

Document Information

Info	Content
Project (full name)	DVFD818x Hardware Specification
Title (full)	Cordless VoIP Multimedia Baseband Processor
Keywords	Baseband processor, ARM926, Saturn DSP core, DECT, , VoIP



Co. Company Restricted

Contents

1 Features	3	9.4	SCRAM - System Controller RAM	211
1.1 Quick Overview	3	9.5	SCROM - System Controller ROM	212
1.2 Block Diagram	5	9.6	SCROM Software (for ordering information on the different SCROM masks see Table 4)	213
1.3 General	6		ADPCM -Adaptive Differential Pulse Code Modulation	219
1.4 Embedded System Controller - SC	6	9.7	AHB2VPB - AHB to VPB Bridge	224
1.5 Audio DSP - DSP	7	9.8	AMU1 to AMU6 - AHB Monitoring Unit	226
1.6 Toplevel and Intercore Blocks	7	9.9	ARB - Arbiter	231
1.7 Burst Mode Processor- BMP (supported at DVFD8185 only)	8	9.10	DAIF - Digital to Analog Interface	232
1.8 Ethernet MAC controller - ETN1	8	9.11	DMAU - Direct Memory Access Unit	238
1.9 SC Firmware	8	9.12	DRT - Digital Receive Transmit	260
1.10 DSP Firmware	9	9.13	EBI1 and EBI2 - External Bus Interfaces	268
2 Application Information	10	9.14	ETM and ETB - Embedded Trace Macrocell and Buffer	308
3 This Document	12	9.15	EXTINT - External Interrupt	316
3.1 Scope and Content of this Document	12	9.16	FIR - Fast IrDA Controller	325
3.2 Disclaimers	12	9.17	GPIO - General Purpose I/O	350
3.3 Trademarks	12	9.18	IIC - I ² C-bus Interface	354
3.4 Abbreviations	13	9.19	IIS - I ² S-bus Interface	364
3.5 Glossary	16	9.20	INTC - Interrupt Controller	369
3.6 Symbol Naming Conventions	16	9.21	IPINT - IOM/PCM Interface	388
3.7 References	17	9.22	KBS - Keyboard Scanner	401
4 Selection Information	19	9.23	PWM2/PWM3 - Pulse Width Modulators	408
4.1 Family types and feature overview	19	9.24	SCTU1/SCTU2 - SC Timer Units	413
5 Ordering Information	20	9.25	SDI - SDRAM Interface	417
6 Package and Pinning Information	20	9.26	SPI1 - Serial Peripheral Interfaces	450
6.1 Package Description	20	9.27	UART1/UART2 - RS232 Interfaces	470
6.2 Pin Listing	23	9.28	USB - Universal Serial Bus	505
6.3 FMP - Fast Memory Port	35	9.29	BMP - Burst Mode Processor	531
6.4 GPM - GPIO Pin Multiplexing	37	10	Introduction	531
6.5 HDP - Hardware Debug Port	42	10.1	Block diagram	533
6.6 HMP - Hybrid Memory Port	49	10.2	Hardware Interface	535
7 Toplevel Blocks	51	10.3	Software interface	536
7.1 CGU and DCXO - Clock Generation Units	51	10.4	Application information	560
7.2 TAP1/2 - JTAG Test Access Ports	77	10.5	ETN1 -10/100 Ethernet MAC	561
7.3 PDCU - Power Down Control Unit	87	11	Features	561
8 DSP Subsystem (DSP)	130	11.1	Block diagram	562
8.1 DSP Block Description	130	11.2	Hardware Interface	563
8.2 DSP Core	133	11.3	Software Interface	564
8.3 Memory Configuration	135	11.4	Basic functionality	590
8.4 I/O Register Map	138	11.5	Functional Description	591
8.5 DSP Interrupt Configuration	140	11.6	Power Management	611
8.6 BBB - Bug Bypass Block	143	11.7	Application information	612
8.7 DCON - DSP Control Unit	146	11.8	Limiting Values	623
8.8 MMR - Memory Monitor Register	148	12	DC Characteristics	623
8.9 MPU - Memory Paging Unit	151	13	Supply Voltage	623
8.10 TIO - Test I/O Interface	154	13.1	Power Dissipation	626
8.11 DMAC - Direct Memory Access Controller	159	13.2	Input Characteristics for Digital Pins	627
8.12 TMU - DSP Mailbox unit	165	13.3	Output Characteristics for Digital Pins	629
8.13 Programmer's Restrictions	175	13.4	AC Characteristics of Digital Pins	631
9 SC Subsystem	176	14	Input Characteristics	631
9.1 ARM926EJ-S - System Controller Core	176	14.1	Output Characteristics	633
9.2 SC Bus Configuration	179	14.2		
9.3 SC Memory Configuration	182			

1. Features

1.1 Quick Overview

The DVFD818x Family is part of the DSP Group highly flexible multimedia systems and ICs.

Beside standard DECT telecom features (supported at DVFD8185 only) the DVFD818x Family allow implementation of advanced multimedia features and low cost IP phone applications. The following features are offered:

- Powerful system controller
 - ARM926EJ-S subsystem, up to 208 MHz clock, 32kB + 32kB instruction and data cache
 - Dedicated instruction set for JAVA support
 - ETM9 embedded trace macrocell with on-chip trace buffer
 - Interfaces:
 - DVFD818x Family: SPI, 2 x UART, 2 x PWM, IIC, IIS, FIR, USB
 - User interface: 8 x 8 keyboard scanner matrix
- External memory interfaces: Burst Flash, synchronous and asynchronous SRAM, SDRAM (not all types are supported at DVFD818x Family see [Section 9.14](#))
 - On-chip SRAM (192kB) and ROM
 - General purpose I/Os and external interrupts
 - High performance multi-layer AHB system bus
- Audio DSP
 - Saturn DSP system running up to 120 MHz for audio feature implementation
 - Debugging and tracing via JTAG, real-time trace via hardware debug port
 - Rich set of firmware features: speech codecs (G.711, G.729AB, G.723, G.722, AMR-NB, AMR-WB, ADPCM, iLBC), wide band full duplex and half duplex acoustic echo cancellation, voice band equalizer, burst noise reduction, automatic volume control, wideband synthesis, wide band speech (G.726 at 16kHz sampling rate), MAS polyphonic music generation, MIDI player, Local and network echo cancellation, Caller ID transmission, Detectors (CLIP1/2, Ringing, Busy/Dial tone, Silence, Fax tone).
- Embedded Burst Mode Processor for DECT (supported at DVFD8185 only)
 - Fully flexible slot/frame formatting under software control
 - Ciphering, scrambling, CRC checking/generation, protected B-fields
 - Serial interface (for synthesizer programming or other functionality)
 - Programmable timing and polarity of radio control signals
 - Phase error measurements and phase error correction in hardware
 - Operation in a PABX environment by synchronization to an external master clock (SYNCPORT or IOM)
 - Configurable switching to half rate in guard space

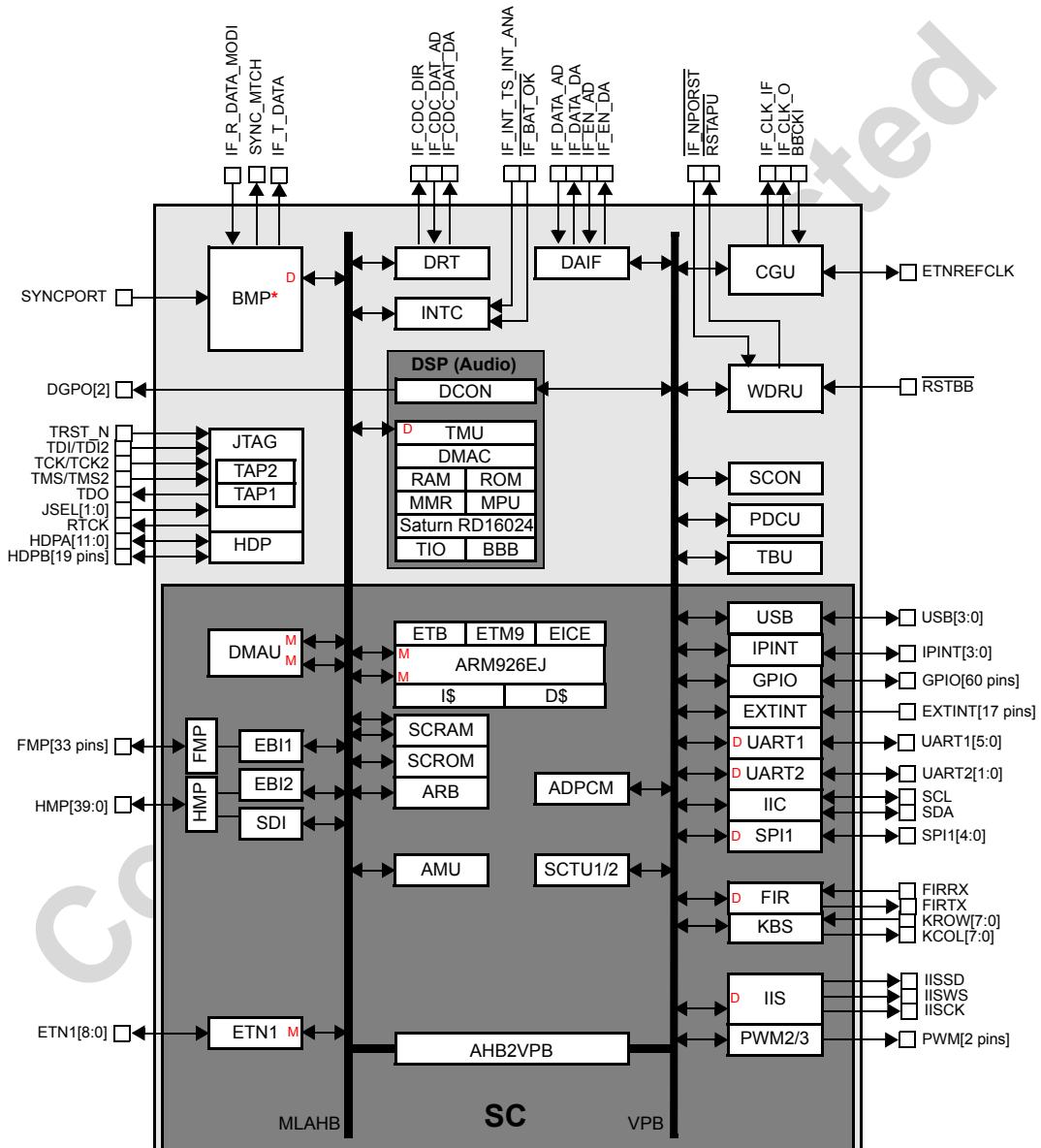
- Single embedded Ethernet MAC controller 1
- 10 or 100Mbps supported 2
- MAC sub layer of IEEE std. 802.3 3
- Attachment of external PHY chip through standard Reduced MII (RMII) interface 4
- Separate AHB interfaces for status/control and DMA traffic 5
- Low power mode 6
- Optional support for Quality of Service (VLAN) 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54

Company Restricted

1.2 Block Diagram

A functional overview of the DVFD8185 & DVFD8187 is presented in [Figure 1](#).

Please note that several pins are multiplexed. This is not shown in [Figure 1](#). Please refer to [Section 6.4](#) for more information on pin multiplexing.



Note : Blocks marked with "*" are not supported at DVFD8187

Fig 1. DVFD8185 & DVFD8187 Block Diagram

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

1.3 General

- Multiband Cordless application
 - DVFD8185 : worldwide DECT frequency bands
- Multimedia features: G.711, G.722, G.723, G.726, G.729AB, AMR-NB, AMR-WB, iLBC, JPEG, MAS polyphonic music generation, MIDI player, wide band full duplex and half duplex acoustic echo cancellation, voice band equalizer, burst noise reduction.
- Process:
 - CMOS090, 0.09 µm feature size for digital
- Voltage domains:
 - core digital supply: 1.20 V typical
 - clock generator supply: 1.20 V typical
 - low voltage I/O supply: 1.80 V typical
 - high voltage I/O supply: 3.00 V typical
 - dual voltage I/O supply: 1.80V or 2.80V typical
 - USB supply: 3.30 V typical
- Ambient temperature range 0 °C to +70 °C
- Low power consumption
- On-chip PLLs for the generation of high frequencies and system reference clocks
- Package:
 - TFBGA196

1.4 Embedded System Controller - SC

- ARM926EJ-S - powerful 32 bit micro controller subsystem
 - I\$, D\$ - instruction and data cache
 - EICE - embedded in-circuit emulator block
 - ETM9/ETB - embedded trace macrocell with on-chip trace buffer
- MLAHB - multi-layer advanced high performance bus
- VPB - VLSI peripheral bus - 32 bit data width, 13 to 104 MHz clock frequency
- AMU - AHB monitoring unit
- AHB2VPB - bridge to connect MLAHB and VPB
- ADPCM - Adaptive Differential Pulse Code Modulation
- DAIF - Digital to Analog Interface (on digital die)
- DMAU - DMA unit for fast peripheral input and output
- DRT - Digital Receive Transmit
- EBI1 - 16/32 bit¹ external bus interface with page and burst mode support (limitations see [Section 9.14](#))
- EBI2 - 16 bit static external bus interface

1. 32-bit are available for memories supporting address/data multiplexing

• EXTINT - external interrupts with debouncing capability	1
• FIR - Fast IrDA controller	2
• FMP - Fast memory port, dedicated to EBI1	3
• GPIO - general purpose I/O pins multiplexed with functional pins	4
• HMP - Hybrid memory port, shared between EBI2 and SDI	5
• IIC - fast and standard multi-master I ² C bus interface	6
• IIS - I ² S Interface	7
• INTC - interrupt control unit	8
• IPINT - IOM/PCM Interface	9
• KBS - keyboard scanner	10
• PWM - pulse width modulator with two output channel	11
• SCTU1/2 - system control timer units	12
• SCRAM - embedded static SC RAM	13
• SCROM - program ROM for bootstrap loader and security features	14
• SDI - single data rate SDRAM interface	15
• SPI1 - serial peripheral interface (master/slave), up to13 Mbps	16
• UART1 - UART, 3.25 Mbps, low-speed IrDA mode, with full handshake signals	17
• UART2 - UART, 3.25 Mbps, low-speed IrDA mode, with handshake signals	18
• USB - Universal Serial Bus (FS supported only)	19
	20
	21
	22
	23
	24
	25
	26

1.5 Audio DSP - DSP

• Saturn (RD16024) - DSP Core	27
• ROM - program and data ROM	28
• RAM - program and data RAM	29
• BBB - bug bypass block to patch bugs in DSP ROM code	30
• DMAC - direct memory access controller	31
• MMR - memory monitor register	32
• MPU - memory paging unit	33
• TIO - test I/O interface	34
	35
	36
	37
	38
	39

1.6 Toplevel and Intercore Blocks

• CGU - clock generation unit with reference clock input	40
• HDP - hardware debug port	41
• PDCU - power down control unit	42
• SCON - system configuration register	43
• TAP1/2 - JTAG test access ports	44
• TBU - Time base unit	45
• TMU - Tandem Mailbox Unit for intercore communication	46
• WDRU - watchdog and reset unit	47
	48
	49
	50
	51
	52

1.7 Burst Mode Processor- BMP (supported at DVFD8185 only)	1
• MCU - Main Control Unit	2
• RCU - Receive Control Unit	3
• RPU - Receive Peripheral Unit	4
• RFCU - RF Control Unit	5
• RFPU - RF Peripheral Unit	6
• SIF - Serial Interface	7
• SPIF - Serial peripheral Interface	8
• SYPO - Synchronization Port	9
• BMPTBU - Time base Unit of the Burst Mode Processor	10
• TCU - Transmit Control Unit	11
• TPU - Transmit Peripheral Unit	12
	13
	14
	15
	16
	17
	18
1.8 Ethernet MAC controller - ETN1	19
• PE-MAC - Ethernet core	20
• MMBD - Ethernet data interface	21
• MMIO - Ethernet I/O Interface	22
• DMA - Ethernet buffers and scatter gather DMA	23
• RMII - Reduced Media Independent Interface	24
	25
	26
	27
1.9 SC Firmware	28
• Flexible boot loader in on-chip ROM	29
• BMP driver for DECT for DVFD8185 in on-chip ROM	30
	31
	32
	33
	34
	35
	36
	37
	38
	39
	40
	41
	42
	43
	44
	45
	46
	47
	48
	49
	50
	51
	52
	53
	54

1.10 DSP Firmware

The DVFD818x Family provides an extremely rich firmware feature set. Table 1 gives an overview of the available firmware features.

Table 1: DSP Firmware Features

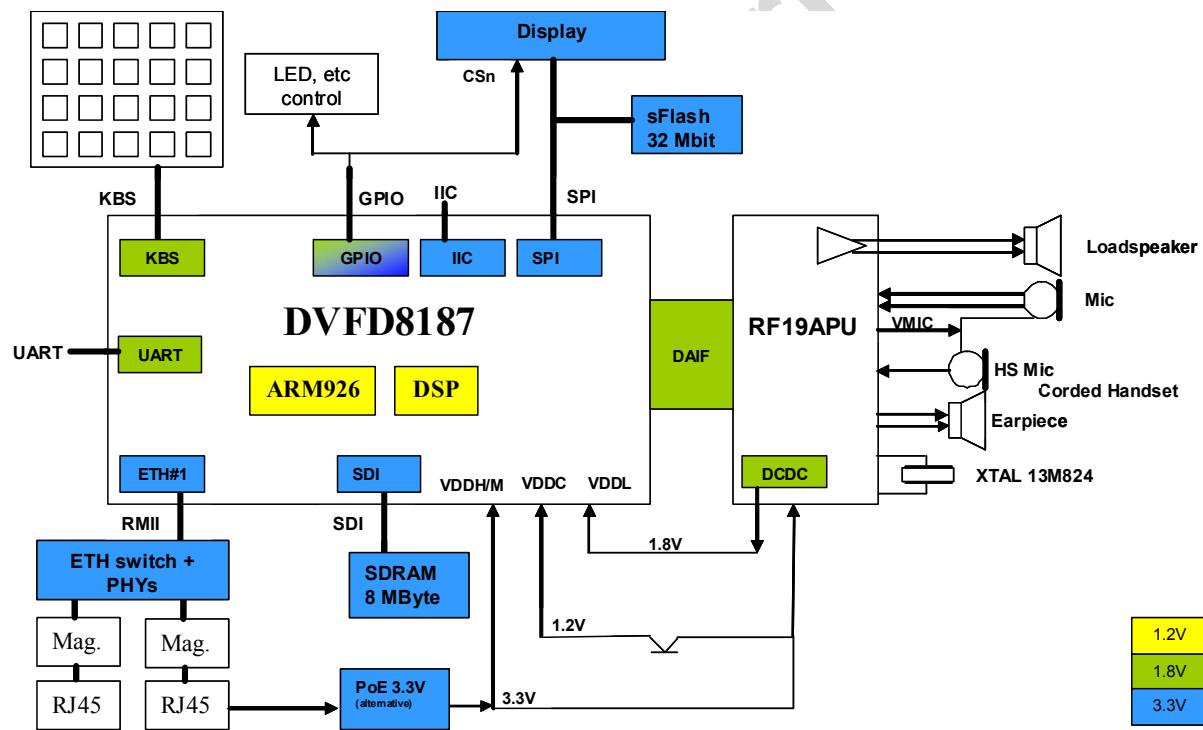
Package	Comment
Features Related to Communication Pipe	
ADPCM	Compatible with G.726
Advanced Speech Features	
MMI controlled audio equalizer	Both in receive and transmit direction
Wide band synthesis	Artificial extension of narrow-band speech (4 kHz) to wide-band speech (8 kHz)
Bad frame cancellation	Soft mute
Burst noise reduction	Radio frequency interference cancellation
Acoustic echo cancellation	Wide band (8kHz) full duplex and half duplex acoustic echo canceller
Tone generators	Generation of sine wave or square wave signals, frequency and amplitude can be set independently for each signal.
Side-tone generation	Acoustic path from microphone to loudspeaker
DTMF, Ring, Dial, Busy and FAX tone detectors	For analog line support
CLIP1/2 and caller ID transmission	For SMS on analog line
Automatic gain control and controlled volumes	For speech recording and signal clipping protection
LEC and NES	Near and far end echo cancellation on analog lines
Shared flow	Allows some audio processing on ARM
Advanced Audio Features	
G.711, G.723, G.722, G.729AB, AMR-NB, AMR-WB	VoIP
ADPCM codecs, G729, AMR-NB	Answering machine
FMS, MAS, MIDI player	Polyphonic ringing tones or music generation
Audio compressor limiter	Compression of the dynamic range of an input signal to a given reference level
Wide band speech	G.726 (ADPCM) at 16kHz sampling rate
Other Features	
Voice memo	
On-chip boot loader	Download of DSP SW via JTAG or TMU
JTAG monitor	Resident on-chip JTAG monitor program in ROM
Real-time trace via JTAG	Allows real-time tracing of DSP internal data (e.g. voice flow) via JTAG

2. Application Information

The DVFD818x Family are part of the cordless chip sets of DSP Group.

In [Figure 2](#) a basic VoIP phone is presented. It comprises the following subsystems:

- The DVFD8187 multimedia processor and the DAP1902 Power management and analog/audio engine
- An external Ethernet Switch
- An external SDRAM code execution and data
- A NOR flash for code/data storage
- MMI parts as Display, keyboard, LED's and acoustic components
- Additional interfaces for connectivity (UART)

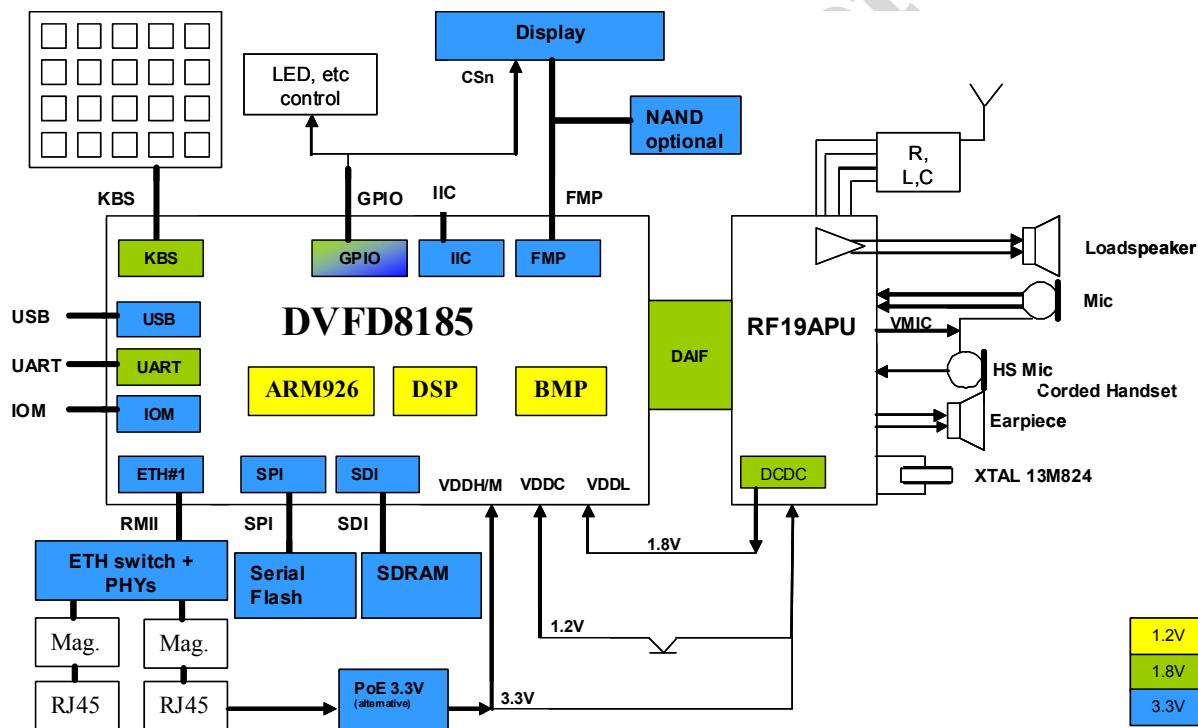


Note: Only a selection of internal blocks are shown, for a complete list, refer to the Block Diagram

[Fig 2. Application Diagram - IP phone based on DVFD8187 + DAP1902 with L2 switch](#)

Figure 3 depicts an IP phone with cordless feature (DECT). It is based on and including an external ethernet switch.

- The DVFD8185 baseband and multimedia processor
- The DRF1902 as a complete RF, analog, audio and power management unit
- An external Ethernet Switch
- An external SDRAM for code execution and data
- A SPI flash for code/data storage (optional a NAND flash can be used for code)
- MMI parts as Display, keyboard, LED's and acoustic components
- Additional interfaces for audio (IOM), connectivity (USB and UART)



Note: Only a selection of internal blocks are shown, for a complete list, refer to the Block Diagram

Fig 3. Application Diagram - IP phone + DECT based on DVFD8187 + DRF1902 with L2 switch

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

3. This Document

3.1 Scope and Content of this Document

This hardware specification describes the functionality of the DVFD818x Family of Cordless VoIP Multimedia Baseband Processors. It is intended to be used as reference by the customer for integrating the ICs into a mobile terminal, and as design documentation for development. The objective specification is based on the description given in the target specification.

This document provides information on the package, pinning, hardware and software interfaces, register layouts and memory maps of the embedded processors, detailed description of the IC functions, system test feature description, and a description of the emulation and debug features.

The description of the Firmware is not available in this document, but is described in [1.]

3.2 Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. DSP Group customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify DSP Group for any damages resulting from such application.

makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Right to make changes — DSP Group reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). DSP Group assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and

Application information — Applications that are described herein for any of these products are for illustrative purposes only. DSP Group make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

JPEG — DSP Group does not give any license or immunity, either directly or by implication, under any Essential Patent. For the avoidance of doubt, this is without prejudice to any rights grants under any other existing agreement. "Essential Patent" means the use or implementation of which is an unavoidable result of compliance with the JPEG or the MPEG-4 Video standard

3.3 Trademarks

ARM and **Thumb** are registered trademarks of ARM Limited

IOM-2 is a trademark of Infineon Technologies

ARM926EJ-S, **Jazelle**, **ARM7**, **AMBA**, **EmbeddedICE-RT**, **ETB**, **ETM**, **ETM9**, and **Embedded Trace Macrocell** are trademarks of ARM Limited

Java is a trademark of Oracle

CellularRAM is a trademark of Micron Technology, Inc., inside the U.S. and a trademark of Infineon Technologies outside the U.S.

Linux is a registered trademark of Linus Torvalds

FCRAM is a trademark or Fujitsu Limited

The I²C-bus logo, **Goodlink™** and **Softconnect™** are trademarks of Koninklijke Philips Electronics N.V.

The USB logo is a trademark of USB Implementers Forum

All other product names are trademarks or registered trademarks of their respective owners

3.4 Abbreviations

This list contains the majority of the abbreviations used within this document, including the acronyms of the blocks. The name of registers and signals are not included with the abbreviations.

Table 2: Abbreviations

AAC	Advanced Audio Coding
ACP	Application Co-Processor
ADC, A/D	Analog to Digital Converter
ADIF	Analog to Digital Interface
AFE	Analog Front End
AHB	Advanced High-performance Bus
AHB2VP	AHB to VPB Bridge
B	
AMBA	Advanced Microcontroller Bus Architecture
AMR (NB and WB)	Adaptive Multi-rate Codec, Narrow and Wide Band
API	Application Programming Interface
APU	Analog Processing Unit
ARB	Arbiter block of AHB
ARM™	Advanced Risc Machines
ASI	Application Specific Instruction
BB	BaseBand
BBB	Bug Bypass Block
BGA	Ball Grid Array
CGU	Clock Generation Unit
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing unit
CRC	Cyclic Redundancy Check
D\$	Data Cache
DAC, D/A	Digital to Analog Converter
DAIF	Digital to Analog interface
DCON	DSP Configuration block
DDF	Digital Decimation Filter
DECT	Digital Enhanced Cordless Telephony
DMAC	Direct Memory Access Controller (DSP)
DMAU	Direct Memory Access Unit (SC)
DNS	Digital Noise Shaper
DPU	Digital Processing Unit
DSP	Digital Signal Processor
DSS	DSP Subsystem
DTL	Device Transaction Level
EBI1	External Bus Interfaces
EBI2	

Table 2: Abbreviations...continued

EEPROM	Electrically Erasable Programmable Read Only Memory
EICE	Embedded In-Circuit Emulator
ETB	Embedded Trace Buffer
ETM	Embedded Trace Macrocell
ETSI	European Telecommunications Standards Institute
ETU	Elementary Time Unit (ISO 7816-3)
FAD	Fast Antenna Diversity
FCI	Flash Card Interface
FIQ	Fast interrupt Request
FIR	Fast IRda or Finite Impulse Response
FM	Frequency Modulation
FMP	Fast Memory Port
FPGA	Field Programmable Gate Array
FR	Frequency Response
GFSK	Gaussian Frequency Shift Keying
GPIO	General Purpose Input Output
GPM	GPIO Pin Multiplexing
HAL	Hardware abstraction layer
HDP	Hardware Debug Port
HiFi	High Fidelity
HMP	Hybrid Memory Port
HW	Hardware
I\$	Instruction Cache
I ² C, IIC	Inter IC Control bus; I ² C-bus design unit
I ² S, IIS	Inter IC Sound bus, I ² S-bus design unit
IC	Integrated Circuit
ICE	In-Circuit Emulator
ICN	Idle Channel Noise
INTC	Interrupt Controller
IOM-2™	ISDN Oriented Modular Interface, Revision 2
IP	Intellectual Property system component It refers to the hardware and software parts of the component that will provide a specific function in the system
IrDA	Infrared Data Association
IRQ	Interrupt Request
ISM	Industrial Scientific and Medical band

Table 2: Abbreviations...continued

ISR	Interrupt Service Routine
ITCM	Instruction Tightly Coupled Memory
JPEG	Joint Picture Experts Group
JTAG	Joint Test Action Group
KBS	Keyboard Scanner
LCD	Liquid Crystal Display
LP	Low Pass
LSB	Least Significant Bit
LSP	Least Significant Part
MAC	Media Access Controller
MCP	Multi Chip Package
MICBIAS	Microphone Bias
MLAHB	Multi-Layer Advanced High Performance Bus
MMBD	Memory Mapped Block Data
MMC	Multi Media Card
MMI	Man Machine Interface
MMIO	Memory Mapped Input Output
MMR	Memory Monitor Register
Mobile-S	SDRAM with additional low power features like TCSR, PASR
MP3	MPEG: Audio Layer 3 (MPEG1, MPEG2 and MPEG2.5)
MPEG	Motion Picture Expert Group
MPU	Memory Paging Unit
MSB	Most Significant Bit
NFC	Near Field Communication
NZIF	Near Zero Intermediate Frequency
OS	Operating System
PASR	Partial Array Self Refresh (for Mobile-SDRAM devices)
PCM	Pulse Code Modulation
PDCU	Power-Down Control Unit
PE-MAC	Packet Engine MAC
PID	Packet Identification (USB specification)
PIO	Parallel IO unit
PLL	Phase Locked Loop
PRAM	Program RAM for DSP
Prediction	The use of a predictor to provide an estimate of a sample or data element currently being decoded
Predictor	A linear combination of previously decoded samples of data elements
PROM	Program ROM for DSP

Table 2: Abbreviations...continued

PWM	Pulse Width Modulator
RAM	Random Access Memory
QoS	Quality of Service
RD16024	Saturn DSP Core
RF	Radio Frequency
RFCU	RF Control Unit
RMII	Reduced Media Independent Interface
ROM	Read Only Memory
Rx	Receive Path
SC	System Controller (ARM926EJ-S)
SCON	System Configuration
SCRAM	SC RAM
SCROM	SC ROM
SCTU	SC Timer Control Units
SDI	SDRAM Interface
SDR	Single Data Rate (applicable to SDRAM)
SDRAM	Synchronous Dynamic Random Access Memory
SIM	Subscriber Identity Module (GSM specification)
SINAD	Signal to Noise and Distortion
SIR	Serial Infrared
SJTAG	SC JTAG
SOF	Start Of Frame (USB specification)
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SW	Software
TAP	Test Access Port
TAPC	TAP Controller
TBU	Time Base Unit
TCB	Test Control Block
TCM	Tightly Coupled Memory
TCSR	Temperature Compensated Self Refresh (for Mobile-SDRAM devices)
TDMA	Time Division Multiple Access
THD	Total Harmonic Distortion
TIO	Test IO
TPR	Test Point Register
TTY	Text Telephony
Tx	Transmit Path
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

Table 2: Abbreviations...continued

USIM	UMTS SIM
VLIW	Very Large Instruction Word
VPB	VLSI Peripheral Bus
WDRU	Watchdog and Reset Unit
WLAN	Wireless Local Area Network (also known as Wi-Fi)
WMA	Windows Media Audio
XYRAM	Data RAM for DSP
XYROM	Data ROM for DSP
ZIF	Zero Intermediate Frequency

3.5 Glossary

This section is intended as short reference and explanation for some concepts and terms used in this document. Further information may be found in the reference documents.

32 kHz — in the complete document with 32 kHz, the frequency of 32000 Hz and alternatively 32768 Hz is intended depending on the context.

Debug features — functionality which supports the software development providing on-line and off-line access in read and write mode to internal circuitry by means of scan.

Die — single piece of silicon containing an integrated functionality.

Emulation features — functionality which supports substitution of a specific circuitry (for instance the target memory of a processor by external emulator memory) the substitution may support the functionality in real-time, or at a slowed down speed.

Flash — electrically erasable programmable non-volatile memory.

JTAGEPP — JTAG Enhanced Parallel Port: this piece of hardware adapts the protocol of a PC parallel port and of a DAI system simulator interface to a JTAG port [7].

Prototyping features — functionality which allows external extension for early development or replacement of internal functionality for debug purposes (usually implemented with the external availability of internal bus systems) prototyping is usually supported at lower speeds than internally available.

Tracing features — non-intrusive analysis of internal signals available on the external pinning with dedicated pins, or multiplexed with functional pins.

RF19 — RF solution for DECT band of DSP Group

3.6 Symbol Naming Conventions

The naming conventions for the symbols used in this document are explained hereafter.

Primary pins — primary inputs and outputs are always written UPPERCASE - active low pins are written as UPPERCASE; primary pin names do not contain underscores, except pins with a double function

Register field access —

- **R**: this field can be accessed in read mode only; should be written with zero; a write access does not have any impact
- **R/W**: this field can be accessed both in write and in read mode
- **R/C**: this field can be accessed both in write and in read mode; a write with one does not have any effect; a write with zero clears the bit if set
- **R/S**: this field can be accessed both in read and write mode; a write with one sets the bit if cleared; a write with zero has no effect

- **W:** this field can only be accessed in write mode; a read access to this field has no effect to the circuit, and returns either zero or generates a bus error on the SC side 1
 - **RaC:** read and auto clear; this bit can only be read; if the bit is set, it is automatically cleared in the cycle following the read access; a write access has not effect 2
 - **SaC:** set and auto clear; setting this bit provokes a certain function inside the circuit; the bit is automatically cleared by the hardware; clearing this bit has no effect 3
 - **WaC:** write and auto clear; setting this bit provokes a certain function inside the circuit; the bit is automatically cleared by the hardware; clearing this bit has no effect 4
- Register descriptions** — for each register field the value noted with asterisk * is the reset value of that field - exceptions are explicitly noted 5
- Register field names** — short acronyms of the function and usually do not contain underscores; always **lower case and bold**, except when used in tables 6
- Register names** — short acronyms of the function and usually do not contain underscores; always **lower case and bold** 7
- Signal names** — may contain underscores and are **lower case and bold** when used in text such as paragraph and tables; in block diagrams signal names are plain text; active low signals are noted with **signal_n** 8

3.7 References

1. "Vega-Firebird DSP FW-ROM 1.0 + API (D95)", Version 1.1 Accepted, 11 May 2006, Key ID DOC-052452, Author Markus Lüdicke 28
2. "UDA1380 Stereo audio coder-decoder for MD, CD and MP3", Product specification, 22nd of April 2004, Philips Semiconductors 29
3. "I2C-Bus Specification", version 2.0, Philips Semiconductors, 1998 30
4. "Saturn Atmosphere, Atmosphere IDE user manual", version 1.1, AT2004.0012, 8th of July 2004, Adelante Technologies B.V., www.adelantetech.com 31
5. "Command line simulation user and reference manual", Version v1.0, AT2002.0110, 5th of August 2004, Adelante Technologies B.V., www.adelantetech.com 32
6. "Saturn RD16024 Programmer s Manual", AT.2004.0064, version 5.2 rev1, AT.2004.0064, 21st of June 2004, Adelante Technologies B.V., www.adelantetech.com 33
7. "JTAG-Adapter Documentation", release 1.0, HEITEC Datentechnik GmbH, 25th of September 1998, www.heitec.de 34
8. "TRACE 32", Online User Manual, Lauterbach Datentechnik GmbH, 2002, www.lauterbach.com 35
9. "IEEE Standard Test Access Port and Boundary Scan Architecture", IEEE Std 1149.1-1990, JTAG, 1990, www.ieee.com 36
10. "RD16024 DSP Sub System User Manual", Version 1.2, 31st of March 2004, NXP Semiconductors DSPIC 37

11. "IOM-2 Interface Reference Guide", revision 1.90, Infineon AG, www.infineon.com	1 2
12. "ARM926EJ-S" Technical Reference Manual, Rev 0, ARM Limited, ARM DDI 0198B, January 2002, www.arm.com	3 4
13. "AMBA Specification", revision 2.0, ARM Limited, ARM IHI 0011A, 1999, www.arm.com	5 6
14. "ETM9", Technical Reference Manual, Rev 2a, ARM Limited, ARH DDI 0157E, 6th of June 2001, www.arm.com	7 8 9
15. "Embedded Trace Macrocell", Specification, ARM Limited, ARM IHI 0014H, 25th of July 2001, www.arm.com	10 11
16. "Embedded Trace Buffer", Specification, ARM Limited, ARM DDI 0242B, 5th of February 2002, www.arm.com	12 13 14
17. "4 Megabit Serial DataFlash - AT45DB041", Atmel, Rev. 0669D - 07/98, www.atmel.com	15 16
18. "Information Technology - Identification cards - Integrated circuit(s) cards with contact - Part 3: electronic signals and transmission protocols-3", ISO 7816-3, 15th December 1997, www.standardsinfo.net	17 18 19
19. "Serial Infrared Physical Layer Specification", version 1.4, Infrared Data Association, 30th may 2001, www.irda.org	20 21 22
20. "Infrared Data Association Serial Infrared Link Access Protocol (IrLAP)", version 1.1, Infrared Data Association, 16th June 1996, www.irda.org	23 24
21. "Rating Systems for Electronic Tubes and Valves and Analogous Semiconductor Devices", IEC 60134, 1961, www.standardsinfo.net	25 26
22. "Universal Serial Bus Specification", Revision 2.0, Compaq - Hewlett-Packard - Intel - Lucent - Microsoft - NEC - Philips, 27th of April 2000, www.usb.org	27 28 29
23. "Open Host Controller Interface Specification for USB", Revision 1.0a, Compaq - Microsoft - National Semiconductors, 14th of September 1999, www.usb.org	30 31
24. User Manual DSP Application Programming Interface, Version 6.2, ABU11-D62-UD001, 3rd of Oct 2007	32 33
25. ETS 300 175-3, DECT Medium Access Control Layer, Edition 2, European Telecommunication Standards Institute September 1996	34 35 36
26. ETS 300 175-2, DECT Physical Layer, Edition 2, European Telecommunication Standards Institute, September 1996	37 38
27. Specification of the Bluetooth System, version 1.2 2nd draft, 17 January 2003	39
28. User Manual Adelante Adelante SDK for RD1602x, NXP Semiconductors, Revision 1.05, 12-Sep-2007	40 41
29. "PNX818x/DVFD818x Boot Code User manual", Doc.Rev. 1.0, 12 Mar 2010, DSP Group	42 43 44
30. "PNX818x/DVFD818x NAND Boot Code User manual", Doc.Rev. 1.0, 12 Mar 2010, DSP Group	45 46 47 48 49 50 51 52

4. Selection Information

4.1 Family types and feature overview

Table 3: Family type and feature overview of the DPU

Device	DVFD8185	DVFD8187
Operating System	Linux	
Main CPU	ARM926EJ-S™ @ 208MHz core	
DSP	R.E.A.L. DSP Core for Audio processing functions with speed up to 120MHz (dual Harvard Architecture)	
MAC	1x 10/100Mbps	
Flash	Serial SPI / NAND	
Package	12*12mm ² 196 TFBGA (0.8mm pitch)	
IP-Phone PCB	2 layers	
PCM I/F	√	
DECT support	√	-
I2C	√	
SPI	√	
UART	√	
USB	√ (Full speed and as device only)	
FIR	√	
KBS	√ (up to 8x8 matrix)	
IIS	√	
PWM	√ (3)	
	DRF1902	DAP1902
Speaker amplifier	> 1 Watt (4 Ohm)	
ADC	4 inputs, 10-bit resolution with low SINAD	
Package	8x8mm QFN68 (0.4 mm pitch)	
Audio codecs	(2 inputs + 2 outputs, all differential)	
Power supplies	Embedded 1.2V (only with internal core DC2DC down converter), 1.8V (and 3.3V opt.) LDO's and DC/DC up converters	
PCB Layout	2 layers	
Battery management	√	
RF sensitivity	-98dBm	N/A
RF output power	+25.5dBm	N/A
Non blind slot support		N/A

5. Ordering Information

Table 4: Ordering information

Type Number	Package		
	Name	Description	Type
DVFD8185AE1ABC	TFBGA196	plastic thin profile quadratic flat ball grid array, 196 balls, SCROM software mask A	TF196-SW5
DVFD8185BE1ABC	TFBGA196	plastic thin profile quadratic flat ball grid array, 196 balls, SCROM software mask B	TF196-SW5
DVFD8187AE1ABC	TFBGA196	plastic thin profile quadratic flat ball grid array, 196 balls, SCROM software mask A	TF196-SW5
DVFD8187BE1ABC	TFBGA196	plastic thin profile quadratic flat ball grid array, 196 balls, SCROM software mask B	TF196-SW5

6. Package and Pinning Information

6.1 Package Description

Table 5: Package details

Chip	DVFD818x Family
Package	TFBGA196
Purpose	production package
Pitch	0.8 mm
Overall dimensions	12x12mm ²
Power and ground pins	51
Signal pins	145
Reserved pins	-
Total pins	196

Compliance Restricted

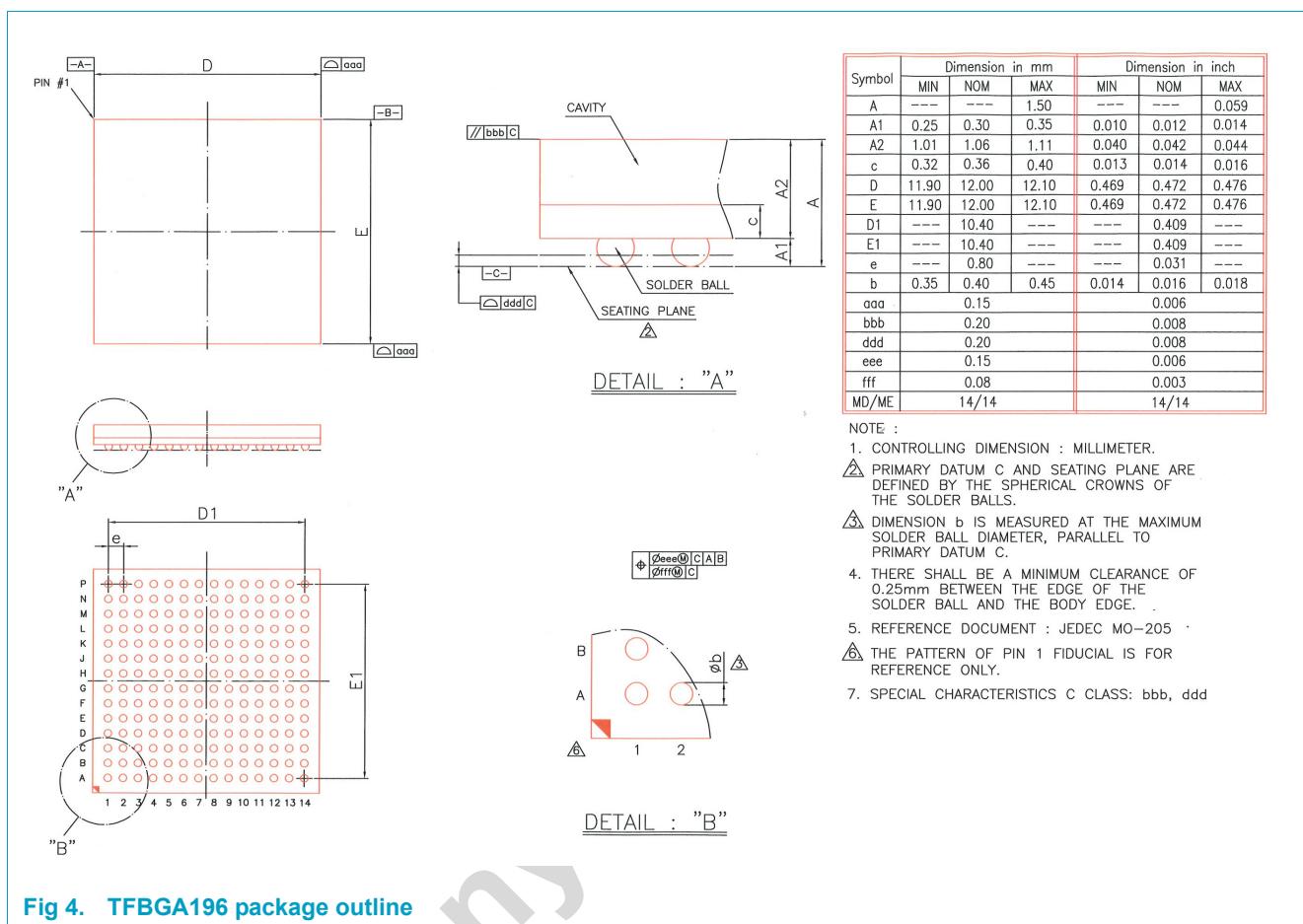


Fig 4. TFBGA196 package outline

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 6: TFBGA196, 12x12x1.24mm, 0,8mm pitch (top transparent view) for DVFD8185 & DVFD8187

NC = Not Connected

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	RSTBB_N	TDI	GPIOA5	ETNREFCLK	GPIOC0 (ETN1TXD0)	GPIOC2 (ETN1RXD0)	GPIOC5 (ETN1RXER)	GPIOA1 (RXD2)	GPIOB26 (FMP42)	GPIOB25 (FMP44)	FMP49	FMP47	FMP39	FMP34
B	GPIOB23 (SYNC_MTCH)	TDO	SCL	GPIOC12	GPIOC1 (ETN1TXD1)	GPIOC3 (ETN1RXD1)	GPIOA11 (TXD2)	GPIOA2	GPIOB24 (FMP53)	FMP48	FMP45	FMP38	FMP35	FMP33
C	IF_CDC_ DAT_AD	TMS	JSEL0	SDA	GPIOC16 (ETNMDC)	GPIOC4 (ETN1CRSDV1)	VSS	VDDH	FMP50	FMP37	FMP36	FMP14	FMP11	FMP7
D	IF_CDC_ DAT_DA	RTCK	TRST_N	GPIOA13	GPIOC15 (ETNMDIO)	GPIOC6 (ETN1TXEN)	VSS	VDDH	FMP46	FMP40	FMP15	FMP13	FMP10	FMP4
E	IF_CDC_DIR	IF_BAT_OK	TCK	JSEL1	VDDC	GPIOC14 (USBVBUS)	VSS	VDDM	VDDM	VDDM	FMP8	FMP12	FMP5	FMP2
F	IF_CLK_O	IF_CLK_IF	VDDINT	VDDL	VSS	VDDC	VSS	VSS	VSS	VSS	FMP6	FMP9	FMP3	FMP1
G	IF_DATA_AD	BBCKI	VSSINT	VSS	VDDCGU	VDDL	VSS	VDDC	VSS	VDDM	VDDM	VDDM	FMP0	HMP37
H	IF_DATA_DA	IF_EN_AD	IF_INT_TS_ INT_ANA	KCOL5	VDDACGU	VSSCGU	VDDC	VSS	VDDC	VSS	VSS	VSS	HMP39	HMP36
J	IF_EN_DA	RSTAPU_N	IF_T_DATA	KCOL4	VSSACGU	VDDH	VSS	VDDC	VDDM	VSS	HMP33	HMP38	HMP35	HMP32
K	IF_NPORST	IF_R_DATA_ MOD_I	KCOL2	KCOL3	VSS	VDDH	VSS	VSS	VDDM	VDDM	HMP29	HMP34	HMP30	HMP31
L	KCOL1	KROW4	KROW5	KCOL0	GPIOA19 (SCLK1)	GPIOA30 (DL_IP)	VDDC	VSS	HMP6	HMP7	HMP16	HMP28	HMP27	HMP26
M	KROW3	GPIOB22 (KROW7)	GPIOB21 (KROW6)	KROW1	GPIOA31 (SDATIN1)	GPIOA26 (DO_IP)	VDDC	VSS	HMP14	HMP19	HMP24	HMP20	HMP21	HMP22
N	KROW2	GPIOB20 (KROL7)	KROW0	USBCN	GPIOA29 (SDATIO1)	GPIOA16 (SCS_N_LOAD1)	HMP3	HMP10	HMP1	HMP13	HMP15	HMP12	HMP23	HMP17
P	GPIOB19 (KCOL6)	USBDP	USBDM	GPIOA25 (FSC_IP)	GPIOA24 (DCK_IP)	HMP2	HMP9	HMP0	HMP4	HMP5	HMP11	HMP8	HMP18	HMP25

1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	----	----	----	----	----

- [1] It is mandatory to connect all supply and ground balls (having the same name) together at the application PCB.

6.2 Pin Listing

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks

Signal	Description	I/O Pad type	Drive [1]	Level during [2] Reset state [5]	Sleep state	Power up	Tristate [3]	Supply [4]	Muxed [4]
BMP RF Control Interface									
SYNCPORT	DECT synchronisation port	I/O TBSHPW2	-	-	oper	-	func	VDDH	M
SYNC_MTCH	Correlator sync match event out	O TBHMP	6	-	oper	-	highz	VDDL	M
IPINT									
DO_IP	IOM data output	O TBSHPW1	-	-	-	-	func	VDDH	M
FSC_IP	IOM frame sync	I/O TBSHPW1	-	-	-	-	func	VDDH	M
DCK_IP	IOM data clock	I/O TBSHPW1	-	-	-	-	func	VDDH	M
DI_IP	IOM data in	I TBSHPW2	-	-	-	-	func	VDDH	M
DSP General Purpose Outputs (DGPO)									
DGPO2	DSP2 general purpose output	O TBSHP	-	-	oper	-	highz	VDDH	M
Hybrid Memory Port (HMP)									
HMP39	HMP signal 39	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP38	HMP signal 38	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP37	HMP signal 37	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP36	HMP signal 36	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP35	HMP signal 35	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP34	HMP signal 34	O TOM_dualV	6	H	oper	-	highz	VDDM	D
HMP33	HMP signal 33	O TOM_dualV	6	L	oper	-	highz	VDDM	D
HMP32	HMP signal 32	O TOM_dualV	6	L	oper	-	highz	VDDM	D
HMP31	HMP signal 31	O TOM_dualV	6	L	oper	-	highz	VDDM	D
HMP30	HMP signal 30	O TBHMP_dualV	6	L	oper	-	func	VDDM	P
HMP29	HMP signal 29	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	P
HMP28	HMP signal 28	O TBHMP_dualV	6	L	oper	-	func	VDDM	P
HMP27	HMP signal 27	O TBHMP_dualV	6	L	oper	-	func	VDDM	P
HMP26	HMP signal 26	O TBHMP_dualV	6	L	oper	-	func	VDDM	P
HMP25	HMP signal 25	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP24	HMP signal 24	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP23	HMP signal 23	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP22	HMP signal 22	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP21	HMP signal 21	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP20	HMP signal 20	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP19	HMP signal 19	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP18	HMP signal 18	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP17	HMP signal 17	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP16	HMP signal 16	O TOM_dualV	6	L	oper	-	func	VDDM	D
HMP15	HMP signal 15	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
HMP14	HMP signal 14	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D

Company Confidential

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2]			Tristate [3]	Supply [3]	Muxed [4]
				Reset state [5]	Sleep state	Power up			
HMP13	HMP signal 13	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP12	HMP signal 12	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP11	HMP signal 11	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP10	HMP signal 10	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP9	HMP signal 9	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP8	HMP signal 8	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP7	HMP signal 7	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP6	HMP signal 6	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP5	HMP signal 5	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP4	HMP signal 4	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP3	HMP signal 3	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP2	HMP signal 2	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP1	HMP signal 1	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
HMP0	HMP signal 0	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
Fast Memory Port (FMP)									
FMP54	FMP signal 54	I TBHMP_dualV	-	-	-	-	-	VDDM	M
FMP53	FMP signal 53	O TBHMP_dualV	6	-	oper		highz	VDDM	M
FMP50	FMP signal 50	O TBHMP_dualV	6	H	oper		highz	VDDM	P
FMP49	FMP signal 49	O TOM_dualV	6	H	oper		highz	VDDM	D
FMP48	FMP signal 48	O TOM_dualV	6	H	oper		highz	VDDM	D
FMP47	FMP signal 47	O TOM_dualV	6	H	oper		highz	VDDM	D
FMP46	FMP signal 46	O TBHMP_dualV	6	H	oper	-	highz	VDDM	P
FMP45	FMP signal 45	O TBHMP_dualV	6	H	oper		highz	VDDM	P
FMP44	FMP signal 44	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
FMP42	FMP signal 42	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
FMP40	FMP signal 40	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP39	FMP signal 39	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP38	FMP signal 38	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP37	FMP signal 37	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP36	FMP signal 36	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP35	FMP signal 35	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP34	FMP signal 34	O TBHMP_dualV	6	L	oper		highz	VDDM	P
FMP33	FMP signal 33	O TBHMP_dualV	6	H	oper		highz	VDDM	P
FMP15	FMP signal 15	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
FMP14	FMP signal 14	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
FMP13	FMP signal 13	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
FMP12	FMP signal 12	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
FMP11	FMP signal 11	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D
FMP10	FMP signal 10	I/O TBHM_dualV	6	R weak	R weak		func	VDDM	D

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2]			Tristate [3]	Supply [3]	Muxed [4]
				Reset state [5]	Sleep state	Power up			
FMP9	FMP signal 9	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP8	FMP signal 8	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP7	FMP signal 7	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP6	FMP signal 6	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP5	FMP signal 5	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP4	FMP signal 4	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP3	FMP signal 3	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP2	FMP signal 2	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP1	FMP signal 1	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
FMP0	FMP signal 0	I/O TBHM_dualV	6	R weak	R weak	-	func	VDDM	D
UART1 Data Interface (UART1)									
TXD1	UART1 transmit data	O TBS1HP	- -	oper	-	highz	VDDH	M	
TXD1_copy	UART1 transmit data (alternate)	O TBS1HP	- -	oper	-	highz	VDDH	M	
RXD1	UART1 receive data	I TBS1HP	- -	-	-	-	VDDH	M	
RXD1_copy	UART1 receive data (alternate)	I TBSHPW2	- -	-	-	-	VDDH	M	
RTS1_N	UART1 request to send	O TBS1HP	- -	oper	-	highz	VDDH	M	
RTS1_N_copy	UART1 request to send (alternate)	O TBSHPW1	- -	oper	-	highz	VDDH	M	
CTS1_N	UART1 clear to send	I TBS1HP	- -	-	-	-	VDDH	M	
DTR1_N	UART1 data terminal ready	O TBS1HP	- -	oper	-	highz	VDDH	M	
DSR1_N	UART1 data set ready	I TBS1HP	- -	-	-	-	VDDH	M	
UART2 Data Interface (UART2)									
TXD2	UART2 transmit data	O TBHMP	6 -	oper	-	highz	VDDL	M	
RXD2	UART2 receive data	I TBHMP	6 -	-	-	-	VDDL	M	
CTS2_N	UART2 clear to send	I TBHMP	6 -	-	-	-	VDDL	M	
Fast IrDA Controller (FIR)									
FIRTX	FIR transmit data	O TBHMP	6 -	oper	-	highz	VDDL	M	
FIRRX	FIR receive data	I TBHMP	6 -	-	-	highz	VDDL	M	
I²C Interfaces (IIC)									
SCL	I ² C-bus serial clock	I/O IICSL	3	Hi-Z	oper	-	func	VDDH	P
SDA	I ² C-bus data	I/O IICSDA	3	Hi-Z	oper	-	func	VDDH	P
Serial Peripheral Interface 1 (SPI1)									
SCLK1	SPI1 serial clock	I/O TBS1HP	- -	oper	-	func	VDDH	M	
SDATIN1	SPI1 serial data input	I TBSHPW2	- -	-	-	-	VDDH	M	
SDATIO1	SPI1 serial data input / output	I/O TBS1HP	- -	oper	-	func	VDDH	M	
SBUSY1	SPI1 device busy	I TBSHP	- -	-	-	-	VDDH	M	
SCS_N_LOAD1	SPI1 chip select in / load pulse	I/O TBSHP	- -	oper	-	func	VDDH	M	
Ethernet MAC 1 RMII interface									
ETN1TXD1	ETN1 Transmit Data [1]	O TBS1HP	- -	oper	-	-	VDDH	M	

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2] Reset state [5]	Sleep state	Power up	Tristate [3]	Supply [3]	Muxed [4]
ETN1TXD0	ETN1 Transmit Data [0]	O TBS1HP	-	-	oper	-	-	VDDH	M
ETN1TXEN	ETN1 Transmit Enable	O TBS1HP	-	-	oper	-	-	VDDH	M
ETN1RXER	ETN1 Receive Error	I TBS1HP	-	-	oper	-	-	VDDH	M
ETN1CRSDV	ETN1 Carrier Sense/Receive Data Valid	I TBS1HP	-	-	oper	-	-	VDDH	M
ETN1RXD1	ETN1 Receive Data [1]	I TBS1HP	-	-	oper	-	-	VDDH	M
ETN1RXD0	ETN1 Receive Data [0]	I TBS1HP	-	-	oper	-	-	VDDH	M
ETNREFCLK	ETN Reference Clock	I/O TBS1PD	-	oper	-	-	-	VDDH	D
ETNMDC	ETN PHY Management Clock	O TBSHPW2	-	-	oper	-	-	VDDH	M
ETNMDIO	ETN PHY DATA input/output	I/O TBSHPW2	-	-	oper	-	-	VDDH	M
Pulse Width Modulator (PWM)									
PWM3	pulse width modulator 3	O TBS1HP	-	-	oper	-	highz	VDDH	M
PWM2	pulse width modulator 2	O TBHMP	6	-	oper	-	highz	VDDL	M
Keyboard Scanner (KBS)									
KCOL7	keyboard column scan 7	I TBHMP	6	-	-	-	-	VDDL	M
KCOL6	keyboard column scan 6	I TBHMP	6	-	-	-	-	VDDL	M
KCOL5	keyboard column scan 5	I TBHMP	6	L weak	-	-	-	VDDL	P
KCOL4	keyboard column scan 4	I TBHMP	6	L weak	-	-	-	VDDL	P
KCOL3	keyboard column scan 3	I TBHMP	6	L weak	-	-	-	VDDL	P
KCOL2	keyboard column scan 2	I TBHMP	6	L weak	-	-	-	VDDL	P
KCOL1	keyboard column scan 1	I TBHMP	6	L weak	-	-	-	VDDL	P
KCOL0	keyboard column scan 0	I TBHMP	6	L weak	-	-	-	VDDL	P
KROW7	keyboard row scan 7	O TBHMP	6	-	oper	-	func	VDDL	M
KROW6	keyboard row scan 6	O TBHMP	6	-	oper	-	func	VDDL	M
KROW5	keyboard row scan 5	O TBHMP	6	H	oper	-	func	VDDL	P
KROW4	keyboard row scan 4	O TBHMP	6	H	oper	-	func	VDDL	P
KROW3	keyboard row scan 3	O TBHMP	6	H	oper	-	func	VDDL	P
KROW2	keyboard row scan 2	O TBHMP	6	H	oper	-	func	VDDL	P
KROW1	keyboard row scan 1	O TBHMP	6	H	oper	-	func	VDDL	P
KROW0	keyboard row scan 0	O TBHMP	6	H	oper	-	func	VDDL	P
External Interrupts (EXTINT)									
EXTINT23	external interrupt 23	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT22	external interrupt 22	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT21	external interrupt 21	I TBSHPW1	-	-	-	-	-	VDDH	M
EXTINT19	external interrupt 19	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT18	external interrupt 18	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT17	external interrupt 17	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT16	external interrupt 16	I TBS1HP	-	-	-	-	-	VDDH	M
EXTINT13	external interrupt 13	I TBS1HP	-	-	-	-	-	VDDH	M

