI}$		A[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access)

2.6.2.3 8 bit processors in mode 4 (Intel, multiplexed)

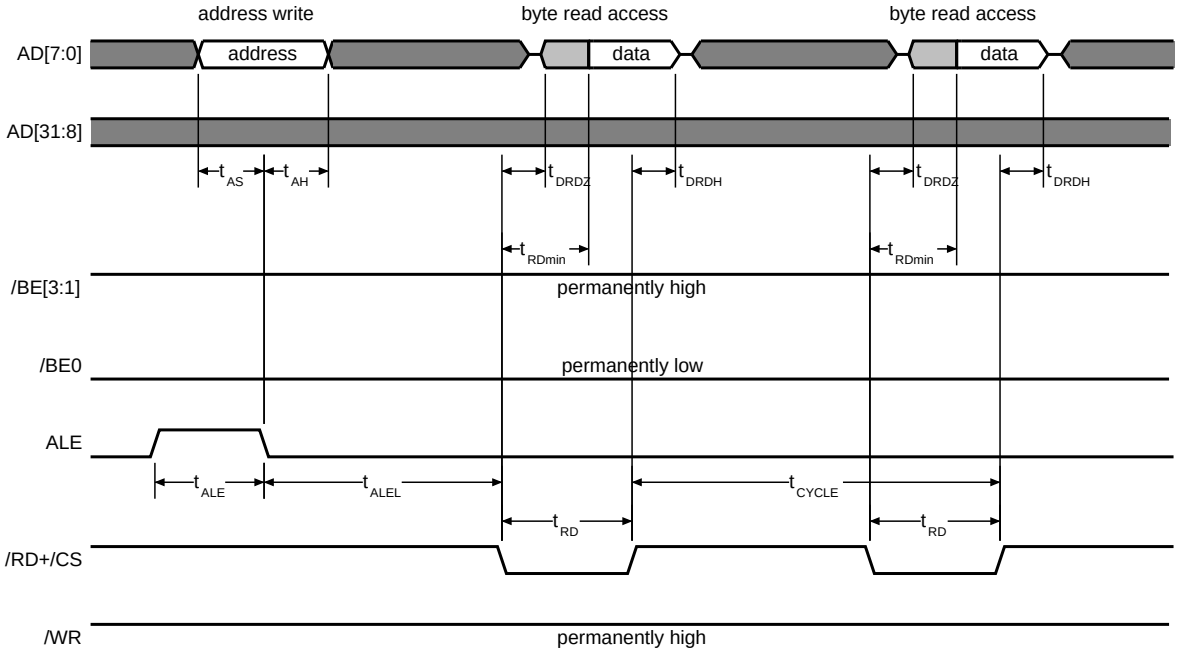


Figure 2.13: *Byte read access from 8 bit processors in mode 4 (Intel, multiplexed)*

8 bit processors read data like shown in Figure 2.13. Timing values are listed in Table 2.22.

/BE3 . . /BE1 must always be '1'. /BE0 can be fixed to '0' or must be low during read access to switch the bus AD7 . . AD0 from tristate into data driven state.

Data can be read in mode 4 (Intel, multiplexed) with [2]

$$\text{/BE0} = \text{'0'} \quad \text{and} \quad (\text{/RD} + \text{/CS}) = \text{'0'} \quad \text{and} \quad \text{/WR} = \text{'1'} \ .$$

The data bus is stable after t_{RDmin} and returns into tristate after t_{DRDH}.

Address and /BE0 (if not fixed to low) require a setup time t_{AS} which starts with the ⌝ of ALE. The hold time of these lines is t_{AH}. If two consecutive read accesses are on the same address, multiple register address write is not required.

[2] /RD + /CS means logical OR function of the two signals.

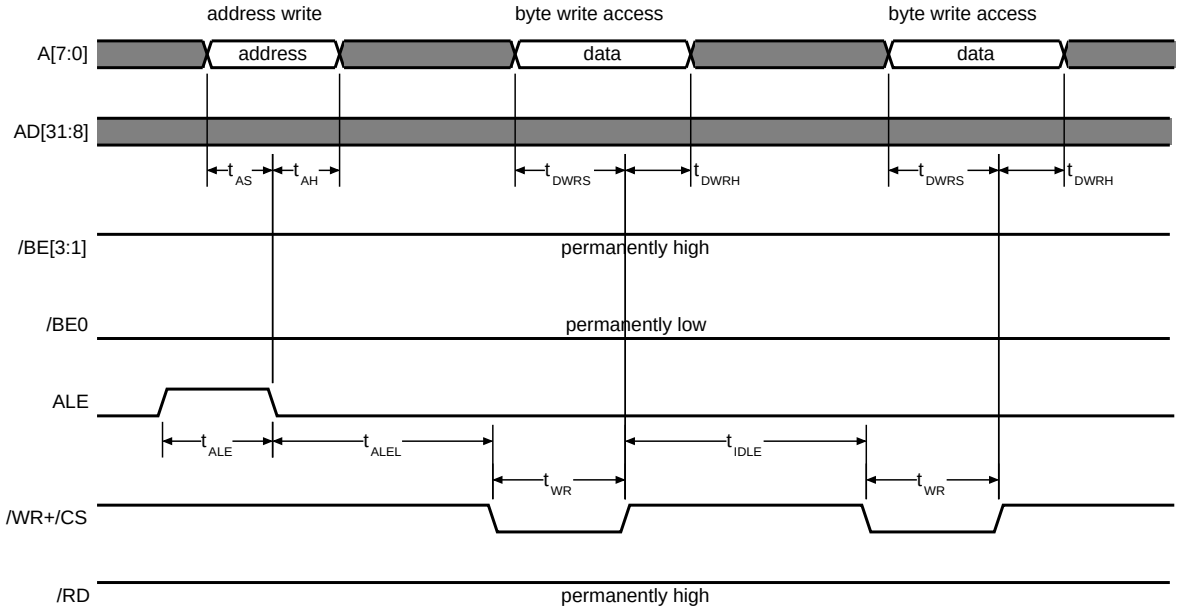


Figure 2.14: *Byte write access from 8 bit processors in mode 4 (Intel, multiplexed)*

8 bit processors write data like shown in Figure 2.14. Timing values are listed in Table 2.23.

/BE3 . . /BE1 must always be '1'. /BE0 controls the bus AD7 . . AD0 during the data phase and can be fixed to '0'.

Data is written with ⌋⌈ of (/WR + /CS) in mode 4 (Intel, multiplexed). HFC-E1 requires a data setup time t_{DWRS} and a data hold time t_{DWRH}.

Address and /BE0 (if not fixed to low) require a setup time t_{AS} which starts with the ⌉⌊ of ALE. The hold time of these lines is t_{AH}. If two consecutive write accesses are on the same address, multiple register address write is not required.

2.6.2.4 16 bit processors in mode 4 (Intel, multiplexed)

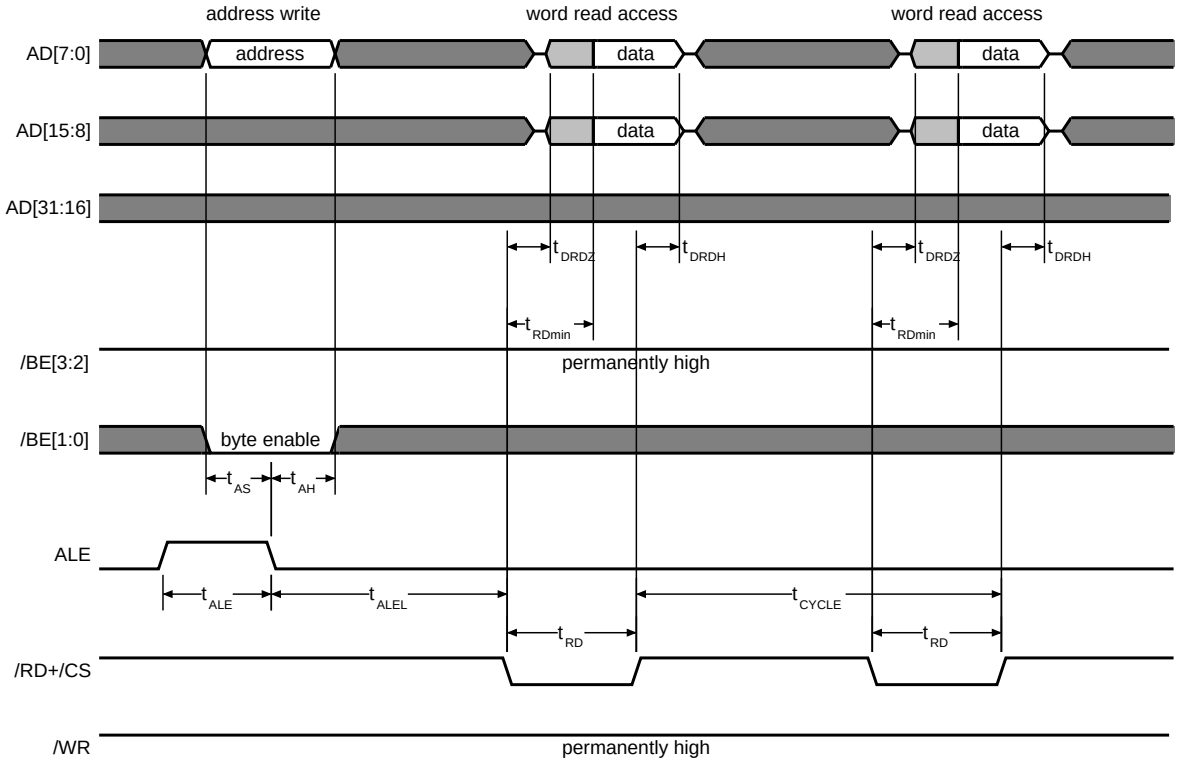


Figure 2.15: *Word read access from 16 bit processors in mode 4 (Intel, multiplexed)*

16 bit processors can either read data with byte or word access. Only 8 bits are used for address decoding. Thus the address on lines AD31 ... AD8 is ignored.

A word read is shown in Figure 2.15. FIFO and F-/Z-counter read access have 8 bit or 16 bit width alternatively. The 16 bit processor must support byte access because all other register read accesses must have a width of 8 bit.

/BE2 and /BE3 must always be '1'. /BE0 and /BE1 switch the bus AD15 .. AD0 during the data phase from tristate into data driven state (see Table 2.21 on page 84).

In mode 4 (Intel, multiplexed) the states

$$\text{/BE} = \text{'0'} \quad \text{and} \quad (\text{/RD} + \text{/CS}) = \text{'0'} \quad \text{and} \quad \text{/WR} = \text{'1'}$$

must be fulfilled to drive data out. The data bus is stable after t_{RDmin} and returns into tristate after t_{DRDH}.

Address and /BE require a setup time t_{AS} which starts with the ⅂ of ALE. The hold time of these lines is t_{AH}. If two consecutive read accesses are on the same address, multiple register address write is not required.

An 8 bit read access of a low byte is performed in the same way as it is done with 8 bit processors. Thus see Figure 2.13 for the timing specification. 8 bit read access of a high byte requires AD0 = '1' because the address is not decoded from /BE. Nevertheless, /BE[1:0] must be '01' to control lines AD15 .. AD0 during the data phase.

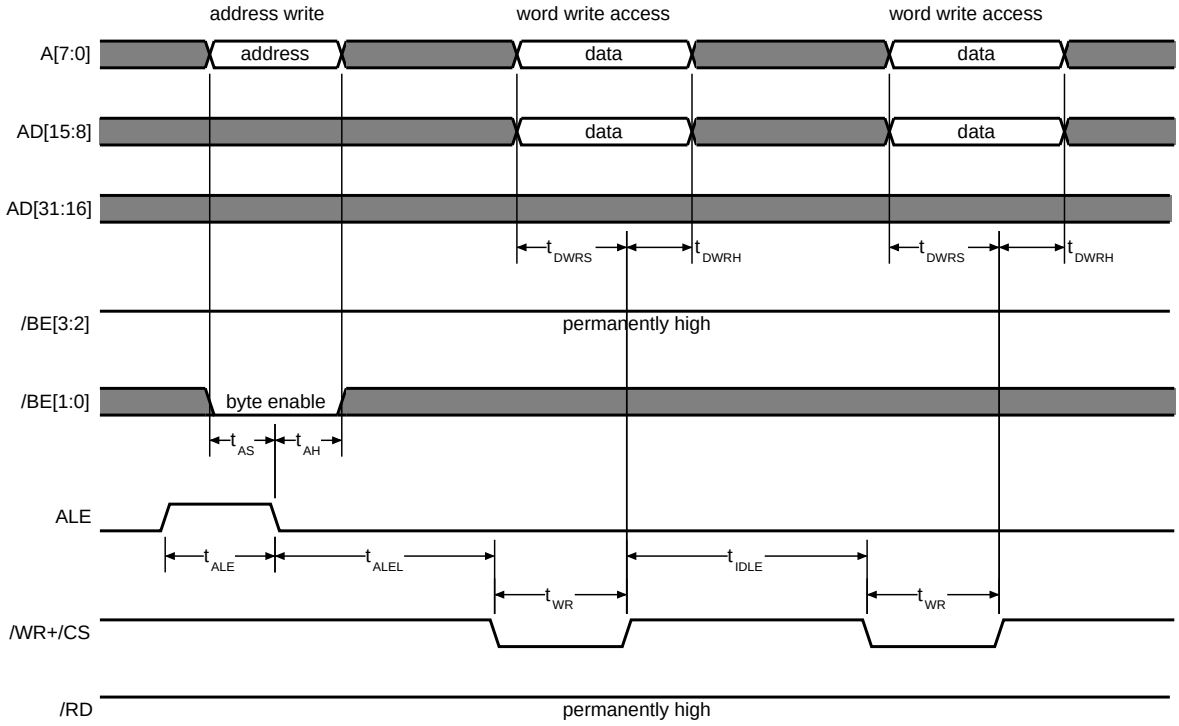


Figure 2.16: *Word write access from 16 bit processors in mode 4 (Intel, multiplexed)*

16 bit processors can either write data with byte or word access. Only 8 bits are used for address decoding. Thus the address on lines AD31 ... AD8 is ignored.

A word write is shown in Figure 2.16. FIFO write access have 8 bit or 16 bit width alternatively. The 16 bit processor must support byte access because all other register write accesses must have a width of 8 bit.

/BE2 and /BE3 must always be '1'. /BE0 and /BE1 control the low byte and high byte of the bus AD15 ..AD0 during the data phase (see Table 2.21 on page 84).

Data is written with ⌋ of /WR + /CS in mode 4 (Intel, multiplexed). HFC-E1 requires a data setup time t_{DWRS} and a data hold time t_{DWRH}.

Address and /BE require a setup time t_{AS} which starts with the ⌊ of ALE. The hold time of these lines is t_{AH}. If two consecutive write accesses are on the same address, multiple register address write is not required.

An 8 bit write access (low byte) is performed in the same way as it is done with 8 bit processors. Thus see Figure 2.14 for the timing specification. 8 bit write access of a high byte requires AD0 = '1' because the address is not decoded from /BE. Nevertheless, /BE[1:0] must be '01' to control the AD15 ..AD0 lines during the data phase.

2.6.2.5 32 bit processors in mode 4 (Intel, multiplexed)

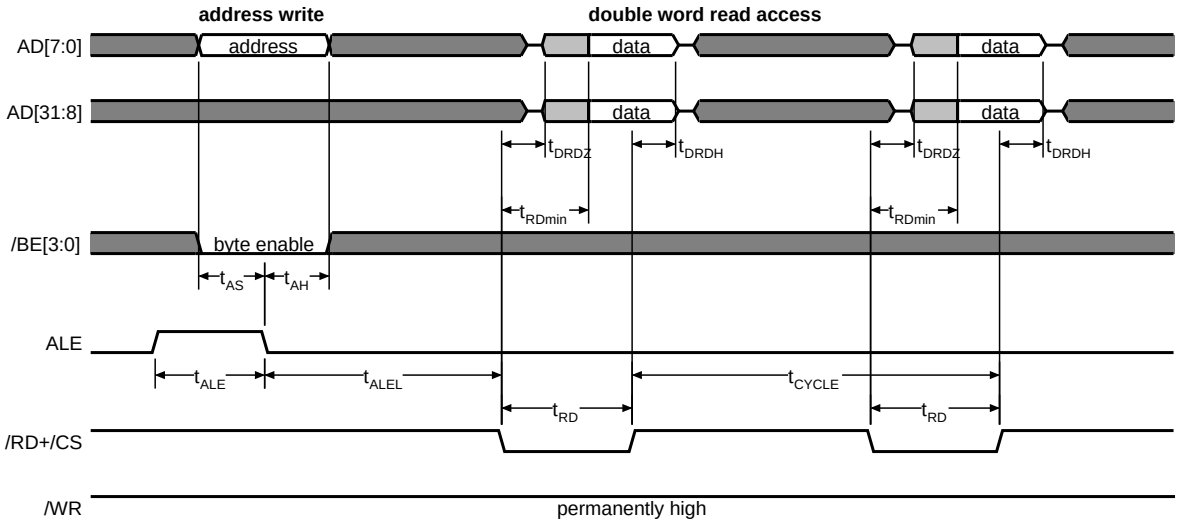


Figure 2.17: *Double word read access from 32 bit processors in mode 4 (Intel, multiplexed)*

32 bit processors can either read data with byte, word or double word access. Only 8 bits are used for address decoding. Thus the address on lines AD31 ... AD8 is ignored.

A double word read is shown in Figure 2.17. FIFO and *Z*-counter read access have 8 bit, 16 bit or 32 bit width alternatively, *F*-counter read access have 8 bit or 16 bit width alternatively. The 32 bit processor must support byte access because all other register read accesses must have a width of 8 bit.

Table 2.21: *Data access width in mode 4*

AD[1]	AD[0]	/BE3	/BE2	/BE1	/BE0	Data access
'X'	'X'	'1'	'1'	'1'	'1'	no access
'0'	'0'	'1'	'1'	'1'	'0'	byte access on AD[7:0]
'0'	'1'	'1'	'1'	'0'	'1'	byte access on AD[15:8]
'1'	'0'	'1'	'0'	'1'	'1'	byte access on AD[23:16]
'1'	'1'	'0'	'1'	'1'	'1'	byte access on AD[31:24]
'0'	'0'	'1'	'1'	'0'	'0'	word access on AD[15:0]
'1'	'0'	'0'	'0'	'1'	'1'	word access on AD[31:16]
'0'	'0'	'0'	'0'	'0'	'0'	double word access

/BE3 ../BE0 switch the bus lines AD31 ..AD0 from tristate into data driven state during read access (see Table 2.21).

In mode 4 (Intel, multiplexed) the states

$$/\mathrm{BE} = \text{'0'} \quad \text{and} \quad (/\mathrm{RD} + /\mathrm{CS}) = \text{'0'} \quad \text{and} \quad /\mathrm{WR} = \text{'1'}$$

must be fulfilled to drive data out. The data bus is stable after t_{RDmin} and returns into tristate after t_{DRDH}.

Address and /BE require a setup time t_{AS} which starts with the ⌝ of ALE. The hold time of these lines is t_{AH}. If two consecutive read accesses are on the same address, multiple register address write is not required.

An 8 bit read access (low byte) is performed in the same way as it is done with 8 bit processors. Thus see Figure 2.13 for the timing specification. Accordingly a 16 bit read access (low word) is performed in the same way as it is done with 16 bit processors. This is shown in Figure 2.15. The requirements of other byte accesses and the high word access is specified in Table 2.21.

Table 2.22: *Symbols of read accesses in Figures 2.13, 2.15 and 2.17*

Symbol	min / ns	max / ns	Characteristic
t_{ALE}	10		Address latch time
t_{ALEL}	0		ALE ⌝ to /RD+/CS ⌝
t_{AS}	10		Address and /BE valid to ALE ⌝ setup time
t_{AH}	10		Address and /BE hold time after ALE ⌝
t_{DRDZ}	2		/RD+/CS ⌝ to data buffer turn on time
t_{DRDH}	2	15	/RD+/CS ⌟ to data buffer turn off time
t_{RD}	20		Read time:
	20		AD[7] = '0' (address range 0 ... 0x7F: normal register access)
	20		AD[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$5 \cdot t_{CLKI}$		AD[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access, FIFO interrupt registers) *
t_{CYCLE}			Cycle time between two consecutive /RD+/CS ⌟
	$1.5 \cdot t_{CLKI}$		AD[7] = '0' (address range 0 ... 0x7F: normal register access)
			AD[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$5.5 \cdot t_{CLKI}$		– after byte access
	$6.5 \cdot t_{CLKI}$		– after word access
	$8.5 \cdot t_{CLKI}$		– after double word access
	$5.5 \cdot t_{CLKI}$		AD[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access, FIFO interrupt registers)

*: See 'Short read method' on page 74.

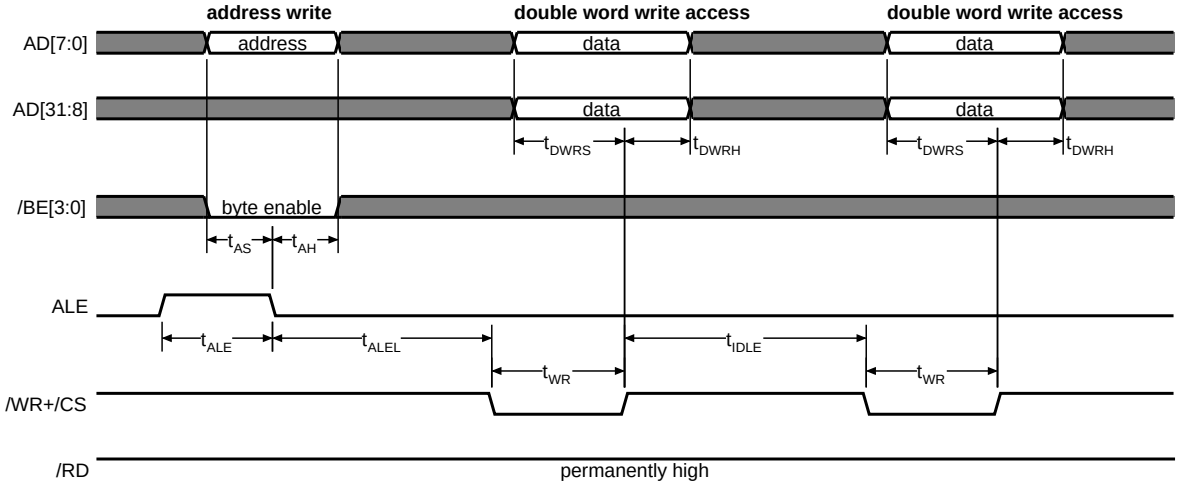


Figure 2.18: *Double word write access from 32 bit processors in mode 4 (Intel, multiplexed)*

32 bit processors can either write data with byte, word or double word access. Only 8 bits are used for address decoding. Thus the address on lines AD31 ... AD8 is ignored.

A double word write is shown in Figure 2.18. FIFO write access have 8 bit, 16 bit or 32 bit width alternatively. The 32 bit processor must support byte access because all other register write accesses must have a width of 8 bit.

/BE3 ../BE0 control the bus lines AD31 ..AD0 during data phase (see Table 2.21).

Data is written with ⌟ of /WR + /CS in mode 4 (Intel, multiplexed). HFC-E1 requires a data setup time t_{DWRS} and a data hold time t_{DWRH}.

Address and /BE require a setup time t_{AS} which starts with the ⌞ of ALE. The hold time of these lines is t_{AH}. If two consecutive write accesses are on the same address, multiple register address write is not required.

An 8 bit write access (low byte) is performed in the same way as it is done with 8 bit processors. Thus see Figure 2.14 for the timing specification. Accordingly a 16 bit write access (low word) is performed in the same way as it is done with 16 bit processors. This is shown in Figure 2.16. The requirements of other byte accesses and the high word access is specified in Table 2.21.

Table 2.23: *Symbols of write accesses in Figures 2.14, 2.16 and 2.18*

Symbol	min / ns	max / ns	Characteristic
t_{ALE}	10		Address latch time
t_{ALEL}	0		ALE ⌝ to /WR+/CS ⌝
t_{AS}	10		Address and /BE valid to ALE ⌝ setup time
t_{AH}	10		Address and /BE hold time after ALE ⌝
t_{DWRS}	20		Write data setup time to /WR+/CS ⌟
t_{DWRH}	10		Write data hold time from /WR+/CS ⌟
t_{WR}	20		Write time
t_{IDLE}			/WR+/CS high time between two consecutive data write accesses
	$1.5 \cdot t_{CLKI}$		AD[7] = '0' (address range 0 ... 0x7F: normal register access)
			AD[7,6] = '10' (address range 0x80 ... 0xBF: FIFO data access)
	$3.5 \cdot t_{CLKI}$		– after byte access
	$4.5 \cdot t_{CLKI}$		– after word access
	$6.5 \cdot t_{CLKI}$		– after double word access
	$3.5 \cdot t_{CLKI}$		AD[7,6] = '11' (address range 0xC0 ... 0xFF: direct RAM access)

2.6.3 Examples of parallel processor connection circuitries

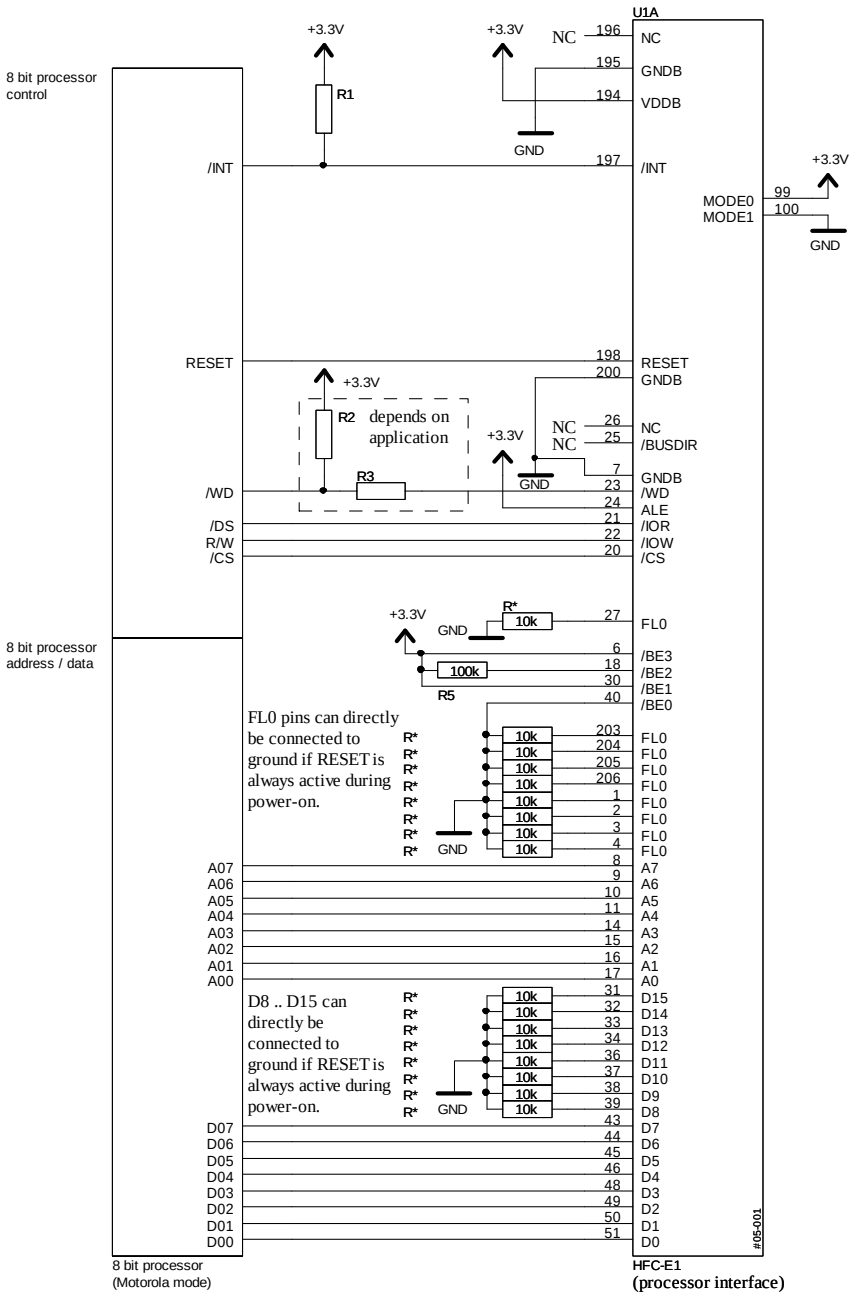


Figure 2.19: *8 bit Intel / Motorola processor circuitry example, mode 2*

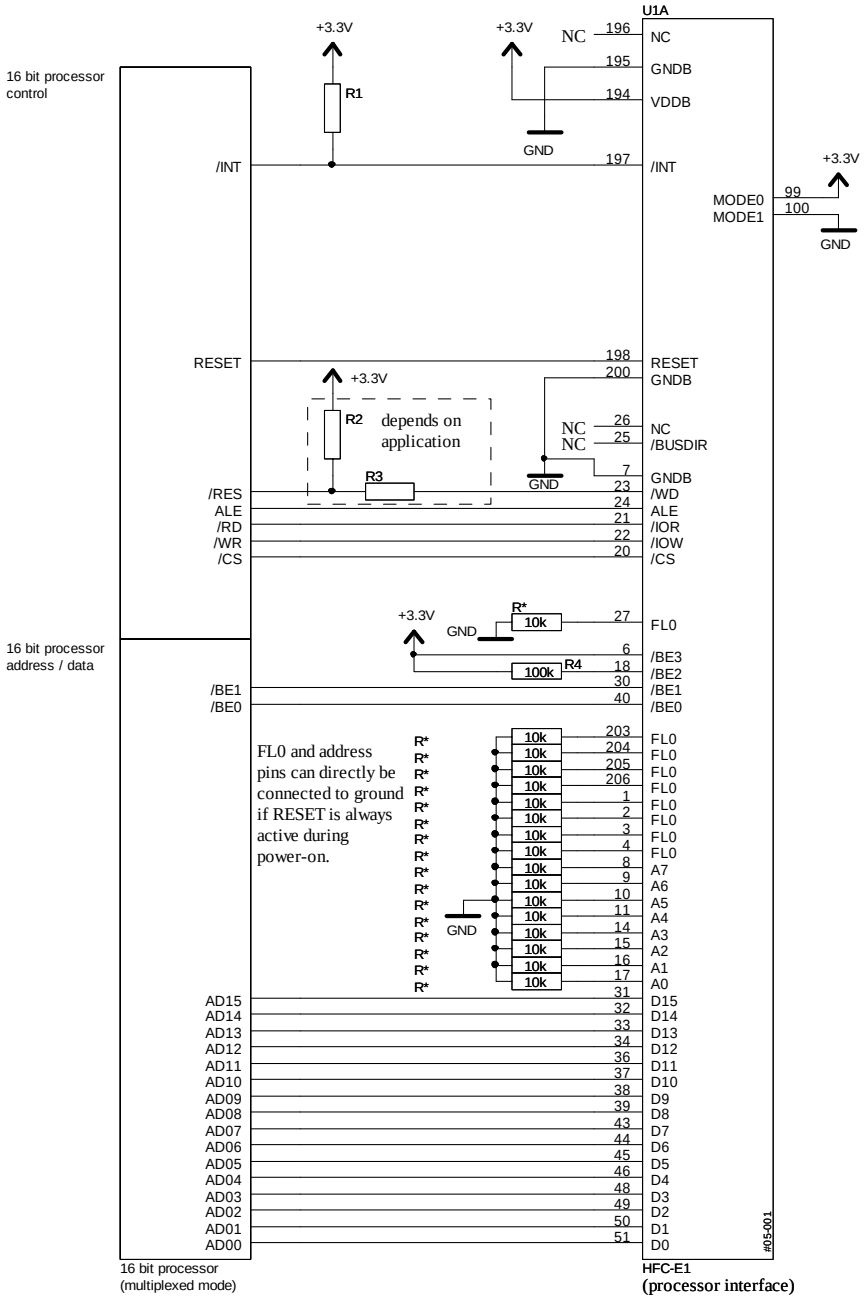


Figure 2.20: *16 bit Intel processor circuitry example, mode 4, multiplexed*

2.7 Serial processor interface (SPI)

Table 2.24: *Overview of the SPI interface pins*

Number	Name	Description
194	/SPISEL	SPI device select low active
195	SPI_RX	SPI receive data input
196	SPI_TX	SPI transmit data output
200	SPICLK	SPI clock input

HFC-E1 has a serial processor interface (SPI) which is compatible with Motorola's SPI. CPUs and MCUs with SPI interface are also available from a variety of semiconductor vendors. The SPI interface has 4 pins as shown in Table 2.24. HFC-E1 supports only SPI slave mode.

The SPI interface mode is selected by MODE0 = '1' and MODE1 = '0' (pins 99 and 100) and connecting pin 200 to SPI clock. /SPISEL must be high during reset. The first positive edge on SPICLK switches the interface from parallel processor interface mode into SPI mode. This may be the first positive clock at the start of an SPI access.

2.7.1 SPI transactions

An SPI transaction of HFC-E1 has always a length of 16 bits. The first byte is a control byte, whereas the second byte contains data with MSB first. Only the first two bits of the control byte are used by the HFC-E1 SPI interface. The other six bits must be zero.

Four different transactions are defined by the two control bits. These are shown in Table 2.25. Depending on the $R/\overline{W}$ bit the data byte is read from HFC-E1 or written into HFC-E1.

Table 2.25: *SPI transaction matrix*

	$\mathbf{R}/\overline{\mathbf{W}}$ = '0'	$\mathbf{R}/\overline{\mathbf{W}}$ = '1'
$\mathbf{A}/\overline{\mathbf{D}}$ = '0'	write data	read data
$\mathbf{A}/\overline{\mathbf{D}}$ = '1'	write address	read address

The waveforms of an address or data write transaction are shown in Figure 2.21. HFC-E1 receives the control byte and the data byte with $R/\overline{W}$ = '0' on line SPI_RX . Pin SPI_TX is not used for write transactions and has tri-state level all the time.

A read transaction is shown in Figure 2.22. HFC-E1 receives the control byte on line SPI_RX and transmits the requested data byte on line SPI_TX afterwards.

Any SPI transactions can be split by the SPI master into the control byte and the data byte with /SPISEL = '1'. In this case the transmission pauses and will be continued after /SPISEL returns to low level. An example for a split read transaction is shown in Figure 2.23. Write transactions can be split in the same way.

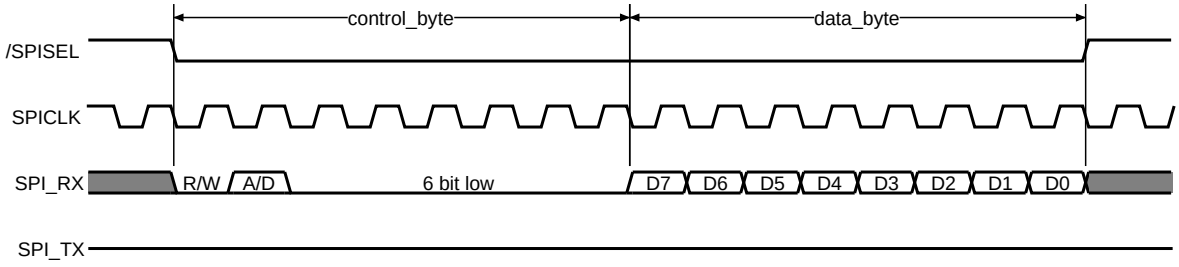


Figure 2.21: *SPI write transaction ($R/\overline{W}$ = '0')*

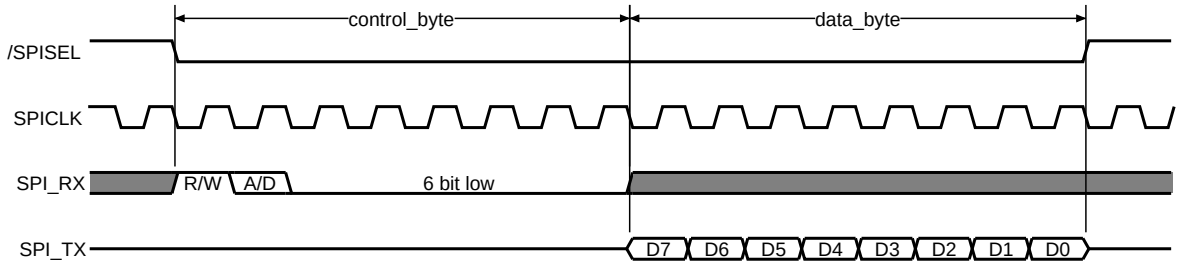


Figure 2.22: *SPI read transaction ($R/\overline{W}$ = '1')*

The SPI host is not allowed to break an SPI transaction by an interrupt service routine. Otherwise master and slave could have different views wether a specific byte is a control or a data byte. If the SPI master cannot ensure this characteristic, interrupts must be disabled between control byte and data byte.

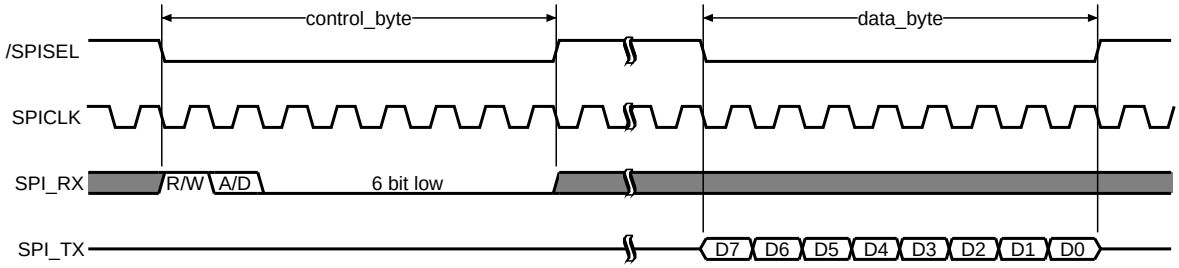


Figure 2.23: *Split SPI read transaction*

Every SPI transaction has a minimal length of 16 clock cycles. With $f_{\mathrm{SPICLK}} = 4\,\mathrm{MHz}$, e.g., a transaction takes 4 µs. HFC-E1 can operate with a maximum SPI clock of 25 MHz which leads to a minimal transaction time of

$$T_{trans,min} = \frac{16}{25\,\mathrm{MHz}} = 640\,\mathrm{ns} \quad .$$

The HFC-E1 SPI protocol defines the following rules for transactions:

1. The SPI master is allowed to stop the SPI clock at any time. When the SPI clock is restarted afterwards, the transaction is continued as if it has not been stopped.
2. When the control byte is split (/SPISEL = '1') during a write or read transaction, it is ignored and the next received byte is expected to be a control byte.

3. When the data byte of a write transaction is split, it is ignored and the next received byte is expected to be the data byte again.
4. When the data byte of a read transaction is split, the transaction quits immediately (SPI_TX is always tri-state when /SPISEL = '1'). The next received byte is expected to be the data byte again.
5. When an invalid control byte is received, it is ignored and the next received byte is expected to be a control byte.

2.7.2 Register write access

A register write access is a sequence of two SPI write transactions as shown in Figure 2.24. With the first transaction the SPI master specifies the register address (control byte is '0100 0000'). Afterwards, the new register value is transferred from the SPI master to HFC-E1 (control byte is '0000 0000').

It is allowed to execute multiple data write transactions to the same register address without address write transactions in between. This is shown in Figure 2.25 and is typically used for transmit FIFO data.

Important !

As SPI accesses are always byte oriented, the following registers cannot be used when HFC-E1 uses the SPI bus interface:

A_Z1, A_Z2, A_Z12,
A_F12,
A_FIFO_DATA1, A_FIFO_DATA2,
A_FIFO_DATA1_NOINC, A_FIFO_DATA2_NOINC.

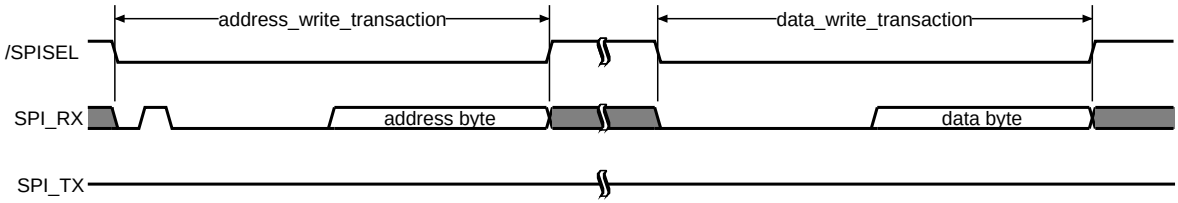


Figure 2.24: *Register write access (transaction sequence))*

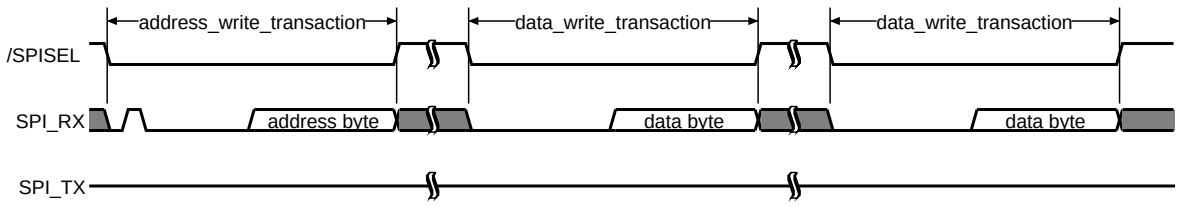


Figure 2.25: *Multiple write access to the same register*

Figures 2.24 and 2.25 show /SPISEL = '1' between the transactions. This splits the sequence into a address write transaction and one or several data write transactions. The split time has an arbitrary duration. It can also decrease to zero, which means that /SPISEL remains '0' for several transactions.

The HFC-E1 SPI protocol defines the following rules for transaction sequences:

1. Every data write transaction stores the received data byte into the register which has been selected with the last address write transaction.
2. When several consecutive address write transactions occur, the last address write transaction specifies the address for the next data access. All previous address write accesses are ignored, i.e. they have no effect to the HFC-E1 status.

2.7.3 Register read access

A register read access is a sequence of an SPI write transaction and an SPI read transaction as shown in Figure 2.26. The SPI master specifies the register address (control byte is '0100 0000') with a write transaction. Afterwards, HFC-E1 transfers the register value to the SPI master (control byte is '1000 0000') with one or several data read transactions.

It is allowed to execute multiple data read transactions to the same register address without address write transactions in between. This is shown in Figure 2.27 and is typically used for receive FIFO data.

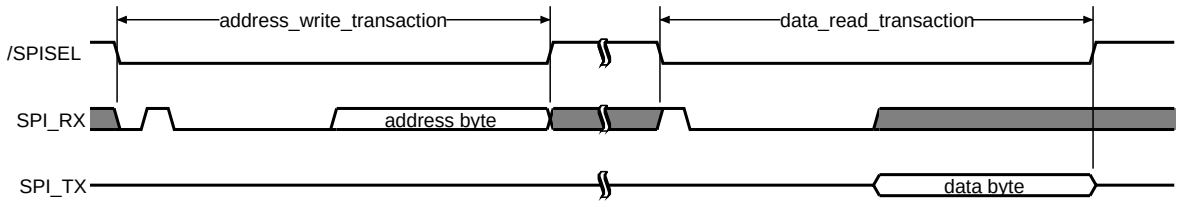


Figure 2.26: *Register read access (transaction sequence)*

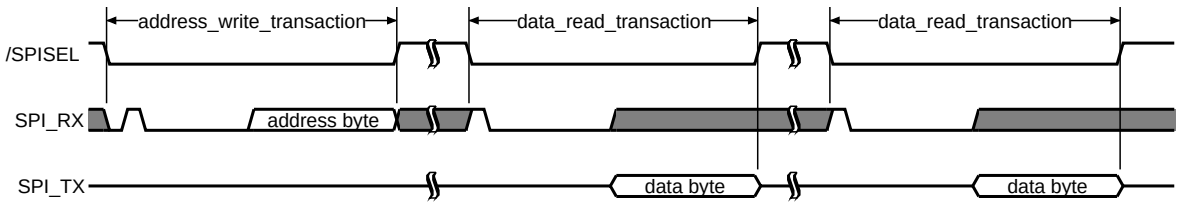


Figure 2.27: *Multiple read access to the same register*

Similar to register write accesses, the pause between transactions shown in Figures 2.26 and 2.27 have an arbitrary duration. They can also decrease to zero, which means that /SPISEL remains '0' for several transactions.

2.7.4 Register access duration

Single data sequence

A register write or read access has a length of

$$T_{reg} = 2T_{trans} = \frac{32}{f_{\mathrm{SPICLK}}}$$

if it is not split. This leads to a minimum access duration

$$T_{reg,min} = \frac{32}{25\,\mathrm{MHz}} = 1.28\,\mu\mathrm{s}$$

or a maximum data transfer rate of more than 760 KByte/s.

Multiple data sequence

Multiple accesses to the same register with a block of N data bytes have an duration of

$$T_{reg,N} = \frac{N+1}{2} T_{reg} = \frac{16(N+1)}{f_{\mathrm{SPICLK}}}$$

if there are split times neither within nor between any transactions. With the maximum SPI clock frequency $f_{\mathrm{SPICLK}} = 25\,\mathrm{MHz}$ and $N = 16$, e.g., the block transfer time is

$$T_{reg,N,min} = \frac{16 \cdot 17}{25\,\mathrm{MHz}} = 10.88\,\mu\mathrm{s} \quad (\text{for } N = 16 \text{ bytes})$$

which means that more than 1.4 MByte/s can be transmitted through the HFC-E1 SPI interface with any block size of $N \geq 16$.

2.7.5 Register address read-back capability

The address read transaction with the control byte '1100 0000' can be executed to read the address of the currently selected register.

Important !

Register address read-back cannot be used with some registers because this would change the register value. This is true for the following registers:

Address	Register name
0x2C	R_SLIP
0x26	R_RX_SL0_1
0x1A	R_BERT_ECL
0x14	R_CONF_OFLOW
0x11	R_IRQ_MISC
0xC8	R_IRQ_FIFO_BL0
..	..
0xCF	R_IRQ_FIFO_BL7

2.7.6 Problems with interrupts during transaction sequences

The address read-back capability is useful for interrupt procedures, e.g., to save and restore the previous state:

```
interrupt procedure:  - execute address read transaction and store
                        the address
                      - ... (execute the interrupt service routine)
                      - address write transaction to restore the
                        previous address
```

This procedure is important to avoid data read or write to an unexpected register address after the transaction sequence has been split between transactions by an interrupt service routine. Please note, that transactions are not allowed to be split from the SPI master (see Section 2.7.1).

Some registers have no address read-back capability (see Section 2.7.5). It is mandatory to ensure that no interrupt service routine can split an SPI access to any of these registers. As these registers are typically accessed at the beginning of interrupt service routines, this requirement should be satisfied in normal applications. Otherwise interrupts must be disabled between the address write transaction and the data read or write transaction.

2.7.7 SPI connection circuitry

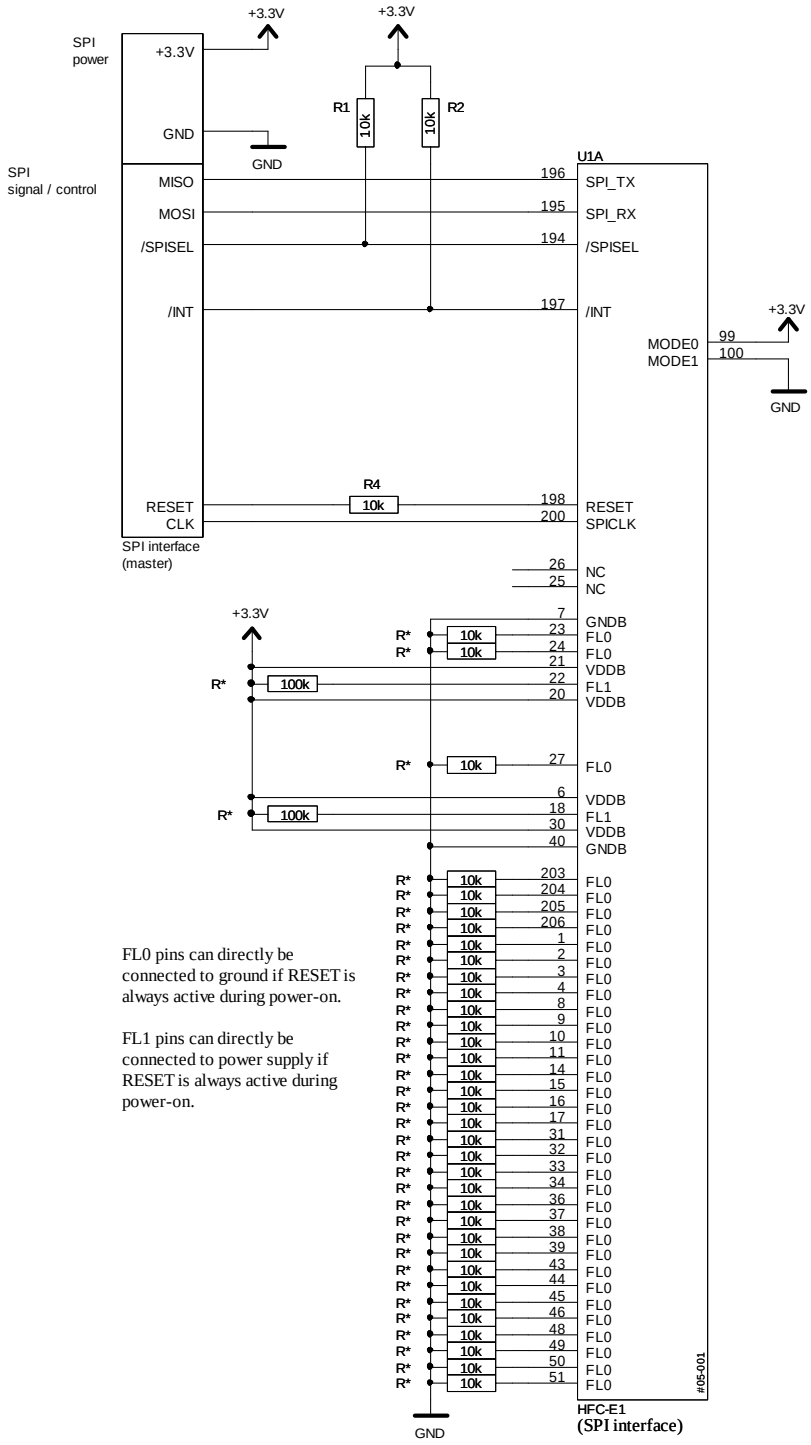


Figure 2.28: *SPI connection circuitry*

2.8 Register description

2.8.1 Write only registers

R_CIRM	(w)	0x00

Interrupt and reset register

All reset bits in this register must be cleared by software. It is not allowed to write any register while reset is pending.

Bits	Reset value	Name	Description
2..0	0	V_IRQ_SEL	**IRQ pin selection in ISA PnP mode** '000' = interrupt lines disable '001' = IRQ0 '010' = IRQ1 '011' = IRQ2 '100' = IRQ3 '101' = IRQ4 '110' = IRQ5 '111' = IRQ6
3	0	V_SRES	**Soft reset (reset group 0)** This reset is similar to the hardware reset. The selected I/O address (CIP) remains unchanged. The reset is active until the bit is cleared. '0' = deactivate reset '1' = activate reset
4	0	V_HFC_RES	**HFC-reset (reset group 1)** Sets all FIFO and HDLC registers to their initial values. The reset is active until the bit is cleared. '0' = deactivate reset '1' = activate reset
5	0	V_PCM_RES	**PCM reset (reset group 2)** Sets all PCM registers to their initial values. The reset is active until the bit is cleared. '0' = deactivate reset '1' = activate reset
6	0	V_E1_RES	**E1-reset (reset group 3)** Sets all E1 interface registers to their initial values. The reset is active until the bit is cleared. '0' = deactivate reset '1' = activate reset
7	0	V_RLD_EPR	**EEPROM reload** '0' = normal operation '1' = reload EEPROM to SRAM This bit must be cleared by software. The reload is started when the bit is cleared.

(For reset group description see Table 12.3 on page 295.)

R_CTRL	(w)	0x01

Common control register

Bits	Reset value	Name	Description
0	0	(reserved)	Must be '0'.
1	0	**V_FIFO_LPRIO**	**FIFO access priority for host accesses** '0' = normal priority '1' = low priority
2	0	**V_SLOW_RD**	**One additional wait cycle for PCI read accesses** '0' = normal operation '1' = additional wait (must be set for 66 MHz PCI operation)
3	0	**V_EXT_RAM**	**Use external RAM** The internal SRAM is switched off when external SRAM is used. '0' = internal SRAM is used in lower 32 kByte address space '1' = external SRAM is used
4	0	(reserved)	Must be '0'.
5	0	**V_CLK_OFF**	**CLK oscillator** '0' = normal operation '1' = CLK oscillator is switched off Except in PCI interface mode, this bit is reset at every write access to HFC-E1. When PCI interface is used, this bit must be reset by software.
7..6	0	(reserved)	Must be '00'.

R_RAM_ADDR0	(w)	0x08

Address pointer, register 0

1st address byte for internal / external SRAM access.

Bits	Reset value	Name	Description
7..0	0x00	**V_RAM_ADDR0**	**Address bits 7..0**

R_RAM_ADDR1	(w)	0x09

Address pointer, register 1

2nd address byte for internal / external SRAM access.

Bits	Reset value	Name	Description
7 . . 0	0x00	**V_RAM_ADDR1**	**Address bits 15 . . 8**

R_RAM_ADDR2	(w)	0x0A

Address pointer, register 2

High address bits for internal / external SRAM access and access configuration.

Bits	Reset value	Name	Description
3 . . 0	0	**V_RAM_ADDR2**	**Address bits 19 . . 16**
5 . . 4		(reserved)	Must be '00'.
6	0	**V_ADDR_RES**	**Address reset** '0' = normal operation '1' = address bits 0 . . 19 are set to zero This bit is automatically cleared.
7	0	**V_ADDR_INC**	**Address increment** '0' = no address increment '1' = automatically increment of the address after every write or read on register R_RAM_DATA **Note:** This feature can only be used with the internal SRAM. When external SRAM is used, this bit must be '0'.

R_RAM_MISC	(w)	0x0C

RAM size setup and miscellaneous functions register

Bits	Reset value	Name	Description
1..0	0	**V_RAM_SZ**	**RAM size** '00' = 32k x 8 '01' = 128k x 8 '10' = 512k x 8 '11' = reserved After setting V_RAM_SZ to a value different from '00' a soft reset should be initiated.
3..2		(reserved)	Must be '00'.
4	0	**V_PWM0_16KHZ**	**16 kHz signal on pin PWM0** '0' = normal PWM0 function '1' = 16 kHz output
5	0	**V_PWM1_16KHZ**	**16 kHz signal on pin PWM1** '0' = normal PWM1 function '1' = 16 kHz output
6		(reserved)	Must be '0'.
7	0	**V_FZ_MD**	**Exchange F-/Z-counter context** (for transmit FIFOs only) '0' = A_Z1L, A_Z1H = $Z1(F1)$ and A_Z2L, A_Z2H = $Z2(F1)$ (normal operation) '1' = A_Z1L, A_Z1H = $Z1(F1)$ and A_Z2L, A_Z2H = $Z2(F2)$ (exchanged operation) This bit can be used to check the actual RAM usage of transmit FIFOs.

2.8.2 Read only registers

R_RAM_USE	(r)	0x15

SRAM duty factor

Usage of SRAM access bandwidth by the internal data processor.

Bits	Reset value	Name	Description
7..0		V_SRAM_USE	**Relative duty factor** 0x00 = 0% bandwidth used 0x7C = 100% bandwidth used

R_CHIP_ID	(r)	0x16

Chip identification register

Bits	Reset value	Name	Description
3..0	0	V_PNP_IRQ	**IRQ assigned by the PnP BIOS** (only in ISA PnP mode) V_IRQ_SEL in register R_CIRM must be set to the value corresponding to the hardware connected IRQ lines.
7..4		V_CHIP_ID	**Chip identification code** '1110' means HFC-E1.

