DSCC4

DMA Supported Serial Communication Controller with 4 Channels

PEB 20534 Version 2.1
PEF 20534 Version 2.1

Datacom

Never stop thinking.

PEB 20534 PEF 20534 **Revision History:**		**Current Version: 2000-05-30**
Previous Version:		Data Sheet 09.98 (V 2.0)
Page (in previous Version)	Page (in current Version)	Subjects (major changes since last revision)
-	-	removed remaining references to command bit GCMDR:IADC
429-438	427-435	corrected timing values #81-#86, #132, #149

For questions on technology, delivery and prices please contact the Infineon Technologies Offices in Germany or the Infineon Technologies Companies and Representatives worldwide:
see our webpage at http://www.infineon.com

Edition 2000-05-30

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
D-81541 München, Germany

Attention please!

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or our Infineon Technologies Representatives worldwide (see address list).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

Preface

The *DMA Supported Serial Communication Controller with 4 Channels* (DSCC4) is a Multi Protocol Controller for a wide range of data communication and telecommunication applications. This document provides complete reference information on hardware and software related issues as well as on general operation.

Organization of this Document

This Data Sheet is divided into 15 chapters. It is organized as follows:

- Chapter 1, Overview
  Gives a general description of the product, lists the key features, and presents some typical applications.

- Chapter 2, Pin Description
  Lists pin locations with associated signals, categorizes signals according to function, and describes signals.

- Chapters 3,4,5,6,7 Functional Description
  These chapters provide detailed descriptions of all DSCC4 internal function blocks.

- Chapter 8, Detailed Protocol Descriptions
  Gives a detailed description of all protocols supported by the serial communication controllers SCCs.

- Chapter 9, Reset and Initialization Procedure
  Gives examples for DSCC4 initialization procedure and operation.

- Chapter 10, Detailed Register Description
  Gives a detailed description of all DSCC4 on chip registers.

- Chapter 11, Host Memory Organization
  Provides an overview of all DSCC4 data structures located in the shared memory

- Chapter 12, JTAG Boundary Scan
  Gives a detailed description of the boundary scan unit.

- Chapter 13, Electrical Characteristics
  Gives a detailed description of all electrical DC and AC characteristics and provides timing diagrams and values for all interfaces.

- Chapter 14, Package Outline

Table of Contents **Page**

1 Overview . . . 18
1.1 Features . . . 19
1.2 Differences between the DSCC4 and the ESCC Family . . . 22
1.2.1 Enhancements to the ESCC Serial Core . . . 22
1.2.2 Simplifications to the ESCC Serial Core . . . 22
1.3 Logic Symbol . . . 23
1.4 Typical Applications . . . 24
1.4.1 Application Examples . . . 26
1.4.1.1 HSSI Application - DCE Adapter . . . 26
1.4.1.2 HSSI Application - DTE Adapter . . . 27
1.4.1.3 General Data Application . . . 28

2 Pin Descriptions . . . 29
2.1 Pin Diagram . . . 29
2.2 Pin Definitions and Functions . . . 30

3 Functional Description . . . 48

4 Microprocessor Bus Interface . . . 51
4.1 PCI Bus Interface . . . 51
4.1.1 Supported PCI Transactions . . . 51
4.1.2 PCI Configuration Space Register Overview . . . 52
4.2 De-multiplexed Bus Interface Extension . . . 53

5 DMA Controller and Central FIFOs . . . 57
5.1 DMAC Operational Description . . . 59
5.1.1 DMAC Register Overview . . . 59
5.1.2 DMAC Control and Data Structures . . . 64
5.1.2.1 DMAC Transmit Descriptor Lists . . . 66
5.1.2.2 DMAC Receive Descriptor Lists . . . 71
5.1.2.3 DMAC Operation Using Hold-Bit Control Mechanism . . . 76
5.1.2.4 DMAC Operation Using Last Descriptor Address Control Mode . . . 78
5.1.3 DMAC Interrupt Controller . . . 81
5.2 Central FIFOs Operational Description . . . 85
5.2.1 Central FIFO Register Overview . . . 85
5.2.2 Central Transmit FIFO (TFIFO) . . . 89
5.2.3 Central Receive FIFO (RFIFO) . . . 92
5.2.4 DMAC Internal Arbitration Scheme . . . 93
5.2.5 DMAC Performance . . . 94
5.2.6 Little / Big Endian Byte Swap Convention . . . 95

6 Multi Function Port (MFP) . . . 96
6.1 Local Bus Interface (LBI) . . . 98
6.1.1 LBI Bus Modes . . . 99

Table of Contents **Page**

6.1.1.1 LBI External Bus Controller (EBC) . . . 99
6.1.1.2 Multiplexed Local Bus Modes . . . 99
6.1.1.3 De-multiplexed Local Bus Modes . . . 100
6.1.1.4 Programmable Bus Characteristics . . . 102
6.1.1.5 Ready Signal Controlled Bus Cycles . . . 104
6.1.1.6 LBI (EBC) Configuration . . . 106
6.1.2 LBI Bus Arbitration . . . 107
6.1.3 PCI to Local Bus Bridge Operation . . . 113
6.1.4 LBI Interrupt Generation . . . 114
6.2 Synchronous Serial Control (SSC) Interface . . . 115
6.2.1 SSC Functional Description . . . 115
6.2.1.1 Overview . . . 115
6.2.1.2 Operational Mode: Full-Duplex Operation: . . . 119
6.2.1.3 Operational Mode: Half Duplex Operation: . . . 122
6.2.1.4 Baud Rate Generation . . . 124
6.2.1.5 Error Detection . . . 124
6.2.2 SSC Interrupt (Vector) . . . 126
6.3 General Purpose Port (GPP) Interface . . . 127
6.3.1 GPP Functional Description . . . 127
6.3.2 GPP Interrupt Vector . . . 127

7 Serial Communication Controller (SCC) Cores . . . 128
7.1 General . . . 128
7.2 Protocol Modes Overview . . . 132
7.3 SCC FIFOs . . . 133
7.3.1 SCC Transmit FIFO . . . 133
7.3.2 SCC Receive FIFO . . . 134
7.4 Clocking System . . . 137
7.4.1 Clock Modes . . . 141
7.4.1.1 Clock Mode 0 (0a/0b) . . . 141
7.4.1.2 Clock Mode 1 . . . 142
7.4.1.3 Clock Mode 2 (2a/2b) . . . 143
7.4.1.4 Clock Mode 3 (3a/3b) . . . 144
7.4.1.5 Clock Mode 4 (High Speed Interface Clock Mode) . . . 145
7.4.1.6 Clock Mode 5 . . . 147
7.4.1.7 Clock Mode 6 (6a/6b) . . . 152
7.4.1.8 Clock Mode 7 (7a/7b) . . . 153
7.4.2 Baud Rate Generator (BRG) . . . 154
7.4.3 Clock Recovery (DPLL) . . . 155
7.5 SCC Interrupt Interface . . . 158
7.6 High Speed Channel Operation (PEB 20534H-52 only) . . . 159
7.7 Serial Bus Configuration Mode . . . 159
7.7.1 Serial Bus Access Procedure . . . 160

Table of Contents **Page**

7.7.2 Serial Bus Collisions and Recovery . . . 160
7.7.3 Serial Bus Access Priority Scheme . . . 161
7.7.4 Serial Bus Configuration Timing Modes . . . 161
7.7.5 Functions Of Signal RTS in Serial Bus Configuration . . . 162
7.8 Data Encoding . . . 162
7.8.1 NRZ and NRZI Encoding . . . 162
7.8.2 FM0 and FM1 Encoding . . . 163
7.8.3 Manchester Encoding . . . 164
7.9 Modem Control Signals (RTS, CTS, CD) . . . 164
7.9.1 RTS/CTS Handshaking . . . 164
7.9.2 Carrier Detect (CD) Receiver Control . . . 166
7.10 Local Loop Test Mode . . . 166

8 Detailed Protocol Description . . . 167
8.1 HDLC/SDLC Protocol Modes . . . 169
8.1.1 HDLC . . . 169
8.1.1.1 Auto Mode . . . 170
8.1.1.2 Non Auto Mode . . . 170
8.1.1.3 Address Mode 1 . . . 171
8.1.1.4 Address Mode 0 . . . 171
8.1.2 HDLC/PPP Protocol Mode . . . 171
8.1.2.1 Bit Synchronous PPP . . . 171
8.1.2.2 Octet Synchronous PPP . . . 172
8.1.2.3 Aynchronous PPP . . . 172
8.1.3 Extended Transparent Mode . . . 172
8.1.4 HDLC Receive Data Processing Overview . . . 172
8.1.5 HDLC Transmit Data Processing Overview . . . 174
8.1.6 Procedural Support (Layer-2 Functions) . . . 176
8.1.7 Full-Duplex LAPB/LAPD Operation . . . 177
8.1.8 Half-Duplex SDLC-NRM Operation . . . 182
8.1.9 Special Functions . . . 185
8.1.9.1 Extended Transparent Transmission and Reception . . . 185
8.1.9.2 Receive Address Handling . . . 185
8.1.9.3 Shared Flags . . . 185
8.1.9.4 One Bit Insertion . . . 186
8.1.9.5 Preamble Transmission . . . 186
8.1.9.6 CRC Generation and Checking . . . 186
8.1.9.7 Data Transparency in PPP Mode . . . 188
8.1.9.8 Receive Length Check Feature . . . 190
8.2 Asynchronous (ASYNC) Protocol Mode . . . 191
8.2.1 Character Framing . . . 191
8.2.2 Data Reception . . . 191
8.2.2.1 Asynchronous Mode . . . 191

Table of Contents **Page**

8.2.2.2 Isochronous Mode . . . 192
8.2.2.3 Storage of Receive Data . . . 192
8.2.3 Data Transmission . . . 193
8.2.4 Special Functions . . . 193
8.2.4.1 Break Detection/Generation . . . 193
8.2.4.2 In-band Flow Control by XON/XOFF Characters . . . 193
8.2.4.3 Out-of-band Flow Control . . . 195
8.3 Character Oriented Synchronous (BISYNC) Protocol Mode . . . 198
8.3.1 Character Framing . . . 198
8.3.2 Data Reception . . . 198
8.3.3 Data Transmission . . . 199
8.3.4 Special Functions . . . 200
8.3.4.1 Preamble Transmission . . . 200
8.3.4.2 CRC Parity Inhibit . . . 200

9 Reset and Initialization Procedure . . . 201
9.1 Reset and Power-On . . . 201
9.2 Initialization Example . . . 203
9.3 Start of Operation . . . 207
9.4 Initialization Example . . . 214
9.4.1 Test Loop For Data Transfer in HDLC Address Mode 0 . . . 214

10 Detailed Register Description . . . 220
10.1 Register Range Overview and Address Mapping . . . 221
10.2 PCI Configuration Space - Detailed Register Description . . . 222
10.3 On-Chip Registers Description . . . 228
10.3.1 Global Registers - Detailed Register Description . . . 228
10.3.1.1 Global Registers Overview . . . 228
10.3.1.2 Global Registers Description . . . 231
10.3.2 SCC Registers - Detailed Register Description . . . 272
10.3.2.1 SCC Registers Overview . . . 272
10.3.2.2 SCC Registers Description . . . 274
10.3.3 Peripheral Registers - Detailed Register Description . . . 349
10.3.3.1 Peripheral Registers Overview . . . 349
10.3.3.2 LBI Registers Description . . . 351
10.3.3.3 SSC Registers Description . . . 355
10.3.3.4 GPP Registers Description . . . 368

11 Host Memory Organization . . . 374
11.1 Linked List Structure . . . 374
11.1.1 Transmit Descriptor Lists . . . 374
11.1.1.1 Transmit Descriptor . . . 374
11.1.2 Receive Descriptor Lists . . . 378
11.1.2.1 Receive Descriptor . . . 379

Table of Contents **Page**

11.1.2.2 Receive Data Section Status Byte (HDLC Mode) . . . 383
11.1.2.3 Receive Data Section Status Byte (ASYNC/BISYNC Modes) . . . 386
11.2 Interrupt Queue Structure . . . 387
11.2.1 Interrupt Queue Overview . . . 387
11.2.2 Interrupt Vector Overview . . . 389
11.2.2.1 Configuration Interrupt Vector . . . 390
11.2.2.2 DMA Controller Interrupt Vector . . . 391
11.2.2.3 SCC Interrupt Vector . . . 393
11.2.2.4 SSC Interrupt Vector . . . 395
11.2.2.5 LBI Interrupt Vector . . . 397
11.2.2.6 GPP Interrupt Vector . . . 398

12 Test Configuration . . . 399
12.1 JTAG Boundary Scan Interface . . . 399

13 Electrical Characteristics . . . 406
13.1 Important Electrical Requirements . . . 406
13.2 Absolute Maximum Ratings . . . 408
13.3 Thermal Package Characteristics . . . 408
13.4 DC Characteristics . . . 409
13.5 Capacitances . . . 410
13.6 AC Characteristics . . . 411
13.6.1 PCI Bus Interface Timing . . . 411
13.6.1.1 PCI Read Transaction . . . 412
13.6.1.2 PCI Write Transaction . . . 414
13.6.2 De-multiplexed Bus Interface . . . 418
13.6.3 Local Bus Interface Timing . . . 420
13.6.4 Local Bus Interface Arbitration Timing . . . 425
13.6.5 PCM Serial Interface Timing . . . 426
13.6.5.1 Clock Input Timing . . . 426
13.6.5.2 Receive Cycle Timing . . . 428
13.6.5.3 Transmit Cycle Timing . . . 430
13.6.5.4 Strobe Timing (clock mode 1) . . . 432
13.6.5.5 Frame Synchronisation Timing (clock mode 5) . . . 433
13.6.5.6 High Speed Receive Cycle Timing . . . 434
13.6.5.7 High Speed Transmit Cycle Timing . . . 435
13.6.6 Reset Timing . . . 436
13.6.7 JTAG-Boundary Scan Timing . . . 437
13.6.8 SSC Serial Interface Timing . . . 438

14 Package Outlines . . . 439

List of Figures

Page

Figure 1 Logic Symbol . . . 23
Figure 2 General System Integration (PCI Bus Interface) . . . 24
Figure 3 General System Integration (De-multiplexed Interface) . . . 25
Figure 4 HSSI Application - DCE Adapter . . . 26
Figure 5 HSSI Application - DTE Adapter . . . 27
Figure 6 General Data Application . . . 28
Figure 7 Pin Configuration . . . 29
Figure 8 DSCC4 Functional Block Diagram . . . 48
Figure 9 Master Single READ Transaction followed by a Master Single WRITE Transaction in De-multiplexed Configuration . . . 54
Figure 10 Master Burst WRITE/READ Transaction in De-multiplexed Configuration . . . 55
Figure 11 DMA Controller Block Diagram . . . 58
Figure 12 DMA Data Flow . . . 64
Figure 13 Transmit Descriptor List Structure . . . 66
Figure 14 Transmit Descriptor Memory Example . . . 70
Figure 15 Receive Descriptor List Structure . . . 71
Figure 16 Receive Descriptor Memory Example . . . 75
Figure 17 Data Transfer controlled via first and last descriptor addresses . . . 78
Figure 18 Example: ’Chain Jump’ Handling per ’Dummy Descriptor’. . . . 80
Figure 19 DSCC4 Logical Interrupt Structure . . . 83
Figure 20 Central Transmit FIFO Section Thresholds . . . 90
Figure 21 Central Receive FIFO Threshold . . . 92
Figure 22 Little/Big Endian Byte Swapping . . . 95
Figure 23 MFP Configurations Overview . . . 96
Figure 24 Multiplexed Bus Cycle . . . 100
Figure 25 De-multiplexed Bus Cycle . . . 101
Figure 26 Memory Cycle Time . . . 103
Figure 27 LRDY Controlled Bus Cycles . . . 104
Figure 28 LRDY Timing . . . 105
Figure 29 External Bus Arbitration (Releasing the Bus) . . . 108
Figure 30 External Bus Arbitration (Regaining the Bus) . . . 109
Figure 31 Connection of the Master and Slave Bus Arbitration Signals . . . 110
Figure 32 Bus Arbitration Sequence . . . 112
Figure 33 Registers and Port Pins associated with the SSC . . . 116
Figure 34 Synchronous Serial Channel SSC Block Diagram . . . 117
Figure 35 Serial Clock Phase and Polarity Options . . . 119
Figure 36 SSC Full Duplex Configuration . . . 120
Figure 37 SSC Half Duplex Configuration . . . 123
Figure 38 SSC Error Interrupt Control . . . 126
Figure 39 SCC Transmit FIFO . . . 133
Figure 40 SCC Receive FIFO . . . 134

List of Figures

Page

Figure 41 Clock Supply Overview . . . 140
Figure 42 Clock Mode 0a/0b Configuration . . . 141
Figure 43 Clock Mode 1 Configuration . . . 142
Figure 44 Clock Mode 2a/2b Configuration . . . 143
Figure 45 Clock Mode 3a/3b Configuration . . . 144
Figure 46 Clock Mode 4 (High Speed) Configuration . . . 145
Figure 47 Selecting one time-slot of programmable delay and width . . . 148
Figure 48 Selecting one or more time-slots of 8-bit width . . . 150
Figure 49 Clock Mode 5 Configuration . . . 151
Figure 50 Clock Mode 6a/6b Configuration . . . 152
Figure 51 Clock Mode 7a/7b Configuration . . . 153
Figure 52 DPLL Algorithm for NRZ and NRZI Coding with Phase Shift Enabled (CCR0:PSD = ‘0’) . . . 156
Figure 53 DPLL Algorithm for NRZ and NRZI Encoding with Phase Shift Disabled (CCR0:PSD = ‘1’) . . . 157
Figure 54 DPLL Algorithm for FM0, FM1 and Manchester Coding . . . 157
Figure 55 Request-to-Send in Bus Operation . . . 162
Figure 56 NRZ and NRZI Data Encoding . . . 163
Figure 57 FM0 and FM1 Data Encoding . . . 163
Figure 58 Manchester Data Encoding . . . 164
Figure 59 RTS/CTS Handshaking . . . 165
Figure 60 SCC Test Loop . . . 166
Figure 61 SCC Receive Data Flow (HDLC Modes) part a) . . . 173
Figure 62 SCC Receive Data Flow (HDLC Modes) part b) . . . 174
Figure 63 SCC Transmit Data Flow (HDLC Modes) . . . 175
Figure 64 Processing of Received Frames in Auto Mode . . . 178
Figure 65 Timer Procedure/Poll Cycle . . . 180
Figure 66 Transmission/Reception of I-Frames and Flow Control . . . 181
Figure 67 Flow Control: Reception of S-Commands and Protocol Errors . . . 181
Figure 68 No Data to Send: Data Reception/Transmission . . . 184
Figure 69 Data Transmission (without error), Data Transmission (with error) . . . 184
Figure 70 PPP Mapping/Unmapping Example . . . 189
Figure 71 Asynchronous Character Frame . . . 191
Figure 72 Out-of-Band DTE-DTE Bi-directional Flow Control . . . 196
Figure 73 Out-of-Band DTE-DCE Bi-directional Flow Control . . . 197
Figure 74 BISYNC Message Format . . . 198
Figure 75 Data Structures in shared Memory before Transmission . . . 214
Figure 76 Data Stuctures in shared Memory after Transmission . . . 219
Figure 77 Transmit Descriptor List Structure . . . 374
Figure 78 Receive Descriptor List Structure . . . 378
Figure 79 ASYNC/BISYNC Receive Status Character Format . . . 386
Figure 80 DSCC4 Logical Interrupt Structure . . . 388

List of Figures

Page

Figure 81 Interrupt Vector Overview . . . 389
Figure 82 Block Diagram of Test Access Port and Boundary Scan Unit . . . 399
Figure 83 Power-up and Power-down scenarios . . . 406
Figure 84 Power-Failure scenarios . . . 407
Figure 85 Input/Output Waveform for AC Tests . . . 411
Figure 86 PCI Output Timing Measurement Waveforms . . . 411
Figure 87 PCI Input Timing Measurement Waveforms . . . 412
Figure 88 PCI Read Transaction . . . 413
Figure 89 PCI Write Transaction . . . 414
Figure 90 PCI Clock Specification . . . 416
Figure 91 Master Single READ Transaction followed by a Master Single WRITE Transaction in De-multiplexed Bus Configuration . . . 418
Figure 92 Master Burst WRITE/READ Access in De-multiplexed Bus Configuration . . . 418
Figure 93 Synchronous LBI Read Cycle Timing Multiplexed Bus . . . 420
Figure 94 Synchronous LBI Write Cycle Timing Multiplexed Bus . . . 420
Figure 95 Synchronous LBI Read Cycle Timing De-multiplexed Bus . . . 422
Figure 96 Synchronous LBI Write Cycle Timing De-multiplexed Bus . . . 422
Figure 97 LRDY Timing . . . 424
Figure 98 LBI Arbitration Timing . . . 425
Figure 99 Clock Input Timing . . . 426
Figure 100 Receive Cycle Timing . . . 428
Figure 101 Transmit Cycle Timing . . . 430
Figure 102 Strobe Timing . . . 432
Figure 103 Frame Synchronisation Timing . . . 433
Figure 104 High Speed Receive Timing . . . 434
Figure 105 High Speed Transmit Timing . . . 435
Figure 106 Reset Timing . . . 436
Figure 107 JTAG-Boundary Scan Timing . . . 437
Figure 108 SSC Interface Timing (Master) . . . 438
Figure 109 Package Outline . . . 439

List of Tables

Page

Table 1 PCI Bus Interface(DEMUX Interface) . . . 31
Table 2 Dedicated Signals . . . 37
Table 3 JTAG Test Port for Boundary Scan according to IEEE 1149.1 . . . 38
Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins . . . 39
Table 5 Serial Communication Controller (SCC) Signals . . . 43
Table 6 PCI Configuration Space Registers . . . 52
Table 7 Non-PCI Signal Extension in the De-multiplexed Bus Interface Mode. 53
Table 8 DEMUX Mode Related Register and Bit-Fields . . . 54
Table 9 Supported Commands in De-multiplexed Bus Mode . . . 56
Table 10 DMA Controller Register Set . . . 59
Table 11 DMAC Commands . . . 62
Table 12 Transmit Descriptor Bit Field Description . . . 67
Table 13 Meaning of ADD in Little/Big Endian Mode . . . 69
Table 14 Receive Descriptor Bit Field Description . . . 72
Table 15 Receive Data Buffer Section . . . 74
Table 16 Central FIFO Control Registers . . . 85
Table 17 MFP Configuration via GMODE Register, Bit Field 'PERCFG': . . . 97
Table 18 LBI Peripheral Transaction Options . . . 98
Table 19 Protocol Modes . . . 132
Table 20 Overview of Clock Modes . . . 137
Table 21 Clock Modes of the SCCs . . . 138
Table 22 BRR Register and Bit-Fields . . . 154
Table 23 Protocol Mode Overview . . . 168
Table 24 Address Comparison Overview . . . 169
Table 25 Error Handling . . . 182
Table 26 Status after Hardware Reset . . . 201
Table 27 Global Configuration of DSCC4 and Initialization of DMAC (Interrupt Channel) . . . 203
Table 28 Initialization of DMAC (Data Channels) . . . 204
Table 29 Initialization of the SCC(s) . . . 204
Table 30 Initialization of the MFP . . . 206
Table 31 Activation of DMAC and SCC . . . 207
Table 32 Continuous Operation of Data Transfer . . . 209
Table 33 Stop Data Transmission . . . 210
Table 34 Stop Data Reception . . . 210
Table 35 Exceptional handling in Case of Receive Data Overflow . . . 212
Table 36 Exceptional handling in Case of Transmit Data Underrun . . . 212
Table 37 Register Initialization for HDLC Transparent Mode 0, Test Loop. . . . 215
Table 38 Register Range and Address Mapping . . . 221
Table 39 DSCC4: PCI Configuration Space Register Set . . . 222
Table 40 PCI Base Address Ranges . . . 223

List of Tables

Page

Table 41 PCI Configuration Space: Status/Command Register 224
Table 42 Status and Command register bits. 225
Table 43 DSCC4 Global Register Overview 228
Table 44 GCMDR: Global Command Register 232
Table 45 GSTAR: Global Status Register. 237
Table 46 GMODE: Global Mode Register. 241
Table 47 IQLENR0: Interrupt Queue Length Register 0 246
Table 48 IQLENR1: Interrupt Queue Length Register 1 248
Table 49 IQSCCiRXBAR:
Interrupt Queue SCCi Receiver Base Address Register (i=0...3) . . . 250
Table 50 IQSCCiTXBAR:
Interrupt Queue SCCi Receiver Base Address Register (i=0...3) . . . 251
Table 51 IQCFGBAR:
Interrupt Queue Configuration Base Address Register 252
Table 52 IQPBAR:
Interrupt Queue Peripheral Base Address Register. 253
Table 53 FIFOCR1: FIFO Control Register 1 254
Table 54 FIFOCR2: FIFO Control Register 2 255
Table 55 FIFOCR3: FIFO Control Register 3 257
Table 56 FIFOCR4: FIFO Control Register 4 259
Table 57 CHiCFG: Channel i Configuration Register (i=3...0) 261
Table 58 CHiBRDA:
Channel i Base Receive Descriptor Address Register (i=3...0) 264
Table 59 CHiBTDA:
Channel i Base Transmit Descriptor Address Register (i=3...0) 265
Table 60 CHiFRDA:
Channel i First (Current) Receive Descriptor
Address Register (i=3...0) 266
Table 61 CHiFTDA:
Channel i First (Current) Transmit Descriptor
Address Register (i=3...0) 267
Table 62 CHiLRDA:
Channel i Last Receive Descriptor Address Register (i=3...0). 268
Table 63 CHiLTDA:
Channel i Last Transmit Descriptor Address Register (i=3...0) 270
Table 64 SCC Register Overview 273
Table 65 CMDR: Command Register 275
Table 66 STAR: Status Register. 279
Table 67 CCR0: Channel Configuration Register 0 283
Table 68 CCR1: Channel Configuration Register 1 288
Table 69 CCR2: Channel Configuration Register 2 296
Table 70 ACCM: PPP ASYNC Control Character Map 306

List of Tables

Page

Table 71 UDAC: User Defined PPP ASYNC Control Character Map 308
Table 72 TTSA: Transmit Time Slot Assignment Register 310
Table 73 RTSA: Receive Time Slot Assignment Register 312
Table 74 PCMMTX: PCM Mask for Transmit Direction. 314
Table 75 PCMMRX: PCM Mask for Receive Direction 316
Table 76 BRR: Baud Rate Register 318
Table 77 TIMR: Timer Register. 320
Table 78 XADR: Transmit Address Register 323
Table 79 RADR: Receive Address Register 325
Table 80 RAMR: Receive Address Mask Register 327
Table 81 RLCR: Receive Length Check Register. 329
Table 82 XNXF: XON/XOFF In-Band Flow Control Character Register 331
Table 83 TCR: Termination Character Register 334
Table 84 TICR: Transmit Immediate Character Register 336
Table 85 SYNCR: Synchronization Character Register 338
Table 86 IMR: Interrupt Mask Register 340
Table 87 ISR: Interrupt Status Register 342
Table 88 DSCC4 Peripheral Register Overview 350
Table 89 LCONF: LBI Configuration Register. 351
Table 90 SSCCON: SSC Control Register 355
Table 91 SSCBR: SSC Baud Rate Register. 360
Table 92 SSC Baud Rate Values 361
Table 93 SSCTB: SSC Transmit Buffer Register 362
Table 94 SSCRB: SSC Receive Buffer Register 363
Table 95 SSCCSE: SSC Chip Select Enable Register. 364
Table 96 SSCIM: SSC Interrupt Mask Register 366
Table 97 GPDIR: GPP Direction Register 368
Table 98 GPDATA: GPP Data Register 370
Table 99 GPIM: GPP Interrupt Mask Register 372
Table 100 Transmit Descriptor 375
Table 101 Receive Descriptor. 379
Table 102 CFGIV: Configuration Interrupt Vectori 390
Table 103 DMA Controller Interrupt Vectori 391
Table 104 SCC Interrupt Vector 393
Table 105 SSC Interrupt Vector 395
Table 106 LBI Interrupt Vector 397
Table 107 GPP Interrupt Vectori. 398
Table 108 Boundary Scan Sequence of the DSCC4 400
Table 109 Boundary Scan Test Modes. 404
Table 110 Absolute Maximum Ratings 408
Table 111 Thermal Package Characteristics 408
Table 112 DC Characteristics (Non-PCI Interface Pins and Power Supply Pins) 409

List of Tables

Page

Table 113 DC Characteristics PCI Interface Pins 410
Table 114 Capacitances (Non-PCI Interface Pins) 410
Table 115 PCI Input and Output Measurement Conditions 412
Table 116 Number of Wait States Inserted by the DSCC4 as Initiator 415
Table 117 Number of Wait States Inserted by the DSCC4 as Slave 415
Table 118 PCI Clock Characteristics 416
Table 119 PCI Interface Signal Characteristics 417
Table 120 Additional De-multiplexed Interface Signal Characteristics 419
Table 121 LBI Timing (synchronous, multiplexed bus) 421
Table 122 LBI Timing (synchronous, de-multiplexed bus) 423
Table 123 LBI LRDY Timing 424
Table 124 LBI Arbitration Timing 425
Table 125 Clock Input Timing (non high speed modes) 426
Table 126 Clock Input Timing (high speed mode) 427
Table 127 Receive Cycle Timing 429
Table 128 Transmit Cycle Timing 431
Table 129 Strobe Timing (clock mode 1) 432
Table 130 Frame Synchronisation Timing (clock mode 5) 433
Table 131 High Speed Receive Timing 434
Table 132 High Speed Transmit Timing 435
Table 133 Reset Timing 436
Table 134 JTAG-Boundary Scan Timing 437
Table 135 SSC Interface Timing 438

1 Overview

The DSCC4 is a DMA Supported Serial Communication Controller with four independent serial channels[1]. The serial channels are derived from updated protocol logic of the ESCC device family providing a large set of protocol support and variety in serial interface configuration. This allows easy integration to different environments and applications.

A 33-MHz/32-bit PCI bus Master/Slave interface with integrated high performance DMA controllers provides data transfer from or to host memory with low bus utilization and easy software handshaking.

An additional de-multiplexed bus interface mode is provided for integration in non-PCI bus environments with little glue-logic depending on the bus type.

The DMA Controller operates on linked lists which are optimized for data communication applications. Different control mechanisms allow easy software development well adapted to the needs of special applications.

Large onchip FIFOs in combination with enhanced threshold control mechanisms allow decoupling of traffic requirements on host bus and serial interfaces with little exception probabilities such as data underuns or overflows.
In a PCI bus application an integrated Local Bus Interface (LBI) provides bridging functionality to non PCI peripherals such as framers or line interface units (LIUs).

A Synchronous Serial Control (SSC) interface as well as a General Purpose Port (GPP) allows covering application specific requirements without additional controllers.

Each of the four Serial Communication Controllers (SCC) contains an independent Baud Rate Generator, DPLL, programmable protocol processing (HDLC, BISYNC, ASYNC and PPP). Data rates of up to 2 Mbit/s (DPLL assisted modes, ASYNC, BISYNC), 10 Mbit/s (HDLC, PPP) and 52 Mbit/s (H-52 version) are supported. The channels can also handle a large set of layer-2 protocol functions reducing bus and host CPU load. Four channel specific timers are provided to support protocol functions.

The DSCC4 devices can be used in LAN-WAN inter-networking applications such as Routers, Switches and Trunk cards and support the common V.35, ISDN BRI (S/T) or Asynchronous Dial-up interfaces. Its new features provide powerful hardware and software interfaces to develop high performance systems.

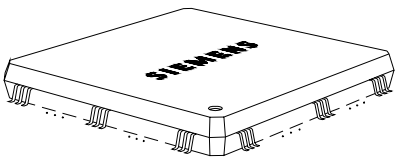
[1] The serial channels are also called ’ports’ or ’cores’ depending on the context.

DMA Supported Serial Communication Controller with 4 Channels
DSCC4

PEB 20534
PEF 20534

Version 2.1 **CMOS**

1.1 Features



P-FQFP-208-7

Serial Communication Controllers (SCCs)

- Four independent channels
- Full duplex data rates on each channel of up to 10 Mbit/s sync - 2 Mbit/s with DPLL, 2 Mbit/s async
- Full duplex data rate of up to 52 Mbit/s on any two channels in high speed mode (HDLC: Address Mode 0 and extended transparent protocol mode); up to 45 Mbit/s on any two channels in high speed mode (HDLC: PPP modes). The aggregate bandwith for all channels is limited to 108 Mbit/s per direction.
- 17 DWORDs deep receive FIFO per SCC (+ 128 DWORDs central receive FIFO).
- 8 DWORDs deep transmit FIFO per SCC (+ 128 DWORDs central transmit FIFO).

Serial Interface

- On-chip clock generation or external clock sources
- On-chip DPLLs for clock recovery
- Baud rate generator
- Clock gating signals
- Clock gapping capability
- Programmable time-slot capability for connection to TDM interfaces (e.g. T1, E1)
- NRZ, NRZI, FM and Manchester data encoding
- Optional data flow control using modem control lines ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, CD)
- Support of bus configuration by collision detection and resolution

Type	Package
PEB 20534 H-10	P-FQFP-208-7
PEF 20534 H-10	P-FQFP-208-7
PEB 20534 H-52	P-FQFP-208-7

- HDLC/SDLC Protocol Modes
  - Automatic flag detection and transmission
  - Shared opening and closing flag
  - Generation of interframe-time fill ’1’s or flags
  - Detection of receive line status
  - Zero bit insertion and deletion
  - CRC generation and checking (CRC-CCITT or CRC-32)
  - Transparent CRC option per channel and/or per frame
  - Programmable Preamble (8 bit) with selectable repetition rate
  - Error detection (abort, long frame, CRC error, short frames)
- Bit Synchronous PPP Mode
  - Bit oriented transmission of HDLC frame (flag, data, CRC, flag)
  - Zero bit insertion/deletion
  - 15 consecutive ’1’ bits abort sequence
- Octet Synchronous PPP Mode
  - Octet oriented transmission of HDLC frame (flag, data, CRC, flag)
  - Programmable character map of 32 hard-wired characters (00_H-$1F_H$)
  - Four programmable characters for additional mapping
  - Insertion/deletion of control-escape character ($7D_H$) for mapped characters
- Asynchronous PPP Mode
  - Character oriented transmission of HDLC frame (flag, data, CRC, flag)
  - Start/stop bit framing of single character
  - Programmable character map of 32 hard-wired characters (00_H-$1F_H$)
  - Four programmable characters for additional mapping
  - Insertion/deletion of control-escape character ($7D_H$) for mapped characters
- Asynchronous (ASYNC) Protocol Mode
  - Selectable character length (5 to 8 bits)
  - Even, odd, forced or no parity generation/checking
  - 1 or 2 stop bits
  - Break detection/generation
  - In-band flow control by XON/XOFF
  - Immediate character insertion
  - Termination character detection for end of block identification
  - Time out detection
  - Error detection (parity error, framing error)
- BISYNC Protocol Mode
  - Programmable 6/8 bit SYN pattern (MONOSYNC)
  - Programmable 12/16 bit SYN pattern (BISYNC)
  - Selectable character length (5 to 8 bits)
  - Even, odd, forced or no parity generation/checking
  - Generation of interframe-time fill ’1’s or SYN characters
  - CRC generation (CRC-16 or CRC-CCITT)
  - Transparent CRC option per channel and/or per frame

  - Programmable Preamble (8 bit) with selectable repetition rate
  - Termination character detection for end of block identification
  - Error detection (parity error, framing error)
- Extended Transparent Mode
  - Fully bit transparent (no framing, no bit manipulation)
  - Octet-aligned transmission and reception
- Protocol and Mode Independent
  - Data bit inversion
  - Data overflow and underrun detection
  - Timer

Protocol Support

- Address Recognition Modes
  - No address recognition (Address Mode 0)
  - 8-bit (high byte) address recognition (Address Mode 1)
  - 8-bit (low byte) or 16-bit (high and low byte) address recognition (Non Auto Mode)
- HDLC Auto Mode
  - 8-bit or 16-bit address generation/recognition
  - Support of LAPB/LAPD
  - Automatic handling of S- and I-frames
  - Automatic processing of control byte(s)
  - Modulo-8 or modulo-128 operation
  - Programmable time-out and retry conditions
  - SDLC Normal Response Mode (NRM) operation for slave

Microprocessor Interface

- 33 MHz/32-bit PCI bus interface option.
- 33 MHz/32-bit De-multiplexed bus interface option.
- 8-channel DMA controller with buffer chaining capability.
  - Master 15-word burst read and write capability (PCI Mode).
  - Master 4-word burst read and write capability (DEMUX Mode).
  - Slave single-word read and write capability.
- Circular interrupt queues with variable size.
- Maskable interrupts for each channel

Other Interfaces

- 8-/16-bit optional Local Bus Interface (LBI) for driving non-PCI peripherals in a PCI environment.
- Synchronous Serial Control interface (SSC) for controlling peripherals.
- 16-bit General Purpose Port (GPP).

General

- On chip Rx and Tx data buffer; the buffer size is 128 32-bit words each.
- Programmable buffer size in transmit direction per channel; buffer allocation in receive direction on request.
- Programmable watermark for receive channels to control transfer of receive data to host memory.
- Two programmable watermarks for each transmit channel. One controlling data loading from host memory and one controlling transfer of transmit data to the corresponding Serial Communication Controller (SCC).
- Internal test loop capability.
- JTAG boundary scan test according to IEEE 1149.1
- Advanced low-power CMOS technology
- TTL-compatible inputs/outputs
  - 3.3 V & 5 V power supply
  - 3.3 V interfaces (TTL levels; 5 V tolerant in 5 V environment)
- P-FQFP-208-7 package
- The 10 MHz version only is available in extended temperature range -40 .. +85 °C (PEF 20534 H-10)

1.2 Differences between the DSCC4 and the ESCC Family

This chapter is useful for all being familiar with the Infineon Technologies' ESCC family.

1.2.1 Enhancements to the ESCC Serial Core

The DSCC4 SCC cores contain the core logic of the ESCC2 V3.2A as the heart of the device. Some enhancements are incorporated in the SCCs. These are:

- Asynchronous PPP protocol support as in Internet RFC-1662
- Octet and Bit Synchronous PPP protocol support as in Internet RFC-1662
- 16-Kbyte packet length byte counter
- Enhanced address filtering (16-bit maskable)
- Enhanced time slot assigner
- Support of high data rates (45 Mbit/s for DS3 or 52 Mbit/s for OC1). Protocol support limited to HDLC Sub-modes without address recognition.

1.2.2 Simplifications to the ESCC Serial Core

The following features of the ESCC core have been removed:

- SDLC Loop mode
- Extended transparent mode 0
  (this mode provided octet buffered data reception without usage of FIFOs; the DSCC4 supports octet buffered reception via appropriate threshold configurations for the SCC receive FIFOs)

1.3 Logic Symbol

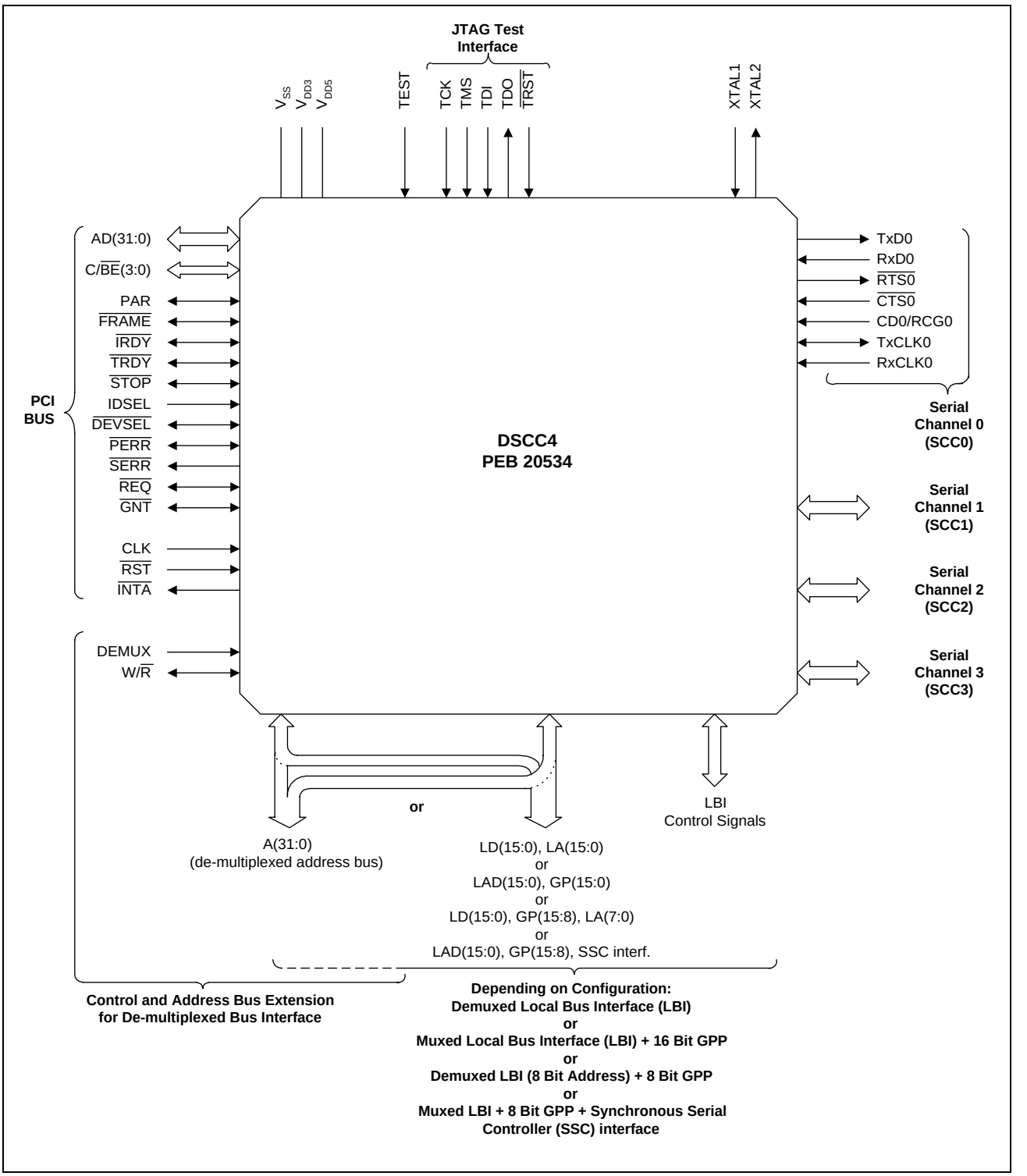


Figure 1 Logic Symbol

1.4 Typical Applications

The DSCC4 is designed to handle up to 4 serial data ports in various configurations, depending on the application. It transfers the data between the serial ports and a shared memory via its 32 bit/33 MHz PCI Bus Interface which can optionally be configured as a generic 32 bit de-multiplexed bus interface in the case that no PCI bus is applicable.

Figure 2 provides a general overview upon system integration of the DSCC4 in a PCI bus environment:

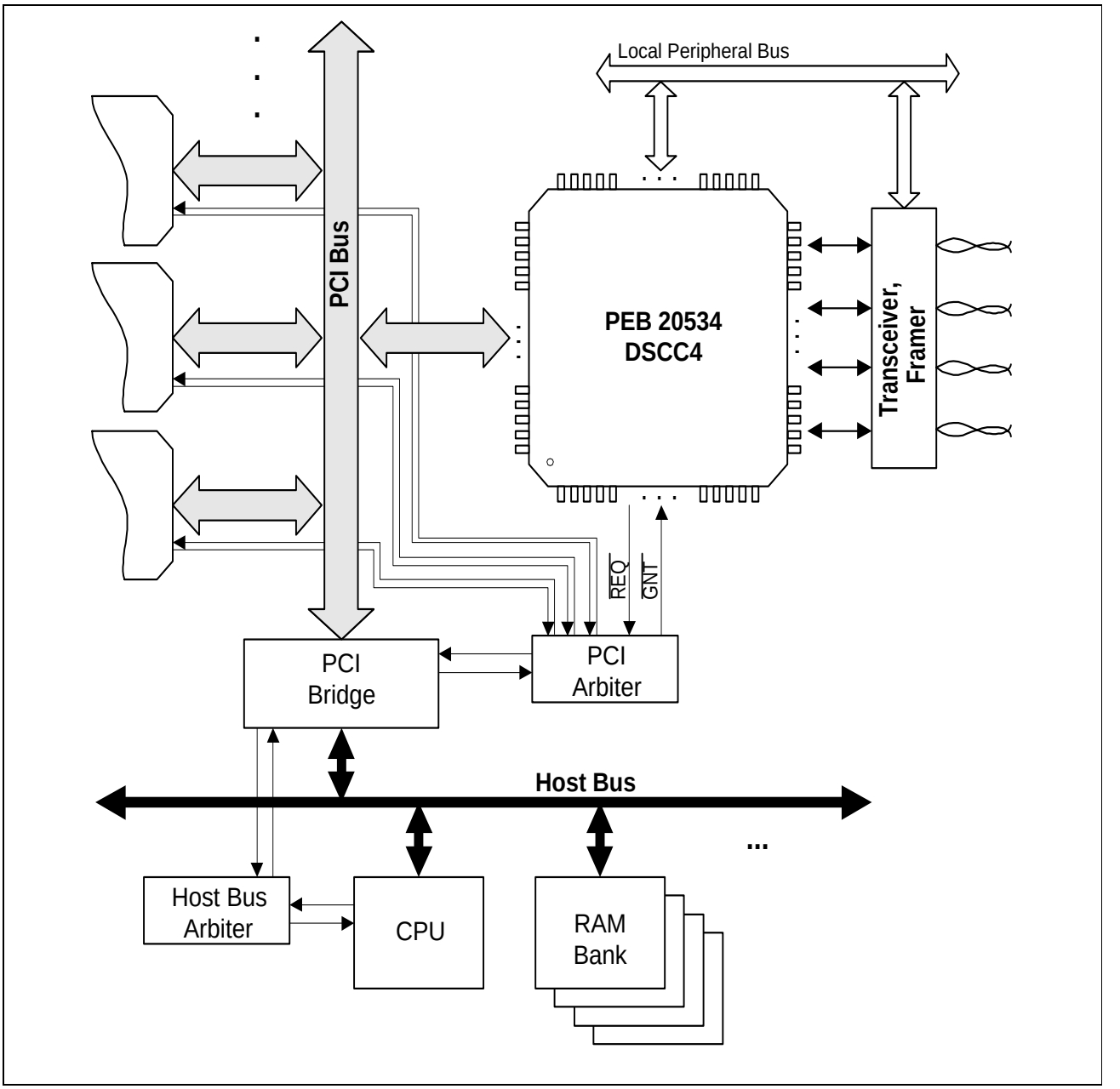


Figure 2 General System Integration (PCI Bus Interface)

Connection of DSCC4 to PCI Bus according to PCI Specification Rev. 2.1 is free of any glue-logic.

Figure 3 provides an overview upon system integration in a non PCI bus environment by the example of a Motorola 68360 CPU bus:

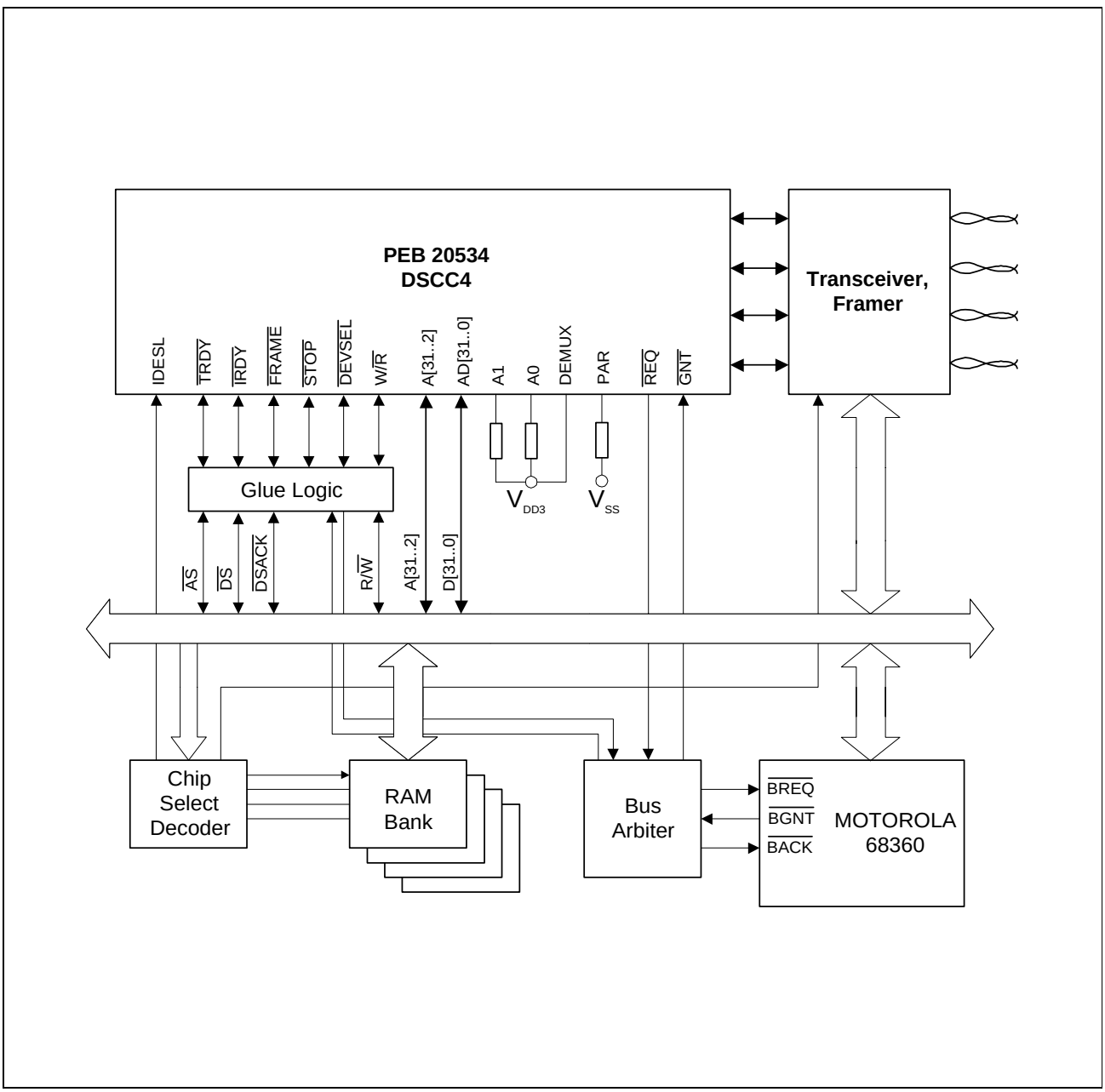


Figure 3 General System Integration (De-multiplexed Interface)

The glue-logic depends on the host bus which the DSCC4 should be connected to. The example in **Figure 3** shows the glue-logic for connection to an Motorola 68360 like de-multiplexed 32 bit bus.

1.4.1 Application Examples

1.4.1.1 HSSI Application - DCE Adapter

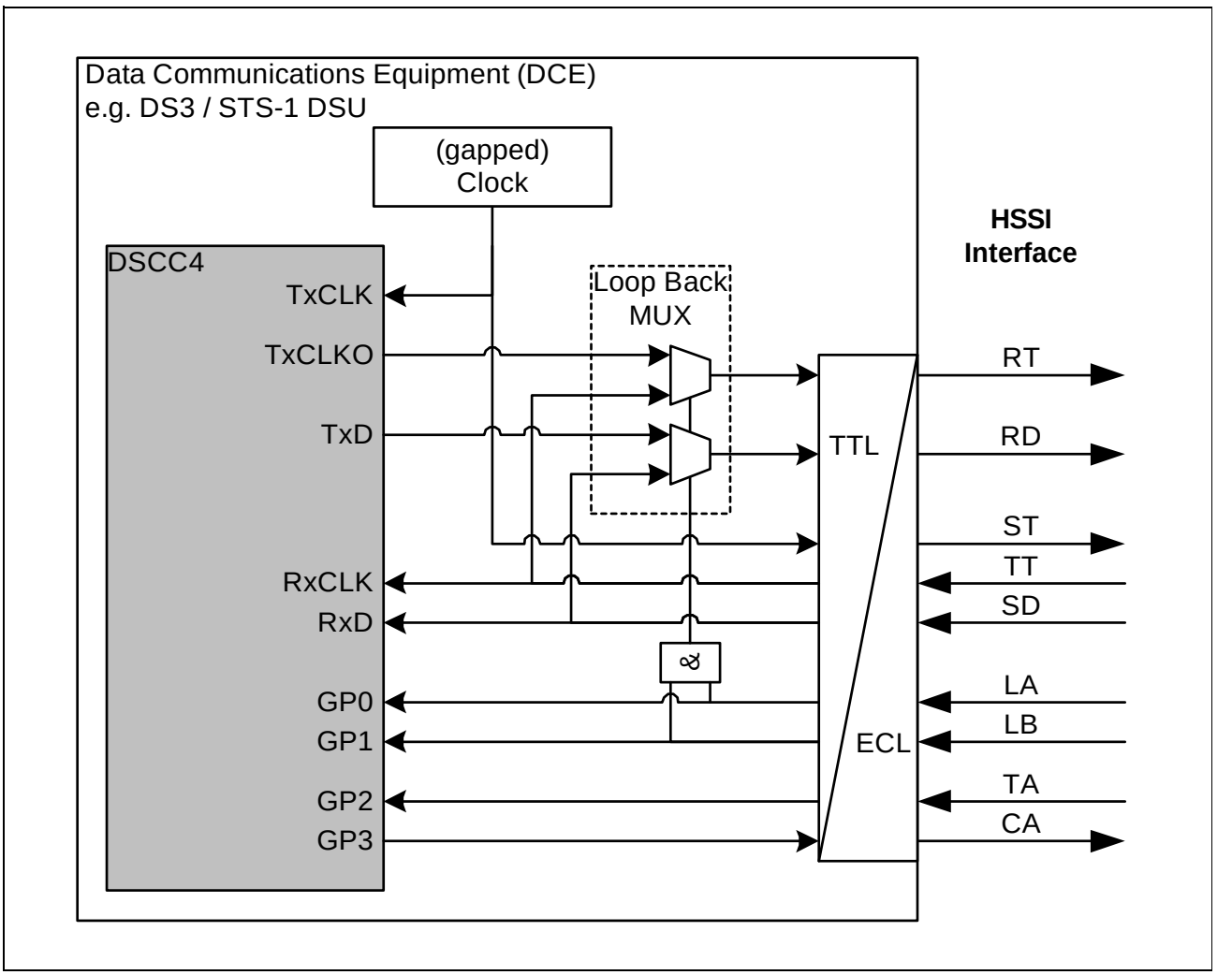


Figure 4 HSSI Application - DCE Adapter

1.4.1.2 HSSI Application - DTE Adapter

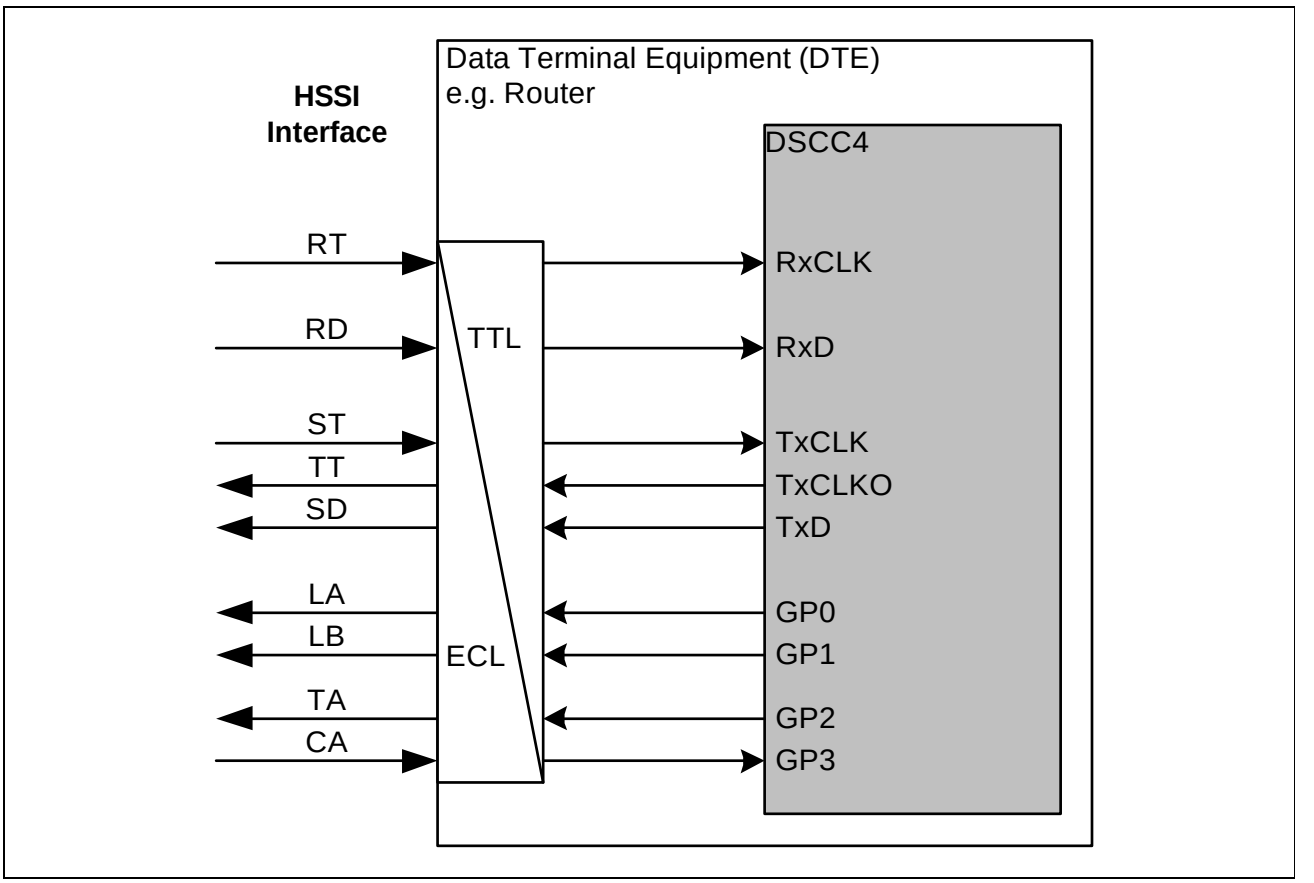


Figure 5 HSSI Application - DTE Adapter

1.4.1.3 General Data Application

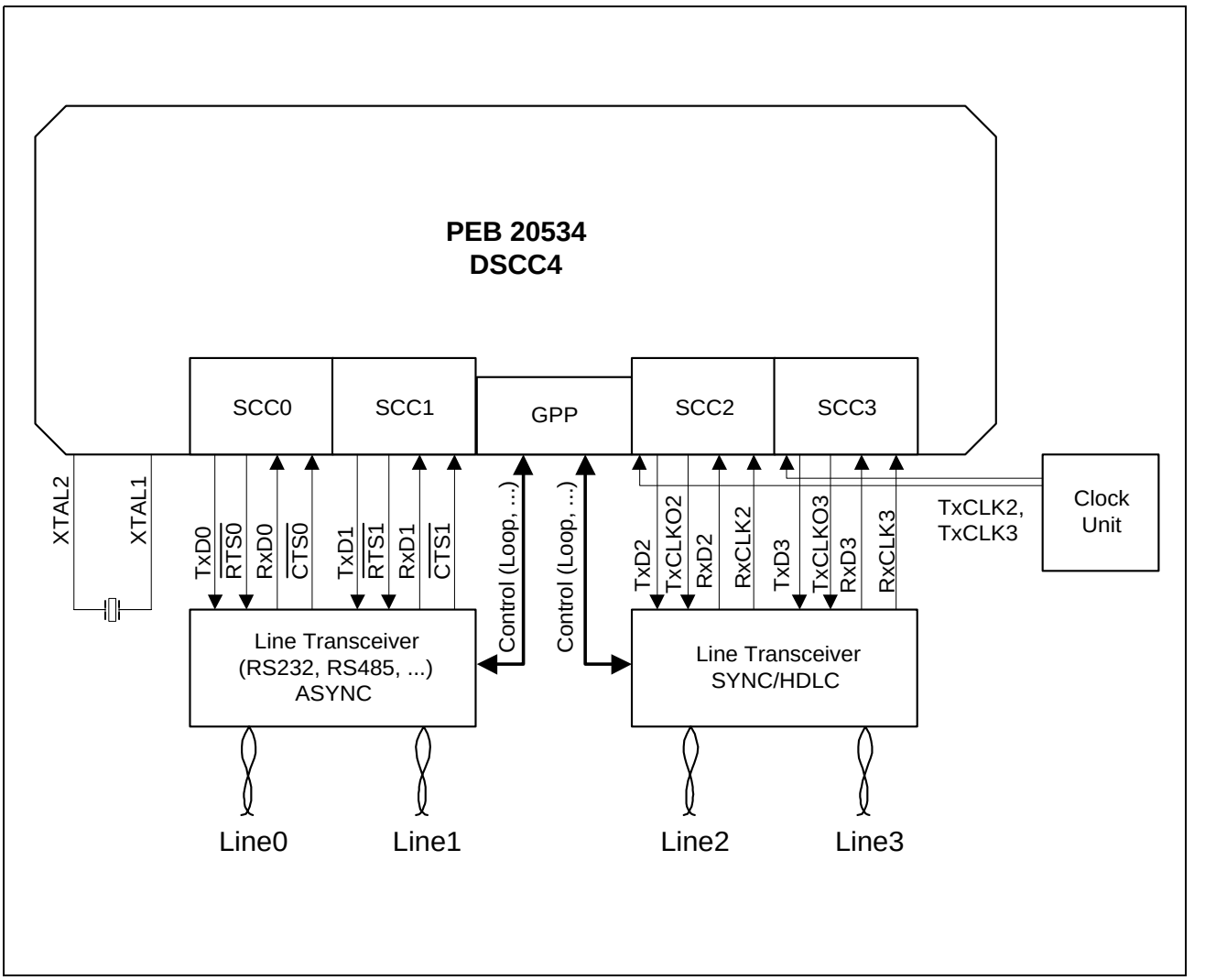


Figure 6 General Data Application

2 Pin Descriptions

2.1 Pin Diagram

(top view)

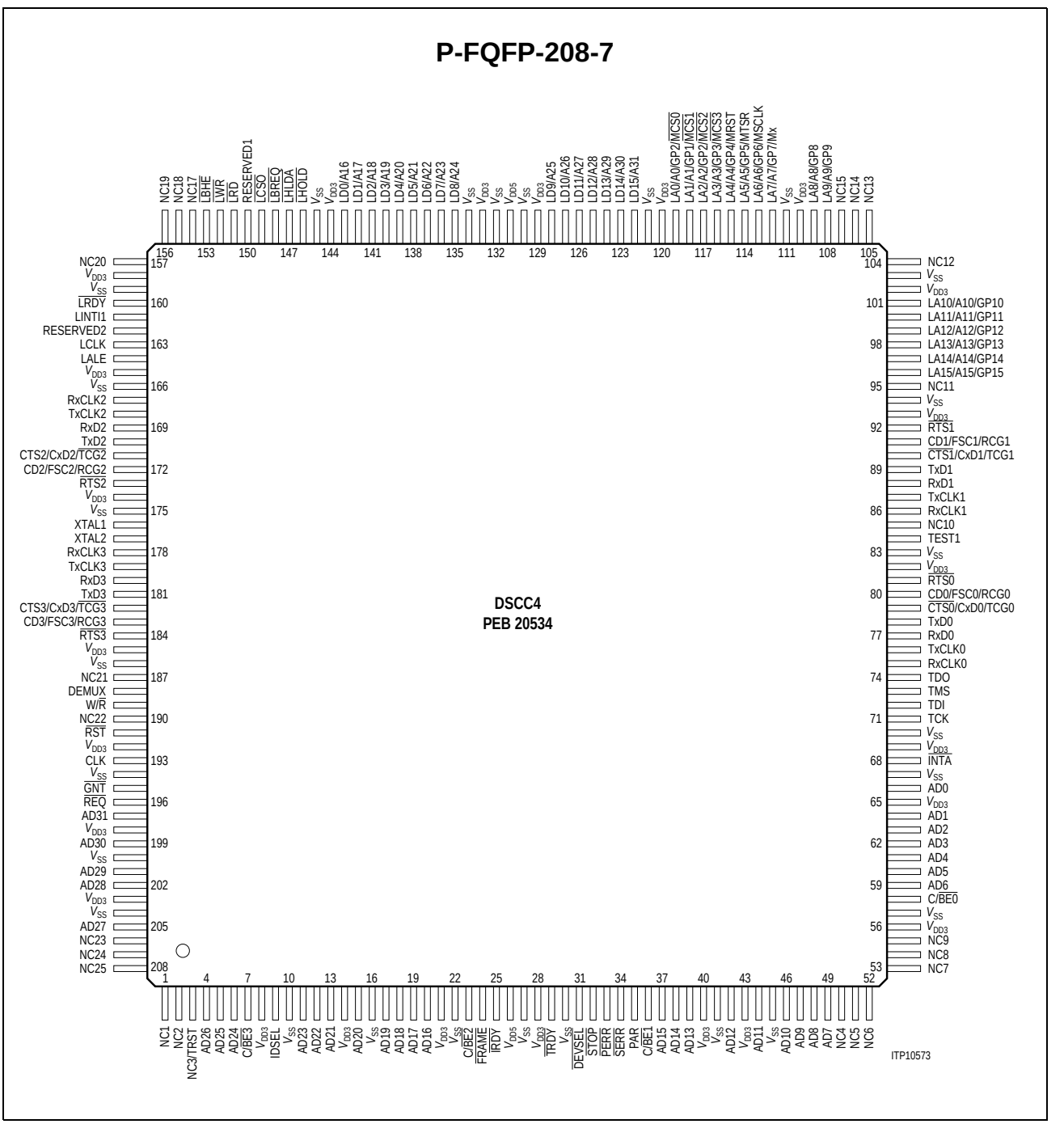


Figure 7 Pin Configuration

2.2 Pin Definitions and Functions

Signal Type Definitions:

The following signal type definitions are mainly taken from the PCI Specification Revision 2.1:

I	**Input** is a standard input-only signal.
O	**Totem Pole Output** is a standard active driver.
t/s, I/O	**Tri-State or I/O** is a bi-directional, tri-state input/output pin.
s/t/s	**Sustained Tri-State** is an active low tri-state signal owned and driven by one agent at a time. (For further information refer to the PCI Specification Revision 2.1)
o/d	**Open Drain** allows multiple devices to share as a wire-OR. A pull-up is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

Signal Name Conventions:

NC	**Not Connected Pin** These pins are not bonded with the silicon. Although any potential at these pins will not impact the device it is recommended to leave them unconnected. NC pins might be used for additional functionality in later versions of the device. Leaving them unconnected will guarentee hardware compatibility to later device versions.
Reserved	**Reserved** pins are for vendor specific use only and should be connected as recommended to guarantee normal operation.

Table 1 PCI Bus Interface(DEMUX Interface)

Pin No.	Symbol	Input (I) Output (O)	Function
197, 199, 201, 202, 205, 4...6, 11...13, 15, 17...20, 37...39, 42, 44, 46...49, 59...64, 66	AD(31:0)	t/s	**Address/Data Bus** A bus transaction consists of an address phase followed by one or more data phases. When DSCC4 is Master, AD(31:0) are outputs in the address phase of a transaction. During the data phases, AD(31:0) remain outputs for write transactions, and become inputs for read transactions. When DSCC4 is Slave, AD(31:0) are inputs in the address phase of a transaction. During the data phases, AD(31:0) remain inputs for write transactions, and become outputs for read transactions. AD(31:0) are updated and sampled on the rising edge of CLK.
7, 23, 36, 58	C/$\overline{\text{BE}}$(3:0)	t/s	**Command/Byte Enable** During the address phase of a transaction, C/$\overline{\text{BE}}$(3:0) define the bus command. During the data phase, C/$\overline{\text{BE}}$(3:0) are used as Byte Enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. C/$\overline{\text{BE0}}$ applies to byte 0 (lsb) and C/$\overline{\text{BE3}}$ applies to byte 3 (msb). When DSCC4 is Master, C/$\overline{\text{BE}}$(3:0) are outputs. When DSCC4 is Slave, C/$\overline{\text{BE}}$(3:0) are inputs. C/$\overline{\text{BE}}$(3:0) are updated and sampled on the rising edge of CLK. *Note: The bus command cycle is not generated (initiator) or evaluated (target) in DEMUX mode.*

Table 1 **PCI Bus Interface**(DEMUX Interface) (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
35	PAR	t/s	**Parity** PAR is even parity across AD(31:0) and C/$\overline{BE}$(3:0). PAR is stable and valid one clock after the address phase. PAR has the same timing as AD(31:0) but delayed by one clock. When DSCC4 is Master, PAR is output during address phase and write data phases. When DSCC4 is Slave, PAR is output during read data phases. Parity errors detected by the DSCC4 are indicated on $\overline{PERR}$ output. PAR is updated and sampled on the rising edge of CLK. *Note: PAR is not generated in DEMUX mode and remains '0'.A Pull-Down resistor to V_{SS} is recommended.*
24	$\overline{FRAME}$	s/t/s	**Frame** $\overline{FRAME}$ indicates the beginning and end of an access. $\overline{FRAME}$ is asserted to indicate a bus transaction is beginning. While $\overline{FRAME}$ is asserted, data transfers continue. When $\overline{FRAME}$ is deasserted, the transaction is in the final phase. When DSCC4 is Master, $\overline{FRAME}$ is an output. When DSCC4 is Slave, $\overline{FRAME}$ is an input. $\overline{FRAME}$ is updated and sampled on the rising edge of CLK.

Table 1 **PCI Bus Interface**(DEMUX Interface) (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
25	$\overline{\text{IRDY}}$	s/t/s	**Initiator Ready** $\overline{\text{IRDY}}$ indicates the bus master's ability to complete the current data phase of the transaction. It is used in conjunction with $\overline{\text{TRDY}}$. A data phase is completed on any clock where both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are sampled asserted. During a write, $\overline{\text{IRDY}}$ indicates that valid data is present on AD(31:0). During a read, it indicates the master is prepared to accept data. Wait cycles are inserted until both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted together. When DSCC4 is Master, $\overline{\text{IRDY}}$ is an output. When DSCC4 is Slave, $\overline{\text{IRDY}}$ is an input. $\overline{\text{IRDY}}$ is updated and sampled on the rising edge of CLK.
29	$\overline{\text{TRDY}}$	s/t/s	**Target Ready** $\overline{\text{TRDY}}$ indicates a slave's ability to complete the current data phase of the transaction. During a read, $\overline{\text{TRDY}}$ indicates that valid data is present on AD(31:0). During a write, it indicates the target is prepared to accept data. When DSCC4 is Master, $\overline{\text{TRDY}}$ is an input. When DSCC4 is Slave, $\overline{\text{TRDY}}$ is an output. $\overline{\text{TRDY}}$ is updated and sampled on the rising edge of CLK.
32	$\overline{\text{STOP}}$	s/t/s	**Stop Signal** $\overline{\text{STOP}}$ is used by a slave to request the current master to stop the current bus transaction. When DSCC4 is Master, $\overline{\text{STOP}}$ is an input. When DSCC4 is Slave, $\overline{\text{STOP}}$ is an output. $\overline{\text{STOP}}$ is updated and sampled on the rising edge of CLK.

Table 1 **PCI Bus Interface**(DEMUX Interface) (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
9	IDSEL	I	**Initialization Device Select** When DSCC4 is slave in a transaction and if IDSEL is active in the address phase and C/$\overline{BE}$(3:0) indicates an config read or write command, the DSCC4 assumes a read or write to a configuration space register. In response, the DSCC4 asserts $\overline{DEVSEL}$ during the subsequent CLK cycle. IDSEL is sampled on the rising edge of CLK. *Note: In DEMUX mode IDSEL is a chipselect for the configuration space registers.*
31	$\overline{DEVSEL}$	s/t/s	**Device Select** When activated by a slave, it indicates to the current bus master that the slave has decoded its address as the target of the current transaction. If no bus slave activates $\overline{DEVSEL}$ within six bus CLK cycles, the master should abort the transaction. When DSCC4 is master, $\overline{DEVSEL}$ is input. If $\overline{DEVSEL}$ is not activated within six clock cycles after an address is output on AD(31:0), the DSCC4 aborts the transaction and generates an $\overline{INTA}$. When DSCC4 is slave, $\overline{DEVSEL}$ is output. *Note: $\overline{DEVSEL}$ is also valid in DEMUX mode.*
33	$\overline{PERR}$	s/t/s	**Parity Error** When activated, indicates a parity error over the AD(31:0) and C/$\overline{BE}$(3:0) signals (compared to the PAR input). It has a delay of two CLK cycles with respect to AD and C/$\overline{BE}$(3:0) (i.e., it is valid for the cycle immediately following the corresponding PAR cycle). $\overline{PERR}$ is asserted relative to the rising edge of CLK.

Table 1 **PCI Bus Interface**(DEMUX Interface) (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
34	$\overline{\text{SERR}}$	o/d	**System Error** The DSCC4 asserts this signal to indicate a fatal system error. $\overline{\text{SERR}}$ is activated on the rising edge of CLK.
196	$\overline{\text{REQ}}$	t/s	**Request** Used by the DSCC4 to request control of the PCI. $\overline{\text{REQ}}$ is activated on the rising edge of CLK.
195	$\overline{\text{GNT}}$	t/s	**Grant** This signal is asserted by the arbiter to grant control of the PCI to the DSCC4 in response to a bus request via $\overline{\text{REQ}}$. After $\overline{\text{GNT}}$ is asserted, the DSCC4 will begin a bus transaction only after the current bus Master has deasserted the $\overline{\text{FRAME}}$ signal. $\overline{\text{GNT}}$ is sampled on the rising edge of CLK.
193	CLK	I	**Clock** Provides timing for all PCI transactions. Most PCI signals are sampled or output driven relative to the rising edge of CLK. The maximum CLK frequency is 33 MHz.
191	$\overline{\text{RST}}$	I	**Reset** An active $\overline{\text{RST}}$ signal brings all PCI registers, sequencers and signals into a consistent state. $\overline{\text{RST}}$ also resets all other blocks beside PCI to their initial state. During RESET - all PCI output signals are driven to their benign state - the TxDn (n=0,..,3) output signals are in high impedance state - the $\overline{\text{RTSn}}$ (n=0,..,3) output signals are inactive - all bi-directional signals are inputs.

Table 1 PCI Bus Interface(DEMUX Interface) (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
68	$\overline{\text{INTA}}$	o/d	**Interrupt Request** When an interrupt status is active and unmasked, the DSCC4 activates this open-drain output. Examples of interrupt sources are transmission/reception error, completion of transmit or receive packets etc. The DSCC4 deactivates $\overline{\text{INTA}}$ when the interrupt status is acknowledged via an appropriate action (e.g., specific register write) and no other unmasked interrupt statuses are active. $\overline{\text{INTA}}$ is activated/ deactivated asynchronous to the CLK.

Note: PCI control signals (type s/t/s) always require pull-up resistors. For the system dependent pull-up recommendation please refer to PCI Specification Revision 2.1 chapter 4.3.3

Note: The function of PCI Bus Interface signals is the same in DEMUX mode where the DSCC4 is operated in a non-PCI bus environment. All recommendations and signal characteristics also apply in DEMUX mode.
Signal PAR is not generated and remains '0' in DEMUX mode.
In DEMUX mode signal IDSEL is a chipselect for the PCI Configuration Space (no bus commands exists in DEMUX mode to address Configuration Space access in addition to IDSEL)

Additional DEMUX Interface Control Signals

188	DEMUX	I	**PCI/De-multiplexed Mode Select** DEMUX = '0' selects normal PCI operation. DEMUX = '1' selects operation in de-multiplexed (DEMUX) mode. (Pull-Up/Down resistors or direct connection to V_{SS}/V_{DD3} possible)
189	W/$\overline{\text{R}}$	I/O	**Write/Read Control** This signal distinguishes between write and read operations in the De-multiplexed mode. It is tristate when the DSCC4 is in PCI mode. **A Pull-Up resistor to V_{DD3} is recommended for PCI operation mode (DEMUX = V_{SS}).**

Table 2 Dedicated Signals

Pin No.	Symbol	Input (I) Output (O)	Function
1, 2, 50, 51, 52, 53, 54, 55, 85, 95, 104, 105, 106, 107, 154, 155, 156, 157, 187, 190, 206, 207, 208	NC1, NC2, NC4...15 NC17...25	-	**No-connect Pin 1, 2, 4…15** **No-connect Pin 17...25** **These pins must be left unconnected.**
194, 200, 204, 10, 16, 22,27, 30, 41, 45, 57, 67, 70, 83, 94, 103, 111, 121, 130, 132, 134, 145, 159, 166, 175, 186	V_{SS}		**Ground** (0 V) All pins must be connected to the same voltage potential.
192, 198, 203, 8, 14, 21, 28, 40, 43, 56, 65, 69, 82, 93, 102, 110, 120, 129, 133, 144, 158, 165, 174, 185	V_{DD3}		**Supply Voltage** 3.3 V ± 0.3 V All pins must be connected to the same voltage potential.
26, 131	V_{DD5}		**Supply Voltage** 5 V ± 0.25 V Both pins must be connected to the same voltage potential.

Table 2 Dedicated Signals (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
84	TEST	I	**Test Input** When connected to V_{DD3} the DSCC4 works in a vendor specific test mode. *It is recommended to connect this pin to V_{SS}.*
150	Reserved 1	-	**Reserved Pin 1** *A Pull-Up resistor to V_{DD3} is required.*
162	Reserved 2	-	**Reserved Pin 2** *A Pull-Up resistor to V_{DD3} is required.*

Table 3 JTAG Test Port for Boundary Scan according to IEEE 1149.1

Pin No.	Symbol	Input (I) Output (O)	Function
71	TCK	I	**JTAG Test Clock** Connection V_{DD3} is recommended if boundaryscan unit is not used.
73	TMS	I	**JTAG Test Mode Select** *Note: An internal pull-up transistor is provided. Pin TMS can be left unconnected if boundary scan operation is not used.*
72	TDI	I	**JTAG Test Data Input** *Note: An internal pull-up transistor is provided. Pin TDI can be left unconnected if boundary scan operation is not used.*
74	TDO	O	**JTAG Test Data Output**
3	$\overline{\text{TRST}}$	I	**JTAG Reset Pin** For normal device operation this pin should be connected to V_{SS} or logical '0' to force deactivation of the boundary scan unit. For boundary scan mode $\overline{\text{TRST}}$ should be connected to high level. *Note: An internal pull-up transistor forces boundary scan operation mode if this pin remains unconnected.*

Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins

Pin No.	Symbol	Input (I) Output (O)	Function
96...101, 108, 109, 112...119	LA(15:0)	I/O	**PCI Mode (DEMUX connected to VSS):** **LBI Address Bus** (GMODE.PERCFG=000_2) These pins provide the 16 bit Address bus for the Local Bus Interface.
	or		
	GP(15:0)	I/O	**General Purpose Port** (GMODE.PERCFG=100_2) A general purpose 16-bit bi-directional parallel port is provided on pins GP(15...0). Every pin is individually programmable via register GPDIR to operate as an output or an input. If defined as output, the state of the pin is directly controlled via the register GPDATA. If defined as input, its current status can be read via GPDATA. All changes may be indicated via an interrupt vector. The interrupts for each single pin can be masked via mask register GPIM.

Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins

Pin No.	Symbol	Input (I) Output (O)	Function
	or		
96...101, 108, 109,	GP (15:8)	I/O	**8 bit General Purpose Port** (GMODE.PERCFG=011_2)
112	MX	I/O	**SSC Interface** (GMODE.PERCFG=011_2): **Dont' Care Pin** This pin is reserved in SSC mode. **A Pull-Up resistor to V_{DD3} is recommended.**
113	MSCLK		**SSC Shift Clock Input/Output**
114	MTSR		**SSC Master Transmit / Slave Receive**
115	MRST		**SSC Master Receive / Slave Transmit**
116	$\overline{\text{MCS3}}$		**SSC Chip select 3**
117	$\overline{\text{MCS2}}$		**SSC Chip select 2**
118	$\overline{\text{MCS1}}$		**SSC Chip select 1**
119	$\overline{\text{MCS0}}$		**SSC Chip select 0**
	or		
96...101, 108, 109, 112...119	GP (15:8)	I/O	**8 bit General Purpose Port** (GMODE.PERCFG=010_2)
	LA(7:0)		**LBI Address Bus** These pins provide the 8 bit Address bus for the Local Bus Interface.
	or		
96...101, 108, 109, 112...119	A(15:0)	I/O	**De-multiplexed Mode (DEMUX connected to VDD3) :** **DEMUX Address Bus (15:0)** These pins provide the 16 least significant address lines for the de-multiplexed interface. **Signals A(1:0) are unused due to 32 bit address alignment. A Pull-Up resistor to V_{DD3} is recommended.**

Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins

Pin No.	Symbol	Input (I) Output (O)	Function
122...128, 135...143	LD (15:0)	I/O	**PCI Mode (DEMUX connected to VSS):** **LBI Data** (GMODE.PERCFG=0XX$_2$) These pins provide the 16 bit Data bus for the Local Bus Interface. **LBI Address/Data** (GMODE.PERCFG=100$_2$) These pins provide the 16 bit multiplexed Address/Data bus for the Local Bus Interface.
	or		
	A(31:16)	I/O	**Demultiplexed Mode (DEMUX connected to VDD3) :** **DEMUX Address Bus (31:16)** These pins provide the 16 most significant address lines for the de-multiplexed interface.
146	$\overline{\text{LHOLD}}$	I	**LBI Hold Request** $\overline{\text{LHOLD}}$ = '1' is used for normal bus drive mode. $\overline{\text{LHOLD}}$ = '0' requests LBI to enter hold mode. **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
148	$\overline{\text{LBREQ}}$	O	**LBI Bus Request** The DSCC4 asserts $\overline{\text{LBREQ}}$ = '0' to request the local bus and deasserts the signal $\overline{\text{LBREQ}}$ = '1' after regaining bus.

Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins

Pin No.	Symbol	Input (I) Output (O)	Function
147	$\overline{\text{LHLDA}}$	I/O	**LBI Hold Status** The function depends on whether DSCC4 is enabled as arbitration master via bit LCONF.HDEN: As an output, $\overline{\text{LHLDA}}$ = '0' confirms that the LBI bus is in $\overline{\text{HOLD}}$ mode (arbitration master). As an input, $\overline{\text{LHLDA}}$ = '1' means that DSCC4 must remain in hold mode (external arbiter). *A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.*
149	$\overline{\text{LCSO}}$	O	**LBI Chip Select Output** Used to select LBI external peripheral
164	LALE	(I)/O	**LBI Address Latch Enable** *This pin is tri-state when unused. A Pull-Down resistor to V_{SS} is recommended if LBI is not used or operated in demultiplexed LBI configuration.*
151	$\overline{\text{LRD}}$	(I)/O	**LBI Read strobe** *This pin is tri-state when the LBI is not the active bus master. Refer to **Page 420** for timing figures.*
152	$\overline{\text{LWR}}$	(I)/O	**LBI Write strobe** *This pin is tri-state when the LBI is not the active bus master. Refer to **Page 420** for timing figures.*
153	$\overline{\text{LBHE}}$	(I)/O	**LBI Byte high enable** *This pin is tri-state when the LBI is not the active bus master. Refer to **Page 420** for timing figures.*
160	$\overline{\text{LRDY}}$	I	**LBI Ready strobe** Control signal for extended bus cycles. This signal is asserted by the bus target to insert wait states.

Table 4 Local Bus Interface (LBI) / General Purpose Port (GPP) / Synchronous Serial Control (SSC) Interface Pins

Pin No.	Symbol	Input (I) Output (O)	Function
161	LINTI1	I	**LBI Interrupt Input from Peripheral 1**
163	LCLK	O	**LBI Clock Output** This is signal provides the internal LBI clock which is frequency of signal CLK divided by n (n must be configured with bit field GMODE.LCD(1:0)). LCLK is provided for connection of synchronous peripherals. *Note: The duty cycle (high to low in %) of LCLK depends on the clock division factor:* *n = 2: 25%* *n = 4: 12.5%* *n = 16: 3.125%* *The high phase is always T_{CLK}/2, i.e. typically 15ns at 33 MHz system clock frequency.*

Table 5 Serial Communication Controller (SCC) Signals

Pin No.	Symbol	Input (I) Output (O)	Function
75 86 167 178	RxCLK0 RxCLK1 RxCLK2 RxCLK3	I	**Receive Clock** The function of these pins depends on the selected clock mode. In each channel, RxCLKn may supply either – the receive clock (clock mode 0, 4), or – the receive and transmit clock (clock mode 1, 5), or – the clock input for the baud rate generator (clock mode 2, 3).
77 88 169 180	RxD0 RxD1 RxD2 RxD3	I	**Receive Data** Serial data is received on these pins.

Table 5 Serial Communication Controller (SCC) Signals (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
76 87 168 179	TxCLK0 TxCLK1 TxCLK2 TxCLK3	I/O	**Transmit Clock** The function of this pin depends on the selected clock mode and the value of the SSEL bit (CCR0 register). If programmed as an input (bit CCR0.TOE='0'), this pin supplies either – the transmit clock for the channel (clock mode 0, 2, 4, 6; SSEL bit in CCR0 is reset), or – a transmit strobe signal for the channel (clock mode 1). If programmed as an output (bit CCR0.TOE='1'), this pin supplies the transmit clock for the channel which is generated either – from the baud rate generator (clock mode 2, 3, 6, 7; SSEL bit in CCR0 is set), or – from the DPLL circuit (clock mode 3, 7; SSEL bit in CCR0 is reset). In clock mode 5 an active-low tri-state control signal marks the programmed transmit time-slot if bit CCR2.TOE is set.
78 89 170 181	TxD0 TxD1 TxD2 TxD3	O, o/d	**Transmit Data** Transmit data is shifted out via these pins. They can be programmed to be either push-pull or open drain output to support bus configurations (register CCR1).

Table 5 Serial Communication Controller (SCC) Signals (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
81 92 173 184	$\overline{RTS}$0 $\overline{RTS}$1 $\overline{RTS}$2 $\overline{RTS}$3	O	**Request to Send** When the $\overline{RTS}$ bit in the CCR1 register is set, the RTS signal goes low. When the $\overline{RTS}$ bit is reset, the signal goes high if the transmitter has finished and there is no further request for a transmission. In bus configuration, $\overline{RTS}$ can be programmed via CCR1 to: – go low during the actual transmission of a frame shifted by one clock period, excluding collision bits. – go low during reception of a data frame. – stay always high ($\overline{RTS}$ disabled). In ASYNC mode, $\overline{RTS}$ can be programmed via CCR1 to: – to be controlled autonomously by the SCCn and be activated when a frame transmission starts and deactivated when transmission is completed (default state). – to be controlled autonomously by the SCCn for bi-directional flow control and to be forced active when shadow part of RFIFO is empty and forced in-active when RFIFO has reached a threshold. – to be controlled by the host.
	or TxCLKO0 TxCLKO1 TxCLKO2 TxCLKO3		**Transmit Clock Out** In clock mode 4 the internal transmit clock is switched to this pin if bit CCR1.TCLKO set. TxCLKOi will be in phase with the corresponding transmit data signal TxDi with regard to the High Speed timing Characteristics. *Note: It is recommended to ignore signal $\overline{RTS}$ in High Speed mode if CCR1.TCLKO is not set. Nevertheless the signal remains an active push/pull output.*

Table 5 Serial Communication Controller (SCC) Signals (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
79 90 171 182	$\overline{\text{CTS0}}$ $\overline{\text{CTS1}}$ $\overline{\text{CTS2}}$ $\overline{\text{CTS3}}$	I	**Clear to Send** A low on the $\overline{\text{CTSn}}$ input enables the respective transmitter. Additionally, an interrupt may be issued if a state transition occurs at the $\overline{\text{CTSn}}$ pin (programmable feature). If no 'Clear To Send' function is required, the $\overline{\text{CTSn}}$ inputs can be directly connected to V_{SS}.
	or		**Collision Data** In a bus configuration, the external serial bus must be connected to the corresponding CxD pin for collision detection. A collision is detected whenever a logical '1' is driven on the open drain TxD output but a logical '0' is detected via CxD input.
	CxD0 CxD1 CxD2 CxD3		
	or		**Transmit Clock Gating** In clock mode 4 these pins are used as Transmit Clock Gating signals.
	$\overline{\text{TCG0}}$ $\overline{\text{TCG1}}$ $\overline{\text{TCG2}}$ $\overline{\text{TCG3}}$		

Table 5 Serial Communication Controller (SCC) Signals (cont'd)

Pin No.	Symbol	Input (I) Output (O)	Function
80 91 172 183	CD0 CD1 CD2 CD3 **or** FSC0 FSC1 FSC2 FSC3 **or** $\overline{\text{RCG0}}$ $\overline{\text{RCG1}}$ $\overline{\text{RCG2}}$ $\overline{\text{RCG3}}$	I	**Carrier Detect** The function of this pin depends on the selected clock mode. It can supply – either a modem control or a general purpose input (clock modes 0, 2, 3, 6, 7). If auto-start is programmed, it functions as a receiver enable signal. – or a receive strobe signal (clock mode 1). Additionally, an interrupt may be issued if a state transition occurs at the CDn pin (programmable feature). **A Pull-Down resistor to V_{SS} is recommended if CD is not used.** **Frame Synchronization** When SCCn is in time-slot mode (e.g. PCM mode) these pins supply the Frame Synchronization inputs (clock mode 5). **Receive Clock Gating** In clock mode 4 these pins are used as Receive Clock Gating signals.
176 177	XTAL1 XTAL2	I O	**Crystal Connection** If the internal oscillator is used for clock generation (clock mode 0b, 6, 7) the external crystal has to be connected to these pins. Moreover, XTAL1 may be used as common clock input for all SCCs provided by an external clock generator (oscillator). **A Pull-Down resistor to V_{SS} is recommended if XTAL1 is not used.**

Note: In general all input and I/O signals should be forced via a pull-up or pull-down resistor to a defined level V_{DD3} or V_{SS} as stated in the pin description table. Pull-up/down resistor values should be less than or equal to 10 kOhm.

3 Functional Description

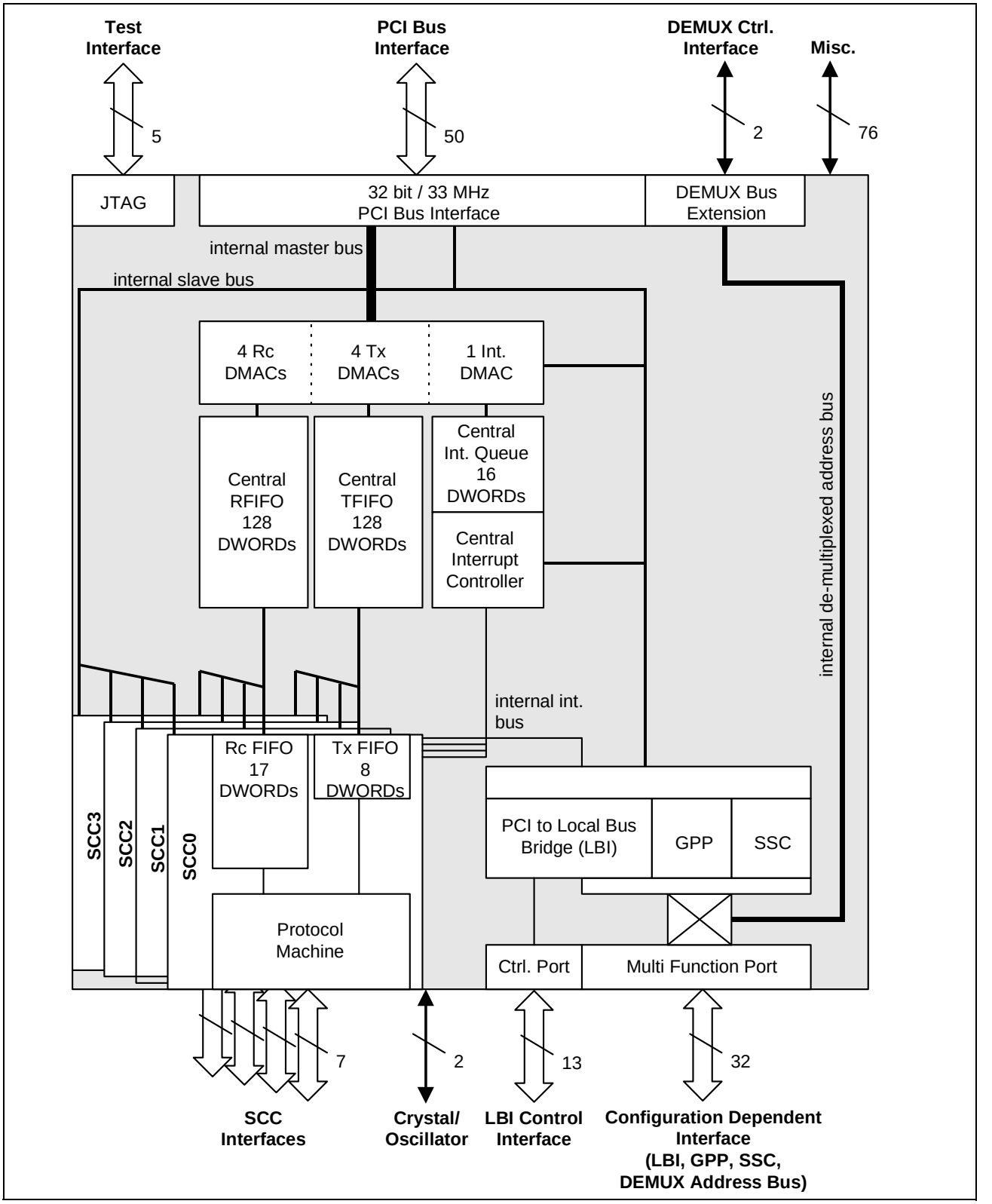


Figure 8 DSCC4 Functional Block Diagram

Abbreviations

Acronyms used in the block diagram are explained in the following:

DMAC - DMA Controller

GPP - General Purpose Port

JTAG - Joint Test Action Group

LBI - Local Bus Interface

MFP - Multi Function Port

PCI - Peripheral Components Interface

Rc- Receive

RFIFO - (Central) Receive FIFO

SCC3...0 - Serial Communication Controller 3...0

SSC - Synchronous Serial Controller

TFIFO - (Central) Transmit FIFO

Tx - Transmit

The functional blocks of the DSCC4 can be partitioned into three different groups:

– The first group provides the data transfer between the shared memory and on chip registers/FIFOs. It consists of the PCI System Bus Interface, the 9 DMACs, the Central RFIFO, the Central TFIFO and the Central Interrupt Queue.
– The second group supports the multiport and multiprotocol serial data communication. It consists of four SCCs.
– The third group provides control functions for external peripherals using the LBI and/ or low speed communication/control functions using SSC and/or GP. The third group includes the peripheral blocks: GPP, SSC, LBI. These peripheral blocks can be used in PCI mode configuration only.

Note: In demultiplexed bus operation mode, the MFP only supplies the De-multiplexed address bus extension of the host interface.

General Data Flow Description

For register read/write transaction, each single block can be accessed by the host CPU via the PCI (or de-multiplexed) bus interface.

The DSCC4 central TFIFO and RFIFO provide space for 128 DWORDs each to be shared by four serial channels (4 SCCs).

Four DMA Controller Channels (DMACs) transfer data associated with the serial channels from shared memory via the PCI interface into the central TFIFO. These data are forwarded from the TFIFO to the four SCCs.

In receive direction the serial channels (4 SCCs) deliver received data into the central RFIFO. Another four DMACs transfer these data from the central RFIFO via the PCI interface into shared memory locations, associated with the serial ports.

The SCCs as well as the peripheral blocks (LBI, GPP, SSC) are able to generate interrupts. Corresponding interrupt vectors are delivered by any single block the central interrupt controller and its 16 DWORDs interrupt queue. A DMAC transfers the interrupt vectors from the interrupt queue into appropriate locations in the shared memory.

Data and interrupt buffering exists in each block as its interface is contending for the internal busses. The access to the internal busses is controlled by arbiters that are not shown in the block diagram.

4 Microprocessor Bus Interface

The DSCC4 may be configured either for 33 MHz/32-bit PCI operation or for a 33 MHz/ 32-bit De-multiplexed bus interface.

The DSCC4's DEMUX input pin is used to select the desired configuration:

DEMUX connected to V_{SS}: PCI operation mode
connected to V_{DD3}: De-multiplexed bus interface mode

The De-multiplexed bus interface mode provides an additional address bus and W/$\overline{R}$ control signal extension to the PCI bus interface.

4.1 PCI Bus Interface

In this configuration, the DSCC4 interfaces directly to a 33 MHz/32-bit PCI bus. During run-time, the DSCC4 operates mostly as a PCI Master. However it may also be accessed by the host processor as a PCI Slave for register read/write or access to peripherals connected to the LBI.
Device configuration is performed via slave transactions to DSCC4 on-chip registers or to the PCI Configuration Space.

The DSCC4 is compliant with PCI specification 2.1 at up to 33 MHz.

In addition, the DSCC4 supports little/big endian byte swapping from/to the serial channels, and unaligned-byte accesses for transmit data sections.

4.1.1 Supported PCI Transactions

Memory accesses as a PCI Master: The DSCC4 supports both the PCI Memory Write and PCI Memory Read commands. For the PCI Memory Write command, it writes to an agent mapped in the memory access space, while for the PCI Read command, it reads from an agent mapped in the memory address space.

I/O accesses as a PCI Master: The DSCC4 does not support the PCI I/O Write nor PCI I/O Read commands.

M**emory accesses as a PCI Slave**: The DSCC4 supports both the PCI Memory Write and PCI Memory Read commands. For the PCI Memory Write command, the DSCC4 is written to as an agent mapped in the memory address space, while for the PCI Memory Read command, the DSCC4 is read from as an agent mapped in the memory address space.

I/O accesses as a PCI Slave: The DSCC4 does not support the PCI I/O Write nor PCI I/O Read commands.

Burst Capability: The DSCC4 supports bursts of up to 3 DWORDs for reading transmit and receive descriptors and bursts of up to 15 DWORDs for reading/writing data.

4.1.2 PCI Configuration Space Register Overview

The PCI Configuration Space Registers of the DSCC4 are listed in **Table 6**.

For a detailed description of the registers see **"PCI Configuration Space - Detailed Register Description" on Page 222** and **"Configuration Space"** described in chapter 6 of the **PCI Specification Rev. 2.1**.

Table 6 PCI Configuration Space Registers

Register Name	Short Name	Access (Read/ Write)	Absolute Address	Reset Value
Device ID / Vendor ID	DID / VID	R	00_H	$2102/110A_H$
Status / Command	STA / CMD	R/W	04_H	$0000/0000_H$
Class Code / Revision ID	CC / RID	R	08_H	$029000/21_H$
Builtin Self Test / Header Type / Latency Timer / Cache Line Size	BIST/HEAD/ LATIM/ CLSIZ	R/W	$0C_H$	00000000_H
Base Address 1	BAR1	R/W	10_H	00000000_H
Base Address 2	BAR2	R/W	14_H	00000000_H
Base Address not used	BARX	R/W	18_H-24_H	00000000_H
Cardbus CIS Pointer	CISP	R	28_H	00000000_H
Subsystem ID / Subsystem Vendor ID	SSID / SSVID	R	$2C_H$	00000000_H
Expansion ROM Base Address	ERBAD	R/W	30_H	00000000_H
Reserved	RES34	R/W	34_H	00000000_H
Reserved	RES38	R/W	38_H	00000000_H
Maximum Latency / Minimum Grant / Interrupt Pin / Interrupt Line	MAXLAT / MINGNT / INTPIN / INTLIN	R/W	$3C_H$	00000000_H
	REG40		40_H	00000000_H
	REG44		44_H	00000000_H
	REG48		48_H	00000000_H
	REG4C		$4C_H$	00000000_H

4.2 De-multiplexed Bus Interface Extension

The DSCC4 may be configured for 33 MHz/32-bit De-multiplexed bus for connection to non PCI systems with de-multiplexed processors such as the i960Hx or MC68EC0x0. The de-multiplexed bus interface is a synchronous interface very similar to the PCI bus with the following exceptions:

1. The W/$\overline{R}$ input/output signal replaces the function of the PCI command nibble in the C/$\overline{BE}$(3:0) bit field.
2. The transaction address is driven or read from the additional address bus A[31..2]
3. The parity signal PAR is not generated as a master or evaluated as a slave

Beside these exceptions all control signals and timings are equal to PCI bus interface mode. Also the PCI Configuration Space must be programmed during configuration.