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive state [1]	Level during [2]			Tristate [3]	Supply [5]	Muxed [4]
				Reset state [5]	Sleep state	Power up			
EXTINT12	external interrupt 12	I TBSHPW1	-	-	-	-	-	VDDH	M
EXTINT10	external interrupt 10	I TBSHP	-	-	-	-	-	VDDH	M
EXTINT7	external interrupt 7	I TBSHP	-	-	-	-	-	VDDH	M
EXTINT5	external interrupt 5	I TBSHPW2	-	-	-	-	-	VDDH	M
EXTINT2	external interrupt 2	I TBHMP	6	-	-	-	-	VDDL	M
EXTINT1	external interrupt 1	I TBHMP	6	-	-	-	-	VDDL	M
General Purpose I/O (GPIO)									
GPIOA31	general purpose I/O bank A	I/O TBSHPW2	-	H weak oper		H weak func	VDDH	P	
GPIOA30	general purpose I/O bank A	I/O TBSHPW2	-	H weak oper		H weak func	VDDH	P	
GPIOA29	general purpose I/O bank A	I/O TBS1HP	-	R weak oper			func	VDDH	P
GPIOA28	general purpose I/O bank A	I/O IICSDA	3	-	oper	-	func	VDDH	M
GPIOA27	general purpose I/O bank A	I/O IICSCL	3	-	oper	-	func	VDDH	M
GPIOA26	general purpose I/O bank A	I/O TBSHPW1	-	L weak oper		L weak func	VDDH	P	
GPIOA25	general purpose I/O bank A	I/O TBSHPW1	-	L weak oper		L weak func	VDDH	P	
GPIOA24	general purpose I/O bank A	I/O TBSHPW1	-	L weak oper		L weak func	VDDH	P	
GPIOA19	general purpose I/O bank A	I/O TBS1HP	-	R weak oper			func	VDDH	P
GPIOA18	general purpose I/O bank A	I/O TBSHPW1	-	-	oper	-	func	VDDH	M
GPIOA16	general purpose I/O bank A	I/O TBSHP	-	R weak oper		-	func	VDDH	P
GPIOA13	general purpose I/O bank A	I/O TBSHP	-	L weak oper		-	func	VDDH	P
GPIOA11	general purpose I/O bank A	I/O TBHMP	6	R weak oper		-	func	VDDL	P
GPIOA5	general purpose I/O bank A	I/O TBSHPW2	-	H weak oper		H weak func	VDDH	P	
GPIOA2	general purpose I/O bank A	I/O TBHMP	6	R weak oper		-	func	VDDL	P
GPIOA1	general purpose I/O bank A	I/O TBHMP	6	R weak oper		-	func	VDDL	P
GPIOB30	general purpose I/O bank B	I/O TBHMP_dualV6	-	oper	-		func	VDDM	M
GPIOB29	general purpose I/O bank B	I/O TBHMP_dualV6	-	oper	-		func	VDDM	M
GPIOB28	general purpose I/O bank B	I/O TBHMP_dualV6	-	oper	-		func	VDDM	M
GPIOB27	general purpose I/O bank B	I/O TBHMP_dualV6	-	oper	-		func	VDDM	M
GPIOB26	general purpose I/O bank B	I/O TBHMP_dualV6	H	oper	-		func	VDDM	P
GPIOB25	general purpose I/O bank B	I/O TBHMP_dualV6	H weak oper		-		func	VDDM	P
GPIOB24	general purpose I/O bank B	I/O TBHMP_dualV6	L weak oper		-		func	VDDM	P
GPIOB23	general purpose I/O bank B	I/O TBHMP	6	L weak oper		-	func	VDDL	P
GPIOB22	general purpose I/O bank B	I/O TBHMP	6	H weak oper		-	func	VDDL	P
GPIOB21	general purpose I/O bank B	I/O TBHMP	6	H weak oper		-	func	VDDL	P
GPIOB20	general purpose I/O bank B	I/O TBHMP	6	L weak oper		-	func	VDDL	P
GPIOB19	general purpose I/O bank B	I/O TBHMP	6	L weak oper		-	func	VDDL	P
GPIOB18	general purpose I/O bank B	I/O TBHMP_dualV6	-	oper	-		func	VDDM	M
GPIOB11	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB10	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB9	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2]			Tristate [3]	Supply [3]	Muxed [4]
				Reset state [5]	Sleep state	Power up			
GPIOB8	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB7	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB6	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB5	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB4	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB3	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB2	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB1	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOB0	general purpose I/O bank B	I/O TBHMP	6	-	oper	-	func	VDDL	M
GPIOC30	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC29	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC28	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC27	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC26	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC25	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC23	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC18	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC17	general purpose I/O bank C	I/O TBHMP_dualV	6	-	oper	-	func	VDDM	M
GPIOC16	general purpose I/O bank C	I/O TBSHPW2	-	H weak	oper	H weak	func	VDDH	P
GPIOC15	general purpose I/O bank C	I/O TBSHPW2	-	H weak	oper	H weak	func	VDDH	P
GPIOC14	general purpose I/O bank C	I/O TBSHP	-	H weak	oper	-	func	VDDH	P
GPIOC12	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC6	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC5	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC4	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC3	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC2	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC1	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
GPIOC0	general purpose I/O bank C	I/O TBS1HP	-	H weak	oper	-	func	VDDH	P
USB Device (USB)									
USBDP	USB differential data plus	I/O USB	-	Hi-Z	oper	-	func	VDDH	D
USBDM	USB differential data minus	I/O USB	-	Hi-Z	oper	-	func	VDDH	D
USBVBUS	USB VBUS indicator	I TBSHP	-	L weak	-	-	-	VDDH	M
USBCN	USB external pullup activation for soft-connect	O TBSHPW1	-	-	oper	-	highz	VDDH	M
USBLED_N	USB GoodLink™ indicator	O TBSHPW2	-	-	oper	-	highz	VDDH	M
IIS-bus Interface									
IISD_copy	I ² S-bus serial data	O TBSHPW1	-	-	oper	-	highz	VDDH	M
IISWS_copy	I ² S-bus word select	O TBSHPW1	-	-	oper	-	highz	VDDH	M

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2] Reset state [5]	Sleep state	Power up	Tristate [3]	Supply [3]	Muxed [4]
IISCK_copy	I ² S-bus serial clock	O TBSHPW1	-	-	oper	-	highz	VDDH	M
Watchdog and Reset Unit (WDRU)									
RSTBB_N	DPU reset	I IHMU	6	H weak	oper	-	highz	VDDL	D
JTAG Test Access Port 1 (TAP1)									
JSEL0	TAP mode selector	I IHMU	6	H weak	-	-	-	VDDL	D
JSEL1	TAP mode selector	I IHMD	6	L weak	-	-	-	VDDL	D
TDI	JTAG data input	I IHMU_nbs	6	H weak	H weak	-	-	VDDL	D
TDO	JTAG data output	O TOM	6	R weak	oper	-	-	VDDL	D
TCK	JTAG clock	I IHMU_nbs	6	H weak	H weak	-	-	VDDL	D
RTCK	JTAG return clock	O TOM_fast	6	L	oper	-	-	VDDL	D
TMS	JTAG mode select	I IHMU_nbs	6	H weak	H weak	-	-	VDDL	D
TRST_N	JTAG reset	I IHMD_nbs	6	L weak	L weak	-	-	VDDL	D
JTAG Test Access Port 2 (TAP2)									
TDI2	JTAG data input	I TBHMP	6	-	-	-	-	VDDL	M
TDO2	JTAG data output	O TBHMP	6	-	oper	-	-	VDDL	M
TCK2	JTAG clock	I TBHMP	6	-	-	-	-	VDDL	M
TMS2	JTAG mode select	I TBHMP	6	-	-	-	-	VDDL	M
Miscellaneous Signals									
FINT0_SC	SC frame interrupt 0	O TBSHPW2	-	-	oper	-	highz	VDDH	M
FINT0_SC_copy	SC frame interrupt 0 (alternate)	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
FRAME_CLOCK	TBU frame clock	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
IRQ_N	ARM interrupt	O TBSHPW1	-	-	oper	-	highz	VDDH	M
FIQ_N	ARM fast interrupt	O TBSHPW1	-	-	oper	-	highz	VDDH	M
SC_IRQ19	PIO interrupt from DSP to SC	O TBS1HP	-	-	oper	-	highz	VDDH	M
Hardware Debug Port (HDP)									
HDPA11	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPA10	hardware debug port	O TBHM_dualV	6	-	oper	-	highz	VDDM	M
HDPA9	hardware debug port	O TBHM_dualV	6	-	oper	-	highz	VDDM	M
HDPA8	hardware debug port	O TBHM_dualV	6	-	oper	-	highz	VDDM	M
HDPA7	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPA6	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPA5	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPA4	hardware debug port	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA3	hardware debug port	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA2	hardware debug port	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA1	hardware debug port	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA0	hardware debug port	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA12_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA11_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2] Reset state [5]	Sleep state	Power up	Tristate [3]	Supply [3]	Muxed [4]
HDPA10_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA9_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA8_copy	hardware debug port (alternate)	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPA7_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA6_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA5_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA4_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA3_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA2_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA1_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPA0_copy	hardware debug port (alternate)	O TBHMP	6	-	oper	-	highz	VDDL	M
HDPB27	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB26	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB25	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB24	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB23	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB22	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB21	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB20	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB16	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB15	hardware debug port	O TBHMP_dualV	6	-	oper	-	highz	VDDM	M
HDPB14	hardware debug port	O TBSHPW2	-	-	oper	-	highz	VDDH	M
HDPB13	hardware debug port	O TBSHPW2	-	-	oper	-	highz	VDDH	M
HDPB12	hardware debug port	O TBSHP	-	-	oper	-	highz	VDDH	M
HDPB12_copy	hardware debug port (alternate)	O TBSHP	-	-	oper	-	highz	VDDH	M
HDPB10	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
HDPB4	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
HDPB3	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
HDPB2	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
HDPB1	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
HDPB0	hardware debug port	O TBS1HP	-	-	oper	-	highz	VDDH	M
DAIF interface									
IF_BAT_OK	Input from low voltage detector logic. Interrupts the ARM with <i>int_pwr_fail</i> at both rising and falling edges.	I IH	6	-	-	-	-	VDDINT	
IF_CDC_DAT_AD	Data signal from the Sigma-Delta converters (ATC) on the APU to the Digital Decimating Filters (DDF).	I IH	6	-	-	-	-	VDDINT	

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2] [5]	Reset state	Sleep state	Power up	Tristate [3]	Supply [3]	Muxed [4]
IF_CDC_DAT_DA	Data signal from the Digital Noise Shapers (DNS) to the DA converters (ARD) on the APU.	O TBSH	6	-	-	-	-	-	VDDINT	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
IF_CDC_DIR	Synchronisation and direction control signal of the audio bit stream frames.	O TBSH	6	-	-	-	-	-	VDDINT	
IF_CLK_IF	clock for data transmission on the 4-wire interface (equal to clk13m)	O TBSH	6	-	-	-	-	-	VDDINT	
IF_CLK_O	12 times oversampled clock of IF_T_DATA	O TBSH	6	-	-	-	-	-	VDDINT	
BBCKI	Baseband main clock input	I IH	6	-	-	-	-	-	VDDINT	
IF_DATA_AD	4 wire interface to control the APU.	I IH	6	-	-	-	-	-	VDDINT	
IF_DATA_DA		O TBSH	6	-	-	-	-	-	VDDINT	
IF_EN_AD		I IH	6	-	-	-	-	-	VDDINT	
IF_EN_DA		O TBSH	6	-	-	-	-	-	VDDINT	
IF_INT_TS	Interrupt signal from APU (touchscreen, voice monitor, on-off, charge detect)	I IH	6	-	-	-	-	-	VDDINT	
RSTAPU	APU reset output from WDRU	O TBSH	6	-	-	-	-	-	VDDINT	
IF_NPORST	Power on reset input of WDRU from APU	I IH	6	-	-	-	-	-	VDDINT	
IF_R_DATA_MOD	RF receive data signals from ABS (OM6115) or in-phase components from RFAPU	I IH	6	-	-	-	-	-	VDDINT	
IF_T_DATA	RF transmit data serial output from BMP.	O TBSH	6	-	-	-	-	-	VDDINT	
Power supply and ground pins										
VDDC	DPU core power supply									
VDDCGU	DPU CGU digital power supply									
VDDACGU	DPU PLL power supply									
VDDL	DPU low voltage pad power supply									
VDDH	DPU high voltage pad power supply									
VDDH	DPU USB pad power supply									
VDDM	DPU HMP voltage selectable domain power supply									
	DPU FMP voltage selectable domain power supply									
	DPU special GPIO voltage selectable domain power supply									
VDDINT	APU and DPU die interface power supply									
VSS	DPU core ground									
	DPU pad ground									

Table 7: DVFD8185 & DVFD8187 pinning grouped by functional blocks ...continued

Signal	Description	I/O Pad type	Drive [1]	Level during [2] Reset state [5]	Level during [2] Sleep state	Level during [2] Power up	Tristate [3]	Supply [3]	Muxed [4]
VSSACGU	DPU PLL analog ground								
VSSCGU	DPU PLL digital ground								
VSSINT	APU and DPU die interface ground								

[1] Output drive is in mA.

[2] Explanation of symbols: L weak means pull-down behavior, H weak means pull-up behavior, R weak means repeater (reset value can be either high or low), Hi-Z means high impedance, H means high level, L means low level, oper means that level is not changed because of state transition.

[3] Explanation of symbols: highz means that pad can be set to tristate by JTAG HIGHZ command only, func means that pad can be set to high impedance by JTAG HIGHZ command and by a functional behavior.

[4] Explanation of symbols: D means direct connection, P means primary connection (multiplexed active after reset), M means multiplexed connection not active after reset.

[5] Pins which are operating in reset state are not affected by RSTBB active.

[6] It is mandatory to connect all supply and ground balls (having the same name) together in the application PCB.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 8: CMOS090 pad characteristic description [1]

Pad type	Power domain	Description	Resistance	Cell mapping	Library mapping
APIO	VDDINT	Analog Input / Output pad	-	api03v3	jliolib_p50_io3v3
APIO noESD_V DDE	VDDINT	Analog Input / Output pad without ESD protection diode to power domain	-	api03v3_noESD_vdde	modified jliolib_p50_io3v3
IH	VDDINT	Bidirectional dual voltage pad used as input at 2.5V, ESH set to 1 (low speed mode).	-	mem1bsptz40p	jliolib_p50_mem1chp
IHM	VDDL	Bidirectional dual voltage pad used as input at 1.8 V, ESH set to 1 (low speed mode).	-	mem1bsptz40p	jliolib_p50_mem1chp
IHMD	VDDL	Bidirectional dual voltage pad used as input at 1.8 V, ESH set to 1 (low speed mode).	pull-down	mem1bsptz40p	jliolib_p50_mem1chp
IHMD_nbs	VDDL	Bidirectional dual voltage pad used as input at 1.8 V, pull-down, no boundary scan, ESH set to 1 (low speed mode).	pull-down	mem1bsptz40p	jliolib_p50_mem1chp
IHMU	VDDL	Bidirectional dual voltage pad used as input at 1.8 V, pull-up, ESH set to 1 (low speed mode).	pull-up	mem1bsptz40p	jliolib_p50_mem1chp
IHMU_nbs	VDDL	Bidirectional dual voltage pad used as input at 1.8 V, pull-up, no boundary scan, ESH set to 1 (low speed mode).	pull-up	mem1bsptz40p	jliolib_p50_mem1chp
IICSL	VDDH	Open drain bidirectional pad buffer, Schmitt trigger input, slew rate controlled, for SCL standard or fast mode	-	iic3m4scl	jliolib_p50_io3v3
IICSDA	VDDH	Open drain bidirectional pad buffer, Schmitt trigger input, slew rate controlled, for SDA standard or fast mode	-	iic3m4sda	jliolib_p50_io3v3
TBHM_du alV	VDDM	Tristate bidirectional dual voltage pad (1.8 V or 2.8 V), fixed programming as repeater.	-	mem1bsptz40p	jliolib_p50_mem1chp
TBHMP	VDDL	Tristate bidirectional dual voltage pad used at 1.8 V, programmable pull down / pull up / repeater / plain input, ESH set to 1 (low speed mode).	yes/no, up/down	mem1bsptz40p	jliolib_p50_mem1chp
TBHM_d ualV	VDDM	Tristate bidirectional dual voltage pad (1.8 V or 2.8 V), programmable pull down / pull up / repeater / plain input.	yes/no, up/down	mem1bsptz40p	jliolib_p50_mem1chp
TBS1HP	VDDH	Tristate bidirectional, programmable pull down / pull up / repeater / plain input, 1ns slew rate.	yes/no, up/down	bspts1chp	jliolib_p50_mem1
TBS1PD	VDDH	Tristate bidirectional, programmable pull down, 1 ns slew rate.	yes/no, down	bspts1cp	jliolib_p50_mem1
TBSH	VDDINT	Tristate bidirectional dual voltage pad (1.8 V or 2.8 V), programmable pull down / pull up / repeater / plain input, ESH set to 1 (low speed at 1.8 V, high speed at 2.8 V)	yes/no, up/down	mem1bsptz40p	jliolib_p50_mem1chp
TBSHP	VDDH	Tristate bidirectional pad, programmable pull up / pull down / repeater / plain input, 3 ns slew rate.	yes/no, up/down	bspts3chp	jliolib_p50_mem1

Table 8: CMOS090 pad characteristic description [1]...continued

Pad type	Power domain	Description	Resistance	Cell mapping	Library mapping
TBSHPW 1	VDDH	Tristate bidirectional pad, programmable pull up / pull down / repeater / plain input, 3 ns slew rate. TARA Pad : delivers L weak at powerup and retains configuration at powerdown (no core power, only IO power).	yes/no, up/down	bspts3chmpdd	jliolib_p50_io3v3 md
TBSHPW 2	VDDH	Tristate bidirectional pad, programmable pull up / pull down / repeater / plain input, 3 ns slew rate. TARA Pad : delivers H weak at powerup and retains configuration at powerdown (no core power, only IO power)	yes/no, up/down	bspts3chmpdu	jliolib_p50_io3v3 md
TOM	VDDL	Bidirectional dual voltage pad used as output at 1.8 V, repeater, ESH set to 1 (low speed mode).		mem1bsptz40p chp	jliolib_p50_mem1
TOM_dual V	VDDM	Tristate bidirectional dual voltage pad (1.8 V or 2.8 V) used as output, fixed programming as repeater.		mem1bsptz40p chp	jliolib_p50_mem1
TOM_fast	VDDL	Bidirectional dual voltage pad used as output at 1.8 V, repeater, ESH set to 0 (high speed mode).		mem1bsptz40p chp	jliolib_p50_mem1
USB	VDDH	USB analog transceiver pad	-	usb11f3t5v	jliolib_p50_usb

[1] Please find the detailed pad characteristic specification in [Section 13](#).

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

6.3 FMP - Fast Memory Port

6.3.1 Features

- 16-bit and 32-bit multiplexed static memory controller (EBI1)
- Up to 256 Mbyte address range and 104 MHz operation for static memory access
- Interface operates at 1.8V or 2.8V supply voltage (VDDM)

6.3.2 Block diagram

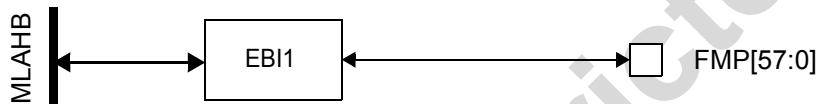


Fig 5. FMP block diagram (Note: Not all FMP pins are available)

6.3.3 Hardware interface

Table 9: FMP pin overview

Pin	Name	I/O	Description
FMP[57:55]	FMP outputs	O	FMP outputs
FMP54	FMP input	I	FMP input
FMP[53:32]	FMP outputs	O	FMP outputs
FMP[31:0]	FMP bidirectionals	I/O	FMP bidirectionals

6.3.4 Software interface

The FMP has no configuration registers and no function to be configured.

6.3.5 Functional description

The FMP should be used to connect fast memories to profit from the high bandwidth. However also any kind of memory mapped peripheral can be connected. Memory devices should be physically located as close as possible to the DVFD818x in order to keep wiring on the PCB as short as possible. If possible, combos should be used to reduce the load on the bus.

Table 10: FMP pin allocation

FMP pins	EBI1 pins	I/O
FMP54	WAIT	I
FMP53	CLKBURST	O
FMP50	<u>OE_R/W</u>	O
FMP49	<u>WE_E</u>	O
FMP48	<u>CS0</u>	O
FMP47	<u>CS1</u>	O
FMP46	<u>CS3</u>	O
FMP45	ADV	O
FMP44	BAA	O
FMP[43:34] ^[1]	ADD[27:18]	O
FMP33	ADD1_BE2	O
FMP[15:0]	IO[15:0]	I/O

[1] FMP43 & FMP41 are not available

6.3.6 Application information

On the DVFD818x there is a single supply rail (VDDM) for all FMP pins.

6.4 GPM - GPIO Pin Multiplexing

6.4.1 Features

- Multiplexing primary pins on 3 GPIO banks

6.4.2 Block diagram

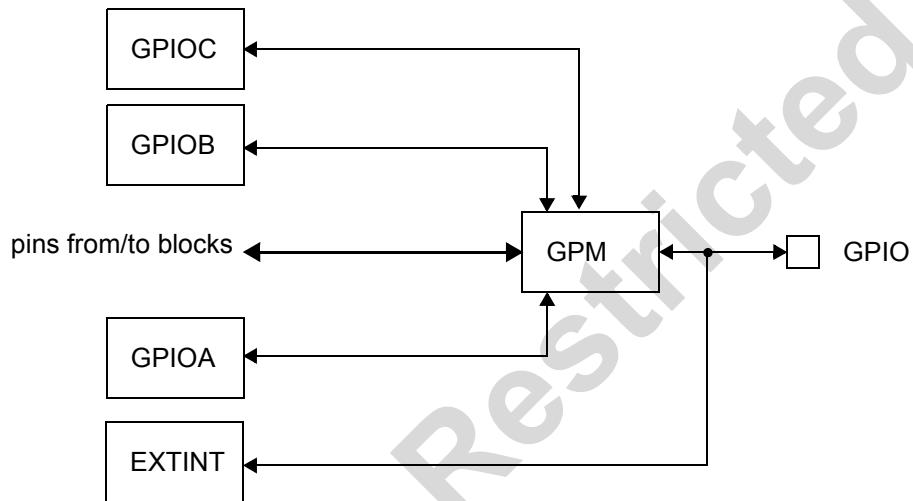


Fig 6. GPM block diagram

6.4.3 Hardware interface

Table 11: GPM pin overview^[1]

Pin	Name	I/O	Description
GPIOC[31:0]	GPIOC bidirectionals	I/O	GPIOC bidirectionals
GPIOB[31:0]	GPIOB bidirectionals	I/O	GPIOB bidirectionals
GPIOA[31:0]	GPIOA bidirectionals	I/O	GPIOA bidirectionals

[1] Not all pins are available

6.4.4 Functional description

This section describes the implemented pin functional multiplexing which allows flexible access to the internal functions of the IC and to customize the IC interface depending on the application.

Table 12: GPIOA pin multiplexing descriptionSignals in **bold** are selected after reset (See **SCON.sysmux** description for details)

Programming field [1]	Multiplexing value				EXTINT [2]
	0b00 [2]	0b01	0b10	0b11	
gpioa31c	GPIOA31	USBLED_n	SDATIN1 [3]	FINT0_SC	-
gpioa30c	GPIOA30	-	-	DI_IP	-
gpioa29c	GPIOA29	-	SC_IRQ19	SDATIO1 [3]	-
gpioa28c	GPIOA28 [6]	SDA [4]	-	-	-
gpioa27c	GPIOA27 [6]	SCL [4]	-	-	-
gpioa26c	GPIOA26	IISWS_copy	DO_IP	FIQ_n	-
gpioa25c	GPIOA25	IISSD_copy	FSC_IP	-	EXTINT21
gpioa24c	GPIOA24	IISCK_copy	DCK_IP	IRQ_n	-
gpioa19c	GPIOA19	-	PWM3	SCLK1 [3]	EXTINT13
gpioa18c	GPIOA18	SIMPWR [8]	USBCN	RTS1_n_copy	EXTINT12
gpioa16c	GPIOA16	HDPB12_copy	SCS_n_LOAD1 [4]	SBUSY1 [3]	EXTINT10
gpioa13c	GPIOA13	DGPO2	-	-	EXTINT7
gpioa11c	GPIOA11	TXD2	FIRTX	-	-
gpioa5c	GPIOA5	SYNCPORT	SIMOFF_n_copy [4][5]	RXD1_copy [4][7]	EXTINT5
gpioa2c	GPIOA2	DMAUCLR	CTS2_n [3]	FIRRX [3]	EXTINT2
gpioa1c	GPIOA1	RXD2 [4]	PWM2	-	EXTINT1

[1] Programming these fields into one of the unspecified values is not permitted and may provoke faulty IC behavior.

[2] GPIO input and EXTINT are always connected to the pad regardless the multiplexing value.

[3] Input internally forced to 0 if not selected by multiplexer.

[4] Input internally forced to 1 if not selected by multiplexer.

[5] It is not allowed to select both SIMOFF at the same time - the circuit would behave in an unpredictable manner.[6] I²C pins have only Open-Drain capabilities, no Push-Pull[7] It is not allowed to select both RXD1 (see also **Table 14**) at the same time - the circuit would behave in an unpredictable manner.

[8] Don't use this selection with DVFD818x. If it is selected at reset, the software has to change it during initialisation phase of the hardware. The reset level has to be taken into account for external circuit design.

Table 13: GPIOB pin multiplexing descriptionSignals in **bold** are selected after reset (See *SCON.sysmux* description for details)

Programming field [1]	Multiplexing value				EXTINT [2]
	0b00	0b01 [2]	0b10	0b11	
gpiob30c	HDPA11	GPIOB30	FMP45	-	-
gpiob29c	HDPA10	GPIOB29	HMP30	-	-
gpiob28c	HDPA9	GPIOB28	HMP29 [3]	-	-
gpiob27c	HDPA8	GPIOB27	HMP28	-	-
gpiob26c	HDPA7	GPIOB26	FMP42	-	-
gpiob25c	HDPA6	GPIOB25	FMP54 [4]	FMP44	-
gpiob24c	HDPA5	GPIOB24	FMP53 [3]	-	-
gpiob23c	HDPA4	GPIOB23	SYNC_MTCH [6]	-	-
gpiob22c [5]	HDPA3	GPIOB22	KROW7	TMS2 [4]	-
gpiob21c [5]	HDPA2	GPIOB21	KROW6	TCK2 [4]	-
gpiob20c [5]	HDPA1	GPIOB20	KCOL7 [3]	TDO2	-
gpiob19c [5]	HDPA0	GPIOB19	KCOL6 [3]	TDI2 [4]	-
gpiob18c	FMP40	GPIOB18	FRAME_CLOCK	FINT0_SC_copy	-
gpiob11c	KROW5	GPIOB11	HDPA12_copy	-	-
gpiob10c	KROW4	GPIOB10	HDPA11_copy	-	-
gpiob9c	KROW3	GPIOB9	HDPA10_copy	-	-
gpiob8c	KROW2	GPIOB8	HDPA9_copy	-	-
gpiob7c	KROW1	GPIOB7	HDPA7_copy	-	-
gpiob6c	KROW0	GPIOB6	HDPA6_copy	-	-
gpiob5c	KCOL5 [3]	GPIOB5	HDPA5_copy	-	-
gpiob4c	KCOL4 [3]	GPIOB4	HDPA4_copy	-	-
gpiob3c	KCOL3 [3]	GPIOB3	HDPA3_copy	-	-
gpiob2c	KCOL2 [3]	GPIOB2	HDPA2_copy	-	-
gpiob1c	KCOL1 [3]	GPIOB1	HDPA1_copy	-	-
gpiob0c	KCOL0 [3]	GPIOB0	HDPA0_copy	-	-

[1] Programming these fields into one of the unspecified values is not permitted and may provoke faulty IC behavior.

[2] GPIO input and EXTINT are always connected to the pad regardless the multiplexing value.

[3] Input internally forced to 0 if not selected by multiplexer.

[4] Input internally forced to 1 if not selected by multiplexer.

[5] If the JTAG test access port pins JSEL are set to 0b10, the JTAG interface pins are selected independent of the setting of this field

[6] Don't use this selection with DVFD8187. If it is selected at reset, the software has to change it during initialisation phase of the hardware. The reset level has to be taken into account for external circuit design.

Table 14: GPIOC pin multiplexing descriptionSignals in **bold** are selected after reset (See **SCON.sysmux** description for details)

Programming field [1]	Multiplexing value				EXTINT [2]
	0b00	0b01 [2]	0b10	0b11	
gpioc30c	FMP50	GPIOC30	-	-	-
gpioc29c	FMP46	GPIOC29	HDPB27	HDPA8_copy	-
gpioc28c	FMP39	GPIOC28	HDPB26	-	-
gpioc27c	FMP38	GPIOC27	HDPB25	-	-
gpioc26c	FMP37	GPIOC26	HDPB24	-	-
gpioc25c	FMP36	GPIOC25	HDPB23	-	-
gpioc24c	FMP35	GPIOC24	HDPB22	-	-
gpioc23c	FMP34	GPIOC23	HDPB21	-	-
gpioc22c	FMP33	GPIOC22	HDPB20	-	-
gpioc18c	HMP27	GPIOC18	HDPB16	-	-
gpioc17c	HMP26	GPIOC17	HDPB15	-	-
gpioc16c	ETNMDC	GPIOC16	HDPB14	-	-
gpioc15c	ETNMDIO	GPIOC15	HDPB13	-	-
gpioc14c	USBVBUS	GPIOC14	HDPB12	-	-
gpioc12c	ETN2TXD1	GPIOC12	HDPB10	FCIDATA3[3]	-
gpioc6c	ETN1TXEN	GPIOC6	HDPB4	-	-
gpioc5c	ETN1RXER	GPIOC5	HDPB3	DSR1_n[4]	EXTINT23
gpioc4c	ETN1CRSDV1	GPIOC4	HDPB2	CTS1_n[3][6]	EXTINT22
gpioc3c	ETN1RXD1	GPIOC3	HDPB1	DTR1_n	EXTINT19
gpioc2c	ETN1RXD0	GPIOC2	HDPB0	TXD1	EXTINT18
gpioc1c	ETN1TXD1	GPIOC1	TXD1_copy	RXD1[4][5]	EXTINT17
gpioc0c	ETN1TXD0	GPIOC0		RTS1_n	EXTINT16

[1] Programming these fields into one of the unspecified values is not permitted and may provoke faulty IC behavior.

[2] GPIO input and EXTINT are always connected to the pad regardless the multiplexing value.

[3] Input internally forced to 0 if not selected by multiplexer.

[4] Input internally forced to 1 if not selected by multiplexer.

[5] It is not allowed to select both RXD1 (see also [Table 12](#)) at the same time - the circuit would behave in an unpredictable manner.[6] It is not allowed to select both CTS1 (see also [Table 12](#)) at the same time - the circuit would behave in an unpredictable manner.

The following notes apply to the pin multiplexing functionality:

- Output pin multiplexing is always a selection between several internal signals.
- Input pin multiplexing depends on the functionality of the pins: only one block input is selected, and the non-selected are either forced to 1 or to 0.
- The EXTINT input channels and the GPIO inputs are connected to the pad also when not selected by the multiplexer.

6.4.4.1 Pin multiplexing example

Figure 7 details the pin multiplexing on the example of the GPIOA1 pin. The signal **gpioa1_in** always represents the physical level available at the output GPIOA1 independent if it is selected or not. The signal **rx2d_in** forced to 1 when not selected. This figure does not show the JTAG high-z control nor any boundary scan details.

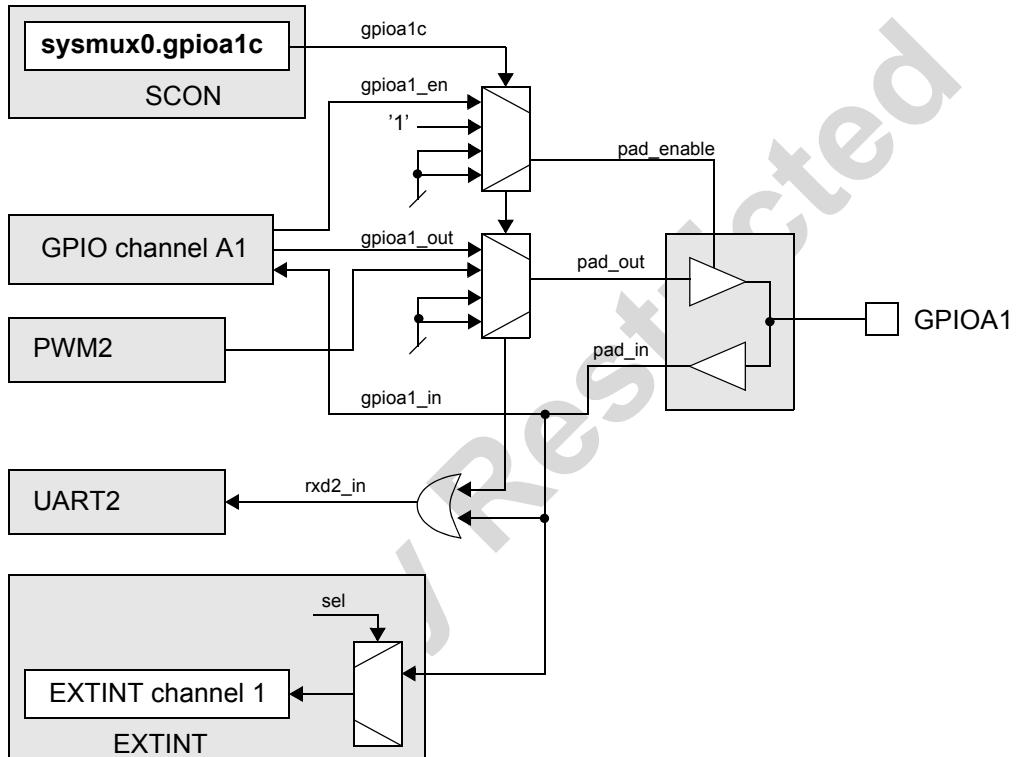


Fig 7. Example for output and input multiplexing including an EXTINT channel

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

6.5 HDP - Hardware Debug Port

6.5.1 Features

- Multiplexing of internal functionality for debug purpose (limited support provided).

6.5.2 Block diagram

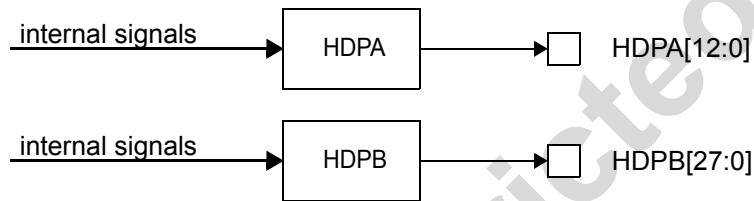


Fig 8. HDP block diagram

6.5.3 Hardware interface

Table 15: HDP pin overview

Pin	Name	I/O	Description
HDPA[12:0]	HDPA outputs	O	HDPA outputs
HDPB[27:0]	HDPB outputs	O	HDPB outputs

6.5.4 Functional description

The debug pin multiplexing allows the user to observe many internal signals and to use some debug functionality like the ETM9 and the TSR.

The HDP function is controlled by SCON block. The activation of the HDP function can be performed as follow:

- set bit **syscon0.hdpn** to 1
- select the proper HDP multiplex value (see Table 16 to 23) thanks to **hdpacon** and **hpdbcon** fields of **syscon0**
- select the HDP function on GPIO multiplexer (see Section 6.4)

6.5.4.1 HDPA

Table 16: Debug pin multiplexing description on HDPA (set 0 to 4)

HDP pin name	multiplex value				
	0 ETM	1 ARM I	2 ETN1	3 DMAU1	4 BMP Correlator Testsignals
HDPA0	TRCPKT0/TRCPKTA0	arm_i_hclk	etn1_hclk	dmau1_hclk	-
HDPA1	TRCPKT1/TRCPKTA1	arm_i_hwrite	etn1_hwrite	dmau1_hwrite	sync_mtch
HDPA2	TRCPKT2/TRCPKTA2	arm_i_hburst0	etn1_hburst0	dmau1_hburst0	bmp_sym_rec_rx_bitclk
HDPA3	TRCPKT3/TRCPKTA3	arm_i_hburst1	etn1_hburst1	dmau1_hburst1	bmp_sym_rec_rx_data
HDPA4	TRCCLK	arm_i_hburst2	etn1_hburst2	dmau1_hburst2	corr_sync_time_i
HDPA5	TRCSYNC/TRCSYNCA	arm_i_htrans0	etn1_htrans0	dmau1_htrans0	sym_sync_time_i
HDPA6	PIPESTAT0/PIPESTAT0A	arm_i_htrans1	etn1_htrans1	dmau1_htrans1	-
HDPA7	PIPESTAT1/PIPESTAT1A	arm_i_hresp0	etn1_hresp0	dmau1_hresp0	-
HDPA8	PIPESTAT2/PIPESTAT2A	arm_i_hready	etn1_hready	dmau1_hready	-
HDPA9	TRCPKT4	arm_i_hprot0	etn1_hprot0	dmau1_hprot0	-
HDPA10	TRCPKT5	arm_i_hprot1	etn1_hprot1	dmau1_hprot1	-
HDPA11	TRCPKT6	arm_i_hprot2	etn1_hprot2	dmau1_hprot2	-

Table 17: Debug pin multiplexing description on HDPA (set 5 to 9)

HDP pin name	multiplex value				
	5 DRT test signals	6 AHB ready	7 SDI0	8 EBI1 [1]	9 VPB bridge
HDPA0	clk_drt	ahb_hclk	sdi0_hclk	ebi1_hclk	vpb_hclk
HDPA1	IF_CLK_IF	ahb_hready_etn1	sdi0_hwrite	ebi1_hwrite	vpb_hwrite
HDPA2	clk_fs4	ahb_hready_etn2	sdi0_hburst0	ebi1_hburst0	-
HDPA3	int_fsi	ahb_hready_dmau1	sdi0_hburst1	ebi1_hburst1	-
HDPA4	IF_CDC_DIR	ahb_hready_dmau2	sdi0_hburst2	ebi1_hburst2	-
HDPA5	dtb_test_out[0]	-	sdi0_htrans0	ebi1_htrans0	vpb_htrans0
HDPA6	dtb_test_in[0]	-	sdi0_htrans1	ebi1_htrans1	vpb_htrans1
HDPA7	dtb_test_out[1]	-	sdi0_hsize1	ebi1_hsize1	vpb_hsize1
HDPA8	dtb_test_in[1]	-	sdi0_hready	ebi1_hready	vpb_hready
HDPA9	cdc_dat	-	sdi0_hprot0	ebi1_hprot0	vpb_hprot0
HDPA10	-	ahb_hready_armi	sdi0_hprot1	ebi1_hprot1	vpb_hprot1
HDPA11	-	ahb_hready_armd	sdi0_hprot2	ebi1_hprot2	vpb_hprot2

[1] EBI signal monitoring applies to the EBI data port only. Configuration register access is not supported.

Table 18: Debug pin multiplexing description on HDPA (set 10 to 14)

HDP pin name	multiplex value				
	10 IRQ set1	11 IRQ set2	12 DMA	13 ETN2	14 BMP test signals
HDPA0	IRQ	IRQ	hclk	etn2_hclk	clk_adpcm
HDPA1	intr1	intr13	dmabreq0	etn2_hwrite	clk_drt
HDPA2	intr2	intr14	dmabreq1	etn2_hburst0	clk_o
HDPA3	intr3	intr15	dmabreq2	etn2_hburst1	clk_bit
HDPA4	intr4	intr16	dmabreq3	etn2_hburst2	tdata / demod_afc
HDPA5	intr5	intr17	dmabreq4	etn2_htrans0	int_fsi
HDPA6	intr6	intr18	dmabreq5	etn2_htrans1	rdata_pll/rdata_demod
HDPA7	intr7	intr19	dmabreq6	etn2_hresp0	symbol_ind
HDPA8	intr8	intr20	dmabreq7	etn2_hready	mod_i
HDPA9	intr9	intr21	dmabreq8	etn2_hprot0	mod_q
HDPA10	intr10	intr22	dmabreq9	etn2_hprot1	-
HDPA11	intr11	intr23	dmabreq10	etn2_hprot2	-

[1] DPU input signal is provided.

[2] DPU output signal is provided.

Table 19: Debug pin multiplexing description on HDPA (set 15 to 17)

HDP pin name	multiplex value		
	15 BMP Vector	16 DSP	17 DAIF
HDPA0	bmp_vec0	dsp_pa0	IF_CDC_DAT_AD
HDPA1	bmp_vec1	dsp_pa1	IF_CDC_DAT_DA
HDPA2	bmp_vec2	dsp_pa2	IF_CDC_DIR
HDPA3	bmp_vec3	dsp_pa3	IF_CLK_IF
HDPA4	bmp_vec4	dsp_pa4	IF_DATA_AD
HDPA5	bmp_vec5	dsp_pa5	IF_DATA_DA
HDPA6	bmp_vec6	dsp_pa6	IF_EN_AD
HDPA7	bmp_vec7	dsp_pa7	IF_EN_DA
HDPA8	-	dsp_pa8	IF_CLK_O
HDPA9	-	dsp_pa9	IF_T_DATA
HDPA10	-	dsp_pa10	IF_R_DATA
HDPA11	-	dsp_pa11	IF_SLC_CTRL_DA

6.5.4.2 HDPB

Table 20: Debug pin multiplexing description on HDPB (set 0 to 4)

HDPBpin name	multiplex value				
	0 ETM	1 ARM D / SCRAM0	2 DMAU2 / SCRAM1	3 Reserved	4 Reserved
HDPB0	TRCPKT0/TRCPKTA0	arm_d_hclk	dmau2_hclk	-	-
HDPB1	TRCPKT1/TRCPKTA1	arm_d_hwrite	dmau2_hwrite	-	-
HDPB2	TRCPKT2/TRCPKTA2	arm_d_hburst0	dmau2_hburst0	-	-
HDPB3	TRCPKT3/TRCPKTA3	arm_d_hburst1	dmau2_hburst1	-	-
HDPB4	TRCPKT4	arm_d_hburst2	dmau2_hburst2	-	-
HDPB10	PIPESTAT0/PIPESTATA0	arm_d_hprot1	dmau2_hprot1	-	-
HDPB12	PIPESTAT2/PIPESTATA2	arm_d_hprot3	dmau2_hprot3	-	-
HDPB13	TRCPKT8/PIPESTATB0	scram0_hwrite	scram1_hwrite	-	-
HDPB14	TRCPKT9/PIPESTATB1	scram0_hburst0	scram1_hburst0	-	-
HDPB15	TRCPKT10/PIPESTATB2	scram0_hburst1	scram1_hburst1	-	-
HDPB16	TRCPKT11/TRCSYNCB	scram0_hburst2	scram1_hburst2	-	-
HDPB20	TRCPKT15/TRCPKTB3	scram0_hready	scram1_hready	-	-
HDPB21		scram0_hprot0	scram1_hprot0	-	-
HDPB22	portmode[0]	scram0_hprot1	scram1_hprot1	-	-
HDPB23	portmode[1]	scram0_hprot2	scram1_hprot2	-	-
HDPB24	TRCPKT4	scram0_hprot3	scram1_hprot3	-	-
HDPB25	TRCPKT5	-	-	-	-
HDPB26	TRCPKT6	-	-	-	-
HDPB27	TRCPKT7	-	-	-	-

Companion document

Table 21: Debug pin multiplexing description on HDPB (set 5 to 9)

HDPBpin name	multiplex value				
	5 Reserved	6 AHB write	7 SDI1 / SDI2	8 EBI2 ^[1] / AHB periph.	9 VPB
HDPB0	-	hclk	sdi1_hclk	ebi2_hclk	vpb1_pclk1
HDPB1	-	-	sdi1_hwrite	ebi2_hwrite	vpb1_pstb
HDPB2	-	-	sdi1_hburst0	ebi2_hburst0	vpb1_pwrite
HDPB3	-	-	sdi1_hburst1	ebi2_hburst1	vpb1_ha16
HDPB4	-	-	sdi1_hburst2	ebi2_hburst2	vpb1_ha15
HDPB10	-	-	sdi1_hprot1	-	vpb2_pwrite
HDPB12	-	-	sdi1_hprot3	-	vpb2_ha14
HDPB13	-	arm_i_hwrite	sdi2_hwrite	ahb_hwrite	vpb2_ha13
HDPB14	-	arm_d_hwrite	sdi2_hburst0	-	vpb2_ha12
HDPB15	-	etn1_hwrite	sdi2_hburst1	-	vpb3_pclk2
HDPB16	-	etn2_hwrite	sdi2_hburst2	-	vpb3_pstb
HDPB20	-	-	sdi2_hready	ahb_hready	vpb3_ha13
HDPB21	-	-	sdi2_hprot0	-	vpb3_ha12
HDPB22	-	-	sdi2_hprot1	-	hsel_vpb1
HDPB23	-	-	sdi2_hprot2	-	hsel_vpb2
HDPB24	-	-	sdi2_hprot3	-	hsel_vpb3
HDPB25	-	-	-	-	-
HDPB26	-	-	-	-	-
HDPB27	-	-	-	-	-

[1] EBI signal monitoring applies to the EBI data port only. Configuration register access is not supported.

Table 22: Debug pin multiplexing description on HDPB (set 10 to 14)

HDPBpin name	multiplex value				
	10 IRQ set 1	11 IRQ set 2	12 DMA	13 IRQ set 3	14 DAIF
HDPB0	intr13	intr25	dmabreq12	intr49	IF_CDC_DAT_AD
HDPB1	intr14	intr26	dmabreq13	intr50	IF_CDC_DAT_DA
HDPB2	intr15	intr27	dmabreq14	intr51	IF_CDC_DIR
HDPB3	intr16	intr28	dmabreq15	intr52	IF_CLK_IF
HDPB4	intr17	intr29	dmabreq16	intr53	IF_DATA_AD
HDPB10	intr23	intr35	dmasreq0	intr59	IF_R_DATA
HDPB12	intr25	intr37	dmasreq12	intr61	IF_INT_TS_INT_ANA
HDPB13	intr26	intr38	dmasreq13	intr62	IF_BAT_OK
HDPB14	intr27	intr39	dmasreq20	intr63	IF_TST
HDPB15	intr28	intr40	dmasreq21	intr64	RSTAPU
HDPB16	intr29	intr41	dmalsreq0	intr65	RSTPO
HDPB20	intr33	intr45	dmau1_htrans1	-	-
HDPB21	intr34	intr46	dmau2_htrans1	-	-
HDPB22	intr35	intr47	hsel_vpb1	-	-
HDPB23	intr36	intr48	hsel_vpb2	IRQ	-
HDPB24	hclk	hclk	hsel_vpb3	hclk	-
HDPB25	pclk1	pclk1	-	pclk1	-
HDPB26	pclk2	pclk2	-	pclk2	-
HDPB27	FIQ	FIQ	-	FIQ	-

Table 23: Debug pin multiplexing description on HDPB (set 15 to 17)

HDPBpin name	multiplex value		
	15 DSP	16 Reserved	17 DSP
HDPB0	fint0_sc	-	dsp_pa13
HDPB1	dsp_fsi	-	dsp_pa14
HDPB2	dsp_fsi_ip	-	dsp_pa15
HDPB3	-	-	-
HDPB4	frame_clock	-	frame_clock
HDPB10	dsp_pa4	-	dsp_xd4

Table 23: Debug pin multiplexing description on HDPB (set 15 to 17)...continued

HDPBpin name	multiplex value		
	15 DSP	16 Reserved	17 DSP
HDPB12	dsp_pa6	-	dsp_xd6
HDPB13	dsp_pa7	-	dsp_xd7
HDPB14	dsp_pa8	-	dsp_xd8
HDPB15	dsp_pa9	-	dsp_xd9
HDPB16	dsp_pa10	-	dsp_xd10
HDPB20	dsp_pa14	-	dsp_xd14
HDPB21	dsp_pa15	-	dsp_xd15
HDPB22	dsp_pwr	-	dsp_xwr
HDPB23	dsp_idle	-	-
HDPB24	-	-	-
HDPB25	dsp_tmu_intreq0	-	-
HDPB26	dsp_tmu_intreq1	-	-
HDPB27	dsp_bcnt0	-	-

6.5.5 Application information

When using the HDPA port the VDDM has to be connected to the VDDL for a common supply level of all debug port pins.

Company Restricted

6.6 HMP - Hybrid Memory Port

6.6.1 Features

- Multiplexing for the following functionality:
 - 16-bit wide SDRAM controller (SDI)
 - Up to 256 Mbyte address range and 104 MHz operation for dynamic memory access
 - 16-bit wide static memory controller (EBI2)
 - Up to 128 kbyte address range and 52 MHz operation for static memory access
- Exclusive static memory choice
- Interface operates at 1.8V or 2.8V supply voltage (VDDM)

6.6.2 Block diagram

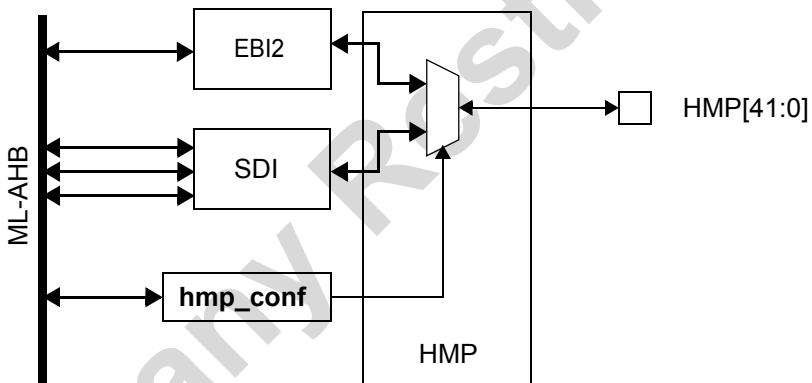


Fig 9. HMP block diagram (not all HMP pins are available)

6.6.3 Hardware interface

Table 24: HMP pin overview

Pin	Name	I/O	Description
HMP[41:30]	HMP outputs	O	HMP outputs
HMP[29]	HMP bidirectionals	I/O	HMP bidirectionals
HMP[28:16]	HMP outputs	O	HMP outputs
HMP[15:0]	HMP bidirectionals	I/O	HMP bidirectionals

6.6.4 Software interface

Table 25: HMP register overview

Symbol	Name	I/O	reset
hmp_conf	HMP configuration	R/W	0x0001

Table 26: Register hmp_conf

Bit	Symbol	Access	Value	Description
15 to 1	-	R	0*	reserved
0	sdiselect	R/W	0	EBI2 selected
			1*	SDI selected

6.6.5 Functional description

The HMP multiplexes the accesses between the EBI2 and the SDI. The choice of the active peripheral is static and exclusive. Preferably the choice is performed during the system boot sequence.

The EBI2 should be used only to connect small memories and any kind of memory mapped peripheral: LCD controllers, polyphonic ICs, camera ICs. The SDI can be used to connect to any commercial SDRAM.

Table 27: HMP pin multiplexing

HMP pins	EBI2 pins	I/O	SDI pins	I/O
HMP39	$\overline{WE_E}$	O	\overline{SDWE}	O
HMP38	$\overline{OE_R/W}$	O	\overline{SDCS}	O
HMP37	$\overline{CS3}$	O	\overline{SDRAS}	O
HMP36	$\overline{CS2}$	O	\overline{SDCAS}	O
HMP35	$\overline{CS1}$	O	\overline{SDUDQM}	O
HMP34	$\overline{CS0}$	O	\overline{SDLDQM}	O
HMP33	$\overline{BE1}$	O	$\overline{SDBA1}$	O
HMP32	ADD1	O	$\overline{SDBA0}$	O
HMP31	$\overline{ADD0_BE0}$	O	\overline{SDCKE}	O
HMP30	IO30	O	\overline{SDCLKO}	O
HMP29	IO29	O	\overline{SDCLKI}	I
HMP[28:16]	IO[28:16]	O	SDA[12:0]	O
HMP[15:0]	IO[15:0]	I/O	SDDQ[15:0]	I/O

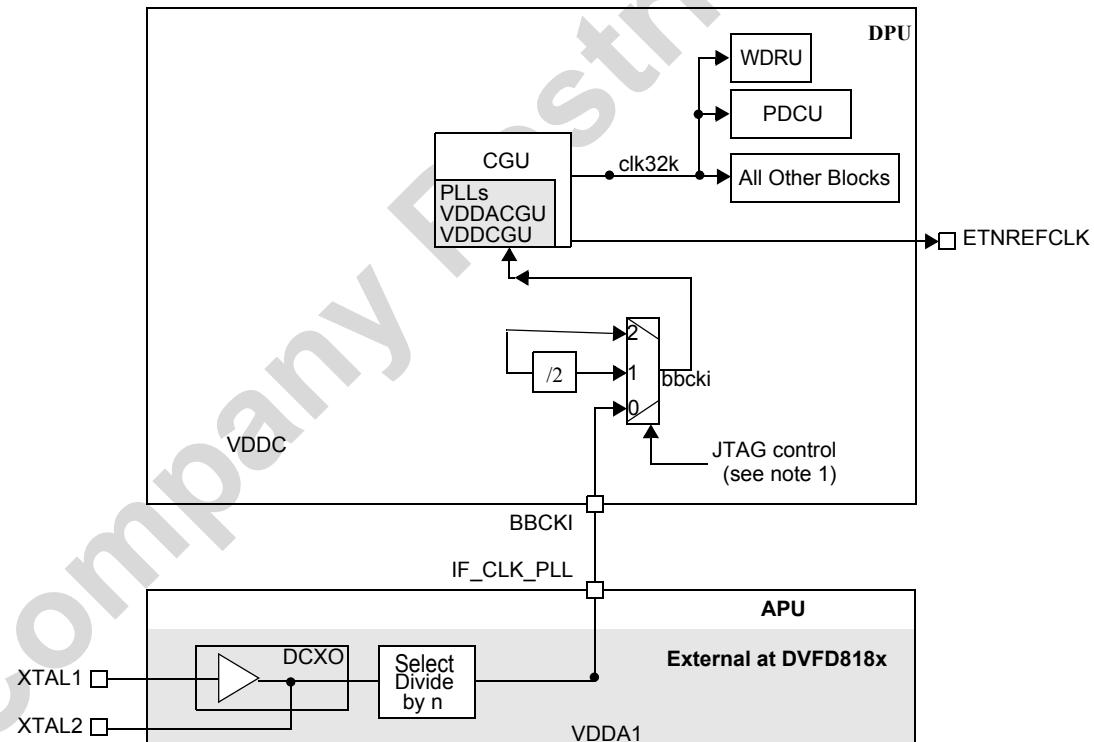
7. Toplevel Blocks

7.1 CGU and DCXO - Clock Generation Units

7.1.1 Features

- Generation and distribution of high frequency clocks
- Separate PLLs for SC, DSP and dedicated peripherals (USB and Ethernet MAC's)
- Smart clock gating for reduction of power dissipation (in sleep and active state)
- Reference clock output for external devices. Frequency depends on application.
- 32768 clock supported via programmable divider

7.1.2 Block diagram



(1) In IC production testing, JTAG can force the multiplexer to position 2

Fig 10. Reference clock and 32768Hz clock input schematic

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

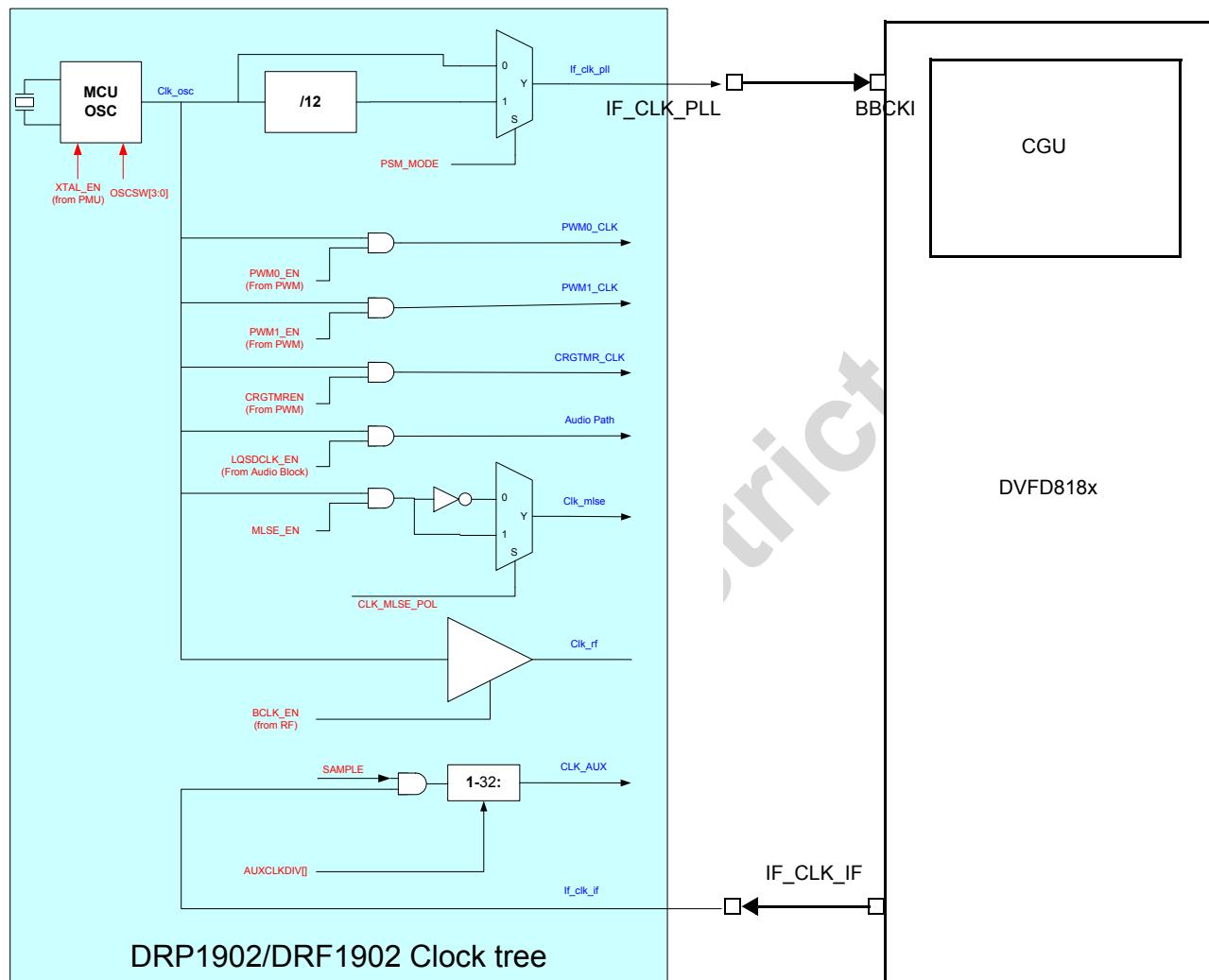


Fig 11. Clock generation for DVFD818x Family (see also DAP1902/DRF1902 datasheets)

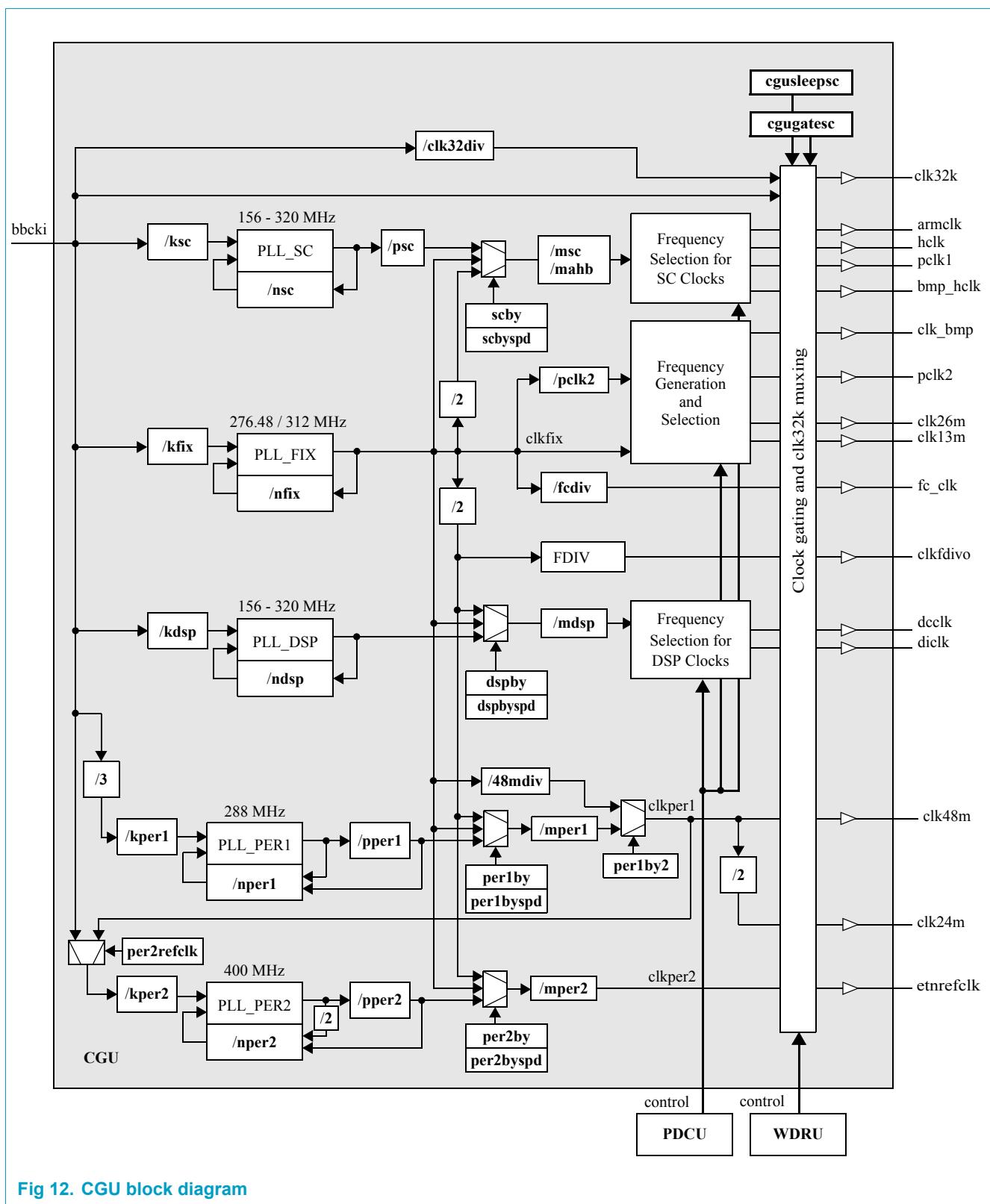


Fig 12. CGU block diagram

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

7.1.3 Hardware interface

Table 28: Clock pin overview

Pin	Name	I/O	Description
BBCKI	DPU reference clock	I	reference clock from APU (application&mode dependent), internally connected to IF_CLK_PLL
IF_CLK_PLL	DPU reference clock	O	reference clock output to DPU from APU (application&mode dependent), internally connected to BBCKI
ETNREFCLK	Ethernet Phy Clock	I/O	Clock from the Ethernet PHY chip.