R_CHIP_RV	(r)	0x1F

HFC-E1 revision

Bits	Reset value	Name	Description
3..0	1	V_CHIP_RV	**Chip revision 1** (Engineering samples were revision 0.)
7..4	0	(reserved)	

2.8.3 Read / write register

R_RAM_DATA	(r/w)	0xC0

SRAM data access

Direct access to internal / external SRAM

Bits	Reset value	Name	Description
7..0		V_RAM_DATA	**SRAM data access** The address must be written into the registers R_RAM_ADDR0 . . R_RAM_ADDR2 in advance.

Chapter 3

HFC-E1 data flow

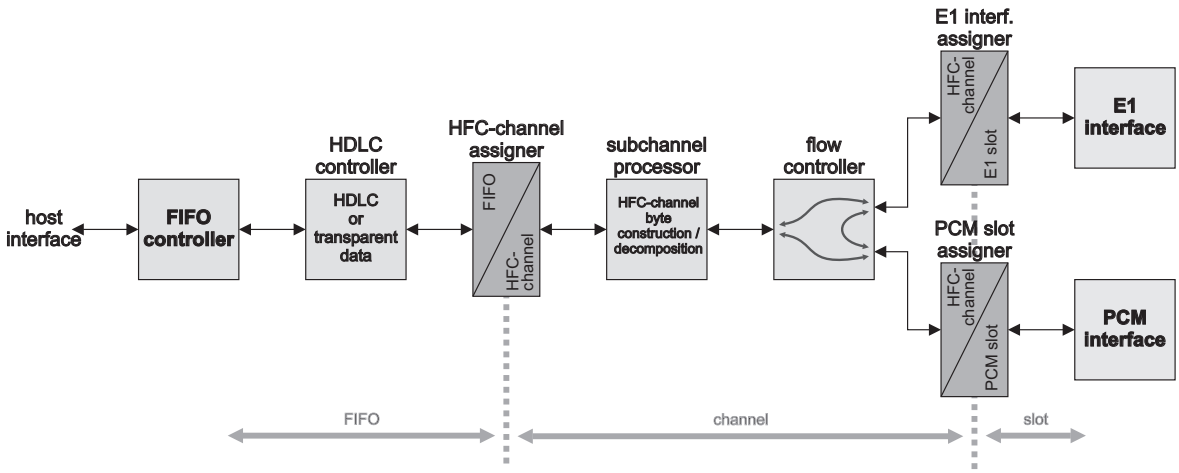


Figure 3.1: *Data flow block diagram*

3.1 Data flow concept

3.1.1 Overview

HFC-E1 has a programmable data flow unit, in which the FIFOs are connected to the PCM and the E1 interface. Moreover the data flow unit can directly connect PCM and E1 interface or two PCM time slots [1].

The fundamental features of the HFC-E1 data flow are as follows:

- programmable interconnection capability between FIFOs, PCM time slots and E1 time slots
- in transmit and receive direction there are
  - up to 32 FIFOs each
  - 32, 64 or 128 PCM time slots each
  - 32 E1 time slots each
  - 32 HFC-channels each to connect the above-mentioned data interfaces
- 3 data flow modes to satisfy different application tasks
- subchannel processing for bitwise data handling

The complete HFC-E1 data flow block diagram is shown in Figure 3.1. Basically, data routing requires an allocation number at each block. So there are three areas where numbering is based on FIFOs, HFC-channels and PCM time slots.

FIFO handling and HDLC controller, PCM and E1 interface are described in Chapters 4 to 6. So this chapter deals with the data flow unit which is located between and including the HFC-channel assigner, the PCM slot assigner and the E1 slot assigner.

3.1.2 Term definitions

Figure 3.2 clarifies the relationship and the differences between the numbering of FIFOs, HFC-channels and PCM time slots. The inner circle symbolizes the HFC-channel oriented part of the data flow, while the outer circle shows the connection of three data sources and data drains respectively. The E1 interface have a fixed mapping between HFC-channels and E1 time slots so that there is no need of a separate E1 time slot numbering.

FIFO: The FIFOs are buffers between the universal bus interface and the PCM and E1 interface. The HDLC controllers are located on the non host bus side of the FIFOs. The number of FIFOs depends on the FIFO size configuration (see Section 4.3) and starts with number 0. The maximum FIFO number is 31. Furthermore data directions transmit and receive are associated with every FIFO number.

HFC-channel: HFC-channels are used to define data paths between FIFOs on the one side and PCM and E1 interface on the other side. The HFC-channels are numbered 0 . . 31. Furthermore data directions transmit and receive are associated with every HFC-channel number.

[1] In this data sheet the shorter expression “slot” instead of “time slot” is also used with the same meaning.

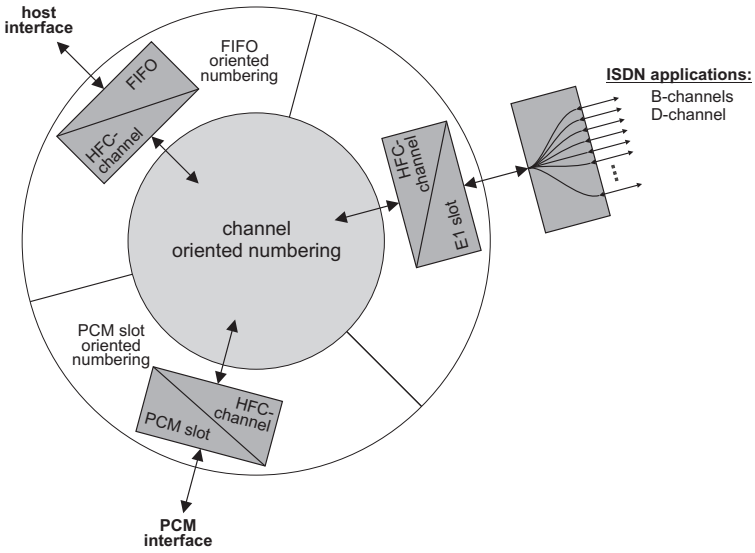


Figure 3.2: *Areas of FIFO oriented, HFC-channel oriented and PCM time slot oriented numbering*

It is important not to mix up the HFC-channels of the here discussed data flow (inner circle of Figure 3.2) with the B-channels and the D-channel of the E1 interface.

PCM time slot: The PCM data stream is organized in time slots. The number of PCM time slots depends on the data rate, i.e. there are 32 time slots with 2 MBit/s (numbered 0 . . 31), 64 time slots with 4 MBit/s (numbered 0 . . 63) or 128 time slots with 8 MBit/s (numbered 0 . . 127). Every PCM time slot exists both in transmit and receive data directions.

E1 time slot: The E1 data stream is organized in time slots. E1 time slots have always the same number and data direction as the associated HFC-channel.

Each FIFO, HFC-channel and PCM time slot number exist for transmit and receive direction. The data rate is always 8 kByte/s for every E1 time slot and every PCM time slot. FIFOs, HFC-channels, E1 time slots and PCM time slots have always a width of 8 bit.

3.2 Flow controller

3.2.1 Overview

The various connections between FIFOs, E1 time slots and PCM time slots are set up by programming the flow controller, the HFC-channel assigner and the PCM slot assigner.

The flow controller sets up connections between FIFOs and the E1 interface, FIFOs and the PCM interface and between the E1 interface and the PCM interface. Bitmap V_DATA_FLOW in register A_CON_HDLC (which exists for every FIFO) configures these connections. The numbering of transmit and corresponding receive FIFOs, HFC-channels and PCM time slots is independent from each other. But in practice the connection table is more clear if the same number is chosen for corresponding transmit and receive direction.

A direct connection between two PCM time slots can be set up inside the PCM slot assigner and will be described in Section 3.3.

The flow controller operates on HFC-channel data. Nevertheless it is programmed with a bitmap of a FIFO-indexed array register. With this concept it is possible to change the FIFO-to-HFC-channel assignment of a ready-configured FIFO without re-programming its parameters again.

The internal structure of the flow controller contains

- 4 switching buffers, i.e. one for the E1 and PCM interface in transmit and receive direction each and
- 3 switches to control the data paths.

3.2.2 Elastic buffers

Any data transmitted to and received from the line interface is stored temporarily in so-called elastic buffers. These are ring buffers with 4 E1 frames each. The offset between read and write access can be programmed with bitmaps V_TX_OFFS and V_RX_OFFS in registers R_TX_OFFS and R_RX_OFFS. Figure 3.3 shows two examples with an offset of two frames and one frame.

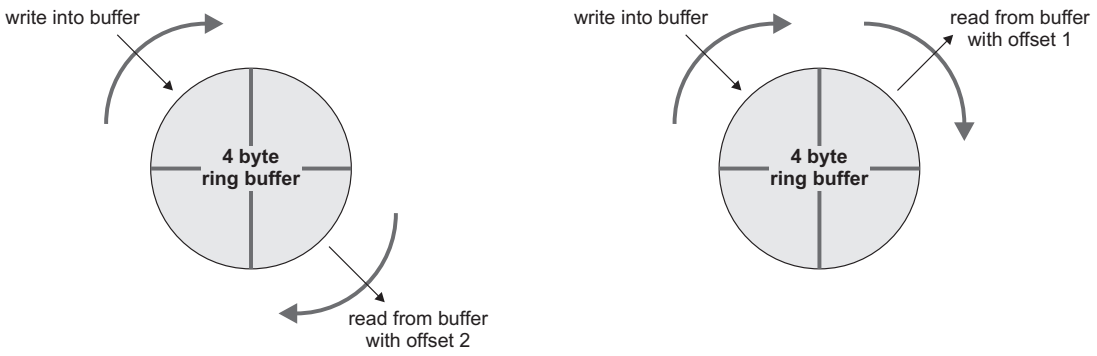


Figure 3.3: *Elastic buffer access*

The buffer size may not be changed during data operation, because any size change causes data loss. A new value of V_TX_OFFS and V_RX_OFFS must be set with corresponding V_TX_INIT and V_RX_INIT set to '1' in the same registers. These initialization bits are reset just after the new buffer size gets valid.

It is recommended to select V_TX_OFFS = V_RX_OFFS = 2 for a save operation. The delay in data transmission is $2 \cdot 125\,\mu s$ in this case.

When very accurate clock frequencies of the HFC-E1 system *and* the connected communication system is ensured, the elastic buffer size might be set to 1. This leads to a shorter delay of 125 µs but a higher risc of data loss.

The reset values '0' of R_TX_OFFS and R_RX_OFFS or a buffer size of 3 frames are not recommended for operation.

Due to different clocks of the HFC-E1 system and the connected communication system read and write access to the elastic buffer might fall into the same buffer frame. This can lead to data error and is reported in register R_SLIP. With each write access to the transmit buffer and each read access to the receive buffer, V_SLIP_RX and V_SLIP_TX are set in register R_SLIP. When one of these bits is set, the buffer must be realigned.

3.2.3 Timed sequence

The data transmission algorithm of the flow controller is FIFO-oriented and handles all FIFOs, and of course all connected HFC-channels, every 125 µs in the following sequence:

1. FIFO[0,TX]
2. FIFO[0,RX]
3. FIFO[1,TX]
4. FIFO[1,RX]

⋮

63. FIFO[n,TX]
64. FIFO[n,RX]

The number of existing FIFOs, and consequently the value of n, depends on the FIFO configuration (see Table 4.3 on page 159). In any case n cannot exceed 31. There are other configurations with $n = 15$, $n = 7$ and $n = 3$.

If a faulty configuration writes data from several sources into the same switching buffer, the last write access overwrites the previous ones. Only in this case it is necessary to know the process sequence of the flow controller.

HFC-E1 has three data flow modes. One of them (*FIFO sequence mode*) is used to configure a programmable FIFO sequence which can be used instead of the ascending FIFO numbering. This is explained in Section 3.4.

3.2.4 Transmit operation (FIFO in transmit data direction)

In transmit operation one HDLC or transparent byte is read from a FIFO and can be transmitted to the E1 and the PCM interface as shown in Figure 3.4. Furthermore, data can be transmitted from the E1 interface to the PCM interface. From the flow controller point of view, the switches select the source for outgoing data. They are controlled by bitmap V_DATA_FLOW[2..1] in register A_CON_HDLC[n,TX] where n is a FIFO number. Transmit operation is configured with V_FIFO_DIR = '0' in the multi-register R_FIFO.

- FIFO data is only transmitted to the E1 interface if V_DATA_FLOW[1] = '0'.
- The PCM interface can transmit a data byte which comes either from the FIFO or from the E1 interface. Bit V_DATA_FLOW[2] selects the source for the PCM transmit slot (see Figure 3.4). The receiving E1 time slot has always the same number as the transmitting E1 time slot .
- Bit V_DATA_FLOW[0] is ignored in transmit operation.

3.2.5 Receive operation (FIFO in receive data direction)

Figure 3.5 shows the flow controller structure in receive operation. The two switches are controlled by bitmap V_DATA_FLOW[1..0] in register A_CON_HDLC[n,RX] where n is a FIFO number. Receive operation is configured with V_FIFO_DIR = '1' in multi-register R_FIFO. FIFO data can either be

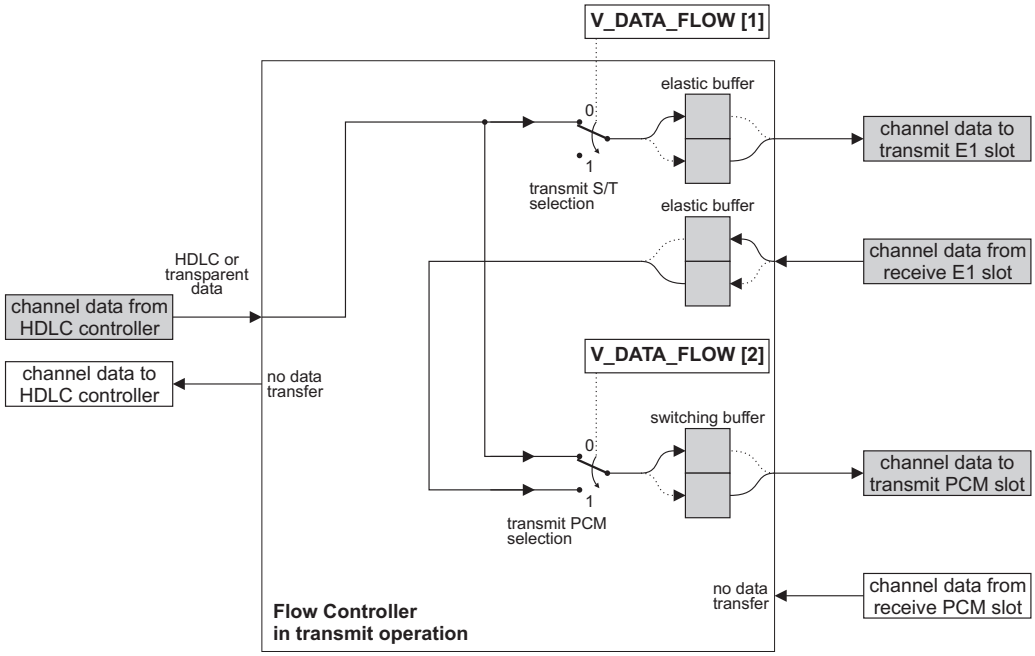


Figure 3.4: *The flow controller in transmit operation*

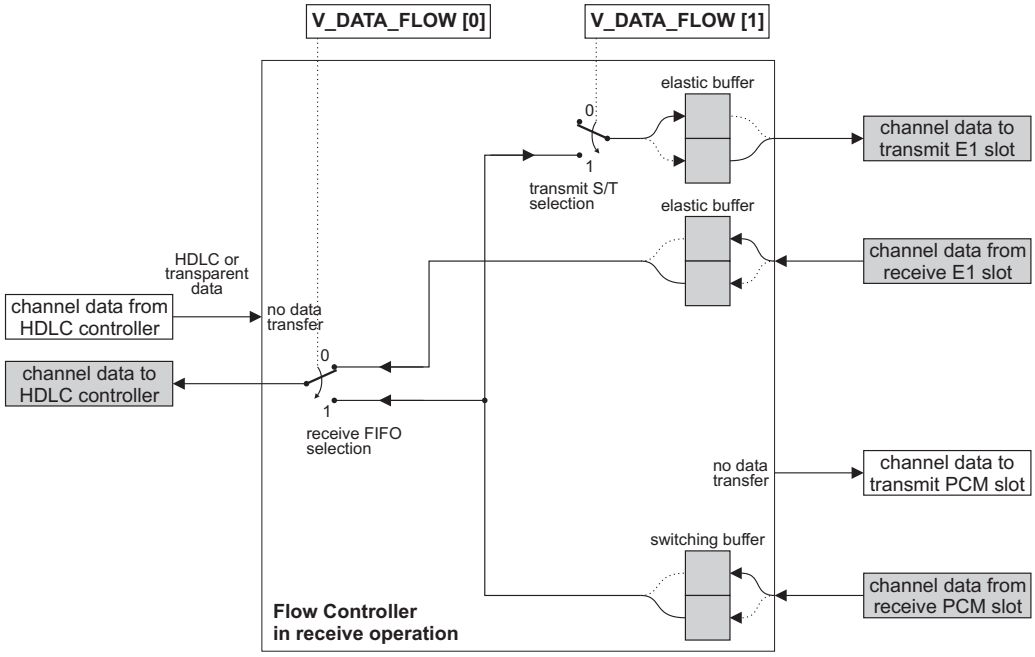


Figure 3.5: *The flow controller in receive FIFO operation*

received from the E1 or PCM interface. Furthermore, data can be transmitted from the PCM interface to the E1 interface.

- Bit V_DATA_FLOW[0] selects the source for the receive FIFO which can either be the PCM or the E1 interface.
- Furthermore, the received PCM byte can be transferred to the E1 interface. This requires bit V_DATA_FLOW[1] = '1'.
- Bit V_DATA_FLOW[2] is ignored in receive FIFO operation.

3.2.6 Connection summary

Table 3.1 shows the flow controller connections as a whole. Bidirectional connections [2] are pointed out with a gray box because they are typically used to establish the data transmissions. All rows have an additional connection to a second destination.

Table 3.1: *Flow controller connectivity*

V_DATA_FLOW	Transmit (V_FIFO_DIR = '0')		Receive (V_FIFO_DIR = '1')	
'000'	FIFO → PCM	FIFO → E1	FIFO ← E1	
'001'	FIFO → E1	FIFO → PCM	FIFO ← PCM	
'010'	FIFO → PCM		FIFO ← E1	E1 ← PCM
'011'		FIFO → PCM	FIFO ← PCM	E1 ← PCM
'100'	E1 → PCM	FIFO → E1	FIFO ← E1	
'101'	FIFO → E1	E1 → PCM	FIFO ← PCM	
'110'		E1 → PCM	E1 ← PCM	FIFO ← E1
'111'		E1 → PCM	E1 ← PCM	FIFO ← PCM

The most important connections are bidirectional data transmissions. For these connections it is possible to manage the configuration programming of V_DATA_FLOW with only three different values for transmit and receive FIFO operations. Table 3.2 shows the suitable programming values which can be used to simplify the programming algorithm.

> **Important !**
>
> There is one execption in data flow programming, when the DTMF controller is required for a FIFO → E1 connection:
>
> V_DATA_FLOW = '100' is recommended in this case.
>
> Please see Section 9.4 on page 275 for a detailed explanation.

[2] In fact, all connections are unidirectional. However, in typical applications there is always a pair of transmit and receive data channels which belong together. Instead of "transmit and corresponding receive data connection" the shorter expression "bidirectional connection" is used in this data sheet.

Table 3.2: *V_DATA_FLOW programming values for bidirectional connections*

Connection			V_FIFO_DIR	Required V_DATA_FLOW	Recommended V_DATA_FLOW
FIFO	→	E1	'0' (TX)	'x0x'	'000'
FIFO	←	E1	'1' (RX)	'xx0'	
FIFO	→	PCM	'0' (TX)	'0xx'	'001'
FIFO	←	PCM	'1' (RX)	'xx1'	
E1	→	PCM	'0' (TX)	'1xx'	'110'
E1	←	PCM	'1' (RX)	'x1x'	

3.3 Assigners

The data flow block diagram in Figure 3.1 contains three assigners. These functional blocks are used to connect FIFOs, E1 time slots and PCM time slots to the HFC-channels.

3.3.1 HFC-channel assigner

The HFC-channel assigner interconnects FIFOs and HFC-channels. Its functionality depends on the data flow mode described in Section 3.4.

3.3.2 PCM slot assigner

The PCM slot assigner can connect each PCM time slot to an arbitrary HFC-channel. Therefore, for a selected time slot [3] the connected HFC-channel number and data direction must be written into register A_SL_CFG[SLOT] as follows:

Register setup:

A_SL_CFG[SLOT] :	V_CH_SDIR	=	<HFC-channel data direction>
:	V_CH_SNUM	=	<HFC-channel number>

Typically, the data direction of a HFC-channel and its connected PCM time slot is the same.

If two PCM time slots are connected to each other, incoming data on a slot is transferred to the PCM slot assigner and stored in the PCM receive switching buffer of the connected HFC-channel. From there it is read (i.e. same HFC-channel) and transmitted to a transmit PCM time slot.

3.3.3 E1 slot assigner

The E1 interface consists of 32 time slots for transmit data and 32 time slots for receive data.

In HFC-E1 applications, these time slots are typically used for ISDN data transfer. Then time slot 0 is reserved for the synchronization process and time slot 16 is normally used to be the D-channel. All the other time slots are assigned to B-channels for the ISDN data transmission.

Between the HFC-channels [4] and the E1 time slots there is a simple assignment:

$$\text{HFC-channel}[n,\text{TX}] \leftrightarrow \text{E1 slot}[n,\text{TX}]$$
$$\text{HFC-channel}[n,\text{RX}] \leftrightarrow \text{E1 slot}[n,\text{RX}]$$

with $n = 0..31$. There is no possibility to change this allocation, so there are no registers for programming the E1 slot assigner.

[3] A time slot is specified by writing its number and data direction into register R_SLOT. Then all accesses to the slot array registers belong to this time slot. Please see Chapter 6 for details.

[4] These channels have nothing to do with the mentioned D-channel and B-channels of the E1 interface, please refer to the inner circle of Figure 3.2.

3.3.4 Assigner summary

The three different assigner types of HFC-E1 are shown in Figure 3.6. Assigner programming is always handled with array registers. This can be a FIFO array register or a PCM slot array register.

- The E1 interface assigner is not programmable. Every HFC-channel is connected to a E1 time slot with the same number and data direction.
- The PCM slot assigner is programmed by register A_SL_CFG[SLOT]. The PCM time slot must be selected before by writing the desired slot number and direction into register R_SLOT.
- The HFC-channel assigner programming depends on the data flow mode which is described in Section 3.4. This section explains in what cases the assigner is programmable and how this can be done. Figure 3.6 gives a hint, that the programming procedure is handled with the array register A_CHANNEL[FIFO]. Please see section 3.4 for details and restrictions.

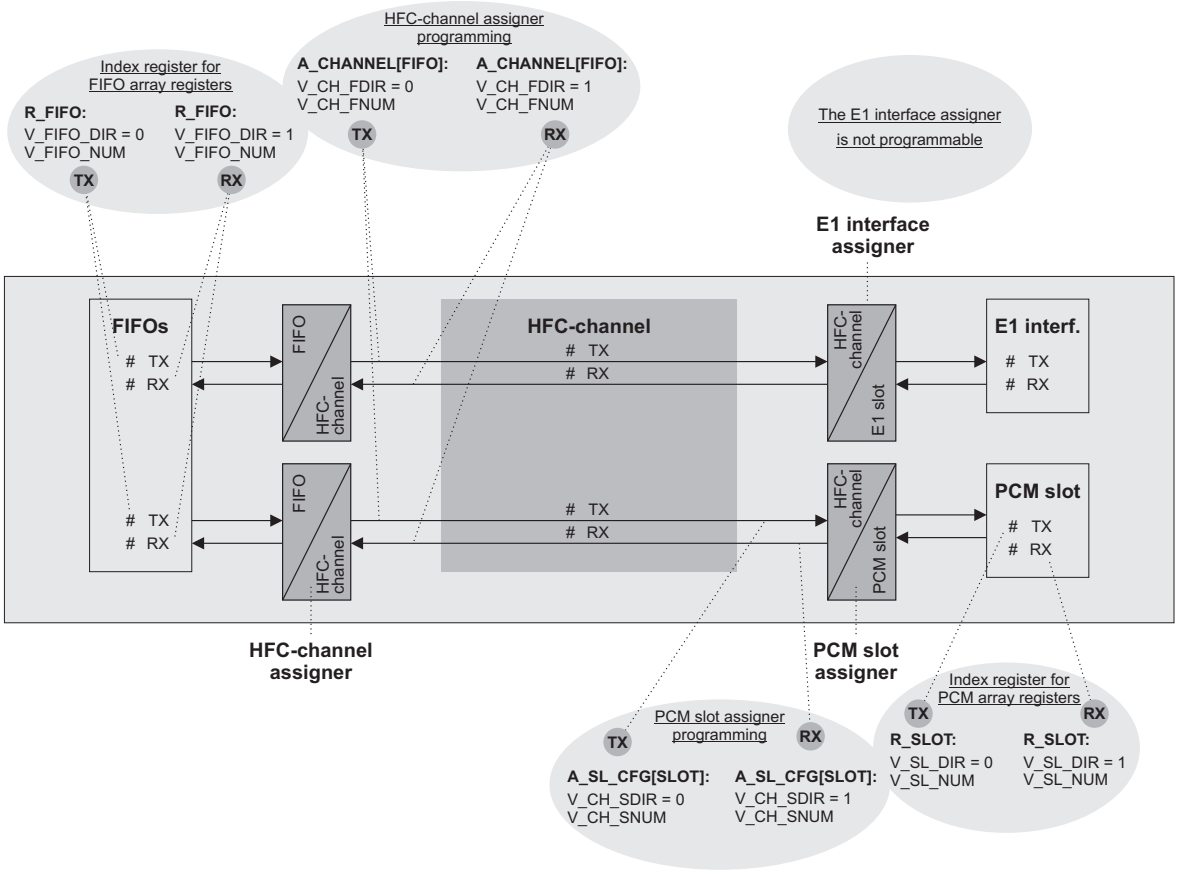


Figure 3.6: *Overview of the assigner programming*

3.4 Data flow modes

The internal operation of the HFC-channel assigner and the subchannel processor depends on the selected data flow mode. Three modes are available and will be described in this section:

- *Simple Mode* (SM),
- *Channel Select Mode* (CSM) and
- *FIFO Sequence Mode* (FSM)

Various array registers are available to configure the data flow. Unused FIFOs and PCM time slots should remain in their reset state.

FIFO array registers are indexed by R_FIFO in most cases. But there are some exceptions depending on the data flow mode and the target array register. Table 3.3 shows all FIFO array registers and their index registers at the different data flow modes.

3.4.1 Simple Mode (SM)

3.4.1.1 Mode description

In *Simple Mode* (SM) only one-to-one connections are possible. That means one FIFO, one E1 time slot or one PCM time slot can be connected to each other. The number of connections is limited by the number of FIFOs. It is possible to establish as many connections as there are FIFOs [5]. The actual number of FIFOs depends on the FIFO setup (see Section 4.3).

Simple Mode is selected with V_DF_MD = '00' in register R_FIFO_MD. All FIFO array registers are indexed by the multi-register R_FIFO (address 0x0F) in this data flow mode.

The FIFO number is always the same as the HFC-channel number. Thus, the HFC-channel assigner cannot be programmed in *Simple Mode*. In contrast to this, the PCM time slot number can be chosen independently from the HFC-channel number.

Due to the fixed correspondence between FIFO number and HFC-channel, a pair of transmit and receive FIFOs is allocated even if a bidirectional data connection between the PCM interface and the E1 interface is established without using the FIFO. Nevertheless, in this case the FIFO must be enabled to enable the data transmission.

A direct coupling of two PCM time slots uses a PCM switching buffer and no FIFO has to be enabled. This connection requires a HFC-channel number (resp. the same FIFO number). An arbitrary HFC-channel number can be chosen. If there are less than 32 transmit and receive FIFOs each, it is usefull to choose a HFC-channel number that is greater than the maximum FIFO number. This saves FIFO resources where no data is stored in a FIFO.

[5] Except PCM-to-PCM connections which do not need a FIFO resource if the involved HFC-channel number is higher than the maximum FIFO number.

Table 3.3: *Index registers of the FIFO array registers (sorted by address)*

Context	Array register name	Array register address	Index register in SM	Index register in CSM	Index register in FSM	Index meaning in SM	Index meaning in CSM	Index meaning in FSM
FIFO data counters	0x04	A_Z1L[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
	⋮							
	0x07	A_Z2H[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
FIFO frame counters	0x0C	A_F1[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
	⋮							
	0x0D	A_F2[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
FIFO configuration	0x0E	A_INC_RES_FIFO[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
FIFO data access	0x80	A_FIFO_DATA0[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
	⋮							
	0x84	A_FIFO_DATA0_NOINC[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
Subchannel processor	0xF4	A_CH_MSK[FIFO]	R_FIFO	R_FIFO	R_FIFO	HFC-channel	HFC-channel	HFC-channel
FIFO configuration	0xFA	A_CON_HDLC[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO
Subchannel Processor	0xFB	A_SUBCH_CFG[FIFO]	R_FIFO	R_FIFO	R_FSM_IDX	FIFO	FIFO	list index
FIFO configuration	0xFC	A_CHANNEL[FIFO]	R_FIFO	R_FIFO	R_FSM_IDX	FIFO	FIFO	list index
FIFO configuration	0xFD	A_FIFO_SEQ[FIFO]	R_FIFO	R_FIFO	R_FSM_IDX	FIFO	FIFO	list index
FIFO configuration	0xFF	A_IRQ_MSK[FIFO]	R_FIFO	R_FIFO	R_FIFO	FIFO	FIFO	FIFO

Please note !

The index of FIFO array registers is always denoted '[FIFO]' even if the meaning is different from this in particular cases (grey marked fields in Table 3.3).

3.4.1.2 Subchannel processing

In most applications the subchannel processor is not used in *Simple Mode*. However, if the data stream of a FIFO does not require full 8 kByte/s data rate, the subchannel processor might be used. Unused bits can be masked out and replaced by bits of an arbitrary mask byte which can be specified in A_CH_MSK.

In transparent mode only the non-masked bits of a byte are processed. Masked bits are taken from register A_CH_MSK. So the effective FIFO data rate always remains 8 kByte/s whereas the usable data rate depends on the number of non-masked bits.

In HDLC mode the data rate of the FIFO is reduced according to how many bits are not masked out.

Please see Section 3.5 on page 138 for details concerning the subchannel processor.

3.4.1.3 Example for SM

Figure 3.7 shows an example with four bidirectional connections (❶ FIFO-to-E1, ❷ FIFO-to-PCM, ❸ PCM-to-E1 and ❹ PCM-to-PCM). The FIFO box on the left side contains the number and direction information of the used FIFOs. The E1 and PCM boxes on the right side contain the E1 time slots and PCM time slot numbers and directions which are used in this example. Black lines illustrate data paths, whereas dotted lines symbolize blocked resources. These are not used for the data transmission, but they are necessary to enable the settings.

> **Please note !**
>
> All settings in Figure 3.7 are configured in bidirectional data paths due to typical applications of HFC-E1. However, transmit and receive directions are independent from each other and could occur one at a time as well.

The following settings demonstrate the required register values to establish the connections. All involved FIFOs have to be enabled with either V_HDLC_TRP = '1' (transparent mode and implicit FIFO enable) or V_TRP_IRQ $\neq 0$ (explicit FIFO enable) in register A_CON_HDLC[FIFO].

The subchannel processor and the conference unit are not used in this example. For this reason, registers A_SUBCH_CFG, A_CH_MSK and A_CONF remain in their reset state.

❶ **FIFO-to-E1**

As HFC-channel and FIFO numbers are the same in SM, a selected E1 time slot specifies the corresponding FIFO (and same in inverse, of course). There is no need of programming the HFC-channel assigner.

To set up a FIFO-to-E1 connection, the desired E1 time slot has to be chosen and the linked FIFO has to be programmed. Due to the user's requirements, V_REV can be programmed either to normal or inverted bit order of the FIFO data.

HDLC or transparent mode (V_HDLC_TRP) can freely be chosen as well. In addition to the settings shown here, a periodic interrupt (in transparent mode) or a *end of frame* interrupt (in HDLC mode) can be enabled.

If HDLC mode is chosen, the FIFO must be enabled with V_TRP_IRQ $\neq 0$.

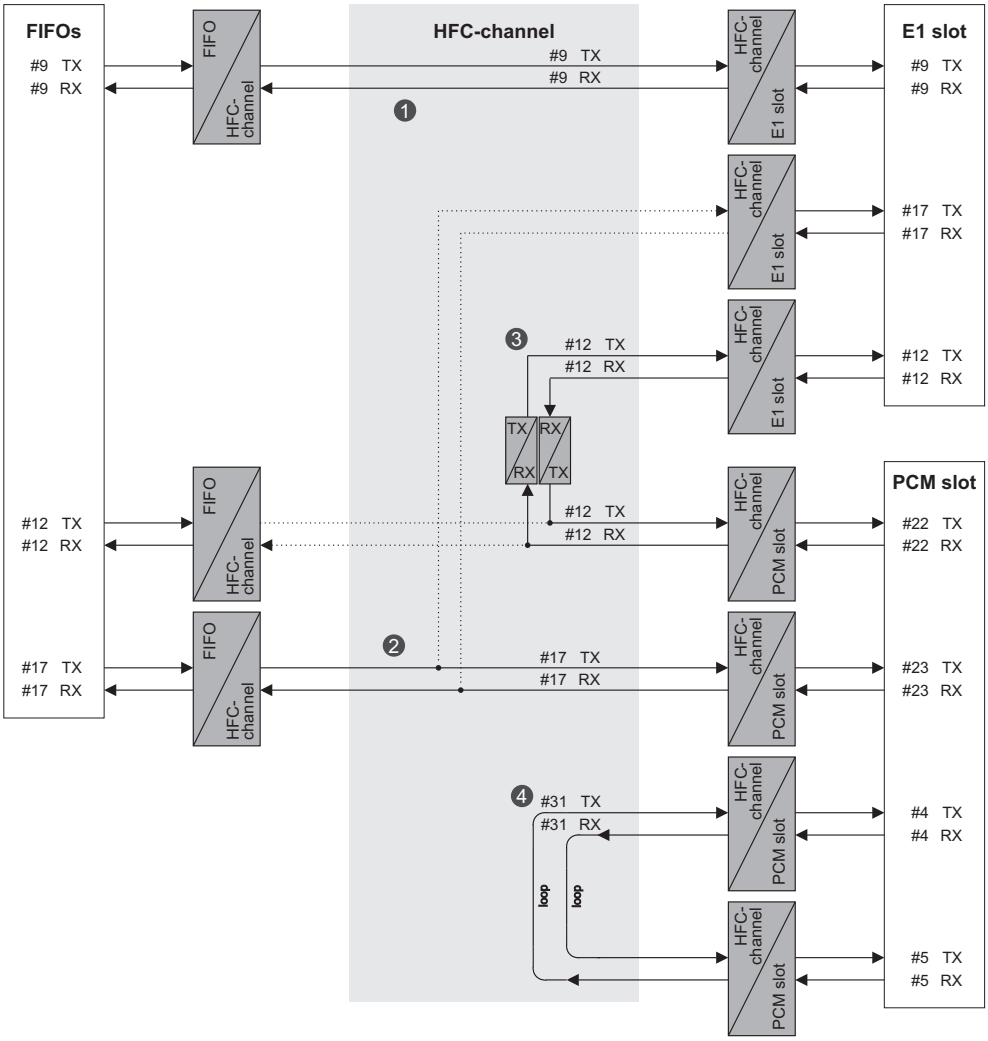


Figure 3.7: *SM example*

Register setup:				(SM ❶ TX)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	9	(FIFO #9)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[9,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO → E1, FIFO → PCM)

Register setup:				(SM ❶ RX)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	9	(FIFO #9)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[9,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO ← E1)

❷ FIFO-to-PCM

The FIFO-to-PCM connection can use different numbers for the involved HFC-channels and PCM time slots. The desired numbers are linked together in the PCM slot assigner.

As the E1 interface assigner links the HFC-channels to the E1 time slots, every used HFC-channel blocks the connected E1 time slot.

Again, V_REV and V_HDLC_TRP can freely be chosen according to the user's requirements. As in the previous setting, a periodic interrupt in transperant mode or a *end of frame* interrupt in HDLC mode can be enabled.

Register setup:				(SM ❷ TX)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	17	(FIFO #17)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[17,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'001'	(FIFO → E1, FIFO → PCM)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	23	(slot #23)
A_SL_CFG[23,TX]	: V_CH_SDIR	=	0	(transmit HFC-channel)
	: V_CH_SNUM	=	17	(HFC-channel #17)
	: V_ROUT	=	'10'	(data to pin STIO1)

Register setup:				(SM ❷ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 17		(FIFO #17)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[17,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO ← PCM)
R_SLOT	: V_SL_DIR	= 1		(receive slot)
	: V_SL_NUM	= 23		(slot #23)
A_SL_CFG[23,RX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 17		(HFC-channel #17)
	: V_ROUT	= '10'		(data from pin STIO2)

❸ PCM-to-E1

A direct PCM-to-E1 coupling is shown in the third connection set. The array registers of FIFO[12,TX] and FIFO[12,RX] contain the data flow settings, so they must be configured and the FIFOs must be enabled to switch on the data transmission. This is done with either V_HDLC_TRP = '1' (transparent mode and implicit FIFO enable) or V_TRP_IRQ $\neq 0$ (explicit FIFO enable) in register A_CON_HDLC[FIFO].

In receive direction, data is stored in the connected FIFO. But it is not used and needs not to be read. A FIFO overflow has no effect and can be ignored. Consequently, the V_HDLC_TRP setting has no effect to the transferred data between the PCM and the E1 interface neither in receive nor in transmit direction. A PCM-to-E1 connection operates always in transparent mode.

For a PCM-to-E1 connection, the data direction changes between the two interfaces. In detail, data is received on a RX line and then transmitted on a TX line to the other interface. Therefore, a TX-RX-exchanger is inserted for this connection. The blocked FIFOs are on the PCM side of the TX-RX-exchanger. Like shown in the register setting below, data direction of FIFO, E1 and PCM lines are never mixed up when programming the assigners in *Simple Mode*.

Register setup:				(SM ❸ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 12		(FIFO #12)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[12,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 0		(interrupt disabled)
	: V_DATA_FLOW	= '110'		(E1 → PCM)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 22		(slot #22)
A_SL_CFG[22,TX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 12		(HFC-channel #12)
	: V_ROUT	= '10'		(data to pin STIO1)

Register setup:				(SM ❸ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 12		(FIFO #12)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[12,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 0		(interrupt disabled)
	: V_DATA_FLOW	= '110'		(FIFO ← E1, E1 ← PCM)
R_SLOT	: V_SL_DIR	= 1		(receive slot)
	: V_SL_NUM	= 22		(slot #22)
A_SL_CFG[22,RX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 12		(HFC-channel #12)
	: V_ROUT	= '10'		(data from pin STIO2)

❹ PCM-to-PCM

A PCM-to-PCM configuration does not occupy any FIFO resources. An example is shown in the last connection set. HFC-channel[31,RX] is used to connect PCM slot[4,RX] to PCM slot[5,TX]. Data from PCM slot[5,RX] to PCM slot[4,TX] is transferred through HFC-channel[31,TX].

A HFC-channel loop is easily established by linking two PCM time slots to the same HFC-channel. Every used HFC-channel blocks the assigned E1 channel. For this reason the shown PCM-to-PCM configuration blocks the E1 channels [31,TX] and [31,RX]. They cannot be used for data transmission in this example.

Register setup:				(SM ❹ no. 1)
R_SLOT	: V_SL_DIR	= 1		(receice slot)
	: V_SL_NUM	= 4		(slot #4)
A_SL_CFG[4,RX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '11'		(data from pin STIO1)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 5		(slot #5)
A_SL_CFG[5,TX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '11'		(data to pin STIO2)

Register setup:				(SM ❹ no. 2)
R_SLOT	: V_SL_DIR	= 1		(receice slot)
	: V_SL_NUM	= 5		(slot #5)
A_SL_CFG[5,RX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '10'		(data from pin STIO2)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 4		(slot #4)
A_SL_CFG[4,TX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '10'		(data to pin STIO1)

Rule

In *Simple Mode* for every used FIFO[*n*] the HFC-channel[*n*] is also used. This is valid in reverse case, too.

3.4.2 Channel Select Mode (CSM)

3.4.2.1 Mode description

The *Channel Select Mode* (CSM) allows an arbitrary assignment between a FIFO and the connected HFC-channel as shown in Figure 3.8 (left side). Beyond this, it is possible to connect several FIFOs to one HFC-channel (Fig. 3.8, right side). This works in transmit and receive direction and can be used to connect one 8 kByte/s E1 time slot or PCM time slot to multiple FIFO data streams, with lower data rate each. In this case the subchannel processor must be used.

Figure 3.8: *HFC-channel assigner in CSM*

Channel Select Mode is selected with V_DF_MD = '01' in register R_FIFO_MD. All FIFO array registers are indexed by the multi-register R_FIFO (address 0x0F) in this data flow mode.

3.4.2.2 HFC-channel assigner

The connection between a FIFO and a HFC-channel can be established by register A_CHANNEL which exists for every FIFO. For a selected FIFO, the HFC-channel to be connected must be written into V_CH_FNUM of register A_CHANNEL. Typically, the data direction in V_CH_FDIR is the same as the FIFO data direction V_FIFO_DIR in the multi-register R_FIFO. With the following register settings the HFC-channel assigner connects the selected FIFO to HFC-channel *n*.

Register setup:

A_CHANNEL[FIFO] : V_CH_FDIR = V_FIFO_DIR
: V_CH_FNUM = *n*

A direct connection between a PCM time slot and an E1 time slot allocates one FIFO although this FIFO does not store any data. In *Channel Select Mode* – in contrast to *Simple Mode* – an arbitrary FIFO can be chosen. This FIFO must be enabled to switch on the data transmission. If there are less than 32 FIFOs in transmit and receive direction, it is necessary to select an existing FIFO number (see Table 4.3 on page 159).

3.4.2.3 Subchannel Processing

If more than one FIFO is connected to one HFC-channel, this HFC-channel number must be written into bitmap V_CH_FNUM of all these FIFOs. In this case every FIFO contributes one or more bits to construct one HFC-channel byte. Unused bits of a HFC-channel byte can be set with an arbitrary mask byte in register A_SUBCH_CFG.

In transparent mode the FIFO data rate always remains 8 kByte/s. In HDLC mode the FIFO data rate

is determined by the number of bits transmitted to the HFC-channel.

Please see Section 3.5 on page 138 for details concerning the subchannel processor.

3.4.2.4 Example for CSM

The example for a *Channel Select Mode* configuration in Figure 3.9 shows three bidirectional connections (❶ FIFO-to-E1, ❷ FIFO-to-PCM and ❸ PCM-to-E1). The black lines illustrate data paths, whereas the dotted lines symbolize blocked resources. These are not used for data transmission, but they are necessary to enable the settings.

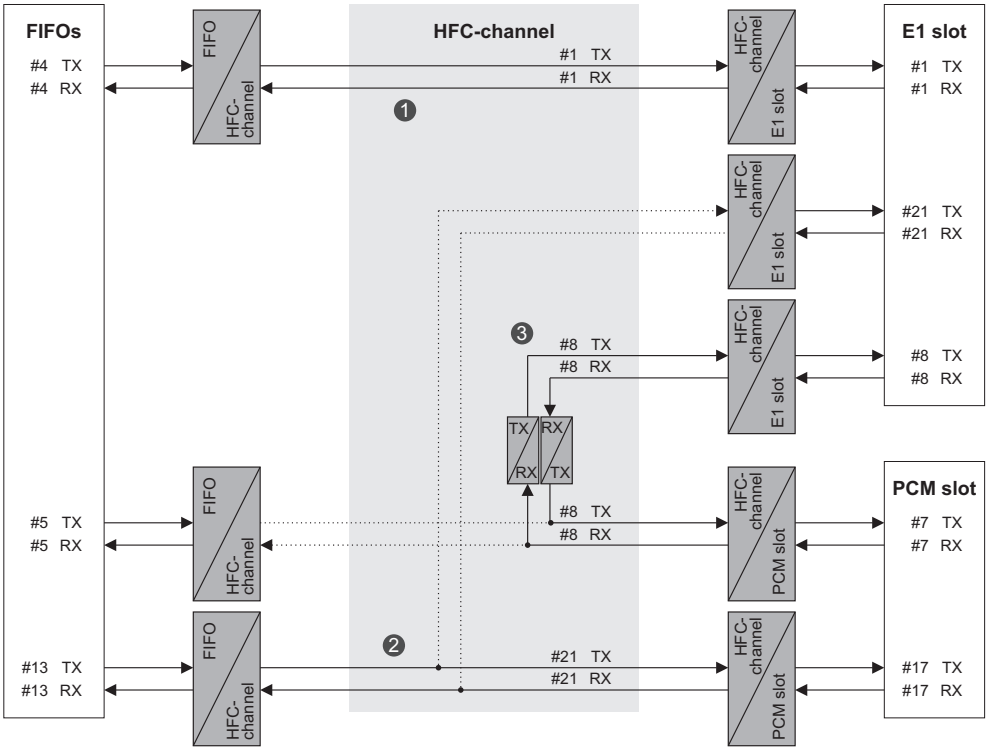


Figure 3.9: *CSM example*

The following settings demonstrate the required register values to establish the connections. All involved FIFOs have to be enabled with either V_HDLC_TRP = '1' (transparent mode and implicit FIFO enable) or V_TRP_IRQ $\neq 0$ (explicit FIFO enable) in register A_CON_HDLC[FIFO].

The subchannel processor and the conference unit are not used in this example. For this reason, registers A_SUBCH_CFG, A_CH_MSK and A_CONF remain in their reset state.

❶ FIFO-to-E1

HFC-channel and FIFO numbers can be chosen independently from each other. This is shown in the FIFO-to-E1 connection.

Due to the user's requirements, V_REV can be programmed either to normal or inverted bit order of the FIFO data.

HDLC or transparent mode (V_HDLC_TRP) can freely be chosen as well. In addition to the settings shown here, a periodic interrupt (in transparent mode) or a *end of frame* interrupt (in

HDLC mode) can be enabled.

If HDLC mode is chosen, the FIFO must be enabled with V_TRP_IRQ $\neq 0$.

Register setup:				(CSM ❶ TX)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	4	(FIFO #4)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[4,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO → E1, FIFO → PCM)
A_CHANNEL[4,TX]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	1	(HFC-channel #1)

Register setup:				(CSM ❶ RX)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	4	(FIFO #4)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[4,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO ← E1)
A_CHANNEL[4,RX]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	1	(HFC-channel #1)

❷ FIFO-to-PCM

The FIFO-to-PCM connection blocks one transmit and one receive E1 time slot .

Again, V_REV and V_HDLC_TRP can freely be chosen according to the user's requirements. As in the previous setting, HDLC mode is selected and the FIFOs are enabled with V_TRP_IRQ $= 1$.

Register setup:				(CSM ❷ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 13		(FIFO #13)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[13,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO → E1, FIFO → PCM)
A_CHANNEL[13,TX]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 21		(HFC-channel #21)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 17		(slot #17)
A_SL_CFG[17,TX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 21		(HFC-channel #21)
	: V_ROUT	= '10'		(data to pin STIO1)

Register setup:				(CSM ❷ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 13		(FIFO #13)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[13,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO ← PCM)
A_CHANNEL[13,RX]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 21		(HFC-channel #21)
R_SLOT	: V_SL_DIR	= 1		(receive slot)
	: V_SL_NUM	= 17		(slot #17)
A_SL_CFG[17,RX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 21		(HFC-channel #21)
	: V_ROUT	= '10'		(data from pin STIO2)

❸ PCM-to-E1

The PCM-to-E1 connection blocks one transmit and one receive FIFO. Although there is no data stored in these FIFOs, they must be enabled to switch on the data transmission between the PCM and the E1 interface.

In receive direction, data is stored in the connected FIFO. But it is not used and needs not to be read. A FIFO overflow has no effect and can be ignored. Consequently, the V_HDLC_TRP setting has no effect to the transferred data between the PCM and the E1 interface neither in receive nor in transmit direction. A PCM-to-E1 connection operates always in transparent mode.

For a PCM-to-E1 connection, the data direction changes between the two interfaces. In detail, data is received on a RX line and then transmitted on a TX line to the other interface. Therefore, a TX-RX-exchanger is inserted for this connection. The blocked FIFOs are on the PCM side

of the TX-RX-exchanger, typically. [6]

Register setup:				(CSM ❸ TX)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	5	(FIFO #5)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[5,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	1	(transparent mode)
	: V_TRP_IRQ	=	0	(interrupt disabled)
	: V_DATA_FLOW	=	'110'	(E1 → PCM)
A_CHANNEL[5,TX]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	8	(HFC-channel #8)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	7	(slot #7)
A_SL_CFG[7,TX]	: V_CH_SDIR	=	0	(transmit HFC-channel)
	: V_CH_SNUM	=	8	(HFC-channel #8)
	: V_ROUT	=	'10'	(data to pin STIO1)

Register setup:				(CSM ❸ RX)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	5	(FIFO #5)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[5,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	1	(transparent mode)
	: V_TRP_IRQ	=	0	(interrupt disabled)
	: V_DATA_FLOW	=	'110'	(FIFO ← E1, E1 ← PCM)
A_CHANNEL[5,RX]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	8	(HFC-channel #8)
R_SLOT	: V_SL_DIR	=	1	(receive slot)
	: V_SL_NUM	=	7	(slot #7)
A_SL_CFG[7,RX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	8	(HFC-channel #8)
	: V_ROUT	=	'10'	(data from pin STIO2)

[6] It is not forbidden to connect the blocked FIFOs at the E1 side of the TX-RX-exchanger. 'Advanced users' might find configurations where this is useful. But all typical configuration settings do not require this exceptional option.

Rule

In *Channel Select Mode*

- every used HFC-channel requires at least one enabled FIFO (except for the PCM-to-PCM connection) with the same data direction and
- every used PCM time slot requires one HFC-channel (except for the PCM-to-PCM connection where a full duplex connection with four time slots allocates only two HFC-channels).

3.4.3 FIFO Sequence Mode (FSM)

3.4.3.1 Mode description

In contrast to the PCM and E1 time slots , the FIFO data rate is not fixed to 8 kByte/s in *FIFO Sequence Mode*. In the previous section the CSM allows the functional capability of a FIFO data rate with less than 8 kByte/s. This section shows how to use FIFOs with a data rate which is higher than 8 kByte/s. In transmit direction one FIFO can cyclically distribute its data to several HFC-channels. In opposite direction, received data from several HFC-channels can be collected cyclically in one FIFO (see Fig. 3.10, right side). A one-to-one connection between FIFO and HFC-channel is also possible in FSM, of course (Fig. 3.10, left side).

Figure 3.10: *HFC-channel assigner in FSM*

FIFO Sequence Mode is selected with V_DF_MD = '11' in register R_FIFO_MD. This data flow mode selects the multi-register R_FSM_IDX at address 0x0F for some FIFO array registers (see Table 3.3 on page 116).

3.4.3.2 FIFO sequence

To achieve a FIFO data rate higher than 8 kByte/s, a FIFO must be connected to more than one HFC-channel. As there is only one register A_CHANNEL[FIFO] for each FIFO, the FSM programming path must differ from the previous modes. Some array registers which are indexed by R_FIFO must be indexed by R_FSM_IDX in *FIFO Sequence Mode* (see Table 3.3).

In FSM all FIFOs are organized in a list with up to 64 entries. Every list entry is assigned to a FIFO. The FIFO configuration can be set up as usual, i.e. HFC-channel allocation, flow controller programming and subchannel processing can be configured as described in the previous sections. Additionally, each list entry specifies the next FIFO of the sequence. The list is terminated by an 'end of list' entry. This procedure is shown in Figure 3.11 with $j+1$ list entries. The first FIFO of the sequence must be specified in register R_FIRST_FIFO.

A quite simple FSM configuration with every FIFO and every HFC-channel specified only one time in the list, would have the same data transmission result as the CSM with an equivalent FIFO $\longleftrightarrow$ HFC-channel setup. But if a specific FIFO is selected n times in the list and connected to n different HFC-channels, the FIFO data rate is $n \cdot 8\,\text{kByte/s}$.

The complete list is processed every 125 µs with ascending list index beginning with 0. Suppose the transmit FIFO m occurs several times in the list. Then the first FIFO byte is transferred to the first connected HFC-channel, the second byte of FIFO m to the second connected HFC-channel and so on. This is similar in receive data direction. The first byte written into FIFO m comes from the first connected HFC-channel, the second byte from the second connected HFC-channel and so on.

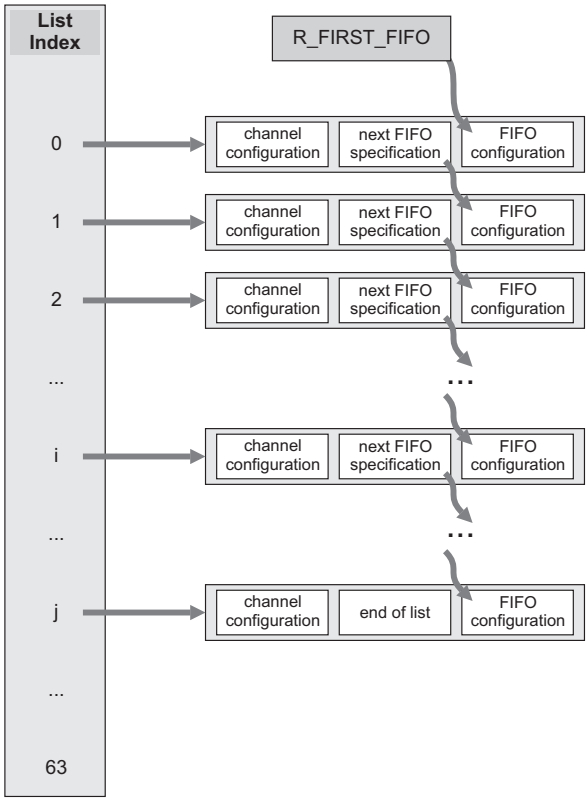


Figure 3.11: *FSM list processing*

3.4.3.3 FSM programming

Register R_FSM_IDX specifies the list index with bitmap V_IDX in the range of $0..63$. R_FSM_IDX is a multi-register and has the same address as R_FIFO because in FSM it replaces R_FIFO for the list programming of the HFC-channel based registers. The array registers A_CHANNEL, A_FIFO_SEQ and A_SUBCH_CFG are indexed with the list index V_IDX instead of the FIFO number (see Table 3.3 on page 116). All other FIFO array registers remain indexed by R_FIFO.

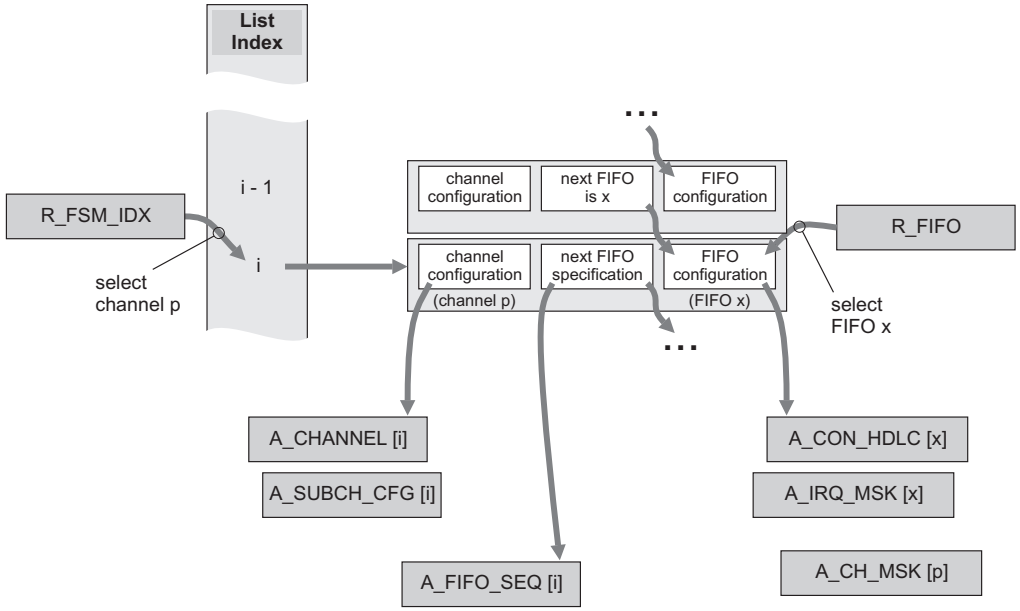


Figure 3.12: *FSM list programming*

The first processed FIFO has to be specified in register R_FIRST_FIFO with the direction bit V_FIRST_FIFO_DIR and the FIFO number V_FIRST_FIFO_NUM. The next FIFO has to be specified in register A_FIFO_SEQ[V_IDX = 0].