Note: In DEMUX mode as in PCI mode, the DSCC4 provides only the first address of a master burst read or write transaction. Address incrementation must be provided externally if required by the target or DSCC4 burst capability is disabled (default value) for DEMUX mode.

Note: Because the PCI command nibble is replaced by a W/$\overline{R}$ control signal only IDSEL distinguishes between PCI Configuration Space and slave register access. Thus IDSEL must be treated as a Configuration Space chipselect and remain deasserted during all other slave register accesses.

Table 7 Non-PCI Signal Extension in the De-multiplexed Bus Interface Mode

Pin description table reference	Symbol	Input (I) Output (O)	Function
Table 1	DEMUX	I	DEMUX = V_{SS} selects PCI mode, DEMUX = V_{DD3} selects De-multiplexed Bus Interface mode.
Table 4	A(31:2)	I/O	**De-multiplexed Address Bus**
Table 1	W/$\overline{R}$	I/O	**Write/Read Control signal**

In DEMUX bus mode the burst capability is limited to 4-dwords and must be enabled via the ’DBE’ (Demux Burst Enable) bit in the Global Mode Register GMODE.

Even in the case that burst capability has been enabled, the target can request the DSCC4 to stop the current transaction by asserting the $\overline{STOP}$ signal as in PCI operation.

The following diagrams illustrate the functional timing waveforms for both single and burst transactions.

Table 8 DEMUX Mode Related Register and Bit-Fields

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0008_H	r/w	CPU	00000000_H	**GMODE**: Global Mode Register
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	1	DBE	0	Demux Bus Mode Burst Enable DBE = '0' Burst capability disabled, only single DWORD transfers are performed; DBE = '1' Burst capability enabled, burst length is limited to 4 DWORDs. *Note: Only valid in De-multiplexed bus mode.*

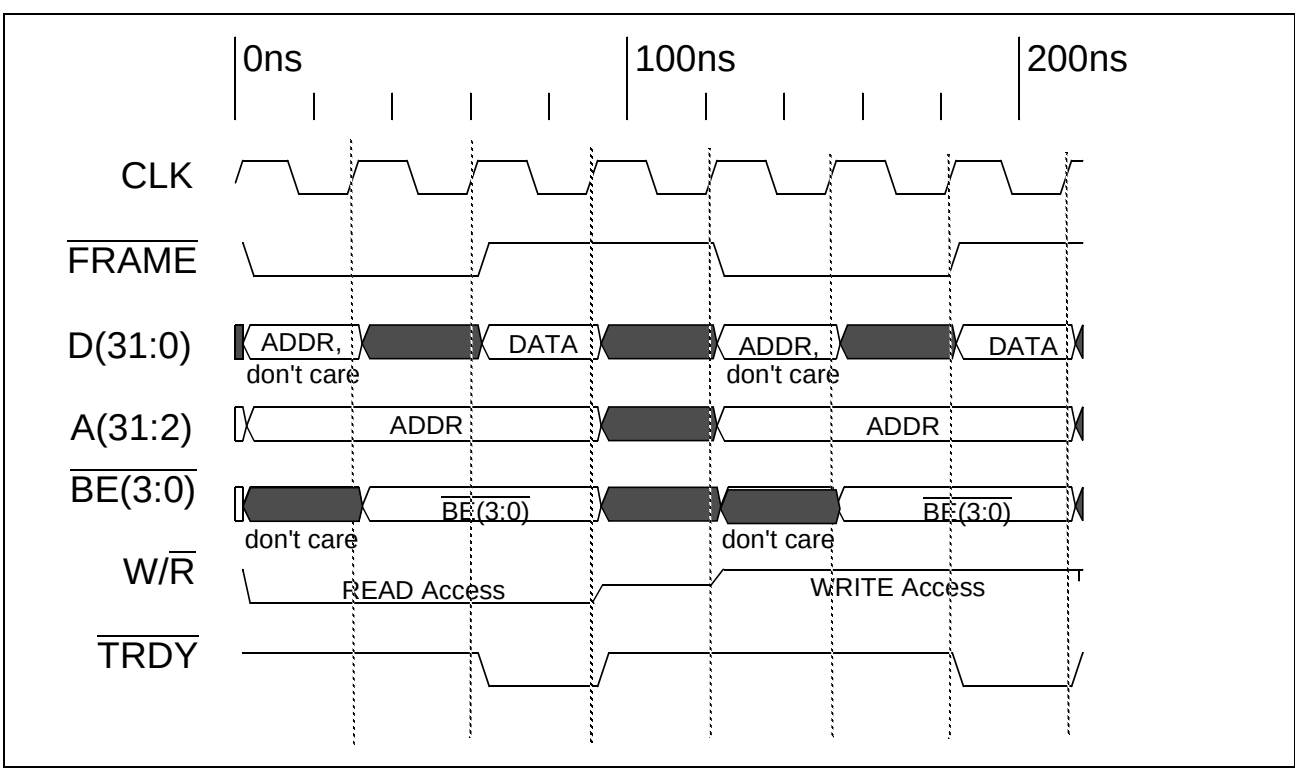


Figure 9 Master Single READ Transaction followed by a Master Single WRITE Transaction in De-multiplexed Configuration

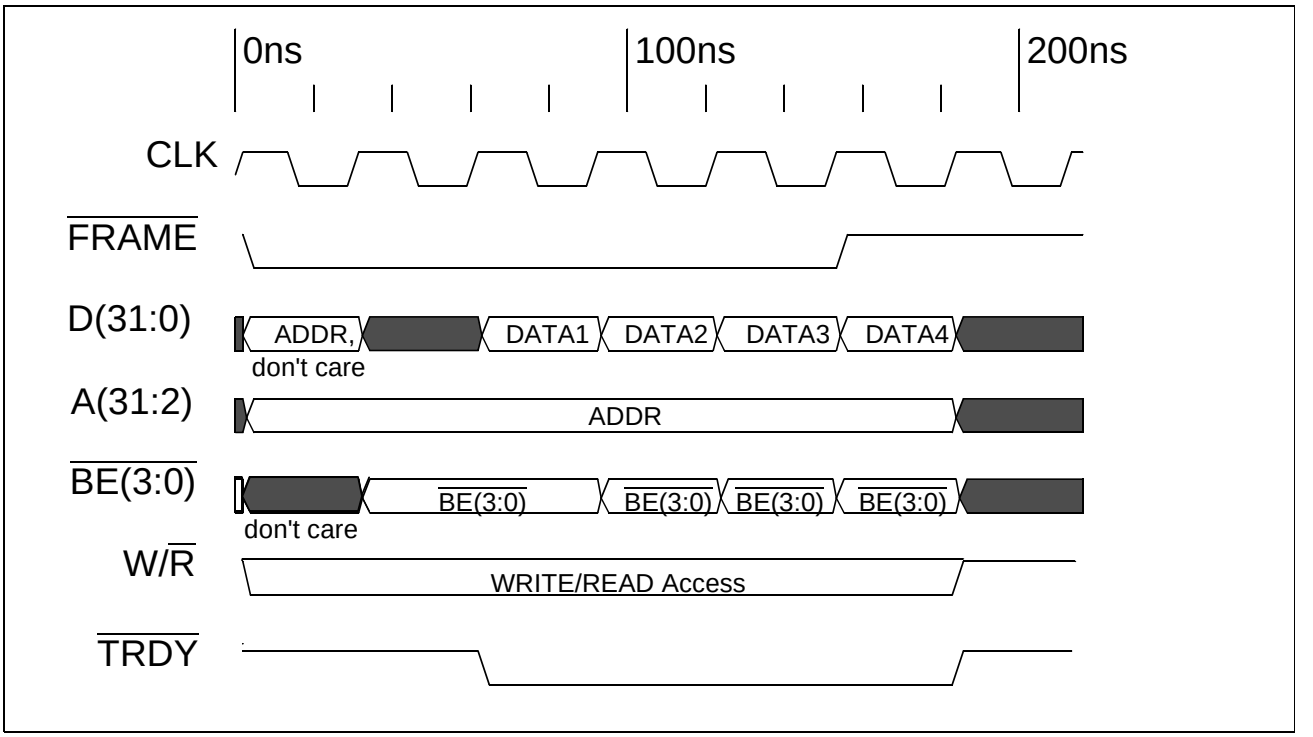


Figure 10 Master Burst WRITE/READ Transaction in De-multiplexed Configuration

When in De-multiplexed bus configuration, the DSCC4 adheres the PCI bus protocol and timing specificaton, except for the address and command handling. In this mode, the addresses are provided on a separate address bus A(31:2) to eliminate the need for external de-multiplexing buffers. The address lines A(31:2) correspond to the address lines AD(31:2) in PCI mode. The address becomes valid with the falling edge of FRAME and stays valid for the standard PCI address phase, the turn-around cycle and the entire data phase. In burst mode the addresses have to be incremented externally for each single transfer.

Moreover, in De-multiplexed mode the command signals are not used. Instead of the command signals a separate pin W/R (I/O) provides the Write/Read strobe signal. The Write/Read becomes valid with the falling edge of FRAME and stays valid for the standard PCI address phase, the turn-around cycle and the entire data phase.

The following four ’commands’ are supported:

Table 9 Supported Commands in De-multiplexed Bus Mode

W/$\overline{R}$	IDSEL	Master Mode	Slave Mode
0	0	memory read	DSCC4 register read
1	0	memory write	DSCC4 register write
0	1	not supported	DSCC4 PCI Configuration read
1	1	not supported	DSCC4 PCI Configuration write

Note: When designing a de-multiplexed system with the DSCC4 in De-multiplexed PCI mode it is the responsibility of the glue logic to meet the bus timing/protocol of the PCI specification and of the memory devices that are used in the system. When the DSCC4 operates in master mode, the bus cycle, for example, can be delayed by the $\overline{TRDY}$ signal.

5 DMA Controller and Central FIFOs

The DSCC4 has an 8-channel flexible DMA controller to perform data transfer with minimal host CPU intervention and high bus efficiency between the DSCC4 and memory.

These DMA channels service receive and transmit FIFOs of the four serial communication controllers (SCCs) transferring data into or out of the central DMA Controller FIFOs.

On the host system side each DMA channel transfers data either from the central receive FIFO to the shared memory (receive direction) or from the shared memory to the central transmit FIFO (transmit direction).

8 DMA channels, each corresponding to an individual SCC transmit or receive, operate on linked lists in the host memory.

A ninth DMA channel transfers interrupt vectors from the central interrupt FIFO to one of 10 interrupt vector queues located in the shared host memory.

Figure 11 shows the block diagram of the DMA Controller.

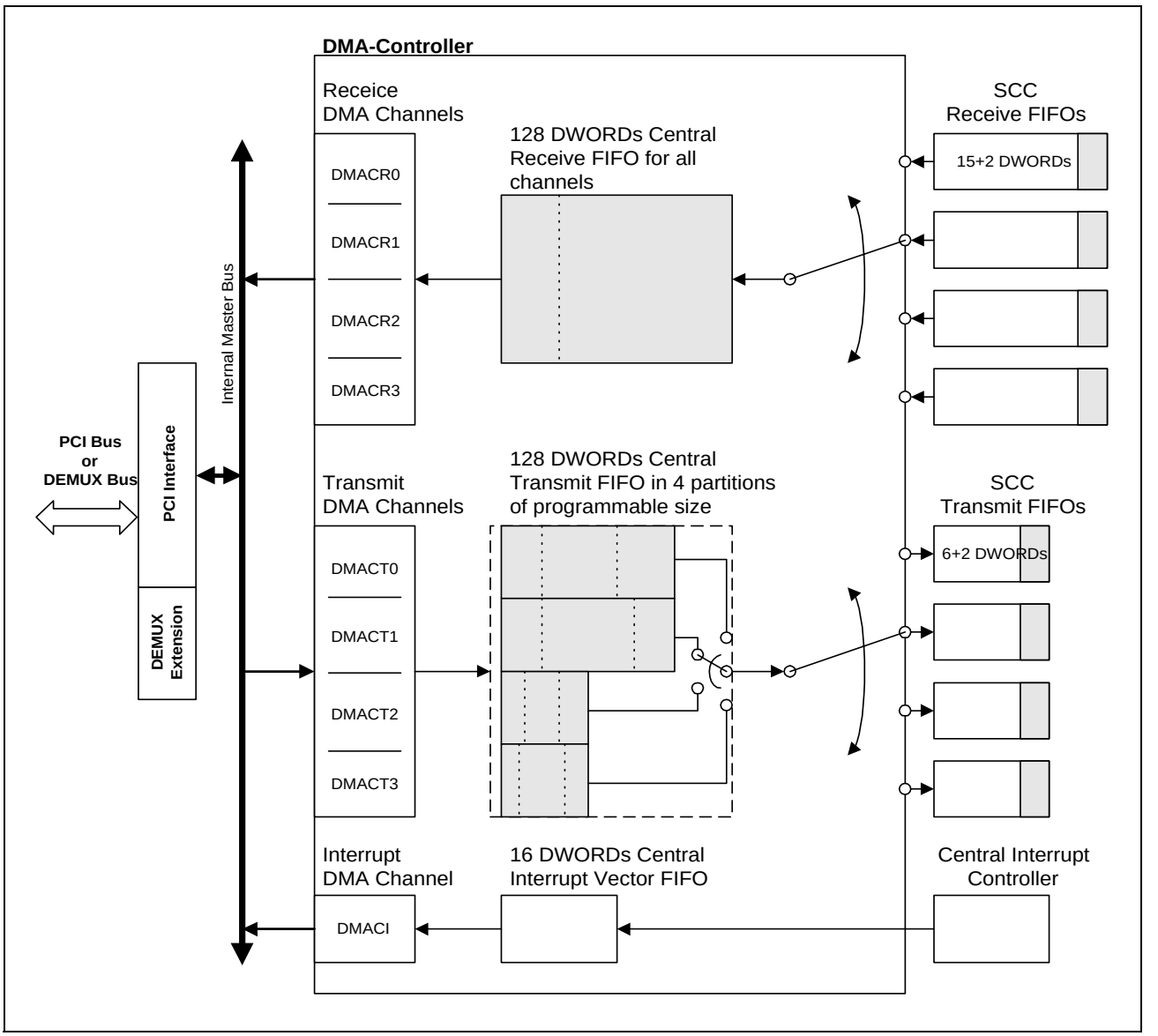


Figure 11 DMA Controller Block Diagram

A detailed description of the DMA channels and central FIFOs is provided in the following chapters.

5.1 DMAC Operational Description

5.1.1 DMAC Register Overview

The following table provides an overview of all DMA Controller related registers. For detailed register description refer to **Chapter 10**.

A summary is provided in the following table:

Table 10 DMA Controller Register Set

Offset Address relative to Base Address 0 (BAR0) in PCI Configuration Space (not DMA channel specific registers)	Register	Meaning
Global Control Registers		
0000_H	GCMDR	Global Command Register (see **Page 232**)
0004_H	GSTAR	Global Status Register (see **Page 237**)
0008_H	GMODE	Global Mode Register (see **Page 241**)
Global Interrupt Queue Control Registers		
$000C_H$	IQLENR0	Interrupt Queue Length Register 0 (transmit/receive interrupt queues) (see **Page 246**)
0010_H	IQLENR1	Interrupt Queue Length Register 1 (configuration and peripheral interrupt queue) (see **Page 248**)
$003C_H$	IQCFGBAR	Interrupt Queue Configuration Base Address (see **Page 252**)
0040_H	IQPBAR	Interrupt Queue Peripheral Base Address (see **Page 253**)
Global Central FIFO Control Registers		

0044_H	FIFOCR1	FIFO Control Register 1 (Transmit FIFO partition size, distinguished for 4 transmit channels) (see **Page 254**)
0048_H	FIFOCR2	FIFO Control Register 2 (Transmit FIFO refill threshold, distinguished for 4 transmit channels) (see **Page 255**)
$004C_H$	FIFOCR3	FIFO Control Register 3 (Receive FIFO threshold, common to all receive channels) (see **Page 257**)
0034_H	FIFOCR4	FIFO Control Register 4 (Transmit FIFO forward threshold, distinguished for 4 transmit channels) (see **Page 259**)

(DMA channel specific registers)

Ch0	**Ch1**	**Ch2**	**Ch3**	**(i=0..3)**	
0014_H	0018_H	$001C_H$	0020_H	IQSCCiRX BAR	Interrupt Queue Receive SCCi Base Address (see **Page 250**)
0024_H	0028_H	$002C_H$	0030_H	IQSCCiTX BAR	Interrupt Queue Transmit SCCi Base Address (see **Page 251**)
0050_H	$005C_H$	0068_H	0074_H	CHiCFG	Channel i Configuration Register (see **Page 261**)
0054_H	0060_H	$006C_H$	0078_H	CHiBRDA	Channel i Base Receive Descriptor Address (see **Page 264**)
0058_H	0064_H	0070_H	$007C_H$	CHiBTDA	Channel i Base Transmit Descriptor Address (see **Page 265**)
0098_H	$009C_H$	$00A0_H$	$00A4_H$	CHiFRDA	Channel i First Receive Descriptor Address (see **Page 266**)
$00C8_H$	$00CC_H$	$00D0_H$	$00D4_H$	CHiLRDA	Channel i Last Receive Descriptor Address (see **Page 268**)

$00B0_H$	$00B4_H$	$00B8_H$	$00BC_H$	CHiFTDA	Channel i First Transmit Descriptor Address (see **Page 267**)
$00E0_H$	$00E4_H$	$00E8_H$	$00EC_H$	CHiLTDA	Channel i Last Transmit Descriptor Address (see **Page 270**)

The following table provides an overview of all DMA Controller commands. For detailed register description refer to **Chapter 10**.

Table 11 DMAC Commands

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0000_H	r/w	CPU	00000200_H	**GCMDR**: Global Command Register (**Page 232**)
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	31..28 27..24 21 20	CFGIQ-SCCiRX and CFGIQ-SCCiTX and CFGIQ CFG, CFGIQP	0	Configure Interrupt Queue: These command bits cause the DMAC to establish or re-configure the dedicated interrupt queue using the values of the corresponding base address registers and interrupt queue length registers. (only performed if action request bit 'AR' is set additionally)
	13..10	TXPRi	0	Transmit Poll Request Channel i: If the DMA transmit channel is stopped on a HOLD condition (HOLD bit detected), this command forces a read transaction on the transmit descriptor verifying the HOLD condition again.
	9	IM	1	Interrupt Mask: If set to '1' the action request acknowledge interrupt is supressed.
	0	AR	0	Action Request: This bit causes the DMAC to execute all commands set in registers GCMDR and CHiCFG.

Table 11 DMAC Commands (cont'd)

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0050_H $005C_H$ 0068_H 0074_H	r/w	CPU	00000000_H	**CHiCFG** (i=0..3): Channel i Configuration Register (**Page 261**)
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	27..24	Interrupt Mask	0	This bit field is used to mask FI and ERR interrupt indications distinguished for transmit and receive DMA channels.
	The following commands are evaluated by the DMAC on any action request (bit 'AR' set in register GCMDR).			
	22	RDR	0	Reset DMA Receiver
	21	RDT	0	Reset DMA Transmitter
	20	IDR	0	Initialize DMA Receiver
	19	IDT	0	Initialize DMA Transmitter

5.1.2 DMAC Control and Data Structures

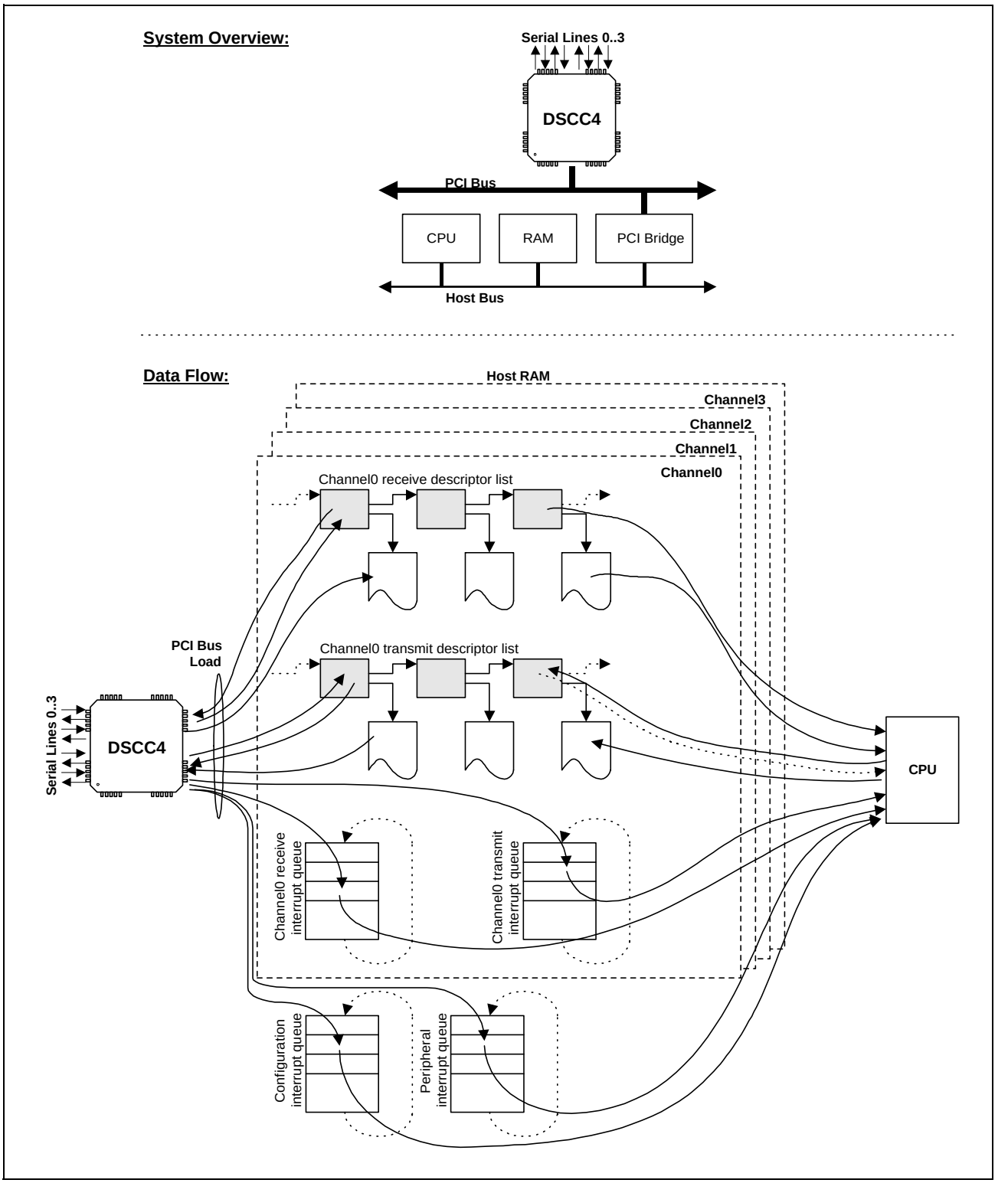


Figure 12 DMA Data Flow

The CPU prepares linked lists for transmit and receive channels in the shared memory. These may be handled by dynamically allocating and linking descriptors and buffers as needed during runtime or by static predefined memory structures e.g. ring-chained-lists (the ’last’ descriptor points back to the first descriptor). A mix of predefined descriptor lists but dynamically handled data buffers may also be an appropriate solution. This strategy depends on the specific application. The DMAC provides multiple control mechanisms supporting all of these combinations in an efficient way.

The descriptors and data buffers can be stored in separate memory spaces within the 32-bit address range allowing full scatter/gather methods of assembling and disassembling of packets.

Each descriptor contains a ’next descriptor address’ field to realize the linked list. Because the DMA controller cannot distinguish between valid and invalid addresses, a ’Hold’ mechanism is needed to prevent the DMA controller from branching to invalid memory locations. A ’next descriptor address’ might be invalid for several reasons:

- no further transmit transaction is requested; therefore no further transmit descriptor is allocated and the ’next descriptor address’ field of the last descriptor is invalid when read by the DMA controller;
- temporarily the software is not able to attach new receive descriptors to the list in time; therefore no further receive descriptor is allocated and the ’next descriptor address’ field of the last descriptor is invalid when read by the DMA controller;
- the receive descriptor list is organized as a ring; the DMA channel must be prevented from branching a descriptor which is not yet serviced by the CPU.

Two alternative control mechanisms are provided to detect and handle descriptor list end (Hold) conditions:

- Hold bit control mode
  (See “DMAC Operation Using Hold-Bit Control Mechanism” on page 76.)
- Last descriptor address control mode
  (See “DMAC Operation Using Last Descriptor Address Control Mode” on page 78.)

The Control Mode applies to all DMA channels transmit and receive and is selected via bit ’CMODE’ in Global Mode Register GMODE.

An HDLC frame may be contained in one buffer connected to one descriptor or it may be contained in several buffers each associated with linked descriptors. A ’frame end’ indication (FE bit) will be set in each descriptor which points to the last buffer of one HDLC frame.

The ’frame end’ indications are stored in the internal FIFOs influencing the FIFO control (threshold) mechanisms. Therefore ’frame end’ indications (FE bit) are also used in non frame oriented protocol modes such as ASYNC mode. They are referred to as ’frame end/block end’ indication in the following chapters.

5.1.2.1 DMAC Transmit Descriptor Lists

Each transmit descriptor consists of 4 consecutive DWORDs located DWORD aligned in the shared memory. The first 3 DWORDs are written by the host and read by the corresponding DMA channel using a burst transaction. They provide information about the next descriptor in the linked list, the attached transmit data buffer, and its size as well as some control bits.

The fourth DWORD is written by the DMA channel indicating that operation on this descriptor is finished.

The CPU will write the address of the first descriptor of each linked list to a dedicated Base Address Register (BTDAi) during initialization procedure. The corresponding DMA channel start serving the descriptor at these addresses.

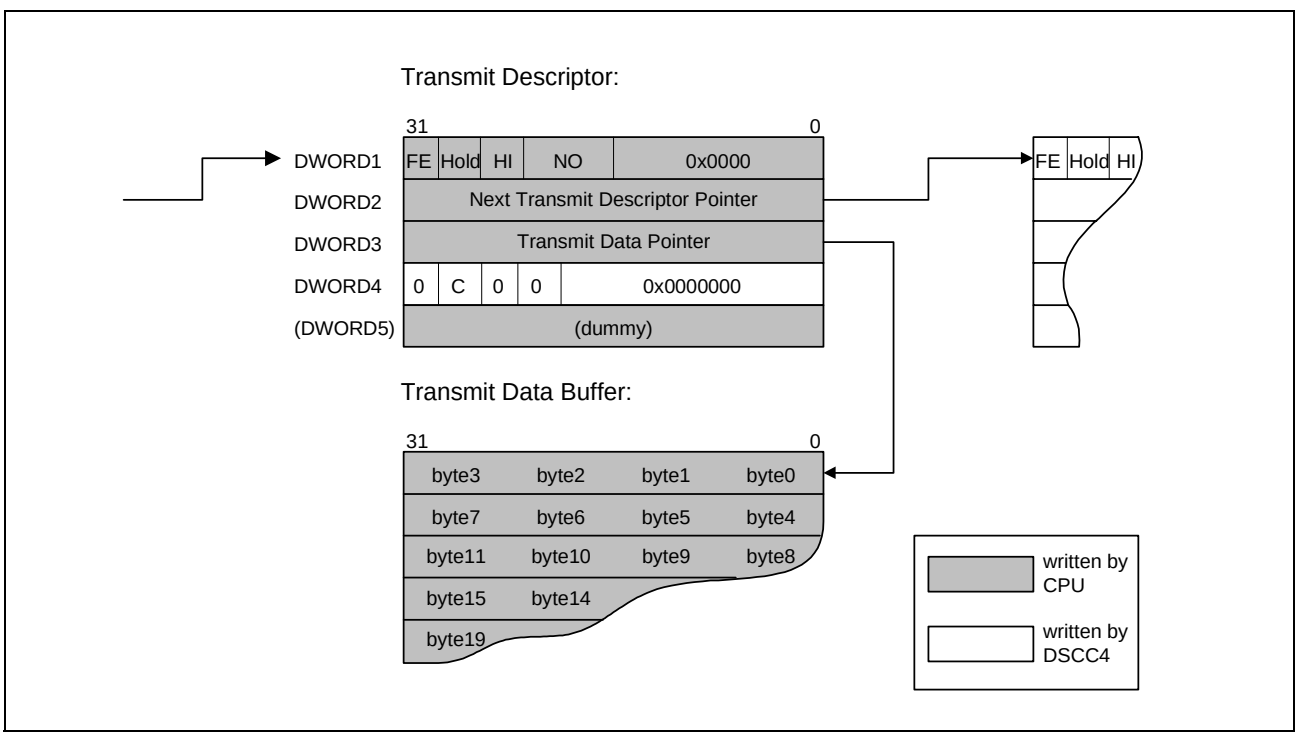


Figure 13 Transmit Descriptor List Structure

Table 12 Transmit Descriptor Bit Field Description

DWORD	Bit Field	Description
1 (read by DMAC, written by CPU)	FE	Frame End Indication: Not evaluated by the DMAC. This bit indicates that this descriptor contains a complete data packet or the last part of a data packet. This indication is forwarded to the corresponding SCC. An 'FI' interrupt is generated after completion of a transmit descriptor with FE='1' setting.
	Hold	Hold Indication: Hold='1' marks the end of the descriptor chain. In this case the DMAC will not branch to the next descriptor address. The DMAC reads and evaluates Hold bit and next descriptor address again on transmit poll request command.(see **Chapter 5.1.2.3**) (this bit is ignored if DMAC is configured in last descriptor address control mode)
	HI	Host Initiated Interrupt: This bit set to '1' causes the DMAC to generate an interrupt after completion of the descriptor and after transfer of the complete data section from Host memory to the central transmit FIFO. This may be used for software control purposes.
	NO	Number OF Bytes: This bit field determines the number of valid data bytes in the transmit data buffer and the data buffer size. The data buffer size is always n DWORDs, whereas n depends on NO and the byte offset address ADD (refer to ADD description on next pages).
2 (read by DMAC, written by CPU)	Next Tx Descr Ptr	Next Transmit Descriptor Address: The DMA Channel will branch to this address when proceeding in the linked list.
3 (read by DMAC, written by CPU)	Tx Data Buffer Ptr	Transmit Data Buffer Start Address: The DMA Channel starts reading transmit data at this address. Read access to transmit data buffer may occur per single DWORD transfers or up to 15 DWORDs burst transfers.

Table 12 Transmit Descriptor Bit Field Description (cont'd)

4 (written by DMAC, read by CPU)	C	Complete Bit: This bit is set by the DMAC after having completed the descriptor and corresponding data section. The software can use this indication for memory and linked list management.
5	-	Dummy DWORD; Only necessary if compatibility between transmit and receive descriptors is needed, i.e. receive descriptors are manipulated by the host and attached to a transmit descriptor list. For transmit operation, DWORD5 is neither used by the DMAC nor by the host CPU.

In detail the DMA controller reads a transmit buffer descriptor, calculates the data buffer address and data buffer length and transfers data up to the burst size from the data buffer into the central transmit FIFO. For a frame longer than the burst size, this operation is repeated as long as the transmit FIFO requests for data. For more information about FIFO control see **Chapter 5.2**.

After the data buffer has been transferred, the controller marks the descriptor "completed" and branches to the next descriptor if applicable. An 'FI' interrupt will be generated, if the currently completed descriptor contained an 'frame end/block end' (FE='1') indication.

In HDLC mode data is transmitted as frames. The host indicates the end of a frame by setting 'FE' bit in the transmit descriptor. When a frame end is detected the DMA channel forwards this information to the SCC. The SCC then terminates the transmission by appending the CRC and the closing flag sequence to the data. If (FE=0 & HOLD=1) or (FE=0 & FTDA=LTDA) an ERR interrupt is generated by the DMA controller (see **Chapter 5.1.2.3** and **Chapter 5.1.2.4**).

Note: In contrast to HDLC mode all other modes (ASYNC, BISYNC, Extended Transparent Mode) are block/character oriented. Since the DMA controller does not distinguish between different protocol modes (HDLC, ASYNC, ...) the 'FE' bit in the last descriptor of the linked list might be set also for the block/character oriented modes as a kind of block end indication or together with an end of list condition.

Although the DSCC4 works only DWORD oriented, it is possible to begin a transmit data section at an uneven (not DWORD aligned) address. The two least significant bits (ADD) of the transmit data pointer determine the beginning of the data section and the number of data bytes in the first DWORD of the data section, respectively. Nevertheless the DSCC4 will always perform DWORD read transfers (all byte enables valid) on transmit data sections marking invalid bytes internally.

Table 13 shows how many bytes can be valid in the first 32-bit word depending on ADD for both little endian mode and big endian mode. Of course, the total number of valid bytes depends on the 'NO' bit field value in the corresponding transmit descriptor.

Table 13 Meaning of ADD in Little/Big Endian Mode

ADD	Number of Valid Bytes	Little Endian (Intel)				Big Endian (Motorola)			
		11	**10**	**01**	**00**	**11**	**10**	**01**	**00**
00	4, if NO > 3	byte 3	byte 2	byte 1	byte 0	byte 0	byte 1	byte 2	byte 3
01	3, if NO > 2	byte 2	byte 1	byte 0	-	-	byte 0	byte 1	byte 2
10	2, if NO > 1	byte 1	byte 0	-	-	-	-	byte 0	byte 1
11	1, if NO > 0	byte 0	-	-	-	-	-	-	byte 0
Example A: NO = 3									
00	3	-	byte 2	byte 1	byte 0	byte 0	byte 1	byte 2	-
Example B: NO = 2									
01	2	-	byte 1	byte 0	-	-	byte 0	byte 1	-
Example C: NO = 8									
10	2	byte 1	byte 0	-	-	-	-	byte 0	byte 1
	4	byte 5	byte 4	byte 3	byte 2	byte 2	byte 3	byte 4	byte 5
	2	-	-	byte 7	byte 6	byte 6	byte 7	-	-
Example D: NO = 1									
11	1	byte 0	-	-	-	-	-	-	byte 0

The following figure provides an example, how a transmit descriptor and its associated data buffer is located in the memory as a result of a memory dump.

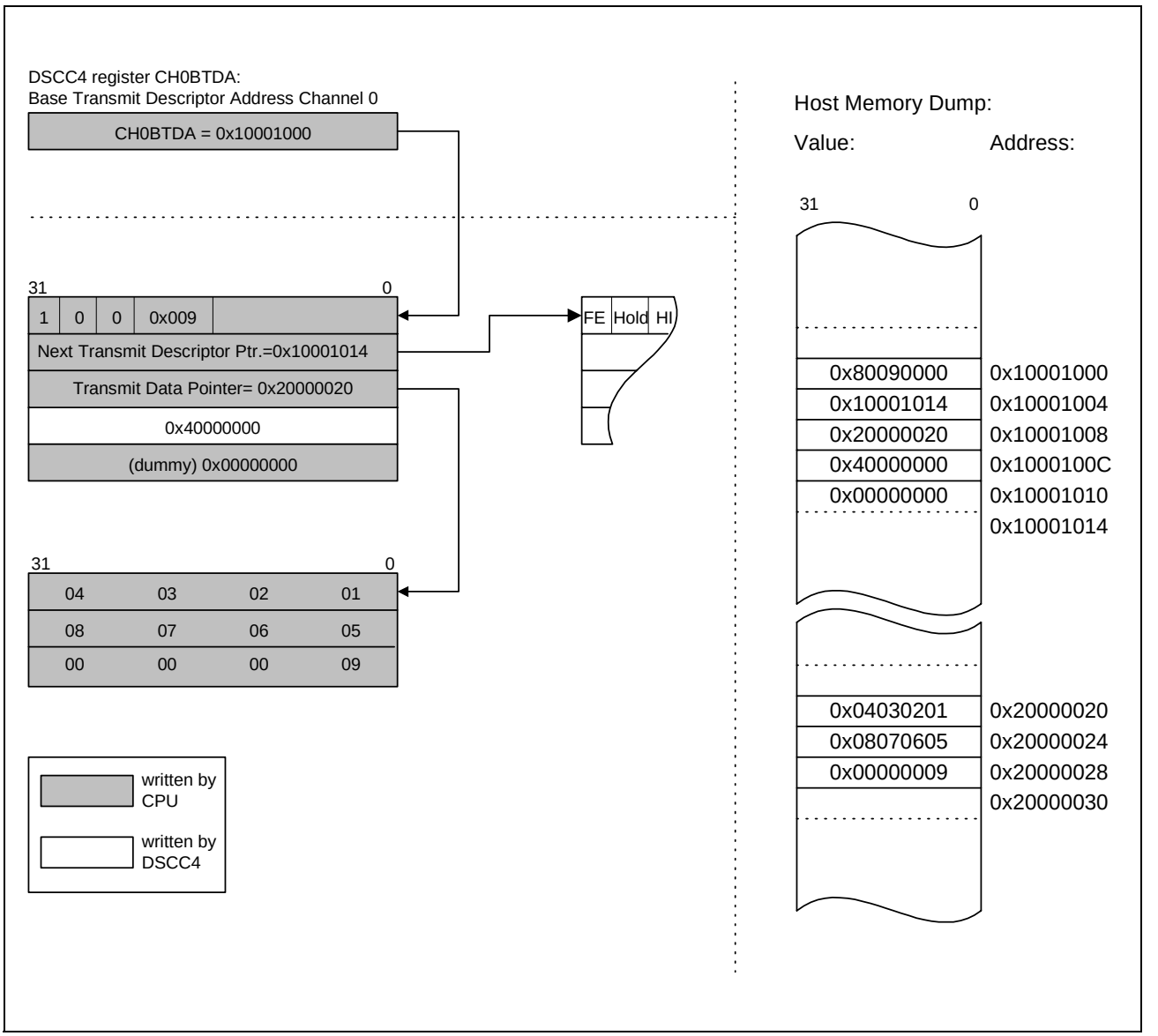


Figure 14 Transmit Descriptor Memory Example

Note: Although transmit descriptors consist of only 4 DWORDs it might be useful to allocate 5 DWORD structures to achieve compatibility between transmit and receive descriptors. In this case only small CPU performed manipulations are necessary to convert a completed receive descriptor into a transmit descriptor re-chained to a transmit descriptor list. This is typical e.g. for frame relay or bridging applications where received data might be sent out again on another DSCC4 port.

5.1.2.2 DMAC Receive Descriptor Lists

Each receive descriptor consists of 5 consecutive DWORDs, located DWORD aligned in the shared memory. The first 3 DWORDs are read by the corresponding DMA channel using a burst transaction and provide information about the buffer size as well as some control bits, the next descriptor in the linked list and the attached receive data buffer.

The fourth DWORD is written by the DMA channel indicating that operation on this descriptor is finished. The fifth DWORD is also written by the DMA channel but only in descriptors containing the first or only data section of an HDLC frame or data block. It is a pointer to the last descriptor containing the frame or block end (’FE’ bit) allowing the software to unchain the complete partial descriptor list containing one frame or block without parsing through the list for ’FE’ indication. It is written after the frame has been completely written to the shared memory.

The CPU will write the address of the first descriptor of each linked list to a dedicated Base Address Register during initialization procedure. The corresponding DMA channels start operating the linked lists at these addresses.

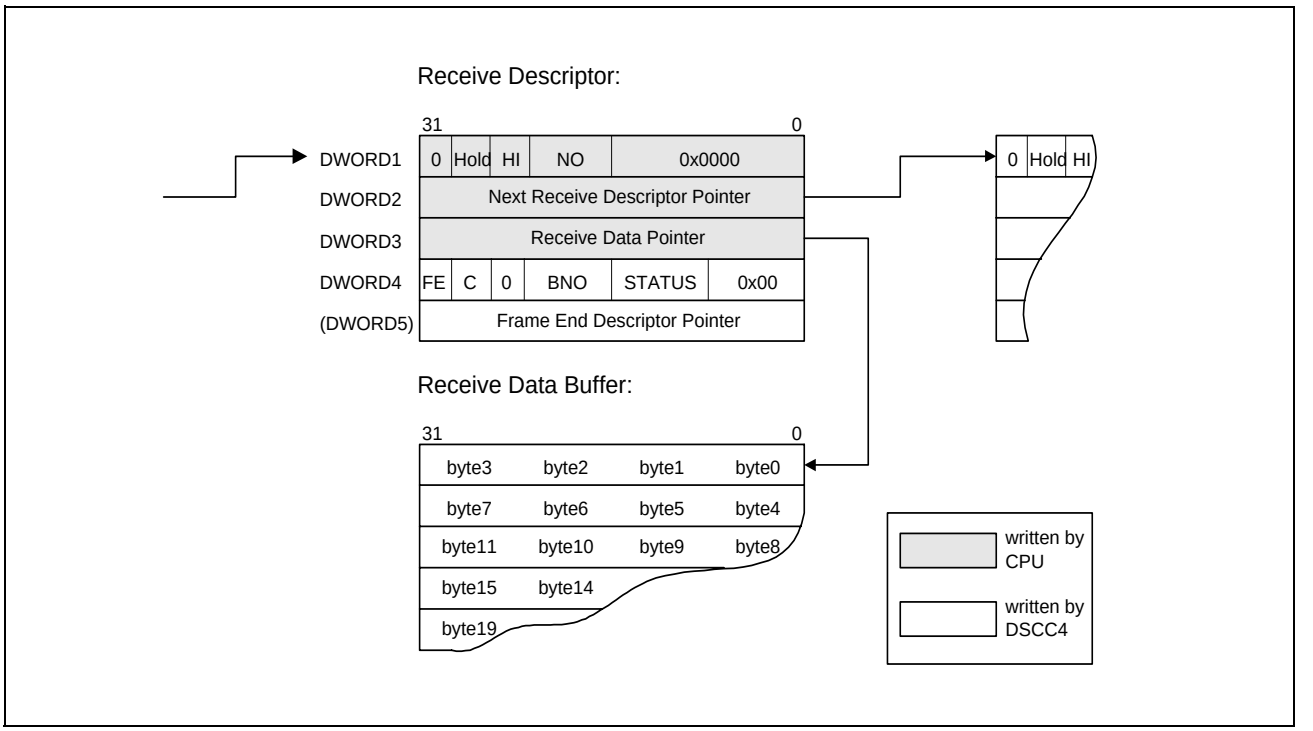


Figure 15 Receive Descriptor List Structure

Table 14 Receive Descriptor Bit Field Description

DWORD	Bit Field	Description
1 (read by DMAC)	Hold	Hold Indication: Hold='1' marks the end of the descriptor chain. In this case the DMAC will not branch to the next descriptor address but stop DMA operation until initialized again (see **Chapter 5.1.2.3**). (this bit is ignored if DMAC is configured in last descriptor address control mode)
	HI	Host Initiated Interrupt: This bit set to '1' causes the DMAC to generate an interrupt after completion of the descriptor and after transfer of the complete data section to the Host memory. This may be used for software control purposes.
	NO	Number Of Bytes: This bit field determines the receive data buffer size and should be a multiple of 4. *Note: The number of received data bytes includes the receive status byte (RSTA) which is generated by the SCC receiver in HDLC mode or all status bytes optionally generated in character oriented protocol modes respectively.*
2 (read by DMAC)	Next Rx Descr Ptr	Next Receive Descriptor Address: The DMA Channel will branch to this address when proceeding in the linked list.
3 (read by DMAC)	Rx Data Buffer Ptr	Receive Data Buffer Start Address: The DMA Channel starts writing receive data at this address. Write access to receive data buffer may occur per single DWORD transfers or up to 15 DWORDs burst transfers.

Table 14 Receive Descriptor Bit Field Description (cont'd)

4 (written by DMAC)	FE	Frame End Indication: This bit set to '1' indicates that this descriptor contains a complete data packet or the last part of a data packet. An 'FI' interrupt is generated after completion of a receive descriptor with FE='1' setting.
	C	Complete Bit: This bit is set by the DMAC after having completed the descriptor and corresponding data section. The software can use this indication for memory and linked list management. *Note: The Frame End Descriptor Pointer (DWORD 5) is written to the receive descriptor after writing the Complete Bit indication to DWORD 4.*
	BNO	Number Of Valid Bytes: This bit field indicates the number of valid bytes stored in the receive data buffer upon completion.
	STATUS	Receive Status. Refer to **Page 379** for details.
5 (written by DMAC)	FE Descr Ptr	Frame End Descriptor Pointer: This address is only written to the first descriptor of a data packet pointing to the descriptor containing the 'FE' indication of the same packet. If the complete packet is stored in the first and only data section this address is equal to the descriptor address.

In details the DMA controller reads a receive buffer descriptor, calculates the maximum data buffer size and the data buffer address and transfers data up to the burst size from the receive FIFO into the data buffer. For a frame longer than the burst size, this operation is repeated as long as data is available in the FIFO. For more information about FIFO control see **Chapter 5.2**).

After the data buffer has been filled, the controller writes the number of stored bytes into the descriptor, marks the descriptor "completed" and branches to the next descriptor. When a frame end (HDLC) or block end (e.g. termination character or time out in ASYNC mode) is detected and all data has been transferred, the DMA controller writes the Frame End Descriptor Pointer to the descriptor containing the beginning of the frame. A maskable interrupt status entry is written into the interrupt queue, if initiated by the host. As a last transaction the DMA controller writes the number of bytes stored in the last buffer as well as a DMA controller related status byte in the descriptor.

The DMA controller related status byte indicates, if the frame or block was ended normally or by a receiver reset command or by a HOLD bit in the current descriptor.

In HDLC mode the last byte in the data section contains the status information of the HDLC frame such as: result of CRC check, data overflow, frame aborted. This byte is forwarded transparently from the SCC to the data buffer following the received data.

In ASYNC mode and BISYNC mode to every data byte an attached status byte can be stored. This byte is forwarded transparently from the SCC to the data buffer and it contains status information such as: parity, parity error, framing error.

Since the threshold of the SCC specific receive FIFOs can be set to 1, 2, 4, 16 or 24 bytes the receive data buffer DWORDs can contain less than four valid bytes (e.g. 1 or 2 bytes) In this case the data buffer contains holes of invalid data bytes. Refer to **Table 69 "CCR2: Channel Configuration Register 2" on Page 296**.

Table 15 provides examples for the receive data section.

Table 15 Receive Data Buffer Section

Mode	BNO	Little Endian				Big Endian			
		11	10	01	00	11	10	01	00
HDLC (Default)	4	RSTA	byte 2	byte 1	byte 0	byte 0	byte 1	byte 2	RSTA
HDLC	7	byte 3	byte 2	byte 1	byte 0	byte 0	byte 1	byte 2	byte 3
(Default)			RSTA	byte 5	byte 4	byte 4	byte 5	RSTA	
ASYNC	6			status 0	data 0	data 0	status 0		
(RFDF=1				status 1	data 1	data 1	status 1		
RFTH=00)				status 2	data 2	data 2	status 2		

Note: In general, BNO counts the valid bytes that have been transferred into the data buffer including RSTA (BNO <= NO). The receive status byte (RSTA) is treated and counted as 'data' by the DMA controller. As an example, an HDLC frame containing 32 bytes to be transferred to the shared memory needs 33 bytes in the receive data buffer due to the receive status byte, which is attached to the data by the SCC. If NO =32 in this example, the receive status byte as well as the frame end indication will be written to the next data buffer and descriptor respectively.

The following figure provides an example how a receive descriptor and its associated data buffer is located in the memory as a result of a memory dump.

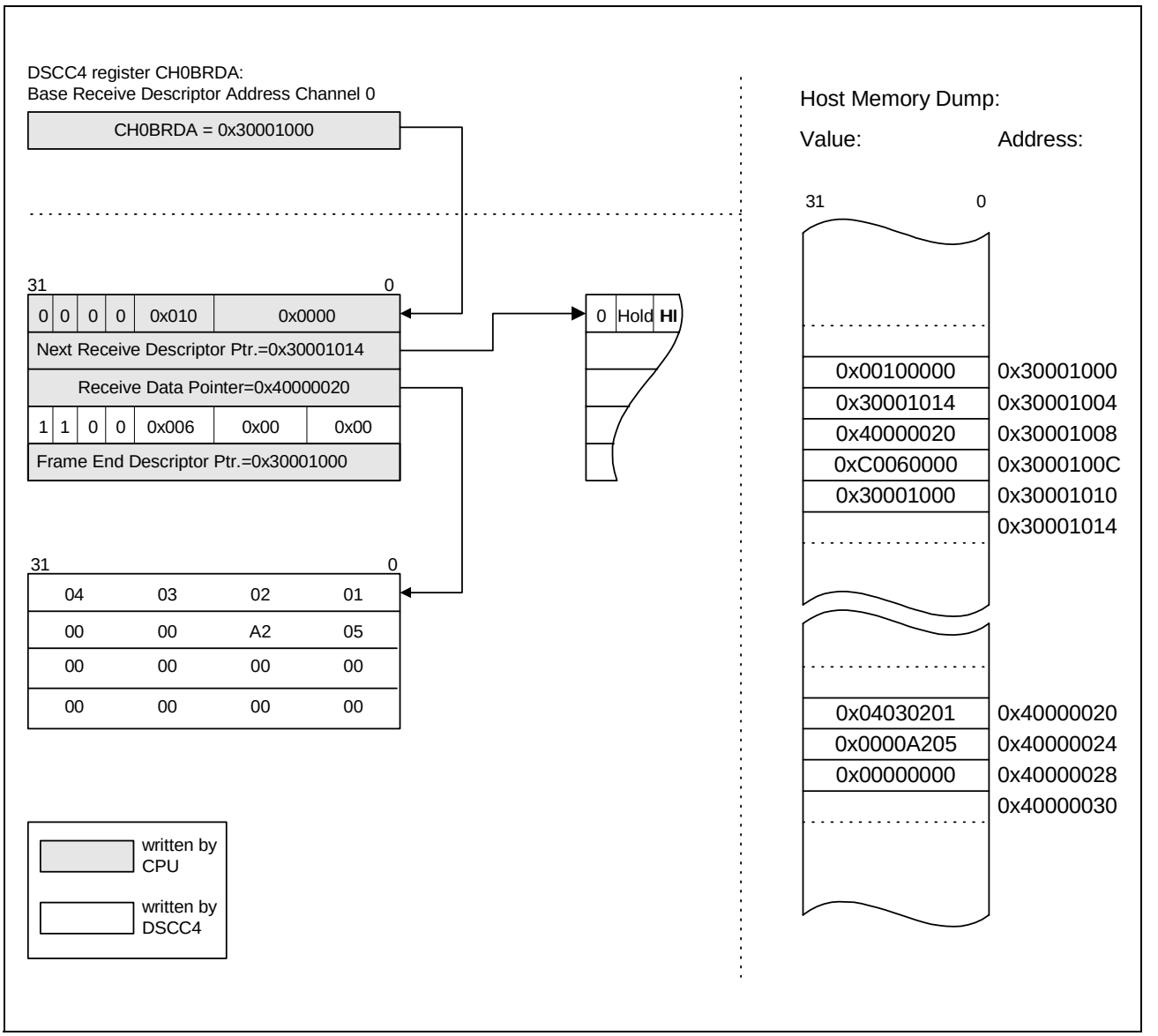


Figure 16 Receive Descriptor Memory Example

5.1.2.3 DMAC Operation Using Hold-Bit Control Mechanism

This mode is selected by setting bit CMODE='0' in register GMODE (see **"GMODE: Global Mode Register" on Page 241**).

The DMA controller operates on linked lists with pointer information stored in the DSCC4 internal configuration section.

The software starts DMAC operation by writing the action request bit 'AR' in the Global Command Register (GCMDR). On this command, the DMAC checks all channel specific Configuration registers CHiCFG for initialization command bits ('IDR', 'IDT').

Each DMA channel which is triggered for initialization by one of the above mentioned commands fetches the Base Transmit/Receive Descriptor Address (BTDA/BRDA) from its CHiBTDA/CHiBRDA register. The DMA channel continues reading the Tx/Rx descriptors from the shared memory which in turn point to the associated data buffer and the next descriptor address.

The external memory associated to a DMA channel is organized as a chained list of buffers (typically 256 bytes) set up by an external host. Each chained list is composed of descriptors and data sections. The descriptor contains the pointer to the next descriptor, the start address of the data buffer and the size of a data section. In transmit direction the data pointer is a byte address. The descriptor also includes control information like frame end (frame end for HDLC modes, block end indication for ASYNC and extended transparent mode), transmission hold and host initiated interrupt.

Transmit:

- In transmit direction the DMA controller reads a transmit descriptor, calculates the data address and fills the DSCC4 central transmit FIFO. When the data transfer of the specified section is completed, the DMA controller marks the buffer as "completed" and branches off to the next transmit descriptor. If a frame end (FE) is indicated in the buffer descriptor (HDLC and block oriented protocol modes), a frame end indication is forwarded to the serial channel after the data has been transferred, and the frame will be closed correctly by the SCC. A maskable interrupt status may be generated at the completion of transmission to be stored in the interrupt queue by the interrupt controller. Transmission of another frame can begin immediately.
  However, if the current transmit buffer descriptor has its "HOLD" bit set, the DMA channel does not branch off to the next descriptor. If no frame end was encountered in the current descriptor, an active "HOLD" bit causes a transmit FIFO underrun to occur, and a frame to be aborted by the serial channel.
  Furthermore an error interrupt is generated anytime a transmit channel detects the HOLD condition without a frame end indication asserted.
  Once, the DSCC4 has sensed the HOLD='1' condition in transmit direction, the data transfer can be restarted by a single poll command (GCMDR:TXPOLLi=1; i=0...3) or by setting the 'AR' bit in the GCMDR register after the configuration register has been updated for the corresponding channel (new channel initialization). The transmit poll command causes the channel to read the HOLD bit (first DWORD of the descriptor)

and the next descriptor address of the current descriptor again. The DMAC branches to the next descriptor address if HOLD bit has been reset to ’0’.
There are three methods to handle the situation in which the next descriptor address pointer is not yet valid in the current transmit descriptor. This happens if the location of the next descriptor is not known when preparing the last descriptor of the chain:
1.Updating the next descriptor address before providing transmit poll request command.
2. Updating BTDA address in register CHiBTDA followed by a channel initialization command (’IDT’).
3. Programming the "next transmit descriptor address" of each descriptor with HOLD bit set pointing to a fixed dummy descriptor with byte number 0 (NO = 0). The next descriptor address of the dummy descriptor can be programmed to point to a newly prepared descriptor list before providing a transmit poll request command. The DMA channel will continue branching to the dummy descriptor, completing it without data transfer and branch to the next descriptor which itself might point to the dummy descriptor again if HOLD bit is set.

Receive:

- If the current receive buffer descriptor has its HOLD-bit set and the current data buffer has been filled, the DMA channel does not branch off to the next descriptor. An active “HOLD” bit causes a receive FIFO overflow to occur. If more data are received, those are discarded by the serial channel.
  Furthermore an error interrupt is generated anytime a receive channel detects the HOLD condition.
  The Host CPU (software) is expected to prepare sufficient amount of receive descriptors which is supported by several control mechanisms. Thus detecting a Hold condition in the receive descriptor list is treated as an exceptional condition by the corresponding DMA channel.
  Once, the DSCC4 has detected the HOLD=’1’ condition in receive direction, an interrupt is generated after completion of the current receive descriptor and the corresponding DMA channel is deactivated for receive direction as long as the host does not request an initalization via action request command in register GCMDR. The data transfer can be restarted by setting the AR bit in the GCMDR register after the channel specific configuration register has been updated (new initialization).

5.1.2.4 DMAC Operation Using Last Descriptor Address Control Mode

This mode is selected by setting bit CMODE='1' in register GMODE (see **"GMODE: Global Mode Register" on Page 241**).

The DMA controller operates on linked lists with pointer information stored in the DSCC4 internal configuration section.

The initialization procedure as well as the CPU to DSCC4 handshaking is equal to the HOLD-Bit control mode as described in **Chapter 5.1.2.3** with the following exception:

The host CPU does not take care about the HOLD bit. The address of the descriptor in the chain at which a 'Hold' is to be exercised is written to the corresponding LTDA/LRDA register. The DMA channel compares its current (first) descriptor address to LTDA/ LRDA. When a match occures, a 'Hold' condition is activated. After attaching at least one new descriptor to the linked list. Also the DMA channel does not take care on the HOLD bit within the descriptors but compares its current descriptor address with LTDA/LRDA register value. In case of address match this condition is equal to the HOLD-condition.

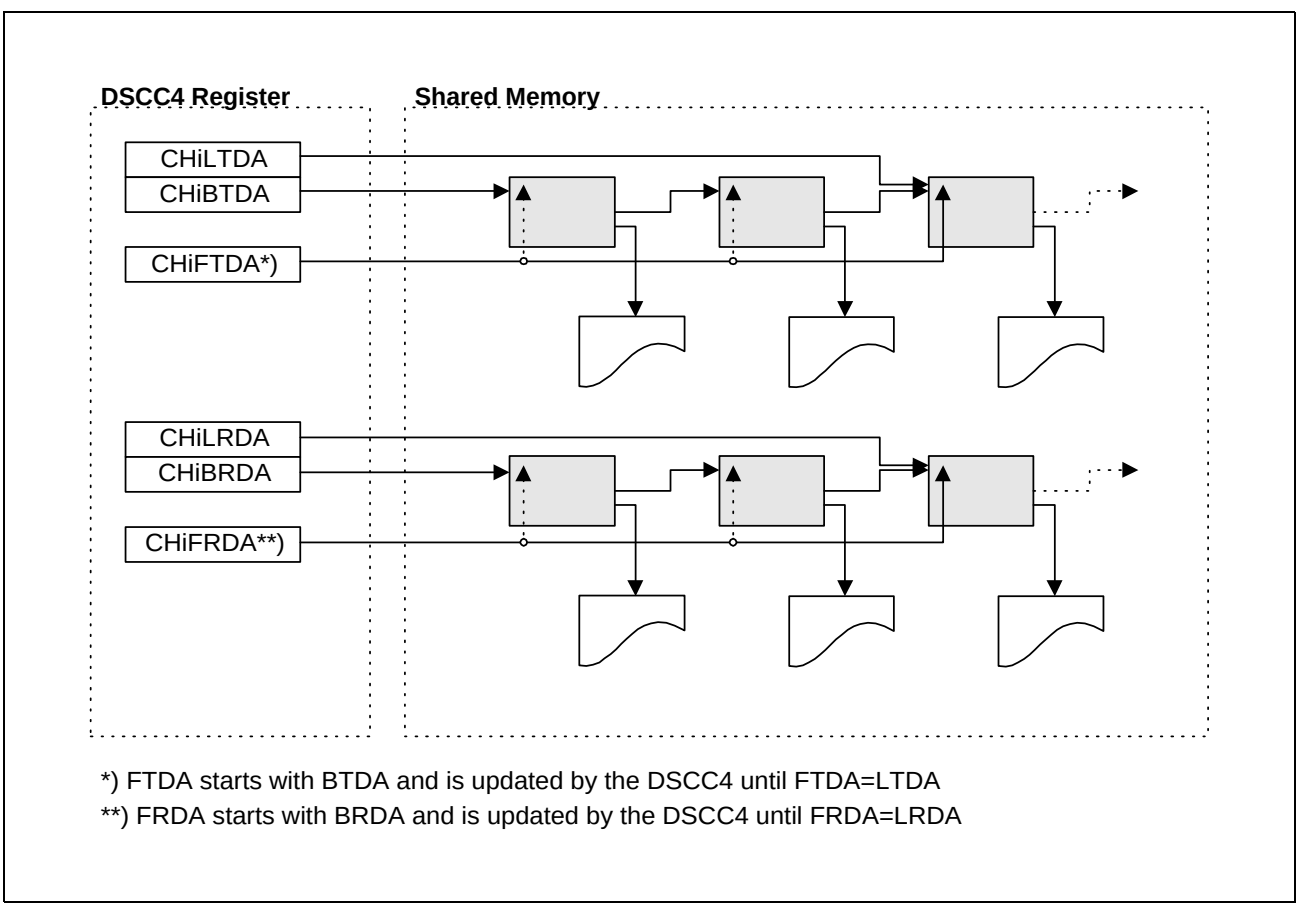


Figure 17 Data Transfer controlled via first and last descriptor addresses

After initialization the DMAC internally starts with the Base Tx/Rx Descriptor Address BTDA/BRDA as the "first descriptor address" since this address points to the first descriptor of the linked list that has to be processed by the DMA channel. The current

descriptor address is stored in register FTDA/FRDA as the (current) first descriptor address. With branching to the next descriptor the first descriptor address register is updated. Thus the host can keep track of the DMAC's progress by reading the FTDA/FRDA register. Beside the base descriptor address the user provides the Last Tx/Rx Descriptor Address in the corresponding LTDA/LRDA register. After transferring all data from the current data section and before branching to the next descriptor, the DMAC compares FTDA/FRDA and LTDA/LRDA. If the corresponding addresses are not identical the DMA channel branches to the next descriptor and the FTDA/FRDA register is updated with "next descriptor address" to continue normal transfer operation. If a match occurs the DMA channel is suspended until the host writes a new last transmit/receive descriptor address to LTDA/LRDA register. After write access to these registers the DSCC4, again, compares FTDA/FRDA and LTDA/LRDA and proceeds as described above.

- In transmit direction the condition LTDA equal to FTDA corresponds to the transmit HOLD condition in HOLD bit controlled mode. In this case updating register LTDA is the equivalent operation to transmit poll request command.
- In receive direction the condition LRDA equal to FRDA corresponds to the receive HOLD condition in HOLD bit controlled mode. Updating register LRDA will cause the DMA channel branching to the next receive descriptor. No new initialization command 'IDR' is necessary as in HOLD bit controlled mode.
  Re-initialization might be necessary if the linked list does not continue at the "next descriptor address" of the last descriptor but on any other address.

There are three methods to handle the situation in which the next descriptor address pointer is not yet valid in the current transmit descriptor. This happens if the location of the next descriptor is not known when preparing the last descriptor of the chain:

1. Programming the "next descriptor address" of each last descriptor pointing to a newly prepared descriptor list before updating register LTDA/LRDA. After write access to LTDA/LRDA register, the DMA controller reads the next descriptor address again and branches to the new descriptor.
2. Updating BTDA/BRDA address in register CHiBTDA/CHiBRDA followed by a channel initialization command ('IDT'/'IDR').
3. Programming the "next descriptor address" of each last descriptor pointing to a fixed dummy descriptor with byte number 0 (NO = 0). The next descriptor address of the dummy descriptor must be programmed to point to a newly prepared descriptor list before updating register LTDA/LRDA. The DMA channel will branch to the dummy descriptor, completing it without data transfer, then branch to the next descriptor which itself might point to the dummy descriptor again.

.

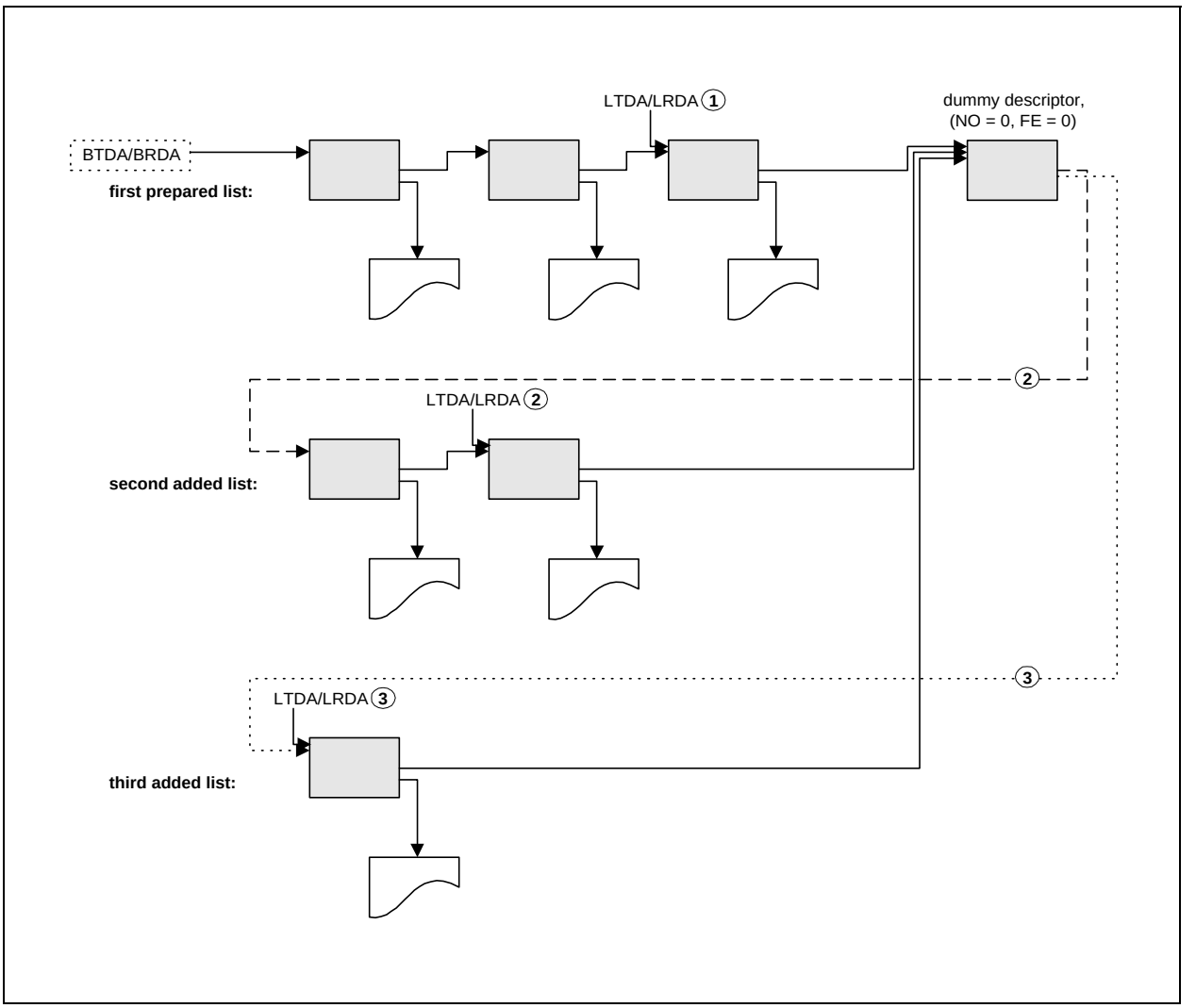


Figure 18 Example: 'Chain Jump' Handling per 'Dummy Descriptor'

5.1.3 DMAC Interrupt Controller

The DSCC4 interrupt concept is based on 32-bit interrupt vectors generated by the different blocks. Interrupt vectors are stored in a central interrupt FIFO which is 16 DWORDs deep. The interrupt controller transfers available vectors to one of ten circular interrupt queues located in the shared memory depending on the source ID of each interrupt vector.

In addition new interrupt vectors are indicated in the global status register GSTAR on a per queue basis and selectively confirmed by writing ’1’ to the corresponding GSTAR bit positions. The PCI interrupt signal $\overline{\text{INTA}}$ is asserted with any new interrupt event and remains asserted until all events are confirmed.

Each interrupt queue length and memory location can be configured via specific interrupt queue base address registers and two shared interrupt queue length registers. The queue length is individually programmable in multiples of 32 DWORDs (see **Page 246**).

One dedicated interrupt queue is provided per SCC channel and direction (IQSCCiRX and IQSCCiTX). Non channel specific interrupt vectors generated by the DMAC itself are transferred to the configuration queue IQCFG. The peripheral interrupt queue IQP is used for vectors generated by one of the blocks GPP, SSC or LBI.

The internal blocks provide mask registers for suppressing interrupt indications. Masked interrupts will neither generate an interrupt vector nor an $\overline{\text{INTA}}$ signal and GSTAR indication. (Refer to figure **“DSCC4 Logical Interrupt Structure” on Page 83**.)

The DMA controller (interrupt controller) itself generates 6 channel specific interrupts regarding the transmit and receive descriptor handling:

- Host Initiated interrupt (HI):
  This interrupt can be forced by setting bit ’HI’ in the receive and transmit descriptor. In this case the DMAC will generate an HI-interrupt with completion of this descriptor i.e. when the DMAC is ready to branch to the next descriptor address. This might be used to monitor the progress of the corresponding DMA channel on the descriptor list. As an example the HI interrupt can be used to dynamically request attachment of new receive descriptors to the list if the DMA channel comes close to the list end.
- Frame Indication interrupt (FI):
  This interrupt is generated with completion of any receive and transmit descriptor with a set ’frame end/block end’ indication, i.e. FE=’1’.
- Error interrupt (ERR):
  Indicates an unexpected descriptor configuration.
  receive descriptor:
  ERR is generated if receive data cannot be transferred to the shared memory completely because the frame (block) does not fit into the current data section and a HOLD condition (HOLD bit or LRDA=FRDA) prevents the DMAC from branching to the next descriptor.

ERR is also generated if an already started DMA transfer is aborted by a receive DMA reset (RDR) command.
transmit descriptor:
In transmit direction an ERR interrupt is generated if one of the following descriptor settings is detected
- HOLD=’1’ and FE=’0’ (the already started transmit frame could not be finished)
- LTDA=FTDA and FE=’0’ (the already started transmit frame could not be finished)
- FE=’0’ and NO=’0’ (a packet of length 0 is supposed to be a ’frame’ with FE bit set)

The DMA controller will continue ’normal’ operation in case of an ERR event. Nevertheless these cases may result in receive data overflows or transmit data underruns.

FI and HI interrupt indications caused by one descriptor will be generated into one interrupt vector with ’HI’ and ’FI’ bit set.

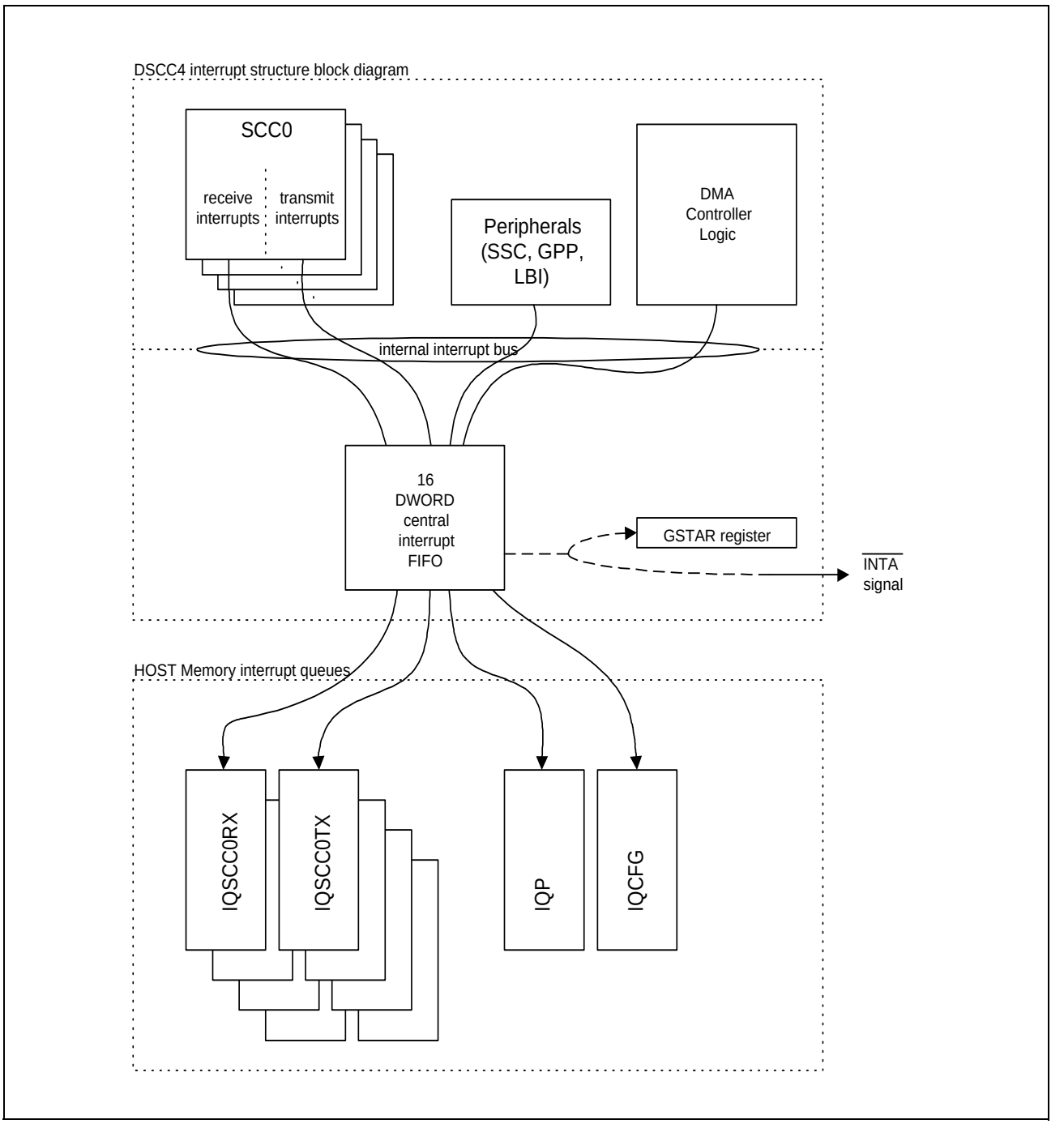


Figure 19 DSCC4 Logical Interrupt Structure

According the interrupt service routine (ISR) two different solutions (or any mix of them) are possible:

- Traditional ISR entry on interrupt event ($\overline{\text{INTA}}$ signal asserted):
  On interrupt event the CPU jumps to the ISR entry point. Within the ISR the DSCC4 global status register GSTAR is read indicating which interrupt queues store at least one new interrupt vector. The interrupt indication is confirmed by writing a ’1’ to the

queue specific bit positions in register GSTAR and all new vectors are read from the corresponding interrupt queues afterwards. It is possible to confirm all indications by one register access writing the value 0xFFFFFFFF$_H$ to GSTAR.

- Interrupt queue polling:
  The interrupt queues are polled periodically for new vectors. Neither $\overline{\text{INTA}}$ signal nor register GSTAR is evaluated.

The two solutions can be mixed. Queues with high interrupt activity might be serviced by periodic queue polling and other queues on $\overline{\text{INTA}}$ and GSTAR indication only. In this case the ISR only takes care on some GSTAR bits but must confirm all indications to reset $\overline{\text{INTA}}$ indication.

Note that any interrupt event will cause an $\overline{\text{INTA}}$ and GSTAR indication.

To distinguish between old and new interrupt vectors in the circular queues, vectors can be overwritten by software with all zeros. Because the number of new vectors is not known, a queue read pointer can be incremented as long as no new vector is found at this location.

5.2 Central FIFOs Operational Description

The large central transmit and receive FIFOs are a major factor according to the system performance depending on the serial lines and the system bus data rates and latencies. Programmable thresholds and partitioning of the transmit FIFO alow optimized adaption to the needs of any application.

5.2.1 Central FIFO Register Overview

The following table provides an overview about all central FIFO related registers. For detailed register description refer to **Chapter 10**.

Table 16 Central FIFO Control Registers

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0044_H	r/w	CPU	00000000_H	**FIFOCR1**: FIFO Control Register 1
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	31..27, 26..22, 21..17, 15..11	TFSIZE0 TFSIZE1 TFSIZE2 TFSIZE3	0, 0, 0, 0	Transmit FIFO Section Size i (i=0..3) Transmit FIFO Section Size for the corresponding channel i in multiples of 4 DWORDs. *Note: The entire size of all FIFO parts must not exceed 128 DWORDs. If the complete FIFO is assigned to only one channel the maximum size is limited to 4*31=124 DWORDs.* *Note: The minimum FIFO section size for active channels is 4 DWORDs which means a “1” programmed to the respective TFSIZEi bit field.*

Table 16 Central FIFO Control Registers (cont'd)

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0048_H	r/w	CPU	00000000_H	**FIFOCR2**: FIFO Control Register 2
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	31..27, 26..22, 21..17, 15..11	TFR THRESH0, TFR THRESH1, TFR THRESH2, TFR THRESH3,	0, 0, 0, 0	Transmit FIFO Refill Threshold i (i=0..3) Transmit FIFO Refill Threshold for the corresponding channel i in number of DWORDs multiplied by its respective multiplier Mx_i. This threshold controls DMAC operation towards the Host memory. A watermark is calculated by: watermark = TFRTHRESHi*Mx_i+1. As soon as the number of valid data in the transmit FIFO section is less than the watermark, the DMA controller requests for new data from shared memory.
	7,5,3,1	M4_i	0	Multiplier 4
	6,4,2,0	M2_i	0	Multiplier 2

Table 16 Central FIFO Control Registers (cont'd)

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
$004C_H$	r/w	CPU	00000000_H	**FIFOCR3**: FIFO Control Register 3
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	6..0	RF THRESH	0	Receive FIFO Threshold Receive FIFO Threshold in number of DWORDs. A watermark is calculated by: watermark = RFTHRESH*M_x. When more data than specified by this watermark is available in the receive FIFO the DMA controller is requested to transfer data to the channel specific data buffers in host memory until the receive FIFO is empty.
	8	M4	0	Multiplier 4
	7	M2	0	Multiplier 2

Table 16 Central FIFO Control Registers (cont'd)

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
0034_H	r/w	CPU	00000000_H	**FIFOCR4**: FIFO Control Register 4
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	31..24, 23..16, 15..8, 7..0	TFF THRESH0, TFF THRESH1, TFF THRESH2, TFF THRESH3,	0, 0, 0, 0	Transmit FIFO Forward Threshold i (i=0..3) Transmit FIFO Forward Threshold for corresponding channel i in DWORDs. A watermark is calculated by: watermark = TFFTHRESHi As soon as the number of valid data belonging to a new frame in the transmit FIFO is greater than the watermark, the DMAC will provide transmit data to the corresponding SCC. Once having started a frame the DMAC will ignore this threshold providing all available data to the SCC. Threshold operation starts again with the beginning of a new frame. Frames shorter than the threshold will be transferred as soon as a frame end indication is detected by the DMAC. *Note: The maximum allowed Transmit FIFO Forward Threshold is: TFFTHRESHi = (TFSIZEi * 4) - 1* *Note: Programming TFFTHRESHi to zero will disable the threshold causing the DMAC to transfer all data immediately. This may be useful with non packet oriented data e.g. in ASYNC protocol mode.*

5.2.2 Central Transmit FIFO (TFIFO)

The central transmit FIFO can be partitioned in 4 sections with 128 DWORDs in total. Each section size can be programmed according to the needs of the corresponding serial port via register FIFOCR1. Criteria for partitioning are serial line speed (nominal data rate) and type of traffic (bursty or constant). The software has to ensure that the sum of section sizes does not exceed the 128 DWORDs total limit. One channel can consume only 124 DWORDs of the central transmit FIFO.

Two thresholds per TFIFO section are provided to optimize TFIFO operation to the serial side as well as the system interface side.

These thresholds have conflicting requirements:

1. Minimizing transmit data underrun probability in case of PCI bus latencies (especially for high speed ports).
2. Reducing bus utilization by making maximum PCI burst transfers possible for loading of transmit data.

As a naming convention Transmit FIFO always means the Transmit FIFO section of the dedicated channel. All considerations apply to one transmit channel.

Requirement 1) is controlled by the Transmit FIFO forward threshold (register FIFOCR4). Transmit data is transferred to its SCC if one of the following conditions is true:

- A complete packet is stored in the TFIFO (Frame End (FE) indication detected). In this case data transfer to the SCC will start although the frame is smaller than the forward watermark.
- The TFIFO is filled beyond the forward threshold.

These two conditions are checked again after every transfer of a complete frame to the SCC.

Consider a small frame is stored in the transmit FIFO and the beginning of a second frame, but the total amount of data is smaller than the forward threshold (DMAC operation may be delayed by bus latency now). The TFIFO will start transferring data to the SCC because a frame end (FE) indication is detected. After transfer of the complete first frame the above two conditions are checked again. Now there is no further FE indication in the TFIFO and the forward threshold is not exceeded. Thus the TFIFO will not start transferring the second frame until additional data is loaded.

Requirement 2) is controlled by the Transmit FIFO refill threshold (register FIFOCR2). In case of an empty transmit FIFO the DMAC always tries to fill the complete TFIFO with data. If a TFIFO full condition occurs, the DMAC gets stopped. The DMAC starts again if the TFIFO fill level falls below the refill threshold and tries to get new data via the PCI bus until the TFIFO is filled again.

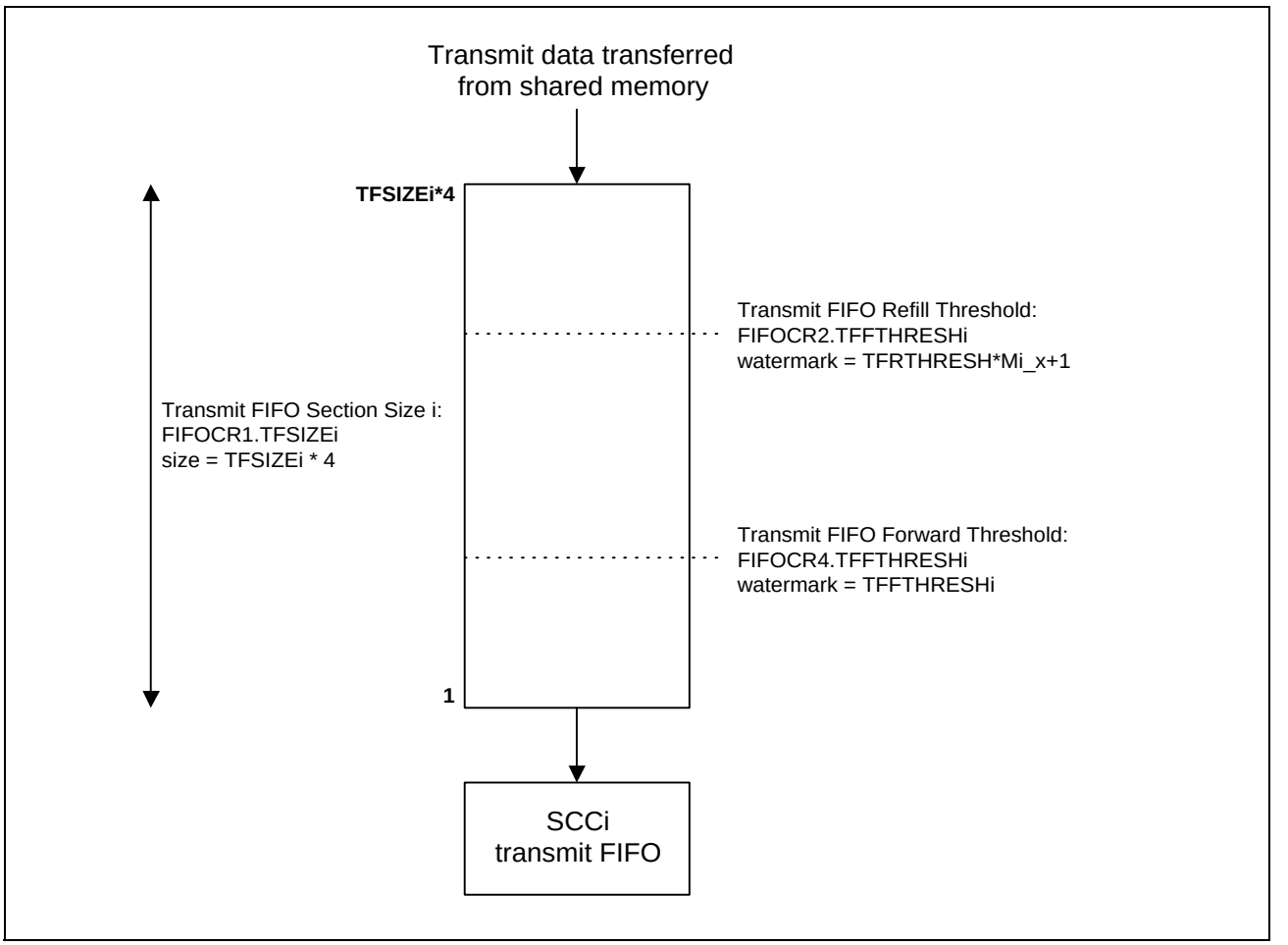


Figure 20 Central Transmit FIFO Section Thresholds

For all operations the burst size is limited to the available TFIFO space or to 15 DWORDs maximum (4 DWORDs in de-multiplexed interface mode).

To guarantee maximum burst length the refill threshold should be programmed such that (TFIFO size - refill watermark) is equal to 15 DWORDs. A lower refill threshold is no guarentee for limited burst size because due to bus latency the available TFIFO space might be greater when transfer starts thus allowing full burst. So 15 DWORDs burst can happen even if the (TFIFO size - refill watermark) < 15 DWORDs.

A refill threshold such that (TFIFO size - refill watermark) is greater than 15 DWORDs (the maximum burst size) does not make sense.

Note: The following condition must be met to avoid deadlock situations:
Refill Watermark > (Forward Watermark + 2 DWORDs)

Consider the following situation:

A short frame is loaded and the beginning of a second frame such that the transmit FIFO is full and DMA operation stops.

After having transferred the first frame to the SCC, the remaining TFIFO fill level is between the two watermarks. Now DMAC operation does NOT start if the refill watermark is not suffused and transfer to the SCC does NOT start due to neither a frame end indication is detected nor the forward watermark is exceeded.

This is a deadlock condition!

Note: The DMAC evaluates the 2 conditions for transfer to SCC with the beginning of each new frame.

Note: The small SCC specific FIFOs always request for data if space is available even if the SCC is in power down condition. If the forward threshold condition is met, data is transferred into the SCC FIFO immediately (6 DWORDs because transfer to the 2 DWORD shadow FIFO does not happen in power-down).

5.2.3 Central Receive FIFO (RFIFO)

The central receive FIFO is shared by all 4 receive channels and dynamically allocated to the SCCs. One central receive FIFO threshold controls data transfer to the host memory.

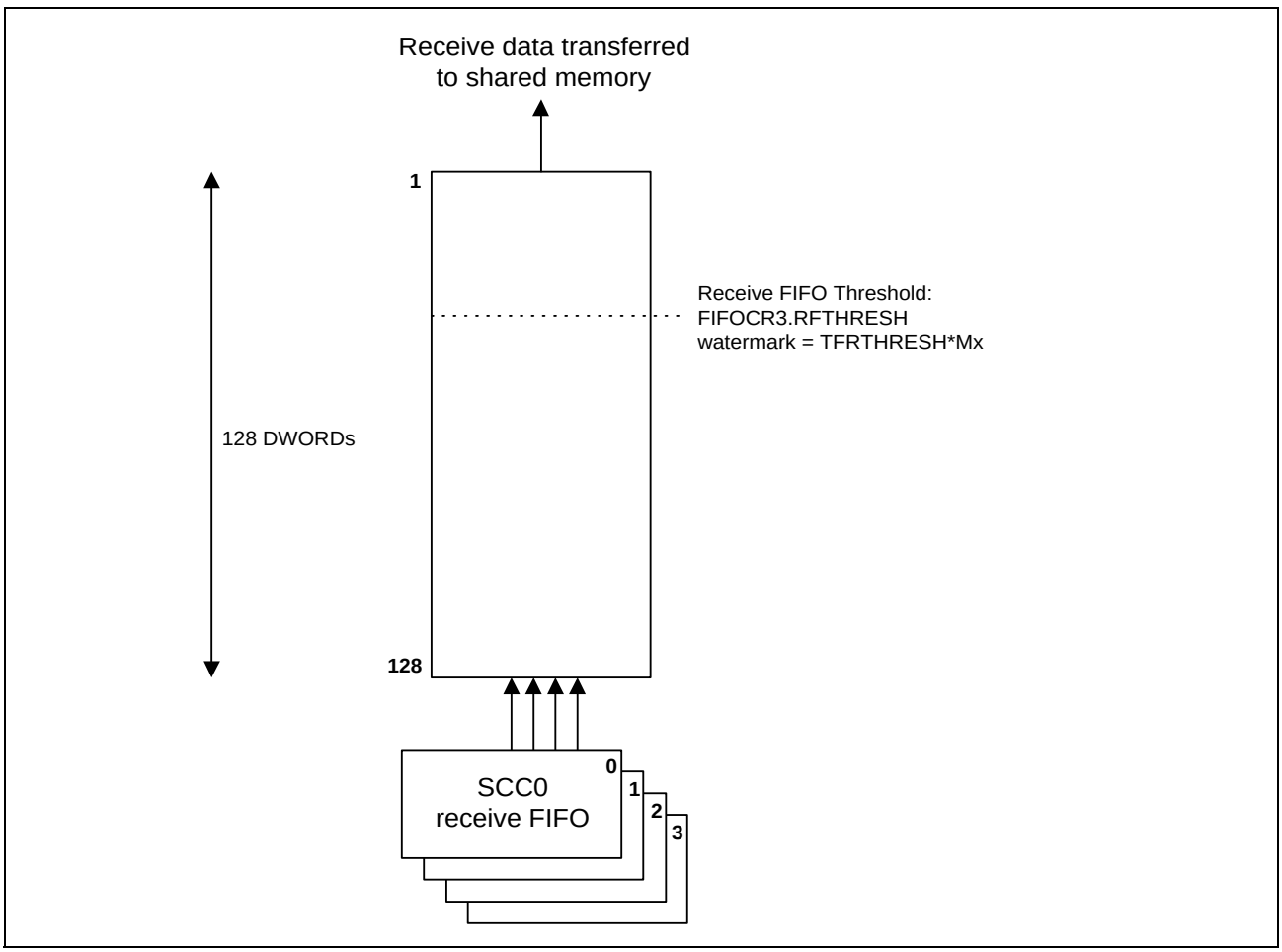


Figure 21 Central Receive FIFO Threshold

Mention, that the SCCs typically forward receive data in portions of 15 DWORDs to the central receive FIFO. These coherent data portions are a precondition for PCI burst transactions.

5.2.4 DMAC Internal Arbitration Scheme

System Interface Arbitration (DMA Arbiter):

Once the DMA controller has been granted PCI bus access, it will attempt to transfer all of its available data in one bus access. However the PCI access time is limited by the PCI bus protocol (bus arbitration and latency timer operation).

There are three priority groups as defined below:

Within each group, the group priority is shared between the channels by the round robbin rules. After each bus access the priority is re-computed for the next access based on the round robbin rules. The DMA arbiter steps through three groups of priority in case of at least one DMA request per group is pending.

Priority groupings:

1. The DMA channel performing the interrupt vector transfer has the highest (first) priority to prevent over-runs and loss of interrupt vectors.
2. The DMA channels performing the data transfer in receive direction have the second priority (receiver group).
3. The DMA channels performing the data transfer in transmit direction have the third priority (transmitter group).

The sub-priority of the DMA channels within the receiver or transmitter group is the same or one channel is treated as ’high priority channel’.

This exclusive priority can be enabled via bit ’SPRI’ in register GMODE whereas the dedicated channel can be selected via bit field ’PCH’ in register GMODE. This setting also effects the internal SCC arbitration.

Internal SCC Arbitration (SCC Arbiter):

Two independent arbiters control the service of the 4 SCC transmit and the 4 SCC receive channels. Each arbiter either works following the round robbin scheme or provides ’high priority’ to one dedicated channel and services the remaining channels in a second priority group using round robbin.

This exclusive priority can be enabled via bit ’SPRI’ in register GMODE whereas the dedicated channel can be selected via bit field ’PCH’ in register GMODE.

The high priority option is useful if one SCC is configured in high speed mode (up to 52 MBit/s) and the others are operating at slow data rates below 10 or 2 MBit/s. In this case data transfer from and to the corresponding SCC FIFOs as well as data transfer from and to host memory is preferred for the high speed transmit/receive channels. In all other cases ’round robbin’ within the described priority groups provide a balanced arbitration solution.

5.2.5 DMAC Performance

The DMAC supports linear bursting of multiple DWORDs for transfers of both data and descriptors:

- On descriptor reads: 3 (Tx) / 3 (Rx) DWORDs at a time.
- On data write/read transactions for full scatter/gather sections up to 15 DWORDs at a time.

The burst size for transmit data is determined by:

- the central transmit FIFO partition and threshold configurations (channel specific)
- the microprocessor interface arbitration and latency

The burst size for receive data is determined by:

- the central receive FIFO threshold configuration
- the SCC receive FIFO threshold
  (Only DWORDs stored in consecutive sub-sections of the central receive FIFO can be transferred to the host memory by one burst transfer. If the SCC receive FIFO threshold is below 15 DWORDs (60 bytes) the consecutive sub-sections in the central receive FIFO might be smaller as well depending on other channels activity. In this case burst transfers of receive data might be limited to the SCC receive threshold value.)
- the microprocessor interface arbitration and latency

In PCI mode up to 15 DWORDs bursts are supported. In de-multiplexed bus interface mode the burst size is limited to 4 DWORDs or burst transactions are suppressed at all.

The DSCC4 DMAC uses PCI (fast) back-to-back transfers to achieve a maximum throughput within one bus arbitration phase.

5.2.6 Little / Big Endian Byte Swap Convention

The DSCC4 operates per default as a little endian device. To support integration into big endian environments the DMAC provides an internal byte swapping mechanism which can be enabled via bit 'ENDIAN' in register GMODE.

The big endian byte swapping applies only to the DATA sections of the receive and transmit descriptor lists in the shared memory.

Data sections might be prepared/evaluated by software using byte pointer operations in the shared memory. The DSCC4 will access these data sections by DWORD read/write transfers. In case of a big endian CPU but little endian DSCC4 mode this will result in wrong byte orders on the serial line and in the receive data sections.

Bad/Good case example:

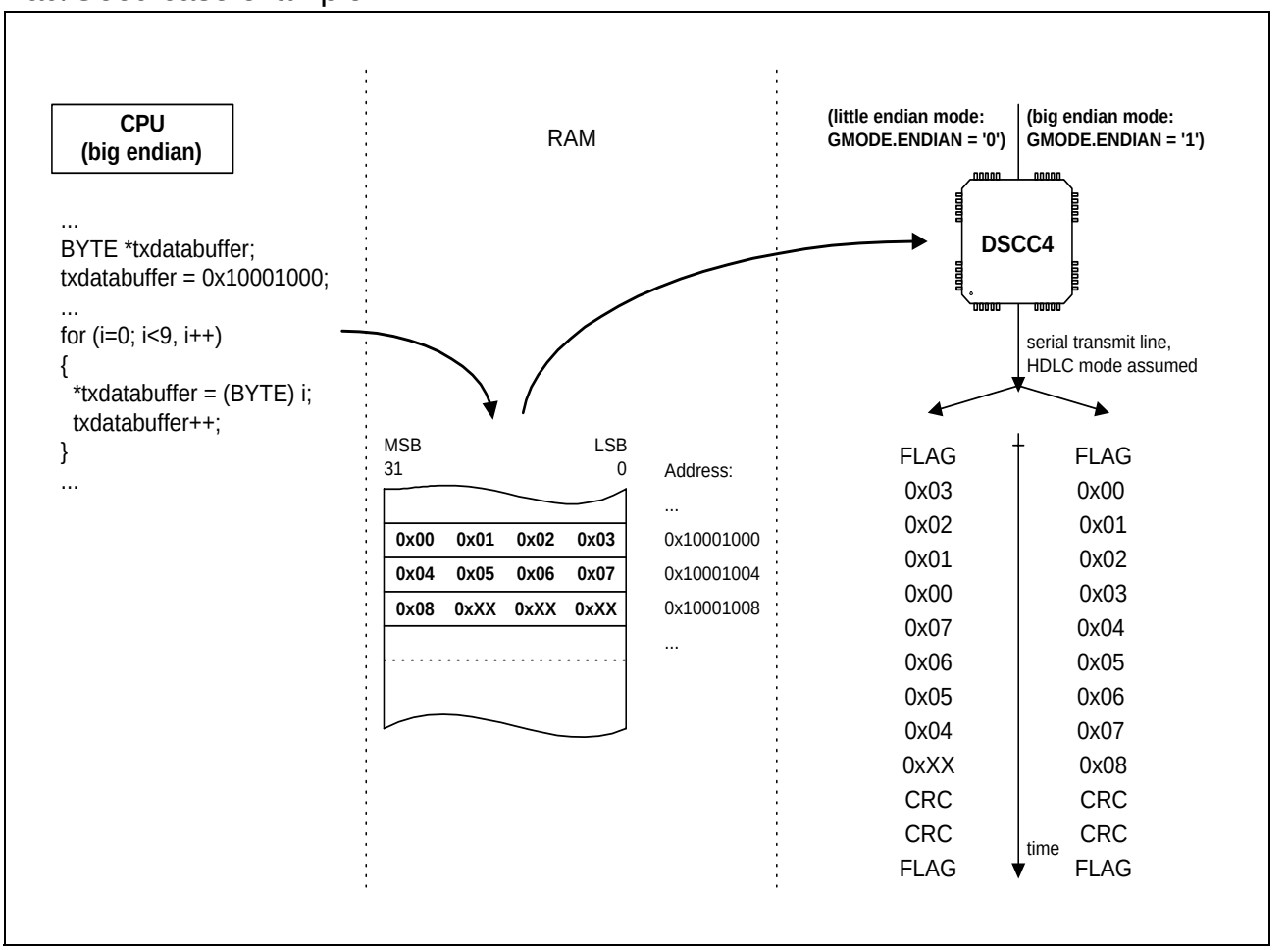


Figure 22 Little/Big Endian Byte Swapping

All other memory structures (descriptors, interrupt vectors) as well as DSCC4 registers are organized DWORD-wise and should be operated by software using 32bit operations only. Therefore no little/big endian ordering mismatches can occur on these structures.

However byte ordering in the local memory, as it appears to the PCI bus view, depends on the local bus (CPU, memory, PCI bridge) realization.

6 Multi Function Port (MFP)

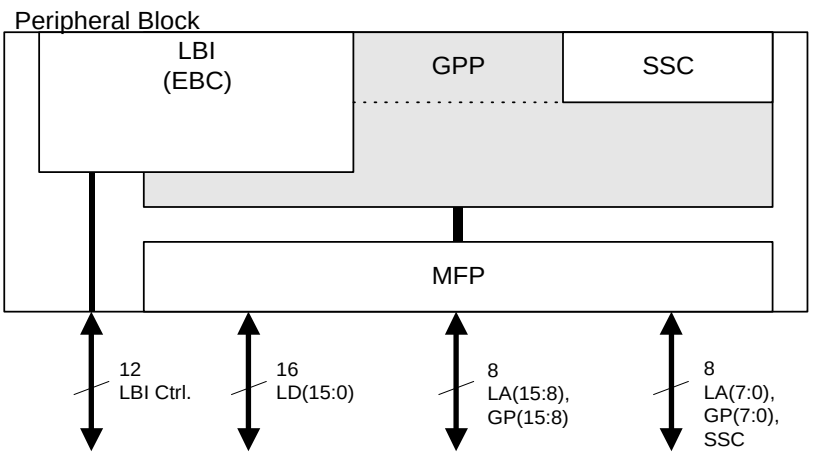
The Multi Function Port consists of a set of I/O signal pins and three internal function blocks sharing these signal pins.

The function blocks are:

- Local Bus Interface (LBI)
- Synchronous Serial Communication Interface (SSC)
- General Purpose I/O Port (GPP)

The MFP is only available in PCI bus operation mode, because in de-multiplexed bus mode, all MFP I/O signals are used for the 32-bit address bus extension.

Various configurations allow simultaneous operation of the three function blocks:



				Configurations:
LBI Ctrl.	LD(15:0)	LA(15:8)	LA(7:0)	LBI only
LBI Ctrl.	LD(15:0)	GP(15:8)	LA(7:0)	LBI + 8 bit GPP
LBI Ctrl.	LD/LA(15:0)	GP(15:8)	SSC	LBI + 8 bit GPP + SSC
LBI Ctrl.	LD/LA(15:0)	GP(15:8)	GP(7:0)	LBI + 16 bit GPP

Figure 23 MFP Configurations Overview

The configuration is selected via bit field ’PERCFG’ in register GMODE. The following table shows all supported ’PERCFG’ bit field options (also refer to **“GMODE: Global Mode Register” on Page 241**):

Table 17 MFP Configuration via GMODE Register, Bit Field 'PERCFG':

PERCFG (2:0)

Peripheral Block Configuration

The peripheral block basically consists of the functions
- Local Bus Interface (LBI)
- General Purpose Port (GPP)
- Synchronous Serial Controller (SSC)

which can be operated in various combinations/configurations.
Bit field 'PERCFG' selects the peripheral configuration and switches the multiplexed signal pins accordingly:

PCI Interface Mode (DEMUX pin connected to V_{SS}):

PERCFG	Signal Pin Groups		
(2:0)	109, 108 101..96	119..112	143..135 128..123
'000'	LA(15..8)	LA(7..0)	LD(15..0)
'001'	*Reserved*. Do not use.		
'010'	GP(15..8)	LA(7..0)	LD(15..0)
'011'	GP(15..8)	SSC	LAD(15..0)
'100'	GP(15..8)	GP(7..0)	LAD(15..0)
'101', '110', '111'	*Reserved*. Do not use.		