Table 29: AC characteristic for PLL_SC, PLL_DSP, PLL_PER1, PLL_PER2 and PLL_FIX

Name	Parameter	Min	Typ	Max	Unit
T _{lock}	PLL lock time - after bbcki stable - after change of pre divider and/or feedback divider ratio - after activation	-	-	250	μs

7.1.4 Software interface

Table 30: SC accessible registers

Name	Description	I/O	Reset
cguscccon	CGU control register for SC clock frequencies	R/W	0x0001 005D
cgufixcon	CGU control register for FIX clock frequencies	R/W	0x000F 03CF
cgudspccon	CGU control register for DSP clock frequencies	R/W	0x0000 207F
cguper1con	CGU control register for USB & IIS clock frequencies	R/W	0x0060 1BE2
cguper2con	CGU control register for Ethernet clock frequencies	R/W	0x0180 1FF3
cguper2bwcon	CGU control register for user defined bandwidth setting for PLL_PER2	R/W	0x0000 038D
cgudivcon	CGU control register for 32kHz clock generation and CLKFDIV0 generation	R/W	0x1023 35E8
cgufdiv	CGU fractional divider increment register	R/W	0x25ED 097C
cugatesc	CGU control register for gating clocks in the SC domain	R/W	0xFFFF FFFF
cgusleepsc	CGU control register for gating clocks in the SC domain in Stop and Sleep state	R/W	0x0000 0000

Table 31: Register cguscccon

Bit	Symbol	Access	Value	Description
31 to 23	-	R	0*	reserved
22 to 21	psc[1:0]	R/W	0*	divider ratio of the PLL_SC post divider binary encoded value, valid range is 0 to 3 (division ratio 2, 4, 8, 16)
20	scdirect	R/W	0*	output post divider selection for PLL_SC
			1	output post divider is used, division ratio is defined in field psc
			1	output clock divider is bypassed (division ratio 1)
19	pllsc_lock	R	0*	PLL_SC lock flag
			1	PLL_SC not locked. Phase between clock in and clock out is not guaranteed
			1	PLL_SC locked. Phase between clock in and clock out is ok for more than 8 clock period
18	pllscen	R/W	0*	control for PLL_SC
			1	PLL_SC is disabled - this setting is ignored if scby is set to 0
			1	PLL_SC is enabled

Table 31: Register cgusccon...continued

Bit	Symbol	Access	Value	Description
17	scbyspd	R/W		control for SC clock bypass speed Only relevant when scby = 1 this bit should be changed only when scby = 0
			0*	bypass is done with clkfix/2
			1	bypass is done with clkfix
16	scby	R/W		control for SC clock bypass [4]
			0	armclk , hclk and pclk1 are generated by PLL_SC
			1*	armclk , hclk and pclk1 are generated by PLL_FIX depending on the value of scbyspd bit. The divider ratios mahb and msc are still relevant and determines the speed of the SC clocks
15 to 12	mahb[3:0]	R/W	0*	defines the divider ratio between hclk , pclk1 [1] and armclk [2][3]
				$\text{hclk}, \text{pclk1} = \frac{\text{armclk}}{\text{mahb} + 1}$
				binary encoded value valid range: mahb = 0 to 11 others: reserved
11 to 10	ksc[1:0]	R/W	0*	divider ratio of the PLL_SC pre divider [2][3] binary encoded value valid range: ksc = 0 to 3
9 to 3	nsc[6:0]	R/W	11*	divider ratio of the PLL_SC feedback divider [2][3] binary encoded value (for a bbcki clock of 13.000MHz) valid range: nsc = 11 to 23 for ksc = 0 nsc = 23 to 48 for ksc = 1 nsc = 35 to 72 for ksc = 2 nsc = 47 to 97 for ksc = 3 others: reserved
2 to 0	msc[2:0]	R/W	5*	divider ratio of the PLL_SC feed forward divider [2][3] binary encoded value valid range: msc = 0 to 5 others: reserved

[1] **pclk1** is the VPB bus clock used for all peripherals connected to VPB1.

[2] Programming restrictions as described in SC section apply. **hclk** and **pclk1** must be a multiple of application reference clock. **armclk** must not be greater than 208 MHz. **hclk**, **pclk1** must not be greater than 104 MHz.

[3] Please refer to [Equation 1](#), [Table 3](#), [Table 4](#), [Table 42](#) and [Table 43](#) for the resulting SC domain frequency depending on the application.

[4] If **scby** is changed from '1' to '0' than PLL_SC is switched on even if bit **cgusccon.pllscen** is not set.

Table 32: Register cgufixcon

Bit	Symbol	Access	Value	Description
31	hwctrlby	R/W	0*	hardware synchronized PSM mode control
			1	clock bypass to bbcki, PLLs switch-off & clk32div update operations performed by SW using clk32kpsm , scclksby , fixclksby , clkbmpby bits; cgudivcon register clk32divpsm , clk32fracpsm , clk32div , clk32frac bits; IF_CLK_PLL division ratio changed then via pdcucon register psm_on , psm_off bits
30 to 28	-	R	0*	reserved
27	clk32kpsm	R/W	0*	clk32k divider ratio selection for PSM mode
			1	cgudivcon register clk32div , clk32frac bits are used
26	scclksby	R/W	0*	armclk , hclk , pclk1 , bmp_hclk clocks bypass control
			1	based on PLL_FIX or PLL_SC chain (no bypass)
25	fixclksby	R/W	0*	clk13m , clk26m , pclk2 clocks bypass control
			1	based on PLL_FIX chain (no bypass)
24	clkbmpby	R/W	0*	clk_bmp clock bypass control
			1	based on PLL_FIX chain (no bypass)
23 to 21	-	R	0*	reserved
20 to 16	reserved		15*	keep reset state
15 to 14	pclk2[1:0]	R/W		defines the pclk2 clock frequencies [2][2]. Frequency depends on fixdiv (0b00, 0b01, 0b10, 0b11) field, respectively
			0b00*	pclk2 = 11.520, 13.824, 13.000, 15.600MHz
			0b01	pclk2 = 23.040, 27.648, 26.000, 31.200MHz
			0b10	pclk2 = 34.560, 34.560, 39.000, 39.000MHz
			0b11	pclk2 = 46.080, 46.080, 52.000, 52.000MHz
13 to 12	fixdiv[1:0]	R/W		selects clk_bmp , clk13m , clk26m , pclk2 clock divider's common division ratios (see Table 46) [2][2]
			0b00*	DECT
			0b01	DECT - Mixed
			0b10	WLAN
			0b11	WLAN - Mixed
11	clk48men	R/W		control for the clk48m clock generation derived from PLL_FIX based on the divider block 48mdiv. Frequency correct only if application reference frequency is 13.000MHz.
			0*	divider block 48mdiv for clk48m clock generation disabled
			1	divider block 48mdiv for clk48m clock generation enabled

Table 32: Register cgufixcon...continued

Bit	Symbol	Access	Value	Description
10	plifix_lock	R	0*	PLL_FIX lock flag. Useful for debug only
				PLL_FIX not locked. Phase between clock in and clock out is not guaranteed
				PLL_FIX locked. Phase between clock in and clock out is ok for more than 8 clock period
9	plifixen	R/W	0	control for PLL_FIX
				PLL_FIX is disabled
				PLL_FIX is enabled
8 to 7	kfix[1:0]	R/W	3*	divider ratio of the PLL_FIX pre divider binary encoded value valid range: kfix = 3 for bbcki = 13.824MHz kfix = 0 for bbcki = 13.000MHz kfix = 3 for bbcki = 12.000MHz kfix = 2 for bbcki = 10.368MHz
6 to 0	nfix[6:0]	R/W	79*	divider ratio of the PLL_FIX feedback divider binary encoded value valid range: nfix = 79 for bbcki = 13.824MHz nfix = 23 for bbcki = 13.000MHz nfix = 103 for bbcki = 12.000MHz nfix = 79 for bbcki = 10.368MHz, others: reserved

[1] For FCI, **pclk1** frequency must be greater than or equal to $fc_clk \times \frac{3}{8}$ frequency.

[2] **pclk2** is the VPB bus clock used for all peripherals connected to VPB2 and VPB3.

Table 33: Register cgudspcon

Bit	Symbol	Access	Value	Description
31 to 20	-	R	0*	reserved
19	dsp_idle	R	0*	State of the DSP IDLE signal. It is driven by the DSP and therefore valid only if DSP power and DSP clock is enabled. However, the reset value of 0 is enforced at reset.
18	diclkidleen	R/W	0*	control for diclk ^[1]
				diclk is disabled when DSP goes into IDLE mode
				diclk is enabled when DSP goes into IDLE mode. (Necessary to get access from SC to DSP memories via TMU)
17	dcclken	R/W	0*	control for dcclk ^[1]
				dcclk is disabled
				dcclk is enabled. If DSP executes IDLE instruction, dcclk is stopped regardless of the state of this bit
16	plldsp_lock	R	0*	PLL_DSP lock flag
				PLL_DSP not locked. Phase between clock in and clock out is not guaranteed
				PLL_DSP locked. Phase between clock in and clock out is ok for more than 8 clock period

Table 33: Register cgudspcon...continued

Bit	Symbol	Access	Value	Description
15	plldspen	R/W		control for PLL_DSP
			0*	PLL_DSP is disabled
			1	PLL_DSP is enabled
14	dspbyspd	R/W		control for DSP clock bypass speed - Only relevant when dspby = 1 this bit should be changed only when dspby = 0
			0*	bypass is done with clkfix/2
			1	bypass is done with clkfix
13	dspby	R/W		control for DSP clock bypass
			0	dcclk , diclk are generated by PLL_DSP
			1*	dcclk , diclk are generated by PLL_FIX from clkfix/2 if dspbyspd = 0 or clkfix if dspbyspd = 1 - The divider ratio mdsp is still relevant and determines the speed of the DSP clocks
12	dspclken	R/W		enable bit for all DSP clocks [1]
			0*	dcclk and diclk are disabled
			1	dcclk and diclk are enabled
11 to 10	kdsp[1:0]	R/W	0*	divider ratio of the PLL_DSP pre divider [2] binary encoded value valid range: kdsp = 0 to 3
9 to 3	ndsp[6:0]	R/W	15*	divider ratio of the PLL_DSP feedback divider [2] binary encoded value (for a bbcki clock of 13.000MHz) valid range: 11 to 23 for kdsp = 0 23 to 48 for kdsp = 1 35 to 72 for kdsp = 2 47 to 97 for kdsp = 3, others: reserved
2 to 0	mdsp[2:0]	R/W	7*	divider ratio of the PLL_DSP feed forward divider [2] binary encoded value valid range: mdsp = 1 to 7 others: reserved

[1] Please refer to [Figure 13](#) for DSP clock gating.

[2] Please refer to [Equation 2](#), [Table 5](#) and [Table 44](#) for the resulting DSP domain frequency depending on the application.

[3] If **dspby** is changed from '1' to '0' than PLL_DSP is switched on even if bit **cgudspcon.plldspen** is not set.

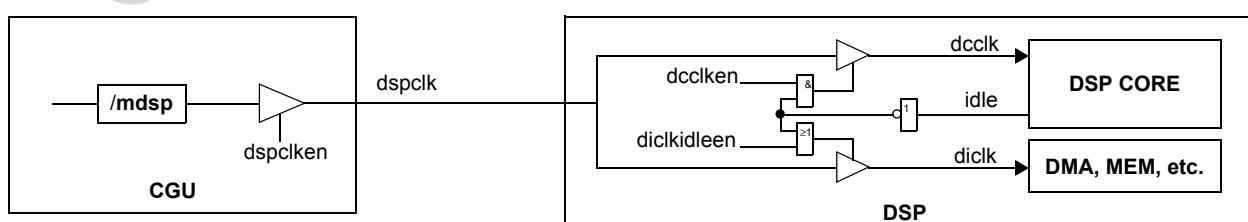


Fig 13. DSP Clock Gating

Table 34: Register cguper1con

Bit	Symbol	Access	Value	Description
31 to 23	-	R	0*	reserved
22	per1by2	R/W		control for PER1 clock bypass 2
			0	clkper1 is generated by using the output of mper1 divider
			1*	bypass is done with clkfix/6.5 using the divider block 48mdiv
21	per1fbclk	R/W		feedback divider clock input selection of the PLL_PER1
			0	selects internal CCO clock output
			1*	selects internal post divider clkout output
20 to 19	pper1[1:0]	R/W	0*	divider ratio of the PLL_PER1 post divider binary encoded value, valid range is 0 to 3 (division ratio 2, 4, 8, 16)
18	per1direct	R/W		output post divider selection for PLL_PER1
			0*	output post divider is used, division ratio is defined in field pper1
			1	output clock divider is bypassed (division ratio 1)
17	pllper1_lock	R		PLL_PER1 lock flag
			0*	PLL_PER1 not locked. Phase between clock in and clock out is not guaranteed
			1	PLL_PER1 locked. Phase between clock in and clock out is ok for more than 8 clock period
16	pllper1en	R/W		control for PLL_PER1
			0*	PLL_PER1 is disabled
			1	PLL_PER1 is enabled
15	per1byspd	R/W		control for PLL_PER1 clock bypass speed - Only relevant when per1by = 1 this bit should be changed only when per1by = 0
			0*	bypass is done with clkfix/2
			1	bypass is done with clkfix
14	per1by	R/W		control for PER1 clock bypass
			0*	per1by2 mux is driven by PLL_PER1
			1	per1by2 mux is driven by PLL_FIX from clkfix/2 if per1byspd = 0 or from clkfix if per1byspd = 1. The divider ratio mper1 is still relevant and determines the speed of the clkper1 clock
13	per1clken	R/W		enable bit for all clocks based on clkper1
			0*	clkper1 is disabled
			1	clkper1 is enabled.
12 to 11	kper1[1:0]	R/W	3*	divider ratio of the PLL_PER1 pre divider binary encoded value, valid range is 0 to 3 (division ratio 1, 2, 3, 4) valid value for 10.368MHz crystal: kper1 = 2 valid value for 13.824MHz crystal: kper1 = 3
10 to 3	nper1[7:0]	R/W	124*	divider ratio of the PLL_PER1 feedback divider binary encoded value to reach fixed output frequency of 144MHz
2 to 0	mper1[2:0]	R/W	2*	divider ratio of the PLL_PER1 feed forward divider binary encoded value valid range: mper1 = 1 to 7

Table 35: Register cguper2con

Bit	Symbol	Access	Value	Description
31 to 25	-	R	0*	reserved
24	etnclkpd[1]	R/W		ETNREFCLK pad pull-down control
			0	disabled
			1*	enabled (effective only when the pad is programmed as input)
23	etnclkoen[1]	R/W		ETNREFCLK pad direction control (see Figure 14)
			0	input
			1*	output
22	per2refclk	R/W		input reference clock selection of the PLL_PER2
			0*	selects bbcki input
			1	selects clkper1 output
21	per2fbclk	R/W		feedback divider clock input selection of the PLL_PER2
			0*	selects internal CCO clock output divided by 2
			1	selects internal post divider clkout output
20 to 19	pper2[1:0][2]	R/W	0*	divider ratio of the PLL_PER2 post divider binary encoded value, valid range is 0 to 3 (division ratio 2, 4, 6, 8)
18	per2direct	R/W		output post divider selection for PLL_PER2
			0*	output post divider is used, division ratio is defined in field pper2
			1	output clock divider is bypassed (division ratio 1)
17	pllper2_lock	R		PLL_PER2 lock flag
			0*	PLL_PER2 not locked. Phase between clock in and clock out is not guaranteed
			1	PLL_PER2 locked. Phase between clock in and clock out is ok for more than 8 clock period
16	pllper2en	R/W		control for PLL_PER2
			0*	PLL_PER2 is disabled
			1	PLL_PER2 is enabled
15	per2byspd	R/W		control for PLL_PER2 clock bypass speed - Only relevant when per2by = 1 this bit should be changed only when per2by = 0
			0*	bypass is done with clkfix/2
			1	bypass is done with clkfix
14	per2by	R/W		control for PER2 clock bypass
			0*	clkper2 is generated by PLL_PER2
			1	clkper2 is generated by PLL_FIX from clkfix/2 if per2byspd = 0 or from clkfix if per2byspd = 1. The divider ratio mper2 is still relevant and determines the speed of the clkper2 clock

Table 35: Register cguper2con...continued

Bit	Symbol	Access	Value	Description
13	per2clken	R/W		enable bit for all clocks based on clkper2
			0*	clkper2 is disabled
			1	clkper2 is enabled.
12 to 9	kper2[3:0] ^[2]	R/W		divider ratio of the PLL_PER2 pre divider binary encoded value, valid range is 0 to 15. Division ratio is:
			0, 1, ... 9	divide by 1, 2, ... 10
			10, 11, 12, 13, 14, 15*	divide by 12, 13, 72, 96, 162, 216, respectively
8 to 3	nper2[5:0] ^[2]	R/W		divider ratio of the PLL_PER2 feedback divider binary encoded value, valid range is 0 to 63. Used for reaching fixed output frequency of 400MHz based on value of kper2 and 13.824MHz bbcki clock input frequency. Division ratio is:
			0, 1, ... 59	divide by 1, 2, ... 60
			60, 61, 62*, 63	divide by 150, 300, 3125, 6250, respectively
2 to 0	mper2[2:0]	R/W	3*	divider ratio of the PLL_PER2 feed forward divider binary encoded value valid range: mper2 = 1 to 7 others: reserved

[1] Please refer to Figure 14 for the Ethernet reference clock output selection

[2] Internal divider ratios must be set when PLL_PER2 is in power down mode (pllper2en = '0'). Divider values change operation and PLL enable/disable operations must be done in two different accesses to the cguper2con register. The reset values are set for 13.824MHz at BBCKI (default DECT Xtal).

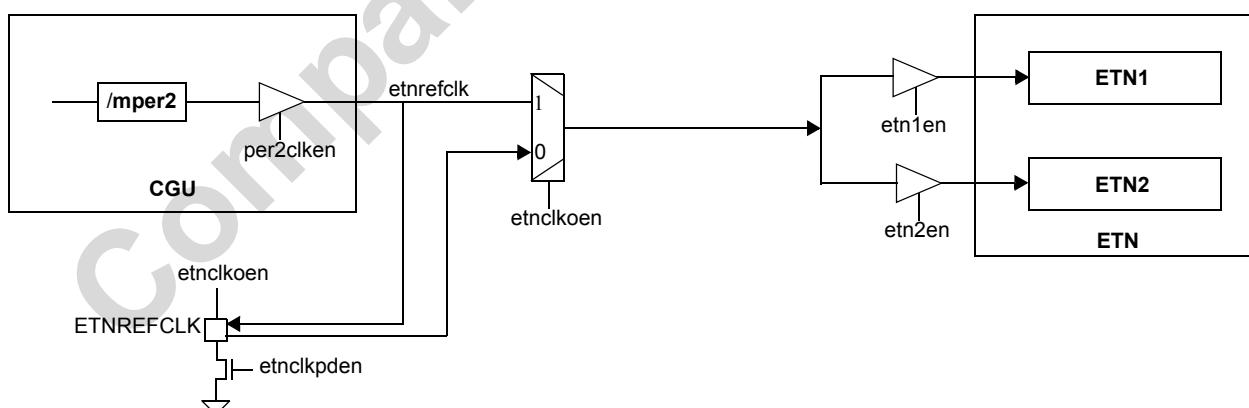


Fig 14. ETNREFCLK pad direction control

Note: At DVFD818x the ETNREFCLK should be used as input only due to high jitter in output mode

Table 36: Register cguper2bwcon

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15	per2bwsel	R/W	0*	user defined bandwidth select bit of the PLL_PER2
			1	default bandwidth settings
			1	user defined bandwidth settings using per2inselr , per2inseli and per2inselp programmable bit fields ^[1]
14 to 11	per2inselr[3:0]	R/W	0b0000* ^[2]	inselr[3:0] user defined bandwidth select inputs of PLL_PER2
10 to 5	per2inseli[5:0]	R/W	0b011100* ^[2]	inseli[5:0] user defined bandwidth select inputs of PLL_PER2
4 to 0	per2inselp[4:0]	R/W	0b01101* ^[2]	inselp[4:0] user defined bandwidth select inputs of PLL_PER2

[1] User defined bandwidth setting feature is for evaluation purposes. It is not recommended to use before consulting DSP Group.

[2] Default user defined bandwidth settings correspond to **nper2** = 24 (division ratio 25). If the PLL is required to operate at a different division ratio with user defined bandwidth settings, these bits must be programmed appropriately prior to enable the PLL.

Table 37: Register cgudivcon

Bit	Symbol	Access	Value	Description
31 to 29	-	R	0*	reserved
28 to 27	xtaldivpsm[1:0]	R/W	0b00	PSM mode XTAL oscillator post divider ratio selection. Selects IF_CLK_PLL (bbcki) frequency only in PSM mode
			0b01	reserved
			0b10	div by 9 (10.368MHz XTAL)
			0b11	div by 12 (12.000MHz / 13.824MHz XTAL)
			0b11	div by 13 (13.000MHz XTAL)
26 to 25	clk32fracpsm[1:0]	R/W ^{[1][2]}	0b00*	PSM mode fractional part of 32kHz clock generation. Following values can be used for correcting the clk32div value
			0b01	adds 0.00 (off)
			0b10	adds 0.50
			0b11	adds 0.33
			0b11	adds 0.25
24 to 16	clk32divpsm[8:0]	R/W	35*	PSM mode division ratio of 32kHz clock generation. Valid range is: 0, 1, 2, ... 511. The division ratio is (div + 1). If the value of clk32frac is non-zero, then the effective division ratio is (div + 1) + frac. Value 0 stops (off) the clk32k counter and it is not recommended to be used (WDRU, PDCU are running with clk32k)
			0b00*	fractional part of 32kHz clock generation. Following values can be used for correcting the clk32div value
			0b01	adds 0.00 (off)
			0b10	adds 0.50
			0b11	adds 0.33
15 to 14	clk32frac[1:0] ^[1]	R/W	0b00*	adds 0.25
			0b01	division ratio of 32kHz clock generation.
			0b10	Valid range is: 0, 1, 2, ... 511. The division ratio is (div + 1). If the value of clk32frac is non-zero, then the effective division ratio is (div + 1) + frac. Value 0 stops (off) the clk32k counter and it is not recommended to be used (WDRU, PDCU are running with clk32k)
			0b11	
13 to 5	clk32div[8:0] ^[1]	R/W	431*	

Table 37: Register cgudivcon...continued

Bit	Symbol	Access	Value	Description
4 to 3	fdivsel[1:0] ^[3]	R/W		define the post-division ratio of the fractional divider
			0b00	divide by 6
			0b01*	divide by 3
			0b10	divide by 2.
			0b11	reserved
2	fraclp ^[4]	R/W		FDIV low power mode - this bit has an effect on the activity rate within the fractional divider
			0*	FDIV runs with maximum activity
			1	FDIV runs with reduced activity (33% power reduction and minimal medium-term jitter increase)
1	fracen	R/W		enable the fractional divider
			0*	fractional part of FDIV block is disabled - CLKFDIVO ^[5] is produced from the PLL_FIX frequency divided by 12 when fdiven = 1
			1	fractional part of FDIV block is enabled and used to produce the CLKFDIVO ^[5] output when fdiven = 1
0	fdiven	R/W		enable the fixed and fractional divider of FDIV block
			0*	CLKFDIVO ^[5] is disabled
			1	CLKFDIVO ^[5] is enabled - Output frequency depends on the fracen setting

[1] When 13.000MHz Xtal is used for WLAN applications, clk32frac[1:0] must be set as 3 and clk32div[8:0] must be set as 405. These values give a fractional division ratio of $(405 + 1) + 0.25 = 406.25$, then the freq. of clk32k = $13.000\text{MHz} / 406.25 = 32.000\text{KHz}$. This will generate 0.25% cycle to cycle jitter on clk32k, but the long term period jitter will be much more lower.

[2] **PSM mode** clk32divpsm[8:0] and clk32fracpsm[1:0] are used for clk32k generation when cgufixcon register bit clk32kpsm = '1'. The clk32divpsm[8:0] and clk32fracpsm[1:0] values must be scaled by the division ratio selected by xtaldivpsm[1:0] bits. If application uses different XTAL frequencies, these bits must be programmed prior to use.

[3] fdivsel[1:0] must be set when fdiven = '0' (when divider is off).

[4] fraclp must be changed when fracen = '0' (when fractional part is off).

[5] Pin CLKFDIVO is not available.

Table 38: Register cgufdiv

Bit	Symbol	Access	Value	Description
31	-	R	0*	reserved
30 to 0	fdiv[30:0]	R/W	0x25ED097C*	fractional divider ratio R - default value provides a 7.68MHz output clock when a 13.824MHz crystal is used. ^[1]

[1] Fractional divider ratio R; any value within the programming restriction described in Section 7.1.5.6 "Fractional divider" is allowed.

Table 39: Register cgugatesc

Bit	Symbol	Access	Value	Description
31	ebi2en	R/W		clock gating: activates the hclk on EBI2 block in all states
			0	hclk is inactive
			1*	hclk is active

Table 39: Register cgugatesc...continued

Bit	Symbol	Access	Value	Description
30	ebi1en	R/W		clock gating: activates the hclk on EBI1 block in all states
			0	hclk is inactive
			1*	hclk is active
29	-	R	1*	reserved
28	sdien	R/W		clock gating: activates the hclk on SDI block in all states
			0	hclk is inactive
			1*	hclk is active
27 to 26	-	R	1*	reserved
25	adpcmen	R/W		clock gating: activates the pclk1 and clk13m on ADPCM block in all states
			0	pclk1 and clk13m are inactive
			1*	pclk1 and clk13m are active
24	firen	R/W		clock gating: activates the pclk1 and clk48m on FIR block in all states
			0	pclk1 and clk48m are inactive
			1*	pclk1 and clk48m are active
23	etn2en ^[1]	R/W		clock gating: activates the hclk and etnrefclk on ETN2 block in all states
			0	hclk and etnrefclk are inactive
			1*	hclk and etnrefclk are active
22	etn1en	R/W		clock gating: activates the hclk and etnrefclk on ETN1block in all states
			0	hclk and etnrefclk are inactive
			1*	hclk and etnrefclk are active
21	daifen	R/W		clock gating: activates the pclk1 and clk13m on DAIF block and the IF_CLK_IF in all states (it is similar to clk_if in the VegaFamily)
			0	pclk1 and clk13m are inactive
			1*	pclk1 and clk13m are active
20	dmauen	R/W		clock gating: activates the hclk on DMAU block in all states
			0	hclk is inactive
			1*	hclk is active
19	drten	R/W		clock gating: activates the hclk DRT block in all states
			0	hclk is inactive
			1*	hclk is active
18	iisen	R/W		clock gating: activates the pclk1 and clk24m on IIS block in all states
			0	pclk1 and clk24m are inactive
			1*	pclk1 and clk24m are active
17	usbden	R/W		clock gating: activates the pclk1 and clk48m on device parts of USB block in all states
			0	pclk1 and clk48m are inactive
			1*	pclk1 and clk48m are active

Company Confidential

Table 39: Register cgugatesc...continued

Bit	Symbol	Access	Value	Description
16	fcien ^[1]	R/W		clock gating: activates pclk1 and fc_clk on FCI block in all states
			0	pclk1 and fc_clk are inactive
			1*	pclk1 and fc_clk are active
15	usimen ^[1]	R/W		clock gating: activates the pclk1 on USIM block in all states
			0	pclk1 is inactive
			1*	pclk1 is active
14	ipinten	R/W		clock gating: activates the pclk1 and clk48m on IPINT block in all states
			0	pclk1 and clk48m are inactive
			1*	pclk1 and clk48m are active
13	pwm3en	R/W		clock gating: activates the pclk2 and clk13mm on PWM3 block in all states
			0	pclk2 and clk13mm are inactive
			1*	pclk2 and clk13mm are active
12	pwm2en	R/W		clock gating: activates the pclk2 and clk13mm on PWM2 block in all states
			0	pclk2 and clk13mm are inactive
			1*	pclk2 and clk13mm are active
11	pwm1en ^[1]	R/W		clock gating: activates the pclk2 and clk13mm on PWM1 block in all states
			0	pclk2 and clk13mm are inactive
			1*	pclk2 and clk13mm are active
10	kbsen	R/W		clock gating: activates the pclk2 on KBS block in all states
			0	pclk2 is inactive
			1*	pclk2 is active
9	gpioen	R/W		clock gating: activates the pclk2 on GPIO block in all states
			0	pclk2 is inactive
			1*	pclk2 is active
8	bmpen	R/W		clock gating: activates the bmp_hclk and clk_bmp on BMP block in all states
			0	bmp_hclk and clk_bmp are inactive
			1*	bmp_hclk and clk_bmp are active
7	uart2en	R/W		clock gating: activates the pclk1 , clk26m and clk13m on UART2 block in all states
			0	pclk1 , clk26m and clk13m are inactive
			1*	pclk1 , clk26m and clk13m are active
6	uart1en	R/W		clock gating: activates the pclk1 , clk26m and clk13m on UART1 block in all states
			0	pclk1 , clk26m and clk13m are inactive
			1*	pclk1 , clk26m and clk13m are active

Table 39: Register cgugatesc...continued

Bit	Symbol	Access	Value	Description
5	iicen	R/W		clock gating: activates the pclk1 on IIC block in all states
			0	pclk1 is inactive
			1*	pclk1 is active
4	spi2en ^[1]	R/W		clock gating: activates the pclk1 on SPI2 block in all states
			0	pclk1 is inactive
			1*	pclk1 is active
3	spi1en	R/W		clock gating: activates the pclk1 on SPI1 block in all states
			0	pclk1 is inactive
			1*	pclk1 is active
2	sctuen	R/W		clock gating: activates the pclk2 and clk13mm on SCTU1 and SCTU2 blocks in all states
			0	pclk2 and clk13mm are inactive
			1*	pclk2 and clk13mm are active
1	extinten	R/W		clock gating: activates the pclk2 on EXTINT block in all states
			0	pclk2 is inactive
			1*	pclk2 is active
0	intcen	R/W		clock gating: activates the hclk on INTC block in all states
			0	hclk is inactive
			1*	hclk is active

[1] Software shall clear it

Table 40: Register cgusleepsc

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11	daifslen	R/W		clock gating: activates pclk1 on DAIF block in sleep/stop state. IF_CLK_IF is not controlled by this bit and remains inactive during sleep/stop
			0*	pclk1 is inactive
			1	pclk1 is active
10	pwmslen	R/W		clock gating: activates pclk2 and clk13mm on PWM block in sleep/stop state
			0*	pclk2 and clk13mm are inactive
			1	pclk2 and clk13mm are active
9	bmphclkslen	R/W		clock gating: activates bmp_hclk clock in sleep/stop state
			0*	bmp_hclk is inactive
			1	bmp_hclk is active
8	bmplen	R/W		clock gating: activates clk_bmp for BMP core in sleep/stop state
			0*	clk_bmp is inactive
			1	clk_bmp is active

Table 40: Register cgusleepsc...continued

Bit	Symbol	Access	Value	Description
7	uart2slen	R/W		clock gating: activates the pclk1 , clk26m and clk13m on UART2 block in sleep/stop state
			0*	pclk1 , clk26m and clk13m are inactive
			1	pclk1 , clk26m and clk13m are active
6	uart1slen	R/W		clock gating: activates the pclk1 , clk26m and clk13m on UART1 block in sleep/stop state
			0*	pclk1 , clk26m and clk13m are inactive
			1	pclk1 , clk26m and clk13m are active
5	iicslen	R/W		clock gating: activates the pclk1 on IIC block in sleep/stop state
			0*	pclk1 is inactive
			1	pclk1 is active
4	spi2slen[1]	R/W		clock gating: activates the pclk1 on SPI2 block in sleep/stop state
			0*	pclk1 is inactive
			1	pclk1 is active
3	spi1slen	R/W		clock gating: activates the pclk1 on SPI1 block in sleep/stop state
			0*	pclk1 is inactive
			1	pclk1 is active
2	sctuslen	R/W		clock gating: activates the pclk2 and clk13mm on SCTU1 and SCTU2 blocks in sleep/stop state
			0*	pclk2 and clk13mm are inactive
			1	pclk2 and clk13mm are active
1	ahbslen	R/W		clock gating: activates hclk for all blocks on AHB in sleep/stop state
			0*	hclk is inactive
			1	hclk is active
0	intcslen	R/W		clock gating: activates the hclk on INTC block in sleep/stop state
			0*	hclk is inactive
			1	hclk is active

[1] Shall not be activated

7.1.5 Functional description

Clock inputs: The clock generation is performed externally for the DVFD818x at DAP1902 or DRF1902 and the distribution by the CGU block. The CGU is placed inside of the DPU.

According to [Figure 11](#), the reference clock generation is at DAP1902 or DRF1902. IF_CLK_PLL is the digital clock output with the crystal frequency in normal mode or divided(see at datasheet for DAP1902 or DRF1902).

For frequency adjustment and oscillator control please consult datasheet of DAP1902 or DRF1902.

7.1.5.1 DPU clock domains

The system reference clock is processed in the CGU by means of PLLs to generate stable clocks to operate SC and DSP at programmable speeds. The SC domains are split up in several smaller clocking regions. Each clock domain can be disabled by software. The SC and the DSP clock domains have independent PLL for maximum flexibility in frequency choice. For all fixed clocks a third PLL is implemented. Two additional PLL's are used to generate the clocks for USB and Ethernet when operating in cordless mode (DECT). The application reference clock is for:

- DECT = 13.824MHz (for DVFD818x)

[Table 41](#) summarizes all clock domains for the SC, DSP, USB and Ethernet.

Table 41: DSP, SC and peripheral clock domains

Clock name	Generated by	Possible frequencies	Related blocks
dcclk	PLL_DSP [1]	18 to 120 MHz	DSP core, BBB and MPU
diclk	PLL_DSP [1]	18 to 120 MHz	DMAC, DCON, TIO
armclk	PLL_SC [1]	24 to 208 MHz	ARM core, IceBreaker, ETM, ETB, I\$, and D\$
hclk	PLL_SC [1]	12 to 104 MHz	DMAU, ARB, EBI1, EBI2, SDI, TMU, DRT, ETN1, ETN2, AHB2VPB, SCRAM and SCROM
pclk1 [2]	PLL_SC [1]	12 to 104 MHz	all VPB peripherals connected to VPB1 bus
pclk2 [2][3]	PLL_FIX	12/13/13.824 MHz 24/26/27.648 MHz 36/39/34.560 MHz 48/52/46.080 MHz	all VPB peripherals connected to VPB2 and VPB3 buses
bmp_hclk	PLL_SC [1]	12 to 104 MHz	BMP bus interface, memory and registers
clk_bmp	PLL_FIX or bbcki	12/13.824/27.648 MHz	BMP core (BMPTBU, etc.)
clk26m [3]	PLL_FIX and pclk1	24/26/27.648 MHz	UART1/2
clk13m [3]	PLL_FIX and pclk1	12/13/13.824 MHz	UART1/2, DAIF, ADPCM
clk13mm [3][4]	PLL_FIX and pclk2	11.52/12/13/13.824 MHz	PWM2/3, SCTU1/2
clk48m	PLL_PER1 or PLL_FIX	48 MHz	FIR, USB, IPINT
clk24m	PLL_PER1 or PLL_FIX	24 MHz	IIS
etnrefclk	PLL_PER2 [1]	25 or 50MHz	ETN1
clk32k	CLK_32DIV	32000 or 32768 Hz	WDRU, PDCU, KBS and EXTINT

[1] Can also be generated from PLL_FIX by using the corresponding bypass field (**scby**, **dspby**)

[2] The clock speed depends on the application reference clock. The clock frequency of pclk2 is based on multiples of the bbcki frequency.

- [3] Valid values are based on the application reference clock.
- [4] clk13mm clock which clocks PWM & SCTU blocks are generated by (sub-multiple) pclk2. Please refer to [Table 32](#) and [Table 46](#) for application specific pclk2 frequency values to calculate clk13mm actual frequency.

7.1.5.2 Changing SC clock speed

The clocking frequency of the SC core and peripherals is controlled by the SC by changing the values in the configuration register **cguSCCON** (for restrictions see [Section 7.1.6](#)).

The following equation applies for register value settings (restriction is that the PLL_SC CCO frequency shall be in the range of 156..320MHz):

$$\text{armclk} = \frac{f(\text{bbcki})}{\text{ksc} + 1} \times \frac{\text{nsc} + 1}{\text{msc} + 1} \quad (1)$$

A change in the SC clocking speed happens, at the latest, 154 ns after writing a new value into **msc** and without glitches on the clock lines. When the feedback divider ratio **nsc** and/or the pre divider ratio **ksc** are changed, then the SW needs to follow the following procedure:

- Set the **cguSCCON.scby** bit - this switches the generation of the **armclk**, **hclk** and **pclk1** to **clkfix/2** (**scbyspd** = 0) or **clkfix** (**scbyspd** = 1) clock derived from **PLL_FIX** (note that **pclk2** is always derived from **PLL_FIX**) - the **msc** and **mab** value might need to be updated as well
- Change the value of the **PLL_SC** feedback divider **nsc** and/or pre divider **ksc**
- Wait during the **PLL** lock time (refer to [Table 29](#)) or **pllsc_lock** = 1
- Clear the **cguSCCON.scby** bit - the **msc** and **mab** value might need to be updated as well

Table 42: Generation of armclk for selected values of nsc (ksc = 0, DECT application, bbcki = 13.824MHz)

Divider nsc	VCO frequency f_{VSC} [MHz]	armclk frequency [MHz]					
		msc = 0	msc = 1	msc = 2	msc = 3	msc = 4	msc = 5
22	317.952	317.952 ^[1]	158.976 ^[2]	105.984 ^[1]	79.488	63.5904	52.992
21	304.128	304.128 ^[1]	152.064 ^[2]	101.376	76.032	60.8256	50.688
20	290.304	290.304 ^[1]	145.152 ^[2]	96.768	72.576	58.0608	48.384
19	276.480	276.480 ^[1]	138.240 ^[2]	92.160	69.120	55.296	46.080
18	262.656	262.656 ^[1]	131.328 ^[2]	87.552	65.664	52.5312	43.776
17	248.832	248.832 ^[1]	124.416 ^[2]	82.944	62.208	49.7664	41.472
16	235.008	235.008 ^[1]	117.504 ^[2]	78.336	58.752	47.0016	39.168
15	221.184	221.184 ^[1]	110.592 ^[1]	73.728	55.296	44.2368	36.864
14	207.360	207.360 ^[2]	103.680	69.120	51.840	41.472	34.560
13	193.536	193.536 ^[2]	96.768	64.512	48.384	38.7072	32.256
12	179.712	179.712 ^[2]	89.856	59.904	44.928	35.9424	29.952
11	165.888	165.888 ^[2]	82.944	55.296	41.472	33.1776	27.648 (default)

[1] These operating points are not guaranteed

[2] These operating points are not guaranteed for all VDDC supply voltages, see [Table 626](#).

Table 43: Generation of hclk and pclk1 out of selected values of armclk (DECT application, bbcki = 13.824MHz)

armclk	hclk, pclk1 frequency [MHz] [3]											
	mahb = 0	mahb = 1	mahb = 2	mahb = 3	mahb = 4	mahb = 5	mahb = 6	mahb = 7	mahb = 8	mahb = 9	mahb = 10	mahb = 11
207.360	207.360 [1]	103.680 [2]	69.120 [2]	51.840	41.472	34.560	29.623	-	-	-	-	-
193.536	193.536 [1]	96.768 [2]	64.512 [2]	48.384	38.707	32.256	27.648	-	-	-	-	-
179.712	179.712 [1]	89.856 [2]	59.904 [2]	44.928	35.942	29.952	-	-	-	-	-	-
165.888	165.888 [1]	82.944 [2]	55.296 [2]	41.472	33.178	27.648	-	-	-	-	-	13.824
152.064	152.064 [1]	76.032 [2]	50.688	38.016	30.413	-	-	-	-	-	13.824	-
138.240	138.240 [1]	69.120 [2]	46.080	34.560	27.648	-	-	-	-	13.824	-	-
124.416	124.416 [1]	62.208 [2]	41.472	31.104	-	-	-	-	13.824	-	-	-
110.592	110.592 [1]	55.296 [2]	36.864	27.648	-	-	-	13.824	-	-	-	-
96.768	96.768 [2]	48.384	32.256	-	-	-	13.824	-	-	-	-	-
82.944	82.944 [2]	41.472	27.648	-	-	13.824	-	-	-	-	-	-
69.120	69.120 [2]	34.560	-	-	13.824	-	-	-	-	-	-	-
55.296	55.296 [2]	27.648	-	13.824	-	-	-	-	-	-	-	-
41.472	41.472	-	13.824	-	-	-	-	-	-	-	-	-
27.648	27.648 (default)	13.824	-	-	-	-	-	-	-	-	-	-

[1] These operating points are not guaranteed

[2] These operating points are not guaranteed for all VDDC supply voltages, see Table 626.

[3] hclk and pclk1 must be a multiple of 13.824 MHz when set below 27.648 MHz (not allowed values are noted with a dash '-' in the table above)

7.1.5.3 Changing DSP clock speed

The clocking frequency of the DSP core and its peripherals is controlled independently by the SC by changing the values in the configuration registers **cgudspcon**.

The following equation applies for register value settings (restriction is that the PLL_DSP CCO frequency shall be in the range of 156..320MHz):

$$f_{dsp} = \frac{f(bbcki)}{kdsp + 1} \times \frac{ndsp + 1}{mdsp + 1} \quad (2)$$

A change in the clocking speed of the DSP happens immediately and without glitches when **mdsp** is changed. When the feedback divider ratio **ndsp** and/or pre divider ratio **kdsp** are changed, then the SW needs to follow the following procedure:

- Set the **cgudspcon.ds bpy** bit - this switches the generation of the **dcclk** and **diclk** to clkfix/2 (**ds bpy spd** = 0) or clkfix (**ds bpy spd** = 1) clock derived from PLL_FIX - the **mdsp** value might need to be updated as well.
- Change the value of the PLL_DSP feedback divider **ndsp** and/or pre divider ratio **kdsp**
- Wait during the PLL lock time (refer to Table 29) or **plldsp_lock** = 1
- Clear the **cgudspcon.ds bpy** bit - the **mdsp** value might need to be updated as well

Table 44: DSP clock frequencies for selected values of ndsp (kdsp = 0, DECT application, bbcki = 13.824MHz)

Divider ndsp	VCO frequency $f_{V_{dsp}}$ [MHz]	DSP frequency f_{dsp} [MHz]							
			mdsp = 1	mdsp = 2	mdsp = 3	mdsp = 4	mdsp = 5	mdsp = 6	mdsp = 7
22	317.952	158.976 [1]	105.984 [2]	79.488 [2]	63.5904 [2]	52.992	45.42171	39.744	
21	304.128	152.064 [1]	101.376 [2]	76.032 [2]	60.8256 [2]	50.688	43.44686	38.016	
20	290.304	145.152 [1]	96.768 [2]	72.576 [2]	58.0608 [2]	48.384	41.472	36.288	
19	276.480	138.240 [1]	92.160 [2]	69.120 [2]	55.296 [2]	46.080	39.49714	34.56	
18	262.656	131.328 [1]	87.552 [2]	65.664 [2]	52.5312	43.776	37.52229	32.832	
17	248.832	124.416 [1]	82.944 [2]	62.208 [2]	49.7664	41.472	35.54743	31.104	
16	235.008	117.504 [2]	78.336 [2]	58.752 [2]	47.0016	39.168	33.57257	29.376	
15	221.184	110.592 [2]	73.728 [2]	55.296 [2]	44.2368	36.864	31.59771	27.648	
14	207.360	103.680 [2]	69.120 [2]	51.840	41.472	34.560	29.62286	25.92	
13	193.536	96.768 [2]	64.512 [2]	48.384	38.7072	32.256	27.648	24.192	
12	179.712	89.856 [2]	59.904 [2]	44.928	35.9424	29.952	25.67314	22.464	
11	165.888	82.944 [2]	55.296 [2]	41.472	33.1776	27.648	23.69829	20.736	
						(default)			

[1] These operating points are not guaranteed

[2] These operating points are not guaranteed for all VDDC supply voltages, see Table 626.

7.1.5.4 Generation of the 24/48MHz clocks

The clock for the USB (48MHz) and the IIS (24MHz) blocks are derived from different sources depending on the application. The clk24m is derived always from the clk48m by a divider with a ratio of 2. The options to generate the 48MHz clock are:

- In a WLAN application based on a 13MHz application reference clock, the PLL_FIX output frequency should be set to 312MHz. The selected input in **cguper1con.per1by2** is the divider block '48mdiv' with a division ratio of 6.5 to generate the clk48m. To activate, the block must be enabled with **cgufixcon.clk48men**. The block requires at most 100 ns to output a stable clock. The PLL_PER1 is switched off to save power.
- In a DECT application based on a 13.824MHz application reference clock, a separate pll (PLL_PER1) is required to generate a multiple of the 48MHz clock. The input selection in **cguper1con.per1by** is the PLL_PER1 output. The input selection for **cguper1con.per1by2** is the mper1 divider output. Depending on the crystal frequency used, the pre-divider ratio of PLL_PER1 needs to be selected as 3 or 4 (for 10.368MHz or 13.824MHz, respectively). The PLL_PER1 feedback divider is fixed at 125 to generate 144MHz out of the phase detector frequency of 1.152MHz derived from the crystal clock by a total division ratio of 9 or 12. The division ratio of the feed forward divider **cguper1con.mper1** is 3.

The following equation applies for register value settings (restriction is that the PLL_PER1 CCO frequency shall be in the range of 156..320MHz):

$$\text{clk48m} = \frac{f(\text{bbcki})}{\text{kper1} + 1} \times \frac{\text{nper1} + 1}{\text{mper1} + 1} \quad (3)$$

Clock gating: Clock gating is implemented to turn off the clocks going to all temporarily or permanently unused peripherals - the related register is **cgugatesc**. This clock gating has the highest priority (refer to [Figure 15](#)).

7.1.5.5 SC sleep and stop clocking

Parts of the SC subsystem need to be clocked in Sleep and Stop state in order to generate wake-ups. In Sleep and Stop state, the CGU activates the **pclk2** signal for KBS and for EXTINT blocks at the speed of the **clk32k** signal. This function is always active and cannot be changed by the SW.

In addition to the KBS and the EXTINT, it is possible to activate **pclk1** or **pclk2** (at 32 kHz) for the blocks defined by register **cgusleepsc**. Also all other blocks connected to the AHB (including AHB2VPB bridges) can be clocked with 32 kHz in Sleep state if required. All other blocks cannot be activated.

The **cgusleepsc** register has a reduced priority with respect of the **cgugatesc** register. In other words the content of **cgusleepsc** for a specific block has only a meaning if the corresponding block is enabled in the **cgugatesc** register.

Table 45: Effect of gating registers to SC clocking

SC mode	cgugatesc	cgusleepsc
active	enable bit is relevant	no effect
idle	enable bit is relevant	no effect
stop	enable bit is relevant	enable bit is relevant if related bit in cgugatesc = 1
sleep	enable bit is relevant	enable bit is relevant if related bit in cgugatesc = 1

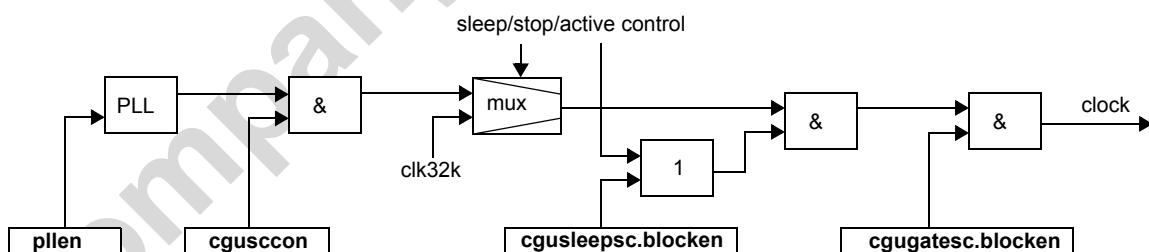


Fig 15. Clock gating priority

To allow BMPTBU activity in STOP and SLEEP mode, the bits **cgusleepsc.bmpslen**, **cgufixcon.bmpclkby** and **cgugatesc.bmpen** have to be set. The bit **cgusleepsc.bmphclslen** enables the bmp_hclk in STOP mode to allow BMP activity while system is stopped. It does also ignore the bit **cgusleepsc.ahbslen** and the bmp_hclk frequency is the normal one (not 32kHz).

Setting the bit **cgufixcon.bmpclkby** the BMP core clock is taken directly from the PLL input (bbcki). This is applicable only for a DECT application with a 13.824MHz crystal. It has to be tight to bmp power save mode controlled in the PDCU (see [Table 62](#)).

7.1.5.6 Fractional divider

A fractional divider block FDIV generates **CLKFDIVO** to cope with inaccurate reference clock frequencies (refer to [Figure 16](#) and [Figure 17](#)). This is a flexible divider that can be used, for example, to create a 7.68MHz clock.

A change in the programmed value, has immediately effect on the R of the fractional divider. **cgufdiv** can be programmed between 0x1FFFFFFF and 0x3FFFFFFF which corresponds of a divider ratio between 4 and 8. The fractional divider allows changing of the **cgufdiv** value at any time (no long roll over waiting).

The fractional divider can be disabled and bypassed without any spikes on the clock signal. The power consumption of the divider can be reduced by using the **cgudivcon.fraclp** field however this increases the inaccuracy / jitter of the divider.

Note that pin CLKDIVO is not available at DVFD8185 and DVFD8187.

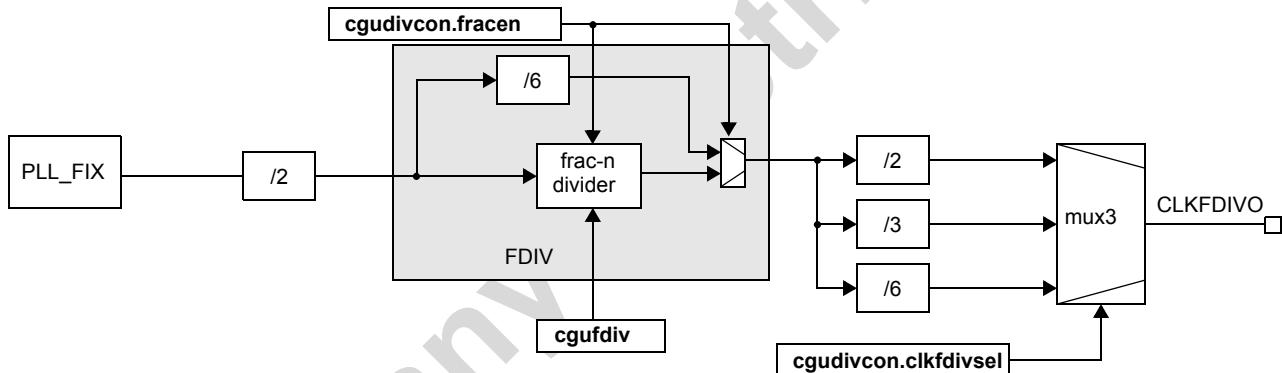


Fig 16. Fractional divider block FDIV and generation of CLKFDIVO

Note: CLKDIVO is not available at DVFD818x

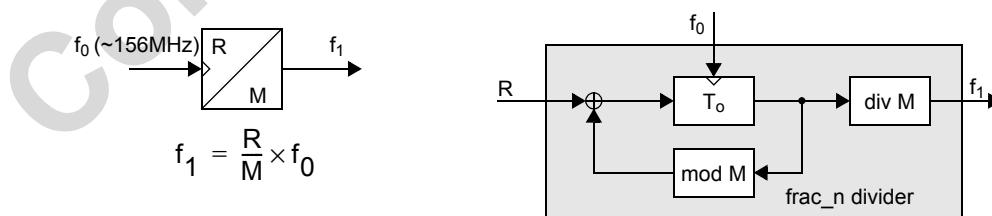


Fig 17. Synchronously clocked fractional divider as implemented in the FDIV block: $M = 2^{32}$

32 kHz clock: The DVFD818x generates a clock of 32000 Hz +/-10% using a programmable divider. The reset value of this divider is chosen to produce exactly 32000Hz when used with a 13.824MHz crystal. For higher average accuracy also the fractional part of this programmable divider can be used. But it will introduce a jitter on the clock (e.g. a **clk32frac[1:0]** = 0b11 leads to an output cycle time sequence of 3 times with input clock / div+1 followed by 1 time with input clock / div+2).

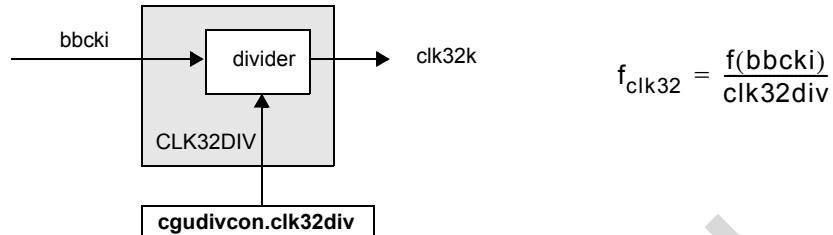


Fig 18. Divider block CLK32DIV and generation of 32kHz clock

7.1.5.7 BMP, SCTU and PWM clocks

The core clocks of the SCTU and PWM blocks are synchronized with the pclk2 (integer division from pclk2). The configuration depicted in the grey fields in Table 46 is not recommended for use. In DECT systems the configuration with **cgufixcon.fixdiv** = 0b01 will require a reprogramming of the SCTU/PWM blocks if the bit **cgufixcon.fixdiv[1]** is changed to keep the programmed timing behaviour of these blocks.

Table 46: BMP, SCTU, PWM and pclk2 clocks

System	bbcki [MHz]	clkfix [MHz]	cgufixcon				clk13m [MHz]	clk26m [MHz]	pclk2 [MHz]	clk13mm for SCTU & PWM [MHz]	BMP clock [MHz]
			fixdiv [1]	fixdiv [0]	pclk2 [1]	pclk2 [0]					
DECT	13.824	276.480	0	0	0	0	13.824	27.648	11.520	11.520	13.824
					0	1				23.040	
					1	0				34.560	
					1	1				46.080	
			0	1	0	0	13.824	13.824			
					0	1				27.648	
					1	0				34.560	11.520
					1	1					46.080

7.1.6 Application information

CAUTION



The AHB bus clock should not be changed during SDRAM accesses.

CAUTION



A watchdog reset applies now also to CGU registers.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

CAUTION

If watchdog is enabled when PSM mode is entered, then scby mux must be set to PLL_FIX before invoking the PSM mode.

CAUTION

If the bit **cguscccon.scdirect** needs to be changed, the following scheme should be applied. Set multiplexer (bit **cguscccon.scby**) to PLL_FIX, then switch off the PLL_SC, then change the bit **cguscccon.scdirect**, then switch on PLL_SC and set multiplexer (bit **cguscccon.scby**) to PLL_SC.