A FIFO handles more than one HFC-channel if a FIFO is specified several times in the 'next FIFO' entries.

The FIFO sequence list terminates with V_SEQ_END = '1' in register A_FIFO_SEQ. The other list entries must specify V_SEQ_END = '0' to continue the sequence processing with the next entry.

Programming of the HFC-channel and FIFO registers is shown in Figure 3.12. The connected HFC-channel array registers are indexed by the list index which is written into register R_FSM_IDX. On the other hand, FIFO array registers are indexed by register R_FIFO as usual.

- After writing the list index i into register R_FSM_IDX, the registers A_CHANNEL[i] and A_SUBCH_CFG[i] can be programmed to assign and configure an HFC-channel.
- The next FIFO in the sequence must be specified in register A_FIFO_SEQ[i].
- Supposed, that the previous list entry $i-1$ has specified A_FIFO_SEQ[i-1] = FIFOx, then the corresponding FIFO array registers have to be programmed by first setting R_FIFO = x. Afterwards, registers A_CON_HDLC[x], A_IRQ_MSK[x] and A_CH_MSK can be programmed in the usual way. Please note, that register A_CH_MSK requires the addressed HFC-channel to be specified in register R_FIFO (see remark on page 139).

3.4.3.4 Example for FSM

Figure 3.13 shows an example with three bidirectional connections (❶ 8 kByte/s-FIFO-to-E1, ❷ 8 kByte/s-FIFO-to-PCM and ❸ 16 kByte/s-FIFO-to-E1). The black lines illustrate data paths, whereas the dotted lines symbolize blocked HFC-channels. These are not used for data transmission, but they are necessary to enable the settings.

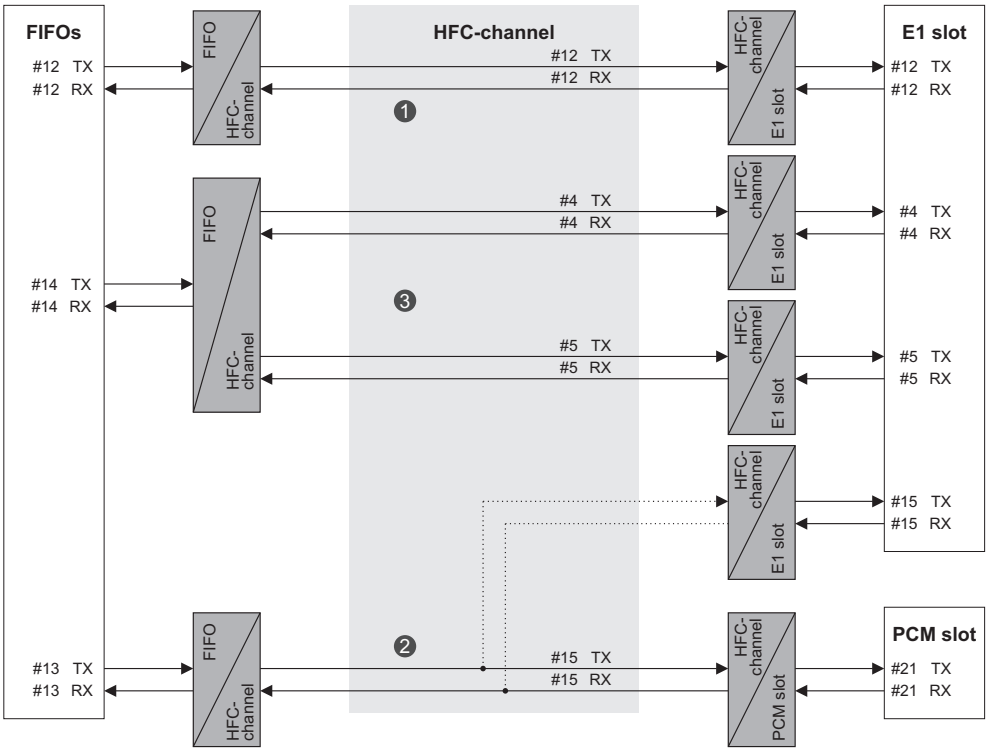


Figure 3.13: *FSM example*

The following settings demonstrate the required register values to establish the connections. All involved FIFOs have to be enabled with either V_HDLC_TRP = '1' (transparent mode and implicit FIFO enable) or V_TRP_IRQ $\neq 0$ (explicit FIFO enable) in register A_CON_HDLC[FIFO].

The subchannel processor and the conference unit are not used in this example. For this reason, registers A_SUBCH_CFG, A_CH_MSK and A_CONF remain in their reset state.

All FIFOs can be arranged in arbitrary order. In the example the list specification of Table 3.4 is chosen. To select FIFO[12,TX] beeing the first FIFO, R_FIRST_FIFO is set as follows:

Register setup:

R_FIRST_FIFO :	V_FIRST_FIFO_DIR	= '0'	(transmit FIFO)
:	V_FIRST_FIFO_NUM	= 12	(FIFO #12)

❶ FIFO-to-E1

The bidirectional FIFO-to-E1 connection use the list indices 0 and 1. Registers A_CHANNEL and A_FIFO_SEQ are indexed by the list index.

Table 3.4: *List specification of the example in Figure 3.13*

Example number	List index	Connection		
❶	0	FIFO[12,TX]	→	E1 slot[12,TX]
❶	1	FIFO[12,RX]	←	E1 slot[12,TX]
❷	2	FIFO[13,RX]	←	PCM time slot[21,RX]
❷	3	FIFO[13,TX]	→	PCM time slot[21,TX]
❸	4	FIFO[14,TX]	→	E1 slot[4,TX]
❸	5	FIFO[14,RX]	←	E1 slot[4,RX]
❸	6	FIFO[14,TX]	→	E1 slot[5,TX]
❸	7	FIFO[14,RX]	←	E1 slot[5,RX]

Register setup:				(FSM ❶ list indices 0 and 1)
R_FSM_IDX	: V_IDX	=	0	(List index #0, used for FIFO[12,TX])
A_CHANNEL[#0]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	12	(HFC-channel #12)
A_FIFO_SEQ[#0]	: V_NEXT_FIFO_DIR	=	1	(next: receive FIFO)
	: V_NEXT_FIFO_NUM	=	12	(next: FIFO #12)
	: V_SEQ_END	=	0	(continue)
R_FSM_IDX	: V_IDX	=	1	(List index #1, used for FIFO[12,RX])
A_CHANNEL[#1]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	12	(HFC-channel #12)
A_FIFO_SEQ[#1]	: V_NEXT_FIFO_DIR	=	1	(next: receive FIFO)
	: V_NEXT_FIFO_NUM	=	13	(next: FIFO #13)
	: V_SEQ_END	=	0	(continue)

The FIFO programming sequence is indexed by the FIFO number and direction. V_REV, V_HDLC_TRP and V_TRP_IRQ can be programmed due to the user's requirements. FIFO[12,TX] and FIFO[12,RX] must both be enabled.

Register setup:				(FSM ❶ FIFO programming for list indices 0 and 1)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	12	(FIFO #12)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[12,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO → E1, FIFO → PCM)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	12	(FIFO #12)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[12,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO ← E1)

❷ FIFO-to-PCM

The following two list entries (indices 2 and 3) define the bidirectional FIFO-to-PCM connection. Two E1 time slots are blocked. But E1 time slot resources are saved because the HFC-channels are assigned to an E-channel which is normally not used and an unused transmit channel.

Register setup:				(FSM ❷ list indices 2 and 3)
R_FSM_IDX	: V_IDX	=	2	(List index #2, used for FIFO[13,RX])
A_CHANNEL[#2]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	15	(HFC-channel #15)
A_FIFO_SEQ[#2]	: V_NEXT_FIFO_DIR	=	0	(next: transmit FIFO)
	: V_NEXT_FIFO_NUM	=	13	(next: FIFO #13)
	: V_SEQ_END	=	0	(continue)
R_FSM_IDX	: V_IDX	=	3	(List index #3, used for FIFO[13,TX])
A_CHANNEL[#3]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	15	(HFC-channel #15)
A_FIFO_SEQ[#3]	: V_NEXT_FIFO_DIR	=	0	(next: transmit FIFO)
	: V_NEXT_FIFO_NUM	=	14	(next: FIFO #14)
	: V_SEQ_END	=	0	(continue)

Register setup:				(FSM ❷ RX FIFO programming for list indices 2 and 3)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	13	(FIFO #13)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[13,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'001'	(FIFO ← PCM)
R_SLOT	: V_SL_DIR	=	1	(receive slot)
	: V_SL_NUM	=	21	(slot #21)
A_SL_CFG[21,RX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	15	(HFC-channel #15)
	: V_ROUT	=	'10'	(data from pin STIO2)

Register setup:				(FSM ❷ TX FIFO programming for list indices 2 and 3)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	13	(FIFO #13)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[13,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	0	(HDLC mode)
	: V_TRP_IRQ	=	1	(enable FIFO)
	: V_DATA_FLOW	=	'001'	(FIFO → E1, FIFO → PCM)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	21	(slot #21)
A_SL_CFG[21,TX]	: V_CH_SDIR	=	0	(transmit HFC-channel)
	: V_CH_SNUM	=	15	(HFC-channel #15)
	: V_ROUT	=	'10'	(data to pin STIO1)

❸ FIFO to multiple E1 time slots

The last setting shows a channel bundling configuration of one FIFO to two B-channels of the E1 interface for both transmit and receive directions. The FIFOs have a data rate of 16 kByte/s each.

Register setup:				(FSM ❸ list indices 4 and 5)
R_FSM_IDX	: V_IDX	= 4		(List index #4, used for FIFO[14,TX])
A_CHANNEL[#4]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 4		(HFC-channel #4)
A_FIFO_SEQ[#4]	: V_NEXT_FIFO_DIR	= 1		(next: receive FIFO)
	: V_NEXT_FIFO_NUM	= 14		(next: FIFO #14)
	: V_SEQ_END	= 0		(continue)
R_FSM_IDX	: V_IDX	= 5		(List index #5, used for FIFO[14,RX])
A_CHANNEL[#5]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 4		(HFC-channel #4)
A_FIFO_SEQ[#5]	: V_NEXT_FIFO_DIR	= 0		(next: transmit FIFO)
	: V_NEXT_FIFO_NUM	= 14		(next: FIFO #14)
	: V_SEQ_END	= 0		(continue)

Register setup:				(FSM ❸ FIFO programming for list indices 4 and 5)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 14		(FIFO #14)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[14,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO → E1, FIFO → PCM)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 14		(FIFO #14)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[14,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	=	'000'	(FIFO ← E1)

When the FIFO[14,TX] and FIFO[14,RX] are used for the second time, they need not to be programmed again. So just the HFC-channels have to programmed for the list indices #6 and #7.

Register setup:				(FSM ❸ list indices 6 and 7)
R_FSM_IDX	: V_IDX	=	6	(List index #6, used for FIFO[14,TX])
A_CHANNEL[#6]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	5	(HFC-channel #5)
A_FIFO_SEQ[#6]	: V_NEXT_FIFO_DIR	=	1	(next: receive FIFO)
	: V_NEXT_FIFO_NUM	=	14	(next: FIFO #14)
	: V_SEQ_END	=	0	(continue)
R_FSM_IDX	: V_IDX	=	7	(List index #7, used for FIFO[14,RX])
A_CHANNEL[#7]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	5	(HFC-channel #5)
A_FIFO_SEQ[#7]	: V_NEXT_FIFO_DIR	=	0	
	: V_NEXT_FIFO_NUM	=	0	
	: V_SEQ_END	=	1	(end of list)

3.5 Subchannel Processing

3.5.1 Overview

Data transmission between a FIFO and the connected HFC-channel can be controlled by the subchannel processor. The behavior of this functional unit depends on the selected data flow mode (SM, CSM or FSM) and the operation mode of the HDLC controller (transparent or HDLC mode). The subchannel controller allows to process less than 8 bits of the transferred FIFO data bytes.

The subchannel processor cannot be used for direct PCM-to-E1 or PCM-to-PCM connections, because a FIFO must participate in the data flow.

A general overview of the subchannel processor in transmit direction is shown as an simplified example in Figure 3.14. Three transmit FIFOs are connected to one HFC-channel. Details of subchannel processing are described in the following sections, partitioned into the different modes of the data flow and the HDLC controller.

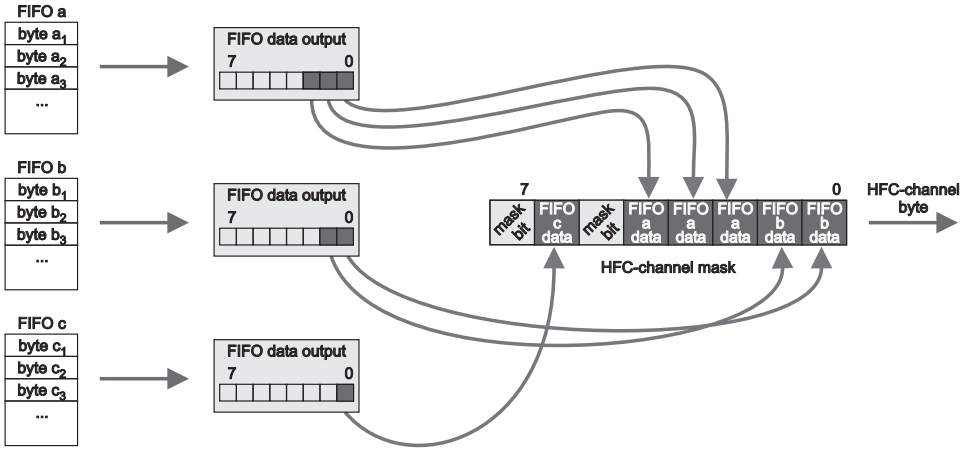


Figure 3.14: *General structure of the subchannel processor shown with an example of three connected FIFOs*

The essence of the subchannel processor is a bit extraction / insertion unit for every FIFO and a byte mask for every HFC-channel. Therefore, the subchannel processor is divided into two parts A and B like shown in Figure 3.15. The behaviour of the FIFO oriented part A depends on the HDLC or transparent mode selection. The HFC-channel oriented part B has a different behaviour due to the selected data flow SM or CSM / FSM.

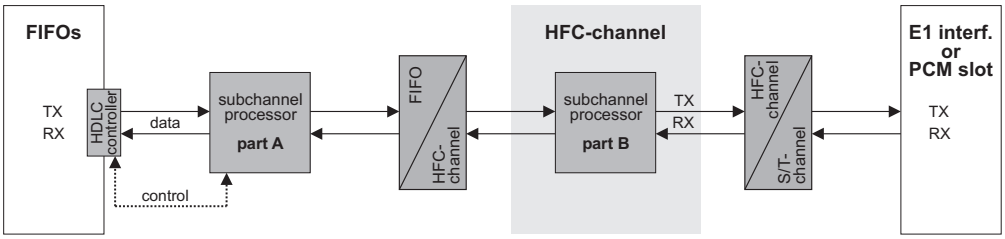


Figure 3.15: *Location of the subchannel parts A and B in the data flow diagram*

3.5.1.1 Registers

The FIFO bit extraction / insertion requires two register settings. V_BIT_CNT defines the number of bits to be extracted / inserted. These bits are always aligned to position 0 in the FIFO data. This bit field can freely be placed in the HFC-channel byte. For this, the start bit can be selected with V_START_BIT in the range of 0 . . 7. Both values are located in register A_SUBCH_CFG[FIFO].

The HFC-channel mask can be stored in register A_CH_MSK[FIFO]. This mask is only used for transmit data. The processed FIFO bits are stored in this register, so it must be re-initialized after changing the settings in A_SUBCH_CFG[FIFO]. Each HFC-channel has its own mask byte. To write this byte for HFC-channel [*n*,TX], the HFC-channel must be written into the multi-register R_FIFO first. The desired mask byte *m* can be written with A_CH_MSK = *m* after this index selection.

> **Important !**
>
> Typically, the multi-register R_FIFO contains always a FIFO index. There is one exception where the R_FIFO value has a different meaning: The HFC-channel mask byte A_CH_MSK is programmed by writing the HFC-channel into register R_FIFO.

The default subchannel configuration in register A_SUBCH_CFG leads to a transparent behavior. That means, only complete data bytes are transmitted in receive and transmit direction.

> **Important !**
>
> For normal data transmission, register A_SUBCH_CFG must be set to 0x00. To use 56 kbit/s restricted mode for U.S. ISDN lines, register A_SUBCH_CFG must be set to 0x07 for B-channels.

3.5.2 Details of the FIFO oriented part of the subchannel processor (part A)

The subchannel processor part A lies between the HDLC controller and the HFC-channel assigner. Figure 3.16 shows the block diagram for both receive and transmit data directions.

At the HDLC controller side, there are a data path and two control lines. These communicate the number of bits to be processed and the HDLC / transparent mode selection between the two modules. In transparent mode always one byte is transferred between the HDLC controller and the subchannel controller part A every 125 µs cycle. In HDLC mode the number of bits is specified by the subchannel bitmap V_BIT_CNT in register A_SUBCH_CFG[FIFO].

On the other side, the data path between subchannel processor part A and the HFC-channel assigner transfers always one byte in transmit and receive direction during every 125 µs cycle.

3.5.2.1 FIFO transmit operation in transparent mode

In transparent mode every FIFO has a data rate of 8 kByte/s. Every 125 µs one byte of a FIFO is processed. The number of bits specified in V_BIT_CNT is placed at position [V_START_BIT +

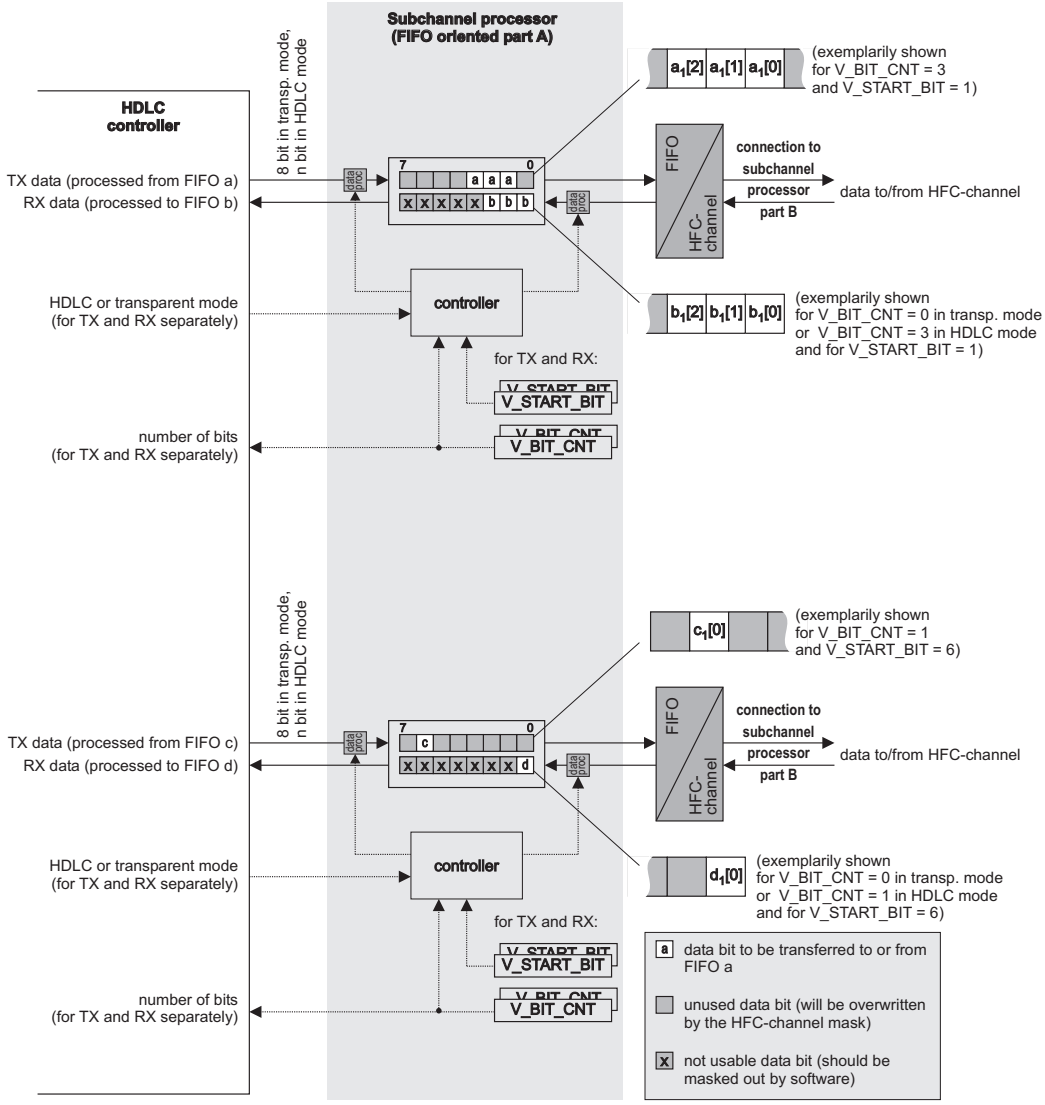


Figure 3.16: *Part A of the subchannel processor*

V_BIT_CNT − 1 .. V_START_BIT] while the other bits are not used and will be overwritten from the HFC-channel mask in part B of the subchannel processor.

3.5.2.2 FIFO transmit operation in HDLC mode

The HDLC mode allows to reduce the data rate of a FIFO. With every 125 µs cycle the subchannel processor requests V_BIT_CNT bits from the HDLC controller. The FIFO data rate is

$$DR_{FIFO} = \begin{cases} \text{V\_BIT\_CNT}\,\text{kBit/s} & : \quad \text{V\_BIT\_CNT} > 0 \\ 8\,\text{kBit/s} & : \quad \text{V\_BIT\_CNT} = 0 \end{cases}$$

or might be a little lower due to the bit stuffing (zero insertion).

3.5.2.3 FIFO receive operation in transparent mode

The subchannel processor part A receives one byte every 125 µs cycle. Typically, only some bits – depending on the usage mode of this receive channel – contain valid data. V_START_BIT defines the position of the valid bit field in the received HFC-channel byte. The subchannel processor part A shifts the valid bit field to position 0 before a whole byte is transferred to the HDLC controller. The invalid bits must be masked out by software. The FIFO data rate is always 8 kByte/s in this configuration.

If transparent mode is selected, V_BIT_CNT must always be '000' in receive direction. The number of valid bits must be handled by the software.

3.5.2.4 FIFO receive operation in HDLC mode

From every received HFC-channel data byte only V_BIT_CNT bits beginning at position V_START_BIT contain valid data. Only these bits are transferred to the HDLC controller. So the FIFO data rate is

$$DR_{FIFO} = \begin{cases} \text{V\_BIT\_CNT}\,\text{kBit/s} & : \quad \text{V\_BIT\_CNT} > 0 \\ 8\,\text{kBit/s} & : \quad \text{V\_BIT\_CNT} = 0 \end{cases}$$

or might be a little lower due to the bit stuffing (zero deletion).

3.5.3 Details of the HFC-channel oriented part of the subchannel processor (part B)

Part B of the subchannel processor is located inside the HFC-channel area. With every 125 µs cycle it transmits and receives always one data byte to / from the connected interface (either PCM or E1 interface). On the other side, to / from every connected HFC-channel assigner one byte is transferred in both transmit and receive directions. Figure 3.17 shows the block diagram of this module.

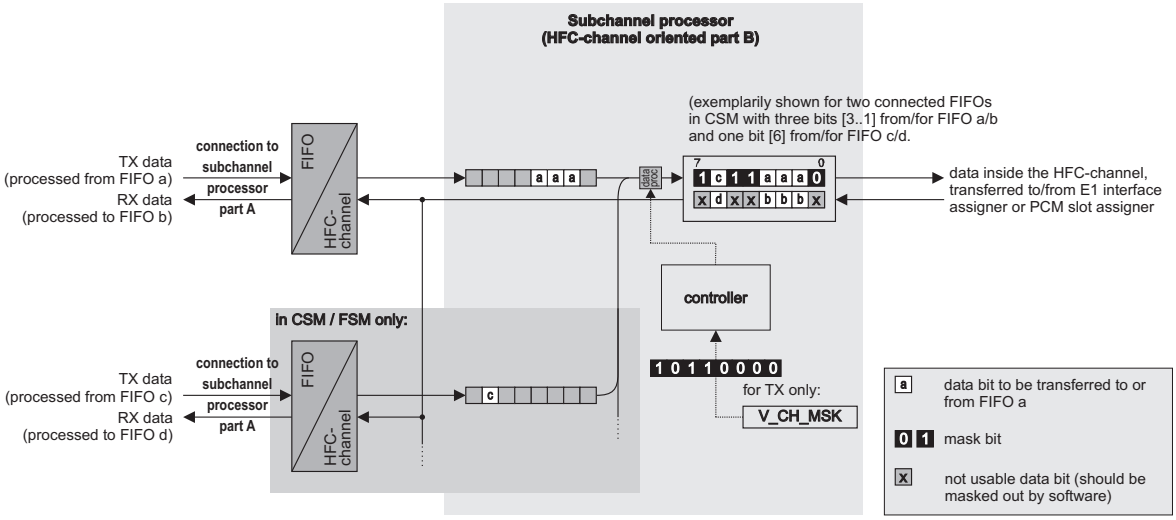


Figure 3.17: *Part B of the subchannel processor*

3.5.3.1 FIFO transmit operation in SM

As the FIFO and HFC-channel numbers are the same in *Simple Mode*, only one FIFO can be connected to a HFC-channel. Subchannel processing can do nothing more than masking out some bits of every transmitted data byte.

The specified bit field is put into the HFC-channel mask byte before the data byte is transmitted to the connected interface.

3.5.3.2 FIFO transmit operation in CSM and FSM

In *Channel Select Mode* and *FIFO Sequence Mode*, several FIFOs can contribute data to one HFC-channel data byte. From every connected HFC-channel assigner, one or more bits are extracted and are joined to a single HFC-channel data byte.

Here, the subchannel processor works in the same way as in *Simple Mode*, except that multiple bit insertion is performed. All FIFOs which contribute data bits to the HFC-channel byte should specify different bit locations to avoid overwriting data.

3.5.3.3 FIFO receive operation in SM

The received data byte is transferred to the HFC-channel assigner without modification. Part B of the subchannel processor has no effect to the receive data. Typically, only some bits contain valid data which will be extracted by the part A of the subchannel processor.

3.5.3.4 FIFO receive operation in CSM and FSM

If there are several FIFOs connected to one receive HFC-channel in *Channel Select Mode* or *FIFO Sequence Mode*, every received data byte is transferred to all connected HFC-channel assigners without modification. Part B of the subchannel processor has no effect to the receive data. Typically, the HFC-channel data byte contains bit fields for several FIFOs which will be extracted by their part A of the subchannel processor.

3.5.4 Subchannel example for SM

The subchannel processing example in Figure 3.18 shows two bidirectional configurations (❶ FIFO-to-E1 and ❷ FIFO-to-PCM) in *Simple Mode*.

> **Please note !**
>
> All subchannel examples in this document have always the same number of bits and the same start bit for corresponding transmit and receive FIFOs. Actually, transmit and receive configuration settings are independently from each other. The settings are chosen for clearness and can simply be reproduced with looped data pathes.

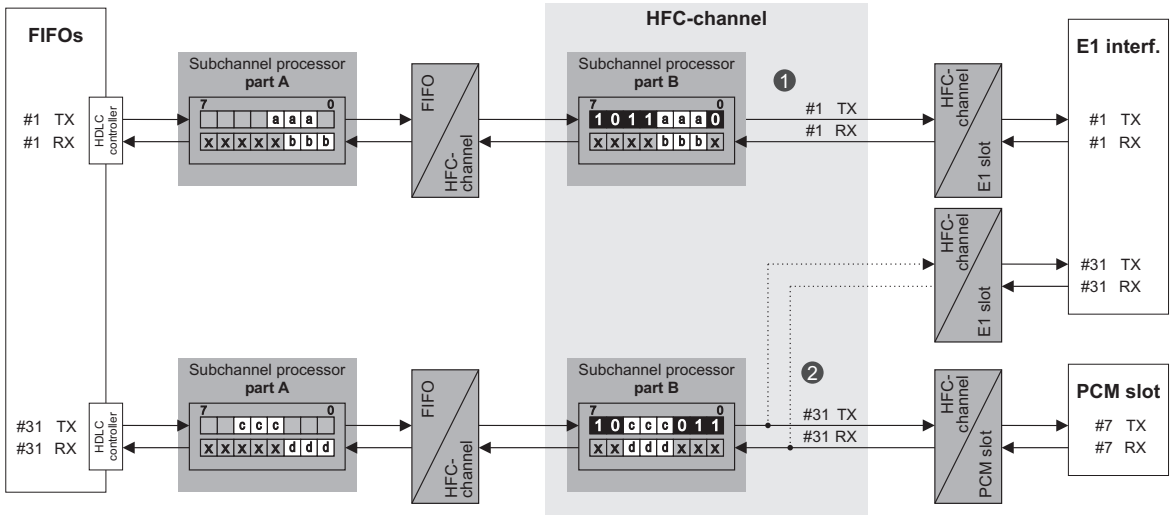


Figure 3.18: *SM example with subchannel processor*

❶ FIFO-to-E1 (TX)

The first setting shows a FIFO-to-E1 data transmission in transparent mode.

Register A_SUBCH_CFG[FIFO] defines three bits [2 . . 0] to be transmitted from each FIFO byte. These bits have the position [3 . . 1] in the HFC-channel data byte.

All other data bits in the HFC-channel byte are defined by the HFC-channel mask V_CH_MSK = '1011 0000' in register A_CH_MSK. This array register must be selected by writing the HFC-channel number and direction into register R_FIFO. The mask bits [3 . . 1] are *don't care* because they are overwritten from the FIFO data.

A detailed overview of the transmitted data is shown in Table 3.5. The first data byte in FIFO[1,TX] is a_1, the second byte is a_2, and so on. In transparent mode only $(a_1[2..0], a_2[2..0], \ldots)$ are placed in the HFC-channel bytes at the location $[3..1]$ and $(a_1[7..3], a_2[7..3], \ldots)$ are ignored and replaced by the HCF-channel mask.

Register setup:				(SM ❶ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 1		(FIFO #1)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[1,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO → E1, FIFO → PCM)
A_SUBCH_CFG[1,TX]	: V_BIT_CNT	= 3		(process 3 bits)
	: V_START_BIT	= 1		(start with bit 1)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit HFC-channel)
	: V_FIFO_NUM	= 1		(HFC-channel #1)
	: V_REV	= 0		(normal bit order)
A_CH_MSK[1,TX]	: V_CH_MSK	= '10110000'		(mask byte)

❶ FIFO-to-E1 (RX)

Only three bits $[3..1]$ from the received HFC-channel byte are assumed to be valid data. Nevertheless, the number of received bits must be set to the value V_BIT_CNT = 0 which means 'one byte'. The start position is specified with V_START_BIT = 1 in register A_SUBCH_CFG. As the received bit field is aligned to position 0, these bits represent FIFO data $b[2..0]$.

A detailed overview of the received data is shown in Table 3.6. The first data byte in FIFO[1,RX] is b_1, the second byte is b_2, and so on. Only $(b_1[2..0], b_2[2..0], \ldots)$ contain valid data and $(b_1[7..3], b_2[7..3], \ldots)$ must be masked out by software.

Register setup:				(SM ❶ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 1		(FIFO #1)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[1,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO ← E1)
A_SUBCH_CFG[1,RX]	: V_BIT_CNT	= 0		(process 8 bits)
	: V_START_BIT	= 1		(start with bit 1)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)

Table 3.5: *Subchannel processing according to Figure 3.18 (SM ❶ TX, transparent mode)*

	7							0
HFC-channel mask:	1	0	1	1	0	0	0	0
HFC-channel transmit byte 1:	1	0	1	1	$a_1[2]$	$a_1[1]$	$a_1[0]$	0
HFC-channel transmit byte 2:	1	0	1	1	$a_2[2]$	$a_2[1]$	$a_2[0]$	0
HFC-channel transmit byte 3:	1	0	1	1	$a_3[2]$	$a_3[1]$	$a_3[0]$	0
...					...			

Table 3.6: *Subchannel processing according to Figure 3.18 (SM ❶ RX, transparent mode)*

	7							0
HFC-channel receive byte 1:	x	x	x	x	$b_1[2]$	$b_1[1]$	$b_1[0]$	x
HFC-channel receive byte 2:	x	x	x	x	$b_2[2]$	$b_2[1]$	$b_2[0]$	x
HFC-channel receive byte 3:	x	x	x	x	$b_3[2]$	$b_3[1]$	$b_3[0]$	x
...					...			

❷ FIFO-to-PCM (TX)

The second *Simple Mode* configuration connects a FIFO in HDLC mode with the PCM interface [7]. Bitmap V_BIT_CNT in register A_SUBCH_CFG[FIFO] defines three FIFO data bits to be transmitted during every 125 µs cycle. The bit field location in the HFC-channel data byte is specified by bitmap V_START_BIT in the same register.

All other data bits in the HFC-channel are defined by the HFC-channel mask in register A_CH_MSK. This array register must be selected by writing the HFC-channel number and direction into register R_FIFO. The mask bits [5 . . 3] are *don't care* because they are overwritten from the FIFO data.

A detailed overview of the transmitted data is shown in Table 3.7. The first data byte in FIFO[31,TX] is c_1, the second byte is c_2, and so on. In HDLC mode, FIFO bytes are dispersed among several HFC-channel bytes.

[7] HDLC bit stuffing is not shown in this example.

Register setup:				(SM ❷ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 31		(FIFO #31)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[31,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO → E1, FIFO → PCM)
A_SUBCH_CFG[31,TX]	: V_BIT_CNT	= 3		(process 3 bits)
	: V_START_BIT	= 3		(start with bit 3)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit HFC-channel)
	: V_FIFO_NUM	= 31		(HFC-channel #31)
	: V_REV	= 0		(normal bit order)
A_CH_MSK[31,TX]	: V_CH_MSK	= '10110011'		(mask byte)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 7		(slot #7)
A_SL_CFG[7,TX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '10'		(data to pin STIO1)

❷ FIFO-to-PCM (RX)

Only three bits [5 . . 3] from the received HFC-channel byte are assumed to be valid data. This is done with the bitmap values V_BIT_CNT = 3 and V_START_BIT = 3 in register A_SUBCH_CFG. The bit field is aligned to position 0 and transferred to the HDLC controller. There, FIFO data bytes are constructed from several received bit fields.

A detailed overview of the received data is shown in Table 3.8. The first data byte in FIFO[31,RX] is d_1, the second byte is d_2, and so on. In HDLC mode, FIFO bytes are constructed from several HFC-channel bytes.

Register setup:				(SM ❷ RX)
R_FIFO	: V_FIFO_DIR	= 1	(receive FIFO)	
	: V_FIFO_NUM	= 31	(FIFO #31)	
	: V_REV	= 0	(normal bit order)	
A_CON_HDLC[31,RX]	: V_IFF	= 0	(0x7E as inter frame fill)	
	: V_HDLC_TRP	= 0	(HDLC mode)	
	: V_TRP_IRQ	= 1	(enable FIFO)	
	: V_DATA_FLOW	= '001'	(FIFO ← PCM)	
A_SUBCH_CFG[31,RX]	: V_BIT_CNT	= 3	(process 3 bits)	
	: V_START_BIT	= 3	(start with bit 3)	
	: V_LOOP_FIFO	= 0	(normal operation)	
	: V_INV_DATA	= 0	(normal data transmission)	
R_SLOT	: V_SL_DIR	= 1	(receive slot)	
	: V_SL_NUM	= 7	(slot #7)	
A_SL_CFG[7,RX]	: V_CH_SDIR	= 1	(receive HFC-channel)	
	: V_CH_SNUM	= 31	(HFC-channel #31)	
	: V_ROUT	= '10'	(data from pin STIO2)	

Table 3.7: *Subchannel processing according to Figure 3.18 (SM ❷ TX, HDLC mode)*

	7							0
HFC-channel mask:	1	0	0	0	0	0	1	1
HFC-channel transmit byte 1:	1	0	$c_1[2]$	$c_1[1]$	$c_1[0]$	0	1	1
HFC-channel transmit byte 2:	1	0	$c_1[5]$	$c_1[4]$	$c_1[3]$	0	1	1
HFC-channel transmit byte 3:	1	0	$c_2[0]$	$c_1[7]$	$c_1[6]$	0	1	1
HFC-channel transmit byte 4:	1	0	$c_2[3]$	$c_2[2]$	$c_2[1]$	0	1	1
...				...				

3.5.5 Subchannel example for CSM

In *Channel Select Mode* up to 8 FIFOs can be assigned to one HFC-channel if only 1 bit is processed by every FIFO. The example in Figure 3.19 shows two bidirectional configurations (❶ FIFO-to-E1 and ❷ FIFO-to-PCM) with two FIFOs per direction each.

❶ FIFO-to-E1 (TX)

In the first setting two transmit FIFOs are connected to one HFC-channel. Transparent mode is selected in this example.

Registers A_SUBCH_CFG[FIFO] of FIFO[4,TX] and FIFO[5,TX] define both, the number of bits to be extracted from the FIFO data bytes and their position in the HFC-channel byte.

Table 3.8: *Subchannel processing according to Figure 3.18 (SM ❷ RX, HDLC mode)*

	7							0
HFC-channel receive byte 1:	x	x	$d_1[2]$	$d_1[1]$	$d_1[0]$	x	x	x
HFC-channel receive byte 2:	x	x	$d_1[5]$	$d_1[4]$	$d_1[3]$	x	x	x
HFC-channel receive byte 3:	x	x	$d_2[0]$	$d_1[7]$	$d_1[6]$	x	x	x
HFC-channel receive byte 4:	x	x	$d_2[3]$	$d_2[2]$	$d_2[1]$	x	x	x
...			...					

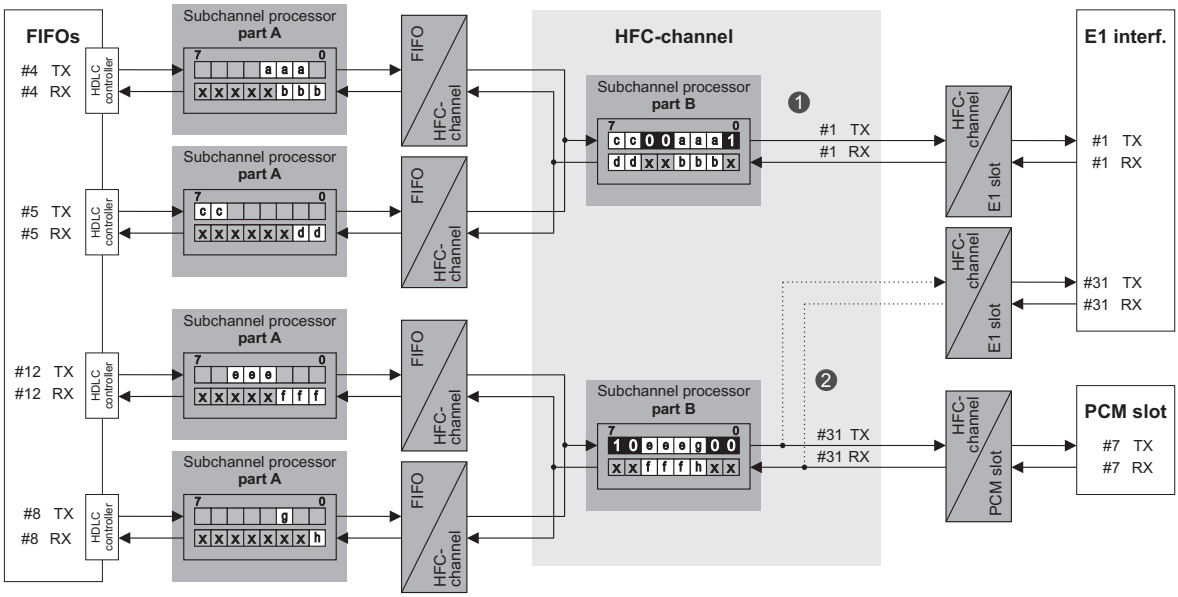


Figure 3.19: *CSM example with subchannel processor*

The HFC-channel mask in register A_CH_MSK defines the bit values that are not used for FIFO data. The array register must be selected by writing the HFC-channel number and direction into register R_FIFO. The mask bits [7 . . 6, 3 . . 1] are *don't care* because they are overwritten from the FIFO data.

A detailed overview of the transmitted data is shown in Table 3.9. The first data byte in FIFO[4,TX] is a_1, the second byte is a_2, and so on. FIFO[5,TX] is represented by the data bytes c_1, c_2, and so on.

Register setup:				(CSM ❶ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 4		(FIFO #4)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[4,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO → E1, FIFO → PCM)
A_CHANNEL[4,TX]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 1		(HFC-channel #1)
A_SUBCH_CFG[4,TX]	: V_BIT_CNT	= 3		(process 3 bits)
	: V_START_BIT	= 1		(start with bit 1)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 5		(FIFO #5)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[5,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO → E1, FIFO → PCM)
A_CHANNEL[5,TX]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 1		(HFC-channel #1)
A_SUBCH_CFG[5,TX]	: V_BIT_CNT	= 2		(process 2 bits)
	: V_START_BIT	= 6		(start with bit 6)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit HFC-channel)
	: V_FIFO_NUM	= 1		(HFC-channel #1)
	: V_REV	= 0		(normal bit order)
A_CH_MSK[0,TX]	: V_CH_MSK	= '0000 0001'		(mask byte)

❶ FIFO-to-E1 (RX)

The received HFC-channel byte is distributed to two FIFOs. Bit fields [7..6] and [3..1] from the received HFC-channel byte are assumed to be valid data. Nevertheless, the number of received bits must be set to the value V_BIT_CNT = 0 which means 'one byte'. The start position is specified with V_START_BIT in register A_SUBCH_CFG. As the received bit fields are aligned to position 0, these bits represent FIFO data $b[2..0]$ and $d[1..0]$.

A detailed overview of the received data is shown in Table 3.10. The first data byte in FIFO[4,RX] is b_1, the second byte is b_2, and so on. FIFO[5,RX] data bytes are d_1, d_2, and so on.

Register setup:				(CSM ❶ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 4		(FIFO #4)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[4,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO ← E1)
A_CHANNEL[4,RX]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 1		(HFC-channel #1)
A_SUBCH_CFG[4,RX]	: V_BIT_CNT	= 0		(process 8 bits)
	: V_START_BIT	= 1		(start with bit 1)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 5		(FIFO #5)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[5,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 1		(transparent mode)
	: V_TRP_IRQ	= 1		(interrupt all 64 bytes)
	: V_DATA_FLOW	= '000'		(FIFO ← E1)
A_CHANNEL[5,RX]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 1		(HFC-channel #1)
A_SUBCH_CFG[5,RX]	: V_BIT_CNT	= 0		(process 8 bits)
	: V_START_BIT	= 6		(start with bit 6)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)

Table 3.9: *Subchannel processing according to Figure 3.19 (CSM ❶ TX, transparent mode)*

	7							0
HFC-channel mask:	0	0	0	0	0	0	0	1
HFC-channel transmit byte 1:	$c_1[1]$	$c_1[0]$	0	0	$a_1[2]$	$a_1[1]$	$a_1[0]$	1
HFC-channel transmit byte 2:	$c_2[1]$	$c_2[0]$	0	0	$a_2[2]$	$a_2[1]$	$a_2[0]$	1
HFC-channel transmit byte 3:	$c_3[1]$	$c_3[0]$	0	0	$a_3[2]$	$a_3[1]$	$a_3[0]$	1
...				...				

❷ FIFO-to-PCM (TX)

A FIFO-to-PCM configuration in HDLC mode with two FIFOs in transmit and receive direction

Table 3.10: *Subchannel processing according to Figure 3.19 (CSM ❶ RX, transparent mode)*

	7							0
HFC-channel transmit byte 1:	$d_1[1]$	$d_1[0]$	x	x	$b_1[2]$	$b_1[1]$	$b_1[0]$	x
HFC-channel transmit byte 2:	$d_2[1]$	$d_2[0]$	x	x	$b_2[2]$	$b_2[1]$	$b_2[0]$	x
HFC-channel transmit byte 3:	$d_3[1]$	$d_3[0]$	x	x	$b_3[2]$	$b_3[1]$	$b_3[0]$	x
...				...				

each is shown in the second example setting [8].

Registers A_SUBCH_CFG[FIFO] of FIFO[12,TX] and FIFO[8,TX] define both, the numbers of FIFO data bits to be transmitted during every 125 µs cycle and their position in the HFC-channel byte.

All other data bits in the HFC-channel are defined by the HFC-channel mask in register A_CH_MSK. This array register must be selected by writing the HFC-channel number and direction into register R_FIFO. The mask bits [5 . . 2] are *don't care* because they are overwritten from the FIFO data.

A detailed overview of the transmitted data is shown in Table 3.11. The first data byte in FIFO[12,TX] is e_1, the second byte is e_2, and so on. FIFO[8,TX] transmits bytes g_1, g_2, and so on. In HDLC mode, FIFO bytes are dispersed among several HFC-channel bytes.

[8] HDLC bit stuffing is not shown in this example.

Register setup:				(CSM ❷ TX)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 12		(FIFO #12)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[12,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO → E1, FIFO → PCM)
A_CHANNEL[12,TX]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 31		(HFC-channel #31)
A_SUBCH_CFG[12,TX]	: V_BIT_CNT	= 3		(process 3 bits)
	: V_START_BIT	= 3		(start with bit 3)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit FIFO)
	: V_FIFO_NUM	= 8		(FIFO #8)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[8,TX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO → E1, FIFO → PCM)
A_CHANNEL[8,TX]	: V_CH_FDIR	= 0		(transmit HFC-channel)
	: V_CH_FNUM	= 31		(HFC-channel #31)
A_SUBCH_CFG[8,TX]	: V_BIT_CNT	= 1		(process 1 bit)
	: V_START_BIT	= 2		(start with bit 2)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 0		(transmit HFC-channel)
	: V_FIFO_NUM	= 31		(HFC-channel #31)
	: V_REV	= 0		(normal bit order)
A_CH_MSK[31,TX]	: V_CH_MSK	= '1000 1100'		(mask byte)
R_SLOT	: V_SL_DIR	= 0		(transmit slot)
	: V_SL_NUM	= 7		(slot #7)
A_SL_CFG[7,TX]	: V_CH_SDIR	= 0		(transmit HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '10'		(data to pin STIO1)

❷ FIFO-to-PCM (RX)

HFC-channel[31,RX] receives data bits that are to be distributed to FIFO[12,RX] and FIFO[8,RX].

Registers A_SUBCH_CFG[FIFO] of FIFO[12,RX] and FIFO[8,RX] define the numbers of valid data bits and their positions in the HFC-channel byte. These bits are dispersed to FIFO[12,RX] and FIFO[8,RX] where they are aligned to bit 0.

A detailed overview of the received data is shown in Table 3.12. The first data byte in FIFO[12,RX] is f_1, the second byte is f_2, and so on. FIFO[8,RX] receives bytes h_1, h_2, and so on. In HDLC mode, FIFO bytes are collected from several HFC-channel bytes.

Register setup:				(CSM ❷ RX)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 12		(FIFO #12)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[12,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO ← PCM)
A_CHANNEL[12,RX]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 31		(HFC-channel #31)
A_SUBCH_CFG[12,TX]	: V_BIT_CNT	= 3		(process 3 bits)
	: V_START_BIT	= 3		(start with bit 3)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_FIFO	: V_FIFO_DIR	= 1		(receive FIFO)
	: V_FIFO_NUM	= 8		(FIFO #8)
	: V_REV	= 0		(normal bit order)
A_CON_HDLC[8,RX]	: V_IFF	= 0		(0x7E as inter frame fill)
	: V_HDLC_TRP	= 0		(HDLC mode)
	: V_TRP_IRQ	= 1		(enable FIFO)
	: V_DATA_FLOW	= '001'		(FIFO ← PCM)
A_CHANNEL[8,RX]	: V_CH_FDIR	= 1		(receive HFC-channel)
	: V_CH_FNUM	= 31		(HFC-channel #31)
A_SUBCH_CFG[8,TX]	: V_BIT_CNT	= 1		(process 1 bit)
	: V_START_BIT	= 2		(start with bit 2)
	: V_LOOP_FIFO	= 0		(normal operation)
	: V_INV_DATA	= 0		(normal data transmission)
R_SLOT	: V_SL_DIR	= 1		(receive slot)
	: V_SL_NUM	= 7		(slot #7)
A_SL_CFG[7,RX]	: V_CH_SDIR	= 1		(receive HFC-channel)
	: V_CH_SNUM	= 31		(HFC-channel #31)
	: V_ROUT	= '10'		(data from pin STIO2)

Table 3.11: *Subchannel processing according to Figure 3.19 (CSM ❷ TX, HDLC mode)*

	7							0
HFC-channel mask:	1	0	0	0	1	1	0	0
HFC-channel transmit byte 1:	1	0	$e_1[2]$	$e_1[1]$	$e_1[0]$	$g_1[0]$	0	0
HFC-channel transmit byte 2:	1	0	$e_1[5]$	$e_1[4]$	$e_1[3]$	$g_1[1]$	0	0
HFC-channel transmit byte 3:	1	0	$e_2[0]$	$e_1[7]$	$e_1[6]$	$g_1[2]$	0	0
HFC-channel transmit byte 4:	1	0	$e_2[3]$	$e_2[2]$	$e_2[1]$	$g_1[3]$	0	0
...				...				

Table 3.12: *Subchannel processing according to Figure 3.19 (CSM ❷ RX, HDLC mode)*

	7							0
HFC-channel transmit byte 1:	x	x	$f_1[2]$	$f_1[1]$	$f_1[0]$	$h_1[0]$	x	x
HFC-channel transmit byte 2:	x	x	$f_1[5]$	$f_1[4]$	$f_1[3]$	$h_1[1]$	x	x
HFC-channel transmit byte 3:	x	x	$f_2[0]$	$f_1[7]$	$f_1[6]$	$h_1[2]$	x	x
HFC-channel transmit byte 4:	x	x	$f_2[3]$	$f_2[2]$	$f_2[1]$	$h_1[3]$	x	x
...				...				

Chapter 4

FIFO handling and HDLC controller

Table 4.1: *Overview of the HFC-E1 FIFO registers*

Write only registers:

Address	Name	Page
0x0B	R_FIRST_FIFO	165
0x0D	R_FIFO_MD	166
0x0E	A_INC_RES_FIFO	167
0x0F	R_FIFO	168
0x0F	R_FSM_IDX	168
0x84	A_FIFO_DATA0_NOINC	169
0x84	A_FIFO_DATA1_NOINC	169
0x84	A_FIFO_DATA2_NOINC	169
0xF4	A_CH_MSK	170
0xFA	A_CON_HDLC	171
0xFB	A_SUBCH_CFG	173
0xFC	A_CHANNEL	174
0xFD	A_FIFO_SEQ	175

Read only register:

Address	Name	Page
0x04	A_Z1L	176
0x05	A_Z1H	177
0x06	A_Z2L	177
0x07	A_Z2H	178
0x0C	A_F1	178
0x0D	A_F2	179
0x88	R_INT_DATA	179

Read / write registers:

Address	Name	Page
0x80	A_FIFO_DATA0	180
0x80	A_FIFO_DATA1	180
0x80	A_FIFO_DATA2	180

4.1 Overview

There are up to 32 receive FIFOs and up to 32 transmit FIFOs with 64 HDLC controllers in whole. The HDLC circuits are located on the E1 interface side of the FIFOs. Thus plain data is always stored in the FIFOs. Automatic zero insertion is done in HDLC mode when HDLC data goes from the FIFOs to the E1 interface or to the PCM bus (transmit FIFO operation). Automatic zero deletion is done in HDLC mode when the HDLC data comes from the E1 interface or PCM bus (receive FIFO operation).

There is a transmit and a receive FIFO for each E1 time slot (even for time slot 0).

The FIFO control registers are used to select and control the FIFOs of HFC-E1. The FIFO register set exists for every FIFO number and receive / transmit direction. The FIFO is selected by the FIFO select register R_FIFO.

All FIFOs are disabled after reset (hardware reset, soft reset or HFC reset). The selected FIFO is enabled with register A_CON_HDLC by setting at least one of V_HDLC_TRP or V_TRP_IRQ to a value different from zero.

4.2 FIFO counters

The FIFOs are realized as ring buffers in the internal or external SRAM. They are controlled by counters. The counter sizes depend on the setting of the FIFO sizes. $Z1$ is the FIFO input counter and $Z2$ is the FIFO output counter.

Each counter points to a byte position in the SRAM. On a FIFO input operation $Z1$ is incremented. On an output operation $Z2$ is incremented. If $Z1 = Z2$ the FIFO is empty.

After every pulse on the F0IO signal, HDLC bytes are written into the E1 interface (from a transmit FIFO) and HDLC bytes are read from the E1 interface (to a receive FIFO). A connection to the PCM interface is also possible.

Additionally there are two counters $F1$ and $F2$ for every FIFO for counting the HDLC frames. Their width is 4 bit for 32 KByte SRAM and 5 bit for larger SRAMs. They form a ring buffer as $Z1$ and $Z2$ do, too.

Table 4.2: F-*counter range with different RAM sizes*

RAM size	F_{MIN}	F_{MAX}
32k x 8	0x00	0x0F
128k x 8	0x00	0x1F
512k x 8	0x00	0x1F

$F1$ is incremented when a complete frame has been received and stored in the FIFO. $F2$ is incremented when a complete frame has been read from the FIFO. If $F1 = F2$ there is no complete frame in the FIFO.

The reset state of the Z- and F-counters are

- $Z1 = Z2 = Z_{MAX}$ [1] and
- $F1 = F2 = F_{MAX}$ [2].

This initialization can be carried out with a soft reset or a HDLC reset. For this, bit V_SRES or bit V_HFC_RES in register R_CIRM has to be set. Individual FIFOs can be reset with bit V_RES_FIFO in register A_INC_RES_FIFO.

In addition, a hardware reset initializes the counters.

> **Important !**
>
> **Busy status after FIFO change, FIFO reset and F1 / F2 incrementation**
>
> Changing a FIFO, reseting a FIFO or incrementing the F-counters causes a short BUSY period of HFC-E1. This means an access to FIFO control registers is not allowed until BUSY status is cleared (bit V_BUSY in register R_STATUS). The maximum duration takes 25 clock cycles ($\sim$1 µs). Status, interrupt and control registers can be read and written at any time.

> **Please note !**
>
> The counter state Z_{MIN} (resp. F_{MIN}) of the Z-counters (resp. F-counters) follows counter state Z_{MAX} (resp. F_{MAX}) in the FIFOs.
>
> Please note that Z_{MIN} and Z_{MAX} depend on the FIFO number and FIFO size (s. Section 4.3 and Table 4.3).

4.3 FIFO size setup

HFC-E1 can operate with 32k x 8 internal or alternatively with 128k x 8 or 512k x 8 external SRAM. Bitmap V_RAM_SZ in register R_RAM_MISC must be set accordingly to the RAM size. Table 4.3 shows how the FIFO size can be varied with the different RAM sizes. Additionally, the initial Z_{max} and Z_{min} values are given in Table 4.3.