DEMUX Interface Mode (DEMUX pin connected to V_{DD3}):

Bit field 'PERCFG' is not valid. All 32 multiplexed signals are used as DEMUX address bus A(31:0):

PERCFG	Signal Pin Groups		
(2:0)	109, 108 101..96	119..112	143..135 128..123
'xxx'	A(15..8)	A(7..0)	A(31..0)

The following sub-chapters describe the three peripheral function blocks LBI, SSC and GPP:

6.1 Local Bus Interface (LBI)

The DSCC4 provides capability for the PCI host system to access peripherals connected to the Local Bus Interface (LBI).

Note: The LBI is only available when the DSCC4 is configured for the PCI bus interface mode. When in de-multiplexed bus interface mode, the LBI address and data pins interface to the host system address bus.

Table 18 LBI Peripheral Transaction Options

Peripheral Type	Corresponding PCI Transaction	Read	Write
Non-Intelligent	Slave	PCI Retry operation	PCI posted write operation

Overview of Transactions:

Standard PCI Slave transactions are used when the PCI host system communicates with non-intelligent LBI peripherals (single address read/write operations).

For reads, a PCI Retry sequence of operations is performed, in which the DSCC4 will immediately terminate the PCI transaction (and request a retry) until it terminates the transaction to the LBI. The DSCC4 uses the retry procedure because the time to complete the data phase will require more than the maximum allowed 16 PCI clocks (from the assertion of $\overline{\text{FRAME}}$ to the completion of the first data phase). Data transfer will be successfully completed within a PCI retry cycle. The number of necessary PCI retry cycles depend on PCI arbitration behavior and the time it needs to terminate the transaction on the local bus; PCI $\overline{\text{TRDY}}$ wait states will not be added for the sequential retry read cycles unless the LBI arbitration time is excessive.

For write transactions, the DSCC4 will store a single data DWORD and then immediately terminate the PCI transaction successfully. It will then arbitrate the local bus and perform the write transaction after being granted, depending on the selected number of wait states and $\overline{\text{LRDY}}$ bus control signal.

Thus write accesses to LBI are performed as ‘posted write’ transactions from the PCI view. A consecutive write transaction results in PCI retry cycles in the case that the preceding write transaction is not yet finished on the local bus.

Note: Note that the DSCC4 performs single word PCI Slave read or write transactions only; Slave burst transactions to LBI are not supported.

6.1.1 LBI Bus Modes

6.1.1.1 LBI External Bus Controller (EBC)

The External Bus Controller (EBC) provides a flexible bus interface to connect a wide range of peripherals. In normal mode this interface is a bus master (and default bus owner) and drives peripheral devices. It provides the ability to select busses of different configuration: 8 bit multiplexed/de-multiplexed or 16 bit multiplexed/de-multiplexed. The configurable pins of address signals/general purpose signals provide alternate functionality to support the LBI pins with additional I/O signals.

The EBC also supports bus arbitration supporting other bus masters connected to the local bus. In this case, the DSCC4 can be configured as arbitration master (default bus owner) or arbitration slave

The function of the EBC is controlled via the global mode register GMODE and the LBI Configuration register LCONF. It specifies the external bus cycles in terms of address (multiplexed/de-multiplexed), data (16-bit/8-bit) and control signal length (wait states).

6.1.1.2 Multiplexed Local Bus Modes

In the 16-bit multiplexed bus mode both the address and data lines use the pins LD(15:0). The address is time-multiplexed with the data and has to be latched externally. The width of the required latch depends on the selected data bus width, i.e. an 8-bit data bus requires an 8-bit latch (the address bits LD15...LD8 on the LBI port do not change, while on LD7...LD0 address and data signals are multiplexed), a 16-bit data bus requires a 16-bit latch (the least significant address line LA0 is not relevant for word accesses). In de-multiplexed mode, the address lines are permanently output on pins LA(15:0) and do not require latches.

The EBC initiates an external access by generating the Address Latch Enable signal (LALE) and then placing an address on the bus. The falling edge of LALE triggers an external latch to capture the address. After a period of time in which the address must have been latched externally, the address is removed from the bus. The EBC now activates the respective command signal ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LRDY}}$) and data is driven onto the bus either by the EBC (for write cycles) or by the external memory/peripheral (for read cycles). After a period of time, which is determined by the access time of the memory/ peripheral, data becomes valid.

Read cycles: Input data is latched and the command signal is now deactivated. This causes the accessed device to remove its data from the bus which is then tri-stated again.

Write cycles: The command signal is now deactivated. The data remain valid on the bus until the next external bus cycle is started.

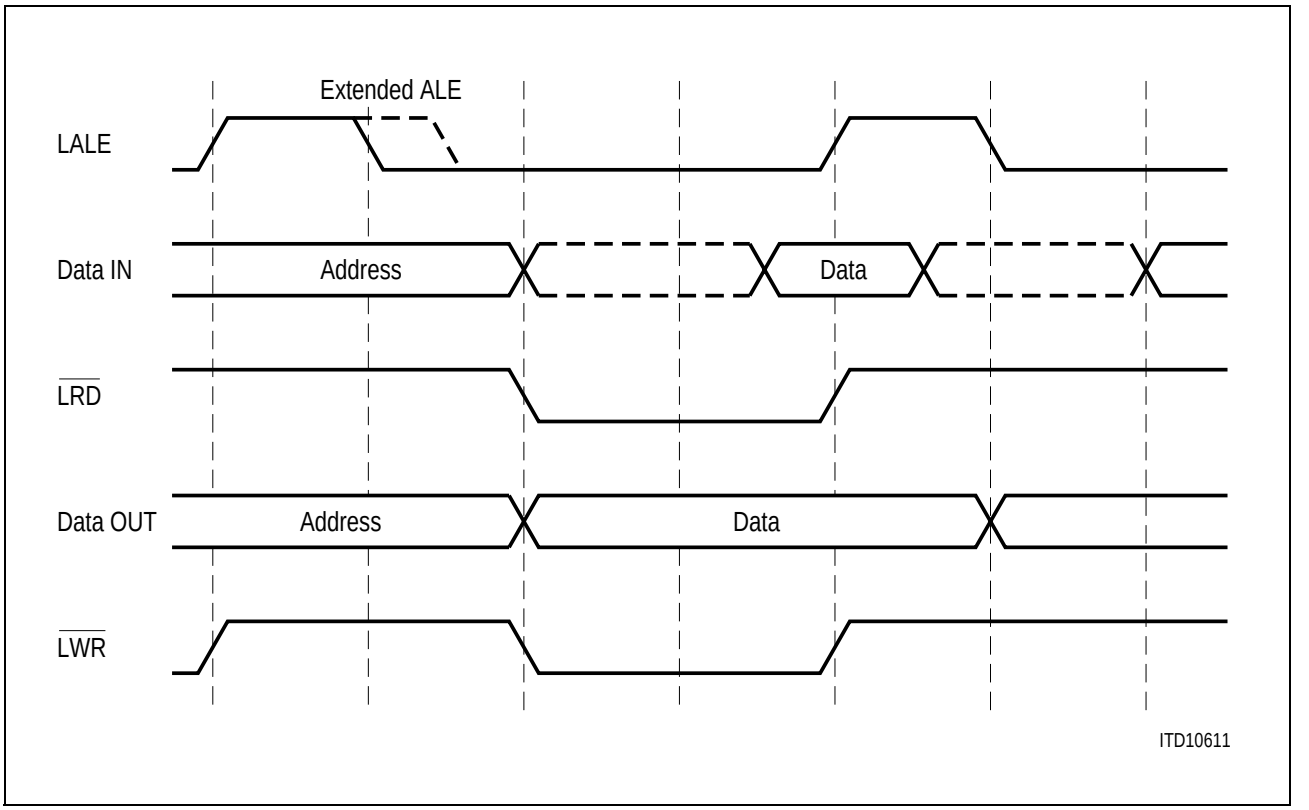


Figure 24 Multiplexed Bus Cycle

6.1.1.3 De-multiplexed Local Bus Modes

The de-multiplexed bus modes use the LBI port pins LA(15:0) for the 16-bit address and the LBI port pins LD(15:0) for 8/16-bit data. The EBC initiates an external access by placing an address on the address bus. The EBC then activates the respective command signal ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LBHE}}$). Data is driven onto the data bus either by the EBC (for write cycles) or by the external memory/peripheral (for read cycles). After a period of time, which is determined by the access time of the memory/peripheral, data become valid.

Read cycles: Input data is latched and the command signal is now deactivated. This causes the accessed device to remove its data from the data bus which is then tri-stated again.

Write cycles: The command signal is now deactivated. If a subsequent external bus cycle is required, the EBC places the respective address on the address bus. The data remain valid on the bus until the next external bus cycle is started.

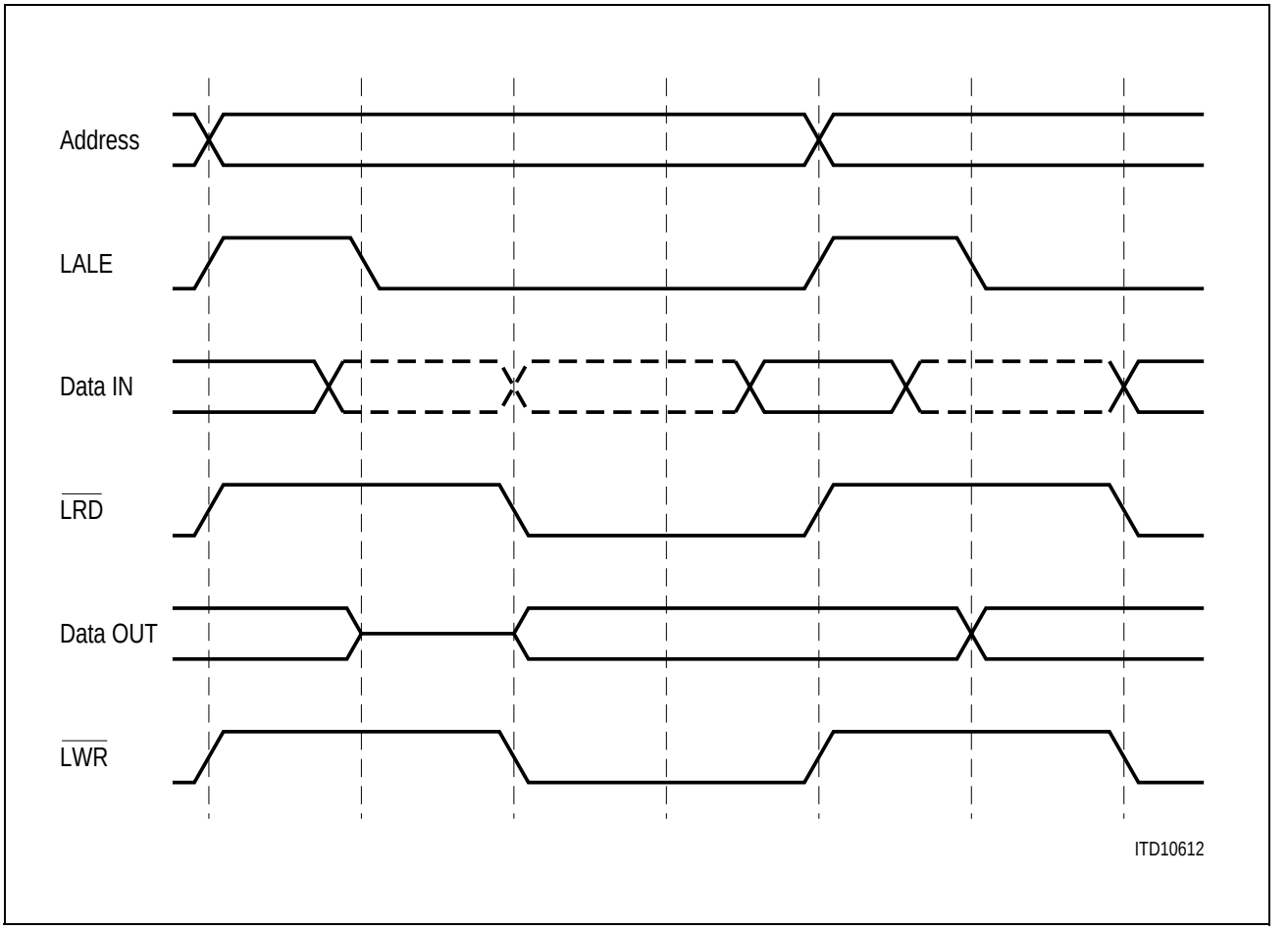


Figure 25 De-multiplexed Bus Cycle

6.1.1.4 Programmable Bus Characteristics

External Data Bus Width:

The EBC can operate on 8-bit or 16-bit wide external memory/peripherals. A 16-bit data bus uses the LBI port pins LD(15:0), while an 8-bit data bus only uses LD(7:0). This saves bus transceivers, bus routing and memory costs at the expense of transfer time. The EBC can control byte accesses on a 16-bit data bus.

Byte accesses on a 16-bit data bus require that the upper and lower half of the memory can be accessed individually. In this case the upper byte is selected with the $\overline{\text{LBHE}}$ signal, while the lower byte is selected with the LA0 signal. The two bytes of the memory can therefore be enabled independent from each other (or together when accessing words).

Devices such as the ESCC2 also provide a $\overline{\text{BHE}}$ input and hence allow byte accesses in 16-bit bus mode.

When reading bytes from an external 16-bit device, whole words may be read and the EBC automatically selects the byte to be input and discards the other.

However, it must be taken care on devices, which change their state when being read, e.g. FIFOs, interrupt status registers, etc. In this case individual bytes should be selected using $\overline{\text{BHE}}$ and LA0.

Switching between the Bus Modes:

The EBC bus type can be switched dynamically by software between different read/write bus transactions. However, the connected peripherals must support the selected bus type(s) (multiplexed mode or de-multiplexed mode).

Arbitration master/slave mode according the local bus handled dynamically by the device itself.

Programmable bus timing characteristics:

Important timing characteristics of the external bus interface have been made user programmable to adapt to the needs of a wide range of different external bus and memory configurations with different types of memories and/or peripherals.

The following parameters of an external bus cycle are programmable:

- **Memory Cycle Time** (extendable with 1...15 wait states) defines the minimum access time by the width of $\overline{\text{LRD}}$/$\overline{\text{LWR}}$ strobe signals;
- **$\overline{\text{LRDY}}$ Control Signal** extends the transcation controlled by the target peripheral.

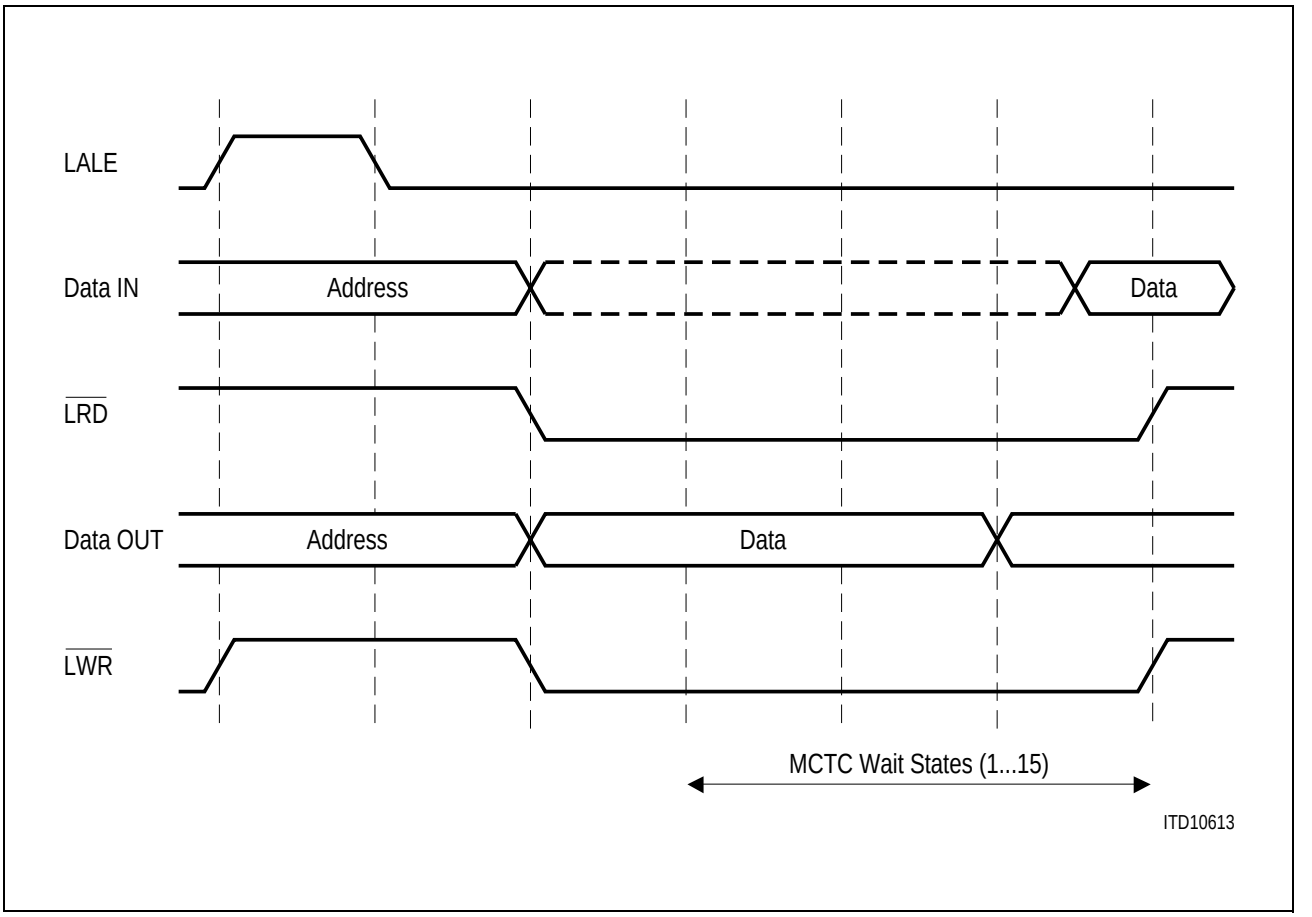


Figure 26 Memory Cycle Time

The external bus cycles of the EBC can be extended for a memory or peripheral, which cannot keep pace with the EBC's maximum speed, by introducing wait states during the access (see figure above).

The minimum $\overline{\text{LRD}}$/$\overline{\text{LWR}}$ strobe active length is 2 LBI clock cycles (with zero MCTC wait states).

The memory cycle time wait states can be programmed in increments of one EBC system clock (LCLKO) within a range from 0 to 15 (default value after reset) via the 'MCTC' bit field in the LBI Configuration register LCONF. (15-<MCTC>) waitstates will be inserted.

6.1.1.5 Ready Signal Controlled Bus Cycles

For situations, where the programmable constant number of 15 wait states is not enough, or where the response (access) time of a peripheral is not constant, the DSCC4 EBC interface provides external bus cycles that are terminated via a $\overline{\text{LRDY}}$ input signal. In this case the EBC first inserts a programmable number of waitstates (0...7) and then monitors the $\overline{\text{LRDY}}$ line to determine the actual end of the current bus cycle. The external device drives $\overline{\text{LRDY}}$ low in order to indicate that data have been latched (write cycle) or are available (read cycle).

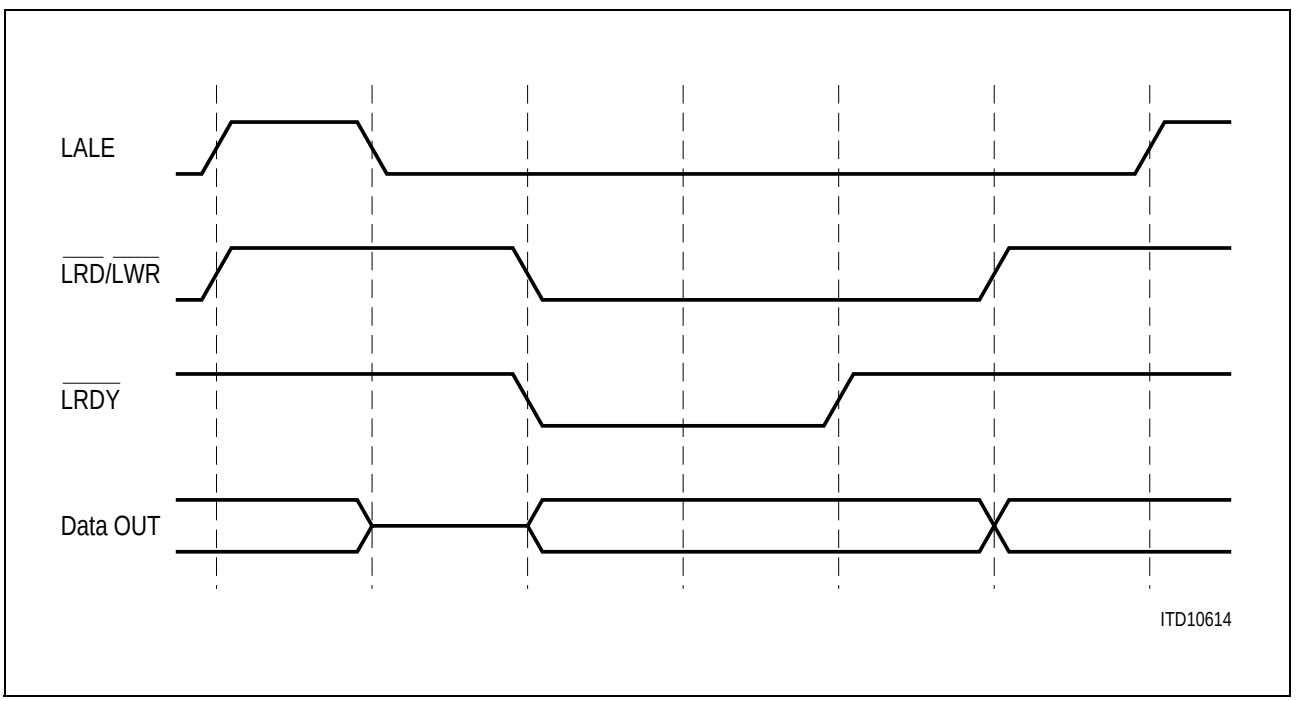


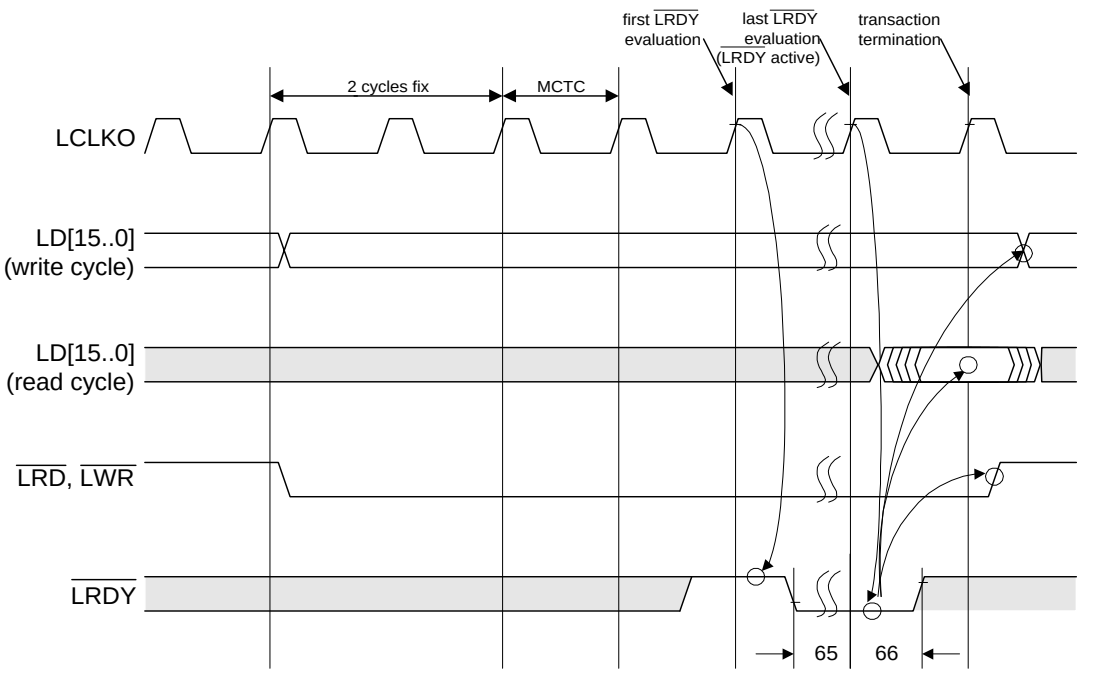
Figure 27 $\overline{\text{LRDY}}$ Controlled Bus Cycles

The $\overline{\text{LRDY}}$ function is enabled via bit 'RDYEN' in the LBI Configuration register LCONF. The $\overline{\text{LRDY}}$ signal is always synchronized at the input port pin. An asynchronous $\overline{\text{LRDY}}$ signal that has been activated by an external device may be deactivated in response to the trailing (rising) edge of the respective command ($\overline{\text{LRD}}$ or $\overline{\text{LWR}}$).

Combining the $\overline{\text{LRDY}}$ function with predefined waitstates is advantageous in two cases. Memory components with a fixed access time and peripherals operating with $\overline{\text{LRDY}}$ may be grouped into the same address window. The (external) wait states control logic in this case would activate $\overline{\text{LRDY}}$ either upon the memory's chip select or with the peripheral's $\overline{\text{LRDY}}$ output. After the predefined number of waitstates the EBC will check its $\overline{\text{LRDY}}$ line to determine the end of the bus cycle. For a memory access it will be low already, for a peripheral access it may be delayed. As memories tend to be faster than peripherals, there should be no impact on system performance.

When using the $\overline{\text{LRDY}}$ function with 'normally-ready' peripherals, it may lead to erroneous bus cycles, if the $\overline{\text{LRDY}}$ line is sampled too early. These peripherals pull their $\overline{\text{LRDY}}$ output low, while they are idle. When they are accessed, they deactivate $\overline{\text{LRDY}}$

until the bus cycle is complete, then drive it low again. By inserting predefined waitstates the first $\overline{\text{LRDY}}$ sample point can be shifted to a time, where the peripheral has safely controlled the $\overline{\text{LRDY}}$ line (e.g. after 2 waitstates, see **Figure 27** and **Figure 28**).



Notes:
- MCTC wait state configuration is assumed to one cycle in this figure
- LRDY is evaluated the first time with the clock cycle following the MCTC related wait states
- Transaction is terminated one clock cycle after detecting LRDY active
- LRDY is evaluated only if LRDY-control is enabled

Figure 28 $\overline{\text{LRDY}}$ Timing

6.1.1.6 LBI (EBC) Configuration

The properties of a bus cycle usage of signal $\overline{\text{LRDY}}$, external bus mode and waitstates are controlled by LBI Configuration register LCONF and global mode register GMODE. This allows the use of memory components or peripherals with different interfaces within the same system, while optimizing accesses to each of them.

LCONF is described in **“LBI Registers Description” on Page 351**.

EBC Idle State:

Upon reset the MFP comes up in 16-bit de-multiplexed bus LBI mode. The EBC can then be programmed to be arbitration master or slave by software.

When the EBC bus interface is enabled in arbitration master mode, but no external access is currently executed, the EBC is idle. During this idle state the external interface behaves in the following way:

Data port LD(15:0) switches in high impedance state (floating);

Address port LA(15:0) drives the last used address value;

$\overline{\text{LRD}}$/$\overline{\text{LWR}}$ signals remain inactive (high).

6.1.2 LBI Bus Arbitration

In high performance systems it may be efficient to share external resources like memory banks or peripheral devices among more than one bus controller. The LBI's EBC block supports this approach with the possibility to arbitrate the access to its external bus, i.e. to the external devices.

This bus arbitration allows an external master to request the EBC's bus via the $\overline{\text{LHOLD}}$ input signal. The EBC acknowledges this request via the $\overline{\text{LHLDA}}$ output signal and will float its bus signals in this case. The new master may now access the peripheral devices or memory banks via the same interface lines as the EBC. During this time the DSCC4 can keep on executing internal processes, as long as it does not need access to the external bus.

When the EBC needs access to its external bus while it is occupied by another bus master, the bus is requested via the $\overline{\text{LBREQ}}$ output signal.

The external bus arbitration is enabled by setting bit 'HLDEN' in the LBI Configuration register LCONF to '1'. This bit is allowed to be cleared by software during the execution of program sequences, where the external resources are required but cannot be shared with other bus masters. In this case the EBC will not answer to $\overline{\text{LHOLD}}$ requests from other external masters.

The pins $\overline{\text{LHOLD}}$, $\overline{\text{LHLDA}}$ and $\overline{\text{LBREQ}}$ keep their function (bus arbitration) even after the arbitration mechanism has been switched off by clearing bit 'HLDEN'.
All three pins are used for bus arbitration after bit 'HLDEN' was set once.

Entering the Hold State:

Access to the EBC's external bus is requested by driving its $\overline{\text{LHOLD}}$ input low. After synchronizing this signal the EBC will complete a current external bus cycle (if any is active), release the external bus and grant access to it by driving the $\overline{\text{LHLDA}}$ output low. During hold state, the address bus, data bus and signals LALE, $\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LBHE}}$ are tri-stated.

Should the DSCC4 require access to its external bus during hold mode, it activates its bus request output $\overline{\text{LBREQ}}$ to notify the arbitration circuitry. $\overline{\text{LBREQ}}$ is activated only during hold mode. It will be inactive during normal operation.

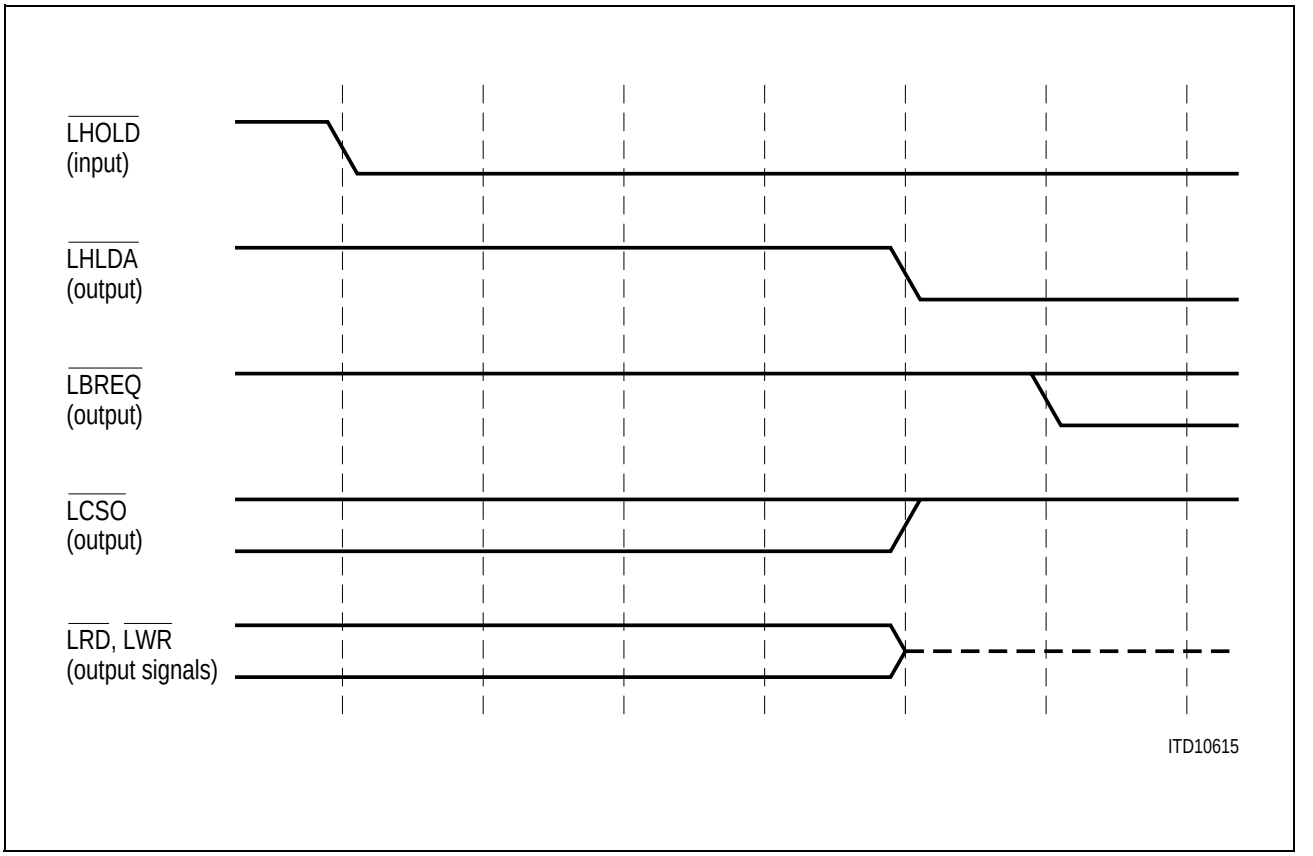


Figure 29 External Bus Arbitration (Releasing the Bus)

Note: The DSCC4 will complete the currently running bus cycle before granting bus access as indicated by the broken lines. This may delay hold acknowledge compared to this figure.
The figure above shows the first possibility for LBREQ to become active.

Exiting the Hold State:

The external bus master returns the access rights to the DSCC4 EBC by driving the LHOLD input high. After synchronizing this signal the EBC will drive the LHLDA output high, actively drive the control signals and resume executing external bus cycles if required.

Depending on the arbitration logic, the external bus can be returned to the EBC under two circumstances:

- The external master does not require access to the shared resources and gives up its own access rights, or
- The DSCC4 EBC needs access to the shared resources and demands this by activating its LBREQ output. The arbitration logic may then deactivate the other master's LHLDA and so free the external bus for the EBC, depending on the priority of the different masters.

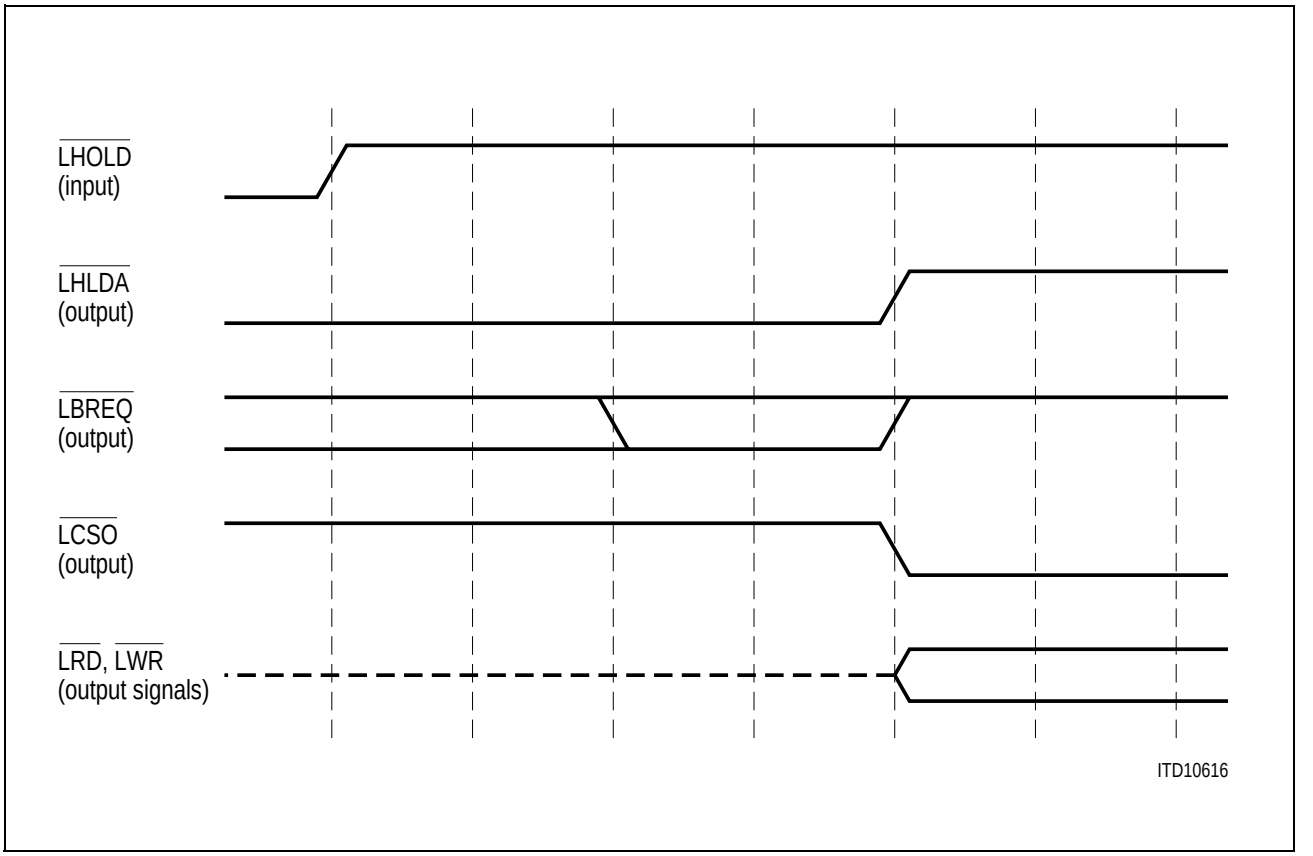


Figure 30 External Bus Arbitration (Regaining the Bus)

The falling LBREQ edge shows the last chance for LBREQ to trigger the indicated regain-sequence. Even if LBREQ is activated earlier the regain-sequence is initiated by LHOLD going high. LBREQ and LHOLD are connected via an external arbitration circuitry. Please note that LHOLD may also be deactivated without the EBC requesting the bus.

Figure 31 below shows the correct connection of the bus arbitration signals between the master and the slave. In order to provide correct levels during initialization of the master and the slave, two external pull-up devices are required. One is connected to the master's LHOLD input, the other to the slave's LHLDA input.

Note: For compatibility reasons with existing applications, these pull-ups can not be integrated into the chip.

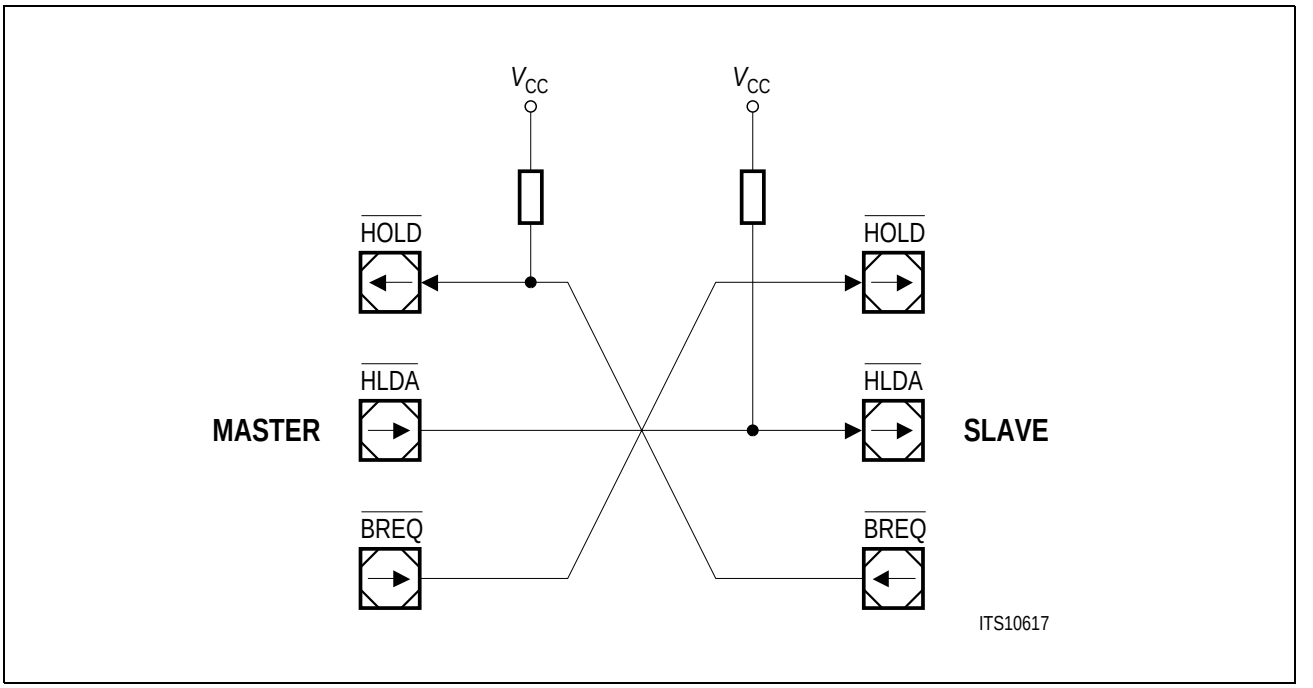


Figure 31 Connection of the Master and Slave Bus Arbitration Signals

Bus Arbitration Master Initialization:

After reset, the master is normally starting execution out of external memory. During reset, the default is the arbitration slave mode. The master arbitration mode must first be selected by setting bit LCONF.ABM=’1’. During the initialization, the ’HLDEN’ bit in register LCONF must be set. Since the $\overline{\text{LHOLD}}$ pin is hold high through the external pull-up, no hold requests can occur even when the slave is not initialized yet.

Note that the HLDEN bit of the master can be reset during normal operation to force the master to ignore hold requests from the slave until HLDEN is set again. However, the pins $\overline{\text{LHOLD}}$, $\overline{\text{LHLDA}}$ and $\overline{\text{LBREQ}}$ are still reserved for the bus arbitration. This is intended to have the option to disable certain critical processes against interruption through hold requests:

Bus Arbitration Slave Initialization

The slave must start using internal resources only after reset. During reset, the default is the slave mode. This is also done by setting bit LCONF.ABM=’0’. This enables the slave mode of the bus arbitration signals. After this, the ’HLDEN’ bit in register LCONF must be set.

Note: After setting the slave’s HLDEN bit, the $\overline{\text{LBREQ}}$ output of the slave might be activated to low for a period of 2 LBI clock periods. If the master does not recognize this hold request (it depends on the master’s transition detection time-slot, whether this short pulse is detected), this pulse has no effect. If the master recognizes this pulse, it might go into hold mode for one cycle. The exact timing in this case will be defined later.

Note: The effect of resetting bit ’HLDEN’ in slave mode (whether the slave is in hold or in normal mode) will be defined later. It is recommended to not reset the slave’s ’HLDEN’ bit after initialization.

Operation of the Master/Slave Bus Arbitration:

The figure below shows the sequence of the bus arbitration signals in a master/slave system. The start-up condition is that the master is in normal mode and operating on the external bus, while the slave is in hold mode, operating from internal memory; the slave’s bus interface is tristated. The marked time points in the diagram are explained in detail in the following.

1) The slave detects that it has to perform an external bus access. It activates $\overline{\text{LBREQ}}$ to low, which issues a hold request from the master.

2) The master activates $\overline{\text{LHLDA}}$ after releasing the bus. This initiates the slave’s exit from hold sequence.

3a) When the master detects that it also has to perform external bus accesses, it activates $\overline{\text{LBREQ}}$ to low. The earliest time for the master to activate $\overline{\text{LBREQ}}$ is one LBI clock after the activation of the master’s $\overline{\text{LHLDA}}$ signal. However, the slave will ignore this signal until it has completely taken over control of the external bus. In this way, it is assured that the slave will at least perform one complete external bus access.

3b) If the master can operate from internal memory while it is in hold mode, it leaves the $\overline{\text{LBREQ}}$ signal high until it detects that an external bus access has to be performed. The slave therefore can stay on the bus as long as the master does not request the bus again.

4) When the master has requested the bus again through activation of its $\overline{\text{LBREQ}}$ signal, the slave will complete the current access and go into hold mode again. After completely tristateing its bus interface, the slave deactivates its $\overline{\text{LBREQ}}$ signal, thus releasing the master out of hold mode.

5) The master has terminated its hold mode and deactivates its $\overline{\text{LHLDA}}$ signal again. Now the master again controls the external bus again.

6) The master deactivates its $\overline{\text{LBREQ}}$ signal again one LBI clock after deactivation of $\overline{\text{LHLDA}}$. From now on (and not earlier), the slave can generate a new hold request from the master. With this procedure it is assured that the master can perform at least one complete bus cycle before requested to go into hold mode again by the slave.

Also shown in the figure below is the sequence of the bus control between the master and the slave.

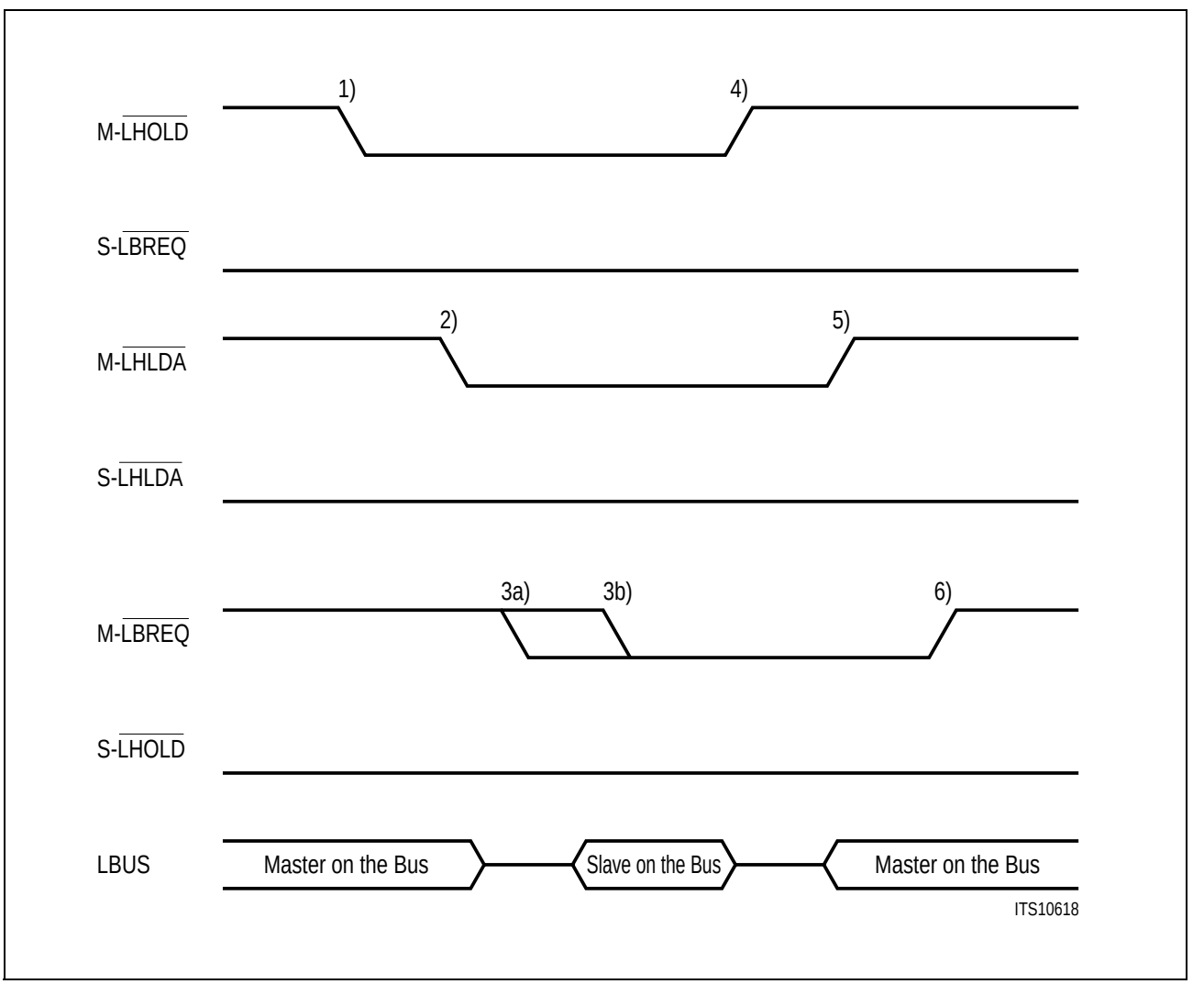


Figure 32 Bus Arbitration Sequence

6.1.3 PCI to Local Bus Bridge Operation

The local bus can work in 8/16-bit multiplexed/de-multiplexed mode. This 8/16-bit organized address space can be mapped into the host memory space using the base address register BAR2 in the PCI Configuration Space which is initialized as part of the device configuration. Other configuration parameters define the clock speed of the local bus, and the number of wait states on local bus transactions.

PCI accesses to this mapped address range result in the assertion of the chip select output $\overline{\text{LCSO}}$ and the corresponding write or read transaction is performed on the local bus.

Register Write to Peripherals:

A PCI write within the local bus address space causes the address and data to be transferred to the peripherals on the local bus.

The DSCC4 will store a single data DWORD (with correct byte enable information) and then immediately terminate the PCI transaction successfully (posted write). The write transaction on the local bus is performed and terminated depending on the selected number of wait states and the $\overline{\text{LRDY}}$ bus control signal.

The PCI 32-bit addresses are automatically modified to appropriate 16 or 8-bit local bus addresses.

Thus write accesses to LBI are performed as ‘posted write’ transactions from the PCI view. A consecutive write transaction results in PCI retry cycles in the case that the preceding write transaction is not yet finished on the local bus.

With this approach, consecutive PCI writes are possible to the local bus address range.

Register Read from Peripherals:

The local bus address space is mapped into the shared memory space, and hence a read operation is similar to a read from memory or any memory mapped register within the PCI address space.

A PCI Retry sequence of operations is performed, in which the DSCC4 will immediately terminate the PCI transaction (and request a retry) until it terminates the transaction to the LBI. The DSCC4 uses the retry procedure because the time to complete the data phase will require more than the maximum allowed 16 PCI clocks (from the assertion of $\overline{\text{FRAME}}$ to the completion of the first data phase). Data transfer will be successfully completed within a PCI retry cycle. The number of necessary PCI retry cycles depend on PCI arbitration behavior and the time it needs to terminate the transaction on the local bus.

Within the local bus, the PCI read address is physically mapped into the 8/16-bit address space of the local bus, and the read cycle is performed to the peripheral. A 8/16 bit data read takes place at the selected local bus speed, and the 8/16 bit data is then passed on to the PCI cycle with the correct number of $\overline{\text{C/BE}}$ (byte enable) bits set.

6.1.4 LBI Interrupt Generation

The LBI block generates interrupts for transitions on input signal LINTI. (Refer to section **“LBI Registers Description” on Page 351**.)

Thus local bus peripherals can generate interrupt indications to the PCI host CPU.

All interrupt events result in an LBI interrupt vector which is transferred into the peripheral interrupt queue (refer to section **“LBI Interrupt Vector” on Page 397**).

6.2 Synchronous Serial Control (SSC) Interface

6.2.1 SSC Functional Description

6.2.1.1 Overview

The Synchronous Serial Control (SSC) Interface provides flexible high-speed serial communication between the DSCC4 and other microcontrollers or external peripherals.

The SSC supports full-duplex and half-duplex synchronous communication up to 8.25 MBaud (@ 33 MHz bus clock). The serial clock signal can be generated by the SSC itself (master mode) or be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible, or Microwire compatible devices. Transmission and reception of data is double-buffered. A 16-bit baud rate generator provides the SSC with a separate internally derived serial clock signal.

The high-speed synchronous serial interface can be configured very flexibly, thus it can be used with other synchronous serial interfaces (e.g., the ASC0 in synchronous mode), serve for master/slave or multimaster interconnections or operate compatible with the popular SPI interface. It allows communicating with shift registers (IO expansion), peripherals (e.g. EEPROMs) or other controllers (networking).

Data is transmitted or received on the pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is an ’output’ or ’input’ on pin MSCLK.

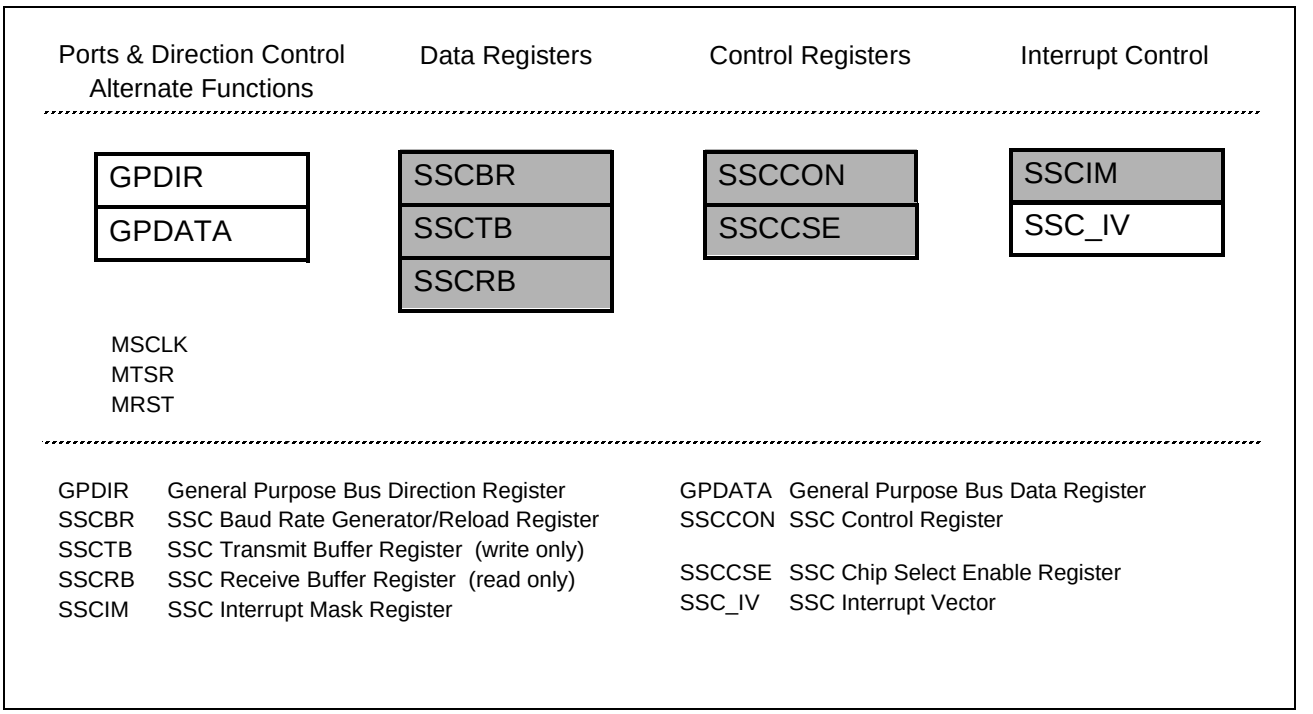


Figure 33 Registers and Port Pins associated with the SSC

If the MFP is programmed for SSC operation (bit field PERCFG='011' in register GMODE), the lower 8 signal pins of the general purpose port (GPP) provide the SSC specific signals whereas the upper 8 signal pins are still available for general purposes. Nevertheless for SSC operation all GPP pins must be programmed for the appropriate direction via register GPDIR.

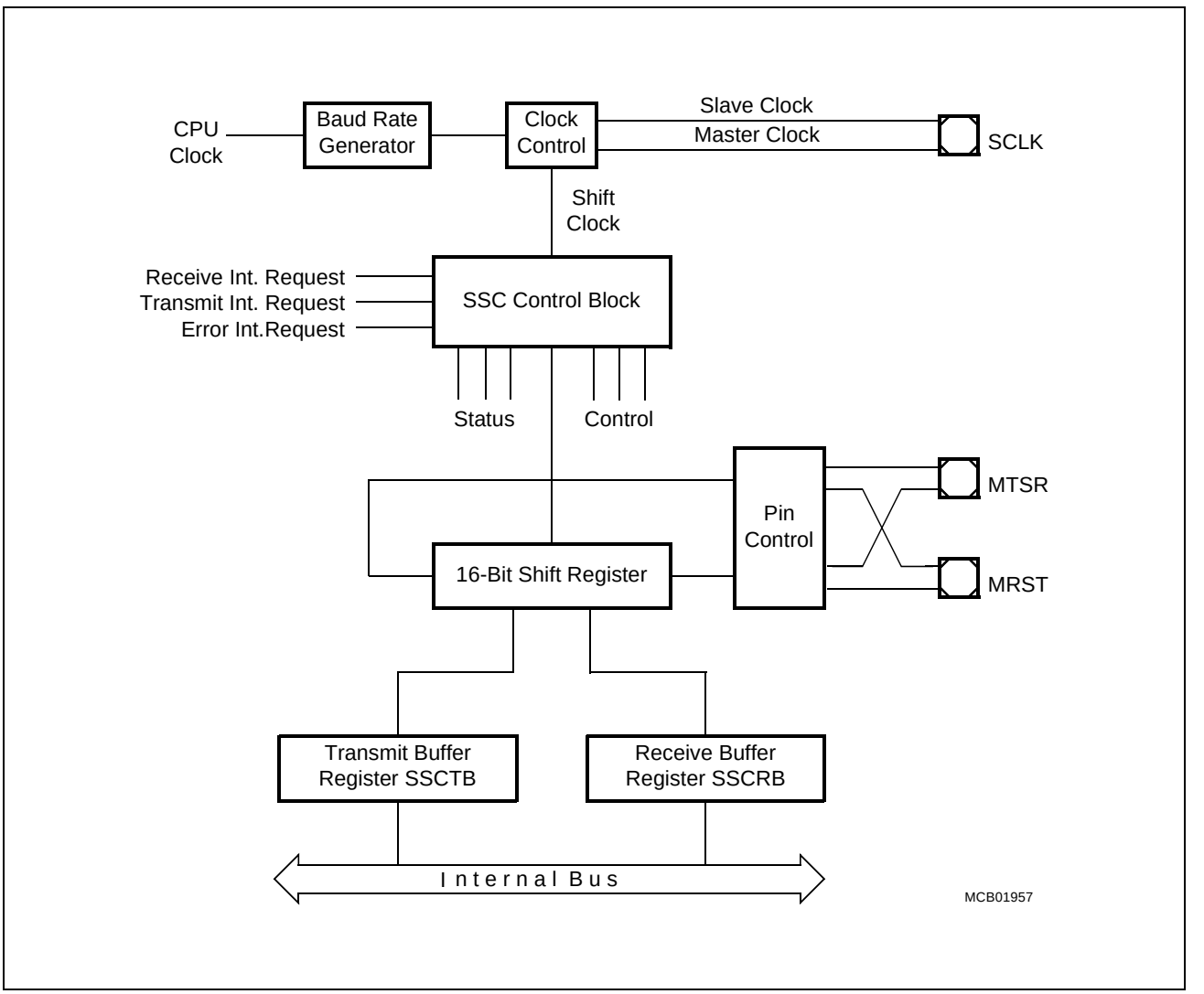


Figure 34 Synchronous Serial Channel SSC Block Diagram

The operating mode of the serial channel SSC is controlled by its bit-addressable control register SSCCON. This register serves for two purposes:

- during programming (SSC disabled by SSCEN=’0’) it provides access to a set of control bits;
- during operation (SSC enabled by SSCEN=’1’) it provides access to a set of status flags.

A detailed control register description for each of the two modes is provided in **“SCC Registers Description” on Page 274**.

The shift register of the SSC is connected to both the transmit pin and the receive pin via the pin control logic (see block diagram). Transmission and reception of serial data is synchronized and takes place at the same time, i.e. the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer SSCTB. It is moved to the shift register as soon as this is empty. An SSC-master (SSCMS=’1’) immediately starts transmitting, while an SSC-slave (SSCMS=’0’) waits for an active shift clock.

When the transfer starts, the busy flag SSCBSY is set and a transmit interrupt request (SSCTXI) will be generated to indicate that SSCTB may be reloaded again. When the programmed number of bits (2 ... 16) has been transferred, the content of the shift register is moved to the Receive Buffer SSCRB and a receive interrupt request (SSCRXI) will be generated. If no further transfer is to take place (SSCTB is empty), SSCBSY will be cleared at the same time. Software should not modify SSCBSY, as this flag is hardware controlled.

Note that only one SSC can be master at a given time.

The transfer of serial data bits may be programmed in many respects:

- The data width may be selected in a range between 2 bits and 16 bits.
- Transfer may start with the LSB or the MSB.
- The shift clock may be idle low or idle high.
- Data bits may be shifted with the leading or trailing edge of the clock signal.
- The baudrate may be set from 152 Baud up to 5 MBaud (@ 20 MHz CPU clock).
- The shift clock can be either generated (master) or received (slave).

This flexible programming allows to adapt the SSC to a wide range of applications, where serial data transfer is required.

The Data Width Selection allows to transfer frames of any length, from 2-bit ’characters’ up to 16-bit ’characters’. Starting with the LSB (SSCHB=’0’) allows communicating e.g. with ASC0 devices in synchronous mode (C166 family) or 8051 like serial interfaces. Starting with the MSB (SSCHB=’1’) allows to operate compatible with the SPI interface. Regardless which data width is selected and whether the MSB or the LSB is transmitted first, the transfer data is always right aligned in registers SSCTB and SSCRB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of SSCTB are ignored, the unselected bits of SSCRB will be not valid and should be ignored by the receiver service routine.

The Clock Control allows to adapt transmit and receive behaviour of the SSC to a variety of serial interfaces. A specific clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in receive data. Bit SSCPH selects the leading edge or the trailing edge for each function. Bit SSCPO selects the level of the clock line in the idle state. Hence for an idle-high clock the leading edge is a falling one, a 1-to-0 transition. The figure below summarizes the clock control.

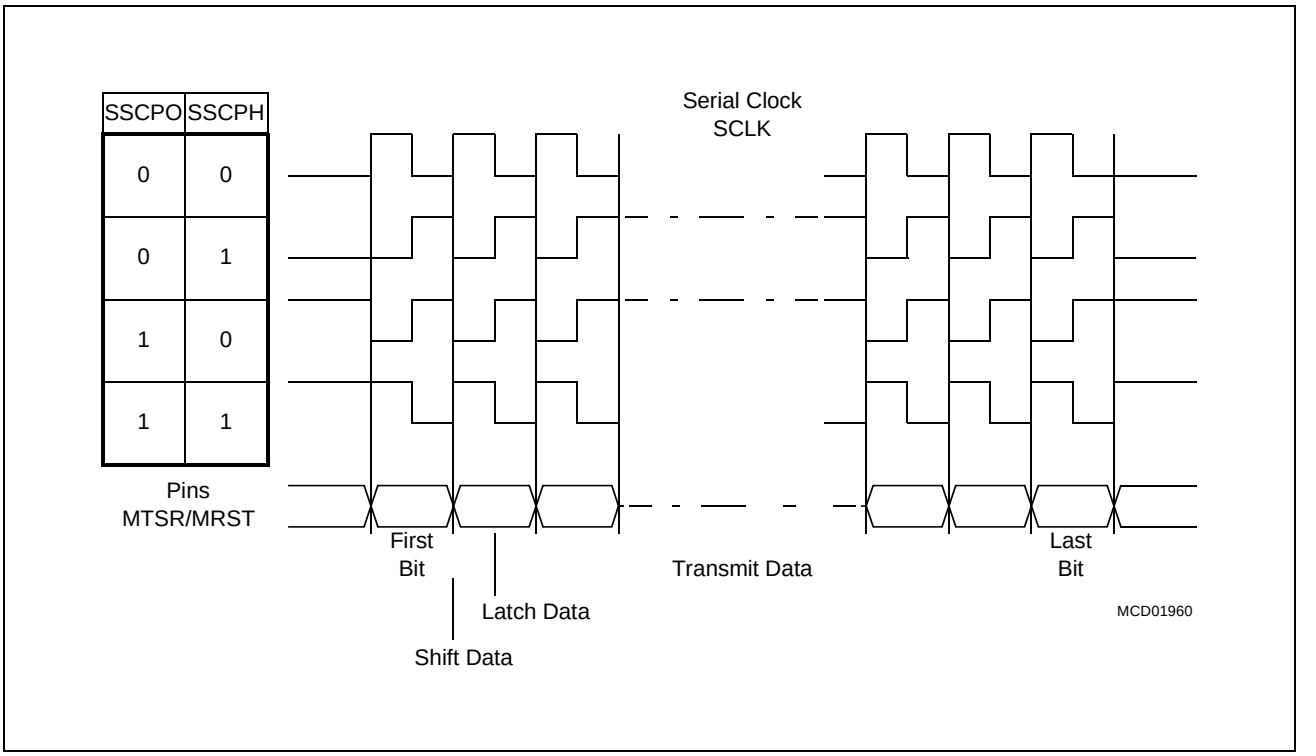


Figure 35 Serial Clock Phase and Polarity Options

6.2.1.2 Operational Mode: Full-Duplex Operation:

The different devices are connected through three lines. The definition of these lines is always determined by the master: The line connected to the master's data output pin MTSR is the transmit line, the receive line is connected to its data input line MRST, and the clock line is connected to pin MSCLK. Only the device selected for master operation generates and outputs the serial clock on pin MSCLK. All slaves receive this clock, so their pin MSCLK must be switched to input mode (GPDIR.p=’0’). The output of the master’s shift register is connected to the external transmit line, which in turn is connected to the slaves’ shift register input. The output of the slaves’ shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

When initializing the devices in this configuration, select one device for master operation (SSCMS=’1’), all others must be programmed for slave operation (SSCMS=’0’). Initialization includes the operating mode of the device's SSC and also the function of the respective port lines (refer to section ’Port Control’).

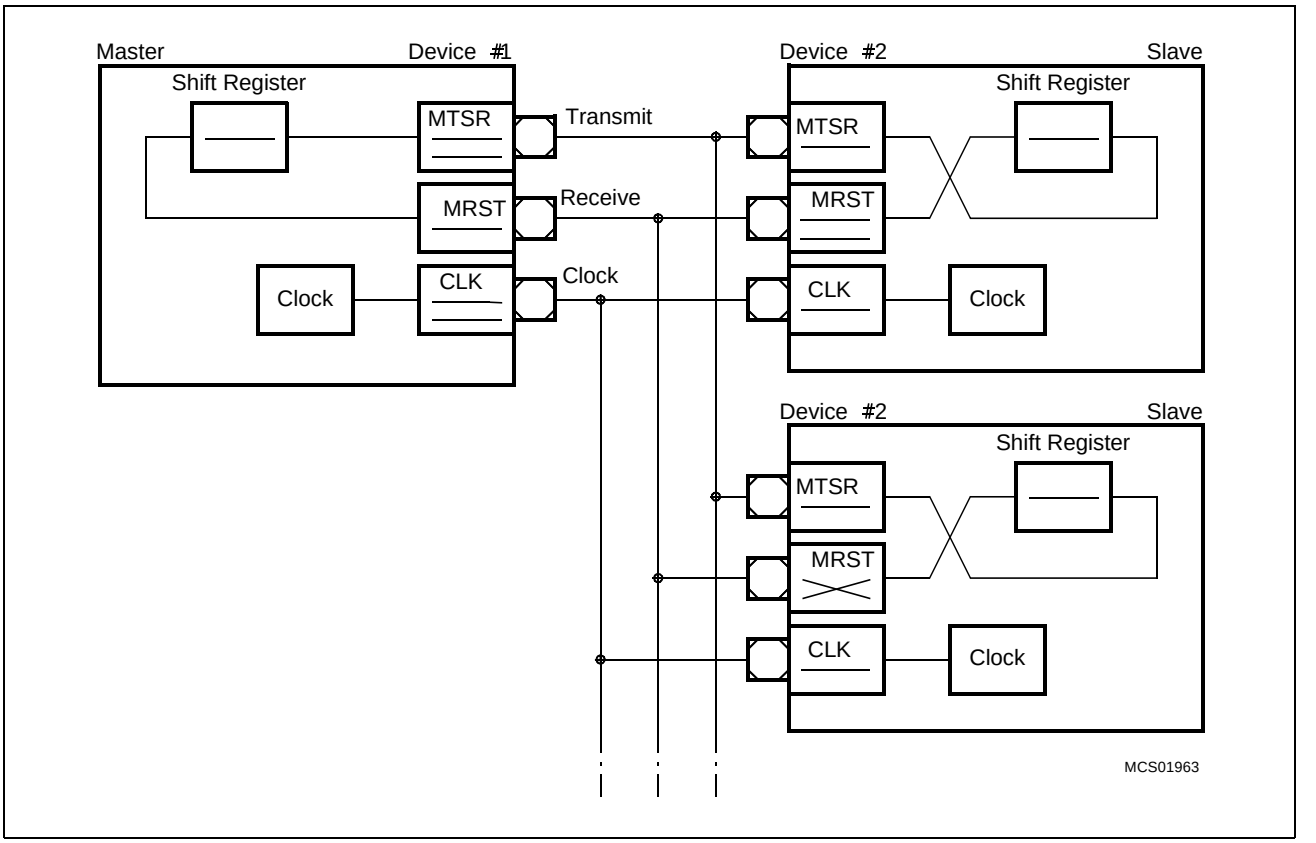


Figure 36 SSC Full Duplex Configuration

Note: The shift direction applies to MSB-first operation as well as to LSB-first operation.

The data output pins MRST of all slave devices are connected together onto the one receive line in this configuration. During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

1. **Only one slave drives the line**, i.e. enables the driver of its MRST pin. All the other slaves have to program their MRST pins to input. So only one slave can put its data onto the master's receive line. Only receiving of data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output, until it gets a deselection signal or command.
2. **The slaves use open drain output on MRST**. This forms a Wired-AND connection. The receive line needs an external pullup in this case. Corruption of the data on the receive line sent by the selected slave is avoided, when all slaves which are not selected for transmission to the master only send ’ones’. Since this high level is not actively driven onto the line, but only held through the pullup device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device, from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initializations of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either '0' or '1', until the first transfer will start. After a transfer the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interface is enabled, the master device can initiate the first data transfer by writing the transmit data into register SSCTB. This value is copied into the shift register (which is assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the MTSR line on the next clock from the baudrate generator (transmission only starts, if SSCEN=’1’). Depending on the selected clock phase, a clock pulse will also be generated on the MSCLK line. With the opposite clock edge the master at the same time latches and shifts in the data detected at its input line MRST. This ’exchanges’ the transmit data with the receive data. Since the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master's shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the preprogrammed number of clock pulses (via the data width selection) the data transmitted by the master is contained in all slaves’ shift registers, while the master's shift register holds the data of the selected slave. In the master and all slaves the content of the shift register is copied into the receive buffer SSCRB and the receive interrupt vector is generated, if enabled.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin MRST, when the content of the transmit buffer is copied into the slave's shift register. It will not wait for the next clock from the baudrate generator, as the master does. The reason is that, depending on the selected clock phase, the first clock edge generated by the master may already be used to clock in the first data bit. Hence the slave's first data bit must already be valid at this time.

*Note: On the SSC always a transmission **and** a reception takes place at the same time, regardless whether valid data has been transmitted or received.*

The initialization of the MSCLK pin on the master requires some attention in order to avoid undesired clock transitions, which may disturb the other receivers. The state of the internal alternate output lines is '1' as long as the SSC is disabled. This alternate output signal is ANDed with the respective port line output latch. Enabling the SSC with an idle-low clock (SSCPO=’0’) will drive the alternate data output and (via the AND) the port pin MSCLK immediately low. To avoid this, use the following sequence:

- select the clock idle level (SSCPO=’x’),
- load the port output latch with the desired clock idle level (GPDATA.p=’x’),
- switch the pin to output (GPDIR.p=’1’),
- enable the SSC (SSCEN=’1’), and
- if SSCPO=’0’: enable alternate data output (GPDATA.p=’1’).

The same mechanism as for selecting a slave for transmission (separate select lines or special commands) may also be used to promote the role of the master to another device in the network. In this case the previous master and the future master (previous slave) will have to toggle their operating mode (SSCMS) and the direction of their port pins (see description above).

Chip Select Control:

There are 4 chip select pins associated with the SSC port: $\overline{\text{MCS0}}$ to $\overline{\text{MCS3}}$. The four chip select lines are automatically activated at the beginning of a transfer and deactivated again after the transfer has ended. Activation of a chip enable line always begins one half bit time before the first data bit is output at the MTSR pin, and the deactivation (except for the continuous transfers) is performed one half bit time after the last bit of the transfer has been transmitted/received completely.

The chip select lines are selected by the control bits ASEL0 to ASEL3 of the SSC Chip Select Enable register SSCCSE (refer to **Page 364**). By setting any of these bits to 0, the corresponding chip select port will be asserted when transmitting data. All other bits of the SSCCSE register have to be set to ‘0‘.

6.2.1.3 Operational Mode: Half Duplex Operation:

In a half duplex configuration only one data line is necessary for both receiving **and** transmitting of data. The data exchange line is connected to both pins MTSR and MRST of each device, the clock line is connected to the MCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to full duplex mode there are two ways to avoid collisions on the data exchange line:

- only the transmitting device may enable its transmit pin driver
- the non-transmitting devices use open drain output and only send ones.

Since the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). This allows to detect any corruptions on the common data exchange line, where the Rx data is not equal to the Tx data.

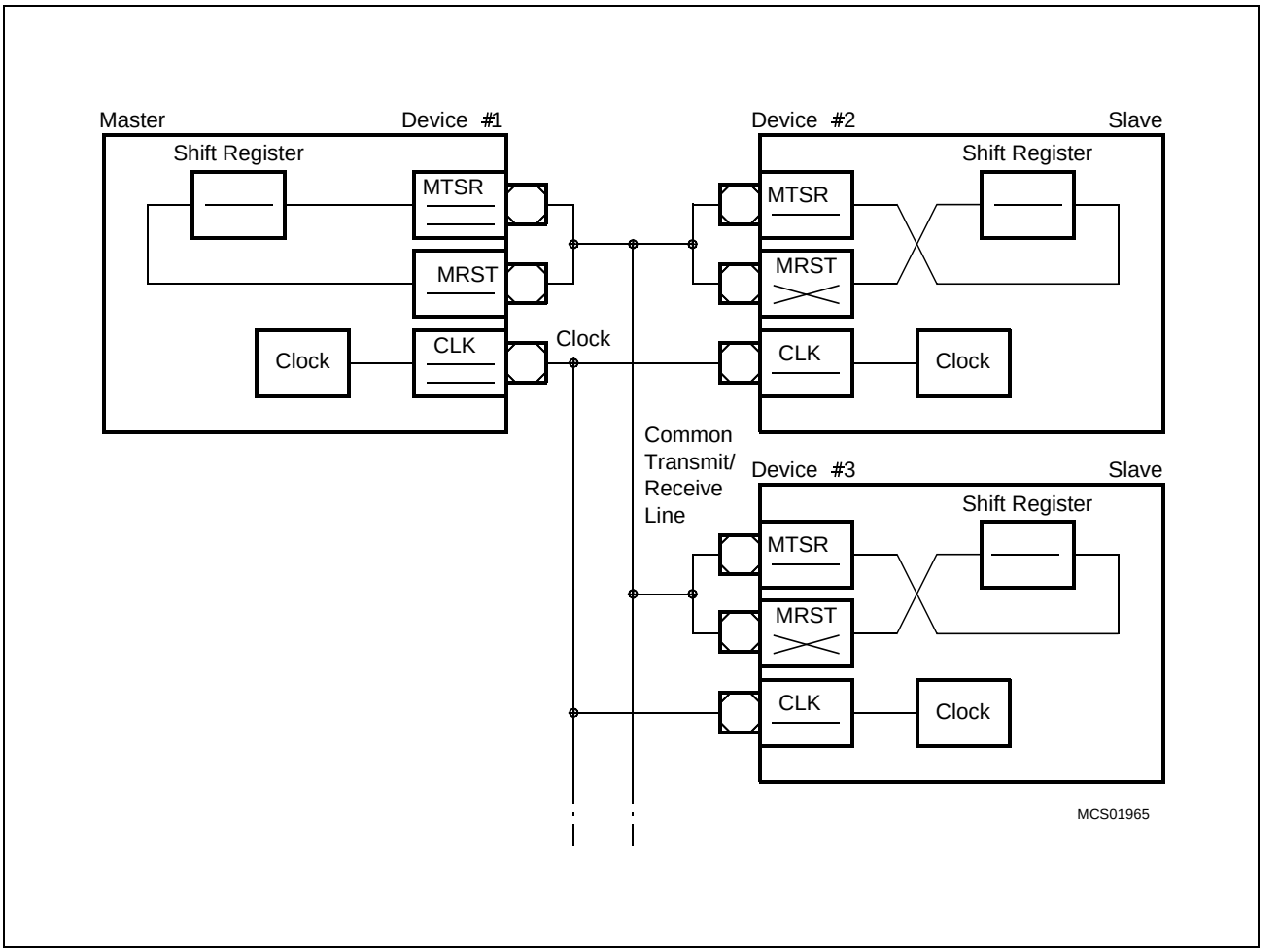


Figure 37 SSC Half Duplex Configuration

Continuous Transfers:

When the transmit interrupt request flag is set, it indicates that the transmit buffer SSCTB is empty and ready to be loaded with the next transmit data. If SSCTB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line there is no gap between the two successive frames. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices which can operate with or require more than 16 data bits per transfer. The length of a total data frame is up to the software. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus.

Note: This feature only applies to multiples of the selected basic data width, since it would require disabling/enabling of the SSC to re-program the basic data width on-the-fly.

Port Control:

The SSC uses three pins to communicate with the external world. Pin MCLK serves as the clock line, while pins MRST (Master Receive/Slave Transmit) and MTSR (Master Transmit/Slave Receive) serve as the serial data input/output lines. The operation of these pins depends on the selected operating mode (master or slave).

The direction of the port lines depends on the operating mode, that is selected via SSCCON:SSCMS.

6.2.1.4 Baud Rate Generation

The SSC interface has its own dedicated 16-bit baud rate generator with 16-bit reload capability, allowing baud rate generation independent from timers.

The baud rate generator is clocked with the CPU clock divided by 2 (10 MHz @ 20 MHz bus clock). The timer is counting downwards and can be started or stopped through the global enable bit SSCEN in the SSC Control register SSCCON.

The register SSCCON (refer to section **“SSC Registers Description” on Page 355**) is the dual-function Baud Rate Generation register. Reading SSCBR, while the SSC is enabled, returns the contents of the timer. Reading SSCBR, while the SSC is disabled, returns the programmed reload value. In this mode the desired reload value can be written to SSCBR.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baudrate:

$$Baudrate_{SSC} = \frac{f_{CPU}}{2 * (<SSCBR> + 1)} \qquad SSCBR = \left(\frac{f_{CPU}}{2 * Baudrate_{SSC}} \right) - 1$$

Bit field <SSCBR> represents the contents of the reload register, taken as an unsigned 16-bit integer.

The maximum baud rate that can be achieved when using a PCI clock of 20 MHz is 5 MBaud. The table below lists some possible baud rates together with the required reload values and the resulting bit times, assuming a PCI clock of 20 MHz. A PCI clock of 33 MHz is also supported.

6.2.1.5 Error Detection

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, while Transmit Error and Baudrate Error only apply to slave mode. When an error is detected, the respective error flag is set. When the corresponding error enable bit is set, also an error interrupt request will be generated by setting SSCERI (see **Figure 38**). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset

automatically (like SSCERI), but rather must be cleared by software after servicing. This allows to service some error conditions via interrupt, while the others may be polled by software.

Note: The error interrupt handler must clear the associated (enabled) errorflag(s) to prevent repeated interrupt requests.

A **Receive Error** (Master or Slave mode) is detected, when a new data frame is completely received, but the previous data was not read out of the Receive Buffer register SSCRB. This condition sets the error flag SSCRE and, when enabled via SSCREN, the error interrupt request flag SSCERI. The old data in the receive buffer SSCRB will be overwritten with the new value and is unretrievably lost.

A **Phase Error** (Master or Slave mode) is detected, when the incoming data at pin MRST (master mode) or MTSR (slave mode), sampled with the same frequency as the CPU clock, changes in a range between one sample before and two samples after the latching edge of the clock signal (refer to section ’Clock Control’). This condition sets the error flag SSCPE and, when enabled via SSCPEN, the error interrupt request flag SSCERI.

A **Baud Rate Error** (Slave mode) is detected, when the incoming clock signal deviates from the programmed baud rate by more than 100%, i.e., it has a value of either more than double or less than half of the expected baud rate. This condition sets the error flag SSCBE and, when enabled via SSCBEN, the error interrupt request flag SSCEIR.

Using this error detection capability requires that the slave's baud rate generator is programmed to the same baud rate as the master device. This feature allows to detect false additional, or missing pulses on the clock line (within a certain frame).

If this error condition occurs and bit SSCAREN=’1’, an automatic reset of the SSC will be performed. This is done to reinitialize the SSC, when too few or too many clock pulses have been detected.

A **Transmit Error** (Slave mode) is detected, when a transfer was initiated by the master (shift clock gets active), but the transmit buffer SSCTB of the slave was not updated since the last transfer. This condition sets the error flag SSCTE and, when enabled via SSCTEN, the error interrupt request flag SSCERI. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which are usually the data received during the last transfer.

This may lead to the corruption of the data on the transmit/receive line in half-duplex mode (open drain configuration), if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ’ones’, i.e. their transmit buffers must be loaded with 'FFFF$_H$' prior to any transfer.

Note: A slave with push/pull output drivers, which is not selected for transmission, will normally have its output drivers switched. However, in order to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

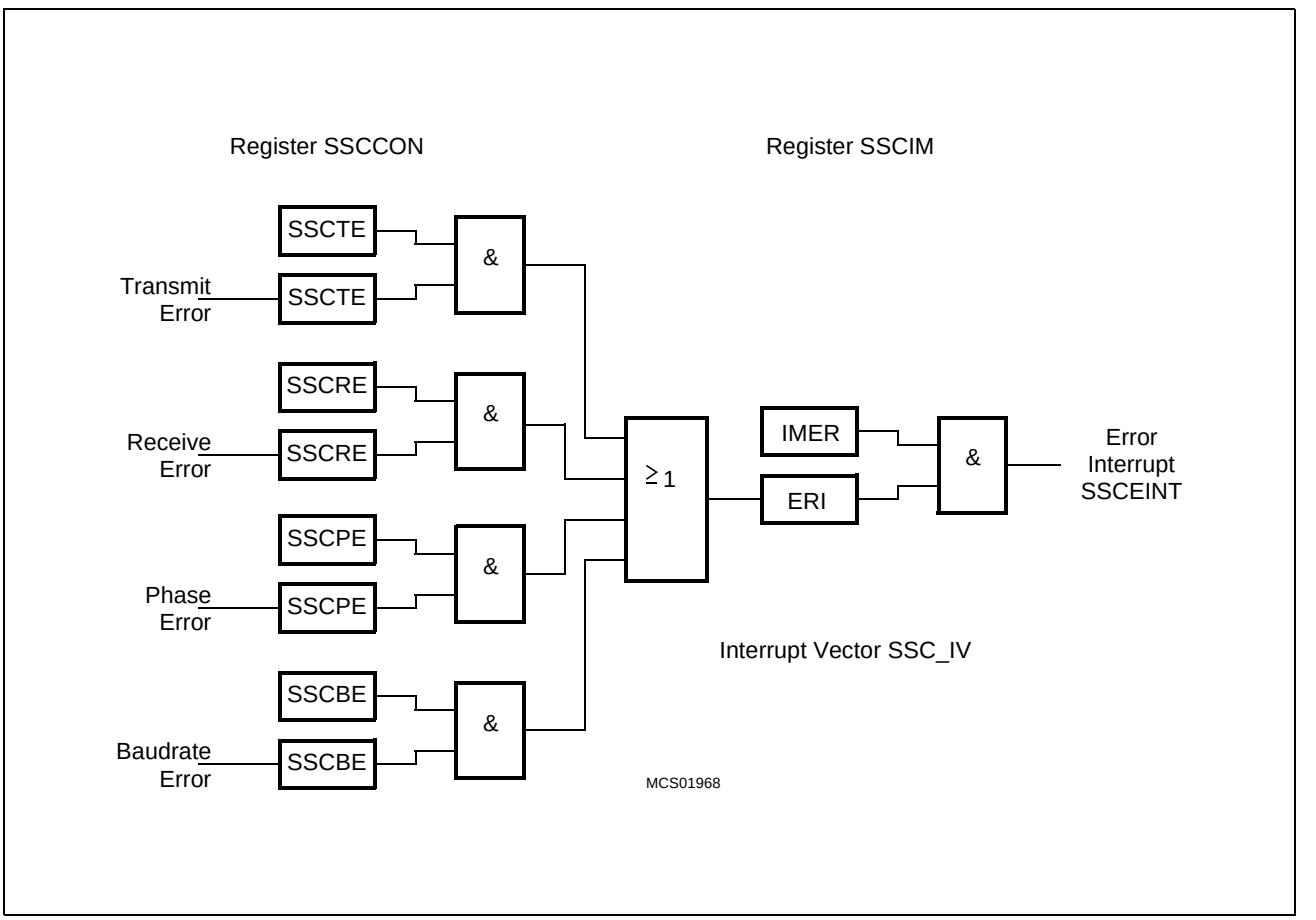


Figure 38 SSC Error Interrupt Control

6.2.2 SSC Interrupt (Vector)

The SSC block generates three kinds of interrupts: transmit, receive and error interrupts. Any of these interrupts can be enabled by setting the corresponding bit of the SSC Interrupt Mask register SSCIM (refer to section **“SSC Registers Description” on Page 355**) to ‘1‘. All other bits of this register have to be set to zero.

All interrupt events result in an SSC interrupt vector which is transferred into the peripheral interrupt queue (refer to section **“SCC Interrupt Vector” on Page 393**).

6.3 General Purpose Port (GPP) Interface

6.3.1 GPP Functional Description

A general purpose 8-/16-bit port is provided on pins GP0...GP15. Every pin is separately programmable via the General Purpose Port Direction Register GPDIR to operate as an output or an input.

The number of available port pins depends on the selected MFP configuration mode (bit field 'PERCFG' in register GMODE).

If defined as output, the state of the pin is directly controlled via the General Purpose Port Data Register GPDATA. Read access to this register delivers the current state of all GPP pins (input and output signals).

If defined as input, the state of the pin is monitored. The value is readable via GPDATA. All changes may be (if desired) indicated via interrupt. Assigned register: General Purpose Port Interrupt Mask Register GPIM. See **"GPP Registers Description" on Page 368**.

6.3.2 GPP Interrupt Vector

The GPP block generates interrupts for transitions on each input (and output) signal. Any pin can be enabled for interrupt generation by setting the corresponding bit of the GPP Interrupt Mask register GPIM (refer to section **"GPP Registers Description" on Page 368**) to '0'.

All interrupt events result in an GPP interrupt vector which is transferred into the peripheral interrupt queue (refer to section **"GPP Interrupt Vector" on Page 398**).

7 Serial Communication Controller (SCC) Cores

7.1 General

The Serial Communication Controller (SCC) distinguishes itself from other communication controllers by its advanced characteristics. The most important are:

– Support of HDLC/SDLC, ASYNC, BISYNC/MONOSYNC, and point-to-point protocols (PPP).
– Support of layer-2 functions (HDLC mode)
  In addition to those bit-oriented functions commonly supported by HDLC controllers, such as bit stuffing, CRC check, flag and address recognition, the SCC provides a high degree of procedural support.
– In a special operating mode (auto-mode), the SCC processes the information transfer and the procedure handshaking (I- and S-frames of HDLC protocol) autonomously. The only restriction is that the window size (= number of outstanding unacknowledged frames) is limited to 1, which is sufficient for many applications. The communication procedures are mainly processed between the communication controllers and not between the attached hosts. Thus the dynamic load on the host and the software expense is greatly reduced.
  The host is informed about the status of the procedure and has mainly to manage the interrupt service. Receive and transmit data are managed by the on chip DMAC autonomously. In order to maintain cost effectiveness and flexibility, the handling of unnumbered (U) frames, and special functions such as error recovery in case of protocol errors, are not implemented in hardware and must be done by the user's software.
– Extended support of different link configurations
  Besides the point-to-point configurations, the SCC allows the implementation of point-to-multipoint or multi-master configurations without additional hardware or software expense.
  In point-to-multipoint configurations, the SCC can be used as a master or as a slave station. Even when working as slave station, the SCC can initiate the transmission of data at any time. An internal function block provides means of idle and collision detection and collision resolution, which are necessary if several stations start transmitting simultaneously. Thus, a multi-master configuration is also possible.
– Telecom specific features
  In a special operating mode, the SCC can transmit or receive data packets in one of up to 128 time-slots of programmable width (clock mode 5). Furthermore, the SCC can transmit or receive variable data portions within a defined window of one or more clock cycles in conjunction with an external strobe signal (clock mode 1). These features make the SCC suitable for applications using time division multiplex methods, such as time-slot oriented PCM systems or systems designed for packet switching.

– Support of PPP Data Link Layer frame transmission
  In a special HDLC sub mode, the SCC provides transmission of PPP Data Link Layer frames in either an asynchronous (start/stop), bit-synchronous or octet-synchronous mode. An escape mechanism is implemented to allow control data such as XON/XOFF to be transmitted transparently via the link, and to remove spurious control data which may be injected into the link by intervening hardware and software.
– FIFO buffers for efficient transfer of data packets
  Since all SCCs are contending for the internal busses, each SCC has an eight 32-bit word deep FIFO in transmit and an seventeen 32-bit word deep FIFO in receive direction for temporary storage of data packets transferred between the serial communications interface and the central FIFOs of the DSCC4. These FIFOs allow overlapping input/output operation (dual-port behavior).
– High Data Rate **(PEB 20534H-52 only)**
  In a special operating mode (clock mode 4, high speed mode) any of the four SCCs can support high data rates: e.g. 45 Mbit/s for DS3 or 52 Mbit/s for OC1. The aggregate bandwidth supported is 108 MBit/s per direction. This allows various configurations, for example:
  - 2 ports 52 MBit/s and 2 ports 2 MBit/s
  - 2 ports 45 MBit/s and 2 ports 8 MBit/s
  - 4 ports 26 MBit/s

Features included by each one of the SCCs:

- Serial Interface
  – On chip clock generation or external clock source
  – On chip DPLL for clock recovery
  – Baud rate generator
  – Programmable time-slot capability
  – NRZ, NRZI, FM0/1 and Manchester data encoding
  – Optional data flow control using modem lines ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, CD)
  – Support of bus configuration by collision detection and resolution
  – Full duplex data rates of up to 10 Mbit/s sync - 2 Mbit/s with DPLL, 2 Mbit/s async
  – Full duplex data rate of up to 52 Mbit/s (HDLC address mode 0, PPP or extended transparent mode) in clock mode 4 (external clock source and clock gating/gapping).
- Bit Processor Functions
  – HDLC/SDLC Mode
    - Automatic flag detection and transmission
    - Shared opening and closing flag
    - Generation of interframe-time fill ’1’ s or flags
    - Detection of receive line status
    - Zero bit insertion and deletion
    - CRC generation and checking (CRC-CCITT or CRC-32)

- Transparent CRC option per channel and/or per frame
- Programmable Preamble (8 bit) with selectable repitition rate
- Error detection (abort, overrun, underrun, CRC error, too long/short frame)

– ASYNC Mode
- Selectable character length (5 to 8 bits)
- Even, odd, forced or no parity generation/checking
- 1 or 2 stop bit insertion in transmit
- Break detection/generation
- Flow control by XON/XOFF character
- Immediate character insertion (for insertion of control characters in data stream)
- Termination character detection for end of block identification
- Time out detection
- Error detection (parity error, framing error)

– BISYNC Mode
- Programmable 6/8 bit SYNC pattern (MONOSYNC)
- Programmable 12/16 bit SYNC pattern (BISYNC)
- Selectable character length (5 to 8 bits)
- Even, odd, forced or no parity generation/checking
- Generation of interframe-time fill ’1’ s or SYNC characters
- CRC generation (CRC-16 or CRC-CCITT)
- Transparent CRC option per channel and/or per frame
- Programmable Preamble (8 bit) with selectable repitition rate
- Termination character detection for end of block identification
- Error detection (parity error, CRC error)

• Protocol Support (provided in HDLC/SDLC Mode)

– Address mode 0
- No address recognition

– Address mode 1
- 8-bit (high byte) address recognition

– Non-auto mode
- 8-bit (low byte) or 16-bit (high and low byte) address recognition

– Auto mode
- 8-bit or 16-bit address generation/recognition
- Automatic handling of S- and I-frames
- Support of LAP-B / LAP-D
- Automatic processing of control byte(s)
- Modulo 8 or modulo 128 operation
- Programmable time-out and retry conditions
- Normal Response Mode operation for slave

– Asynchronous PPP mode
- Character oriented transmission of HDLC frame (flag, data, CRC, flag)
- Start/stop bit framing of each single character
- Automatic flag detection and transmission

- Shared opening and closing flag
- Generation of interframe-time fill ’1’ s or flags
- Detection of receive line status
- No zero bit insertion/deletion
- CRC generation and checking (CRC-CCITT or CRC-32)
- Transparent CRC option per channel and/or per frame
- Programmable Preamble (8 bit) with selectable repitition rate
- Error detection (abort, overrun, underrun, long frame, CRC error, short frames)
- Escape mechanism for control data
- Programmable character map for escape mechanism (00_H..$1F_H$ selectable from fixed character map, 4 additional programmable characters)

– Bit synchronous PPP mode
- Automatic flag detection and transmission
- Shared opening and closing flag
- Generation of interframe-time fill ’1’ s or flags
- Detection of receive line status
- Zero bit insertion and deletion
- 15 bit ’1’ abort sequence
- CRC generation and checking (CRC-CCITT or CRC-32)
- Transparent CRC option per channel and/or per frame
- Programmable Preamble (8 bit) with selectable repetition rate
- Error detection (abort, overrun, underrun, long frame, CRC error, short frames)

– Octet synchronous PPP mode
- Automatic flag detection and transmission
- Shared opening and closing flag
- Generation of interframe-time fill ’1’ s or flags
- Detection of receive line status
- CRC generation and checking (CRC-CCITT or CRC-32)
- Transparent CRC option per channel and/or per frame
- Programmable Preamble (8 bit) with selectable repetition rate
- Error detection (abort, overrun, underrun, long frame, CRC error, short frames)
- Escape mechanism for control data
- Programmable character map for escape mechanism (0x00..0x1F selectable from fixed character map, 4 additional programmable characters)

– Extended transparent mode
- Bit-transparent data transmission/reception (No HDLC-framing, no bit stuffing...)

• Protocol and Mode Independent
- Data inversion
- Data over- and underflow detection
- Timer for software support
- Internal test loop capability

7.2 Protocol Modes Overview

The SCC is a multi-protocol communication controller. Three major protocol blocks are implemented: HDLC/SDLC, ASYNC and BISYNC.

These protocol blocks provide different protocol modes which are listed in **Table 19**.

Table 19 Protocol Modes

Protocol Blocks	Protocol Modes
HDLC/SDLC PPP	HDLC auto mode (16-bit)
	HDLC auto mode (8-bit)
	HDLC non auto mode (16-bit)
	HDLC non auto mode (8-bit)
	HDLC address mode 1
	HDLC address mode 0
	Asynchronous PPP mode
	Bit synchronous PPP mode
	Octet synchronous PPP mode
	Extended transparent mode[1)]
ASYNC	Asynchronous mode
	Isochronous mode
BISYNC	Bisynchronous mode
	Monosynchronous mode

1) Extended transparent is a fully bit-transparent transmit/reception mode which is treated as a sub-mode of the HDLC/SDLC block.

The protocol modes are described in details in **Chapter 8**, " Detailed Protocol Description"

7.3 SCC FIFOs

Each SCC provides its own transmit and receive FIFOs to handle internal arbitration of the central FIFOs.

7.3.1 SCC Transmit FIFO

The SCC transmit FIFO is divided into two parts of 6 and 2 DWORDs. The interface between the two parts provides clock synchronization between the system clock domain and the protocol logic working with the serial transmit clock.

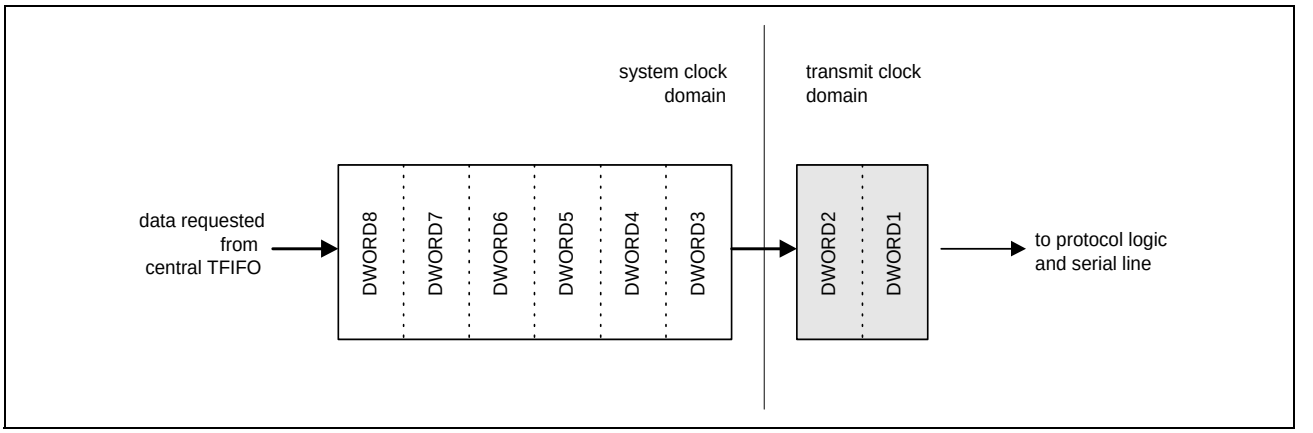


Figure 39 SCC Transmit FIFO

The 6 DWORDs system clocked FIFO part always requests transmit data from the central TFIFO if at least 4 DWORDs free space is available even if the SCC is in power-down condition (register CCR0 bit PU=’0’).

The only exception is a transmit data underrun (XDU) event. In case of an XDU event (e.g. after excessive PCI bus latency), the FIFO will neither request more data from the central TFIFO nor transfer another DWORD to the protocol logic.

This XDU blocking mechanism prevents unexpected serial data and must be cleared by a transmitter reset command. In case of a transmitter reset command (register CMDR bit XRES=’1’) the complete SCC transmit FIFO is cleared and will immediately request new transmit data from the central TFIFO.

Transfer of data to the 2 DWORD shadow part only takes place if the SCC is in power-up condition and an appropriate transmit clock is provided depending on the selected clock mode.

Serial data transmission will start as soon as at least one DWORD is transferred into the 2 DWORD shadow FIFO and transmission is enabled depending on the selected clock mode ($\overline{\text{CTS}}$ signal active, clock strobe signal active, valid timeslot or clock gapping signal inactive).

7.3.2 SCC Receive FIFO

The SCC receive FIFO is divided into two parts of 15 and 2 DWORDs. The interface between the two parts provides clock synchronization between the system clock domain and the protocol logic working with the serial receive clock.