7.2 TAP1/2 - JTAG Test Access Ports

7.2.1 Features

- Two JTAG compliant [9.] test access ports (TAP) and four TAP controllers (TAPC)
- Independent parallel TAPC access
- Externally selectable smart multiplexing for the four TAPCs on the DPU
- Emulation feature control
- Debugging of SC code by means of scanning by EmbeddedICE-RT
- Debugging and evaluation of several hardware features
- Full boundary scan of digital pins
- Programming of the functions of the ETM9 and ETB block
- Production testing (not the scope of this specification)

7.2.2 Block diagram

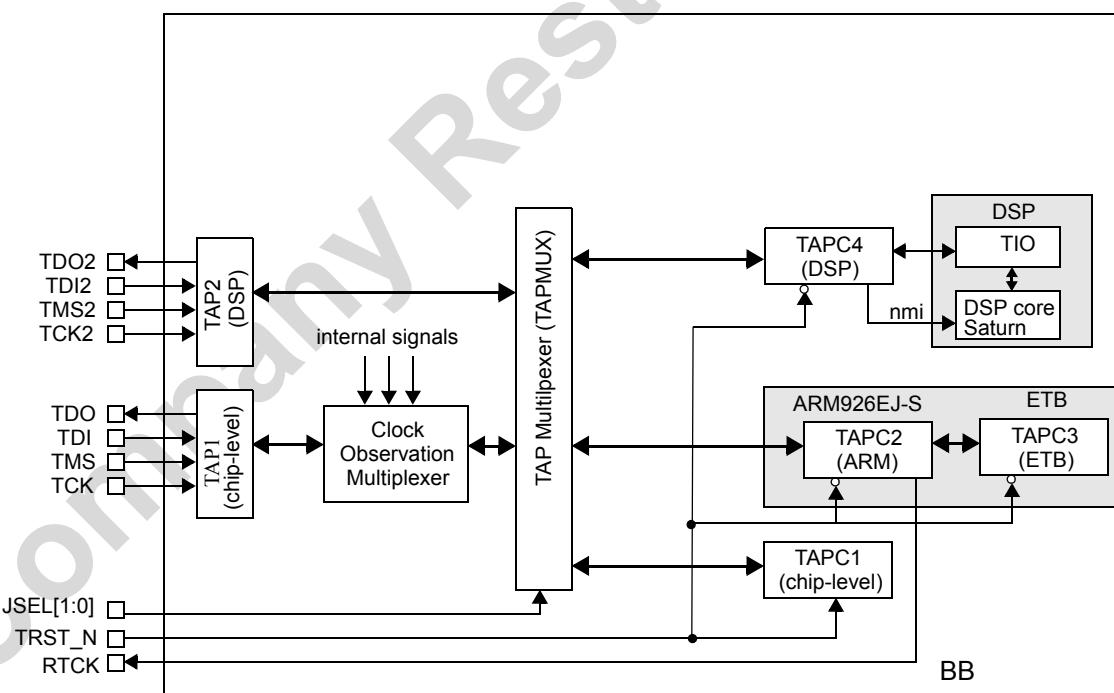


Fig 19. TAP1/2/3/4 Block Diagram

7.2.3 Hardware interface

Table 47: General Signals

Pin	Name	I/O	Description
TRST_N	test reset	I	JTAG asynchronous reset for TAPC1, TAPC2, TAPC3 and TAPC4
JSEL[1:0]	JTAG selector	I	selects between TAPC1, TAPC2/TAPC3, TAPC2/TAPC3/TAPC4 and TAPC4 (see Table 51)
RTCK	return test clock	O	JTAG return test clock from TAPC2. RTCK is the TCK synchronized to the internal ARM clock armclk . RTCK is required for external debug HW

Table 48: TAP1 overview

Pin	Name	I/O	Description
TCK	test clock	I	JTAG scan clock
TMS	test mode select	I	JTAG test mode select
TDI	test data input	I	JTAG data input
TDO	test data output	O	JTAG data output

Table 49: TAP2 overview (only available in development package)

Pin	Name	I/O	Description
TCK2	test clock	I	JTAG scan clock
TMS2	test mode select	I	JTAG test mode select
TDI2	test data input	I	JTAG data input
TDO2	test data output	O	JTAG data output

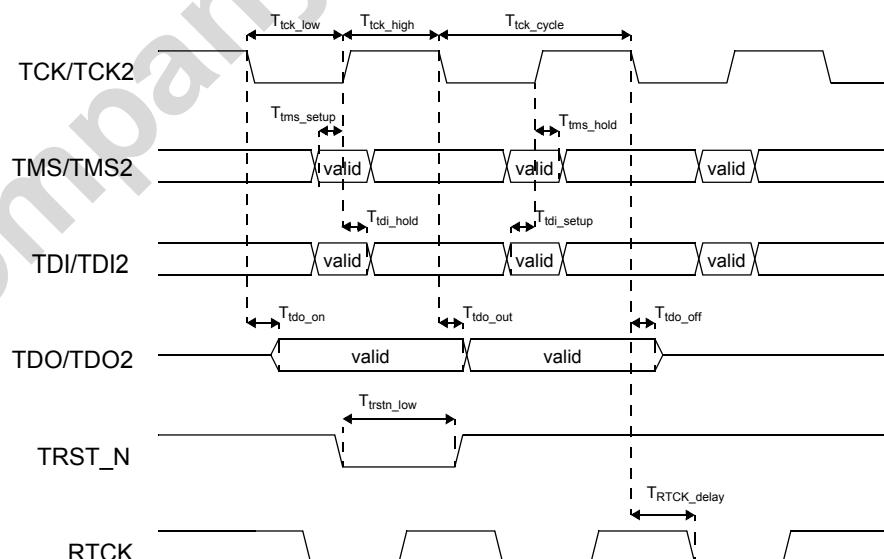


Fig 20. Timing diagram of the JTAG interfaces

Table 50: AC characteristics of test access port TAP1, TAP2, TAP3

Symbol	Description	Min	Typ	Max	Unit
T _{tck_cycle}	TCK/TCK2 clock period	50	-	-	ns
T _{tck_high}	TCK/TCK2 high time	25	-	-	ns
T _{tck_low}	TCK/TCK2 low time	25	-	-	ns
T _{tms_setup}	TMS/TMS2 setup time to rising edge of TCK/TCK2	10	-	-	ns
T _{tms_hold}	TMS/TMS2 hold time after rising edge of TCK/TCK2	5	-	-	ns
T _{tdi_setup}	TDI/TDI2 setup time to rising edge of TCK/TCK2	10	-	-	ns
T _{tdi_hold}	TDI/TDI2 hold time after rising edge of TCK/TCK2	5	-	-	ns
T _{tdo_on}	TDO/TDO2 enable time after falling edge of TCK/TCK2	-	-	20	ns
T _{tdo_out}	TDO/TDO2 valid after falling edge of TCK/TCK2	-	-	5	ns
T _{tdo_off}	TDO/TDO2 disable time after falling edge of TCK/TCK2	-	-	5	ns
T _{trstn_low}	TRST_N low pulse width	100	-	-	ns
T _{RTCK_delay}	RTCK delay after TCK	-	3xT _{armclk}	-	ns

7.2.4 Functional description

Two JTAG test access ports (TAP) and four test access controllers (TAPC) are available. They are used for boundary scan testing, IC production test and simultaneous debugging of SC and DSP cores. The test access ports and controllers are fully IEEE standard 1149.1 compliant [9.]¹.

7.2.4.1 Test access ports

Two test access ports are available providing independent parallel debugging of DSP and the SC. Typically DSP debuggers [6.] are connected to TAP2 while the ARM JTAG debugger is connected to TAP1. This allows scanning, debugging, and tracing with minimum interference to the application. TAP2 is multiplexed with HDP pins. Selection is done via GPM.

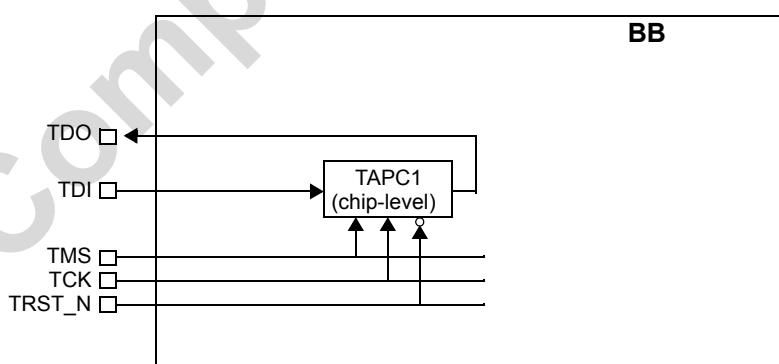
In cases where the HDP is used for other purposes (i.e. the pins are not available for TAP2), it is still possible to alternatively access all two subsystems via TAP1 by means of the TAP multiplexer shown in Figure 19. The TAP multiplexer is controlled via JSEL[1:0] input pins according to Table 51.

1. All mode behaves as a IEEE1149.1 compliant device as long as it is used in stand-alone mode.

Table 51: TAPMUX control via JSEL[1:0]

Mode	JSEL[1:0]	Description
0	0b00	DSP only In this mode TAP1 is connected to TAPC4. TAP2 as well as TAPC1, TAPC2 and TAPC3 are not connected. This mode is useful in the product package for debugging of DSP.
1	0b01	Board test mode In this mode the TAP1 is connected to TAPC1 (see Figure 21) and all other tap controllers are disconnected (TAPC2, TAPC3 and TAPC4).
2	0b10	Dual debug ^[1] In this mode the TAP1 is connected to the daisy chain of TAPC2 and TAPC3 (see Figure 22). TAP2 is connected to the TAPC4. This mode allows simultaneous debugging of SC and DSP.
3	0b11	Four TAPC chain ^[1] In this mode the TAP1 is connected to a daisy chain of TAPC1, TAPC2, TAPC3 and TAPC4 (see Figure 23). A bypass multiplexer is available in this mode. The setting of this bypass multiplexer depends on the content of the TAPC1 instruction register: In the shift-IR state, all four TAP controllers are part of the daisy chain. In the Shift-DR state, the bypass multiplexer switches to the TDO output of TAPC1, when a standard chip-level IEEE1149.1 instruction is activated in the TAPC1 (see Table 57). This mode is the standard mode used for IC and PCB testing. It also allows to debug both cores, even in the product package where only TAP1 is available. A disadvantage is that the debug tools must support daisy chained TAP controllers.

[1] This mode behaves like a IEEE1149.1 compliant device, if the ARM clock is running six times faster than TCK (internal clock synchronization of TCK and armclk within TAPC2.)

**Fig 21. Board test mode (mode 1- JSEL[1:0]=0b01)**

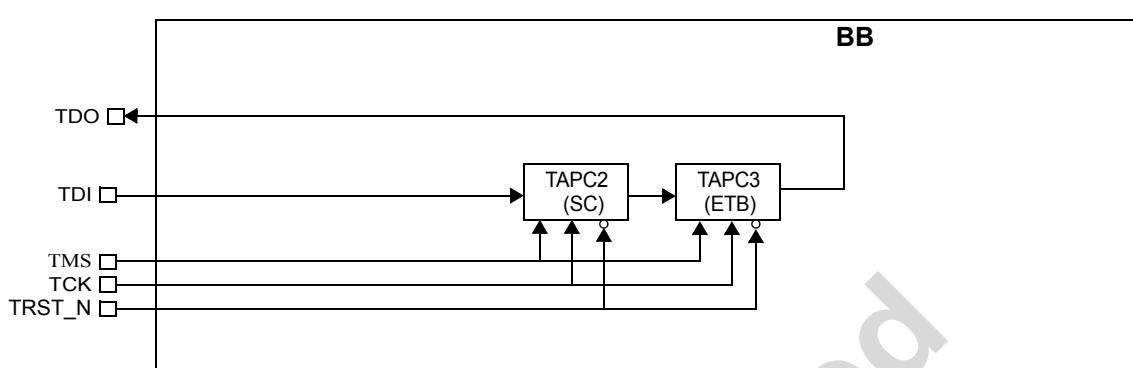


Fig 22. Chaining of TAP controllers (mode 2 - JSEL[1:0]=0b10)

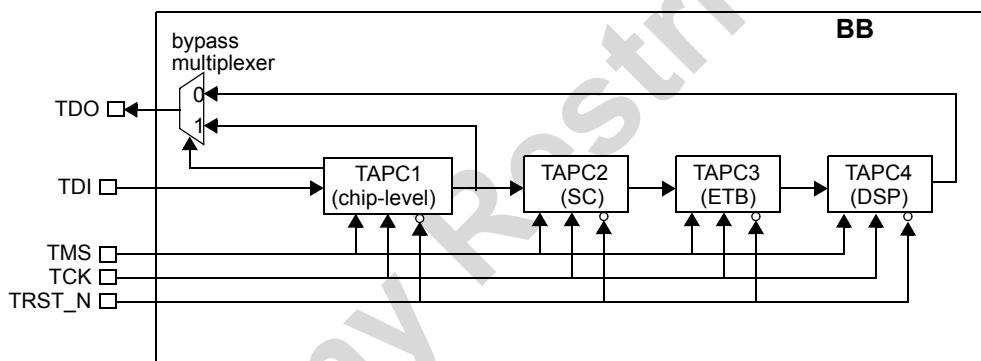


Fig 23. Chaining of TAP controllers (mode 3 - JSEL[1:0]=0b11)

The JSEL[1:0] signals have an effect to the total chained instruction register length. Please be aware that the TAPC1 and TAPC4 have an 8-bit instruction register, whereas TAPC2 and TAPC3 have a 4-bit instruction register. The control of the chained TAPCs differs depending on the setting of the JSEL[1:0] signals.

7.2.4.2 JTAG resets

The pin TRST_N resets TAPC1, TAPC2, TAPC3 and TAPC4 asynchronously. All TAP controllers can also be initialized applying the standard JTAG reset sequence (5 logic ones on TMS).

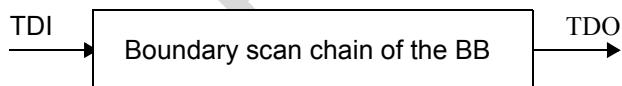
During parallel core debugging (JSEL[1:0]= 0b10), it is advised to connect the TRST_N to the ARM JTAG debugger. The DSP emulator software resets the TAPC4 with 5 logic ones on TMS2. During single core debugging (JSEL[1:0] <> 0b10), the reset of DSP emulator should not be connected to the DVFD818x.

Table 52: Reset pin connection

TASK	TAP1	TAP2
Parallel debugging of all two cores (JSEL[1:0] = 0b10)	JTAG debugger to TRST_N	reset TAPC4 by 5 logic ones
Single DSP core debugging (JSEL[1:0] = 0b10)	reset TAPC4 by 5 logic ones	-
Single SC core debugging (JSEL[1:0] = 0b10)	JTAG debugger to TRST_N	-
Chained JTAGs for boundary scan test (JSEL[1:0] = 0b11)	tester connects to TRST_N	-

7.2.4.3 Boundary scan chain

The DVFD818x provides a complete boundary scan chain on the digital pins of the circuit. By means of the EXTEST instruction, it is possible to connect the pins of the DVFD818x boundary scan chain between the TDI and TDO pins. **Figure 24** visualizes the boundary scan chain.



(1) The exact sequence of the cells can be delivered on demand in form of a BDSL file.

Fig 24. Complete boundary scan chain

7.2.4.4 Clock Observation Mode

For debugging purposes it is possible to externally observe some internal signals without the need to run any software. The internal signals to be observed are selected by using combinations of the JTAG signals TCK TMS and TDI while TRST_N is kept low. The application mode is equivalent to the un-driven state of the JTAG interfaces. The functionality is described in **Table 53**.

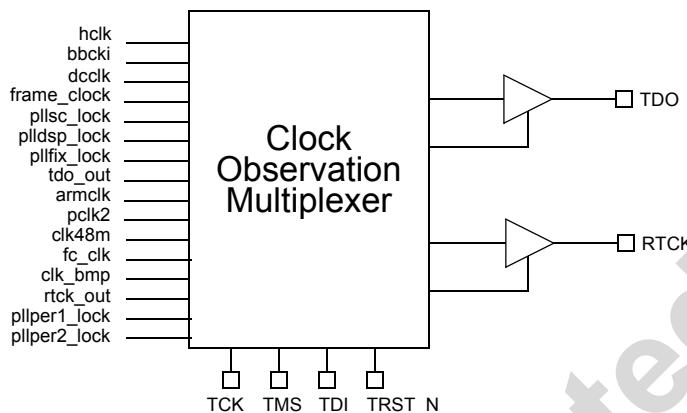


Fig 25. Clock observation multiplexer

Table 53: Clock observation mode

Inputs				Output		Mode description
TRST_N	TCK	TMS	TDI	TDO	RTCK	
pull-down	pull-up	pull-up	pull-up	-	-	tristate
0	0	0	0	armclk [1]	pllsc_lock [1]	JTAG reset - CGU enabled
0	0	0	1	bbcki [1]	frame_clock [1]	JTAG reset - CGU enabled
0	0	1	0	dcclk [1]	plldsp_lock [1]	JTAG reset - CGU enabled
0	0	1	1	pclk2 [1]	plifix_lock [1]	JTAG reset - CGU enabled
0	1	0	0	hclk [1]	clk_bmp [1]	JTAG reset - CGU enabled
0	1	0	1	clkper1 [1]	pllper1_lock [1]	JTAG reset - CGU enabled
0	1	1	0	fc_clk [1]	pllper2_lock [1]	JTAG reset - CGU enabled
0	1	1	1	Z	Z	application mode
1	0	0	0	tdo_out	rtck_out	Normal JTAG mode
1	0	0	1	tdo_out	rtck_out	Normal JTAG mode
1	0	1	0	tdo_out	rtck_out	Normal JTAG mode
1	0	1	1	tdo_out	rtck_out	Normal JTAG mode
1	1	0	0	tdo_out	rtck_out	Normal JTAG mode
1	1	0	1	tdo_out	rtck_out	Normal JTAG mode
1	1	1	0	tdo_out	rtck_out	Normal JTAG mode
1	1	1	1	tdo_out	rtck_out	Normal JTAG mode

[1] The clock and lock signals in this table are explained in [Section 7.1](#).

RSTBB signal needs to be high in order for the clock observation mode to work.

7.2.4.5 ID codes

The ID codes are hard coded within the TAP controllers and can be shifted out using the IDCODE instruction. There are no ID codes for TAPC4.

Table 54: ID code of the TAPC1

Bit number	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 9	8 8	7 7	6 6	5 5	4 4	3 3	2 2	1 1	0 0
Field	version	part number																	0	manufacturer									1			
Content	0 0 0 0 1	1 0 1 1 0	0 0 0 1 1	0 0 0 1 0	0 0 0 0 0	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0																						
Meaning	1	V	F	B															0	Philips									1			

Table 55: ID code of TAPC2

Bit number	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 9	8 8	7 7	6 6	5 5	4 4	3 3	2 2	1 1	0 0
Field	version	part number																	0	manufacturer									1			
Content	0 0 0 0 1	0 1 1 1 1	0 0 1 0 0	0 0 0 1 1	0 0 0 0 0	1 0 1 0	0 1 0 1	0 1 0 1	1 0 1 0	1 0 1 0																						
Meaning	1	ARM926																	0	Philips									1			

Table 56: ID code of TAPC3

Bit number	3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9 9	8 8	7 7	6 6	5 5	4 4	3 3	2 2	1 1	0 0
Field	version	part number																	0	manufacturer									1			
Content	0 0 0 0 1	1 0 1 1 1	0 0 1 0 0	0 0 0 0 0	1 0 1 0	0 1 0 1	0 1 0 1	1 0 1 0	1 0 1 0																							
Meaning	1	ETB																	0	Philips									1			

7.2.4.6 TAPC1 description

TAPC1 is the chip level TAP controller. It features a set of hard coded commands built in an 8-bit instruction set. The instruction set includes all mandatory instructions defined by the JTAG standard [9.] Other instructions also support production testing (not the scope of this document). Table 57 lists the instructions available for testing, debugging and tracing of the cores, and production testing.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

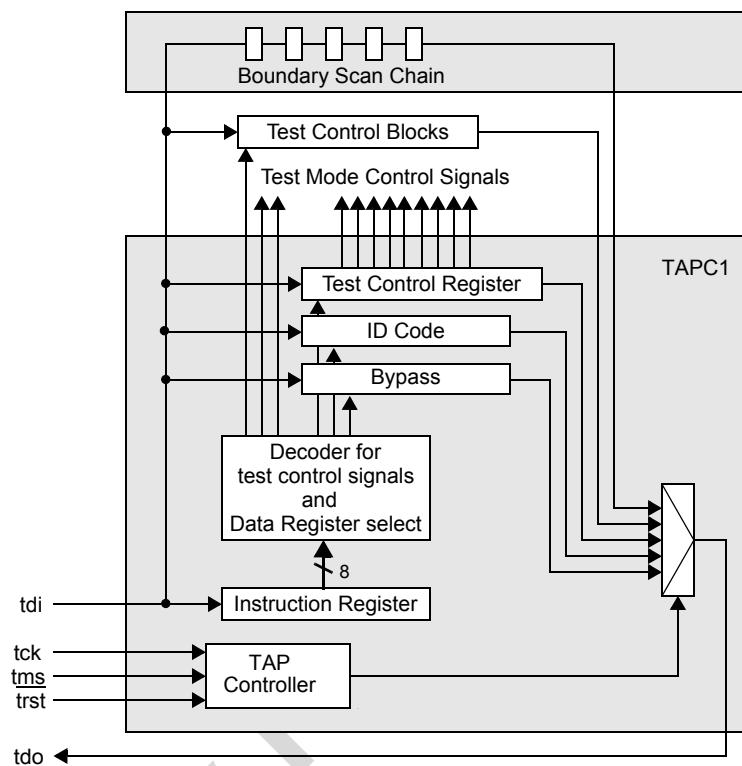


Fig 26. TAPC1 block diagram including boundary scan chain

Table 57: TAPC1 instruction set

Code	Instruction/ Test mode	Short description	Part of IEEE standard
0000 0000	EXTEST [1]	boundary Scan Chain mode: allows testing of off-chip circuitry and board level interconnections	standard
1111 1111	BYPASS [1]	boundary Scan Bypass mode: sets one register between serial data input and output - allows to skip the IC when it is not used during board level testing	standard
0000 0001	SAMPLE/PRELOAD [1]	boundary Scan Chain mode: can be used to sample the inputs and outputs during normal operation of the IC - also allows to preload data into the latched outputs of the boundary scan cells	standard
0000 0010	HIGHZ [1]	boundary Scan Bypass mode: sets all outputs of the IC to high impedance	optional
0000 0011	CLAMP [1]	boundary Scan Bypass mode: drives the preset values onto the outputs of the IC which have been defined with the PRELOAD command - useful for board level testing when not all devices support boundary scan	optional
0000 0100	IDCODE [1]	TAPC1 identification: read the ID code from ID register	optional
0000 0110	INTEST [1]	boundary scan chain mode: allows testing of internal circuitry of the IC	optional
0000 1000	PROG_TCB [2]	program all the TCB blocks which are connected in one long TCB scan chain	no

Table 57: TAPC1 instruction set...continued

Code	Instruction/ Test mode	Short description	Part of IEEE standard
0000 1001	PROG_TPR ^[2]	program all the TPR blocks of the DPU which are connected in one long TPR scan chain	no
0001 1000	AC_TEST ^[2]	test mode for pad delay measurement	no
0001 1001	RESET_CON ^[2]	Trigger or block a RSTBB reset. (see Section 7.6.5)	no
0001 1101	DBG_BYPASS ^[2]	The DBG_BYPASS instruction is selected instead of the standard BYPASS instruction, to select the bypass register, but not activate the bypass multiplexer (i.e. the bypass multiplexer is in position 0).	no

[1] For standard IEEE1149.1 instructions the bypass multiplexer is activated (position 1)

[2] For non-standard instructions the bypass multiplexer is not activated (position 0)

7.2.4.7 TAPC2/TAPC3 description

TAPC2 and TAPC3 is use for debugging of the SC. TAPC2 is integrated in the ARM926EJ-S subsystem. TAPC3 is integrated in the ETB. Please refer to the ARM documentation for more information.

7.2.4.8 TAPC4 description

TAPC4 is used for debugging of DSP. It features a set of hard coded commands built in 8-bit instruction set. [Table 58](#) lists the available instructions.

Table 58: TAPC4 instruction set

Code	Instruction/ Test mode	Short description	Part of IEEE standard
1111 1111	BYPASS	boundary Scan Bypass mode: sets one register between serial data input and output - allows to skip TAPC4 when not used	standard
0001 0000	TESTIN	select ti register of TIO (DSP)	no
0001 0001	TESTOUT	select to register of TIO (DSP)	no
0001 0010	STATUS	select tios register of TIO (DSP) - the trap flag is automatically cleared by this access	no
0001 0011	DSPNMI	generate non-maskable interrupt for DSP	no

7.2.5 Application information

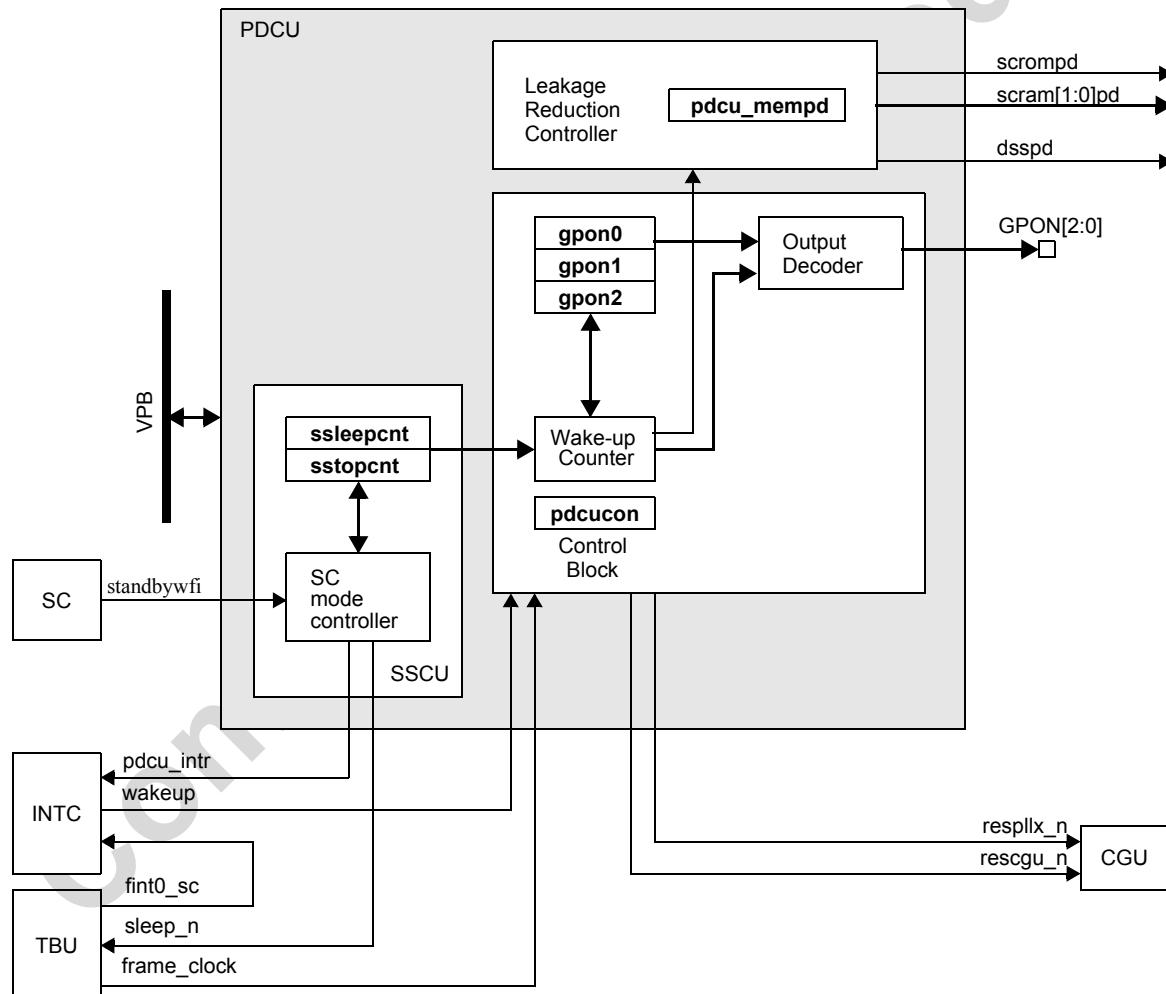
The TRST_N input should be kept low (or not driven to use internal pull down) until the supplies are settled.

7.3 PDCU - Power Down Control Unit

7.3.1 Features

- Control of the activity states of the digital circuit
- Control of the activity modes of the SC
- Flexible power switch control for SCROM, SCRAM and DSP subsystem

7.3.2 Block diagram



respllx_n: PLL_x enable

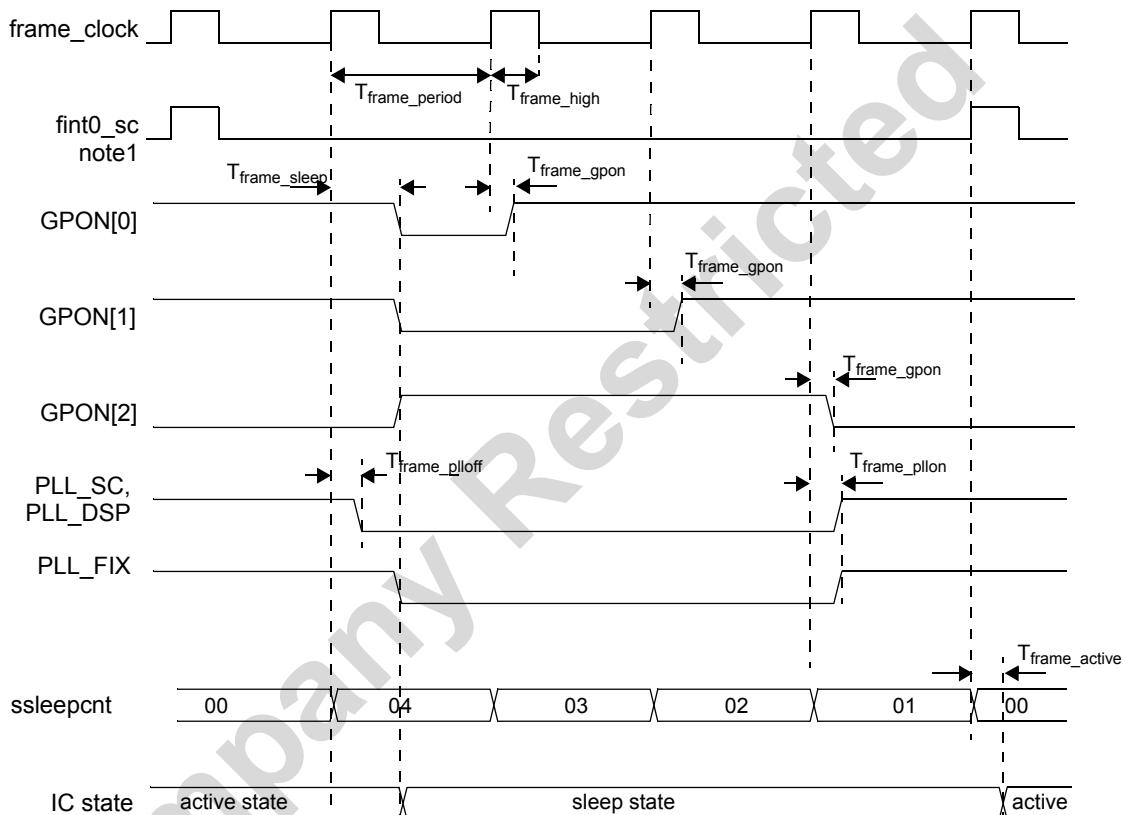
rescgu_n: CGU enable

Fig 27. PDCU block diagram (Note: Pins GPON[2:0] are not available at DVFD818x)

7.3.3 Hardware interface

Table 59: PDCU pin overview

Pin	Name	I/O	Description
GPON[2:0]	general purpose on	O	control signals to define the activation of external devices (Note: These pins are not available at DVFD8185 and DVFD8187)



(1) The signal fint0_sc is generated by the TBU and is depicted here to explain the function of the PDCU.

Fig 28. Timing diagram for the PDCU in sleep state entry and wake-up due to sleep counter expiration;
gp0n2[4:0] = 0b00001, gp0n1[4:0] = 0b00010, gp0n0[4:0] = 0b00011, gp0n_mask[2:0] = 0b000,
gp0n_p0l[2:0] = 0b011

Note : At DVFD818x the GPON[x] pins are not available

Table 60: AC characteristics of the PDCU

Symbol	Description	Min	Typ	Max	Unit
T_{frame_period}	period of the TBU frame in stop state	-	5	-	ms
T_{frame_high}	frame clock high phase in active state	-	$4 T_{clk32k}$	-	μs
	frame clock high phase in sleep state	-	$4 T_{clk32k}$	-	μs
T_{frame_gpon}	frame clock rising edge to GPON change	-	$2 T_{clk32}$	-	μs
T_{frame_sleep}	frame clock rising edge to start of sleep	-	$8 T_{clk32}$	-	μs

Table 60: AC characteristics of the PDCU...continued

Symbol	Description	Min	Typ	Max	Unit
T _{frame_active}	frame clock rising edge to start of active	-	3 T _{clk32}	-	μs
T _{frame_ploff}	frame clock rising edge to disable of pll's	-	3 T _{clk32}	-	μs
T _{frame_plon}	frame clock rising edge to enable of pll's	-	3 T _{clk32}	-	μs

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

7.3.4 Software interface

The PDCU control register can only be accessed by the SC.

Table 61: SC accessible registers

Name	Description	I/O	Reset
pdcucon	power-down control unit register	R/W	0x0001
gpon0	GPON[0] control register	R/W	0x0007
gpon1	GPON[1] control register	R/W	0x0007
gpon2	GPON[2] control register	R/W	0x0007
sstopcnt	SC stop counter register	R/W	0x0000
ssleepcnt	SC sleepcounter register	R/W	0x0000
pdcu_mempd	power-off control for memories and DSP subsystem	R/W	0x0000 0000

Table 62: Register pdcucon

Bit	Symbol	Access	Value	Description
15 to 12		R	0*	reserved
11	pdcu_int	R/C	0*	interrupt not pending
			1	interrupt pending - this bit is asserted at the beginning of the first active frame after a SC Sleep, Stop or Idle mode
10 to 9	-	R	0*	reserved
8	psm_stat	R	0*	Status of bmppsm mode
			1	bmppsm mode inactive - IF_CLK_PLL is running at full speed.
			0*	bmppsm mode active - IF_CLK_PLL running at 1/12 normal rate
7	psm_off ^[2]	R/S	0*	Exit from bmppsm mode
			1	no change to current state
			0*	Initiate exit from bmppsm mode - completed when psm_stat becomes '0'. (This bit is auto-cleared)
6	psm_on ^[2]	R/S	0*	Enter bmppsm mode
			1	no change to current state
			0*	Initiate entry into bmppsm mode - completed when psm_stat becomes '1'. (This bit is auto-cleared)
5 to 3	gpon_mask[2:0] ^[2]	R/W	0b000*	GPON[2:0] mask definition
			0	the corresponding GPON pin is defined by the sleep state
			1 [1]	the corresponding GPON pin is a general purpose output and has the value set in gpon_pol field
2 to 0	gpon_pol[2:0] ^[2]	R/W	0b001*	GPON[2:0] polarity definition
			0	the corresponding GPON pin is low active
			1	the corresponding GPON pin is high active

[1] The gpon_mask[0] bit should not be set to 1 because it is used internally.

[2] Not supported at DVFD818x (keep reset state)

Table 63: Register gpon0

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	gpon0[4:0]	R/W	0b00111*	number of frames (before the end of SLEEP state) when GPON[0] becomes active (pin not available at DVFD8185 and DVFD8187, but function for programming wake up by PDCU can be used)

Table 64: Register gpon1

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	gpon1[4:0]	R/W	0b00111*	number of frames (before the end of SLEEP state) when GPON[1] becomes active (pin not available at DVFD8185 and DVFD8187, but function for programming wake up by PDCU can be used)

Table 65: Register gpon2

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	gpon2[4:0]	R/W	0b00111*	number of frames (before the end of SLEEP state) when GPON[2] becomes active (pin not available at DVFD8185 and DVFD8187, but function for programming wake up by PDCU can be used)

Table 66: Register ssleepcnt

Bit	Symbol	Access	Value	Description
15 to 0	ssleepcnt[15:0]	R/W	0*	write access: number of frames to sleep; read access in the first frame after wake-up: number of frames left before programmed wake-up [1][2]

[1] ssleepcnt and sstopcnt can't be used at the same time - only last written register is valid.

[2] 0xFFFF should never be written and indicates an abort condition.

Table 67: Register sstopcnt

Bit	Symbol	Access	Value	Description
15 to 0	sstopcnt[15:0]	R/W	0*	write access: number of frames to stop; read access in the first frame after wake-up: number of frames left before programmed wake-up [1][2]

[1] ssleepcnt and sstopcnt can't be used at the same time - only last written register is valid.

[2] 0xFFFF should never be written and indicates an abort condition.

Table 68: Register pdcu_mempd^[2]

Bit	Symbol	Access	Value	Description
31 to 30	scrompd	R/W		SCROM PowerDown (PD) mode control
			0b00*	SCROM is in ON mode
			0b01	SCROM powerdown controlled by IC sleep state (HW control)
			0b10	SCROM is in WU (wakeup, pre charge) mode
			0b11	SCROM is in global powerdown (GPD) mode
29 to 28	scrompos	R/W		SCROM Power Off Switch (POS) control
			0b00*	SCROM is always powered

Table 68: Register pdcu_mempd^[2]...continued

Bit	Symbol	Access	Value	Description
			0b01	SCROM is powered-off in IC sleep state (HW control)
			0b10	SCROM pre charge
			0b11	SCROM is powered-off immediately
27 to 12	-	R	0*	reserved
11 to 10	dsspd[1:0]	R/W		controls the power supply of the DSS (DSP subsystem) ^[1]
			0b00*	DSS is always powered
			0b01	DSS is powered-off in IC sleep state (HW control)
			0b10	DSS pre charge
			0b11	DSS is powered-off immediately
9 to 4	-	R	0*	reserved
3 to 2	scram1pd[1:0]	R/W		controls the power supply of the SCRAM bank 1 ^[1]
			0b00*	memory is always powered
			0b01	memory is powered-off in IC sleep state (HW control)
			0b10	memory pre charge
			0b11	memory is powered-off immediately
1 to 0	scram0pd[1:0]	R/W		controls the power supply of the SCRAM bank 0 ^[1]
			0b00*	memory is always powered
			0b01	memory is powered-off in IC sleep state (HW control)
			0b10	memory pre charge
			0b11	memory is powered-off immediately

[1] The content of the RAM is lost when the power is switched-off.

[2] SCROM has both a powerdown control (PD) in bits 31 to 30 and a power switch (POS) control in bits 29 to 28. DSP subsystem and SCRAM have only a power switch control.

CAUTION

Do not access the TMU if DSP is powered-off!



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

7.3.5 Functional description

7.3.5.1 General concept

From power-down point of view, the DSP is considered as a peripheral of the SC. Although it has own controls of idle mode, its full swing of power modes (active, idle, stop, off) can be only controlled by the SC by writing into the appropriate registers of the CGU and PDCU.

7.3.5.2 Operating modes

The DVFD818x Family is designed for low power consumption. The power-down control unit reduces the current drawn by the IC when no activity is required. This is achieved by setting the device into different modes and states. Note that a state is specific for the chip set and its individual ICs, while a **mode** is specific to a core. For convenience, modes and states have the same naming: sleep, stop and active. Power-down modes can be controlled for the SC and the DSP independently.

The definition of the Operating modes is given in [Table 69](#) and [Table 70](#).

Table 69: SC operating modes

Mode	Description	Wake-up
Active	SC core and peripheral clocks are running; some peripheral clocks can be switched off individually (refer to Section 7.1)	none
Idle	SC core, TCM, ETM and cache clock is switched off; the peripheral clocks are running for all enabled peripherals (refer to Section 7.1); to enter Idle mode the user has to: <ul style="list-style-type: none"> • do not load the ssleepcnt/sstopcnt registers • enter in ARM Wait For Interrupt mode [1] 	all enabled interrupts except the following interrupts related to trace and debug: etb_etbfull, etb_acqcomp, arm9_commrx, arm9_commtx
Stop	SC core and peripheral clocks are switched off; Crystal oscillator is running as all enabled PLLs. To enter Stop mode the user has to: <ul style="list-style-type: none"> • load stop counter register sstopcnt[3] • enter in ARM Wait For Interrupt mode [1] 	<ul style="list-style-type: none"> • expiration of stop counter • BMP - int_bmptime • KBS • EXTINT • EXTINT alternate routing • APU - int_pwr_fail / int_ana • TBU - fint0_sc_irq • see [4]

Table 69: SC operating modes...continued

Mode	Description	Wake-up
Sleep	<p>SC core and peripheral clocks are switched off; all PLLs are switched off.</p> <p>BMPTBU timing can be kept by running clk_bmp directly from bbcki by setting cgufixcon.bmpclkby to 1. Wake up latency is defined by the value of the gpon[2:0] registers and is always at least 1 frame.</p> <p>To enter Sleep mode the user has to:</p> <ul style="list-style-type: none"> • load sleep counter register ssleepcnt^[3] • enter in ARM Wait For Interrupt mode ^[1] 	<ul style="list-style-type: none"> • expiration of sleep counter • BMP - int_bmptime • KBS • EXTINT • EXTINT alternate routing • APU - int_pwr_fail / int_ana • TBU - fint0_sc_irq • see ^[4]
bmppsm	<p>SC core and peripheral clocks are switched off; all PLLs are switched off, BBCKI input frequency is reduced by a factor of 12. clk_bmp can continue to run from this reduced clock. SC can also run on this reduced clock by connecting its clocks directly to BBCKI in the CGU. Alternatively (and most likely) bmppsm mode can be combined with idle or sleep mode, to achieve maximum power saving.</p> <p>To enter bmppsm mode the user has to:</p> <ul style="list-style-type: none"> • enable SC/FIX PLL bypasses for the SC, AHB and BMP by setting cgufixcon.scclksby, cgufixcon.fixclksby, and cgufixcon.clkbmpby to 1. These clocks are now fed by BBCKI. • switch off all PLLs. • reduce/switch off unneeded power supplies. • enter bmppsm mode via pdcucon.psm_on • update cgudivcon.clk32kdiv to match the new BBCKI frequency • If sleep mode is also required, load the sleep counter ssleepcnt • If sleep or idle is required, enter in ARM Wait For Interrupt mode ^[1] <p>Alternatively, bmppsm mode can be entered via hardware control by:</p> <ul style="list-style-type: none"> • setting the cgufixcon.hwctrlby bit in the CGU • entering bmppsm mode via pdcucon.psm_on <p>In this second case, all bypasses are automatically activated, PLLs are switched off and so on. Refer to the CGU section for details.</p> <p>To exit bmppsm mode, the user has to write 1 to pdcucon.psm_off.</p> <p>If idle or sleep mode is also entered, then the same wakeup sources defined for these modes can be used to re-activate the SC, which will in turn be able to exit bmppsm mode via pdcucon.psm_off.</p>	<ul style="list-style-type: none"> • expiration of sleep counter • BMP - int_bmptime • KBS • EXTINT • EXTINT alternate routing • APU - int_pwr_fail / int_ana • TBU - fint0_sc_irq
Off	<p>All clocks including crystal oscillator are switched off. Required power supply rails are set to a min. level, other rails are switched off.</p> <p>To enter Off mode the user has to:</p> <ul style="list-style-type: none"> • enable SC PLL bypass • switch off all PLLs • power down any unneeded blocks using pdcu_mempd • reduce/switch off unneeded power supplies • switch off crystal oscillator in register daif_rxctrl (dcxo_off)^[2] for PNX818x • enter in ARM Wait For Interrupt mode ^{[1][2]} 	<p>wake-up only possible by applying a RESET. Possible sources:</p> <ul style="list-style-type: none"> • NPORST (power cycle) • <u>RSTBB</u> (as <u>async reset</u> source forces RSTAPU)

[1] Refer to [12.] for more details on how to enter the ARM Wait For Interrupt mode with cp15 instruction

[2] After writing to register **daif_rxctrl** it takes additional 20 cycles of clk_if to transfer this request (Section 9.11 "DAIF - Digital to Analog Interface"). It must be enough time to safely enter the ARM wait for interrupt mode.

- [3] The counters bmp_ref and stopcnt/ssleepcnt are running independently.
In a DECT application based on DSP Group software the int_bmptime should be used for wake up. Therefore the wakeup time of sstopcnt/ssleepcnt should be programmed larger than the bmp_ref value
- [4] Beside the interrupts indicated as wake up sources any other block able to raise an interrupt can wake up the system if enabled. But it is not recommended to take use of it because these blocks operate than outside the specification.

The DSP has limited power-down control functionality: its operating mode does not interfere with IC state.

Table 70: DSP operating modes

Mode	Description	Wake-up
Active	DSP core and peripheral clocks are running; some peripheral clocks can be switched off individually (refer to Section 7.1)	none
Idle	DSP core clock is switched off; the peripheral clocks are running for all enabled peripherals (refer to Section 7.1) to enter Idle mode the user has to either: <ul style="list-style-type: none"> execute idle instruction (while cgudspcon.diclkidleen bit is set) two nop instructions or <ul style="list-style-type: none"> clear the cgudspcon.dcclken bit with the SC 	all enabled interrupts
Stop	DSP core and peripheral clocks are switched off. To enter Stop mode the user has to either: <ul style="list-style-type: none"> execute idle instruction (while cgudspcon.diclkidleen bit is not set) two nop instructions or <ul style="list-style-type: none"> clear the cgudspcon.dspclken bit with the SC 	either: <ul style="list-style-type: none"> all enabled interrupts (if entered via Idle instruction) or <ul style="list-style-type: none"> set the cgudspcon.dspclken bit with the SC
Off	All DSP core clock are switched off; DSS power supply is switched off. To enter Off mode the user has to: <ul style="list-style-type: none"> disable the dsp clocks in the cgu (refer to Section 7.1) switch off dss power supply (pdcu_mempd.dsspd) 	<ul style="list-style-type: none"> switch on power supply (pdcu_mempd.dsspd) enable clocks.

Table 71: Active clocks at operating states

Name	Reset	Active	IDLE	STOP [1]	Sleep [1][2]	PSM+ Sleep [1][2]	Off
armclk	clk32k	✓	-	-	-	-	-
hclk	clk32k	✓	✓	clk32k	clk32k	clk32k	-
pclk1	clk32k	✓	✓	clk32k	clk32k	clk32k	-
pclk2	clk32k	✓	✓	clk32k	clk32k	clk32k	-
bmp_hclk	clk32k	✓	✓	clk32k	clk32k	clk32k	-
clk_bmp	clk32k	✓	✓	bbcki	bbcki	bbcki	-
clk26m	clk32k	✓	✓	clk32k	clk32k	clk32k	-
clk13m	clk32k	✓	✓	clk32k	clk32k	clk32k	-
clk13mm	clk32k	✓	✓	clk32k	clk32k	clk32k	-
clk32k	✓	✓	✓	✓	✓	✓	-
bbcki	✓	✓	✓	✓	✓	1.152MHz	-

[1] Clocks can be activated at speed of clk32k in register see [Table 40](#)

[2] Proposed configuration for sleep and bmppsm mode for a cordless handset

7.3.5.3 Sleep and Stop modes

The duration of Stop and Sleep modes can be programmed with a granularity of TBU frames in the sleep and stop counters. The Stop and Sleep modes are terminated by the expiration of the sleep and stop counters or by any pending and enabled interrupt. Once a stop or sleep counter has reached 0, it does not continue to decrement and stays at 0. If the Power-down mode is left because of an interrupt, and the counters have not expired yet, they continue to decrement until they reach the value 0.

Writing n into the stop/sleep count register during frame m causes the frame interrupt to occur at the beginning of the TBU frame $n+m+1$. Thus the shortest duration of stop and sleep is two TBU frames. The sleep and stop counters are always loaded at the begin of a frame. Writing a 0x0000 into the **sstopcnt/ssleepcnt** registers clears the stop/sleep counters at the next frame clock.

The **pdcu_int** interrupt is asserted after the SC leaves the Idle, Stop and Sleep modes. Leaving the Sleep mode the **pdcu_int** interrupt is only asserted if it was forced by a wake up from the INTC.

To leave the STOP/Sleep mode when stop/sleep counter expires, the **fint0_sc** has to be enabled in the INTC.

7.3.5.4 Sleep, stop, and idle entry

The Sleep and Stop modes are entered at the next pulse of frame clock after the SC has entered idle mode (WFI instruction executed).

If the enter in ARM Wait For Interrupt mode happens shortly before a new frame starts, the Power-down mode lasts one frame longer than expected.

For the wake-up, the CGU hardware takes care that the core clocks are switched on two cycles after the peripheral clocks.

It is not allowed for the SC to go to stop/sleep mode when data transfers on the serial interfaces are still ongoing (including DMAU transfers). On the contrary it is always possible to switch to idle mode.

7.3.5.5 Wake-up counter

The PDCU features a wake-up counter which is used to generate the recovery timings when waking-up from sleep. This counter has an effect on the PLLs and CGU. The timings are performed based on the values programmed in **gpon0**, **gpon1**, and **gpon2** (see [Figure 28](#)).

When entering sleep, the wake-up counter is loaded with the maximum value
 programmed in **gpon0**, **gpon1** and **gpon2**: wake-up
 $\text{counter} = \max(\text{gpon0}, \text{gpon1}, \text{gpon2})$. Care should be taken to program at least one
gpon register different than zero. The wake-up counter starts to decrement when

- An interrupt occurs
- One of the stop/sleep counters has the same value than the value loaded into the wake-up counter.

The PLLs are always activated one frame before the end of sleep. The CGU is activated at the end of sleep.

PDCU special cases

1. If an interrupt occurs between the entry of ARM Wait For Interrupt mode and the next rising edge of the frame clock, then the sleep or stop Power-down mode is not entered - the SW detects this situation by the abort condition (sleep or stop counters = 0xFFFF)
2. If there is a pending interrupt when entering the ARM Wait For Interrupt mode, then the Power-down mode is not entered as this condition corresponds to a wake-up - the SW detects this situation as no PDCU interrupt is generated
3. It is not possible to go to stop or sleep in the same frame where the wake-up has occurred
4. If a sleep counter value is equal to the maximum value of gpon0, gpon1, gpon2 at the entry of sleep, then the wake-up counter is loaded with the maximum value decremented by one. This implies that:
 $\text{if } (\text{sleep counter at entry} = \max(\text{gpon0}, \text{gpon1}, \text{gpon2})) \text{ then } (\text{wake-up counter at entry} = \max(\text{gpon0}, \text{gpon1}, \text{gpon2}) - 1)$
 The external ICs are not switched off for one frame
5. If an interrupt occurs less than 1 **hclk** cycle after ARM Wait For Interrupt mode has been entered, the sleep state is not entered and the SC wakes-up at the begin of the next TBU frame - in this case the ABORT condition is not loaded into the counters - the SW detects this situation as no PDCU interrupt is generated.

7.3.5.6 RFCU inactive mode control

It is possible to switch the RFCU signals to inactive in order to avoid cross-currents into the RF-IC in case it is not powered in sleep state. This mode is handled by the BMP (**bmp_rfpu_con**) and not from the PDCU.

7.3.5.7 Leakage reduction by memory power-off

The DVFD818x Family implements advanced power switches to control the power supply for some blocks. In particular these switches are used to control the power supply of blocks which are not used by the application. Furthermore it is possible to switch-off the supply power for these blocks if the volatile content can be lost during sleep state. This functionality considerably decreases the static power dissipated during stand-by.

The control of the supply power is performed by on-die power switches in the VDDC domain. The I/O supply domains are not impacted when some areas of the IC are powered down. In particular the VDDCGU domain implements level shifters to cope with changes in the VDDC power domain.

The switches provoke a negligible power drop on the supply lines. The user does not observe any performance degradation when the power dissipation of the blocks increase as long as the operation is within the limits specified in the rating section.

7.3.5.8 Leakage reduction control

the register **pdcu_mempd** allows to control the power supply of the following blocks:

- SCROM
- DSS
- SCRAM, each bank separately

After reset or watchdog reset all blocks of the DVFD818x Family are powered. When the SC switches-off the power of a specific domain, the block is no longer accessible, all its inputs are driven to 0V and all the outputs of the block drive their reset value. The powered sections of the IC, see a block which is in its reset state. Due to the power of that block is internally switched-off, its static power dissipation is eliminated. When the SC tries to access a block which is not powered (SCRAM), it receives fixed dummy values.

The register **pdcu_mempd** allows to control the power of these blocks with two mechanisms:

- immediate action through SC programming - when the SC programs one of the power down fields with 0b11, then the power of the related block is immediately switched-off - all non-volatile data is lost and activity is stopped - before using a block that was powered-off, the SW has to program the register bits with 0b10 to pre charge and to respect the pre charge time. During this time the block is still in reset state. After the pre charge time, the setting in the **pdcu_mempd** has to be changed to operation 0b00.
- delayed action during baseband sleep state - when the SC programs one of the power down fields with 0b01, then the power of the related block is turned-off during all following sleep states in the frame (length defined by the application) after the transition to sleep - the data inside all registers and memories is lost and has to be retrieved after sleep state if required.

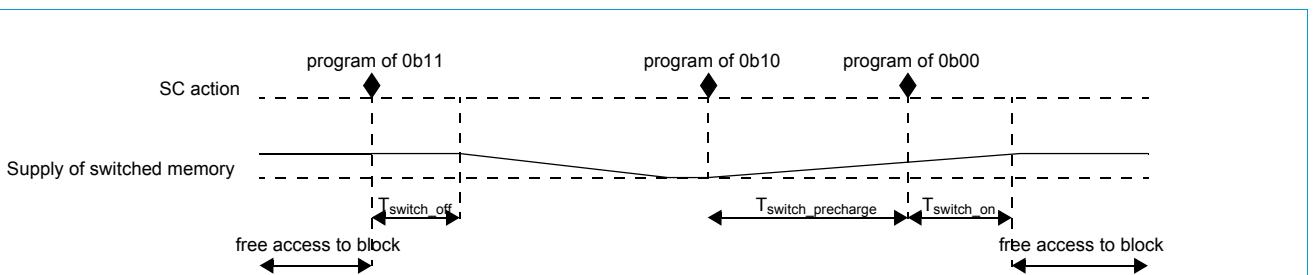


Fig 29. Power switch timing diagram

Table 72: Power switch timings

Symbol	Description	Typ	Unit
T _{switch_off}	time it takes for the power switches to stop conducting	1	μs
T _{switch_preamble}	time it takes pre charge block with small transistor	10	μs
T _{switch_on}	time it takes for the power switches to allow full operation	1	μs

7.3.5.9 Additional note

The control of the power switches provided by the DVFD818x Family is vital in order to extend battery life. The user has to consider to reduce the leakage further by reducing the VDDC supply voltage during the sleep state.