After changing the FIFO size or RAM size a soft reset should be initiated.

[1] See Z_{max} value in Table 4.3.
[2] See F_{max} value in Table 4.2.

4.4 External SRAM

The maximum FIFO size depends on the SRAM size. The internal 32k x 8 SRAM will satisfy most applications needs. An external SRAM can be used to enlarge the memory size. The following characteristics are required:

- Asynchronous SRAM
- 128k x 8 or 512k x 8
- high speed (20 ns for $f_{SYS} = 24.576\,\text{MHz}$, 10 ns for $f_{SYS} = 49.152\,\text{MHz}$)

Table 4.3: *FIFO size setup*

		32k x 8 RAM (internal) V_RAM_SZ = 0x00 F_{MIN} = 0x00, F_{MAX} = 0x0F				128k x 8 RAM (external) V_RAM_SZ = 0x01 F_{MIN} = 0x00, F_{MAX} = 0x1F				512k x 8 RAM (external) V_RAM_SZ = 0x02 F_{MIN} = 0x00, F_{MAX} = 0x1F			
V_FIFO_MD	V_FIFO_SZ	FIFO number	Z_{MIN}	Z_{MAX}	FIFO size (byte)	FIFO number	Z_{MIN}	Z_{MAX}	FIFO size (byte)	FIFO number	Z_{MIN}	Z_{MAX}	FIFO size (byte)
'00'	'00'	0 .. 31	0x80	0x1FF	384	0 .. 31	0xC0	0x07FF	1856	0 .. 31	0xC0	0x1FFF	8000
'10'	'00'	0 .. 15	0x80	0x0FF	128	0 .. 15	0xC0	0x03FF	832	0 .. 15	0xC0	0x0FFF	3904
		16 .. 31	0x00	0x1FF	512	16 .. 31	0x00	0x07FF	2048	16 .. 31	0x00	0x1FFF	8192
'10'	'01'	0 .. 23	0x80	0x0FF	128	0 .. 23	0xC0	0x03FF	832	0 .. 23	0xC0	0x0FFF	3904
		24 .. 31	0x00	0x3FF	1024	24 .. 31	0x00	0x0FFF	4096	24 .. 31	0x00	0x3FFF	16384
'10'	'10'	0 .. 27	0x80	0x0FF	128	0 .. 27	0xC0	0x03FF	832	0 .. 27	0xC0	0x0FFF	3904
		28 .. 31	0x00	0x7FF	2048	28 .. 31	0x00	0x1FFF	8192	28 .. 31	0x00	0x7FFF	32768
'10'	'11'	0 .. 29	0x80	0x0FF	128	0 .. 29	0xC0	0x03FF	832	0 .. 29	0xC0	0x0FFF	3904
		30 .. 31	0x00	0xFFF	4096	30 .. 31	0x00	0x3FFF	16384	30 .. 31	0x00	0xFFFF	65536
'11'	'00'	0 .. 15	0x00	0x0FF	256	0 .. 15	0x00	0x03FF	1024	0 .. 15	0x00	0x0FFF	4096
		16 .. 31	0x00	0x1FF	512	16 .. 31	0x00	0x07FF	2048	16 .. 31	0x00	0x1FFF	8192
'11'	'01'	0 .. 7	0x00	0x1FF	512	0 .. 7	0x00	0x07FF	2048	0 .. 7	0x00	0x1FFF	8192
		8 .. 15	0x00	0x3FF	1024	8 .. 15	0x00	0x0FFF	4096	8 .. 15	0x00	0x3FFF	16384
'11'	'10'	0 .. 3	0x00	0x3FF	1024	0 .. 3	0x00	0x0FFF	4096	0 .. 3	0x00	0x3FFF	16384
		4 .. 7	0x00	0x7FF	2048	4 .. 7	0x00	0x1FFF	8192	4 .. 7	0x00	0x7FFF	32768
'11'	'11'	0 .. 1	0x00	0x7FF	2048	0 .. 1	0x00	0x1FFF	8192	0 .. 1	0x00	0x7FFF	32768
		2 .. 3	0x00	0xFFF	4096	2 .. 3	0x00	0x3FFF	16384	2 .. 3	0x00	0xFFFF	65536

4.5 FIFO operation

> **Important !**
>
> Without F0IO and C4IO clocks the HDLC controller does not work!

4.5.1 HDLC transmit FIFOs

Data can be transmitted from the host bus interface to the FIFO with write access to registers A_FIFO_DATA0 and A_FIFO_DATA0_NOINC. HFC-E1 converts the data into HDLC code and tranfers it from the FIFO to the E1 or the PCM bus interface.

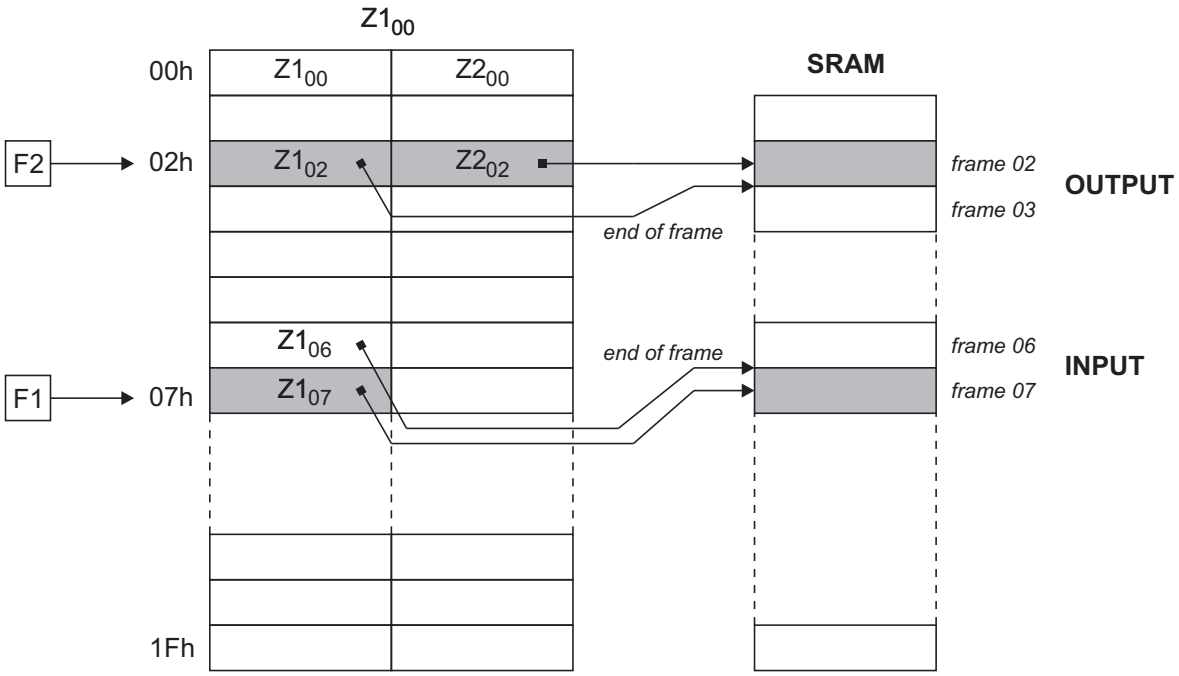


Figure 4.1: *FIFO organization*

HFC-E1 checks $Z1$ and $Z2$. If $Z1 = Z2$ (FIFO empty), HFC-E1 generates a HDLC flag ('01111110') or continuous '1's (depending on bit V_IFF in register A_CON_HDLC) and transmits it to the E1 interface. In this case $Z2$ is not incremented. If also $F1 = F2$ only HDLC flags or continuous '1's are sent to the E1 interface and all counters remain unchanged. If the frame counters are unequal $F2$ is incremented and HFC-E1 tries to transmit the next frame. At the end of a frame ($Z2$ reaches $Z1$) it automatically generates the 16 bit CRC checksum and adds an ending flag. If there is another frame in the FIFO ($F1 \neq F2$) the $F2$ counter is incremented again.

With every byte being written from the host bus side to the FIFO, $Z1$ is incremented automatically. If a complete frame has been sent into the FIFO $F1$ must be incremented to transmit the next frame. If the frame counter $F1$ is incremented the Z-counters may also change because $Z1$ and $Z2$ are functions of $F1$ and $F2$. Thus there are $Z1(F1)$, $Z2(F1)$, $Z1(F2)$ and $Z2(F2)$ (see Fig. 4.1).

$Z1(F1)$ is used for the frame which is just written from the host bus side. $Z2(F2)$ is used for the

frame which is just being transmitted to the PCM or E1 interface side of HFC-E1. $Z1(F2)$ is the end of frame pointer of the current output frame.

In the transmit HFC-channels $F1$ is only incremented from the host interface side if the software driver wants to say "end of transmit frame". This is done by setting bit V_INC_F in register A_INC_RES_FIFO. Then the current value of $Z1$ is stored, $F1$ is incremented and $Z1$ is used as start address of the next frame. $Z2(F2)$ can not be accessed while $Z2(F1)$ can be accessed for transmit FIFOs: If V_FZ_MD in register R_RAM_MISC is set, then $Z2(F2)$ replaces $Z2(F1)$. With this setting the actual filling of the entire RAM space for a FIFO can be calculated.

Please note !

HFC-E1 begins to transmit the bytes from a FIFO at the moment the FIFO is changed (writing R_FIFO) or the $F1$ counter is incremented. Switching to the FIFO that is already selected also starts the transmission. Thus by selecting the same FIFO again transmission can be started.

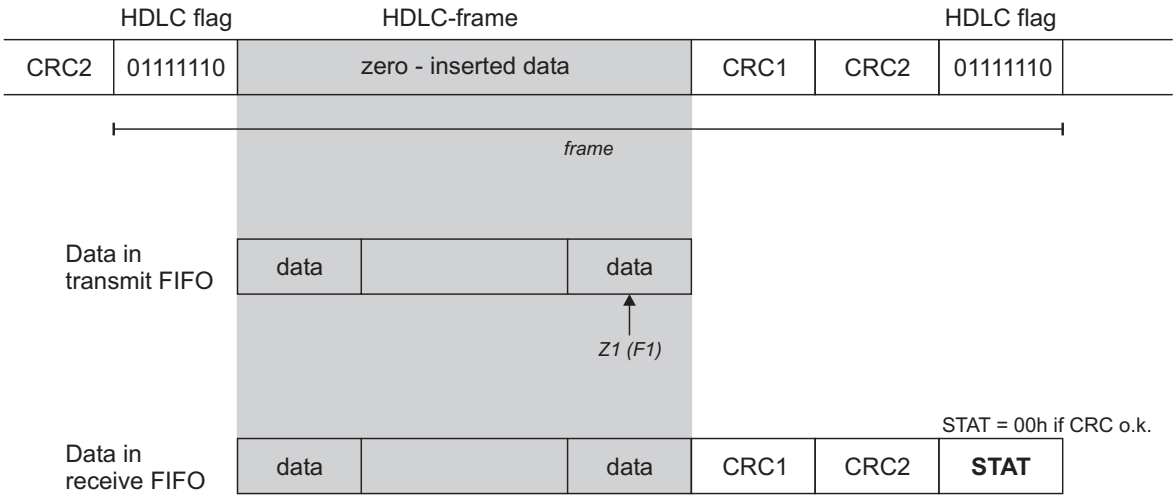


Figure 4.2: *FIFO data organization in HDLC mode*

4.5.2 FIFO full condition in HDLC transmit HFC-channels

Due to the limited number of registers of HFC-E1, the driver software must maintain a list of frame start and end addresses to calculate the actual FIFO size and to check the FIFO full condition. Because there is a maximum of 32 (resp. 16 with 32k RAM) frame counter values and the start address of a frame is the incremented value of the end address of the last frame the memory table needs to have only 32 (resp. 16) values of 16 bit instead of 64 (resp. 32) values.

Remember that an increment of Z-value Z_{MAX} is Z_{MIN} in all FIFOs!

There are two different FIFO full conditions. The first one is met when the FIFO contents comes up to 31 frames (128k or 512k RAM) or 15 frames (32k RAM). There is no possibility for HFC-E1 to manage more frames even if the frames are very small. The second limitation is the overall size of the FIFO. If V_FZ_MD in register R_RAM_MISC is set, the actual FIFO size can be calculated as $Z1(F1) - Z2(F2) + 1$.

4.5.3 HDLC receive FIFOs

The receive HFC-channels receive data from the E1 or PCM bus interface read registers. The data is converted from HDLC into plain data and sent to the FIFO. The data can then be read via the host bus interface.

HFC-E1 checks the HDLC data coming in. If it finds a flag or more than 5 consecutive '1's it does not generate any output data. In this case $Z1$ is not incremented. Proper HDLC data being received is converted by HFC-E1 into plain data. After the ending flag of a frame, HFC-E1 checks the HDLC CRC checksum. If it is correct one byte with all '0's is inserted behind the CRC data in the FIFO named STAT (see Fig. 4.2). This last byte of a frame in the FIFO is different from all '0's if there is no correct CRC field at the end of the frame.

If the STAT value is 0xFF, the HDLC frame ended with at least 8 bits '1's. This is similar to an abort HDLC frame condition.

The ending flag of a HDLC frame can also be the starting flag of the next frame.

After a frame is received completely, $F1$ is incremented by HFC-E1 automatically and the next frame can be received.

After reading a frame via the host bus interface $F2$ has to be incremented. If the frame counter $F2$ is incremented also the Z-counters may change because $Z1$ and $Z2$ are functions of $F1$ and $F2$. Thus there are $Z1(F1)$, $Z2(F1)$, $Z1(F2)$ and $Z2(F2)$ (see Fig. 4.1).

$Z1(F1)$ is used for the frame which is just received from the E1 interface side of HFC-E1. $Z2(F2)$ is used for the frame which is just beeing transmitted to the host bus interface. $Z1(F2)$ is the end of frame pointer of the current output frame.

To calculate the length of the current receive frame the software has to evaluate $Z1 - Z2 + 1$. When $Z2$ reaches $Z1$ the complete frame has been read.

In the receive HFC-channels $F2$ must be incremented from the host interface side after the software detects an end of receive frame ($Z1 = Z2$) and $F1 \neq F2$. Then the current value of $Z2$ is stored, $F2$ is incremented and $Z2$ is copied as start address of the next frame. This is done by setting bit V_INC_F in register A_INC_RES_FIFO. If $Z1 = Z2$ and $F1 = F2$ the FIFO is totally empty. $Z1(F1)$ can not be accessed.

> **Important !**
>
> Before reading a new frame, a change FIFO operation (write access to register R_FIFO) has to be done even if the desired FIFO is already selected. The change FIFO operation is required to update the internal buffer of HFC-E1. Otherwise the first 4 bytes of the FIFO will be taken from the internal buffer and may be invalid.

4.5.4 FIFO full condition in HDLC receive HFC-channels

Because of the E1 time slots not having a hardware based flow control there is no possibility to stop input data if a receive FIFO is full.

Thus there is no FIFO full condition implemented in HFC-E1. HFC-E1 assumes that the FIFOs are deep enough that the host processor's hardware and software is able to avoid any overflow of the

receive FIFOs. Overflow conditions are again more than 31 input frames (resp. 15 frames with 32k RAM) or a memory overflow of the FIFO because of excessive data.

Because HDLC procedures only know a window size of 7 frames no more than 7 frames are sent without software intervention. Due to the great large size of the HFC-E1 FIFOs it is easy to poll HFC-E1 even in large time intervalls without having to fear a FIFO overflow condition.

To avoid any undetected FIFO overflows the software driver should check $F1 - F2$, i.e. the number of frames in the FIFO. If $F1 - F2$ is less than the number in the last reading, an overflow took place if there was no reading of a frame in between.

After a detected FIFO overflow condition this FIFO must be reset by setting the FIFO reset bit V_RES_FIFO in register A_INC_RES_FIFO.

4.5.5 Transparent mode of HFC-E1

It is possible to switch off the HDLC operation for each FIFO independently by bit V_HDLC_TRP in register A_CON_HDLC. If this bit is set, data from the FIFO is sent directly to the E1 or PCM bus interface and data from the E1 or PCM bus interface is sent directly to the FIFO.

Be sure to switch into transparent mode only if $F1 = F2$. Being in transparent mode the F-counters remain unchanged. $Z1$ and $Z2$ are the input and output pointers respectively. Because $F1 = F2$, the Z-counters are always accessable and have valid data for FIFO input and output.

If a transmit FIFO changes to FIFO empty condition no CRC is generated and the last data byte written into the FIFO is repeated until there is new data.

Normally the last byte is undefined because of the Z-counter pointing to a previously unwritten address. To define the last byte, the last write access to the FIFO must be done without Z increment (see register A_FIFO_DATA0_NOINC).

In receive HFC-channels there is no check on flags or correct CRCs and no status byte added.

Unlike in HDLC mode, where byte synchronization is achieved with HDLC flags, the byte boundaries are not arbitrary. The data is just the same as it comes from or is sent to the E1 or PCM bus interface.

Transmit and receive transparent data can be done in two ways. The usual way is transporting FIFO data to the E1 interface with the LSB first as usual in HDLC mode. The second way is transmitting the bytes in reverse bit order as usual for PCM data. So the first bit is the MSB. The bit order can be reversed by setting bit V_REV in register R_FIFO when the FIFO is selected.

> **Important !**
>
> For normal data transmission, register A_SUBCH_CFG must be set to 0x00. To use 56 kbit/s restricted mode for U.S. ISDN lines, register A_SUBCH_CFG must be set to 0x07 for B-channels.

4.5.6 Reading F- and Z-counters

For all asynchronous host accesses to HFC-E1 there is a small chance that a register is changed just in the moment when it is read. Because of slightly different delays of individual bits, it is even possible

that the read value is fully invalid. Therefore we advise to read a F- or Z-counter register until two consecutive readings find the same value.

This is not necessary for a time period of at least 125 µs after writing R_FIFO. It is also not necessary for Z-counters of receive FIFOs if $F1 \neq F2$. Then a whole frame has been received and the counters $Z1(F2)$ and $Z2(F2)$ are stable and valid.

4.5.7 FIFO loop

FIFO data can automatically be transmitted in a loop. This function is normaly used to play a sound signal repeatedly in a PCM time slot. Thus transparent mode should be used.

To set up a FIFO data loop, the following register values have to be implemented:

Register setup:

R_FIFO	: V_FIFO_DIR	= '0'	(select a transmit FIFO)
	: V_FIFO_NUM	= n	(select FIFO [n,TX])
Wait until the FIFO is empty			
A_INC_RES_FIFO	: V_RES_FIFO	= '1'	(reset FIFO)
Wait until not busy (V_BUSY = '0' in register R_STATUS)			
Wait at least for 125 µs			
A_CON_HDLC	: V_HDLC_TRP	= '1'	(select transparent mode)
A_SUBCH_CFG	: V_LOOP_FIFO	= '1'	(enable FIFO loop)
Write data into the FIFO			
Write any value to R_FIFO to start the transmission (change FIFO command)			

An established FIFO loop can easily be stopped by writing V_LOOP_FIFO = '0' into register A_SUBCH_CFG.

4.6 Register description

4.6.1 Write only registers

R_FIRST_FIFO	(w)	0x0B

First FIFO of the FIFO sequence

This register is only used in *FIFO Sequence Mode*, see register R_FIFO_MD for data flow mode selection.

Bits	Reset value	Name	Description
0	0	**V_FIRST_FIFO_DIR**	**Data direction** This bit defines the data direction of the first FIFO in FIFO sequence. '0' = transmit FIFO data '1' = receive FIFO data
5 . . 1	0x00	**V_FIRST_FIFO_NUM**	**FIFO number** This bitmap defines the number of the first FIFO in the FIFO sequence.
7 . . 6		(reserved)	Must be '00'.

R_FIFO_MD	(w)	0x0D

FIFO mode configuration

This register defines the FIFO arrangement and the working mode of the FIFOs and HDLC controllers.

Bits	Reset value	Name	Description
1..0	0	**V_FIFO_MD**	**FIFO mode** This bitmap and V_FIFO_SZ are used to organize the FIFOs in the internal or external SRAM.
3..2	0	**V_DF_MD**	**Data flow mode selection** '00' = *Simple Mode* (SM) '01' = *Channel Select Mode* (CSM) '10' = not used '11' = *FIFO Sequence Mode* (FSM)
5..4	0	**V_FIFO_SZ**	**FIFO size** This bitmap and V_FIFO_MD are used to organize the FIFOs in the internal or external SRAM. The actual FIFO sizes also depend on the used SRAM size (see R_RAM_MISC).
7..6		(reserved)	Must be '00'.

(See Table 4.3 on page 159 for suitable V_FIFO_MD and V_FIFO_SZ values.)

A_INC_RES_FIFO [FIFO] **(w)** **0x0E**

Increment and reset FIFO register

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
0		**V_INC_F**	**Increment the *F*-counters of the selected FIFO** '0' = no increment '1' = increment This bit is automatically cleared after the counter increment has been processed.
1		**V_RES_FIFO**	**FIFO reset** '0' = no reset '1' = reset selected FIFO (*F*- and *Z*-counters and channel mask A_CH_MSK are reset) This bit is automatically cleared after the FIFO reset has been processed.
2		**V_RES_LOST**	**LOST error bit reset** '0' = no reset '1' = reset LOST This bit is automatically cleared with the LOST error bit reset.
7 . . 3		(reserved)	Must be '00000'.

R_FIFO	(w)	0x0F

FIFO selection register

This register is used to select a FIFO. Before a FIFO array register can be accessed, this index register must specify the desired FIFO number and data direction.

Note: This register is a multi-register. It is selected with bitmap V_DF_MD less than '11' in register R_FIFO_MD (SM and CSM selected).

Bits	Reset value	Name	Description
0	0	**V_FIFO_DIR**	**FIFO data direction** '0' = transmit FIFO data '1' = receive FIFO data
5..1	0x00	**V_FIFO_NUM**	**FIFO number**
6		(reserved)	Must be '0'.
7	0	**V_REV**	**Bit order** '0' = LSB first '1' = MSB first LSB first is used in HDLC mode while MSB first is used in transparent mode. The bit order is being reversed for the data stored into the FIFO or when the data is read from the FIFO.

R_FSM_IDX	(w)	0x0F

Index register of the FIFO sequence

This register is used to select a list number in *FIFO Sequence Mode*. Some FIFO array registers are indexed by this list number. Before these registers can be accessed, this index register must specify the desired list number.

Note: This register is a multi-register. It is selected with bitmap V_DF_MD = '11' in register R_FIFO_MD. In FSM only few FIFO array registers are indexed by this multi-register, but most FIFO array registers remain indexed by R_FIFO.

Bits	Reset value	Name	Description
5..0	0	**V_IDX**	**List index** The list index must be in the range 0..63.
7..6		(reserved)	Must be '00'.

A_FIFO_DATA0_NOINC [FIFO] **(w)** **0x84**

FIFO data register

This address can also be accessed to write word and double word width which accesses two or four data bytes (see registers A_FIFO_DATA1_NOINC and A_FIFO_DATA2_NOINC).

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		**V_FIFO_DATA0_NOINC**	**Data byte** Write one byte to the FIFO selected with register R_FIFO without incrementing *Z*-counter.

(This register can be used to store the last FIFO byte in transparent transmit mode. Then this byte is repeately transmitted automatically.)

A_FIFO_DATA1_NOINC [FIFO] **(w)** **0x84**

FIFO data register

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
15..0		**V_FIFO_DATA1_NOINC**	**Data word** Write one word to the FIFO selected with register R_FIFO without incrementing *Z*-counter.

A_FIFO_DATA2_NOINC [FIFO] **(w)** **0x84**

FIFO data register

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
31..0		**V_FIFO_DATA2_NOINC**	**Data double word** Write two words to the FIFO selected with register R_FIFO without incrementing *Z*-counter.

A_CH_MSK [FIFO] **(w)** **0xF4**

HFC-channel data mask for the selected transmit HFC-channel

For receive FIFOs this register is ignored.

Before writing this array register the HFC-channel must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0	0xFF	**V_CH_MSK**	**Mask byte** This bitmap defines bit values for not processed bits of a HFC-channel. All not processed bits of a HFC-channel are set to the value defined in this register. This register has only a meaning when V_BIT_CNT $\neq 0$ in register A_SUBCH_CFG.

A_CON_HDLC [FIFO] **(w)** **0xFA**

HDLC and connection settings of the selected FIFO

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
0	0	**V_IFF**	**Inter frame fill** '0' = write HDLC flags 0x7E as inter frame fill '1' = write all '1's as inter frame fill
1	0	**V_HDLC_TRP**	**HDLC mode / transparent mode selection** '0' = HDLC mode '1' = transparent mode **Note:** For a D-channel this bit must be '0'.
4..2	0	**V_TRP_IRQ**	**Transparent mode interrupt selection** This bitmap has a different meaning in HDLC and transparent mode. **Transparent mode:** The FIFO is enabled and an interrupt is generated all 2^n bytes when the bits [n-1:0] of the $Z1$-counter (in transmit direction) or the $Z2$-counter (in receive direction) change from all '1's to all '0's. $n =$ V_TRP_IRQ $+ 5$. 0 = FIFO enabled, no interrupt 1 = all $2^6 = 64$ bytes an interrupt is generated 2 = all $2^7 = 128$ bytes an interrupt is generated 3 = all $2^8 = 256$ bytes an interrupt is generated 4 = all $2^9 = 512$ bytes an interrupt is generated 5 = all $2^{10} = 1024$ bytes an interrupt is generated 6 = all $2^{11} = 2048$ bytes an interrupt is generated 7 = all $2^{12} = 4096$ bytes an interrupt is generated **Notes:** (1) No interrupt occurs, if the Z-counters do never reach the selected value. This depends on the Z_{MAX} setting. (2) The first interrupt after $Z = Z_{min}$ comes after $2^n - Z_{min}$ bytes if $2^n - 1 > Z_{min}$. Only the following interrupts come after 2^n bytes until Z reaches Z_{min} again. **HDLC mode:** The FIFO is enabled with any value $\neq 0$. A FIFO interrupt can be generated at end of frame. 0 = FIFO disabled, no interrupt 1..7 = FIFO enabled, interrupt enabled **Note:** A FIFO must be enabled even for connections between line interface and PCM interface. No data transmission is performed with disabled FIFO.

(continued on next page)

(continued from previous page)

Bits	Reset value	Name	Description
7..5	0	V_DATA_FLOW	**Data flow configuration** In transmit operation (V_FIFO_DIR = '0' in register R_FIFO): '000', '001' = FIFO → E1, FIFO → PCM '010', '011' = FIFO → PCM '100', '101' = FIFO → E1, E1 → PCM '110', '111' = E1 → PCM In receive operation (V_FIFO_DIR = '1' in register R_FIFO): '000', '100' = FIFO ← E1 '001', '101' = FIFO ← PCM '010', '110' = FIFO ← E1, E1 ← PCM '011', '111' = FIFO ← PCM, E1 ← PCM

(For details on bitmap V_DATA_FLOW see Fig. 3.4 and 3.5 on page 110.)

Important !

A FIFO is disabled if V_HDLC_TRP + V_TRP_IRQ = 0 in register A_CON_HDLC[FIFO]. This setting is useful to reduce RAM accesses if a FIFO is not used at all.

If HFC-channel data is routed through the switches of the flow controller (Fig. 3.4 and 3.5) the FIFO must be enabled. That applies to all connections except the PCM-to-PCM data transmission.

A_SUBCH_CFG [FIFO] **(w)** **0xFB**

Subchannel parameters for bit processing of the selected FIFO

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
2..0	0	**V_BIT_CNT**	**Number of bits to be processed in the HFC-channel byte** In HDLC mode, only this number of bits is read from or written into the FIFO. In transparent mode always a whole FIFO byte is read or written, but only V_BIT_CNT bits contain valid data. '000' = process 8 bits (64 kbit/s) '001' = process 1 bit (8 kbit/s) '010' = process 2 bits (16 kbit/s) '011' = process 3 bits (24 kbit/s) '100' = process 4 bits (32 kbit/s) '101' = process 5 bits (40 kbit/s) '110' = process 6 bits (48 kbit/s) '111' = process 7 bits (56 kbit/s)
5..3	0	**V_START_BIT**	**Start bit in the HFC-channel byte** This bitmap specifies the position of the bit field in the HFC-channel byte. The bit field is located at position V_START_BIT in the HFC-channel byte. V_BIT_CNT + V_START_BIT must not be greater than 7 to get the bit field completely inside the HFC-channel byte. Any value greater than 7 produces an undefined behavior of the subchannel processor.
6	0	**V_LOOP_FIFO**	**FIFO loop** '0' = normal operation '1' = repeat current FIFO data (useful only in transparent mode) **Note:** This bit is only used for transmit FIFOs and is ignored for receive FIFOs.
7	0	**V_INV_DATA**	**Inverted data** '0' = normal data transmission '1' = inverted data transmission

A_CHANNEL [FIFO] **(w)** **0xFC**

HFC-channel assignment for the selected FIFO

This register is only used in *Channel Select Mode* and *FIFO Sequence Mode*.

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
0		**V_CH_FDIR**	**HFC-channel data direction** '0' = HFC-channel for transmit data '1' = HFC-channel for receive data **Reset value:** This bitmap is reset to the same value as the current FIFO, i.e. V_CH_FDIR of A_CHANNEL[*number*,*direction*] = *direction*.
5..1		**V_CH_FNUM**	**HFC-channel number** (0..31) **Reset value:** This bitmap is reset to the same value as the current FIFO, i.e. V_CH_FNUM of A_CHANNEL[*number*,*direction*] = *number*.
7..6	0	(reserved)	Must be '00'.

A_FIFO_SEQ [FIFO] **(w)** **0xFD**

FIFO sequence list

This register is only used in *FIFO Sequence Mode*.

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
0		**V_NEXT_FIFO_DIR**	**FIFO data direction** This bit defines the data direction of the next FIFO in FIFO sequence. '0' = transmit FIFO data '1' = receive FIFO data **Reset value:** This bitmap is reset to the same value as the current FIFO, i.e. V_NEXT_FIFO_DIR of A_FIFO_SEQ[*number*,*direction*] = *direction*.
5 . . 1		**V_NEXT_FIFO_NUM**	**FIFO number** This bitmap defines the FIFO number of the next FIFO in the FIFO sequence. **Reset value:** This bitmap is reset to the same value as the current FIFO, i.e. V_NEXT_FIFO_NUM of A_FIFO_SEQ[*number*,*direction*] = *number*.
6	0	**V_SEQ_END**	**End of FIFO list** '0' = FIFO list goes on '1' = FIFO list is terminated after this FIFO (V_NEXT_FIFO_DIR and V_NEXT_FIFO_NUM are ignored)
7	0	(reserved)	Must be '0'.

4.6.2 Read only registers

A_Z1L [FIFO]	(r)	0x04

FIFO input counter *Z*1, low byte access

This address can also be accessed with word and double word width to read the complete *Z*1-counter or *Z*1- and *Z*2-counters together (see registers A_Z1 and A_Z12).

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		V_Z1L	Bits [7..0] counter value of *Z*1

(See Table 4.3 for reset value.)

A_Z1 [FIFO]	(r)	0x04

FIFO input counter *Z*1, word access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
15..0		V_Z1	Bits [15..0] counter value of *Z*1

(See Table 4.3 for reset value.)

A_Z12 [FIFO]	(r)	0x04

FIFO input counters *Z*1 and *Z*2, double word access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
31..0		V_Z12	Bits [15..0] are counter value of *Z*1 and bits [31..16] are counter value of *Z*2

(See Table 4.3 for reset value.)

A_Z1H [FIFO]	(r)	0x05

FIFO input counter *Z*1, high byte access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7 . . 0		V_Z1H	**Bits [15..8] counter value of *Z*1**

(See Table 4.3 for reset value.)

A_Z2L [FIFO]	(r)	0x06

FIFO output counter *Z*2, low byte access

This address can also be accessed with word width to read the complete *Z*2-counter (see register A_Z2).

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7 . . 0		V_Z2L	**Bits [7..0] counter value of *Z*2**

(See Table 4.3 for reset value.)

A_Z2 [FIFO]	(r)	0x06

FIFO output counter *Z*2, word access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
15 . . 0		V_Z2	**Bits [15..0] counter value of *Z*2**

(See Table 4.3 for reset value.)

A_Z2H [FIFO]	**(r)**	**0x07**

FIFO output counter *Z*2, high byte access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		**V_Z2H**	**Bits [15..8] counter value of *Z*2**

(See Table 4.3 for reset value.)

A_F1 [FIFO]	**(r)**	**0x0C**

FIFO input HDLC frame counter *F*1, byte access

This address can also be accessed with word width to read the *F*1- and *F*2-counters together (see register A_F12).

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		**V_F1**	**Counter value** Up to 31 HDLC frames (resp. 15 with 32k RAM) can be stored in every FIFO.

(See Table 4.3 for reset value.)

A_F12 [FIFO]	**(r)**	**0x0C**

FIFO input HDLC frame counters *F*1 and *F*2, word access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
15..0		**V_F12**	**Bits [7..0] are counter value of *F*1 and bits [15..8] are counter value of *F*2** Up to 31 HDLC frames (resp. 15 with 32k RAM) can be stored in every FIFO.

(See Table 4.3 for reset value.)

A_F2 [FIFO]	(r)	0x0D

FIFO output HDLC frame counter *F2*, byte access

Before reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		V_F2	**Counter value** Up to 31 HDLC frames (resp. 15 with 32k RAM) can be stored in every FIFO.

(See Table 4.3 for reset value.)

R_INT_DATA	(r)	0x88

Internal data register

This register can be read to access data with short read signal.

Bits	Reset value	Name	Description
7..0		V_INT_DATA	**Internal data buffer**

See ‘Short read method’ on page 74.

4.6.3 Read / write registers

A_FIFO_DATA0 [FIFO]	(r/w)	0x80

FIFO data register

This address can also be accessed with word and double word width to access two or four data bytes (see registers A_FIFO_DATA1 and A_FIFO_DATA2).

Before writing or reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
7..0		**V_FIFO_DATA0**	**Data byte** Read / write one byte from / to the FIFO selected with register R_FIFO and increment *Z*-counter by 1.

A_FIFO_DATA1 [FIFO]	(r/w)	0x80

FIFO data register

Before writing or reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
15..0		**V_FIFO_DATA1**	**Data word** Read / write one word from / to the FIFO selected with register R_FIFO and increment *Z*-counter by 2.

A_FIFO_DATA2 [FIFO]	(r/w)	0x80

FIFO data register

Before writing or reading this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
31..0		**V_FIFO_DATA2**	**Data double word** Read / write two words from / to the FIFO selected with register R_FIFO and increment *Z*-counter by 4.

Chapter 5

E1 interface

Table 5.1: *Overview of the E1 interface pins*

Number	Name	Description
184	T_B	E1 interface transmit data B
185	T_A	E1 interface transmit data A
187	ADJ_LEV	E1 interface level generator
188	R_B	E1 interface receive input B
189	LEV_B	E1 interface level detect B
190	LEV_A	E1 interface level detect A
191	R_A	E1 interface receive input A

Table 5.2: *Overview of the E1 interface registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x20	R_E1_WR_STA	202	0x20	R_E1_RD_STA	218
0x22	R_LOS0	202	0x24	R_SYNC_STA	219
0x23	R_LOS1	203	0x25	R_RX_SL0_0	220
0x24	R_RX0	204	0x26	R_RX_SL0_1	221
0x25	R_RX_SL0_CFG0	205	0x27	R_RX_SL0_2	222
0x26	R_RX_SL0_CFG1	207	0x2B	R_JATT_STA	222
0x28	R_TX0	208	0x2C	R_SLIP	223
0x29	R_TX1	209	0x30	R_FAS_ECL	223
0x2C	R_TX_SL0_CFG0	210	0x31	R_FAS_ECH	224
0x2D	R_TX_SL0	211	0x32	R_VIO_ECL	224
0x2E	R_TX_SL0_CFG1	212	0x33	R_VIO_ECH	225
0x2F	R_JATT_CFG	213	0x34	R_CRC_ECL	225
0x30	R_RX_OFFS	214	0x35	R_CRC_ECH	225
0x31	R_SYNC_OUT	215	0x36	R_E_ECL	226
0x34	R_TX_OFFS	216	0x37	R_E_ECH	226
0x35	R_SYNC_CTRL	217	0x38	R_SA6_VAL13_ECL	226
			0x39	R_SA6_VAL13_ECH	227
			0x3A	R_SA6_VAL23_ECL	227
			0x3B	R_SA6_VAL23_ECH	228

5.1 Overview

The E1 interface module consists of the receive and transmit data pathes with a clock processing unit each, the clock distribution unit and the E1 state machine. The overall connection of these units is shown in Figure 5.1.

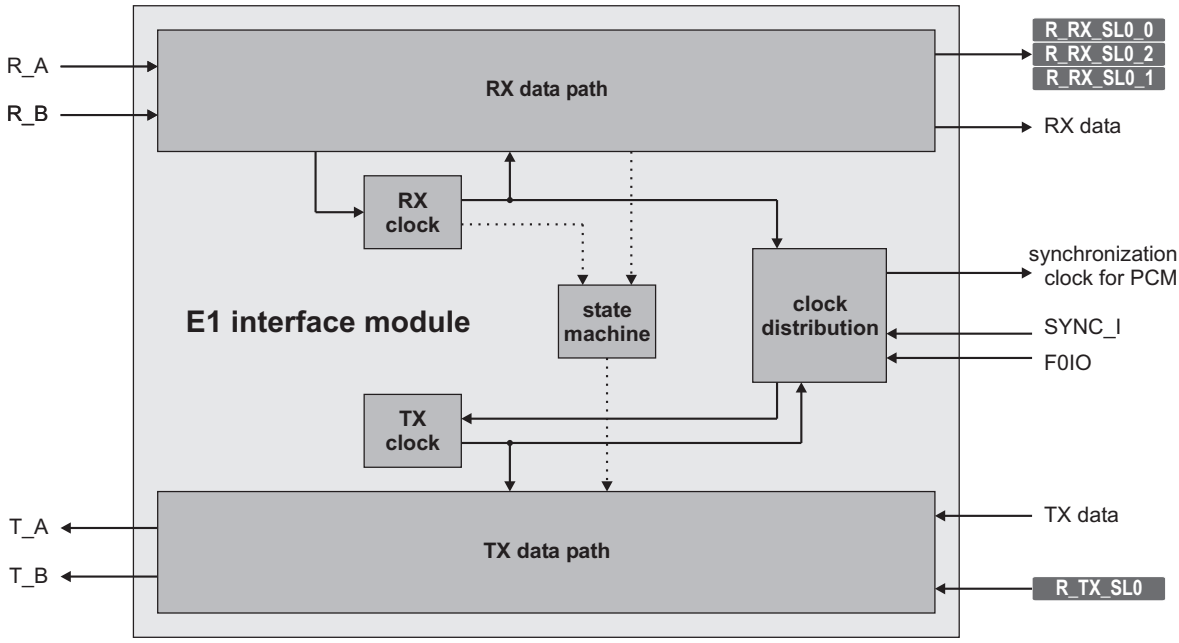


Figure 5.1: *Overview of the E1 interface module*

Received data from pins R_A and R_B is passed through the RX data path. A bit clock and a frame clock are derived from the received data. These clocks are used to synchronize the RX data path timing to the incoming data stream. The frame clock can be passed for synchronization purposes to the TX data path and the PCM timing control as well.

The transmit data clock can be generated from several clocks which are obtained from the clock distribution unit. A jitter attenuator inside the TX clock unit ensures jitter and wander reduction. This TX clock is passed back to the clock distribution unit.

The state machine takes several signals from the RX data path and the RX clock unit. The TX data path is controlled by the state machine's output signal.

A detailed block diagram of the E1 interface module is shown in Figure 5.2.

5.2 Frame structure

Every frame of the E1 interface consists of 256 bits which are organized in 32 channels with 8 bytes each. E1-channels are also called *E1 time slots* and are numbered 0 . . 31.

Time slot 0 is used for synchronization purposes. Its structure is described in Section 5.3.

When ISDN D-channel is used for E1 interface operation, time slot 16 is normally occupied for this task.

All other E1 channels 1 . . 15 and 17 . . 31 can be used for ISDN B-channels.

5.3 Time slot 0 frame configuration

Time slot 0 can operate either in normal doubleframe mode or in multiframe mode according to ITU G.704. The modes can be selected independently from each other in transmit and receive direction.

For both modes there are three ways of data processing:

- Semiautomatic mode where E1 interface time slot 0 data is partly handled automatically. Manually data handling uses register accesses to write and read time slot 0 data.
- Transparent mode where all E1 interface time slot 0 data bits are handled by the host processor. HFC-channel 0 is used to transmit and receive time slot 0 data.
- Transparent mode where all E1 interface time slot 0 data bits are handled by the host processor. Transmitted and received time slot 0 data are processed with memory window accesses.

Additionally, in semiautomatic mode some bits can be specified to be processed in transparent mode. Details of semiautomatic and transparent mode handling are explained in section 5.4 from page 188.

5.3.1 Normal doubleframe mode

Normal doubleframe mode is selected with V_TX_MF = '0' in register R_TX_SL0_CFG1 for transmit direction and with V_RX_MF = '0' in register R_RX_SL0_CFG1 for receive direction. Transmit and receive data direction can have different modes.

Time slot 0 has alternating data bytes in this mode. Table 5.3 shows its structure.

Table 5.3: *Allocation of bits 1 to 8 of the frame (Time slot 0)*

Alternating frames	Bit number							
	1	2	3	4	5	6	7	8
Frame containing the FAS	S_i (FAS)	0	0	1	1	0	1	1
		Frame alignment signal (FAS)						
Frame not containing the FAS	S_i (NFAS)	1	A	S_{a4}	S_{a5}	S_{a6}	S_{a7}	S_{a8}

The spare bits $S_i(FAS)$ and $S_i(NFAS)$ are reserved for international use. If not used, they should be fixed to '1'.

The frame alignment signal (FAS) has the value '0011011' and it is used for synchronization purposes.

A constant '1' at bit number 2 in the NFAS byte is used as a qualifier bit for the two doubleframe bytes.

The remote alarm indication (RAI) bit A indicates a loss of the frame alignment condition. It has the value A = '0' in undisturbed operation and it indicates an alarm condition with A = '1'.

Additional spare bits $S_{a4}..S_{a8}$ are defined for national use. They should be set to '1' if they are not used.

5.3.2 CRC-4 multiframe mode

CRC-4 multiframe mode is selected with V_TX_MF = '1' in register R_TX_SL0_CFG1 for transmit direction and with V_RX_MF = '1' in register R_RX_SL0_CFG1 for receive direction. Transmit and receive data direction can have different modes. A complete multiframe structure consists of 16 bytes as shown in Table 5.4.

Table 5.4: *CRC-4 multiframe structure in time slot 0*

Sub-multiframe	Frame number		Bit number							
(SMF)	in SMF	in MF	1	2	3	4	5	6	7	8
I	0	0	C_1	0	0	1	1	0	1	1
	1	1	0	1	A	S_{a4}	S_{a5}	S_{a61}	S_{a7}	S_{a8}
	2	2	C_2	0	0	1	1	0	1	1
	3	3	0	1	A	S_{a4}	S_{a5}	S_{a62}	S_{a7}	S_{a8}
	4	4	C_3	0	0	1	1	0	1	1
	5	5	1	1	A	S_{a4}	S_{a5}	S_{a63}	S_{a7}	S_{a8}
	6	6	C_4	0	0	1	1	0	1	1
	7	7	0	1	A	S_{a4}	S_{a5}	S_{a64}	S_{a7}	S_{a8}
II	0	8	C_1	0	0	1	1	0	1	1
	1	9	1	1	A	S_{a4}	S_{a5}	S_{a61}	S_{a7}	S_{a8}
	2	10	C_2	0	0	1	1	0	1	1
	3	11	1	1	A	S_{a4}	S_{a5}	S_{a62}	S_{a7}	S_{a8}
	4	12	C_3	0	0	1	1	0	1	1
	5	13	E_1	1	A	S_{a4}	S_{a5}	S_{a63}	S_{a7}	S_{a8}
	6	14	C_4	0	0	1	1	0	1	1
	7	15	E_2	1	A	S_{a4}	S_{a5}	S_{a64}	S_{a7}	S_{a8}

The cyclic redundancy check bits (CRC-4) $C_1..C_4$ are transmitted at bit number 1 of every even frame number. The odd frames contain the CRC-4 multiframe alignment signal '001011' at bit number 1.

Two CRC-4 error indication bits E_1, E_2 can be used to indicate remote CRC-4 errors. V_TX_E must be set to '1' in register R_TX_SL0_CFG1 for this function. An error received in sub-multiframe I is indicated with $E_1 = 0$, while an error received in sub-multiframe II is indicated with $E_2 = 0$. Correct CRC-4 results have $E_{1,2} = 1$. If the E-bits are not used, they should be fixed to '1'. An inverted E-bit functionallity can be achieved with V_INV_E = '1' in register R_TX_SL0_CFG1.

The E-bits can be used for another purpose instead of CRC-4 error indication with V_TX_E = '0'. In this case, they are called XS_{13} and XS_{15} bits due to their frame number in the multiframe. The XS-bits get their value from register R_TX_SL0_CFG1, namely XS_{13} = V_XS13 and XS_{15} = V_XS15.

The remote alarm indication bit A in transmit direction is set to '1' after a loss of the frame alignment condition is detected. It has the value A = '0' in undisturbed operation.

Additional spare bits $S_{a4}..S_{a8}$ are defined for national use. Except of S_{a6} these spare bits have the same value in every frame of the multiframe structure. S_{a6} forms a sequence $S_{a61}..S_{a64}$ in every sub-multiframe. This sequence can be used for error indication. A valid S_{a6} sequence must occur three times in a row. When this happens, V_SA6_OK is set to '1' in register R_RX_SL0_1. V_SA6_CHG in the same register can be read to take note of a changed S_{a6} pattern.

Sixteen different S_{a6} patterns are possible. Three of them are counted by two 16 bit counters. Every received pattern '0001' or '0011' increments the counter R_SA6_VAL13_ECL

$$EC(S_{a6}(1,3)) = 256 \cdot R\_SA6\_VAL13\_ECL + R\_SA6\_VAL13\_ECL$$

and every received pattern '0010' or '0011' increments the counter R_SA6_VAL23_ECL

$$EC(S_{a6}(2,3)) = 256 \cdot R\_SA6\_VAL23\_ECL + R\_SA6\_VAL23\_ECL \ .$$

Both counters roll over with an increment on the value 65535. The counter behaviour can be configured with bit V_AUTO_ERR_RES in register R_TX1 [1].

With V_AUTO_ERR_RES = '0' the counters are reset to zero with any read access to the counter's high byte. So the low byte must be read first (this buffers the high byte). Please note, that the counters do never stop counting in this mode. After reaching the maximum value the counters restart with zero again.

In practice, the error rate *errors/second* is most interesting. This mode can be selected with V_AUTO_ERR_RES = '1'. An automatic counter reset is initiated every second after the value has been buffered. Any read access to the error counter register has no influenced to the counter value.

All frames with an even number contain the FAS '0011011' at bit number 2..8. This is used for synchronization purposes. Bit number 2 of all odd numbered frames has constant '0' to distinguish from frames with FAS.

[1] Please note, that V_AUTO_ERR_RES configures the behaviour of all six error counters.

5.4 Frame access in semiautomatic and transparent modes

5.4.1 Semiautomatic mode

The semiautomatic mode is selected with V_TRP_SL0 = '0' in register R_TX_SL0_CFG1. Additionally, R_TX_SL0_CFG0 must be 0x00.

S_i **bits:** There is one $S_i(FAS)$ and one $S_i(NFAS)$ bit per doubleframe. These bits must be written into V_TX_FAS and V_TX_NFAS of register R_TX_SL0 for transmit direction. S_i bits are not used in multiframe mode.

In receive direction, these bits can be read from V_SI_FAS and V_SI_NFAS in register R_RX_SL0_0.

A **bit:** The remote alarm indication bit A is used in both doubleframe and multiframe data structures. In semiautomatic mode it is generated from the state machine.

The received remote alarm indication bit A can be read from V_A in register R_RX_SL0_0.

$S_{a4}..S_{a8}$ **bits:** In transmit direction, the spare bits $S_{a4}..S_{a8}$ must be written into V_TX_SA4 .. V_TX_SA8 of register R_TX_SL0. This is usable in doubleframe mode. But in multiframe mode this setting specifies

$$S_{a61} = S_{a62} = S_{a63} = S_{a64} = S_{a6}$$

which is not the typical application case. Different multiframe values for $S_{a61}..S_{a64}$ can be written in mixed semiautomatic and transparent mode as it is explained in section 5.4.3 on page 190.

The received spare bits $S_{a4}..S_{a8}$ can be read from V_SA4 .. V_SA8 in register R_RX_SL0_2. As there are four S_{a6} bits ($S_{a61}..S_{a64}$) in every multiframe, V_SA6 contains a valid value for S_{a6} only in doubleframe mode. In multiframe mode $S_{a61}..S_{a64}$ can be read from V_SA61 .. V_SA64 in register R_RX_SL0_1.

$C_1..C_4$ **bits:** CRC-4 bits $C_1..C_4$ are generated automatically for E1 interface time slot 0. Received CRC-4 bits are interpreted by HFC-E1 automatically. The CRC result can be read from bit V_CRC_OK in register R_RX_SL0_0.

$E_{1,2}$ **bits:** CRC-4 error indication bits E_1 and E_2 are generated automatically for E1 interface time slot 0. Transmitted E-bits can be read back from V_TX_E1 and V_TX_E2 in register R_RX_SL0_0.

Received CRC-4 error indication bits influence the HFC-E1 state machine. They can be read from V_RX_E1 and V_RX_E2 in register R_RX_SL0_0.

R_RX_SL0_0 and R_RX_SL0_2 are updated once every 250 µs while R_RX_SL0_1 is updated once every 2 ms. Register R_TX_SL0 is processed once every 250 µs as well.

5.4.2 Transparent mode

The doubleframe and multiframe data can be processed in transparent mode. In this case, V_TRP_SL0 = '1' must be written into register R_TX_SL0_CFG1. All bits are handled by the host processor in transmit direction.

Two different methods of data handling are selectable in transparent mode. Time slot 0 data can either be processed via HFC-channel 0 or memory window access (see below). This can be selected for each data direction separately.

Registers R_TX_SL0_CFG0 and R_TX_SL0 are ignored in transparent mode. Received doubleframe and multiframe data is stored in the registers R_RX_SL0_0, R_RX_SL0_1 and R_RX_SL0_2; so it can be accessed via register access as well as via HFC-channel 0 or memory window access.

5.4.2.1 HFC-channel 0

Transmitted time slot 0 data is processed through HFC-channel [0,TX] with V_TRP_SL0 = '1' and V_TX_SL0_RAM = '0' in register R_TX_SL0_CFG1. A transmit FIFO has to be assigned to the HFC-channel [0,TX] and must be enabled in transparent mode to establish the data connection between the host processor and the E1 interface.

Received time slot 0 data is processed through HFC-channel [0,RX] with V_TRP_SL0 = '1' in register R_TX_SL0_CFG1 and V_RX_SL0_RAM = '0' in register R_RX_SL0_CFG1. A receive FIFO has to be assigned to the HFC-channel [0,RX] and must be enabled in transparent mode to establish the data connection between the host processor and the E1 interface.

The whole doubleframe structure (S_i bits, A bit and $S_{a4}..S_{a8}$ bits) or multiframe structure ($C_1..C_4$ bits, E_1 and E_2 bits, A bit, $S_{a4}..S_{a8}$ and $S_{a61}..S_{a64}$ bits) must be handled by the host processor in both receive and transmit directions.

5.4.2.2 Memory window access

The memory window access is selected with V_TRP_SL0 = '1' and V_TX_SL0_RAM = '1' in register R_TX_SL0_CFG1 for transmitted time slot 0 data. The whole doubleframe or multiframe must be written into the memory at RAM addresses 'RRRR 1001 01TX XXXD'.

Received time slot 0 data is processed with memory window access if V_TRP_SL0 = '1' in register R_TX_SL0_CFG1 and V_RX_SL0_RAM = '1' in register R_RX_SL0_CFG1. The whole multiframe can be read from the memory at RAM addresses 'RRRR 1001 01TX XXXD'.

'D' is the direction flag which is '0' for transmit time slot 0 and it is '1' for receive time slot 0.

'XXXX' is the 16 byte address range for the doubleframe and multiframe data.

The RAM address range dependes on the used SRAM. 'RRRR' has the value '0001' if the internal 32 KByte SRAM is used. With external 128 KByte or 512 KByte SRAM, 'RRRR' = '0010' must be set.

The RAM area is organized as an alternating buffer. So one half can be read or written by the host processor while the other half sends or receives data via the E1 interface. V_ALT_FR_TX and V_ALT_FR_RX show the value of 'T' for the memory area which is currently in transmit and receive process. The alternate memory area can be written and read during this period.

Valid memory addresses for transmit time slot 0 data are

$$\text{address}_{\text{TX}} = \begin{cases} \text{0x1940} + \text{0x20} \cdot \overline{\text{V\_ALT\_FR\_TX}} + 2i & : \quad \text{internal SRAM, } i = 0..15 \\ \text{0x2940} + \text{0x20} \cdot \overline{\text{V\_ALT\_FR\_TX}} + 2i & : \quad \text{external SRAM, } i = 0..15 \end{cases}$$

while

$$\text{address}_{\text{RX}} = \begin{cases} \text{0x1940} + \text{0x20} \cdot \overline{\text{V\_ALT\_FR\_RX}} + 2i + 1 & : \text{ internal SRAM, } i = 0..15 \\ \text{0x2940} + \text{0x20} \cdot \overline{\text{V\_ALT\_FR\_RX}} + 2i + 1 & : \text{ external SRAM, } i = 0..15 \end{cases}$$

is valid for receive time slot 0 data. It is recommended to check V_ALT_FR_TX and V_ALT_FR_RX afterwards to ensure that the alternating buffer has not switched meanwhile [2].

V_ALT_FR_TX and V_ALT_FR_RX are not synchronized to each other and change every 2 ms.

Memory window access is very similar for both doubleframe and multiframe modes. The alternating memory area has always 2×16 byte. Thus eight doubleframes are processed before the alternating buffer changes the 'T' bit of the memory address. The alternating memory has the same size as a complete multiframe. For this reason, the 'T' bit changes its value after every multiframe.

5.4.3 Mixed semiautomatic and transparent mode

When the doubleframe or multiframe data is processed in semiautomatic mode, some time slot 0 data bits can be processed in transparent mode via HFC-channel 0 or memory window access.

Semiautomatic mode must be selected with V_TRP_SL0 = '0' in register R_TX_SL0_CFG1. With every bit which is set in register R_TX_SL0_CFG0, the corresponding doubleframe or multiframe bit is handled in transparent mode. The desired data path – HFC-channel 0 or memory window access – is selected in the same way as described above.

As an example, the following procedure describes how to handle multiframe transmit data in semiautomatic mode, except spare bits $S_{a61}..S_{a64}$ which are written into memory. As $S_{a61}..S_{a64}$ are constant over the whole data connection, they have to be programmed only once.

- V_TX_MF = '1'; select multiframe processing
- V_TRP_SL0 = '0'; select semiautomatic mode
- V_TX_SL0_RAM = '1'; transparent time slot 0 bits come from alternating RAM buffer
- V_TRP_FAS = '0'; S_i(FAS) bit in semiautomatic mode
- V_TRP_NFAS = '0'; S_i(NFAS) bit in semiautomatic mode
- V_TRP_RAL = '0'; remote alarm bit A will be generated from the state machine
- V_TRP_SA4 = '00100'; S_{a4}, S_{a5}, S_{a7} and S_{a8} are taken from V_TX_SA4, S_{a6} is processed in transparent mode
- Write $[S_{a64}..S_{a61}] = [0011]$, e.g., with memory window access:

  R_RAM_ADDR2 = 0x00

  R_RAM_ADDR1 = 0x19 if the internal 32 KByte SRAM is used, or
  R_RAM_ADDR1 = 0x29 if the external 128 KByte or 256 KByte SRAM is used.