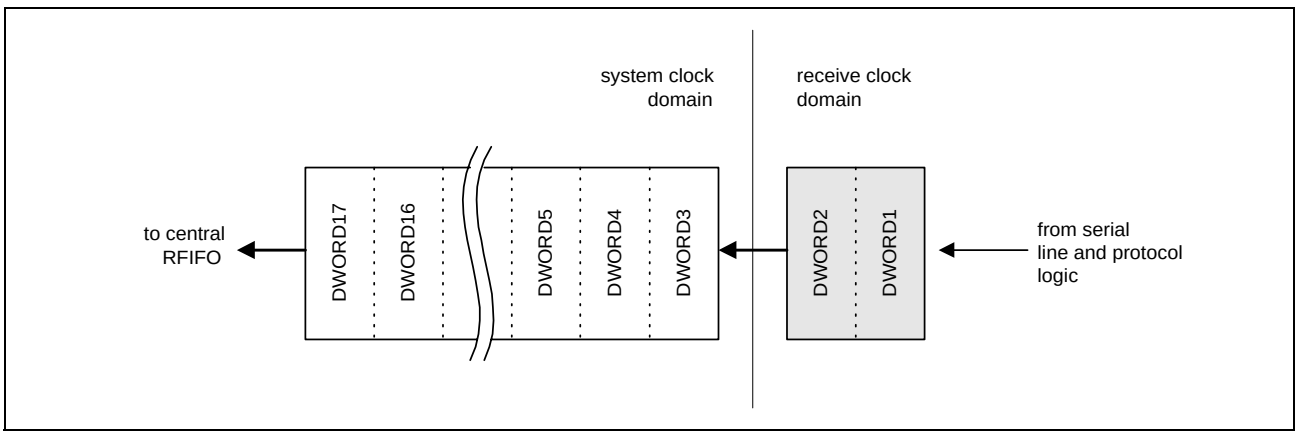


Figure 40 SCC Receive FIFO

With standard register settings (i.e. the SCC receive FIFO threshold is not reduced, refer to **Table 69 "CCR2: Channel Configuration Register 2" on Page 296**), the SCC receive FIFO requests data transfer to the central RFIFO if the 15 DWORDs part is completely filled or a frame end / block end condition is detected.

This SCC receive FIFO size is optimized for high speed channel configurations. The 15 DWORDs FIFO part is transferred to the central RFIFO allocating one consecutive block of RFIFO memory. This guarantees full 15 DWORDs burst length on the PCI/De-multiplexed system interface which can be performed on consecutive RFIFO sections only (refer to **Chapter 5.2.3**, **"** Central Receive FIFO (RFIFO)**"**).

Nevertheless this FIFO depth might cause too long delay in low speed channel configurations on data transfer to the host memory (especially ASYNC/BISYNC protocol modes). Therefore the SCC receive FIFO threshold can be lowered in some steps downto 1 data byte causing the SCC to request data transfer to the central RFIFO as soon as this threshold is reached. The threshold is adjusted by bit field ’RFTH’ in register CCR2.

In addition data stored in the SCC receive FIFO can be transferred to the central RFIFO any time on request by setting command ’RFRD’ in register CMDR. Prior to issuing a ’RFRD’ command, the "receive FIFO not empty condition" can be tested by the host CPU by reading bit ’RFNE’ in register STAR.

Furthermore in ASYNC mode this ’RFRD’ command can be generated automatically on a time out condition if enabled via bit ’TOIE’ in register CCR1. In ASYNC applications characters are often send in blocks which means a small time gap between characters of these blocks; but also single control characters may be interleaved. In this case the receive FIFO threshold might be adjusted to the length of expected ASYNC character

blocks or higher but single characters not exceeding the threshold are also forwarded to the central RFIFO after time out. A time out condition is detected if the line idle time exceeds a programmable time period (see register CCR1 bit field ’TOLEN’).

Mention that any ’RFRD’ command generated by write access to register CMDR or automatically by time out mechanism in ASYNC/BISYNC modes always forces a ’frame end/block end’ condition (FE=’1’) causing the DMA controller to finish the current receive descriptor.

If the SCC receive FIFO is completely filled further incoming data is ignored and a receive data overflow condition (RDO) is detected. As soon as the receive FIFO provides empty space receive data is accepted again waiting for a frame end or frame abort sequence. The automatically generated receive status byte (RSTA) will contain an RDO indication in this case and the next incoming frame will be received in a normal way.

Therefore no further CPU intervention is necessary to recover the SCC from an RDO condition.

A "frame" with RDO status might be a mixture of a frame partly received before the RDO event occured and the rest of this frame received after the receive FIFO again accepted data and the frame was still incoming. A quite arbitrary series of data or complete frames might get lost in case of an RDO event. Every frame which must be completely discarded because of an RDO condition generates an RFO interrupt.

The SCC receive FIFO can be cleared by command ’RRES’ in register CMDR. Note that clearing the receive FIFO during operation might delete a frame end / block end indication. A frame which was already partly transferred to the central RFIFO cannot be "closed" in this case because the DMA controller will not get the corresponding frame end indication. A new frame received after receiver reset command will be appended to this "open" frame.

In ASYNC and BISYNC protocol modes, a frame end / block end indication can be forced by command ’RFRD’ in register CMDR to avoid this unexpected behaviour.

In this case a useful sequence of clearing the SCC receive FIFO during operation is:

	Register	Bit field	Description
ASYNC/BISYNC Modes	CCR2	RAC=’0’	Switches the receive protocol logic to inactive state not accepting any more data from the serial line.
	CMDR	RRES=’1’	Resets the receive protocol logic and clears the SCC receive FIFO (self-resetting command bit).
	CMDR	RFRD=’1’	Generates a frame end indication which is transferred to the central RFIFO terminating any partial frame. If no partial frame was stored, this will result in a "frame" with byte number zero.
	CCR2	RAC=’1’	Switches the receive protocol logic to active state accepting receive data from the serial line.

7.4 Clocking System

The DSCC4 includes an internal Oscillator (OSC) as well as four independent Baud Rate Generators (BRG) and four Digital Phase Locked Loop (DPLL) circuits.

The transmit and receive clock can be generated either

- externally, and supplied directly via the RxCLK and/or TxCLK pins (called external clock modes)
- internally, by selecting
  - the internal oscillator (OSC) and/or the channel specific baud rate generator (BRG)
  - the internal DPLL, recovering the receive (and optionally transmit) clock from the receive data stream.

  (called internal clock modes)

There are a total of 13 different clocking modes programmable via bit field 'cm' in register CCR0, providing a wide variety of clock generation and clock pin functions, as shown in **Table 20**.

The transmit clock pins (TxCLK) may also be configured as output clock and control signals in certain clock modes if enabled via bit 'TOE' in register CCR0.

The clocking source for the DPLL's is always the internal channel specific BRG; the scaling factor (divider) of the BRG can be programmed through BRR register.

There are two channel specific internal operational clocks in the SCC:

One operational clock (= transmit clock) for the transmitter part and one operational clock (= receive clock) for the receiver part of the protocol logic.

Note: The internal timers always run using the internal transmit clock.

Table 20 Overview of Clock Modes

Clock			
Type	**Source**	**Generation**	**Clock Mode**
Receive Clock	RxCLK Pins	Externally	0, 1, 4, 5
	OSC, DPLL, BRG,	Internally	2, 3a, 6, 7a 3b, 7b
Transmit Clock	TxCLK Pins, RxCLK Pins	Externally	0a, 2a, 4, 6a 1,5
	OSC, DPLL, BRG/BCR, BRG	Internally	3a, 7a 2b, 6b 0b, 3b, 7b

The internal structure of each SCC channel consists of 3 clocking domains, transmit, receive, and system. These three function blocks are clocked with internal transmit frequency $\mathbf{f_{TRM}}$, internal receive frequency $\mathbf{f_{REC}}$ and system frequency $\mathbf{f_{PCI}}$, respectively (system frequency $\mathbf{f_{PCI}}$ only supplies the SCC receive and transmit FIFO part facing the DMA controller). The internal FIFO interfaces are used to transfer data between the different clock domains.

The clocks $\mathbf{f_{TRM}}$ and $\mathbf{f_{REC}}$ are internal clocks only and need not be identical to external clock inputs e.g. $\mathbf{f_{TRM}}$ and TxCLK input pin.

The features of the different clock modes are summarized in **Table 21**.

Table 21 Clock Modes of the SCCs

Channel Configuration		Clock Sources				Control Sources				
Clock Mode CCR0: CM2, CM1,CM0	CCR0: SSEL	to BRG	to DPLL	to REC	to TRM	CD	R- Strobe	X- Strobe	Frame-Sync	Output via TxCLK (if CCR0: TOE = '1')
0a	0	–	–	RxCLK	TxCLK	CD	–	–	–	–
0b	1	OSC	–	RxCLK	BRG	CD	–	–	–	BRG
1	X	–	–	RxCLK	RxCLK	–	CD	TxCLK	–	–
2a	0	RxCLK	BRG	DPLL	TxCLK	CD	–	–	–	–
2b	1	RxCLK	BRG	DPLL	BRG/16	CD	–	–	–	BRG/16
3a	0	RxCLK	BRG	DPLL	DPLL	CD	–	–	–	DPLL
3b	1	RxCLK	–	BRG	BRG	CD	–	–	–	BRG
4	X	–	–	RxCLK	TxCLK	–	$\overline{\text{RCG}}$	$\overline{\text{TCG}}$	–	-
5	X	–	–	RxCLK	RxCLK	–	(TSAR/ PCMMRX)	(TSAX/ PCMMTX)	FSC	TS-Control
6a	0	OSC	BRG	DPLL	TxCLK	CD	–	–	–	–
6b	1	OSC	BRG	DPLL	BRG/16	CD	–	–	–	BRG/16
7a	0	OSC	BRG	DPLL	DPLL	CD	–	–	–	DPLL
7b	1	OSC	–	BRG	BRG	CD	–	–	–	BRG

Note: If asynchronous operation is selected (asynchronous PPP, ASYNC), some clock mode frequencies can or must be divided by 16 as selected by the Bit Clock Rate bit CCR0:BCR:

Clock Mode	$\mathbf{f_{REC}}$	$\mathbf{f_{TRM}}$
0a	f_{RxCLK}/BCR	f_{TxCLK}
0b	f_{RxCLK}/BCR	f_{BRG}
1	f_{RxCLK}/BCR	f_{RxCLK}/BCR
3b, 7b	f_{BRG}/BCR	f_{BRG}/BCR

When bit clock rate is '16' (bit BCR = '1'), oversampling (3 samples) in conjunction with majority decision is performed. BCR has no effect when using clock mode 2, 3a, 4, 5, 6, or 7a.

Note: If one of the clock modes 0b, 6 or 7 is selected the internal oscillator (OSC) is enabled which allows connection of an external crystal to pins XTAL1-XTAL2. The output signal of the OSC can be used for one serial channel, or for all serial channels (independent baud rate generators and DPLLs). Moreover, XTAL1 alone can be used as input for an externally generated clock.

The first two columns of **Table 21** list all possible clock modes configured via bit field ’CM’ and bit ’SSEL’ in register CCR0.

For example, clock mode 6b is choosen by writing a ’6’ to register CCR1.CMi and by setting bit CCR0.SSEL equal to ’1’. The following 4 columns (grouped as ’Clock Sources’) specify the source of the internal clocks. Columns REC and TRM correspond to the domain clock frequencies $\mathbf{f_{REC}}$ and $\mathbf{f_{TRM}}$.

The columns grouped as ’Control Sources’ cover additional clock mode dependent control signals like strobe signals (clock mode 1), clock gating signals (clock mode 4) or synchronization signals (clock mode 5). The last column describes the function of signal TxCLK which in some clock modes can be enabled as output signal monitoring the effective transmit clock or providing a time slot control signal (clock mode 5).

The following is an example of how to read **Table 21**:

For clock mode 6b (row ’6b’) the TRM clock (column ’TRM’) is supplied by the baudrate generator (BRG) output divided by 16 (source BRG/16). The BRG (column ’BRG’) is derived from the internal oscillator which is supplied by pin XTAL1 and XTAL2.

The REC clock (column ’REC’) is supplied by the internal DPLL which itself is supplied by the baud rate generator (column ’DPLL’) again.

Note: The REC clock is DPLL clock divided by 16.

If enabled by bit ’TOE’ in register CCR2 the resulting transmit clock can be monitored to pin TxCLK (last column, row ’6b’).

The clocking concept is illustrated in a block diagram manner in the following figure:

Additional control signals are not illustrated (please refer to the detailed clock mode descriptions below).

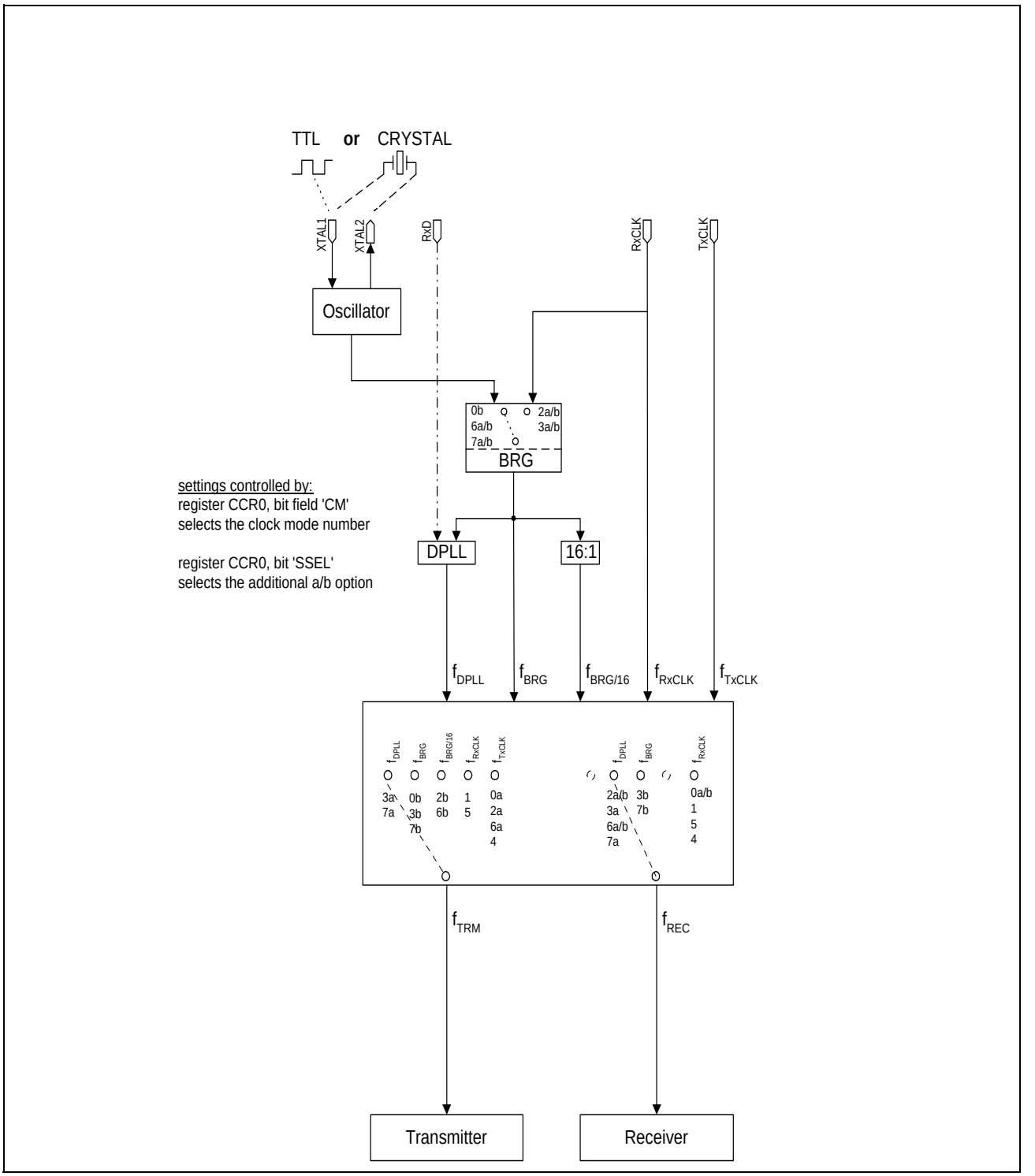


Figure 41 Clock Supply Overview

7.4.1 Clock Modes

7.4.1.1 Clock Mode 0 (0a/0b)

Separate, externally generated receive and transmit clocks are supplied to the SCC via their respective pins. The transmit clock may be directly supplied by pin TxCLK (clock mode 0a) or generated by the internal baud rate generator from the clock supplied at pin XTAL1 (clock mode 0b).

In clock mode 0b the resulting transmit clock can be driven out to pin TxCLK if enabled via bit 'TOE' in register CCR2.

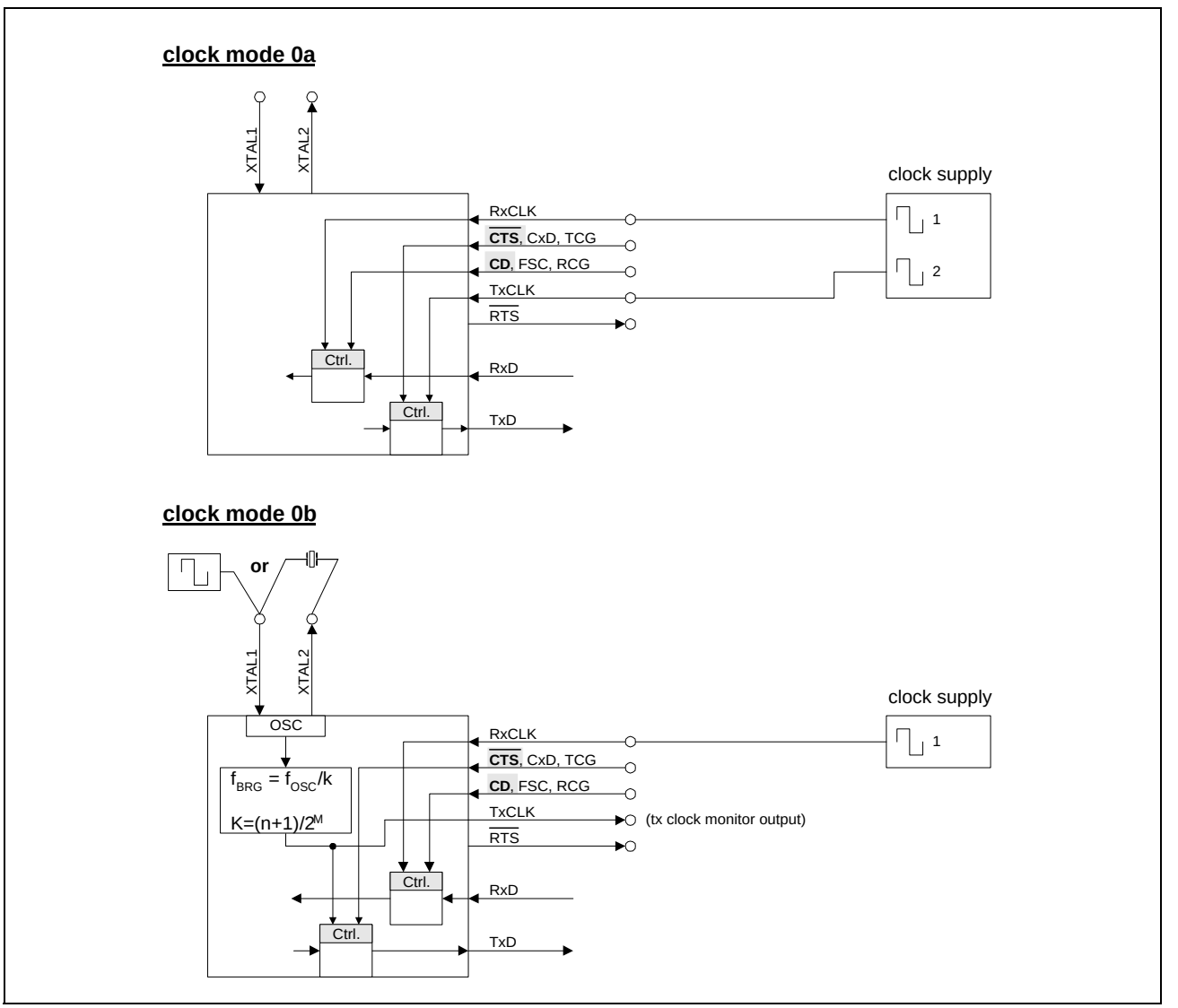


Figure 42 Clock Mode 0a/0b Configuration

7.4.1.2 Clock Mode 1

Externally generated RxCLK is supplied to both the receiver and transmitter. In addition, a receive strobe can be connected via CD and a transmit strobe via TxCLK pin. These strobe signals work on a per bit basis. This operating mode can be used in time division multiplex applications or for adjusting disparate transmit and receive data rates.

Note: In Extended Transparent Mode (HDLC/SDLC), the above mentioned strobe signals provide byte synchronization (byte alignment).
In ASYNC Mode, the above mentioned strobe signals provide character synchronization (character alignment).
This means that the strobe signal needs to be detected once only to transmit or receive a complete byte or character respectively.

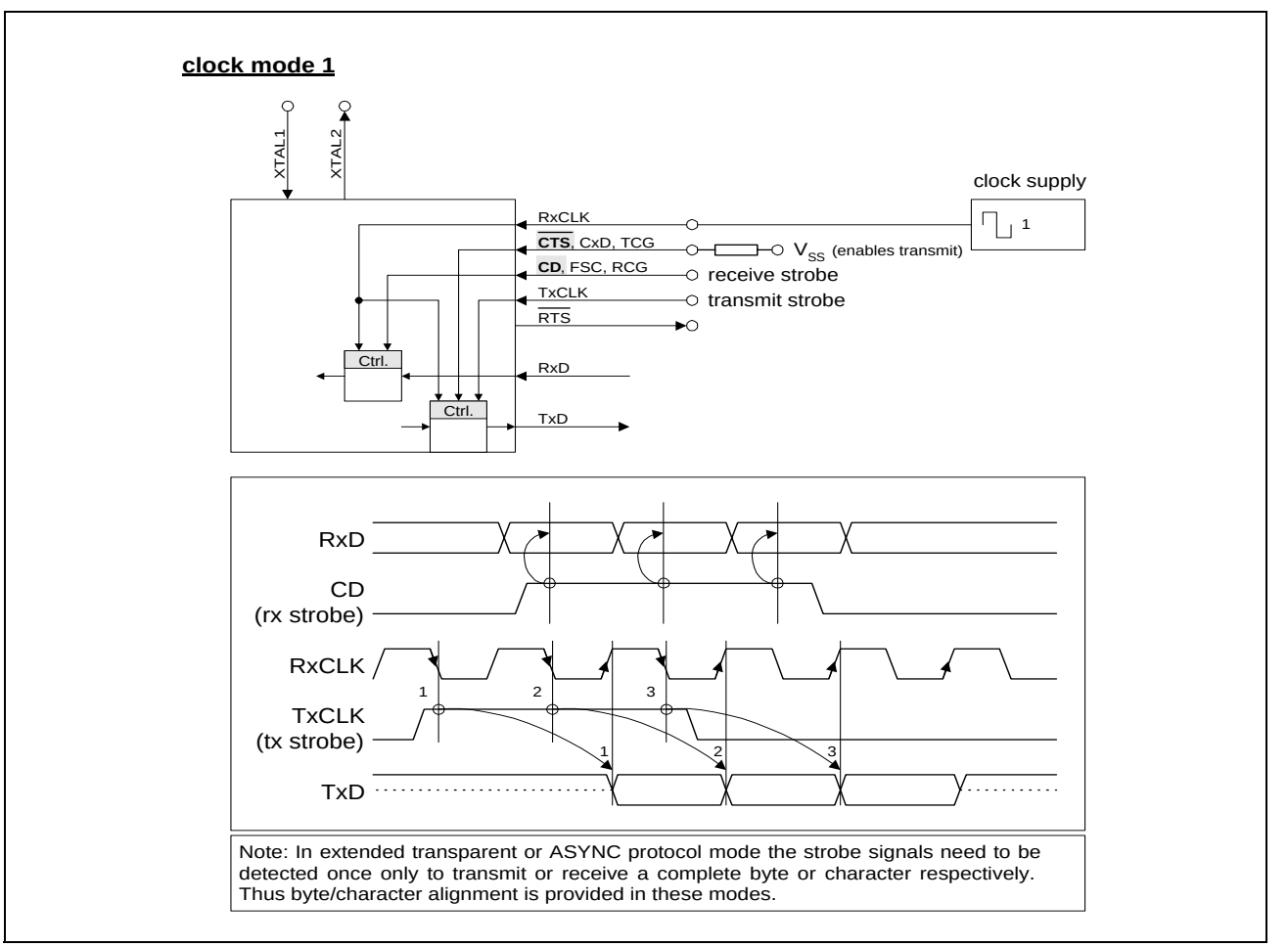


Figure 43 Clock Mode 1 Configuration

7.4.1.3 Clock Mode 2 (2a/2b)

The BRG is driven by an external clock (RxCLK pin) and delivers a reference clock for the DPLL which is 16 times of the resulting DPLL output frequency which in turn supplies the internal receive clock. Depending on the programming of register CCR0 bit 'SSEL', the transmit clock will be either an external input clock signal provided at pin TxCLK in clock mode 2a or the clock delivered by the BRG divided by 16 in clock mode 2b. In the latter case, the transmit clock can be driven out to pin TxCLK if enabled via bit 'TOE' in register CCR2.

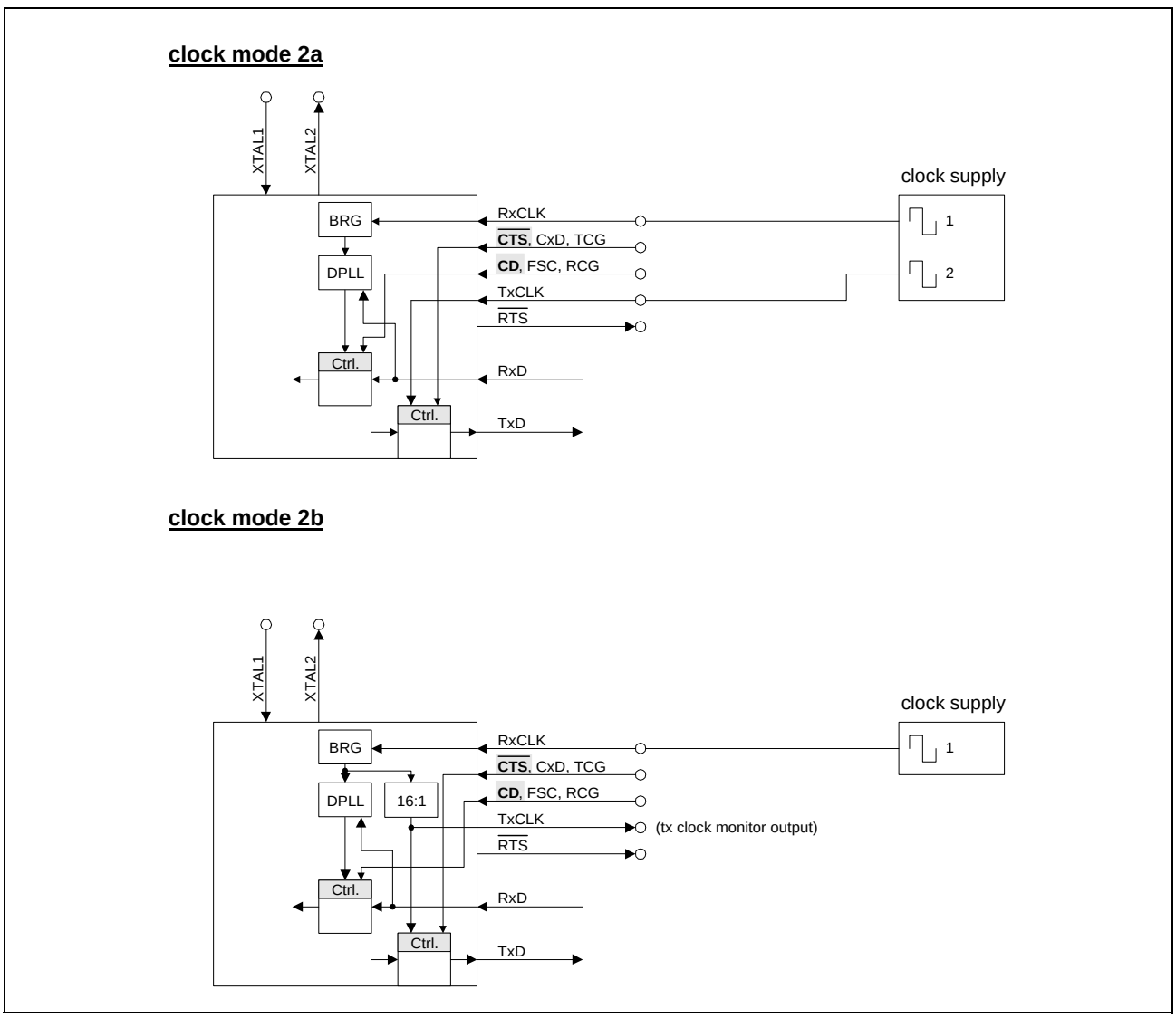


Figure 44 Clock Mode 2a/2b Configuration

7.4.1.4 Clock Mode 3 (3a/3b)

The BRG is fed with an externally generated clock via pin RxCLK. Depending on the value of bit 'SSEL' in register CCR0 the BRG delivers either a reference clock for the DPLL which is 16 times of the resulting DPLL output frequency or delivers directly the receive and transmit clock (clock mode 3b). In the first case (clock mode 3a) the DPLL output clock is used as receive and transmit clock.

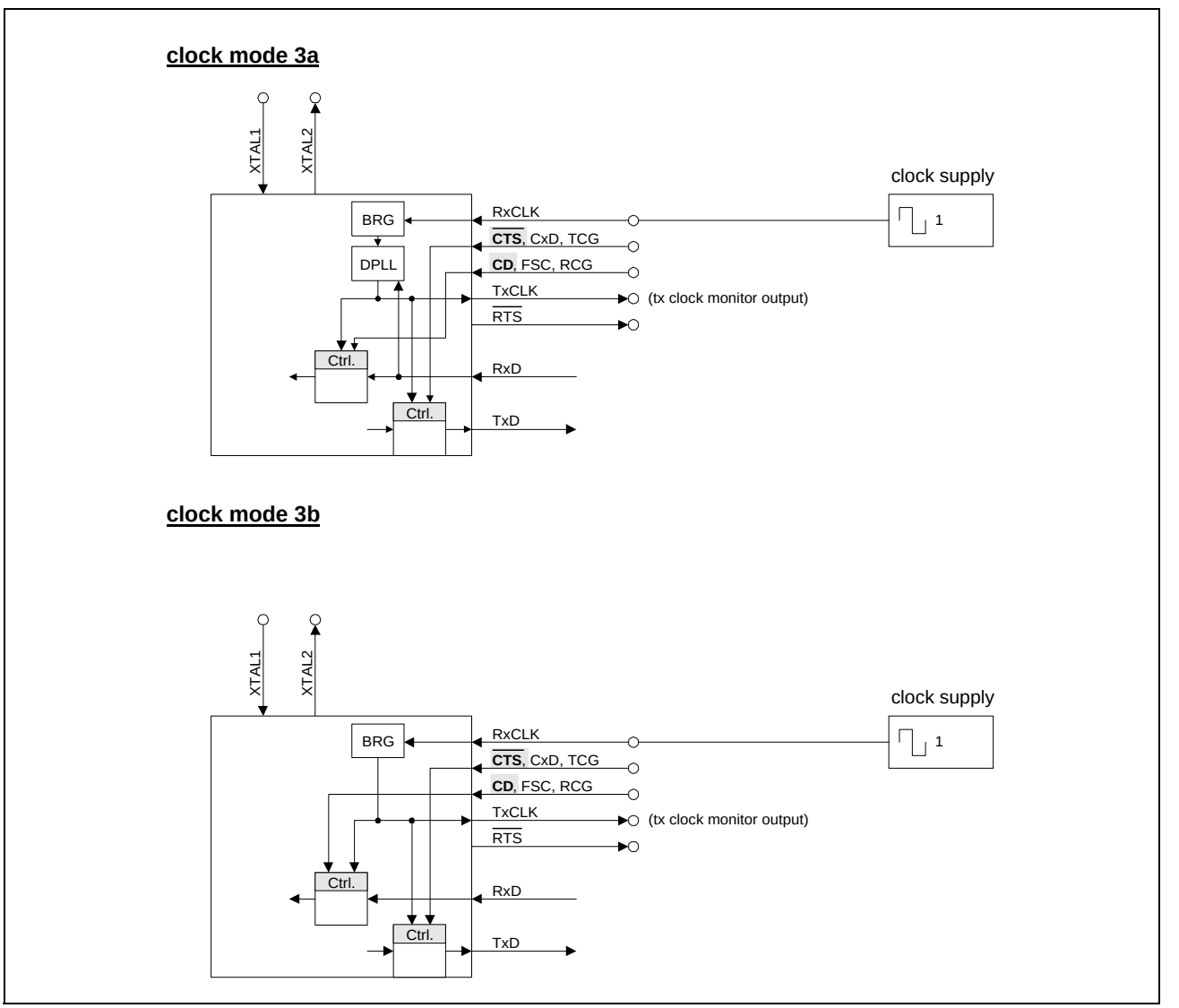


Figure 45 Clock Mode 3a/3b Configuration

7.4.1.5 Clock Mode 4 (High Speed Interface Clock Mode)

Separate, externally generated receive and transmit clocks are supplied via pins RxClk and TxClk. In addition separate receive and transmit clock gating signals are supplied via pins $\overline{\text{RCG}}$ and $\overline{\text{TCG}}$. These gating signals work on a per bit basis.

Note: Clock mode 4 can only be applied in combination with High Speed Serial Mode (register CCR0 bit HS ='1'). This setting optimizes the internal signal and clock trees for high speed timing requirements.

Note: For correct operation only HDLC Address Mode 0, PPP and Extended Transparent Mode should be used.

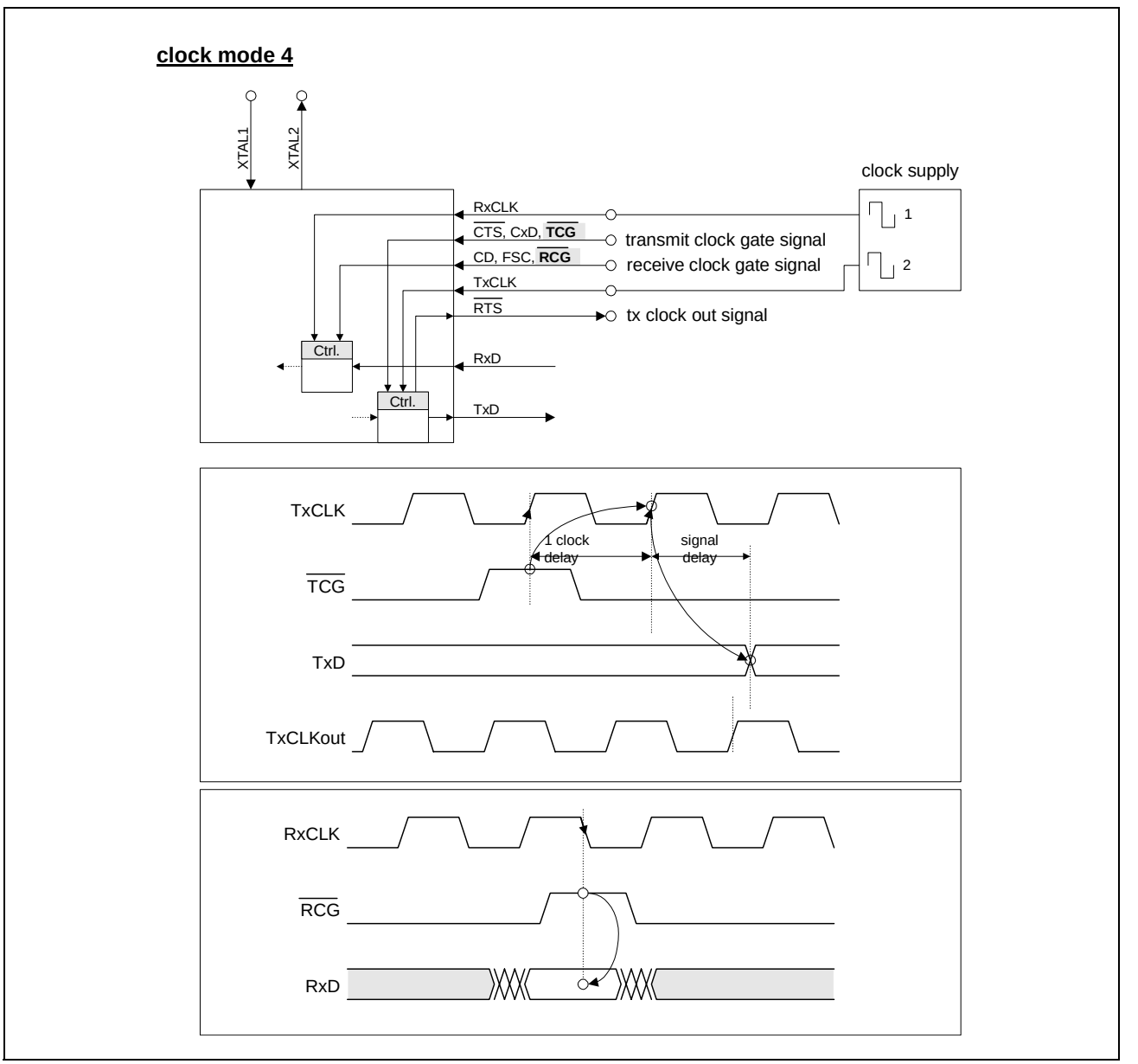


Figure 46 Clock Mode 4 (High Speed) Configuration

The transmit clock supplied at pin TxCLK can be switched out to pin $\overline{\text{RTS}}$ in phase with the transmit data on pin TxD if bit 'TCLKO' in register CCR1 is set. Due to internal signal delays the transmit data output signal delay to TxCLK may be high with regard to the total clock period of 19.2 nanoseconds which is the minimum high speed clock period. Therefore the transmit clock is supplied to pin $\overline{\text{RTS}}$ such that TxD signal has a small delay to this monitor clock which might be useful for connection to external transceiver devices.

7.4.1.6 Clock Mode 5

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

Note: For correct operation NRZ data coding/encoding should be used.

The receive and transmit clock are common for each channel and must be supplied externally via pin RxCLK. The SCC receives and transmits only during fixed time-slots. Either one time-slot

– of programmable width (1 … 512 bit, via TSAR and TSAX registers), and
– of programmable location with respect to the frame synchronization signal (via pin FSC)

or up to 32 time-slots

– of constant width (8 bits), and
– of programmable location with respect to the frame synchronization signal (via pin FSC)

can be selected.

The time-slot locations can be programmed independently for receive and transmit direction via TSAX/TSAR and PCMMTX/PCMMRX registers.

Depending on the value programmed via those registers, the receive/transmit time-slot starts with a delay of 1 (minimum delay) up to 1024 clock periods following the frame synchronization signal.

Figure 47 shows how to select a time-slot of programmable width and location and **Figure 48** shows how to select one or more time-slots of 8-bit width.

If bit ’TOE’ in register CCR0 is set, the selected transmit time-slot(s) is(are) indicated at an output status signal via pin TxCLK, which is driven to ‘low’ during the active transmit window.

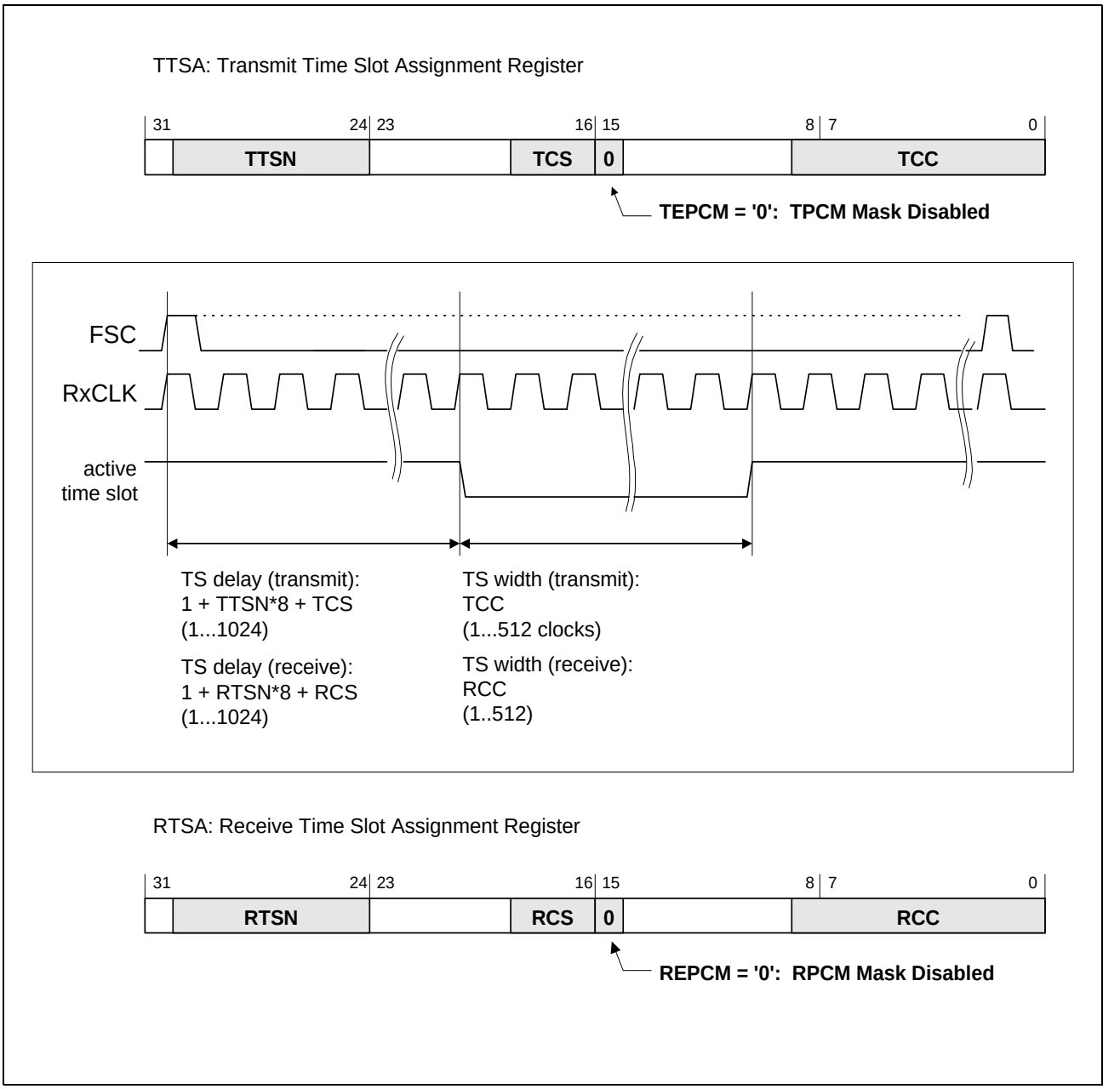


Figure 47 Selecting one time-slot of programmable delay and width

*Note: If time-slot 0 is to be selected, the DELAY has to be as long as the PCM frame itself to achieve synchronization (at least for the 2nd and subsequent PCM frames): DELAY = PCM frame length = 1 + xTSN*8 + xCS. xTSN and xCS have to be set appropriately.*

Example: Time-slot 0 in E1 (2.048 Mbit/s) system has to be selected.
*PCM frame length is 256 clocks. 256 = 1+ xTSN*8 + xCS. => xTSN = 31, xCS = 7.*

Note: In extended transparent mode the width xCC of the selected time-slot has to be n × 8 bit because of character synchronization (byte alignment). In all other modes the width can be used to define windows down to a minimum length of one bit.

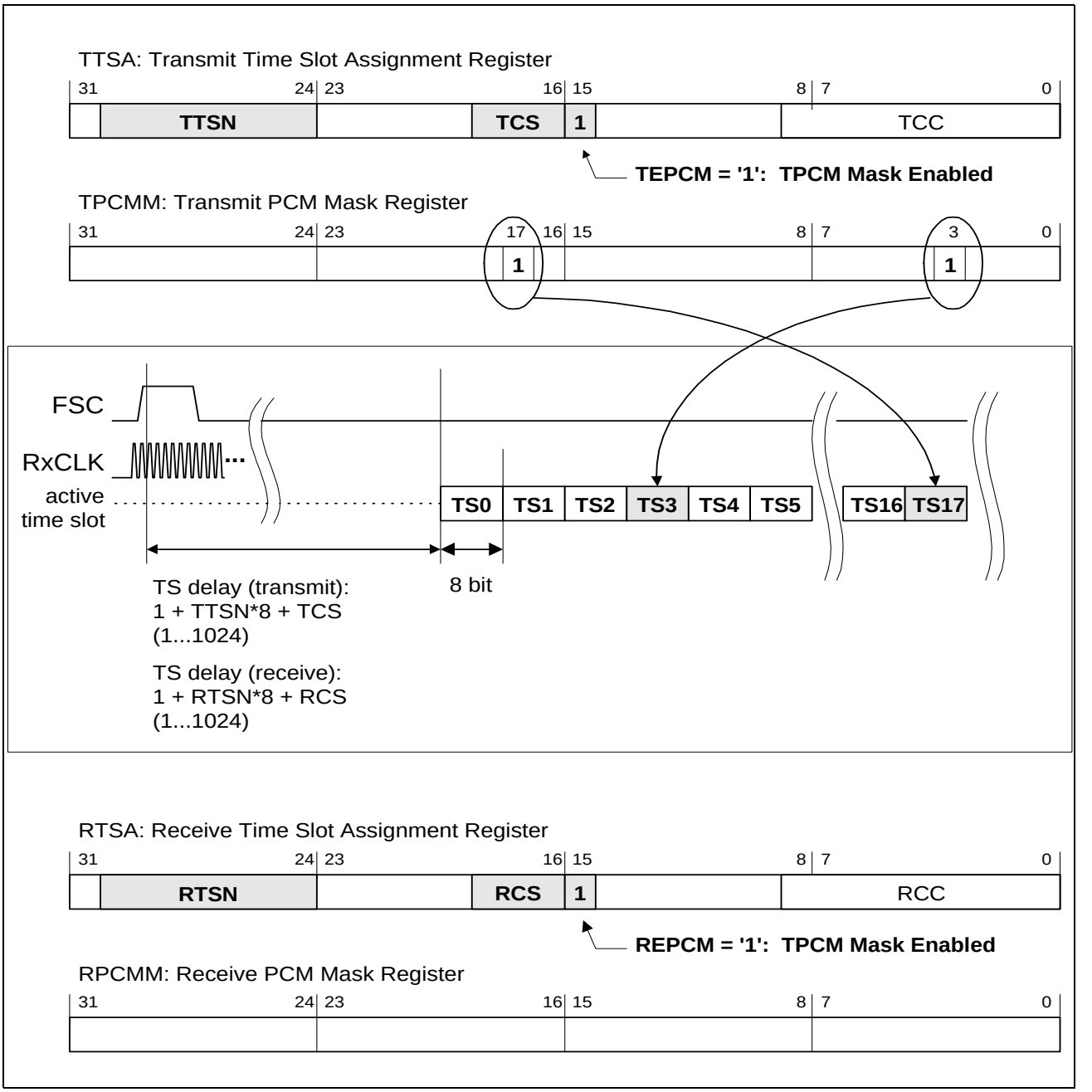


Figure 48 Selecting one or more time-slots of 8-bit width

The common transmit and receive clock is supplied at pin RxCLK and the common frame synchronisation signal at pin FSC. The "strobe signals" for active time slots are generated internally by the time slot assigner block (TSA) independent in transmit and receive direction.

When the transmit and receive PCM masks are enabled, bit fields ’TCC’ and ’RCC’ are ignored because of the constant 8-bit time slot width.

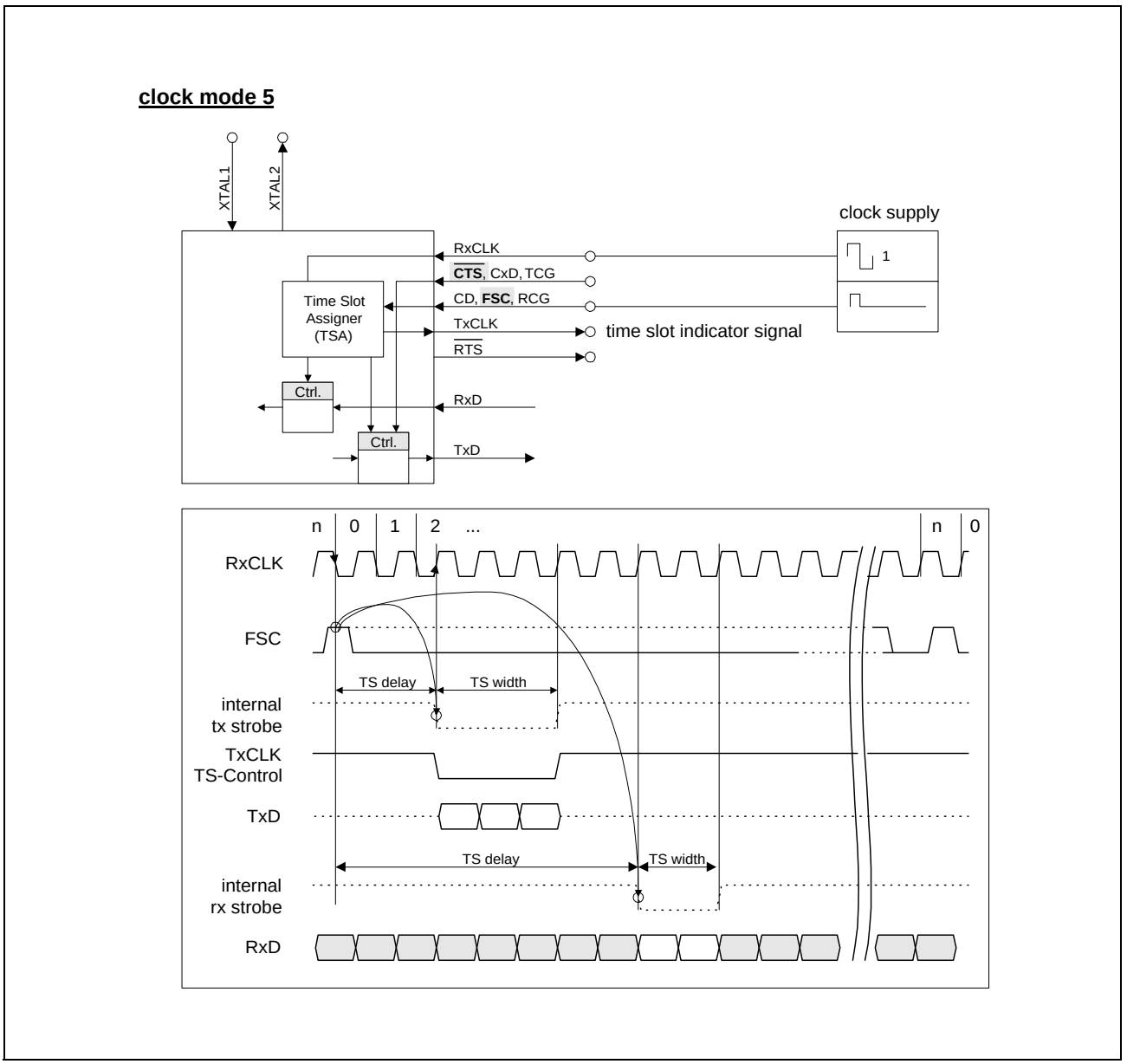


Figure 49 Clock Mode 5 Configuration

Note: The transmit time slot delay and width is programmable via bit fields 'TTSN', 'TCS' and 'TCC' in register TTSA.
The receive time slot delay and width is programmable via bit fields 'RTSN', 'RCS' and 'RCC' in register RTSA.

7.4.1.7 Clock Mode 6 (6a/6b)

This clock mode is identical to clock mode 2a/2b except that the clock source of the BRG is supplied at pin XTAL1.

The BRG is driven by the internal oscillator and delivers a reference clock for the DPLL which is 16 times the resulting DPLL output frequency which in turn supplies the internal receive clock. Depending on the programming of register CCR0 bit 'SSEL', the transmit clock will be either an external input clock signal provided at pin TxCLK in clock mode 6a or the clock delivered by the BRG divided by 16 in clock mode 6b. In the latter case, the transmit clock can be driven out to pin TxCLK if enabled via bit 'TOE' in register CCR2.

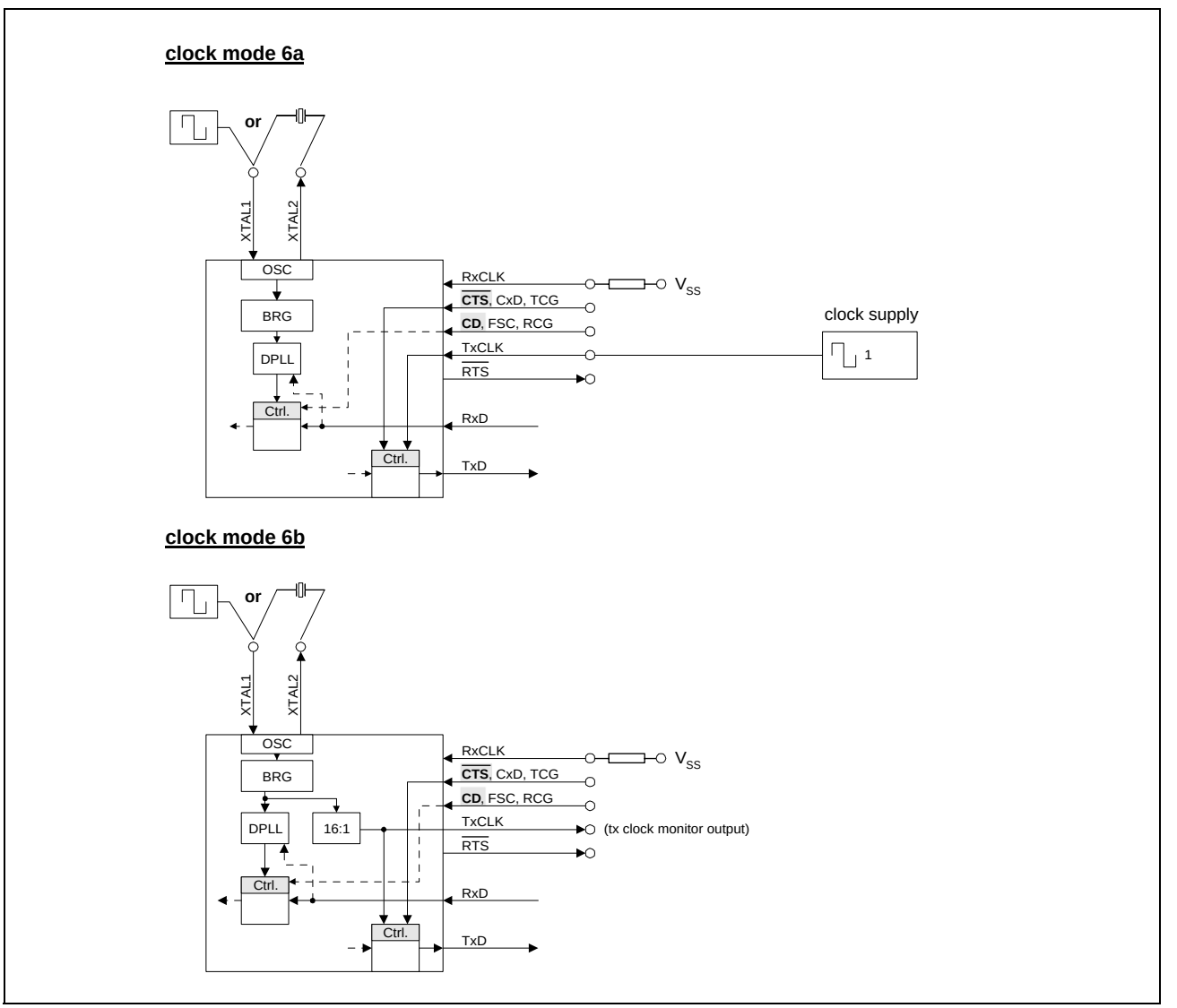


Figure 50 Clock Mode 6a/6b Configuration

7.4.1.8 Clock Mode 7 (7a/7b)

This clock mode is identical to clock mode 3a/3b except that the clock source of the BRG is supplied at pin XTAL1.

The BRG is driven by the internal oscillator. Depending on the value of bit 'SSEL' in register CCR0 the BRG delivers either a reference clock for the DPLL which is 16 times the resulting DPLL output frequency (clock mode 7a) or delivers directly the receive and transmit clock (clock mode 7b). In clock mode 7a the DPLL output clocks receive and transmit data.

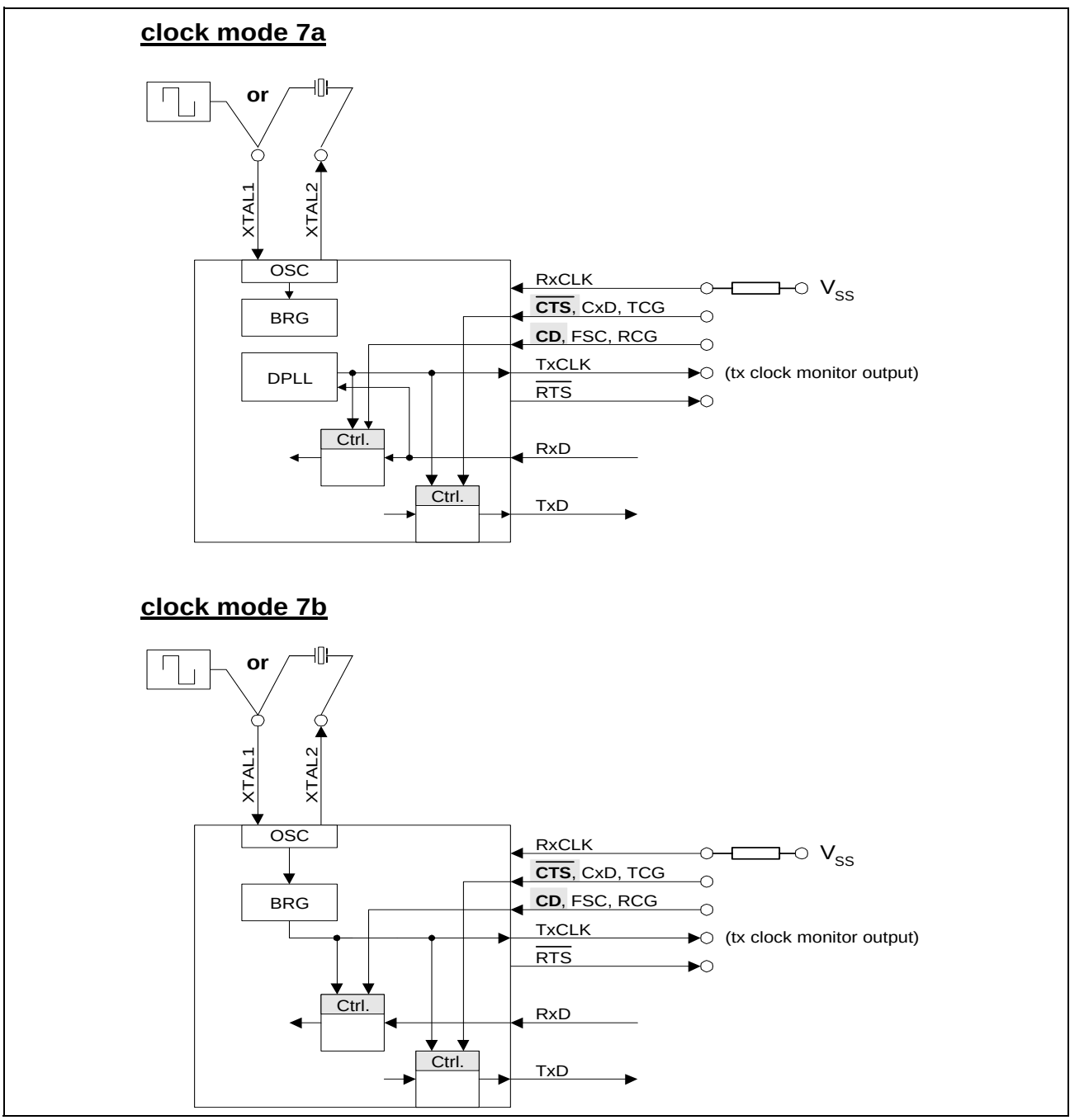


Figure 51 Clock Mode 7a/7b Configuration

7.4.2 Baud Rate Generator (BRG)

Each serial channel provides a baud rate generator (BRG) whose division factor is controlled by register BRR. The function of the BRG depends on the selected clock mode.

Table 22 BRR Register and Bit-Fields

Offset Addr.	Access Type	Controlled by	Reset Value	Register Name
$012C_H$ $01AC_H$ $022C_H$ $02AC_H$	r/w	CPU	00000000_H	**BRR**: Baud Rate Register
	Bit-Fields			
	Pos.	**Name**	**Default**	**Description**
	11..8	BRM	0	Baud Rate Factor M, range M = 0..15
	5..0	BRN	0	Baud Rate Factor N range N = 0..63

The clock division factor k is calculated by:

$$k = (N + 1) \times 2^M$$

$$f_{BRG} = f_{in} / k$$

7.4.3 Clock Recovery (DPLL)

The SCC offers the advantage of recovering the received clock from the received data by means of internal DPLL circuitry, thus eliminating the need to transfer additional clock information via a separate serial clock line. For this purpose, the DPLL is supplied with a ‘reference clock’ from the BRG which is 16 times the expected data clock rate (clock mode 2, 3a, 6, 7a). The transmit clock may be obtained by dividing the output of the BRG by a constant factor of 16 (clock mode 2b, 6b; bit ’SSEL’ in register CCR0 set) or also directly from the DPLL (clock mode 3a, 7a).

The main task of the DPLL is to derive a receive clock and to adjust its phase to the incoming data stream in order to enable optimal bit sampling.

The mechanism for clock recovery depends on the selected data encoding (see **“Data Encoding” on Page 162**).

The following functions have been implemented to facilitate a fast and reliable synchronization:

Interference Rejection and Spike Filtering

Two or more edges in the same directional data stream within a time period of 16 reference clocks are considered to be interference and consequently no additional clock adjustment is performed.

Phase Adjustment (PA)

Referring to **Figure 52**, **Figure 53** and **Figure 54**, in the case where an edge appears in the data stream within the PA fields of the time window, the phase will be adjusted by 1/16 of the data.

Phase Shift (PS) (NRZ, NRZI only)

Referring to **Figure 52** in the case where an edge appears in the data stream within the PS field of the time window, a second sampling of the bit is forced and the phase is shifted by 180 degrees.

Note: Edges in all other parts of the time window will be ignored.

This operation facilitates a **fast** and reliable synchronization for most common applications. Above all, it implies a very fast synchronization because of the phase shift feature: one edge on the received data stream is enough for the DPLL to synchronize, thereby eliminating the need for synchronization patterns, sometimes called preambles. However, in case of **extremely** high jitter of the incoming data stream the reliability of the clock recovery cannot be guaranteed.

The SCC offers the option to disable the Phase Shift function for NRZ and NRZI encodings by setting bit 'PSD' in register CCR0. In this case, the PA fields are extended as shown in **Figure 53**.

Now, the DPLL is more insensitive to high jitter amplitudes but needs **more time** to reach the optimal sampling position. To ensure correct data sampling, preambles should precede the data information.

Figure 52, **Figure 53** and **Figure 54** explain the DPLL algorithms used for the different data encodings.

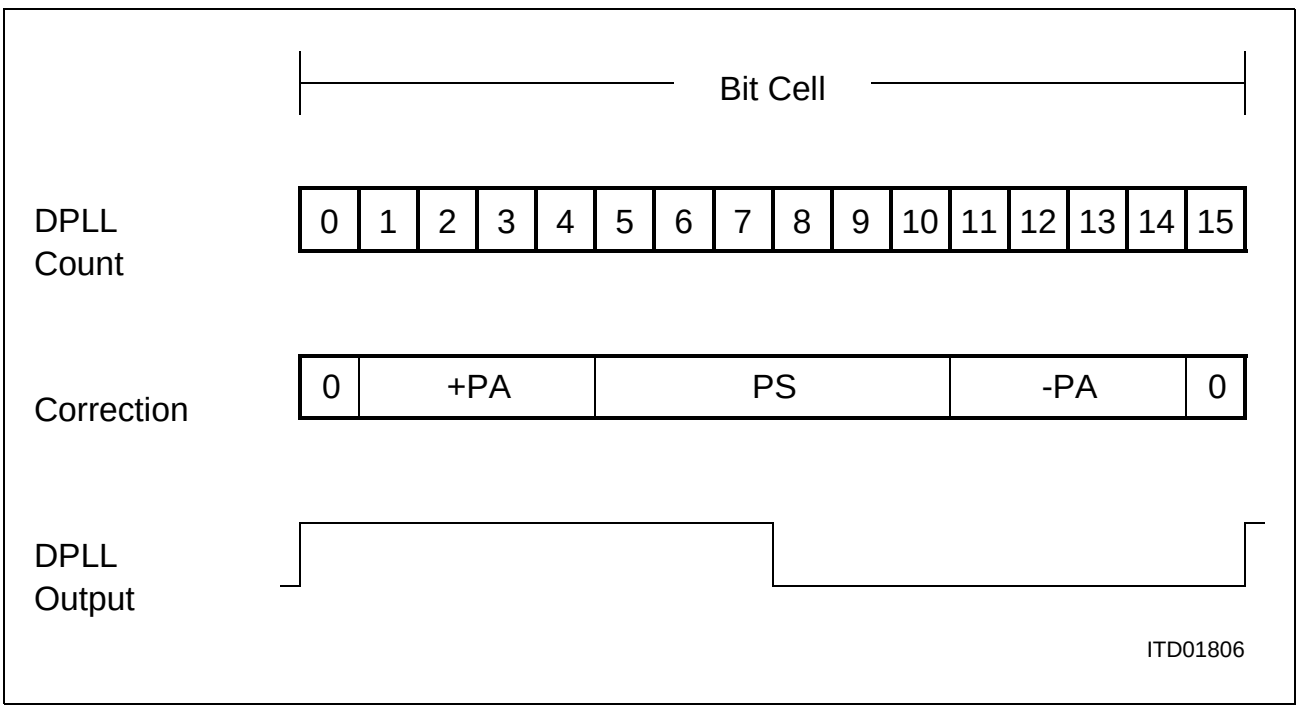


Figure 52 DPLL Algorithm for NRZ and NRZI Coding with Phase Shift Enabled (CCR0:PSD = '0')

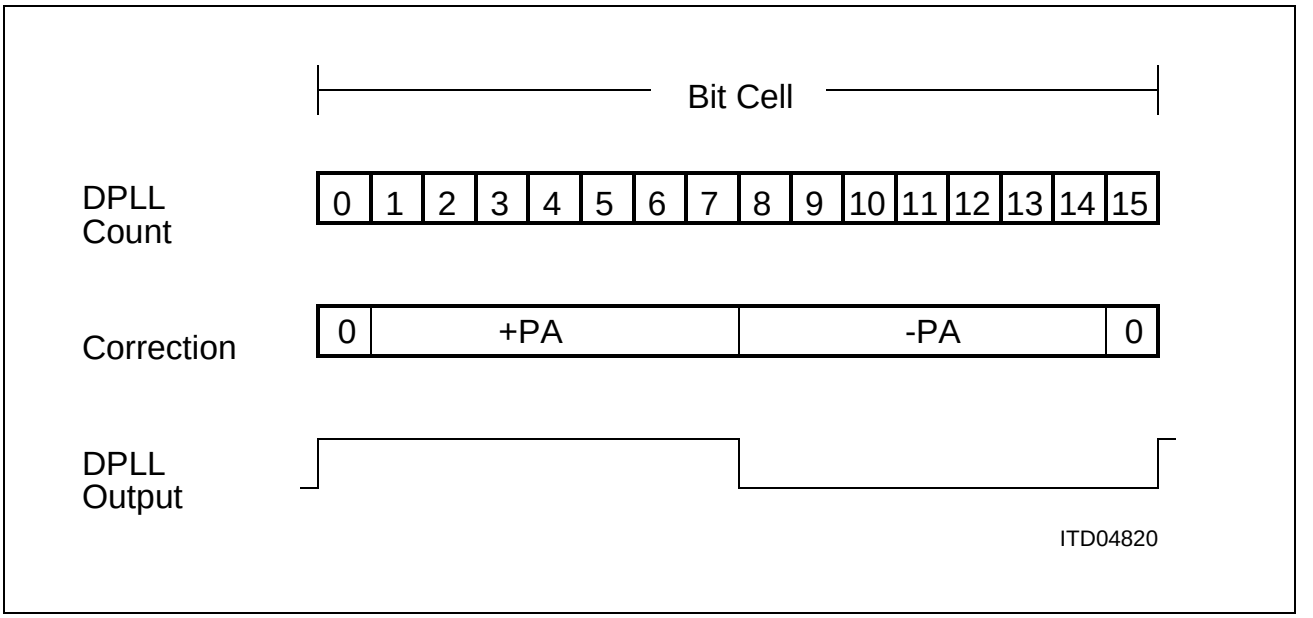


Figure 53 DPLL Algorithm for NRZ and NRZI Encoding with Phase Shift Disabled (CCR0:PSD = '1')

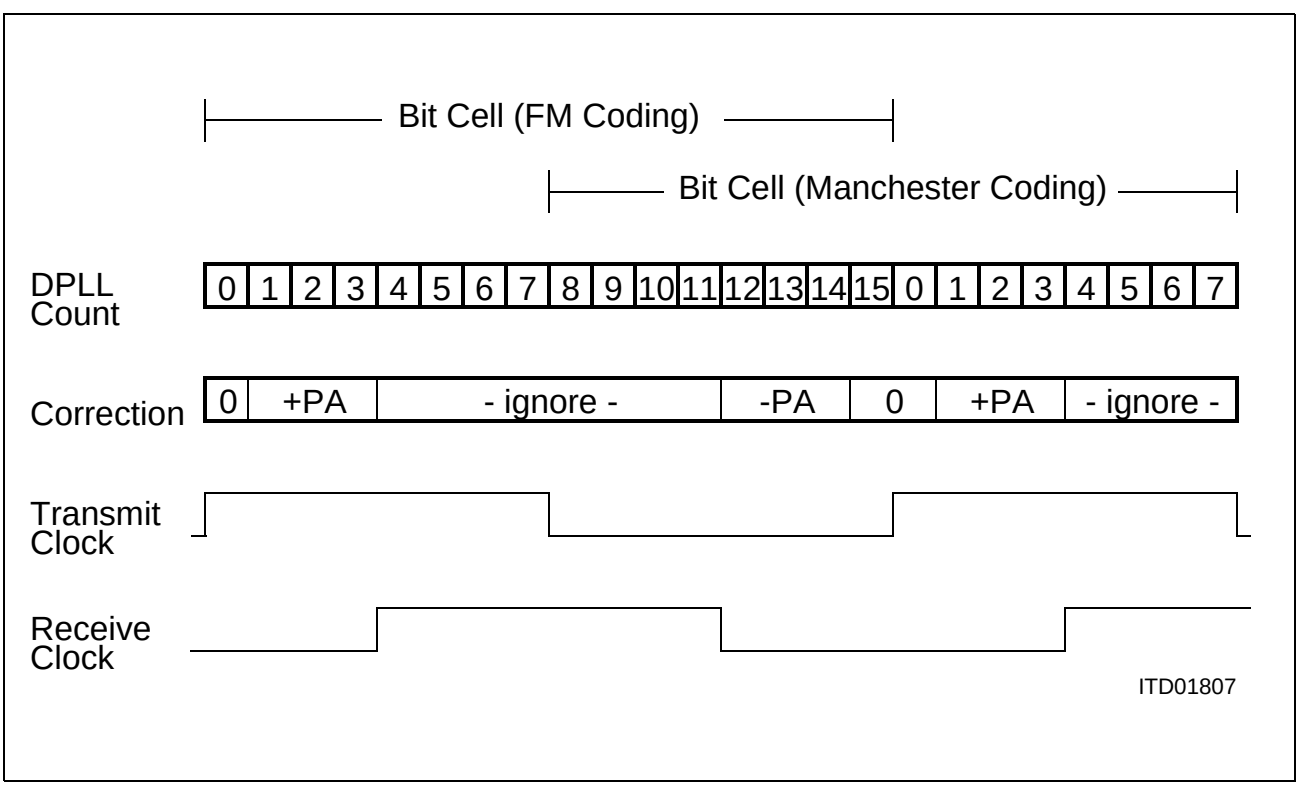


Figure 54 DPLL Algorithm for FM0, FM1 and Manchester Coding

To supervise correct function when using bi-phase encoding, a status flag and a maskable interrupt inform about synchronous/asynchronous state of the DPLL.

7.5 SCC Interrupt Interface

Special events in the SCC are indicated by interrupts either for receive direction or for transmit direction. Accordingly individual transmit and receive interrupt queues located in the shared memory are provided for each of the 4 SCCs. The interrupts generated by the SCC are forwarded as interrupt vectors to the channel specific and direction specific interrupt queues. See **“Interrupt Queue Overview” on Page 387** and **“SCC Interrupt Vector” on Page 393**.

Each interrupt indicated by the interrupt vector or interrupt status register (ISR) can selectively be masked (disabled) by setting the corresponding bit in the interrupt mask register IMR.

Note: A dedicated transmit interrupt vector may contain receive interrupt indications. In that case the receive interrupt indications can be ignored, since a dedicated receive interrupt vector is generated, additionally.

Note: A dedicated receive interrupt vector may contain transmit interrupt indications. In that case the transmit interrupt indications can be ignored, since a dedicated transmit interrupt vector is generated, additionally.

In interrupt visible mode (CCR0:VIS = ’1’), masked interrupt status bits neither activate the interrupt signal nor generate an interrupt vector, but are indicated in the respective interrupt status register ISR.

This mode is useful when some interrupt status bits are to generate an interrupt vector and other status bits are to be polled in the individual interrupt status register.

When using visible mode, only unmasked interrupt status bits are reset when the interrupt status register is read.

7.6 High Speed Channel Operation (PEB 20534H-52 only)

Any channel of the DSCC4 may be configured for operation at data rates up to 52 MBit/s. To configure one SCC for high speed operation, bit 'HS' in register CCR0 has to be set (allows data rates above 10 MBit/s) and clock mode 4 has to be selected to configure the internal clock unit appropriately.

The protocols supported at this high data rate are limited to HDLC address mode 0, PPP and extended transparent mode (PPP modes up to 45 MBit/s).

The high speed mode will operate with clocks provided on TxCLK (transmitter) and RxCLK (receiver). Clock gating is supplied on $\overline{TCG}$ and $\overline{RCG}$ pins to allow external transceivers (HSSI, DS3, etc.) to enable/disable frame/block bursts. When the clock gating signals are active, the SCC will not latch data from pin RxD nor will it output data on pin TxD (idle '1' is transmitted when $\overline{TCG}$ is active).

Beside clock gating via signals $\overline{TCG}$/$\overline{RCG}$, clock gapping is also supported in high speed mode.

In transmit direction one clock delay is inserted between detecting an active transmit gate signal $\overline{TCG}$ and the first transmit data bit driven to the line.

A transmit clock which is in phase to the data bits on pin TxD can be provided on pin $\overline{RTS}$ by setting bit 'TxCLKO' in register CCR1.

Depending on the expected data traffic the user can select appropriate depth for the central transmit FIFO. In addition a second transmit FIFO threshold (register FIFOCR4) controls data transfer to the SCC to prevent data underrun conditions when a new frame is started.

The central receive FIFO is allocated dynamically by the SCC as the data are received on the serial line.

7.7 Serial Bus Configuration Mode

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

In a pt-mpt configuration, comprising a central station (master) and several peripheral stations (slaves), or in a multimaster configuration, data transmission can be initiated by each station over a common transmit line (bus). In case more than one station attempts to transmit data simultaneously (collision), the bus has to be assigned to only one station.

– In HDLC/SDLC mode, a collision-resolution procedure is implemented by the SCC. Bus assignment is based on a priority mechanism with rotating priorities. This allows each station a bus access within a predetermined maximum time delay (deterministic CSMA/CD), no matter how many transmitters are connected to the serial bus.
– In BISYNC mode, the collision-resolution is implemented by the microprocessor.

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

Note: In high speed operation the usage of bus configuration is not supported.

Prerequisites for bus operation are:

- NRZ encoding
- ‘OR’ing of data from every transmitter on the bus (this can be realized as a wired-OR, using the TxD open drain capability)
- Feedback of bus information (CxD input).

The bus configuration is selected via register CCR0.

Note: Central clock supply for each station is not necessary if both the receive and transmit clock is recovered by the DPLL (clock modes 3a, 7a). This minimizes the phase shift between the individual transmit clocks.

The bus configuration mode operates independently of the clock mode, e.g. also together with clock mode 1 (receive and transmit strobe operation).

7.7.1 Serial Bus Access Procedure

The idle state of the bus is identified by eight or more consecutive ‘1’s. When a device starts transmission of a frame, the bus is recognized to be busy by the other devices at the moment the first ‘zero’ is transmitted (e.g. first ‘zero’ of the opening flag in HDLC mode).

After the frame has been transmitted, the bus becomes available again (idle).

Note: If the bus is occupied by other transmitters and/or there is no transmit request in the SCC, logical ‘1’ will be continuously transmitted on TxD.

7.7.2 Serial Bus Collisions and Recovery

During the transmission, the data transmitted on TxD is compared with the data on CxD. In case of a mismatch (‘1’ sent and ‘0’ detected, or vice versa) data transmission is immediately aborted, and idle (logical ‘1’) is transmitted.

HDLC/SDLC: Transmission will be initiated again by the SCC as soon as possible if the first part of the frame is still present in the SCC transmit FIFO. If not, an XMR interrupt is generated.

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

BISYNC: Transmitter and SCC transmit FIFO are reset and pin TxD goes to ‘1’. The XMR interrupt is provided which requests the microprocessor to repeat the whole message or block of characters.

ASYNC: Bus configuration is not recommended.

Note: If a wired-OR connection has been realized by an external pull-up resistor without decoupling, the data output (TxD) can be used as an open drain output and connected directly to the CxD input.
For correct identification as to which frame is aborted and thus has to be repeated after an XMR interrupt has occurred, the contents of SCC transmit FIFO have to be unique, i.e. SCC transmit FIFO as well as the central transmit FIFO should not contain data of more than one frame. For this purpose new data my be provided to the DMA controller only after ’ALLS’ interrupt status is detected.

7.7.3 Serial Bus Access Priority Scheme

To ensure that all competing stations are given a fair access to the transmission medium. Once a station has successfully completed the transmission of a frame, it is given a lower level of priority. This priority mechanism is based on the requirement that a station may attempt transmitting only when a determined number of consecutive ‘1’s are detected on the bus.

Normally, a transmission can start when eight consecutive ‘1’s on the bus are detected (through pin CxD). When an HDLC frame has been successfully transmitted, the internal priority class is decreased. Thus, in order for the same station to be able to transmit another frame, ten consecutive ‘1’s on the bus must be detected. This guarantees that the transmission requests of other stations are satisfied before the same station is allowed a second bus access. When ten consecutive ‘1’s have been detected, transmission is allowed again and the priority class (of all stations) is increased (to eight ‘1’s).

Inside a priority class, the order of transmission (individual priority) is based on the HDLC address, as explained in the preceding paragraph. Thus, when a collision occurs, it is always the station transmitting the only ‘zero’ (i.e. all other stations transmit a ‘one’) in a bit position of the address field that wins, all other stations cease transmission immediately.

7.7.4 Serial Bus Configuration Timing Modes

If a bus configuration has been selected, the SCC provides two timing modes, differing in the time interval between sending data and evaluation of the transmitted data for collision detection.

- Timing mode 1 (CCR0:SC1, SC0 = ‘01’)
  Data is output with the rising edge of the transmit clock via the TxD pin, and evaluated 1/2 a clock period later at the CxD pin with the falling clock edge.

- Timing mode 2 (CCR0:SC1, SC0 = ‘11’)
  Data is output with the falling clock edge and evaluated with the next falling clock edge. Thus one complete clock period is available between data output and collision detection.

7.7.5 Functions Of Signal $\overline{\text{RTS}}$ in Serial Bus Configuration

In clock modes 0 and 1, the $\overline{\text{RTS}}$ output can be programmed via register CCR1 (SOC bits) to be active when data (frame or character) is being transmitted. This signal is delayed by one clock period with respect to the data output TxD, and marks all data bits that could be transmitted without collision (see **Figure 55**). In this way a configuration may be implemented in which the bus access is resolved on a local basis (collision bus) and where the data are sent one clock period later on a separate transmission line.

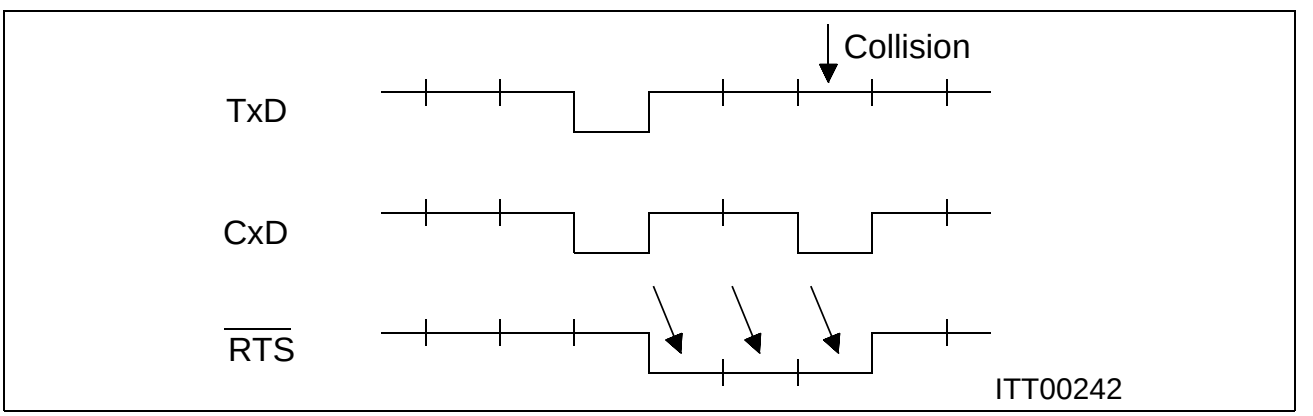


Figure 55 Request-to-Send in Bus Operation

Note: For details on the functions of the $\overline{\text{RTS}}$ pin refer to ***“Modem Control Signals (RTS, CTS, CD)” on Page 164****.*

7.8 Data Encoding

The SCC supports the following coding schemes for serial data:

– Non-Return-To-Zero (NRZ)
– Non-Return-To-Zero-Inverted (NRZI)
– FM0 (also known as Bi-Phase Space)
– FM1 (also known as Bi-Phase Mark)
– Manchester (also known as Bi-Phase)

7.8.1 NRZ and NRZI Encoding

NRZ: The signal level corresponds to the value of the data bit. By programming bit DIV (CCR1 register), the SCC may invert the transmission and reception of data.

NRZI: A logical ‘0’ is indicated by a transition and a logical ‘1’ by no transition at the beginning of the bit cell.

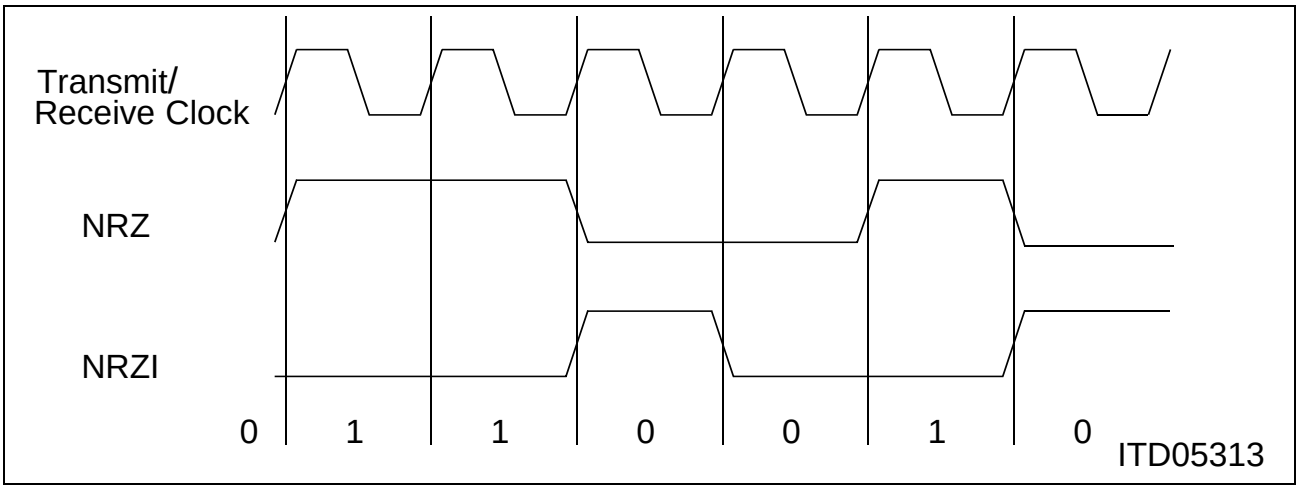


Figure 56 NRZ and NRZI Data Encoding

7.8.2 FM0 and FM1 Encoding

FM0: An edge occurs at the beginning of every bit cell. A logical ‘0’ has an additional edge in the center of the bit cell, whereas a logical ‘1’ has none. The transmit clock precedes the receive clock by 90°.

FM1: An edge occurs at the beginning of every bit cell. A logical ‘1’ has an additional edge in the center of the bit cell, a logical ‘0’ has none. The transmit clock precedes the receive clock by 90°.

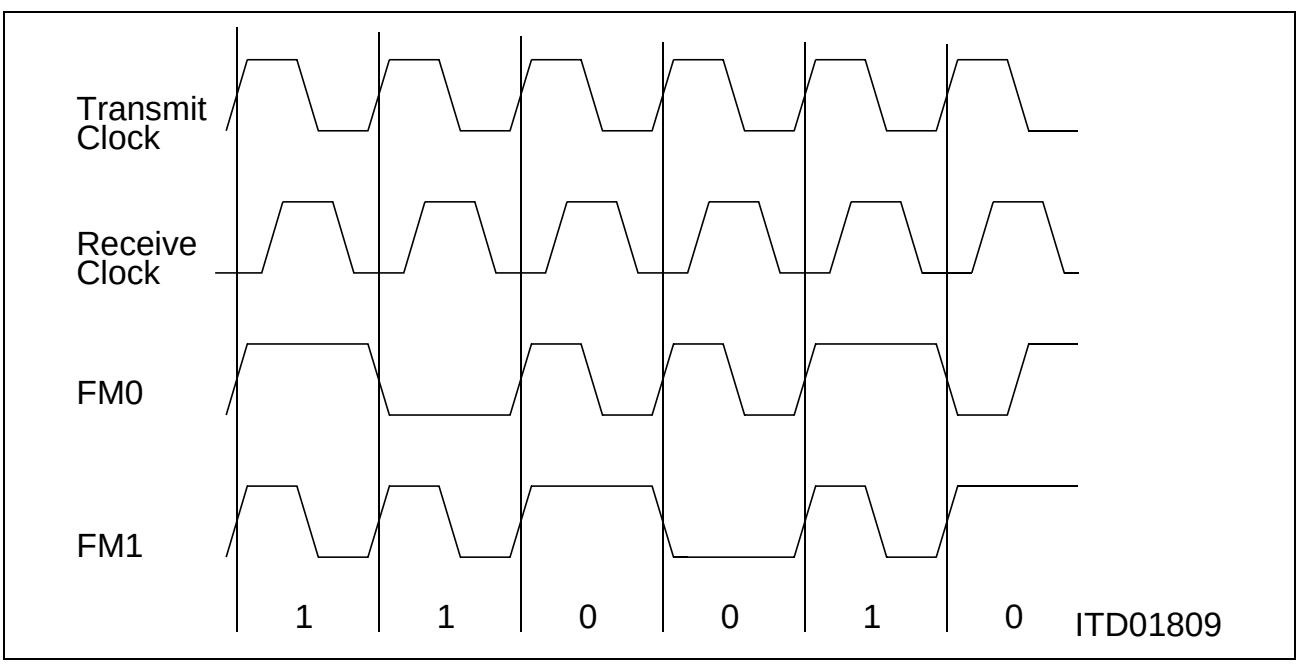


Figure 57 FM0 and FM1 Data Encoding

7.8.3 Manchester Encoding

Manchester: In the first half of the bit cell, the physical signal level corresponds to the logical value of the data bit. At the center of the bit cell this level is inverted. The transmit clock precedes the receive clock by 90°. The bit cell is shifted by 180° in comparison with FM coding.

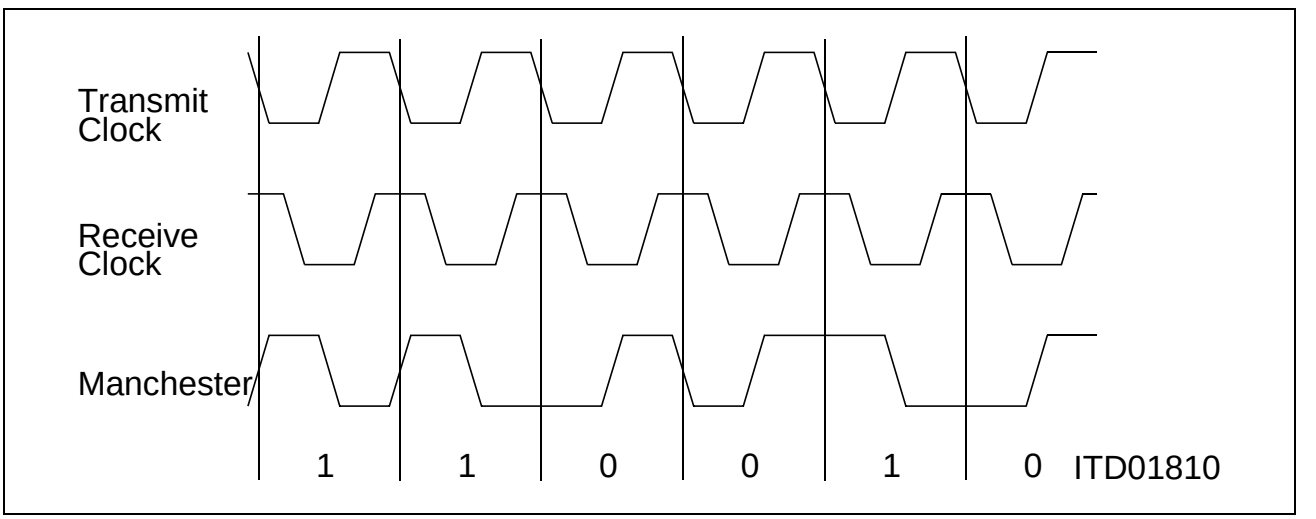


Figure 58 Manchester Data Encoding

7.9 Modem Control Signals ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$, CD)

7.9.1 $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ Handshaking

The SCC provides two pins ($\overline{\text{RTS}}$, $\overline{\text{CTS}}$) per serial channel supporting the standard request-to-send modem handshaking procedure for transmission control.

A transmit request will be indicated by outputting logical ‘0’ on the request-to-send output ($\overline{\text{RTS}}$). It is also possible to control the $\overline{\text{RTS}}$ output by software. After having received the permission to transmit ($\overline{\text{CTS}}$) the SCC starts data transmission.

HDLC/SDLC and **BISYNC:** In the case where permission to transmit is withdrawn in the course of transmission, the frame is aborted and IDLE is sent. After transmission is enabled again by re-activation of $\overline{\text{CTS}}$, and if the beginning of the frame is still available in the SCC, the frame will be re-transmitted (self-recovery). However, if the permission to transmit is withdrawn after the data available in the shadow part of the SCC transmit FIFO has been completely transmitted and the pool is released, the transmitter and the SCC transmit FIFO are reset, the $\overline{\text{RTS}}$ output is deactivated and an interrupt (XMR) is generated.

Note: For correct identification as to which frame is aborted and thus has to be repeated after an XMR interrupt has occurred, the contents of SCC transmit FIFO *have to be unique, i.e. SCC transmit FIFO as well as central transmit FIFO should not contain data of more than one frame, which could happen if transmission of a new*

frame is started by providing new data to the DMA controller too early. For this purpose the All Sent interrupt (ISR.ALLS) has to be waited for before forwarding new data to the DMA controller.

ASYNC: In the case where permission to transmit is withdrawn, transmission of the current character is completed. After that, IDLE is sent. After transmission is enabled again by re-activation of $\overline{\text{CTS}}$, the next available character is sent out.

Note: In the case where permission to transmit is not required, the $\overline{CTS}$ input can be connected directly to V_{SS}.

Additionally, any transition on the $\overline{\text{CTS}}$ input pin will generate an interrupt indicated via register ISR, if this function is enabled by setting the 'CSC' bit in register IMR to '0'.

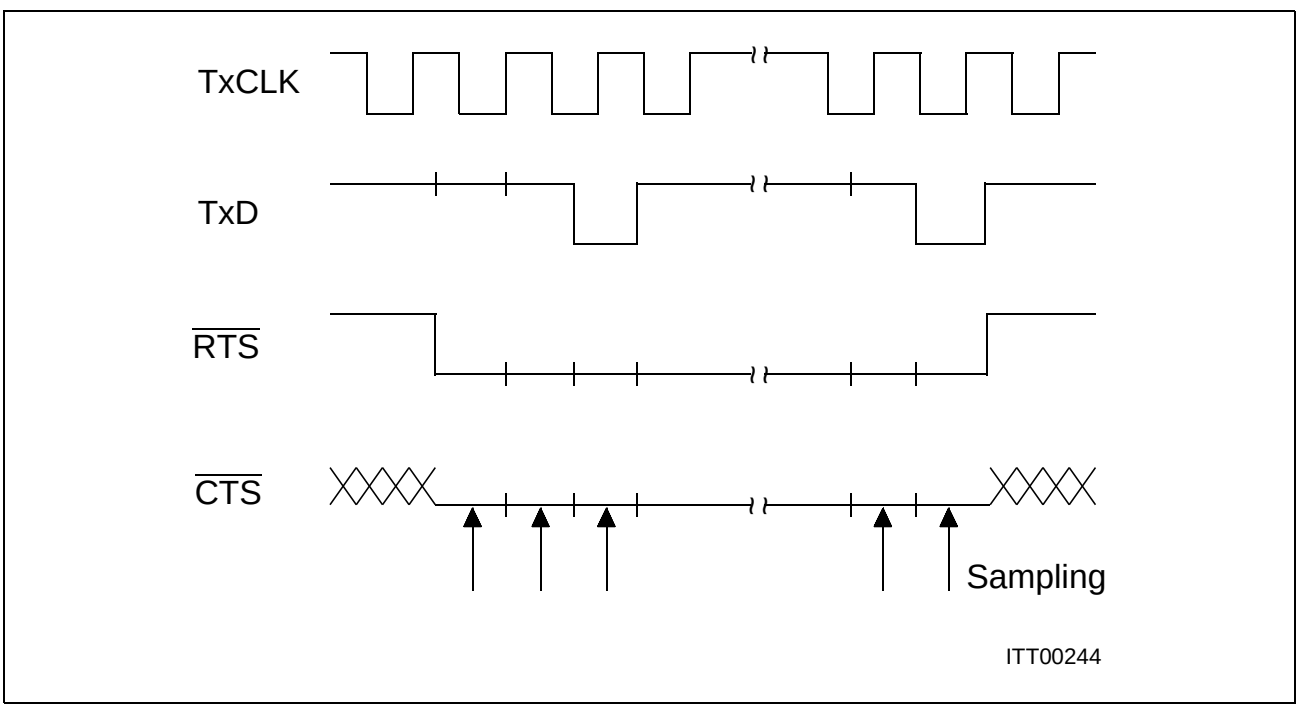


Figure 59 $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ Handshaking

Beyond this standard $\overline{\text{RTS}}$ function, signifying a transmission request of a frame (Request To Send), in HDLC mode the $\overline{\text{RTS}}$ output may be programmed for a special function via SOC1, SOC0 bits in the CCR1 register. This is only available if the serial channel is operating in a bus configuration mode in clock mode 0 or 1.

- If SOC1, SOC0 bits are set to '11', the $\overline{\text{RTS}}$ output is active (= low) during the reception of a frame.
- If SOC1, SOC0 bits are set to '10', the $\overline{\text{RTS}}$ output function is disabled and the $\overline{\text{RTS}}$ pin remains always high.

7.9.2 Carrier Detect (CD) Receiver Control

Similar to the $\overline{\text{RTS}}$/$\overline{\text{CTS}}$ control for the transmitter, the SCC supports the carrier detect modem control function for the serial receiver if the Carrier Detect Auto Start (CAS) function is programmed by setting the 'CAS' bit in register CCR1. This function is always available in clock modes 0, 2, 3, 6, 7 via the CD pin. In clock mode 1 the CD function is not supported. See **Table 21** for an overview.

If the CAS function is selected, the receiver is enabled and data reception is started when the CD input is detected to be high. If CD input is set to 'low', reception of the current character (byte) is still completed.

7.10 Local Loop Test Mode

To provide fast and efficient testing, the SCC can be operated in a test mode by setting the 'TLP' bit in register CCR1. The on-chip serial data input and output signals (TxD,RxD) are connected, generating a local loopback. As a result, the user can perform a self-test of the SCC.

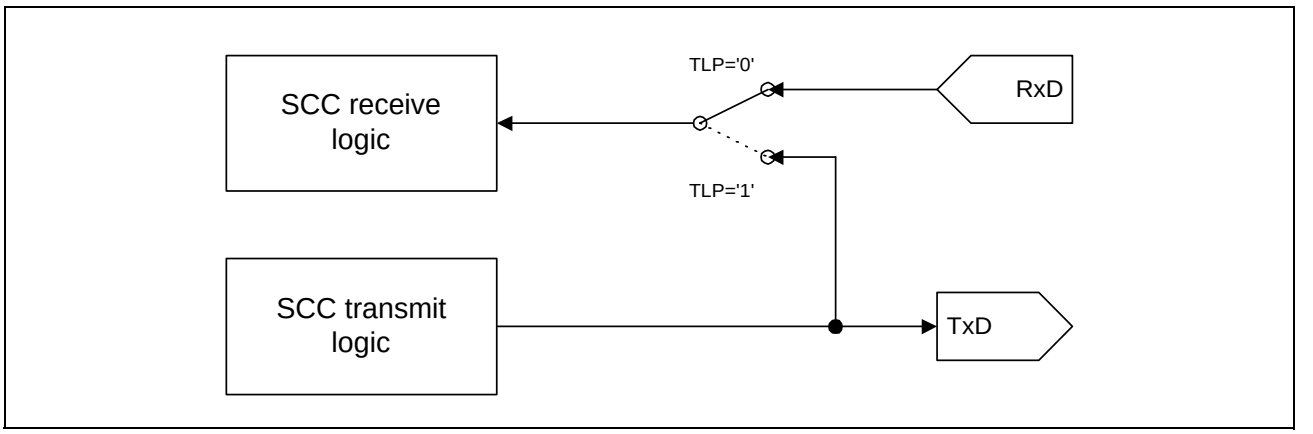


Figure 60 SCC Test Loop

Note: The transmit data is not disconnected from pin TxD during test loop operation, i.e. transmit data is always provided to pin TxD.

Note: An sufficient clock mode must be used for test loop operation such that receiver and transmitter operate with the same frequencies depending on the clock supply (e.g. clock mode 2b or 6b).

Note: Test loop operation is not supported in high speed channel operation (clock mode 4).

8 Detailed Protocol Description

The following **Table 23** provides an overview of all supported protocol modes and their assignment to three major protocol engines HDLC, ASYNC, BISYNC.The protocol engine of each SCC is selected via bit field ’SM’ in register CCR0. The HDLC Sub Modes are selected via additional bit fields in registers CCR0 and CCR1.

Table 23 Protocol Mode Overview

Protocol Engine:	Protocol Mode:			
Register CCR0 Setting:			**Register CCR1 Setting:**	
			bit fields MDS, ADM	**bit field PPPM**
HDLC/ SDLC SM = ’00’	HDLC auto mode	16 bit	MDS = ’00’ ADM = ’1’	PPPM = ’00’
		8 bit	MDS = ’00’ ADM = ’0’	
	HDLC non auto mode	16 bit	MDS = ’01’ ADM = ’1’	
		8 bit	MDS = ’01’ ADM = ’0’	
	HDLC address mode 1		MDS = ’10’ ADM = ’1’	
	HDLC address mode 0		MDS = ’10’ ADM = ’0’	
	PPP mode asynchronous		MDS=’10’ ADM=’0’	PPPM = ’10’
	PPP mode bit synchronous			PPPM = ’11’
	PPP mode octet synchronous			PPPM = ’01’
	Extended transparent mode[1)]		MDS = ’11’ ADM=’1’	PPPM = ’00’
			Register CCR0 Setting (bit BCR):	
ASYNC SM = ’11’	Asynchronous mode		BCR = ’1’	
	Isochronous mode		BCR = ’0’	
			Register CCR1 Setting (bit EBIM):	
BISYNC SM = ’10’	Bisynchronous mode		EBIM = ’1’	
	Monosynchronous mode		EBIM = ’0’	

1) Extended transparent is a fully bit-transparent transmit/reception mode which is treated as a sub-mode of the HDLC/SDLC block.