7.3.6 Application information

7.3.6.1 Timing diagrams for PDCU operation

Note: The GPON[2:0] pins are not available at DVFD818x

Table 73: PDCU application for sleep state entry

Register	Setting	Register	Setting	Register	Setting
gpon0	-	gpon_pol[0]	0	gpon_mask[0]	0
gpon1	-	gpon_pol[1]	1	gpon_mask[1]	0
gpon2	-	gpon_pol[2]	0	gpon_mask[2]	0

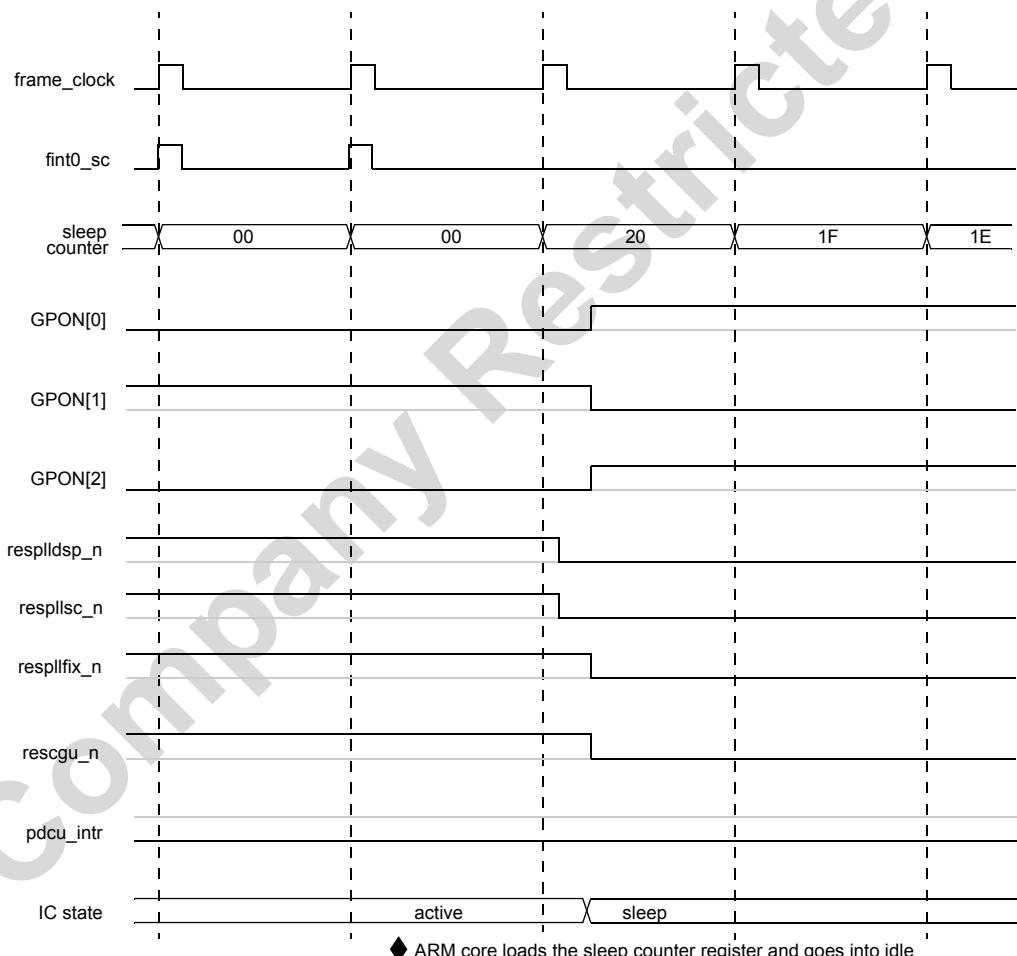


Fig 30. Application timing diagram for sleep state entry

Note: The GPON[2:0] pins are not available at DVFD818x

Table 74: PDCU application in sleep state wake-up due to expiration of sleep counter

Register	Setting	Register	Setting	Register	Setting
gpon0	3	gpon_pol[0]	1	gpon_mask[0]	0
gpon1	1	gpon_pol[1]	0	gpon_mask[1]	0
gpon2	1	gpon_pol[2]	1	gpon_mask[2]	0

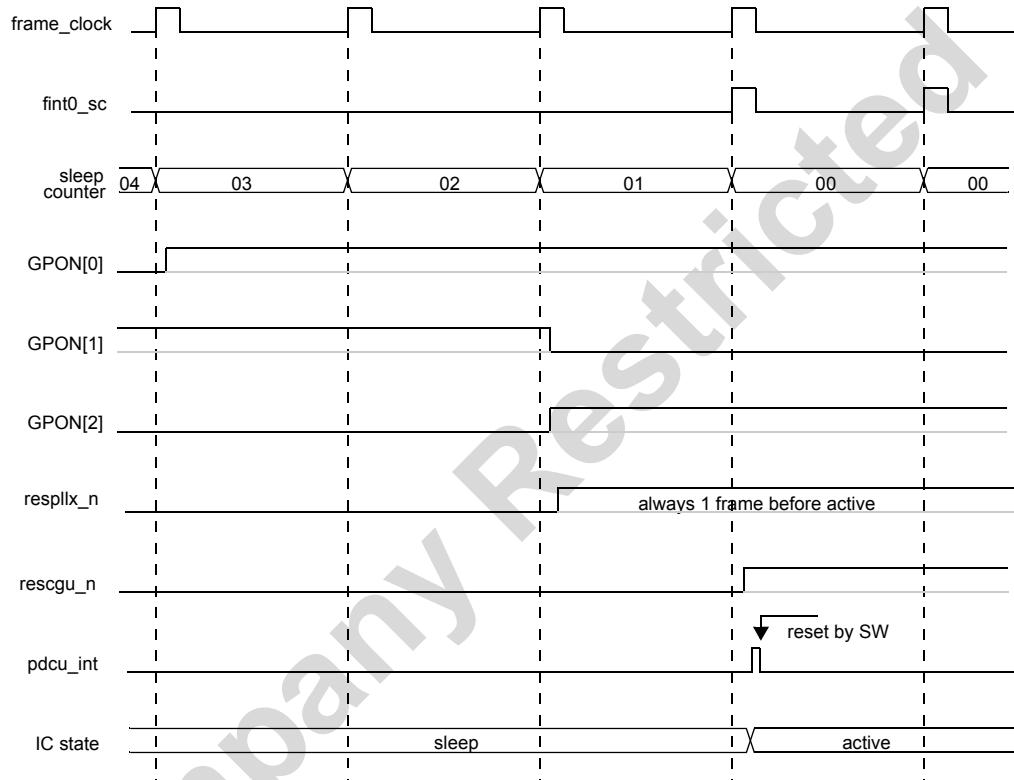


Fig 31. Application timing diagram for the PDCU in sleep state wake-up due to expiration of sleep counter
Note: The GPON[2:0] pins are not available at DVFD818x

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 75: PDCU application in sleep state wake-up due to external interrupt

Register	Setting	Register	Setting	Register	Setting
gpon0	2	gpon_pol[0]	1	gpon_mask[0]	0
gpon1	1	gpon_pol[1]	1	gpon_mask[1]	1
gpon2	1	gpon_pol[2]	1	gpon_mask[2]	0

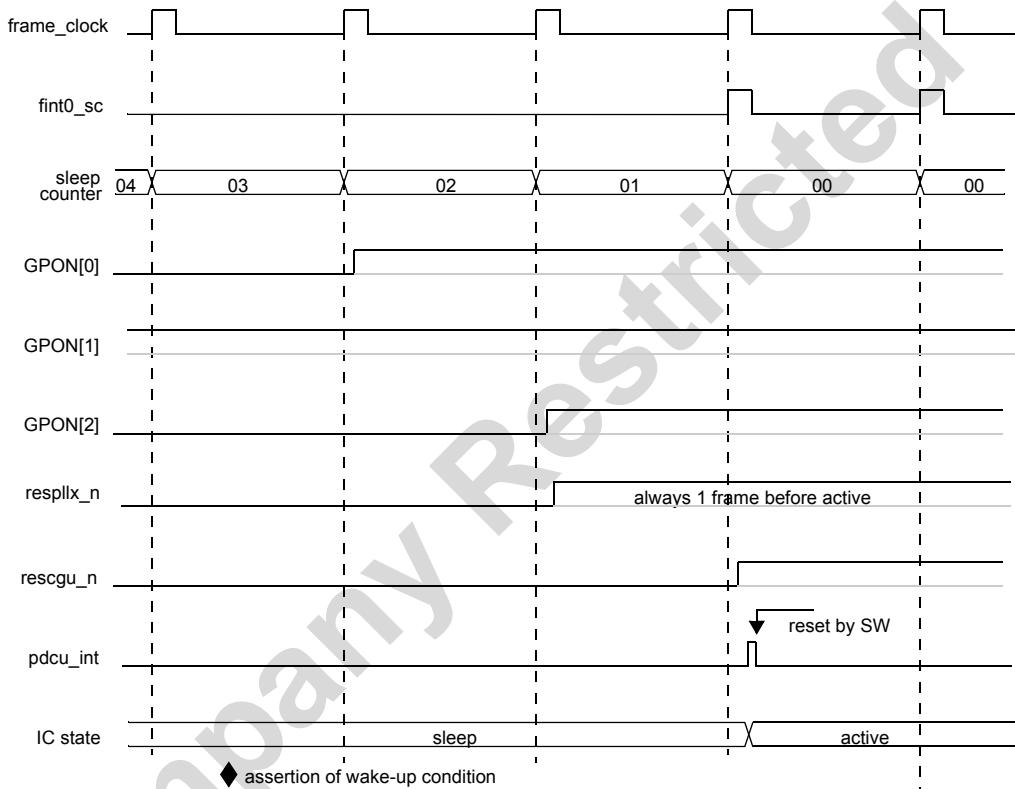


Fig 32. Application timing diagram for the PDCU in sleep state wake-up due to external interrupt - note: because of gpon_mask[1] = 1, gpon1 is not relevant

Note: The GPON[2:0] pins are not available at DVFD818x

Table 76: PDCU application in stop state wake-up due to external interrupt

Register	Setting	Register	Setting	Register	Setting
gpon0	-	gpon_pol[0]	0	gpon_mask[0]	1
gpon1	-	gpon_pol[1]	1	gpon_mask[1]	1
gpon2	-	gpon_pol[2]	1	gpon_mask[2]	1

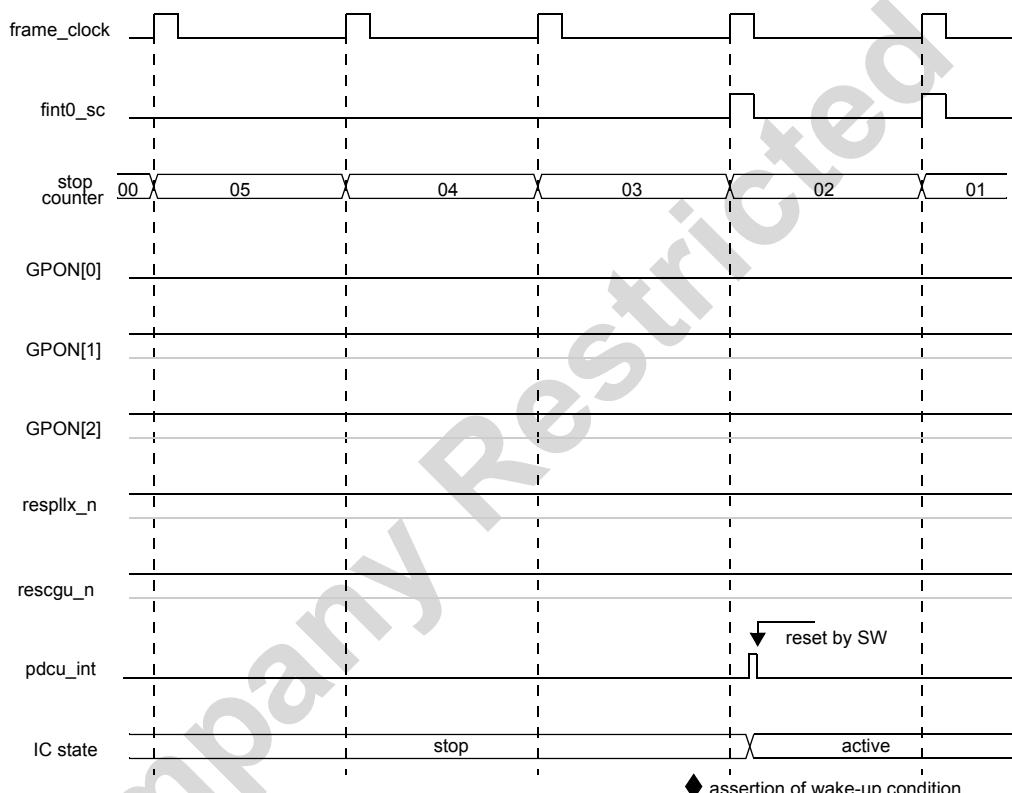
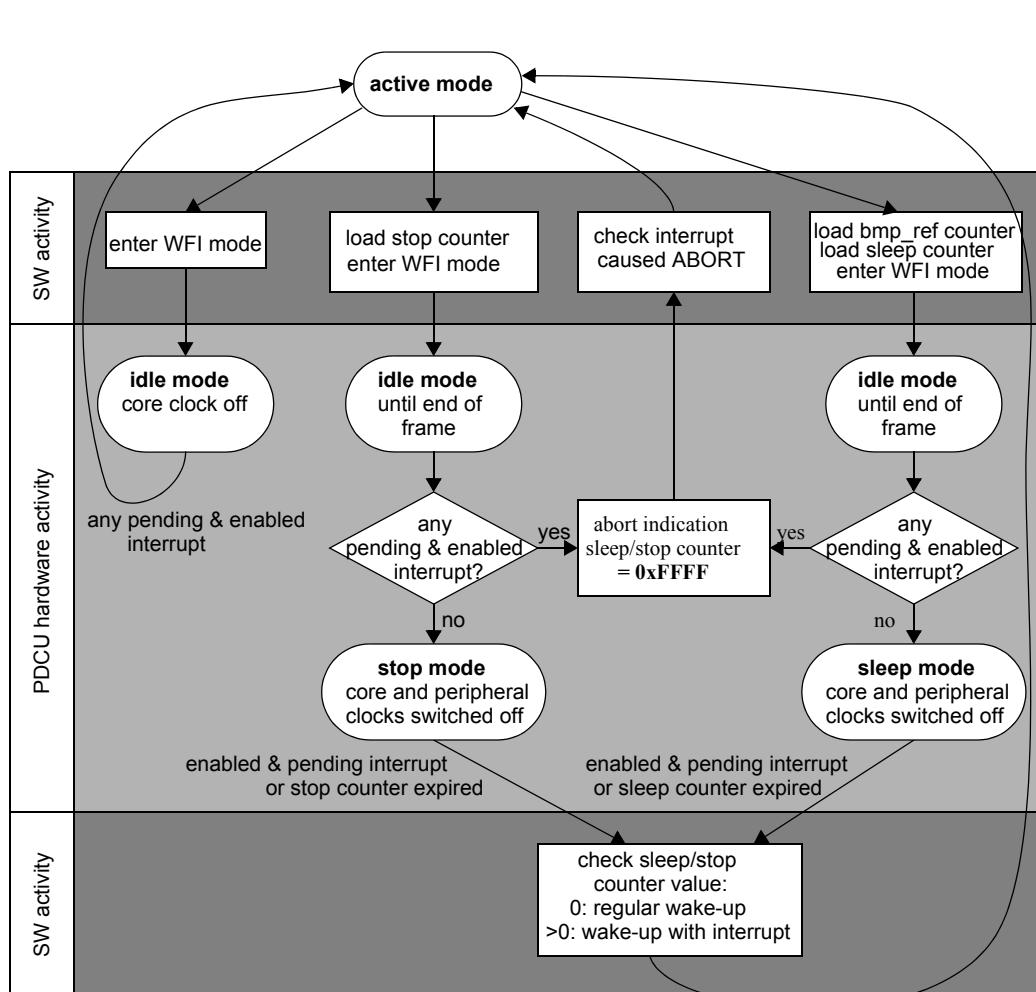


Fig 33. Application timing diagram for the PDCU wake-up from stop due to external interrupt - note: in stop state the GPON do not change

Note: The GPON[2:0] pins are not available at DVFD818x

7.3.6.2 Using the Power-down modes for the SC

A description of the SC software handling is shown in [Figure 34](#). Only the major cases are shown - for a description of the special cases please refer to [Section "PDCU special cases"](#).



(1) WFI refer to ARM Wait For Interrupt through cp15 access

Fig 34. SC Power-down mode sequence

7.3.6.3 Wake up with bmptime

To allow BMPTBU activity in STOP and SLEEP mode for wake up by bmptime, both the bit **cgusleepsc.bmpslen** and the bit **cgufixcon.bmpclkby** have to be set. In that configuration the BMP core clock is taken directly from the PLL input (bbcki). This is applicable only for a DECT application with a 13.824MHz crystal.

7.3.6.4 ARM/AHB clock gating in sleep mode

It is not recommended to enable the ARM/AHB clock in sleep mode (bit **cgu-sleepsc.ahbsen**). If enabled the ARM will start code execution after assertion of the wake up condition with clk32k while PDCU is still in sleep state (counting down and PLLs are disabled). Accesses or configurations of peripherals can give wrong value back or are not applied.

For debugging the bit needs to be set (to generate RTCK). In that case after assertion of wake up condition the wake up routine should poll on fint0_sc true before code execution is continued.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Company Restricted

7.4 SCON - System Configuration Block

7.4.1 Features

- Mask-programmable IC version register
- Allows the programming of several system controller configurations
- Allows the setting of functional pin multiplexing
- Pull-up/pull-down/repeater control for GPIOA, GPIOB GPIOC lines
- System protection control
- HMP/FMP pad speed control (for 1.8V supply level only)

7.4.2 Block diagram

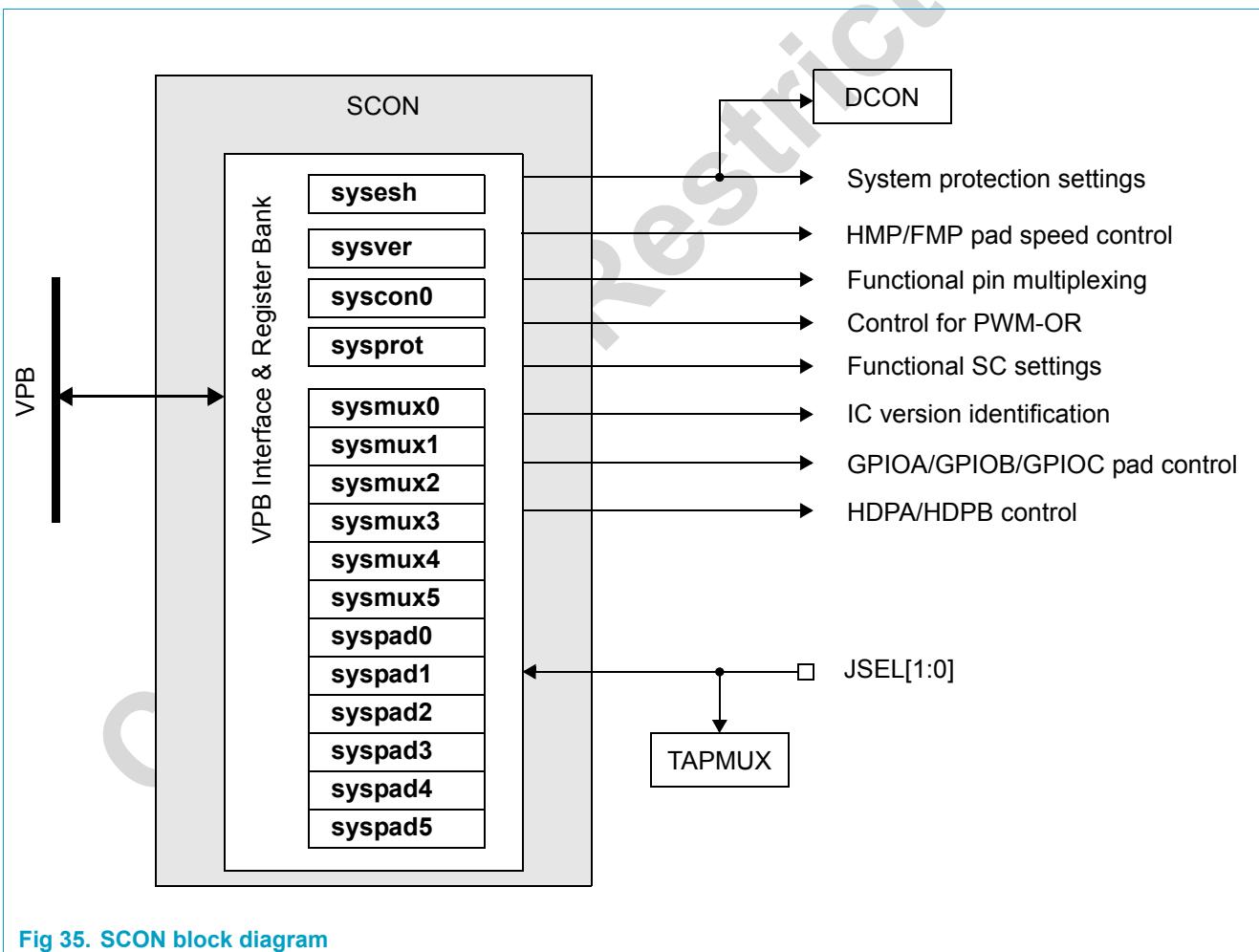


Fig 35. SCON block diagram

7.4.3 Hardware interface

The SCON block is connected to the JSEL[1:0] pins which are described in [Section 7.2](#).

7.4.4 Software interface

Table 77: Register overview of SCON

Name	Description	I/O	reset
sysver	system version register	R	0x4122 [1]
syscon0	system configuration register 0	R/W	- [2]
sysprot	system protection register	R/W	0x0000 0002
sysmux0	system multiplexing register 0	R/W	0x0005 0141
sysmux1	system multiplexing register 1	R/W	0x0140 0010
sysmux2	system multiplexing register 2	R/W	0x0000 0000
sysmux3	system multiplexing register 3	R/W	0xAA95 5540
sysmux4	system multiplexing register 4	R/W	0x5555 5555
sysmux5	system multiplexing register 5	R/W	0x0000 0001
syspad0	system I/O pad configuration register 0	R/W	0x1F5A 5296
syspad1	system I/O pad configuration register 1	R/W	0x043F 5761
syspad2	system I/O pad configuration register 2	R/W	0xAFF FFFF
syspad3	system I/O pad configuration register 3	R/W	0xA6A3 C3EB
syspad4	system I/O pad configuration register 4	R/W	0x0000 0000
syspad5	system I/O pad configuration register 5	R/W	0xAAAA A568
sysesh	system HMP/FMP pad speed control register	R/W	0xFFFF FFFF

[1] The reset value might changes for the different versions of DVFD818x

[2] The reset value depends as well on the level of JSEL[1:0]

Table 78: Register sysver

Bit	Symbol	Access	Value	Description
15 to 12	icfab[3:0]	R		silicon foundry identification
			0b0001	SSMC
			0b0010	MOS4
			0b0011	reserved
			0b0100*	TSMC
			0b1111	Quickturn or FPGA
11 to 8	icid[3:0]	R		IC identification
			0b0001*	OM48381
			0b0010	reserved
			0b0011	reserved
			0b0100	reserved
			0b0101	reserved
7 to 4	icmask[3:0]	R		full mask set version
			0b0001	icid[3:0]-1x
			0b0010*	icid[3:0]-2x (and so on)
3 to 0	icmet[3:0]	R		metal mask set version
			0b0001	icid[3:0]-xA
			0b0010*	icid[3:0]-xB (and so on)

Table 79: Register syscon0

Bit	Symbol	Access	Value	Description
31 to 27	hdpbcon[4:0]	R/W	0*	HDPB[27:0] pin configuration according to Section 6.5
26 to 24	ebi1_delay_ctrl	R/W		Delay of the EBI1 control, address and data signals versus CLOCKBURST (FMP53)
			0b000*	no delay
			0b001	delay by 1/2 cycle of hclk
			0b010	2.5ns delay typical (5ns max.)
			0b011	3.5ns delay typical (6.5ns max.)
			0b100	reserved
			0b101	reserved
			0b110	reserved
			0b111	reserved
23 to 21	-	R	0b000000*	reserved
20	pad_vref_ei	R/W		Internal pad reference generator control. Used for test only
			0*	normal mode. Pad reference is generated, pads are fully functional
			1	power down mode. Pad reference is off, consumption is reduced. Pads are still functional, but speed is reduced by a factor of 1.2 to 2
19 to 17	-	R	0b00*	reserved
16	dminus	R	[2]	D- detection
			0	USBDM signal low
			1	USBDM signal high
15	dplus	R	[2]	D+ detection
			0	USBDP signal low
			1	USBDP signal high
14	vbus	R	[2]	USB VBUS detection
			0	USB VBUS (VDDH) not available
			1	USB VBUS available
13 to 12	jselstat[1:0]	R	[2]	status of JSEL[1:0] input
11 to 10	pwmor3[1:0]	R/W		PWM3 or control
			0b00*	PWM3 is the output signal of PWM3
			0b01	PWM3 is the logic OR between the outputs of PWM3 and PWM1
			0b10	PWM3 is the logic OR between the outputs of PWM3 and PWM2
			0b11	PWM3 is the logic OR between the outputs of PWM3 and PWM2 and PWM1
9 to 8	pwmor2[1:0]	R/W		PWM2 or control
			0b00*	PWM2 is the output signal of PWM2
			0b01	PWM2 is the logic OR between the outputs of PWM2 and PWM1
			0b10	PWM2 is the logic OR between the outputs of PWM2 and PWM3
			0b11	PWM2 is the logic OR between the outputs of PWM3 and PWM2 and PWM1
7 to 6	reserved		0b00*	should be set to 0b00
5 to 1	hdpacon[4:0]	R/W	0*	HDPA[12:0] pin configuration according to Section 6.5

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 79: Register syscon0...continued

Bit	Symbol	Access	Value	Description
0	hdpen	R/W		HDP port select
			0	interface is disabled (no activity)
			1*	interface is active

[1] The switching of these bits may only happen during pseudo-static situations - the SC needs to make sure that no access performed at the same time to the involved circuitry.

[2] The reset value of these bits depend on the value available at the primary pin.

Table 80: Register sysprot

Bit	Symbol	Access	Value	Description
31 to 5	-	R	0*	reserved
4	jtag_dis	R/W		JTAG disable bit
			0*	the DVFD818x JTAG debugging access is allowed
			1	the DVFD818x JTAG / debugging access is avoided In this mode access to the EmbeddedICE-RT and to the ETM9/ETB by means of the JTAG port are disabled. In addition ETB access via AHB is disabled. The ARM926EJ-S debug enable pin is forced inactive. Furthermore it is not possible to generate NMI interrupts towards DSP.
3 to 2	-	R	0*	reserved
1	security [1]	R/W		secure mode enable bit
			0	the DVFD818x is operating in debug mode
			1*	the DVFD818x is operating in secure mode
0	lock	R/S	0*	bits in register sysprot can be set and cleared
			1	bits in register sysprot can be set at any time, but not cleared anymore - this behavior also holds for lock itself - as a consequence only a reset can clear this bit

[1] This bit is reset with $\overline{\text{RSTBB}}$ only (i.e. not reset by a watchdog reset)

Table 81: Register sysmux0

Bit	Symbol	Access	Value	Description
27 to 26	gpioa13c	R/W	0b00*	GPIOA13 pin multiplexing
23 to 22	gpioa11c	R/W	0b00*	GPIOA11 pin multiplexing
11 to 10	gpioa5c	R/W	0b00*	GPIOA5 pin multiplexing
5 to 4	gpioa2c	R/W	0b00*	GPIOA2 pin multiplexing
3 to 2	gpioa1c	R/W	0b00*	GPIOA1 pin multiplexing

[1] See [Section 6.4](#) for more information

Table 82: Register sysmux1

Bit	Symbol	Access	Value	Description
31 to 30	gpioa31c	R/W	0b00*	GPIOA31 pin multiplexing
29 to 28	gpioa30c	R/W	0b00*	GPIOA30 pin multiplexing
27 to 26	gpioa29c	R/W	0b00*	GPIOA29 pin multiplexing
25 to 24	gpioa28c	R/W	0b01*	GPIOA28 pin multiplexing
23 to 22	gpioa27c	R/W	0b01*	GPIOA27 pin multiplexing
21 to 20	gpioa26c	R/W	0b00*	GPIOA26 pin multiplexing
19 to 18	gpioa25c	R/W	0b00*	GPIOA25 pin multiplexing
17 to 16	gpioa24c	R/W	0b00*	GPIOA24 pin multiplexing
7 to 6	gpioa19c	R/W	0b00*	GPIOA19 pin multiplexing
5 to 4	gpioa18c	R/W	0b01*	GPIOA18 pin multiplexing
1 to 0	gpioa16c	R/W	0b00*	GPIOA16 pin multiplexing

[1] See [Section 6.4](#) for more information

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 83: Register sysmux2

Bit	Symbol	Access	Value	Description
23 to 22	gpiob11c	R/W	0b00*	GPIOB11 pin multiplexing
21 to 20	gpiob10c	R/W	0b00*	GPIOB10 pin multiplexing
19 to 18	gpiob9c	R/W	0b00*	GPIOB9 pin multiplexing
17 to 16	gpiob8c	R/W	0b00*	GPIOB8 pin multiplexing
15 to 14	gpiob7c	R/W	0b00*	GPIOB7 pin multiplexing
13 to 12	gpiob6c	R/W	0b00*	GPIOB6 pin multiplexing
11 to 10	gpiob5c	R/W	0b00*	GPIOB5 pin multiplexing
9 to 8	gpiob4c	R/W	0b00*	GPIOB4 pin multiplexing
7 to 6	gpiob3c	R/W	0b00*	GPIOB3 pin multiplexing
5 to 4	gpiob2c	R/W	0b00*	GPIOB2 pin multiplexing
3 to 2	gpiob1c	R/W	0b00*	GPIOB1 pin multiplexing
1 to 0	gpiob0c	R/W	0b00*	GPIOB0 pin multiplexing

[1] See [Section 6.4](#) for more information

Table 84: Register sysmux3

Bit	Symbol	Access	Value	Description
29 to 28	gpiob30c	R/W	0b10*	GPIOB30 pin multiplexing
27 to 26	gpiob29c	R/W	0b10*	GPIOB29 pin multiplexing
25 to 24	gpiob28c	R/W	0b10*	GPIOB28 pin multiplexing
23 to 22	gpiob27c	R/W	0b10*	GPIOB27 pin multiplexing
21 to 20	gpiob26c	R/W	0b01*	GPIOB26 pin multiplexing
19 to 18	gpiob25c	R/W	0b01*	GPIOB25 pin multiplexing
17 to 16	gpiob24c	R/W	0b01*	GPIOB24 pin multiplexing
15 to 14	gpiob23c	R/W	0b01*	GPIOB23 pin multiplexing
13 to 12	gpiob22c	R/W	0b01*	GPIOB22 pin multiplexing
11 to 10	gpiob21c	R/W	0b01*	GPIOB21 pin multiplexing
9 to 8	gpiob20c	R/W	0b01*	GPIOB20 pin multiplexing
7 to 6	gpiob19c	R/W	0b01*	GPIOB19 pin multiplexing
5 to 4	gpiob18c	R/W	0b00*	GPIOB18 pin multiplexing

[1] See [Section 6.4](#) for more information

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 85: Register sysmux4

Bit	Symbol	Access	Value	Description
31 to 30	gpioc15c	R/W	0b01*	GPIOC15 pin multiplexing
29 to 28	gpioc14c	R/W	0b01*	GPIOC14 pin multiplexing
25 to 24	gpioc12c	R/W	0b01*	GPIOC12 pin multiplexing
13 to 12	gpioc6c	R/W	0b01*	GPIOC6 pin multiplexing
11 to 10	gpioc5c	R/W	0b01*	GPIOC5 pin multiplexing
9 to 8	gpioc4c	R/W	0b01*	GPIOC4 pin multiplexing
7 to 6	gpioc3c	R/W	0b01*	GPIOC3 pin multiplexing
5 to 4	gpioc2c	R/W	0b01*	GPIOC2 pin multiplexing
3 to 2	gpioc1c	R/W	0b01*	GPIOC1 pin multiplexing
1 to 0	gpioc0c	R/W	0b01*	GPIOC0 pin multiplexing

[1] See Section 6.4 for more information

Table 86: Register sysmux5

Bit	Symbol	Access	Value	Description
29 to 28	gpioc30c	R/W	0b00*	GPIOC30 pin multiplexing
27 to 26	gpioc29c	R/W	0b00*	GPIOC29 pin multiplexing
25 to 24	gpioc28c	R/W	0b00*	GPIOC28 pin multiplexing
23 to 22	gpioc27c	R/W	0b00*	GPIOC27 pin multiplexing
21 to 20	gpioc26c	R/W	0b00*	GPIOC26 pin multiplexing
19 to 18	gpioc25c	R/W	0b00*	GPIOC25 pin multiplexing
17 to 16	gpioc24c	R/W	0b00*	GPIOC24 pin multiplexing
15 to 14	gpioc23c	R/W	0b00*	GPIOC23 pin multiplexing
13 to 12	gpioc22c	R/W	0b00*	GPIOC22 pin multiplexing
5 to 4	gpioc18c	R/W	0b00*	GPIOC18 pin multiplexing
3 to 2	gpioc17c	R/W	0b00*	GPIOC17 pin multiplexing
1 to 0	gpioc16c	R/W	0b01*	GPIOC16 pin multiplexing

[1] See Section 6.4 for more information

Table 87: Register syspad0

Bit	Symbol	Access	Value	Description
27 to 26	gpioa13p[2:1]	R/W		GPIOA13 pad configuration
			0b00	pull-up
			0b01	repeater
			0b10	plain-input
			0b11*	pull-down
23 to 22	gpioa11p[2:1]	R/W	0b01*	GPIOA11 pad configuration
11 to 10	gpioa5p[2:1]	R/W	0b00*	GPIOA5 pad configuration
5 to 4	gpioa2p[2:1]	R/W	0b01*	GPIOA2 pad configuration
3 to 2	gpioa1p[2:1]	R/W	0b01*	GPIOA1 pad configuration

[1] See Section 6.2 for more information

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 88: Register syspad1

Bit	Symbol	Access	Value	Description
31 to 30	gpioa31p[2:1]	R/W	0b00*	GPIOA31 pad configuration
29 to 28	gpioa30p[2:1]	R/W	0b00*	GPIOA30 pad configuration
27 to 26	gpioa29p[2:1]	R/W	0b01*	GPIOA29 pad configuration
25 to 24	gpioa28p[2:1]	R/W	0b00*	reserved, iic3m4sda pad has no configuration
23 to 22	gpioa27p[2:1]	R/W	0b00*	reserved, iic3m4scl pad has no configuration
21 to 20	gpioa26p[2:1]	R/W	0b11*	GPIOA26 pad configuration
19 to 18	gpioa25p[2:1]	R/W	0b11*	GPIOA25 pad configuration
17 to 16	gpioa24p[2:1]	R/W	0b11*	GPIOA24 pad configuration
7 to 6	gpioa19p[2:1]	R/W	0b01*	GPIOA19 pad configuration
5 to 4	gpioa18p[2:1]	R/W	0b10*	GPIOA18 pad configuration
1 to 0	gpioa16p[2:1]	R/W	0b01*	GPIOA16 pad configuration

[1] See [Section 6.2](#) for more information

Table 89: Register syspad2

Bit	Symbol	Access	Value	Description
23 to 22	gpiob11p[2:1]	R/W	0b11*	GPIOB11 pad configuration
21 to 20	gpiob10p[2:1]	R/W	0b11*	GPIOB10 pad configuration
19 to 18	gpiob9p[2:1]	R/W	0b11*	GPIOB9 pad configuration
17 to 16	gpiob8p[2:1]	R/W	0b11*	GPIOB8 pad configuration
15 to 14	gpiob7p[2:1]	R/W	0b11*	GPIOB7 pad configuration
13 to 12	gpiob6p[2:1]	R/W	0b11*	GPIOB6 pad configuration
11 to 10	gpiob5p[2:1]	R/W	0b11*	GPIOB5 pad configuration
9 to 8	gpiob4p[2:1]	R/W	0b11*	GPIOB4 pad configuration
7 to 6	gpiob3p[2:1]	R/W	0b11*	GPIOB3 pad configuration
5 to 4	gpiob2p[2:1]	R/W	0b11*	GPIOB2 pad configuration
3 to 2	gpiob1p[2:1]	R/W	0b11*	GPIOB1 pad configuration
1 to 0	gpiob0p[2:1]	R/W	0b11*	GPIOB0 pad configuration

[1] See [Section 6.2](#) for more information

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 90: Register syspad3

Bit	Symbol	Access	Value	Description
29 to 28	gpiob30p[2:1]	R/W	0b10*	GPIOB30 pad configuration
27 to 26	gpiob29p[2:1]	R/W	0b01*	GPIOB29 pad configuration
25 to 24	gpiob28p[2:1]	R/W	0b10*	GPIOB28 pad configuration
23 to 22	gpiob27p[2:1]	R/W	0b10*	GPIOB27 pad configuration
21 to 20	gpiob26p[2:1]	R/W	0b10*	GPIOB26 pad configuration
19 to 18	gpiob25p[2:1]	R/W	0b00*	GPIOB25 pad configuration
17 to 16	gpiob24p[2:1]	R/W	0b11*	GPIOB24 pad configuration
15 to 14	gpiob23p[2:1]	R/W	0b11*	GPIOB23 pad configuration
13 to 12	gpiob22p[2:1]	R/W	0b00*	GPIOB22 pad configuration
11 to 10	gpiob21p[2:1]	R/W	0b00*	GPIOB21 pad configuration
9 to 8	gpiob20p[2:1]	R/W	0b11*	GPIOB20 pad configuration
7 to 6	gpiob19p[2:1]	R/W	0b11*	GPIOB19 pad configuration
5 to 4	gpiob18p[2:1]	R/W	0b10*	GPIOB18 pad configuration

[1] See Section 6.2 for more information

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 91: Register syspad4

Bit	Symbol	Access	Value	Description
31 to 30	gpioc15p[2:1]	R/W	0b00*	GPIOC15 pad configuration
29 to 28	gpioc14p[2:1]	R/W	0b00*	GPIOC14 pad configuration
25 to 24	gpioc12p[2:1]	R/W	0b00*	GPIOC12 pad configuration
13 to 12	gpioc6p[2:1]	R/W	0b00*	GPIOC6 pad configuration
11 to 10	gpioc5p[2:1]	R/W	0b00*	GPIOC5 pad configuration
9 to 8	gpioc4p[2:1]	R/W	0b00*	GPIOC4 pad configuration
7 to 6	gpioc3p[2:1]	R/W	0b00*	GPIOC3 pad configuration
5 to 4	gpioc2p[2:1]	R/W	0b00*	GPIOC2 pad configuration
3 to 2	gpioc1p[2:1]	R/W	0b00*	GPIOC1 pad configuration
1 to 0	gpioc0p[2:1]	R/W	0b00*	GPIOC0 pad configuration

[1] See [Section 6.2](#) for more information

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 92: Register syspad5

Bit	Symbol	Access	Value	Description
29 to 28	gpioc30p[2:1]	R/W	0b10*	GPIOC30 pad configuration
27 to 26	gpioc29p[2:1]	R/W	0b10*	GPIOC29 pad configuration
25 to 24	gpioc28p[2:1]	R/W	0b10*	GPIOC28 pad configuration
23 to 22	gpioc27p[2:1]	R/W	0b10*	GPIOC27 pad configuration
21 to 20	gpioc26p[2:1]	R/W	0b10*	GPIOC26 pad configuration
19 to 18	gpioc25p[2:1]	R/W	0b10*	GPIOC25 pad configuration
17 to 16	gpioc24p[2:1]	R/W	0b10*	GPIOC24 pad configuration
15 to 14	gpioc23p[2:1]	R/W	0b10*	GPIOC23 pad configuration
13 to 12	gpioc22p[2:1]	R/W	0b10*	GPIOC22 pad configuration
5 to 4	gpioc18p[2:1]	R/W	0b10*	GPIOC18 pad configuration
3 to 2	gpioc17p[2:1]	R/W	0b10*	GPIOC17 pad configuration
1 to 0	gpioc16p[2:1]	R/W	0b00*	GPIOC16 pad configuration

[1] See [Section 6.2](#) for more information

Table 93: Register sysesh

Bit	Symbol	Access	Value	Description
31 to 3	-	R	1*	reserved
2	esh_hmp	R/W	0	speed control for HMP pads at VDDM [1]
			1*	high speed performance
			1*	low speed and low noise performance
1	esh_fmp	R/W	0	speed control for FMP pads at VDDM [1]
			1*	high speed performance
			1*	low speed and low noise performance
0	esh_muxed	R/W	0	speed control for muxed HMP/FMP pads at VDDM [1]
			0	high speed performance
			1*	low speed and low noise performance

[1] Speed control is only allowed for mobil memory supply level (see [Table 625](#)) otherwise it has to be set to '1'.

7.4.5 Functional description

The SCON block is connected to the peripheral bus system of the SC. The **syscon0** register is used by the system controller to define its operation. The version register **sysver** is read-only. It contains information about the chip version. The content is stored in a mask programmable array. The **sysmux** registers are used to define the pin multiplexing of the DVFD818x Family. The **sysesh** register is used to control the speed of the HMP and FMP memory pads. At mobil memory supply level (see [Table 625](#)) the pads can be configured for low speed with low noise or for high speed. At standard memory supply level the high speed selection is permitted.

7.4.5.1 System protection

The system protection register provides functionality in order to avoid hacker access to sensitive areas of the baseband. The idea is that the code in the SCROM sets one or more of the flags inside **sysprot** depending on the result of a certain number of tests. In case the SCROM sets also the **lock** bit, it is not possible anymore for the user to clear any of the bits which are set. The functions are locked by hardware, and only a HW reset can unblock the register. However a HW reset results again in the setting of the same bits inside **sysprot**.

If the **jtag_dis** bit is set then the following debug features are disabled:

- JTAG based access to the EmbeddedICE-RT and to ETB(ETM9 not available at DVFD8185 and DVFD8187).
- AHB access to ETB.
- NMI generation to DSP.

jtag_dis is set by SCROM depending on **security** bit and mode under host control. The **security** bit is set after an external hardware reset (RSTBB). It is not affected by a watchdog reset. Debugger access is only possible if the SCROM SW clears **jtag_dis** before setting the **lock** bit. The **security** bit can be read by DSP via the DCON unit.

The **scrom_prot[2:0]** field permits the user to avoid changes to the SCROM protection registers. As an example, it is possible to only permit code fetches from SCROM after the corresponding **scrom_prot** field has been set. This would prevent reading the code stored in the SCROM for hacking purposes.

Remark: all the SCROM code which needs to be protected has to be contiguous.

Remark: the constant data generated by the C-compiler has to be placed to areas outside of the boundaries of areas protected for code fetch-only.

7.5 TBU - Time Base Unit

7.5.1 Features

- 16-bit counter
- Operation with 32 kHz clock (clk32k) in all states
- Generation of interrupt at the beginning of the frame
- Register programmable by the SC

7.5.2 Block diagram

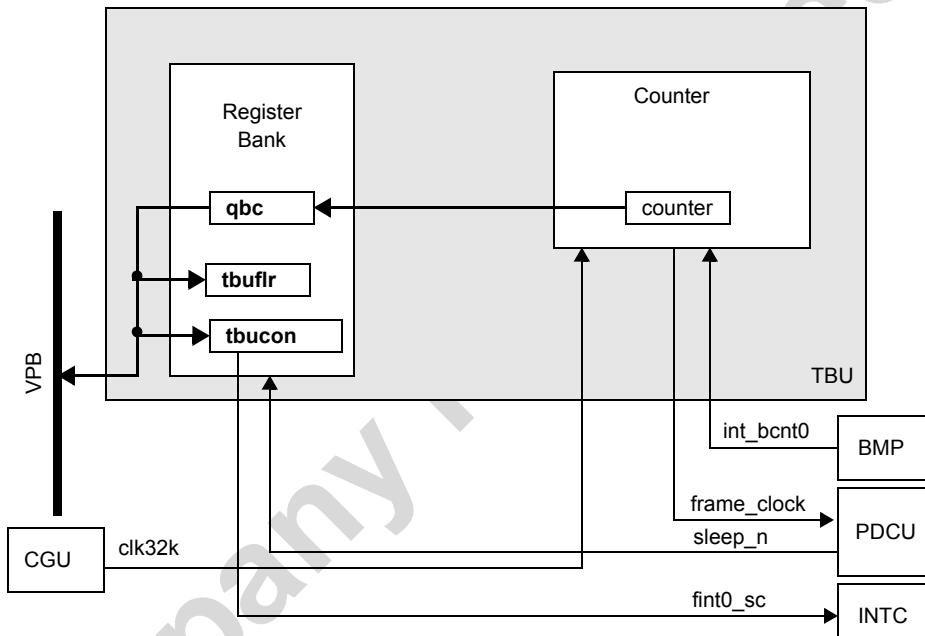


Fig 36. Block diagram of the TBU

7.5.3 Software interface

Table 94: Register overview of the SC

Name	Description	I/O	Reset
qbc	clk32k bit counter	R	0x0000
tbucon	time base unit configuration register	R/W	0x0000
tbuflr	time base unit frame length register	R/W	0x00A0 ^[1]

[1] These TBU registers are doubled buffered. As an effect of this, a new programming only takes effect at the begin of the next frame compared to programming (when qbc changes to zero).

Table 95: Register qbc

Bit	Symbol	Access	Value	Description
15 to 0	qbc[15:0]	R	0*	bit counter

Table 96: Register tbucon

Bit	Symbol	Access SC	Value	Description
15 to 4	-	R	0*	reserved
3	restbu	R/SaC	0*	reset for the bit counter of the TBU; this bit is automatically cleared when the reset has been performed (at most 1 bit later)
			0*	reset inactive
			1	reset active
2	fint0sc	R/C	0*	fint0_sc interrupt inactive
			1	fint0_sc interrupt active
1	fint0scen	R/W	0*	fint0_sc interrupt is disabled
			1	fint0_sc interrupt is enabled and goes high when qbc = 0
0	reserved	R	0*	

Table 97: Register tbuflr

Bit	Symbol	Access SC	Value	Description
15 to 0	flr[15:0] ^[1]	R/W	0xA0*	length of next frame with a duration of t _{clk32}

[1] Note: Minimal value for flr[15:0] must exceed the PLL locking time

7.5.4 Functional description

7.5.4.1 Bit counter

The time base unit contains the 16-bit counter **qbc**, which is clocked and incremented in all states with **clk32k**. The current bit value is available at the output of the block in the signal **qbc[7:0]**. The **qbc** counts with a resolution of **clk32k** modulo the frame length. The frame length can be programmed in the frame length register **tbuflr** but defaults to 160 (0xA0) bits.

7.5.4.2 Frame clock and frame interrupts

At the beginning of each frame the **frame_clock** pulse is generated, and if enabled, the **fint0_sc** interrupt is asserted.

The clearing of the SC interrupt happens by explicit clearing of the interrupt flag in **tbucon**.

The setting of the SC interrupt flag **fint0sc** happens independently from the setting of interrupt enable bit **fint0scen**. Therefore the user has to consider the context when switching the interrupt enable bit in order to avoid unwanted interrupts.

The interrupt is also disabled when the core is in Sleep mode (**sleep_n** active). Also the interrupt flag is not set. In the first frame when the core switches from sleep to active the frame interrupt **fint0_sc** is asserted if enabled.

7.5.4.3 TBU resets

At power-on, the TBU starts to operate after the **respon_n** signal has been inactivated. Then the IC is in reset state which for the TBU is equivalent to the sleep state: the operation of the TBU is defined by the reset values of the registers. For this purpose the PDCU forces the TBU in sleep state as long as the wake-up counter has not reached the value zero.

The TBU bit counter block including its control logic can be initialized by the SW by setting the **restbu** bit. For the TBU, the **restbu** bit and the **respon_n** signal are functionally equivalent, however the **respon_n** signal is synchronized to **clk32k**. The register interface is reset by the **hreset_n** signal.

When the **restbu** is cleared, the bit counter is set to zero, **fint0_sc**, is asserted if enabled, and the double buffered registers are loaded into the working registers.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

7.6 WDRU - Watchdog and Reset Unit

7.6.1 Features

- Generation of all internal resets during power-on
- Control of internal resets after power-on
- Watchdog timer allowing to reset DSP, SC and all peripherals after a SW deadlock

7.6.2 Block diagram

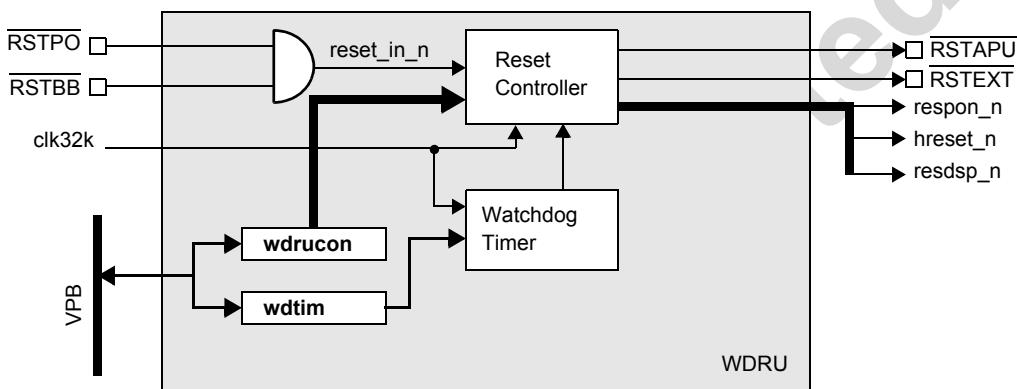


Fig 37. WDRU block diagram (Note: Pin RSTEXT is not available at DVFD818x)

7.6.3 Hardware interface

Table 98: WDRU pin overview

Pin	Name	I/O	Description
RSTPO	power on reset	I	power on reset input for the DPU, connected to the IF_NPORST output of the APU
RSTBB	baseband processor reset	I	main reset input for the DPU
RSTAPU	apu reset	O	reset output for the apu, connected to the IF_NARES input of the APU

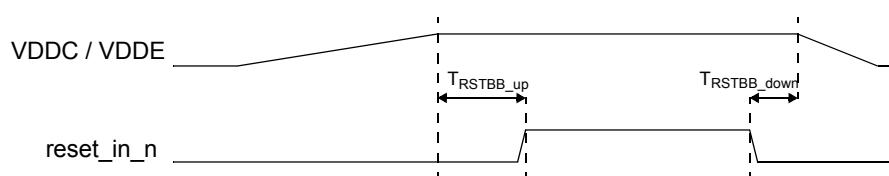


Fig 38. Timing diagram of the WDRU signals

Table 99: AC characteristics of the WDRU

Symbol	Description	Min	Typ	Max	Unit
T _{RSTBB_up}	RSTBB active after power is connected	1	-	-	μs
T _{RSTBB_down}	RSTBB inactive before power is cut off	0	-	-	μs

7.6.4 Software interface

Table 100:SC accessible registers

Name	Description	I/O	Reset
wdrucon	Reset control register	R/W	0x0000
wdtim	Watchdog period register	R/W	0x0000

Table 101: Register wdrucon

Bit	Symbol	Access	Value	Description
15 to 4		R	0*	reserved
3	resapu	R/W	0*	apu reset active; the pin changes one pclk2 cycle after the SC writes this field
			1	$\overline{\text{RSTAPU}}$ active
			0*	$\overline{\text{RSTAPU}}$ inactive
2	resdsp	R/W	0*	reset for the DSP (core and peripherals); in order to have full effect, this field need to be active at least 10 cycles of dcclk and at least 250 ns (see also Section 8.13.1)
			1	resdsp_n active
			0*	resdsp_n inactive
1	wdtstat	R	0*	status of the watchdog
			1	watchdog has not been loaded after reset, i.e. watchdog is disabled
			0*	watchdog has been loaded after reset, i.e. watchdog is enabled
0	reserved		0*	shall be set to 0

[1] **wdrucon** is reset by **hreset_n**.

Table 102: Register wdtim

Bit	Symbol	Access	Value	Description
15	wdtexp	R/C	0*	watchdog expiration flag
			1	watchdog was not responsible for the last performed reset sequence
			1	watchdog had expired before the last performed reset sequence
14	wdthalf	R	0*	half of programmed period of the watchdog
			1	watchdog time less than half of the programmed period
			0*	watchdog time greater than half of programmed period
13 to 0	wdtim[13:0]	R/W	0*	watchdog time out in $32/f_{\text{clk32k}}$ seconds steps - in this field the programmed value is read

[1] **wdtim** is reset by **respon_n**. The **wdtim** register is not reset by the expiration of the watchdog.

7.6.5 Functional description

7.6.5.1 DVFD818x Family resets

The WDRU and the PDCU are the chip masters during reset and Power-down modes. This means that these blocks manage the IC by control signals to the CGU to distribute the 32 kHz clock to the necessary blocks.

The WDRU reset inputs are the RSTBB and RSTPO pins. These 2 pins are joined together using an AND gate to form the signal **reset_in_n**. When **reset_in_n** is active (low) all internal resets are set active. Other reset sources are the watchdog timer and the register **wdruccon**, accessible by the SC. Depending on these reset sources, the WDRU generates all internal resets according to [Table 103](#).

When **reset_in_n** is active, all output pins of the DVFD818x Family acquire their reset value asynchronously as in [Table 5](#).

Table 103: Overview of DPU resets

Signal	Description	Activated by					Generated by
		Power-on reset	Watchdog expiration	SC SW control	Sleep state	JTAG control	
RSTBB	Reset input - activates all internal resets	-	-	-	-	-	PIN
RSTPO	Power on reset - activates all internal resets	✓	-	-	-	-	APU
reset_in_n	DPU reset - activates all internal resets	✓	-	-	-	-	<u>RSTBB</u> and <u>RSTPO</u>
RSTAPU	APU reset output	✓	✓	✓	-	-	WDRU
respon_n	internal power-on reset for:	✓	-	-	-	-	<u>RSTBB</u> and <u>RSTPO</u>
hreset_n	reset for all VPB and AHB participants:	✓	✓	-	-	-	WDRU
	• ARM core and caches						
	• DMAU, EBI1, SDI, BMP, DRT, ETN1						
	• all SC peripherals except the bit sysprot.security in SCON						
	• wdruccon register in WDRU						
resdsp_n	resets the DSP and all of its peripherals:	✓	✓	✓	-	-	WDRU
rescgu_n	resets the CGU and stops all high frequency clocks inside the chip	✓	-	-	✓	✓	JTAG and PDCU
resplldsp_n	force PLL_DSP into power down mode	✓	-	✓ ^[2]	✓	✓	JTAG and PDCU
respllper1_n	force PLL_PER1 into power down mode	✓	-	✓ ^[2]	✓	✓	JTAG and PDCU
respllper2_n	force PLL_PER2 into power down mode	✓	-	✓ ^[2]	✓	✓	JTAG and PDCU
respllsc_n	force PLL_SC into power down mode	✓	-	✓ ^[2]	✓	✓	JTAG and PDCU
resplifix_n	force PLL_FIX into power down mode	✓	-	✓ ^[2]	✓	✓	JTAG and PDCU and CGU

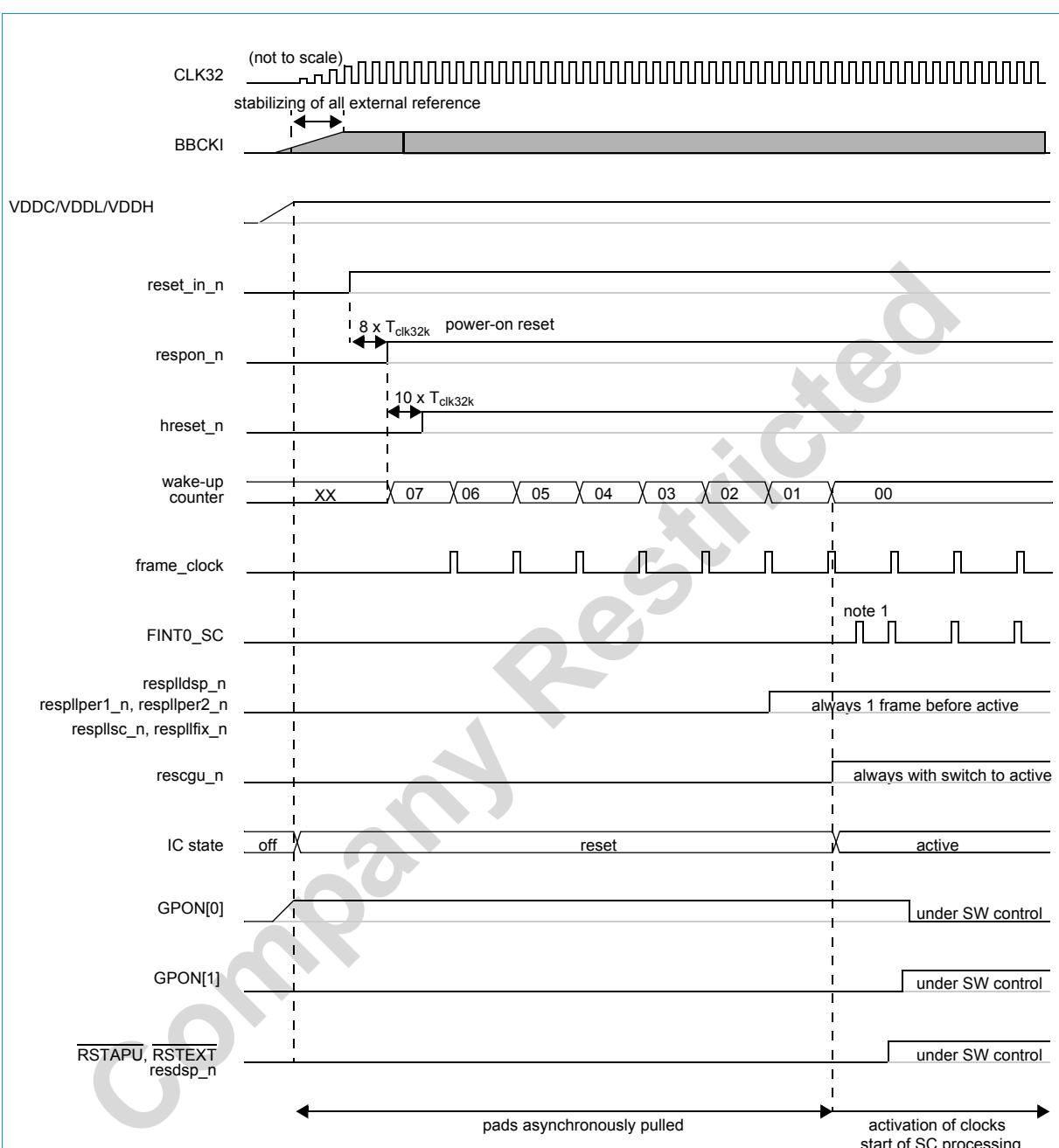
- [1] All reset signals are active low.
- [2] The PLLs can be powered down with the respective bits in the CGU configuration registers: **cguscccon.pllscen**, **cgudspcon.plldspen**, **cguper1con.pllper1en** and **cguper2con.pllper2en**, **cgufixcon.pllfixen**. See chapter 7.1.4.

The SC core does not have a dedicated reset instruction, the watchdog can be used to reset the SC. Memories have to be initialized by SW.

7.6.5.2 Power-on reset

The following sequence is executed after power is connected to the DPU:

1. The RSTBB and RSTPO signals are inputs to the WDRU; they are joined together with an AND gate to form the signal **reset_in_n**; this signal is synchronized to the **clk32k** domain by the WDRU; as long as **reset_in_n** is active, all internal resets are activated
2. Following the rising edge of **reset_in_n**, the **respon_n** remains active for 8 more cycles of **clk32k**; the deactivation of **respon_n**, activates the **clk32k** domains in the PDCU.
3. All resets but **respon_n** are active; the clock multiplexers of the CGU are set such that the whole system is clocked by **clk32k**; in this position of the clock multiplexers, the whole system is reset; the WDRU keeps the internal resets active for 10 **clk32k** pulses (about 0.3 ms), to ensure that all flip flops receive at least one clock edge
4. The WDRU sets **hreset_n** inactive - the system controller wakes-up after the PDCU wake-up counter (with default settings from **gpon** registers) has elapsed - the SC can check the reason of its wake-up in the set of registers which can generate one
5. The signals **resdsp_nand** RSTAPU remain active until they are set by the SW.



(1) The FINT0 is pending as soon as the IC switches to active state. The interrupt must be enabled by the SW.

Fig 39. Power-on reset sequence (Note: Pins GPON[2:0] and RSTEXT are not available DVFD818x)

7.6.5.3 JTAG control of the reset

It is possible to trigger or block the main (RSTBB) reset of the system via the JTAG interface. This is controlled via the JTAG register RESET_CON (see [Section 7.2.4](#))

7.6.5.4 The watchdog timer

The DPU contains a watchdog timer. The watchdog timer is disabled after power-on reset and after watchdog reset. The watchdog timer starts after writing a value into **wdtim**. The maximum time range is 16 seconds. Once the watchdog timer has been started, it cannot be disabled by SW. If the watchdog has been activated, the SC must reload the timer within the period programmed in the **wdtim** register, or a watchdog reset sequence is performed.

The bit **wdthalf** indicates when the watchdog counter is greater than half the programmed **wdtim** value. If the watchdog timer expires during active or idle, the **wdtexp** flag is set and the watchdog reset sequence is performed. The **wdtexp** is reset by SW or by the activation of **respon_n**. As a consequence the watchdog expiration resets **wdrucon** but not **wdtim**.

The watchdog timer is frozen while the SC is in Stop, Sleep or Debug mode. In this way, a reset triggered by the watchdog expiration during power-down and debugging via EICE is avoided.

Writing a zero into **wdtim** while the watchdog is disabled, triggers a watchdog reset after one 32 kHz cycle.

7.6.5.5 Watchdog timer reset

If the watchdog timer is enabled and expires (in case of a software deadlock), the DPU is reset (except for **respon_n**). The following sequence is performed if the SC watchdog timer expires.

1. BBCI is not interrupted as the GPON signals do not change - the **clk32k** is not interrupted - the **respon_n** signal is kept inactive
2. The WDRU activates **hreset_n**, **resdsp_n** and **RSTAPU** active (**respon_n** inactive).
3. The clocks for the SC and its peripherals are enabled for 10 **clk32k** periods.
4. The WDRU sets **hreset_n** inactive: the system controller starts executing code from the SCROM - the SC can check the reason of its wake-up in the set of registers which can generate one - since the SC finds the **wdtim.wdtexp** flag set, it is possible to act accordingly.
5. The SW should clear the **wdtexp** flag in order to be able to detect future watchdog expirations.

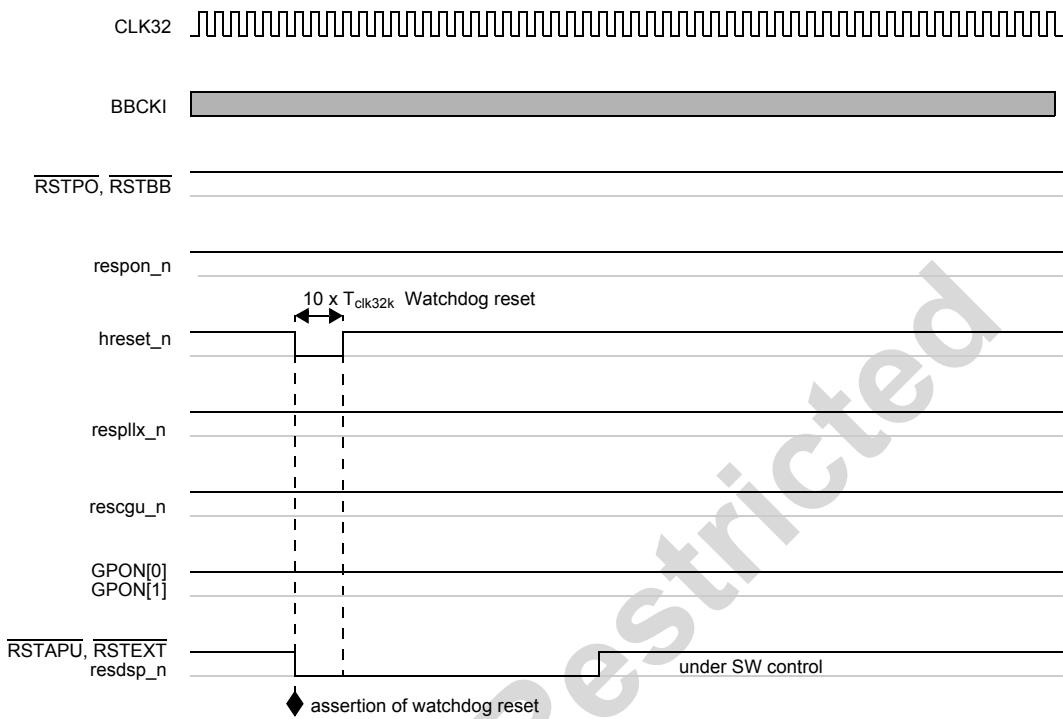


Fig 40. Watchdog reset sequence (Note: Pins GPON[2:0] and RSTEXT are not available DVFD818x)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

7.7 Debug, Trace and Prototyping

7.7.1 General

The DVFD818x Family supports debugging, tracing and prototyping, in order to support the software development and the target system development.