  R_RAM_ADDR0 = 0x40 + 0x02; R_RAM_DATA = 0x04; S_{a61} of SMF I in 1st altern. buffer
  R_RAM_ADDR0 = 0x60 + 0x02; R_RAM_DATA = 0x04; S_{a61} of SMF I in 2nd altern. buffer

  R_RAM_ADDR0 = 0x40 + 0x06; R_RAM_DATA = 0x04; S_{a62} of SMF I in 1st altern. buffer
  R_RAM_ADDR0 = 0x60 + 0x06; R_RAM_DATA = 0x04; S_{a62} of SMF I in 2nd altern. buffer

[2] These bits can also be used in semiautomatic mode to check if the next time slot 0 has been processed.

R_RAM_ADDR0 = 0x40 + 0x0A; R_RAM_DATA = 0x00; S_{a63} of SMF I in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x0A; R_RAM_DATA = 0x00; S_{a63} of SMF I in 2nd altern. buffer

R_RAM_ADDR0 = 0x40 + 0x0E; R_RAM_DATA = 0x00; S_{a64} of SMF I in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x0E; R_RAM_DATA = 0x00; S_{a64} of SMF I in 2nd altern. buffer

R_RAM_ADDR0 = 0x40 + 0x12; R_RAM_DATA = 0x04; S_{a61} of SMF II in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x12; R_RAM_DATA = 0x04; S_{a61} of SMF II in 2nd altern. buffer

R_RAM_ADDR0 = 0x40 + 0x16; R_RAM_DATA = 0x04; S_{a62} of SMF II in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x16; R_RAM_DATA = 0x04; S_{a62} of SMF II in 2nd altern. buffer

R_RAM_ADDR0 = 0x40 + 0x1A; R_RAM_DATA = 0x00; S_{a63} of SMF II in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x1A; R_RAM_DATA = 0x00; S_{a63} of SMF II in 2nd altern. buffer

R_RAM_ADDR0 = 0x40 + 0x1E; R_RAM_DATA = 0x00; S_{a64} of SMF II in 1st altern. buffer
R_RAM_ADDR0 = 0x60 + 0x1E; R_RAM_DATA = 0x00; S_{a64} of SMF II in 2nd altern. buffer

Now the complete multiframe is specified for transmit direction.

5.5 State machine

HFC-E1 is equipped with a fully ETSI TBR 4 [2] compliant E1 interface which handles 32 time slots of 8 bits each. The time slots are numbered 0 . . 31. Time slot 0 is used for synchronization purposes and for the CRC-4 procedure which checks for data integrity. All other slots can be used for data transmission. In ISDN environments time slot 16 is normally used as D-channel.

HFC-E1 provides the current F state of the E1 interface in register R_E1_RD_STA. A new state can be set by writing its number into bitmap V_E1_SET_STA of register R_E1_WR_STA. The new state is loaded with V_E1_LD_STA = '1' in the same register. This can be done to overwrite the automatic E1 state machine. The state machine can be set back to automatic mode with V_E1_LD_STA = '0'.

The state machine receives several signals from the frame synchronization block and the receive data path blocks as shown in figure 5.2 on page 195. These signals can also be read from the host processor by register access:

- Receiving alarm indication signal (AIS)
- Remote alarm indication (RAI, A-bit of the received doubleframe or multiframe)
- CRC status of the received data
- Remote CRC status (received $E_{1,2}$ bits)
- Frame synchronization status
- Receive LOS status

These signals influence the automatic state machine which passes the current state to the *transmit frame and data controller* (see Figure 5.2 on page 195).

The E1 interface activation and deactivation layer 1 of the finite state matrix in NT mode is shown in Table 5.5. This is a modified state matrix concerning specification [3]. The HFC-E1 state machine has no states G 2 and G 4. These can easily be implemented in software: Internal network failures FC 1 and FC 3 can be notified by sending AIS.

Table 5.6 shows the E1 interface activation and deactivation layer 1 of the finite state matrix in TE mode (for more details see [3]).

Table 5.5: *E1 interface activation/deactivation layer 1 matrix for NT mode*

State name:	Power off at NT	Operational	FC 2	FC 4	Power on at NT
State number:	G 0	G 1	G 3	G 5	G 6
Signal:	No signal	Normal operational frames	Normal operational frames	RAI	No signal
Event:					
Loss of NT power	/	G 0	G 0	G 0	G 0
Return of NT power	G 6	/	/	/	/
Normal operational frames, no internal network failure	/	—	G 1	G 1	/
Internal network failure FC 2	/	G 3	G 3	G 3	G 3
Reception of RAI FC 2	/	G 5	—	G 5	G 5
Internal network failure FC 4	/	G 5	G 5	G 5	G 5
Loss of operational frames FC 4	/	G 3	G 3	—	G 3

Legend:

—	No state change
/	Impossible situation
FC 2, FC 4	Fault conditions, see [2, 3] for more information

Table 5.6: *E1 interface activation/deactivation layer 1 matrix for TE mode*

State name:	Power off at TE	Operational	FC 1	FC 2	FC 3	FC 4	Power on at TE
State number:	F 0	F 1	F 2	F 3	F 4	F 5	F 6
Signal:	No signal	Normal operational frames	Normal operational frames	Frames with RAI	Frames with RAI	Normal operational frames	No signal
Event:							
Loss of TE power	/	F 0	F 0	F 0	F 0	F 0	F 0
Return of TE power	F 6	/	/	/	/	/	/
Normal operational frames from NT side	/	—	F 1	F 1	F 1	F 1	/
Reception of RAI	/	F 2	—	F 2	F 2	F 2	F 2
Loss of signal or frame alignment	/	F 3	F 3	—	F 3	F 3	F 3
Reception of AIS	/	F 4	F 4	F 4	—	F 4	F 4
Reception of RAI and continuous CRC error report	/	F 5	F 5	F 5	F 5	—	F 5

Legend:

—	No state change
/	Impossible situation
FC 1 . . FC 4	Fault conditions, see [2, 3] for more information

5.6 Synchronization clocks

Details of the E1 interface clock synchronization is shown in Figure 5.2 on page 195. Clock distribution is divided into the *receive clock unit* of the E1 interface and the *transmit clock unit*.

5.6.1 Receive clock unit

The receive data path is synchronizes with a clock which is derived from the received bit stream. Several signals are extracted from the data stream inside the frame synchronization block. These are reported in register R_SYNC_STA. Missing or wrong FAS are counted with a 16 bit counter with the low byte in register R_FAS_ECL and the high byte in R_FAS_ECH. Any code error of the AMI or HDB3 code is counted in R_VIO_ECL (low byte) and R_VIO_ECH (high byte). The counter behaviour can be configured with bit V_AUTO_ERR_RES in register R_TX1 [3].

With V_AUTO_ERR_RES = '0' the counters are reset to zero with any read access to the counter's high byte. So the low byte must be read first. Please note, that the counters do never stop counting in this mode. After reaching the maximum value the counters restart with zero again.

In practice, the error rate *errors/second* is most interesting. This mode can be selected with V_AUTO_ERR_RES = '1'. An automatic counter reset is initiated every second after the value has been buffered. Any read access to the error counter register has no influenced to the counter value.

The condition for loss of receive signal (LOS) can be set in the registers R_LOS0 and R_LOS1. A LOS condition is reported in bit V_SIG_LOS of register R_SYNC_STA and by changing the state of the state machine accordingly.

5.6.2 Transmit clock unit

The transmit clock unit can select from three frame synchronization clock inputs:

1. Receive frame synchronization clock
2. F0IO clock (from pin 119)
3. SYNC_I clock (from pin 97)

The synchronization select block 'sync select' in Figure 5.2 generates the frame clock for the E1 transmit data as shown in Figure 5.3.

The following jitter attenuator block (JATT) contains a PLL to generate a very accurate 32.768 MHz clock which is synchronized to the selected frame clock. For normal operation, the JATT PLL should be enabled with V_JATT_OFF = '0' and set to automatic mode with V_JATT_MAN = '0' in register R_SYNC_CTRL. Register R_JATT_CFG must be set to 0x9C.

The JATT PLL status can be read from R_JATT_STA. The frequency is adjusted when there is a frequency difference between the synchronization source and the transmit frame clock. This is reported with V_JATT_ADJ = '01'. When the frequency is stable, the phase adjustment starts and V_JATT_ADJ has the value '10'. After this, only small readjustments of frequency and phase are necessary. This stable state is reported with V_JATT_ADJ = '11'.

[3] Please note, that V_AUTO_ERR_RES configures the behaviour of all six error counters.

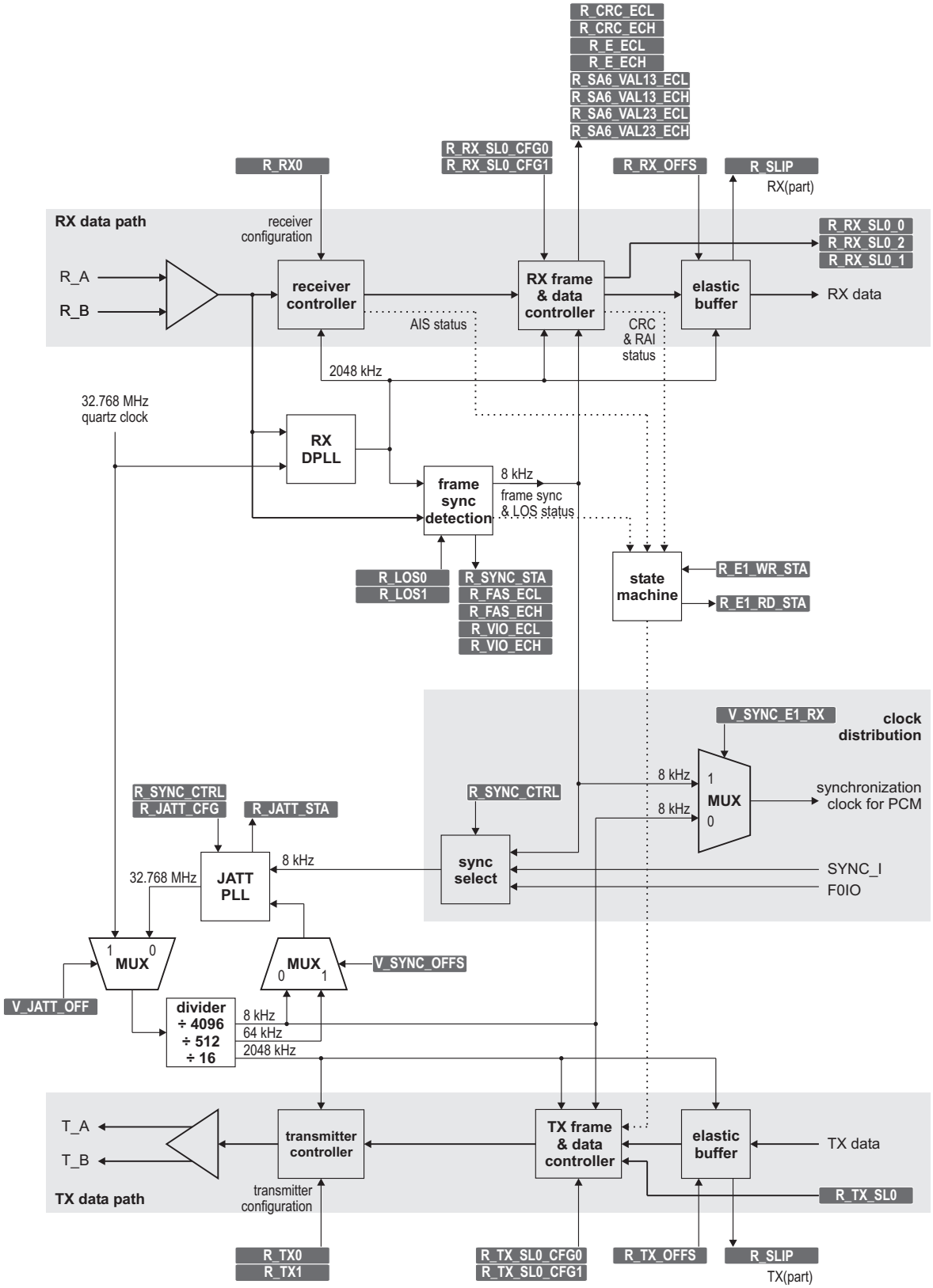


Figure 5.2: *Block diagram of the E1 interface module*

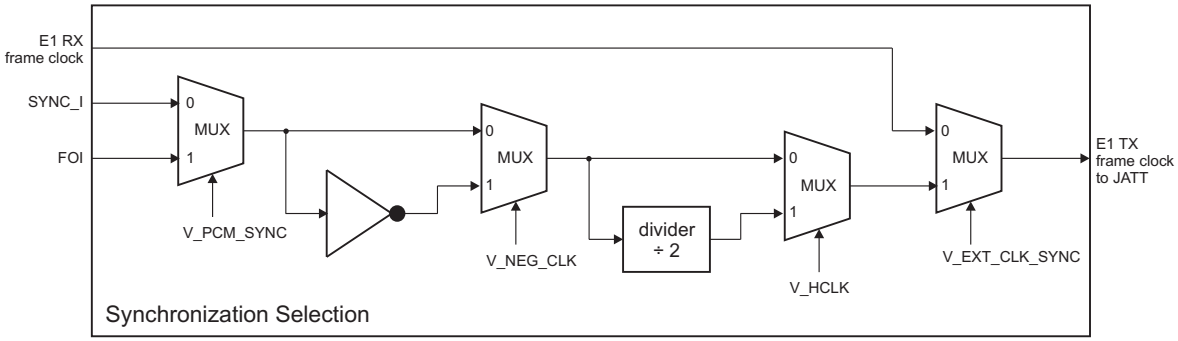


Figure 5.3: *Detail of the E1 interface synchronization selection shown in Figure 5.2*

5.7 Transmitter and receiver configuration

Fundamental interface mode selections can be done by writing registers R_RX0 for receive direction and R_TX0 and R_TX1 for transmit direction.

V_TX_CODE of register R_TX0 and V_RX_CODE of register R_RX0 are used to select the data coding. AMI code or HDB3 code can be chosen for wire connections, while NRZ mode is typically used for fiber optic data transmission. In this case R_A is data input and R_B is clock input in receive direction; T_A is data output and T_B is clock output accordingly.

For normal E1 operation with the interface circuitry of Figure 5.4 and 5.5, V_RX_CODE should be set to '01' in register R_RX0 while the other bits in this register remain in their reset state '0'.

5.8 External circuitries

Figures 5.4 and 5.5 show the standard E1 circuitries.

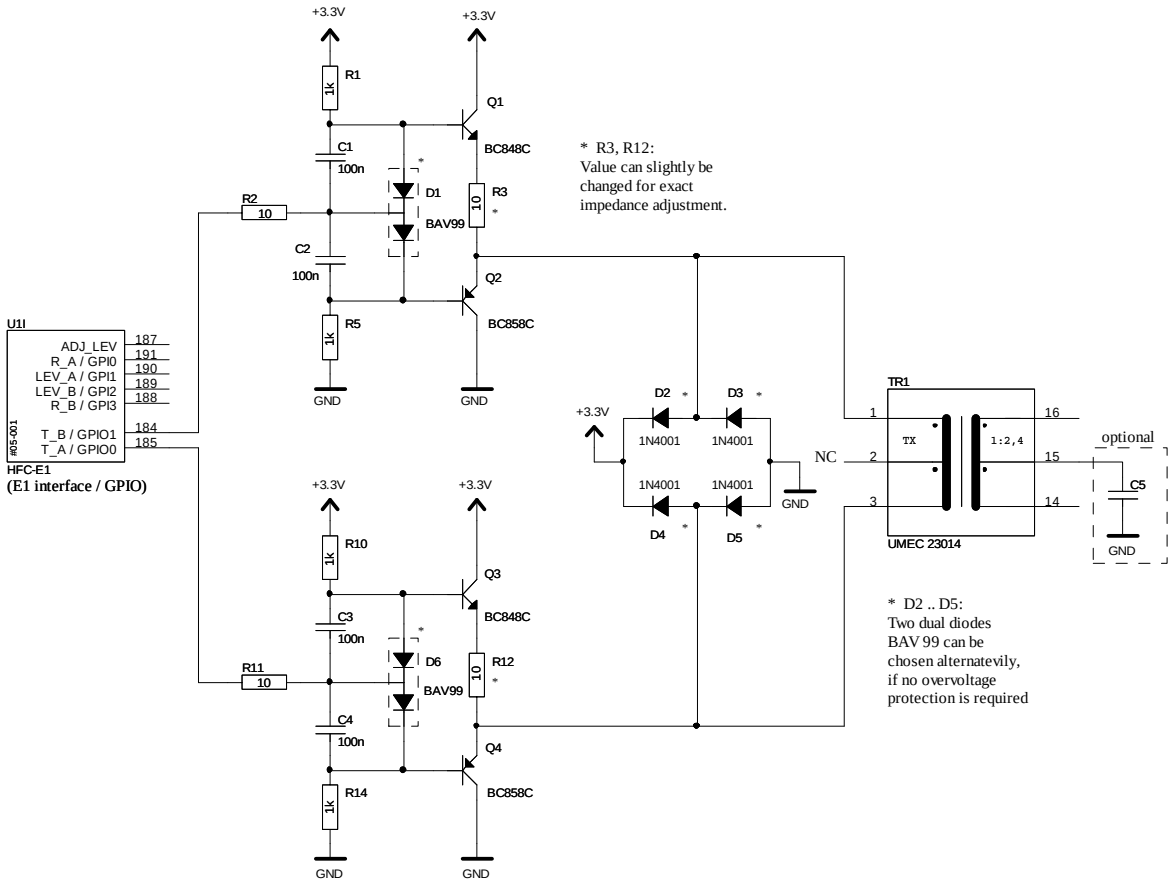


Figure 5.4: *External E1 transmit circuitry*

The E1 transmit pins are supplied from the power pin VDD_E1 whose voltage level is responsible for the E1 signal amplitude. Figure 5.6 shows an example how to generate a software programmable VDD_E1 voltage. This circuitry uses pin PWM0 for voltage adjustment.

Many applications do not require software adjustment. In that case any other circuitry which can provide a stable voltage can be used. Typically, VDD_E1 must be in the range 2.5 V . . 3.2 V depending on the actual transformer and the circuitry dimensioning.

The RJ45 connection is shown for LT and TE mode in Figures 5.7 and 5.8.

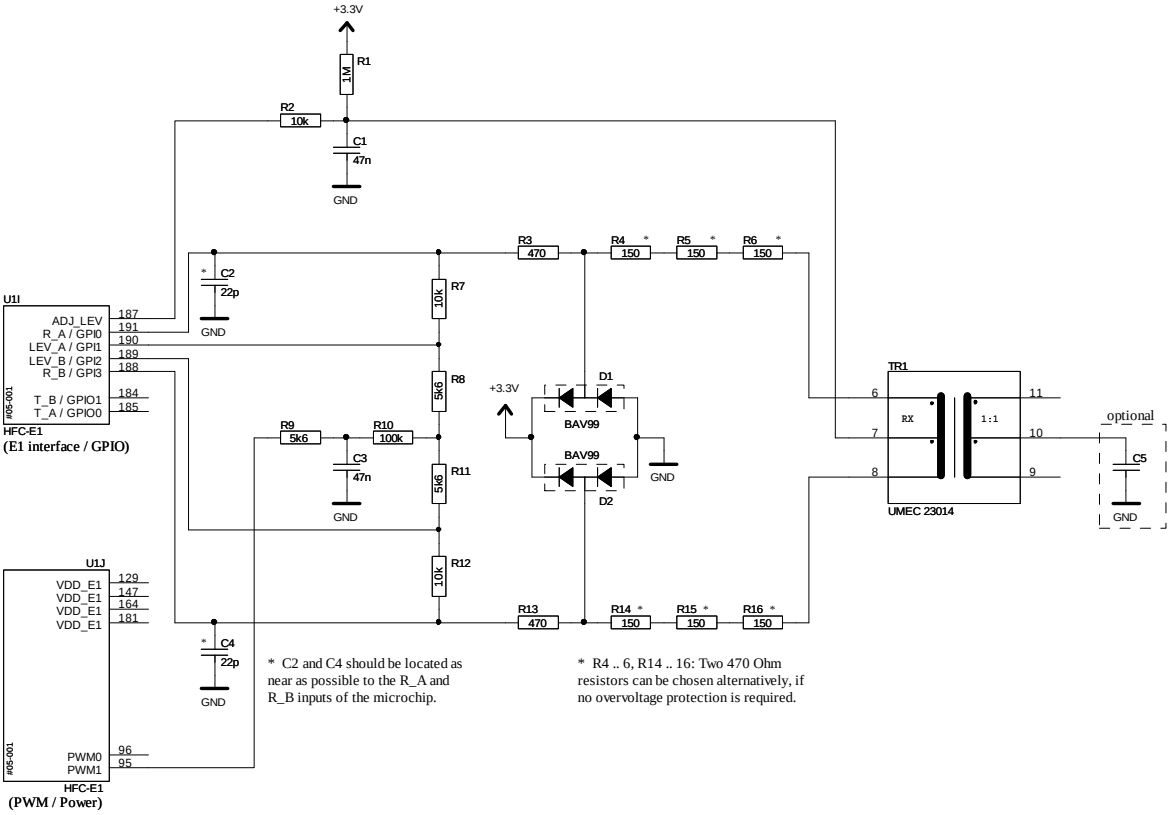


Figure 5.5: *External E1 receive circuitry*

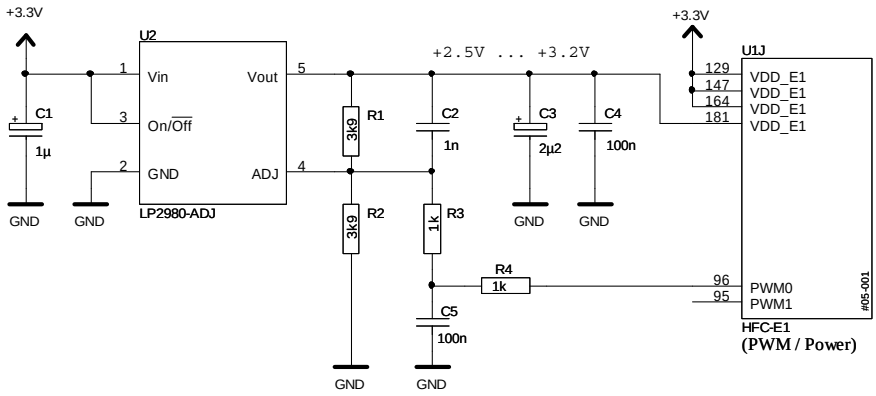


Figure 5.6: *VDD_E1 voltage generation*

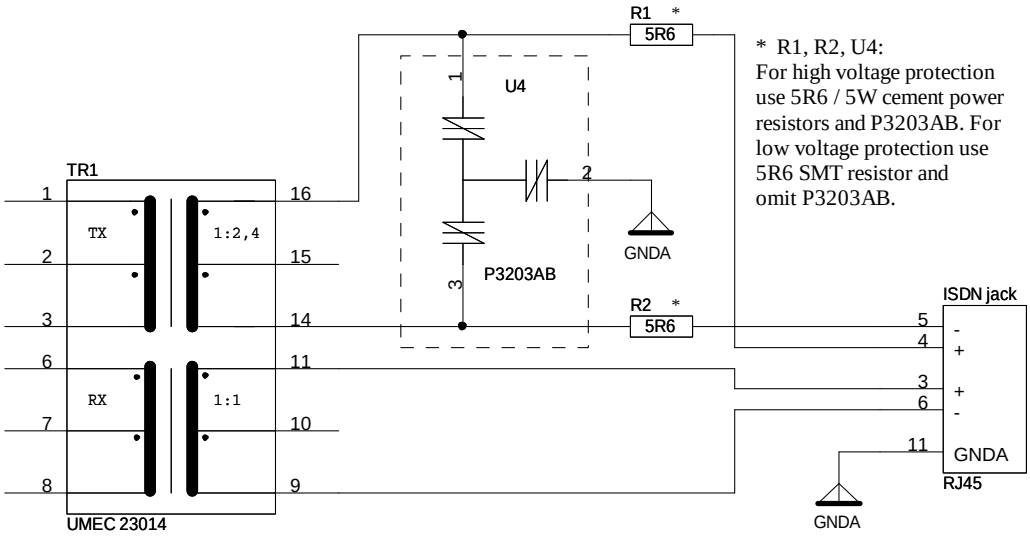


Figure 5.7: *Connector circuitry in LT mode (shown without termination)*

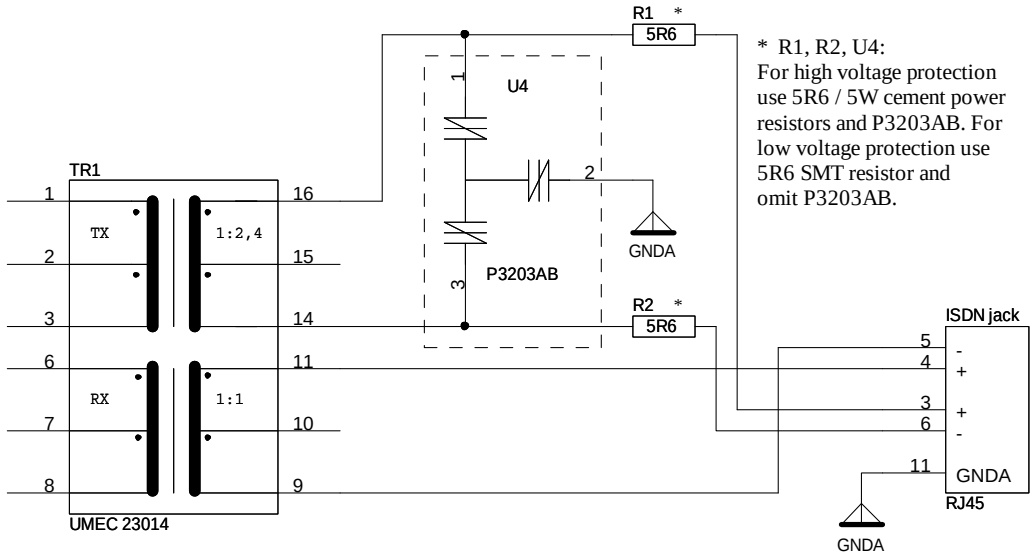


Figure 5.8: *Connector circuitry in TE mode (shown without termination)*

5.9 E1 transformers

Customers of Cologne Chip can choose from a variety of E1 transformers for Primary Rate Interface. All transformers are compatible to HFC-E1 that fulfill two criteria:

- Turns ratio of 2.4 : 1 (line side : chip side) for transmit and 1:1 for receive path.
- Center tap on the chip side (required for Cologne Chip receiver circuitry)

Several companies provide transformers and transformer modules that can be used with our ISDN Primary Rate Interface controller. Most popular are SMD dual transformer modules. Part numbers and manufacturers are listed in Table 5.7. A more extensive and regularly updated list can be found on Cologne Chip's website `http://www.colognechip.com`.

The transformer list has not been compiled under aspects of RoHS compliance. For the current RoHS status of the listed parts, please contact the transformer manufacturers straight.

Table 5.7: *E1 transformer part numbers and manufacturers*

E1 module part number	Manufacturer
Dual transformer module: S553-6500-A5 (SMD) S553-6500-A6 (SMD)	**Bel Fuse Inc.** United Kingdom URL: http://www.belfuse.com
Dual Transformer Module: TG91-1505N1 (SMD) TG91-1505NZ (SMD) TGSP-S225NZ (SMD, with choke)	**Halo Electronics, Inc.** United States URL: http://www.haloelectronics.com
Dual transformer module: T1176 (SMD, with choke) T1219 (SMD, with choke) T1308 (SMD) TX1321 (SMD)	**Pulse Engineering, Inc.** United States URL: http://www.pulseeng.com
Dual transformer module: MJM-022	**Talema Elektronik GmbH** Germany URL: http://www.talema.net
Transformer module: UT 23014	**UMEC GmbH** Germany, Taiwan, United States URL: http://www.umec.de

(continued on next page)

Table 5.7: *E1 transformer part numbers and manufacturers*

(continued from previous page)

E1 module part number		Manufacturer
Dual transformer module:	7-M5014-X011 (SMD) 7-M5026-X022 (SMD)	**Vacuumschmelze GmbH & Co. KG** Germany URL: http://www.vacuumschmelze.com
Dual transformer module:	503 16 007 00 (SMD)	**Vogt electronic AG** Germany URL: http://www.vogt-electronic.com

Please note: Cologne Chip cannot take any liability concerning the product names, characteristics and availability. Products can change without notice. Please refer to the manufacturer in case of doubt.

5.10 Register description

5.10.1 Write only registers

R_E1_WR_STA	(w)	0x20

E1 state machine register

This register is used to set a new state. The current state can be read from register R_E1_RD_STA.

Bits	Reset value	Name	Description
2..0	0	**V_E1_SET_STA**	**Binary value of new F state in TE mode** V_E1_LD_STA must also be set to load the state.
3		(reserved)	Must be '0'.
4	1	**V_E1_LD_STA**	**Load the new state** '0' = enable the state machine '1' = load the prepared state (V_E1_SET_STA) and stops the state machine **Note:** After writing an invalid state the state machine goes to deactivated state.
7..5		(reserved)	Must be '000'.

R_LOS0	(w)	0x22

Alarm set value for loss of input signal

Bits	Reset value	Name	Description
7..0	0	**V_LOS0**	**LOS alarm** LOS alarm will be active in V_SIG_LOS of register R_SYNC_STA if the incoming data stream has no transitions in (V_LOS0 + 1) ·16 consecutive data bit times. Maximum time is $256 \cdot 16 \cdot 488\,\text{ns} = 2\,\text{ms}$.

R_LOS1	(w)	0x23

Alarm clear value for loss of input signal

Bits	Reset value	Name	Description
7..0	0	V_LOS1	**LOS alarm** LOS alarm will be cleared in V_SIG_LOS of register R_SYNC_STA if the incoming data stream has V_LOS1 +1 transitions in LOS0 time interval. A new LOS0 time interval starts after LOS alarm is cleared.

R_RX0	(w)	0x24

E1 receiver configuration register 0

Bits	Reset value	Name	Description
1..0	0	**V_RX_CODE**	**Receive code** '00' = NRZ (pin R_A is data input and pin R_B is clock input in NRZ mode) '01' = HDB3 code '10' = AMI code '11' = reserved
2	0	**V_RX_FBAUD**	**Full / half bauded** '0' = receive pulse is half bit long '1' = receive pulse is full bit long This bit can be used to configure the pulse width in AMI and HDB3 code. It should be set to '1' in NRZ mode.
4..3	0	(reserved)	Must be '00'.
5	0	**V_RX_INV_CLK**	**Polarity of clock** '0' = non-inverted clock '1' = inverted clock This bit inverts the received clock from pin R_B . It is ignored if HDB3 or AMI code is selected.
6	0	**V_RX_INV_DATA**	**Polarity of received data** '0' = non-inverted data '1' = inverted data Data can be inverted after decoding from NRZ, HDB3 or AMI code to binary data.
7	0	**V_AIS_ITU**	**AIS alarm specification** '0' = according to ETS 300233 '1' = according to ITU-T G.775

R_RX_SL0_CFG0	(w)	0x25

E1 time slot 0 configuration (receive direction), register 0

Bits	Reset value	Name	Description
0	0	V_NO_INSYNC	**Disable frame synchronization** '0' = frame synchronization is continiously derived from input data '1' = disable frame synchronization to input data This bit can be used to switch off the frame synchronization. This should only be done for test purposes after frame synchronization is achieved.
1	0	V_AUTO_RESYNC	**Automatic resynchronization** '0' = multiframe resynchronization must manually be started after loss of synchronization '1' = the search for new double- and multiframing is automatically initiated after loss of synchronization This bit is only valid in CRC-4 multiframe mode and it is ignored in doubleframe mode.
2	0	V_AUTO_RECO	**Automatic error recovery** '0' = the number of CRC-4 errors has no influence to the synchronization state '1' = if there are more than 914 CRC errors in one second, the receiver will search for new double- and multiframing This bit is only valid in CRC-4 multiframe mode and it is ignored in doubleframe mode.
3	0	V_NFAS_COND	**FAS and NFAS condition** '0' = loss of synchronization if 3 or 4 (depending on V_FAS_LOSS) consecutive incorrect FAS or NFAS are received '1' = incorrect NFAS has no influence to the synchronization state
4	0	V_FAS_LOSS	**Loss of FAS and NFAS** '0' = loss of synchronization if 3 consecutive incorrect FAS or NFAS are received '1' = loss of synchronization if 4 consecutive incorrect FAS or NFAS are received
5	0	V_XCRC_SYNC	**Extended CRC-4 to non-CRC-4** '0' = according to ITU-T G.706 '1' = according to ITU-T G.706 except that the synchronizer will still search for multiframing even if the 400 ms is expired

(continued on next page)

(continued from previous page)

Bits	Reset value	Name	Description
6	0	**V_MF_RESYNC**	**Multiframe resynchronization** When this bit is set, the resynchronization of CRC-4 multiframe alignment is initiated without influencing doubleframe synchronous state. If V_AUTO_RESYNC is enabled and multiframe alignment can not be regained, a new search of doubleframe is initiated. **Note:** This bit is only valid in CRC-4 multiframe format. It is reset automatically.
7	0	**V_RESYNC**	**Resynchronization** '0' = normal operation '1' = initiate resynchronization of receive frame This bit is reset automatically.

R_RX_SL0_CFG1	(w)	0x26

E1 time slot 0 configuration (transmit direction), register 1

Bits	Reset value	Name	Description
0	0	V_RX_MF	**Multiframe mode** '0' = normal doubleframe mode '1' = multiframe mode (CRC-4)
1	0	V_RX_MF_SYNC	**Multiframe alignment error** '0' = normal operation '1' = MFA error leads to loss of synchronization
2	0	V_RX_SL0_RAM	**Time slot 0 data destination** '0' = time slot 0 data is written into HFC-channel 0 '1' = time slot 0 data is written into alternating RAM buffer
4..3		(reserved)	Must be '00'.
5	0	V_ERR_SIM	**Error simulation** This bit is for diagnostic purpose only. '0' = no action '1' = increment all error counters once
6	0	V_RES_NMF	**Reset 'no multiframe found' (NMF) status** '0' = no action '1' = reset no MFA found status (V_NO_MF_SYNC) which is set in V_NO_MF_SYNC of register R_SYNC_STA after 400 ms of MFA searching This bit is automatically cleared.
7		(reserved)	Must be '0'.

R_TX0	(w)	0x28

E1 transmitter configuration, register 0

Bits	Reset value	Name	Description
1..0	0	**V_TX_CODE**	**Transmit code** '00' = NRZ (pin T_A is data output and pin T_B is clock output in NRZ mode) '01' = HDB3 code '10' = AMI code '11' = reserved
2	0	**V_TX_FBAUD**	**Full / half bauded** '0' = transmit pulse is half bit long '1' = transmit pulse is full bit long This bit can be used to configure the pulse width in AMI and HDB3 code. It should be set to '1' in NRZ mode.
4..3	0	(reserved)	Must be '00'.
5	0	**V_TX_INV_CLK**	**Polarity of clock** '0' = non-inverted clock '1' = inverted clock This bit inverts the clock to be transmitted on pin T_B in NRZ mode. It is ignored if HDB3 or AMI code is selected.
6	0	**V_TX_INV_DATA**	**Polarity of data** '0' = non-inverted data '1' = inverted data Binary data can be inverted before it is coded in NRZ, HDB3 or AMI code.
7	0	**V_OUT_EN**	**Buffer enable** '0' = output buffers disabled (tristate) '1' = output buffers enabled

Important !

Transmit data is only generated if bit V_OUT_EN of register R_TX0 is set to '1'.

R_TX1	(w)	0x29

E1 transmitter configuration, register 1

Bits	Reset value	Name	Description
0	0	(reserved)	Must be '0'.
1	0	**V_EXCHG**	**TxD-exchange** '0' = normal operation '1' = exchange data output pins T_A and T_B
2	0	**V_AIS_OUT**	**Generate AIS output signal** '0' = normal operation '1' = continuous '1's are generated
4..3		(reserved)	Must be '00'.
5	0	**V_ATX**	**Transmitter mode** '0' = standard transmitter '1' = analog transmitter tandem mode
6	0	**V_NTRI**	**No tristate** '0' = tristate for gap between pulses enabled '1' = tristate for gap between pulses disabled
7	0	**V_AUTO_ERR_RES**	**Error counter mode** '0' = normal counter operation after reaching maximum count or any read access to the counter's high byte, a counter starts at 0 again '1' = every second the error counters will be reset automatically after they are latched **Note:** This register applies to all six error counters.

Please note !

The typical operational values of register R_TX1 differ from the reset values.

The default operational settings are:
(reserved): '0'
V_EXCHG: '0'
V_ATX: '1'
V_NTRI: '1'

R_TX_SL0_CFG0	(w)	0x2C

E1 time slot 0 configuration (transmit direction), register 0

This register is only used if V_TRP_SL0 in register R_TX_SL0_CFG1 is not set.

Bits	Reset value	Name	Description
0	0	**V_TRP_FAS**	**Transparent S_i(FAS) bit** '0' = semiautomatic mode (S_i bit will be taken from V_TX_FAS in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
1	0	**V_TRP_NFAS**	**Transparent S_i(NFAS) bit** '0' = semiautomatic mode (S_i bit will be taken from V_TX_NFAS in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
2	0	**V_TRP_RAL**	**Transparent remote alarm** '0' = semiautomatic mode (remote alarm bit will be generated internally from the state machine) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
3	0	**V_TRP_SA4**	**Transparent S_{a4} bit** '0' = semiautomatic mode (S_{a4} bit will be taken from V_TX_SA4 in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
4	0	**V_TRP_SA5**	**Transparent S_{a5} bit** '0' = semiautomatic mode (S_{a5} bit will be taken from V_TX_SA5 in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
5	0	**V_TRP_SA6**	**Transparent S_{a6} bit** '0' = semiautomatic mode (S_{a6} bit will be taken from V_TX_SA6 in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)

(continued on next page)

(continued from previous page)

Bits	Reset value	Name	Description
6	0	**V_TRP_SA7**	**Transparent S_{a7} bit** '0' = semiautomatic mode (S_{a7} bit will be taken from V_TX_SA7 in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)
7	0	**V_TRP_SA8**	**Transparent S_{a8} bit** '0' = semiautomatic mode (S_{a8} bit will be taken from V_TX_SA8 in register R_TX_SL0) '1' = transparent mode (HFC-channel 0 data or RAM data will be used, see V_TX_SL0_RAM of register R_TX_SL0_CFG1)

R_TX_SL0 **(w)** **0x2D**

E1 time slot 0 transmit data

This register is only used if V_TRP_SL0 in register R_TX_SL0_CFG1 is not set.

Bits	Reset value	Name	Description
0	0	**V_TX_FAS**	**S_i(FAS) bit** This bit is only used in doubleframe format.
1	0	**V_TX_NFAS**	**S_i(NFAS) bit** This bit is only used in doubleframe format.
2	0	**V_TX_RAL**	**Remote alarm bit** '0' = remote alarm bit is generated from the state machine '1' = remote alarm bit is fixed to '1'
3	0	**V_TX_SA4**	**S_{a4} bit**
4	0	**V_TX_SA5**	**S_{a5} bit**
5	0	**V_TX_SA6**	**S_{a6} bit**
6	0	**V_TX_SA7**	**S_{a7} bit**
7	0	**V_TX_SA8**	**S_{a8} bit**

R_TX_SL0_CFG1	(w)	0x2E

E1 time slot 0 configuration (transmit direction), register 1

Bits	Reset value	Name	Description
0	0	**V_TX_MF**	**Framing selection** '0' = doubleframe format '1' = multiframe mode (CRC-4)
1	0	**V_TRP_SL0**	**Time slot 0 transparent mode** '0' = semiautomatic mode '1' = transparent mode (HFC-channel 0 data or RAM data will be used and the registers R_TX_SL0_CFG0 and R_TX_SL0 are ignored)
2	0	**V_TX_SL0_RAM**	**Time slot 0 data source** '0' = time slot 0 data comes from HFC-channel 0 '1' = time slot 0 data comes from alternating RAM buffer
3		(reserved)	Must be '0'.
4	0	**V_TX_E**	**Automatic transmission of submultiframe status** '0' = XS13 and XS15 bits from V_XS13 and V_XS15 of this register are transmitted '1' = E-bits are transmitted (CRC-4 calculation result)
5	0	**V_INV_E**	**Polarity of E-bits** '0' = non-inverted E-bits '1' = inverted E-bits This bit is ignored if V_TX_E is set to '0'.
6	0	**V_XS13**	**Transmit spare bit XS13** (Frame 13 of multiframe) '0' = XS13 is '0' '1' = XS13 is '1' **Note:** This bit is only valid in CRC-4 multiframe.
7	0	**V_XS15**	**Transmit spare bit XS15** (Frame 15 of multiframe) '0' = XS15 is '0' '1' = XS15 is '1' **Note:** This bit is only valid in CRC-4 multiframe.

R_JATT_CFG	(w)	0x2F

Jitter attenuation configuration register

This register can be used to change the JATT PLL behaviour. The automatic JATT adjustment should be used in normal operation. Manual JATT adjustment is only usefull for diagnostic purposes.

It is strongly recommended to set the operation value 0x9C for this register.

Bits	Reset value	Name	Description
0	0	**V_JATT_FRQ_EN**	**Enable JATT frequency adjustment** '0' = frequency adjustment disabled '1' = frequency adjustment enabled This bit is ignored if the JATT PLL is in automatic mode. **Note:** Only one bit of V_JATT_FRQ_EN and V_JATT_PH_EN should be set at a time to avoid interdependend adjustment conflicts.
1	0	**V_JATT_PH_EN**	**Enable JATT phase adjustment** '0' = phase adjustment disabled '1' = phase adjustment enabled This bit is ignored if the JATT PLL is in automatic mode. **Note:** Only one bit of V_JATT_FRQ_EN and V_JATT_PH_EN should be set at a time to avoid interdependend adjustment conflicts.
3..2	0	**V_JATT_FRQ_IV**	**JATT frequency adjustment interval** '00' = 1024 ms '01' = 512 ms '10' = 256 ms '11' = 128 ms
7..4	0	**V_JATT_PAR**	**JATT configuration parameter** This bitmap should be set to '1001' for automatic and non-automatic JATT operation.

Please note !

Register R_JATT_CFG should be set to 0x9C for normal operation and the automatic mode of the JATT PLL should be enabled with V_JATT_MAN = '0' in register R_SYNC_CTRL.

V_JATT_FRQ_EN	=	'0'	(irrelevant in auto mode)
V_JATT_PH_EN	=	'0'	(irrelevant in auto mode)
V_JATT_FRQ_IV	=	'11'	(128 ms frequency adjust intervall)
V_JATT_PAR	=	'1001'	(standard operational parameter)

R_RX_OFFS	(w)	0x30

E1 receive buffer configuration register

Bits	Reset value	Name	Description
1..0	0	**V_RX_OFFS**	**Buffer pointer offset** Elastic buffer offset in number of frames (0..3)
2	0	**V_RX_INIT**	**Buffer initialization** Some data may be lost when this bit is set. This bit is automatically cleared.
7..3		(reserved)	Must be '00000'.

R_SYNC_OUT	(w)	0x31

E1 synchronization source selection for PCM master

Bits	Reset value	Name	Description
0	0	V_SYNC_E1_RX	**PCM master synchronization** '0' = PCM master synchronizes from the E1 TX end of frame (EOF) signal '1' = PCM master synchronizes from the E1 RX end of frame (EOF) signal
4 . . 1		(reserved)	Must be '00000'.
5	0	V_IPATS0	**RAI pulse configuration for IPATS test** '0' = normal opeartion '1' = delete short RAI low pulses, increase RAI to a minimum of more than 1 ms **Note:** This bit is only used for passing IPATS test equipment.
6	0	V_IPATS1	**CRC configuration for IPTAS test** '0' = normal operation '1' = delete CRC reporting over E-bits up to 8 ms after MFA synchronization **Note:** This bit is only used for passing IPATS test equipment.
7	0	V_IPATS2	**JATT configuration for IPATS test** '0' = normal operation '1' = stop jitter attenuator (JATT) adaptation when in F3 state **Note:** This bit is only used for passing IPATS test equipment.

R_TX_OFFS	(w)	0x34

E1 transmit buffer configuration register

Bits	Reset value	Name	Description
1..0	0	**V_TX_OFFS**	**Buffer pointer offset** Elastic buffer offset in number of frames (0..3)
2	0	**V_TX_INIT**	**Buffer initialization** Some data may be lost when this bit is set. This bit is automatically cleared.
7..3		(reserved)	Must be '00000'.

R_SYNC_CTRL	(w)	0x35

E1 transmit clock sychronization register

Bits	Reset value	Name	Description
0	0	**V_EXT_CLK_SYNC**	**E1 synchronization source selection for transmit data** '0' = clock synchronization derived from receive data '1' = synchronization is determined from V_PCM_SYNC, V_NEG_CLK and V_HCLK
1	0	**V_SYNC_OFFS**	**E1 synchronization type selection** '0' = TX and RX frame synchronization phase offset 0 '1' = TX and RX frame synchronization phase offset arbitrary **Note:** If this bit is set the synchronization process is faster because the phase offset can be arbitrary.
2	0	**V_PCM_SYNC**	**E1 synchronization source selection** '0' = synchronization from PCM pin SYNC_I '1' = synchronization from PCM pin F0IO
3	0	**V_NEG_CLK**	**External synchronization clock polarity** '0' = positive edge '1' = negative edge
4	0	**V_HCLK**	**Half clock frequency** '0' = normal operation '1' = external synchronization clock is divided by 2
5	0	**V_JATT_AUTO10**	**Restricted frequency search** '0' = automatic frequency search is initiated after 3 frequency mismatches in 0.5 s '1' = automatic frequency search is initiated after 10 frequency mismatches in 0.5 s
6	0	**V_JATT_MAN**	**Automatic or manual JATT adjustment** '0' = automatic JATT adjustment enabled '1' = automatic JATT adjustment disabled
7	0	**V_JATT_OFF**	**Enable or disable JATT PLL** '0' = JATT PLL is enabled '1' = JATT PLL is disabled (transmit clock is generated from crystal clock)

5.10.2 Read only registers

R_E1_RD_STA	(r)	0x20

E1 state machine register

Bits	Reset value	Name	Description
2..0	0	V_E1_STA	**E1 state** Binary value of current F state in TE mode.
5..3		(reserved)	
6	0	V_ALT_FR_RX	**Alternating RAM bank** Shows which RAM bank of time slot 0 data is currently used for receive data. This bit is toggled every 2 ms after a complete multiframe has been received.
7	0	V_ALT_FR_TX	**Alternating RAM bank** Shows which RAM bank of time slot 0 data is currently used for transmit data. This bit is toggled every 2 ms after a complete multiframe has been transmitted.

R_SYNC_STA	(r)	0x24

E1 synchronization status

Bits	Reset value	Name	Description
1..0	0	**V_RX_STA**	**Receive status** '00' = not synchronized '01' = FAS found '10' = NFAS found after FAS '11' = synchronized (FAS - NFAS - FAS found) This bitmap indicates the doubleframe synchronous state.
2	0	**V_FR_SYNC**	**Frame synchronization status** Frame synchronization status according to the selected doubleframe or multiframe mode. '0' = no frame synchronization achieved '1' = frame synchronization achieved (V_RX_STA = '11' in doubleframe or multiframe mode, V_MFA_STA = '10' in multiframe mode)
3	0	**V_SIG_LOS**	**LOS status** Loss of receive signal detected. The condition for LOS indication can be specified in register R_LOS0. V_SIG_LOS is automatically cleared when the condition defined in register R_LOS1 is reached.
5..4	0	**V_MFA_STA**	**Status of multiframe alignment (MFA)** '00' = no doubleframe or multiframe alignment found '01' = one MFA pattern found '10' = two consecutive MFA patterns found, MFA synchronous state achieved '11' = reserved
6	0	**V_AIS**	**Receiving Alarm Indication Signal (AIS)** This bit is set to '1' when a data stream with continous '1's is received.
7	0	**V_NO_MF_SYNC**	**No multiframe (NMF) synchronization** '1' = no multiframe synchronization found for 400 ms (V_MFA_STA has not achieved '10') This bit is reset by asserting V_RES_NMF in register R_RX_SL0_CFG1.

R_RX_SL0_0	(r)	0x25

E1 time slot 0 receive information, register 0

Bits	Reset value	Name	Description
0	0	**V_SI_FAS**	S_i(FAS) **in time slot 0**
1	0	**V_SI_NFAS**	S_i(NFAS) **in time slot 0**
2	0	**V_A**	**A-bit of time slot 0**
3	0	**V_CRC_OK**	**CRC result** '1' = CRC-4 ok
4	0	**V_TX_E1**	**E1-bit for transmit multiframe**
5	0	**V_TX_E2**	**E2-bit for transmit multiframe**
6	0	**V_RX_E1**	**E1-bit of receive multiframe**
7	0	**V_RX_E2**	**E2-bit of receive multiframe**

R_RX_SL0_1	(r)	0x26

E1 time slot 0 receive information, register 1

Bits	Reset value	Name	Description
0	0	**V_SA61**	$S_{A6}[1]$ **bit of time slot 0**
1	0	**V_SA62**	$S_{A6}[2]$ **bit of time slot 0**
2	0	**V_SA63**	$S_{A6}[3]$ **bit of time slot 0**
3	0	**V_SA64**	$S_{A6}[4]$ **bit of time slot 0**
5..4		(reserved)	
6	0	**V_SA6_OK**	S_{A6} **OK** '0' = The last three received sub-multiframes were not the same '1' = The same S_{A6} patterns were received in the last three consecutive SMFs
7	0	**V_SA6_CHG**	S_{A6} **pattern has changed** This bit is set to '1' when the received S_{A6} patterns of a sub-multiframe are different from the patterns of the previous sub-multiframe. V_SA6_CHG is automatically cleared after register read.

R_RX_SL0_2	(r)	0x27

E1 time slot 0 receive information, register 2

Bits	Reset value	Name	Description
0	0	**V_SA4**	S_{a4} **bit** This bitmap contains the last received S_{a4} bit.
1	0	**V_SA5**	S_{a5} **bit** This bitmap contains the last received S_{a5} bit.
2	0	**V_SA6**	S_{a6} **bit** This bitmap contains the last received S_{a6} bit. In multiframe mode S_{a6} is a 4 bit sequence and can be read from V_SA61 in register R_RX_SL0_1.
3	0	**V_SA7**	S_{a7} **bit** This bitmap contains the last received S_{a7} bit.
4	0	**V_SA8**	S_{a8} **bit** This bitmap contains the last received S_{a8} bit.
7..5		(reserved)	

R_JATT_STA	(r)	0x2B

Jitter attenuation state register

Bits	Reset value	Name	Description
4..0	0	(reserved)	
6..5		**V_JATT_ADJ**	This bitmap reports the JATT PLL state. '00' = JATT reset '01' = frequency adjustment is running '10' = phase adjustment is running '11' = frequency and phase readjustment in locked state
7	0	(reserved)	

R_SLIP (r) **0x2C**

Frequency slip warning register

Bits	Reset value	Name	Description
0	0	**V_SLIP_RX**	**Frequency slip in receive data direction** This bit is set when an overflow of the elastic receive buffer has occured as a result of a frequency slip. This bit is automatically cleared with a write access to register R_RX_OFFS with V_RX_INIT set to '1'.
3..1	0	(reserved)	
4	0	**V_SLIP_TX**	**Frequency slip in transmit data direction** This bit is set when an overflow of the elastic transmit buffer has occured as a result of a frequency slip. This bit is automatically cleared with a write access to register R_TX_OFFS with V_TX_INIT set to '1'.
7..5	0	(reserved)	

R_FAS_ECL (r) **0x30**

Error counter for missing or wrong FAS, low byte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_FAS_ECL**	**Bits [7..0] of FAS error count** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_FAS_ECH	(r)	0x31

Error counter for missing or wrong FAS, high byte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_FAS_ECH**	**Bits [15..8] of FAS error count** In ring counter mode, the counter is reset after the read access to the high byte.

R_VIO_ECL	(r)	0x32

Error counter for code violation of AMI or HDB3 code, low byte

Only code errors are counted. Valid code violations are not counted.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_VIO_ECL**	**Bits [7..0] of code violation error count** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_VIO_ECH	(r)	0x33

Error counter for code violation of AMI or HDB3 code, high byte

Only code errors are counted. Valid code violations are not counted.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	V_VIO_ECH	**Bits [15..8] of code violation error count** In ring counter mode, the counter is reset after the read access to the high byte.

R_CRC_ECL	(r)	0x34

Receive CRC-4 error count, low byte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	V_CRC_ECL	**Bits [7..0] of CRC-4 error count** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_CRC_ECH	(r)	0x35

Receive CRC-4 error count, high byte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	V_CRC_ECH	**Bits [15..8] of CRC-4 error count** In ring counter mode, the counter is reset after the read access to the high byte.

R_E_ECL	(r)	0x36

Error counter for received E_1 and E_2 bits (remote CRC-4 error reporting), low byte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_E_ECL**	**Bits [7..0] of CRC-4 error count** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_E_ECH	(r)	0x37

Error counter for received E_1 and E_2 bits (remote CRC-4 error reporting), highbyte

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_E_ECH**	**Bits [15..8] of CRC-4 error count** In ring counter mode, the counter is reset after the read access to the high byte.

R_SA6_VAL13_ECL	(r)	0x38

SA6-bit error indication counter for SA6 values '0001' and '0011', low byte

This counter increments with every received SA6 sequence which has the value '0001' or '0011'.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_SA6_VAL13_ECL**	**Bits [7..0] of the error counter** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_SA6_VAL13_ECH	(r)	0x39

SA6-bit error indication counter for SA6 values '0001' and '0011', high byte

This counter increments with every overflow of R_SA6_VAL13_ECL.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_SA6_VAL13_ECH**	**Bits [15..8] of the error counter** In ring counter mode, the counter is reset after the read access to the high byte.

R_SA6_VAL23_ECL	(r)	0x3A

SA6-bit error indication counter for SA6 values '0010' and '0011', low byte

This counter increments with every received SA6 sequence which has the value '0010' or '0011'.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_SA6_VAL23_ECL**	**Bits [7..0] of the error counter** In ring counter mode, reading the low byte buffers the high byte of this counter. Thus the low byte must be read first.

R_SA6_VAL23_ECH	(r)	0x3B

SA6-bit error indication counter for SA6 values '0010' and '0011', high byte

This counter increments with every overflow of R_SA6_VAL23_ECL.

The error counters can operate either as ring counters or as one-second counters which are reset automatically every second. This is selected for all six error counters together with bit V_AUTO_ERR_RES in register R_TX1.

Bits	Reset value	Name	Description
7..0	0	**V_SA6_VAL23_ECH**	**Bits [15..8] of the error counter** In ring counter mode, the counter is reset after the read access to the high byte.

Chapter 6

PCM interface

Table 6.1: *Overview of the HFC-E1 PCM interface registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x10	R_SLOT	244	0x18	R_F0_CNTL	255
0x14	R_PCM_MD0	245	0x19	R_F0_CNTH	255
0x15	R_SL_SEL0	246			
0x15	R_SL_SEL1	247			
0x15	R_SL_SEL2	247			
0x15	R_SL_SEL3	248			
0x15	R_SL_SEL4	248			
0x15	R_SL_SEL5	249			
0x15	R_SL_SEL6	249			
0x15	R_SL_SEL7	250			
0x15	R_PCM_MD1	251			
0x15	R_PCM_MD2	252			
0x15	R_SH0L	253			
0x15	R_SH0H	253			
0x15	R_SH1L	253			
0x15	R_SH1H	254			
0xD0	A_SL_CFG	254			

Table 6.2: *Overview of the HFC-E1 PCM pins (* : Second pin function)*

PCM pins:

Number	Name	Description
97	SYNC_I	Synchronization Input
98	SYNC_O	Synchronization Output
117	C2O	PCM bit clock output
118	C4IO	PCM double bit clock I/O
119	F0IO	PCM frame clock I/O (8 kHz)
120	STIO1	PCM data bus 1, I or O per time slot
121	STIO2	PCM data bus 2, I or O per time slot

CODEC select via enable lines:

Number	Name	Description
107	F1_7	PCM CODEC enable 7
108	F1_6	PCM CODEC enable 6
109	F1_5	PCM CODEC enable 5
110	F1_4	PCM CODEC enable 4
111	F1_3	PCM CODEC enable 3
112	F1_2	PCM CODEC enable 2
113	F1_1	PCM CODEC enable 1
114	F1_0	PCM CODEC enable 0

CODEC select via time slot number:

Number	Name	Description
106 *	NC	
107 *	F1_7	PCM CODEC enable 7
108 *	F1_6	PCM CODEC enable 6
109 *	F1_5	PCM CODEC enable 5
110 *	F1_4	PCM CODEC enable 4
111 *	F1_3	PCM CODEC enable 3
112 *	F1_2	PCM CODEC enable 2
113 *	F1_1	PCM CODEC enable 1
114 *	F1_0	PCM CODEC enable 0

6.1 PCM interface function

HFC-E1 can operate in PCM master mode or PCM slave mode. This is selected with V_PCM_MD in register R_PCM_MD0.