All modes are discussed in details in this chapter.

8.1 HDLC/SDLC Protocol Modes

The HDLC controller of each serial channel (SCC) can be programmed to operate in various modes, which are different in the treatment of the HDLC frame in receive direction. Thus, the receive data flow and the address recognition features can be performed in a very flexible way satisfying almost any application specific requirements.

There are 4 different HDLC operating modes which can be selected via register CCR1. Two more protocol modes (PPP, Extended Transparent Mode) are treated as sub-modes of the HDLC controller part.

8.1.1 HDLC

The following table provides an overview of the different address comparison mechanisms in HDLC operating modes:

Table 24 Address Comparison Overview

Mode	Address Field	High Address Byte	Low Address Byte
Auto Mode, Non-Auto Mode	16 bit	FE_H 1111 11 C/R 0_2	RAL1
			RAL2
		FC_H 1111 11 C/R 0_2	RAL1
			RAL2
		RAH1	RAL1
		RAH1	RAL2
		RAH2	RAL1
		RAH2	RAL2
	8 bit	RAL1	-
		RAL2	-
Address Mode 1	8 bit	FE_H	-
		FC_H	-
		RAH1	-
		RAH2	-
Address Mode 0	None	-	-

8.1.1.1 Auto Mode

Characteristics: Window size 1, random message length, address recognition.

The SCC processes autonomously all numbered frames (S-, I-frames) of an HDLC protocol. The HDLC control field, I-field data of the frames and an additional status byte are temporarily stored in the SCC receive FIFO.

Depending on the selected address mode, the SCC can perform a 2-byte or 1-byte address recognition. If a 2-byte address field is selected, the high address byte is compared with the fixed value FE_H or FC_H (group address) as well as with two individually programmable values in RAH1 and RAH2 registers. According to the ISDN LAPD protocol, bit 1 of the high byte address will be interpreted as COMMAND/ RESPONSE bit (C/R), dependent on the setting of the CRI bit in RAH1, and will be excluded from the address comparison.

Similarly, two comparison values can be programmed in special registers (RAL1, RAL2) for the low address byte. A valid address will be recognized in case the high and low byte of the address field correspond to one of the compare values. Thus, the SCC can be called (addressed) with 6 different address combinations, however, only the logical connection identified through the address combination RAH1, RAL1 will be processed in the auto-mode, all others in the non auto-mode. HDLC frames with address fields that do not match any of the address combinations, are ignored by the SCC.

In the case of a 1-byte address, RAL1 and RAL2 will be used as comparison values in the RADR register. According to the X.25 LAPB protocol, the value in RAL1 will be interpreted as COMMAND and the value in RAL2 as RESPONSE.

The address bytes can be masked to allow selective broadcast frame recognition. For further information see **“Receive Address Handling” on Page 185**.

8.1.1.2 Non Auto Mode

Characteristics: address recognition, arbitrary window size.

All frames with valid addresses (address recognition identical to auto-mode) are forwarded directly to the system memory.

The HDLC control field, I-field data and an additional status byte are temporarily stored in the SCC receive FIFO.

In non-auto-mode, all frames with a valid address are treated similarly.

The address bytes can be masked to allow selective broadcast frame recognition.

8.1.1.3 Address Mode 1

Characteristics: address recognition high byte.

Only the high byte of a 2-byte address field will be compared. The address byte is compared with the fixed value FE_H or FC_H (group address) as well as with two individually programmable values RAH1 and RAH2 in the RADR register. The whole frame excluding the first address byte will be stored in the SCC receive FIFO.

The address bytes can be masked to allow selective broadcast frame recognition.

8.1.1.4 Address Mode 0

Characteristics: no address recognition

No address recognition is performed and each complete frame will be stored in the SCC receive FIFO.

8.1.2 HDLC/PPP Protocol Mode

PPP (as described in RFC1662) can work over 3 modes: asynchronous HDLC, synchronous HDLC, and octet synchronous. The DSCC4 supports asynchronous HDLC PPP over ISDN or DDS circuits as well as synchronous HDLC PPP for use over dial-up connections. The octet synchronous mode of PPP protocol (RFC 1662) supports PPP over SONET applications.

Both the asynchronous HDLC PPP mode, as well as the synchronous HDLC PPP modes, are submodes of the HDLC mode. Either mode is selected by configuring the DSCC4 for the standard HDLC mode. In addition the appropriate PPP mode is selected via bit field ’PPPM’ in register CCR2.

The DSCC4 provides logic to convert an HDLC frame to an ASYNC character stream with the specified mapping functions. Layer 3 PPP functions are normally implemented in software.

The PPP-support hardware allows software to perform segmentation and reassembly of PPP payloads, and allows the DSCC4 to perform the asynchronous HDLC PPP **or** the synchronous HDLC PPP protocol conversions as required for the network interface.

8.1.2.1 Bit Synchronous PPP

The DSCC4 transfers a data block from the shared memory, inserts HDLC Header (Opening Flag), and appends the HDLC Trailer (CRC, Ending Flag). Zero-bit stuffing algorithm is also performed. No character mapping is performed. The bit-synchronous PPP mode differs from the HDLC mode (address mode 0) only in the abort sequence:

HDLC requires at least 7 ’1’ bit whereas PPP requires at least 15 ’1’ bit abort sequence.

For receive operation the DSCC4 monitors the incoming data stream for the Opening Flag (7E Hex) to identify the beginning of a HDLC packet. Subsequent bytes are part of data and are processed as normal HDLC packet including checking of CRC.

8.1.2.2 Octet Synchronous PPP

The DSCC4 transfers a data block from the shared memory, inserts HDLC Header (Opening Flag), and appends the HDLC Trailer (CRC, Ending Flag). Beside this standard HDLC operation, zero-bit stuffing is not performed, but character mapping is performed.

For receive operation the DSCC4 monitors the incoming data stream for the Opening Flag (7E Hex) to identify the beginning of a HDLC packet. Subsequent bytes are part of data and are processed as normal HDLC packet including checking of CRC. Received mapped characters are unmapped.

8.1.2.3 Aynchronous PPP

For transmit operation, the DSCC4 reads a data block from host memory, inserts the HDLC header (Opening Flag), and appends the HDLC trailer (CRC, Ending Flag). Each octet (including HDLC framing flags and idle flags) is converted into async character format (1 start, 8 data bits, 1 stop bit) and then transmitted using the asynchronous character formatter block.

In receive direction any async character is transferred into the DSCC4's ASYNC Character De-Formatting logic block, where it is translated back into the original information octet. The information octets are then transferred to host the memory as in HDLC address mode 0 operation.

8.1.3 Extended Transparent Mode

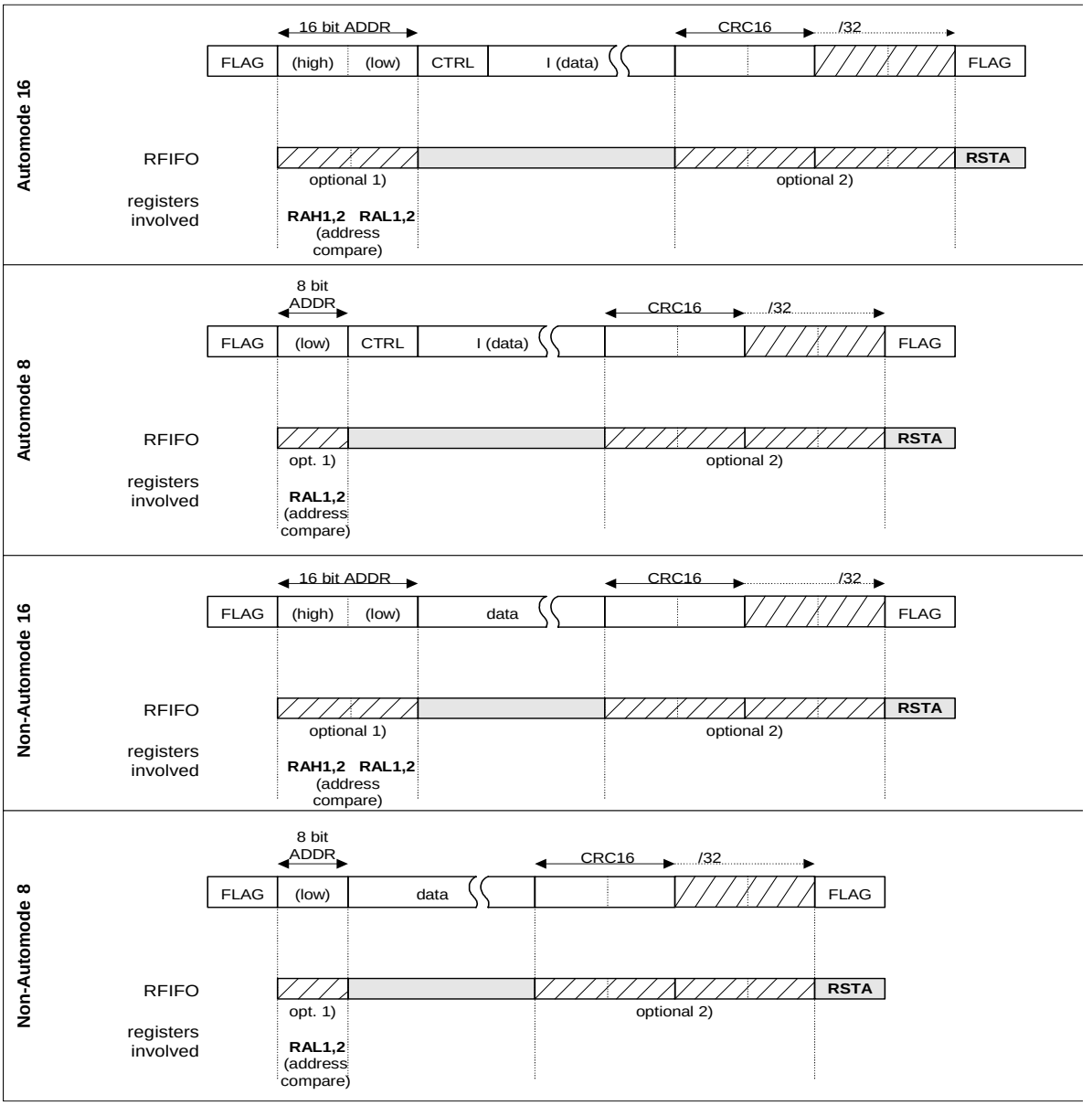
Characteristics: fully transparent

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

In clock mode 1 and clock mode 5 byte alignment is provided.

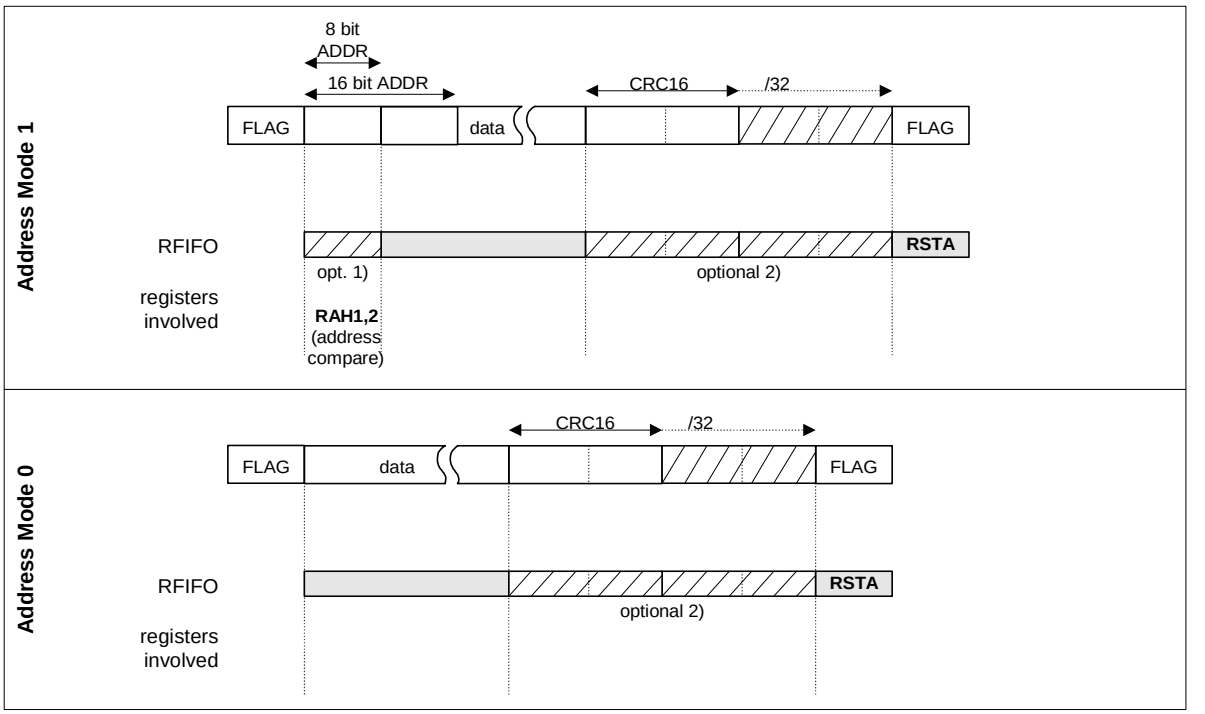
8.1.4 HDLC Receive Data Processing Overview

The following two figures give an overview about the management of the received frames in the different HDLC operating modes.



option1)
The 8 or 16 bit address field can optionally be forwarded to the RFIFO (bit 'RADD' in register CCR2)
option 2)
The 16 or 32 bit CRC field can optionally be forwarded to the RFIFO (bit 'RCRC' in register CCR2)

Figure 61 SCC Receive Data Flow (HDLC Modes) part a)



option1)
The address field (8 bit address, 16 bit address or the high byte of a 16 bit address) can optionally be forwarded to the RFIFO (bit 'RADD' in register CCR2)
option 2)
The 16 or 32 bit CRC field can optionally be forwarded to the RFIFO (bit 'RCRC' in register CCR2)

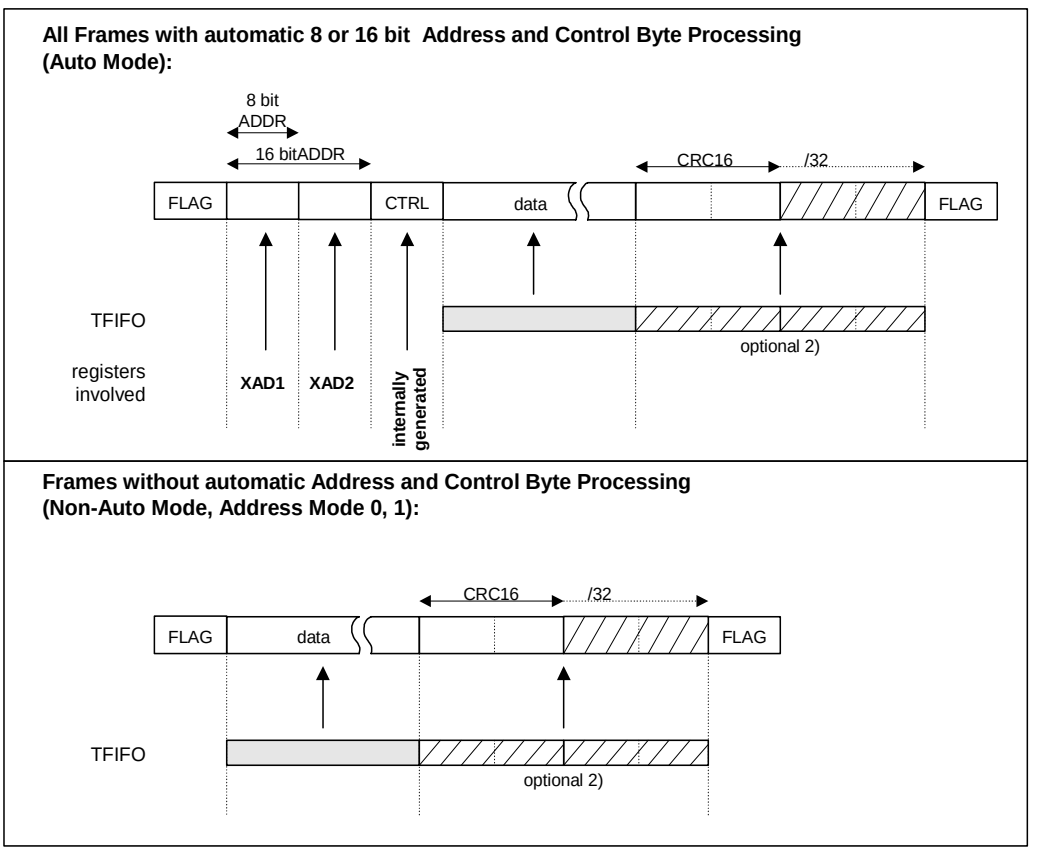
Figure 62 SCC Receive Data Flow (HDLC Modes) part b)

8.1.5 HDLC Transmit Data Processing Overview

Two different types of frames can be transmitted:

– I-frames and
– transparent frames

as shown below.



option 2)
Generation of the 16 or 32 bit CRC field can optionally be disabled by setting bit 'XCRC' in register CCR2, in which case the CRC must be calculated and written into the last 2 or 4 bytes of the transmit FIFO, to immediately proceed closing flag.

Figure 63 SCC Transmit Data Flow (HDLC Modes)

For transmission of I-frames (selected via bit ’SXIF’ in register CCR2), the address and control fields are generated autonomously by the SCC and the data in the corresponding transmit data buffer is entered into the information field of the frame. This is possible only if the SCC is operated in auto mode.

For (address) transparent frames, the address and the control fields have to be entered in the transmit data buffer by software. This is possible in all operating modes and used also in auto-mode for sending U-frames.

If bit ’XCRC’ in register CCR2 is set, the CRC checksum will not be generated internally. The checksum has to be provided via the transmit data buffer as the last two or four bytes by software. The transmitted frame will be closed automatically only with a (closing) flag.

Note: The SCC does not check whether the length of the frame, i.e. the number of bytes, to be transmitted makes sense according the HDLC protocol or not.

8.1.6 Procedural Support (Layer-2 Functions)

When operating in the auto mode, the SCC offers a high degree of protocol support. In addition to address recognition, the SCC autonomously processes all (numbered) S- and I-frames (window size 1 only) with either normal or extended control field format (modulo-8 or modulo-128 sequence numbers – selectable via register CCR1 bit 'MCS').

The following functions will be performed:

– updating of transmit and receive counter
– evaluation of transmit and receive counter
– processing of S commands
– flow control with RR/RNR
– generation of responses
– recognition of protocol errors
– transmission of S commands, if acknowledgement is not received
– continuous status query of remote station after RNR has been received
– programmable timer/repeater functions.

In addition, all unnumbered frames are forwarded directly to the processor. The logical link can be initialized by software at any time (Reset HDLC Receiver by RRES command in register CMDR).
Additional logical connections can be operated in parallel by software.

8.1.7 Full-Duplex LAPB/LAPD Operation

Initially (i.e. after RESET), the LAP controllers of the two serial channels are configured to function as a combined (primary/secondary) station, where they autonomously perform a subset of the balanced X.25 LAPB/ISDN LAPD protocol.

Reception of Frames:

The logical processing of received S-frames is performed by the SCC without interrupting the host. The host is merely informed by interrupt of status changes in the remote station (receiver ready / receiver not ready) and protocol errors (unacceptable N(R), or S-frame with I field).

I-frames are also processed autonomously and checked for protocol errors. The I-frame will not be accepted in the case of sequence errors (no interrupt is forwarded to the host), but is immediately confirmed by an S-response. If the host sets the SCC into a ‘receive not ready’ status, an I-frame will not be accepted (no interrupt) and an RNR response is transmitted. U-frames are always stored in the RFIFO and forwarded directly to the host. The logical sequence and the reception of a frame in auto mode is illustrated in **figure 64.**

Note: The state variables N(S), N(R) are evaluated within the window size 1, i.e. the SCC checks only the least significant bit of the receive and transmit counter regardless of the selected modulo count.

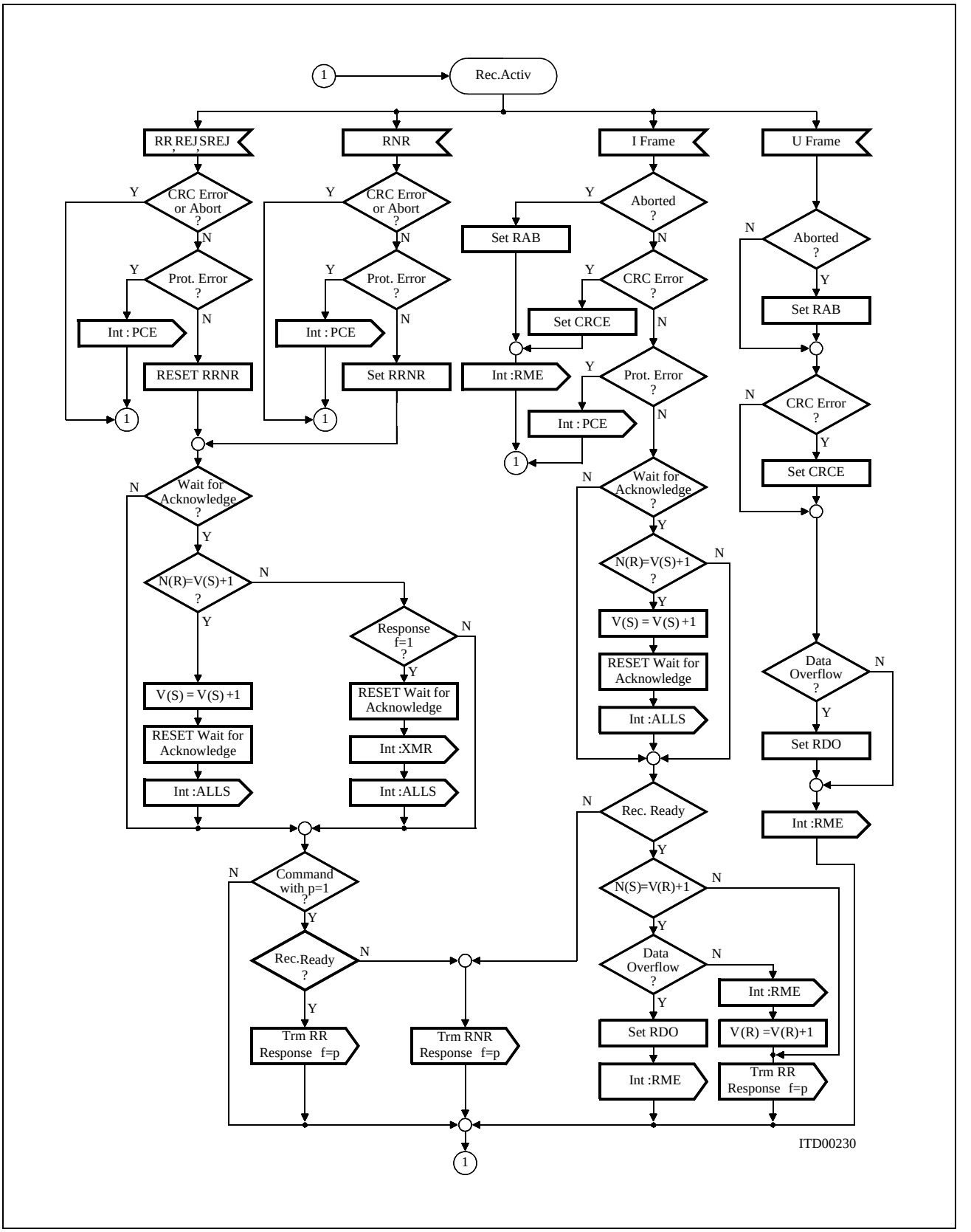


Figure 64 Processing of Received Frames in Auto Mode

Transmission of Frames:

The SCC autonomously transmits S commands and S responses in the auto mode. Either transparent or I-frames can be transmitted by the user. The software timer has to be operated in the internal timer mode to transmit I-frames. After the frame has been transmitted, the timer is self-started, the XFIFO is inhibited, and the SCC waits for the arrival of a positive acknowledgement. This acknowledgement can be provided by means of an S- or I-frame.

If no positive acknowledgement is received during time t_1, the SCC transmits an S-command (p = ‘1’), which must be answered by an S-response (f = ‘1’). If the S-response is not received, the process is performed n1 times (in HDLC known as N2, refer to register TIMR).

Upon the arrival of an acknowledgement or after the completion of this poll procedure the XFIFO is enabled and an interrupt is generated. Interrupts may be triggered by the following:

- message has been positively acknowledged (ALLS interrupt)
- message must be repeated (XMR interrupt)
- response has not been received (TIN interrupt).

In automode only when the ALLS interrupt has been issued data of a new frame may be provided to the DMA controller!

Upon arrival of an RNR frame, the software timer is started and the status of the remote station is polled periodically after expiration of t_1, until the status ‘receive ready’ has been detected. The user is informed via the appropriate interrupt. If no response is received after n1 times, a TIN interrupt, and t_1 clock periods thereafter an ALLS interrupt is generated and the process is terminated.

Note: The internal timer mode should only be used in the auto mode.

Transparent frames can be transmitted in all operating modes.

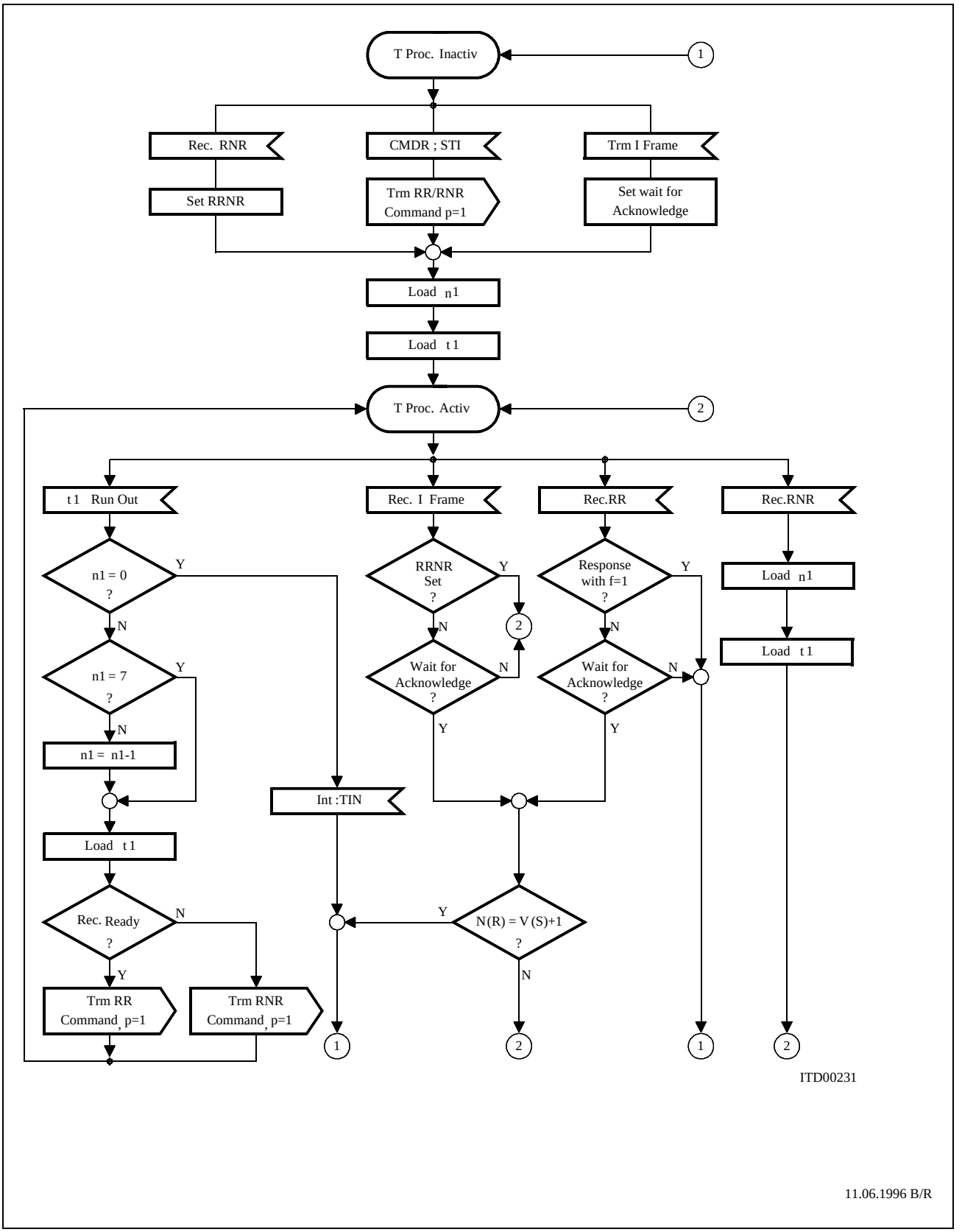


Figure 65 Timer Procedure/Poll Cycle

Examples

The interaction between SCC and the host during transmission and reception of I-frames is illustrated in the following two figures. The flow control with RR/RNR of I-frames during transmission/reception is illustrated in **figure 66**. Both, the sequence of the poll cycle and protocol errors are shown in **figure 67**.

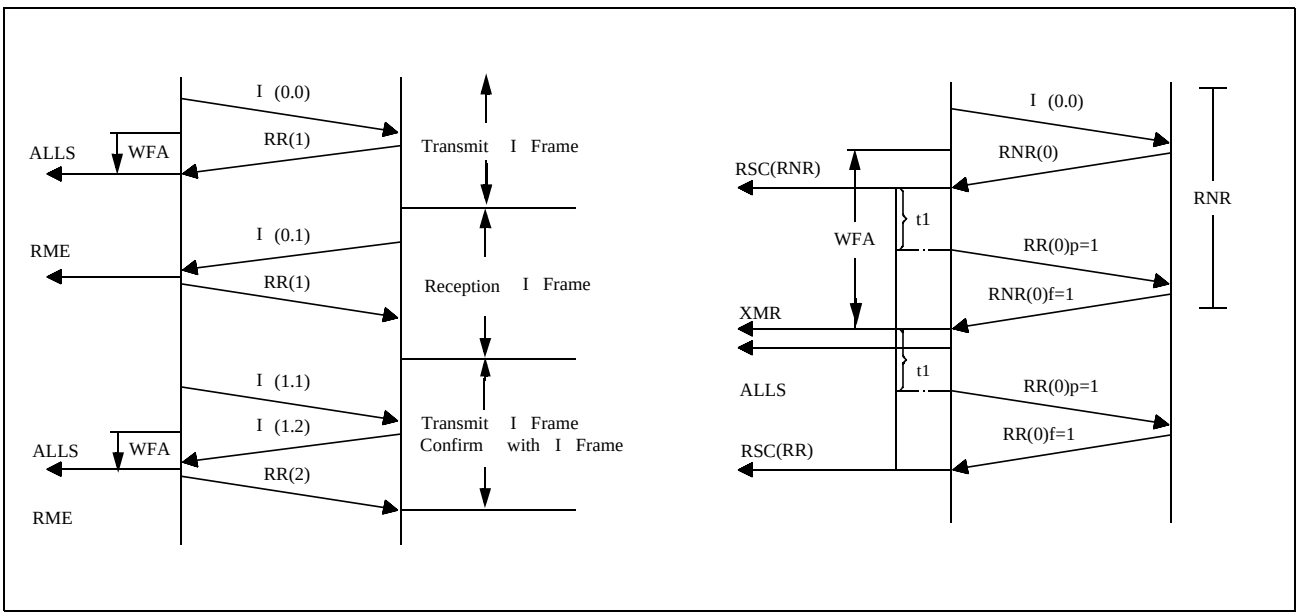


Figure 66 Transmission/Reception of I-Frames and Flow Control

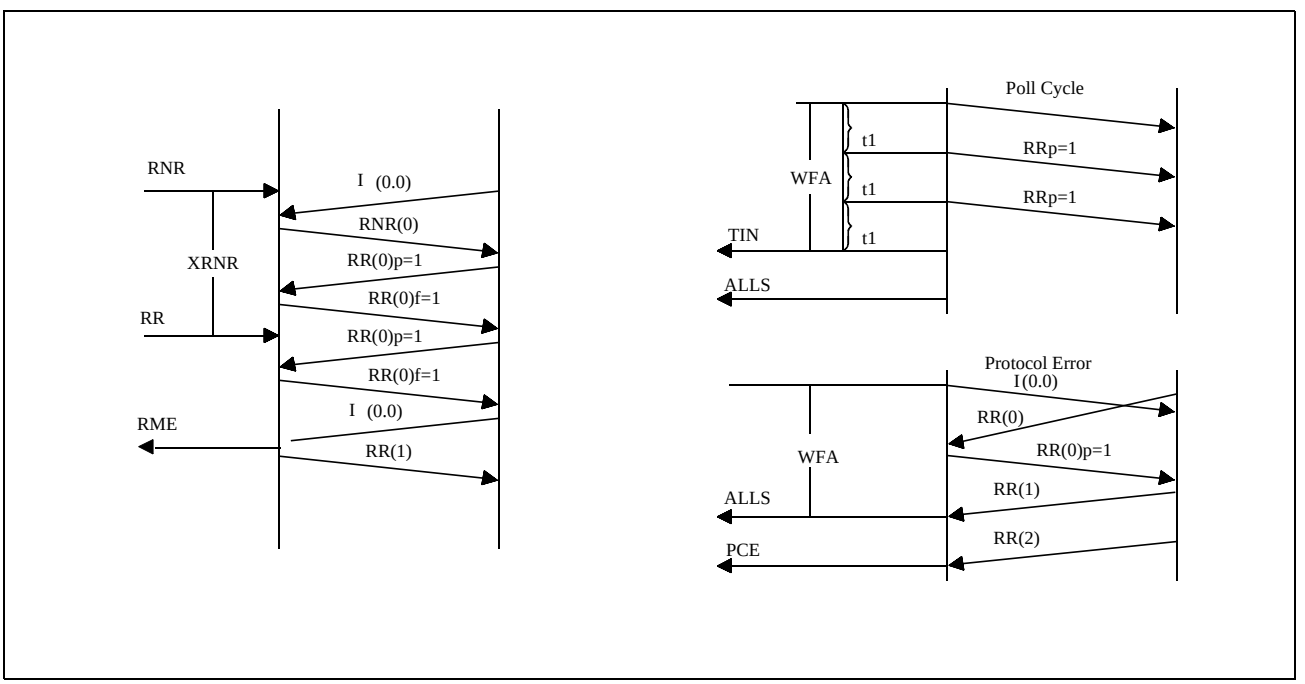


Figure 67 Flow Control: Reception of S-Commands and Protocol Errors

Protocol Error Handling:

Depending on the error type, erroneous frames are handled according to **table 25**.

Table 25 Error Handling

Frame Type	Error Type	Generated Response	Generated Interrupt	Rec. Status
I	CRC error aborted unexpec. N(S) unexpec. N(R)	– – S-frame –	FI (from DMAC) FI (from DMAC) – PCE	CRC error abort – –
S	CRC error aborted unexpec. N(R) with I-field	– – – –	– – PCE PCE	– – – –

Note: The station variables (V(S), V(R)) are not changed.

8.1.8 Half-Duplex SDLC-NRM Operation

The LAP controllers of the two serial channels can be configured to function in a half-duplex Normal Response Mode (NRM), where they operate as a slave (secondary) station, by setting the NRM bit in the CCR1 register of the corresponding channel.

In contrast to the full-duplex LAP B/LAP D operation, where the combined (primary + secondary) station transmits both commands and responses and may transmit data at any time, the NRM mode allows only responses to be transmitted **and** the secondary station may transmit only when instructed to do so by the master (primary) station. The SCC gets the permission to transmit from the primary station via an S-, or I-frame with the poll bit (p) **set**.

The NRM mode can be profitably used in a point-to-multipoint configuration with a fixed master-slave relationship, which guarantees the absence of collisions on the common transmit line. It is the responsibility of the master station to poll the slaves periodically and to handle error situations.

Prerequisite for NRM operation is:

– auto mode with 8-bit address field selected
  Register CCR1 bit fields MDS1, MDS0, ADM = ‘000’
– external timer mode
  Register TIMR bit TMD = ‘0’
– same transmit and receive addresses, since only responses can be transmitted, i.e.
  Register XADR bit fields XAD1 = XAD2 and register RADR bit fields RAL1 = RAL2 (address of secondary).

Note: The broadcast address may be programmed in bit field RAL2 if broadcasting is required.
In this case bit fields RAL1 and RAL2 are not equal.

The primary station has to operate in transparent HDLC mode.

Reception of Frames:

The reception of frames functions similarly to the LAPB/LAPD operation (see **“Full-Duplex LAPB/LAPD Operation” on Page 177**).

Transmission of Frames:

The SCC does **not** transmit S-, or I-frames if not instructed to do so by the primary station via an S-, or I-frame with the poll bit set.

The SCC can be prepared to send an I-frame by the host by setting bit ’SXIF’ in register CCR2. The transmission of the frame, however, will not be initiated by the SCC until reception of either an

- RR, or
- I-frame

with poll bit set (p = ‘1’).

After the frame has been transmitted (with the final bit set), the host has to wait for an ALLS or XMR interrupt.

Since the on-chip timer of the SCC must be operated in the external timer mode (a secondary does not poll the primary for acknowledgements), timer supervision must be done by the primary station.

Upon the arrival of an acknowledgement the SCC transmit FIFO is enabled and an interrupt is forwarded to the host, either the

– message has been positively acknowledged (ALLS interrupt), or the
– message must be repeated (XMR interrupt).

Additionally, the timer can be used **under host control** to provide timer recovery of the secondary if no acknowledgements are received at all.

Note: A secondary will transmit transparent frames only if the permission to send is given by receiving an S-frame or I-frame with poll bit set (p = ‘1’).

Examples:

A few examples of SCC/host interaction in the case of normal response mode (NRM) mode are shown in **figure 68** and **68**.

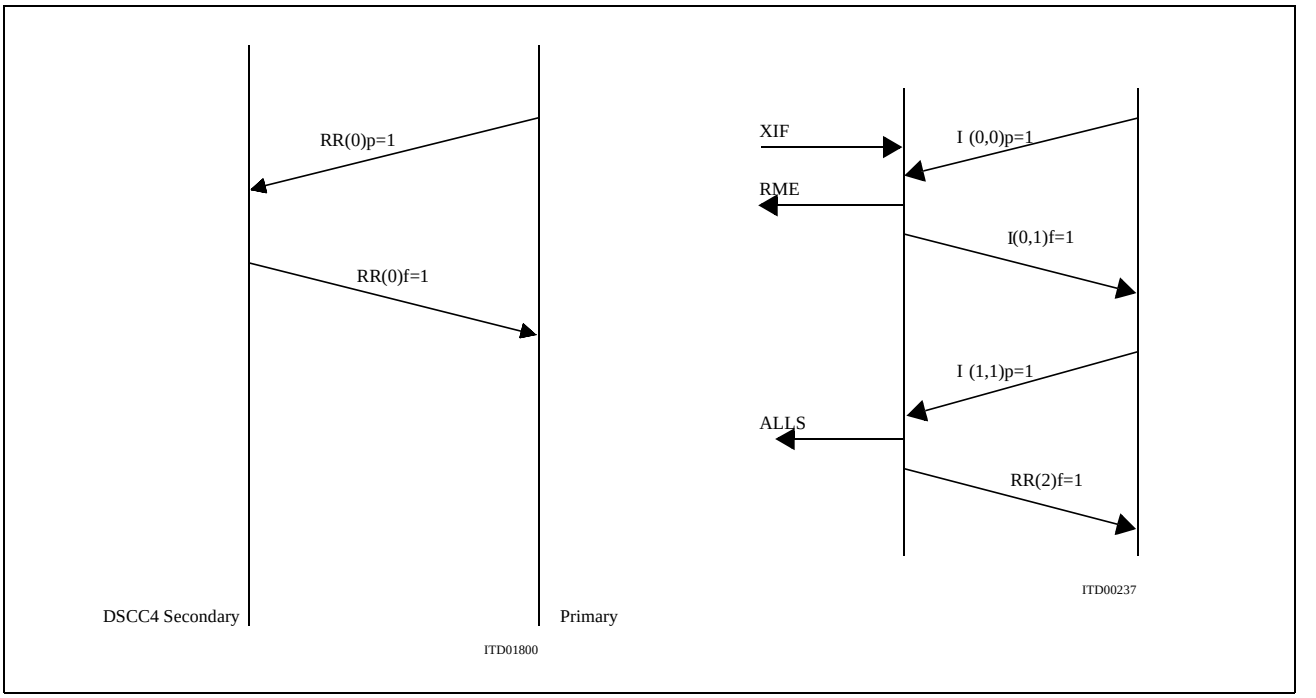


Figure 68 No Data to Send: Data Reception/Transmission

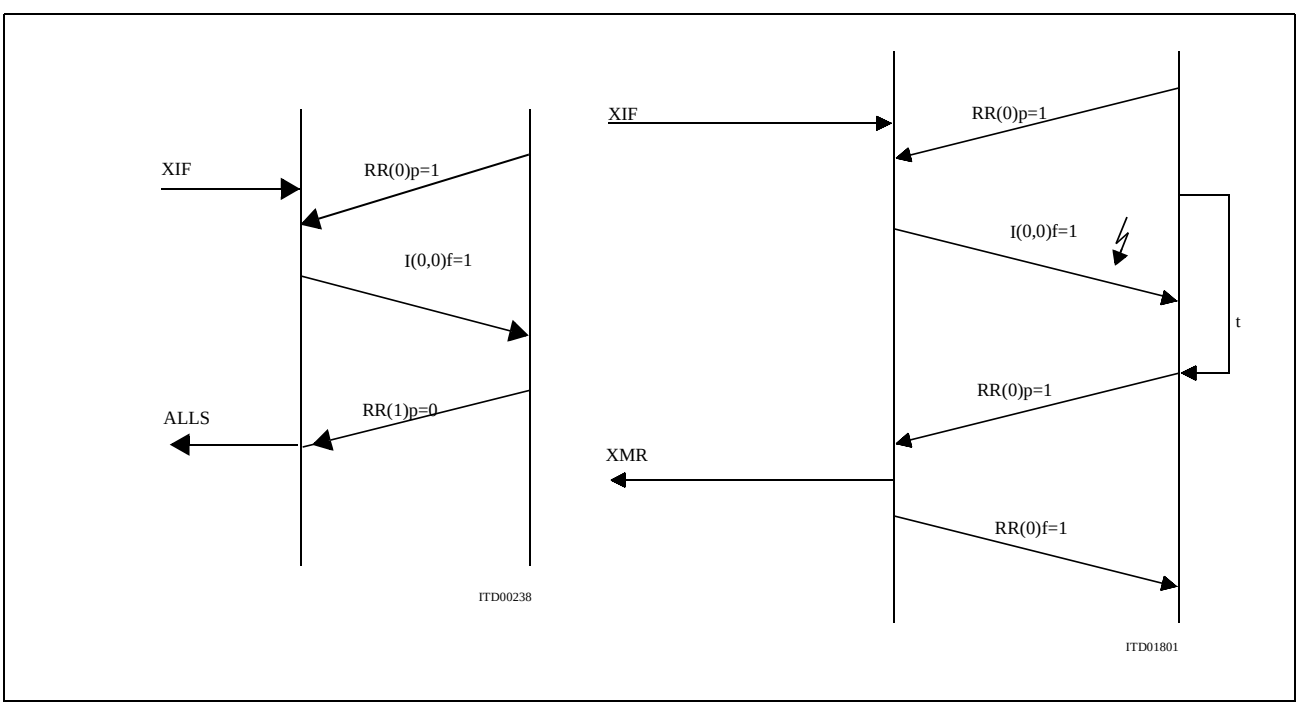


Figure 69 Data Transmission (without error), Data Transmission (with error)

8.1.9 Special Functions

8.1.9.1 Extended Transparent Transmission and Reception

When programmed in the extended transparent mode via the MODE register (MODE:MDS1, MDS0, ADM = ‘111’), the SCC performs fully transparent data transmission and reception without HDLC framing, i.e. without

- FLAG insertion and deletion
- CRC generation and checking
- bit stuffing.

This feature can be profitably used e.g. for:

- user specific protocol variations
- line state monitoring, or
- test purposes, in particular for monitoring or intentionally generating HDLC protocol rule violations (e.g. wrong CRC)

Character or octet boundary synchronization can be achieved by using clock mode 1 with an external receive strobe input to pin CD.

8.1.9.2 Receive Address Handling

The Receive Address Low/High Bytes (RAL1/RAH1 and RAL2/RAH2) in register RADR can be masked on a per bit basis by setting the corresponding bits in the mask register RAMR. This allows extended broadcast address recognition. Masked bit positions always match in comparison of the received frame address with the respective address fields in register RADR.

This feature is applicable to all HDLC protocol modes with address recognition (auto mode, non-auto mode and address mode 1). It is disabled if all bits of mask bit fields AML and AMH are set to ‘zero’ (which is the RESET value).

The function of RADR:RAL2/RADR:RAH2 and detection of the fixed group address FE_H or FC_H if applicable to the selected operating mode remains unchanged.

As an option in the auto mode, non-auto mode and address mode 1, the 8/16 bit address field of received frames can be pushed to the receive data buffer (first one/two bytes of the frame). This function is especially useful in conjunction with the extended broadcast address recognition. It is enabled by setting control bit ’RADD’ in register CCR2.

8.1.9.3 Shared Flags

If the ‘Shared Flag’ feature is enabled by setting bit ’SFLG’ in register CCR1 the closing flag of a previously transmitted frame simultaneously becomes the opening flag of the following frame if there is one already available in the SCC transmit FIFO.

In receive direction the SCC always expects and handles ’Shared Flags’. ’Shared Zeroes’ of consecutive flags are also supported.

8.1.9.4 One Bit Insertion

Similar to the zero bit insertion (bit stuffing) mechanism, as defined by the HDLC protocol, the SCC offers a completely new feature of inserting/deleting a ’one’ after seven consecutive ‘zeros’ into the transmit/receive data stream, if the serial channel is operating in bus configuration mode. This method is useful if clock recovery is performed by DPLL.

Since only NRZ data encoding is supported in a bus configuration, there are possibly long sequences without edges in the receive data stream in case of successive ‘0’s received, and the DPLL may lose synchronization.

Enabling the one bit insertion feature by setting bit ’OIN’ in register CCR2, it is guaranteed that at least after

– 5 consecutive ‘1’s a ‘0’ will appear (bit stuffing), and after
– 7 consecutive ‘0’s a ‘1’ will appear (one insertion)

and thus a correct function of the DPLL is ensured.

Note: As with the bit stuffing, the ‘one insertion’ is fully transparent to the user, but it is not in accordance with the HDLC protocol, i.e. it can only be applied in proprietary systems using circuits that also implement this function, such as the SAB 82525/ SAB 82526.

8.1.9.5 Preamble Transmission

If enabled via bit ’EPT’ in register CCR2, a programmable 8-bit pattern is transmitted with a selectable number of repetitions after Interframe Timefill transmission is stopped and a new frame is ready to be sent out. The 8 bit preamble pattern can be programmed in bit field ’PRE’ and the repetition time in bit field ’PREREP’ of register CCR2.

Note: Zero Bit Insertion is disabled during preamble transmission. To guarantee correct function the programmed preamble value should be different from Receive Address Byte values defined for any of the connected stations.

8.1.9.6 CRC Generation and Checking

In HDLC/SDLC mode, error protection is done by CRC generation and checking.

In standard applications, CRC-CCITT algorithm is used. The Frame Check Sequence at the end of each frame consists of two bytes of CRC checksum.

If required, the CRC-CCITT algorithm can be replaced by the CRC-32 algorithm, enabled via bit ’C32’ in register CCR1. In this case the Frame Check Sequence consists of four bytes.

As an option in non-auto mode or address mode 0, the internal handling of received and transmitted CRC checksum can be influenced via control bits ’RCRC’ and ’XCRC’ in register CCR2.

Receive direction:
The received CRC checksum is always assumed to be in the 2 (CRC-CCITT) or 4 (CRC-32) last bytes of a frame, immediately preceding a closing flag. If bit ’RCRC’ is set, the received CRC checksum is treated as data and will be written to the receive data buffer in the shared memory where it precedes the frame status byte. Nevertheless the received CRC checksum is additionally checked for correctness. If non-auto mode is selected, the limits for ‘Valid Frame’ check are modified (refer to description of the Receive Status Byte (RSTA)).

Transmit direction:
If bit ’XCRC’ is set, the CRC checksum is not generated internally. The checksum has to be provided via the transmit data buffer by software. The transmitted frame will only be closed automatically with a (closing) flag.

Note: The SCC does not check whether the length of the frame, i.e. the number of bytes, to be transmitted makes sense or not according the HDLC protocol.

8.1.9.7 Data Transparency in PPP Mode

When transporting bit-files (as opposed to text files), or compressed files, the characters could easily represent MODEM control characters (such as CTRL-Q, CTRL-S) which the MODEM would not pass through. The DSCC4 maintains an Async Control Character Map (ACCM) for characters 00-1F Hex. Whenever there is a mapped character in the data stream, the transmitter precedes that character with a control-escape character of $7D_H$. After the control-escape, the character itself is transmitted with bit 5 inverted. character e.g. 13_H is mapped to $7D_H$, 33_H).

At the receive end, a $7D_H$ character is discarded and the following character is modified by inverting bit 5 (e.g. if $7D_H$, 33_H is received, the $7D_H$ is discarded and the 33_H is changed to 13_H the original character).

In addition to the ACCM, 4 user programmable characters (especially outside the range 00-1F Hex) can also be mapped using the control-escape sequence described above. These characters are specified in register UADC.

The receiver discards all characters which are received unmapped, but expected to be mapped because of ACCM and UDAC register contents. If this occurs within an HDLC frame, the unexpected characters are discarded before forwarded to the receive CRC checking unit.

$7D_H$ (control-escape) and $7E_H$ (flag) octets in the data stream are mapped in general. The sequence of mapping control logic is:

1. $7D_H$ and $7E_H$ octets,
2. ACCM,
3. UDAC.

The 32 lookup octet values (00_H-$1F_H$) are stored within the device. One dword programmable register is used to select which of the 32 fixed characters have to be mapped using the control-escape sequence. This is maintained by register ACCM. Register UDAC provides the additional 4 user programmable characters to be mapped.

This mechanism is applied to asynchronous HDLC PPP mode as well as to octet synchronous HDLC PPP mode.

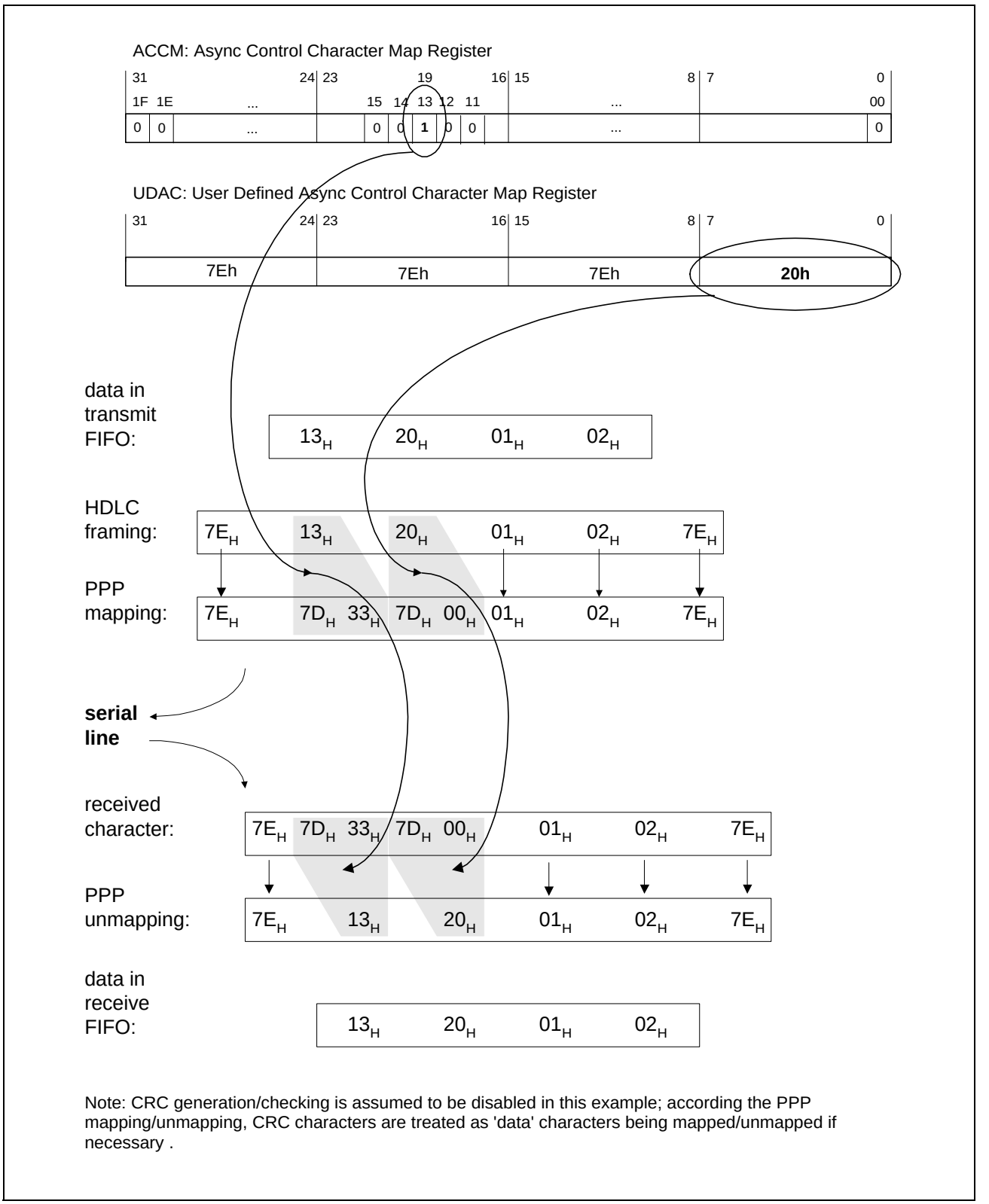


Figure 70 PPP Mapping/Unmapping Example

8.1.9.8 Receive Length Check Feature

The SCC offers the possibility to supervise the maximum length of received frames and to terminate data reception in the case that this length is exceeded.

This feature is controlled via the special Receive Length Check Register RLCR.

The function is enabled by setting bit ’RC’ (Receive Check) and the maximum frame length to be checked is programmed via bit field ’RL’. The maximum receive length can be determined as a multiple of 32-byte blocks as follows:

MAX_LENGTH = (RL + 1) $\times$ 32 ,

MAX_LENGTH = 32 ... 65536

where RL is the value written to bit field ’RL’.

All frames exceeding this length are treated as if they had been aborted by the remote station, i.e. the CPU is informed via

– a FLEX interrupt is generated by the SCC, and
– the receive abort indication ’RAB’ in the Receive Status Byte (RSTA) is set.

Receive operation continues with the beginning of the next receive frame.

8.2 Asynchronous (ASYNC) Protocol Mode

8.2.1 Character Framing

Character framing is achieved by start and stop bits. Each data character is preceded by one Start bit and terminated by one or two stop bits. The character length is selectable from 5 up to 8 bits. Optionally, a parity bit can be added which complements the number of ones to an even or odd quantity (even/odd parity). The parity bit can also be programmed to have a fixed value (Mark or Space). The character format configuration is performed via appropriate bit fields in register CCR2. **Figure 71** shows the asynchronous character format.

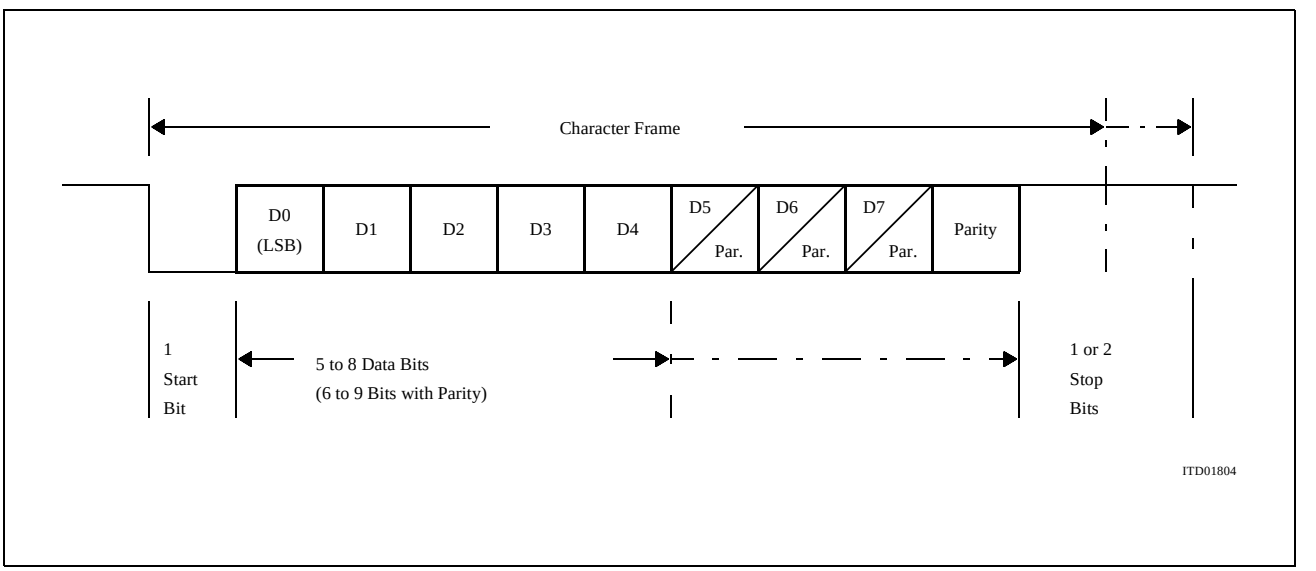


Figure 71 Asynchronous Character Frame

8.2.2 Data Reception

The SCC offers the flexibility to combine clock modes, data encoding and data sampling in many different ways. However, only definite combinations make sense and are recommended for correct operation:

8.2.2.1 Asynchronous Mode

Prerequisites:

- Bit clock rate 16 selected (register CCR0, bit BCR = ‘1’)
- Clock mode 0, 1, 3b, 4, or 7b selected (register CCR0, bit field ’CM’)
- NRZ data encoding selected (register CCR0, bit field ’SC’)

The receiver which operates with a clock rate equal to 16 times the nominal (expected) data bit rate, synchronizes itself to each character by detecting and verifying the start bit. Since character length, parity and stop bit length is known, the ensuing valid bits are

sampled. Oversampling (3 samples) around the nominal bit center in conjunction with majority decision is provided for every received bit (including start bit).

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

8.2.2.2 Isochronous Mode

Prerequisites:

- Bit clock rate 1 selected (register CCR0 bit BCR = ‘0’)
- Clock mode 2, 3a, 6, or 7a (DPLL mode) has to be used in conjunction with FM0, FM1 or Manchester encoding (register CCR0 bit fields ’CM’ and ’SC’).

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

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

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

8.2.2.3 Storage of Receive Data

If the receiver is enabled, received data is stored in the SCC receive FIFO (the LSB is received first). Moreover, the CD input may be used to control data reception. Character length, number of stop bits and the optional parity bit are checked. Storage of parity bits can be disabled. Errors are indicated via interrupts. Additionally, the character specific error status (framing and parity) can optionally be stored in the SCC receive FIFO.

Filling of the the SCC receive FIFO is controlled by

- a programmable threshold level (bit field ’RFTH’ in register CCR2),
- the selected data format (bit ’RFDF’ in register CCR2),
- the parity storage selection (bit ’DPS’ in register CCR2),
- detection of the programmable Termination Character (bit ’TCDE’ and bit field ’TC’ in register TCR).

Additionally, the time-out event interrupt as an optional status information indicates that a certain time (refer to register CCR1) has elapsed since the reception of the last character.

8.2.3 Data Transmission

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

Transmission of the contents of the SCC transmit FIFO starts after providing data to the DMA controller. The character frame for each character, consisting of start bit, the character itself with defined character length, optionally generated parity bit and stop bit(s) is assembled.

After finishing transmission (indicated by the ‘ALLS’ interrupt), IDLE sequence (logical ‘1’) is transmitted on transmit pin TxD.

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

8.2.4 Special Functions

8.2.4.1 Break Detection/Generation

Break generation:

On issuing the transmit break command (bit ’XBRK’ in register CCR2), the TxD pin is immediately forced to physical ‘0’ level with the next following clock edge, and released with the first clock edge after this command is reset again by software.

Break detection:

The SCC recognizes the break condition upon receiving consecutive (physical) ‘0’s for the defined character length, the optional parity and the selected number of stop bits (‘zero’ character and framing error). The ‘zero’ character is not pushed to RFIFO. If enabled, the ’Break’ interrupt (BRK) is generated.

The break condition will be present until a ‘1’ is received which is indicated by the ‘Break Terminated’ interrupt (BRKT).

8.2.4.2 In-band Flow Control by XON/XOFF Characters

Programmable XON and XOFF characters:

The XNXF register contains the programmable values for XON and XOFF characters. The number of significant bits in a register is determined by the programmed character length via bit field ’CHL’ in register CCR2.

Additionally, two programmable eight-bit values ’MXN’ and ’MXF’ serve as masks for the characters XON and XOFF, respectively:

A ‘1’ in any mask bit position has the effect that no comparison is performed between the corresponding bits in the received characters (‘don’t cares’) and the XON/XOFF value. At RESET, the masks are ‘zero’ed, i.e. all bit positions will be compared.

A received character is considered to be recognized as a valid XON or XOFF character

– if it is correctly framed (correct length),
– if its bits match the ones in the XON or XOFF registers over the programmed character length,
– if it has correct parity (if applicable).

Received XON and XOFF characters are stored in the SCC receive FIFO, as any other characters, when bit DXS is set to ’0’ in register CCR2. Otherwise they are not stored in the receive FIFO.

In-Band Flow Control of Transmitted Characters:

Recognition of an XON or XOFF character causes always a corresponding maskable interrupt status to be generated.

Further action depends on the setting of control bit ’FLON’ (Flow Control On) in register CCR2:

0: No further action is automatically taken by the SCC.

1: The reception of an XOFF character automatically turns off the transmitter after the currently transmitted character (if any) has been shifted out completely (entering XOFF state). The reception of an XON character automatically makes the transmitter resume transmitting (entering XON state).

After hardware RESET, bit CCR2:FLON is ‘0’.

When bit CCR2:FLON is programmed from ‘0’ to ‘1’, the transmitter is first in the ‘XON state’, until an XOFF character is received.
When bit CCR2:FLON is programmed from ‘1’ to ‘0’, the transmitter always goes in the ‘XON state’, and transmission is only controlled by the user and by the $\overline{\text{CTS}}$ signal input.

The in-band flow control of the transmitter via received XON and XOFF characters can be combined with control via $\overline{\text{CTS}}$ pin, i.e. the effect of the $\overline{\text{CTS}}$ pin is independent of whether in-band control is used or not. The transmitter is enabled only if $\overline{\text{CTS}}$ is ‘low’ and XON state has been reached.

Transmitter Status Bit:

The status bit ‘Flow Control Status’ (bit ’FCS’ in register STAR) indicates the current state of the transmitter, as follows:

0: if the transmitter is in XON state,

1: if the transmitter is in XOFF state.

Note: The transmitter cannot be turned off by software without disrupting data possibly remaining in the transmit FIFO.

Flow Control for Received Data:

After writing a character value to register TICR (Transmit Immediate Character) its character contents is inserted into the outgoing character stream

- immediately upon writing this register by the microprocessor if the transmitter is in IDLE state. If no further characters (transmit FIFO empty) are to be transmitted, i.e.

the transmitter returns to IDLE state after transmission of the TIC and an ALLS (All Sent) interrupt will be generated.

- after the end of a character currently being transmitted if the transmitter is not in IDLE state. This does not affect the contents of the transmit FIFO. Transmission of characters from transmit FIFO is resumed after the TIC is send out.

Transmission via this register is possible even when the transmitter is in XOFF state (however, $\overline{\text{CTS}}$ must be ‘low’).

The TIC value is an eight-bit value. The number of significant bits is determined by the programmed asynch character length. Parity value (if programmed) and selected number of stop bits are automatically appended, equal to the characters provided via the transmit data buffer. The usage of TIC is independent of in-band flow control mechanism, i.e. is not affected by bit ’FLON’ in register CCR2 anyway.

To control multiple accesses to register TICR, an additional status bit STAR:TEC (TIC Executing) is provided which signals that the transmission command of currently programmed TIC is accepted but not yet completely executed. Further access to register TIC is only allowed if bit STAR:TEC is ‘0’ again.

8.2.4.3 Out-of-band Flow Control

Transmitter:

The transmitter output is enabled if $\overline{\text{CTS}}$ signal is ‘LOW’ AND the XON state is reached in case of in-band flow control is enabled. If the in-band flow control is disabled (CCR2:FLON = ‘0’), the transmitter is only controlled by the $\overline{\text{CTS}}$ signal.

Nevertheless setting bit **CCR1:FCTS = ‘1’ allows the transmitter to send data independent of the condition of the $\overline{\text{CTS}}$ signal,** the in-band flow control (XON/XOFF) mechanism would still be operational if enabled via bit CCR2:FLON = ‘1’.

Receiver:

For some applications it is desirable to provide means of out-of-band flow control to indicate to the far end transmitter that the local receiver’s buffer is getting full.

This flow control can be used between two DTEs as shown in **figure 72** and between a DTE and a DCE (MODEM) as shown in **figure 73** that supports this kind of bi-directional flow control.

Setting bit CCR1:FRTS = ‘1’ and CCR1:RTS = ‘0’ invokes this out-of-band flow control for the receiver. When the shadow part of the SCC receive FIFO has reached a pre-defined threshold of 20 bytes, the $\overline{\text{RTS}}$ signal is forced inactive (HIGH). When the shadow part of the receive FIFO is empty, the $\overline{\text{RTS}}$ is re-asserted (‘LOW’). Note that the data is immediately transferred from the shadow receive FIFO to the DMA accessible FIFO (as long as there is space available). Thus when the shadow FIFO reaches the 20 bytes threshold, there are 4 more bytes storage available before an overflow can occur. This provides sufficient time for the far end transmitter to react to the change in the $\overline{\text{RTS}}$ signal and stop sending more data.

Figure 72 shows the connection between two SCC devices as DTEs. The $\overline{RTS}$ of DTE-A (SCC) feeds the $\overline{CTS}$ input of the second DTE-B (another SCC). For example while DTE-A is receiving data and its receive FIFO threshold is reached, the $\overline{RTS}$ signal goes in-active 'HIGH' forcing the $\overline{CTS}$ of DTE-B to become in-active indicating that transmission has to stop after finishing the current character. Both DTE devices should also be using the $\overline{CTS}$ signal to flow control their transmitters. When the shadow receive FIFO in DTE-A is cleared its $\overline{RTS}$ goes active 'LOW' and this signals the far end DTE-B to resume transmission. Data flow control from DTE,-B to DTE-A works in the same way.

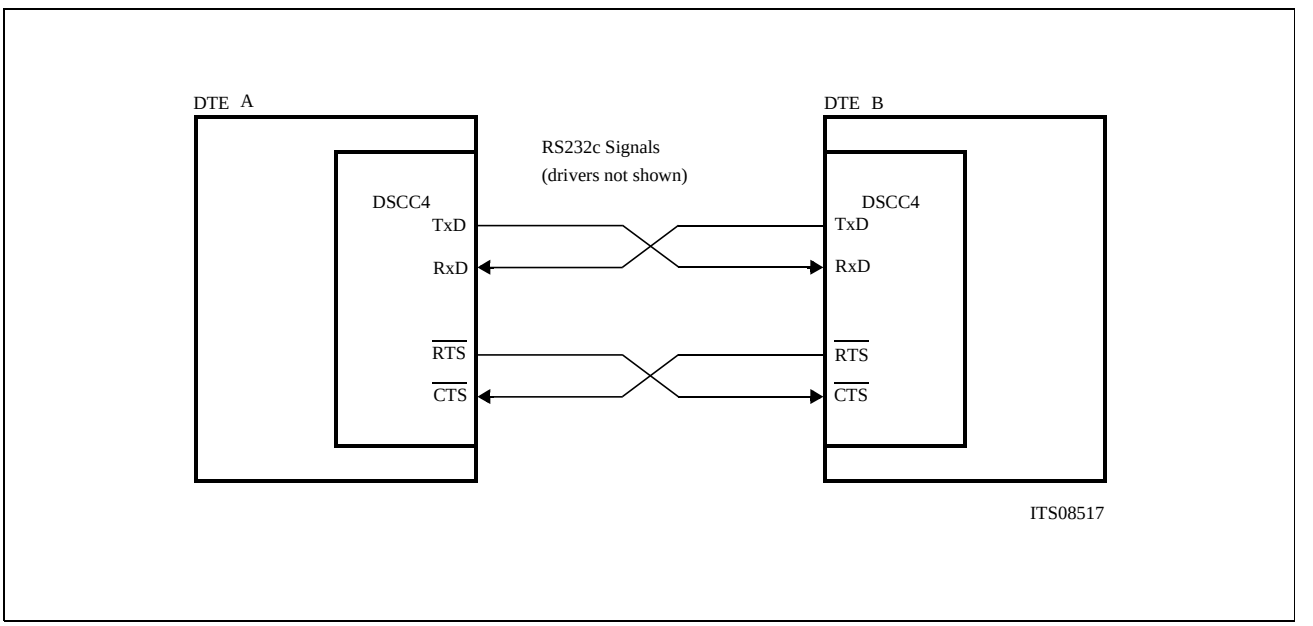


Figure 72 Out-of-Band DTE-DTE Bi-directional Flow Control

Figure 73 shows an SCC as a DTE connected to a DCE (MODEM equipment).

The $\overline{RTS}_A$ feeds the $\overline{RTS}_B$ input of the DCE (MODEM equipment) that supports bi-directional flow control. So when the DTE-A's receiver threshold is reached, the $\overline{RTS}_A$ signal goes active 'HIGH' which is sensed by the DCE and it stops transmitting. Similarly if the DCE's receiver threshold is reached, it deactivates the $\overline{CTS}_B$ ('HIGH') and causes the DTE to stop transmission. These types of DCEs have fairly deep buffers to ensure that it can continue to receive data from the line even though it is unable to pass the data to the DTE for short periods of time. Note that a SCC can also be used in the DCE equipment as shown. Exchange of signals (e.g. $\overline{RTS}$ to $\overline{CTS}$) is necessarily inside the DCE equipment.

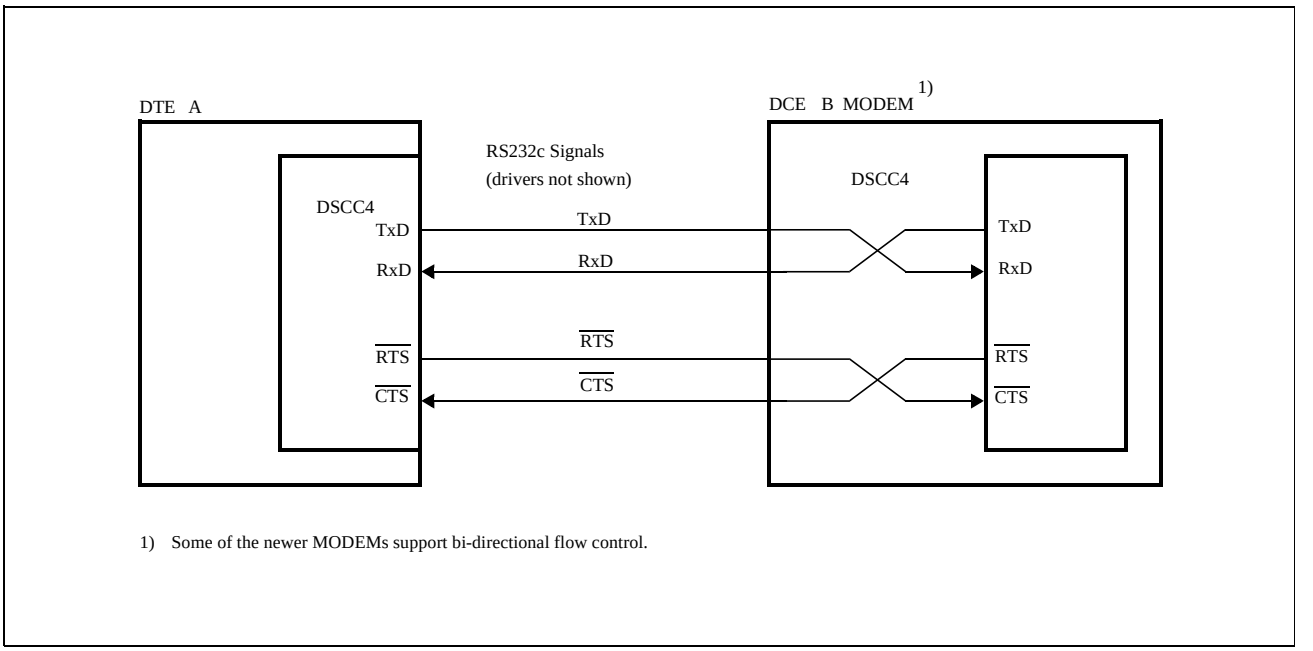


Figure 73 Out-of-Band DTE-DCE Bi-directional Flow Control

$\overline{RTS}$ and $\overline{CTS}$ are used to indicate when the local receiver's buffer is nearly full. This alerts the far end transmitter to stop transmission.

The combination of transmitter and receiver out-of-band control features mentioned above enables data to be exchanged between two devices without software intervention for flow control.

8.3 Character Oriented Synchronous (BISYNC) Protocol Mode

8.3.1 Character Framing

Character oriented protocols achieve synchronization between transmitting and receiving station by means of special SYN characters. Two examples are the MONOSYNC and IBM's BISYNC procedures. BISYNC has two starting SYN characters while MONOSYNC uses only one SYN. **Figure 74** gives an example of the message format.

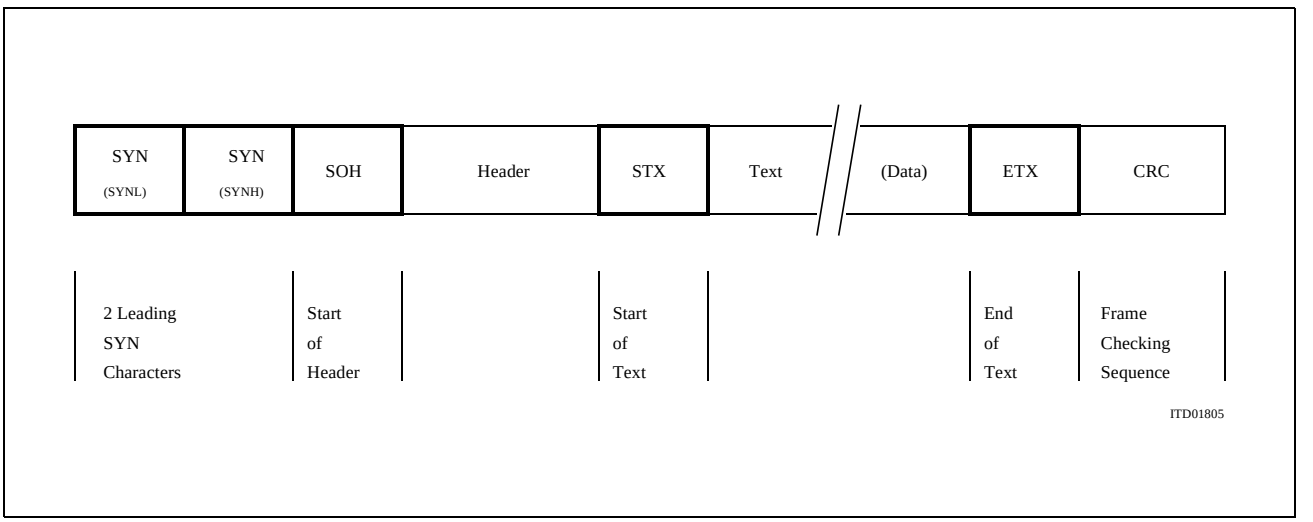


Figure 74 BISYNC Message Format

The SYN character, its length, the length of data characters and additional parity generation are programmable:

- 1 SYN character with 6 or 8 bit length (MONOSYNC), programmable via register SYNCR.
- 2 SYN characters with 6 or 8 bit length each (BISYNC), programmable via registers SYNCR.
- Data character length may vary from 5 to 8 bits (bit field 'CHL' in register CCR2).
- Parity information (even/odd parity, mark, space) may be appended to the character (bit 'PARE' and bit field 'PAR' in register CCR2).

8.3.2 Data Reception

The receiver is generally activated by setting bit 'RAC' in register CCR2. Additionally, the CD signal may be used to control data reception depending on the selected clock mode. After issuing the HUNT command, the receiver monitors the incoming data stream for the presence of specified SYN character(s). However, data reception is still disabled. If synchronization is gained by detecting the SYN character(s), an SCD interrupt is generated and **all** following data is pushed to the receive FIFO, i.e. control sequences, data characters and optional CRC frame checking sequence (the LSB is received first).

In normal operation, SYN characters are excluded from storage to receive FIFO. SYN character length can be specified independently of the selected data character length. If required, the character parity bit and/or parity status is stored together with each data byte in the receive FIFO.

As an option, the loading of SYN characters in receive FIFO may be enabled by setting the bit 'SLOAD' in register CCR2. Note that in this case SYN characters are treated as data. Consequently, for correct operation it must be guaranteed that SYN character length equals the character length + optional parity bit. This is the user's responsibility by appropriate software settings.

Filling of the receive FIFO is controlled by a programmable threshold level.

Reception is stopped if

1. the receiver is deactivated by resetting the RAC bit, or
2. the CD signal goes inactive (if Carrier Detect Auto Start is enabled), or
3. the HUNT command is issued again, or
4. the Receiver Reset command (RRES) is issued, or
5. a programmed Termination Character has been found (optional).

On actions 1. and 2., reception remains disabled until the receiver is activated again. After this is done, and generally in cases 3. and 4., the receiver returns to the (non-synchronized) Hunt state. In case 5. a HUNT command has to be issued. Reception of data is internally disabled until synchronization is regained.

Note: Further checking of frame length, extraction of text or data information and verifying the Frame Checking Sequence (e.g. CRC) has to be done by the microprocessor.

8.3.3 Data Transmission

Transmission of data provided in the shared memory is started after the DMA controller forwards the first data bytes to the SCC transmit FIFO (the LSB is sent out first). Additionally, the $\overline{\text{CTS}}$ signal may be used to control data transmission. The message frame is assembled by appending all data characters to the specified SYN character(s) until Transmit Message End condition is detected (FE indication via DMAC). Internally generated parity information may be added to each character (SYN, CRC and Preamble characters are excluded).

If enabled via CRC Append bit (bit 'CAPP' in register CCR2), the internally calculated CRC checksum (16 bit) is added to the message frame. Selection between CRC-16 and CRC-CCITT algorithms is provided.

Note: - Internally generated SYN characters are always excluded from CRC calculation,
- CRC checksum (2 bytes) is sent without parity.

The internal CRC generator is automatically initialized before transmission of a new frame starts. The initialization value is selectable.

After finishing data transmission, Interframe Timefill (SYN characters or IDLE) is automatically sent.

8.3.4 Special Functions

8.3.4.1 Preamble Transmission

If enabled via register CCR2, a programmable 8-bit pattern (bit field ’PRE’) is transmitted with a selectable number of repetitions after Interframe Timefill transmission is stopped and a new frame is ready to be sent out.

Note: If the preamble pattern equals the SYN pattern, reception is triggered by the preamble.

8.3.4.2 CRC Parity Inhibit

If the internal CRC generator is not used for calculation of the Frame Check Sequence, an externally calculated checksum (16 bits) can be appended to the message frame without internally generated parity information, although parity is enabled for data characters.

Prerequisites are:

- CRC generator disabled (CAPP = ‘0’),
- Frame/Block End indication has to be issued with the checksum provided in shared memory.

The programmed character length has no influence on this function.

9 Reset and Initialization Procedure

9.1 Reset and Power-On

The DSCC4 offers several reset functions. An external low signal on signal $\overline{\text{RST}}$ resets the DSCC4. It immediately drives all PCI outputs to their benign state, pin $\overline{\text{REQ}}$ is driven to *tristate*. The TxDn output signals are driven to high impedance, the $\overline{\text{RTSn}}$ output signals are driven to inactive. All bi-directional signals (e.g. Multi Function Port (MFP)) are switched to input function.

All registers and functions are initialized to known states (RESET values).

After $\overline{\text{RST}}$ is deasserted, the functional blocks PCI, SCCs, LBI, SSC and GPP are in reset state or standby mode.

Table 26 Status after Hardware Reset

Block	Reset Status
PCI interface	Standby PCI Config Space registers accessible.
Global Registers	Standby Slave registers accessible; Host Bus Interface enabled in PCI mode!
DMAC	Standby Slave registers accessible.
SCC	Standby Slave registers accessible Reset Status: – power-down mode – HDLC mode – NRZ coding – … Refer to reset values in **“SCC Registers - Detailed Register Description” on Page 272**.
LBI	Reset Slave register LCONF has reset value.
SSC	Standby Slave registers accessible.
GPP	Standby Slave registers accessible.

Software Reset

Software reset control bits are also available to reset single sections of the DSCC4. Reset of

– DMA receiver is performed via CHiCFG.RDR bit (channel specific)
– DMA transmitter is performed via CHiCFG.RDT bit (channel specific)
– transmitter in SCC is performed via CMDR.XRES bit (channel specific)
– receiver in SCC is performed via CMDR.RRES bit (channel specific)
– EBC (LBI, SSC, GPP) block is performed via LCONF.$\overline{\text{EBCRES}}$ bit.

Note: The software reset only affects the internal state machines. The registers are not reset.

9.2 Initialization Example

The first step of initialization is done already via hardware configuration. If DEMUX pin is connected to V_{DD3}, the DSCC4 is configured in de-multiplexed bus interface mode. Otherwise (DEMUX connected to V_{SS}) the DSCC4 is configured in PCI mode and the MFP is available as LBI port (Reset state).

After hardware reset, the host has to write a minimum set of registers to initialize the functional blocks. The following tables provide initialization sequences assuming that in parallel the host will reserve/prepare memory space for interrupt ring buffers and linked list data structures before activating the DSCC4's DMAC by setting GCMDR.AR (See **"Start of Operation" on Page 207**). During the initialization phase the DSCC4 is operating in slave mode.

Table 27 Global Configuration of DSCC4 and Initialization of DMAC (Interrupt Channel)

Step	Action	
1	Select Little-/Big-Endian mode via bit GMODE.ENDIAN.	
2	Select Burst-/No-Burst mode via bit GMODE.DBE. (Valid only in DEMUX mode)	
3	Select Priority Scheme via bit GMODE.SPRI and GMODE.CHN.	
4	Configure MFP via bit fields GMODE.PERCFG and GMODE.LCD as: - LD15..0, LA15...0 (LBI mux/demux) - LD15..0, GP 15...8, LA7...0 (GPP + LBI) - LD15..0, GP15...8, SSC (GPP + SSC + LBI) - LD15..0, GP15...0 (GPP + LBI)	
5	Set interrupt queue base addresses in registers IQSCCnRXBAR, IQSCCnTXBAR, IQCFGBAR, IQPBAR	In parallel reserve memory space for interrupt queues (ring buffers) in shared memory
6	Set interrupt queue lengths in registers IQLENR1, IQLENR2	

Table 28 Initialization of DMAC (Data Channels)

Step	Action		
1	Select Control Mode of the DMA controller via GMODE.CMODE bit.		In parallel prepare linked list(s) in shared memory.
	GMODE.CMODE = ’0’ (causes the DMAC to check HOLD bit before branching to next descriptor).	**GMODE.CMODE = ’1’** (causes the DMAC to compare FTDA/FRDA and LTDA/LRDA before branching to next descriptor).	
2	Select FIFO size and thresholds: FIFOCR1...4		
3	Set base descriptor addresses: BRDA, BTDA.		
4	Verify/set HOLD=’1’ in last element of linked list.	Write last transmit/receive descriptor address to LTDA/ LRDA register.	
5	Configure channels: CHiCFG - Interrupt Mask (RFI, TFI, RERR, TERR) - DMAC Command (RDR, RDT, IDR, IDT)		

For the SCC, first, the serial mode, the configuration of the serial port and the clock mode have to be defined. The host may switch the SCC between power-up and power-down mode. This has no influence upon the contents of the registers, i.e. the internal state remains stored. In power-down mode, however, all internal clocks are disabled, no interrupts are forwarded to the host. This state can be used as a standby mode, when the SCC is temporarily not used, thus substantially reducing power consumption.

Table 29 Initialization of the SCC(s)

Step	Action
1	Set clock mode specific features: CCR0, BRR, TTSA, RTSA, TPCMM, RPCMM.
2	Set serial mode (HDLC, ASYNC, BISYNC): CCR0
3	Set serial port configuration (Encoding, output driver select, handshaking mechansim): CCR0, CCR1

Table 29 Initialization of the SCC(s) (cont'd)

Step	Action
4	Set serial mode specific features for **HDLC** Auto Mode, NRM Mode: CCR1, TIMR, XADR, RADR, RAMR Non Auto Mode: CCR1, CCR2, RADR, RAMR Transparent Mode: CCR1, CCR2, RADR, RAMR Extended Transparent Mode: CCR1 Asynchronous PPP: CCR1, CCR2, UDAC, ACCM Synchronous PPP: CCR1, CCR2, UDAC, ACCM General: Shared flags, CRC reset level: CCR1 Preamble, ITF/OIN, SCC RFIFO configuration: CCR2 Receive Length Check: RLCR **ASYNC** Bit clock rate: CCR0 Data format, SCC RFIFO configuration, Flow control: CCR2, XNXF Character Insertion: TICR Termination Character: TCR **BISYNC** Bisync/Monosync: CCR1 Sync character: SYNCR Data format, SCC RFIFO configuration, Preamble, CRC: CCR2 Termination character: TCR
5	Set interrupt mask: IMR

In PCI mode the MFP can be used to access external peripherals like ESCC2, HSCX, FALC54, when LBI configuration is selected. In LBI configuration even data can be exchanged with a local microprocessor. Moreover, a General Purpose Port (GPP) can be used for control purposes. Alternatively, serial communication can be performed via the SSC port.

Depending on the configuration selected in GMODE register (bit field GMODE.PERCFG), the appropriate register set of the MFP has to be initialized, too (LBI, SSC, GPP registers).

Table 30 Initialization of the MFP

Step	Action
1	When the MFP has been configured as – LBI port: configure LCONF – GP port: configure GPDIR, GPIM – SSC port: configure SSCCON, SSCBR, SSCCSE, SSCIM.

9.3 Start of Operation

After having performed the initialization, the host requests the activation of the DSCC4 by enabling the SCC(s) and setting the corresponding bits in the GCMDR.

A correct sequence is:

- activation of the SCC
- activation of the DMAC (receive and transmit)
- enabling the SCC receiver

Table 31 Activation of DMAC and SCC

Step	Action by Host	Action by DSCC4
1	Set appropriate command bits for interrupt queue initialization and Action Request (AR) bit in GCMDR.	
2		The DMAC sets up the interrupt queues. When the configuration was successful, the DMAC sets GSTAR.ARACK=’1’. If enabled, INT signal is activated and the DMAC stores the corresponding configuration interrupt vector in the interrupt queue IQCFG located in the shared memory.
3	Serve interrupts.	
4	Set the SCC to power-up mode via CCR0.PU. The SCC receiver should remain disabled (CCR2.RAC=’0’).	
5	Reset SCC transmitter.	
6		The SCC requests for data from the central TFIFO and, if data are available, data transmission is started.
7		Receive/transmit interrupts caused by the SCC are forwarded as interupt vectors through the central interrupt queue to the appropriate interrupt buffers in shared memory.
8	Serve interrupts.	

Table 31 Activation of DMAC and SCC (cont'd)

Step	Action by Host	Action by DSCC4
9	Set appropriate channel configuration bits (receive and transmit) and Action Request (AR) bit in GCMDR.	
10		After the DSCC4 has become bus master the DMAC the internal DMA channels for data transfer. When the configuration was successful, the DMAC sets GSTAR.ARACK='1'. If enabled, INT signal is activated and the DMAC stores the corresponding configuration interrupt vector in the interrupt queue IQCFG located in the shared memory. Moreover the DMAC branches to the base receive/transmit descriptor (referenced by BRDA/ BTDA) of the linked list. In transmit direction it starts transferring data - if available - from shared memory into the on chip central transmit FIFO. Since the SCCs receivers are not activated so far, no data are received from SCCs.
11	Read GSTAR for interrupt information and acknowledge interrupts by writing a '1' back to the bits, which indicate an interrupt	
12	Enable the SCC receiver (CCR2.RAC='1').	SCC starts data reception. The received data are transferred to the central RFIFO. As soon as the threshold of the RFIFO has been reached, the DMAC transfers the data into the shared memory.

After start procedure the continuous operation of data transfer is essentially controlled by the host and the DMAC sharing the data structures and interrupt queues located in the memory.

Table 32 Continuous Operation of Data Transfer

Step	Action by Host		Action by DSCC4 (DMAC)	
	HOLD bit ctrld.	*LTDx/FTDx ctrld.*	*HOLD bit ctrld.*	*LTDx/FTDx ctrld.*
1	Add list elements to linked list, if available.		The DMAC transfers data between shared memory and on chip FIFOs. It branches to the next descriptor as long as HOLD = ’0’.	The DMAC transfers data between shared memory and on chip FIFOs. It branches to the next descriptor as long as FxDA is not equal to LxDA.
	Set HOLD=’1’ in new last element of linked list and reset HOLD bit in the previously last descriptor.	Write new last transmit/receive descriptor address in LTDA/ LRDA register.		
2	Set appropriate poll bit in GCMDR register GCMDR.TXPRi.			
3			If HOLD=1 has been sensed before, the DMAC reads current descriptor again and branches (HOLD=0) to next descriptor. Otherwise the poll request is ignored.	If FTDA/FRDA = LTDA/LRDA has been sensed before, the DMAC compares FTDA/ FRDA to updated LTDA/LRDA and branches to next descriptor.
4			Generally, host initiated interrupts might be generated.	
5	Serve interrupts appropriately.			

The sequence of functional steps shown in **table 32** are repeated as long as data transmission is required and no error does occur.

The procedures to stop data transmission and/or reception are shown in **table 33** and **table 34**.

Table 33 Stop Data Transmission

Step	Action by Host		Action by DSCC4
	HOLD bit ctrld.	*LTDx/FTDx ctrld*.	
1	Initialize CHiCFG register - Interrupt Mask: (RFI, TFI, RERR, TERR) - DMAC Command **(RDT)**		
2	Set GCMDR.AR.		
3			The DMAC transmit channel discards the transmit data stored internally and returns to its reset state. No additional data are read from the shared memory.
4	Reset transmitter in SCC.		
5			Data stored internally in SCC are discarded and transmission stops.

Table 34 Stop Data Reception

Step	Action by Host	Action by DSCC4
1	Disable receiver in SCC via CCR2.RAC.	
2		SCC stops data reception. The received data stored internally are transferred to the shared memory.

Beside normal operation, exceptions might happen such as:

– no memory available for received data (HOLD=’1’ or FRDA=LRDA in current receive descriptor), which leads to a **receive data overflow**
– no data available for transmission (HOLD=’1’, FE=’0’ or FTDA=LTDA, FE=’0’ in current transmit descriptor), which leads to **transmit data underrun**
– general failure, which can cause a software restart.

Table 35 Exceptional handling in Case of Receive Data Overflow

Step	Action by Host		Action by DSCC4
	HOLD bit ctrld.	*LTDx/FTDx ctrld.*	
No Host action has to be performed in case of a receive data overflow event. The DSCC4 marks the receive descriptor (data section) containing incomplete data with an 'RDO' indication in the receive status byte (RSTA). The DMA controller proceeds with the next receive descriptor; no difference to handling of a non-exceptional frame.			

Table 36 Exceptional handling in Case of Transmit Data Underrun

Step	Action by Host		Action by DSCC4
	HOLD bit ctrld.	*LTDx/FTDx ctrld.*	
1	Prepare linked list for future data transmission in shared memory.		
	Set HOLD='1' in last element of linked list.	-	
2	Initialize CHiCFG register - Interrupt Mask: (RFI, TFI, RERR, TERR) - DMAC Command **(RDT)**		
3	Reset transmitter in SCC via CMDR.XRES.		SCC starts requesting for transmit data from central TFIFO.
4	-	Write last transmit descriptor address in LTDA register.	
	Update BTDA. Initialize CHiCFG register - Interrupt Mask: (RFI, TFI, RERR, TERR) - DMAC Command **(IDT and AR)**		
5			SCC gets new data (if available) from the central TFIFO and Data transmission starts.

Table 36 Exceptional handling in Case of Transmit Data Underrun (cont'd)

Step	Action by Host	Action by DSCC4
7	Set GCMDR.AR.	
8		After the DSCC4 has become bus master the DMAC sets up the internal DMA channels for data transmission. If availablle, data are transferred from shared memory to on chip TFIFO.

In case of a general restart the initialization and start sequences have to be performed as described in **tables 31** to **32**.

9.4 Initialization Example

9.4.1 Test Loop For Data Transfer in HDLC Address Mode 0

The data in the transmit data buffer referenced by Tx Data Pointer are transmitted via SCC1's test loop and stored after reception in the receive data buffer referenced by Rx Data Buffer.

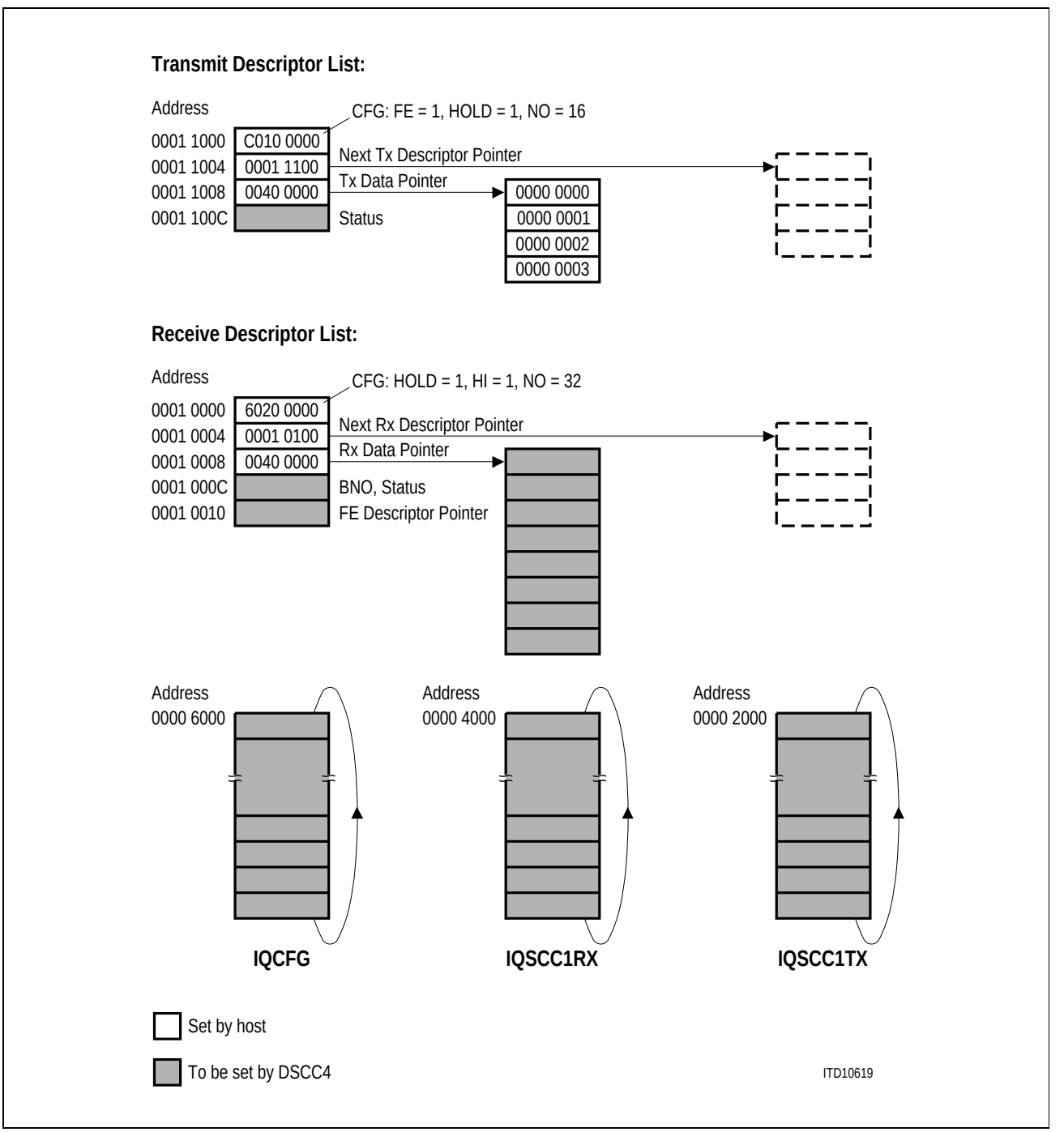


Figure 75 Data Structures in shared Memory before Transmission

Table 37 Register Initialization for HDLC Transparent Mode 0, Test Loop

Register	Access <= (write) => (read)	Value	Meaning
GMODE	<=	0000 0000	RESET Value: - DMAC is controlled by HOLD bit - Little Endian - Default Priority Scheme - MFP configured as LBI (not needed in this example)
IQLENR1	<=	0000 0000	RESET Value: Size of ring buffers: 32 entries
IQLENR2	<=	0000 0000	RESET Value: Size of ring buffers: 32 entries
IQSCC1RXBAR	<=	0000 2000	IQ Base Address for SCC1,RX
IQSCC1TXBAR	<=	0000 4000	IQ Base Address for SCC1,TX
IQCFGBAR	<=	0000 6000	IQ Base Address for CFG
FIFOCR1	<=	07C0 0000	max. possible buffer of TFIFO reserved for SCC1: 124 32-bit words
FIFOCR2	<=	0040 0000	Watermark of TFIFO (SCC1 portion) is set to 2 (example). (As soon as less than two DWORDs are in the central TFIFO buffer, the TFIFO requests for more data.)
FIFOCR3	<=	0000 0000	RESET Value: Watermark of RFIFO is set to one. (As soon as one 32-bit word is stored in the RFIFO, the RFIFO requests for data transfer to shared memory.
FIFOCR4	<=	0000 0000	RESET Value: Watermark of TFIFO forward threshold (SCC1 portion) is set to one. (As soon as at least one 32-bit word is in the central TFIFO, the TFIFO transfers data to SCC1 transmit FIFO.)