Two JTAG test access ports called TAP1 and TAP2 are available. They are independent and can be connected to different tools: A DSP emulator [4.] running on a PC can be connected to TAP2, and the JTAG-debugger or any other of ARM supporting hardware can be connected to TAP1. It is also possible to debug ARM and DSP through TAP1 only, by switching the JSEL[1:0] pins (see [Section 7.2](#) for more information).

The HDP is a multi-functional port which allows tracing of several internal signals. These pins are multiplexed between the ETM9 and other internal signals.

Table 104: Implementation of features

Features	SC	DSP	Miscellaneous
Debug	Embedded ICE-RT and SJTAG	TIO and TAP2	-
Trace	ETM9/ETB	TMU	HDP
Prototyping	EBI2	-	HDP

[1] Note that on DVFD8185 and DVFD8187 not all debug features are available due to there are not all HDP pins available (e.g. no ETM9).

In addition the DSP subsystem provides a large amount of program and data RAM. While the stable FW modules are stored in PROM/XYROM, new FW modules can be loaded into program RAM and executed/debugged from there.

Thanks to the register **sysprot** described in [Section 7.4](#) it is possible to permanently disable all debugging functionality.

7.7.2 Debugging features

7.7.2.1 Features

- Two independent JTAG test access ports and TAP controllers dedicated for debugging - , TAP1/TAPC2/TAPC3 (SC/ETB), TAP2/TAPC4 (DSP)
- Real-time SC debugging - EmbeddedICE-RT and ETB via TAPC2 and TAPC3
- Real-time DSP debugging - monitor based DSP emulator via TAPC4
- JTAG-debugger scan based debugging of SC
- Breaking of the SC - EmbeddedICE-RT
- Single step execution for the SC - EmbeddedICE-RT
- DSP status can be read out via TMU
- DSP general purpose output (DGPO[2]) can be used to indicate internal DSP status or for trigger functions.
- Large internal SRAM for DSP running at full speed supporting development of new firmware - PRAM and XYRAM
- External flash is emulated with external SRAM on target hardware - EBI1.

7.7.2.2 Functional description

The two cores can be run simultaneously in the target system in order to verify the function of the SW. During this operation, the SC subsystem can be traced via the on-chip trace buffer ETB. The emulation function is provided by the flexible breaking functionality of the EmbeddedICE-RT block.

The debugging is mainly performed by means of the JTAG test access ports (TAP). Through TAP1, the complete feature set of the EmbeddedICE-RT can be used. Through TAP2 port, it is possible to exchange data between an external PC and DSP by means of a monitor program located in PROM.

TMU can be used to monitor DSP FW activity with the SC.

The ARM926EJ-S can in principle not be debugged if it is not clocked (for instance when it is in Idle, Stop or Sleep mode). However in Sleep and Stop mode, it is possible to clock the SC with **clk32k** if the **cgsleepsc.ahbslen** is set, which as a consequence allows the user to debug the ARM926EJ-S. Note that due to RTCK synchronization, this result on a TCK clock around 15 kHz.

The SCTU is frozen while the SC is in Debug mode. This allows to maintain system timer (like OS ticks) not affected by breakpoints.

7.7.3 Tracing features**7.7.3.1 Features**

- Hardware debug port featuring several sets of signals for tracing - HDP
- Monitoring of DSP status - TMU
- ARM926EJ-S tracing - ETB.

7.7.3.2 Functional description

The DVFD818x Family features tracing for SC program execution. Monitoring of DSP status is done via status read operation through TMU. Tracing of the SC is done via the embedded trace buffer (ETB).

7.7.4 Prototyping of SC peripherals**7.7.4.1 Features**

- EBI1 or EBI2 interface for prototyping SC peripherals
- External interrupts and external DMA requestors available

7.7.4.2 Description

The EBI1 and EBI2 support prototyping, which provides the option for early IP development and verification in a known baseband environment.

Note that on DVFD818x Family the EBI1 and EBI2 pins are reduced and therefore less functions are supported.

In order to prototype external blocks, external interrupts are provided and an external DMAU channel is available.

8. DSP Subsystem (DSP)

The DVFD818x includes a Saturn DSP subsystems (DSP) with embedded RAM and ROM.

8.1 DSP Block Description

The DSP subsystem executes all audio algorithms. It includes the Saturn DSP core with embedded RAM and ROM, and a set of peripherals. [Figure 41](#) shows the block diagram of the DSP subsystem.

The building blocks of the DSP subsystem are:

- Saturn DSP core RD16024
- DMA controller (DMAC) capable of performing data transfers between memory and I/O
- Static random access memory (PRAM and XYRAM): on-chip program and data RAM consisting of several banks of static RAM; program and data RAM are not shared
- Static random access memory for application specific instructions (ASIRAM); on-chip RAM to store 256 application specific instructions (ASI); each ASI is 96-bit wide
- Read-only memory (PROM and XYROM): on-chip program and data ROM includes the entire signal processing firmware.
- Bug Bypass Block (BBB): the BBB allows to patch bugs in DSP ROM code
- Memory Paging Unit (MPU): memory paging allows to extend program and data memory address spaces to more than the default 64 kwords
- Memory Monitor Register (MMR): the MMR monitors the DSP memory instance access and checks for address conflicts on shared X and Y memory
- Test I/O port (TIO): the TIO port allows data exchange between the DSP core and an external device via the JTAG interface
- DSP control unit (DCON): the DCON is used to control security settings of the DSP firmware.
- Tandem Mailbox unit (TMU): the TMU is used for the access of the system controller to the DSP memory areas and for interchanging interrupts between the DSP and the SC.

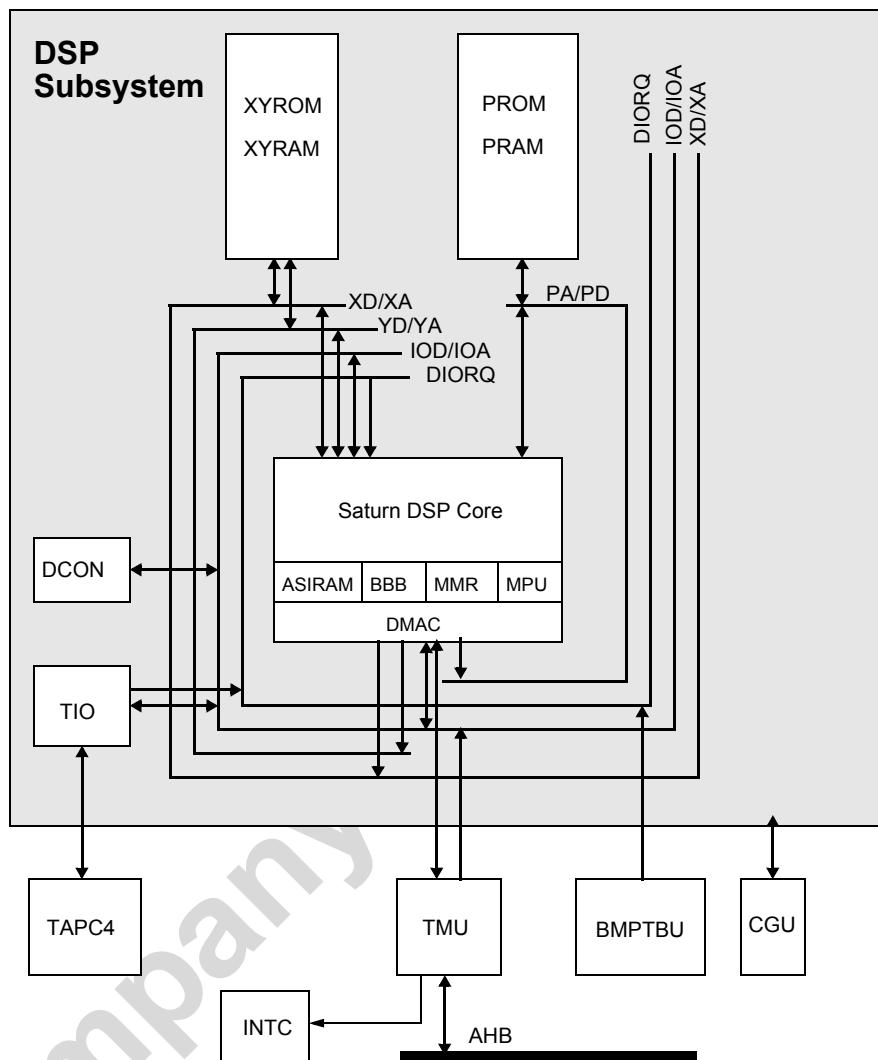


Fig 41. DSP block diagram

The **DSP core** communicates with its memory via three data buses and three address buses:

- The program address bus PA carries the address of the instruction currently fetched from on-chip or external memory. The instruction is transferred on the program data bus PD. The program data bus PD is bidirectional in order to allow write access to PRAM.
- For two parallel data transfers (both in parallel to the program fetch) two address/data bus pairs are available (XA for address of data space X, YA for address of data space Y, XD and YD for the data transfers respectively).

The **DSP peripheral units** are connected to the core via a dedicated peripheral bus (IO-bus) consisting of the data bus XD and the I/O address bus IOA that selects the registers in the peripherals. The DIORQ-bus carries the peripherals I/O requests (interrupts and DMAs).

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The following top-level and intercore blocks are related to the DSP subsystem:

- The BMPTBU generates speech and frame interrupt signals for the DSP.
- Via the TMU the system controller can access the DSP memory to exchange program code and application data.
- The CGU is delivering the clocks for the DSP block.
- The TAPC4 is the JTAG test and emulation control interface to the DSP block.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.2 DSP Core

8.2.1 DSP core description

Detailed information on the Saturn DSP core can be found in [6.].

A simplified block diagram of the Saturn DSP core is shown in Figure 42. A detailed description of the DSP core is given in [6.]. Additional programmer's restrictions are given also in Section 8.13.

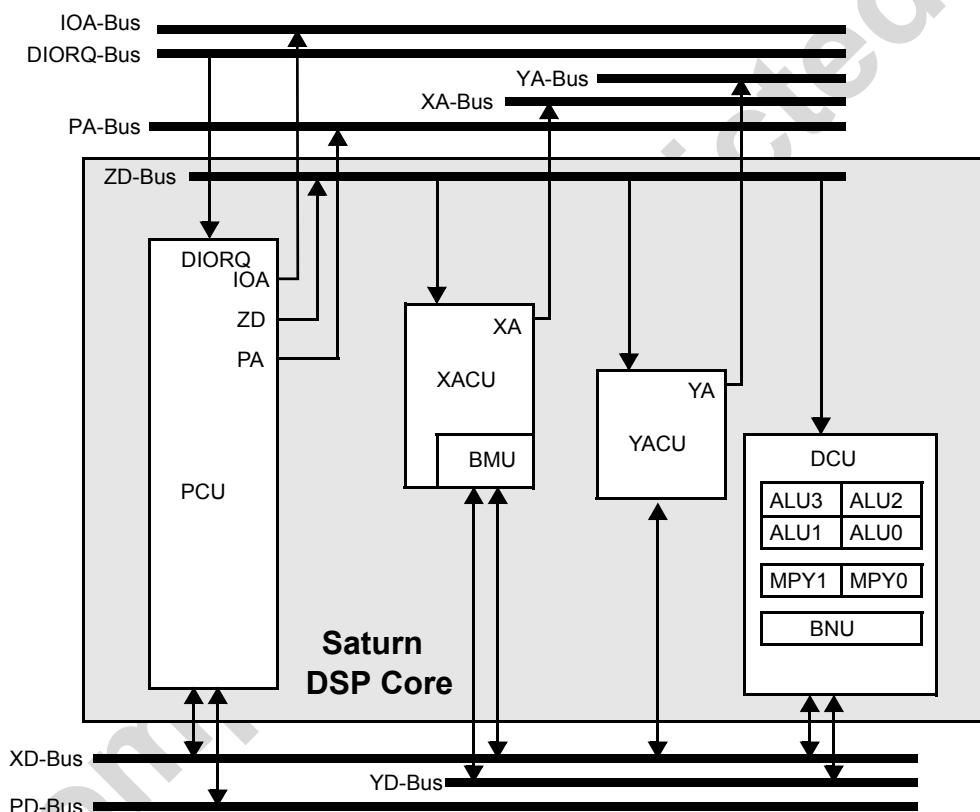


Fig 42. DSP core block diagram

8.2.2 DSP core performance

The DSP can run at lower voltage when the clock speed is reduced. This results in a significantly reduced power consumption. Please refer to Section 13.

8.2.3 DSP core ID number

The Saturn DSP core version is **0x0611**. The core ID can be read from the core register **cidr** with register address 0x1A.

Example:

```
x0 = cidr;
*pym0++ = cidr;
```

8.2.4 Power Down

The DSP may set itself into idle mode by using the IDLE instruction. To optimize power consumption, the DSP core clock, dcclk is automatically switched off by the CGU. If the **cgudspcon.diclkidleen** bit is not set then the diclk will also be automatically switched off.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.3 Memory Configuration

8.3.1 Overview

The memory map of DSP is divided into program (P) and shared data (X/Y) memory. One DSP word equals 16 bits. A RAM bank may only be accessed by one bus at a time. Bus conflicts (X and Y) are signalled in register **mmrdata** but not prevented.

The following memory is integrated.

Table 105: DSP memory configuration

Memory type	Size [1]
PROM	72 kwords
PRAM	24 kwords
XYROM	66 kwords
XYRAM	30 kwords
ASIRAM	256 x 96

[1] A word corresponds to a 16-bit location

The PRAM is organized as on-chip memory banks, which can be accessed by the PD bus only. XYRAM memory banks can be accessed either by the XD or by the YD bus.

The Saturn features 256 ASI which can be defined by the programmer. Each ASI is 96-bit wide and all of them are defined in a 256×96 -bit look-up table called ASIRAM.

CAUTION



After switching a data memory page one additional clock cycle is required before the program can access it.

8.3.2 Memory map

The DSP memory map is described in [Table 106](#) and shown in [Figure 43](#).

Table 106: DSP Memory Map

Address Range	Page	Memory Bank	Size [1]
Program Memory Space			
0xFFFF to 0x8000	1	ROM bank PROM2	32 kwords
0xFFFF to 0x8000	0	ROM bank PROM1	32 kwords
0x7FFF to 0x6000	-	RAM bank PRAM2	8 kwords
0x5FFF to 0x4000	-	RAM bank PRAM1	8 kwords
0x3FFF to 0x2000	-	RAM bank PRAM0	8 kwords
0x1FFF to 0x0000	-	ROM bank PROM0	8 kwords
X and Y Data Memory Space			
0xFFFF...0x8000	1	XYROM2	32 kwords
0xFFFF...0x8000	0	XYROM1	32 kwords
0x7FFF...0x7000	-	XYRAM11	4 kwords
0x6FFF...0x6000	-	XYRAM10	4 kwords
0x5FFF...0x5000	-	XYRAM9	4 kwords
0x4FFF...0x4800	-	XYRAM8	2 kwords
0x47FF...0x4000	-	XYROM0	2 kwords
0x3FFF...0x3800	-	XYRAM7	2 kwords
0x37FF...0x3000	-	XYRAM6	2 kwords
0x2FFF...0x2800	-	XYRAM5	2 kwords
0x27FF...0x2000	-	XYRAM4	2 kwords
0x1FFF...0x1800	-	XYRAM3	2 kwords
0x17FF...0x1000	-	XYRAM2	2 kwords
0x0FFF...0x0800	-	XYRAM1	2 kwords
0x07FF...0x0000	-	XYRAM0	2 kwords

[1] A word corresponds to a 16-bit location

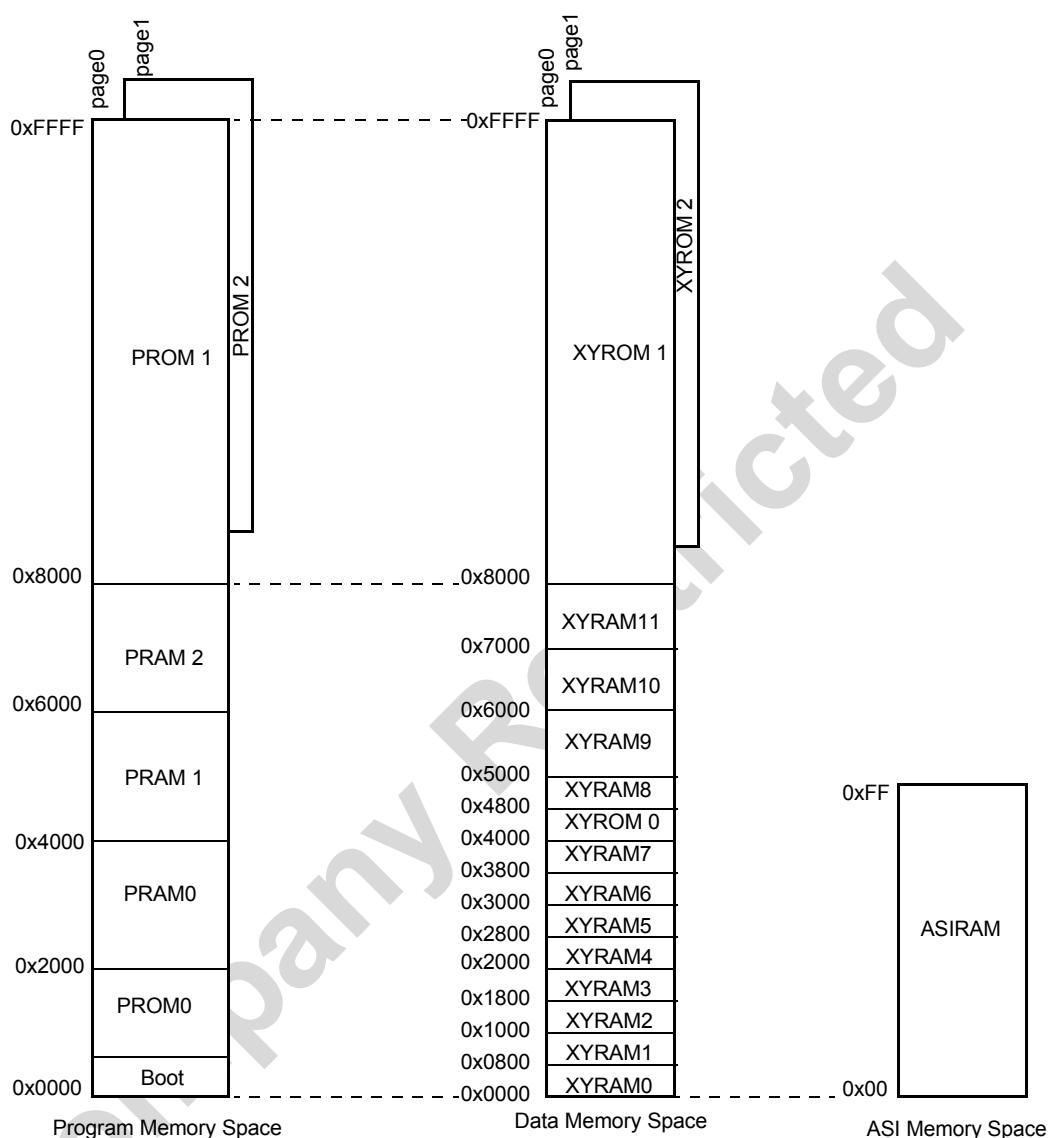


Fig 43. DSP Memory Map

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.4 I/O Register Map

Table 107 shows the I/O register map of DSP. The total number of available address locations is 64.

Note that not all the bits of all registers are used. The unused bits are marked as reserved in the symbol row of the register layout tables. Read operations on any reserved bit always result in 0. Reserved bits must be written with 0.

Table 107:I/O register map of DSP

IOA	Name	I/O [1]	Description
Reserved			
0x0-0x2B	-	-	reserved
DCON			
0x2C	dcon0	R/W	DSP control register
Reserved			
0x2D	-	-	reserved
MPU			
0x2E	pmpc	R/W	MPU program memory paging control register
0x2F	xmpc	R/W	MPU X data memory paging control register
0x30	ympc	R/W	MPU Y data memory paging control register
0x31	cmpc	R/W	MPU PXY common memory paging control register
Reserved			
0x32	-	-	reserved
TMU			
0x33	tmuctrl	R/W	TMU control register
TIO			
0x34	ti	R	TIO data input register
	to	W	TIO data output register
0x35	toc	R/W	TIO control register
MMR			
0x36	mmrpntr	R/W	MMR memory monitor pointer register
0x37	mmrdata	R/C	MMR memory monitor data register
DMAC			
0x38	dmair	R/W	DMAC instruction register
0x39	dmadr	R/W	DMAC data register (virtual register)
BBB			
0x3A	bbbdr	R/W	BBB data register (virtual register)
0x3B	bbbir	R/W	BBB instruction register
Reserved			
0x3C	-	-	reserved
0x3D	-	-	reserved
0x3E	-	-	reserved
0x3F	-	-	reserved

[1] The size of all registers is 16 bits.

Most of the I/O registers are decoupled from the data buses by a bridge. While this bridge is transparent to the user, it has the following timing impact.

- I/O registers belonging to BBB, DMAC, TIO, MPU and MMR are not impacted. All other I/O registers are impacted and are called “impacted I/O registers” in the following.
- All other I/O registers are affected as follows:
 - one wait state is inserted for each read access
 - one wait state is inserted for each write access which is immediately followed by a read access to an “impacted I/O register”
 - no wait state is inserted for all other write accesses

The flags in the **isr** register related to “impacted I/O registers” are cleared with one cycle delay with respect to the clearing in the peripheral.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.5 DSP Interrupt Configuration

8.5.1 Block Diagram

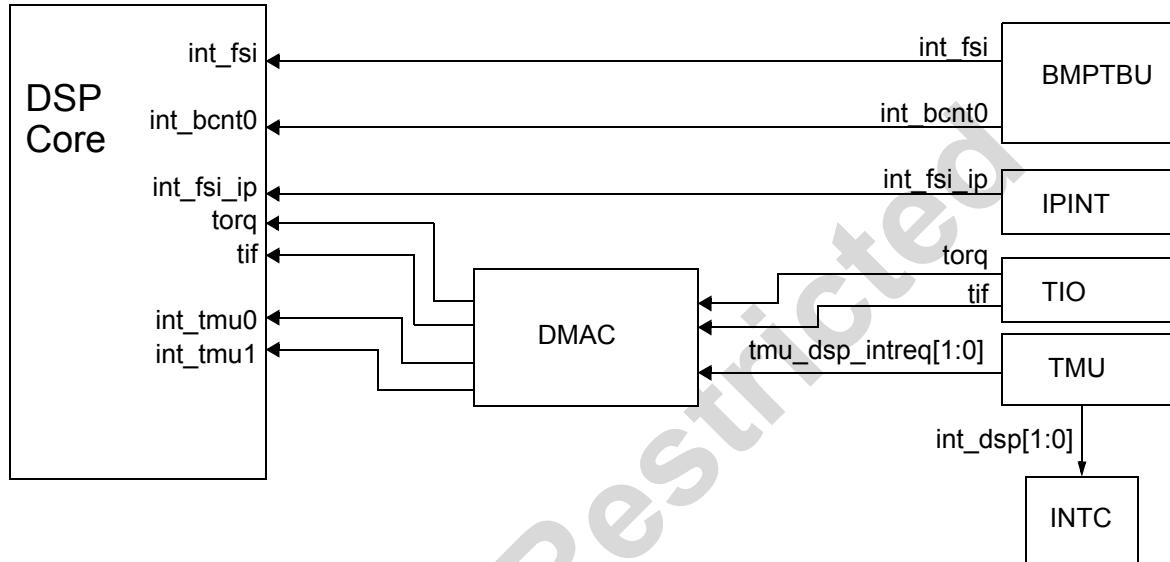


Fig 44. DSP interrupt block diagram

8.5.2 Software interface

Table 108: Interrupt related registers in DSP

Name	Description	I/O	Reset
ibr	interrupt base register	R/W	0x0000
ier	interrupt enable register	R/W	0x0000
isr	interrupt status register	R	0x0000

Table 109: Register ibr

Bit	Symbol	Access	Value	Description
15 to 0	ibr[15:0]	R/W	0x00*	interrupt base address

Table 110: Register ier

Bit	Symbol	Access	Value	Description
15	torq	R/W	0*	to output request interrupt enabled if set
14	tif	R/W	0*	ti full interrupt enabled if set
13	-	R	0*	reserved
12	int_bcnt0	R/W	0*	bmp frame interrupt enabled if set
11	-	R	0*	reserved
10	tmu_dsp_intreq1	R/W	0*	tmu output request interrupt 1 enabled if set
9	-	R	0*	reserved
8	tmu_dsp_intreq0	R/W	0*	tmu output request interrupt 0 enabled if set

Table 110: Register ier...continued

Bit	Symbol	Access	Value	Description
7	-	R	0*	reserved
6	-	R	0*	reserved
5	-	R	0*	reserved
4	int_fsi_ip	R/W	0*	ipint speech frame interrupt enabled if set
3	-	R	0*	reserved
2	int_fsi	R/W	0*	bmp speech frame interrupt enabled if set
1	-	R	0*	reserved
0	-	R	0*	reserved

Table 111: Register isr

Bit	Symbol	Access	Value	Description
15	torq	R	0*	to output request interrupt pending if set
14	tif	R	0*	ti full interrupt pending if set
13	-	R	0*	reserved
12	int_bcnt0	R	0*	bmp frame interrupt pending if set
11	-	R	0*	reserved
10	tmu_dsp_intreq1	R	0*	tmu interrupt 1 pending if set
9	-	R	0*	reserved
8	tmu_dsp_intreq0	R	0*	tmu interrupt 0 pending if set
7	-	R	0*	reserved
6	-	R	0*	reserved
5	-	R	0*	reserved
4	int_fsi_ip	R	0*	ipint speech frame interrupt pending if set
3	-	R	0*	reserved
2	int_fsi	R	0*	bmp speech frame interrupt pending if set
1	-	R	0*	reserved
0	-	R	0*	reserved

8.5.3 Functional description

The start address after reset (reset vector) is located at 0x0 on bank PROM0.

The interrupt base register **ibr** defines the memory address at which interrupt service routines are located.

Table 112: DSP interrupt vector addresses

Priority	Name	Address	Description	DMA	Related block	Clearing
highest	reset	0x0	DSP core reset start address	-	DSP core [1]	-
	NMI	ibr + 0x00	non-maskable interrupt	-	TAPC4 [1]	-
	-	ibr + 0x04	reserved	-	-	-
	-	ibr + 0x08	reserved	-	-	-
	int_fsi	ibr + 0x0c	bmp speech frame interrupt	-	BMP	jump [4]
	-	ibr + 0x10	reserved	-	-	-
	int_fsi_ip	ibr + 0x14	ipint speech frame interrupt	-	-	jump [4]
	-	ibr + 0x18	reserved	-	-	-
	-	ibr + 0x1c	reserved	-	-	-
	-	ibr + 0x20	reserved	-	-	-
	tmu_dsp_intreq0	ibr + 0x24	tmu interrupt 0 pending if set	[3]	TMU	write tmuctrl
	-	ibr + 0x28	reserved	-	-	-
	tmu_dsp_intreq1	ibr + 0x2c	tmu interrupt 1 pending if set	[3]	TMU	write tmuctrl
	-	ibr + 0x30	reserved	-	-	-
	int_bcnt0	ibr + 0x34	bmp frame interrupt	-	BMPTBU	jump [4]
	-	ibr + 0x38	reserved	-	-	-
	tif	ibr + 0x3c	ti register full interrupt	yes [2]	TIO	read ti
	torq	ibr + 0x40	to register empty I/O request	yes [2]	TIO	write to
	trap	ibr + 0x44	subroutine entry for real time debugging	-	DSP core [1]	-
lowest	bbbtrap	ibr + 0x48	subroutine entry for BBB	-	BBB [1]	-

[1] Reset and trap are SW interrupts. Reset, trap and bbbtrap cannot be masked. NMI is disabled by HW if the **security** bit in DCON is set.

[2] When the DMA channel is enabled, the I/O request signal from the peripheral triggers the DMA channel to transfer data. After completion of a programmable number of DMA transfers, the DMA channel asserts an interrupt signal to the DSP. This signal is connected to the interrupt described in this table. When the DMA channel is disabled, then the I/O request signal from the peripheral is connected to the interrupt line described in this table. For more details see [Section 8.11](#).

[3] An external DMA channel is used by TMU. In contrast to [2] the external DMA has no effect on the TMU interrupt requests. These are under strict program control by the DSP or the SC (see [Section 8.12](#)).

[4] These interrupts are automatically cleared as soon as the DSP enters the interrupt routine.

8.6 BBB - Bug Bypass Block

The BBB allows to patch parts of the Saturn DSP firmware ROM. Up to 7 program addresses can be defined to trigger the bbb trap, which causes the processor to execute the bbb trap subroutine.

8.6.1 Features

- Seven trap address registers are available
- All trap addresses can be enabled independently
- A dedicated BBB trap routine is triggered when PA bus matches one of the enabled trap addresses

CAUTION



The BBB is not sensitive to memory paging. It has to be reloaded by SW whenever the page is changed.

8.6.2 Block diagram

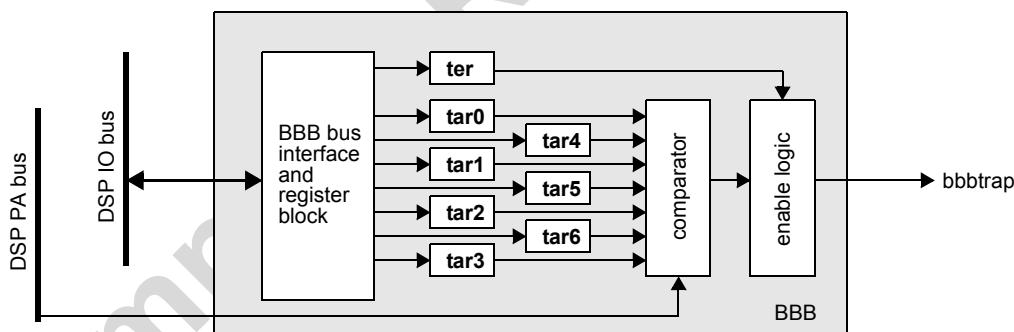


Fig 45. BBB block diagram

8.6.3 Software interface

Table 113: Register overview of BBB

Name	Description	I/O	Reset
bbbir	BBB instruction register (pointer)	R/W	0x0000
bbbdr	BBB data register (virtual register)	R/W	[1]

[1] Reset value differs for the various data registers. See Table 114, Table 115 and Table 116 for details.

The data register **bbbdr** is a virtual register allowing access to the eight registers **tar0** to **tar6** and **ter**. The selection is done with the pointer register **bbbir**.

Table 114: Register bbbir

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2 to 0	reg[2:0]	R/W		BBB data register to be accessed - this field features an auto-increment: on each access of the bbbdr (read and write), the value of the field reg[2:0] is incremented by one
	0b000*			tar0 - trap address register 0
	0b001			tar1 - trap address register 1
	0b010			tar2 - trap address register 2
	0b011			tar3 - trap address register 3
	0b100			tar4 - trap address register 4
	0b101			tar5 - trap address register 5
	0b110			tar6 - trap address register 6
	0b111			ter - trap enable register

Table 115: Register bbbdr (tar0 - trap address register 0)

Bit	Symbol	Access	Value	Description
15 to 0	tadr0[15:0]	R/W	~*	trap address 0

The trap address registers **tar1/tar2/tar3/tar4/tar5/tar6** are identical **tar0**.

Table 116: Register bbbdr (ter - trap enable register)

Bit	Symbol	Access	Value	Description
15 to 7		R	0*	reserved
6	ten6	R/W	0*	trap address 6 enabled if set
5	ten5	R/W	0*	trap address 5 enabled if set
4	ten4	R/W	0*	trap address 4 enabled if set
3	ten3	R/W	0*	trap address 3 enabled if set
2	ten2	R/W	0*	trap address 2 enabled if set
1	ten1	R/W	0*	trap address 1 enabled if set
0	ten0	R/W	0*	trap address 0 enabled if set

8.6.4 Functional description

The Bug Bypass Block (BBB) allows to patch parts of DSP firmware ROM. This functionality is typically used to bypass buggy program code. The bypass is achieved by inserting a BBB trap into the instruction instead of the instruction from the PD bus. The trap-instruction is a single-word instruction which triggers a subroutine call at a pre-defined address. The BBB trap differs from the normal trap instruction only in the vector which determines the subroutine address. The BBB trap executes the subroutine defined at the address $IBR + 0x48$, whereas the normal trap instruction uses address $IBR + 0x44$. To determine the original BBB trap address, bbb handling subroutine can read the **ra** register, which contains the original program address plus one. In the subroutine the **ra** register might be changed to transfer the program flow to a corrected program part (see [6.]).

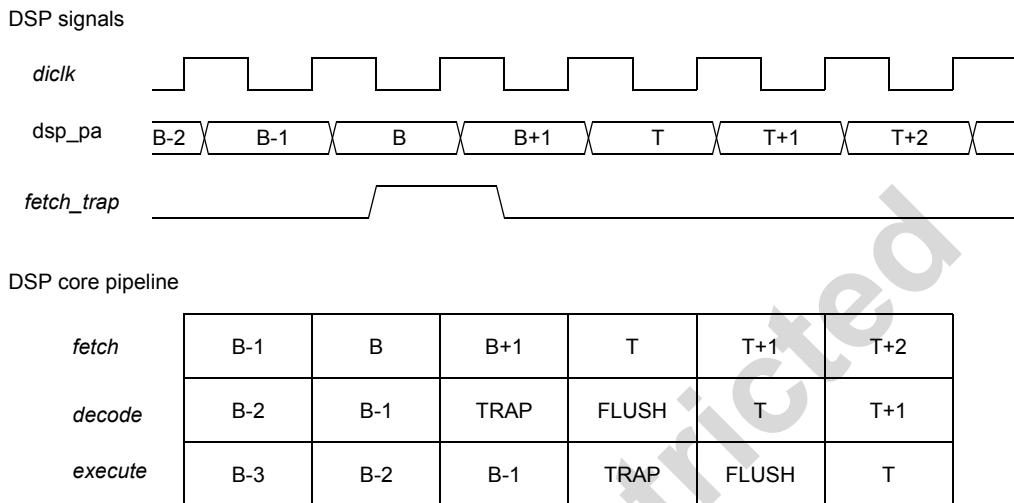


Fig 46. introduction of a trap signal 'B' is trapped by the BBB. The trap routine is located at address "T"

The BBB contains seven trap address registers (**tar0** to **tar6**). Each **tar** has a corresponding enable bit in the trap enable register (**ter**). Bit x of the **tar** corresponds to **tarx**. The BBB trap routine is be called if the program execution reaches an address programmed in an enabled **tar**.

The internal registers in the BBB can be modified by accessing the **bbbdr**. The **bbbir** determines which internal register can be accessed by the **bbbdr**. Each time the **bbbdr** is accessed via the IO-bus, the **bbbir** is auto-incremented. The **bbbir** and the **bbbdr** can also be read out via the IO-bus. The **bbbir**-addresses, by which the internal registers are accessible are listed in [Table 114](#).

The BBB is completely controlled by the enable register **ter** (contains the enable bits) and the trap address registers (**tar0** to **tar6**). All BBB registers are programmed via the IO-interface.

8.6.5 Limitations

A **tar** has no effect if it points to a second word of a double word instruction. A **tar** which points to the first word of a double word instruction, does work as previously described.

The **tar** does not influence read operations from the program memory initiated by the DMAC or by executing the instructions '`*py1++=y1, y1=*pp++`' or '`*pp++=y1, y1=*py1++`'.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.7 DCON - DSP Control Unit

8.7.1 Feature List

- Security bit used to enable security relevant DSP firmware features
- One general purpose output is available at DVFD818x.

8.7.2 Block Diagram

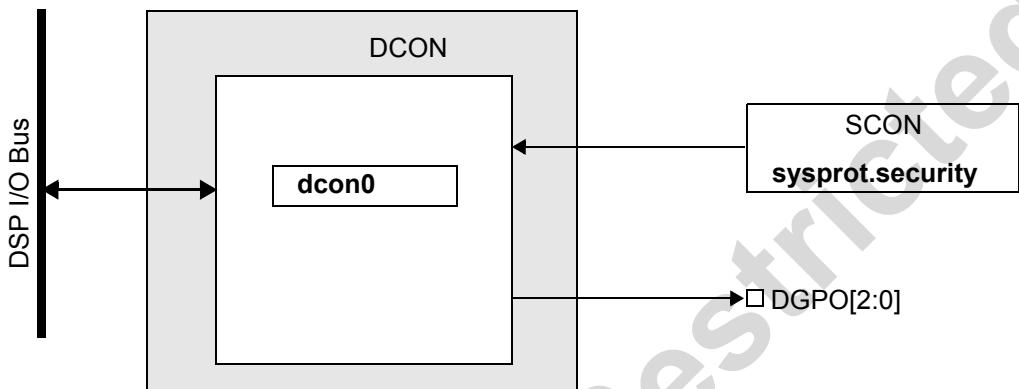


Fig 47. DCON block diagram

Note: at DVFD818x only DGPO[2] is available

8.7.3 Software Interface

Table 117: Register overview of the DCON

Name	Description	I/O	Reset
dcon0	DSP control register	R/W	0x0000

Table 118: Register dcon0

Bit	Symbol	Access	Value	Description
15	security	R		secure mode enable copy of sysprot.security
			0*	secure mode disabled
			1	secure mode enabled
14 to 3	-	R	0*	reserved
2 to 0	dgpo[2:0]	R/W	0*	three general purpose output bits a'0' sets the corresponding output pad (DGPO[2:0]) to low level, a '1' sets it to high level.

8.7.4 Functional Description

The DCON provides a **security** bit which is polled by the on-chip firmware. This bit is a copy of the **security** bit in the **sysprot** register of the SCON block. When **security** is set then the firmware no longer allows to perform SW download and monitor-based debugging via TAPC4 and TIO.

Additional the DCON can control 1 general purpose output bit DGPO[2]. It can be used to provide blinking effects synchronized with music.

8.7.5 Application informations

To read data from the DCON two consecutive read operations have to be performed. The second read operation gives the actual data in **dcon0**.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.8 MMR - Memory Monitor Register

8.8.1 Features

The MMR monitors the DSP memory instance access and checks for address conflicts on shared X and Y memory banks.

8.8.2 Software interface

Table 119: Register overview of MMR

Name	Description	I/O	Reset
mmrpntr	memory monitor pointer register	R/W	0x0000
mmrdata	memory monitor data register	R/W	0x0000

Table 120: Register mmrpntr

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2 to 0	pntr	R/W	0b000*	pointer to the internal register accessible by mmrdata (no post-increment/decrement function)
			0b011	selects mmr_pmem
			0b100	selects mmr_dmem
			others	reserved

Table 121: Register mmrdata (mmr_pmem)

Bit	Symbol	Access	Value	Description
15 to 6	-	R	0*	reserved
5	prom[2]	R/W	0*	monitors access to program memory bank PROM2
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
4	prom[1]	R/W	0*	monitors access to program memory bank PROM1
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
3	pram[2]	R/W	0*	monitors access to program memory bank PRAM2
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
2	pram[1]	R/W	0*	monitors access to program memory bank PRAM1
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
1	pram[0]	R/W	0*	monitors access to program memory bank PRAM0
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
0	prom[0]	R/W	0*	monitors access to program memory bank PROM0
			1	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)

Table 122: Register mmrdata (mmr_dmem)

Bit	Symbol	Access	Value	Description	
15	conflict	R/W		memory conflict flag - a simultaneous X and Y access to the same memory bank leads into a conflict, which is indicated with this flag	1
			0*	no conflict detected	2
			1	conflict detected	3
14	xyrom[2]	R/W		monitors access to data memory bank XYROM2	4
			0*	bank not accessed	5
13	xyrom[1]	R/W		monitors access to data memory bank XYROM1	6
			0*	bank not accessed	7
12	xyram[11]	R/W		monitors access to data memory bank XYRAM11	8
			0*	bank not accessed	9
11	xyram[10]	R/W		monitors access to data memory bank XYRAM10	10
			0*	bank not accessed	11
10	xyram[9]	R/W		monitors access to data memory bank XYRAM9	12
			0*	bank not accessed	13
9	xyram[8]	R/W		monitors access to data memory bank XYRAM8	14
			0*	bank not accessed	15
8	xyrom[0]	R/W		monitors access to data memory bank XYROM0	16
			0*	bank not accessed	17
7	xyram[7]	R/W		monitors access to data memory bank XYRAM7	18
			0*	bank not accessed	19
6	xyram[6]	R/W		monitors access to data memory bank XYRAM6	20
			0*	bank not accessed	21
5	xyram[5]	R/W		monitors access to data memory bank XYRAM5	22
			0*	bank not accessed	23
4	xyram[4]	R/W		monitors access to data memory bank XYRAM4	24
			0*	bank not accessed	25
3	xyram[3]	R/W		monitors access to data memory bank XYRAM3	26
			0*	bank not accessed	27
			1	bank accessed (the bit does not change until explicitly cleared)	28

Table 122: Register mmrdata (mmr_dmem) ...continued

Bit	Symbol	Access	Value	Description
2	xyram[2]	R/W		monitors access to data memory bank XYRAM2
			0*	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
1	xyram[1]	R/W		monitors access to data memory bank XYRAM1
			0*	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)
0	xyram[0]	R/W		monitors access to data memory bank XYRAM0
			0*	bank not accessed
			1	bank accessed (the bit does not change until explicitly cleared)

8.8.3 Functional description

The registers inside the MMR can be used for debugging purposes. The access to the two internal registers **mmr_pmem** and **mmr_dmem** is done by first selecting one of the two registers in **mmrpntr**, and then accessing the **mmrdata** register.

The main feature of this block is to detect simultaneous X/Y access to the same memory bank for debugging purposes. In such a case the common conflict bit will be set if a simultaneous access via the X and Y bus to the same memory bank occurs. Such accesses are not prevented and may lead to wrong behavior of the algorithm. The indicator bits and the memory conflict flag must be explicitly cleared by writing a 0 to the corresponding bit in the **mmr_dmem** register. The other bits indicate just an access to the associated data memory bank. The memory conflict flag is also set, when both X and Y buses access the same bank at the same address.

Please note that this is not a real error condition. The memory is accessed with correct address and the same (correct) data is placed on both buses. An application may use this feature to calculate efficiently the sum of squares of a vector.

8.9 MPU - Memory Paging Unit

8.9.1 Features

- Separate registers for P, X and Y address range; allowing parallel access to different pages
- Common page register for P, X and Y address range, to speed-up page settings
- Up to 32 pages supported per address range

8.9.2 Software interface

Table 123: Register overview of the MPU

Name	Description	I/O	Reset
pmpc	program memory paging control register	R/W	0x0000
xmpc	X data memory paging control register	R/W	0x0000
ympc	Y data memory paging control register	R/W	0x0000
pxympc	P/X/Y memory paging control register	R/W	0x0000

Table 124: Register pmpc

Bit	Symbol	Access	Value	Description
15 to 5	-	R		reserved
4 to 0	pmpa[4:0]	R/W		page number of active program memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31

Table 125: Register xmpc

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	xmpa[4:0]	R/W		page number of active X data memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31

Table 126: Register ympc

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	ympa[4:0]	R/W		page number of active Y data memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31

Table 127: Register pxympc

Bit	Symbol	Access	Value	Description
15	-	R	0*	reserved
14 to 10	pmpa[4:0]	R/W		page number of active program memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31
9 to 5	xmpa[4:0]	R/W		page number of active X data memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31
4 to 0	ympa[4:0]	R/W		page number of active Y data memory page
			0b00000*	page #0
			0b00001	page #1
		
			0b11111	page #31

8.9.3 Functional description

Memory paging is used to extend the DSP memory address spaces from 64 kwords to 1056 kwords. This is done separately for program and data memory space by selecting one out of 32 possible memory pages of 32 kwords size.

The MPU contains the registers **pmpc**, **xmpc** and **ympc** and controls the access of the selected pages. Having three memory paging control registers allows simple calculation of the page selection. Besides three separate paging control registers, one general (P/X/Y) paging control register (**pxympc**) is available, allowing to change all three pages in parallel with only one register access. The bit fields **pmpa**, **xmpa** and **ympa** in the **pxympc** register are always identical with the corresponding bit fields in the **pmpc**, **xmpc** and **ympc** registers.

8.9.3.1 Program memory paging and page switching

The Saturn DSP core supports paging for the upper 32 kwords of the address spaces (0x8000 ... 0xFFFF) only. All addresses below (0x0000 ... 0x7FFF) refer to the unpaged part of the address space. In case an access (program, x-data or y-data) falls into the paged address range, the accessed page is defined by the related paging register fields (**pmpa**, **xmpa** or **ympa**).

If an address in the lower half of the address space is applied, either the PRAM0/1/2 or the PROM0 bank is accessed.

Page switching means changing the **pmpa** value followed by a jump or a subroutine call to the upper half of the address space. There is no NOP instruction needed before jumping to the new page (except the page switching is done on address 0x7FFF). Page switching must only be done by program code located in the lower half of the address space. The **pmpa** value must not be changed by program code located in pages 0 to 31.

From pages 0 to 31 it is only possible to jump back or return from subroutine either to the same page or to the unpaged lower half of the program memory address space. The hardware does not guarantee correct direct switches between pages 0 to 31.

8.9.3.2 Data memory paging

As for program memory paging, the paged address range is the upper half of the data memory address space. Whenever data with an address in this range is requested, the 32 kword section which corresponds to the actual setting of **xmpa** or **ympa** is accessed, depending whether X or Y data is requested. By having two separate registers for X and Y memory page, it is possible to have X and Y data located on two different pages without intermediate page switching.

Page switching for data memory is done by setting **xmpa** and **ympa**.

CAUTION



Data memory must not be accessed in the instruction immediately following the write instruction of the **xmpa** and/or **ympa**.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.10 TIO - Test I/O Interface

8.10.1 Features

- The TIO is the interface between the DSP and the JTAG TAP controller TAPC4
- It allows the transfer of words between the DSP and the serial interface of the JTAG for debug purpose
- A DMA channel can be used for fast transfer of memory blocks to the JTAG port (trace functionality)

8.10.2 Block diagram

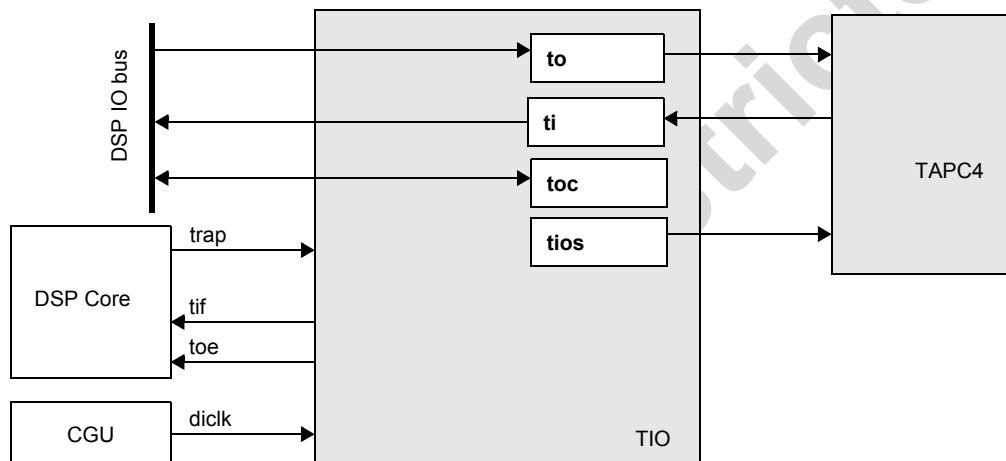


Fig 48. TIO block diagram

8.10.3 Software interface

Table 128: Register overview of the DSP side of the TIO

Name	Description	I/O	Reset
toc	TIO control register	R/W	0x8000
ti	TIO input data register	R	undefined
to	TIO output data register	W	undefined

Table 129: Register toc

Bit	Symbol	Access	Value	Description
15	toe	R		DSP output data flag
			0	DSP output data register full
			1*	DSP output data register empty

Table 129: Register toc...continued

Bit	Symbol	Access	Value	Description
14	tomode	R/W		operation mode select
			0*	automatic mode
			1	normal mode ^[1]
13 to 0	tocnt[13:0]	R/W	0*	counter value for automatic mode

[1] When the DMAC channel is used to transfer words to the JTAG output port or the JTAG port is used for debugging, this bit has to be set (disable Automatic mode).

Table 130: Register ti

Bit	Symbol	Access	Value	Description
15 to 0	ti[15:0]	R	undefined*	data input register of DSP

Table 131: Register to

Bit	Symbol	Access	Value	Description
15 to 0	to[15:0]	W	undefined*	data output register of DSP

8.10.4 Debug interface via JTAG

Table 132: Register overview of the JTAG side of the TIO

Name	Description	I/O	Reset
tios	TIO status register	R	0b1000
ti	TIO input data register	W	undefined
to	TIO output data register	R	undefined

[1] In this chapter the read/write access is seen from the external point of view, i.e., R means the JTAG port can read this bit/register, W means the JTAG port can write this bit/register.

Table 133: Register tios

Bit	Symbol	Access	Value	Description
3	tie	R	1*	shows the status of the ti register, ti is empty when set
2	tof	R	0*	shows the status of the to register, to is full when set
1	trap	R	0*	trap reached: the bit is set when the trap-service routine is started and cleared after tios is read via JTAG; trap is not set, if a bbbtrap is triggered by BBB.
0	-	R	0*	reserved

Table 134: Register ti

Bit	Symbol	Access	Value	Description
15 to 0	ti[15:0]	W	undefined*	data input register of DSP

Table 135: Register to

Bit	Symbol	Access	Value	Description
15 to 0	to[15:0]	R	undefined*	data output register of DSP

8.10.5 Functional description

8.10.5.1 Transfer from DSP to JTAG

Following a DSP reset, the **toe** flag is set as the **to** register is empty. The **tios.tof** flag, accessible by JTAG, is cleared when no output data is available.

Any write operation to the **to** register by DSP will clear the **toc.toe** flag and set the **tios.tof** flag. A write to the **to** register when **to** is full is not allowed by the application and may result in undefined data.

Three different transfer modes can be distinguished.

- Normal mode without DMA
- Normal mode with DMA
- Automatic mode without DMA

Normal mode without DMA

The following sequence is performed when **tomode** is set and the internal DMA channel 1 is disabled. Any read operation to the **to** register via JTAG sets the **toe** flag and clears the **tof** flag. In case the **torq** interrupt is enabled, the **isr.torq** flag is set. A read attempt via JTAG of **to** when **to** is empty is not allowed and results in undefined data.

Normal mode with DMA

The following sequence is performed when **tomode** is set and the internal DMA channel 1 is enabled. Any read operation to the **to** register via JTAG sets the **toe** flag and clears the **tof** flag. The next value to the **to** register is transferred by the DMAC, depending on its settings. The **torq** flag in the **isr** register is set depending on the settings of the DMAC channel (e.g. a programmed number of DMA transfers). A read attempt via JTAG **to** when **to** is empty is not allowed by the application and results in undefined data.

CAUTION



The DMAC will continue after the DMAC has notified the application by an interrupt. Additional data transfers may happen in case the interrupt cannot be executed immediately due to another active interrupt routine.

Automatic mode without DMA

The following sequence is performed when **tomode** is reset and the internal DMA channel 1 is disabled. A DSP interrupt **torq** is only issued when the **tocnt** value is non-zero. Any write to the **to** register causes a decrement of the **tocnt**. In this way a simple way of transferring a block of data from DSP to JTAG is possible. DSP sets up the data and the interrupt, and then writes the number of words to transfer to **tocnt**.

After the interrupt has transferred the appropriate number of words the interrupt is automatically blocked by **tocnt** reaching zero. Note that in this mode, the **torq** interrupt is forwarded directly to the **isr** register as the DMA channel is disabled.

8.10.5.2 Transfer from JTAG to DSP

During reset the **isr.tif** flag is cleared as no input data is available in **ti**. The **tios.tie** flag accessible via JTAG is set as data may be written into **ti**.

Any completed write operation via JTAG to the **ti** register will clear the **tie** flag (**tios** register) and set the **isr.tif** flag. A write to the **ti** register via JTAG when the **tios.tie** flag is clear is not allowed and may result in undefined data in **ti**.

In case an interrupt is enabled at DSP side for the **isr.tif** flag DSP will enter an interrupt routine in order to read the word from the **ti** register.

Any read operation of DSP to the **ti** register clears the **isr.tif** flag and sets the **tios.tie** flag. A read to the **ti** register when **isr.tif** flag is clear is not allowed and may result in reading undefined data from the **ti** register.

8.10.6 Application information

To program the different modes of the TIO output as described in [Section 8.10.5.1 "Transfer from DSP to JTAG"](#), the following register settings are recommended (**torq** interrupt in **ier** can be enabled/disabled as usual).