The PCM data rate is programmable for PCM master mode as shown in Table 6.3. F0IO has always a frequency of 8 kHz. Each time slot has a width of eight bits.

Table 6.3: *PCM master mode*

V_PCM_DR in register R_PCM_MD1	C4IO clock output	Number of time slots	Data rate
'00'	4.096 MHz	32	2 MBit/s
'01'	8.192 MHz	64	4 MBit/s
'10'	16.384 MHz	128	8 MBit/s
'11'			unused

HFC-E1 has two PCM data pins STIO1 and STIO2 which can both be input or output. Data direction can be selected for every time slot independently.

6.2 PCM data flow

The PCM data flow is shown in Figure 6.1. The input and output behavior has to be programmed with bitmap V_ROUT in the array register A_SL_CFG[SLOT].

The PCM output behavior is always setup from transmit slots. V_ROUT = '00' disables the PCM output, i.e. both output buffers are tristated and no data is transferred from the HFC-channel to the PCM module within this time slot. Any other value of V_ROUT enables the data transmission from the HFC-channel:

- V_ROUT = '10' enables the STIO1 output buffer.
- V_ROUT = '11' enables the STIO2 output buffer.
- Finally, V_ROUT = '01' disables both output buffers but enables the data transmission from the HFC-channel. This setting – in connection with the corresponding setting of the PCM input data path – is used to loop data internally without influencing the PCM bus.

PCM input data selects one of three data sources:

- V_ROUT = '10' receives data from STIO2 .
- V_ROUT = '11' receives data from STIO1 .
- V_ROUT = '01' is used for an internal data loop.

The data transfer to the receive HFC-channel can be disabled with V_ROUT = '00'.

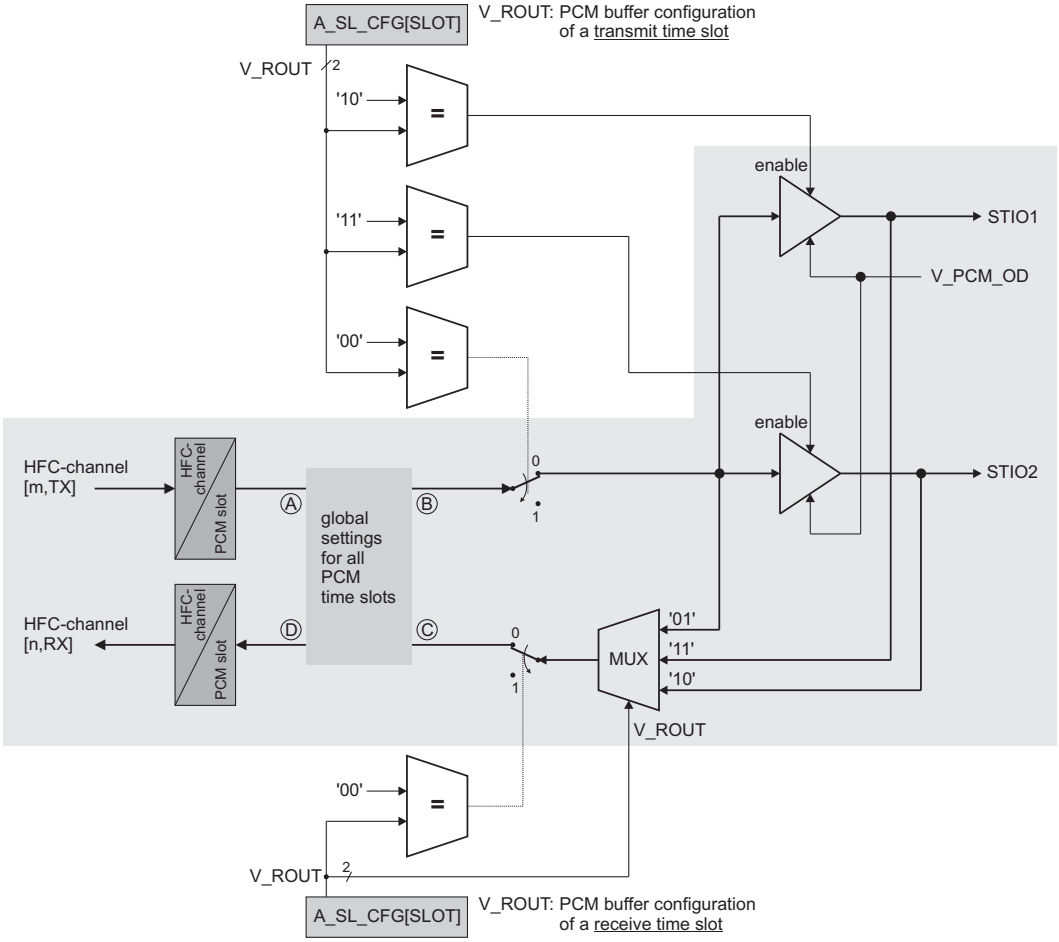


Figure 6.1: *PCM data flow for transmit and receive time slots (see Figure 6.2 for additional setting of all PCM time slots between Ⓐ..Ⓓ)*

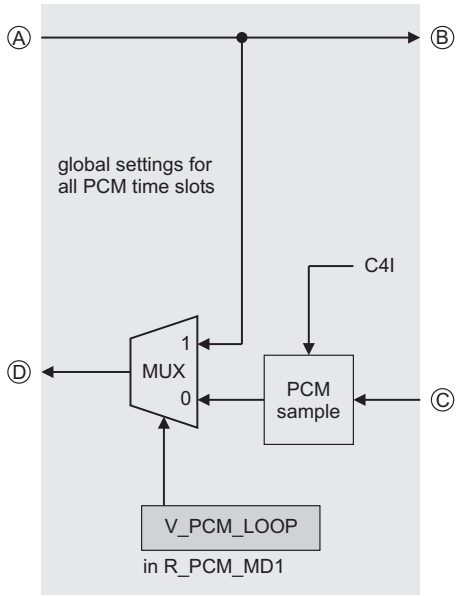


Figure 6.2: *Global settings for all PCM time slots (detail of Figure 6.1), normally used for test loop setup*

A corresponding transmit / receive pair of PCM time slots is typically programmed with the same value for V_ROUT in both directions. Table 6.4 summarizes these settings.

Table 6.4: *PCM data flow programming with the same value for V_ROUT in corresponding transmit and receive time slots*

V_ROUT	Data transmission from / to the HFC-channel *1	STIO1 I/O path *2	STIO2 I/O path *2	Description
'00'	disabled	tristated	tristated	PCM time slot not used
'01'	enabled	tristated	tristated	internal data loop
'10'	enabled	output	input	bidirectional data transfer
'11'	enabled	input	output	bidirectional data transfer

*1: PCM data flow configuration of a receive time slot

*2: PCM data flow configuration of a transmit time slot

Figure 6.1 shows the PCM data flow which can be programmed for each PCM time slot individually. Global settings to the PCM data flow are avaliable between Ⓐ . . Ⓓ as shown in Figure 6.2. When V_PCM_LOOP = '1' in register R_PCM_MD1, the PCM data is looped internally.

6.3 PCM initialization

After hard or soft reset the PCM interface starts an initialization sequence to set all A_SL_CFG registers of the PCM time slots to the reset value 0. This can be done only if valid C4IO and F0IO signals exist, which means that in slave mode there must be external C4IO and F0IO clocks. The initialization process stops after 2 F0IO periods. To check if the initialization sequence is finished after a reset, register R_F0_CNTL value must be equal or greater than 2.

Important !

The PCM data rate must be set immediately (about 40 µs) after PCM reset to ensure the complete array register reset procedure. Only the number of PCM time slots which are available, are initialized during reset. Thus it is not possible to change the PCM data rate later without manually array register reset.
This is important in slave mode to avoid uncontrolled data transmission caused by F0IO pulses from an external device.

6.4 PCM timing

The PCM interface of HFC-E1 can operate either in slave mode or master mode. Slave mode is default selection after HFC-E1 reset.

To configure HFC-E1 as PCM bus master, bit V_PCM_MD in register R_PCM_MD0 must be set to '1'. C4IO and F0IO signals are generated from HFC-E1 in this case and both pins have output characteristic.

Slave mode is selected with V_PCM_MD = '0'. C4IO and F0IO are input pins in slave mode. There must be external signals connected to C4IO and F0IO in this mode because these signals are used from the E1 interface and the flow controller as well.

The PCM bit rate is configured to either 2 MBit/s, 4 MBit/s or 8 MBit/s by bitmap V_PCM_DR in register R_PCM_MD1.

6.4.1 Master mode

Figure 6.3 shows the timing diagram for PCM master mode. The timing characteristics are specified in Table 6.5. STIO1 is shown as data input and STIO2 as data output of HFC-E1. However, both pins can change their I/O characteristic with every PCM time slot.

The F0IO pulse is one C4IO pulse long with the default value V_F0_LEN = '0' in register R_PCM_MD0. F0IO starts one C4IO clock earlier if bit V_F0_LEN = '1'.

6.4.2 Slave mode

Figure 6.4 shows the timing diagram for PCM slave mode. The timing characteristics are specified in Table 6.6. STIO1 is shown as data input and STIO2 as data output of HFC-E1. However, both pins can change their I/O characteristic with every PCM time slot.

The F0IO pulse is expected to be one C4IO pulse long with the default value V_F0_LEN = '0' in register R_PCM_MD0. F0IO is expected to start one C4IO clock earlier if bit V_F0_LEN = '1'.

If the E1 interface is synchronized from C4IO in LT mode, the frequency stability must be at least $\pm 10^{-4}$.

6.5 PCM clock synchronization

6.5.1 Overview

The PCM clock synchronization is shown in Figure 6.5. It is associated with the E1 clock synchronization which is shown in Figure 5.2 on page 195. Various programming features are implemented to fulfill many different application needs.

6.5.2 PLL programming for F0IO generation

C4IO is adjusted from the PCM DPLL (see Figure 6.5) during the last PCM time slot to synchronize the PCM interface with the E1 interface [1]. The maximum number of edge adjustments during one 125 µs cycle can be configured in the range 1 . . 4 by the bitmap value V_PLL_ADJ in register R_PCM_MD1. This automatic adjustment is enabled with V_PLL_MAN = '0' in register R_PCM_MD2.

[1] C4IO adjustment is only in operation when the PCM DPLL receives both 8 kHz reference clocks.

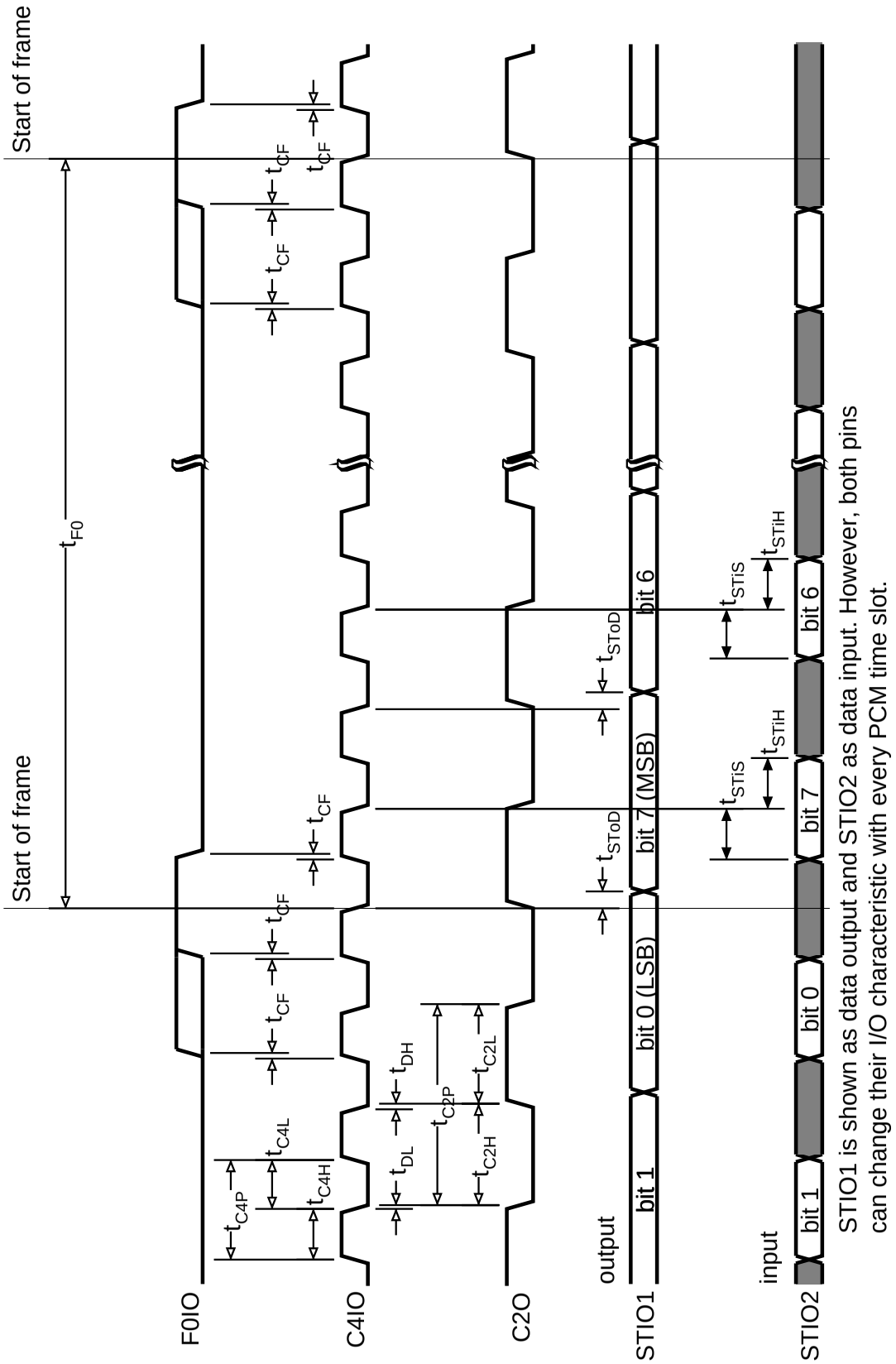


Figure 6.3: *PCM timing for master mode*

Table 6.5: *Symbols of PCM timing for master mode in Figure 6.3 (All values with 50 pF load. Larger load capacitance will increase output delays.)*

Symbol	min / ns	typ / ns	max / ns	Characteristic
t_C				Basic C4IO pulse width (not shown in the timing diagram)
		122.070		4.096 MHz C4IO clock for 2 MB/s
		61.035		8.192 MHz C4IO clock for 4 MB/s
		30.518		16.384 MHz C4IO clock for 8 MB/s
t_{adj}		20.345		Adjust time is half a period of 24.576 MHz clock (not shown in the timing diagram)
t_{C4H}	$t_C - 6 - t_{adj}$		$t_C + 6$	C4IO high width for 2 MBit/s and 4 MBit/s
	$4/3 \cdot t_C - 6 - t_{adj}$		$4/3 \cdot t_C + 6$	C4IO high width for 8 MBit/s
t_{C4L}	$t_C - 6$		$t_C + 6 + t_{adj}$	C4IO low width for 2 MBit/s and 4 MBit/s
	$2/3 \cdot t_C - 6$		$2/3 \cdot t_C + 6 + t_{adj}$	C4IO low width for 8 MBit/s
t_{C4P}	$2 \cdot t_C - 6 - t_{adj}$		$2 \cdot t_C + 6 + t_{adj}$	C4IO clock period
t_{C2H}	$2 \cdot t_C - 6 - t_{adj}$		$2 \cdot t_C + 6 + t_{adj}$	C2O high width
t_{C2L}	$2 \cdot t_C - 6 - t_{adj}$		$2 \cdot t_C + 6 + t_{adj}$	C2O low width
t_{C2P}	$4 \cdot t_C - 6 - t_{adj}$		$4 \cdot t_C + 6 + t_{adj}$	C2O clock period
t_{F0}	124994	125000	125006	F0IO cycle time without adjustment
	$124994 - t_{adj}$		$125006 + t_{adj}$	1 half clock adjustment
	$124994 - 2 \cdot t_{adj}$		$125006 + 2 \cdot t_{adj}$	2 half clocks adjustment
	$124994 - 3 \cdot t_{adj}$		$125006 + 3 \cdot t_{adj}$	3 half clocks adjustment
	$124994 - 4 \cdot t_{adj}$		$125006 + 4 \cdot t_{adj}$	4 half clocks adjustment
t_{DL}	3.92	5.04	7.54	C4IO ⎿ to C2O ⎿ delay
t_{DH}	3.87	5.07	7.69	C4IO ⎿ to C2O ⎾ delay
t_{CF}	0.5		8	C4IO ⎾ to F0IO ⎾ or C4IO ⎾ to F0IO ⎿
t_{STiS}	10			Data valid to C4IO ⎿ setup time
t_{STiH}	10			Data valid to C4IO ⎿ hold time
t_{SToD}	2		10	STIO output delay from C4IO ⎿

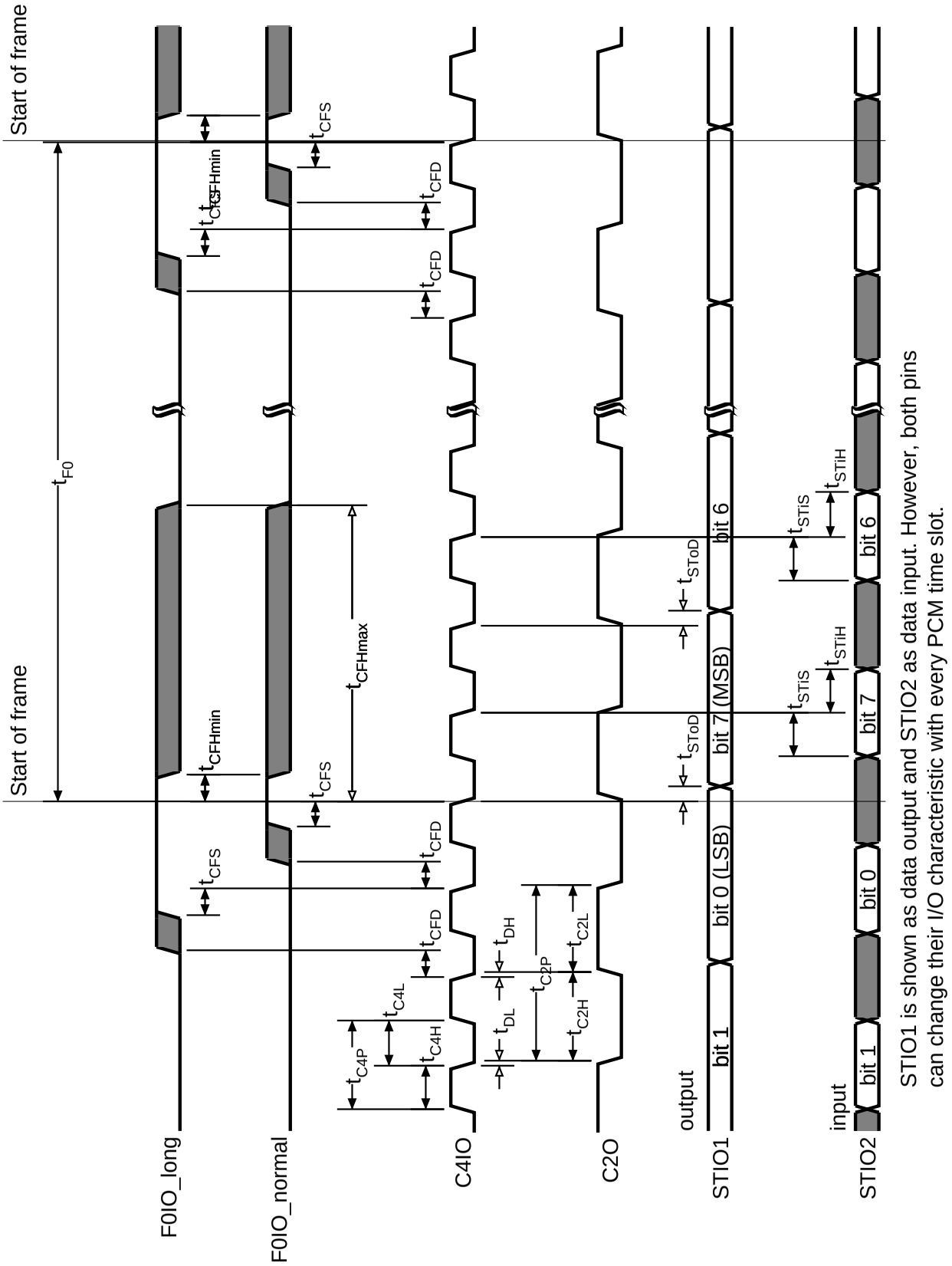


Figure 6.4: *PCM timing for slave mode*

Table 6.6: *Symbols of PCM timing for slave mode in Figure 6.4 (All values with 50 pF load. Larger load capacitance will increase output delays.)*

Symbol	min / ns	typ / ns	max / ns	Characteristic
t_C				Basic C4IO pulse width (not shown in the timing diagram)
		122.070		4.096 MHz C4IO clock for 2 MB/s
		61.035		8.192 MHz C4IO clock for 4 MB/s
		30.518		16.384 MHz C4IO clock for 8 MB/s
t_{C4H}	20	t_C		C4IO high width
t_{C4L}	20	t_C		C4IO low width
t_{C4P}		$2 \cdot t_C$		C4IO clock period
t_{C2H}		$2 \cdot t_C$		C2O high width
t_{C2L}		$2 \cdot t_C$		C2O low width
t_{C2P}		$4 \cdot t_C$		C2O clock period
t_{F0}		125000		F0IO cycle time
t_{DL}	3.92	5.04	7.54	C4IO ⎿ to C2O ⎿ delay
t_{DH}	3.87	5.07	7.69	C4IO ⎿ to C2O ⌐ delay
t_{CFS}	15	t_C		F0IO ⌐ to C4IO ⎿ setup time
t_{CFHmin}	15	t_C		F0IO ⎿ to C4IO ⎿ hold time
t_{CFHmax}	15	t_C	100000	F0IO high time after start of frame
t_{CFD}	15	t_C		C4IO ⎿ to F0IO ⌐ delay
t_{STiS}	10			Data valid to C4IO ⎿ setup time
t_{STiH}	10			Data valid to C4IO ⎿ hold time
t_{SToD}	2		10	STIO output delay from C4IO ⎿

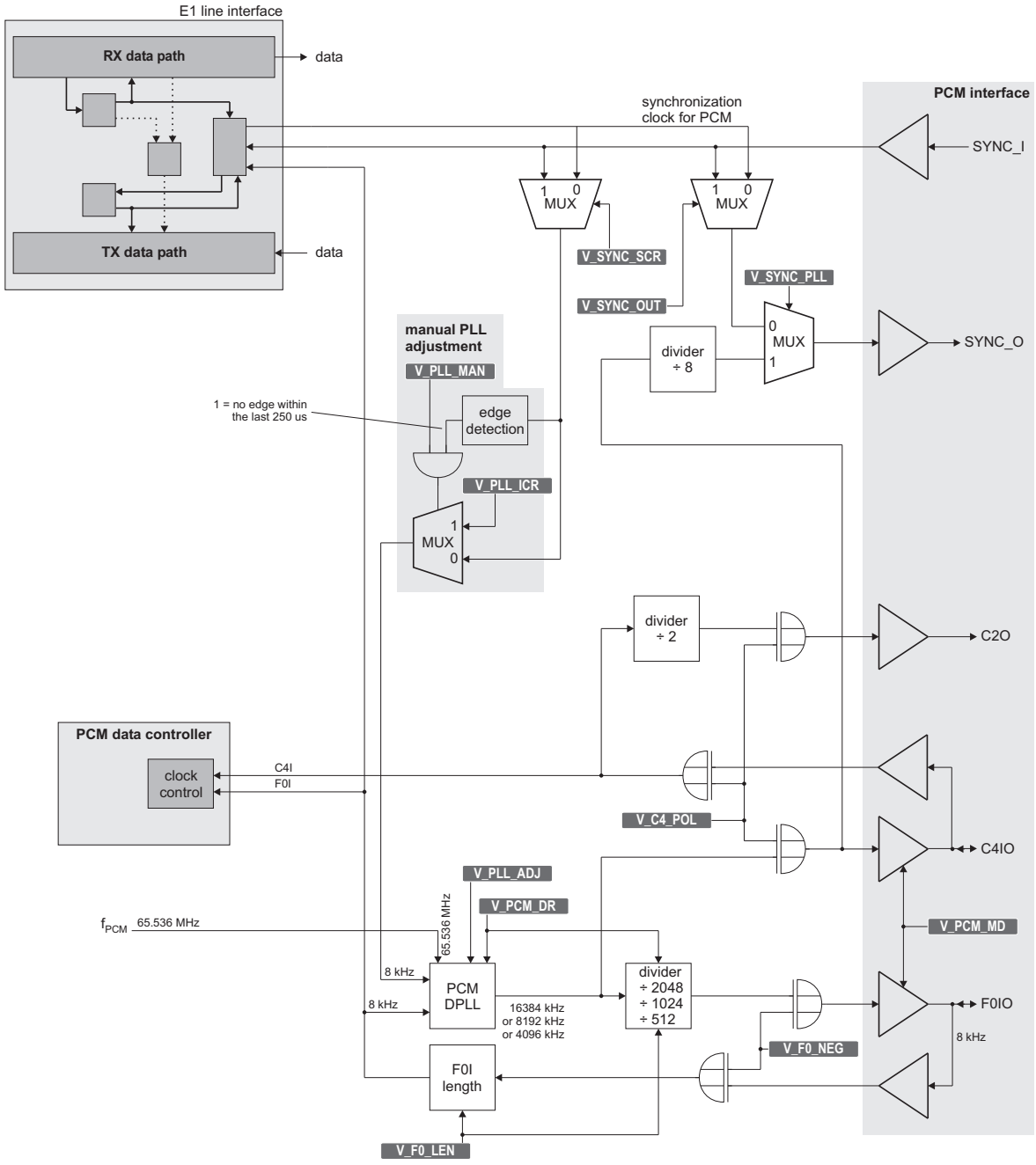


Figure 6.5: *PCM clock synchronization (see Figure 5.1 on page 183 for details on the E1 interface module)*

6.5.3 Manual PLL adjustment

In normal operation mode, the synchronization input signal is passed from the V_SYNC_SRC controlled multiplexer to the PCM DPLL as shown in Figure 6.5. For this V_PLL_MAN has to be '0' in register R_PCM_MD2.

The PLL output frequency can manually be adjusted if no synchronization source is available. This software controlled PLL adjustment is enabled with V_PLL_MAN = '1'. The V_SYNC_SRC controlled multiplexer must feed a 8 kHz signal in any case.

The time of the signal edges can be increased or reduced in the last time slot of the PCM frame. V_PLL_ICR = '0' results in a frequency reduction while V_PLL_ICR = '1' leads to a frequency increase. The number of adjusted edges is specified in the range $1..4$ with bitmap V_PLL_ADJ in register R_PCM_MD1.

V_PLL_MAN must be set back to '0' to stop the frequency regulation of the synchronization input signal.

6.5.4 C2O generation

The C2O output signal is derived from C4IO by a frequency divider. In fact, there is an additional building block (not shown in Figure 6.5) which ensures a specific phase relation at the start of a frame. According to the timing diagrams in Figures 6.3 and 6.4, C2O has its rising edge at the start of a frame. From this follows that C4IO has a falliung edge with every edge of C2O .

6.6 External CODECs

External CODECs can be connected to the HFC-E1 PCM interface. There are two ways of programming the PCM–CODEC–interconnection. First, a set of eight CODEC enable lines allow to connect up to eight external CODECs to HFC-E1. The second way uses the current time slot number that must be decoded to a CODEC's select signal. Then up to 128 external CODECs can be connected to HFC-E1. The choice of these connectivities is done with V_CODEC_MD in register R_PCM_MD1.

6.6.1 CODEC select via enable lines

HFC-E1 has eight CODEC enable signals F1_7 .. F1_0 . Every external CODEC has to be assigned to a PCM time slot via the bitmaps V_SL_SEL7 .. V_SL_SEL0 of the registers R_SL_SEL7 .. R_SL_SEL0.

Two shape signals can be programmed. The last bit determines the inactive level by which non-inverted and inverted shape signals can be programmed. Every external CODEC can choose one of the two shape signals with the bits V_SH_SEL7 .. V_SH_SEL0 of the registers R_SL_SEL7 .. R_SL_SEL0.

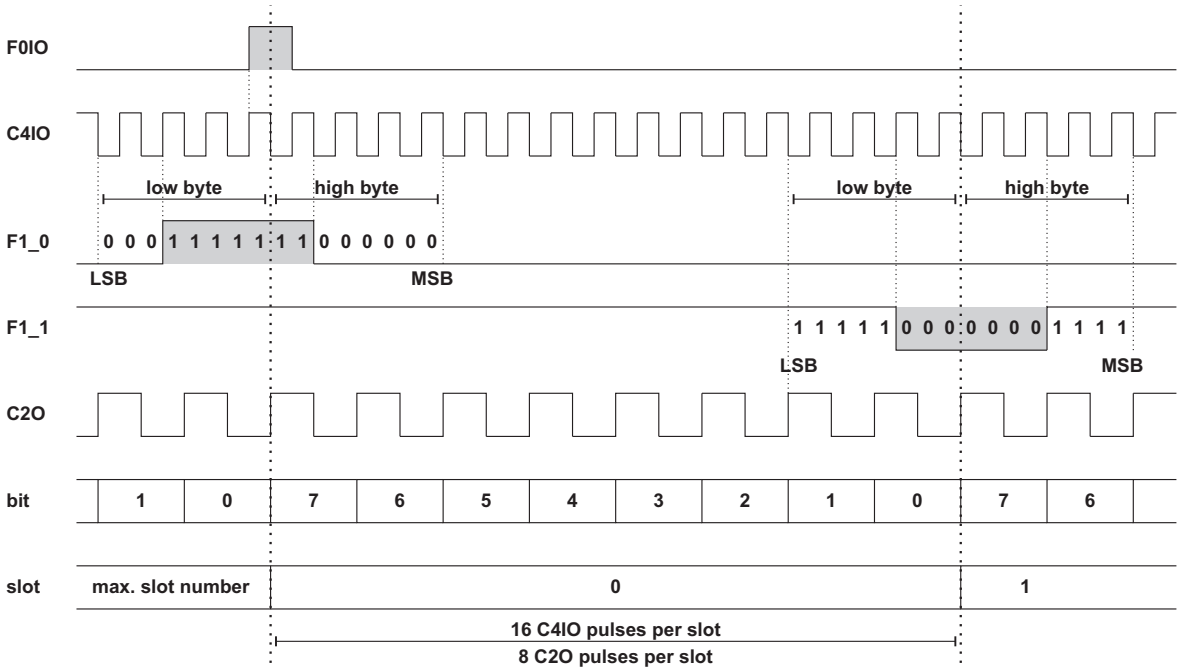


Figure 6.6: *Example for two CODEC enable signal shapes*

Figure 6.6 shows an example with two external CODECs with F1_0 and F1_1 enable signals. Time slot 0 starts with the F0IO pulse. In this example – assuming that PCM30 is configured – F1_0 enables the first CODEC on time slot 0 and shape bytes on R_SH0L and R_SH0H with the following register settings.

Register setup:

R_PCM_MD0	: V_PCM_IDX = 0		(R_SL_SEL0 register accessible)
R_SL_SEL0	: V_SL_SEL0 = 0x1F		(time slot #0)
	: V_SH_SEL0 = 0		(shape bytes R_SH0L and R_SH0H)

The second CODEC on time slot 1 and shape bytes on R_SH1L and R_SH1H must be configured as shown below.

Register setup:

R_PCM_MD0	: V_PCM_IDX = 1		(R_SL_SEL1 register accessible)
R_SL_SEL1	: V_SL_SEL1 = 0		(time slot #1)
	: V_SH_SEL1 = 1		(shape bytes R_SH1L and R_SH1H)

The shown shape signals have to be programmed in reverse bit order by the following register settings.

Register setup:

R_PCM_MD0	: V_PCM_IDX = 0xC		(R_SH0L register accessible)
R_SH0L	: V_SH0L = 0xF8		(0xF8 = '11111000' $\stackrel{reverse}{\longrightarrow}$ '00011111')
R_PCM_MD0	: V_PCM_IDX = 0xD		(R_SH0H register accessible)
R_SH0H	: V_SH0H = 0x03		(0x03 = '00000011' $\stackrel{reverse}{\longrightarrow}$ '11000000')
R_PCM_MD0	: V_PCM_IDX = 0xE		(R_SH1L register accessible)
R_SH1L	: V_SH1L = 0x1F		(0x1F = '00011111' $\stackrel{reverse}{\longrightarrow}$ '11111000')
R_PCM_MD0	: V_PCM_IDX = 0xF		(R_SH1H register accessible)
R_SH1H	: V_SH1H = 0xF0		(0xF0 = '11110000' $\stackrel{reverse}{\longrightarrow}$ '00001111')

6.6.2 CODEC select via time slot number

Alternatively, external CODECs can be enabled by decoding the time slot number. In this case, two programmable shape signals SHAPE0 and SHAPE1 are put out as second pin functions of pins 114 and 113 with every time slot. These signals can be used to mask the desired slot number. The current time slot number is issued on pins F_Q6 .. F_Q0 .

The shape signals can be programmed. The example in Figure 6.7 shows shape signals that are programmed in the same way as shown above (see Section 6.6.1).

F_Q6 .. F_Q0 must be decoded externally to generate CODEC select signals in dependence on the PCM time slot.

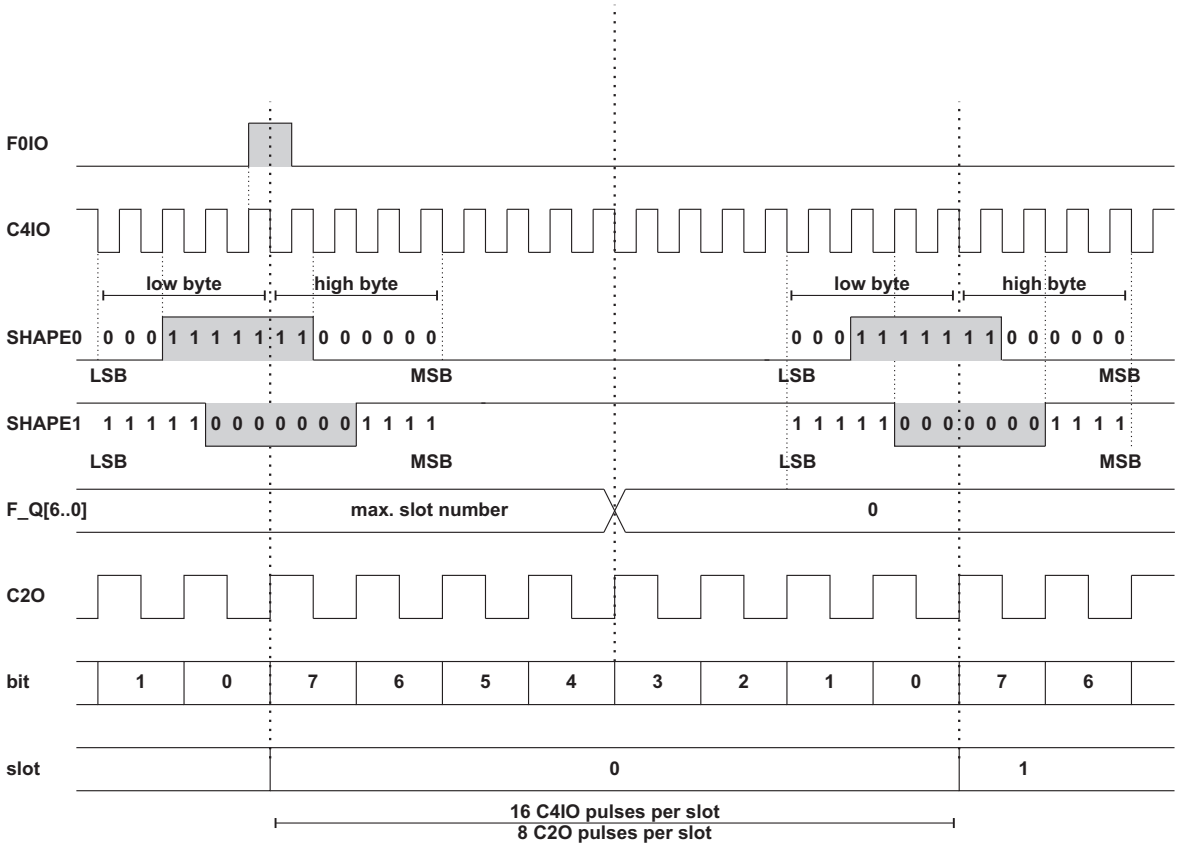


Figure 6.7: *Example for two CODEC enable signal shapes with SHAPE0 and SHAPE1 .*

6.7 Register description

6.7.1 Write only registers

R_SLOT	(w)	0x10

PCM time slot selection

This register is used to select a PCM time slot. Before a PCM slot array register can be accessed, this index register must specify the desired slot number and data direction. Depending on the V_PCM_DR value in register R_PCM_MD1, either 32, 64 or 128 time slots are available for each data direction.

Bits	Reset value	Name	Description
0	0	**V_SL_DIR**	**PCM time slot data direction** '0' = transmit PCM data '1' = receive PCM data
7..1	0x00	**V_SL_NUM**	**PCM time slot number**

R_PCM_MD0	(w)	0x14

PCM mode, register 0

Bits	Reset value	Name	Description
0	0	**V_PCM_MD**	**PCM bus mode** '0' = slave (pins C4IO and F0IO are inputs) '1' = master (pins C4IO and F0IO are outputs) If no external C4IO and F0IO signal is provided this bit must be set for operation.
1	0	**V_C4_POL**	**Polarity of C4IO clock** '0' = pin F0IO is sampled on negative clock transition of C4IO '1' = pin F0IO is sampled on positive clock transition of C4IO
2	0	**V_F0_NEG**	**Polarity of F0IO signal** '0' = positive pulse '1' = negative pulse
3	0	**V_F0_LEN**	**Duration of F0IO signal** '0' = active for one C4IO clock (244 ns at 4 MHz) '1' = active for two C4IO clocks (488 ns at 4 MHz) The specified signal duration is generated in PCM master mode and it is expected in PCM slave mode.
7..4	0	**V_PCM_IDX**	**Index value to select the register at address 15** At address 15 a so-called multi-register is accessible. 0 = R_SL_SEL0 register accessible 1 = R_SL_SEL1 register accessible 2 = R_SL_SEL2 register accessible 3 = R_SL_SEL3 register accessible 4 = R_SL_SEL4 register accessible 5 = R_SL_SEL5 register accessible 6 = R_SL_SEL6 register accessible 7 = R_SL_SEL7 register accessible 9 = R_PCM_MD1 register accessible 0xA = R_PCM_MD2 register accessible 0xC = R_SH0L register accessible 0xD = R_SH0H register accessible 0xE = R_SH1L register accessible 0xF = R_SH1H register accessible

R_SL_SEL0	(w)	0x15

Slot selection register for pin F1_0

This multi-register is selected with bitmap V_PCM_IDX = 0 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_0 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL0**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_0 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL0**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

Important !

For selecting slot 0 the value that has to be written into bitmap V_SL_SEL0 . . V_SL_SEL7 of register R_SL_SEL0 . . R_SL_SEL7 depends on the PCM data rate:

PCM data rate	Value
PCM30	0x1F
PCM64	0x3F
PCM128	0x7F

Please note that time slot 0 for PCM128 can only be used with V_SH_SEL0 . . V_SH_SEL7 = '0' (SHAPE 0) in registers R_SL_SEL0 . . R_SL_SEL7.

R_SL_SEL1	(w)	0x15

Slot selection register for pin F1_1

This multi-register is selected with bitmap V_PCM_IDX = 1 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_1 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL1**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_1 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL1**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL2	(w)	0x15

Slot selection register for pin F1_2

This multi-register is selected with bitmap V_PCM_IDX = 2 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_2 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL2**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_2 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL2**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL3	(w)	0x15

Slot selection register for pin F1_3

This multi-register is selected with bitmap V_PCM_IDX = 3 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_3 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL3**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_3 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL3**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL4	(w)	0x15

Slot selection register for pin F1_4

This multi-register is selected with bitmap V_PCM_IDX = 4 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_4 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL4**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_4 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL4**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL5	(w)	0x15

Slot selection register for pin F1_5

This multi-register is selected with bitmap V_PCM_IDX = 5 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_5 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL5**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_5 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL5**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL6	(w)	0x15

Slot selection register for pin F1_6

This multi-register is selected with bitmap V_PCM_IDX = 6 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_6 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL6**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_6 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL6**	**Shape selection** '0' = use shape 1 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_SL_SEL7	(w)	0x15

Slot selection register for pin F1_7

This multi-register is selected with bitmap V_PCM_IDX = 7 in register R_PCM_MD0.

Note: By setting all 8 bits to '1' pin F1_7 is disabled.

Bits	Reset value	Name	Description
6..0	0x7F	**V_SL_SEL7**	**PCM time slot selection** The selected slot number is V_SL_SEL1 +1 for F1_7 . Slot number 0 is selected with the maximum slot number of the selected PCM speed.
7	1	**V_SH_SEL7**	**Shape selection** '0' = use shape 0 set by registers R_SH0L and R_SH0H '1' = use shape 1 set by registers R_SH1L and R_SH1H

R_PCM_MD1	(w)	0x15

PCM mode, register 1

This multi-register is selected with bitmap V_PCM_IDX = 9 in register R_PCM_MD0.

Bits	Reset value	Name	Description
0	0	**V_CODEC_MD**	**CODEC enable signal selection** '0' = CODEC enable signals on F1_0 .. F1_7 '1' = SHAPE 0 pulse on pin SHAPE0 , SHAPE 1 pulse on pin SHAPE1 and CODEC count on F_Q0 .. F_Q6 for up to 128 external CODECs.
1	0	(reserved)	Must be '0'.
3..2	0	**V_PLL_ADJ**	**DPLL adjust speed** '00' = C4IO clock is adjusted in the last time slot of the PCM frame 4 times by one half clock cycle of system clock '01' = C4IO clock is adjusted in the last time slot of the PCM frame 3 times by one half clock cycle of system clock '10' = C4IO clock is adjusted in the last time slot of the PCM frame twice by one half clock cycle of system clock '11' = C4IO clock is adjusted in the last time slot of the PCM frame once by one half clock cycle of system clock
5..4	0	**V_PCM_DR**	**PCM data rate** '00' = 2 MBit/s (C4IO is 4.096 MHz, 32 time slots) '01' = 4 MBit/s (C4IO is 8.192 MHz, 64 time slots) '10' = 8 MBit/s (C4IO is 16.384 MHz, 128 time slots) '11' = unused Every time slot exists in transmit and receive data direction.
6	0	**V_PCM_LOOP**	**PCM test loop** When this bit is set, the PCM output data is looped to the PCM input data internally for all PCM time slots.
7		(reserved)	Must be '0'.

R_PCM_MD2	(w)	0x15

PCM mode, register 2

This multi-register is selected with bitmap V_PCM_IDX = 0xA in register R_PCM_MD0.

Bits	Reset value	Name	Description
0		(reserved)	Must be '0'.
1	0	**V_SYNC_PLL**	**SYNC_O with internal PLL output** '0' = V_SYNC_OUT is used for synchronization selection '1' = SYNC_O has a frequency of the PCM PLL output signal C4O divided by 8 (512 kHz, 1024 kHz or 2048 kHz depending on the PCM data rate)
2	0	**V_SYNC_SRC**	**PCM PLL synchronization source selection** '0' = E1 interface (see R_SYNC_CTRL for further synchronization configuration) '1' = SYNC_I input (8 kHz)
3	0	**V_SYNC_OUT**	**SYNC_O signal source selection** This bit is only used if V_SYNC_PLL = '0'. '0' = E1 interface '1' = SYNC_I is connected to SYNC_O
5..4		(reserved)	Must be '00'.
6	0	**V_PLL_ICR**	**Increase PCM frame time** This bit is only valid if V_PLL_MAN is set. '0' = PCM frame time is reduced as selected by bitmap V_PLL_ADJ in register R_PCM_MD1 '1' = PCM frame time is increased as selected by bitmap V_PLL_ADJ in register R_PCM_MD1
7	0	**V_PLL_MAN**	**Manual PLL adjustment** '0' = PCM PLL is automatically adjusted to the SYNC_I signal or to the line interface synchronization pulse (depending on V_SYNC_OUT setting) '1' = PCM PLL is manually adjusted according to V_PLL_ICR. This can be used to make synchronization by software if no synchronization source is available. **Note:** Manual PLL adjustment is automatically disabled when a synchronization pulse is available.

R_SH0L (w) **0x15**

CODEC enable signal SHAPE0 , low byte

This multi-register is selected with bitmap V_PCM_IDX = 0xC in register R_PCM_MD0.

Bits	Reset value	Name	Description
7..0	0	**V_SH0L**	**Shape bits 7..0** Every bit is used for 1/2 C4IO clock cycle.

R_SH0H (w) **0x15**

CODEC enable signal SHAPE0 , high byte

This multi-register is selected with bitmap V_PCM_IDX = 0xD in register R_PCM_MD0.

Bits	Reset value	Name	Description
7..0	0	**V_SH0H**	**Shape bits 15..8** Every bit is used for 1/2 C4IO clock cycle. Bit 7 of V_SH0H defines the value for the rest of the period.

R_SH1L (w) **0x15**

CODEC enable signal SHAPE1 , low byte

This multi-register is selected with bitmap V_PCM_IDX = 0xE in register R_PCM_MD0.

Bits	Reset value	Name	Description
7..0	0	**V_SH1L**	**Shape bits 7..0** Every bit is used for 1/2 C4IO clock cycle.

R_SH1H	(w)	0x15

CODEC enable signal SHAPE1 , high byte

This multi-register is selected with bitmap V_PCM_IDX = 0xF in register R_PCM_MD0.

Bits	Reset value	Name	Description
7..0	0	**V_SH1H**	**Shape bits 15..8** Every bit is used for 1/2 C4IO clock cycle. Bit 7 of V_SH1H defines the value for the rest of the period.

A_SL_CFG [SLOT]	(w)	0xD0

HFC-channel assignment for the selected PCM time slot and PCM output buffer configuration

With this register a HFC-channel can be assigned to the selected PCM time slot. Additionally, the PCM buffers can be configured.

Before writing this array register the PCM time slot must be selected by register R_SLOT.

Bits	Reset value	Name	Description
0	0	**V_CH_SDIR**	**HFC-channel data direction** '0' = HFC-channel for transmit data '1' = HFC-channel for receive data
5..1	0	**V_CH_SNUM**	**HFC-channel number** (0..31)
7..6	0	**V_ROUT**	**PCM data flow configuration** For transmit time slots: '00' = data transmission from HFC-channel disabled, output buffers disabled '01' = loop PCM data internally, output buffers disabled '10' = output buffer for STIO1 enabled '11' = output buffer for STIO2 enabled For receive time slots: '00' = data transmission to HFC-channel disabled, input data is ignored '01' = loop PCM data internally '10' = receive data from STIO2 '11' = receive data from STIO1

(See Figure 6.1 on page 232 for detailed information).

6.7.2 Read only registers

R_F0_CNTL	(r)	0x18

F0IO pulse counter, low byte

This register is the low byte of a 16 bit ring counter. The high byte can be read from register R_F0_CNTH. This register is incremented every 125 µs.

Bits	Reset value	Name	Description
7..0	0x00	**V_F0_CNTL**	**Bits [7..0] of the F0IO counter** This register should be read first to latch the value of register R_F0_CNTH.

R_F0_CNTH	(r)	0x19

F0IO pulse counter, high byte

This register is the high byte of a 16 bit ring counter. The low byte can be read from register R_F0_CNTL. This register is incremented every 32 ms.

Bits	Reset value	Name	Description
7..0	0	**V_F0_CNTH**	**Bits [15..8] of the F0IO counter** The low byte should be read first (see register R_F0_CNTL)

Chapter 7

Pulse width modulation (PWM) outputs

Table 7.1: *Overview of the HFC-E1 PWM pins*

Number	Name	Description
95	PWM1	Pulse Width Modulator Output 1
96	PWM0	Pulse Width Modulator Output 0

Table 7.2: *Overview of the HFC-E1 PWM registers*

Address	Name	Page
0x38	R_PWM0	259
0x39	R_PWM1	259
0x46	R_PWM_MD	260

7.1 Overview

HFC-E1 has two PWM output lines PWM0 and PWM1 with programmable output characteristic.

The output lines can be configured as open drain, open source and push / pull by setting V_PWM0_MD respectively V_PWM1_MD in register R_PWM_MD.

7.2 Standard PWM usage

The duty cycle of the output signals can be set in registers R_PWM0 and R_PWM1. The register value defines the number of clock periods where the output signal is high during the cycle time

$$T = \frac{256}{f_{SYS}} = 256 \cdot 40.69\,\text{ns} = 10.42\,\mu\text{s}$$

for the normal system clock $f_{SYS} = 24.576\,\text{MHz}$.

The variable duty cycle

$$\frac{t_{high}}{t_{low}} = \frac{i}{256 - i}\ , i = \text{value of R\_PWM1 or R\_PWM1}$$

of the PWM output pins can be set from 1:255 to 255:1. The register value 0 generates an output signal which is permanently low. The duty cycle 1:1 is achieved with $i = 128$.

There are always i pulses within the period T. Each pulse-width is a multiple $1/f_{SYS}$. Due to the setup values, different pulse-width might occur. Pulses with longer or shorter width than the mostly appearing width are distributed through the whole period.

The ouput signal of the PWM unit can be used for analog settings by using an external RC filter which generates a voltage that can be adapted by changing the PWM register value.

7.3 Alternative PWM usage

The PWM output lines can be programmed to generate a 16 kHz signal. This signal can be used as analog metering pulse for POTS interfaces. Each PWM output line can be switched to 16 kHz signal by setting V_PWM0_16KHZ or V_PWM1_16KHZ in register R_RAM_MISC. In this case the output characteristic is also determined by the R_PWM_MD register settings.

7.4 Register description

R_PWM0	(w)	0x38

Modulator register for pin PWM0

Bits	Reset value	Name	Description
7..0	0	V_PWM0	**PWM duty cycle** The value specifies the number of clock periods where the output signal of PWM0 is high during a 256 clock periods cycle, e.g. 0x00 = no pulse, always low 0x80 = 1/1 duty cycle 0xFF = 1 clock period low after 255 clock periods high

R_PWM1	(w)	0x39

Modulator register for pin PWM1

Bits	Reset value	Name	Description
7..0	0	V_PWM1	**PWM duty cycle** The value specifies the number of clock periods where the output signal of PWM1 is high during a 256 clock periods cycle, e.g. 0x00 = no pulse, always low 0x80 = 1/1 duty cycle 0xFF = 1 clock period low after 255 clock periods high

R_PWM_MD	(w)	0x46

PWM output mode register

Bits	Reset value	Name	Description
2..0	0	(reserved)	Must be '000'.
3	0	**V_EXT_IRQ_EN**	**External interrupt enable** '0' = normal operation '1' = external interrupt from GPI24 .. GPI31 enable **Note:** The GPI pins must be connected to a pull-up resistor to VDD . Any low input signal on one of the lines will generate an external interrupt.
5..4	0	**V_PWM0_MD**	**Output buffer configuration for pin PWM0** '00' =PWM output tristate (disable) '01' = PWM push / pull output '10' = PWM push to 0 only '11' = PWM pull to 1 only
7..6	0	**V_PWM1_MD**	**Output buffer configuration for pin PWM1** '00' = PWM output tristate (disable) '01' = PWM push / pull output '10' = PWM push to 0 only '11' = PWM pull to 1 only

Chapter 8

Multiparty audio conferences

Table 8.1: *Overview of the HFC-E1 conference registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x18	R_CONF_EN	268	0x14	R_CONF_OFLOW	270
0xD1	A_CONF	269			

8.1 Conference unit description

HFC-E1 has a built in conference unit which allows up to 8 conferences with an arbitrary number of members each. The conference unit is located in the data stream going out to the PCM interface. So the normal outgoing data is replaced by the conference data. The number of conference members that can be combined to one conference is limited by the number of receive HFC-channels. Thus the total number of conference members is 32. Each PCM time slot can only be part of one conference.

All PCM values combined to a conference are added in one 125 µs time intervall. Then for every conference member the added value for this member is substracted so that every member of a conference hears all the others but not himself. This is done on a alternating buffer scheme for every 125 µs time intervall.

To enable the conference unit, bit V_CONF_EN in register R_CONF_EN must be set. If this is done, there are additional accesses to the SRAM of HFC-E1 which reduces performance of the on-chip processor on the other hand. Thus conference cannot be used with 8 Mbit/s PCM data rate where 128 slots are used, except the chip operates in double clock mode (see Chapter 12.1 on page 292). When the conference unit is switched on or off during data processing, erroneous data may be transmitted during the 125 µs cycle in progress.

To add a PCM time slot to a conference the slot number must be written into register R_SLOT. If the time slot has not yet been linked to an HFC-channel this can be done by writing the HFC-channel number and the channel's source / destination (input / output pins) into A_SL_CFG register. Afterwards the conference number must be written into bitmap V_CONF_NUM of register A_CONF. Additionally, the conference must be enabled for this time slot with V_CONF_SL set to '1' in the same register. Noise suppression, threshold and input attenuation level can be configured independently for each time slot.

> **Important !**
>
> Register A_CONF must be initialized for every time slot. There is no specific default value after reset. Time slots which are not member of a conference must have A_CONF = 0x00.
>
> The conference unit should never be enabled before all registers A_CONF[SLOT] are initialized.

To remove a time slot from a conference the time slot must be selected by writing its number into register R_SLOT. Then 0x00 must be written into register A_CONF.

8.2 Overflow handling

The data summation of those HFC-channels that are involved in a conference, can cause signal overflows. The conference unit internally works with signed 16 bit words. In case of an overflow the amplitude value is limited to the maximum amplitude value.

Overflow conditions can be checked with register R_CONF_OFLOW. Every bit of this register indicates that an overflow has occured in one of the eight corresponding conferences.

The more conference members are involved in a conference, the higher is the probability of signal overflows. In this case the signal attenuation can be reduced by bitmap V_ATT_LEV in register

A_CONF. This can be done on-the-fly to improve the signal quality of a conference.

8.3 Conference including the E1 interface

As the conference unit is located in the PCM transmit data path, some additional explanations for conference members on the E1 interface have to be made.

Conference members can also be time slots of the E1 interface. In this case, a pair of transmit / receive PCM time slots have to be configured to loop back the data.

In detail, the conference signal received by the E1 gets assigned to a transmit HFC-channel, this gets assigned to a transmit PCM time slot where it passes the summing circuit. The PCM interface loopes-back the signal to a receive time slot. This is asigned to a receive HFC-channel which finally passes the signal to a transmit E1 time slot.

The data transmission on both the receive and transmit HFC-channels require one transmit and one receive FIFO to be enabled, although the FIFOs are not used to store data (see Section 3.4).

8.4 Conference setup example for CSM

The following example shows the register settings for a conference with three members. Two members are located on the PCM interface side while the other one is located on the E1 interface side. The example uses conference number 2. It is specified in Table 8.2.