Table 37 Register Initialization for HDLC Transparent Mode 0, Test Loop

Register	Access <= (write) => (read)	Value	Meaning
GCMDR	<=	2220 0001	Command Bits: - Configure IQ SCC1 RX - Configure IQ SCC1 TX - Configure IQ CFG - Action Request
DSCC4 performs the configuration and requests for the bus to transfer a CFG interrupt vector to the IQCFG in shared memory			
GSTAR	=>	0020 0001	Indication Bits: - CFG interrupt indicated - Action Request Acknowledge indicated
GSTAR	<=	0020 0001	Acknowledge the interrupt indications: - CFG interrupt - Action Request Acknowledge
1st entry of IQCFG in shared memory	=>	A000 0001	Indication Bits: - CFG interrupt queue ID - Action Request Acknowledge indicated
CCR0 (SCC1)	<=	8000 0016	Power Up NRZ HDLC Clock Mode 6b, assuming that a clock is provided on XTAL1
CCR1 (SCC1)	<=	0204 8100	TxD output driver select HDLC Transparent Mode 0 Test Loop FCTS=’1’
CCR2 (SCC1)	<=	0803 0008	Receiver active Continuous FLAG sequences as interframe time fill RFTH=’011’ (default value)
IMR (SCC1)	<=	FFFA EF3D	Interrupts are enabled as follows: ALLS, XDU, XPR, RDO, RFS, RFO
CMDR (SCC1)	<=	0101 0000	Commands: - Transmitter Reset - Receiver Reset

Table 37 Register Initialization for HDLC Transparent Mode 0, Test Loop

Register	Access <= (write) => (read)	Value	Meaning
SCC1 resets transmitter and receiver. After transmitter reset an XPR interrupt is generated. A corresponding interrupt vector is forwarded through the interrupt queue to the appropriate interrupt ring buffer in shared memory.			
GSTAR	=>	0200 0000	Indication Bit: - SCC1 TX interrupt indicated
GSTAR	<=	0200 0000	Indicated interrupt is acknowledged.
1st entry of IQSCC1TX in shared memory	=>	5200 1000	Indication Bits: - SCC1 TX interrupt queue ID - Caused by SCC - XPR interrupt indicated
CH1CFG	<=	0018 0000	Configuration of DMAC channels: - Enable all interrupts: RFI, TFI, RERR, TERR -Set commands: IDR, IDT
CH1BRDA	<=	0001 0000	Set base receive descriptor address. (See **figure 75**.)
CH1BTDA	<=	0001 1000	Set base transmit descriptor address. (See **figure 75**.)
GCMDR	<=	0000 0001	Command Bit: - Action Request
DSCC4 checks the CHiCFG registers and performs the configuration of the DMA channels as required. After configuration an appropriate interrupt vector is generated and forwarded to IQCFG in shared memory			
After configuration the DMAC transfers transmit data from the tx data buffer to the central TFIFO. These data are forwarded to the SCC1, which loops back the data at the serial port . The received data are forwarded to the central RFIFO. Then the DMAC transfers the receive data to the rx data buffer in shared memory. The data transmission is completed with appropriate interrupts: ALLS, RFS, HI, FI.			
GSTAR	=>	2220 0001	Indication Bits: - SCC1 RX interrupt indicated - SCC1 TX interrupt indicated - CFG interrupt indicated - Action Request Acknowledge indicated
GSTAR	<=	2220 0001	Indicated interrupts are acknowledged.

Table 37 Register Initialization for HDLC Transparent Mode 0, Test Loop

Register	Access <= (write) => (read)	Value	Meaning
2nd entry of IQCFG in shared memory	=>	A000 0001	Indication Bits: - CFG interrupt queue ID - Action Request Acknowledge indicated
2nd entry of IQSCC1TX in shared memory	=>	5002 0000	Indication Bits: - DMA(SCC) TX interrupt queue ID - Caused by DMA - FI interrupt indicated
3rd entry of IQSCC1TX in shared memory	=>	5204 0000	Indication Bits: - SCC1 TX interrupt queue ID - Caused by SCC - ALLS interrupt indicated
1st entry of IQSCC1RX in shared memory	=>	1200 0040	Indication Bits: - SCC1 RX interrupt queue ID - Caused by SCC - RFS interrupt indicated
2nd entry of IQSCC1RX in shared memory	=>	1006 0000	Indication Bits: - SCC1 RX interrupt queue ID - Caused by DMAC - HI, FI interrupt indicated

See **figure 76** for data structure in shared memory after data transmission.

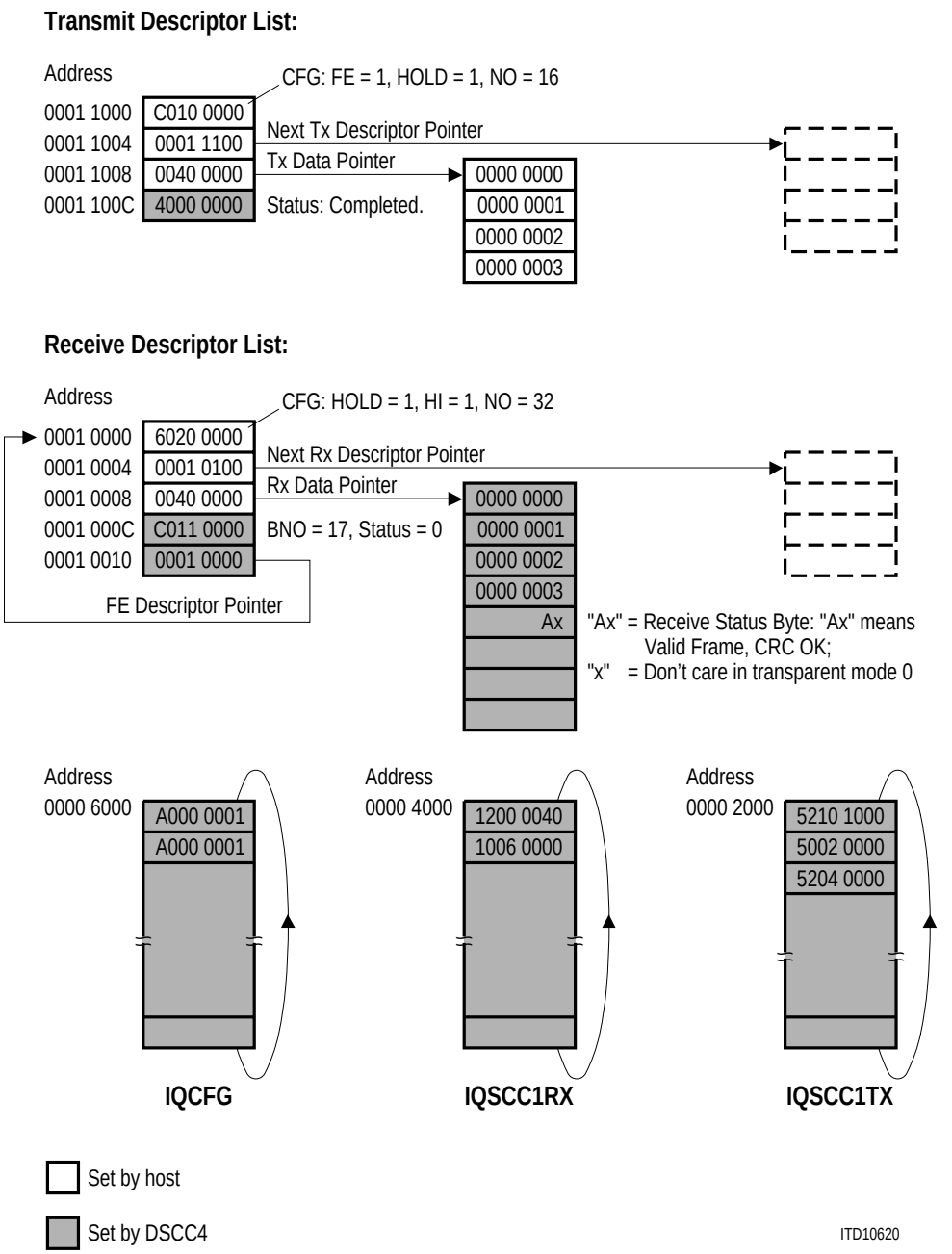


Figure 76 Data Stuctures in shared Memory after Transmission

10 Detailed Register Description

The DSCC4 has a set of PCI Configuration Space registers and several PCI memory mapped on-chip registers which allow configuration and control of the different functions within the DSCC4. Additionally, receive/transmit descriptors and data sections as well as interrupt status queues are located in the shared memory. The on-chip registers as well as the data structures and interrupt queues in the shared memory are described in this chapter.

10.1 Register Range Overview and Address Mapping

Table 38 Register Range and Address Mapping

Base Address	Range Offset Address	Register/Address Range		Description
		Total Range	Number of used DWORD registers	
None, selected via IDSEL signal (and bus commands in PCI interface mode)		2 KByte 000_H...$7FF_H$	16 000_H...$03C_H$	PCI Configuration Space Register Set
BAR1	0000_H	256 Byte 0000_H...$00FF_H$	59 0000_H...$00EC_H$	Global Registers
	0100_H	128 Byte 0100_H...$017F_H$	23 0100_H...0158_H	SCC0 Registers
	0180_H	128 Byte 0180_H...$01FF_H$	23 0180_H...$02D8_H$	SCC1 Registers
	0200_H	128 Byte 0200_H...$027F_H$	23 0200_H...0258_H	SCC2 Registers
	0280_H	128 Byte 0280_H...$02FF_H$	23 0280_H...$02D8_H$	SCC3 Registers
	0300_H	128 Byte 0300_H...$037F_H$	1 0300_H	LBI Control Register
	0380_H	128 Byte 0380_H...$03FF_H$	6 0380_H...0394_H	SSC Control Registers
	0400_H	128 Byte 0400_H...$047F_H$	3 0400_H...0408_H	GPP Control Registers
	0480_H	896 Byte 0480_H...$07FF_H$	0	(unused)
BAR2	0000_H	64 KByte 0000_H...$FFFF_H$	(16384)	Local Bus address range mapped into PCI (HOST) memory address space

10.2 PCI Configuration Space - Detailed Register Description

According to the PCI Specification V2.1 the DSCC4 supports the register layout of the predefined header region of the PCI Configuration Space.

Table 39 DSCC4: PCI Configuration Space Register Set

31	16	15	0	
Device ID (=2102$_H$)		Vendor ID(=110A$_H$)		00$_H$
Status (=0280$_H$)		Command (=0000$_H$)		04$_H$
Class Code (=028000$_H$)			Revision ID (=21$_H$)	08$_H$
BIST (=00$_H$)	Header Type (=00$_H$)	Latency Timer (=00$_H$)	Cache Line Size (=00$_H$)	0C$_H$
BAR1 (Base Address Register 1): base address of DSCC4 on-chip registers				10$_H$
BAR2 (Base Address Register 2): base address of Local Bus Interface				14$_H$
Base Address Register 3 (not used)				18$_H$
Base Address Register 4 (not used)				1C$_H$
Base Address Register 5 (not used)				20$_H$
Base Address Register 6 (not used)				24$_H$
Reserved				28$_H$
Reserved				2C$_H$
Expansion ROM Base Address				30$_H$
Reserved				34$_H$
Reserved				38$_H$
Max_Lat (=0A$_H$)	Min_Gnt (=03$_H$)	Interrupt Pin (=01$_H$)	Interrrupt Line (=00$_H$)	3C$_H$

Predefined header region of the PCI Configuration Space

The predefined header region has a size of 64 bytes and consists of fields that uniquely identify the device and allow the device to be controlled.

The DSCC4 supports the 64-byte header portion of the configuration space of PCI Specification Rev 2.1 to identify the device upon initial installation and power-up. None

of the additional locations 64-255 are used, nevertheless the DSCC4 responds to configuration read/write cycles within the address range 00_H to FC_H.

These configuration registers are addressed only using the PCI Configuration read/write cycles and using the IDSEL/DEVSEL handshake signals.

The PCI Configuration Space is also valid in de-multiplexed bus interface mode, i.e. pin DEMUX connected to V_{DD3}. In this case only signal IDSEL is used to select the PCI Configuration register set on read/write transactions.

The Base Address fields in the configuration space define the memory base addresses and the corresponding address range the DSCC4 will respond to. The first Base Address is the base address of the DSCC4's on-chip register range (Global control registers, SCC registers, LBI control registers, GPP Control registers, SSC registers). The second Base Address is the base address of the memory mapped LBI address space.

According to the PCI Specification, mapped address ranges are evaluated by writing all ones to the base address registers and reading back the value. The number of leading zeros determine the supported address range:

Table 40 PCI Base Address Ranges

Offset Address	Register	Register Read Value (after writing $0xFFFFFFFF_H$)	Supported Address Range
10_H	BAR1	$0xFFFFF800_H$	2 KByte (DSCC4 on chip registers)
14_H	BAR2	$0xFFFF0000_H$	64 KByte (Local bus address range mapped to PCI (HOST) address space)

The status/command register (offset address 04_H) of the PCI Configuration Space describes and determines the DSCC4 PCI system behavior and is described in details in **Table 41**:

Table 41 PCI Configuration Space: Status/Command Register

CPU Accessibility: **read/write (PCI Configuration Cycles)**
Reset Value: **0280 000$_H$**
Offset Address: **04$_H$ (PCI Configuration Space Offset Address)**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PCI Status Information															
	DPE	SSE	RMA	RTA	0	0	1	DPED	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PCI Command Bits															
	0	0	0	0	0	0	FBBE	SERRE	0	PER	0	0	SC	BM	MS	IOS

*Note: Bit locations containing a ’0’ or ’1’ are hardwired status and configuration settings specifying a fixed device behavior. These bit locations are also described in **Table 42**.*

Table 42 Status and Command register bits

Bit Location	Symbol	Description
31	DPE	**Detected Parity Error** This bit is set by the device whenever it detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the Command register).
30	SSE	**Signaled System Error** This bit will be set when • the $\overline{\text{SERR}}$ (SERRE) Enable bit is set in the Command register and one of the following events occured: 1. A transaction in which the DSCC4 acts as a master is terminated with master abort. 2. A transaction in which the DSCC4 acts as a master is terminated with target abort by the involved target. 3. The transaction has an address parity error and the Parity Error Response bit is set.
29	RMA	**Received Master Abort** This bit is set whenever the DSCC4 aborts a transaction with master abort. This occurs when no device responds.
28	RTA	**Received Target Abort** This bit is set whenever a device responds to a master transaction of the DSCC4 with a target abort.
27	0_B	**Signaled Target Abort** The DSCC4 will never signal “Target Abort”.
26..25	01_B	**DEVSEL Timing** The DSCC4 is a medium device.
24	DPED	**Data Parity Error Detected** This bit is set when the following three conditions are met: 1. the device asserted PERR itself or observed PERR asserted; 2. the device setting the bit acted as the bus master for the transaction in which the error occurred; 3. and the Parity Error Response bit is set in the Command register.
23	1_B	**Fast Back-to-Back Capable** The DSCC4 is fast Back-to-Back capable.

Table 42 Status and Command register bits (cont'd)

22	0_B	**UDF Supported** No UDFs are supported by the DSCC4.
21	0_B	**66 MHz Capable** The DSCC4 is not 66 MHz capable.
20...16	00000_B	Reserved
15...10	000000_B	Reserved
9	FBBE	**Fast Back-to-Back enable** A value of '1' means the DSCC4 is allowed to generate fast Back-to-Back transactions to different agents. A value of '0' means the DSCC4 is only allowed to generate fast Back-to-Back transaction to the same agent.
8	SERRE	**$\overline{\text{SERR}}$ Enable** A value of '1' enables the $\overline{\text{SERR}}$ driver. A value of '0' disables the $\overline{\text{SERR}}$ driver.
7	0_B	Wait Cycle Control The DSCC4 does never perform address/data stepping.
6	PER	**Parity Error Response** When this bit is set the DSCC4 will take its normal action when a parity error is detected. When this bit is '0' the DSCC4 ignores any parity errors that it detects and continues normal operation.
5	0_B	VGA Palette Snoop The DSCC4 is no VGA-Device.
4	0_B	Memory Write and Invalidate Enable The "Invalidate" command is not supported by the DSCC4.
3	SC	**Special Cycles** All special cycles are ignored. *Note: Although this bit can be set it has no effect.*
2	BM	**Bus Master** A value of '1' enables the bus master capability. *Note: Before giving the first action request it is necessary to set this bit.*
1	MS	**Memory Space** A value of '1' allows the DSCC4 to respond to Memory Space Addresses. *Note: This bit must be set before the first read/write transactions to the DSCC4 will be started.*

Table 42 Status and Command register bits (cont'd)

0	IOS	**IO Space** I/O Space accesses to the DSCC4 are not supported. *Note: Although this bit can be set it has no effect.*

10.3 On-Chip Registers Description

10.3.1 Global Registers - Detailed Register Description

10.3.1.1 Global Registers Overview

The DSCC4 global registers are used to configure and control the DMA controller, central FIFOs and genral device functions.

The full 32 bit address location of each global register consists of:

- Base Address Register 0 (PCI Configuration Space, address location 10_H)
- Register address offset, which is in the range 0000_H ...$00FC_H$

All registers are 32-bit organized registers.

Table 43 provides an overview about all global registers:

Table 43 DSCC4 Global Register Overview

Offset	Register	Meaning
General registers:		
0000_H	GCMDR	Global Command Register
0004_H	GSTAR	Global Status Register
0008_H	GMODE	Global Mode Register
Interrupt Queue (IQ) specific registers (and FIFO Control 4 register):		
$000C_H$	IQLENR0	IQ Length Register 0
0010_H	IQLENR1	IQ Length Register 1
0014_H	IQSCC0RXBAR	IQ SCC0 RX Base Address Reg.
0018_H	IQSCC1RXBAR	IQ SCC1 RX Base Address Reg.
$001C_H$	IQSCC2RXBAR	IQ SCC2 RX Base Address Reg.
0020_H	IQSCC3RXBAR	IQ SCC3 RX Base Address Reg.
0024_H	IQSCC0TXBAR	IQ SCC0 TX Base Address Reg.
0028_H	IQSCC1TXBAR	IQ SCC1 TX Base Address Reg.
$002C_H$	IQSCC2TXBAR	IQ SCC2 TX Base Address Reg.
0030_H	IQSCC3TXBAR	IQ SCC3 TX Base Address Reg.
0034_H	FIFOCR4	FIFO Control Register 4
0038_H	*RESERVED*	-

Table 43 DSCC4 Global Register Overview (cont'd)

Offset	Register	Meaning
$003C_H$	IQCFGBAR	IQ CFG Base Address Reg.
0040_H	IQPBAR	IQ Peripheral Base Address Reg.
DMA Controller (DMAC) specific registers:		
0044_H	FIFOCR1	FIFO Control Register 1
0048_H	FIFOCR2	FIFO Control Register 2
$004C_H$	FIFOCR3	FIFO Control Register 3
0050_H	CH0CFG	Channel 0 Configuration Register
0054_H	CH0BRDA	Channel 0 Base Rx Descr Address
0058_H	CH0BTDA	Channel 0 Base Tx Descr Address
$005C_H$	CH1CFG	Channel 1 Configuration Register
0060_H	CH1BRDA	Channel 1 Base Rx Descr Address
0064_H	CH1BTDA	Channel 1 Base Tx Descr Address
0068_H	CH2CFG	Channel 2 Configuration Register
$006C_H$	CH2BRDA	Channel 2 Base Rx Descr Address
0070_H	CH2BTDA	Channel 2 Base Tx Descr Address
0074_H	CH3CFG	Channel 3 Configuration Register
0078_H	CH3BRDA	Channel 3 Base Rx Descr Address
$007C_H$	CH3BTDA	Channel 3 Base Tx Descr Address
0080_H	*RESERVED*	-
...		
0094_H		
0098_H	CH0FRDA	Channel 0 First Rx Descr Address
$009C_H$	CH1FRDA	Channel 1 First Rx Descr Address
$00A0_H$	CH2FRDA	Channel 2 First Rx Descr Address
$00A4_H$	CH3FRDA	Channel 3 First Rx Descr Address
$00A8_H$	*RESERVED*	-
$00AC_H$	*RESERVED*	-
$00B0_H$	CH0FTDA	Channel 0 First Tx Descr Address
$00B4_H$	CH1FTDA	Channel 1 First Tx Descr Address
$00B8_H$	CH2FTDA	Channel 2 First Tx Descr Address

Table 43 DSCC4 Global Register Overview (cont'd)

Offset	Register	Meaning
$00BC_H$	CH3FTDA	Channel 3 First Tx Descr Address
$00C0_H$	*RESERVED*	-
$00C4_H$	*RESERVED*	-
$00C8_H$	CH0LRDA	Channel 0 Last Rx Descr Address
$00CC_H$	CH1LRDA	Channel 1 Last Rx Descr Address
$00D0_H$	CH2LRDA	Channel 2 Last Rx Descr Address
$00D4_H$	CH3LRDA	Channel 3 Last Rx Descr Address
$00D8_H$	*RESERVED*	-
$00DC_H$	*RESERVED*	-
$00E0_H$	CH0LTDA	Channel 0 Last Tx Descr Address
$00E4_H$	CH1LTDA	Channel 1 Last Tx Descr Address
$00E8_H$	CH2LTDA	Channel 2 Last Tx Descr Address
$00EC_H$	CH3LTDA	Channel 3 Last Tx Descr Address
$00F0_H$	*RESERVED*	-
...		
$00FC_H$		

10.3.1.2 Global Registers Description

Each register description is organized in three parts:

- a head with general information about reset value, access type (read/write), offset address and usual handling;
- a table containing the bit information (name of bit positions);
- a table containing the detailed description of each bit.

Table 44 GCMDR: Global Command Register

CPU Accessibility: **read/write**
Reset Value: **0000 0200$_H$**
Offset Address: **0000$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Interrupt Queue Configuration Commands															
	CFG IQ SCC 3 RX	CFG IQ SCC 2 RX	CFG IQ SCC 1 RX	CFG IQ SCC 0 RX	CFG IQ SCC 3 TX	CFG IQ SCC 2 TX	CFG IQ SCC 1 TX	CFG IQ SCC 0 TX	0	0	CFG IQ CFG	CFG IQ P	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Transmit Poll Requests / Interrupt Mask AR								Action Request (AR)							
	0	0	TXPR3	TXPR2	TXPR1	TXPR0	IMAR	0	0	0	0	0	0	0	0	AR

CFGIQ SCC3RX	**Configure Interrupt Queue SCC3 Receive**	(Channel RX 3)
CFGIQ SCC2RX	**Configure Interrupt Queue SCC2 Receive**	(Channel RX 2)
CFGIQ SCC1RX	**Configure Interrupt Queue SCC1 Receive**	(Channel RX 1)
CFGIQ SCC0RX	**Configure Interrupt Queue SCC0 Receive**	(Channel RX 0)

Only valid, if action request bit ’AR’ is set.
The DSCC4 DMA (interrupt) controller will transfer interrupt vectors generated by the dedicated SCC receiver (3..0) to the corresponding interrupt queue which must be configured via ’CFGIQSCCiRX’ command bits:

bit=’0’ The DSCC4 DMA (interrupt) controller does NOT configure/re-configure the corresponding interrupt queue, if action request bit ’AR’ is set to ’1’.

bit=’1’ Causes the DSCC4 DMA (interrupt) controller to configure/re-configure the corresponding interrupt queue, if action request bit ’AR’ is set to ’1’.
On action request, the DMA (interrupt) controller will evaluate the corresponding interrupt queue base address and length registers which must have been programmed by software before.

CFGIQ SCC3TX **Configure Interrupt Queue SCC3 Transmit** (Channel TX 3)

CFGIQ SCC2TX **Configure Interrupt Queue SCC2 Transmit** (Channel TX 2)

CFGIQ SCC1TX **Configure Interrupt Queue SCC1 Transmit** (Channel TX 1)

CFGIQ SCC0TX **Configure Interrupt Queue SCC0 Transmit** (Channel TX 0)

Only valid, if action request bit ’AR’ is set.
The DSCC4 DMA (interrupt) controller will transfer interrupt vectors generated by the dedicated SCC transmitter (3..0) to the corresponding interrupt queue which must be configured via ’CFGIQSCCiTX’ command bits:

bit=’0’ The DSCC4 DMA (interrupt) controller does NOT configure/re-configure the corresponding interrupt queue, if action request bit ’AR’ is set to ’1’.

bit=’1’ Causes the DSCC4 DMA (interrupt) controller to configure the corresponding interrupt queue, if action request bit ’AR’ is set to ’1’.
On action request, the DMA (interrupt) controller will evaluate the corresponding interrupt queue base address and length registers which must have been programmed by software before.

CFGIQCFG **Configure Interrupt Queue Configuration** (-)

Only valid, if action request bit ’AR’ is set.
The DSCC4 DMA (interrupt) controller will transfer action request acknowledge/failure interrupt vectors to the configuration interrupt queue which must be configured via ’CFGIQCFG’ command bits:

bit=’0’ The DSCC4 DMA (interrupt) controller does NOT configure/re-configure the configuration interrupt queue, if action request bit ’AR’ is set to ’1’.

bit=’1’ Causes the DSCC4 DMA (interrupt) controller to configure the configuration interrupt queue, if action request bit ’AR’ is set to ’1’.
On action request, the DMA (interrupt) controller will evaluate the configuration interrupt queue base address and length registers which must have been programmed by software before.

CFGIQP **Configure Interrupt Queue Peripheral** (-)

Only valid, if action request bit ’AR’ is set.
The DSCC4 DMA (interrupt) controller will transfer interrupt vectors generated by the Local Bus Interface (LBI) to the peripheral interrupt queue which must be configured via ’CFGIQP’ command bits:

bit=’0’ The DSCC4 DMA (interrupt) controller does NOT configure/re-configure the peripheral interrupt queue, if action request bit ’AR’ is set to ’1’.

bit=’1’ Causes the DSCC4 DMA (interrupt) controller to configure the peripheral interrupt queue, if action request bit ’AR’ is set to ’1’.
On action request, the DMA (interrupt) controller will evaluate the peripheral interrupt queue base address and length registers which must have been programmed by software before.

TXPRi (i=3...0) **Transmit Poll Request Channel i** (Channel TX 3...0)

Self-clearing command bit, only valid in ’HOLD’ bit controlled DMA controller mode (bit CMODE = ’0’ in register GMODE):

TXPRi=’0’ No Transmit Poll Request is performed. The corresponding DMA controller transmit channel is stopped when HOLD=’1’ has been detected in the current transmit descriptor.

TXPRi=’1’ Setting this bit to ’1’, when HOLD=’1’ has been detected in the current transmit descriptor, will cause the DSCC4 to poll the ’HOLD’ bit in the current transmit descriptor, i.e. the DSCC4 reads the configuration word (DWORD 0) and next descriptor address (DWORD 1) of the current descriptor again. If the ’HOLD’ bit is detected cleared (’0’), the DMA controller will branch to the next descriptor. When the DMA controller is not in ’HOLD’ state, this command is discarded.

IMAR **Interrupt Mask Action Request** (-)

On any action request, the DSCC4 will generate either an ’action request acknowledge’ or an ’action request failed’ interrupt vector which is transferred into the configuration interrupt queue. These interrupts can be masked via bit ’IMAR’:

IMAR=’0’ ’action request acknowledge’ and ’action request failed’ interrupt vectors respectively are generated and transferred into the configuration interrupt queue.

IMAR=’1’ (Reset value)
’action request acknowledge’ and ’action request failed’ interrupt vectors respectively are NOT generated (and thus NOT transferred into the configuration interrupt queue).

AR **Action Request** (-)

Self-clearing command bit:

AR=’0’ No action request is performed.

AR=’1’ If this bit is set to ’1’, the DMA controller will evaluate:

- register GCMDR for all interrupt queue configuration commands;
- all DMA channel specific configuration registers (CHiCFG, i=3...0) for reset and initialization commands.

Any command (command bit set to ’1’) will cause the corresponding configuration process to start.
A ’action request acknowledge’ or ’action request failed’ interrupt is generated after completion of all configuration processes and a corresponding status bit is set in register GSTAR.

Table 45 GSTAR: Global Status Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

Offset Address: **0004$_H$**

typical usage: written by DSCC4 as interrupt indication

evaluated by CPU and written as interrupt confirmation

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Queue Specific Interrupt Indication															
	II SCC 3 RX	II SCC 2 RX	II SCC 1 RX	II SCC 0 RX	II SCC 3 TX	II SCC 2 TX	II SCC 1 TX	II SCC 0 TX	0	0	II CFG	0	II P SSC	II P LBI	0	II P GPP

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									Action Request Result Status							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ARF	ARACK

The Global Status Register indicates whether an action request was executed successfully or not. It also gives information about the interrupt source and which interrupt queue has been written to when $\overline{\text{INTA}}$ is activated.

Ten interrupt queues are provided:

– four queues for receive interrupt vectors of the SCCs (SCCi, i=0...3)
– four queues for transmit interrupt vectors of the SCCs (SCCi, i=0...3)
– one queue for configuration interrupt vectors (action request acknowledge/failed)
– one queue for interrupts of the internal peripherals (SSC, LBI, and GPP).

To clear any bit in the status register, the host CPU must set the corresponding bit to “1“ by register write access. Signal $\overline{\text{INTA}}$ will be deasserted by the DSCC4 if ALL GSTAR indications are cleared.

IISCC3RX **Interrupt Indication Queue SCC3 Receive** (Channel RX 3)
IISCC2RX **Interrupt Indication Queue SCC2 Receive** (Channel RX 2)
IISCC1RX **Interrupt Indication Queue SCC1 Receive** (Channel RX 1)
IISCC0RX **Interrupt Indication Queue SCC0 Receive** (Channel RX 0)

These bits indicate whether at least one new interrupt vector was transferred into the corresponding receive interrupt queue:

bit='0' No new interrupt vector was transferred into the corresponding queue.

bit='1' At least one new interrupt vector was transferred into the corresponding queue.

IISCC3TX **Interrupt Indication Queue SCC3 Transmit** (Channel TX 3)
IISCC2TX **Interrupt Indication Queue SCC2 Transmit** (Channel TX 2)
IISCC1TX **Interrupt Indication Queue SCC1 Transmit** (Channel TX 1)
IISCC0TX **Interrupt Indication Queue SCC0 Transmit** (Channel TX 0)

These bits indicate whether at least one new interrupt vector was transferred into the corresponding transmit interrupt queue:

bit='0' No new interrupt vector was transferred into the corresponding queue.

bit='1' At least one new interrupt vector was transferred into the corresponding queue.

IICFG **Interrupt Indication Configuration Queue** (-)

This bit indicates whether at least one new interrupt vector was transferred into the configuration interrupt queue:

bit='0' No new interrupt vector was transferred into the configuration interrupt queue.

bit='1' At least one new interrupt vector was transferred into the configuration interrupt queue.

IIPSSC **Interrupt Indication Peripheral Queue (SSC Interrupt)** (-)

This bit indicates whether at least one new **SSC** interrupt vector was transferred into the peripheral interrupt queue:

bit='0' No new **SSC** interrupt vector was transferred into the peripheral interrupt queue.

bit='1' At least one new **SSC** interrupt vector was transferred into the peripheral interrupt queue.

IIPLBI **Interrupt Indication Peripheral Queue (LBI Interrupt)** (-)

This bit indicates whether at least one new **LBI** interrupt vector was transferred into the peripheral interrupt queue:

bit='0' No new **LBI** interrupt vector was transferred into the peripheral interrupt queue.

bit='1' At least one new **LBI** interrupt vector was transferred into the peripheral interrupt queue.

IIPGPP **Interrupt Indication Peripheral Queue (GPP Interrupt)** (-)

This bit indicates whether at least one new **GPP** interrupt vector was transferred into the peripheral interrupt queue:

bit='0' No new **GPP** interrupt vector was transferred into the peripheral interrupt queue.

bit='1' At least one new **GPP** interrupt vector was transferred into the peripheral interrupt queue.

ARF **Action Request Failed Status** (-)

This bit indicates that an action request command was completed with an 'action request failed' condition:

bit='0' No action request was performed or no 'action request failed' condition occured completing an action request.

bit=’1’ The last action request command was completed with an ’action request failed’ condition.

ARACK **Action Request Acknowledge Status** (-)

This bit indicates that an action request command was completed successfully:

bit=’0’ No action request was performed or completed successfully.

bit=’1’ The last action request command was completed successfully.

Table 46 GMODE: Global Mode Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0008$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	General Configuration															
	0	0	0	0	0	0	0	0	0	0	OSC PD	LCD(1:0)		PERCFG(2:0)		

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	General Configuration															
	SPRI	CHN(1:0)		0	0	0	0	0	0	0	0	0	0	ENDIAN	DBE	CMODE

OSCPD **Oscillator Power Down** (-)

This bit switches the internal oscillator (used if a crystal is connected to pins XTAL1 and XTAL2) in power-down (stand-by) mode:

OSCPD=’0’ Normal operation. The internal oscillator works, if a crystal is connected to pins XTAL1 and XTAL2.

OSCPD=’1’ The internal oscillator is in power-down mode.

LCD(1:0) **LBI Clock Division** (-)

The internal LBI operating clock (which is monitored on output pin LCLKO) is internally derived from the PCI clock input pin CLK and a clock division unit. The division factor can be selected via this bit field:

LCD = ’00’ *Reserved*, do not use.

LCD = ’01’ LCLK = CLK / 2

LCD = ’10’ LCLK = CLK / 4

LCD = ’11’ LCLK = CLK / 16

Note: The LBI clock signal monitored on pin LCLKO is an asymmetric clock signal. The LBI clock high phase time is always equal the PCI clock high phase time (typical 15 nano seconds). The LBI clock low phase time is extended respectively.

PERCFG (2:0) **Peripheral Block Configuration** (-)

The peripheral block basically consists of the functions
- Local Bus Interface (LBI)
- General Purpose Port (GPP)
- Synchronous Serial Controller (SSC)

which can be operated in various combinations/configurations.
Bit field ’PERCFG’ selects the peripheral configuration and switches the multiplexed signal pins accordingly:

PCI Interface Mode (DEMUX pin connected to V_{SS}):

PERCFG	Signal Pin Groups		
(2:0)	109, 108 101..96	119..112	143..135 128..123
’000’	LA(15..8)	LA(7..0)	LD(15..0)
’001’	*Reserved*. Do not use.		
’010’	GP(15..8)	LA(7..0)	LD(15..0)
’011’	GP(15..8)	SSC	LAD(15..0)
’100’	GP(15..8)	GP(7..0)	LAD(15..0)
’101’, ’110’, ’111’	*Reserved*. Do not use.		

DEMUX Interface Mode (DEMUX pin connected to V_{DD3}):

Bit field ’PERCFG’ is not valid. All 32 multiplexed signals are used as DEMUX address bus A(31:0):

PERCFG	Signal Pin Groups		
(2:0)	109, 108 101..96	119..112	143..135 128..123
’xxx’	A(15..8)	A(7..0)	A(31..0)

SPRI **Select Priority** (-)

This bit selects one DMA channel (transmit and receive) as a high priority channel. The dedicated channel 3..0 is selected via bit field ’CHN’.
If enabled, the selected serial channel (SCCi) is serviced by the DMA controller with high priority, whereas the remaining channels are in a second round-robbin priority group:

SPRI=’0’ No SCC is selected as high priority channel. All SCCs are serviced in a round-robbin manner.
Bit field ’CHN’ is don’t care.

SPRI=’1’ The SCC selected via bit field ’CHN’ is serviced as high priority channel, i.e. requests to central receive and transmit FIFOs are priorized against the other channels.

CHN(1:0) **Channel Number (for highest priority)** (-)

This bit field is only valid if bit ’SPRI’ is set to ’1’ and specifies the serial channel which is serviced with highest priority by the DMA controller:

CHN=’00’ Channel 0 (transmit and receive) is selected as high priority channel.

CHN=’01’ Channel 1 (transmit and receive) is selected as high priority channel.

CHN=’10’ Channel 2 (transmit and receive) is selected as high priority channel.

CHN=’11’ Channel 3 (transmit and receive) is selected as high priority channel.

ENDIAN **Endian Selection** (-)

This bit selects whether receive and transmit data buffers are handled with Intel or Motorola like byte ordering:

ENDIAN=’0’ The DWORDs of receive and transmit data buffers are evaluated based on a little endian (Intel like) byte ordering.

ENDIAN=’1’ The DWORDs of receive and transmit data buffers are evaluated based on a big endian (Motorola like) byte ordering. Therefore the byte ordering is automatically swapped by the DMA controller.

Note: The little/big endian selection (byte-swapping) effects only DSCC4 operation on receive and transmit data buffer sections. Descriptor reads and writes as well as register access is not effected anyway.

DBE **DEMUX Burst Enable** (-)

This bit is only valid if the DSCC4 is running in de-multiplexed bus interface mode, i.e. pin DEMUX connected to V_{DD3}.
By default value, the burst functionality is disabled in DEMUX mode and can be enabled via setting this bit. However burst length is limited to 4 DWORDs in DEMUX mode (15 DWORDs in PCI mode):

DBE=’0’ Burst functionality is disabled. The DSCC4 will perform all transactions to the host memory using single DWORD read/write bus transfers.

DBE=’1’ Burst functionality is enabled. The DSCC4 performs burst transfers for operation on descriptors and data sections (like in PCI mode). Burst length is limited to 4 DWORDs maximum.

CMODE **DMA Control Mode** (-)

This bit selects between the two global DMA controller mechanisms for handling descriptor chain end conditions:

CMODE=’0’ ’HOLD’ bit control mode.
The descriptor chain end condition is controlled via the ’HOLD’ bit in each receive/transmit descriptor.

CMODE=’1’ Last Receive/Transmit Descriptor Address mode.
The descriptor chain end condition is controlled via registers LRDA/LTDA.

Table 47 IQLENR0: Interrupt Queue Length Register 0

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **000C$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Interrupt Queues Length Configuration															
	IQSCC0RXLEN				IQSCC1RXLEN				IQSCC2RXLEN				IQSCC3RXLEN			

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Interrupt Queues Length Configuration															
	IQSCC0TXLEN				IQSCC1TXLEN				IQSCC2TXLEN				IQSCC3TXLEN			

IQSCC3 RXLEN	**Interrupt Queue SCC3 Receive Length**	(Channel RX 3)
IQSCC2 RXLEN	**Interrupt Queue SCC2 Receive Length**	(Channel RX 2)
IQSCC1 RXLEN	**Interrupt Queue SCC1 Receive Length**	(Channel RX 1)
IQSCC0 RXLEN	**Interrupt Queue SCC0 Receive Length**	(Channel RX 0)

These bit fields determine the length of the corresponding receive interrupt queue (related to the respective SCC receive channel):

Queue length: Queue Length = (1 + 'IQSCCiRXLEN') * 32 DWORDS,

'IQSCCiRXLEN' = 0...15

IQSCC3 TXLEN	**Interrupt Queue SCC3 Transmit Length**	(Channel TX 3)
IQSCC2 TXLEN	**Interrupt Queue SCC2 Transmit Length**	(Channel TX 2)
IQSCC1 TXLEN	**Interrupt Queue SCC1 Transmit Length**	(Channel TX 1)
IQSCC0 TXLEN	**Interrupt Queue SCC0 Transmit Length**	(Channel TX 0)

These bit fields determine the length of the corresponding transmit interrupt queue (related to the respective SCC transmit channel):

Queue length: Queue Length = (1 + 'IQSCCiTXLEN') * 32 DWORDS,

'IQSCCiTXLEN' = 0...15

Table 48 IQLENR1: Interrupt Queue Length Register 1

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0010$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Interrupt Queues Length Configuration															
	0	0	0	0	0	0	0	0	IQCFGLEN				IQPLEN			

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IQCFGLEN **Interrupt Queue Configuration Length** (-)

These bit field determins the length of the configuration interrupt queue:

Queue length: Queue Length = (1 + 'IQCFGLEN') * 32 DWORDS,

'IQCFGLEN' = 0...15

IQPLEN **Interrupt Queue Peripheral Length** (-)

These bit field determins the length of the peripheral interrupt queue:

Queue length: Queue Length = (1 + 'IQPLEN') * 32 DWORDS,

'IQPLEN' = 0...15

Table 49 IQSCCiRXBAR:
Interrupt Queue SCCi Receiver Base Address Register (i=0...3)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0014$_H$**	**0018$_H$**	**001C$_H$**	**0020$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IQSCCiRXBAR(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IQSCCiRXBAR(15:2)														0	0

IQSCCi RXBAR **i = 3...0** (RX Channel 3...0)

These registers determine the base address of the respective receive interrupt queue and can be located anywhere in the 32 bit address range.

Note: The interrupt queue base addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to ’0’.

Table 50 IQSCCiTXBAR:
Interrupt Queue SCCi Receiver Base Address Register (i=0...3)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0024$_H$**	**0028$_H$**	**002C$_H$**	**0030$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IQSCCiTXBAR(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IQSCCiTXBAR(15:2)														0	0

IQSCCi TXBAR **i = 3...0** (TX Channel 3...0)

These registers determine the base address of the respective transmit interrupt queue and can be located anywhere in the 32 bit address range.

Note: The interrupt queue base addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to '0'.

**Table 51 IQCFGBAR:
Interrupt Queue Configuration Base Address Register**

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **003C$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IQCFGBAR(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IQCFGBAR(15:2)														0	0

IQCFGBAR (-)

This register determins the base address of the configuration interrupt queue and can be located anywhere in the 32 bit address range.

Note: The interrupt queue base address must be 32-bit aligned, i.e. bit 1 and 0 must be set to '0'.

Table 52 IQPBAR:
Interrupt Queue Peripheral Base Address Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0040$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IQPBAR(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IQPBAR(15:2)														0	0

IQPBAR (-)

This register determins the base address of the peripheral interrupt queue and can be located anywhere in the 32 bit address range.

Note: The interrupt queue base address must be 32-bit aligned, i.e. bit 1 and 0 must be set to '0'.

Table 53 FIFOCR1: FIFO Control Register 1

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0044$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Central Transmit FIFO Partition Size															
	TFSIZE0					TFSIZE1					TFSIZE2					0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Central Transmit FIFO Partition Size															
	TFSIZE3					0	0	0	0	0	0	0	0	0	0	0

TFSIZEi (i = 0...3) **Transmit FIFO Size (Partition Channel i)** (TX Channel i)

These bit fields determine the channel specific partition size of the central transmit FIFO in multiples of 4 DWORDs:
partition size i = 'TFSIZEi' * 4 DWORDs,
range 0...124 DWORDs

Note: The entire size of all FIFO partitions must not exceed the total FIFO size of 128 DWORDs which is the responsibility of the software.

Note: If the complete central transmit FIFO is assigned to only one channel, the maximum partition size is 124 DWORDs.

Note: The minimum allowed partition size for used channels is 4 DWORDs, i.e. 'TFSIZEi' = 1.

Table 54 FIFOCR2: FIFO Control Register 2

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0048$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Central Transmit FIFO Refill Threshold															
	TFRTHRES0					TFRTHRES1					TFRTHRES2					0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Central Transmit FIFO Refill Threshold								Multipliers							
	TFRTHRES3					0	0	0	M4_0	M2_0	M4_1	M2_1	M4_2	M2_2	M4_3	M2_3

TFRTHRESi (i = 0...3) **Transmit FIFO Refill Threshold Channel i** (TX Channel i)

These bit fields determine the channel specific Transmit FIFO Refill Threshold for the corresponding channel i in number of DWORDs multiplied by its respective multiplier Mx_i.
This threshold controls DMAC operation towards the Host memory.
A watermark is calculated by:
*watermark = TFRTHRESH*Mx_i+1*.
As soon as the number of valid data in the transmit FIFO is less than *watermark* the DMA controller requests for new data from shared memory.
After initialization the DMAC fills the complete transmit FIFO; when the number of data in the transmit FIFO decreases below the *watermark* the transmit FIFO requests for refill. The *number* of data to be transferred into the transmit FIFO is calculated by:
number=TFSIZEi-watermark.

The transmit FIFO cannot be filled when the DMA channel is in internal HOLD state, i.e. 'HOLD' bit has been detected (GMODE.CMODE='0') or Last Descriptor Address matches the current descriptor address FTDA = LTDA (GMODE.CMODE='1').

Note: (1) The watermark has to be equal or lower than the specified size of the corresponding central transmit FIFO partition.
(2) The refill watermark must be at least 2 DWORDs higher than the forward watermark (refer also to register FIFOCR4).
(3) The minimum allowed TFRTHRESHi value for used channels is "1".

Mx_i (i = 0...3) **Multiplier 2 or 4** (TX Channel i)

These bits enable a multiplier 2 or 4 respectively for the corresponding 'TFRTHRESi' value:

M2_i = '0' The multiplier 'by 2' is disabled

M2_i = '1' The 'TFRTHRESi' bit field value is multiplied by 2.

M4_i = '0' The multiplier 'by 4' is disabled

M4_i = '1' The 'TFRTHRESi' bit field value is multiplied by 4.

Note: It is recommended not to set both multiplier enable bits of one channel to '1'.

Table 55 FIFOCR3: FIFO Control Register 3

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **004C$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	(unused)							Multipliers		Central Receive FIFO Threshold						
	0	0	0	0	0	0	0	M4	M2	RFTHRES						

RFTHRES **Receive FIFO Threshold** (-)

This bit field determins the central Receive FIFO Threshold in number of DWORDs multiplied by its respective multiplier ’M2’ or ’M4’.
This threshold controls DMAC operation towards the Host memory.
A watermark is calculated by:
*watermark = RFTHRES*Mx* .
When more data than specified by this watermark are available in the receive FIFO the DMA controller is requested to transfer the received data to the channel specific data buffers in the host memory until the central receive FIFO is empty. If no host memory buffer is available for a channel, since the internal HOLD state is reached, i.e. ’HOLD’ bit has been detected (GMODE.CMODE=’0’) or Last Descriptor Adress matches the current descriptor address FRDA = LRDA (GMODE.CMODE=’1’), no data can be transferred.

Note: The watermark has to be lower than the maximum central receive FIFO size of 128 DWORDs.

Mx **Multiplier 2 or 4** (-)

These bits enable a multiplier 2 or 4 respectively for the ’RFTHRES’ value:

M2 = ’0’ The multiplier ’by 2’ is disabled

M2 = ’1’ The ’RFTHRES’ bit field value is multiplied by 2.

M4 = ’0’ The multiplier ’by 4’ is disabled

M4 = ’1’ The ’RFTHRES’ bit field value is multiplied by 4.

Note: It is recommended not to set both multiplier enable bits to ’1’.

Table 56 FIFOCR4: FIFO Control Register 4

CPU Accessibility: **read/write**
Reset Value: **0000 000$_H$**
Offset Address: **0034$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Central Transmit FIFO Forward Thresholds															
	TFFTHRES3								TFFTHRES2							

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Central Transmit FIFO Forward Thresholds															
	TFFTHRES1								TFFTHRES0							

TFFTHRESi (i = 0...3) **Transmit FIFO Forward Threshold Channel i** (TX Channel i)

These bit fields determine the channel specific Transmit FIFO Forward Threshold for the corresponding channel i in number of DWORDs. This threshold controls DMAC operation towards the serial channels (SCCi).
A watermark is calculated by:
watermark = TFFTHRESH.
As soon as the number of valid data belonging to a new frame in the central transmit FIFO is greater than the watermark, the DMAC will provide transmit data to the corresponding SCC. Once having started one frame, the DMAC will ignore this threshold providing all available data of the current frame to the SCC. Threshold operation starts again with the beginning of a new frame. Frames shorter than the threshold will be transferred as soon as a frame end indication is detected by the DMAC.

*Note: The maximum allowed Transmit FIFO Forward Threshold is: TFFTHRESHi = (TFSIZEi * 4) - 1 DWORDs,*
whereas 'TFSIZEi' is the channel specific central transmit FIFO partition size programmed in register FIFOCR1

Note: Programming TFFTHRESHi to zero will disable the threshold causing the DMAC to transfer all data immediately. This may be useful for not frame/packet oriented data transmission, e.g. in ASYNC protocol mode.

Table 57 CHiCFG: Channel i Configuration Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0050$_H$**	**005C$_H$**	**0068$_H$**	**0074$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DMA Channel Interrupt Masks								DMA Channel Commands							
	0	0	0	0	MRFI	MTFI	MRERR	MTERR	0	RDR	RDT	IDR	IDT	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MRFI **Mask Receive FI-Interrupt (Channel i)** (RX Channel)

This bit enables/disables the receive FI-interrupt indication for the DMA channel, the register is dedicated to (i=3..0):

MRFI=’0’ FI-interrupt generation is enabled for the dedicated DMA receive channel.

MRFI=’1’ FI-interrupt generation is disabled for the dedicated DMA receive channel.

MTFI **Mask Transmit FI-Interrupt (Channel i)** (TX Channel)

This bit enables/disables the transmit FI-interrupt indication for the DMA channel, the register is dedicated to (i=3..0):

MTFI=’0’ FI-interrupt generation is enabled for the dedicated DMA transmit channel.

MTFI=’1’ FI-interrupt generation is disabled for the dedicated DMA transmit channel.

MRERR **Mask Receive ERR-Interrupt (Channel i)** (RX Channel)

This bit enables/disables the receive ERR-interrupt indication for the DMA channel, the register is dedicated to (i=3..0):

MRERR=’0’ ERR-interrupt generation is enabled for the dedicated DMA receive channel.

MRERR=’1’ ERR-interrupt generation is disabled for the dedicated DMA receive channel.

MTERR **Mask Transmit ERR-Interrupt (Channel i)** (TX Channel)

This bit enables/disables the transmit ERR-interrupt indication for the DMA channel, the register is dedicated to (i=3..0):

MTERR=’0’ ERR-interrupt generation is enabled for the dedicated DMA transmit channel.

MTERR=’1’ ERR-interrupt generation is disabled for the dedicated DMA transmit channel.

RDR **Reset DMA Receiver (Channel i)** (RX Channel)

Self-clearing command bit.
This command resets the specific DMA controller receive channel. After reset, the respective DMA channel is in its initial state equal to the reset state after power on.

RDT **Reset DMA Transmitter (Channel i)** (TX Channel)

Self-clearing command bit.
This command resets the specific DMA controller transmit channel. After reset, the respective DMA channel is in its initial state equal to the reset state after power on.

IDR **Initialize DMA Receiver (Channel i)** (RX Channel)

Self-clearing command bit.
This command causes the specific DMA receive channel to fetch the base descriptor address from register CHiBRDA and to branch to the corresponding descriptor. Afterwards normal DMA operation on the receive descriptor list is performed depending on the selected DMA control mode.

Note: To avoid unexpected DMA controller behavior, it is recommended to apply 'IDR' command only, if the specific DMA channel is in reset state.

IDT **Initialize DMA Transmitter (Channel i)** (TX Channel)

Self-clearing command bit.
This command causes the specific DMA transmit channel to fetch the base descriptor address from register CHiBTDA and to branch to the corresponding descriptor. Afterwards normal DMA operation on the transmit descriptor list is performed depending on the selected DMA control mode.

Note: To avoid unexpected DMA controller behavior, it is recommended to apply 'IDT' command only, if the specific DMA channel is in reset state.

Table 58 CHiBRDA:
Channel i Base Receive Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0054$_H$**	**0060$_H$**	**006C$_H$**	**0078$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiBRDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiBRDA(15:2)														0	0

CHiBRDA **i = 3...0** (RX Channel 3...0)

These registers determine the base address of the channel specific receive descriptor chain and can be located anywhere in the 32 bit address range.

Note: The descriptor base addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to '0'.

Table 59 CHiBTDA:
Channel i Base Transmit Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0058$_H$**	**0064$_H$**	**0070$_H$**	**007C$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiBTDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiBTDA(15:2)														0	0

CHiBTDA **i = 3...0** (TX Channel 3...0)

These registers determine the base address of the channel specific transmit descriptor chain and can be located anywhere in the 32 bit address range.

Note: The descriptor base addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to '0'.

Table 60 CHiFRDA:
Channel i First (Current) Receive Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**0098$_H$**	**009C$_H$**	**00A0$_H$**	**00A4$_H$**
typical usage:	written by DSCC4			
	evaluated by CPU			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiFRDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiFRDA(15:2)														0	0

CHiFRDA **i = 3...0** (RX Channel 3...0)

The DMA controller writes the first/current address of the channel specific receive descriptor chain to these registers, i.e. the address of the receive descriptor, the DMA receive channel i is currently working on. These registers are only valid, if the DMA controller is operating in Last Descriptor Address Mode (bit CMODE set to ’1’ in register GMODE).

Table 61 CHiFTDA:
Channel i First (Current) Transmit Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**00B0$_H$**	**00B4$_H$**	**00B8$_H$**	**00BC$_H$**
typical usage:	written by DSCC4			
	evaluated by CPU			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiFTDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiFTDA(15:2)														0	0

CHiFTDA **i = 3...0** (TX Channel 3...0)

The DMA controller writes the first/current address of the channel specific transmit descriptor chain to these registers, i.e. the address of the transmit descriptor, the DMA transmit channel i is currently working on.
These registers are only valid, if the DMA controller is operating in Last Descriptor Address Mode (bit CMODE set to ’1’ in register GMODE).

Table 62 CHiLRDA:
Channel i Last Receive Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**00C8$_H$**	**00CC$_H$**	**00D0$_H$**	**00D4$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiLRDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiLRDA(15:2)														0	0

CHiLRDA **i = 3...0** (RX Channel 3...0)

These registers determine the last descriptor address of the channel specific receive descriptor chain and can be located anywhere in the 32 bit address range. The last descriptor address is written by the CPU and marks the corresponding descriptor as the last descriptor in the receive descriptor chain.

After write access to one of these registers, the DMA controller again compares the first (current) receive descriptor address (register CHiFRDA) with the last descriptor address (register CHiLRDA), if the corresponding DMA controller channel was in internal HOLD state. If these addresses do not match any more, the DMA channel leaves the internal HOLD state, re-reads the next descriptor address of the current receive descriptor and continues operation.

These registers are only valid, if the DMA controller is operating in Last Descriptor Address Mode (bit CMODE set to ’1’ in register GMODE).

Note: The last descriptor addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to ’0’.

Table 63 CHiLTDA:
Channel i Last Transmit Descriptor Address Register (i=3...0)

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	Channel 0	Channel 1	Channel 2	Channel 3
Offset Address:	**00E0$_H$**	**00E4$_H$**	**00E8$_H$**	**00EC$_H$**
typical usage:	written by DSCC4			
	evaluated by CPU			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHiLTDA(31:16)															

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHiLTDA(15:2)														0	0

CHiFTDA **i = 3...0** (TX Channel 3...0)

These registers determine the last descriptor address of the channel specific transmit descriptor chain and can be located anywhere in the 32 bit address range. The last descriptor address marks the corresponding descriptor as the last descriptor in the transmit descriptor chain.

After write access to one of these registers, the DMA controller again compares the first (current) transmit descriptor address (register CHiFTDA) with the last descriptor address (register CHiLTDA), if the corresponding DMA controller channel was in internal HOLD state.If these addresses do not match any more, the DMA channel leaves the internal HOLD state, re-reads the next descriptor address of the current transmit descriptor and continues operation.

These registers are only valid, if the DMA controller is operating in Last Descriptor Address Mode (bit CMODE set to ’1’ in register GMODE).

Note: The last descriptor addresses must be 32-bit aligned, i.e. bit 1 and 0 must be set to ’0’.

10.3.2 SCC Registers - Detailed Register Description

10.3.2.1 SCC Registers Overview

The SCC registers are used to configure and control each of the four Serial Communication Controller (SCC). The complete SCC register set exists four times, i.e. once for each SCC, distinguished by a SCC specific offset address.

The full 32 bit address location of each SCC register consists of:

- Base Address Register 0 (PCI Configuration Space, address location 10_H)
- SCC specific offset address:
  SCC0: 0100_H
  SCC1: 0180_H
  SCC2: 0200_H
  SCC3: 0280_H

- Register address offset, which is in the range 00_H ...58_H

Most registers and register bit positions are shared by all SCC protocol modes (HDLC, ASYNC, BISYNC). Nevertheless the meaning (and name) of single bit positions might defer between different protocol modes. All registers are 32-bit organized registers.

Table 64 provides an overview about all SCC registers:

Table 64 SCC Register Overview

Register Offset	Access Type	Register		Valid in Mode(s)
00_H	w	CMDR	Command Register	H/A/B
04_H	r	STAR	Status Register	H/A/B
08_H	r/w	CCR0	Channel Configuration Register 0	H/A/B
$0C_H$	r/w	CCR1	Channel Configuration Register 1	H/A/B
10_H	r/w	CCR2	Channel Configuration Register 2	H/A/B
14_H	r/w	ACCM	ASYNC Control Character Map	H (PPP)
18_H	r/w	UDAC	User Defined ASYNC Character	H (PPP)
$1C_H$	r/w	TTSA	Transmit Time Slot Assignment Register	H/A/B
20_H	r/w	RTSA	Receive Time Slot Assignment Register	H/A/B
24_H	r/w	PCMMTX	PCM Mask for Transmit	H/A/B
28_H	r/w	PCMMRX	PCM Mask for Receive	H/A/B
$2C_H$	r/w	BRR	Baud Rate Register	H/A/B
30_H	r/w	TIMR	Timer Register	H/A/B
34_H	r/w	XADR	Transmit Address Register	H
38_H	r/w	RADR	Receive Address Register	H
$3C_H$	r/w	RAMR	Receive Address Mask Register	H
40_H	r/w	RLCR	Receive Length Check Register	H
44_H	r/w	XNXFR	XON/XOFF Register	A
48_H	r/w	TCR	Termination Character Register	A/B
$4C_H$	r/w	TICR	Transmit Immediate Character Register	A
50_H	r/w	SYNCR	Synchronization Character Register	B
54_H	r/w	IMR	Interrupt Mask Register	H/A/B
58_H	r	ISR	Interrupt Status Register	H/A/B

10.3.2.2 SCC Registers Description

Each register description is organized in three parts:

- a head with general information about reset value, access type (read/write), channel specific offset addresses and usual handling;
- a table containing the bit information (name of bit positions) distinguished for the three major protocol modes HDLC (H), ASYNC (A) and BISYNC (B);
- a table containing the detailed description of each bit; the corresponding modes, the bit is valid for, are marked again by a bracket term right beside the full bit name.

Table 65 CMDR: Command Register

CPU Accessibility: **write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0100$_H$**	**0180$_H$**	**0200$_H$**	**0280$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Transmitter Commands								Receiver Commands							
H	0	0	0	0	0	0	0	XRES	0	0	0	0	0	0	0	RRES
A	0	0	0	0	0	0	0	XRES	0	0	0	0	0	0	RFRD	RRES
B	0	0	0	0	0	0	0	XRES	0	0	0	0	0	HUNT	RFRD	RRES

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Internal Commands								Remote Control							
H	0	0	0	0	0	0	0	STI	0	0	0	0	0	0	0	RNR
A	0	0	0	0	0	0	0	STI	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	STI	0	0	0	0	0	0	0	0

XRES **Transmitter Reset Command** (all modes)

Self-clearing command bit:

XRES=’1’ The SCC transmit FIFO is cleared and the transmitter protocol engines are reset to their initial state.
The SCC transmit FIFO requests new transmit data from the central TFIFO immediately after transmitter reset procedure.
A transmitter reset command is recommended after all changes in protocol mode configurations (switching between the protocol engines HDLC/ASYNC/BISYNC or sub-modes of HDLC).

HUNT **Enter Hunt State Command** (bisync mode)

Self-clearing command bit:

HUNT=’1’ This command forces the receiver to enter its ’HUNT’ state immediately. Thus synchronization is ’lost’ and the receiver starts searching for new SYNC characters.

RFRD **Receive FIFO Read Enable Command** (async/bisync mode)

Self-clearing command bit:

RFRD=’1’ This command forces insertion of a ’block end’ indication in the SCC receive FIFO. If the receive FIFO is not empty (bit ’RFNE’ set in register STAR) and data was not transferred to the central RFIFO because neither the receive threshold is exceeded nor a block end indication is stored, this command forces data transfer to the central RFIFO.

Note: This command always generates a ’block-end’ indication. If the receive FIFO was empty, the DMA Controller will finish the current receive descriptor with receive byte number zero and frame-end/ block-end indication bit set.

RRES **Receiver Reset Command** (all modes)

Self-clearing command bit:

RRES=’1’ The SCC receive FIFO is cleared and the receiver protocol engines are reset to their initial state.
The SCC receive FIFO accepts new receive data from the protocol engine immediately after receiver reset procedure.
It is recommended to disable data reception before issuing a receiver reset command by setting bit CCR2.RAC = ’0’ and enabling data reception afterwards.
A ’receiver reset command’ is recommended after all changes in protocol mode configurations (switching between protocol the engines HDLC/ASYNC/BISYNC or sub-modes of HDLC).

STI **Start Timer Command** (all modes)

Self-clearing command bit:
HDLC Automde:
In HDLC Automode the timer is operating in ’internal timer mode’. The timer is started automatically by the SCC when an I-Frame is sent out and needs to be acknowledged.
If the ’STI’ command is issued by software:

STI=’1’ An S-Frame with poll bit set is sent out and the internal timer is started expecting an acknowledge from the remote station via an I- or S-Frame.
The timer is stopped after receiving an acknowledge otherwise the timer expires generating a timer interrupt.

All protocol modes except HDLC Automode:
In these modes the timer is operating in ’external timer mode’.

STI=’1’ This commands starts timer operation in ’external timer mode’.
The timer can be stopped by rewriting register TIMR. If the timer expires a timer interrupt is generated.

Note: Internal/External timer operation mode must be selected by bit 'TMD' in register TIMR.

RNR **Receiver Not Ready Command** (hdlc mode)

NON self-clearing command bit:
This command bit is significant in Automode only.

RNR='0' Forces the receiver to enter its 'receiver-ready' state. The receiver acknowledges received poll or I-Frames with a 'receiver-ready' indication.

RNR='1' Forces the receiver to enter its 'receiver-not-ready' state. The receiver acknowledges received poll or I-Frames with a 'receiver-not-ready' indication.

Table 66 STAR: Status Register

CPU Accessibility: **read**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0104$_H$**	**0184$_H$**	**0204$_H$**	**0284$_H$**
typical usage:	updated by DSCC4			
	read and evaluated by CPU			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Command Status				Transmitter Status				Receiver Status					Automode Status		
H	0	0	0	CEC	0	0	0	CTS	0	0	CD	RLI	DPLA	WFA	XRNR	RRNR
A	0	0	TEC	CEC	FCS	0	0	CTS	0	RFNE	CD	0	DPLA	0	0	0
B	0	0	0	CEC	0	0	0	CTS	SYNC	RFNE	CD	0	DPLA	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	unused															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TEC **TIC Executing** (async mode)

TIC='0' No TIC (transmit immediate character) is currently in transmission. Access to register TICR is allowed to initiate a TIC transmission.

TIC='1' A TIC command (write access to register TICR) is accepted but not completely executed. No further write access to register TICR is allowed until 'TIC' bit is cleared by the DSCC4.

CEC **Command Executing** (all modes)

CEC='0' No command is currently in execution. The command register CMDR can be written by CPU.

CEC='1' A command (written previously to register CMDR) is currently in execution. No further command can be written to register CMDR by CPU.

Note: CEC will stay active if the SCC is in power-down mode or if no serial clock, needed for command execution, is available.

FCS **Flow Control Status** (async mode)

If (in-band) flow control mechanism is enabled via bit 'FLON' in register CCR2 this bit indicates the current state of transmitter:

FCS='0' Transmitter is ready (always after transmitter reset command or XON-character detected).

FCS='1' Transmitter is stopped (XOFF-character detected).

CTS **Clear To Send Input Signal State** (all modes)

CTS='0' $\overline{\text{CTS}}$ input signal is inactive (high level)

CTS='1' $\overline{\text{CTS}}$ input signal is active (low level)

Note: A transmit clock must be provided in order to detect the signal state of the $\overline{\text{CTS}}$ pin.
Optionally this input can be programmed to generate an interrupt on signal level changes.

SYNC **Synchronization Status** (bisync mode)

Indicates whether the receiver is in synchronized state. After a ’HUNT’ command ’SYNC’ bit is cleared and the receiver starts searching for a SYNC character. When found the ’SYNC’ status bit is set immediately, an SCD-interrupt is generated (if enabled) and receive data is forwarded to the receive FIFO.

SYNC=’0’ Synchronization is lost or not yet achieved.
(after reset, after new ’HUNT’ command has been issued before SYNC character is found)

SYNC=’1’ The receiver is in synchronized state.

RFNE **(SCC) Receive FIFO Not Empty** (async/bisync modes)

This status bit is set if the SCC receive FIFO stores at least one valid byte which might be not transferred to the central RFIFO because the programmed threshold level is not exceeded and no frame end/ block end condition is generated.

RFNE=’0’ SCC receive FIFO is empty.

RFNE=’1’ SCC receive FIFO is not empty.

CD **CD (Carrier Detect) Input Signal State** (all modes)

CD=’0’ CD input signal is inactive (low level)

CD=’1’ CD input signal is active (high level)

Note: A receive clock must be provided in order to detect the signal state of the CD pin.
Optionally this input can be programmed to generate an interrupt on signal level changes.

RLI **Receive Line Inactive** (hdlc mode)

This bit indicates that neither flags as interframe time fill nor data are received via the receive line.

RLI=’0’ Receive line is active, no constant high level is detected.

RLI=’1’ Receive line is inactive, i.e. more than 7 consecutive ’1’ are detected on the line.

DPLA **DPLL Asynchronous** (all modes)

This bit is only valid if the receive clock is recovered by the DPLL and FM0, FM1 or Manchester data encoding is selected. It is set when the DPLL has lost synchronization. In this case reception is disabled (receive abort condition) until synchronization has been regained. In addition transmission is interrupted in all cases where transmit clock is derived from the DPLL (clock mode 3a, 7a). Interruption of transmission is performed the same way as on deactivation of the $\overline{\text{CTS}}$ signal.

DPLA='0' DPLL is synchronized.

DPLA='1' DPLL is asynchronous (re-synchronization process is started automatically).

WFA **Wait For Acknowledgement** (hdlc mode)

This status bit is significant in Automode only. It indicates whether the Automode state machine expects an acknowledging I- or S-Frame for a previously sent I-Frame.

WFA='0' No acknowledge I/S-Frame is expected.

WFA='1' The Automode state machine is waiting for an achnowledging S- or I-Frame.

XRNR **Transmit RNR Status** (hdlc mode)

This status bit is significant in Automode only. It indicates the receiver status of the local station (SCC).

XRNR='0' The receiver is ready and will automatically answer poll-frames with a S-Frame with 'receiver-ready' indication.

XRNR='1' The receiver is NOT ready and will automatically answer poll-frames with a S-Frame with a 'receiver-not-ready' indication.

RRNR **Received RNR Status** (hdlc mode)

This status bit is significant in Automode only. It indicates the receiver status of the remote station.

RRNR='0' The remote station receiver is ready.

RRNR='1' The remote receiver is NOT ready.
(A 'receiver-not-ready' indication was received from the remote station)

Table 67 CCR0: Channel Configuration Register 0

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0108$_H$**	**0188$_H$**	**0208$_H$**	**0288$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Power Mode				-				Serial Port Configurationt				Protocol Engine Selection			
H	PU	0	0	0	0	0	0	0	0	SC2	SC1	SC0	0	0	SM(1:0) = ’00’	
A	PU	0	0	0	0	0	0	0	0	SC2	SC1	SC0	0	0	SM(1:0) = ’11’	
B	PU	0	0	0	0	0	0	0	0	SC2	SC1	SC0	0	0	SM(1:0) = ’10’	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Interrupts				DPLL				Clock Source				Clock Mode			
H	0	0	0	VIS	0	0	0	PSD	BCR	0	TOE	SSEL	HS	CM2	CM1	CM0
A	0	0	0	VIS	0	0	0	PSD	BCR	0	TOE	SSEL	0	CM2	CM1	CM0
B	0	0	0	VIS	0	0	0	PSD	0	0	TOE	SSEL	0	CM2	CM1	CM0

PU **Power Up** (all modes)

PU=’0’ The SCC is in ’power-down’ mode. The protocol engines are switched off (standby) and no operation is performed. This may be used to save power when SCC is not in use.

Note: The DMA Controller accessible part of the SCC transmit FIFO requests for transmit data even in ’power-down’ mode.

PU=’1’ The SCC is in ’power-up’ mode.

SC(2..0) **Serial Port Configuration** (all modes)

This bit field selects the line coding of the serial port.
Note, that special operation modes and settings may require or exclude operation in special line coding modes. Refer to the ’prerequisites’ in the dedicated mode descriptions.

SC = ’000’ NRZ data encoding

SC = ’001’ Bus configuration, timing mode 1;
NRZ data encoding

SC = ’010’ NRZI data encoding

SC = ’011’ Bus configuration, timing mode 2;
NRZ data encoding

SC = ’100’ FM0 data encoding

SC = ’101’ FM1 data encoding

SC = ’110’ Manchester data encoding

SC = ’111’ *Reserved*
(do not use)

Note: If bus configuration mode is selected, only NRZ data encoding is supported.

SM(1..0) **Serial Port Mode** (all modes)

This bit field selects one of the three protocol engines.
Depending on the selected protocol engine the SCC related registers change or special bit positions within the registers change their meaning.

SM = ’00’ HDLC/SDLC protocol engine

SM = ’01’ *Reserved*
(do not use)

SM = ’10’ BISYNC protocol engine

SM = ’11’ ASYNC protocol engine

VIS **Masked Interrupts Visible** (all modes)

VIS=’0’ Masked interrupt status bits are not visible on interrupt status register (ISR) read accesses.

VIS=’1’ Masked interrupt status bits are visible and automatically cleared after interrupt status register (ISR) read access.

Note: Masked interrupts will not generate an interrupt vector to the interrupt controller.

PSD **DPLL Phase Shift Disable** (all modes)

This option is only applicable in the case of NRZ or NRZI line encoding is selected.

PSD=’0’ Normal DPLL operation.

PSD=’1’ The phase shift function of the DPLL is disabled. The windows for phase adjustment are extended.

BCR **Bit Clock Rate** (async PPP, ASYNC modes)

This bit is only valid in asynchronous PPP and ASYNC protocol mode and only in clock modes not using the DPLL (0, 1, 3b, 7b). It is also invalid in high speed operation clock mode 4.

BCR=’0’ Selects isochronous operation with bit clock rate 1. Data bits are sampled once.

BCR=’1’ Selects standard asynchronous operation with bit clock rate 16. Using 16 samples per bit, data bits are sampled 3 times around the nominal bit center. The resulting bit value is determined by majority decision of the 3 samples. For correct operation NRZ data encoding has to be selected.

TOE **Transmit Clock Out Enable** (all modes)

For clock modes 0b, 2b, 3a, 3b, 6b, 7a and 7b, the internal transmit clock can be monitored on pin TxCLK as an output signal. In clock mode 5, a time slot control signal marking the active transmit time slot is output on pin TxCLK.
Bit ’TOE’ is invalid for all other clock modes.

TOE=’0’ TxCLK pin is input.

TOE=’1’ TxCLK pin is switched to output function if applicable for the selected clock mode.

SSEL **Clock Source Select** (all modes)

Distinguishes between the ’a’ and ’b’ option of clock modes 0, 2, 3, 6 and 7.

SSEL=’0’ Option ’a’ is selected.

SSEL=’1’ Option ’b’ is selected.

HS **High Speed (PEB 20534H-52 only)** (hdlc mode)

Configures the SCC for High Speed operation.

HS=’0’ Normal, non high speed operation.

HS=’1’ Switches the internal clocking and data paths for high speed operation up to 52 MBit/s and enables the clock gating mechanism.

Note: Bit ’HS’ should be set only in conjunction with clock mode 4 selection.

CM(2..0) **Clock Mode** (all modes)

This bit field selects one of main clock modes 0..7. For a detailed description of the clock modes refer to **Chapter 7.4.1**.

CM = ’000’ clock mode 0

CM = ’001’ clock mode 1

CM = ’010’ clock mode 2

CM = ’011’ clock mode 3

CM = ’100’ clock mode 4 (high speed operation clock mode) **(PEB 20534H-52 only)**

CM = ’101’ clock mode 5 (time slot oriented clocking mode)

CM = ’110’ clock mode 6

CM = ’111’ clock mode 7

Table 68 CCR1: Channel Configuration Register 1

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**010C$_H$**	**018C$_H$**	**020C$_H$**	**028C$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Output Control															
H	0	0	SOC1	SOC0	0	DIV	ODS	0	0	ICD	TCLKO	RTS	FRTS	FCTS	CAS	0
A	0	0	0	0	0	DIV	ODS	0	0	ICD	0	RTS	FRTS	FCTS	CAS	0
B	0	0	0	0	0	DIV	ODS	0	0	ICD	0	RTS	FRTS	FCTS	CAS	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmitter/Receiver Configuration															
H	MDS1	MDS0	ADM	NRM	PPM1	PPM0	MCS	TLP	SFLG	0	0	0	0	0	CRL	C32
A	0	0	0	0	0	0	0	TLP	TOIE	TOLEN						
B	0	0	0	0	SLEN	BISNC	0	TLP	0	0	0	0	0	0	0	0

SOC(1..0) **Serial Output Control** (hdlc mode)

This bit field selects the $\overline{\text{RTS}}$ signal output function.
(This bit field is only valid in bus configuration modes selected via bit field SC(2:0) in register CCR0).

SOC = ’0X’ $\overline{\text{RTS}}$ ouput signal is active during transmission of a frame (active low).

SOC = ’10’ $\overline{\text{RTS}}$ ouput signal is always inactive (high).

SOC = ’11’ $\overline{\text{RTS}}$ ouput signal is active during reception of a frame (active low).

DIV **Data Inversion** (all modes)

This bit is only valid if NRZ data encoding is selected via bit field SC(2:0) in register CCR0.

DIV=’0’ No Data Inversion.

DIV=’1’ Data is transmitted/received inverted (on a per bit basis). In HDLC and HDLC Synchronous PPP modes the continuous ’1’ idle sequence is NOT inverted. Interframe time fill flag transmission is inverted.

Note: It is recommended not to use DIV=’1’ in combination with high speed operation, i.e. clock mode 4 and CCR0.HS=’1’.

ODS **Output Driver Select** (all modes)

The transmit data output pin TxD can be configured as push/pull or open drain output chracteristic.

ODS=’0’ TxD pin is open drain output.

ODS=’1’ TxD pin is push/pull output.

ICD **Invert Carrier Detect Pin Polarity** (all modes)

ICD=’0’ Carrier Detect (CD) input pin is active high.

ICD=’1’ Carrier Detect (CD) input pin is active low.

TCLKO **Transmit Clock Output** (hdlc mode)

This bit is only valid in high speed operation mode, i.e. clock mode 4 and CCR0.HS = ’1’.
In high speed mode the internal transmit clock is supplied to pin $\overline{\text{RTS}}$ as an output.

TCLKO=’0’ Pin $\overline{\text{RTS}}$ is an unused input signal in high speed operation mode.
It should be connected to a known level via a pull-up/down resistor.

TCLKO=’1’ The internal transmit clock is supplied as an output signal to pin $\overline{\text{RTS}}$.

RTS **Request To Send pin control** (all modes)

The request to send pin $\overline{\text{RTS}}$ can be controlled by the DSCC4 as an output autonomously or via setting/clearing bit ’RTS’.
This bit is not valid in high speed operation mode.

RTS=’0’ $\overline{\text{RTS}}$ (output) pin is controlled by the DSCC4 autonomously.
- In HDLC modes $\overline{\text{RTS}}$ is activated during transmission.
- In ASYNC and BISYNC mode, the function depends on bit ’FRTS’ in register CCR1.
- In bus configuration mode the functionality depends on bit field ’SOC’ setting.

RTS=’1’
- In HDLC modes $\overline{\text{RTS}}$ (output) is activated (low) until this bit is cleared by software again.
- In ASYNC and BISYNC mode, the function depends on bit ’FRTS’ in register CCR1.

Note: For RTS pin control a transmit clock is necessary.

FRTS **Flow Control (using signal $\overline{\text{RTS}}$)** (all modes)

Bit 'FRTS' together with bit 'RTS' determine the function of signal $\overline{\text{RTS}}$:

FRTS, RTS		
0,	0	Pin $\overline{\text{RTS}}$ is controlled by the DSCC4 autonomously. $\overline{\text{RTS}}$ is activated (low) as soon as transmit data is available within the SCC transmit FIFO.
1,	0	Pin $\overline{\text{RTS}}$ is controlled by the DSCC4 autonomously supporting bi-directional data flow control. $\overline{\text{RTS}}$ is activated (low) if the shadow part of the SCC receive FIFO is empty and de-activated (high) when the SCC receive FIFO fill level reaches its receive FIFO threshold.
0,	1	Forces pin $\overline{\text{RTS}}$ to active state (low).
1,	1	Forces pin $\overline{\text{RTS}}$ to inactive state (high).

Note: For RTS pin control a transmit clock is necessary.

FCTS **Flow Control (using signal $\overline{\text{CTS}}$)** (all modes)

This bit controls the function of pin $\overline{\text{CTS}}$.
This bit is not valid in high speed operation mode.

FCTS = '0' The transmitter is stopped if $\overline{\text{CTS}}$ input signal is inactive (high) and enabled if active (low).

Note: In character oriented protocol modes (ASYNC, BISYNC, asynchronous PPP), the current byte is completely sent even if $\overline{\text{CTS}}$ becomes inactive during transmission.

FCTS = '1' The transmitter is enabled disregarding $\overline{\text{CTS}}$ input signal.

Note: In ASYNC mode the transmitter is additionally controlled by in-band flow control mechanism (if enabled).

CAS **Carrier Detect Auto Start** (all modes)

CAS = ’0’ The CD pin is used as general input.
In clock mode 1, 4 and 5, clock mode specific control signals must be provided at this pin (receive strobe, receive gating $\overline{\text{RCG}}$, frame sync pulse FSC).
A pull-up/down resistor is recommended if unused.

CAS = ’1’ The CD pin enables/disables the receiver for data reception. (Polarity of CD pin can be configured via bit ’ICD’.)

Note: (1) In clock modes 1, 4 and 5 this bit must be set to ’0’
(2) In ASYNC mode the transmitter is additionally controlled by in-band flow control mechanism (if enabled).
(3) A receive clock must be provided in order to detect the signal state of the CD input pin.

MDS(1..0) **Mode Select** (hdlc modes)

This bit field selects the HDLC protocol sub-mode including the ’extended transparent mode’.

MDS = ’00’ Automode.

MDS = ’01’ Non-Automode.

MDS = ’10’ Address Mode 0/1.
(Option ’0’ or ’1’ is selected via bit ’ADM’.)

MDS = ’11’ Extended transparent mode (bit transparent transmission/ reception).

Note: MDS(1:0) must be set to ’10’ if PPP operation is selected.

ADM **Address Mode Select** (hdlc mode)

The meaning of this bit depends on the selected protocol sub-mode:

Automode, Non-Automode:
Determines the address field length of a HDLC frame.

ADM = ’0’ 8-bit address field.

ADM = ’1’ 16-bit address field.

Address mode 0/1:
Determines whether address mode 0 or 1 is selected.

ADM = ’0’ Address Mode 0 (no address recognition).

ADM = ’1’ Address Mode 1 (high byte address recognition).

NRM **Normal Response Mode** (hdlc mode)

This bit is valid in HDLC Automode operation only and determines the function of the Automode LAP-Controller:

NRM = ’0’ Full-duplex LAP-B / LAP-D operation.

NRM = ’1’ Half-duplex normal response mode (NRM) operation.

PPPM(1..0) **PPP Mode Select** (hdlc mode)

This bit field enables and selects the HDLC PPP protocol modes:

PPPM = ’00’ No PPP protocol operation. The HDLC sub-mode is determined by bit field ’MDS’.

PPPM = ’01’ Octet synchronous PPP protocol operation.

PPPM = ’10’ Asynchronous PPP protocol operation.

Note: Bit ’BCR’ in register CCR0 must be set to ensure proper asynchronous reception.

PPPM = ’11’ Bit synchronous PPP protocol operation.

SLEN **SYNC Character Length** (bisync mode)

This bit selects the SYNC character length in BISYNC/MONOSYNC operation mode:

SLEN = ’0’ 6 bit (MONOSYNC), 12 bit (BISYNC).

SLEN = ’1’ 8 bit (MONOSYNC), 16 bit (BISYNC).

BISNC **Enable BISYNC Mode** (bisync mode)

This bit is selects BISYNC/MONOSYNC operation:

BISNC = ’0’ MONOSYNC operation.

BISNC = ’1’ BISYNC operation.

MCS **Modulo Count Select** (hdlc mode)

This bit is valid in HDLC Automode operation only and determines the control field format:

MCS = ’0’ Basic operation, one byte control field (modulo 8 counter operation).

MCS = ’1’ Extended operation, two bytes control field (modulo 128 counter operation).

TLP **Test Loop** (all modes)

This bit controls the internal test loop between transmit and receive data signals. The test loop is closed at the far end of serial transmit and receive line just before the respective TxD and RxD pins:

TLP = ’0’ Test loop disabled.

TLP = ’1’ Test loop enabled.
The software is responsible to select a clock mode which allows correct reception of transmit data depending on the external clock supply. Transmit data is also sent out via pin TxD, but receive input pin RxD is internally disconnected during test loop operation.

Note: It is recommended not to use the test loop in high speed operation mode (clock mode 4). A non high speed clock mode should be selected for test loop operation.

SFLAG **Shared Flags Transmission** (hdlc mode)

This bit enables ’shared flag transmission’ in HDLC protocol mode. If another transmit frame begin is stored in the SCC transmit FIFO, the closing flag of the preceding frame becomes the opening flag of the next frame (shared flags):

SFLAG = ’0’ Shared flag transmission disabled.

SFLAG = ’1’ Shared flag transmission enabled.

Note: The receiver always supports shared flags and shared zeros of consecutive flags.

TOIE **Time Out Indication Enable** (async mode)

If this bit is selected in ASYNC mode, any time out event will automatically generate a ’RFRD’ command thus inserting a ’frame end/ block end’ indication into the receive FIFO. This causes the SCC receive FIFO to forward received data to the DMA Controller even if the receive FIFO threshold is not exceeded. The DMA controller is forced to finish the current receive descriptor with an ’frame end / block - end’ indication:

TOIE = ’0’ Automatic Time Out processing disabled.

TOIE = ’1’ Automatic Time Out processing enabled.

TOLEN (6..0) **Time Out Length** (async mode)

This bit field determines the time out period. If there is no receive line activity for the configured period of time, a time out indication is generated if enabled via bit 'TOIE'.
The period of time is programmable in multiples of character frame length (CFL) time equivalents including start, parity and stop bits:

TOLEN T = ((TOLEN + 1) * 4) * CFL

CRL **CRC Reset Value** (hdlc mode)

This bit defines the initial value of the internal transmit/receive CRC generators:

CRL='0' Initial value is $0xFFFF_H$ (16 bit CRC), $0xFFFFFFFF_H$ (32 bit CRC).
Default value for most HDLC/SDLC applications.

CRL='1' Initial value is $0x0000_H$ (16 bit CRC), $0x00000000_H$ (32 bit CRC).

C32 **CRC 32 Select** (hdlc mode)

This bit enables 32-bit CRC operation for transmit and receive.

C32='0' 16-bit CRC-CCITT generation/checking.

C32='1' 32-bit CRC generation/checking.

Note: The internal 'valid frame' criteria is updated depending on the selected number of CRC-bytes.

Table 69 CCR2: Channel Configuration Register 2

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0110$_H$**	**0190$_H$**	**0210$_H$**	**0290$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Receiver Configuration															
H	0	0	0	0	RAC	0	0	0	0	DRCRC	RCRC	RADD	RFDF	RFTH(2:0)		
A	0	0	CHL1	CHL0	RAC	DXS	XBRK	STOP	PAR1	PAR0	PARE	DPS	RFDF	RFTH(2:0)		
B	0	0	CHL1	CHL0	RAC	0	0	SLOAD	PAR1	PAR0	PARE	DPS	RFDF	RFTH(2:0)		

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmitter Configuration															
H	PRE(7:0)								EPT	0	NPRE(1:0)		ITF	SXIF	OIN	XCRC
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FLON
B	PRE(7:0)								EPT	0	NPRE(1:0)		ITF	CRL	CAPP	CRCM

CHL(1..0) **Character Length** (async/bisync modes)

This bit field selects the number of data bits within a character:

CHL = ’00’ 8-bit data.

CHL = ’01’ 7-bit data.

CHL = ’10’ 6-bit data.

CHL = ’11’ 5-bit data.

RAC **Receiver active** (all modes)

Switches the receiver between operational/inoperational states:

RAC=’0’ Receiver inactive, receive line is ignored.

RAC=’1’ Receiver active.

DXS **Disable Storage of XON/XOFF Characters** (async mode)

In ASYNC mode, XON/XOFF characters might be filtered out or stored to the SCC receive FIFO:

DXS=’0’ All received characters including XON/XOFF characters are stored in the receive FIFO.

DXS=’1’ XON/XOFF characters are filtered out and not stored in the receive FIFO.

XBRK **Transmit Break** (async mode)

XBRK=’0’ Normal transmit operation.

XBRK=’1’ Forces the TxD pin to ’low’ level immediately (break condition), regardless of any character being currently transmitted. This command is executed immediately with the next rising edge of the transmit clock and further transmission is disabled. The currently sent character is lost.
Data stored in the SCC transmit FIFO will be sent as soon as the break condition is cleared (XBRK=’0’). A transmit reset command (bit ’XRES’ in register CMDR) does NOT clear the break condition automatically.

STOP **Stop Bit number** (async mode)

This bit selects the number of stop bits per ASYNC character:

STOP=’0’ 1 stop bit per character.

STOP=’1’ 2 stop bits per character.

SLOAD **Enable SYNC Character Load** (bisync mode)

In BISYNC mode, SYNC characters might be filtered out or stored to the SCC receive FIFO:

SLOAD=’0’ SYNC characters are filtered out and not stored in the receive FIFO.

SLOAD=’1’ All received characters including SYNC characters are stored in the receive FIFO.

PAR(1..0) **Parity Format** (async/bisync modes)

This bit field selects the parity generation/checking mode:

PAR = ’00’ SPACE (’0’), a constant ’0’ is inserted as parity bit.

PAR = ’01’ Odd parity.

PAR = ’10’ Even parity.

PAR = ’11’ MARK (’1’), a constant ’1’ is inserted as parity bit.

The received parity bit is stored in the SCC receive FIFO depending on the selected character format:

- as leading bit immediately preceding the data bits if character length is 5, 6 or 7 bits and bit ’DPS’ in register CCR2 is cleared (’0’).
- as LSB of the status byte belonging to the character if character length is 8 bits and the corresponding receive FIFO data format is selected (RFDF = ’1’).

A parity error is indicated in the MSB of the status byte belonging to each character if enabled. In addition, a parity error interrupt can be generated.

PARE **Parity Enable** (async/bisync modes)

PARE=’0’ Parity generation/checking is disabled.

PARE=’1’ Parity generation/checking is enabled.

DPS **Data Parity Storage** (async/bisync modes)

Only valid if parity generation/checking is enabled via bit ’PARE’:

DPS=’0’ The parity bit is stored.

DPS=’1’ The parity bit is not stored in the data byte containing character data.
The parity bit is always stored in the status byte.

DRCRC **Disable Receive CRC Checking** (hdlc mode)

DRCRC=’0’ The receiver expects a 16 or 32 bit CRC within a HDLC frame. CRC processing depends on the setting of bit ’RCRC’.
Frames shorter than expected are marked ’invalid’ or are discarded (refer to RSTA description).

DRCRC=’1’ The receiver does not expect any CRC within a HDLC frame. The criteria for ’valid frame’ indication is updated accordingly (refer to RSTA description).
Bit ’RCRC’ is ignored.

RCRC **Receive CRC Checking Mode** (hdlc mode)

This bit is only valid in Non-Automode and Address Mode 0:

RCRC=’0’ The received checksum is evaluated, but not forwarded to the receive FIFO.

RCRC=’1’ The received checksum (2 or 4 bytes) is evaluated and forwarded to the receive FIFO as data. In Non-Automode the criteria for ’valid frame’ is updated (refer to RSTA description).

RADD **Receive Address Pushed to RFIFO** (hdlc mode)

This bit is only valid if a HDLC sub-mode with address field support is selected (Automode, Non-Automode, Address Mode 1):

RADD=’0’ The received HDLC address field (either 8 or 16 bit depending on bit ’ADM’) is evaluated, but NOT forwarded to the receive FIFO.

RADD=’1’ The received HDLC address field (either 8 or 16 bit depending on bit ’ADM’) is evaluated and forwarded to the receive FIFO.

RFDF **Receive FIFO Data Format** (all modes)

HDLC modes:

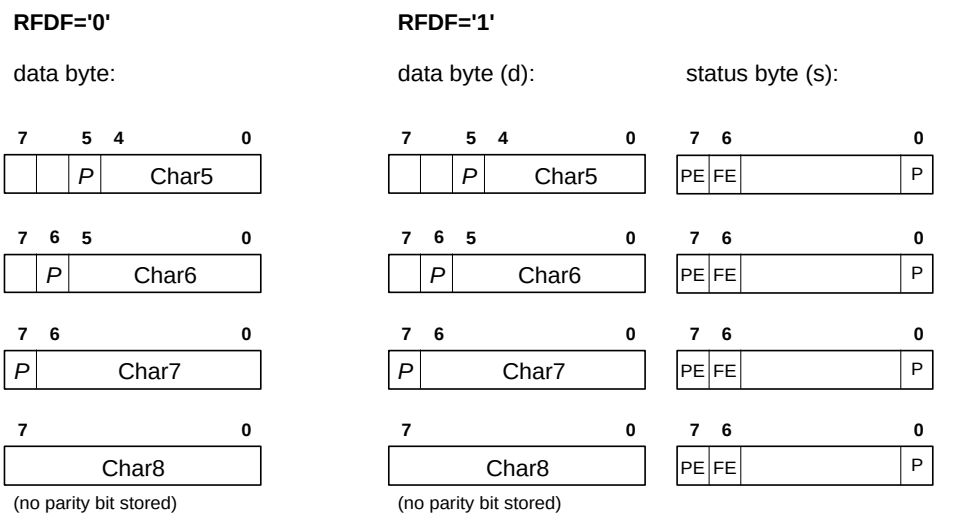
This bit is only valid if the minimum receive FIFO threshold is selected (bit field RFTH(2..0) = ’000’):

RFDF=’0’ A minimum of one byte is stored in the receive FIFO before forwarded to the central RFIFO.

RFDF=’1’ A minimum of two bytes is stored in the receive FIFO before forwarded to the central RFIFO.

ASYNC/BISYNC modes:

In ASYNC/BISYNC modes, the character format is determined as follows:



P: Parity bit stored in data byte, can be disabled via bit 'DPS'
PE: Parity Error
FE: Frame Error
P: Parity bit stored in second data byte (= status byte)

RFTH(2..0) **Receive FIFO Threshold** (all modes)

This bit field defines the level up to which the SCC receive FIFO is filled with valid data before transfer to the central RFIFO is requested. (Transfer always starts immediately in case of a ’frame end / block end’ condition.)
The meaning depends on the selected protocol engine:

HDLC Modes:

RFTH(2..0)	Threshold level in number of data (d) and status (s) bytes depending on bit ’RFDF’:	
	RFDF = ’0’	RFDF = ’1’
’000’	1d	2d
’001’	4d	don’t care
’010’	16d	don’t care
’011’	24d	don’t care
’100’	32d	don’t care
’101’	60d	don’t care
’110’	1d	2d
’111’	1d	2d

ASYNC/BISYNC Modes:

RFTH(2..0)	Threshold level in number of data (d) and status (s) bytes depending on bit ’RFDF’:	
	RFDF = ’0’	RFDF = ’1’
’000’	1d	1d+1s
’001’	4d	2d+2s
’010’	16d	8d+8s
’011’	24d	12d+12s
’100’	32d	16d+16s
’101’	60d	30d+30s
’110’	1d	1d+1s
’111’	1d	1d+1s

PRE(7..0) **Preamble** (hdlc/bisync modes)

This bit field determines the preamble pattern which is send out during preamble transmission.

Note: In HDLC-mode, zero-bit insertion is disabled during preamble transmission.

EPT **Enable Preamble Transmission** (hdlc/bisync modes)

This bit enables preamble transmission. The preamble is started after interframe timefill (ITF) transmission is stopped because a new frame is ready to be transmitted. The preamble pattern consists of 8 bits defined in bit field 'PRE(7..0)' which is send repetitively. The number of repetitions is determined by bit field 'PRE(1..0)':

EPT='0' Preamble transmission is disabled.

EPT='1' Preamble transmission is enabled.

Note: Preamble operation does NOT influence HDLC shared flag transmission if enabled.

NPRE(1..0) **Number of Preamble Repetitions** (hdlc/bisync modes)

This bit field determines the number of preambles transmitted:

PRE = '00' 1 preamble.

PRE = '01' 2 preambles.

PRE = '10' 4 preambles.

PRE = '11' 8 preambles.

ITF **Interframe Time Fill** (hdlc/bisync modes)

This bit selects the idle state of the transmit pin TxD:

ITF='0' Continuous logical '1' is send during idle phase.

ITF='1' Continuous flag sequences are sent ('01111110' flag pattern).

Note: It is recommended to clear bit 'ITF' in bus configuration modes, i.e. continuous ones are send as idle sequence and data encoding is NRZ.

SXIF **Selects Transmission of I-Frames** (hdlc mode)

This bit is valid in HDLC-Automode only:

SXIF=’0’ The SCC in Automode transmits transparent HDLC frames (U-frames). No acknowledgement is awaited.

SXIF=’1’ Causes the SCC in Automode to transmit a HDLC frame as an I-frame. Additionally to the opening flag sequence, the address and control field of the frame is automatically added by the SCC. An all-sent (ALLS) interrupt is generated after receiving the corresponding acknowledgement

CRL **CRC Reset Value** (bisync mode)

This bit defines the initial value of the internal transmit/receive CRC generators:

CRL=’0’ Initial value is $0xFFFF_H$ (16 bit CRC), $0xFFFFFFFF_H$ (32 bit CRC).

CRL=’1’ Initial value is $0x0000_H$ (16 bit CRC), $0x00000000_H$ (32 bit CRC).

OIN **One Insertion** (hdlc mode)

In HDLC mode a one-insertion mechanism similar to the zero-insertion can be activated:

OIN=’0’ The ’1’ insertion mechanism is disabled.

OIN=’1’ In transmit direction a logical ’1’ is inserted to the serial data stream after 7 consecutive zeros.
In receive direction a ’1’ is deleted from the receive data stream after receiving 7 consecutive zeros.
This enables clock information to be recovered from the receive data stream by means of a DPLL, even in the case of NRZ data encoding, because a transition at bit cell boundary occurs at least every 7 bits.

CAPP **CRC Append** (bisync mode)

In BISYNC mode the CRC generator can be activated:

CAPP=’0’ No CRC generation/checking is active in BISYNC mode.

CAPP=’1’ The CRC generator is activated:

1. The CRC generator is initialized every time the transmission of a new ’frame’ starts. The CRC initialization value can be selected via bit ’CRL’ in register CCR2 (for BISYNC operation).
2. The CRC is automatically appended to the last transmitted data of a ’frame’.

CRCM **CRC Mode Select** (bisync mode)

In BISYNC mode the CRC generator can be configured for two different generator polynoms:

CRCM=’0’ CRC-16:
The polynominal is $x^{16}+x^{15}+x^{2}+1$.

CRCM=’1’ CRC-CCITT:
The polynominal is $x^{16}+x^{12}+x^{5}+1$.

XCRC **Transmit CRC Checking Mode** (hdlc mode)

XCRC=’0’ The transmit checksum (2 or 4 bytes) is generated and appended to the transmit data automatically.

XCRC=’1’ The transmit checksum is not generated automatically. The checksum is expected to be provided by software as the last 2 or 4 bytes in the transmit data buffer.

Note: The transmitter does NOT check whether the number of data bytes makes sense, i.e. a valid frame length.

FLON **Flow Control Enable** (async mode)

In ASYNC mode, in-band flow control is supported:

FLON=’0’ No automatic in-band flow-control is performed. However recognition of a flow control character (XON/XOFF) causes always a maskable interrupt event.

FLON=’1’ Automatic in-band flow-control is performed.
Reception of a XOFF character (defined in register XNXF) turns off the transmitter after the currently transmitted character has been shifted out completely (XOFF state).
Reception of a XON character (defined in register XNXF) resumes the transmitter from XOFF into XON state ready to send available transmit data bytes.
The current flow control state is indicated via bit ’FCS’ in register Star.
Any transmitter reset switches the flow-control logic to XON state.

Table 70 ACCM: PPP ASYNC Control Character Map

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0114$_H$**	**0194$_H$**	**0214$_H$**	**0294$_H$**

typical usage: written by CPU, valid in HDLC PPP protocol mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	ASYNC Character Control Map (high)															
H	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	ASYNC Character Control Map (low)															
H	0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ACCM **ASYNC Character Control Map** (hdlc mode)

This bit field is valid in HDLC asynchronous and octet-synchronous PPP mode only:

Each bit selects the corresponding character (indicated as hex value $1F_H..00_H$ in the register description table) as control character which has to be mapped into the transmit data stream.

Table 71 UDAC: User Defined PPP ASYNC Control Character Map

CPU Accessibility: **read/write**

Reset Value: **7E7E 7E7E$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0118$_H$**	**0198$_H$**	**0218$_H$**	**0298$_H$**

typical usage: written by CPU, valid in HDLC PPP protocol mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	ASYNC Character 3								ASYNC Character 2							
H	AC3								AC2							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	ASYNC Character 1								ASYNC Character 0							
H	AC1								AC0							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AC3..0 **User Defined ASYNC Character Control Map** (hdlc mode)

This bit field is valid in HDLC asynchronous and octet-synchronous PPP mode only:

These bit fields define user determined characters as control characters which have to be mapped into the transmit data stream.
In register ACCM only characters $00_H..1F_H$ can be selected as control characters. Register UDAC allows to specify any four characters in the range $00_H..FF_H$.
The default value is a $7E_H$ flag which must be always mapped. Thus no additional character is mapped if $7E_H$'s are programed to bit fields AC3...0 (reset value).
($7E_H$ is mapped automatically, even if not defined via a AC bit field.)

Table 72 TTSA: Transmit Time Slot Assignment Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**011C$_H$**	**019C$_H$**	**021C$_H$**	**029C$_H$**

typical usage: written by CPU, valid in HDLC clock mode 5 only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Tx Time Slot Number												Tx Clock Shift			
H	0	TTSN(6:0)							0	0	0	0	0	TCS(2:0)		
A	0	TTSN(6:0)							0	0	0	0	0	TCS(2:0)		
B	0	TTSN(6:0)							0	0	0	0	0	TCS(2:0)		

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmit Time Slot Control								Transmit Channel Capacity							
H	TEPCM	0	0	0	0	0	0		TCC(8:0)							
A	TEPCM	0	0	0	0	0	0		TCC(8:0)							
B	TEPCM	0	0	0	0	0	0		TCC(8:0)							

The following bit fields allow flexible assignment of time-slots to the serial channel. For more detailed information refer to **Chapter 7.4.1.6**, **“Clock Mode 5” on Page 147**.

TTSN(6:0) **Transmit Time Slot Number** (all modes)

This bit field selects one of 128 possible time-slots in which data is allowed to be transmitted. The number of bits per time-slot can be programmed via bit field ’TCC’.

TCS(2:0) **Transmit Clock Shift** (all modes)

This bit field determines the transmit clock shift.

TEPCM **Enable PCM Mask Transmit** (all modes)

This bit selects the additional Transmit PCM Mask (refer to register PCMMTX):

TEPCM=’0’ Standard time-slot configuration.

TEPCM=’1’ The time-slot width is constant 8 bit, bit fields ’TTSN’ and ’TCS’ determine the offset of the PCM mask and ’TCC’ is ignored. Each time-slot selected via register PCMMTX is an active transmit timeslot.

TCC(8:0) **Transmit Channel Capacity** (all modes)

This bit field determines the transmit time-slot width in standard time-slot configuration (bit TEPCM=’0’):
Number of bits = TCC + 1, (1...512 bits/time-slot)

Table 73 RTSA: Receive Time Slot Assignment Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0120$_H$**	**01A0$_H$**	**0220$_H$**	**02A0$_H$**

typical usage: written by CPU, valid in HDLC clock mode 5 only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Rx Time Slot Number												Rx Clock Shift			
H	0	RTSN(6:0)							0	0	0	0	0	RCS(2:0)		
A	0	RTSN(6:0)							0	0	0	0	0	RCS(2:0)		
B	0	RTSN(6:0)							0	0	0	0	0	RCS(2:0)		

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Receive Time Slot Control								Receive Channel Capacity							
H	REPCM	0	0	0	0	0	0		RCC(8:0)							
A	REPCM	0	0	0	0	0	0		RCC(8:0)							
B	REPCM	0	0	0	0	0	0		RCC(8:0)							

The following bit fields allow flexible assignment of time-slots to the serial channel. For more detailed information refer to **Chapter 7.4.1.6**, **“Clock Mode 5” on Page 147**.

RTSN(6:0) **Receive Time Slot Number** (all modes)

This bit field selects one of 128 possible time-slots in which data is received. The number of bits per time-slot can be programmed via bit field ’RCC’.

RCS(2:0) **Receive Clock Shift** (all modes)

This bit field determines the receive clock shift.

REPCM **Enable PCM Mask Receive** (all modes)

This bit selects the additional Receive PCM Mask (refer to register PCMMRX):

REPCM=’0’ Standard time-slot configuration.

REPCM=’1’ The time-slot width is constant 8 bit, bit fields ’RTSN’ and ’RCS’ determine the offset of the PCM mask and ’RCC’ is ignored. Each time-slot selected via register PCMMRX is an active receive timeslot.