8.10.6.1 Normal mode without DMA

```

#define DMAC_EA      0x0050
#define DMAC_SA      0x0040
#define DMAC_CNT     0x0030
#define DMAC_DES     0x0020
#define DMAC_SRC      0x0010
#define DMAC_CMD      0x0000
#define DMAC_CHANNEL_TORQ 0x0002
#define DMAC_DEST_IO   (2 << 3)
#define DMAC_SRC_XMEM  (1 << 1)
#define DMAC_SRC_PMEM  (3 << 1)
#define DMAC_SRC_PTR_INC (1 << 8)
#define DMAC_TM_WORD_MODE (1 << 6)
#define DMAC_ENABLE    (1 << 0)
#define DMAC_TM_FORWARDING DMAC_TM_WORD_MODE
#define TOC_TOMODE     (1 << 14)
toc = TOC_TOMODE;           // set normal mode
dma_ir = DMAC_CHANNEL_TOE | DMAC_CMD; // select command register of TOE channel
dma_dr = DMAC_TM_FORWARDING; // disable channel and set transparent mode

```

8.10.6.2 Normal mode with DMA

```

toc = TOC_TOMODE;           // set normal mode
dma_ir = DMAC_CHANNEL_TOE | DMAC_EA; // select end address register of TOE channel
dma_dr = TO_BUFFER_EA;       // initialize end address
dma_dr = TO_BUFFER_SA;       // initialize start address

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```
dma_dr = BUFFER_SIZE;          1
dma_dr = TO_ADDRESS;           2      // set destination to I/O address
dma_dr = TO_BUFFER_SA;         3      // source is to output buffer
// start DMAC transfer from XMEM to I/O with burst mode and pointer increment of
// pointer to XMEM
dma_dr =  DMAC_DEST_IO |       4
        DMAC_SRC_XMEM |       5
        DMAC_SRC_PINC |       6
        DMAC_TM_WORD_MODE |  7
        DMAC_ENABLE;          8
                                9
```