Table 8.2: *Conference example specification*

Conference member		Connection		
❶	1st PCM member :	PCM slot[5,RX]	→	HFC-channel[31,RX]
	:	PCM slot[5,TX]	←	HFC-channel[31,RX]
❷	2nd PCM member :	PCM slot[20,RX]	→	HFC-channel[27,RX]
	:	PCM slot[20,TX]	←	HFC-channel[27,RX]
❸	E1 member :	S/T interf. #1, RX B1	→	PCM slot[6,TX]
	:	S/T interf. #1, TX B1	←	PCM slot[6,RX]

Only two FIFOs are used in this example. Channel select mode should be selected to avoid unnecessary FIFO usage [1]. A PCM member allocates a single HFC-channel to establish the data loop via the switching buffer (see Fig. 3.4 and 3.4).

❶ **1st PCM conference member**

A PCM conference member can be looped over an arbitrary receive HFC-channel. In this example HFC-channel[31,TX] is used for the first PCM conference member. The conference is enabled only on the transmit time slot of the PCM interface.

[1] Remember that in *Simple Mode* FIFO numbers are equal to HFC-channel numbers. In the example four HFC-channels are enabled, so that in *Simple Mode* all FIFOs with the same number are blocked.

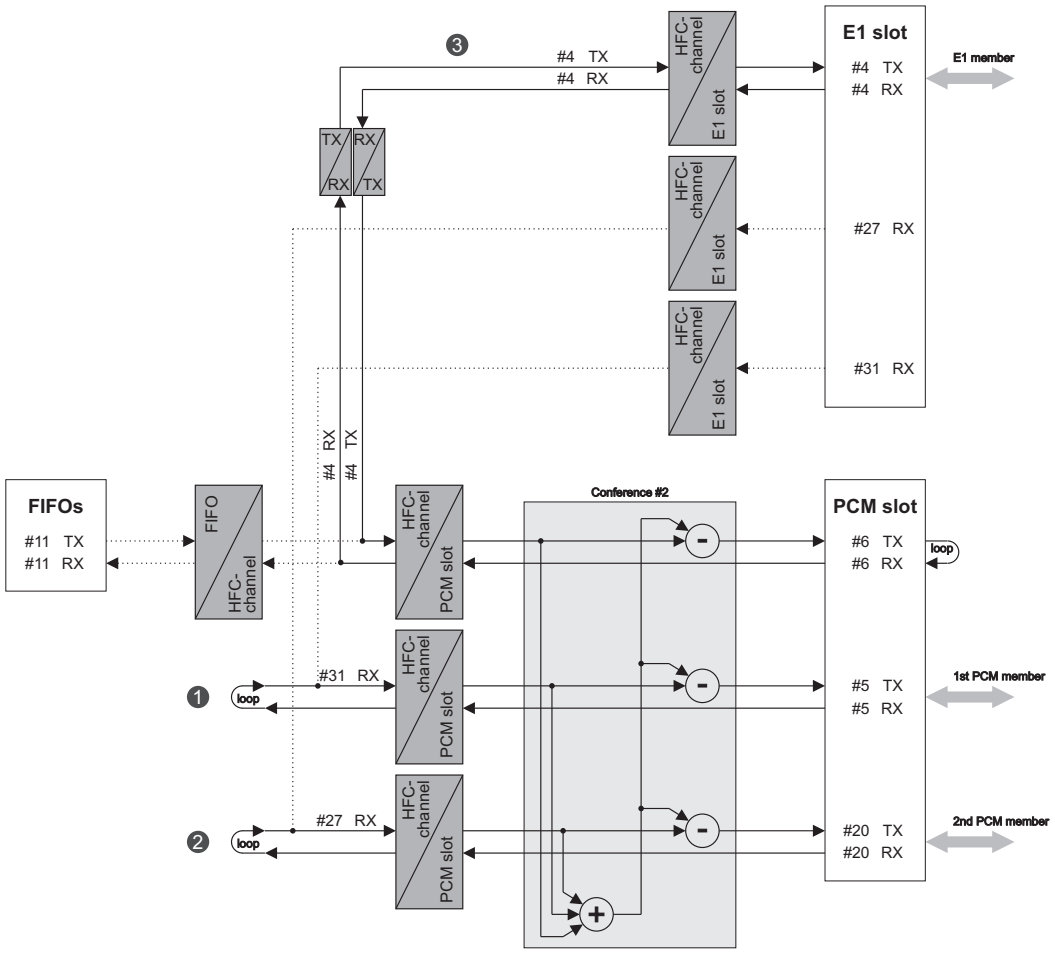


Figure 8.1: *Conference example*

Register setup:				(CSM ❶ 1st PCM conference member)
R_SLOT	: V_SL_DIR	=	1	(receive slot)
	: V_SL_NUM	=	5	(slot #5)
A_SL_CFG[5,RX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	31	(HFC-channel #31)
	: V_ROUT	=	'10'	(data from pin STIO2)
A_CONF[5,RX]	: V_CONF_NUM	=	0	(conference #0)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	0	(slot is not added to the conference)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	5	(slot #5)
A_SL_CFG[5,TX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	31	(HFC-channel #31)
	: V_ROUT	=	'10'	(data to pin STIO1)
A_CONF[5,TX]	: V_CONF_NUM	=	2	(conference #2)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	1	(slot is added to the conference)

❷ 2nd PCM conference member

The settings for the second PCM conference member is quite similar.

Register setup:				(CSM ❷ 2nd PCM conference member)
R_SLOT	: V_SL_DIR	=	1	(receive slot)
	: V_SL_NUM	=	20	(slot #20)
A_SL_CFG[20,RX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	27	(HFC-channel #27)
	: V_ROUT	=	'10'	(data from pin STIO2)
A_CONF[20,RX]	: V_CONF_NUM	=	0	(conference #0)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	0	(slot is not added to the conference)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	20	(slot #20)
A_SL_CFG[20,TX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	27	(HFC-channel #27)
	: V_ROUT	=	'10'	(data to pin STIO1)
A_CONF[20,TX]	: V_CONF_NUM	=	2	(conference #2)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	1	(slot is added to the conference)

❸ E1 conference member

Finally the E1 conference member must loop back its data via the PCM interface. This is normally done internally, i.e. the PCM output buffers are both disabled (see Chapter 6 for details). A pair of FIFOs is used to configure the PCM-to-E1 connection but no data is stored in these FIFOs.

Register setup:				(CSM ❸ E1 conference member)
R_FIFO	: V_FIFO_DIR	=	0	(transmit FIFO)
	: V_FIFO_NUM	=	11	(FIFO #11)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[11,TX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	1	(transparent mode)
	: V_TRP_IRQ	=	0	(interrupt disabled)
	: V_DATA_FLOW	=	'110'	(E1 → PCM)
A_CHANNEL[11,TX]	: V_CH_FDIR	=	0	(transmit HFC-channel)
	: V_CH_FNUM	=	4	(HFC-channel #4)
R_SLOT	: V_SL_DIR	=	0	(transmit slot)
	: V_SL_NUM	=	6	(slot #6)
A_SL_CFG[6,TX]	: V_CH_SDIR	=	0	(transmit HFC-channel)
	: V_CH_SNUM	=	4	(HFC-channel #4)
	: V_ROUT	=	'01'	(internal PCM data transmission)
A_CONF[6,TX]	: V_CONF_NUM	=	2	(conference #2)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	1	(slot is added to the conference)
R_FIFO	: V_FIFO_DIR	=	1	(receive FIFO)
	: V_FIFO_NUM	=	11	(FIFO #11)
	: V_REV	=	0	(normal bit order)
A_CON_HDLC[11,RX]	: V_IFF	=	0	(0x7E as inter frame fill)
	: V_HDLC_TRP	=	1	(transparent mode)
	: V_TRP_IRQ	=	0	(interrupt disabled)
	: V_DATA_FLOW	=	'110'	(FIFO ← E1, E1 ← PCM)
A_CHANNEL[11,RX]	: V_CH_FDIR	=	1	(receive HFC-channel)
	: V_CH_FNUM	=	4	(HFC-channel #4)
R_SLOT	: V_SL_DIR	=	1	(receive slot)
	: V_SL_NUM	=	6	(slot #6)
A_SL_CFG[6,RX]	: V_CH_SDIR	=	1	(receive HFC-channel)
	: V_CH_SNUM	=	4	(HFC-channel #4)
	: V_ROUT	=	'01'	(internal PCM data loop)
A_CONF[6,RX]	: V_CONF_NUM	=	0	(conference #0)
	: V_NOISE_SUPPR	=	0	(no noise suppression)
	: V_ATT_LEV	=	0	(0 dB attenuation level)
	: V_CONF_SL	=	0	(slot is not added to the conference)

- **Global conference enable**

  After the configuration procedure of settings ❶ to ❸, the conference unit can be switched on and the data coding can be chosen. Both is done by setting register R_CONF_EN.

Register setup:

R_CONF_EN	: V_CONF_EN	= '1'	(enable conference unit)
	: V_ULAW	= '1'	(µ-law data coding)

8.5 Register description

8.5.1 Write only registers

R_CONF_EN	(w)	0x18

Conference mode register

Bits	Reset value	Name	Description
0	0	**V_CONF_EN**	**Global conference enable** '0' = disable '1' = enable **Note:** When this bit is changed, erroneous data may be processed during the 125 µs cycle in progress.
6 . . 1		(reserved)	Must be '000000'.
7	0	**V_ULAW**	**Data coding of the conference unit** '0' = A-Law '1' = µ-Law

A_CONF [SLOT] **(w)** **0xD1**

Conference parameter register for the selected PCM time slot

Before writing this array register the PCM time slot must be selected by register R_SLOT.

Note: This register has no specific default value after reset.

Bits	Reset value	Name	Description
2..0		**V_CONF_NUM**	**Conference number** (0..7)
4..3		**V_NOISE_SUPPR**	**Noise suppression threshold** '00' = no noise suppression '01' = data values less or equal to 5 are set to 0 '10' = data values less or equal to 9 are set to 0 '11' = data values less or equal to 16 are set to 0
6..5		**V_ATT_LEV**	**Input attenuation level** '00' = 0 dB '01' = -3 dB '10' = -6 dB '11' = -9 dB
7		**V_CONF_SL**	**Conference enable for the selected PCM time slot** '0' = slot is not added to the conference '1' = slot is added to the conference

8.5.2 Read only registers

R_CONF_OFLOW	(r)	0x14

Conference overflow indication register

Specifies the conference numbers where an overflow has occured. Reading this register clears the bits.

Bits	Reset value	Name	Description
0	0	V_CONF_OFLOW0	Overflow occured in conference 0
1	0	V_CONF_OFLOW1	Overflow occured in conference 1
2	0	V_CONF_OFLOW2	Overflow occured in conference 2
3	0	V_CONF_OFLOW3	Overflow occured in conference 3
4	0	V_CONF_OFLOW4	Overflow occured in conference 4
5	0	V_CONF_OFLOW5	Overflow occured in conference 5
6	0	V_CONF_OFLOW6	Overflow occured in conference 6
7	0	V_CONF_OFLOW7	Overflow occured in conference 7

Chapter 9

DTMF controller

Table 9.1: *Overview of the HFC-E1 DTMF registers*

Write only registers:		
Address	**Name**	**Page**
0x1C	R_DTMF	279
0x1D	R_DTMF_N	280

9.1 Overview

HFC-E1 has an on-chip DTMF[1] detection machine. This contains the recursive part of the Goertzel filter while the non-recursive part has to be calculated by firmware.

The Goertzel filter is well known in literature. It describes a special form of the DFT algorithm on the base of an *infinite-duration impulse response* filter (IIR filter). The Goertzel algorithm calculates only a single spectral line. It is very fast and has low memory requirements.

9.2 DTMF calculation principles

The transmission of dialed numbers on analog lines is normaly done by DTMF (Dual Tone Multi-Frequency). This means that pairs of two frequencies are used to determine one key of a keypad like shown in Table 9.2.

Table 9.2: *DTMF tones on a 16-key keypad*

Keypad				Frequencies	
1	2	3	A	697	
4	5	6	B	770	low tones
7	8	9	C	852	(f/Hz)
*	0	#	D	941	
1209	1336	1477	1633	high tones (f/Hz)	

Thus there are 4 low tones and 4 high tones and therefore 16 combinations of 2 tones. Because the ISDN network has several interfaces to the old-fashioned POTS analog network, in-band number dialing with DTMF can take place. To decode this DTMF information, HFC-E1 has a built in DTMF detection engine.

The detection is done by the digital processing of the HFC-channel data by the so-called Goerzel Algorithm

$$W_n = K \cdot W_{n-1} - W_{n-2} + x \,, \tag{9.1}$$

where W_n is a DTMF coefficient calculated from the 2 previous coefficients W_{n-1} and W_{n-2}. The factor

$$K = 2\cos\left(2\pi \cdot \frac{f}{8000\,\mathrm{Hz}}\right)$$

is a constant for each frequency and x is a new HFC-channel value every 125 µs. The start condition is $W_{-1} = W_{-2} = 0$.

After processing equation (9.1) for $N+1$ times with $n = 0..N$ the real power amplitude

$$A^2 = W_N^2 + W_{N-1}^2 - K \cdot W_N \cdot W_{N-1} \,. \tag{9.2}$$

[1] DTMF: Dual tone multi-frequency

has to be calculated by the host processor.

The calculation of equation (9.1) is done for all 8 frequencies and for every new HFC-channel value (every 125 µs). Optionally also the second harmonic (double frequency) is also investigated. The K factors are values concerning to the DTMF frequencies. If the DTMF calculation is implemented in integer arithmetic, it is useful to multiply K with 2^{14} to exploit the whole 16 bit value range. These K values are listed in Table 9.3.

Table 9.3: *16-bit K factors for the DTMF calculation*

1st harmonic		2nd harmonic	
f/Hz	**K · 2^14**	**f/Hz**	**K · 2^14**
697	27 980	1406 *	14 739
770	26 956	1555 *	11 221
852	25 701	1704	7 549
941	24 219	1882	3 032
1209	19 073	2418	-10 565
1336	16 325	2672	-16 503
1477	13 085	2954	-22 318
1633	9 315	3266	-27 472

*: These frequencies are modified to achieve a better detection compared with the high fundamental tones.

9.3 DTMF controller implementation

The DTMF controller picks up the values of the HFC-channels and stores the temporary or final results in the internal or external RAM. Figure 9.1 shows how the DTMF controller is embedded between the HFC-channels of the PCM part and the internal or external RAM. K factors are read from the chip internal ROM.

After reset, the DTMF controller is disabled. It is to be enabled by setting bit V_DTMF_EN in register R_DTMF. This bit can be changed at any time.

W coefficients of all 32 transmit or receive channels can be calculated if only the first harmonic (i.e. 8 frequencies) are chosen with V_HARM_SEL = '0' in register R_DTMF. With V_HARM_SEL = '1', the second harmonics are calculated, too. Then the DTMF coefficients of only 16 HFC-channels are calculated, namely #0 . . #15 if V_CHBL_SEL = '0' or #16 . . #31 if V_CHBL_SEL = '1'. In any case either transmit or receive HFC-channels of the PCM part can be selected with V_DTMF_RX_CH in the same register.

The W coefficients are stored in the internal or external RAM after calculation. These are always 256 values; either from 8 frequencies of 32 HFC-channels or from 16 frequencies of 16 HFC-channels.

Equation 9.1 is calculated $N+1$ times to obtain the power amplitude (see Equation 9.2). Then, after $(N+1) \cdot 125\,\mu s$ the result can be read from the RAM. The value of N can be programmed with register R_DTMF_N. A good balance between the bandwith of the Goerzel filter and the length of the investigation is $N = 102$, e.g.

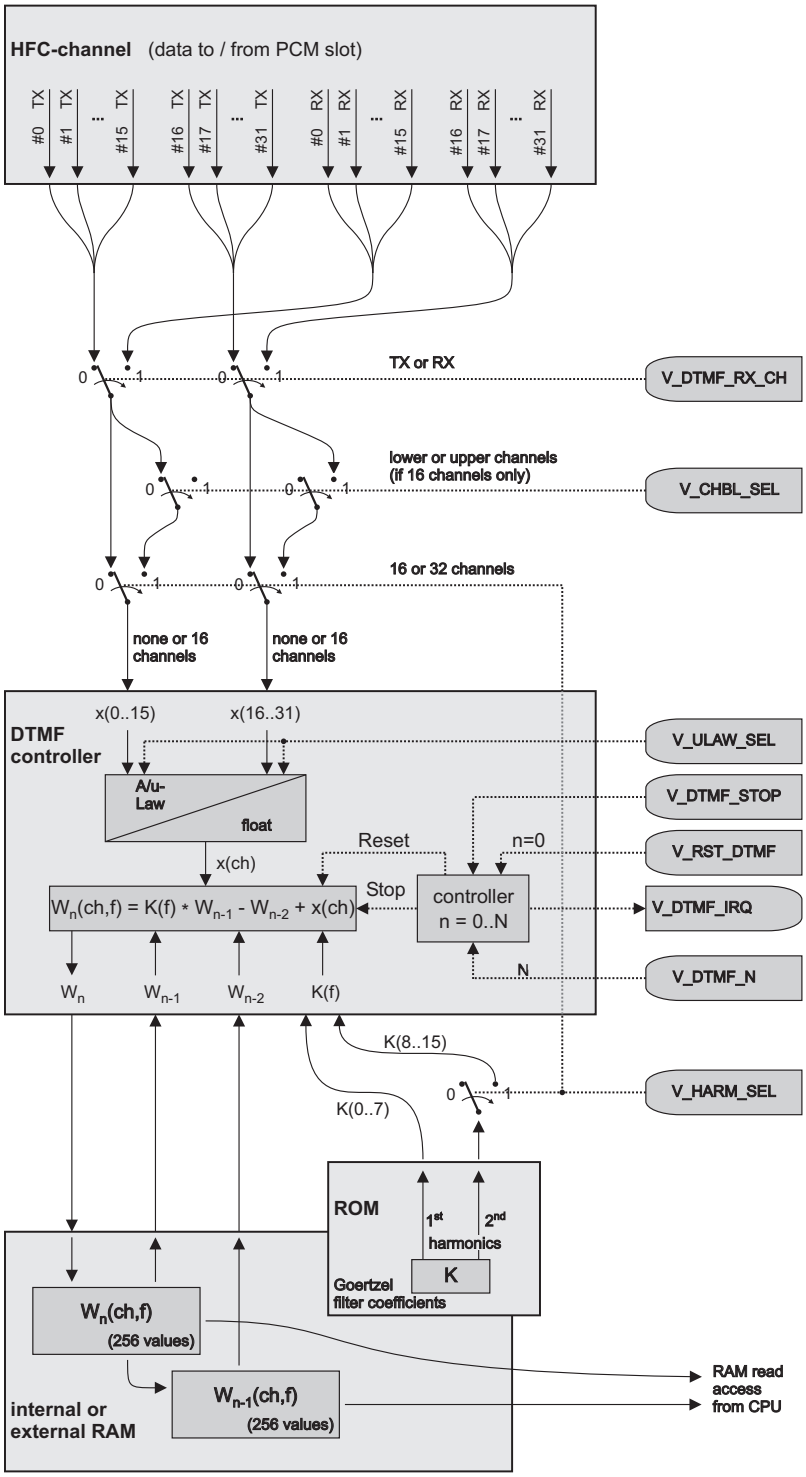


Figure 9.1: *DTMF controller block diagram*

When $n = N$, the V_DTMF_IRQ flag in register R_IRQ_MISC is set to alert the termination of the power amplitude calculation. This bit can either be polled by software or it can trigger an interrupt if the mask bit V_DTMF_IRQMSK in register R_IRQMSK_MISC is set.

Bit V_DTMF_STOP should be set to '1' in register R_DTMF to stop the DTMF engine after every calculation of W_N and W_{N-1}. The software has time to read the DTMF coefficients from the RAM. After this, a new calculation can be started with V_RST_DTMF is set to '1' in register R_DTMF. Depending on the time required from the software, some x values might not be considered by the next DTMF detection [2].

9.4 Data flow programming

As the DTMF controller operates on the PCM switching buffers shown in Figures 3.4 and 3.5 on pages 110 and 110, DTMF calculation requires a data path to a PCM switching buffer.

For PCM-to-E1 connections as well as for FIFO-to-E1 connections, no special data flow programming is required. V_DTMF_RX_CH has to be set up due to the data direction.

When DTMF calculation is desired for a FIFO-to-E1 connection with data transmitted from the line interface, the standard data flow programming V_DATA_FLOW = '000' can be used and V_DTMF_RX_CH = '0' has to be set up.

A special data flow programming value is required when DTMF calculation is desired for a FIFO-to-E1 connection with data received by the line interface. In this case V_DATA_FLOW[2] must be set to '1' to route data to the PCM switching buffer as well. So V_DATA_FLOW = '100' is recommended and V_DTMF_RX_CH = '1' has to be set up.

9.5 Access to DTMF coefficients

The host processor should read the two W_N and W_{N-1} 16-bit coefficients for 8 or 16 frequencies for the desired channels. The coefficients are located in the internal or external SRAM. The memory address is calculated by

$$\text{address} = \text{base address} + \text{frequency offset} + \text{channel offset} + \text{W-byte offset} \; . \tag{9.3}$$

The individual address components are shown in Table 9.4.

The DTMF coefficients have a special float format which increases the precision by some bits. They have 16 bit length and are coded *sign – exponent – mantissa* with a width of 1 – 3 – 12 bit.

The exponent is always as small as possible. In other words, an exponent value greater than zero requires the most significant mantissa bit to be '1'. Thus this bit has not to be coded. The following pseudocode shows how to convert float values to linear values:

```
// input: w_float (16 bit, word)
// output: w_linear (32 bit singned integer)

// mantissa and exponent extraction:
```

[2] It is *not* recommended to set up a continous DTMF detection with V_DTMF_STOP = '0'. This would require hard time constraints and a special software algorithm to read the W_N and W_{N-1} coefficients. Please ask the Cologne Chip support team if this procedure is desired anyway.

Table 9.4: *Memory address calculation for DTMF coefficients related to equation (9.3)*

base address	RAM size	address	RAM size	address
	32k	0x1000	128k	0x2000
			512k	0x2000

frequency offset	1st harmonic	offset	2nd harmonic	offset
(low tones)	697 Hz	0x00	1406 Hz	0x40
	770 Hz	0x80	1555 Hz	0xC0
	852 Hz	0x100	1704 Hz	0x140
	941 Hz	0x180	1882 Hz	0x1C0
(high tones)	1209 Hz	0x200	2418 Hz	0x240
	1336 Hz	0x280	2672 Hz	0x2C0
	1477 Hz	0x300	2954 Hz	0x340
	1633 Hz	0x380	3266 Hz	0x3C0

channel offset	number	offset	number	offset
	0	0x00	16	0x40
	1	0x04	17	0x44
	2	0x08	18	0x48
	3	0x0C	19	0x4C
	4	0x10	20	0x50
	5	0x14	21	0x54
	6	0x18	22	0x58
	7	0x1C	23	0x5C
	8	0x20	24	0x60
	9	0x24	25	0x64
	10	0x28	26	0x68
	11	0x2C	27	0x6C
	12	0x30	28	0x70
	13	0x34	29	0x74
	14	0x38	30	0x78
	15	0x3C	31	0x7C

W-byte offset	W_{N-1}	offset	W_N	offset
	low byte	0	low byte	2
	high byte	1	high byte	3

```
mantissa = w_float and 0xFFF; // bits[11..0]
exponent = (w_float shr 12) and 0x7; // bits[14..12]

// sign evaluation:
if w_float and 0x8000 <> 0 then
{
  mantissa = mantissa or 0xFFFFF000; // set bits[31..12]
}

// exponent evaluation:
if exponent <> 0 then
{
  mantissa = mantissa xor 0x1000; // restore bit[12]
  mantissa = mantissa shl (exponent-1);
}

// return result:
w_linear = mantissa;
```

9.6 DTMF tone detection

DTMF tones are detected by evaluation of the power amplitude A^2 (see Equation 9.2). This procedure is explained in [1] and is widely presented in literature with different approaches.

The discrimination process should consider the maximum power amplitude of a pair of low tone and high tone frequencies and the timing requirements given by the DTMF specification in ITU-T Q.24 [4]. Additionally, the second highest power amplitude can be investigated to ensure a sufficient distance to the maximum amplitude. In this way, the software can determine if a DTMF signal has been on the line or not. If a DTMF signal has been there, the tone pair is detected and so the dialed digit is decoded.

If potential DTMF tones are superimposed on arbitrary voice signal, it is helpfull to investigate not only the 8 DTMF tones but also their second harmonics. For DTMF tones the second harmonics should have no significant amplitude.

9.7 Register description

R_DTMF	(w)	0x1C

DTMF configuration register

Bits	Reset value	Name	Description
0	0	**V_DTMF_EN**	**Global DTMF enable** '0' = disable DTMF unit '1' = enable DTMF unit
1	0	**V_HARM_SEL**	**Harmonics selection** Investigation of the 2nd harmonics of the DTMF frequencies can be enabled to improve the detection algorithm. '0' = 8 frequencies in 32 channels (only 1st harmonics are processed) '1' = 16 frequencies in 16 channels (1st and 2nd harmonics are processed) **Note:** If 2nd harmonics are processed, only 16 HFC-channels can be considered (see V_CHBL_SEL).
2	0	**V_DTMF_RX_CH**	**DTMF data source** '0' = transmit HFC-channels are used for DTMF detection '1' = receive HFC-channels are used for DTMF detection
3	0	**V_DTMF_STOP**	**Stop DTMF unit** '0' = continuous DTMF processing '1' = DTMF processing stops after *n* processed samples
4	0	**V_CHBL_SEL**	**HFC-Channel block selection** HFC-Channel block selection (only if 16 channels are used, see V_HARM_SEL) '0' = lower 16 channels (0 . . 15) '1' = upper 16 channels (16 . . 31)
5		(reserved)	Must be '0'.
6	0	**V_RST_DTMF**	**Restart DTMF prosessing** '0' = no action '1' = enables new DTMF calculation phase after stop, automatically cleared
7	0	**V_ULAW_SEL**	**Data coding for DTMF detection** '0' = A-Law code '1' = µ-Law code

R_DTMF_N	(w)	0x1D

Number of samples

This register defines the number of samples which are calculated in the recursive part of the Goertzel filter.

Bits	Reset value	Name	Description
7..0	0	V_DTMF_N	**Number of samples** The recursive part of the Goertzel filter is looped V_DTMF_N +1 times ($n = 0..$ V_DTMF_N = 0..N) to calculate one pair of DTMF coefficients W_N and W_{N-1} (1 PCM value every 125 µs, i.e. 1 pair of DTMF coefficients every $(N+1) \cdot 125\,\mu s$). **Note:** V_DTMF_N must be specified before the DTMF unit is enabled in register R_DTMF.

Chapter 10

Bit Error Rate Test (BERT)

Table 10.1: *Overview of the HFC-E1 BERT registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x1B	R_BERT_WD_MD	286	0x17	R_BERT_STA	287
0xFF	A_IRQ_MSK	306	0x1A	R_BERT_ECL	287
			0x1B	R_BERT_ECH	288

10.1 BERT functionality

Bit Error Rate Test (BERT) is a very important test for communication lines. The bit error rate should be as low as possible. Increasing bit error rate is an early indication of a malfunction of components or the communication wire link itself.

HFC-E1 includes a high performance pseudo random bit generator (PRBG) and a pseudo random bit receiver with automatic synchronization capability. The error rate can be checked by the also implemented Bit Error counter (BERT counter).

> **Please note !**
>
> Transparent mode must be selected for *Bit Error Rate Test*.

10.2 BERT transmitter

The PRBG can be set to a variety of different pseudo random bit patterns. Continous '0', continous '1' or pseudo random bit patterns with one of 6 selectable sequence length's from $2^9 - 1$ bit to $2^{23} - 1$ bit can be configured with bitmap V_PAT_SEQ in register R_BERT_WD_MD. All bit sequences are defined in the ITU-T O.150 [6] and O.151 [5] specifications.

The BERT patterns are passed through the HFC-channel assigner if V_BERT_EN = '1' in register A_IRQ_MSK[FIFO]. For this reason, either a FIFO-to-E1 or a FIFO-to-PCM configuration must be selected. Furthermore, the allocated FIFO must be enabled to switch on the data path.

BERT patterns are generated if at least one FIFO has its bit V_BERT_EN set to '1'. When more than one transmit FIFO are using BERT patterns, all these patterns are generated from the same pseudo random generator. They are distrubuted to the FIFOs in the order of the FIFO processing sequence (see Section 3.2.3 on page 109).

Subchannel processing can be used together with the *Bit Error Rate Test*. Then the number of bits taken from the PRBG is V_BIT_CNT.

> **Please note !**
>
> To test a connection and the error detection capability of the BERT error counter, a BERT error can be generated on the receiver side of an E1 link. Setting bit V_BERT_ERR in register R_BERT_WD_MD generates one wrong BERT bit in the outgoing data stream. V_BERT_ERR is automatically cleared afterwards.

10.3 BERT receiver

The BERT receiver has an automatic synchonization capability. When the incoming bit stream is synchronized with the PRBG, bit V_BERT_SYNC in register R_BERT_STA is set to '1'.

A 16 bit BERT error counter is available in registers R_BERT_ECL and R_BERT_ECH. The low

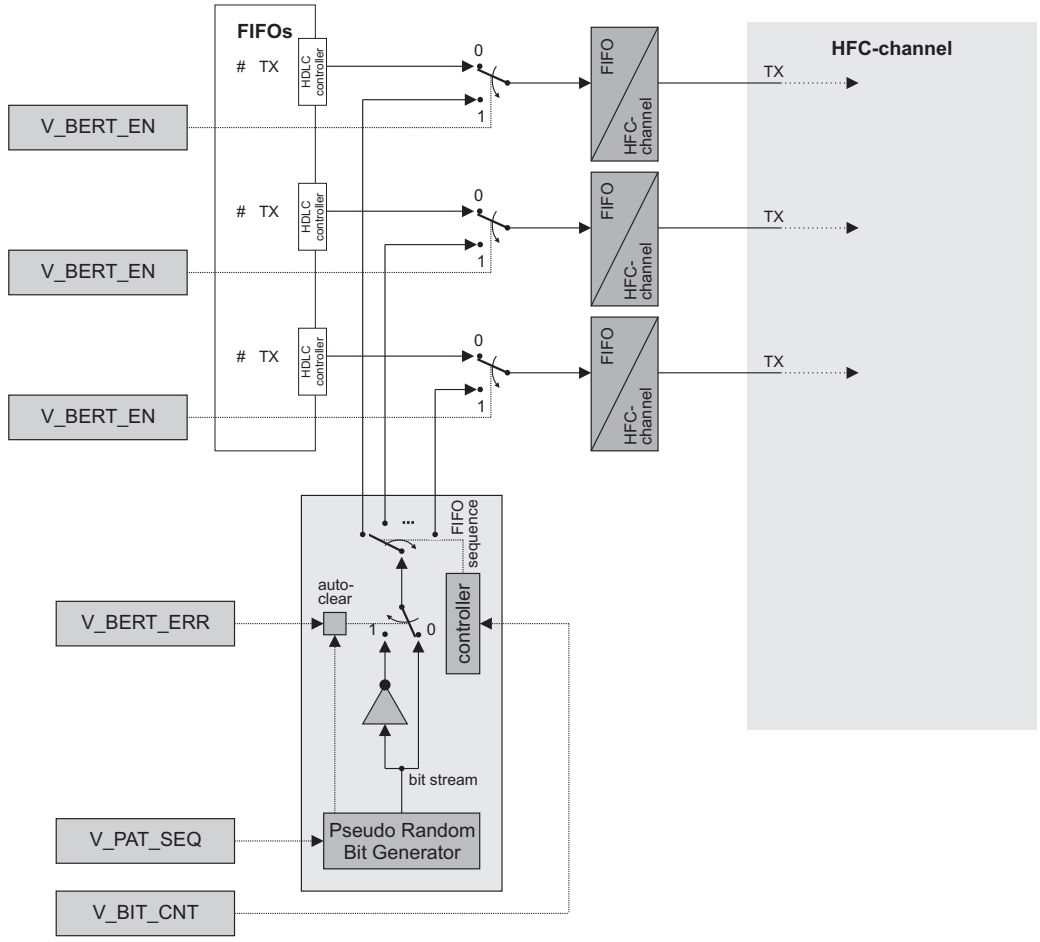


Figure 10.1: *BERT transmitter block diagram*

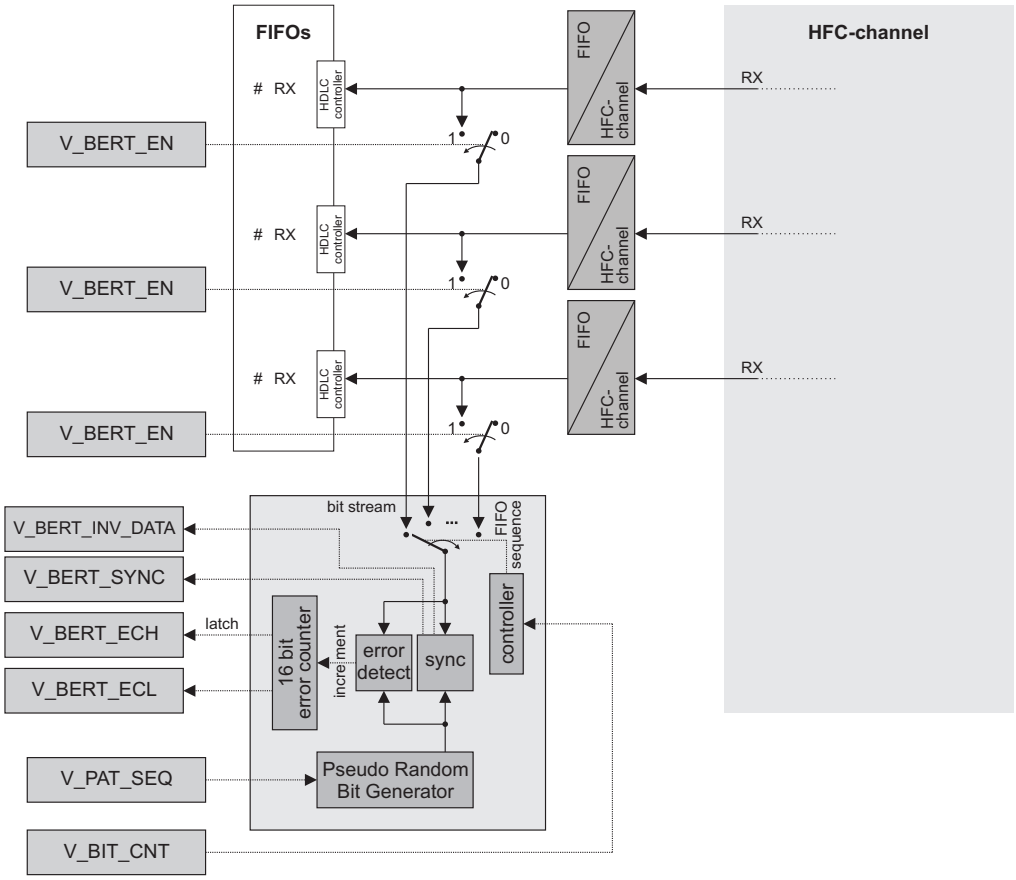


Figure 10.2: *BERT receiver block diagram*

byte R_BERT_ECL should be read first to latch the high byte. Then the high byte can be read from register R_BERT_ECH. A read access to the low byte R_BERT_ECL clears the 16 bit counter.

The BERT procedure should first wait for the synchronization state. After this, the BERT error counter should be cleared by reading register R_BERT_ECL.

Received BERT data is passed through the HFC-channel assigner if V_BERT_EN = '1' in register A_IRQ_MSK[FIFO]. For this reason, either a FIFO-to-E1 or a FIFO-to-PCM configuration must be selected. Furthermore, the allocated FIFO must be enabled to switch on the data path. Received BERT data is stored in the FIFO but it needs not to be read out. Received BERT data is collected from all FIFOs which have V_BERT_EN = '1' in the order of the FIFO processing sequence (see Section 3.2.3 on page 109).

Subchannel processing can be used together with the *Bit Error Rate Test*. Then V_BIT_CNT bits taken passed to the BERT receiver.

Inverted BERT data is automatically detected and can be checked with V_BERT_INV_DATA in register R_BERT_STA.

The automatically synchronization works only if the bit error rate is less than $4 \cdot 10^{-2}$. Synchronization state will not be achieved with a higher error rate. It is lost when many bit errors occur during a short time period. In this case, the re-synchronization starts automatically and a high bit error counter value indicates that a re-synchronization might has happened.

10.4 Register description

10.4.1 Write only registers

R_BERT_WD_MD	(w)	0x1B

Bit error rate test (BERT) and watchdog mode

Bits	Reset value	Name	Description
2..0	0	V_PAT_SEQ	**Continuous '0' / '1' or pseudo random pattern sequence for BERT** '000' = continuous '0' pattern '001' = continuous '1' pattern '010' = sequence length $2^9 - 1$ bits '011' = sequence length $2^{10} - 1$ bits '100' = sequence length $2^{15} - 1$ bits '101' = sequence length $2^{20} - 1$ bits '110' = sequence length $2^{20} - 1$ bits, but maximal 14 bits are zero '111' = pseudo random pattern seq. $2^{23} - 1$ **Note:** These sequences are defined in ITU-T O.150 and O.151 specifications.
3	0	V_BERT_ERR	**BERT error** Generates 1 error bit in the BERT data stream '0' = no error generation '1' = generates one error bit This bit is automatically cleared.
4		(reserved)	Must be '0'.
5	0	V_AUTO_WD_RES	**Automatically watchdog timer reset** '0' = watchdog is only reset by V_WD_RES '1' = watchdog is reset after every access to the chip
6		(reserved)	Must be '0'.
7	0	V_WD_RES	**Watchdog timer reset** '0' = no action '1' = manual watchdog timer reset This bit is automatically cleared.

10.4.2 Read only registers

R_BERT_STA	(r)	0x17

Bit error rate test status

Bits	Reset value	Name	Description
3..0	0	(reserved)	
4	0	V_BERT_SYNC	**BERT synchronization status** '0' = BERT not synchronized to input data '1' = BERT sync to input data
5	0	V_BERT_INV_DATA	**BERT data inversion** '0' = BERT receives normal data '1' = BERT receives inverted data
7..6	0	(reserved)	

R_BERT_ECL	(r)	0x1A

BERT error counter, low byte

Bits	Reset value	Name	Description
7..0	0	V_BERT_ECL	**Bits 7..0 of the BERT error counter** This register should be read first to latch the value of register R_BERT_ECH. **Note:** The BERT counter is cleared after reading this register.

R_BERT_ECH	(r)	0x1B

BERT error counter, high byte

Bits	Reset value	Name	Description
7..0	0	**V_BERT_ECH**	**Bits 15..8 of the BERT error counter** **Note:** Low byte should be read first (see register R_BERT_ECL).

Chapter 11

Auxiliary interface

Please note !

Please contact the Cologne Chip support team if you want to use the auxiliary interface.

Chapter 12

Clock, reset, interrupt, timer and watchdog

Table 12.1: *Overview of the HFC-E1 clock pins*

Number	Name	Description
90	OSC_IN	Oscillator Input Signal
91	OSC_OUT	Oscillator Output Signal
92	CLK_MODE	Clock Mode

Table 12.2: *Overview of the HFC-E1 reset, timer and watchdog registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x11	R_IRQMSK_MISC	303	0x10	R_IRQ_OVIEW	307
0x13	R_IRQ_CTRL	304	0x11	R_IRQ_MISC	308
0x1A	R_TI_WD	305	0x1C	R_STATUS	309
0xFF	A_IRQ_MSK	306	0xC8	R_IRQ_FIFO_BL0	310
			0xC9	R_IRQ_FIFO_BL1	311
			0xCA	R_IRQ_FIFO_BL2	312
			0xCB	R_IRQ_FIFO_BL3	313
			0xCC	R_IRQ_FIFO_BL4	314
			0xCD	R_IRQ_FIFO_BL5	315
			0xCE	R_IRQ_FIFO_BL6	316
			0xCF	R_IRQ_FIFO_BL7	317

12.1 Clock

12.1.1 Clock distribution

HFC-E1 uses two internal clock frequencies $f_{SYS} = f_{E1}$ and f_{PCM}. They are generated from one oscillator clock as shown in Figure 12.1.

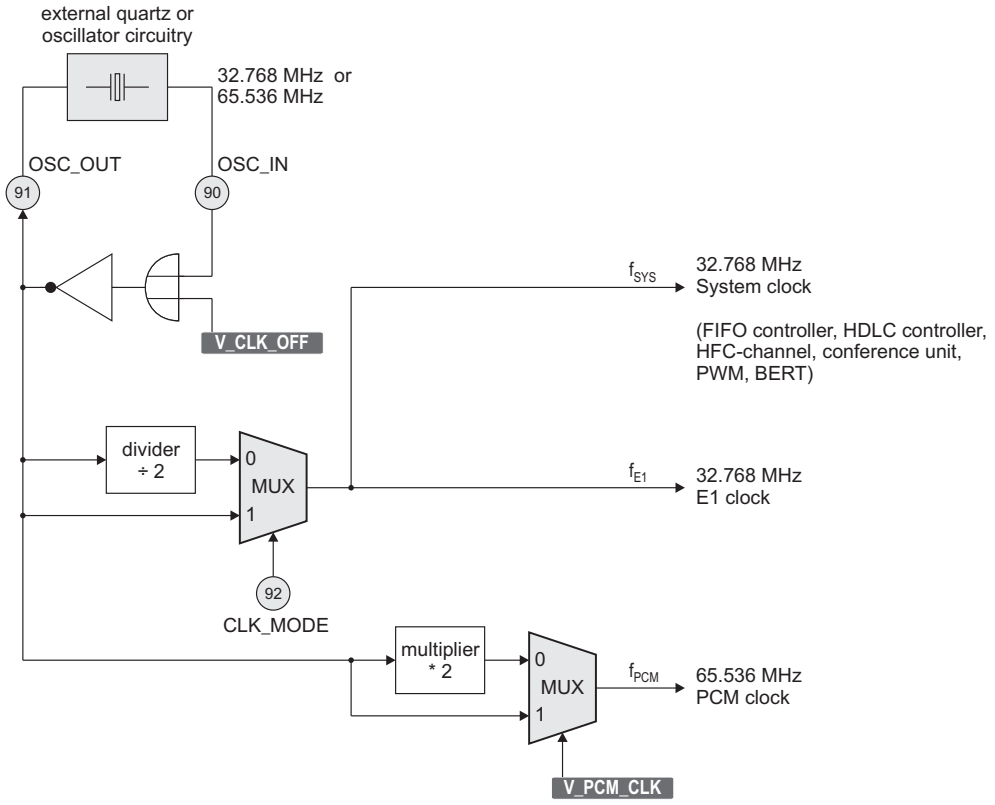


Figure 12.1: *Clock distribution*

The PCM clock must be set up to $f_{PCM} = 65.536\,\text{MHz}$. This is done with bit V_PCM_CLK in register R_BRG_PCM_CFG.

Pin CLK_MODE must be set as shown in Figure 12.1 to ensure a system clock of 32.768 MHz.

12.1.2 Clock oscillator circuitry

There are different ways to provide the internal clocks of HFC-E1. This section describes the Pierce oscillator circuitry, gives a hint to 3rd overtone oscillator, Crystal oscillator circuitry and, finally, shows how to connect the clock oscillator circuitry of several HFC-E1 in a cascade.

12.1.2.1 Frequency accuracy

E1 applications need an exact clock frequency. By the ISDN specification a precision of $\pm 100\,\text{ppm}$ is minimum requirement for passing the ISDN type approval. In respect to temperature dependence and ageing behavior a crystal with $\pm 50\,\text{ppm}$ is recommended.

12.1.2.2 Pierce oscillator

A typical clock oscillator circuitry using a 32.768 MHz crystal is shown in Figure 12.2. This Pierce oscillator is very popular for clock generation and is widely known from literature.

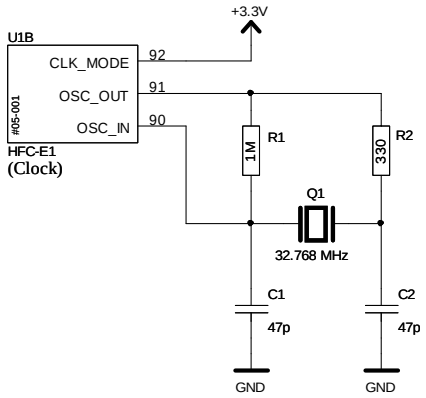


Figure 12.2: *Standard HFC-E1 quartz circuitry*

The feedback resistor R1 determines the DC operation point and is typically in the range 100 kΩ . . 10 MΩ for CMOS inverters.

The capacitive load C_L of the crystal is given in its data sheet. C1 and C2 should be chosen to fulfill

$$C_L = \frac{\mathsf{C1} \cdot \mathsf{C2}}{\mathsf{C1} + \mathsf{C2}} + C_S$$

where C_S is the stray capacitance. It is given by the input and output capacitances of the inverter and the shunt capacitance between the crystal terminals. Typically, C1 and C2 are chosen to be equal.

Finally, the resistor R2 is chosen to be roughly equal to the capacitive reactance of C2 at the frequency of oscillation.

$$\mathsf{R2} \sim \frac{1}{2\pi f_{\mathsf{Q1}} \cdot \mathsf{C2}}$$

R2 and C2 provide a low-pass filter that prevents the circuit from oscillating at a higher harmonic of the crystal frequency.

The minimum value of R2 depends on the recommended power consumption of the crystal. A too small value may damage the crystal or shorten the lifetime. When R2 is too large, the oscillation might not start. As HFC-E1 has a bufffered inverter between pins OSC_IN and OSC_OUT the value of R2 can be increased. A factor of about 2 . . 3, e.g., is well.

The circuitry shown in Figure 12.2 is based on a crystal with $C_L = 30\,\mathrm{pF}$ and a stray capacitance of $C_S = 5\,\mathrm{pF}$. This leads to

$$\mathsf{C1} = \mathsf{C2} = 2 \cdot (C_L - C_S) = 59\,\mathrm{pF} \sim 47\,\mathrm{pF}$$

and

$$\mathsf{R2} = \frac{3}{2\pi f_{\mathsf{Q1}} \cdot \mathsf{C2}} = 310\,\Omega \sim 330\,\Omega \quad .$$

12.1.2.3 3rd overtone oscillator

A different oscillator frequency with double frequency 65.536 MHz can be used alternatively. For this a 3rd overtone crystal or a clock oscillator can be used.

12.1.2.4 Crystal oscillator circuitry

It is possible to feed the OSC_IN input of HFC-E1 with a standard 3.3 V crystal oscillator. The input switching level is close to $V_{DD}/2$ (CMOS level) and HFC-E1 can accept at least a duty cycle of 45 % high/55 % low to 55 % high/45 % low.

12.1.2.5 HFC-E1 cascade

Figure 12.3 shows how to connect several HFC-E1 to only one quartz circuitry.

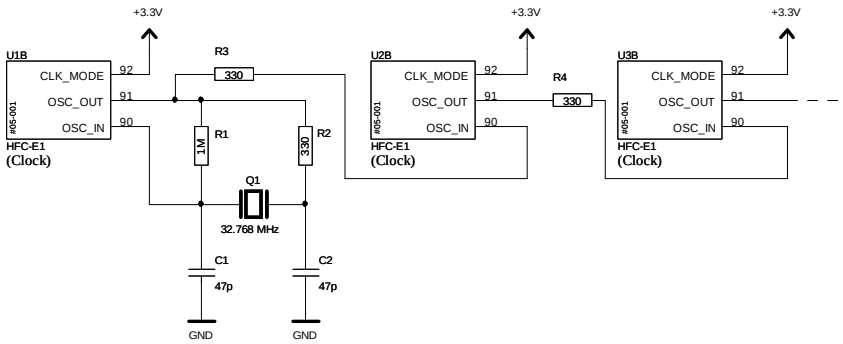


Figure 12.3: *Cascade-connected HFC-E1 with only one quartz circuitry*

12.2 Reset

HFC-E1 has a level sensitive RESET input at pin 198. This is low active in PCI mode (pin name RST#) and high active in all other modes (pin name RESET). Pins MODE0 and MODE1 must be valid during RESET and /SPISEL must be '1' (inactive) when the SPI bus interface mode is selected. The reset pulse must not be shorter than 10 ns.

After RESET, HFC-E1 enters an initialization sequence. Its duration depends on the number of FIFOs and has a maximum length of 100 µs. When the initialization process is finished, bit V_BUSY changes from '1' to '0'.

PCM initialization requires the F0IO clock. The PCM data rate should be set during 125 µs after reset to asure that all PCM array registers are initialized. Only 32 time slots are initialized if V_PCM_DR is left in its reset state '00' (see also Section 6.3).

HFC-E1 has 4 different soft resets. The FIFO registers, PCM registers and E1 registers can be reset independently with the bits of register R_CIRM which are listed in Table 12.3. The reset bits must be set and cleared by software.

A hardware reset implies the soft reset, of course.

Table 12.3: *HFC-E1 reset groups*

Reset name	Reset group	Register bit	Description
Soft Reset	0	V_SRES	Reset for FIFO, PCM and E1 registers of HFC-E1. Soft reset is the same as reset of all partial reset registers.
HFC Reset	1	V_HFC_RES	Reset for all FIFO registers of HFC-E1.
PCM Reset	2	V_PCM_RES	Reset for all PCM registers of HFC-E1.
E1 Reset	3	V_E1_RES	Reset for all E1 registers of HFC-E1.
Hardware reset	H	–	Hardware reset initiated by RESET input pin.

Information about the allocation of the registers to the different reset groups can be found in the register list on pages 20 and 22. Many registers are allocated to more than one reset group, and some are not resetable by software.

12.3 Interrupt

12.3.1 Common features

HFC-E1 is equipped with a maskable interrupt engine. A big variety of interrupt sources can be enabled and disabled. All interrupts except FIFO interrupts are reported on reading the interrupt status registers independently of masking the interrupt or not. Only mask enabled interrupts are used to generate an interrupt on the interrupt pin of HFC-E1. Reading the interrupt status register resets the bits. Interrupt bits set during the reading are reported at the next reading of the interrupt status registers.

Pin 197 is the interrupt output line for all bus interface modes. ISA PnP uses additional pins 106 . . 112 for interrupt purposes. After reset, all interrupts are disabled. The interrupt lines must be enabled with V_GLOB_IRQ_EN set to '1' in register R_IRQ_CTRL. The polarity of the interrupt signals can be changed by an optional external transistor from *active low* to *active high*. This must be configured with bitmap V_IRQ_POL in register R_IRQ_CTRL.

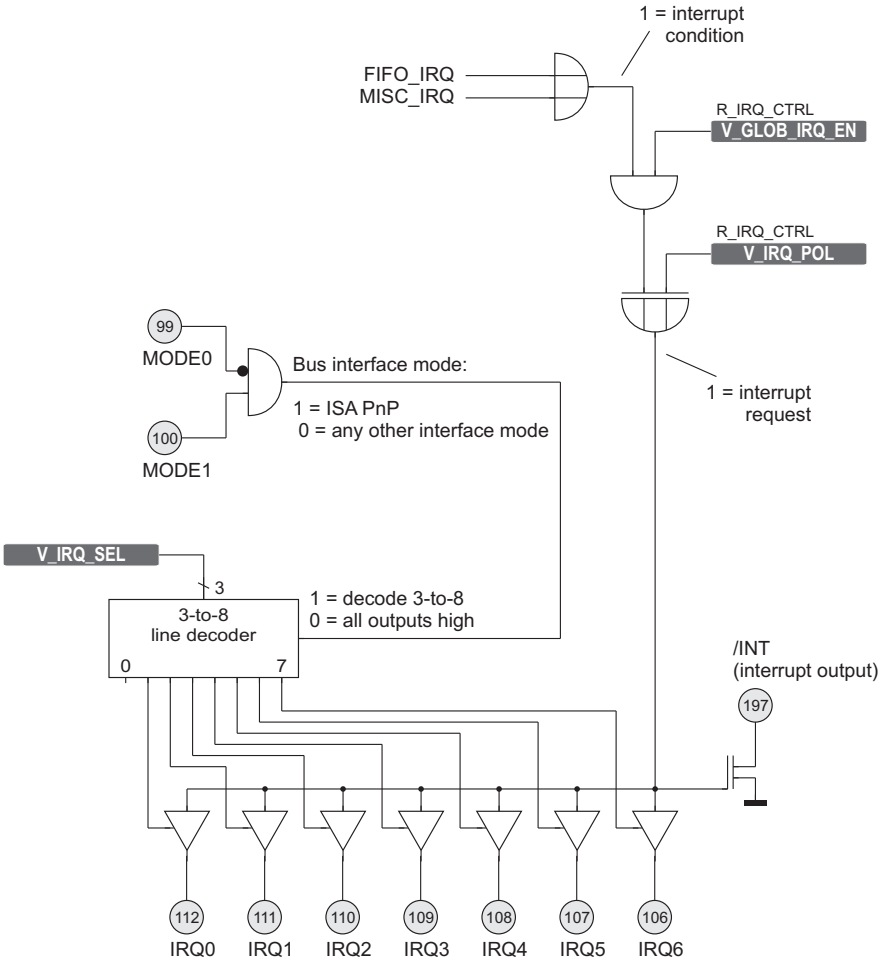


Figure 12.4: *Interrupt output*

12.3.2 FIFO interrupt

FIFO interrupts can be enabled or disabled by setting bit V_IRQ in register A_IRQ_MSK[FIFO]. Because there are 64 interrupts there are 8 interrupt status registers for FIFO interrupts. To determine which interrupt register must be read in an interrupt routine there is an interrupt overview register which shows in which status register at least one interrupt bit is set (R_IRQ_OVIEW). Reading this register does not clear any interrupt. The following reading of an interrupt register (R_IRQ_FIFO_BL0 . . R_IRQ_FIFO_BL7) clears the reported interrupts.

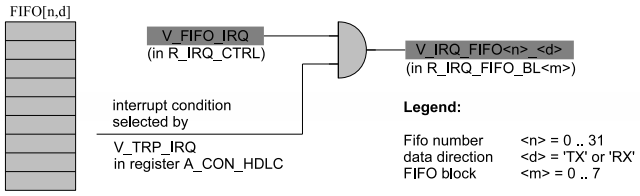


Figure 12.5: *Enable FIFO interrupt condition with V_FIFO_IRQ*

The FIFO interrupts must be enabled with the global bit V_FIFO_IRQ in register R_IRQ_CTRL as shown if Figure 12.5.