RCC(8:0) **Receive Channel Capacity** (all modes)

This bit field determines the receive time-slot width in standard time-slot configuration (bit REPCM=’0’):
Number of bits = RCC + 1, (1...512 bits/time-slot)

Table 74 PCMMTX: PCM Mask for Transmit Direction

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0124$_H$**	**01A4$_H$**	**0224$_H$**	**02A4$_H$**

typical usage: written by CPU, valid in HDLC clock mode 5 only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	PCM Mask for Transmit Direction (high)															
H	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
A	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
B	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	PCM Mask for Transmit Direction (low)															
H	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00
A	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00
B	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00

PCMMTX **PCM Mask for Transmit Direction** (hdlc mode)

This bit field is valid in HDLC clock mode 5 only and the PCM mask must be enabled via bit ’TEPCM’ in register TTSA.

Each bit selects one of 32 (8-bit) transmit time-slots. The offset of time-slot zero to the frame sync pulse can be programmed via register TTSA bit field ’TTSN’.

Table 75 PCMMRX: PCM Mask for Receive Direction

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0128$_H$**	**01A8$_H$**	**0228$_H$**	**02A8$_H$**

typical usage: written by CPU, valid in HDLC clock mode 5 only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	PCM Mask for Receive Direction (high)															
H	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
A	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16
B	T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	PCM Mask for Receive Direction (low)															
H	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00
A	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00
B	T15	T14	T13	T12	T11	T10	T09	T08	T07	T06	T05	T04	T03	T02	T01	T00

PCMMRX **PCM Mask for Receive Direction** (hdlc mode)

This bit field is valid in HDLC clock mode 5 only and the PCM mask must be enabled via bit ’REPCM’ in register RTSA.

Each bit selects one of 32 (8-bit) receive time-slots. The offset of time-slot zero to the frame sync pulse can be programmed via register RTSA bit field ’RTSN’.

Table 76 BRR: Baud Rate Register

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	SCC0	SCC1	SCC2	SCC3
Offset Address:	**012C$_H$**	**01AC$_H$**	**022C$_H$**	**02AC$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Baud Rate Generator Factors															
H	0	0	0	0	BRM(3:0)				0	0	BRN(5:0)					
A	0	0	0	0	BRM(3:0)				0	0	BRN(5:0)					
B	0	0	0	0	BRM(3:0)				0	0	BRN(5:0)					

BRM(3:0) **Baud Rate Factor 'M'** (all modes)

BRN(5:0) **Baud Rate Factor 'N'** (all modes)

This bit fields determine the division factor of the internal baud rate generator. The baud rate generator input clock and the usage of baud rate generator output depends on the selected clock mode.
The division factor k is calculated by:

$$k = (N + 1) \times 2^{M}$$

with M=0..15 and N=0..63.

$$f_{BRG} = f_{in} / k$$

Table 77 TIMR: Timer Register

CPU Accessibility:	**read/write**			
Reset Value:	**0000 0000$_H$**			
	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0130$_H$**	**01B0$_H$**	**0230$_H$**	**02B0$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Timer Configuration								Timer Value							
H	SRC	0	0	TMD	0	CNT(3:0)			TVALUE(23:16)							
A	SRC	0	0	0	0	CNT(3:0)			TVALUE(23:16)							
B	SRC	0	0	0	0	CNT(3:0)			TVALUE(23:16)							

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Timer Value															
H	TVALUE(15:0)															
A	TVALUE(15:0)															
B	TVALUE(15:0)															

SRC **Clock Source** (all modes)

This bit selects the clock source of the internal timer:

SRC = ’0’ The timer is clocked by the effective transmit clock.

SRC = ’1’ The timer is clocked by the frame-sync synchronization signal supplied via the CD pin in clock mode 5. (Valid in clock mode 5 only.)

TMD **Timer Mode** (HDLC mode)

This bit selects between internal and external timer operation mode:

TMD=’0’ External timer mode:
The timer is controlled by the CPU via access to registers CMDR and TIMR.
The timer can be started any time by setting bit ’STI’ in register CMDR. The timer stops automatically after it has expired and generates a timer interrupt. The timer can be stopped any time by writing zero to the value bit field.

TMD=’1’ Internal timer mode: (valid in HDLC Automode only)
The timer is used by the DSCC4 for protocol specific time-out and retry transactions.

CNT(2..0) **Counter** (all modes)

The meaning of this bit field depends on the timer mode.
In 'internal timer mode' (HDLC Automode and bit TMD='1'):

- Retry Counter (in HDLC protocol known as 'N2'):
  Bit field 'CNT' indicates the number of S-Command frames (with poll bit set) which are transmitted autonomously by the DSCC4 after every expiration of the time out period 't' (determined by 'VALUE'), in case an I-Frame gets not acknowledged by the opposite station. The maximum value is 6 S-command frames. If 'CNT' is set to '7', the number of S-commands is unlimited in case of no acknowledgement.

In external timer mode (bit TMD='0'):

- Restart Counter :
  Bit field 'CNT' indicates the number of automatic restarts which are performed by the DSCC4 after every expiration of the time out period 't', in case the timer is not stopped by writing a zero to bit field 'TVALUE'.The maximum value is 6 restarts. If 'CNT' is set to '7', a timer interrupt is generated periodically with time period 't' determined by bit field 'TVALUE'.

TVALUE (23:0) **Timer Expiration Value** (all modes)

This bit field determines the timer expiration period 't':

$$t = (TVALUE + 1) \cdot CP$$

('CP' is the clock period depending on bit 'SRC'.)

Table 78 XADR: Transmit Address Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0134$_H$**	**01B4$_H$**	**0234$_H$**	**02B4$_H$**

typical usage: written by CPU, valid in HDLC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmit Address (low)								Transmit Address (high)							
H	XAD2 (low byte)								XAD1 (high byte)						0	XAD 1_0
	or XAD2 (RESPONSE)								**or** XAD1 (COMMAND)							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XAD1 and XAD2 bit fields are valid in HDLC modes with automatic address field handling only (Automode, Address Mode 1, Non-Automode). They can be programmed with one individual address byte which is inserted automatically into the address field (8 or 16 bit) of a HDLC transmit frame. The function depends on the selected protocol mode and address field size (bit 'ADM' in register CCR1).

XAD2 **Transmit Address 2** (hdlc mode)

2-byte address field:
Bit field XAD2 constitutes the low byte of the 2-byte address field. (In ISDN LAP-D, the low byte is known as 'TEI'.)
1-byte address field:
According to the X.25 LAP-B protocol, XAD2 is the address of a 'RESPONSE' frame.

XAD1 **Transmit Address 1** (hdlc mode)

2-byte address field:
Bit field XAD1 constitutes the high byte of the 2-byte address field. Bit 1 must be set to '0'. According to the ISDN LAP-D protocol, bit 1 is interpreted as the C/R (COMMAND/RESPONSE) bit. This bit is manipulated automatically by the DSCC4 according to the setting of bit 'CRI' in register RADR:

	bit 1	*(C/R)*
Commands Transmit	1	0
Responses Transmit	0	1
	CRI=1	CRI=0

(In ISDN LAP-D, the low byte is known as 'SAPI'.)
1-byte address field:
According to the X.25 LAP-B protocol, XAD2 is the address of a 'COMMAND' frame.

Table 79 RADR: Receive Address Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0138$_H$**	**01B8$_H$**	**0238$_H$**	**02B8$_H$**

typical usage: written by CPU, valid in HDLC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Receive Address 1 (high)								Receive Address 1 (low)							
H	RAH1						CRI	RAH 1_0	RAL1							
	or RAH1								RAL1							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Receive Address 2 (high)								Receive Address 2 (low)							
H	RAH2								RAL2							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In operating modes that provide address recognition, the high/low byte of the received address is compared with the individually programmable values in register RADR. This addresses can be masked on a per bit basis by setting the corresponding bits in register RAMR to allow extended broadcast address recognition. This feature is applicable to all HDLC sub-modes with address recognition.

RAH1 **Receive Address 1 Byte High** (hdlc mode)

In HDLC Automode bit '1' is reserved for 'CRI' (Command Response Indication). In all other modes RAH1 is an 8 bit address.

CRI **Command/Response Indication**

The setting of this bit effects the meaning of the 'C/R' bit in the receive status byte (RSTA):

C/R meaning	*C/R Value*	
Command received	0	1
Response received	1	0
	CRI=1	CRI=0

Note: If 1-byte address field is selected in HDLC Automode, RAH1 must be set to $0x00_H$.

RAL1 **Receive Address 1 Byte Low** (hdlc mode)

The general function whether it must be written or read by the CPU and its meaning depends on the selected operating mode:

- **Auto- / Non-Automode (16-bit address)**
  RAL1 can be programmed with the value of the first individual low address byte.
- **Auto- / Non-Automode (8-bit address)**
  According to X.25 LAP-B protocol, the address in RAL1 is considered as the address of a 'COMMAND' frame.

RAH2 **Receive Address 2 Byte High** (hdlc mode)

RAL2 **Receive Address 2 Byte Low** (hdlc mode)

Value of the second individually programmable high/low address byte. If a 1-byte address field is selected, RAL2 is considered as the address of a 'RESPONSE' frame according to X.25 LAP-B protocol.

Table 80 RAMR: Receive Address Mask Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**013C$_H$**	**01BC$_H$**	**023C$_H$**	**02BC$_H$**

typical usage: written by CPU, valid in HDLC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Receive Mask Address 2 (high)								Receive Mask Address 2 (low)							
H	AMRAH2								AMRAL2							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Receive Mask Address 1 (high)								Receive Mask Address 1 (low)							
H	AMRAH1								AMRAL1							
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AMRAH2 **Receive Mask Address 2 Byte High** (hdlc mode)

AMRAL2 **Receive Mask Address 2 Byte Low** (hdlc mode)

AMRAH1 **Receive Mask Address 1 Byte High** (hdlc mode)

AMRAL1 **Receive Mask Address 1 Byte Low** (hdlc mode)

Setting a bit in this bit field to ’1’ masks the corresponding bit in bit field {’RAH2’/’RAL2’/’RAH1’/’RAL1’} of register RADR. A masked bit position always matches when comparing the received frame address with bit field {’RAH2’/’RAL2’/’RAH1’/’RAL1’} allowing extended broadcast mechanism.

bit = ’0’ The dedicated bit position is NOT masked. This bit position in the received address must match with the corresponding bit position in bit field {’RAH2’/’RAL2’/ ’RAH1’/’RAL1’} to accept the frame.

bit = ’1’ The dedicated bit position is masked. This bit position in the received address NEED NOT match with the corresponding bit position in bit field {’RAH2’/’RAL2’/ ’RAH1’/’RAL1’} to accept the frame.

Table 81 RLCR: Receive Length Check Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0140$_H$**	**01C0$_H$**	**0240$_H$**	**02C0$_H$**

typical usage: written by CPU, valid in HDLC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Receive Length Control					Receive Length Limit										
H	RCE	0	0	0	0	RL(10:0)										
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

RCE **Receive Length Check Enable** (hdlc mode)

This bit is valid in HDLC mode only and enables/disables the receive length check function:

RCE = ’0’ No receive length check on received HDLC frames is performed.

RCE = ’1’ The receive length check is enabled. All bytes of a HDLC frame which are transferred to the receive FIFO (depending on the selected protocol sub-mode and receive CRC handling) are counted and checked against the maximum length check limit which is programmed in bit field ’RL’.
A frame exceeding the maximum length is treated as if it were aborted on the receive line (receive abort ’RAB’ interrupt and status indication).
In addition a ’FLEX’ interrupt is generated if enabled.

Note: The Receive Status Byte (RSTA) is part of the frame length checking.
Thus it is guaranteed, that the number of bytes transferred to the host memory for one frame never exceeds the value programmed to bit field ’RL’.

RL(10:0) **Receive Length Check Limit** (hdlc mode)

This bit-field defines the receive length check limit if checking is enabled via bit ’RCE’:

RL(10:0) The receive length limit is calculated by:

$$Limit = (RL + 1) \cdot 32$$

Table 82 XNXF: XON/XOFF In-Band Flow Control Character Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0144$_H$**	**01C4$_H$**	**0244$_H$**	**02C4$_H$**

typical usage: written by CPU, valid in ASYNC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	XON Character)								XOF Character							
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	XON								XOFF							
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	XON Character Mask								XOFF Character Mask							
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	MXON								MXOFF							
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XON **XON Character** (async mode)

This bit field specifies the XON character for in-band flow control in ASYNC protocol mode. The number of significant bits starting with the LSB depends on the character length (5..8 bits) selected via bit field 'CHL' in register CCR2.
A received character is recognized as a valid XON-character, if

- the character was correctly framed (character length as programmed and correct parity if checking is enabled)
- each bit position of the received character which is not masked via bit field 'MXON' matches with the corresponding bit in bit field 'XON'.

Received characters recognized as 'XON' character are always stored in the receive FIFO as normal receive data, but generate an appropriate XON interrupt if enabled and switch the transmitter into 'XON' state if in-band flow control is enabled via bit 'FLON' in register CCR2.

XOFF **XOFF Character** (async mode)

This bit field specifies the XOFF character for in-band flow control in ASYNC protocol mode. The number of significant bits starting with the LSB depends on the character length (5..8 bits) selected via bit field 'CHL' in register CCR2.
A received character is recognized as a valid XOFF-character, if

- the character was correctly framed (character length as programmed and correct parity if checking is enabled)
- each bit position of the received character which is not masked via bit field 'MXOFF' matches with the corresponding bit in bit field 'XOFF'.

Received characters recognized as 'XOFF' character are always stored in the receive FIFO as normal receive data, but generate an appropriate XOFF interrupt if enabled and switch the transmitter into 'XOFF' state if in-band flow control is enabled via bit 'FLON' in register CCR2.

MXON **XON Character Mask** (async mode)

Setting a bit in this bit field to ’1’ masks the corresponding bit in bit field ’XON’ of register XNXF. A masked bit position always matches when comparing the received character with bit field ’XON’.

bit = ’0’ The dedicated bit position is NOT masked. This bit position in the received character must match with the corresponding bit position in bit field ’XON’ to recognize the received character as a ’XON’ character.

bit = ’1’ The dedicated bit position is masked. This bit position in the received address NEED NOT match with the corresponding bit position in bit field ’XON’ to recognize the received character as a ’XON’ character.

MXOFF **XOFF Character Mask** (async mode)

Setting a bit in this bit field to ’1’ masks the corresponding bit in bit field ’XOFF’ of register XNXF. A masked bit position always matches when comparing the received character with bit field ’XOFF’.

bit = ’0’ The dedicated bit position is NOT masked. This bit position in the received character must match with the corresponding bit position in bit field ’XOFF’ to recognize the received character as a ’XOFF’ character.

bit = ’1’ The dedicated bit position is masked. This bit position in the received address NEED NOT match with the corresponding bit position in bit field ’XOFF’ to recognize the received character as a ’XOFF’ character.

Table 83 TCR: Termination Character Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0148$_H$**	**01C8$_H$**	**0248$_H$**	**02C8$_H$**

typical usage: written by CPU, valid in ASYNC/BISYNC modes only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Termination Character Control								Termination Character							
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	TCDE	0	0	0	0	0	0	0	TC(7:0)							
B	TCDE	0	0	0	0	0	0	0	TC(7:0)							

TCDE **Termination Character Detection Enable** (async/bisync modes)

This bit is valid in ASYNC/BISYNC modes only and enables/disables the termination character detection mechanism:

TCDE = ’0’ No receive termination character detection is performed.

TCDE = ’1’ The termination character detection is enabled. The receive data stream is monitored for the occurence of a termination character (TC) programmed via bit field ’TC’. When this character is detected, an internal ’frame end / block end’ indication is generated. This causes the DMA controller to complete the current receive descriptor and branch to the next receive descriptor address.

Note: If the programmed character length (bit field ’CHL’ in register CCR2) is less than 8 bits, the most significant unused bits of bit field ’TC’ must be set to ’0’. Otherwise no termination character will be detected.

TC(7:0) **Termination Character** (async/bisync modes)

This bit-field defines the termination character which is monitored on the receive data stream if enabled via bit ’TCDE’.

Table 84 TICR: Transmit Immediate Character Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**014C$_H$**	**01CC$_H$**	**024C$_H$**	**02CC$_H$**

typical usage: written by CPU, valid in ASYNC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	(unused)								Transmit Immediate Character							
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	TIC(7:0)							
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TIC **Transmit Immediate Character** (async mode)

On write access to this register the ASYNC protocol engine will automatically insert the character defined by bit field 'TIC' into the transmit data stream.
This happens

- immediately after write access to register TICR if the transmitter is in IDLE state (no other character is currently transmitted). The transmitter returns to IDLE state after transmission of the TIC.
- immediately after the character which is currently in transmission is completed. After transmission of the TIC, the transmitter continues with transmission of characters which are still stored in the transmit FIFO. Thus the TIC is inserted into the data stream between the characters provided via the transmit FIFO.

The TIC transmission is independent off in-band flow control. Thus the TIC is send out even if the transmitter is in XOFF-state. However the transmitter must be enabled via signal $\overline{\text{CTS}}$ (depending on bit 'FCTS' in register CCR1).

The number of significant bits (starting with the LSB) depends on the character length programmed in bit field 'CHL' in register CCR2. All character framing related settings in register CCR2 (start bit, parity generation, number of stop bits) also apply to the TIC character framing.

As long as the TIC character is not completely send, status bit TIC Execution ('TEC') in status register STAR is set to '1' by the DSCC4. No further write access to register TICR is allowed until 'TEC' status indication is cleared by the DSCC4.

Table 85 SYNCR: Synchronization Character Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0150$_H$**	**01D0$_H$**	**0250$_H$**	**02D0$_H$**

typical usage: written by CPU, valid in BISYNC mode only
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Synchronization Character(s)															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	SYNCH(7:0)								SYNCL(7:0)							

SYNCH(7:0) Synchronization Character (high) (bisync mode)

SYNCL(7:0) Synchronization Character (low) (bisync mode)

This register is only valid in BISYNC protocol mode.
The synchronization (SYNC) character format depends on the setting of bit ’BISNC’ and ’SLEN’ in register CCR1:

- MONOSYNC Mode (CCR1.BISNC = ’0’)
  The SYNC character is defined by bit field ’SYNCL’:
  a) SLEN = ’0’: the 6 bit SYNC character is specified by bits (5..0)
  b) SLEN = ’1’: the 8 bit SYNC character is specified by bits (7..0).
- BISYNC Mode (CCR1.BISNC = ’1’)
  The SYNC character is defined by bit fields ’SYNCL’ and ’SYNCH’:
  a) SLEN = ’0’: the 12 bit SYNC character is specified by bits (5..0) of each bit field, i.e. SYNC(11..0) = SYNCH(5..0), SYNCL(5..0)
  b) SLEN = ’1’: the 16 bit SYNC character is specified by bits (7..0) of each bit field, i.e. SYNC(15..0) = SYNCH(7..0), SYNCL(7..0).

In transmit direction the SYNC character is sent continuously if no data has to be transmitted and interframe timefill control is enabled by setting bit ’ITF’ to ’1’ in register CCR2.

In receive direction the receiver monitors the data stream for occurence of the specified SYNC pattern if operating in ’HUNT’ mode (bit ’HUNT’ in register CMDR).

Table 86 IMR: Interrupt Mask Register

CPU Accessibility:	**read/write**			
Reset Value:	**FFFF FFFF$_H$**			
	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0154$_H$**	**01D4$_H$**	**0254$_H$**	**02D4$_H$**
typical usage:	written by CPU			
	evaluated by DSCC4			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Output Control															
H	1	1	1	1	1	1	1	1	1	1	1	1	1	ALLS	1	XDU
A	1	1	1	1	1	1	1	1	1	1	1	1	1	ALLS	1	XOFF
B	1	1	1	1	1	1	1	1	1	1	1	1	1	ALLS	1	XDU

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmitter/Receiver Configuration															
H	TIN	CSC	XMR	XPR	1	1	1	1	RDO	RFS	RSC	PCE	PLLA	CDSC	RFO	FLEX
A	TIN	CSC	XON	XPR	1	1	BRK	BRKT	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	1
B	TIN	CSC	XMR	XPR	1	1	1	1	TCD	1	PERR	SCD	PLLA	CDSC	RFO	1

(31:0) **Interrupt Mask Bits** (all modes)

Each SCC interrupt event can generate an interrupt vector as well as an interrupt signal indication to pin $\overline{\text{INTA}}$. Each bit position of register IMR is a mask for the corresponding interrupt event in the interrupt status register ISR. Masked interrupt events neither generate an interrupt vector nor an interrupt indication to via pin $\overline{\text{INTA}}$.

bit = ’0’ The corresponding interrupt event is NOT masked and will generate an interrupt vector as well as an interrupt indication via pin $\overline{\text{INTA}}$.

bit = ’1’ The corresponding interrupt event is masked and will NEITHER generate an interrupt vector NOR an interrupt indication via pin $\overline{\text{INTA}}$.

Moreover, masked interrupt events are:

- not displayed in the interrupt status register ISR if bit ’VIS’ in register CCR0 is programmed to ’0’.
- are displayed in interrupt status register ISR if bit ’VIS’ in register CCR0 is programmed to ’1’.

Note: After RESET, all interrupt events are masked.

For detailed interrupt event description refer to the corresponding bit position in register ISR.

Table 87 ISR: Interrupt Status Register

CPU Accessibility:	**read**			
Reset Value:	**0000 0000$_H$**			
	SCC0	SCC1	SCC2	SCC3
Offset Address:	**0158$_H$**	**01D8$_H$**	**0258$_H$**	**02D8$_H$**
typical usage:	written by DSCC4			
	evaluated by CPU			

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	Output Control															
H	0	0	0	0	0	0	0	0	0	0	0	0	0	ALLS	0	XDU
A	0	0	0	0	0	0	0	0	0	0	0	0	0	ALLS	0	XOFF
B	0	0	0	0	0	0	0	0	0	0	0	0	0	ALLS	0	XDU

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	Transmitter/Receiver Configuration															
H	TIN	CSC	XMR	XPR	0	0	0	0	RDO	RFS	RSC	PCE	PLLA	CDSC	RFO	FLEX
A	TIN	CSC	XON	XPR	0	0	BRK	BRKT	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	0
B	TIN	CSC	XMR	XPR	0	0	0	0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	0

ALLS **ALL Sent Interrupt** (all modes)

HDLC Mode:

This bit is set to ’1’:

- if the last bit of the current HDLC frame is sent out via pin TxD,
- if an I-frame is sent out completely via pin TxD and either a valid acknowledge S-frame has been received or a time-out condition occured because no valid acknowledge S-frame has been received in time (Automode).

ASYNC/BISYNC Mode:

This bit is set to ’1’, if the last character is completely sent via pin TxD and no further data is stored in the SCC transmit FIFO, i.e. the transmit FIFO is empty.

XDU **Transmit Data Underrun Interrupt** (hdlc/bisync mode)

HDLC Mode:

This bit is set to ’1’, if the current frame was terminated by the SCC with an abort sequence, because neither a ’frame end / block end’ indication was detected in the FIFO (to complete the current frame) nor more data is available in the SCC transmit FIFO.

Note: The transmitter is stopped if this condition occurs and needs to be reset via command bit ’XRES’ in register CMDR. Furthermore the XDU interrupt indication MUST be cleared by generating an interrupt vector, thus bit ’XDU’ should not be masked via register IMR.

BISYNC Mode:

This bit is set to ’1’, if the current transmission was terminated with IDLE sequence because no more data is available in the SCC transmit FIFO.

Note: The transmitter is stopped if this condition occurs and needs to be reset via command bit ’XRES’ in register CMDR. Furthermore the XDU interrupt indication MUST be cleared by generating an interrupt vector, thus bit ’XDU’ should not be masked via register IMR.

XOFF **XOFF Character Detected Interrupt** (async mode)

ASYNC Mode:

This bit is set to ’1’, if the currently received character matched the XOFF character programmed in bit field ’XOFF’ in register XNXF and indicates, that the transmitter is switched to XOFF-state if in-band flow control is enabled via bit ’FLON’ in register CCR2.

TIN **Timer Interrupt** (all modes)

This bit is set to ’1’, if the internal timer was activated and has expired (refer also to description of timer register TIMR).

CSC **$\overline{\text{CTS}}$ Status Change** (all modes)

This bit is set to ’1’, if a transition occurs on signal $\overline{\text{CTS}}$. The current state of signal $\overline{\text{CTS}}$ is monitored by status bit ’CTS’ in status register STAR.

XMR **Transmit Message Repeat** (hdlc/bisync modes)

HDLC Mode:

This bit is set to ’1’, if transmission of the last frame has to be repeated (by software), because

- the SCC has received a negative acknowledge to an I-frame in HDLC Automode operation;
- a collision occured after at least four bytes of data has been completely sent out, i.e. automatic re-transmission cannot be performed by the SCC;
- $\overline{\text{CTS}}$ signal was deasserted after at least four bytes of data has been completely sent out.

Note: For easy recovery from a collision event (in bus configuration only), the SCC transmit FIFO should not contain more than one complete frame. This can be achieved by using the ’ALLS’ interrupt to control the corresponding DMA controller transmit channel forwarding a new frame on all sent (ALLS) event only.

BISYNC Mode:

This bit is set to ’1’, if transmission of the last block of characters has to be repeated (by software), because $\overline{\text{CTS}}$ signal was deasserted after at least four bytes of data has been completely sent out.

XON **XON Character Detected Interrupt** (async mode)

ASYNC Mode:

This bit is set to ’1’, if the currently received character matched the XON character programmed in bit field ’XON’ in register XNXF and indicates, that the transmitter is switched to XON-state if in-band flow control is enabled via bit ’FLON’ in register CCR2.

XPR **Transmit Pool Ready Interrupt** (all modes)

This bit is set to ’1’, if a transmitter reset command was executed successfully (command bit ’XRES’ in register CMDR) and transmit data can be written to the FIFO by the DMA controller.

A ’XPR’ interrupt is not generated, if no sufficient transmit clock is available (depending on the selected clock mode).

BRK **Break Interrupt** (async mode)

This bit is set to ’1’, if a break condition was detected on the receive line, i.e. a low level for a time equal to (character length + parity bit + stop bit(s)) bits depending on the selected ASYNC character format.

BRKT **Break Terminated Interrupt** (async mode)

This bit is set to ’1’, if a previously detected break condition on the receive line is terminated by a low to high transition.

RDO **Receive Data Overflow Interrupt** (hdlc mode)

This bit is set to ’1’, if receive data of the current frame got lost because of a SCC receive FIFO full condition. However the rest of the frame is received and discarded as long as the receive FIFO remains full and is stored as soon as FIFO space is available again. The receive status byte (RSTA) of such a frame contains an ’RDO’ indication.

TCD **Termination Character Detected Interrupt** (async/bisync modes)

This bit is set to ’1’, if a termination character is detected in the receive data stream. The SCC will insert a ’frame end / block end’ indication to the SCC receive FIFO which causes the DMAC to finish the current receive descriptor.

RFS **Receive Frame Start Interrupt** (hdlc mode)

This bit is set to ’1’, if the beginning of a valid frame is detected by the receiver. A valid frame is detected either if a valid address field is recognized (in all operating modes with address recognition) or if a start flag is recognized (in all operating modes with no address recognition).

TIME **Time Out Interrupt** (async mode)

This bit is set to ’1’, if the time out limit is exceeded, i.e. no new character was received in a programmable period of time (refer to register CCR1 bit fields ’TOIE’ and ’TOLEN’ for more information).

RSC **Receive Status Change Interrupt** **(hdlc mode)**

This bit is valid in HDLC Automode only.

It is set to ’1’, if a status change of the remote station receiver has been detected by receiving a S-frame with receiver ready (RR) or receiver not ready (RNR) indication. Because only a status change is indicated via this interrupt, the current status can be evaluated by reading bit ’RRNR’ in status register STAR.

PERR **Parity Error Interrupt** (async/bisync modes)

This bit is only valid if parity checking/generation is enabled via bit ’PARE’ in register CCR2.

It is set to ’1’, if a character with wrong parity has been received. If enabled via bit ’RFDF’, this error status is additionally stored in the receive status byte generated for each receive character.

PCE **Protocol Error Interrupt** (hdlc mode)

This bit is valid in HDLC Automode only.

It is set to ’1’, if the receiver has detected a protocol error, i.e. one of the following events occured:

- an S- or I-frame was received with wrong N(R) counter value;
- an S-frame containing an I-control field was received.

FERR **Framing Error Interrupt** (async mode)

This bit is set to ’1’, if a character framing error is detected, i.e. a ’0’ was ampled at a position where a stop bit ’1’ was expected due to the selected character format.

SCD **SYN Character Detected Interrupt** (bisync mode)

This bit is set to ’1’, if a synchronization character (SYNC) was detected after the receiver was switched to HUNT-mode (by command bit ’HUNT’ in register CMDR).

PLLA **DPLL Asynchronous Interrupt** (all modes)

This bit is only valid, if the receive clock is derived from the internal DPLL and FM0, FM1 or Manchester data encoding is selected (depending on the selected clock mode and data encoding mode). It is set to ’1’ if the DPLL has lost synchronization. Reception is disabled until synchronization has been regained again. If the transmitter is supplied with a clock derived from the DPLL, transmission is also interrupted.

CDSC **Carrier Detect Status Change Interrupt** (all modes)

This bit is set to ’1’, if a state transition has been detected at signal CD. Because only a state transition is indicated via this interrupt, the current status can be evaluated by reading bit ’CD’ in status register STAR.

RFO **Receive FIFO Overflow Interrupt** (all modes)

HDLC Mode:

This bit is set to ’1’, if the SCC receive FIFO is full and a complete frame must be discarded. This interrupt can be used for statistical purposes and might indicate that the DMAC was not able to service the SCC receive FIFO quickly enough, e.g. PCI bus latencies are too bad.

ASYNC/BISYNC Mode:

This bit is set to ’1’, if the SCC receive FIFO is full and another received character must be discarded. This interrupt can be used for statistical purposes and might indicate that the DMAC was not able to service the SCC receive FIFO quickly enough, e.g. PCI bus latencies are too bad.

FLEX **Frame Length Exceeded Interrupt** (hdlc mode)

This bit is set to ’1’, if the frame length check feature is enabled and the current received frame is aborted because the programmed frame length limit was exceeded (refer to register RLCR for detailed description).

10.3.3 Peripheral Registers - Detailed Register Description

10.3.3.1 Peripheral Registers Overview

The DSCC4 Peripheral registers are used to configure and control the function blocks LBI, SSC and GPP.

The full 32 bit address location of each global register consists of:

- Base Address Register 0 (PCI Configuration Space, address location 10_H)
- Register address offset, which is in the range 0300_H ...$07FF_H$

All registers are 32-bit organized registers.

Table 88 provides an overview about all global registers:

Table 88 DSCC4 Peripheral Register Overview

Offset	Register	Meaning
LBI block specific registers:		
0300_H	LCONF	LBI Configuration Register
0304_H	*Reserved*	-
...		
$037C_H$		
SSC block specific registers:		
0380_H	SSCCON	SSC Control Register
0384_H	SSCBR	SSC Baud Rate Generator Register
0388_H	SSCTB	SSC Transmit Buffer
$038C_H$	SSCRB	SSC Receive Buffer
0390_H	SSCCSE	SSC Chip Select Enable Register
0394_H	SSCIM	SSC Interrupt Mask Register
0398_H	*Reserved*	-
...		
$03FC_H$		
GPP block specific registers:		
0400_H	GPDIR	GPP Direction Configuration Register
0404_H	GPDATA	GPP Data I/O Register
0408_H	GPIM	GPP Interrupt Mask Register
$040C_H$	*Reserved*	-
...		
$07FC_H$		

10.3.3.2 LBI Registers Description

Table 89 LCONF: LBI Configuration Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0300$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LBI General Configuration															
	LINTIC	0	0	0	0	0	0	0	0	$\overline{\text{EBCRES}}$	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LBI General Configuration															
	0	0	0	0	0	0	EALE	HDEN	BTYP(1:0)		RDEN	ABM	MCTC(3:0)			

LINTIC **LBI Interrupt Input Control** (-)

This bit selects whether the LBI interrupt input pin LINTI is a low or high active input signal:

LINTIC=’0’ LINTI input pin ia a low active input signal, i.e. LINTI=’0’ generates an interrupt indication.

LINTIC=’1’ LINTI input pin ia a high active input signal, i.e. LINTI=’1’ generates an interrupt indication.

$\overline{\text{EBCRES}}$ **LBI External Bus Controller Reset** (-)

Via this bit the complete LBI block can be reset (disabled) or enabled:

$\overline{\text{EBCRES}}$ =’0’ The LBI block is forced into its reset state. Also all dedicated pins are in reset state (same as hardware reset).

$\overline{\text{EBCRES}}$ =’1’ The LBI block is enabled. The function depends to the selected configuration.

Note: This Reset control bit is not self clearing. The LBI block remains in its reset state, until a ’1’ is written to bit ’$\overline{\text{EBCRES}}$’.

EALE **LBI Extended ALE** (-)

This bit selects whether the LBI ALE output signal is generated for one LBI clock period or an extended period:

EALE=’0’ ALE signal high time is 1 LBI clock period.

EALE=’1’ ALE signal high time is 1 LBI clock period + 1 PCI clock high time.

HDEN **LBI HOLD Enable** (-)

This bit selects whether the LBI bus arbitration interface (pins $\overline{\text{LHOLD}}$, $\overline{\text{LHDLA}}$, $\overline{\text{LBREQ}}$) is enabled or disabled:

HDEN=’0’ The LBI bus arbitration interface is disabled. The DSCC4 (LBI) is always active bus master.

HDEN=’1’ The LBI bus arbitration interface is enabled. The DSCC4 (LBI) shares bus mastership with one or more other bus masters.
The DSCC4 can be default arbitration master or arbitration slave depending on the setting of bit ’ABM’.

BTYP(1:0) **LBI Bus Type** (-)

The Local Bus Interface (LBI) supports 4 different bus configurations which are selected via this bit field:

BTYP = ’00’ 8 bit address/data de-multiplexed bus

BTYP = ’01’ 8 bit address/data multiplexed bus

BTYP = ’10’ 16 bit address/data de-multiplexed bus

BTYP = ’11’ 16 bit address/data multiplexed bus

Note: The Peripheral Configuration must be selected accordingly (bit field ’PERCFG’ in register GMODE).

RDEN **LBI LRDY Enable** (-)

This bit selects whether the $\overline{\text{LRDY}}$ control input signal is evaluated or ignored by the LBI:

RDEN=’0’ Input signal $\overline{\text{LRDY}}$ is ignored (but should be connected to a defined level). The bus cycle depends only on the selected number of wait states (bit field ’MCTC’).

RDEN=’1’ Input signal $\overline{\text{LRDY}}$ is evaluated after the number of selected wait states have been inserted. The bus transaction is terminated after the first detection of $\overline{\text{LRDY}}$=’0’ (active).

ABM **LBI Arbitration Master** (-)

The DSCC4 (LBI) is always master (initiator) of bus transactions on the local bus. Nevertheless the local bus can be shared with other master peripherals. In this case the bus arbitration interface must be enabled by setting bit ’HDEN’ to ’1’.
Bit ’ABM’ selects whether the DSCC4 (LBI) is arbitration default master or arbitration slave, i.e. another peripheral is arbitration default bus master.

ABM=’0’	The DSCC4 (LBI) is arbitration slave. Pin $\overline{\text{LHDLA}}$ of the bus arbitration interface is an input signal.
ABM=’1’	The DSCC4 (LBI) is arbitration default master. Pin $\overline{\text{LHDLA}}$ of the bus arbitration interface is an output signal.

MCTC(3:0) **LBI Memory Cycle Time Control** (-)

Via this bit field, a constant number of wait states can be selected for each LBI bus cycle (read and write). The wait states are inserted into the read and write strobe signal ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$) active time (based on LBI clock cycles):

MCTC	Wait States:
’0000’	15
’0001’	14
...	...
’1111’	0

Note: The minimum active time of read and write strobe signals is 2 LBI clocks. MCTC wait states are additional. If LRDY control is enabled, further wait states may be inserted depending on the $\overline{\text{LRDY}}$ input signal which is generated by the connected peripherals.

10.3.3.3 SSC Registers Description

Table 90 SSCCON: SSC Control Register

CPU Accessibility: **read/(write)**
(Do not write in operating mode,
i.e. write access with SSCEN = ’1’.)

Reset Value: **0000 0000$_H$**

Offset Address: **0380$_H$**

typical usage: written by CPU for control (configuration mode),
read by CPU for status information (operating mode)
evaluated/updated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mode	(unused)															
Control	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Status	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mode	SSC General Configuration And Status															
Control	SSCEN=’0’	SSCMS	0	0	SSCBEN	SSCPEN	SSCREN	SSCTEN	0	SSCPO	SSCPH	SSCHB	SSCBM(3:0)			
Status	SSCEN=’1’	SSCMS	0	SSCBSY	SSCBE	SSCPE	SSCRE	SSCTE	0	0	0	0	SSCBC(3:0)			

SSCEN **SSC Enable** (-)

This bit selects whether the SSC is in '**configuration mode**' or in normal '**operation mode**'. The meaning of bits 14..0 depends on the setting of this bit:

SSCEN='0' The SSC is in configuration mode. Bits 14..0 provide control bits for SSC configuration.

SSCEN='1' The SSC is in normal operation mode. Bits 14..0 provide status information bits.

SSCMS **SSC Master Select** (configuration/operation mode)

This bit selects whether the SSC is operating in master or in slave mode. Bit 'SSCMS' is valid in configuration and operation mode of register SSCCON:

SSCMS='0' The SSC is slave. Operation is performed by the shift clock, supplied at pin MCLK (input).

SSCMS='1' The SSC is master. Operation is performed by the internally generated shift clock which is monitored at pin MCLK (output).

SSCBSY **SSC Busy Status Flag** (operation mode)

This bit indicates that a transfer is currently in process:

SSCBSY ='0' No transfer is in process.

SSCBSY ='1' A transfer is currently in process.

SSCBEN **SSC Baud Rate Error Enable** (configuration mode)

This bit selects whether the SSC ignores or checks baud rate errors:

SSCREN ='0' The SSC ignores baud rate errors.

SSCREN ='1' The SSC checks baud rates errors.

SSCBE **SSC Baud Rate Status Flag** (operation mode)

This bit indicates a baud rate mismatch:

SSCBE='0' No baud rate mismatch is detected.

SSCBE='1' A baud rate mismatch is detected, i.e. the slaves baudrate deviates from the expected baud rate more than a factor of 2 or 0.5.

SSCPEN **SSC Phase Error Enable** (configuration mode)

This bit selects whether the SSC ignores or checks phase errors:

SSCPEN ='0' The SSC ignores phase errors.

SSCPEN ='1' The SSC checks phase errors.

SSCPE **SSC Baud Rate Status Flag** (operation mode)

This bit indicates a phase error:

SSCPE='0' No phase error is detected.

SSCPE='1' A phase error is detected, i.e. a transition occured on the receive data signal within a guard window around the sampling clock edge.

SSCREN **SSC Receive Error Enable** (configuration mode)

This bit selects whether the SSC ignores or checks receive errors:

SSCREN ='0' The SSC ignores receive errors.

SSCREN ='1' The SSC checks receive errors.

SSCRE **SSC Receive Status Flag** (operation mode)

This bit indicates a receive error:

SSCRE='0' No receive error is detected.

SSCRE='1' A receive error is detected, i.e. reception is completed before the receive buffer was read by the CPU.

SSCTEN **SSC Transmit Error Enable** (configuration mode)

This bit selects whether the SSC ignores or checks transmit errors:

SSCTEN ='0' The SSC ignores transmit errors.

SSCTEN ='1' The SSC checks transmit errors.

SSCTE **SSC Transmit Status Flag** (operation mode)

This bit indicates a transmit error:

SSCTE='0' No transmit error is detected.

SSCTE='1' A transmit error is detected, i.e. transmission starts before the transmit buffer has been updated by the CPU.

SSCPO **SSC Polarity Control** (configuration mode)

This bit selects the polarity of the clock:

SSCPO='0' The idle clock line is 'low'. Leading clock edge is a low-to-high transition.

SSCPO='1' The idle clock line is 'high'. Leading clock edge is a high-to-low transition.

SSCPH **SSC Clock Phase Control** (configuration mode)

This bit selects the active clock phase for operation. The definition of the leading clock edge depends on the setting of bit ’SSCPO’:

SSCPH=’0’ Transmit data is shifted with the leading clock edge, receive data is latched with the trailing clock edge.

SSCPH=’1’ Transmit data is shifted with the trailing clock edge, receive data is latched with the leading clock edge.

SSCHB **SSC Heading (Bit Order) Control** (configuration mode)

This bit selects if LSB or MSB is transmitted/received first:

SSCHB=’0’ LSB first operation on transmit and receive.

SSCHB=’1’ MSB first operation on transmit and receive.

SSCBM (3:0) **SSC Data Width Control** (configuration mode)

Via this bit field, the data width (active part of the transmit and receive buffers) can be selected in the range 2 to 16 bit:

SSCBM	Data width:
’0000’	*Reserved*. Do not use.
’0001’	Data width is 2 bit.
’0010’	Data width is 3 bit.
...	...
’1111’	Data width is 16 bit.

SSCBC (3:0) **SSC Shift Counter** (operation mode)

This bit field is used by the SCC as shift counter and is updated with every bit shift operation.

Table 91 SSCBR: SSC Baud Rate Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0384$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSC Baud Rate Generator															
	SSCBR(15:0)															

These bits define the baud rate used for data transfer via the SSC interface. Reading SSCBR (while SSC is enabled via bit 'SSCEN' in register SSCCON) returns the timer value. Reading SSCBR (while SSC is disabled) returns the programmed reload value. The desired reload value of the baud rate can be written to SSCBR when the SSC interface is disabled. **Table 92** shows example values for register SSCBR, assuming a PCI clock frequency of 20 MHz.

Table 92 SSC Baud Rate Values

SSCBR(15:0)	Baud Rate	Bit Time
0000_H	Reserved. Use a reload value > 0.	---
0001_H	5 MBaud	200 ns
0002_H	3.3 MBaud	300 ns
0003_H	2.5 MBaud	400 ns
0004_H	2.0 MBaud	500 ns
0009_H	1.0 MBaud	1 µs
0063_H	100 KBaud	10 µs
$03E7_H$	10 KBaud	100 µs
$270F_H$	1.0 KBaud	1 ms
$FFFF_H$	152.6 Baud	6.6 ms

Note: The contents of SSCBR must always be > 0.
Never write to SSCBR, while the SSC is enabled.

Table 93 SSCTB: SSC Transmit Buffer Register

CPU Accessibility: **write**
Reset Value: **0000 0000$_H$**
Offset Address: **0388$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSC Transmit Buffer															
	SSCTB(15:0)															

Bit field ’SSCTB’ is written by the CPU and contains the transmit data to be transmitted. The number of valid bits depend on bit field ’SSCBM’ in register SSCCON (configuration mode).

Table 94 SSCRB: SSC Receive Buffer Register

CPU Accessibility: **read**
Reset Value: **0000 0000$_H$**
Offset Address: **038C$_H$**
typical usage: written by DSCC4
evaluated by CPU

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSC Receive Buffer															
	SSCRB(15:0)															

Bit field ’SSCRB’ is read by the CPU and contains the receive data. The number of valid bits depend on bit field ’SSCBM’ in register SSCCON (configuration mode).

Table 95 SSCCSE: SSC Chip Select Enable Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0390$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSC Chip Select Control															
	0	0	0	0	0	0	0	0	ASEL3	ASEL2	ASEL1	ASEL0	0	0	0	0

ASEL3 **SSC Chipselect 3** (-)

ASEL2 **SSC Chipselect 2** (-)

ASEL1 **SSC Chipselect 1** (-)

ASEL0 **SSC Chipselect 0** (-)

These bits determine the function of the chipselect signals $\overline{\text{MCSi}}$ (i=3..0):

ASELi=’0’ The $\overline{\text{MCSi}}$ chipselect pin is active low (constant ’0’).

ASELi=’1’ The $\overline{\text{MCSi}}$ chipselect pin is controlled automatically by the SSC transmitter.

Table 96 SSCIM: SSC Interrupt Mask Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0394$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SSC Interrupt Mask Control															
	0	0	0	0	0	0	0	0	0	0	0	0	0	IMTX	IMER	IMRX

IMRX **SSC Receive Interrupt Mask** (-)

This bit enables/disables receive interrupt indications:

IMRX=’0’ SSC receive interrupts are disabled.

IMRX=’1’ SSC receive interrupts are enabled.

IMER **SSC Error Interrupt Mask** (-)

This bit enables/disables error interrupt indications:

IMER=’0’ SSC error interrupts are disabled.

IMER=’1’ SSC error interrupts are enabled.

IMTX **SSC Transmit Interrupt Mask** (-)

This bit enables/disables transmit interrupt indications:

IMTX=’0’ SSC transmit interrupts are disabled.

IMTX=’1’ SSC transmit interrupts are enabled.

Note: The transmit interrupt notifies the CPU about the start of a transmission. The receive interrupt notifies transfer of the received data to the shared memory. The error interrupt notifies the CPU about different error conditions related to data transmission and reception. To further specify what sort of error interrupt the user wants to trace, the corresponding bits of the SSC control register SSCCON has to be set. The SSC error conditions that can be checked are transmit errors (SSCCON(bit 8) =‘1‘), phase errors (SSCCON(bit 10)=‘1‘) and baud rate errors (SSCCON(bit 11)=‘1‘). If any of these error conditions shall not be checked, the corresponding enable bit has to be set to ‘0‘.

Example

To check for transmit errors only:

SSCIM(bit 1)=‘1‘, SSCCON(bit 8)=‘1‘, SSCCON(bit 10)=‘0‘,SSCCON(bit 11)=‘0‘

10.3.3.4 GPP Registers Description

Table 97 GPDIR: GPP Direction Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0400$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GPP I/O Signal Direction Control															
	GPDIR(15:0)															

GPDIR (15:0) **GPP I/O Signal Direction Control** (-)

Each bit of this bit field controls the direction (input/output) of the corresponding General Purpose Pin. E.g. GPDIR bit 8 determins the I/O characteristic of pin GP8.

Note: Even if not configured for GPP operation (bit field 'PERCFG' in register GMODE), this register must be programmed appropriately for correct SSC or LBI operation.
(For detailed information refer to the chapters describing SSC and LBI operation.)

GPDIR(i)='0' Pin GPi is configured as input pin.

GPDIR(i)='1' Pin GPi is configured as output pin.

Table 98 GPDATA: GPP Data Register

CPU Accessibility: **read/write**

Reset Value: **0000 0000$_H$**

Offset Address: **0404$_H$**

typical usage: read by CPU for input signals, written by CPU for output signals;

written by DSCC4 for output signals, evaluated by DSCC4 for input signals;

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GPP I/O Data Control															
	GPDATA(15:0)															

GPDATA (15:0) **GPP I/O Data Control** (-)

Each bit of this bit field is related to the corresponding GPP signal pin. The function of each bit depends on the I/O configuration of the dedicated GPP pin:

Output pins (GPDIR(i)=’1’ in register GPDIR):

Write access:

GPDATA(i) =’0’ Pin GPi output pin is set to ’0’ (low).

GPDATA(i) =’1’ Pin GPi output pin is set to ’1’ (high).

Read access:

GPDATA(i) =’0’ Current level of GPi output pin is ’0’ (low).

GPDATA(i) =’1’ Current level of GPi output pin is ’1’ (high).

Input pins (GPDIR(i)=’0’ in register GPDIR):

Write access:

Write access to GPDATA bit locations related to input signals is ignored.

Read access:

GPDATA(i) =’0’ Current level of GPi input pin is ’0’ (low).

GPDATA(i) =’1’ Current level of GPi input pin is ’1’ (high).

Table 99 GPIM: GPP Interrupt Mask Register

CPU Accessibility: **read/write**
Reset Value: **0000 0000$_H$**
Offset Address: **0408$_H$**
typical usage: written by CPU
evaluated by DSCC4

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	(unused)															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	GPP Interrupt Mask Control															
	GPIM(15:0)															

GPIM(15:0) GPP Interrupt Mask Control (-)

Each bit of this bit field enables/disables interrupt generation in case of transitions on the corresponding GPP pins. Even if not configured for GPP operation (bit field ’PERCFG’ in register GMODE), this register should be programmed appropriately for correct SSC or LBI operation (masking interrupt generation).
Interrupt generation should be disabled for GPP pins, configured as output pins via bit field ’GPDIR’ in register GPDIR.

GPIM(i)=’0’ Pin GPi interrupt generation is enabled.

GPIM(i)=’1’ Pin GPi interrupt generation is disabled.

11 Host Memory Organization

11.1 Linked List Structure

11.1.1 Transmit Descriptor Lists

Each transmit descriptor consists of 4 consecutive DWORDs located DWORD aligned in the shared memory. The first 3 DWORDs are read by the corresponding DMA channel using a burst transaction and provide information about the next descriptor in the linked list, the attached transmit data buffer and its size as well as some control bits.

The fourth DWORD is written by the DMA channel indicating that operation on this descriptor is finished.

The CPU will write the address of the first descriptor of each linked list to a dedicated Base Address Register during initialization procedure. The corresponding DMA channels start operating the linked lists at these addresses.

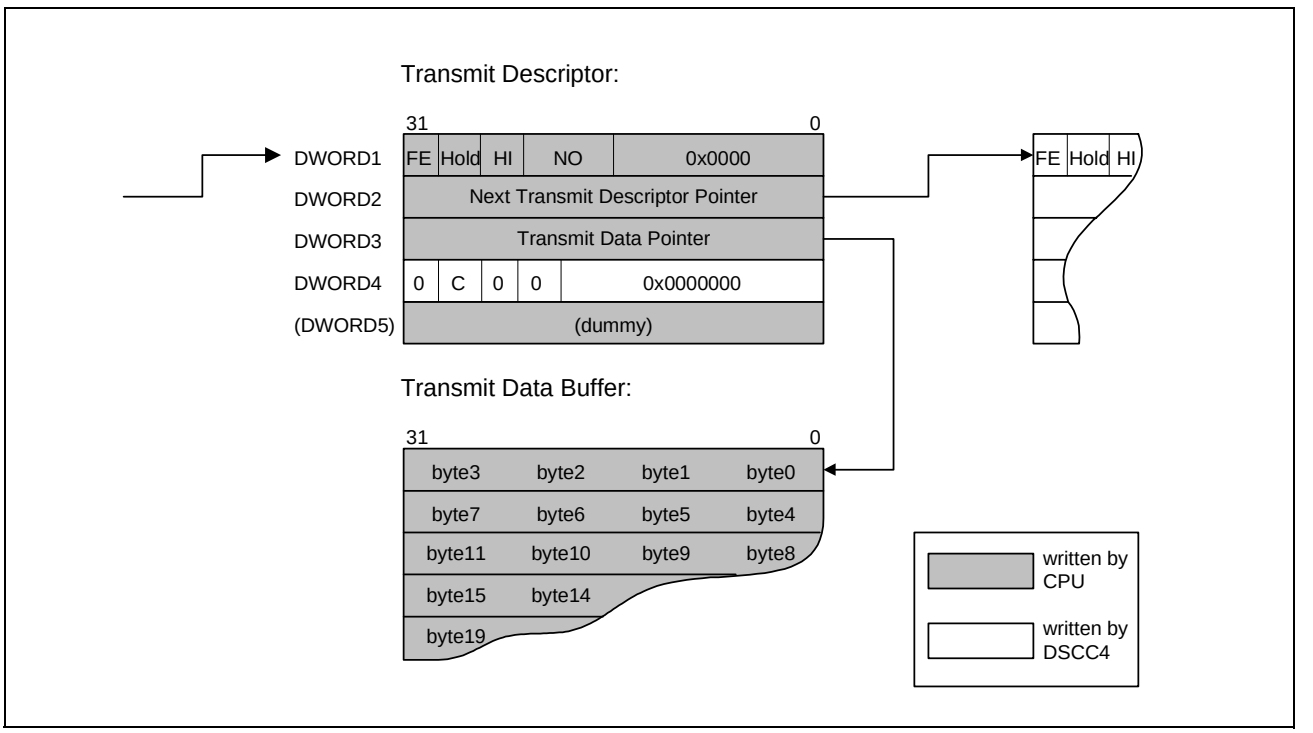


Figure 77 Transmit Descriptor List Structure

11.1.1.1 Transmit Descriptor

The transmit descriptor lists are prepared by the host within the shared memory and read by DSCC4 DMA controller, when requested to do by the host either via an 'AR' (Action Request) command or an transmit poll command or after branching from previous transmit descriptors. The handling of transmit descriptor lists is described in details in

chapter **“DMAC Transmit Descriptor Lists” on Page 66**. The transmit descriptor contains 4 DWORDs which are described in the following table:

Table 100 Transmit Descriptor

DWORD	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	FE	HOLD	HI	NO												
1	Next Transmit Descriptor Pointer															
2	Transmit Data Pointer															
3	0	**C**	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DWORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Next Transmit Descriptor Pointer															
2	Transmit Data Pointer															
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FE: **Frame End, set by the host**
It indicates that the current transmit data section (addressed by Transmit Data Pointer) contains the end of a frame (HDLC, PPP) or the end of data block (ASYNC, BISYNC). When transferring the last data from this transmit data section into the internal FIFO the DMAC marks these data with an 'frame end / block end' indication bit.
GMODE.CMOD='0':
After that it checks the HOLD bit stored in the on-chip memory. If HOLD='0', it branches to the next transmit descriptor. Otherwise the corresponding DMAC transmit channel is deactivated as long as the host CPU does not request reactivation via the GCMDR register (either transmit poll request or action request with 'IDT' command).
GMODE.CMOD='1':
After that it checks if the first (current) transmit descriptor address (LTDA) is equal to the last transmit descriptor address (LTDA) stored in the corresponding channel specific on-chip registers. When both addresses differ, it branches to the next transmit descriptor. Otherwise the corresponding DMAC transmit channel is deactivated as long as the host CPU does not write a new LTDA value to LTDA register or provides an action request with 'IDT' command.

HOLD: **Hold** (only valid when GMODE.CMODE=0)
It indicates whether the current descriptor is the last element of a linked list or not:

HOLD='0': A next descriptor is available in the shared memory; after checking the HOLD bit stored in the on-chip memory the DMAC branches to next transmit descriptor

HOLD='1': The current descriptor is the last one that is available for the DMAC. The corresponding DMAC channel is deactivated for transmit direction as long as the microprocessor does not request an activation via the CMDR register.

NO: **Byte Number**
This byte number defines the number of bytes stored in the data section to be transmitted. Thus the maximum length of data buffer is 8191 bytes (i.e. NO = $1FFF_H$). A transmit descriptor and the corresponding data section must contain at least either one data byte or a frame end indication. Otherwise an DMA controller interrupt with 'ERR' bit set is generated.

HI: **Host Initiated Interrupt**
If the HI bit is set, the corresponding DMAC generates an interrupt with set HI bit after transferring all data bytes of the current data section.

Next Transmit Descriptor Pointer: This 32-bit pointer contains the start address of the next transmit descriptor. After sending the indicated number of data bytes, DSCC4 branches to the next transmit descriptor to continue transmission. The transmit descriptor is read entirely at the beginning of transmission and stored in on-chip memory. Therefore all information in the next descriptor must be valid when the DSCC4 branches to this descriptor.
This pointer is not used if a transmitter reset or initialization channel command is detected while the DSCC4 still reads data from the current transmit descriptor. In this case BTDA value in the BTDA register is used as a pointer for the next transmit descriptor to be branched to.

Transmit Data Pointer: This 32-bit pointer contains the start address of the transmit data section. Although DSCC4 works long word oriented, it is possible to begin transmit data section at byte addresses.

C: **Complete**
This bit is set by the DSCC4 if
- it completes reading data section normally
- it was aborted by a transmitter reset command**.**

11.1.2 Receive Descriptor Lists

Each receive descriptor consists of 5 consecutive DWORDs located DWORD aligned in the shared memory. The first 3 DWORDs are read by the corresponding DMA channel using a burst transaction and provide information about the next descriptor in the linked list, the attached receive data buffer and its size as well as some control bits.

The fourth DWORD is written by the DMA channel indicating that operation on this descriptor is finished. The fifth DWORD is also written by the DMA channel but only in descriptors containing the first or only data section of an HDLC frame or data block. It is a pointer to the last descriptor containing the frame or block end (’FE’ bit) allowing the software to unchain the complete partial descriptor list containing one frame or block without parsing through the list for ’FE’ indication.

The CPU will write the address of the first descriptor of each linked list to a dedicated Base Address Register during initialization procedure. The corresponding DMA channels start operating the linked lists at these addresses.

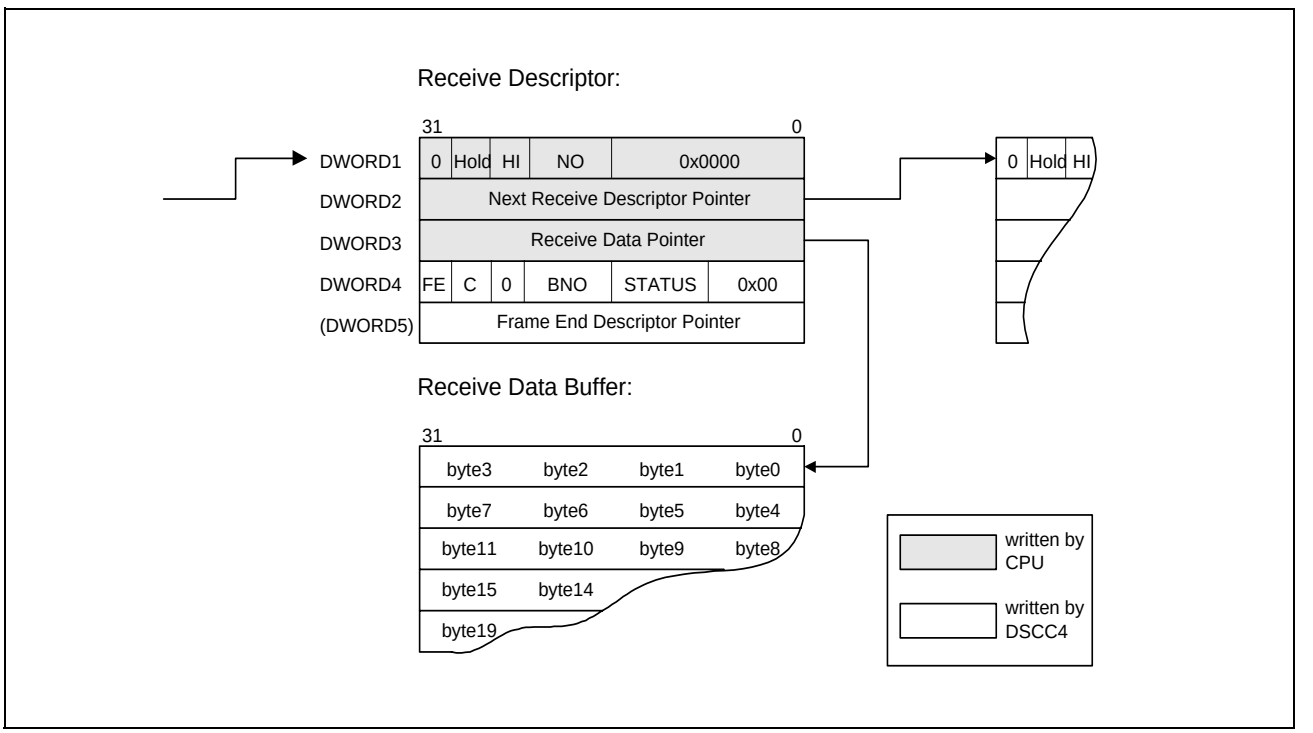


Figure 78 Receive Descriptor List Structure

11.1.2.1 Receive Descriptor

The receive descriptor lists are prepared by the host within the shared memory and read by DSCC4 DMA controller, when requested to do by the host either via an ’AR’ (Action Request) command or after branching from previous receive descriptors. The handling of receive descriptor lists is described in details in chapter **“DMAC Receive Descriptor Lists” on Page 71**. The receive descriptor contains 5 DWORDs which are described in the following table:

Table 101 Receive Descriptor

DWORD	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	HOLD	HI	NO												
1	Next Receive Descriptor Pointer															
2	Receive Data Pointer															
3	FE	**C**	0	BNO												
4	Frame End Descriptor Pointer															

DWORD	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	Next Receive Descriptor Pointer															
2	Receive Data Pointer															
3	STATUS								0	0	0	0	0	0	0	0
4	Frame End Descriptor Pointer															

HOLD: **Hold** (only valid when GMODE.CMODE=0)
It indicates whether the current descriptor is the last element of a linked list or not:

HOLD='0': A next descriptor is available in the shared memory; after checking the HOLD bit stored in the on-chip memory the DMAC branches to next receive descriptor

HOLD='1': The current descriptor is the last one that is available for the DMAC. After completion of the current receive descriptor an interrupt is generated and the corresponding DMAC channel is deactivated for receive direction as long as the microprocessor does not request an activation via the CMDR register.

HI: **Host Initiated Interrupt**
If the HI bit is set, the corresponding DMAC generates an interrupt with set HI bit after transferring all data bytes into the current data section.

NO: **Byte Number**
This byte number defines the size of the receive data section allocated by the host. It has to be a multiple of 4 bytes which is responsibility of the software. The maximum buffer length is 8188 bytes (i.e. NO = $1FFC_H$).

Next Receive Descriptor Pointer: This 32-bit pointer contains the start address of the next receive descriptor. After completion of the current receive descriptor the DSCC4 branches to the next receive descriptor to continue reception. The receive descriptor is read entirely at the beginning of reception and stored in on-chip memory. Therefore all information in the next descriptor must be valid when the DSCC4 branches to this descriptor.
This pointer is not used if a receiver reset command is detected while the DSCC4 still writes data to the current receive descriptor. In this case BRDA is used as a pointer for the next receive descriptor to be branched to.

Receive Data Pointer: This 32-bit pointer contains the start address of the receive data section. The start address must be DWORD aligned.

FE: **Frame End**
It indicates that the current receive data section (addressed by Receive Data Pointer) contains the end of a frame (HDLC, PPP) or the end of data block (ASYNC, BISYNC, XTRANS). This bit is set by the DMAC after transferring the last data from the internal FIFO (indicated by the END bit) into the receive data section. Moreover the BNO and STATUS is updated and the 'C' bit is set by the DMAC.
GMODE.CMODE='0':
After that it checks the HOLD bit stored in the on-chip memory. If HOLD='0', it branches to the next receive descriptor. Otherwise the corresponding DMAC receive channel is deactivated as long as the host CPU does not request reactivation via the GCMDR register (action request with 'IDR' command).
GMODE.CMODE='1':
After that it checks if the first (current) receive descriptor address (LRDA) is equal to the last receive descriptor address (LRDA) stored in the corresponding channel specific on-chip registers. When both addresses differ, it branches to the next receive descriptor. Otherwise the corresponding DMAC receive channel is deactivated as long as the host CPU does not write a new LRDA value to LRDA register or provides an action request with 'IDR' command.

C: **Complete**
This bit is set by the DSCC4 if
- it completed filling data section normally
- it was aborted by a receiver reset command
- end of frame (HDLC, PPP) or end of block (ASYNC, BISYNC, BTRANS) was stored in the receive data section.

BNO: **Byte Number of Received Data**
DSCC4 writes the number of data bytes it has stored in the current data section into BNO

Frame End Descriptor Pointer: This 32-bit pointer is valid only in the descriptor, that contains the data pointer to the first data section of an HDLC frame or ASYNC/BISYNC/BTRANS block. This pointer is updated by the DSCC4 with the address of the descriptor that contains the data pointer to the last section (FE) of the HDLC frame or ASYNC/BISYNC/BTRANS block.

Receive descriptor STATUS bit field:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	**RA**	0

RA: **Receive Abort**
This bit indicates that the reception of a frame (HDLC, PPP) or block (ASYNC, BISYNC) was ended by a DMA receiver reset command or by a HOLD bit in the current receive descriptor or by a FRDA=LRDA.condition.

11.1.2.2 Receive Data Section Status Byte (HDLC Mode)

In HDLC protocol mode, the last byte of a frame (Receive Status Byte, RSTA) - located in the data section - contains error indications caused by the SCC (e.g. CRC, receive abort, …).

RSTA

7	6	5	4	3	2	1	0
VFR	RDO	CRC	RAB	HA1	HA0	C/R	LA

The contents of the RSTA byte relates to the received HDLC frame and is generated when end-of-frame is recognized at the serial receive interface.

VFR… **Valid Frame**

Determines whether a valid frame has been received.
1…valid
0…invalid
An invalid frame is either a frame which is not an integer number of 8 bits (n * 8 bits) in length (e.g. 25 bits), or a frame which is too short taking into account the operation mode selected via CCR1 (MDS1, MDS0, ADM) and the selected CRC algorithm (CCR1:C32) as follows:
for CCR2:DRCRC = ’0’: (CCR2:RCRC has no affect)

- auto-/non-auto mode (16-bit address)
  4 bytes (CRC-CCITT) or 6 (CRC-32)
- auto-/non-auto mode (8-bit address)
  3 bytes (CRC-CCITT) or 5 (CRC-32)
- transparent mode 1:
  3 bytes (CRC-CCITT) or 5 (CRC-32)
- transparent mode 0:
  2 bytes (CRC-CCITT) or 4 (CRC-32)

Shorter frames are not reported anyway.
for CCR2:DRCRC = ’1’: (CCR2:RCRC has no affect)

- auto-/non-auto mode (16-bit address):
  2 bytes
- auto-/non-auto mode (8-bit address):
  1 byte
- transparent mode 1:
  1 byte
- transparent mode 0:
  1 byte

Shorter frames are not reported anyway.

RDO… **Receive Data Overflow**

A data overflow has occurred during reception of the frame. Additionally, an interrupt can be generated (refer to ISR:RDO/IMR:RDO).

CRC… **CRC Compare/Check**

0…CRC check failed, received frame contains errors.
1…CRC check OK, received frame is error-free.

RAB… **Receive Message Aborted**

The received frame was aborted from the transmitting station. According to the HDLC protocol, this frame must be discarded by the receiver station.

HA1… **High Byte Address Compare**

HA0… Significant only if an HDLC mode with automatic address handling has been selected. In operating modes which provide high byte address recognition, the DSCC4 compares the high byte of a 2-byte address with the contents of two individually programmable addresses (RADR:RAH1, RADR:RAH2) and the fixed values FE_H and FC_H (broadcast address). Dependent on the result of this comparison, the following bit combinations are possible:
10…RAH1 has been recognized.
00…RAH2 has been recognized.
01…broadcast address has been recognized.
If RAH1, RAH2 contain identical values, a match is indicated by ‘10’.

C/R… **Command/Response**

Significant only if 2-byte address mode has been selected. Value of the C/R bit (bit in high address byte) in the received frame. The interpretation depends on the setting of the CRI bit in the RADR register. Refer also to the description of RADR register.

LA… **Low Byte Address Compare**

Not significant in transparent and extended transparent operating mode. the below byte address of a 2-byte address field, or the single address byte of a 1-byte address field is compared with two addresses (RADR:RAL1, RADR:RAL2).
0…RAL2 has been recognized.
1…RAL1 has been recognized.
According to the X.25 LAPB protocol, RAL1 is interpreted as the address of a COMMAND frame and RAL2 is interpreted as the address of a RESPONSE frame.

11.1.2.3 Receive Data Section Status Byte (ASYNC/BISYNC Modes)

In ASYNC/BISYNC protocol mode additionally to every data byte, an attached status byte can be stored (CCR2:RFDF=’1’).

The data character and status character format is determined as follows:

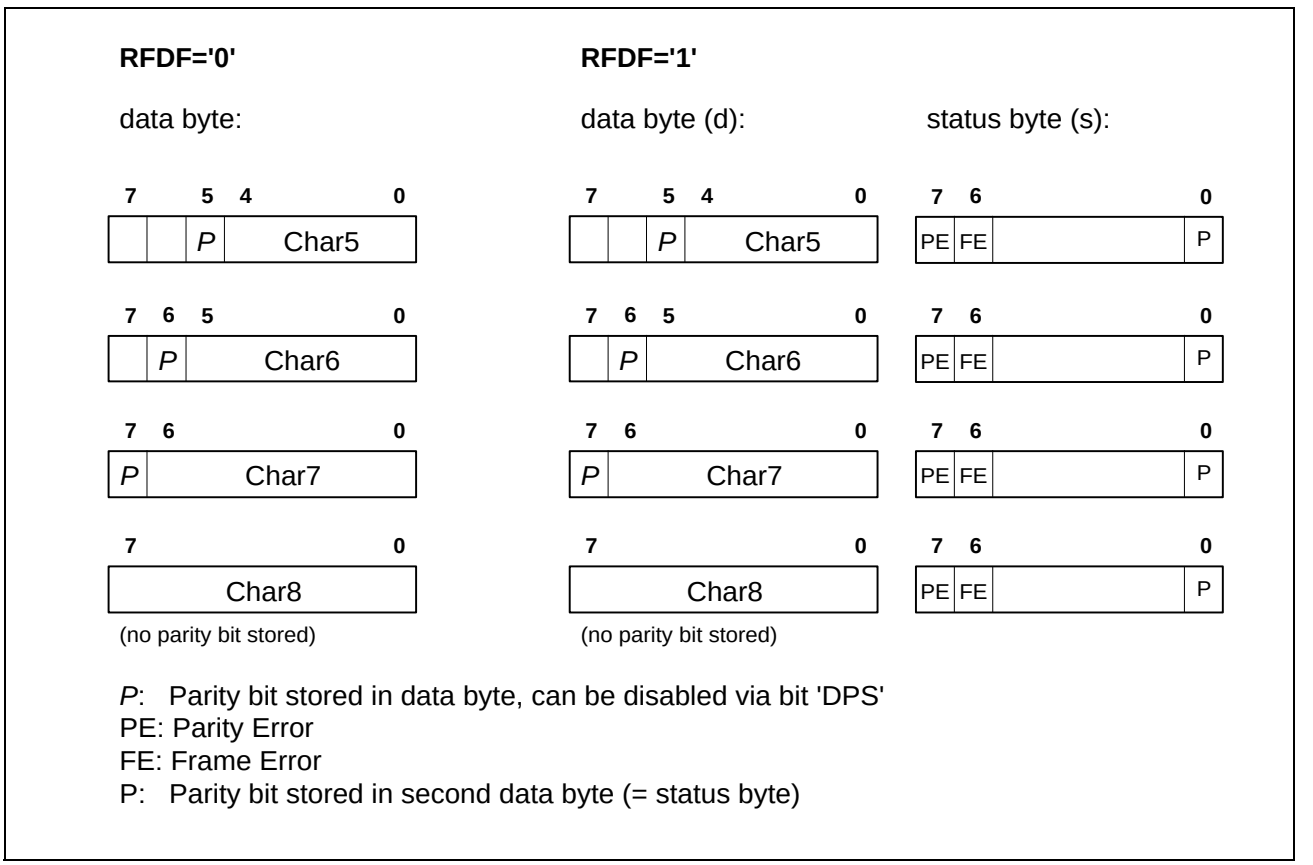


Figure 79 ASYNC/BISYNC Receive Status Character Format

Note: The ’Frame Error’ (FE) status bit is only valid in ASYNC protocol mode.

11.2 Interrupt Queue Structure

11.2.1 Interrupt Queue Overview

The DSCC4 interrupt concept is based on 32-bit interrupt vectors generated by the different blocks. Interrupt vectors are stored in a central interrupt FIFO which is 16 DWORDs deep. The interrupt controller transfers available vectors in one of ten circular interrupt queues located in the shared memory. Each queue is dedicated to the interrupt source.

In addition new interrupt vectors are indicated in the global status register GSTAR on a per queue basis and selectively confirmed by writing ’1’ to the corresponding GSTAR bit positions. The PCI interrupt signal $\overline{\text{INTA}}$ is asserted with any new interrupt event and remains asserted until all events are confirmed.

(For more detailed information refer to chapter **“DMAC Interrupt Controller” on Page 81**.)

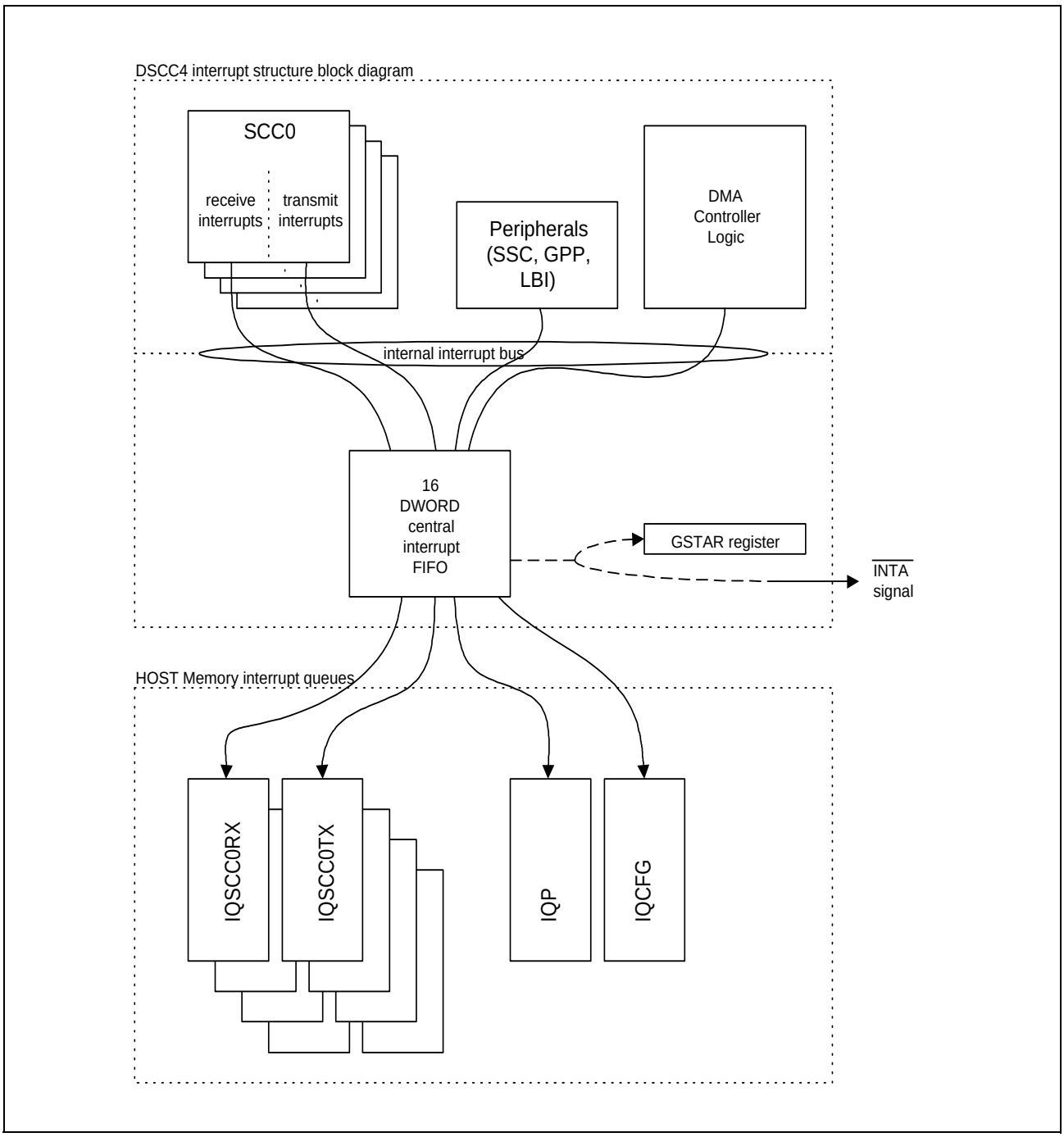


Figure 80 DSCC4 Logical Interrupt Structure

11.2.2 Interrupt Vector Overview

Figure 81provides an overview about all DSCC4 interrupt vectors. The different interrupt sources (types) are distinguished by the most significant 8 bit of the 32-bit interrupt vector. The dedicated host memory interrupt queues, the vectors are transferred to, are referenced in brackets under each interrupt vector name:

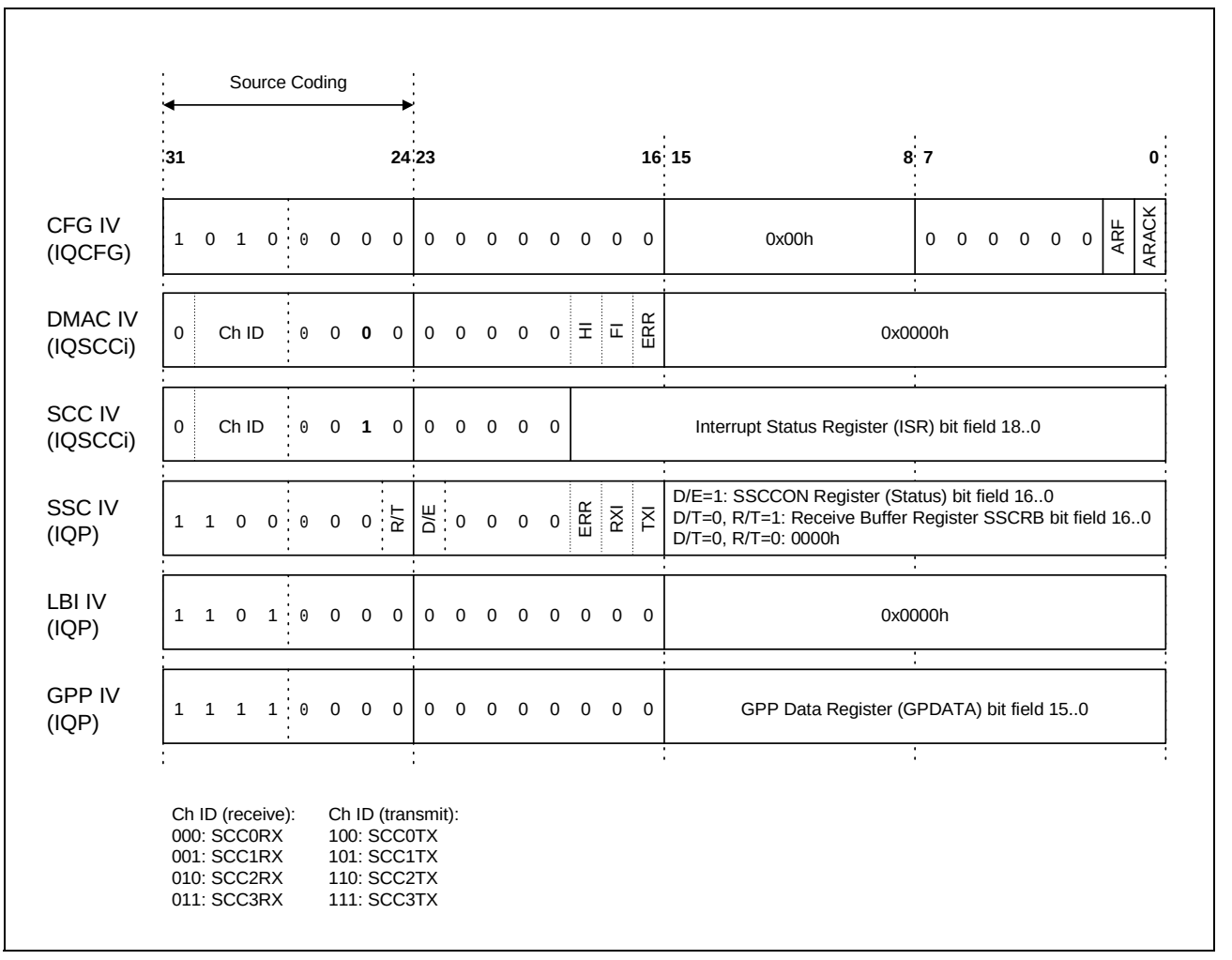


Figure 81 Interrupt Vector Overview

11.2.2.1 Configuration Interrupt Vector

Configuration interrupt vectors are transferred to Configuration Interrupt Queue 'IQCFG'.

Table 102 CFGIV: Configuration Interrupt Vectori

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Source Id=**1010**				0	0	0	0	0	0	0	0	0	0	0	0

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

Source ID **1010_2** Configuration Interrupt Vector (IQCFG)

ARF **Action Request Failed Interrupt**

This bit indicates that an action request command was completed with an 'action request failed' condition:

ARF='0' No action request was performed or no 'action request failed' condition occured completing an action request.

ARF='1' The last action request command was completed with an 'action request failed' condition.

ARACK **Action Request Acknowledge Interrupt**

This bit indicates that an action request command was completed successfully:

ARACK='0' No action request was performed or completed successfully.

ARACK='1' The last action request command was completed successfully.

11.2.2.2 DMA Controller Interrupt Vector

DMA controller interrupt vectors are transferred to the corresponding channel and direction specific interrupt queues IQSCCiRX and IQSCCiTX respectively.

Table 103 DMA Controller Interrupt Vectori

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	Source ID			0	0	**0**	0	0	0	0	0	0	HI	FI	ERR

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

Source ID

$\mathbf{000_2}$	Receive Channel 0 Interrupt Vector (IQSCC0RX)
$\mathbf{001_2}$	Receive Channel 1 Interrupt Vector (IQSCC1RX)
$\mathbf{010_2}$	Receive Channel 2 Interrupt Vector (IQSCC2RX)
$\mathbf{011_2}$	Receive Channel 3 Interrupt Vector (IQSCC3RX)
$\mathbf{100_2}$	Transmit Channel 0 Interrupt Vector (IQSCC0TX)
$\mathbf{101_2}$	Transmit Channel 1 Interrupt Vector (IQSCC1TX)
$\mathbf{110_2}$	Transmit Channel 2 Interrupt Vector (IQSCC2TX)
$\mathbf{111_2}$	Transmit Channel 3 Interrupt Vector (IQSCC3TX)

HI **Host Initiated interrupt** (Rx/Tx Channel)

This bit indicates that an Host Initiated (HI) interrupt occured, i.e. the corresponding DMA controller channel detects the ’HI’ bit set to ’1’ in the receive or transmit descriptor before branching to the next descriptor.

HI=’0’ No Host Initiated (HI) interrupt is indicated by this vector.

HI=’1’ An Host Initiated (HI) interrupt is indicated by this vector.

FI **Frame Indication interrupt** (Rx/Tx Channel)

This bit indicates that an Frame Indication (FI) interrupt occured.
Receive direction:
FI=’1’ indicates, that a frame has been received completely or was stopped by a DMAC receiver reset command or a hold condition set in a receive descriptor. It is set when the DSCC4 branches from the last descriptor belonging to the current frame (or block) (FE=’1’) to the first descriptor of a new frame. It is also set when the descriptor in which the frame/block is finished contained a hold condition.

Transmit direction:
Issued if the ’FE’ bit is detected in the transmit descriptor. It is set when the DSCC4 branches to the next transmit descriptor, belonging to a new frame or when ’HOLD’ bit is set in conjunction with ’FE’ bit. Only ’ERR’ indication without ’FI’ is set, if a transmit descriptor contains a ’HOLD’ (hold condition) but no ’FE’ bit.

FI=’0’	No Frame Indication (FI) interrupt is indicated by this vector.
FI=’1’	An Frame Indication (FI) interrupt is indicated by this vector.

ERR **ERROR Indication interrupt** (Rx/Tx Channel)

This bit indicates that an Error interrupt occured.
Receive direction:
Issued if the current frame/block could not be transferred to the shared memory completely, because of a hold condition in a receive descriptor not providing enough bytes for the frame/block or the frame/block was aborted by a DMAC receiver reset command.
Transmit direction:
Issued if a transmit descriptor contains a hold condition but FE=’0’ or if the last descriptor had NO=0 and FE=’0’.

ERR=’0’	No Error (ERR) interrupt is indicated by this vector.
ERR=’1’	An Error (ERR) interrupt is indicated by this vector.

11.2.2.3 SCC Interrupt Vector

Serial Channel (SCC) related interrupt vectors are transferred to the corresponding channel and direction specific interrupt queues IQSCCiRX and IQSCCiTX respectively.

Note: Interrupt vectors generated by the SCCs might contain interrupt indications for both, receive AND transmit direction. But in receive interrupt queues only the receive interrupt indications need to be served and in transmit interrupt queues only transmit interrupt indications need to be served by the software.

Table 104 SCC Interrupt Vector

Mode	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
H	0	Source ID			0	0	**1**	0	0	0	0	0	0	ALLS	0	XDU
A	0	Source ID			0	0	**1**	0	0	0	0	0	0	ALLS	0	XOFF
B	0	Source ID			0	0	**1**	0	0	0	0	0	0	ALLS	0	XDU

Mode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H	TIN	CSC	XMR	XPR	0	0	0	0	RDO	RFS	RSC	PCE	PLLA	CDSC	RFO	FLEX
A	TIN	CSC	XON	XPR	0	0	BRK	BRKT	TCD	TIME	PERR	FERR	PLLA	CDSC	RFO	0
B	TIN	CSC	XMR	XPR	0	0	0	0	TCD	0	PERR	SCD	PLLA	CDSC	RFO	0

Source ID	$\mathbf{000_2}$	Receive Channel 0 Interrupt Vector (IQSCC0RX)
	$\mathbf{001_2}$	Receive Channel 1 Interrupt Vector (IQSCC1RX)
	$\mathbf{010_2}$	Receive Channel 2 Interrupt Vector (IQSCC2RX)
	$\mathbf{011_2}$	Receive Channel 3 Interrupt Vector (IQSCC3RX)
	$\mathbf{100_2}$	Transmit Channel 0 Interrupt Vector (IQSCC0TX)
	$\mathbf{101_2}$	Transmit Channel 1 Interrupt Vector (IQSCC1TX)
	$\mathbf{110_2}$	Transmit Channel 2 Interrupt Vector (IQSCC2TX)
	$\mathbf{111_2}$	Transmit Channel 3 Interrupt Vector (IQSCC3TX)

Bit field 18..0 of the SCC interrupt vector is a copy of the SCC Interrupt Status Register ISR. The meaning of the bit field depends on the selected protocol mode (HDLC (H), ASYNC (A), BISYNC (B)).
(For detailed information on Interrupt Status Register ISR refer to **Table 87 "ISR: Interrupt Status Register" on Page 342**.)

11.2.2.4 SSC Interrupt Vector

SSC interrupt vectors are transferred to the peripheral interrupt queue IQP.

Table 105 SSC Interrupt Vector

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	0	0	0	0	0	R/T	D/E	0	0	0	0	ERR	RX	TX

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISW(15:0)															

(Source ID) bit field 31..28 **1100_2** SSC Interrupt Vector (IQP)

R/T **Receive/Transmit Indicator**

This bit indicates whether the SSC vector is generated for transmit or receive interrupt events:

R/T=’0’ SSC transmit interrupt vector.

R/T=’1’ SSC receive interrupt vector.

D/E **Data/Error Indicator**

This bit indicates whether the SSC vector is generated for data (transmit/ receive) or error interrupt events:

D/E=’0’ SSC error interrupt vector.

D/E=’1’ SSC data interrupt vector.

ERR **Error interrupt**

This bit indicates that an error interrupt occured:

ERR=’0’ No error interrupt is indicated by this vector.

ERR=’1’ An error interrupt is indicated by this vector.
Bit field 15..0 contains the corresponding bit field 15..0 of register SSCCON (operation mode).

RX **Receive interrupt**

This bit indicates that a receive interrupt occured:

RX=’0’ No receive interrupt is indicated by this vector.

RX=’1’ A receive interrupt is indicated by this vector.
Bit field 15..0 contains the corresponding bit field 15..0 of register SSCRB (receive buffer containing the received data bits).

TX **Transmit interrupt**

This bit indicates that a transmit interrupt occured:

TX=’0’ No transmit interrupt is indicated by this vector.

TX=’1’ A transmit interrupt is indicated by this vector.
Bit field 15..0 contains a constant zero value. This interrupt means, that the transmit buffer can be reloaded with new transmit data (register SSCTB).

ISW(15:0) **Interrupt Status Word**

The contents of this bit field depends on the SSC interrupt type:

Type:	Meaning:
error int. (D/E=’0’)	ISW(15:0) = SSCCON(15:0) (Register SSCCON in operational mode.)
receive int. (D/E=’1’, R/T=’1’)	ISW(15:0) = SSCRB(15:0) (Register SSCRB.)
transm. int. (D/E=’1’, R/T=’0’)	ISW(15:0) = 0000_H

11.2.2.5 LBI Interrupt Vector

LBI interrupt vectors are transferred to the peripheral interrupt queue IQP.

Table 106 LBI Interrupt Vector

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

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

(Source ID) bit field 31..28 **1101_2** LBI Interrupt Vector (IQP)

This interrupt vector (with no additional bit information) is generated, if a state transition inactive to active is detected at LBI interrupt input signal LINTI1.
The polarity (high or low active) of input signal LINTI1 is determined by bit ’LINTIC’ in register LCONF.

11.2.2.6 GPP Interrupt Vector

GPP interrupt vectors are transferred to the peripheral interrupt queue IQP.

Table 107 GPP Interrupt Vectori

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPSTA(15:0)															

(Source ID) bit field 31..28 **1111_2** GPP Interrupt Vector (IQP)

GPSTA (15:0) **General Purpose Port Status Information**

This bit field 15..0 reflects the changes (high-to-low or low-to-high transition) detected on the GPP pins. The actual state (input level) of these pins can be determined by reading the general purpose data register GPDATA.
Whenever a transition on at least one general purpose pin is detected, a GPP interrupt vector is generated (if unmasked).

12 Test Configuration

12.1 JTAG Boundary Scan Interface

In the DSCC4 a Test Access Port (TAP) controller is implemented. The essential part of the TAP is a finite state machine (16 states) controlling the different operational modes of the boundary scan. Both, TAP controller and boundary scan, meet the requirements given by the JTAG standard: IEEE 1149.1. **Figure 82** gives an overview about the TAP controller.

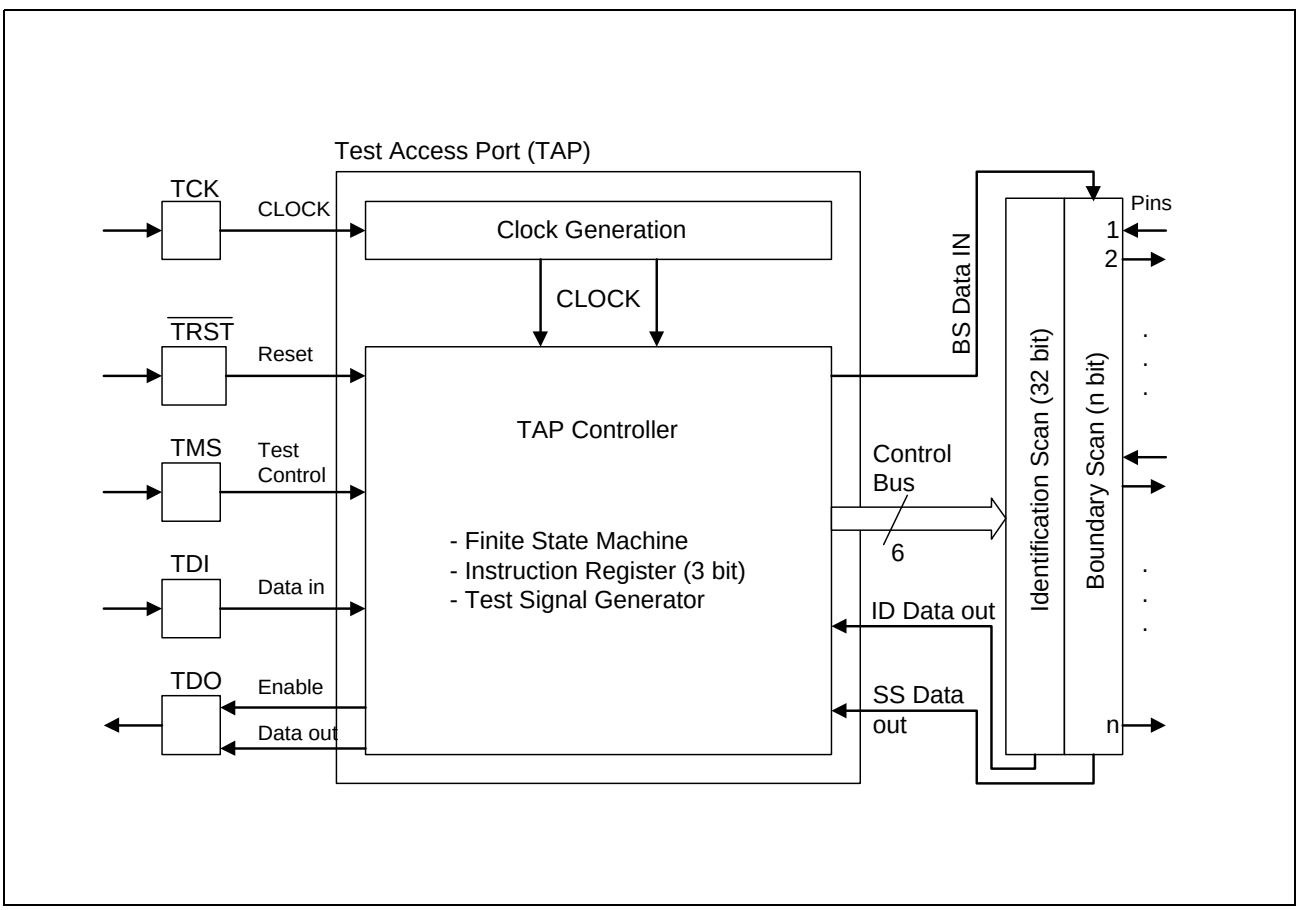


Figure 82 Block Diagram of Test Access Port and Boundary Scan Unit

If no boundary scan operation is planned $\overline{TRST}$ has to be connected with V_{SS}. TMS and TDI do not need to be connected since pull-up transistors ensure high input levels in this case. Nevertheless it would be a good practice to put the unused inputs to defined levels. In this case, if the JTAG is not used:
TMS = TCK = '1' is recommended.

Test handling (boundary scan operation) is performed via the pins TCK (Test Clock), TMS (Test Mode Select), TDI (Test Data Input) and TDO (Test Data Output) when the TAP controller is not in its reset state, i.e. $\overline{TRST}$ is connected to V_{DD} or it remains unconnected due to its internal pull-up. Test data at TDI are loaded with a 4-MHz clock

signal connected to TCK. ‘1’ or ‘0’ on TMS causes a transition from one controller state to another; constant ’1’ on TMS leads to normal operation of the chip.

Table 108 Boundary Scan Sequence of the DSCC4

TDI ->

Seq. No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
1	$\overline{\text{INTA}}$	I/O	3	**001**
2	AD0	I/O	3	**100**
3	AD1	I/O	3	**000**
4	AD2	I/O	3	**000**
5	AD3	I/O	3	**001**
6	AD4	I/O	3	**101**
7	AD5	I/O	3	**100**
8	AD6	I/O	3	**000**
9	C/$\overline{\text{BE0}}$	I/O	3	**100**
10	AD7	I/O	3	**000**
11	AD8	I/O	3	**11**0
12	AD9	I/O	3	000
13	AD10	I/O	3	000
14	AD11	I/O	3	000
15	AD12	I/O	3	000
16	AD13	I/O	3	000
17	AD14	I/O	3	000
18	AD15	I/O	3	000
19	C/$\overline{\text{BE1}}$	I/O	3	000
20	PAR	I/O	3	000
21	$\overline{\text{SERR}}$	I/O	3	000
22	$\overline{\text{PERR}}$	I/O	3	000
23	$\overline{\text{STOP}}$	I/O	3	000
24	$\overline{\text{DEVSEL}}$	I/O	3	000
25	$\overline{\text{TRDY}}$	I/O	3	000
26	$\overline{\text{IRDY}}$	I/O	3	000

Seq. No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
27	$\overline{\text{FRAME}}$	I/O	3	000
28	$\overline{\text{C/BE2}}$	I/O	3	000
29	AD16	I/O	3	000
30	AD17	I/O	3	000
31	AD18	I/O	3	000
32	AD19	I/O	3	000
33	AD20	I/O	3	000
34	AD21	I/O	3	000
35	AD22	I/O	3	000
36	AD23	I/O	3	000
37	IDSEL	I/O	3	000
38	C/$\overline{\text{BE3}}$	I/O	3	000
39	AD24	I/O	3	000
40	AD25	I/O	3	000
41	AD26	I/O	3	000
42	AD27	I/O	3	000
43	AD28	I/O	3	000
44	AD29	I/O	3	000
45	AD30	I/O	3	000
46	AD31	I/O	3	000
47	$\overline{\text{REQ}}$	I/O	3	000
48	$\overline{\text{GNT}}$	I/O	3	000
49	CLK	I	1	0
50	$\overline{\text{RST}}$	I/O	3	000
51	W/$\overline{\text{R}}$	I/O	3	000
52	DEMUX	I/O	3	000
53	$\overline{\text{RTS3}}$	I/O	3	000
54	CD3	I/O	3	000
55	$\overline{\text{CTS3}}$	I/O	3	000
56	TXD3	I/O	3	000

Seq. No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
57	RXD3	I/O	3	000
58	TXCLK3	I/O	3	000
59	RXCLK3	I/O	3	000
60	$\overline{\text{RTS2}}$	I/O	3	000
61	CD2	I/O	3	000
62	$\overline{\text{CTS2}}$	I/O	3	000
63	TXD2	I/O	3	000
64	RXD2	I/O	3	000
65	TXCLK2	I/O	3	000
66	RXCLK2	I/O	3	000
67	LALE	I/O	3	000
68	LINTO	I/O	3	000
69	LINTI2	I/O	3	000
70	LINTI1	I/O	3	000
71	$\overline{\text{LRDY}}$	I/O	3	000
72	$\overline{\text{LBHE}}$	I/O	3	000
73	$\overline{\text{LWR}}$	I/O	3	000
74	$\overline{\text{LRD}}$	I/O	3	000
75	$\overline{\text{LCSI}}$	I/O	3	000
76	$\overline{\text{LCSO}}$	I/O	3	000
77	$\overline{\text{LBREQ}}$	I/O	3	000
78	$\overline{\text{LHLDA}}$	I/O	3	000
79	$\overline{\text{LHOLD}}$	I/O	3	000
80	LD0	I/O	3	000
81	LD1	I/O	3	000
82	LD2	I/O	3	000
83	LD3	I/O	3	000
84	LD4	I/O	3	000
85	LD5	I/O	3	000
86	LD6	I/O	3	000

Seq. No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
87	LD7	I/O	3	000
88	LD8	I/O	3	000
89	LD9	I/O	3	000
90	LD10	I/O	3	000
91	LD11	I/O	3	000
92	LD12	I/O	3	000
93	LD13	I/O	3	000
94	LD14	I/O	3	000
95	LD15	I/O	3	000
96	LA0	I/O	3	000
97	LA1	I/O	3	000
98	LA2	I/O	3	000
99	LA3	I/O	3	000
100	LA4	I/O	3	000
101	LA5	I/O	3	000
102	LA6	I/O	3	000
103	LA7	I/O	3	000
104	LA8	I/O	3	000
105	LA9	I/O	3	000
106	LA10	I/O	3	000
107	LA11	I/O	3	000
108	LA12	I/O	3	000
109	LA13	I/O	3	000
110	LA14	I/O	3	000
111	LA15	I/O	3	000
112	RTS1	I/O	3	000
113	CD1	I/O	3	000
114	CTS1	I/O	3	000
115	TXD1	I/O	3	000
116	RXD1	I/O	3	000

Seq. No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
117	TXCLK1	I/O	3	000
118	RXCLK1	I/O	3	000
119	TEST	I	1	0
120	$\overline{\text{RTS0}}$	I/O	3	000
121	CD0	I/O	3	000
122	$\overline{\text{CTS0}}$	I/O	3	000
123	TXD0	I/O	3	000
124	RXD0	I/O	3	000
125	TXCLK0	I/O	3	000
126	RXCLK0	I/O	3	000

-> TDO

An input pin (I) uses one boundary scan cell (data in), an output pin (O) uses two cells (data out, enable) and an I/O-pin (I/O) uses three cells (data in, data out, enable). Note that some functional output and input pins of the DSCC4 are tested as I/O pins in boundary scan, hence using three cells. The boundary scan unit of the DSCC4 contains a total of n = 374 scan cells.

The right column of **Table 108** gives the initialization values of the cells.

The desired test mode is selected by serially loading a 3-bit instruction code into the instruction register via TDI (LSB first); see **Table 109.**

Table 109 Boundary Scan Test Modes

Instruction (Bit 2 … 0)	Test Mode
000	EXTEST (external testing)
001	INTEST (internal testing)
010	SAMPLE/PRELOAD (snap-shot testing)
011	IDCODE (reading ID code)
111	BYPASS (bypass operation)
others	handled like BYPASS

EXTEST is used to examine the interconnection of the devices on the board. In this test mode at first all input pins **capture** the current level on the corresponding external interconnection line, whereas all output pins are held at constant values (‘0’ or ‘1’,

according to **Table 108**). Then the contents of the boundary scan is **shifted** to TDO. At the same time the next scan vector is loaded from TDI. Subsequently all output pins are **updated** according to the new boundary scan contents and all input pins again capture the current external level afterwards, and so on.