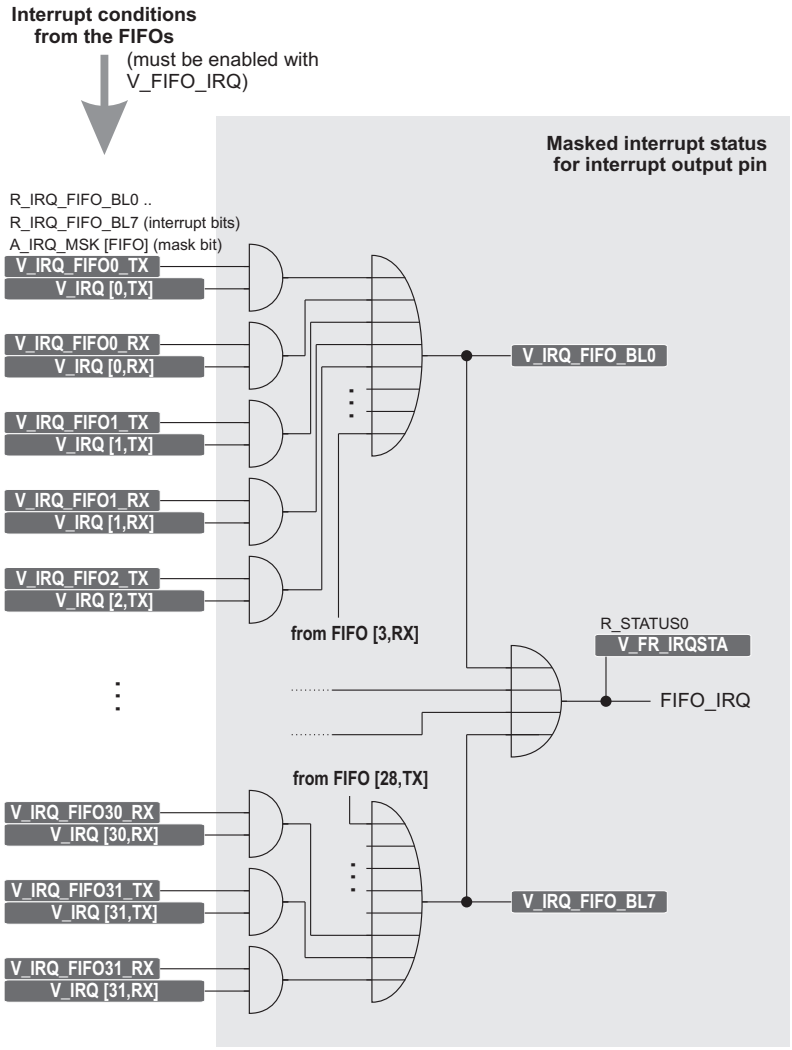


Figure 12.6: *FIFO interrupt*

12.3.3 Miscelleanous interrupts

12.3.3.1 E1 interface interrupt

A state change condition of the state machine can be signaled from an interrupt. When bit V_STA_IRQMSK in register R_IRQMSK_MISC is set to '1', the interrupt is enabled. Bit V_STA_IRQ in register R_IRQ_MISC contains the state change condition even if the interrupt is disabled.

12.3.3.2 Timer interrupt

HFC-E1 includes a timer with interrupt capability. The timer counts F0IO pulses, i.e. it is incremented every 125 µs.

A timer event is indicated with V_TI_IRQ = '1' in register R_IRQ_MISC. This event generates an interrupt if the mask bit V_TI_IRQMSK is set to '1' in register R_IRQMSK_MISC.

A timer event is generated every $2^{V\_EV\_TS} \cdot 250\,\mu s$ where V_EV_TS = 0..15 in register R_TI_WD. This leads to a timer event frequency from 250 µs to 8.192 s.

12.3.3.3 125 µs interrupt

HFC-E1 changes every 125 µs from non processing into processing state. This event can be reported with an interrupt. Bit V_PROC_IRQMSK in register R_IRQMSK_MISC must be set to '1' to enable this interrupt capability. In case of an interrupt, bit V_IRQ_PROC in register R_IRQ_MISC has the value '1'.

12.3.3.4 DTMF interrupt

When DTMF detection has been finished, V_DTMF_IRQ in register R_IRQ_MISC is set to '1'.

An interrupt occurs, when bitmap V_DTMF_IRQMSK in register R_IRQMSK_MISC is set to '1'. The interrupt is cleared with a read access to register R_IRQMSK_MISC.

12.3.3.5 One-second interrupt

An interrupt event can be generated every second, when bitmap V_IRQ1S_MSK in register R_IRQMSK_MISC is set to '1'. The interrupt is cleared with a read access to register R_IRQMSK_MISC.

The one-second interval is reset exceptionally, when multiframe alignment is lost. This is indicated by a change of V_MFA_STA from '10' to '00' in register R_SYNC_STA.

12.3.3.6 External interrupt

The GPI[31..24] pins have interrupt capability. Figure 12.8 shows the block diagram of this external interrupt capability.

The external interrupt occurs, when at least one of the eight GPI lines have low input signal. For this reason all unused GPI lines must be connected to VDD when the external interrupt is enabled. Bit

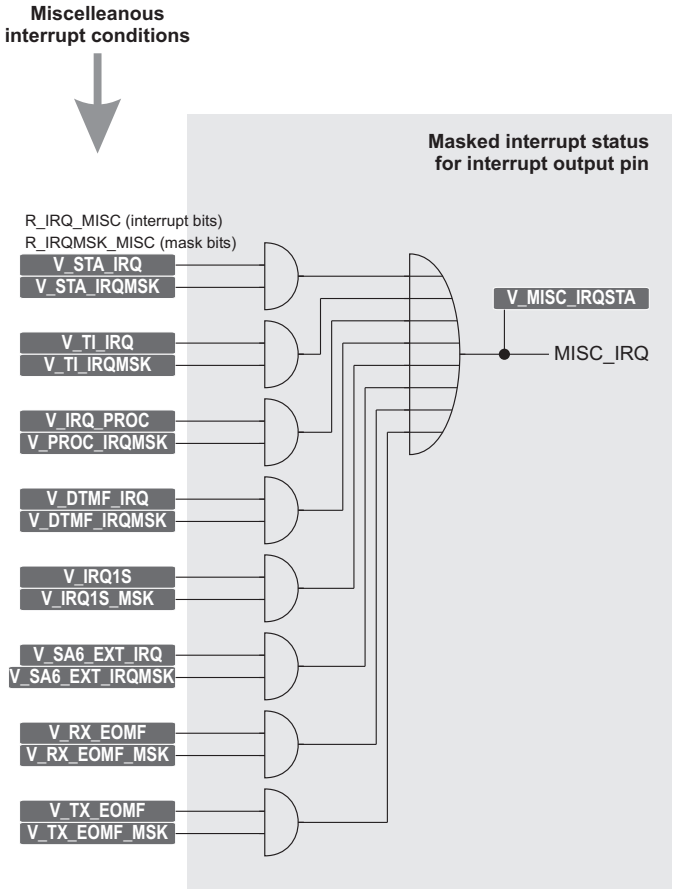


Figure 12.7: *Miscelleanous interrupts*

V_EXT_IRQ_EN must be set to '1' in register R_PWM_MD to enable the external interrupt unit. The current state of the joined GPI signals can be read from bit V_EXT_IRQSTA in register R_STATUS.

From this signal an interrupt can be generated if bit V_SA6_EXT_IRQMSK in register R_IRQMSK_MISC is set to '1'. In case of an interrupt event, V_SA6_EXT_IRQ can be read to determine whether the external interrupt occured.

Another interrupt condition is linked to bit V_EXT_IRQSTA, that is when SA6 pattern has changed.

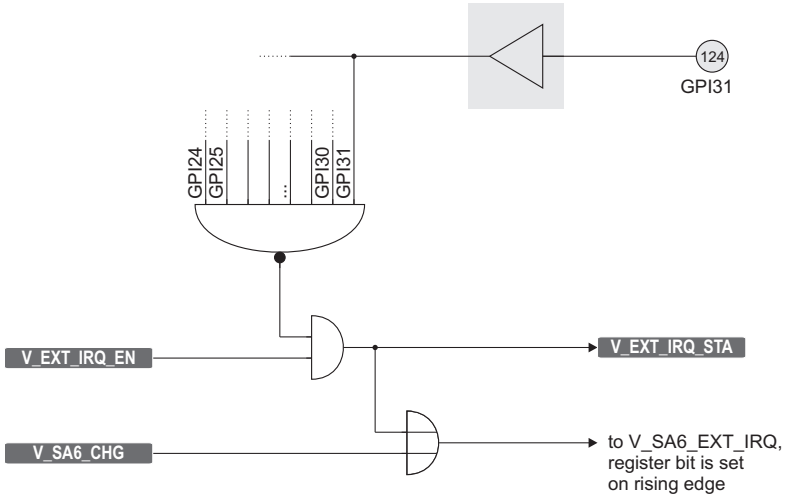


Figure 12.8: *External interrupt block diagram*

12.3.3.7 End of multiframe interrupt

When a complete multiframe has been received or transmitted, bitmap V_RX_EOMF_MSK or V_TX_EOMF_MSK respectively in register R_IRQMSK_MISC is set to '1'. The interrupt is cleared with a read access to register R_IRQMSK_MISC.

12.4 Watchdog reset

The parallel processor interface of HFC-E1 includes a watchdog functionality.

A watchdog event generates a low signal at pin /WD (pin 23). The watchdog timer can either be reset manually or automatically.

- Manual watchdog reset is selected with V_AUTO_WD_RES = '0' in register R_BERT_WD_MD. Then, writing V_WD_RES = '1' into register R_BERT_WD_MD resets the watchdog timer. This bit is automatically cleared afterwards.
- V_AUTO_WD_RES = '1' must be set to switch on the automatically watchdog reset. In this case every access to the chip clears the watchdog timer.

The watchdog counter is incremented every 2 ms. An event occurs after $2^{V\_WD\_TS} \cdot 2\,\text{ms}$ where V_WD_TS = 0..15 in register R_TI_WD. This leads to a watchdog event frequency from 2 ms to 65.536 s.

12.5 Register description

12.5.1 Write only registers

R_BRG_PCM_CFG	(w)	0x02

PCM configuration register

Bits	Reset value	Name	Description
4..0		(reserved)	Must be '00000'.
5	0	**V_PCM_CLK**	**Clock of the PCM module** '0' = f_{PCM} is twice the input clock at pin OSC_IN '1' = f_{PCM} is equal to the input clock at pin OSC_IN PCM clock must be set up to65.536 MHz.
7..6	0	**V_ADDR_WRDLY**	**Address write delay** Delay from rising edge of pin /SR_WR to address change for external RAM '00' = delay is approximately 3 ns '01' = delay is approximately 5 ns '10' = delay is approximately 7 ns '11' = delay is approximately 9 ns

R_IRQMSK_MISC	(w)	0x11

Miscellaneous interrupt status mask register

'0' means that the interrupt is not used for generating an interrupt on the interrupt pin 197. '1' enables the interrupt generation in case of the committed event.

Bits	Reset value	Name	Description
0	0	V_STA_IRQMSK	State of state machine changed interrupt mask bit
1	0	V_TI_IRQMSK	Timer elapsed interrupt mask bit
2	0	V_PROC_IRQMSK	Processing / nonprocessing transition interrupt mask bit (every 125 µs)
3	0	V_DTMF_IRQMSK	DTMF detection interrupt mask bit
4	0	V_IRQ1S_MSK	1 second interrupt mask bit
5	0	V_SA6_EXT_IRQMSK	SA6 pattern changed or external interrupt mask bit
6	0	V_RX_EOMF_MSK	Receive end of multiframe mask bit
7	0	V_TX_EOMF_MSK	Transmit end of multiframe mask bit

R_IRQ_CTRL	(w)	0x13

Interrupt control register

Bits	Reset value	Name	Description
0	0	**V_FIFO_IRQ**	**FIFO interrupt** '0' = all FIFO interrupts disabled '1' = all FIFO interrupts enabled
2..1		(reserved)	Must be '00'.
3	0	**V_GLOB_IRQ_EN**	**Global interrupt signal enable** The interrupt lines are either pins 106..112 in ISA PnP mode or pin 197 in all other bus interface modes. '0' = disable '1' = enable
4	0	**V_IRQ_POL**	**Polarity of interrupt signal** '0' = active low signal '1' = active high signal
7..5		(reserved)	Must be '000'.

R_TI_WD	(w)	0x1A

Timer and watchdog control register

Bits	Reset value	Name	Description
3..0	0	**V_EV_TS**	**Timer event after** $2^n \cdot 250\,\mu s$ 0 = 250 µs 1 = 500 µs 2 = 1 ms 3 = 2 ms 4 = 4 ms 5 = 8 ms 6 = 16 ms 7 = 32 ms 8 = 64 ms 9 = 128 ms 0xA = 256 ms 0xB = 512 ms 0xC = 1.024 s 0xD = 2.048 s 0xE = 4.096 s 0xF = 8.192 s
7..4	0	**V_WD_TS**	**Watchdog event after** $2^n \cdot 2\,ms$ 0 = 2 ms 1 = 4 ms 2 = 8 ms 3 = 16 ms 4 = 32 ms 5 = 64 ms 6 = 128 ms 7 = 256 ms 8 = 512 ms 9 = 1.024 s 0xA = 2.048 s 0xB = 4.096 s 0xC = 8.192 s 0xD = 16.384 s 0xE = 32.768 s 0xF = 65.536 s

A_IRQ_MSK [FIFO] **(w)** **0xFF**

Interrupt register for the selected FIFO

Before writing this array register the FIFO must be selected by register R_FIFO.

Bits	Reset value	Name	Description
0	0	**V_IRQ**	**Interrupt mask for the selected FIFO** '0' = disabled '1' = enabled
1	0	**V_BERT_EN**	**BERT enable** '0' = BERT disabled, normal data is transmitted and received '1' = BERT enabled, output of BERT generator is transmitted and received data is checked by BERT
2	0	**V_MIX_IRQ**	**Mixed interrupt generation** '0' = disabled (normal operation) '1' = frame interrupts and transparent mode interrupts are both generated in HDLC mode
7..3		(reserved)	Must be '00000'.

12.5.2 Read only registers

R_IRQ_OVIEW	(r)	0x10

FIFO interrupt overview register

Every bit with value '1' indicates that an interrupt has occured in the FIFO block. A FIFO block consists of 4 transmit and 4 receive FIFOs. The exact FIFO can be determined by reading the R_IRQ_FIFO_BL0 . . R_IRQ_FIFO_BL7 registers that belong to the specified FIFO block.

Reading any R_IRQ_FIFO_BL0 . . R_IRQ_FIFO_BL7 register clears the corresponding bit in this register. Reading this overview register does not clear any interrupt bit.

Bits	Reset value	Name	Description
0		**V_IRQ_FIFO_BL0**	**Interrupt overview of FIFO block 0** (FIFOs 0 . . 3)
1		**V_IRQ_FIFO_BL1**	**Interrupt overview of FIFO block 1** (FIFOs 4 . . 7)
2		**V_IRQ_FIFO_BL2**	**Interrupt overview of FIFO block 2** (FIFOs 8 . . 11)
3		**V_IRQ_FIFO_BL3**	**Interrupt overview of FIFO block 3** (FIFOs 12 . . 15)
4		**V_IRQ_FIFO_BL4**	**Interrupt overview of FIFO block 4** (FIFOs 16 . . 19)
5		**V_IRQ_FIFO_BL5**	**Interrupt overview of FIFO block 5** (FIFOs 20 . . 23)
6		**V_IRQ_FIFO_BL6**	**Interrupt overview of FIFO block 6** (FIFOs 24 . . 27)
7		**V_IRQ_FIFO_BL7**	**Interrupt overview of FIFO block 7** (FIFOs 28 . . 31)

R_IRQ_MISC	(r)	0x11

Miscellaneous interrupt status register

All bits of this register are cleared after a read access.

Bits	Reset value	Name	Description
0	0	**V_STA_IRQ**	**State change interrupt status** '1' = state of E1 state machine has changed
1	0	**V_TI_IRQ**	**Timer interrupt status** '1' = timer elapsed
2	0	**V_IRQ_PROC**	**Processing / non processing transition interrupt status** '1' = HFC-E1 has changed from processing to non processing phase (every 125 µs).
3	0	**V_DTMF_IRQ**	**DTMF detection interrupt status** '1' = DTMF detection has been finished. The results can be read from the RAM.
4	0	**V_IRQ1S**	**1 second interrupt status** '1' = 1 second elapsed
5	0	**V_SA6_EXT_IRQ**	**SA6 pattern has changed or external interrupt**
6	0	**V_RX_EOMF**	**End of multiframe received**
7	0	**V_TX_EOMF**	**End of multiframe transmited**

R_STATUS	(r)	0x1C

HFC-E1 status register

Bits	Reset value	Name	Description
0		V_BUSY	**BUSY / NOT BUSY status** '0' = HFC-E1 is not busy, all accesses are allowed '1' = HFC-E1 is BUSY after initialising Reset FIFO, increment *F*-counter or change FIFO
1		V_PROC	**Processing / non processing status** '0' = HFC-E1 has finished the processing phase during the 125 µs cycle '1' = HFC-E1 is in processing phase (once every 125 µs cycle) **Note:** The processing / non processing transition can be notified with an interrupt (see V_IRQ_PROC in register R_IRQ_MISC).
2		(reserved)	
3		V_LOST_STA	**LOST error (frames have been lost)** This means HFC-E1 did not process all data in 125 µs. So data may be corrupted. Bit V_RES_LOST in register A_INC_RES_FIFO must be set to reset this bit.
4		V_SYNC_IN	**Synchronization input** Value of the SYNC_I input pin
5		V_EXT_IRQSTA	**External interrupt** The external interrupt signal is currently set.
6		V_MISC_IRQSTA	**Any miscellaneous interrupt** All enabled miscellaneous interrupts of register R_IRQ_MISC are 'ored'.
7		V_FR_IRQSTA	**Any FIFO interrupt** All enabled FIFO interrupts of the registers R_IRQ_FIFO_BL0 . . R_IRQ_FIFO_BL7 are 'ored'.

R_IRQ_FIFO_BL0	(r)	0xC8

FIFO interrupt register for FIFO block 0

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO0_TX**	**Interrupt occured in transmit FIFO 0**
1	0	**V_IRQ_FIFO0_RX**	**Interrupt occured in receive FIFO 0**
2	0	**V_IRQ_FIFO1_TX**	**Interrupt occured in transmit FIFO 1**
3	0	**V_IRQ_FIFO1_RX**	**Interrupt occured in receive FIFO 1**
4	0	**V_IRQ_FIFO2_TX**	**Interrupt occured in transmit FIFO 2**
5	0	**V_IRQ_FIFO2_RX**	**Interrupt occured in receive FIFO 2**
6	0	**V_IRQ_FIFO3_TX**	**Interrupt occured in transmit FIFO 3**
7	0	**V_IRQ_FIFO3_RX**	**Interrupt occured in receive FIFO 3**

R_IRQ_FIFO_BL1	(r)	0xC9

FIFO interrupt register for FIFO block 1

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO4_TX**	**Interrupt occured in transmit FIFO 4**
1	0	**V_IRQ_FIFO4_RX**	**Interrupt occured in receive FIFO 4**
2	0	**V_IRQ_FIFO5_TX**	**Interrupt occured in transmit FIFO 5**
3	0	**V_IRQ_FIFO5_RX**	**Interrupt occured in receive FIFO 5**
4	0	**V_IRQ_FIFO6_TX**	**Interrupt occured in transmit FIFO 6**
5	0	**V_IRQ_FIFO6_RX**	**Interrupt occured in receive FIFO 6**
6	0	**V_IRQ_FIFO7_TX**	**Interrupt occured in transmit FIFO 7**
7	0	**V_IRQ_FIFO7_RX**	**Interrupt occured in receive FIFO 7**

R_IRQ_FIFO_BL2	(r)	0xCA

FIFO interrupt register for FIFO block 2

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO8_TX**	**Interrupt occured in transmit FIFO 8**
1	0	**V_IRQ_FIFO8_RX**	**Interrupt occured in receive FIFO 8**
2	0	**V_IRQ_FIFO9_TX**	**Interrupt occured in transmit FIFO 9**
3	0	**V_IRQ_FIFO9_RX**	**Interrupt occured in receive FIFO 9**
4	0	**V_IRQ_FIFO10_TX**	**Interrupt occured in transmit FIFO 10**
5	0	**V_IRQ_FIFO10_RX**	**Interrupt occured in receive FIFO 10**
6	0	**V_IRQ_FIFO11_TX**	**Interrupt occured in transmit FIFO 11**
7	0	**V_IRQ_FIFO11_RX**	**Interrupt occured in receive FIFO 11**

R_IRQ_FIFO_BL3	(r)	0xCB

FIFO interrupt register for FIFO block 3

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO12_TX**	**Interrupt occured in transmit FIFO 12**
1	0	**V_IRQ_FIFO12_RX**	**Interrupt occured in receive FIFO 12**
2	0	**V_IRQ_FIFO13_TX**	**Interrupt occured in transmit FIFO 13**
3	0	**V_IRQ_FIFO13_RX**	**Interrupt occured in receive FIFO 13**
4	0	**V_IRQ_FIFO14_TX**	**Interrupt occured in transmit FIFO 14**
5	0	**V_IRQ_FIFO14_RX**	**Interrupt occured in receive FIFO 14**
6	0	**V_IRQ_FIFO15_TX**	**Interrupt occured in transmit FIFO 15**
7	0	**V_IRQ_FIFO15_RX**	**Interrupt occured in receive FIFO 15**

R_IRQ_FIFO_BL4	(r)	0xCC

FIFO interrupt register for FIFO block 4

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO16_TX**	**Interrupt occured in transmit FIFO 16**
1	0	**V_IRQ_FIFO16_RX**	**Interrupt occured in receive FIFO 16**
2	0	**V_IRQ_FIFO17_TX**	**Interrupt occured in transmit FIFO 17**
3	0	**V_IRQ_FIFO17_RX**	**Interrupt occured in receive FIFO 17**
4	0	**V_IRQ_FIFO18_TX**	**Interrupt occured in transmit FIFO 18**
5	0	**V_IRQ_FIFO18_RX**	**Interrupt occured in receive FIFO 18**
6	0	**V_IRQ_FIFO19_TX**	**Interrupt occured in transmit FIFO 19**
7	0	**V_IRQ_FIFO19_RX**	**Interrupt occured in receive FIFO 19**

R_IRQ_FIFO_BL5	(r)	0xCD

FIFO interrupt register for FIFO block 5

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO20_TX**	**Interrupt occured in transmit FIFO 20**
1	0	**V_IRQ_FIFO20_RX**	**Interrupt occured in receive FIFO 20**
2	0	**V_IRQ_FIFO21_TX**	**Interrupt occured in transmit FIFO 21**
3	0	**V_IRQ_FIFO21_RX**	**Interrupt occured in receive FIFO 21**
4	0	**V_IRQ_FIFO22_TX**	**Interrupt occured in transmit FIFO 22**
5	0	**V_IRQ_FIFO22_RX**	**Interrupt occured in receive FIFO 22**
6	0	**V_IRQ_FIFO23_TX**	**Interrupt occured in transmit FIFO 23**
7	0	**V_IRQ_FIFO23_RX**	**Interrupt occured in receive FIFO 23**

R_IRQ_FIFO_BL6	(r)	0xCE

FIFO interrupt register for FIFO block 6

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO24_TX**	**Interrupt occured in transmit FIFO 24**
1	0	**V_IRQ_FIFO24_RX**	**Interrupt occured in receive FIFO 24**
2	0	**V_IRQ_FIFO25_TX**	**Interrupt occured in transmit FIFO 25**
3	0	**V_IRQ_FIFO25_RX**	**Interrupt occured in receive FIFO 25**
4	0	**V_IRQ_FIFO26_TX**	**Interrupt occured in transmit FIFO 26**
5	0	**V_IRQ_FIFO26_RX**	**Interrupt occured in receive FIFO 26**
6	0	**V_IRQ_FIFO27_TX**	**Interrupt occured in transmit FIFO 27**
7	0	**V_IRQ_FIFO27_RX**	**Interrupt occured in receive FIFO 27**

R_IRQ_FIFO_BL7	(r)	0xCF

FIFO interrupt register for FIFO block 7

In HDLC mode the *end of frame* is signaled, while in transparent mode the frequency of interrupts is set in bitmap V_TRP_IRQ of register A_CON_HDLC.

The bit value '1' indicates that the corresponding FIFO generated an interrupt. If a bit is '0', no interrupt occured in the corresponding FIFO.

Reading this register clears all set bits and the corresponding bit of register R_IRQ_OVIEW.

Bits	Reset value	Name	Description
0	0	**V_IRQ_FIFO28_TX**	**Interrupt occured in transmit FIFO 28**
1	0	**V_IRQ_FIFO28_RX**	**Interrupt occured in receive FIFO 28**
2	0	**V_IRQ_FIFO29_TX**	**Interrupt occured in transmit FIFO 29**
3	0	**V_IRQ_FIFO29_RX**	**Interrupt occured in receive FIFO 29**
4	0	**V_IRQ_FIFO30_TX**	**Interrupt occured in transmit FIFO 30**
5	0	**V_IRQ_FIFO30_RX**	**Interrupt occured in receive FIFO 30**
6	0	**V_IRQ_FIFO31_TX**	**Interrupt occured in transmit FIFO 31**
7	0	**V_IRQ_FIFO31_RX**	**Interrupt occured in receive FIFO 31**

Chapter 13

General purpose I/O pins (GPIO) and input pins (GPI)

(For an overview of the GPIO and GPI pins see Tables 13.3 and 13.3 on page 321.)

Table 13.1: *Overview of the HFC-E1 general purpose I/O registers*

Write only registers:			Read only registers:		
Address	**Name**	**Page**	**Address**	**Name**	**Page**
0x40	R_GPIO_OUT0	324	0x40	R_GPIO_IN0	328
0x41	R_GPIO_OUT1	325	0x41	R_GPIO_IN1	329
0x42	R_GPIO_EN0	325	0x44	R_GPI_IN0	329
0x43	R_GPIO_EN1	326	0x45	R_GPI_IN1	330
0x44	R_GPIO_SEL	327	0x46	R_GPI_IN2	330
			0x47	R_GPI_IN3	331

13.1 GPIO and GPI functionality

HFC-E1 has up to 16 general purpose I/O (GPIO) and up to 31 general purpose input (GPI) pins. As shown in Tables 13.3 and 13.2, some of them – two GPIO and four GPI pins – are shared with the E1 interface pins and can only be used if this interface is not in use.

E1 interface pins are always called 1st pin function. GPIO and GPI pins are called 2nd pin function even if a pin has no functionality at its 1st function.

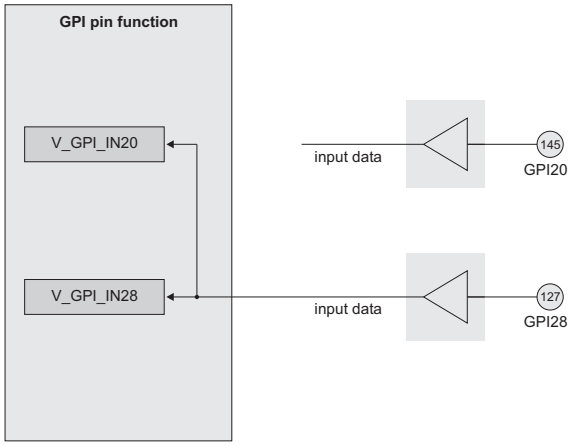
GPIOs must be switched into GPIO mode with register R_GPIO_SEL if they should be used as outputs. The input functionality of all GPIOs and GPIs is allways enabled (see Figure 13.1). The output values for the GPIOs are set in registers R_GPIO_OUT0 and R_GPIO_OUT1. The output function can be enabled in registers R_GPIO_EN0 and R_GPIO_EN1. If disabled, the output drivers are tristated. A detailed GPIO block diagram is shown in Figure 13.2.

The input values for the GPIO[0..15] can be read from registers R_GPIO_IN0 and R_GPIO_IN1. The input values for GPI[0..19,21..31] can be read from registers R_GPI_IN0, R_GPI_IN1, R_GPI_IN2 and R_GPI_IN3.

Important !

Due to a faulty implementation of the GPI function, GPI[20] does not exist. Any read access to GPI[20] returns the GPI[28] value as shown right.

The GPI[20] pin should be treated as any other input pin if not used, i.e. it should be connected to ground or power supply to avoid floating input problems.



13.2 GPIO output voltage

The GPIO output high voltage can be influenced for each set of 4 GPIOs by connecting the appropriate VDD_E1 pin to a voltage different from VDD . The voltage must not exceed 3.6 V. Table 13.3 shows the allocation of power supply pins to the GPIO output drivers.

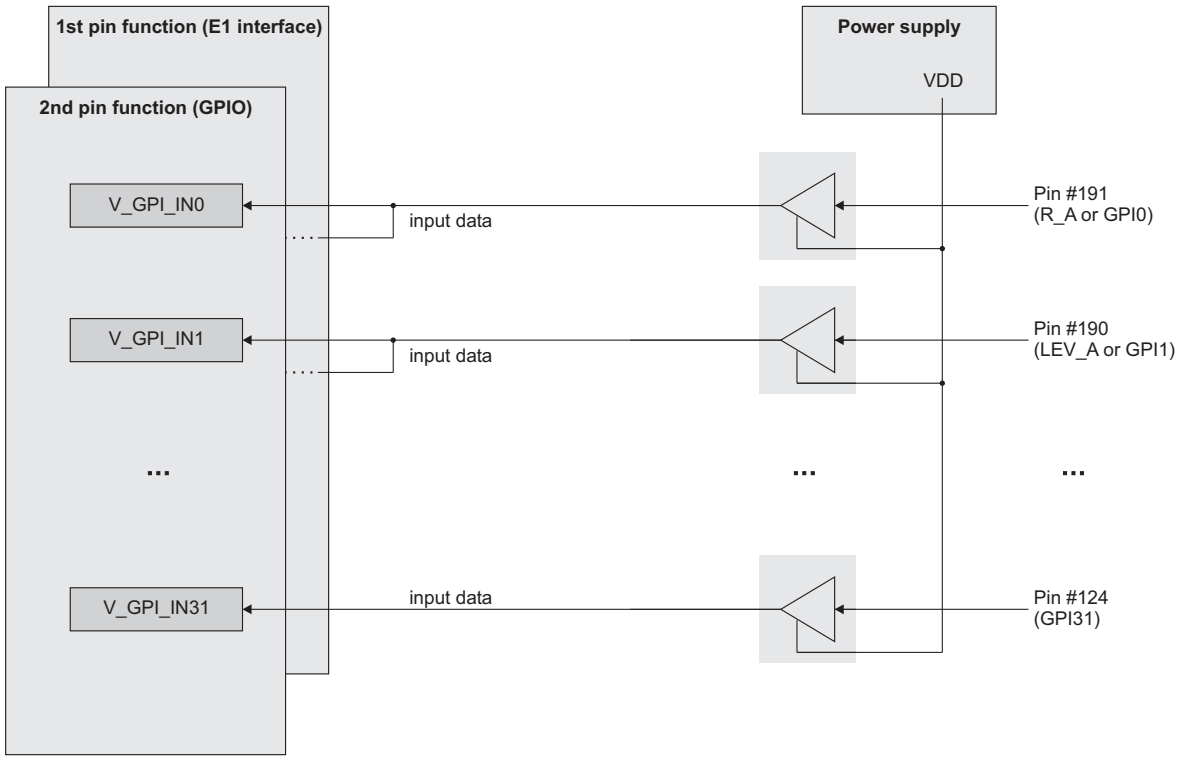


Figure 13.1: *GPI block diagram*

Table 13.2: *GPI pins of HFC-E1*

GPIO pin		Shared with interface	GPIO pin		Shared with interface
124	GPI31	–	159	GPI15	–
125	GPI30	–	160	GPI14	–
126	GPI29	–	161	GPI13	–
127	GPI28	–	162	GPI12	–
136	GPI27	–	171	GPI11	–
137	GPI26	–	172	GPI10	–
138	GPI25	–	173	GPI9	–
139	GPI24	–	174	GPI8	–
142	GPI23	–	176	GPI7	–
143	GPI22	–	177	GPI6	–
144	GPI21	–	178	GPI5	–
145	– *1	–	179	GPI4	–
154	GPI19	–	188	GPI3	E1
155	GPI18	–	189	GPI2	E1
156	GPI17	–	190	GPI1	E1
157	GPI16	–	191	GPI0	E1

*1: GPI[20] is not available, see remark on page 320.

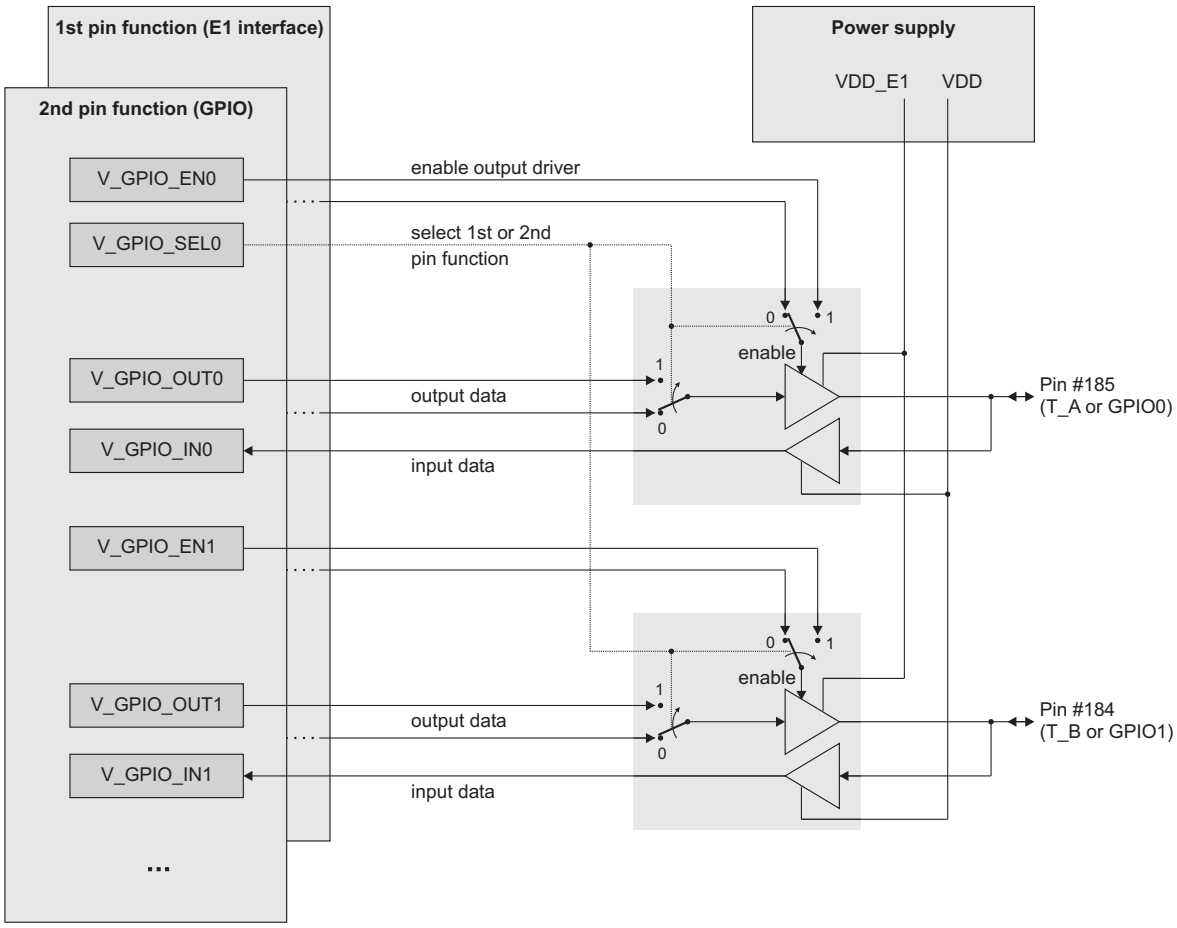


Figure 13.2: *GPIO block diagram (GPIO0 and GPIO1 exemplarily)*

Table 13.3: *GPIO pins of HFC-E1*

GPIO pin		Shared with interface	Output supply pin		GPIO pin		Shared with interface	Output supply pin	
130	GPIO15	–	129	VDD_E1	165	GPIO7	–	164	VDD_E1
131	GPIO14	–	129	VDD_E1	166	GPIO6	–	164	VDD_E1
132	GPIO13	–	129	VDD_E1	167	GPIO5	–	164	VDD_E1
133	GPIO12	–	129	VDD_E1	168	GPIO4	–	164	VDD_E1
148	GPIO11	–	147	VDD_E1	182	GPIO3	–	181	VDD_E1
149	GPIO10	–	147	VDD_E1	183	GPIO2	–	181	VDD_E1
150	GPIO9	–	147	VDD_E1	184	GPIO1	E1	181	VDD_E1
151	GPIO8	–	147	VDD_E1	185	GPIO0	E1	181	VDD_E1

13.3 Unused GPIO and GPI pins

Unused GPI pins should be connected to ground to avoid floating input buffers.

Unused GPIO pins should be configured as output ports. This sets the level of the input buffer – which is always active at the same pin – to a stable level and avoids floating input effect.

13.4 Register description

13.4.1 Write only registers

R_GPIO_OUT0	(w)	0x40

GPIO output data bits 7 . . 0

Bits	Reset value	Name	Description
0	0	V_GPIO_OUT0	Output data bit for pin GPIO0
1	0	V_GPIO_OUT1	Output data bit for pin GPIO1
2	0	V_GPIO_OUT2	Output data bit for pin GPIO2
3	0	V_GPIO_OUT3	Output data bit for pin GPIO3
4	0	V_GPIO_OUT4	Output data bit for pin GPIO4
5	0	V_GPIO_OUT5	Output data bit for pin GPIO5
6	0	V_GPIO_OUT6	Output data bit for pin GPIO6
7	0	V_GPIO_OUT7	Output data bit for pin GPIO7

R_GPIO_OUT1	(w)	0x41

GPIO output data bits 15 . . 8

Bits	Reset value	Name	Description
0	0	V_GPIO_OUT8	Output data bit for pin GPIO8
1	0	V_GPIO_OUT9	Output data bit for pin GPIO9
2	0	V_GPIO_OUT10	Output data bit for pin GPIO10
3	0	V_GPIO_OUT11	Output data bit for pin GPIO11
4	0	V_GPIO_OUT12	Output data bit for pin GPIO12
5	0	V_GPIO_OUT13	Output data bit for pin GPIO13
6	0	V_GPIO_OUT14	Output data bit for pin GPIO14
7	0	V_GPIO_OUT15	Output data bit for pin GPIO15

R_GPIO_EN0	(w)	0x42

GPIO output enable bits 7 . . 0

Every bit value '1' enables the allocated output driver. If an output driver is disabled (bit value '0'), the pin is used for data input.

Bits	Reset value	Name	Description
0	0	V_GPIO_EN0	Output enable for pin GPIO0
1	0	V_GPIO_EN1	Output enable for pin GPIO1
2	0	V_GPIO_EN2	Output enable for pin GPIO2
3	0	V_GPIO_EN3	Output enable for pin GPIO3
4	0	V_GPIO_EN4	Output enable for pin GPIO4
5	0	V_GPIO_EN5	Output enable for pin GPIO5
6	0	V_GPIO_EN6	Output enable for pin GPIO6
7	0	V_GPIO_EN7	Output enable for pin GPIO7

R_GPIO_EN1	(w)	0x43

GPIO output enable bits 15..8

Every bit value '1' enables the allocated output driver. If an output driver is disabled (bit value '0'), the pin is used for data input.

Bits	Reset value	Name	Description
0	0	V_GPIO_EN8	Output enable for pin GPIO8
1	0	V_GPIO_EN9	Output enable for pin GPIO9
2	0	V_GPIO_EN10	Output enable for pin GPIO10
3	0	V_GPIO_EN11	Output enable for pin GPIO11
4	0	V_GPIO_EN12	Output enable for pin GPIO12
5	0	V_GPIO_EN13	Output enable for pin GPIO13
6	0	V_GPIO_EN14	Output enable for pin GPIO14
7	0	V_GPIO_EN15	Output enable for pin GPIO15

R_GPIO_SEL	(w)	0x44

GPIO selection register

This register allows to select the first or second function of GPIO pins. Always two pins are controlled by one register bit. Every bit controls only the output driver, whereas the input functionality needs no programming.

Bits	Reset value	Name	Description
0	0	V_GPIO_SEL0	**GPIO0 and GPIO1** '0' = pins T_A and T_B enabled '1' = pins GPIO0 and GPIO1 enabled
1	0	V_GPIO_SEL1	**GPIO2 and GPIO3** '0' = pins GPIO2 and GPIO3 disabled '1' = pins GPIO2 and GPIO3 enabled
2	0	V_GPIO_SEL2	**GPIO4 and GPIO5** '0' = pins GPIO4 and GPIO5 disabled '1' = pins GPIO4 and GPIO5 enabled
3	0	V_GPIO_SEL3	**GPIO6 and GPIO7** '0' = pins GPIO6 and GPIO7 disabled '1' = pins GPIO6 and GPIO7 enabled
4	0	V_GPIO_SEL4	**GPIO8 and GPIO9** '0' = pins GPIO8 and GPIO9 disabled '1' = pins GPIO8 and GPIO9 enabled
5	0	V_GPIO_SEL5	**GPIO10 and GPIO11** '0' = pins GPIO10 and GPIO11 disabled '1' = pins GPIO10 and GPIO11 enabled
6	0	V_GPIO_SEL6	**GPIO12 and GPIO13** '0' = pins GPIO12 and GPIO13 disabled '1' = pins GPIO12 and GPIO13 enabled
7	0	V_GPIO_SEL7	**GPIO14 and GPIO15** '0' = pins GPIO14 and GPIO15 disabled '1' = pins GPIO14 and GPIO15 enabled

13.4.2 Read only registers

R_GPIO_IN0	(r)	0x40

GPIO input data bits 7 . . 0

Bits	Reset value	Name	Description
0		V_GPIO_IN0	Input data bit from pin GPIO0
1		V_GPIO_IN1	Input data bit from pin GPIO1
2		V_GPIO_IN2	Input data bit from pin GPIO2
3		V_GPIO_IN3	Input data bit from pin GPIO3
4		V_GPIO_IN4	Input data bit from pin GPIO4
5		V_GPIO_IN5	Input data bit from pin GPIO5
6		V_GPIO_IN6	Input data bit from pin GPIO6
7		V_GPIO_IN7	Input data bit from pin GPIO7

R_GPIO_IN1	(r)	0x41

GPIO input data bits 15..8

Bits	Reset value	Name	Description
0		V_GPIO_IN8	Input data bit from pin GPIO8
1		V_GPIO_IN9	Input data bit from pin GPIO9
2		V_GPIO_IN10	Input data bit from pin GPIO10
3		V_GPIO_IN11	Input data bit from pin GPIO11
4		V_GPIO_IN12	Input data bit from pin GPIO12
5		V_GPIO_IN13	Input data bit from pin GPIO13
6		V_GPIO_IN14	Input data bit from pin GPIO14
7		V_GPIO_IN15	Input data bit from pin GPIO15

R_GPI_IN0	(r)	0x44

GPI input data bits 7..0

Note: Unused GPI pins should be connected to ground or power supply.

Bits	Reset value	Name	Description
0		V_GPI_IN0	Input data bit from pin GPI0
1		V_GPI_IN1	Input data bit from pin GPI1
2		V_GPI_IN2	Input data bit from pin GPI2
3		V_GPI_IN3	Input data bit from pin GPI3
4		V_GPI_IN4	Input data bit from pin GPI4
5		V_GPI_IN5	Input data bit from pin GPI5
6		V_GPI_IN6	Input data bit from pin GPI6
7		V_GPI_IN7	Input data bit from pin GPI7

R_GPI_IN1	(r)	0x45

GPI input data bits 15 . . 8

Note: Unused GPI pins should be connected to ground or power supply.

Bits	Reset value	Name	Description
0		V_GPI_IN8	Input data bit from pin GPI8
1		V_GPI_IN9	Input data bit from pin GPI9
2		V_GPI_IN10	Input data bit from pin GPI10
3		V_GPI_IN11	Input data bit from pin GPI11
4		V_GPI_IN12	Input data bit from pin GPI12
5		V_GPI_IN13	Input data bit from pin GPI13
6		V_GPI_IN14	Input data bit from pin GPI14
7		V_GPI_IN15	Input data bit from pin GPI15

R_GPI_IN2	(r)	0x46

GPI input data bits 23 . . 16

Note: Unused GPI pins should be connected to ground or power supply.

Bits	Reset value	Name	Description
0		V_GPI_IN16	Input data bit from pin GPI16
1		V_GPI_IN17	Input data bit from pin GPI17
2		V_GPI_IN18	Input data bit from pin GPI18
3		V_GPI_IN19	Input data bit from pin GPI19
4		(reserved)	
5		V_GPI_IN21	Input data bit from pin GPI21
6		V_GPI_IN22	Input data bit from pin GPI22
7		V_GPI_IN23	Input data bit from pin GPI23

R_GPI_IN3	(r)	0x47

GPI input data bits 31 . . 24

Note: Unused GPI pins should be connected to power supply to prevent from external interrupt requests due to floating input pins. Unused GPI pins can be connected to ground, alternatively, if the external interrupt functionality is disabled.

Bits	Reset value	Name	Description
0		V_GPI_IN24	Input data bit from pin GPI24
1		V_GPI_IN25	Input data bit from pin GPI25
2		V_GPI_IN26	Input data bit from pin GPI26
3		V_GPI_IN27	Input data bit from pin GPI27
4		V_GPI_IN28	Input data bit from pin GPI28
5		V_GPI_IN29	Input data bit from pin GPI29
6		V_GPI_IN30	Input data bit from pin GPI30
7		V_GPI_IN31	Input data bit from pin GPI31

Chapter 14

Electrical characteristics

Absolute maximum ratings [*1]

Parameter	Symbol	Min.	Max.
Power supply	V_{DD}	-0.3 V	+4.6 V
Input voltage	V_I	-0.3 V	6.0 V [*2]
Operating temperature	T_{opr}	-30 °C	+70 °C
Junction temperature	T_{jnc}	0 °C	+100 °C
Storage temperature	T_{stg}	-55 °C	+125 °C

Recommended operating conditions

Parameter	Symbol	Min.	Typ.	Max	Conditions
Power supply	V_{DD}	3.0 V	3.3 V	3.6 V	
Operating temperature	T_{opr}	0 °C		+70 °C	

Electrical characteristics for 3.3 V power supply

Parameter	Symbol	Min.	Typ.	Max	Conditions
Low input voltage	V_{IL}	-0.3 V		$0.2V_{DD}$	
High input voltage	V_{IH}	$0.7V_{DD}$		5.5 V [*2]	
Low output voltage	V_{OL}	0 V		0.4 V	
High output voltage	V_{OH}	2.4 V		V_{DD}	
Power supply current	I_{opr}		80 mA [*3]		$T_{opr} = 25$ °C

[*1]: Stresses beyond those listed under 'Absolute maximum ratings' may cause permanent damage to the device. These are stress ratings only, and operation of the device at these or at any other conditions above those given in this data sheet is not implied. Exposure to limiting values for extended periods may affect device reliability.

[*2]: Maximum voltage for oscillator pins is V_{DD}.

[*3]: Power supply current at full operating condition without external E1 interface circuitry.

Chapter 15

HFC-E1 package dimensions

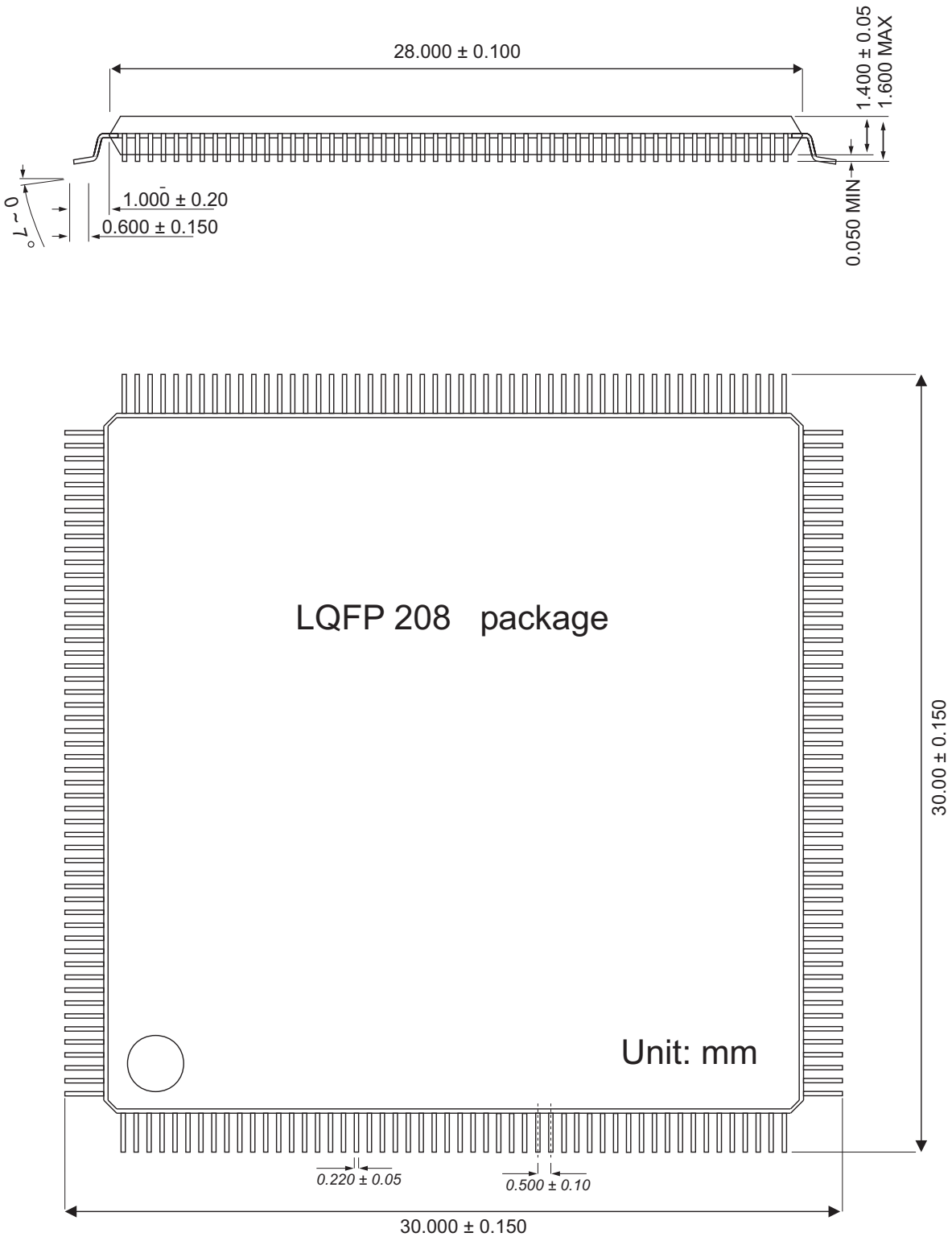


Figure 15.1: *HFC-E1 package dimensions*

References

[1] Cologne Chip AG. *DTMF Algorithm for HFC-8S / HFC-4S. Application Note*, September 2002.

[2] European Telecommunications Standards Institute. *Technical Basis for Regulation (TBR 4): Integrated Services Digital Network (ISDN); Attachment requirements for terminal equipment to connect to an ISDN using ISDN primary rate access*.

[3] Telecommunication Standardization Sector of International Telecommunication Union (ITU). *ITU-T I.431: Integrated services digital network (ISDN); ISDN user-network interfaces. Primary rate user-network interface – Layer 1 specification*, March 1993.

[4] Telecommunication Standardization Union (ITU). Q.24: Multifrequency push-button signal reception. In *General recommendations on telephone switching and signalling. International Automatic and semi-automatic working*. ITU-T (Telecommunication Standardization Sector of ITU), 1988, 1993.

[5] Telecommunication Standardization Union (ITU). O.151: Error performance measuring equipment operating at the primary rate and above. In *Specification of measurement equipment*. ITU-T (Telecommunication Standardization Sector of ITU), 1992.

[6] Telecommunication Standardization Union (ITU). O.150: Equipment for the measurement of digital and analogue/digital parameters. In *General requirements for instrumentation for performance measurements on digital transmission equipment*. ITU-T (Telecommunication Standardization Sector of ITU), 1996.

List of register and bitmap abbreviations

This list shows all abbreviations which are used to define the register and bitmap names. Appended digits are not shown here except they have a particular meaning.

16KHZ	16 kHz
A	A-bit
ADDR	address
ADJ	adjust
AIS	alarm indication signal
ALT	alternate
ATT	attenuation
ATX	analog transmitter
AUTO	automatic
BERT	bit error rate test
BIT	bit
BL	block
BRG	bridge
BUSY	busy
C4	C4IO clock (PCM double bit clock)
CFG	configuration
CH	HFC-channel
CHANNEL	HFC-channel
CHBL	HFC-channel block
CHG	changed
CHIP	microchip
CIRM	configuration, interrupt and reset
CLK	clock
CNT	counter
CNTH	counter, high byte
CNTL	counter, low byte
CODE	code
CODEC	CODEC
CON	connection settings
COND	condition
CONF	conference
CRC	cyclic redundancy check
CTRL	control
DATA	data
DF	data flow
DIR	direction
DR	data rate
DTMF	dual tone multiple frequency
E	CRC-4 error indication bits
E1	E1 interface
E1	CRC-4 error indication bit E1
E2	CRC-4 error indication bit E2
ECH	error counter, high byte
ECL	error counter, low byte
EN	enable
END	end
EOMF	end of multiframe
EPR	EEPROM
ERR	error
EV	event
EXCHG	exchange
EXT	external
F	*F*-counter
F0	frame synchronization signal
F1	*F*1-counter
F12	*F*1- and *F*2-counters
F2	*F*2-counter
FAS	frame alignment signal
FBAUD	full bauded
FDIR	direction (FIFO-related)
FIFO	FIFO
FIRST	first
FLOW	flow
FNUM	number (FIFO-related)
FR	frame
FRQ	frequency
FSM	FIFO sequence mode
FZ	*F*- and *Z*-counters
GLOB	global
GPI	general purpose input
GPIO	general purpose input / output
HARM	harmonic
HCLK	half clock (frequency)
HDLC	high-level data link control

HFC	HDLC FIFO controller
ICR	increase
ID	identifier
IDX	index
IFF	inter frame fill
IN	input
INC	increment
INIT	initialization
INSYNC	synchronization to input data
INT	internal
INV	invert, inversion
IPATS	IPATS test
IRQ	interrupt
IRQ1S	one-second interrupt
IRQMSK	interrupt mask
IRQSTA	interrupt status
ITU	International Telecommunication Union
IV	interval
JATT	jitter attenuator
LD	load
LEN	length
LEV	level
LOOP	loop
LOS	loss of signal
LOSS	loss of synchronization signal
LOST	frame data lost
LPRIO	low priority
MAN	manual
MD	mode
MF	multiframe
MFA	multiframe alignment
MISC	miscellaneous
MIX	mixed
MSK	mask
N	number of samples
NEG	negative
NEXT	next
NFAS	no frame alignment signal
NMF	no multiframe
NO	no
NOINC	no increment
NOISE	noise
NTRI	no tristate
NUM	number
OFF	off
OFFS	offset
OFLOW	overflow
OK	ok
OUT	output
OVIEW	overview
PAR	parameter
PAT	pattern
PCM	pulse code modulation
PH	phase
PLL	phase locked loop
PNP	plug and play
POL	polarity
PROC	processing
PWM	pulse width modulation
RAL	remote alarm
RAM	RAM
RD	read
RECO	recovery
RES	reset
RESYNC	resynchronization
REV	reverse
RLD	reload
ROUT	routing
RST	restart
RV	revision
RX	receive
SA	spare bits S_a
SA6	spare bit S_{a6}
SA61	spare bit $S_{a6}[1]$
SA62	spare bit $S_{a6}[2]$
SA63	spare bit $S_{a6}[3]$
SA64	spare bit $S_{a6}[4]$
SDIR	direction (slot-related)
SEL	select, selection
SEQ	sequence
SET	setup
SGN	sign
SH	shape
SH0H	shape 0, high byte
SH0L	shape 0, low byte
SH1H	shape 1, high byte
SH1L	shape 1, low byte
SI	spare bits S_i
SIG	signal
SIM	simulation
SL	time slot
SL0	time slot 0
SLIP	frequency slip
SLOT	time slot
SLOW	slow
SNUM	number (slot-related)
SRAM	SRAM
SRC	source
SRES	soft reset
ST	S/T interface
STA	state, status
START	start
STATUS	status
STOP	stop
SUBCH	subchannel
SUPPR	suppression
SYNC	synchronize, synchronization
SZ	size
TI	timer
TRP	transparent
TS	time step
TX	transmit
ULAW	μ-law
USE	use, usage
VAL	value
VIO	code violation
WD	watchdog timer
WR	write

WRDLY	write delay
XCRC	extended CRC-4
XS13	Spare bit of frame 13
XS15	Spare bit of frame 15
Z1	*Z*1-counter
Z12	*Z*1- and *Z*2-counters
Z1H	*Z*1-counter, high byte
Z1L	*Z*1-counter, low byte
Z2	*Z*2-counter
Z2H	*Z*2-counter, high byte
Z2L	*Z*2-counter, low byte

Cologne Chip AG

Data Sheet of HFC-E1