INTEST supports internal testing of the chip, i.e. the output pins **capture** the current level on the corresponding internal line whereas all input pins are held on constant values (‘0’ or ‘1’, according to **Table 108**). The resulting boundary scan vector is **shifted** to TDO. The next test vector is serially loaded via TDI. Then all input pins are **updated** for the following test cycle.

Note: In capture IR-state the code ‘001’ is automatically loaded into the instruction register, i.e. if INTEST is wanted the shift IR-state does not need to be passed.

SAMPLE/PRELOAD is a test mode which provides a snap-shot of pin levels during normal operation.

IDCODE: A 32-bit identification register is serially read out via TDO. It contains the version number (4 bits), the device code (16 bits) and the manufacturer code (11 bits). The LSB is fixed to ‘1’.

TDI ->	0011	0000 0000 0011 0110	0000 1000 001	1	-> TDO

Note: Since in test logic reset state the code ‘011’ is automatically loaded into the instruction register, the ID code can easily be read out in shift DR state which is reached by TMS = 0, 1, 0, 0.

BYPASS: A bit entering TDI is shifted to TDO after one TCK clock cycle.

13 Electrical Characteristics

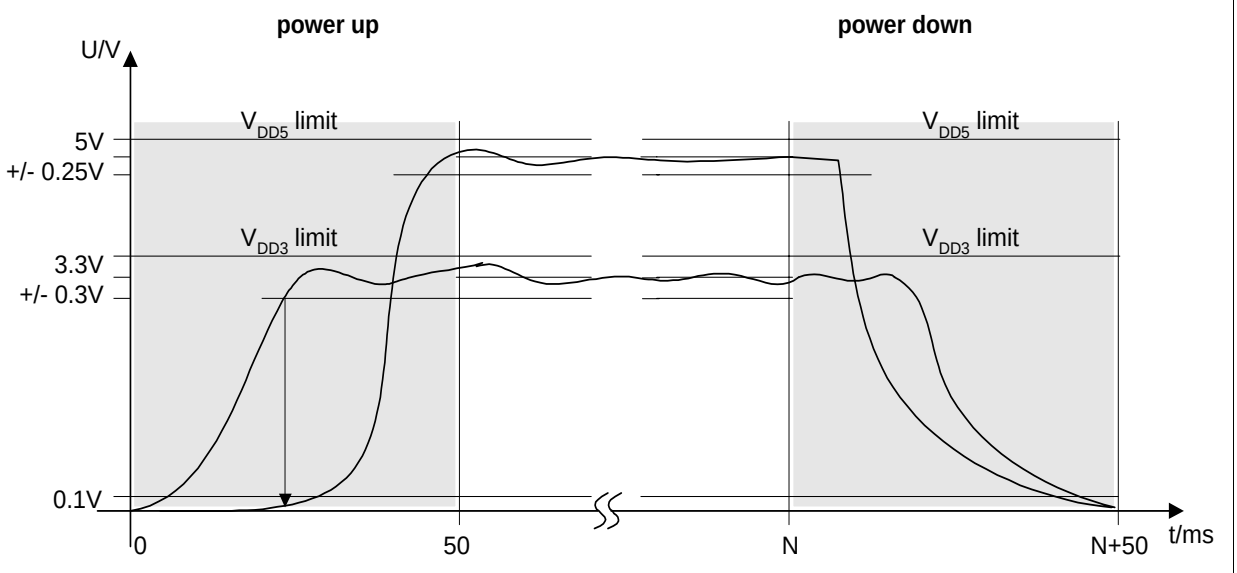
13.1 Important Electrical Requirements

V_{DD3} = 3.3 V ± 0.3 V **$V_{DD3\ max}$ = 3.6 V**

V_{DD5} = 5.0 V ± 0.25 V **$V_{DD5\ max}$ = 5.25 V**

During all DSCC4 power-up and power-down situations the difference $|V_{DD5} - V_{DD3}|$ may not exceed 3.6V. The absolute maximums of V_{DD5} and V_{DD3} should never be exceeded.

Figure 83 shows that both V_{DD3} and V_{DD5} can take on any time sequence, not exceeding a voltage difference of 3.6V, for up to 50 milliseconds at power-up and power-down.Within 50 milliseconds of power-up the voltages must be within their respective absolute voltage limits. At power-down, within 50 milliseconds of either voltage going outside its operational range, the voltage difference should not exceed 3.6V and both voltages must be returned below 0.1V:



Within the grey boxes any shape of V_{DD3} and V_{DD5} signal is allowed with the requirements that the absolute limits of each signal are not exceeded, the slew rate recommendation for V_{DD3} is met to guarantee proper boundary scan reset and the voltage difference does not exceed 3.6V.

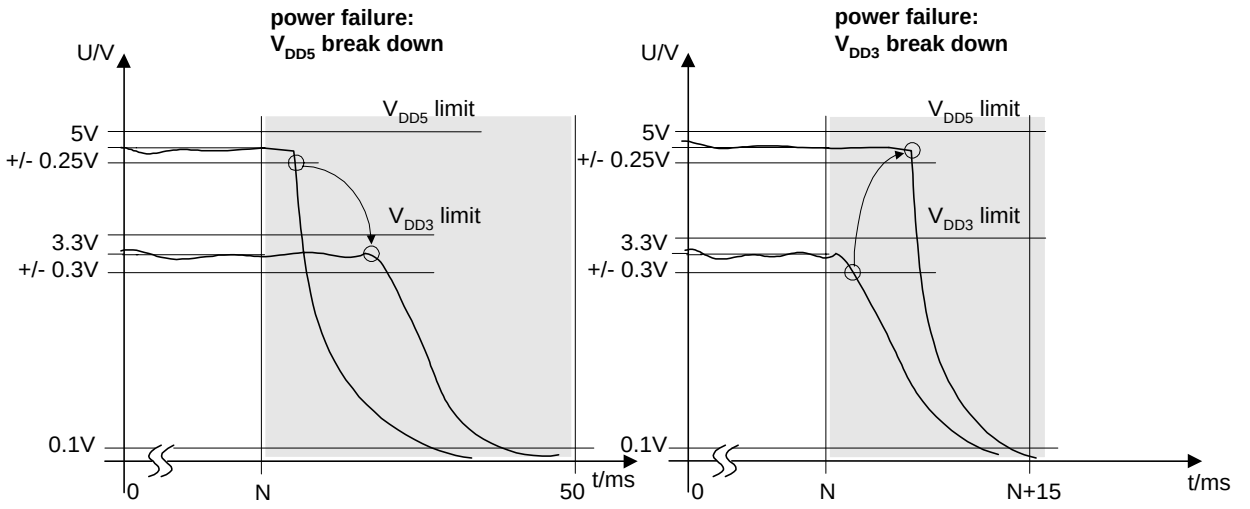
Outside the grey boxes the voltages provided to V_{DD3} and V_{DD5} should be inside the normal operation range.

In this power-up example V_{DD5} is enabled after V_{DD3} reached its minimum operation value which is a typical implementation.

For power-down V_{DD5} is switched off before V_{DD3}.

Figure 83 Power-up and Power-down scenarios

Similar criteria also apply to power down in case of power failure situations:



Within the grey boxes any shape of V_{DD3} and V_{DD5} signal is allowed with the requirements that the absolute limits of each signal are not exceeded and the specified voltage differences are not exceeded.

a. In case of V_{DD5} break-down the 3.6V difference is not exceeded anyway. The voltages must return below 0.1V within 50 milliseconds.
b. In case of V_{DD3} break-down the maximum voltage difference must not exceed 4.5 V for a maximum of 15 milliseconds.The voltages must return below 0.1V within 50 milliseconds.
This scenario is allowed for 2000 power failure cycles.

Figure 84 Power-Failure scenarios

Additional recommendations:

The pin TEST has to be tied to V_{SS} (refer to pin description table).

13.2 Absolute Maximum Ratings

Table 110 Absolute Maximum Ratings

Parameter	Symbol	Limit Values		Unit
		min.	max.	
Ambient temperature under bias	T_A	0	70	°C
Junction temperature under bias	T_J		125	°C
Storage temperature	T_{stg}	– 65	125	°C
Voltage at any pin with respect to ground	V_S	– 0.4	V_{DD5} + 0.4	V

Note: Stresses above those listed here may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

13.3 Thermal Package Characteristics

Table 111 Thermal Package Characteristics

Parameter		Symbol	Value	Unit
Thermal Package Resistance Junction to Ambient				
Airflow:	Ambient Temperature:			
without airflow	T_A=-40°C	$\theta_{JA(0,-40)}$	42.3	°C/K
without airflow	T_A=+25°C	$\theta_{JA(0,25)}$	37.2	°C/K
airflow 1 m/s	T_A=+25°C	$\theta_{JA(1,25)}$	34.9	°C/K
airflow 2 m/s	T_A=+25°C	$\theta_{JA(2,25)}$	33.3	°C/K
airflow 3 m/s	T_A=+25°C	$\theta_{JA(3,25)}$	32.2	°C/K

13.4 DC Characteristics

a) Non-PCI Interface Pins and Power Supply Pins

Table 112 DC Characteristics (Non-PCI Interface Pins and Power Supply Pins)

T_A = 0 to + 70 °C; V_{DD5} = 5 V ± 5 %, V_{DD3} = 3.3 V ± 0.3 V, V_{SS} = 0 V

Parameter		Symbol	Limit Values		Unit	Test Condition
			min.	max.		
L-input voltage		V_{IL}	– 0.4	0.8	V	
H-input voltage		V_{IH}	2.0	V_{DD5} + 0.4	V	
L-output voltage		V_{QL}		0.45	V	I_{QL} = 7 mA (pin TXD) I_{QL} = 2 mA (all others / non-PCI)
H-output voltage		V_{QH}	2.4	V_{DD3}	V	I_{QH} = – 400 µA
Power supply current V_{DD3}	operational	I_{CC3}		< 300	mA	inputs at V_{SS}/V_{DD}, no output loads
	power down (no clocks)	I_{CC3}		< 2	mA	
Power supply current V_{DD5}		I_{CC5}		< 2	mA	
Power dissipation		P		< 1090	mW	
typical values	operational current	$I_{CC3\ typ.}$		< 250	mA	V_{DD3} = 3.3V, inputs at V_{SS}/V_{DD}, no output loads
	power dissipation	$P_{typ.}$		< 830	mW	
Input leakage current Output leakage current		I_{LI} I_{LQ}		1	µA	0 V < V_{IN} < V_{DD} to 0 V 0 V < V_{OUT} < V_{DD} to 0 V (pins with internal pull-ups excluded)

Note: 1. The listed characteristics are ensured over the operating range of the integrated circuit. Typical characteristics specify mean values expected over the production spread. If not otherwise specified, typical characteristics apply at T_A = 25 °C and the given supply voltage.

Note: 2. The electrical characteristics described in ***Section 13.2*** *also apply here!*

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995
(Chapter 4.2.1: Electrical DC Specifications for 5 V signaling)

.

Table 113 DC Characteristics PCI Interface Pins

T_A = 0 to + 70 °C; V_{DD5} = 5 V ± 5 %, V_{DD3} = 3.3 V ± 0.3 V, V_{SS} = 0 V

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
L-input voltage	V_{IL}	– 0.5	0.8	V	
H-input voltage	V_{IH}	2.0	V_{DD5} + 0.5	V	
L-output voltage	V_{QL}		0.45	V	I_{QL} = 3mA
H-output voltage	V_{QH}	2.4	V_{DD3}	V	I_{QH} = – 2 mA

13.5 Capacitances

a) Non-PCI Interface Pins

Table 114 Capacitances (Non-PCI Interface Pins)

T_A = 25 °C; V_{DD5} = 5 V ± 5%, V_{DD3} = 3.3 V ± 0.3 V, V_{SS} = 0 V

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Input capacitance	C_{IN}	1	5	pF	
Output capacitance	C_{OUT}	5	10	pF	
I/O-capacitance	C_{IO}	6	15	pF	

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995 (Chapter 4: Electrical Specification for 5 V signalling)

13.6 AC Characteristics

a) Non-PCI Interface Pins

T_A = 0 to + 70 °C; V_{DD5} = 5 V ± 5%; V_{DD3} = 3.3 V ± 0.3 V

Inputs are driven to 2.4 V for a logical “1” and to 0.4 V for a logical “0”. Timing measurements are made at 2.0 V for a logical “1” and at 0.8 V for a logical “0”.

The AC testing input/output waveforms are shown below.

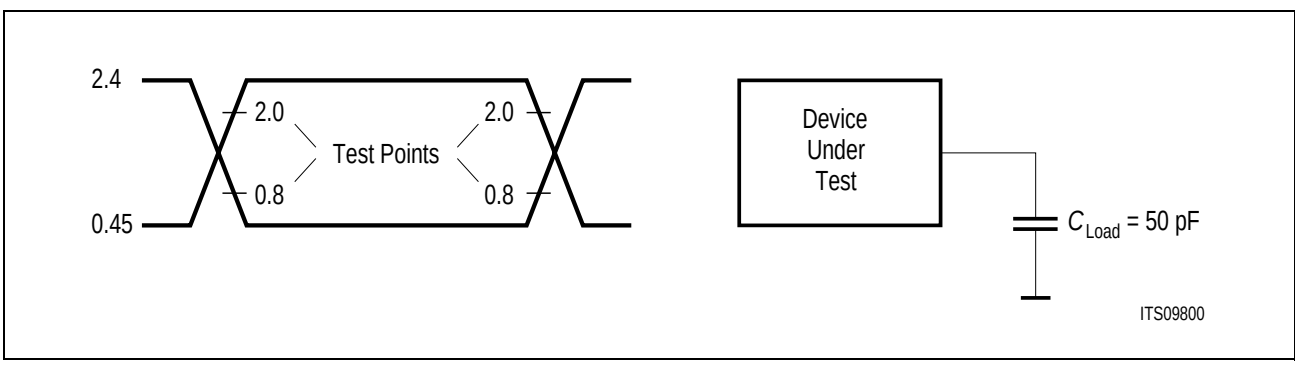


Figure 85 Input/Output Waveform for AC Tests

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995 (Chapter 4: Electrical Specification for 5 V signalling)

13.6.1 PCI Bus Interface Timing

The AC testing input/output waveforms are shown in **Figure 86** and **Figure 87** below.

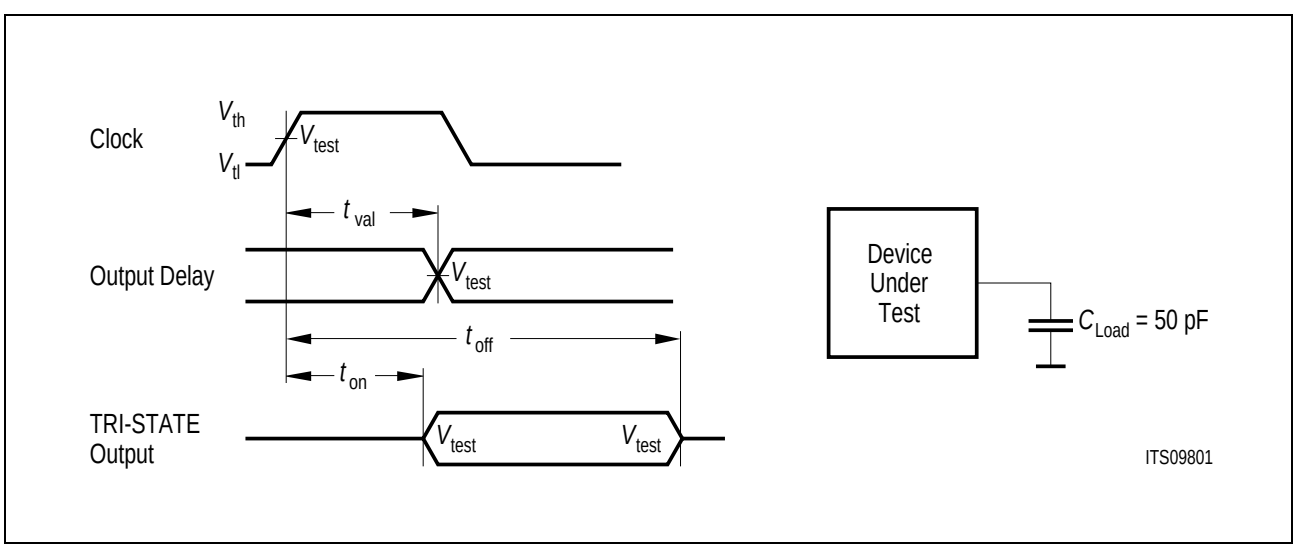


Figure 86 PCI Output Timing Measurement Waveforms

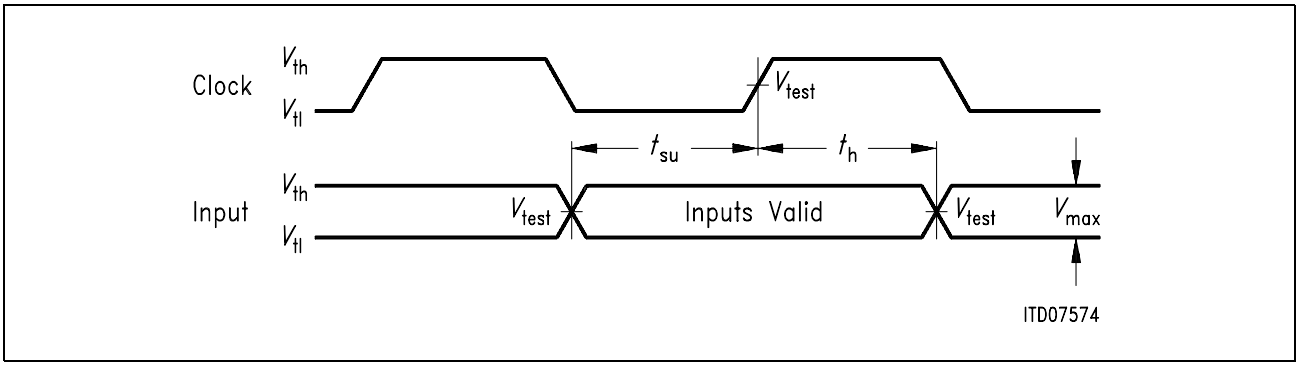


Figure 87 PCI Input Timing Measurement Waveforms

Table 115 PCI Input and Output Measurement Conditions

Symbol	Value	Unit
V_{th}	2.4	V
V_{tl}	0.4	V
V_{test}	1.5	V
V_{max}	2.0	V

The timings below show the basic read and write transaction between an initiator (Master) and a target (Slave) device. The DSCC4 is able to work both as master and slave.

As a master the DSCC4 reads/writes data from/to host memory using DMA and burst. The slave mode is used by an CPU to access the DSCC4 PCI Configuration Space, the on-chip registers and to access peripherals connected to the DSCC4 Local Bus Interface (LBI).

13.6.1.1 PCI Read Transaction

The transaction starts with an address phase which occurs during the first cycle when $\overline{\text{FRAME}}$ is activated (clock 2 in **Figure 88**). During this phase the bus master (initiator) outputs a valid address on AD(31:0) and a valid bus command on $\overline{\text{C/BE}}$(3:0). The first clock of the first data phase is clock 3. During the data phase $\overline{\text{C/BE}}$ indicate which byte lanes on AD(31:0) are involved in the current data phase.

The first data phase on a read transaction requires a turn-around cycle. In **Figure 88** the address is valid on clock 2 and then the master stops driving AD. The target drives the AD lines following the turnaround when $\overline{\text{DEVSEL}}$ is asserted. ($\overline{\text{TRDY}}$ cannot be driven until $\overline{\text{DEVSEL}}$ is asserted.) The earliest the target can provide valid data is clock 4. Once enabled, the AD output buffers of the target stay enabled through the end of the transaction.

A data phase may consist of a data transfer and wait cycles. A data phase completes when data is transferred, which occurs when both IRDY and TRDY are asserted. When either is deasserted a wait cycle is inserted. In the example below, data is successfully transferred on clocks 4, 6 and 8, and wait cycles are inserted on clocks 3, 5 and 7. The first data phase completes in the minimum time for a read transaction. The second data phase is extended on clock 5 because TRDY is deasserted. The last data phase is extended because IRDY is deasserted on clock 7.

The Master knows at clock 7 that the next data phase is the last. However, the master is not ready to complete the last transfer, so IRDY is deasserted on clock 7, and FRAME stays asserted. Only when IRDY is asserted can FRAME be deasserted, which occurs on clock 8.

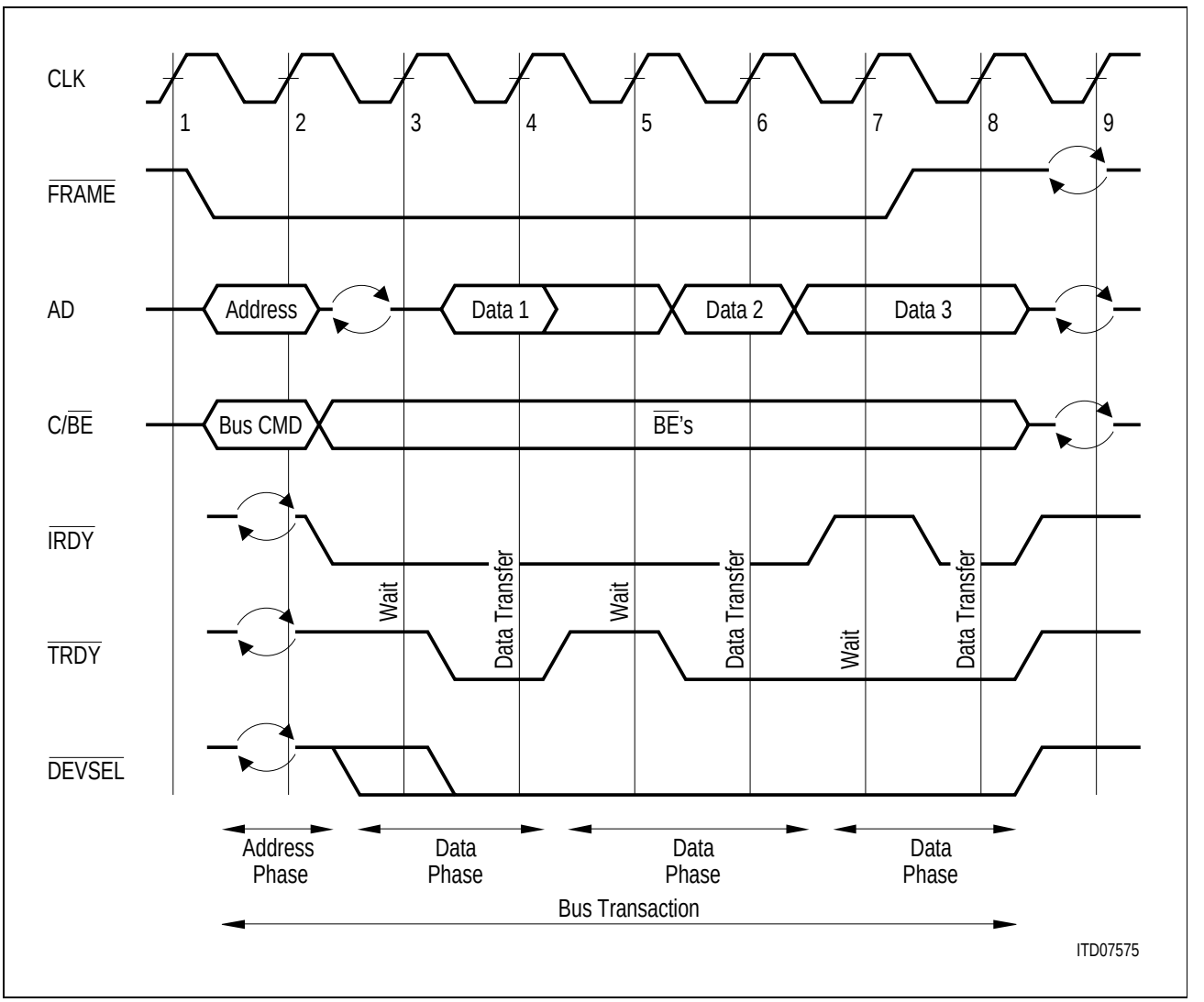


Figure 88 PCI Read Transaction

13.6.1.2 PCI Write Transaction

The transaction starts when FRAME is activated (clock 2 in **Figure 89**). A write transaction is similar to a read transaction except no turnaround cycle is required following the address phase. In the example, the first and second data phases complete with zero wait cycles. The third data phase has three wait cycles inserted by the target. Both initiator and target insert a wait cycle on clock 5. In the case where the initiator inserts a wait cycle (clock 5), the data is held on the bus, but the byte enables are withdrawn. The last data phase is characterized by IRDY being asserted while the FRAME signal is deasserted. This data phase is completed when TRDY goes active (clock 8).

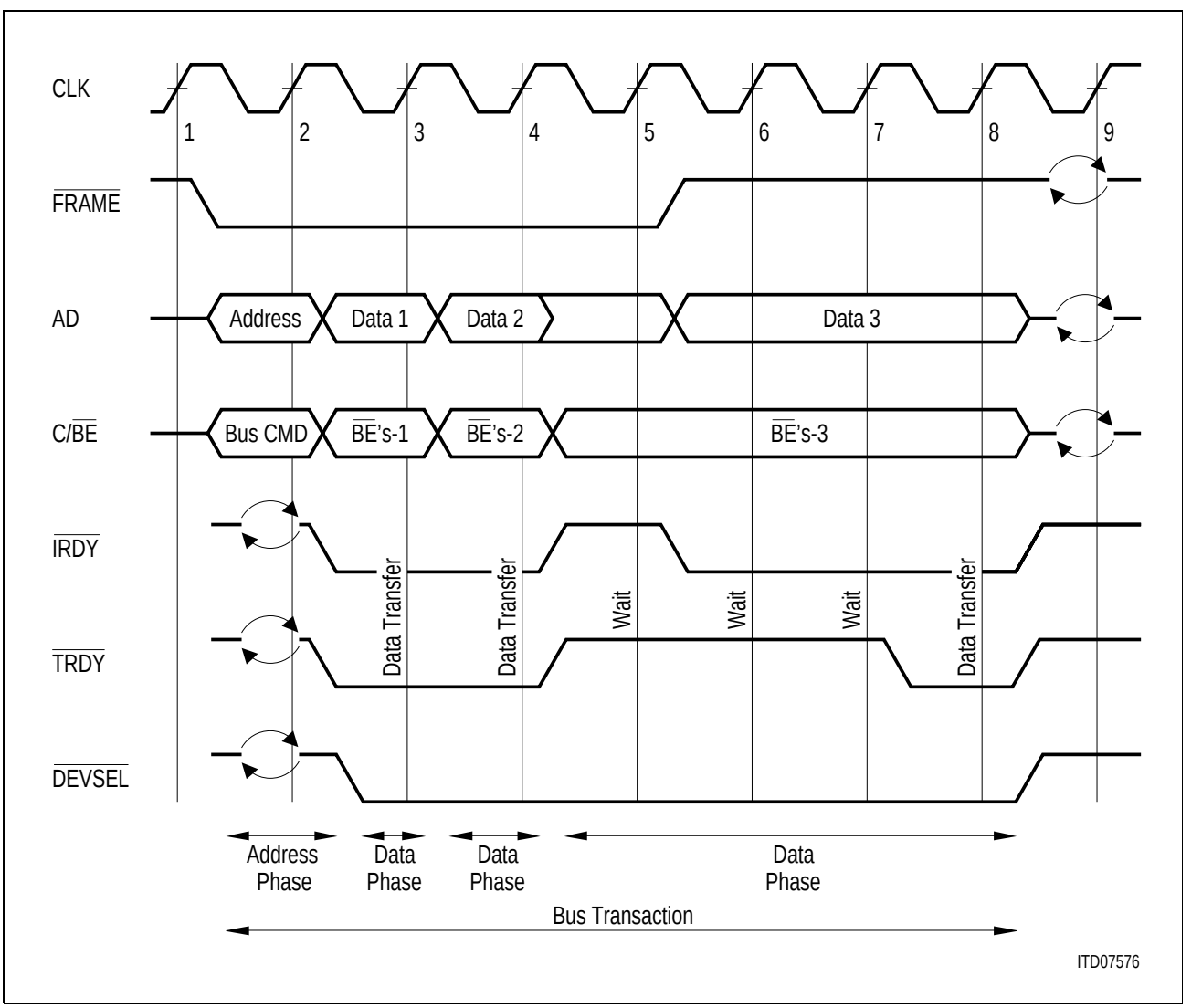


Figure 89 PCI Write Transaction

13.6.1.3 PCI Timing Characteristics

When the DSCC4 operates as a PCI Master (initiator) and it either reads or writes a burst – as controlled by the on-chip DMA controller – it does not deactivate $\overline{\text{IRDY}}$ between consecutive data. In other words, no wait states are inserted by the DSCC4 as a transaction initiator. The numbers of wait states, inserted by the DSCC4 as initiator are listed in **Table 116**.

Table 116 Number of Wait States Inserted by the DSCC4 as Initiator

Transaction	Number of Wait States	
	1st Data Cycle	2nd and Subsequent Data Cycles
Memory read burst	0	0
Memory write burst	0	0
Fast Back-to-back burst; 1st transaction	0	0
Fast Back-to-back burst; 2nd and subsequent transactions	1	0

When the DSCC4 operates as a PCI Slave (target), it inserts wait cycles by deactivating $\overline{\text{TRDY}}$. The numbers of wait states, typically inserted by the DSCC4 are listed in **Table 116:**

Table 117 Number of Wait States Inserted by the DSCC4 as Slave

Transaction	Number of Wait States
Configuration read	2
Configuration write	0
Register read	3
Register write	0
LBI read	3
LBI write	0

The number of wait states inserted by the DSCC4 as target is not critical because accesses to/via the DSCC4 are usually kept to a minimum in a system.

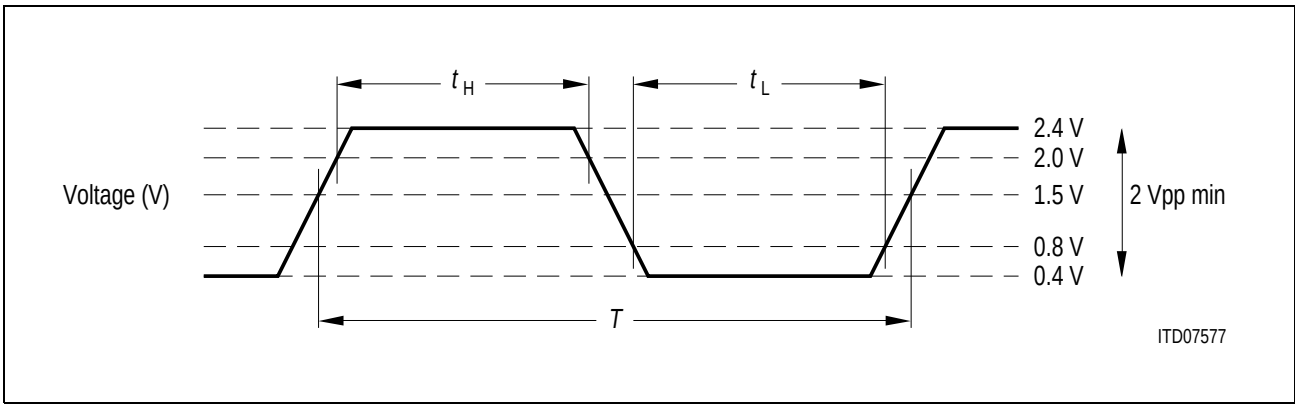


Figure 90 PCI Clock Specification

Table 118 PCI Clock Characteristics

Parameter	Symbol	Limit Values			Unit
		min.	typ.	max.	
CLK cycle time	T	30			ns
CLK high time	t_H	11			ns
CLK low time	t_L	11			ns
CLK slew rate (see note)		1		4	V/ns

*Note: Rise and fall times are specified in terms of the edge rate measured in V/ns. This slew rate must be met across the minimum peak-to-peak portion of the clock waveform as shown in **Figure 90**.*

Table 119 PCI Interface Signal Characteristics

Parameter	Limit Values			Unit	Remarks
	min.	typ.	max.		
CLK to signal valid delay bussed signals	(2)		11	ns	Notes 1, 2
CLK to signal valid delay point-to-point	(2)		12	ns	Notes 1, 2
Float to active delay	2		(3)	ns	
Active to float delay			20	ns	
Input setup time to CLK bussed signals	7			ns	Note 2
Input setup time to CLK point-to-point	10			ns	Note 2
Input hold time from CLK	0			ns	

Note 1Minimum times are measured with 0 pF equivalent load; maximum times are measured with 50 pF equivalent load.

Note 2$\overline{REQ}$ and $\overline{GNT}$ are point-to-point signals. All other signals are bussed $\overline{GNT}$ setup (min) time: 10ns

13.6.2 De-multiplexed Bus Interface

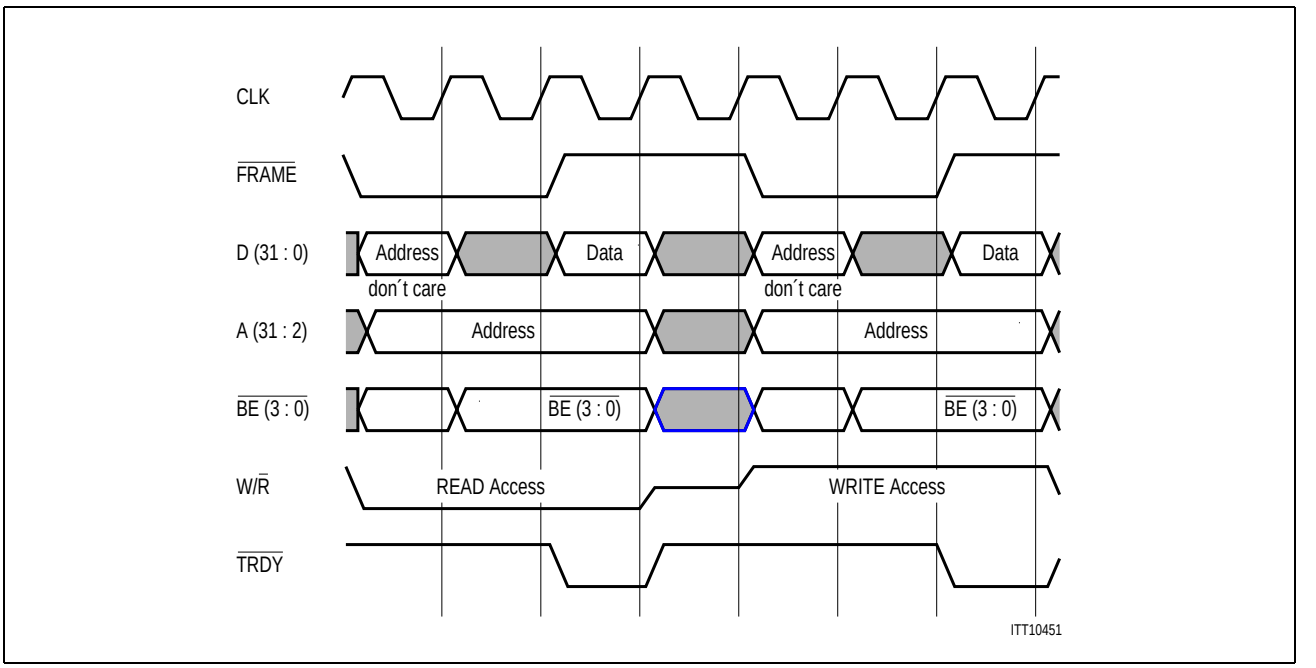


Figure 91 Master Single READ Transaction followed by a Master Single WRITE Transaction in De-multiplexed Bus Configuration

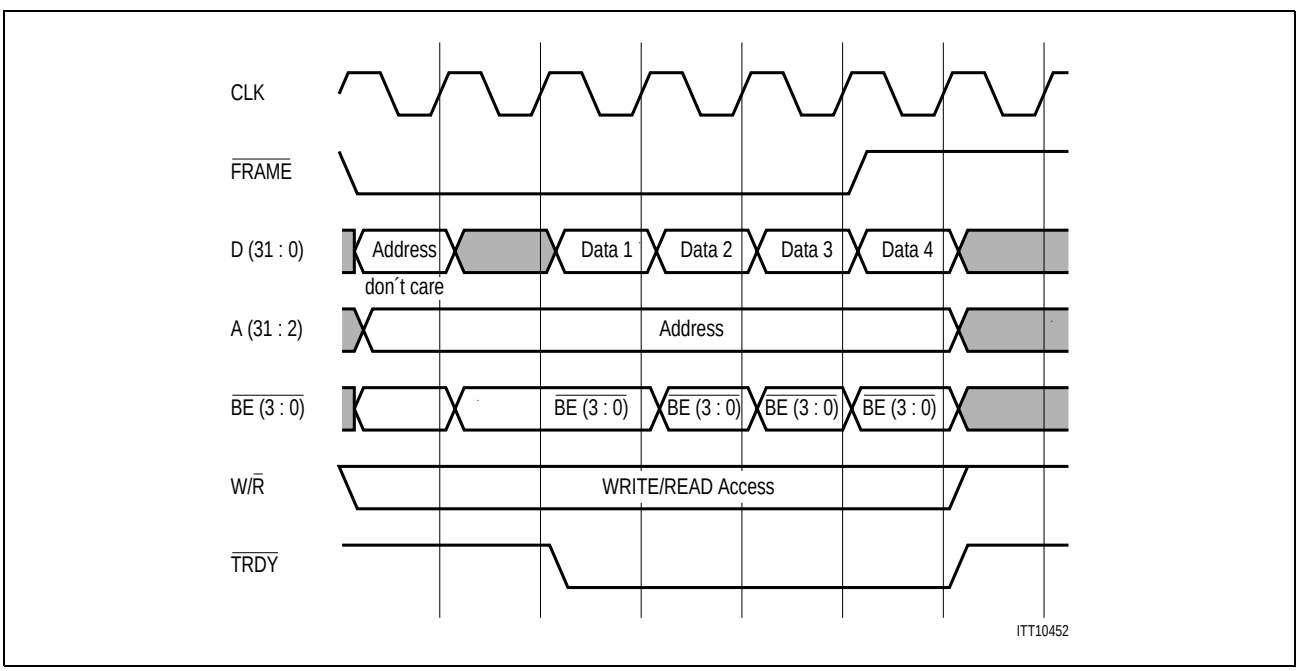


Figure 92 Master Burst WRITE/READ Access in De-multiplexed Bus Configuration

The timing provided in **Table 118** and **Table 119** can also be applied to the de-multiplexed bus interface.

Table 120 Additional De-multiplexed Interface Signal Characteristics

Parameter	Limit Values			Unit	Remarks
	min.	typ.	max.		
CLK to address bus signal valid delay			12	ns	
CLK to W/$\overline{R}$ signal valid delay			12	ns	
Address bus Input setup time to CLK	8			ns	
Address bus Input hold time to CLK	0			ns	
W/$\overline{R}$ signal Input setup time to CLK	8			ns	
W/$\overline{R}$ signal Input hold time to CLK	0			ns	

Note: The PCI parity signal PAR is not generated in de-multiplexed mode. It is driven active low by the DSCC4.

13.6.3 Local Bus Interface Timing

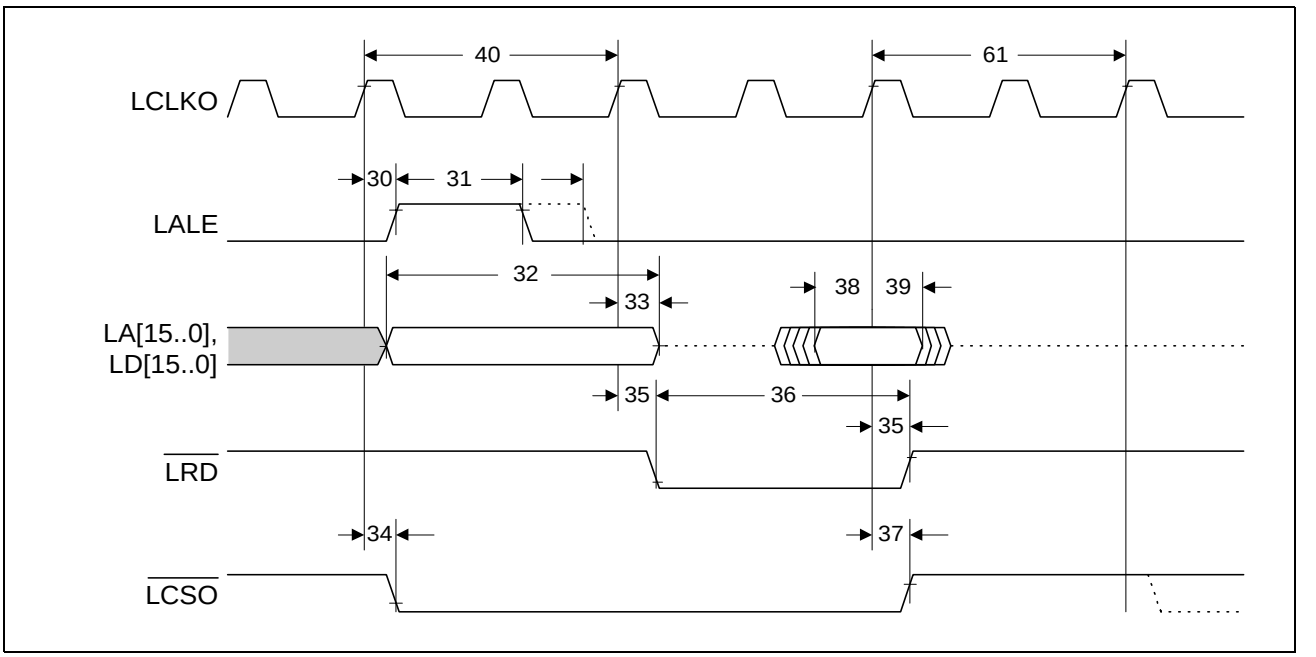


Figure 93 Synchronous LBI Read Cycle Timing Multiplexed Bus

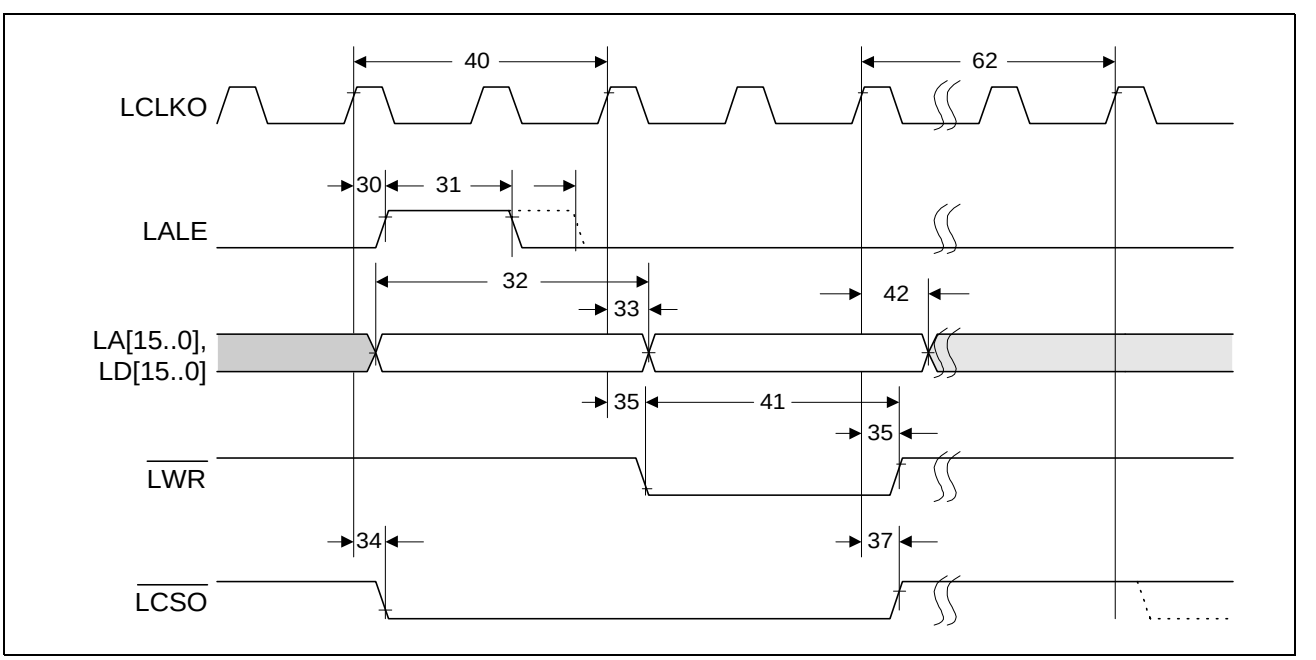


Figure 94 Synchronous LBI Write Cycle Timing Multiplexed Bus

Table 121 LBI Timing (synchronous, multiplexed bus)

No.	Parameter	Limit Values		Unit
		min.	max.	
30	LALE to LCLKO delay	5	20	ns
31	LALE pulse width	1 T_{LBICLK} (1.25 T_{LBICLK})[1]		(ns)
32	Address phase width	2 T_{LBICLK}		(ns)
33	LA, $\overline{LBHE}$ hold to LCLKO delay	5	20	ns
34	$\overline{LCSO}$ active to LCLKO delay	5	20	ns
35	$\overline{LRD}$, $\overline{LWR}$ active/inactive to LCLKO delay	5	20	ns
36	$\overline{LRD}$ pulse width	$2T_{LBICLK}$ +MCTC+T_{LRDY}		(ns)
37	$\overline{LCSO}$ inactive to LCLKO delay	5	20	ns
38	LD to LCLKO setup time	0		ns
39	LD to LCLKO hold time	25		ns
40	$\overline{LRD}$, $\overline{LWR}$ active to cycle start delay	2 T_{LBICLK}		(ns)
61	cycle end to next cycle start delay	2 T_{LBICLK}		(ns)
41	$\overline{LWR}$ pulse width	$2T_{LBICLK}$ +MCTC+T_{LRDY}		(ns)
42	LD to LCLKO hold delay	1 T_{PCICLK} + 5	1 T_{PCICLK} + 20	ns
	LD to $\overline{LWR}$ inactive delay	1 T_{PCICLK}		(ns)
	LD tristate delay to LCLKO		1 T_{PCICLK} + 20	ns
62	cycle end to next cycle start delay	5 T_{LBICLK}		(ns)

[1] If extended LALE timing is selected via bit 'EALE' in register LCONF

Note: T_{LBICLK} is the LBI clock time period which depends on the LBI clock division factor. MCTC is the number of master clock wait states (in LBI clock cycles) selected in register LCONF.
$T_{\overline{LRDY}}$ is the number of additional wait states (in LBI clock cycles) introduced by $\overline{LRDY}$ control signal if enabled via bit 'RDEN' in register LCONF.
T_{PCICLK} is the PCI clock time period.

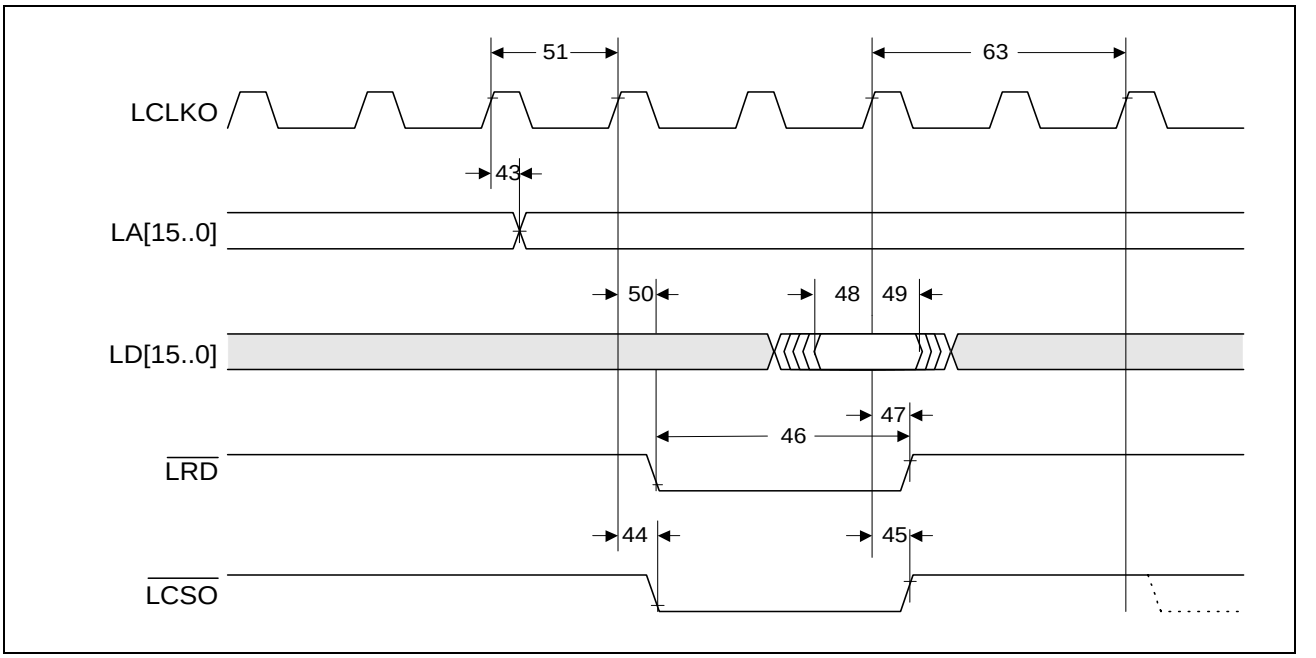


Figure 95 Synchronous LBI Read Cycle Timing De-multiplexed Bus

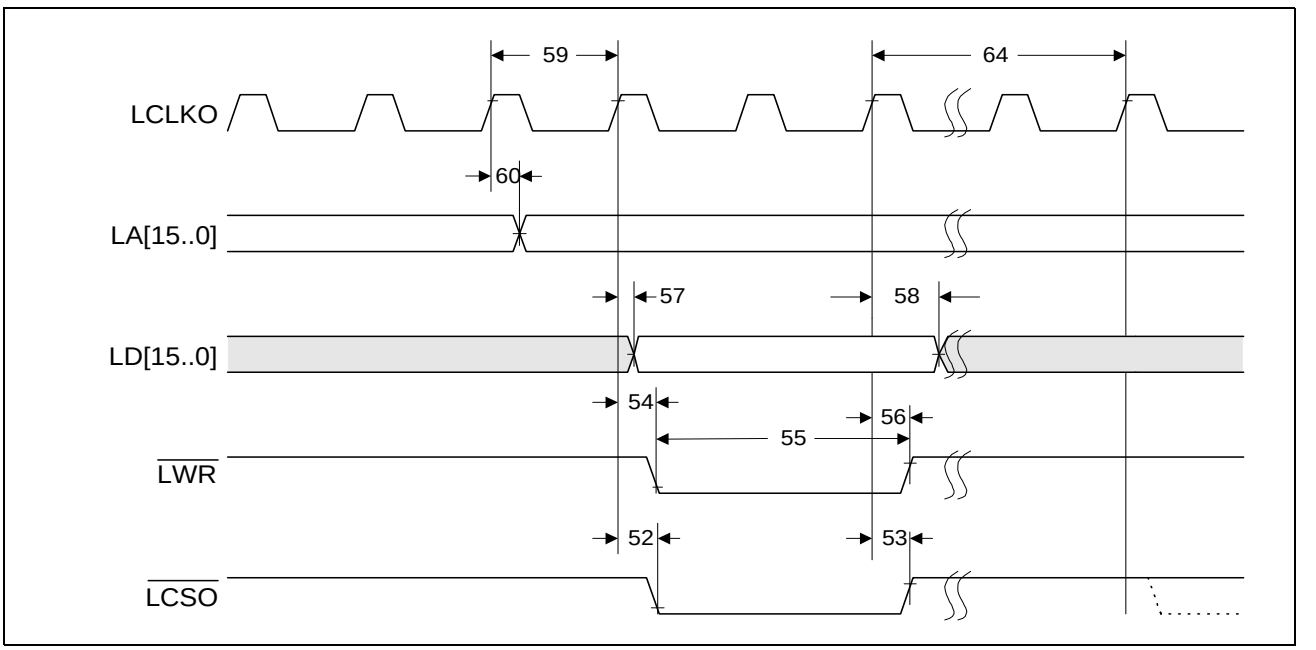
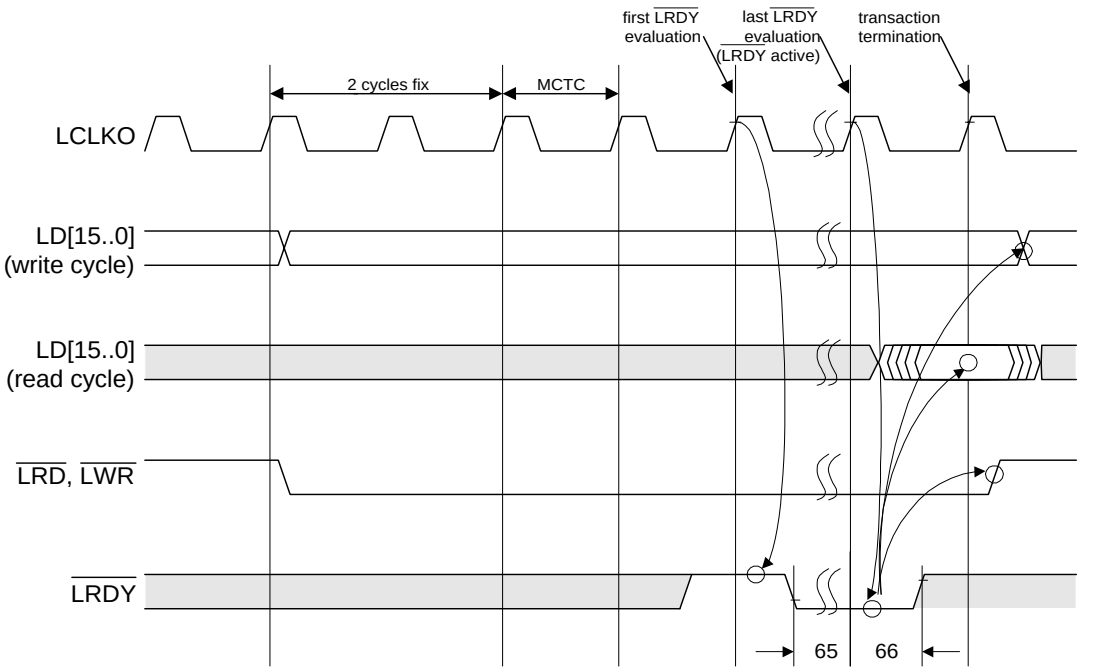


Figure 96 Synchronous LBI Write Cycle Timing De-multiplexed Bus

Table 122 LBI Timing (synchronous, de-multiplexed bus)

No.	Parameter	Limit Values		Unit
		min.	max.	
43	LA to LCLKO delay	5	20	ns
44	$\overline{\text{LCSO}}$ active to LCLKO delay	5	20	ns
45	$\overline{\text{LCSO}}$ inactive to LCLKO delay	5	20	ns
46	$\overline{\text{LRD}}$ pulse width	$2T_{LBICLK}$ +MCTC+T_{LRDY}		(ns)
47	$\overline{\text{LRD}}$, $\overline{\text{LWR}}$ inactive to LCLKO delay	5	20	ns
48	LD to LCLKO setup time		0	ns
49	LD to LCLKO hold time		25	ns
50	$\overline{\text{LRD}}$, $\overline{\text{LWR}}$ active to LCLKO delay	5	20	ns
51	$\overline{\text{LRD}}$ active to cycle start delay	1 T_{LBICLK}		(ns)
63	cycle end to next cycle start delay	2 T_{LBICLK}		ns
52	$\overline{\text{LCSO}}$ active to LCLKO delay	5	20	ns
53	$\overline{\text{LCSO}}$ inactive to LCLKO delay	5	20	ns
54	$\overline{\text{LWR}}$ active to LCLKO delay	5	20	ns
55	$\overline{\text{LWR}}$ pulse width	$2T_{LBICLK}$ +MCTC+T_{LRDY}		(ns)
56	$\overline{\text{LRD}}$, $\overline{\text{LWR}}$ inactive to LCLKO delay	5	20	ns
57	LD valid after LCLKO delay	5	20	ns
58	LD hold after LCLKO delay	1 T_{PCICLK} + 5	1 T_{PCICLK} + 20	ns
	LD to $\overline{\text{LWR}}$ inactive delay	1 T_{PCICLK}		(ns)
	LD tristate delay after LCLKO		1 T_{PCICLK} + 20	ns
59	$\overline{\text{LWR}}$ active to cycle start delay	1 T_{LBICLK}		(ns)
60	LA to LCLKO delay	5	20	ns
64	cycle end to next cycle start delay	5 T_{LBICLK}		(ns)

Note: T_{LBICLK} is the LBI clock time period which depends on the LBI clock division factor. MCTC is the number of master clock wait states (in LBI clock cycles) selected in register LCONF.
T_{LRDY} is the number of additional wait states (in LBI clock cycles) introduced by $\overline{\text{LRDY}}$ control signal if enabled via bit 'RDEN' in register LCONF.
T_{PCICLK} is the PCI clock time period.



Notes:
- MCTC wait state configuration is assumed to one cycle in this figure
- LRDY is evaluated the first time with the clock cycle following the MCTC related wait states
- Transaction is terminated one clock cycle after detecting LRDY active
- LRDY is evaluated only if LRDY-control is enabled

Figure 97 LRDY Timing

Table 123 LBI LRDY Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
65	LRDY to LCLKO setup time		0	ns
66	LRDY to LCLKO hold time		25	ns

13.6.4 Local Bus Interface Arbitration Timing

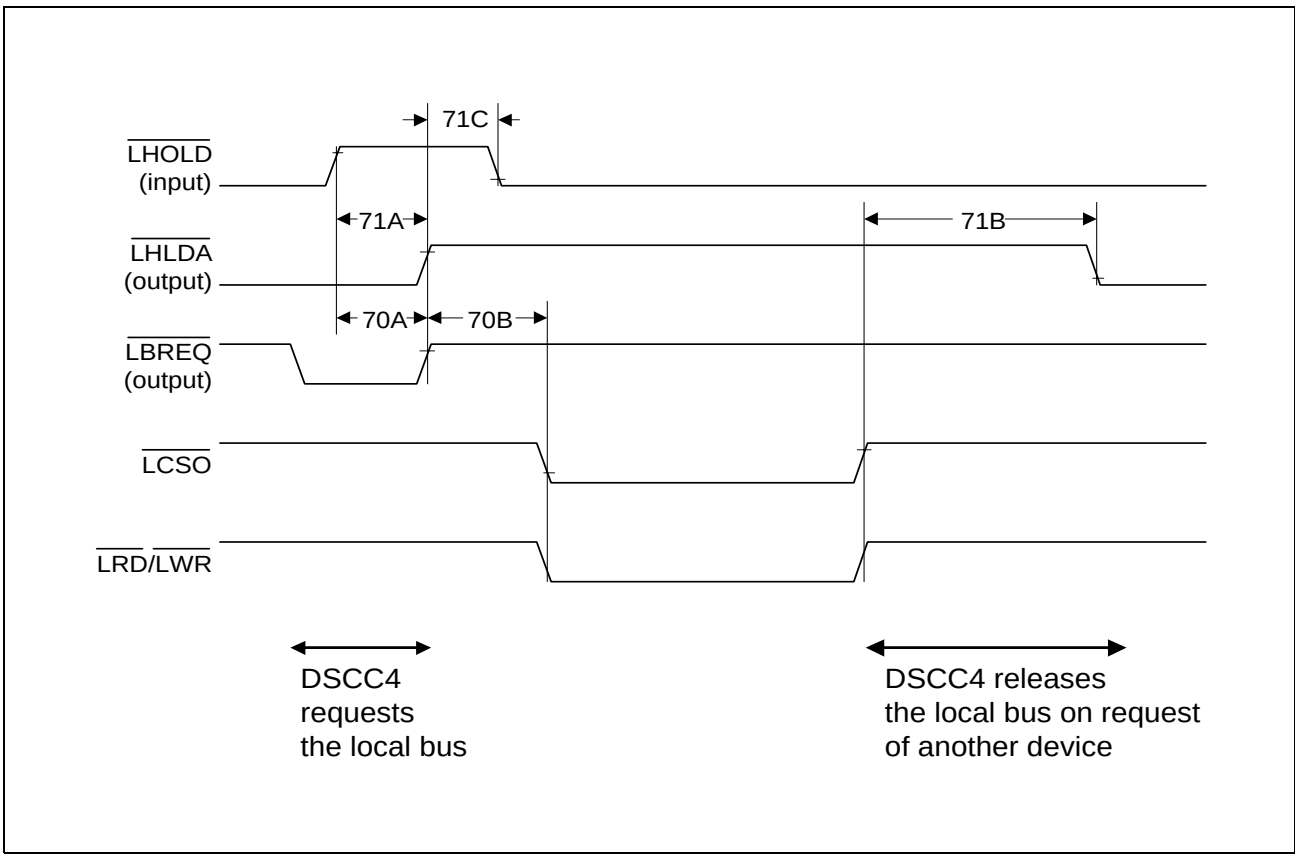


Figure 98 LBI Arbitration Timing

Table 124 LBI Arbitration Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
70A	$\overline{\text{LHOLD}}$ inactive to $\overline{\text{LBREQ}}$ inactive delay	30		ns
70B	First master cycle start to $\overline{\text{LHOLDA}}$ inactive delay	120		ns
71A	$\overline{\text{LHOLD}}$ inactive to $\overline{\text{LHOLDA}}$ inactive delay	30		ns
71B	Last master cycle terminated to $\overline{\text{LHLDA}}$ active again	90		ns
71C	LBREQ active again to $\overline{\text{LHLDA}}$ inactive	60		ns

13.6.5 PCM Serial Interface Timing

13.6.5.1 Clock Input Timing

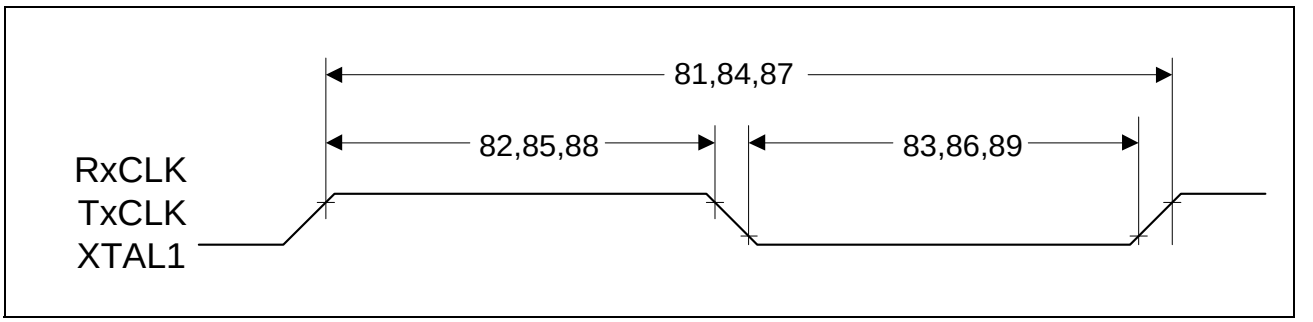


Figure 99 Clock Input Timing

Table 125 Clock Input Timing (non high speed modes)

No.	Parameter	Limit Values		Unit
		min.	max.	
81	RxCLK clock period	30		ns
82	RxCLK high time	13		ns
83	RxCLK low time	13		ns
84	TxCLK clock period	30		ns
85	TxCLK high time	13		ns
86	TxCLK low time	13		ns
87	XTAL1 clock period (internal oscillator used)	50		ns
	XTAL1 clock period (TTL clock signal supplied)	30		ns
88	XTAL1 high time (internal oscillator used)	23		ns
	XTAL1 high time (TTL clock signal supplied)	13		ns
89	XTAL1 low time (internal oscillator used)	23		ns
	XTAL1 low time (TTL clock signal supplied)	13		ns

Table 126 Clock Input Timing (high speed mode)

No.	Parameter	Limit Values		Unit
		min.	max.	
81	RxCLK clock period	19.2		ns
82	RxCLK high time	8.6		ns
83	RxCLK low time	8.6		ns
84	TxCLK clock period	19.2		ns
85	TxCLK high time	8.6		ns
86	TxCLK low time	8.6		ns

13.6.5.2 Receive Cycle Timing

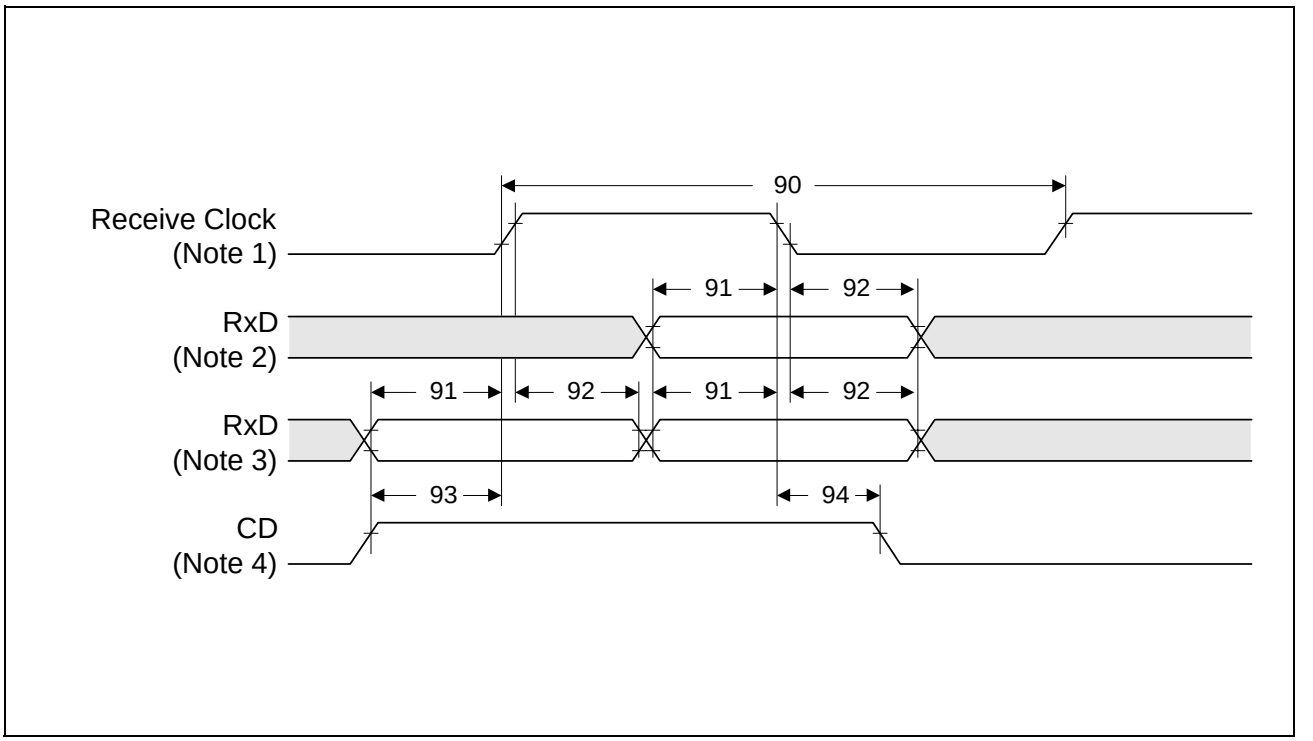


Figure 100 Receive Cycle Timing

Note:

1. Whichever supplies the receive clock depending on the selected clock mode:
   externally clocked via RxCLK or XTAL1 or
   internally clocked via DPLL, BRG or BCR.
   (No edge relation can be measured if the internal receive clock is derived from the external clock source by devision stages (BGR, BCR) or DPLL)
2. NRZ, NRZI and Manchester data encoding
3. FM0 and FM1 data encoding
4. If Carrier Detect auto start feature enabled (not for clock modes 1 and 5)
5. A receive clock must be present to detect input level changes of signal CD.

Table 127 Receive Cycle Timing

No.	Parameter		Limit Values		Unit
			min.	max.	
Receive data rates		externally clocked (HDLC, high speed mode)		52	MBit/s
		externally clocked (non high speed)		10	MBit/s
		internally clocked (DPLL modes)		2	MBit/s
		internally clocked (non DPLL modes)		2	MBit/s
90	Clock period	externally clocked	19.2		ns
		internally clocked (DPLL modes)	480		ns
		internally clocked (non DPLL modes)	480		ns
91	RxD to RxCLK setup time		5		ns
92	RxD to RxCLK hold time		15		ns
93	CD to RxCLK rising edge setup time		10		ns
94	CD to RxCLK falling edge hold time		10		ns

13.6.5.3 Transmit Cycle Timing

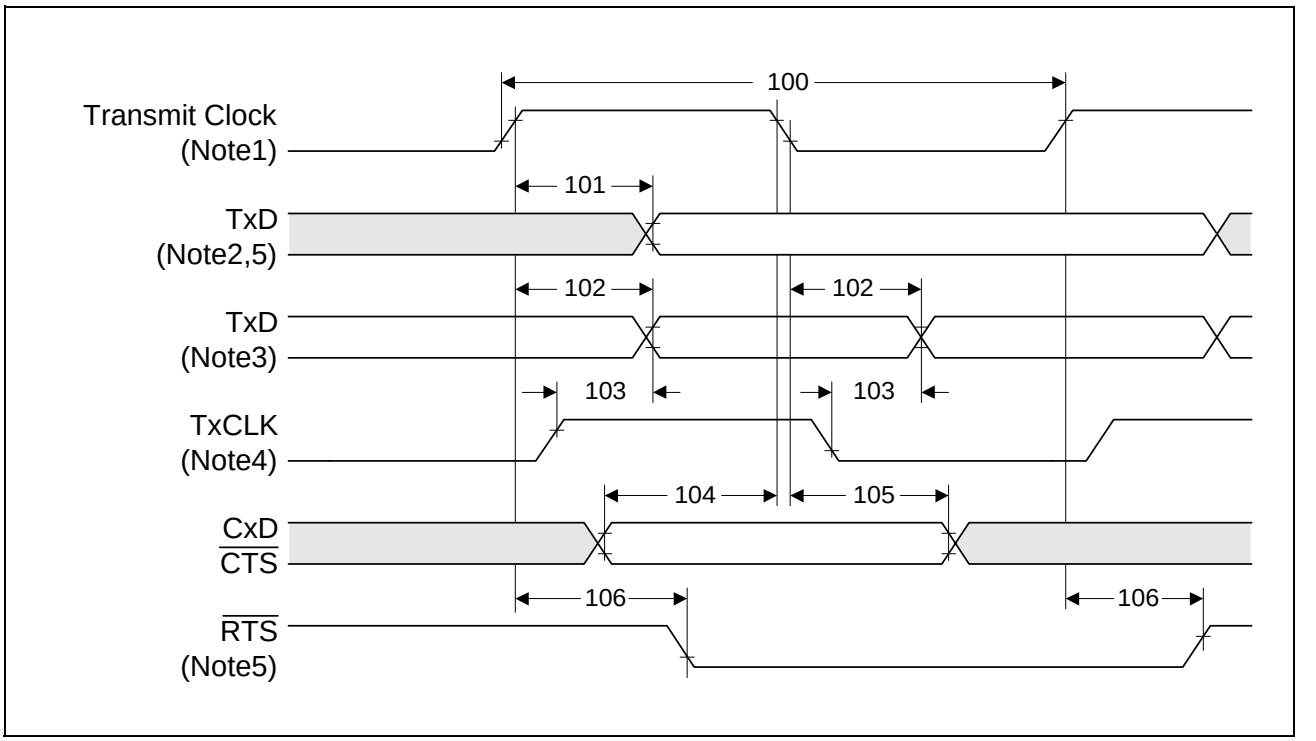


Figure 101 Transmit Cycle Timing

Note:

1. Whichever supplies the transmit clock depending on the selected clock mode:
   externally clocked via TxCLK, RxCLK or XTAL1 or
   internally clocked via DPLL, BRG or BCR.
   (No edge relation can be measured if the internal transmit clock is derived from the external clock source by devision stages (BGR, BCR) or DPLL)
2. NRZ, NRZI and Manchester data encoding
3. FM0 and FM1 data encoding
4. If TxCLK output feature is enabled (only in some clock modes)
5. The timing is valid for non bus configuration modes and bus configuration mode 1. In bus configuration mode 2, TxD and $\overline{\text{RTS}}$ are right shifted for 0.5 TxCLK periods i.e. driven by the falling TxCLK edge.
6. A transmit clock must be present to detect input level changes of signal $\overline{\text{CTS}}$ and to change the output level of signal $\overline{\text{RTS}}$.

Table 128 Transmit Cycle Timing

No.	Parameter		Limit Values		Unit
			min.	max.	
Transmit data rates		externally clocked (HDLC, high speed mode)		52	MBit/s
		externally clocked (non high speed)		10	MBit/s
		internally clocked (DPLL modes)		2	MBit/s
		internally clocked (non DPLL modes)		2	MBit/s
100	Clock period	externally clocked	19.2		ns
		internally clocked (DPLL modes)	480		ns
		internally clocked (non DPLL modes)	480		ns
101	TxD to TxCLK delay (NRZ, NRZI encoding)			35	ns
102	TxD to TxCLK delay (FM0, FM1, Manchester encoding)			35	ns
103	TxD to TxCLK(out) delay (if TxCLKO is enabled)		0	5	ns
104	CxD to TxCLK setup time, $\overline{\text{CTS}}$ to TxCLK setup time		10		ns
105	CxD to TxCLK hold time, $\overline{\text{CTS}}$ to TxCLK hold time		10		ns
106	$\overline{\text{RTS}}$ to TxCLK delay (not bus configuration mode)			35	ns
	$\overline{\text{RTS}}$ to TxCLK delay (bus configuration mode)			35	ns

13.6.5.4 Strobe Timing (clock mode 1)

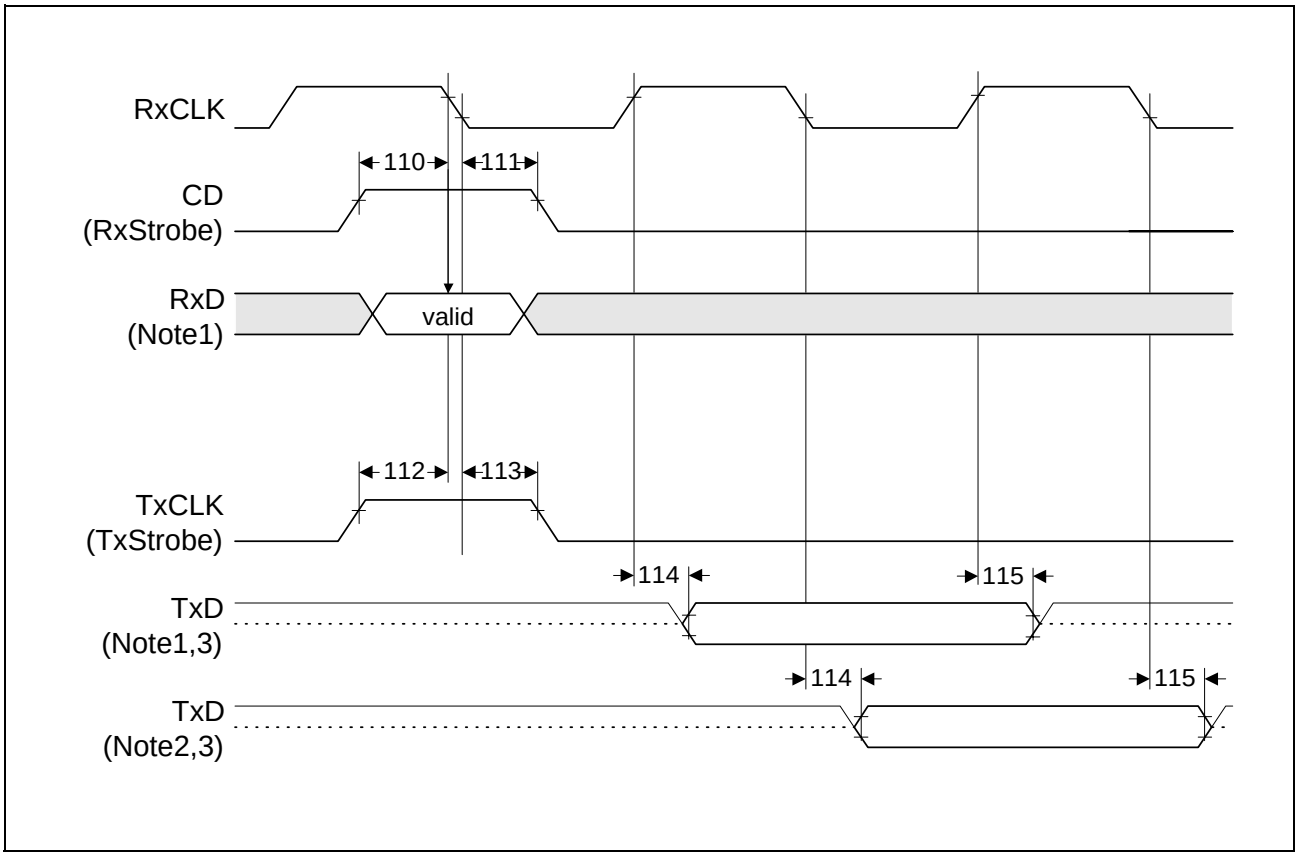


Figure 102 Strobe Timing

Note:

1. No bus configuration mode and bus configuration mode 1
2. Bus configuration mode 2
3. TxD Idle is either *active high* or *high impedance* if 'open drain' output type is selected.
4. A receive clock must be present to detect input level changes of signal CD.

Table 129 Strobe Timing (clock mode 1)

No.	Parameter	Limit Values		Unit
		min.	max.	
110	Receive strobe to RxCLK setup	5		ns
111	Receive strobe to RxCLK hold	15		ns
112	Transmit strobe to RxCLK setup	5		ns
113	Transmit strobe to RxCLK hold	15		ns

Table 129 Strobe Timing (clock mode 1)

No.	Parameter	Limit Values		Unit
		min.	max.	
114	TxD to RxCLK delay	35		ns
115	TxD to RxCLK high impedance delay	35		ns

13.6.5.5 Frame Synchronisation Timing (clock mode 5)

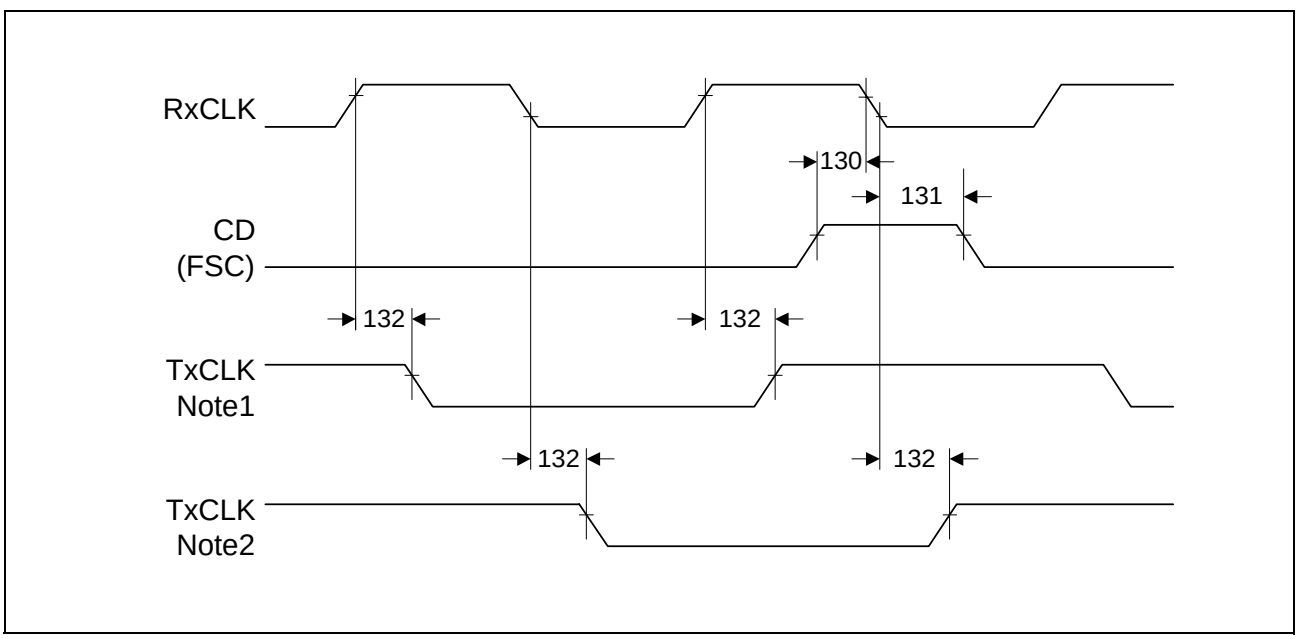


Figure 103 Frame Synchronisation Timing

Note:

1. No bus configuration mode and bus configuration mode 1
2. Bus configuration mode 2

Table 130 Frame Synchronisation Timing (clock mode 5)

No.	Parameter	Limit Values		Unit
		min.	max.	
130	Sync pulse to RxCLK setup time	5		ns
131	Sync pulse to RxCLK hold time	5		ns
132	TxCLKout to RxCLK delay (time slot monitor)	10	40	ns

13.6.5.6 High Speed Receive Cycle Timing

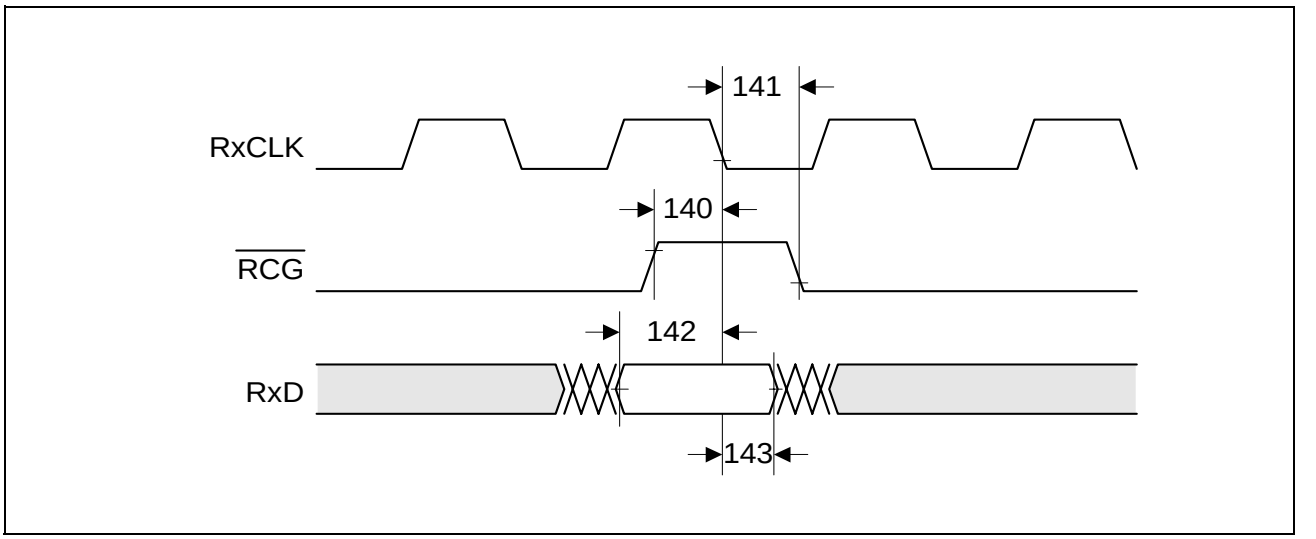


Figure 104 High Speed Receive Timing

Table 131 High Speed Receive Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
140	$\overline{\text{RCG}}$ setup time	5		ns
141	$\overline{\text{RCG}}$ hold time	5		ns
142	RxD setup time	5		ns
143	RxD hold time	5		ns

13.6.5.7 High Speed Transmit Cycle Timing

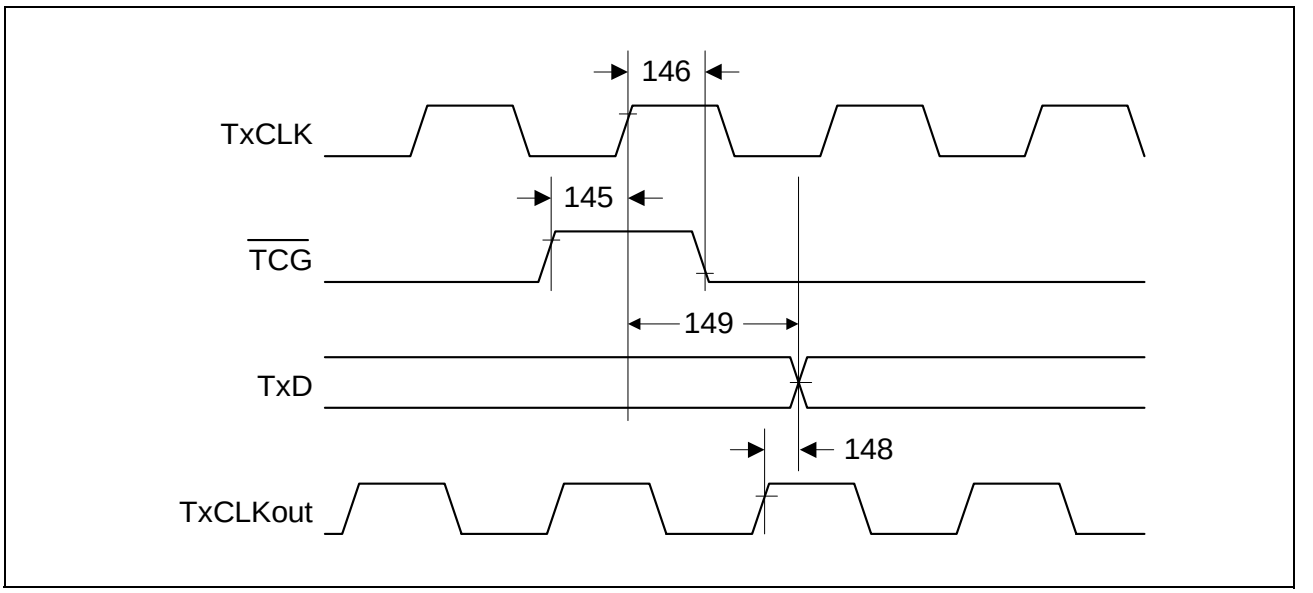


Figure 105 High Speed Transmit Timing

Table 132 High Speed Transmit Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
145	$\overline{TCG}$ setup time	5		ns
146	$\overline{TCG}$ hold time	5		ns
148	TxCLKout to TxD delay	0	5	ns
149	TxCLK to TxD delay	5	16.5	ns

13.6.6 Reset Timing

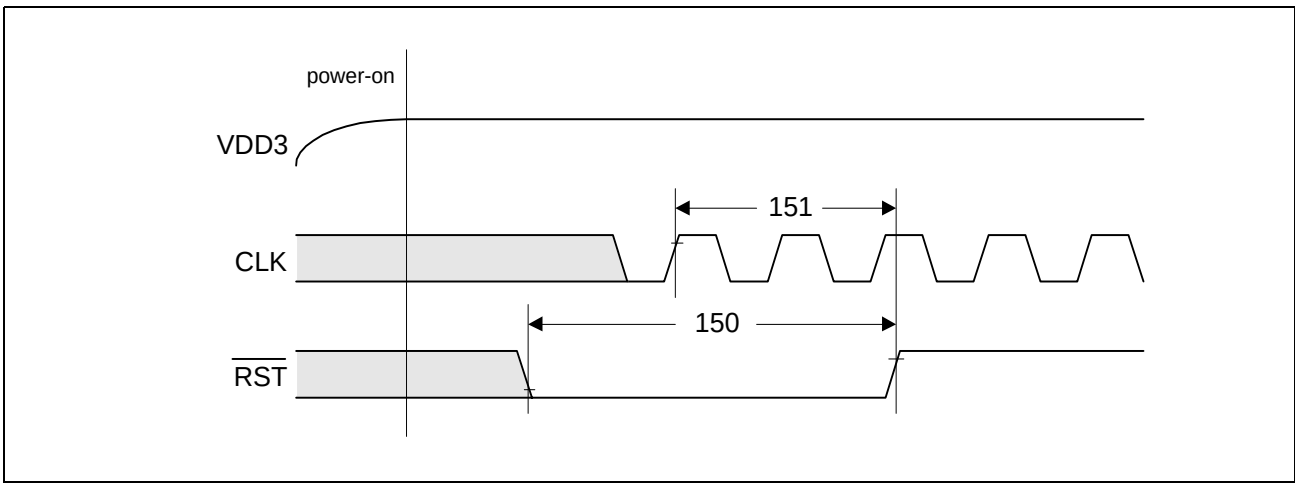


Figure 106 Reset Timing

Table 133 Reset Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
150	RESET pulse width	120		ns
150	Number of CLK cycles during $\overline{RST}$ active	2		CLK cycles

Note: $\overline{RST}$ may be asynchronous to CLK when asserted or deasserted. RST may be asserted during power-up or asserted after power-up. Nevertheless deassertion must be clean, bounce-free edge as recommended by PCI Spec. Revision 2.1.

Note: $\overline{RST}$ signal timing is independent of whether PCI or De-multiplexed mode is selected via pin DEMUX.

13.6.7 JTAG-Boundary Scan Timing

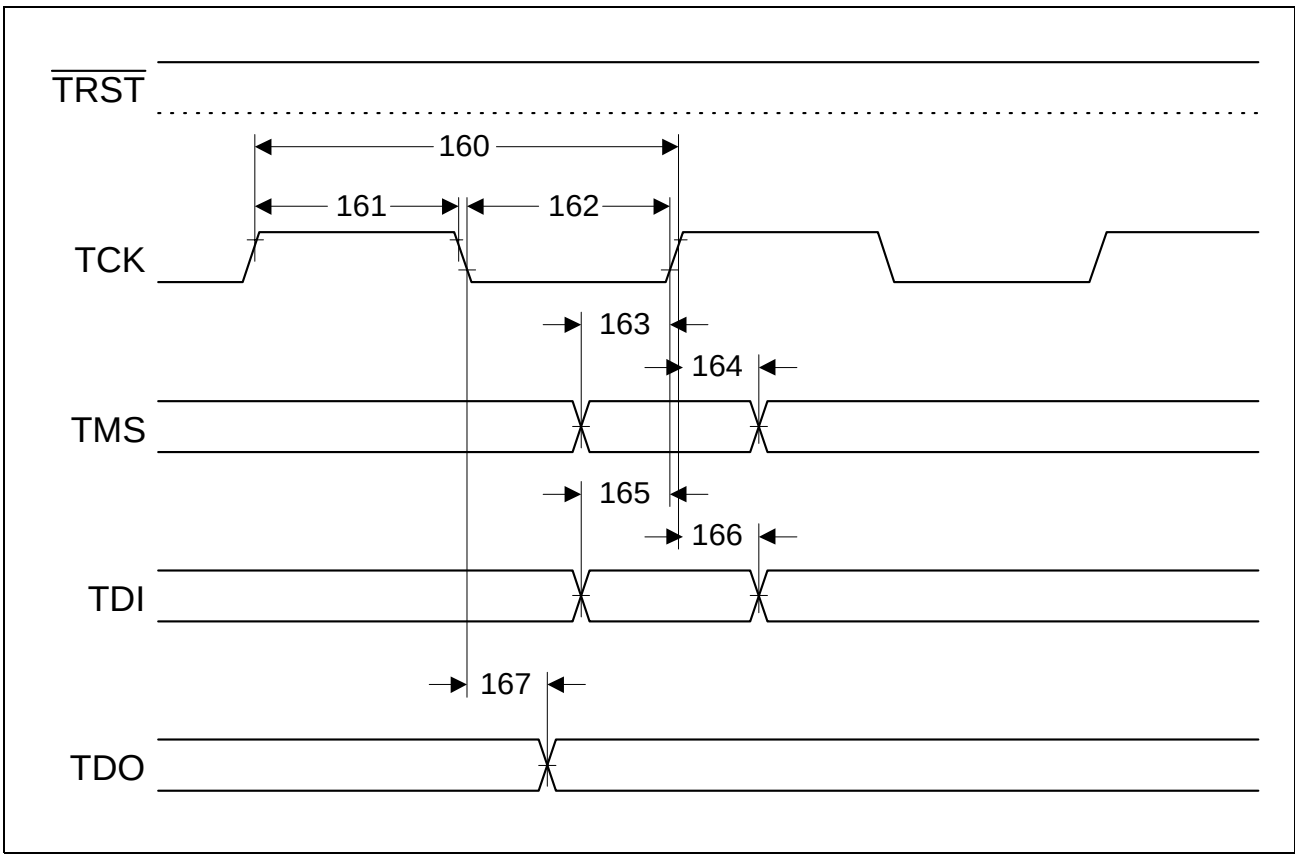


Figure 107 JTAG-Boundary Scan Timing

Table 134 JTAG-Boundary Scan Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
160	TCK period	166	∞	ns
161	TCK high time	80		ns
162	TCK low time	80		ns
163	TMS setup time	30		ns
164	TMS hold time	10		ns
165	TDI setup time	30		ns
166	TDI hold time	20		ns
167	TDO valid delay	60		ns

13.6.8 SSC Serial Interface Timing

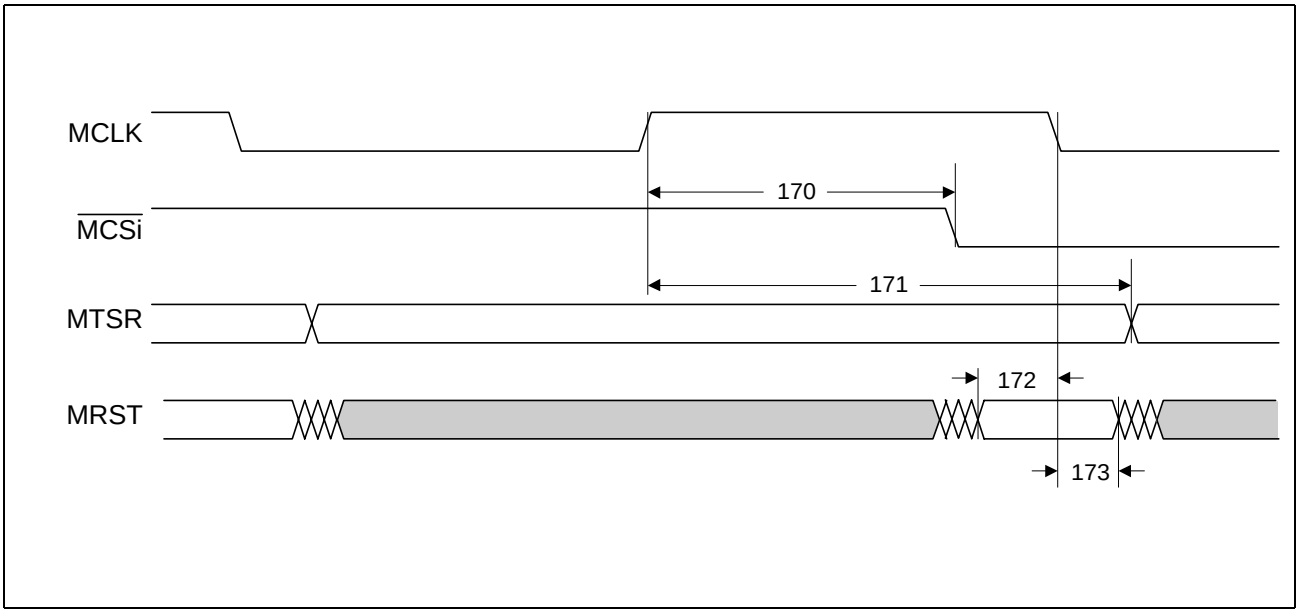


Figure 108 SSC Interface Timing (Master)

Table 135 SSC Interface Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
170	MCLK high to $\overline{\text{MCSi}}$ active delay		$2T_{CLK}$ + 40	ns
171	MCLK high to MTSR delay (Master)		$2T_{CLK}$ + 40	ns
	MCLK high to MRST delay (Slave)[1)]		$4T_{CLK}$ + 40	ns
172	MRST setup time	T_{CLK}+20		ns
173	MRST hold time	T_{CLK}+20		ns

1) In SSC 'Slave Mode', signal MRST is output and MTSR is input.

Note: T_{CLK} is the CLK signal time period.

14 Package Outlines

P-FQFP-208-7
(Plastic Metric Quad Flat Package)

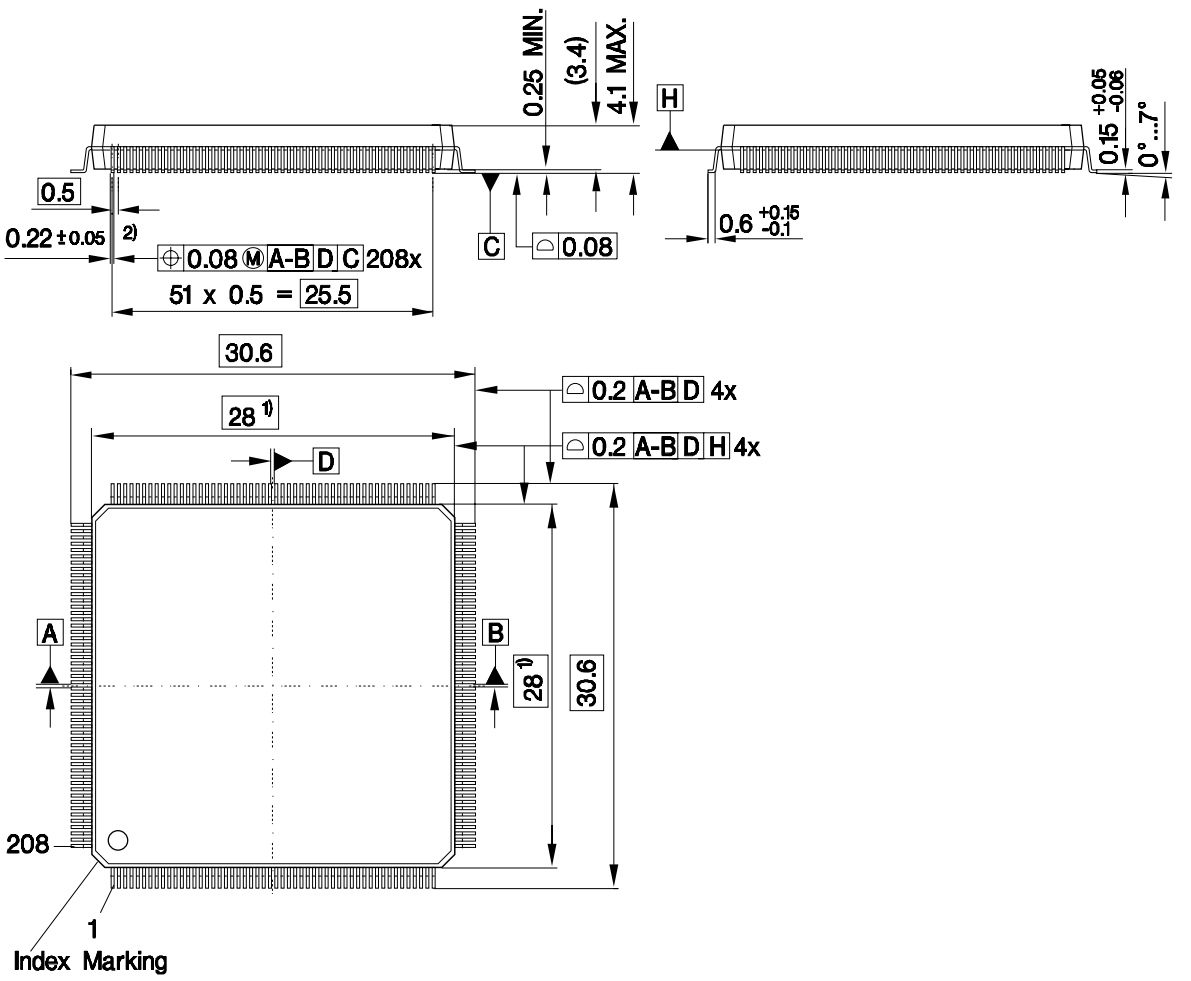


Figure 109 Package Outline

Sorts of Packing
Package outlines for tubes, trays etc. are contained in our Data Book “Package Information”.

SMD = Surface Mounted Device

Dimensions in mm