8.10.6.3 Automatic mode without DMA

```
dma_ir = DMAC_CHANNEL_TOE | DMAC_CMD; // select command register of TOE channel 14
dma_dr = DMAC_TM_FORWARDING;        // disable channel and set transparent mode 15
toc = NUMBER_OF_WORDS_TO_TRANSFER;   // the number of words has to be in the range of 16
                                    // 1 to 16383 (= 0x3FFF)
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

8.11 DMAC - Direct Memory Access Controller

8.11.1 Features

- Direct memory to I/O registers transfers for all X, Y or P address spaces
- Two internal DMA channels controlled by the DSP core
 - Channel 0 for **ti** (input channel for **tif**)
 - Channel 1 for **to** (output channel for **torq**)
- One external DMA channel controlled by the TMU
 - Channel for **tmu**
- With or without address pointer update
- One word per two **dclk**-cycles in word-transfer mode
- DMA bus access priority over DSP core
- No intermediate cycle between DMAs on different channels
- DMA latency of two **dclk**-cycles on a single channel
- Interrupt generation after completed block transfer of arbitrary size
- For the internal channels:
 - Modulo protection
 - DMAC programmable using only two DSP I/O registers
 - I/O interrupt forwarding for disabled internal DMA channels

CAUTION



The DMAC is not sensitive to memory paging. DMA is only allowed in un-paged memory space.

Be aware that the DMAC unit does not stop its operation after the predefined number of words are transferred. The counter is just reloaded with its original value. The minimum number of words has to be > 1.

It is not recommended to use the DMAC unit for TIO data transfers.

8.11.2 Block diagram

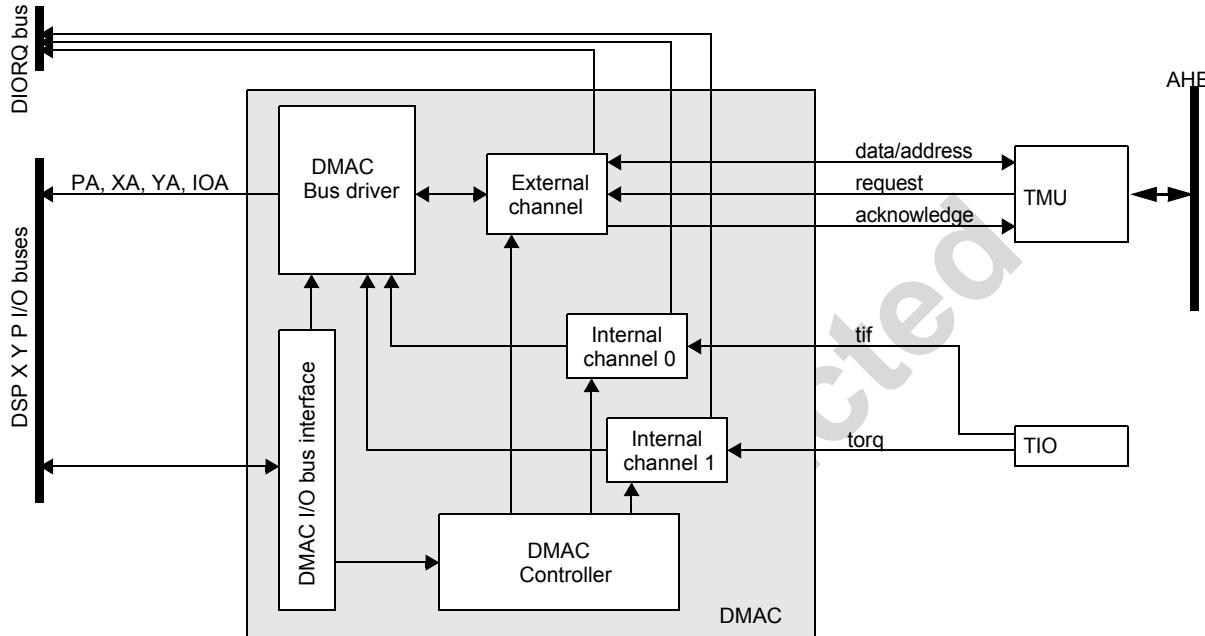


Fig 49. DMAC block diagram

8.11.3 Software interface

Table 136: Register overview of DMAC

Name	Description	I/O	Reset
dmair	DMA instruction register	R/W	0x0000
dmadr	DMA data register (virtual register)	R/W	[1]

[1] Reset value differs for the various data registers. See Table 138 to Table 144 for details.

The data register **dmadr** is a virtual register allowing access to the seven physical data registers **cmd**, **src**, **des**, **cnt**, **sa**, **ea** and **req** which are described in this section. The selection is done by writing to the **reg[2:0]** field of **dmair**.

The **reg[2:0]** field in the **dmair** has an auto-decrement. On each access of the (virtual) data register **dmadr**, either read or write, the value of the field **reg[2:0]** will be decremented by one. This makes it simpler to read or write from or to all registers belonging to an internal DMA channel.

Table 137: Register dmair

Bit	Symbol	Access	Value	Description
15 to 7	-	R	0*	reserved

Table 137: Register dmair...continued

Bit	Symbol	Access	Value	Description
6 to 4	reg[2:0]	R/W		DMA channel register to be accessed
			0b000*	cmd - command register
			0b001	src - source pointer
			0b010	des - destination pointer
			0b011	cnt - counter register
			0b100	sa - start address for modulo protection
			0b101	ea - end address for modulo protection
			0b110	reserved
			0b111	req - request register (independent of content of chan[3:0])
3 to 0	chan[3:0]	R/W		DMA channel number , ordered by priorities (see Table 145)
			0b0000*	internal channel tif - ti register full interrupt
			0b0001	internal channel torq - to register empty interrupt
			others	reserved

Table 138: Register dmadr (cmd - command register)

Bit	Symbol	Access	Value	Description
15 to 11	-		0*	reserved
10	mp	R/W		modulo protection on the source register
			0*	disabled
			1	enabled
9	dppi	R/W		destination pointer post-increment
			0*	disabled
			1	enabled
8	sppi	R/W		source pointer post-increment
			0*	disabled
			1	enabled
7	pdi	R/W		DMA processing during interrupt request and processing to the DSP core
			0*	disabled (Disables the DMA processing during the time the DMAC waits for the interrupt acknowledge from the DSP core)
			1	enabled (Allows to continue the DMA processing regardless whether the interrupt has been acknowledged by the DSP core or not.) CAUTION: This may lead to missing interrupts in case the interrupt acknowledge is postponed by other ongoing interrupt processing.)
6	tm	R/W		in case the channel is enabled (bit en set): Transfer mode
			0	word-transfer
			1*	burst-transfer (not supported by the DSP peripherals)
				in case the channel is disabled (bit en reset): forwarding DMA/interrupt request to DSP core (corresponding DSP interrupt, see Section 8.5)
			0	forwarding off
			1*	forwarding on

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 138: Register dmadr (cmd - command register)...continued

Bit	Symbol	Access	Value	Description
5	swap	R/W		swap transfer direction
			0*	no swap
			1	swap source and destination address
4 to 3	dsel[1:0]	R/W		destination selection
			0b00*	XMEM
			0b01	YMEM
			0b10	I/O
2 to 1	ssel[1:0]	R/W		source selection
			0b00*	XMEM
			0b01	YMEM
			0b10	I/O
0	en	R/W		enable/disable the channel
			0*	disabled
			1	enabled

Table 139: Register dmadr (src - source pointer)

Bit	Symbol	Access	Value	Description
15 to 0	src[15:0]	R/W	0*	source pointer: pointer to either P, X or Y memory or to I/O space, determined by the command loaded in the command register

Table 140: Register dmadr (des - destination pointer)

Bit	Symbol	Access	Value	Description
15 to 0	des[15:0]	R/W	0*	destination pointer: pointer to either P, X or Y memory or to I/O space, determined by the command loaded in the command register

Table 141: Register dmadr (cnt - counter register)

Bit	Symbol	Access	Value	Description
15 to 0	cnt[15:0]	R/W	0*	counter register defining the length of one burst: counts down on every transfer when its value is unequal to zero; when this register equals value one, a DMA interrupt signal is generated and the counter is reloaded with its initial value; if cnt = 0, the counter is disabled

Table 142: Register dmadr (sa - start address for modulo protection)

Bit	Symbol	Access	Value	Description
15 to 0	sa[15:0]	R/W	0*	start address for modulo protection of the source pointer

Table 143: Register dmadr (ea - end address for modulo protection)

Bit	Symbol	Access	Value	Description
15 to 0	ea[15:0]	R/W	0*	end address for modulo protection of the source pointer

Table 144: Register dmadr (req - request register) [1]

Bit	Symbol	Access	Value	Description
15 to 4		R	0*	reserved
3	torq	R	0*	TIO DMA request status
			0*	DMA request inactive in previous cycle
			1	DMA request active in previous cycle
2	tif	R	0*	TIO DMA request status
			0*	DMA request inactive in previous cycle
			1	DMA request active in previous cycle
1	-	R	0*	reserved
0	tmu	R	0*	TMU DMA request status
			0*	DMA request inactive in previous cycle
			1	DMA request active in previous cycle

[1] The value of the fields in this register do not depend on the value programmed in **dmair.chan[3:0]**.

8.11.4 Functional description

The DMAC is able to transfer words from I/O registers to any of the RAM blocks (P and shared XY) and vice versa. It is not possible to transfer data from memory to memory.

The source and the destination of the transfer are stored in two registers, the **src** register and the **des** register. The internal DMAC channels can be programmed to post-increment the source and destination register. On the source register, the post-increment can be modulo protected by programming the start address **sa** and the end address **ea**.

A counter can be used in order to generate an interrupt to the DSP core after a certain number of DMAs have been executed. This enables the DSP idle mode in which the DSP clock is disabled while the DMAC is collecting data from, e.g., I/O peripherals. The interrupt to the core wakes-up the DSP core after finishing the data transfer. In order to do so, the count register **cnt** must be loaded with the appropriate value. Each DMA transfer then decrements the value of **cnt** by one. Once the count register reaches one, it is restored to its initial value and a new series of DMA can be processed. The counter as well as the interrupt generation are disabled by setting **cnt** to its reset value 0x0000.

The DMAC uses the same buses as the DSP core to access the memories. Therefore the DSP core is stalled whenever the DMAC performs a transfer.

The execution of a DMA requires two DSP clock cycles. The first cycle is needed to detect a DMA request and to set-up the appropriate buses in the requested channel. In the second cycle the actual execution takes place and cannot be interrupted anymore. While one DMA is executed, the next DMA can be decoded. Hence, there is no interval between DMAs of different channels.

The DMAC neither protects nor prevents over-writing the current program that is being executed. This exception has to be handled correctly by the programmer.

8.11.4.1 The command register

Modulo protection is enabled by setting the bit **mp** in the **cmd** register. To allow modulo protection on the destination register and to change the direction of the DMA transfer easily, the source and destination register can be swapped: By setting bit **swap** in the **cmd** register, the source and destination as selected with **ssel[1:0]** and **dsel[1:0]** bit in **cmd** are swapped without changing the **ssel[1:0]** and **dsel[1:0]** bits. In doing so, the source register becomes the destination register whereas the destination register becomes the source register.

The bit **tm** in the **cmd** register selects between Burst-transfer mode and Word-transfer mode. Burst-transfer mode is not supported by the DSP peripherals and must not be used. In Word-transfer mode, each execution cycle is preceded by a set-up cycle. In this mode the maximum DMA transfer rate is half of the clock rate of the DSP.

If a DMA channel is not enabled and the bit **tm** is set in the **cmd** register, each interrupt of the peripheral is forwarded to the interrupt input of the DSP directly, i.e. the corresponding bit in the register **isr** will be set. The request can then be handled as a normal interrupt.

If bit **pdi** in the **cmd** register is cleared and an interrupt to the DSP has been asserted, then all other DMAs on this channel are ignored until the core interrupt routine is finished. If bit **pdi** is set, new DMAs on the channel will be serviced.

8.11.4.2 DMA channel priorities

The priorities of the DMA channels are according to the DMA controller specification.

Table 145: DMA channel priorities

Channel	Peripheral	Action	Priority	IORQ
external channel 0	TMU	read/write	1 (highest)	
external channel 1	-	-	2	-
Internal channel 0	TIO	read ti	3	tif
Internal channel 1	TIO	write to	4 (lowest)	torq

8.12 TMU - DSP Mailbox unit

8.12.1 Overview

The mailbox unit is based on the TANDEM mailbox unit (see [10.]) and links the Saturn DSP with the integrated DMA controller (see [10.]) to the AHB system bus. Through this unit any AHB bus master can perform read and write accesses to the DSP's local data and program memories. Using the semaphore and interrupt facilities provided by the Mailbox Unit, the ARM system controller and DSP communicate with each other through shared locations in the DSP data memories.

The place of the Mailbox Unit in the system is presented in [Figure 50](#)

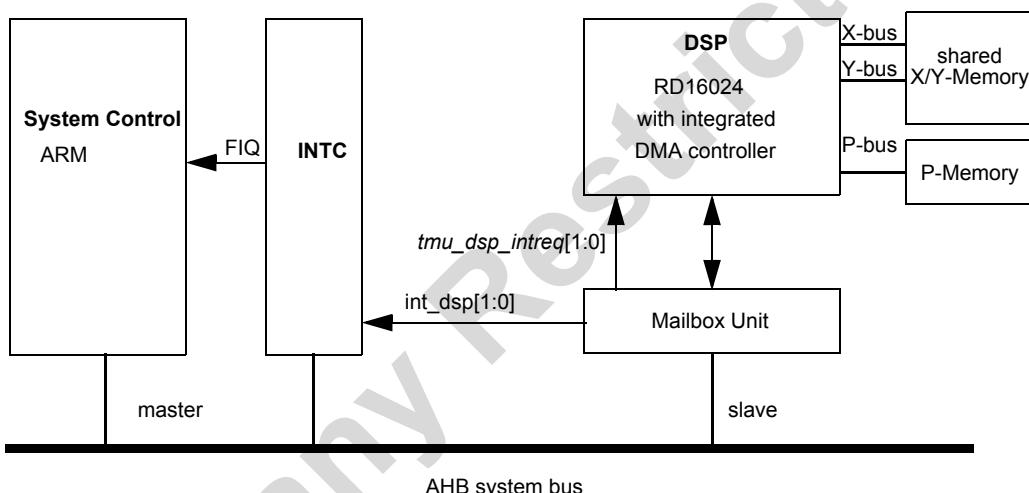


Fig 50. Dual core SC/DSP System with Mailbox Unit

8.12.2 Operation Specification

The TMU has three communication mechanisms:

- AHB system bus access to Mailbox Registers,
- DSP Access to the Mailbox Registers,
- DMA Operation (AHB system bus access to DSP memories).

8.12.2.1 AHB System Bus Communication

The Mailbox Unit is implemented as a AHB system bus slave device with a 32-bit wide data bus (see [10.]).

8.12.2.2 AHB System Bus Access to Mailbox Registers

The Mailbox Unit contains several registers mapped in the AHB address space. The mapping of the location within the total AHB address space is defined by the address decoder of the AHB. The address map is defined in [Section 8.12.3](#).

8.12.2.3 DSP Access to the Mailbox Register

The Mailbox Unit contains one register mapped in the DSP IO address space. The mapping of the location within the total IO space is defined by an address decoder external to the Mailbox Unit. The address decoder asserts the `dsp_tmu_sel` signal when the IO address points to the TMU register.

Both the read and write access is performed in one `dcclk` cycle. Hence, if both AHB and DSP are accessing the TMU registers, the DSP access has a higher priority to guarantee single cycle access.

8.12.2.4 DMA Operation (AHB Access to DSP Memories)

The TMU uses an external channel of the DMA facility of the Saturn DSP core [10.] to transfer data between the AHB and one of the DSP memories. The TMU selects a memory and supplies an address within the memory space, the access direction (read/write) and data (for a write access) to the external DMAC channel. Once these signals are stable, the TMU asserts the DMA request line to start the direct memory access. The access is complete when the DMAC asserts the acknowledge input of the TMU.

The values of the `tmu_dma_addr[18:0]` and the `tmu_dma_dir` are derived from the address and data direction of the AHB (see [Section 8.12.3](#)).

All signals between the TMU and the DMA/DSP have to be synchronous to the rising edge of `diclk`. The number of DMA operations to be performed sequentially per AHB operation, depend on the AHB operation code:

- A half-word (16-bit) transmission results in a single cycle DMA operation,
- A word (32-bit) transmission leads to two consecutive DMA operations (burst transfer).

8.12.2.5 Interrupts

The communication between the system controller and the DSP is restricted by the common accessible memory on the DSP and the TMU registers (for the AHB side see [Table 147 - 150](#), for the DSP side see [Table 153](#)).

Therefore only software defined protocols are usable. The signalling has to be done by employing the `tmu_dsp_intreq[1:0]` and the `int_dsp[1:0]` interrupts, or by regular inspection (polling) of the related status information in the TMU communication registers:

- `DSP_INT_STATUS` indicates the presence of a communication request from the SC towards the DSP,
- `SC_INT_STATUS` indicates the presence of a communication request from the DSP towards the SC.

They can be set and cleared indirectly through write actions to the corresponding “interrupt status set” and “interrupt status clear” fields.

- The bit DSP_INT_ENABLE enables - when set - the generation of an interrupt request towards the DSP via the tmu_dsp_intreq[1:0] lines. The bit can be accessed by the DSP only. Its state can be changed through write actions to the corresponding DSP_INT_ENABLE_SET and DSP_INT_ENABLE_CLEAR bits. 1
2
3
4
- The bit SC_INT_ENABLE enables - when set - the generation of an interrupt request towards the SC via the int_dsp[1:0] lines and the INTC. The bit can be directly read and written by the SC only. 5
6
7
8

8.12.2.6 Example application

A possible application to pass information from the SC to DSP OS might look like (see also Fig 51.):

1. The SC writes the information to be transferred at a certain defined location in the DSP data memory. 13
14
2. The SC generates for example the tmu_dsp_intreq0 interrupt to inform the DSP about the communication request. 15
16
3. The DSP core gets the interrupt request and starts to process the interrupt service function. 18
19
4. The DSP interrupt function reads the information provided by the SC in the DSP data memory, 20
21
5. and disables the DSP_INT_ENABLE0 flag and clears the interrupt request using the DSP_INT_STATUS_CLEAR0. 23
24
6. The DSP interrupt function starts the required actions, based on the passed information 25
26
7. The DSP enables the DSP_INT_STATUS_SET0 again and returns to its normal processing. 27
28

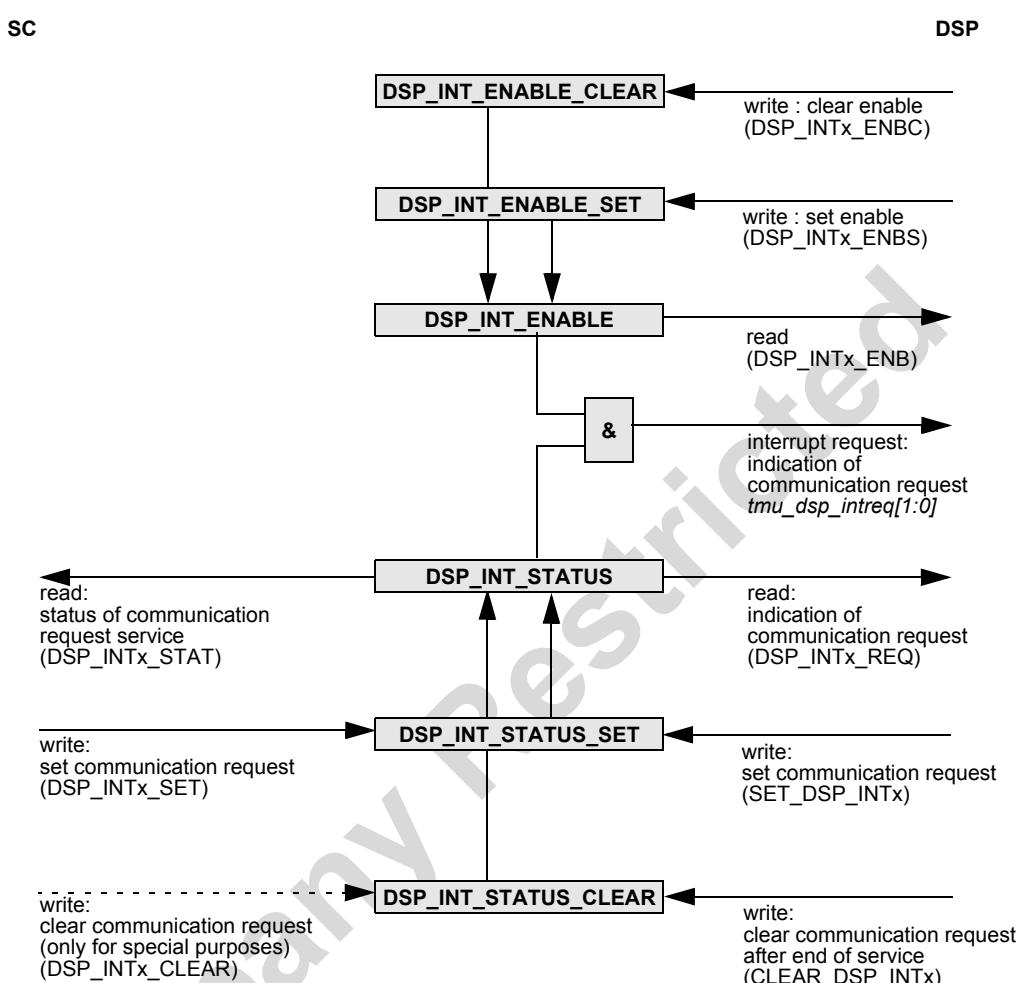


Fig 51. SC to DSP Communication SW Protocol Support

Other protocols might be implemented.

Similar protocols can be realized for the communication from the DSP to the SC (see Fig 52.)

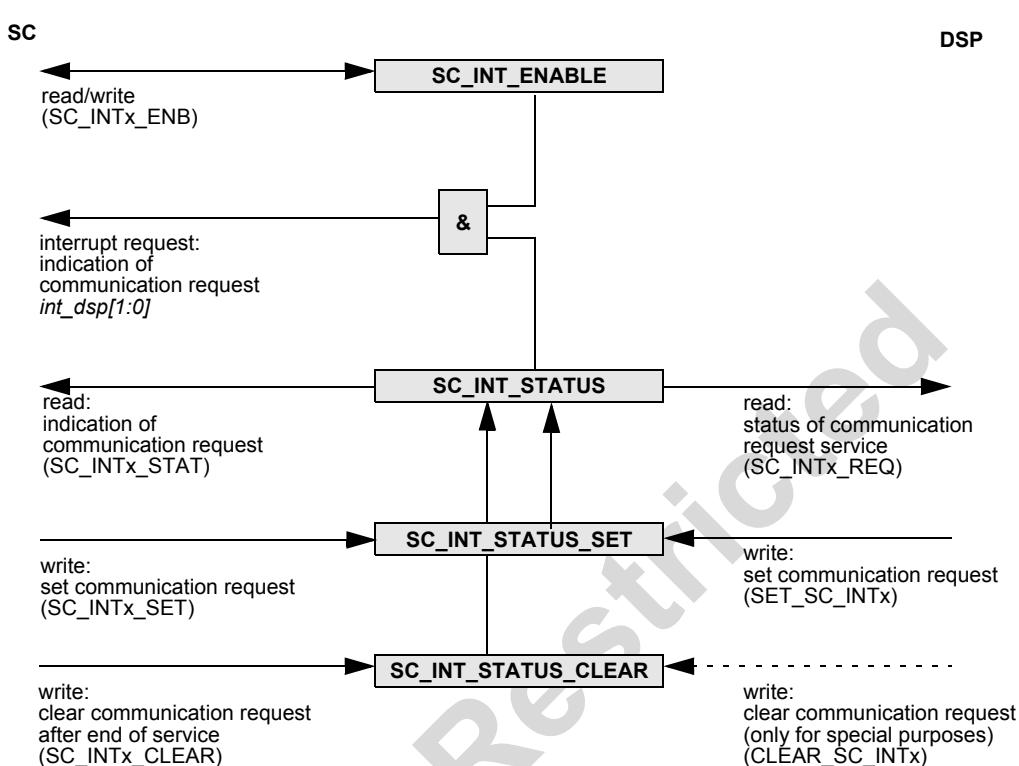


Fig 52. DSP to SC Communication SW Protocol Support

8.12.2.7 Stall DSP Program Execution

DSP program execution can be stalled by clearing the DSP_STALL bit in the register **mu_ctrl**. DSP_STALL is set by the AHB reset signal and remains set until it is cleared by the SW of the system controller. Note that by clearing **mu_ctrl.DSP_STALL**, any DSP IO operation (DMA, BBB, etc.) will be impossible as well.

The applications of the DSP_STALL bit are:

- When DSP program memory is implemented as RAM, then it is required to initialize the memory content before DSP program execution is started. The download of the DSP software into the program memory can be performed by an AHB master (ARM or JTAG debug port).
- For debug reasons, to have unrestricted access to the DSP memories. The DSP program execution is stalled temporarily.

8.12.2.8 Power Management

The interrupt mechanism of the Mailbox Unit terminates system (SC) or DSP power management modes:

- A DSP interrupt request terminates the DSP Idle mode and restarts DSP core clock.

- A SC interrupt request terminates SC or AHB level power management modes and restarts clock within the SC and to AHB agents.

8.12.2.9 Reset

An AHB system bus reset (BnRES) initializes the circuitry within the Mailbox Unit. The reset signal has to be active for a period of at least 3 AHB clock periods + 3 DSP clock periods in which both clocks are enabled.

8.12.3 SC Software Interface Specification (AHB side)

The TMU manages an address space of 512 KByte on the AHB system bus (AHB) which is partitioned in four sections each of 128 KByte:

- TMU Register section
- the “DSP X-Memory” section,
- the “DSP Y-Memory” section,
- the “DSP P-Memory” section, and
- the “DSP IO Register” section

Through the memory address sections, DSP local data memory (X/Y) and program memory (P) can be read or write accessed by the AHB. DSP memory accesses from AHB may be 16-bit or 32-bit wide; 8-bit wide accesses are illegal.

After enabling the dcclk or diclk via **cgudspcon.dcclken** and **cgudspcon.diclken** a latency of up to 5 (**tbd**) ARM clocks has to be taken into account before writing to the TMU control registers. Reasons are clock enable synchronization in CGU and extra VPB bridge latency for CGU control registers compared to TMU control registers on AHB.

Table 146: SC accessible TMU registers

Name	Description	I/O	Reset
mu_int_status	Mailbox Interrupt status	R	0x0000 0000
mu_int_enable	Mailbox Interrupt enable	R/W	0x0000 0000
mu_int_set	Mailbox Interrupt Set	W	0x0000 0000
mu_int_clear	Mailbox Interrupt clear	W	0x0000 0000
mu_ctrl	Mailbox Control	R/W	0x0000 0000

Table 147: Register mu_int_status - TMU interrupt status register

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	sc_int1_stat	R	0*	SC Interrupt Status 1 (set by the DSP)
			0*	no SC interrupt int_dsp1 pending (= no DSP communication request towards the SC)
			1	SC interrupt int_dsp1 pending (= DSP communication request towards the SC)

Table 147: Register mu_int_status - TMU interrupt status register

Bit	Symbol	Access	Value	Description
2	dsp_int1_stat	R		DSP Interrupt Status 1 (set by the SC)
			0*	no DSP interrupt tmu_dsp_intreq1 pending (= no SC communication request towards the DSP)
			1	DSP interrupt tmu_dsp_intreq1 pending (= SC communication request towards the DSP)
1	sc_int0_stat	R		SC Interrupt Status 0 (set by the DSP)
			0*	no SC interrupt int_dsp0 pending (= no DSP communication request towards the SC)
			1	SC interrupt int_dsp0 pending (= DSP communication request towards the SC)
0	dsp_int0_stat	R		DSP Interrupt Status 0 (set by the SC)
			0*	no DSP interrupt tmu_dsp_intreq0 pending (= no SC communication request towards the DSP)
			1	DSP interrupt tmu_dsp_intreq0 pending (= SC communication request towards the DSP)

Table 148: Register mu_int_enable - TMU interrupt enable register

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	sc_int1_enb	R/W		SC Interrupt Enable 1
			0*	disable SC interrupt request 1
			1	enable SC interrupt request 1
2	-	R/W	0*	reserved
1	sc_int0_enb	R/W		SC Interrupt Enable 0
			0*	disable SC interrupt request 0
			1	enable SC interrupt request 0
0	-	R	0*	reserved

Table 149: Register mu_int_set - TMU interrupt set command register

Bit	Symbol	Access	Value	Description
31 to 4	-	-	0*	reserved
3	sc_int1_set	W		SC Interrupt Status Set 1
			0	no effect
			1	set SC_INT_STATUS1 bit (= SC issues a communication request to itself; only for special purposes)
2	dsp_int1_set	W		DSP Interrupt Status Set 1
			0	no effect
			1	set DSP_INT_STATUS1 bit (= issue SC a communication request towards DSP)
1	sc_int0_set	W		SC Interrupt Status Set 0
			0	no effect
			1	set SC_INT_STATUS0 bit (= SC issues a communication request to itself; only for special purposes)

Table 149: Register mu_int_set - TMU interrupt set command register

Bit	Symbol	Access	Value	Description
0	dsp_int0_set	W		DSP Interrupt Status Set 0
			0*	no effect
			1	set DSP_INT_STATUS0 bit (= issue SC a communication request towards DSP)

Table 150: Register mu_int_clear - TMU interrupt clear command register

Bit	Symbol	Access	Value	Description
31 to 4	-	-	0*	reserved
			1	SC Interrupt Clear 1
			0	no effect
3	sc_int1_clear	W	1	clear SC_INT_STATUS1 bit (= indicate that a DSP communication request has been served by the SC)
			0	DSP Interrupt Clear 1
			1	no effect
2	dsp_int1_clear	W	1	clear DSP_INT_STATUS1 bit (= cancel a SC communication request towards the DSP; only for special purposes)
			0	DSP Interrupt Clear 1
			1	no effect
1	sc_int0_clear	W	1	SC Interrupt Clear 0
			0	no effect
			1	clear SC_INT_STATUS0 bit (= indicate that a DSP communication request has been served by the SC)
0	dsp_int0_clear	W	1	DSP Interrupt Clear 0
			0	no effect
			1	clear DSP_INT_STATUS0 bit (= cancel a SC communication request towards the DSP; only for special purposes)

Table 151: Register mu_ctrl - TMU control register

Bit	Symbol	Access	Value	Description
31 to 1	-	-	0*	reserved
0	dsp_stall	R/W		Control of DSP Program Execution
			0*	request to stall DSP program execution and IO operations (e.g. DMA). All DSP clocks will be stalled when it is not in stand alone debug mode.
			1	no effect

8.12.4 Software Interface (DSP-IO side)

This register provides Interrupt Status fields that indicate the presence of SC and DSP interrupt conditions plus an Interrupt Enable field that controls whether an interrupt request towards the DSP is enabled, or not.

Manipulation of Interrupt Status and Enable fields can be performed through write actions to corresponding “Clear” and “Set” fields.

Table 152: DSP accessible TMU register

Name	Description	I/O	Reset
tmuctrl	Mailbox control register	R/W	0x0000

Table 153: Register tmuctrl

Bit	Symbol	Access	Value	Description	
15 to 12	-	R	0*	reserved	1 2 3 4 5 6 7 8
11	clear_sc_int1	W		Clear SC interrupt status 1	9
			0*	no effect	10
			1	clear SC_INT_STATUS1 bit (= cancel a DSP communication request towards the SC; for special purposes only)	11 12 13 14 15 16 17 18
10	set_sc_int1	W		Set SC interrupt status 1	19
			0	no effect	20
			1	set SC_INT_STATUS1 bit (= issue DSP communication request towards the SC)	21 22 23 24 25 26 27 28 29 30 31
9	sc_int1_req	R		Read SC interrupt status 1	32
			0*	no SC interrupt pending (= no DSP communication request, or DSP communication request has been serviced by the SC)	33
			1	SC interrupt pending (= DSP communication request and not yet serviced by the SC)	34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
8	dsp_int1_enbc	W		Clear DSP interrupt enable 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0*	no effect	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	clear DSP_INT_ENABLE1 bit	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
7	dsp_int1_enbs	W		Set DSP interrupt enable 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0	no effect	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	set DSP_INT_ENABLE1 bit	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
6	dsp_int1_enb	R		Read DSP interrupt enable 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0*	interrupt request towards the DSP is disabled	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	interrupt request towards the DSP is enabled. An interrupt request towards the DSP is generated when the DSP_INT_STATUS1 bit is set.	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
5	clear_dsp_int1	W		Clear DSP interrupt status 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0*	no effect	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	clear DSP_INT_STATUS1 bit (= indicate that a SC communication request has been served by the DSP)	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
6	set_dsp_int1	W		Set DSP interrupt status 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0	no effect	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	set DSP_INT_STATUS1 bit (= DSP issues a communication request to itself; only for special purposes)	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
5	dsp_int1_req	R		Read DSP interrupt status 1	19 20 21 22 23 24 25 26 27 28 29 30 31
			0*	no SC interrupt pending (= no SC communication request, or SC communication request has been serviced by the DSP)	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	DSP interrupt pending (= SC communication request and not yet serviced by the DSP)	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
5	clear_sc_int0	W		Clear SC interrupt status 0	19 20 21 22 23 24 25 26 27 28 29 30 31
			0*	no effect	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
			1	clear SC_INT_STATUS0 bit (= cancel a DSP communication request towards the SC; for special purposes only)	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54

Table 153: Register tmuctrl...continued

Bit	Symbol	Access	Value	Description
4	set_sc_int0	W		Set SC interrupt status 0
			0	no effect
			1	set SC_INT_STATUS0 bit (= issue DSP communication request towards the SC)
	sc_int0_req	R		Read SC interrupt status 0
			0*	no SC interrupt pending (= no DSP communication request, or DSP communication request has been serviced by the SC)
			1	SC interrupt pending (= DSP communication request and not yet serviced by the SC)
3	dsp_int0_enbc	W		Clear DSP interrupt enable 0
			0*	no effect
			1	clear DSP_INT_ENABLE0 bit
2	dsp_int0_enbs	W		Set DSP interrupt enable 0
			0	no effect
			1	set DSP_INT_ENABLE0 bit
	dsp_int0_enb	R		Read DSP interrupt enable 0
			0*	interrupt request towards the DSP is disabled
			1	interrupt request towards the DSP is enabled. An interrupt request towards the DSP is generated when the DSP_INT_STATUS0 bit is set.
1	clear_dsp_int0	W		Clear DSP interrupt status 0
			0*	no effect
			1	clear DSP_INT_STATUS0 bit (= indicate that a SC communication request has been served by the DSP)
0	set_dsp_int0	W		Set DSP interrupt status 0
			0	no effect
			1	set DSP_INT_STATUS0 bit (= DSP issues a communication request to itself; only for special purposes)
	dsp_int0_req	R		Read DSP interrupt status 0
			0*	no SC interrupt pending (= no SC communication request, or SC communication request has been serviced by the DSP)
			1	DSP interrupt pending (= SC communication request and not yet serviced by the DSP)

[1] Simultaneous setting and clearing of the same interrupt line in the register will lead to an undefined value of that bit in the register.

8.13 Programmer's Restrictions

The actual list of the restrictions the programmer has to take care can be found in [6.]

8.13.1 Software reset

the following sequence should be applied for a SW controlled DSP reset:

- enable DSP clocks (both dcclk and diclk) while DSP reset is inactive (resdsp = 0).
- wait (recommended value is 5 μ s)
- activate DSP reset by resdsp = 1
- wait (recommended value is 5 μ s)
- disable DSP clocks (both dcclk and diclk)
- deactivate DSP reset by resdsp = 0
- enable DSP clocks

Company Restricted

9. SC Subsystem

9.1 ARM926EJ-S - System Controller Core

9.1.1 Features

- ARM9EJ-S microprocessor core
 - ARMv5TEJ architecture
 - 32-bit ARM instruction set for high performance code
 - 16-bit Thumb instruction set for increased code density
 - Jazelle™ HW accelerator to accelerate Java bytecode execution
 - Harvard architecture with separate instruction and data paths
 - DSP instructions extension and single cycle MAC
 - 32 kbytes instruction cache and 32 kbytes data cache
 - Memory Management Unit to support complex operating systems
 - Clock rate up to 208 MHz (See [Table 626](#))
- The core, including caches, is running at up to 208 MHz (See [Table 626](#)) while Multi-layer AHB buses are running at up to 104 MHz (See [Table 626](#))
- Little endian operation

9.1.2 Block diagram

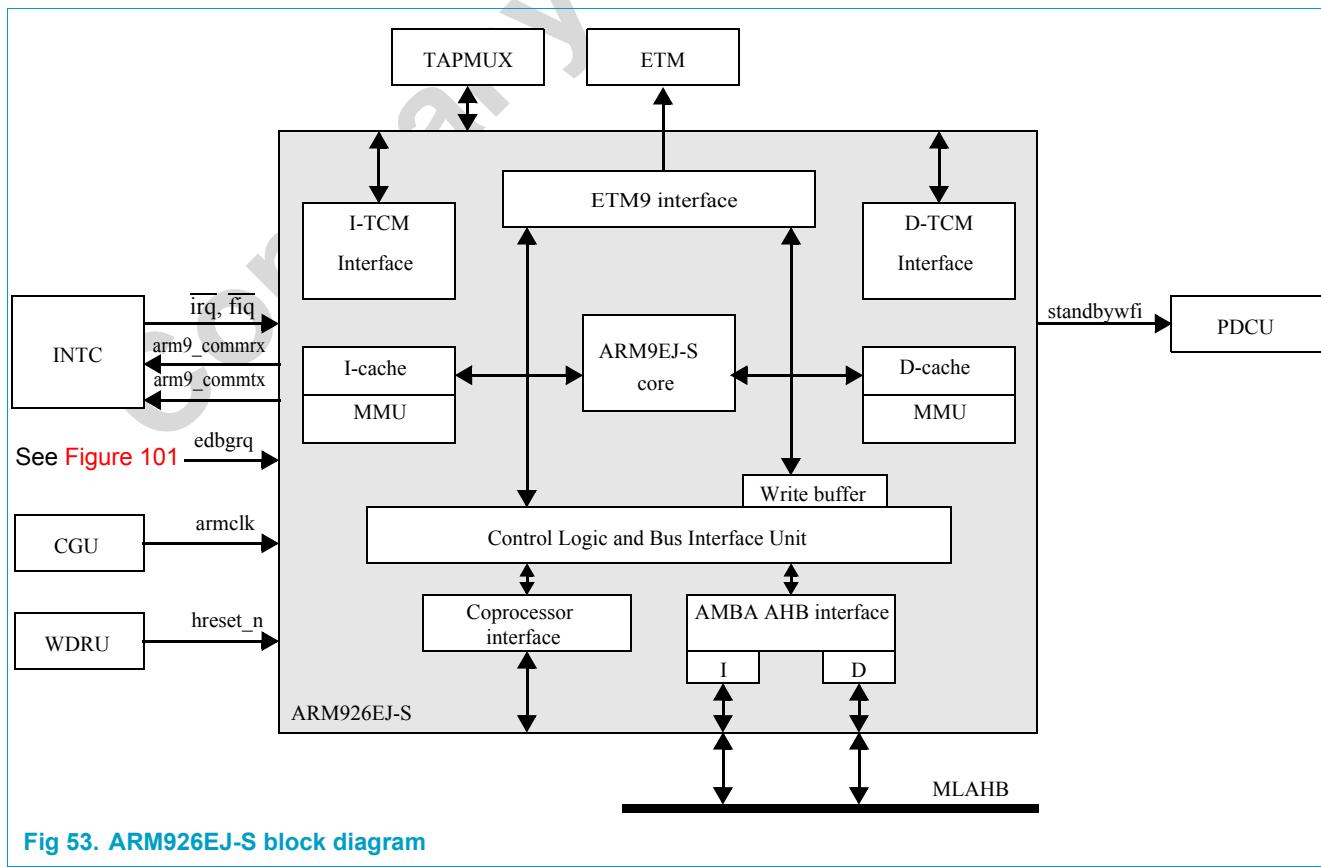


Fig 53. ARM926EJ-S block diagram

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.1.3 Functional description

The ARM926EJ-S is a powerful 32-bit RISC processor from ARM Ltd. A complete description is available in the ARM926EJ-S technical reference manual [12]. The ARM926EJ-S consists of an ARM9EJ-S processor core, 32 kbyte instruction cache and 32 kbyte data cache, a memory management unit, two AMBA AHB bus interfaces and a write buffer. The ARM926EJ-S processor is backwards code compatible with the ARM7 thumb family. The ARM9EJ-S core is an implementation of the ARMv5TEJ architecture with new instructions such as MAC and saturation arithmetic functions.

The system coprocessor (CP15) is used to configure and control the ARM926EJ-S processor. The caches, Memory Management Unit (MMU) and most of the other system options are controlled using CP15 registers. These registers are accessible only with MCR and MRC instructions and in privilege mode. CDP, LDC, STC, MCRR, MRRC instructions and unprivileged MRC or MCR instructions to CP15 cause the UNDEFINED instruction exception to be taken. Refer to [12] for more details.

Even though the ARM core and some peripherals support both little- and big-endian configurations, only the little-endian mode is supported by the SC sub-system. Therefore, when peripherals endianess can be configured, it should be configured in little-endian mode.

9.1.3.1 Addresses in an ARM926EJ-S system

Three distinct types of addresses exist in an ARM926EJ-S system (see Table 154).

Table 154: Address types in ARM926EJ-S

Domain	ARM926EJ-S	Caches and MMU	TCM and buses
Address type	Virtual address (VA)	Modified Virtual Address (MVA)	Physical Address (PA)

This is an example of the address manipulation that occurs when the ARM9EJ-S core requests an instruction:

1. The VA of the instruction is issued by the ARM9EJ-S

The VA is translated using the FCSE PID value to the MVA. The instruction cache (ICache) and Memory Management Unit (MMU) detect the MVA. See [12] page 2-34

2. If the protection check carried out by the MMU on the MVA does not abort and the MVA tag is in the ICache, the instruction data is returned to the ARM9EJ-S core.
3. If the protection check carried out by the MMU on the MVA does not abort, and the cache misses (the MVA TAG is not in the cache), then the MMU translates the MVA to produce the PA. This address is given to the AMBA AHB bus interface to perform an external access.

9.1.3.2 Wait For Interrupt Mode

The ARM926EJ-S can be put in low power mode by writing to register 7 of CP15. When entering this mode, the ARM926EJ-S flushes its write-buffers and request to stop the clock by using the **standbywfi** signal to PDCU. This suspends execution of further instructions until an interrupt (fiq, irq or edbgrq) is asserted.

In case of nIRQ or nFIQ, the processor core is woken up regardless of whether the interrupts are enabled or disabled. If interrupts are enabled, the ARM926EJ-S is guaranteed to take the interrupt before executing the instruction after the wait for interrupt. Refer to [12.] for more details.

Additional low power modes can be enabled with the PDCU, see [Section 7.3](#) for details.

9.1.3.3 Caches

The instruction and data cache are 32kByte each and organized as:

- 4-way associative
- 256 tags per way
- 8 words (32bytes) per line

The data cache supports write-back and write through. In case of write back the physical address is used (stored in addition to the tag ram). In all other cases the MMU is used for modified virtuell addressing.

9.2 SC Bus Configuration

9.2.1 Overview

The architecture of the SC is based on a Multi-layer AHB bus which is a matrix interconnecting the masters and the slaves. Many masters can access different slaves simultaneously. All datapaths are 32-bit wide.

The Multi-layer AHB bus contains the address decoding (one address decoder per master) and the arbitration (one arbiter per slave).

As shown in [Figure 54](#), there are 6 masters (and therefore 6 layers) with 11 slaves (or slave layers). [Table 155](#) shows the slaves which can be accessed by each master. As long as they don't access the same slave, all masters can simultaneously perform accesses.

Whenever a slave port's current master generates a non-sequential or an idle access, the slave port is re-arbitrated. One **hclk** wait cycle is introduced before the slave can be accessed again. Therefore sequential transfers have speed advantages.

Any access from a master to a slave for which there is no physical connection generates a bus error.

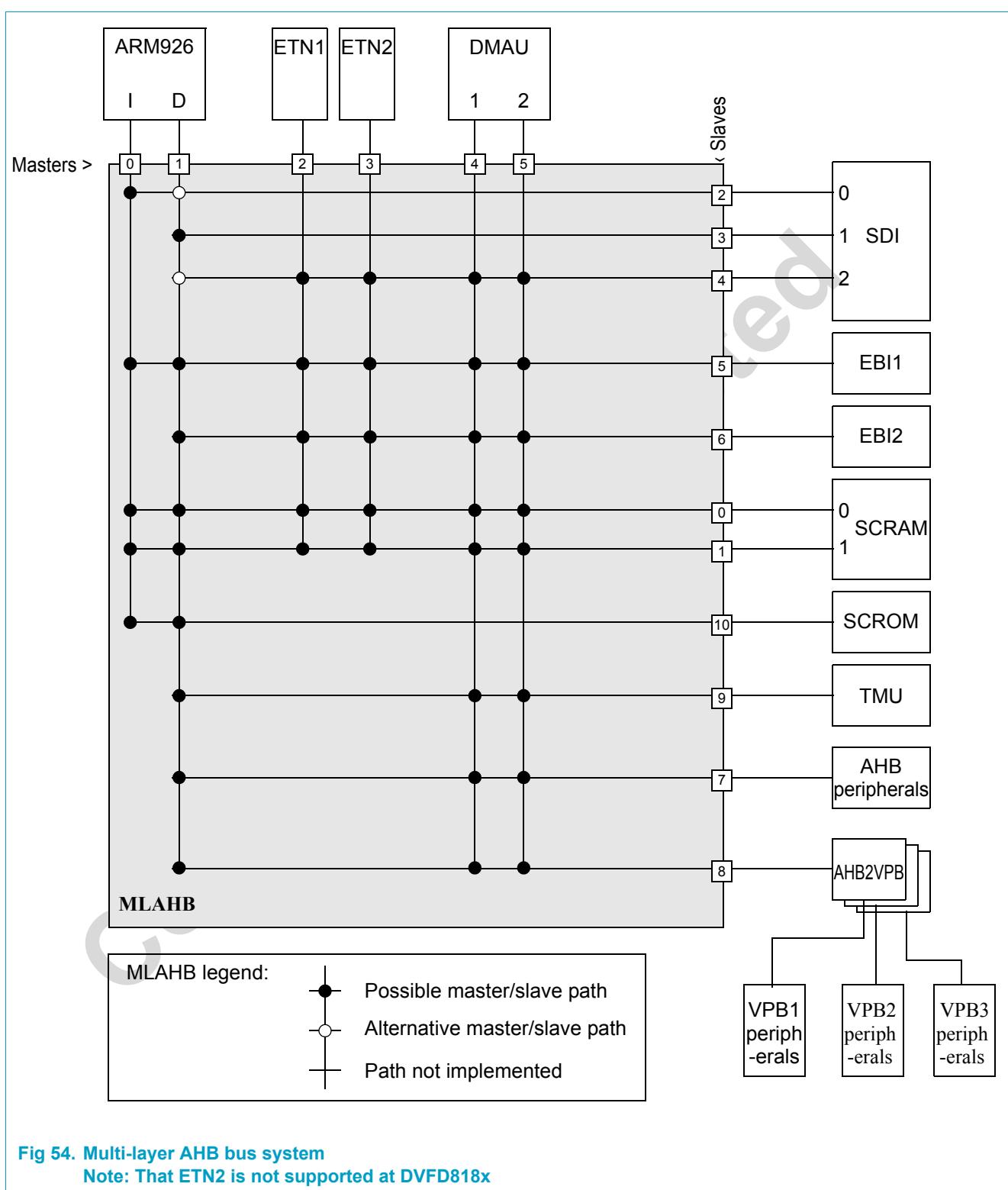
Table 155: Multi-layer AHB bus configuration ^[1]

Masters -> Slaves		ARM 926-I	ARM 926-D	ETN 1	ETN 2 ^[3]	DMAU 1	DMAU 2
	number	0	1	2	3	4	5
SCRAM	bank #0	0	X	X	X	X	X
	bank #1	1	X	X	X	X	X
SDI	port #0	2	X	(X) ^[2]			
	port #1	3		X ^[2]			
EBI1	port #2	4	(X) ^[2]	X	X	X	X
		5	X	X	X	X	X
EBI2		6		X	X	X	X
AHB slaves		7		X		X	X
VPB slaves		8		X		X	X
TMU		9		X		X	X
SCROM		10	X	X			

[1] Only the datapaths with X are implemented. For example, ETN1 can access SDI port #2 but can't access SDI port #0

[2] ARM926-D use port #1 by default when using the SDRAM base address, however, for debug, ARM926-D access can be forced on SDRAM port #0 or port #2 by using other address (More information can be found on [Table 157](#))

[3] Not supported at DVFD8185 and DVFD8187

**Fig 54. Multi-layer AHB bus system**

Note: That ETN2 is not supported at DVFD818x

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 156:HPROT values

Master		HPROT[3]	HPROT[2]	HPROT[1]	HPROT[0]
Name	Number	0 = not cacheable 1 = cacheable	0 = not bufferable 1 = bufferable	0 = user mode 1 = privileged mode	0 = Instruction fetch 1 = Data access
ARM926	I	0	Automatic [1]		
	D	1	Automatic [1]		
ETN1		2	Programmable [2]		1
DMAU	1	4	Programmable [2][3]		1
	2	5	Programmable [2][3]		1

[1] Automatic means depending on the kind of the transfer requested by the ARM926EJ-S, some combinations may not be possible, check ARM926EJ-S documentation for details

[2] Programmable by register setting in the master

[3] Each DMAU channel has its own hprot value

9.2.2 Clocking

The ARM926EJ-S core, including caches, can run at up to 208 MHz. The Multi-layer AHB bus is synchronous with the processor and can run up to 104 MHz.

The frequency of the SC processor can be set by increment depending on the clock crystal used.

On-chip memories connected to the Multi-layer AHB bus run at the same frequency as the bus.

VPB1 runs at the same frequency as the Multi-layer AHB bus.

VPB2 and VPB3 runs at a fixed (independent from the Multi-layer AHB bus) programmable frequency.

9.3 SC Memory Configuration

9.3.1 Overview

The DVFD818x Family features the following on-chip memory.

- SCRAM - internal 192 kbytes (1.5 Mbit total) of SRAM, 32-bit wide, 0/1 wait-state for respectively sequential/non-sequential access, running at Multi-layer AHB bus speed
- SCROM - internal 128 kbytes (1 Mbits) of ROM, 32-bit wide, 1 wait-state, running at AHB speed

It is possible to emulate a TCM by locking-down part of the ARM926EJ-S caches.
Refer to [12.] for more details.

The DVFD818x Family also comprises two off-chip memory ports.

- FMP - Fast Memory Port (reduced function)
featuring connection to the EBI1 with up to 6 static memories
- HMP - Hybrid Memory Port (reduced function)
featuring connection to the EBI2 with up to 6 memory mapped devices or to the SDI for connection to SDRAM

9.3.2 SC memory map

See Table 157 on next page.

Table 157:SC memory map

Start Address	End Address	Size (bytes)	Memory names	Comment
0xFFFFE 0000	0xFFFFF FFFF	128 k	SCROM ^[1] , not mirrored	128 kbytes (32k x 32-bit)
0xFF00 0000	0xFFFFD FFFF	16256 k	Reserved for SCROM extension	
0xC300 0000	0xC30F FFFF	1 M	TMU ^[3]	
0xC220 0000	0xC22F FFFF	1 M	VPB3 peripherals	refer to Table 158
0xC210 0000	0xC21F FFFF	1 M	VPB2 peripherals	refer to Table 158
0xC200 0000	0xC20F FFFF	1 M	VPB1 peripherals	refer to Table 158
0xC1F0 0000	0xC1FF FFFF	1 M	reserved	
0xC1E0 0000	0xC1EF FFFF	1 M	reserved	
0xC1D0 0000	0xC1DF FFFF	1 M	reserved	
0xC1C0 0000	0xC1CF FFFF	1 M	reserved	
0xC1B0 0000	0xC1BF FFFF	1 M	reserved	
0xC1A0 0000	0xC1AF FFFF	1 M	reserved	
0xC190 0000	0xC19F FFFF	1 M	reserved	
0xC180 0000	0xC18F FFFF	1 M	BMP registers ^[4]	
0xC160 0000	0xC16F FFFF	1 M	ETN1 registers	
0xC150 0000	0xC15F FFFF	1 M	DRT registers	
0xC140 0000	0xC14F FFFF	1 M	ETB RAM	See 9.15 for details
0xC130 0000	0xC13F FFFF	1 M	ETB registers	
0xC120 0000	0xC12F FFFF	1 M	SDI registers	
0xC110 0000	0xC11F FFFF	1 M	INTC registers	
0xC100 0000	0xC10F FFFF	1 M	DMAU registers	
0xC038 0000	0xC03F FFFF	512 k	EBI2 registers	
0xC018 0000	0xC01F FFFF	512 k	off-chip static memory, EBI2, CS3	only first 128 kbyte are available
0xC010 0000	0xC017 FFFF	512 k	off-chip static memory, EBI2, CS2	only first 128 kbyte are available
0xC008 0000	0xC00F FFFF	512 k	off-chip static memory, EBI2, CS1	only first 128 kbyte are available
0xC000 0000	0xC007 FFFF	512 k	off-chip static memory, EBI2, CS0	only first 128 kbyte are available
0xBC00 0000	0xBFFF FFFF	64 M	EBI1 registers	
0xB000 0000	0xB3FF FFFF	64 M	off-chip static memory, EBI1, CS3	
0x9000 0000	0x9FFF FFFF	256 M	off-chip static memory, EBI1, CS1	
0x8000 0000	0x8FFF FFFF	256 M	off-chip static memory, EBI1, CS0	
ARM926-I ARM926-D Others masters				
0x6000 0000 ^[2]	0x7FFF FFFF	512 M	- SDI port #2	- for debug only, only first
0x4000 0000 ^[2]	0x5FFF FFFF	512 M	- SDI port #0	- 256 ^[5] Mbyte are available
0x2000 0000 ^[2]	0x3FFF FFFF	512 M	SDI port #0 SDI port #1	SDI port #2 only first 256 ^[5] Mbyte are available
0x0002 0000	0x0002 FFFF	64 k	SCRAM bank #1	64 kbyte (16k x 32-bit)
0x0000 0000	0x0001 FFFF	128 k	SCRAM bank #0	128 kbyte (32k x 32-bit)

[1] The SCROM contain the boot strap loader. Boot address is 0xFFFF 0000. To store user code, SCRAM or external memory is used.

[2] See also [Figure 54](#). For debugging purposes, the ARM926-D can also access SDRAM through SDI port #0 (resp. #2) by using address range 0x4000 0000 to 0x5FFF FFFF (resp. 0x6000 0000 to 0x7FFF FFFF).

[3] The TMU supports only the lower 512Kbyte of the address range. Any access to the upper 512kByte will be mirrored to the lower area.

[4] Not supported at DVFD8187

[5] Only first 64MByte are supported

9.3.3 Boot

The ARM926EJ-S is configured in such a way that, after reset, exception vectors are located from 0xFFFF 0000 to 0xFFFF 001C (signal VINITHI = 1). This can be changed later on by reprogramming bit V = bit[13] of CP15, reg1 (see [\[12.\]](#)).

Therefore, after reset, the ARM core starts executing code from address 0xFFFF 0000 which corresponds to address in the SCROM.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.3.4 SC register map

ABORT exception on the AHB [1] is triggered under the following conditions:

- Access to an address located outside the decoded Multi-layer AHB bus area [2]
- Access from a master to a slave for which there is no physical connection in the matrix [2]
- Access in un-aligned data (i.e. half-word at address other than 0xxxxxxxx0 or word at address other than 0xxxxxxxx00) [2]
- Access in byte in TMU [3]
- Write access to SCROM [3]
- Access to a protected area for SCROM [4]

ABORT exception can also be generated by the ARM926EJ-S MMU for its accesses. This accesses are not issued on the Multi-layer AHB bus.

Note that no data ABORT is triggered on undefined or illegal VPB peripheral access. The Multi-layer AHB bus supports the following accesses.

- Sub-word register accesses are not supported for read and write operations (e.g. byte or half word access)
- All registers connected to Multi-layer AHB bus are aligned on 32-bits boundaries (i.e. 0x...0, 0x...4, 0x...8 or 0x...C).

The VPB supports the following accesses.

- The address assigned to each VPB peripheral register is selected according to the following rules:
 - an address space of 4 kbytes is reserved for each peripheral
 - All registers connected to VPB bus are aligned on 32-bits boundaries (i.e. 0x...0, 0x...4, 0x...8 or 0x...C).
- Sub-size accesses are not supported for write operations in register (e.g. byte access on half word register). Writing sub-size has unpredictable result, except for data transfer where sub-size is handled by the peripheral itself (**spi_dat**). This exception is needed to allow DMA data storing optimization (e.g. SPI data width can be configurated as less or equal than 8 bits since data register is 16 bits; in this case, byte data write is allowed)

Note: sub-size means ‘size lower than effective register size’ - sub-word means ‘size lower than 32-bit size’.

Peripherals are connected either to Multi-layer AHB bus (see [Table 157](#)) or to one of the 3 peripheral buses: VPB1, VPB2 and VPB3 (see [Table 158](#)).

1. ABORT exception can also be generated by the ARM926EJ-S MMU for ARM-D and ARM-I accesses. No AHB access is issued
2. Protection is done globally in the Multi-layer AHB bus
3. ABORT is generated locally by the AHB slave
4. Protection is done locally in SCROM protection unit

Table 158:VPB peripherals registers mapping

VPB bus	Peripheral	Base address	
VPB1	DAIF	0xC200 0000	1
	IIC	0xC200 1000	2
	SPI1	0xC200 2000	3
	UART1	0xC200 4000	4
	UART2	0xC200 5000	5
	IIS	0xC200 6000	6
	USB	0xC200 9000	7
	reserved	0xC200 A000	8
	reserved	0xC200 B000	9
	IPINT	0xC200 C000	10
	ADPCM	0xC200 D000	11
	AMU1 to AMU6, ARB, SCROM, ETB, HMP configuration registers	0xC200 E000	12
	reserved	0xC200 F000	13
	reserved	0xC201 0000	14
	reserved	0xC201 1000	15
	reserved	0xC201 2000	16
	FIR	0xC201 3000	17
	reserved	0xC201 4000	18
VPB2	reserved	0xC210 0000	19
	reserved	0xC210 1000	20
	SCTU1	0xC210 2000	21
	SCTU2	0xC210 3000	22
	GPIOA to GPIOC	0xC210 4000	23
	EXTINT	0xC210 5000	24
	KBS	0xC210 6000	25
	reserved	0xC210 7000	26
	PWM2	0xC210 9000	27
	PWM3	0xC210 A000	28
VPB3	CGU	0xC220 0000	29
	TBU	0xC220 1000	30
	PDCU	0xC220 2000	31
	WDRU	0xC220 3000	32
	SCON	0xC220 4000	33
	reserved	0xC220 5000	34
	reserved	0xC220 6000	35
	reserved	0xC220 7000	36
	reserved	0xC220 8000	37

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals

Address	Name	R/W	Size [1]	Description
AHB Peripherals				
EBI1				
0xBC00 0000	ebi1_maincfg0	R/W	W	main configuration register for chip select 0
0xBC00 0004	ebi1_readcfg0	R/W	W	read timing configuration register for chip select 0
0xBC00 0008	ebi1_writecfg0	R/W	W	write timing configuration register for chip select 0
0xBC00 000C	ebi1_burstcfg0	R/W	W	burst/page configuration register for chip select 0
0xBC00 0020	ebi1_maincfg1	R/W	W	main configuration register for chip select 1
0xBC00 0024	ebi1_readcfg1	R/W	W	read timing configuration register for chip select 1
0xBC00 0028	ebi1_writecfg1	R/W	W	write timing configuration register for chip select 1
0xBC00 002C	ebi1_burstcfg1	R/W	W	burst/page configuration register for chip select 1
0xBC00 0040	ebi1_maincfg2	R/W	W	main configuration register for chip select 2
0xBC00 0044	ebi1_readcfg2	R/W	W	read timing configuration register for chip select 2
0xBC00 0048	ebi1_writecfg2	R/W	W	write timing configuration register for chip select 2
0xBC00 004C	ebi1_burstcfg2	R/W	W	burst/page configuration register for chip select 2
0xBC00 0060	ebi1_maincfg3	R/W	W	main configuration register for chip select 3
0xBC00 0064	ebi1_readcfg3	R/W	W	read timing configuration register for chip select 3
0xBC00 0068	ebi1_writecfg3	R/W	W	write timing configuration register for chip select 3
0xBC00 006C	ebi1_burstcfg3	R/W	W	burst/page configuration register for chip select 3
0xBC00 0080	ebi1_maincfg4	R/W	W	main configuration register for chip select 4
0xBC00 0084	ebi1_readcfg4	R/W	W	read timing configuration register for chip select 4
0xBC00 0088	ebi1_writecfg4	R/W	W	write timing configuration register for chip select 4
0xBC00 008C	ebi1_burstcfg4	R/W	W	burst/page configuration register for chip select 4
0xBC00 00A0	ebi1_maincfg5	R/W	W	main configuration register for chip select 5
0xBC00 00A4	ebi1_readcfg5	R/W	W	read timing configuration register for chip select 5
0xBC00 00A8	ebi1_writecfg5	R/W	W	write timing configuration register for chip select 5
0xBC00 00AC	ebi1_burstcfg5	R/W	W	burst/page configuration register for chip select 5
0xBC00 0100	ebi1_always_clkcfg	R/W	W	EBI always clock configuration register
0xBC00 0300	ebi1_apb_err_cfg	R/W	W	APB Error configuration register
0xBC00 0304	ebi1_apb_latency_cfg	R/W	W	APB Read data latency configuration register
0xBC00 0FF0	ebi1_sw_resetcfg	R/W	W	EBI SW reset register
0xBC00 0FF4	ebi1_power_downcfg	R/W	W	EBI power down configuration register
0xBC00 0FFC	ebi1_ip_id	R	W	IP identification register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
EBI2				
0xC038 0000	ebi2_maincfg0	R/W	W	main configuration register for chip select 0
0xC038 0004	ebi2_readcfg0	R/W	W	read timing configuration register for chip select 0
0xC038 0008	ebi2_writecfg0	R/W	W	write timing configuration register for chip select 0
0xC038 000C	ebi2_burstcfg0	R/W	W	burst/page configuration register for chip select 0
0xC038 0020	ebi2_maincfg1	R/W	W	main configuration register for chip select 1
0xC038 0024	ebi2_readcfg1	R/W	W	read timing configuration register for chip select 1
0xC038 0028	ebi2_writecfg1	R/W	W	write timing configuration register for chip select 1
0xC038 002C	ebi2_burstcfg1	R/W	W	burst/page configuration register for chip select 1
0xC038 0040	ebi2_maincfg2	R/W	W	main configuration register for chip select 2
0xC038 0044	ebi2_readcfg2	R/W	W	read timing configuration register for chip select 2
0xC038 0048	ebi2_writecfg2	R/W	W	write timing configuration register for chip select 2
0xC038 004C	ebi2_burstcfg2	R/W	W	burst/page configuration register for chip select 2
0xC038 0060	ebi2_maincfg3	R/W	W	main configuration register for chip select 3
0xC038 0064	ebi2_readcfg3	R/W	W	read timing configuration register for chip select 3
0xC038 0068	ebi2_writecfg3	R/W	W	write timing configuration register for chip select 3
0xC038 006C	ebi2_burstcfg3	R/W	W	burst/page configuration register for chip select 3
0xC038 0080	ebi2_maincfg4	R/W	W	main configuration register for chip select 4
0xC038 0084	ebi2_readcfg4	R/W	W	read timing configuration register for chip select 4
0xC038 0088	ebi2_writecfg4	R/W	W	write timing configuration register for chip select 4
0xC038 008C	ebi2_burstcfg4	R/W	W	burst/page configuration register for chip select 4
0xC038 00A0	ebi2_maincfg5	R/W	W	main configuration register for chip select 5
0xC038 00A4	ebi2_readcfg5	R/W	W	read timing configuration register for chip select 5
0xC038 00A8	ebi2_writecfg5	R/W	W	write timing configuration register for chip select 5
0xC038 00AC	ebi2_burstcfg5	R/W	W	burst/page configuration register for chip select 5
0xC038 0100	ebi2_always_clkcfg	R/W	W	EBI always clock configuration register
0xC038 0300	ebi2_apb_err_cfg	R/W	W	APB Error configuration register
0xC038 0304	ebi2_apb_latency_cfg	R/W	W	APB Read data latency configuration register
0xC038 OFF0	ebi2_sw_resetcfg	R/W	W	EBI SW reset register
0xC038 OFF4	ebi2_power_downcfg	R/W	W	EBI power down configuration register
0xC038 OFFC	ebi2_ip_id	R	W	IP identification register
DMAU				
0xC100 0000	dmau_int_status	R	W	interrupt status
0xC100 0004	dmau_int_tc_status	R	W	terminal count interrupt status
0xC100 0008	dmau_int_tc_clear	W	W	terminal count interrupt clear
0xC100 000C	dmau_int_error_status	R	W	error interrupt status
0xC100 0010	dmau_int_error_clr	W	W	error interrupt clear
0xC100 0014	dmau_raw_int_tc_status	R	W	terminal count interrupt status prior masking
0xC100 0018	dmau_raw_int_error_status	R	W	error interrupt status prior masking
0xC100 001C	dmau_enblid_chns	R	W	enabled channels

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC100 0020	dmau_soft_breq	R/S	W	software burst DMA requests	1
0xC100 0024	dmau_soft_sreq	R/S	W	software single DMA requests	2
0xC100 0030	dmau_configuration	R/W	W	global configuration register	3
0xC100 0034	dmau_sync	R/W	W	synchronization control register	4
0xC100 0100	dmau_c0_src_addr	R/W	W	source address for channel 0	5
0xC100 0104	dmau_c0_dest_addr	R/W	W	destination address for channel 0	6
0xC100 0108	dmau_c0_lli	R/W	W	linked list address for channel 0	7
0xC100 010C	dmau_c0_control	R/W	W	control for channel 0	8
0xC100 0110	dmau_c0_configuration	R/W	W	configuration for channel 0	9
0xC100 0120	dmau_c1_src_addr	R/W	W	source address for channel 1	10
0xC100 0124	dmau_c1_dest_addr	R/W	W	destination address for channel 1	11
0xC100 0128	dmau_c1_lli	R/W	W	linked list address for channel 1	12
0xC100 012C	dmau_c1_control	R/W	W	control for channel 1	13
0xC100 0130	dmau_c1_configuration	R/W	W	configuration for channel 1	14
0xC100 0140	dmau_c2_src_addr	R/W	W	source address for channel 2	15
0xC100 0144	dmau_c2_dest_addr	R/W	W	destination address for channel 2	16
0xC100 0148	dmau_c2_lli	R/W	W	linked list address for channel 2	17
0xC100 014C	dmau_c2_control	R/W	W	control for channel 2	18
0xC100 0150	dmau_c2_configuration	R/W	W	configuration for channel 2	19
0xC100 0160	dmau_c3_src_addr	R/W	W	source address for channel 3	20
0xC100 0164	dmau_c3_dest_addr	R/W	W	destination address for channel 3	21
0xC100 0168	dmau_c3_lli	R/W	W	linked list address for channel 3	22
0xC100 016C	dmau_c3_control	R/W	W	control for channel 3	23
0xC100 0170	dmau_c3_configuration	R/W	W	configuration for channel 3	24
0xC100 0180	dmau_c4_src_addr	R/W	W	source address for channel 4	25
0xC100 0184	dmau_c4_dest_addr	R/W	W	destination address for channel 4	26
0xC100 0188	dmau_c4_lli	R/W	W	linked list address for channel 4	27
0xC100 018C	dmau_c4_control	R/W	W	control for channel 4	28
0xC100 0190	dmau_c4_configuration	R/W	W	configuration for channel 4	29
0xC100 01A0	dmau_c5_src_addr	R/W	W	source address for channel 5	30
0xC100 01A4	dmau_c5_dest_addr	R/W	W	destination address for channel 5	31
0xC100 01A8	dmau_c5_lli	R/W	W	linked list address for channel 5	32
0xC100 01AC	dmau_c5_control	R/W	W	control for channel 5	33
0xC100 01B0	dmau_c5_configuration	R/W	W	configuration for channel 5	34
0xC100 01C0	dmau_c6_src_addr	R/W	W	source address for channel 6	35
0xC100 01C4	dmau_c6_dest_addr	R/W	W	destination address for channel 6	36
0xC100 01C8	dmau_c6_lli	R/W	W	linked list address for channel 6	37
0xC100 01CC	dmau_c6_control	R/W	W	control for channel 6	38
0xC100 01D0	dmau_c6_configuration	R/W	W	configuration for channel 6	39
0xC100 01E0	dmau_c7_src_addr	R/W	W	source address for channel 7	40

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC100 01E4	dmau_c7_dest_addr	R/W	W	destination address for channel 7	1
0xC100 01E8	dmau_c7_lll	R/W	W	linked list address for channel 7	2
0xC100 01EC	dmau_c7_control	R/W	W	control for channel 7	3
0xC100 01F0	dmau_c7_configuration	R/W	W	configuration for channel 7	4
0xC100 0200	dmau_c8_src_addr	R/W	W	source address for channel 8	5
0xC100 0204	dmau_c8_dest_addr	R/W	W	destination address for channel 8	6
0xC100 0208	dmau_c8_lll	R/W	W	linked list address for channel 8	7
0xC100 020C	dmau_c8_control	R/W	W	control for channel 8	8
0xC100 0210	dmau_c8_configuration	R/W	W	configuration for channel 8	9
0xC100 0220	dmau_c9_src_addr	R/W	W	source address for channel 9	10
0xC100 0224	dmau_c9_dest_addr	R/W	W	destination address for channel 9	11
0xC100 0228	dmau_c9_lll	R/W	W	linked list address for channel 9	12
0xC100 022C	dmau_c9_control	R/W	W	control for channel 9	13
0xC100 0230	dmau_c9_configuration	R/W	W	configuration for channel 9	14
0xC100 0240	dmau_c10_src_addr	R/W	W	source address for channel 10	15
0xC100 0244	dmau_c10_dest_addr	R/W	W	destination address for channel 10	16
0xC100 0248	dmau_c10_lll	R/W	W	linked list address for channel 10	17
0xC100 024C	dmau_c10_control	R/W	W	control for channel 10	18
0xC100 0250	dmau_c10_configuration	R/W	W	configuration for channel 10	19
0xC100 0260	dmau_c11_src_addr	R/W	W	source address for channel 11	20
0xC100 0264	dmau_c11_dest_addr	R/W	W	destination address for channel 11	21
0xC100 0268	dmau_c11_lll	R/W	W	linked list address for channel 11	22
0xC100 026C	dmau_c11_control	R/W	W	control for channel 11	23
0xC100 0270	dmau_c11_configuration	R/W	W	configuration for channel 11	24
INTC					25
0xC110 0000	intc_priomask_irq	R/W	B	interrupt IRQ target priority threshold register	26
0xC110 0004	intc_priomask_fiq	R/W	B	interrupt FIQ target priority threshold register	27
0xC110 0100	intc_vector_irq	R	W	interrupt IRQ target vector register	28
0xC110 0104	intc_vector_fiq	R	W	interrupt FIQ target vector register	29
0xC110 0200	intc_pending_1	R	W	status of interrupt requests 31...1 register	30
0xC110 0204	intc_pending_2	R	W	status of interrupt requests 63...32 register	31
0xC110 0208	intc_pending_3	R	W	status of interrupt requests 66...64 register	32
0xC110 0300	intc_features	R	W	interrupt controller features register	33
0xC110 0404	intc_request1	R/W	W	interrupt request configuration	34
0xC110 0408	intc_request2	R/W	W	interrupt request configuration	35
0xC110 040C	intc_request3	R/W	W	interrupt request configuration	36
0xC110 0410	intc_request4	R/W	W	interrupt request configuration	37
0xC110 0414	intc_request5	R/W	W	interrupt request configuration	38
0xC110 0418	intc_request6	R/W	W	interrupt request configuration	39
0xC110 041C	intc_request7	R/W	W	interrupt request configuration	40

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC110 0420	intc_request8	R/W	W	interrupt request configuration	1
0xC110 0424	intc_request9	R/W	W	interrupt request configuration	2
0xC110 0428	intc_request10	R/W	W	interrupt request configuration	3
0xC110 042C	intc_request11	R/W	W	interrupt request configuration	4
0xC110 0430	intc_request12	R/W	W	interrupt request configuration	5
0xC110 0434	intc_request13	R/W	W	interrupt request configuration	6
0xC110 0438	intc_request14	R/W	W	interrupt request configuration	7
0xC110 043C	intc_request15	R/W	W	interrupt request configuration	8
0xC110 0440	intc_request16	R/W	W	interrupt request configuration	9
0xC110 0444	intc_request17	R/W	W	interrupt request configuration	10
0xC110 0448	intc_request18	R/W	W	interrupt request configuration	11
0xC110 044C	intc_request19	R/W	W	interrupt request configuration	12
0xC110 0450	intc_request20	R/W	W	interrupt request configuration	13
0xC110 0454	intc_request21	R/W	W	interrupt request configuration	14
0xC110 0458	intc_request22	R/W	W	interrupt request configuration	15
0xC110 045C	intc_request23	R/W	W	interrupt request configuration	16
0xC110 0460	intc_request24	R/W	W	interrupt request configuration	17
0xC110 0464	intc_request25	R/W	W	interrupt request configuration	18
0xC110 0468	intc_request26	R/W	W	interrupt request configuration	19
0xC110 046C	intc_request27	R/W	W	interrupt request configuration	20
0xC110 0470	intc_request28	R/W	W	interrupt request configuration	21
0xC110 0474	intc_request29	R/W	W	interrupt request configuration	22
0xC110 0478	intc_request30	R/W	W	interrupt request configuration	23
0xC110 047C	intc_request31	R/W	W	interrupt request configuration	24
0xC110 0480	intc_request32	R/W	W	interrupt request configuration	25
0xC110 0484	intc_request33	R/W	W	interrupt request configuration	26
0xC110 0488	intc_request34	R/W	W	interrupt request configuration	27
0xC110 048C	intc_request35	R/W	W	interrupt request configuration	28
0xC110 0490	intc_request36	R/W	W	interrupt request configuration	29
0xC110 0494	intc_request37	R/W	W	interrupt request configuration	30
0xC110 0498	intc_request38	R/W	W	interrupt request configuration	31
0xC110 049C	intc_request39	R/W	W	interrupt request configuration	32
0xC110 04A0	intc_request40	R/W	W	interrupt request configuration	33
0xC110 04A4	intc_request41	R/W	W	interrupt request configuration	34
0xC110 04A8	intc_request42	R/W	W	interrupt request configuration	35
0xC110 04AC	intc_request43	R/W	W	interrupt request configuration	36
0xC110 04B0	intc_request44	R/W	W	interrupt request configuration	37
0xC110 04B4	intc_request45	R/W	W	interrupt request configuration	38
0xC110 04B8	intc_request46	R/W	W	interrupt request configuration	39
0xC110 04BC	intc_request47	R/W	W	interrupt request configuration	40

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC110 04C0	intc_request48	R/W	W	interrupt request configuration	1
0xC110 04C4	intc_request49	R/W	W	interrupt request configuration	2
0xC110 04C8	intc_request50	R/W	W	interrupt request configuration	3
0xC110 04CC	intc_request51	R/W	W	interrupt request configuration	4
0xC110 04D0	intc_request52	R/W	W	interrupt request configuration	5
0xC110 04D4	intc_request53	R/W	W	interrupt request configuration	6
0xC110 04D8	intc_request54	R/W	W	interrupt request configuration	7
0xC110 04DC	intc_request55	R/W	W	interrupt request configuration	8
0xC110 04E0	intc_request56	R/W	W	interrupt request configuration	9
0xC110 04E4	intc_request57	R/W	W	interrupt request configuration	10
0xC110 04E8	intc_request58	R/W	W	interrupt request configuration	11
0xC110 04EC	intc_request59	R/W	W	interrupt request configuration	12
0xC110 04F0	intc_request60	R/W	W	interrupt request configuration	13
0xC110 04F4	intc_request61	R/W	W	interrupt request configuration	14
0xC110 04F8	intc_request62	R/W	W	interrupt request configuration	15
0xC110 04FC	intc_request63	R/W	W	interrupt request configuration	16
0xC110 0500	intc_request64	R/W	W	interrupt request configuration	17
0xC110 0504	intc_request65	R/W	W	interrupt request configuration	18
0xC110 0508	intc_request66	R/W	W	interrupt request configuration	19
0xC110 0600	intc_edge_detect_sync	R	W	status of synchronous edge detection	20
0xC110 0604	intc_edge_detect_async	R	W	status of asynchronous edge detection	21
0xC110 0608	intc_edge_clear	W	W	edge detection clear register	22
0xC110 060C	intc_edge_ctrl0	R/W	W	edge detection control register 0	23
0xC110 0610	intc_edge_ctrl1	R/W	W	edge detection control register 1	24
0xC110 0FFC	intc_mod_id	R	W	interrupt controller module ID	25

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
SDI				
0xC120 0000	sdi_cfg1	R/W	W	SDRAM controller configuration 1
0xC120 0004	sdi_pwr_ctrl	R/W	W	Power savings parameters
0xC120 0008	sdi_rfrsh1	R/W	W	Refresh parameters 1
0xC120 000C	sdi_rfrsh2	R/W	W	Refresh parameters 2
0xC120 0010	sdi_tim1	R/W	W	Timings parameters 1
0xC120 0014	sdi_tim2	R/W	W	Timings parameters 2
0xC120 0018	sdi_mode	R/W	W	Mode
0xC120 001C	sdi_ext_mode	R/W	W	Extended mode
0xC120 0020	sdi_cfg2	R/W	W	SDRAM controller configuration 2
0xC120 0100	sdi_sched0	R/W	W	Data port #0 scheduler
0xC120 0104	sdi_sched1	R/W	W	Data port #1 scheduler
0xC120 0108	sdi_sched2	R/W	W	Data port #2 scheduler
0xC120 0300	sdi_wtag0	R/W	W	Data port #0 write tag enable
0xC120 0304	sdi_wtag1	R/W	W	Data port #1 write tag enable
0xC120 0308	sdi_wtag2	R/W	W	Data port #2 write tag enable
0xC120 0310	sdi_tam	R/W	W	Tag address mask
0xC120 0400	sdi_sw_ctrl	R/W	W	Software control register
0xC120 0800	sdi_stat	R	W	Status register
0xC120 0FFC	sdi_ip_id	R	W	IP identification
ETB registers				
0xC130 0000	etb_id	R	W	Identification register
0xC130 0004	etb_ram_depth	R	W	RAM depth register
0xC130 0008	etb_ram_width	R	W	RAM width register
0xC130 000C	etb_status	R	W	Status register
0xC130 0010	etb_ram_data	R	W	RAM data register
0xC130 0014	etb_ram_read_ptr	R/W	W	RAM read pointer register
0xC130 0018	etb_ram_write_ptr	R/W	W	RAM write pointer register
0xC130 001C	etb_trig_cnt	R/W	W	Trigger counter register
0xC130 0020	etb_ctrl	R/W	W	Control register
DRT				
0xC150 0000	drt_global	R/W	W	DRT global control register
0xC150 0004	drt_cdc1_di12	R	W	CODEC1 Inbound buffer storing samples 1 and 2 written by the digital decimation filter to be read by the system controller
0xC150 0008	drt_cdc1_di34	R	W	CODEC1 Inbound buffer storing samples 3 and 4 written by the digital decimation filter to be read by the system controller
0xC150 000C	drt_cdc1_di5	R	W	CODEC1 Inbound buffer storing sample 5 written by the digital decimation filter to be read by the system controller

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
0xC150 0010	drt_cdc1_do12	R/W	W	CODEC1 Outbound buffer storing samples 1 and 2 written by the system controller to be read by the digital noise shaper
0xC150 0014	drt_cdc1_do34	R/W	W	CODEC1 Outbound buffer storing samples 3 and 4 written by the system controller to be read by the digital noise shaper
0xC150 0018	drt_cdc1_do5	R/W	W	CODEC1 Outbound buffer storing sample 5 written by the system controller to be read by the digital noise shaper
0xC150 001C	drt_cdc2_di12	R	W	CODEC2 Inbound buffer storing samples 1 and 2 written by the digital decimation filter to be read by the system controller
0xC150 0020	drt_cdc2_di34	R	W	CODEC2 Inbound buffer storing samples 3 and 4 written by the digital decimation filter to be read by the system controller
0xC150 0024	drt_cdc2_di5	R	W	CODEC2 Inbound buffer storing sample 5 written by the digital decimation filter to be read by the system controller
0xC150 0028	drt_cdc2_do12	R/W	W	CODEC2 Outbound buffer storing samples 1 and 2 written by the system controller to be read by the digital noise shaper
0xC150 002C	drt_cdc2_do34	R/W	W	CODEC2 Outbound buffer storing samples 3 and 4 written by the system controller to be read by the digital noise shaper
0xC150 0030	drt_cdc2_do5	R/W	W	CODEC2 Outbound buffer storing sample 5 written by the system controller to be read by the digital noise shaper
0xC150 0034	drt_con	R/W	W	DRT Control Register
0xC150 0038	drt_codtr	R/W	W	CODEC Test Register defining Digital Dithering functions
ETN1				
0xC160 0000	etn1_mac1	R/W	W	MAC configuration register 1
0xC160 0004	etn1_mac2	R/W	W	MAC configuration register 2
0xC160 0008	etn1_ipgt	R/W	W	Back-to-Back Inter-Packet-Gap register
0xC160 000C	etn1_ipgr	R/W	W	Non Back-to-Back Inter-Packet-Gap register
0xC160 0010	etn1_clrt	R/W	W	Collision window / Retry register
0xC160 0014	etn1_maxf	R/W	W	Maximum frame register
0xC160 0018	etn1_supp	R/W	W	PHY support register
0xC160 001C	etn1_test	R/W	W	Test register
0xC160 0020	etn1_mcfg	R/W	W	MII Mgmt configuration register
0xC160 0024	etn1_mcmd	R/W	W	MII Mgmt command register
0xC160 0028	etn1_madr	R/W	W	MII Mgmt address register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC160 002C	etn1_mwtd	R	W	MII Mgmt write data register	1
0xC160 0030	etn1_mrdd	R	W	MII Mgmt read data register	2
0xC160 0034	etn1_mind	R	W	MII Mgmt indicators register	3
0xC160 0040	etn1_sa0	R/W	W	Station address 0 register	4
0xC160 0044	etn1_sa1	R/W	W	Station address 1 register	5
0xC160 0048	etn1_sa2	R/W	W	Station address 2 register	6
0xC160 0100	etn1_command	R/W	W	Command register	7
0xC160 0104	etn1_status	R	W	Status register	8
0xC160 0108	etn1_rxdescriptor	R/W	W	Receive descriptor base address register	9
0xC160 010C	etn1_rxstatus	R/W	W	Receive status base address register	10
0xC160 0110	etn1_rxdescriptornumber	R/W	W	Receive number of descriptors register	11
0xC160 0114	etn1_rxproduceindex	R	W	Receive produce index register	12
0xC160 0118	etn1_rxconsumeindex	R/W	W	Receive consume index register	13
0xC160 011C	etn1_txdescriptor	R/W	W	Non real-time transmit descriptor base address register	14
0xC160 0120	etn1_txstatus	R/W	W	Non real-time transmit status base address register	15
0xC160 0124	etn1_txdescriptornumber	R/W	W	Non real-time transmit number of descriptors register	16
0xC160 0128	etn1_txproduceindex	R/W	W	Non real-time transmit produce index register	17
0xC160 012C	etn1_txconsumeindex	R	W	Non real-time transmit consume index register	18
0xC160 0130	etn1_txrtdescriptor	R/W	W	Real-time transmit descriptor base address register	19
0xC160 0134	etn1_txrtstatus	R/W	W	Real-time transmit status base address register	20
0xC160 0138	etn1_txrtdescriptornumber	R/W	W	Real-time transmit number of descriptors register	21
0xC160 013C	etn1_txrtproduceindex	R/W	W	Real-time transmit produce index register	22
0xC160 0140	etn1_txrtconsumeindex	R	W	Real-time transmit consume index register	23
0xC160 0148	etn1_qostimeout	R/W	W	Transmit quality of service time-out register	24
0xC160 0158	etn1_tsv0	R	W	Transmit status vector 0 register	25
0xC160 015C	etn1_tsv1	R	W	Transmit status vector 1 register	26
0xC160 0160	etn1_rsv	R	W	Receive status vector register	27
0xC160 0170	etn1_flowcontrolcounter	R/W	W	Flow control counter register	28
0xC160 0174	etn1_flowcontrolstatus	R	W	Flow control status register	29
0xC160 0200	etn1_rxfilterctrl	R/W	W	Receive filter control register	30
0xC160 0204	etn1_rxfilterwolstatus	R	W	Receive filter WoL status register	31
0xC160 0208	etn1_rxfilterwolclear	W	W	Receive filter WoL clear register	32
0xC160 0210	etn1_hashfilterl	R/W	W	Hash filter LSBs register	33
0xC160 0214	etn1_hashfilterh	R/W	W	Hash filter MSBs register	34
0xC160 0FE0	etn1_intstatus	R	W	Interrupt status register	35
0xC160 0FE4	etn1_intenable	R/W	W	Interrupt enable register	36
0xC160 0FE8	etn1_intclear	W	W	Interrupt clear register	37
0xC160 0FEC	etn1_intset	W	W	Interrupt set register	38
0xC160 0FF4	etn1_powerdown	R/W	W	Power-down register	39
0xC160 0FF8	reserved				40

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
0xC160 0FFC	etn1_moduleid	R	W	Module ID register
BMP ^[2]				
0xC180 0000	bmp_global	R/W	W	BMP global control register
0xC180 0004	bmp_bcnt_ref	R/W	W	BCNT reference values
0xC180 0008	bmp_bcnt_s_ref	R/W	W	BCNT_S reference values
0xC180 000C	bmp_bcnt_val	R	W	Bit counter value
0xC180 0010	bmp_ref	R/W	W	Reference value for system controller wakeup
0xC180 0014	bmp_cnt_mod	R/W	W	Modulo values bit counters
0xC180 0018	bmp_com_addr0	R/W	W	Command word array address offsets 0
0xC180 001C	bmp_com_addr1	R/W	W	Command word array address offsets 1
0xC180 0020	bmp_com_addr2	R/W	W	Command word array address offsets 2
0xC180 0024	bmp_com_addr3	R/W	W	Command word array address offsets 3
0xC180 0028	bmp_cp_x_time	R/W	W	X_time_buf copy signals
0xC180 002C	bmp_crc_con0	R/W	W	CRC control register 0
0xC180 0030	bmp_crc_con1	R/W	W	CRC control register 1
0xC180 0034	bmp_crc_con2	R/W	W	CRC control register 2
0xC180 0038	bmp_enc_dat0	W	W	Cipher engine initialization data 0
0xC180 003C	bmp_enc_dat1	W	W	Cipher engine initialization data 1
0xC180 0040	bmp_enc_dat2	W	W	Cipher engine initialization data 2
0xC180 0044	bmp_enc_dat3	W	W	Cipher engine initialization data 3
0xC180 0048	bmp_enc_dat4	R/W	W	Cipher engine initialization data 4
0xC180 004C	bmp_enc_dat5	R/W	W	Cipher engine initialization data 5
0xC180 0050	bmp_enc_dat6	R/W	W	Cipher engine initialization data 6
0xC180 0054	bmp_fcnt_mod	R/W	W	Frame counter modulo value
0xC180 0058	bmp_fcnt_val	R/W	W	Frame counter value
0xC180 005C	reserved	R/W	W	reserved
0xC180 0060	reserved	R/W	W	reserved
0xC180 0064	reserved	R/W	W	reserved
0xC180 0068	reserved	R	W	reserved
0xC180 006C	bmp_m_pointer	R/W	W	MCU command array pointer
0xC180 0070	bmp_max_error	R/W	W	Maximum measurable phase error
0xC180 0074	bmp_phase_con	R/W	W	Phase Correction Control
0xC180 0078	bmp_phase_error	R/W	W	Phase Error
0xC180 007C	bmp_phase_offset	R/W	W	Phase Offset
0xC180 0080	bmp_r_buf_con	R/W	W	Length of r_buf
0xC180 0084	bmp_r_buf0	R	W	Receive data buffer 0
0xC180 0088	bmp_r_buf1	R	W	Receive data buffer 1
0xC180 008C	bmp_r_pointer	R/W	W	RCU command array pointer
0xC180 0090	bmp_rf_pointer	R/W	W	RFCU command array pointer
0xC180 0094	bmp_rfpu_con	R/W	W	RFPU control register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC180 0098	bmp_rpu_con0	R/W	W	Receive Peripheral Units control register 0	1
0xC180 009C	bmp_rpu_con1	R/W	W	Receive Peripheral Units control register 1	2
0xC180 00A0	bmp_rpu_pat	R	W	Receive Peripheral Units pattern register	3
0xC180 00A4	bmp_rpu_stat0	R/W	W	Receive Peripheral Units status register 0	4
0xC180 00A8	bmp_rpu_stat1	R/W	W	Receive Peripheral Units status register 1	5
0xC180 00AC	bmp_scnt_mod	R/W	W	Modulo values of slot counters Scnt0 and Scnt1	6
0xC180 00B0	bmp_scnt_val	R/W	W	Current values of slot counters Scnt0 and Scnt1	7
0xC180 00B4	bmp_sif_buf0	R/W	W	Serial interface buffer 0	8
	bmp_spif_adr			Serial peripheral interface address register	9
0xC180 00B8	bmp_sif_buf1	R/W	W	Serial interface buffer 1	10
	bmp_spif_datout			Serial peripheral interface data out register	11
0xC180 00BC	bmp_sif_buf2	R/W	W	Serial interface buffer 2	12
	bmp_spif_datin			Serial peripheral interface data in register	13
0xC180 00C0	bmp_sif_con	R/W	W	Serial interface control register	14
	bmp_spif_con			Serial peripheral interface control register	15
0xC180 00C4	bmp_start_time	R	W	Receive Control Unit start time register	16
0xC180 00C8	bmp_sync_pat0	R/W	W	Synchronization pattern 0	17
	bmp_symrec_con0	R/W	W	Symbol recovery control register 0	18
0xC180 00CC	bmp_sync_pat1	R/W	W	Synchronization pattern 1	19
	bmp_symrec_con1	R/W	W	Symbol recovery control register 0	20
0xC180 00D0	bmp_sypo_con	R/W	W	Synchronization port control register	21
0xC180 00D4	bmp_sypo_ref_cnt	R	W	SYNCPORT reference value	22
0xC180 00D8	bmp_t_buf_len	R/W	W	Length of T_BUFL	23
0xC180 00DC	bmp_t_buf0	R/W	W	Transmit data buffer 0	24
0xC180 00E0	bmp_t_buf1	R/W	W	Transmit data buffer 1	25
0xC180 00E4	bmp_t_pointer	R/W	W	TCU command array pointer	26
0xC180 00E8	bmp_tpu_con	R/W	W	Transmit Peripheral Unit Control register	27
0xC180 00EC	bmp_com_add_rdata	R/W	W	BMP RAM RDATA cache, START and END addresses	28
0xC180 00F0	bmp_com_add_tdata	R/W	W	BMP RAM TDATA cache, START and END addresses	29
0xC180 00F4	bmp_rdata_pointer	R/W	W	BMP RAM RDATA cache pointer	30
0xC180 00F8	bmp_tdata_pointer	R/W	W	BMP RAM TDATA cache pointer	31
0xC180 00FC	bmp_demod_isi	R/W	W	Demodulator, inter symbol interference [2]	32
0xC180 0100	bmp_demod_modul	R/W	W	Demodulator, modulation index [2]	33
0xC180 0104	bmp_demod_offset	R/W	W	Demodulator, frequency offset [2]	34
0xC180 0108	bmp_demod_weighting	R/W	W	Demodulator, FIR TAP summing coefficient [2]	35
0xC180 010C	bmp_crc_con3	R/W	W	32bit CRC configuration register	36
0xC180 0110	bmp_crc_con4	R/W	W	32bit CRC configuration register	37
0xC180 0114	bmp_crc_con5	R/W	W	32bit CRC configuration register	38
0xC180 0118	bmp_rpu_stat2	R/W	W	Receive peripheral unit status register 2 (32bit CRC dump)	39

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
0xC180 011C	bmp_rpu_stat3	R/W	W	Receive peripheral unit status register 3 (for correlator)
0xC180 1000 to 0xC180 17FC	bmp_ram	R/W	W	Command word array (2kByte)
TMU				
0xC300 0000	mu_int_status	R	W	Mailbox Interrupt status
0xC300 0004	mu_int_enable		W	Mailbox Interrupt enable
0xC300 0008	mu_int_set	W	W	Mailbox Interrupt Set
0xC300 000C	mu_int_clear	W	W	Mailbox Interrupt clear
0xC300 0010	mu_ctrl	R/W	W	Mailbox Control
0xC301 0000 to 0xC301 FFFF	dsp_io	R/W	W/H	DSP IO registers
0xC302 0000 to 0xC303 FFFF	dsp_xymem	R/W	W/H	DSP X/Y shared data memory (accessed via X-bus). Any access to the upper half of this memory is paged and is routed to the page defined by the xmpa setting.
0xC304 0000 to 0xC305 FFFF	dsp_xymem	R/W	W/H	DSP X/Y shared data memory (accessed via Y-bus) Any access to the upper half of this memory is paged and is routed to the page defined by the ympa setting.
0xC306 0000 to 0xC307 FFFF	dsp_pmem	R/(W)	W/H	DSP program memory Any access to the upper half of this memory is paged and is routed to the page defined by the pmpa setting.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
VPB1 Peripherals				
DAIF ^[3]				
IIC				
0xC200 1000	iicrx	R	B	receive FIFO
0xC200 1000	iictx	W	H	transmit FIFO
0xC200 1004	iicsts	R/C	H	status register
0xC200 1008	iicctl	R/W	H	control register
0xC200 100C	iicclkhi	R/W	H	clock high time control register
0xC200 1010	iicclklo	R/W	H	clock low time control register
0xC200 1014	iicaddr	R/W	B	slave address
0xC200 1018	iicholddat	R/W	B	data hold time control register
0xC200 1028	iictxs	W	B	transmit FIFO in Slave mode
SPI1				
0xC200 2000	spi1_global	R/W	H	SPI1 global control register
0xC200 2004	spi1_con	R/W	W	SPI1 control register
0xC200 2008	spi1_frm	R/W	H	SPI1 frame count register
0xC200 200C	spi1_ier	R/W	H	SPI1 interrupt enable register
0xC200 2010	spi1_stat	R/W	H	SPI1 status register
0xC200 2014	spi1_dat	R/W	H	SPI1 data buffer
0xC200 2018	spi1_dat_mask	W	H	SPI1 data buffer for using Mask mode
0xC200 201C	spi1_mask	R/W	H	mask register
0xC200 2020	spi1_addr	R/W	H	address register
0xC200 2400	spi1_timer_ctrl	R/W	H	timer control registers
0xC200 2404	spi1_timer_count	R/W	H	timer counter registers
0xC200 2408	spi1_timer_status	R/W	H	timer status registers
UART1				
0xC200 4000	uart1_rbr	R	B	receive buffer register
0xC200 4000	uart1_thr	W	B	transmit holding register
0xC200 4004	uart1_ier	R/W	B	interrupt enable
0xC200 4008	uart1_iir	R	B	interrupt identification
0xC200 4008	uart1_fcr	W	B	FIFO control
0xC200 400C	uart1_lcr	R/W	B	line control
0xC200 4010	uart1_mcr	R/W	B	modem control
0xC200 4014	uart1_lsr	RaC	B	line status
0xC200 4018	uart1_msr	RaC	B	modem status
0xC200 401C	uart1_scr	R/W	B	scratch pad
0xC200 4000	uart1_dll	R/W	B	divisor latch LSB
0xC200 4004	uart1_dlm	R/W	B	divisor latch MSB
0xC200 401C	uart1_acr	R/W	B	autobaud control
0xC200 4400	uart1_timer_ctrl	R/W	B	timed IRQ/DMA control

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
0xC200 4404	uart1_timer_count	R/W	H	timed IRQ/DMA counter value
0xC200 4408	uart1_timer_status	R/W	H	timed IRQ/DMA interrupt status
0xC200 4800	uart1_irda	R/W	B	IrDA control
0xC200 4C00	uart1_fdiv_ctrl	R/W	B	fractional divider control
0xC200 4C04	uart1_fdiv_m	R/W	H	fractional divider M value
0xC200 4C08	uart1_fdiv_n	R/W	H	fractional divider N value
UART2				
0xC200 5000	uart2_rbr	R	B	receive buffer register
0xC200 5000	uart2_thr	W	B	transmit holding register
0xC200 5004	uart2_iер	R/W	B	interrupt enable
0xC200 5008	uart2_iir	R	B	interrupt identification
0xC200 5008	uart2_fcr	W	B	FIFO control
0xC200 500C	uart2_lcr	R/W	B	line control
0xC200 5010	uart2_mcr	R/W	B	modem control
0xC200 5014	uart2_lsr	RaC	B	line status
0xC200 5018	uart2_msr	RaC	B	modem status
0xC200 501C	uart2_scr	R/W	B	scratch pad
0xC200 5000	uart2_dll	R/W	B	divisor Latch LSB
0xC200 5004	uart2_dlm	R/W	B	divisor Latch MSB
0xC200 501C	uart2_acr	R/W	B	autobaud Control
0xC200 5400	uart2_timer_ctrl	R/W	B	timed IRQ/DMA control
0xC200 5404	uart2_timer_count	R/W	H	timed IRQ/DMA counter value
0xC200 5408	uart2_timer_status	R/W	H	timed IRQ/DMA interrupt status
0xC200 5800	uart2_irda	R/W	B	IrDA control
0xC200 5C00	uart2_fdiv_ctrl	R/W	B	fractional divider control
0xC200 5C04	uart2_fdiv_m	R/W	H	fractional divider M value
0xC200 5C08	uart2_fdiv_n	R/W	H	fractional divider N value
IIS				
0xC200 6000	iiscon	R/W	H	IIS configuration register
0xC200 6004	iisstat	R/C	H	IIS status register
0xC200 6008	iisdata	R/W	H	IIS data register for left and right channels

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
USB				
0xC200 9000	usbintstat	R	W	interrupt status register
0xC200 9004	usbinten	R/W	H	interrupt enable register
0xC200 9008	usbintclr	W	H	interrupt clear register
0xC200 900C	usbintset	W	H	interrupt set register
0xC200 9010	usbcmddcode	W	W	command code register (for USB handler)
0xC200 9014	usbcmddata	R	B	command data register (for USB handler)
0xC200 9018	usbrxdata	R	W	receive data register
0xC200 901C	usbttxdata	W	W	transmit data register
0xC200 9020	usbrxpckl	R	H	receive packet length register
0xC200 9024	usbtxpckl	R/W	H	transmit packet length register
0xC200 9028	usbcon	R/W	B	control register
0xC200 902C	usbfiqsel	W	B	usb_fiq interrupt selection register
IPINT				
0xC200 C000	ipint_global	R/W	H	
0xC200 C004	ipint_cntl0	R/W	H	PCM/IOM port control (0)
0xC200 C008	ipint_cntl1	R/W	H	PCM/IOM port control (1)
0xC200 C00C	ipint_hpout0	R/W	H	PCM/IOM port data out - slot 0 and slot 1
0xC200 C010	ipint_hpout1	R/W	H	PCM/IOM port data out - slot 2 and slot 3
0xC200 C014	ipint_hpout2	R/W	H	PCM/IOM port data out - slot 4 and slot 5
0xC200 C018	ipint_hpout3	R/W	H	PCM/IOM port data out - slot 6 and slot 7
0xC200 C01C	ipint_hpout4	R/W	H	PCM/IOM port data out - slot 8 and slot 9
0xC200 C020	ipint_hpout5	R/W	H	PCM/IOM port data out - slot 10 and slot 11
0xC200 C078	ipint_hpin0	R	H	PCM/IOM port data in - slot 0 and slot 1
0xC200 C07C	ipint_hpin1	R	H	PCM/IOM port data in - slot 2 and slot 3
0xC200 C080	ipint_hpin2	R	H	PCM/IOM port data in - slot 4 and slot 5
0xC200 C084	ipint_hpin3	R	H	PCM/IOM port data in - slot 6 and slot 7
0xC200 C088	ipint_hpin4	R	H	PCM/IOM port data in - slot 8 and slot 9
0xC200 C08C	ipint_hpin5	R	H	PCM/IOM port data in - slot 10 and slot 11
ADPCM				
0xC200 D000	adp_status	R	H	ADPCM status register (covers all channels)
0xC200 D004	adp1_enc_conf	R/W	H	ADPCM channel 1 encoder configuration register
0xC200 D008	adp1_enc_in	R/W	H	ADPCM channel 1 encoder input register
0xC200 D00C	adp1_enc_out	R	H	ADPCM channel 1 encoder output register
0xC200 D010	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)
0xC200 D014	adp1_dec_conf	R/W	H	ADPCM channel 1 decoder configuration register
0xC200 D018	adp1_dec_in	R/W	H	ADPCM channel 1 decoder input register
0xC200 D01C	adp1_dec_out	R	H	ADPCM channel 1 decoder output register
0xC200 D020	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)
0xC200 D024	adp2_enc_conf	R/W	H	ADPCM channel 2 encoder configuration register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
0xC200 D028	adp2_enc_in	R/W	H	ADPCM channel 2 encoder input register	1
0xC200 D02C	adp2_enc_out	R	H	ADPCM channel 2 encoder output register	2
0xC200 D030	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)	3
0xC200 D034	adp2_dec_conf	R/W	H	ADPCM channel 2 decoder configuration register	4
0xC200 D038	adp2_dec_in	R/W	H	ADPCM channel 2 decoder input register	5
0xC200 D03C	adp2_dec_out	R	H	ADPCM channel 2 decoder output register	6
0xC200 D040	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)	7
0xC200 D044	adp3_enc_conf	R/W	H	ADPCM channel 3 encoder configuration register	8
0xC200 D048	adp3_enc_in	R/W	H	ADPCM channel 3 encoder input register	9
0xC200 D04C	adp3_enc_out	R	H	ADPCM channel 3 encoder output register	10
0xC200 D050	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)	11
0xC200 D054	adp3_dec_conf	R/W	H	ADPCM channel 3 decoder configuration register	12
0xC200 D058	adp3_dec_in	R/W	H	ADPCM channel 3 decoder input register	13
0xC200 D05C	adp3_dec_out	R	H	ADPCM channel 3 decoder output register	14
0xC200 D060	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)	15
0xC200 D064	adp4_enc_conf	R/W	H	ADPCM channel 4 encoder configuration register	16
0xC200 D068	adp4_enc_in	R/W	H	ADPCM channel 4 encoder input register	17
0xC200 D06C	adp4_enc_out	R	H	ADPCM channel 4 encoder output register	18
0xC200 D070	equal to 0xC200 D000	R	H	ADPCM status register (covers all channels)	19
0xC200 D074	adp4_dec_conf	R/W	H	ADPCM channel 4 decoder configuration register	20
0xC200 D078	adp4_dec_in	R/W	H	ADPCM channel 4 decoder input register	21
0xC200 D07C	adp4_dec_out	R	H	ADPCM channel 4 decoder output register	22
AMU1					
0xC200 E000	amu1_cfg0	R/W	W	configuration for area 0	23
0xC200 E004	amu1_start0	R/W	W	start address for area 0	24
0xC200 E008	amu1_stop0	R/W	W	stop address for area 0	25
0xC200 E010	amu1_cfg1	R/W	W	configuration for area 1	26
0xC200 E014	amu1_start1	R/W	W	start address for area 1	27
0xC200 E018	amu1_stop1	R/W	W	stop address for area 1	28
0xC200 E020	amu1_cfg2	R/W	W	configuration for area 2	29
0xC200 E024	amu1_start2	R/W	W	start address for area 2	30
0xC200 E028	amu1_stop2	R/W	W	stop address for area 2	31
0xC200 E080	amu1_area_status	R/C	H	area status register	32
0xC200 E084	amu1_bus_status	R	W	bus status register	33
0xC200 E088	amu1_address_status	R	W	address status register	34

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
AMU2				
0xC200 E100	amu2_cfg0	R/W	W	configuration for area 0
0xC200 E104	amu2_start0	R/W	W	start address for area 0
0xC200 E108	amu2_stop0	R/W	W	stop address for area 0
0xC200 E110	amu2_cfg1	R/W	W	configuration for area 1
0xC200 E114	amu2_start1	R/W	W	start address for area 1
0xC200 E118	amu2_stop1	R/W	W	stop address for area 1
0xC200 E120	amu2_cfg2	R/W	W	configuration for area 2
0xC200 E124	amu2_start2	R/W	W	start address for area 2
0xC200 E128	amu2_stop2	R/W	W	stop address for area 2
0xC200 E180	amu2_area_status	R/C	H	area status register
0xC200 E184	amu2_bus_status	R	W	bus status register
0xC200 E188	amu2_address_status	R	W	address status register
AMU3				
0xC200 E200	amu3_cfg0	R/W	W	configuration for area 0
0xC200 E204	amu3_start0	R/W	W	start address for area 0
0xC200 E208	amu3_stop0	R/W	W	stop address for area 0
0xC200 E210	amu3_cfg1	R/W	W	configuration for area 1
0xC200 E214	amu3_start1	R/W	W	start address for area 1
0xC200 E218	amu3_stop1	R/W	W	stop address for area 1
0xC200 E220	amu3_cfg2	R/W	W	configuration for area 2
0xC200 E224	amu3_start2	R/W	W	start address for area 2
0xC200 E228	amu3_stop2	R/W	W	stop address for area 2
0xC200 E280	amu3_area_status	R/C	H	area status register
0xC200 E284	amu3_bus_status	R	W	bus status register
0xC200 E288	amu3_address_status	R	W	address status register
AMU4				
0xC200 E300	amu4_cfg0	R/W	W	configuration for area 0
0xC200 E304	amu4_start0	R/W	W	start address for area 0
0xC200 E308	amu4_stop0	R/W	W	stop address for area 0
0xC200 E310	amu4_cfg1	R/W	W	configuration for area 1
0xC200 E314	amu4_start1	R/W	W	start address for area 1
0xC200 E318	amu4_stop1	R/W	W	stop address for area 1
0xC200 E320	amu4_cfg2	R/W	W	configuration for area 2
0xC200 E324	amu4_start2	R/W	W	start address for area 2
0xC200 E328	amu4_stop2	R/W	W	stop address for area 2
0xC200 E380	amu4_area_status	RAC	W	area status register
0xC200 E384	amu4_bus_status	R	W	bus status register
0xC200 E388	amu4_address_status	R	W	address status register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
AMU5				
0xC200 E400	amu5_cfg0	R/W	W	configuration for area 0
0xC200 E404	amu5_start0	R/W	W	start address for area 0
0xC200 E408	amu5_stop0	R/W	W	stop address for area 0
0xC200 E410	amu5_cfg1	R/W	W	configuration for area 1
0xC200 E414	amu5_start1	R/W	W	start address for area 1
0xC200 E418	amu5_stop1	R/W	W	stop address for area 1
0xC200 E420	amu5_cfg2	R/W	W	configuration for area 2
0xC200 E424	amu5_start2	R/W	W	start address for area 2
0xC200 E428	amu5_stop2	R/W	W	stop address for area 2
0xC200 E480	amu5_area_status	RAC	W	area status register
0xC200 E484	amu5_bus_status	R	W	bus status register
0xC200 E488	amu5_address_status	R	W	address status register
AMU6				
0xC200 E500	amu6_cfg0	R/W	W	configuration for area 0
0xC200 E504	amu6_start0	R/W	W	start address for area 0
0xC200 E508	amu6_stop0	R/W	W	stop address for area 0
0xC200 E510	amu6_cfg1	R/W	W	configuration for area 1
0xC200 E514	amu6_start1	R/W	W	start address for area 1
0xC200 E518	amu6_stop1	R/W	W	stop address for area 1
0xC200 E520	amu6_cfg2	R/W	W	configuration for area 2
0xC200 E524	amu6_start2	R/W	W	start address for area 2
0xC200 E528	amu6_stop2	R/W	W	stop address for area 2
0xC200 E580	amu6_area_status	RAC	W	area status register
0xC200 E584	amu6_bus_status	R	W	bus status register
0xC200 E588	amu6_address_status	R	W	address status register
ARB				
0xC200 EA00	arb_cfg0	R/W	H	ARB configuration for master #0
0xC200 EA04	arb_cfg1	R/W	H	ARB configuration for master #1
0xC200 EA08	arb_cfg2	R/W	H	ARB configuration for master #2
0xC200 EA0C	arb_cfg3	R/W	H	ARB configuration for master #3
0xC200 EA10	arb_cfg4	R/W	H	ARB configuration for master #4
0xC200 EA14	arb_cfg5	R/W	H	ARB configuration for master #5
HMP				
0xC200 ED00	hmp_conf	R/W	H	Hybrid Memory port configuration

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
FIR				
0xC201 3000	fir_ip_id	R	W	ID Register
0xC201 3010	fir_con	R/W	W	Control Register
0xC201 3014	fir_conf	R/W	W	Configuration Register
0xC201 3018	fir_para	R/W	W	Parameter Register
0xC201 301C	fir_dv	R/W	W	Divider Register
0xC201 3020	fir_stat	R	W	Status Register
0xC201 3024	fir_tfs	W	W	Transmission Frame Size Register
0xC201 3028	fir_rfs	R	W	Reception Frame Size Register
0xC201 302C	fir_txb	W	W	Transmission Buffer Register
0xC201 3030	fir_rxb	R	W	Reception Buffer Register
0xC201 30E8	fir_imsc	R/W	W	Interrupt Mask Control Register
0xC201 30EC	fir_ris	R	W	Raw Interrupt Status Register
0xC201 30F0	fir_mis	R	W	Masked Interrupt Status Register
0xC201 30F4	fir_icr	W	W	Interrupt Clear Register
0xC201 30F8	fir_isr	W	W	Interrupt Set Register
0xC201 30FC	fir_dma	R/W	W	DMA Control Register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
VPB2 Peripherals				
SCTU1				
0xC210 2000	tim1cr	R/W	H	timer 1 control register
0xC210 2004	tim1rr	R/W	H	timer 1 reload register
0xC210 2008	tim1wr	R	H	timer 1 work register
0xC210 200C	tim1c0	R/W	H	timer 1 channel 0 register
0xC210 2010	tim1c1	R/W	H	timer 1 channel 1 register
0xC210 2014	tim1c2	R/W	H	timer 1 channel 2 register
0xC210 2018	tim1c3	R/W	H	timer 1 channel 3 register
0xC210 201C	tim1sr	R/C	B	timer 1 status register
0xC210 2020	tim1pr	R/W	B	timer 1 pre-scaler reload register
SCTU2				
0xC210 3000	tim2cr	R/W	H	timer 2 control register
0xC210 3004	tim2rr	R/W	H	timer 2 reload register
0xC210 3008	tim2wr	R	H	timer 2 work register
0xC210 300C	tim2c0	R/W	H	timer 2 channel 0 register
0xC210 3010	tim2c1	R/W	H	timer 2 channel 1 register
0xC210 3014	tim2c2	R/W	H	timer 2 channel 2 register
0xC210 3018	tim2c3	R/W	H	timer 2 channel 3 register
0xC210 301C	tim2sr	R/C	B	timer 2 status register
0xC210 3020	tim2pr	R/W	B	timer 2 pre-scaler reload register
GPIOA				
0xC210 4000	gpioa_pins	R	W	GPIOA pins value register
0xC210 4004	gpioa_or	R/W	W	GPIOA output register
0xC210 4008	gpioa_dr	R/W	W	GPIOA direction register
GPIOB				
0xC210 4200	gpiob_pins	R	W	GPIOB pins value register
0xC210 4204	gpiob_or	R/W	W	GPIOB output register
0xC210 4208	gpiob_dr	R/W	W	GPIOB direction register
GPIOC				
0xC210 4400	gpioc_pins	R	W	GPIOC pins value register
0xC210 4404	gpioc_or	R/W	W	GPIOC output register
0xC210 4408	gpioc_dr	R/W	W	GPIOC direction register

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description	
EXTINT					
0xC210 5004	extint_cfg1	R/W	B	EXTINT1 configuration register	1
0xC210 5008	extint_cfg2	R/W	B	EXTINT2 configuration register	2
0xC210 5014	extint_cfg5	R/W	B	EXTINT5 configuration register	3
0xC210 501C	extint_cfg7	R/W	B	EXTINT7 configuration register	4
0xC210 5028	extint_cfg10	R/W	B	EXTINT10 configuration register	5
0xC210 5030	extint_cfg12	R/W	B	EXTINT12 configuration register	6
0xC210 5034	extint_cfg13	R/W	B	EXTINT13 configuration register	7
0xC210 503C	extint_cfg15	R/W	B	EXTINT15 configuration register	8
0xC210 5040	extint_cfg16	R/W	B	EXTINT16 configuration register	9
0xC210 5044	extint_cfg17	R/W	B	EXTINT17 configuration register	10
0xC210 5048	extint_cfg18	R/W	B	EXTINT18 configuration register	11
0xC210 504C	extint_cfg19	R/W	B	EXTINT19 configuration register	12
0xC210 5054	extint_cfg21	R/W	B	EXTINT21 configuration register	13
0xC210 5058	extint_cfg22	R/W	B	EXTINT22 configuration register	14
0xC210 505C	extint_cfg23	R/W	B	EXTINT23 configuration register	15
0xC210 5060	extint_enable1	R/W	W	EXTINT interrupt1 mask	16
0xC210 5064	extint_enable2	R/W	W	EXTINT interrupt2 mask	17
0xC210 5068	extint_enable3	R/W	W	EXTINT interrupt3 mask	18
0xC210 506C	extint_status	R/C	W	EXTINT interrupt status	19
0xC210 5070	extint_signal	R	W	EXTINT signal status	20
KBS					
0xC210 6000	kbs_deb	R/W	B	Keypad debounce counter register	21
0xC210 6004	kbs_state_cond	R	B	Keypad IO control state condition register	22
0xC210 6008	kbs_it	R/W	B	Keypad interrupt register	23
0xC210 6010	kbs_fast_tst	R/W	B	Keypad fast_tst register	24
0xC210 6014	kbs_matrix_dim	R/W	B	Keypad matrix_dim register	25
0xC210 6040	kbs_data0	R	B	Keypad data0 register	26
0xC210 6044	kbs_data1	R	B	Keypad data1 register	27
0xC210 6048	kbs_data2	R	B	Keypad data2 register	28
0xC210 604C	kbs_data3	R	B	Keypad data3 register	29
0xC210 6050	kbs_data4	R	B	Keypad data4 register	30
0xC210 6054	kbs_data5	R	B	Keypad data5 register	31
0xC210 6058	kbs_data6	R	B	Keypad data6 register	32
0xC210 605C	kbs_data7	R	B	Keypad data7 register	33
PWM2					
0xC210 9000	pwm2_pf	R/W	H	PWM2 frequency register	34
0xC210 9004	pwm2_tmr	R/W	H	PWM2 comparator value register	35
0xC210 9008	pwm2_ctrl	R/W	H	PWM2 control register	36

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
PWM3				
0xC210 A000	pwm3_pf	R/W	H	PWM3 frequency register
0xC210 A004	pwm3_tmr	R/W	H	PWM3 comparator value register
0xC210 A008	pwm3_ctrl	R/W	H	PWM3 control register

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [?]	Description
VPB3 Peripherals				
	CGU			
0xC220 0000	cgudspcon	R/W	W	CGU control register for DSP clocks
0xC220 0004	cgusccon	R/W	W	CGU control register for SC clocks
0xC220 0008	cgugatesc	R/W	W	control register for gating SC clocks
0xC220 000C	cgusleepsc	R/W	W	control register for gating SC clocks in sleep state
0xC220 0010	cgufdiv	R/W	W	CGU fractional divider increment register
0xC220 0014	cguper1con	R/W	W	CGU control register for USB & IIS clock frequencies
0xC220 0018	cguper2con	R/W	W	CGU control register for Ethernet clock frequencies
0xC220 001C	cguper2bwcon	R/W	W	CGU control register for user defined bandwidth setting for PLL_PER2
0xC220 0020	cgufixcon	R/W	W	CGU control register of PLL_FIX
0xC220 0024	cgudivcon	R/W	W	control registers for clock dividers (7.68MHz, 32kHz)
	TBU			
0xC220 1000	qbc	R	H	bit counter register
0xC220 1008	tbucon	R/W	H	TBU control register
0xC220 100C	tbuflr	R/W	H	TBU frame length register
	PDCU			
0xC220 2000	pdcucon	R/W	H	power-down control register
0xC220 2004	gpon0	R/W	H	GPON[0] control register
0xC220 2008	gpon1	R/W	H	GPON[1] control register
0xC220 200C	gpon2	R/W	H	GPON[2] control register
0xC220 2010	sstopcnt	R/W	H	SC stop count register
0xC220 2014	ssleepcnt	R/W	H	SC sleep counter
0xC220 2018	pdcu_mempd	R/W	W	power-off control for memories
	WDRU			
0xC220 3000	wdrucon	R/W	H	WDRU reset control register
0xC220 3004	wdtim	R/W	H	WDRU watchdog period register
	SCON			
0xC220 4000	sysver	R	H	system version register
0xC220 4004	syscon0	R/W	W	system configuration register 0
0xC220 4008	sysprot	R/W	W	system protection control register
0xC220 400C	sysmux0	R/W	W	system multiplexer register 0
0xC220 4010	sysmux1	R/W	W	system multiplexer register 1
0xC220 4014	sysmux2	R/W	W	system multiplexer register 2
0xC220 4018	sysmux3	R/W	W	system multiplexer register 3
0xC220 401C	sysmux4	R/W	W	system multiplexer register 4
0xC220 4020	sysmux5	R/W	W	system multiplexer register 5
0xC220 4034	syspad0	R/W	W	system I/O pad configuration register 0
0xC220 4038	syspad1	R/W	W	system I/O pad configuration register 1

1
2
3
4
5

Table 159: Register map of the system controller peripherals...continued

Address	Name	R/W	Size [1]	Description
0xC220 403C	syspad2	R/W	W	system I/O pad configuration register 2
0xC220 4040	syspad3	R/W	W	system I/O pad configuration register 3
0xC220 4044	syspad4	R/W	W	system I/O pad configuration register 3
0xC220 4048	syspad5	R/W	W	system I/O pad configuration register 3
0xC220 4080	sysesh	R/W	W	system HMP/FMP pad speed control register

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit.

[2] Function not available at DVFD8187

[3] consult datasheet of DAP1902 or DRF1902.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.4 SCRAM - System Controller RAM

9.4.1 Features

- Internal 2 banks, one of 128kbytes, one of 64kbytes (192kB total), of SRAM, 32-bit wide, 0/1 wait-state for respectively sequential/non-sequential access, running at Multi-layer AHB bus speed
- Banks are on separate Multi-layer AHB bus slaves in order to allow simultaneous memory access from 2 Multi-layer AHB bus masters.
- 8, 16 or 32 bits read/write allowed

9.4.2 Block diagram

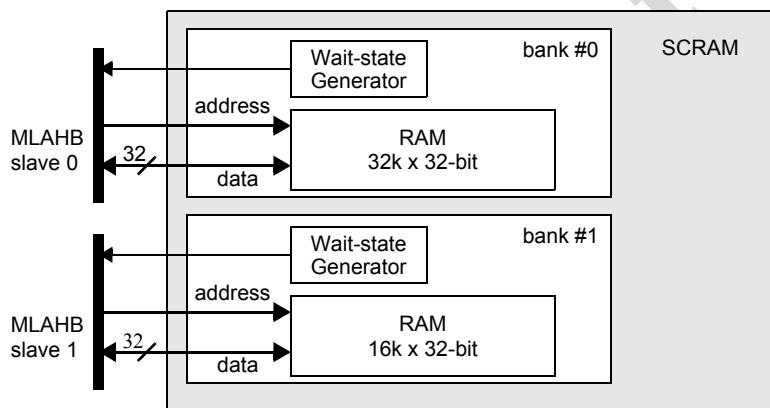


Fig 55. SCRAM block diagram

9.4.3 Functional description

Access of 8, 16 or 32 bits data is allowed only at aligned address (i.e. every address for byte access, even addresses for half-word access, address multiple of 4 for word access).

The SCRAM is accessed with zero wait-state for sequential accesses and with one wait-state for non-sequential accesses. The wait-state insertion is done automatically by the hardware.

9.5 SCROM - System Controller ROM

9.5.1 Features

- Internal 128 kbytes of ROM; 32-bit wide; 0/1 wait-state for respectively sequential/non-sequential access; running at Multi-layer AHB bus speed
- 8, 16 or 32 bits read allowed

9.5.2 Block diagram

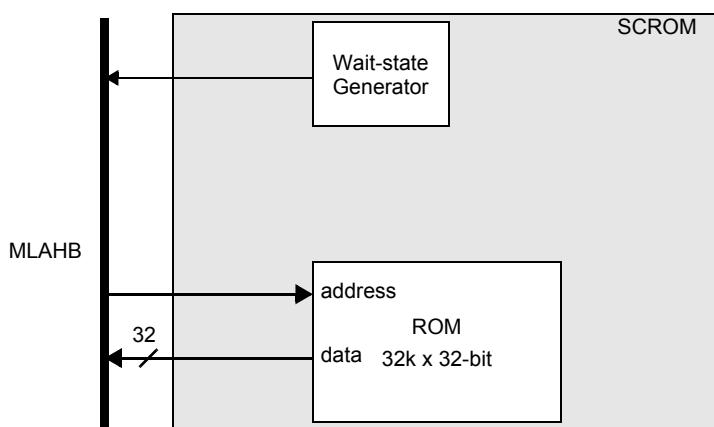


Fig 56. SCROM block diagram

9.5.3 Functional description

The SCROM is accessed with zero wait-state for sequential accesses and with one wait-state for non-sequential accesses. The wait-state insertion is done automatically by the hardware.

Read of 8, 16 or 32 bits data is allowed only at aligned address (i.e. every address for byte access, even addresses for half-word access, address multiple of 4 for word access).

Remark: The SCROM can only be accessed by the ARM processor.

Remark: Boot address is 0xFFFF 0000 (which is not the first SCROM address, see Table 157)

Any attempt to write to the SCROM generates an abort.

Any access to reserved area of the SCROM (Table 157) generates dummy data.

The SCROM contains the BOOT software (see Section 9.6).

9.6 SCROM Software (for ordering information on the different SCROM masks see [Table 4](#))

9.6.1 Features

The SCROM includes the (primary) boot loader software used for production or during chip testing phases. Details and restrictions can be found in [\[29.\]](#). The following features are supported:

1. Boot from different sources (parallel NOR flash, serial SPI flash or external host connection via dedicated UART2 port).
The SCROM mask B does supports additionally direct boot from NAND flash.
2. Hardware port configuration with 2 GPIO pins can be used to determine the preferred boot device to increase the boot speed and robustness.
3. Apply chip configuration based on a boot parameter table located in an external non-volatile memory (e.g. EEPROM or SPI-flash)

9.6.2 Boot sequence

The boot sequence for SCROM mask A is presented at [Figure 57](#) and for SCROM mask B at [Figure 58](#). The simplified scheme is:

1. Basic Init with
 - a. Disable all interrupts
 - b. Disable all ARM926 internal stuff (MMU/Cache/TLB/..)
 - c. Release of RESEXT pin for reset of external devices
 - d. Configure the EBI1 for chip select 0 (CS0)
 - e. Set system mode
 - f. Read GPIO's (GPIOB23 and GPIOA13) for boot configuration
 - g. In case of boot GPIO configuration equal (1-0) jump to the start address of the EBI1 chip select 0 (CS0) for code execution of the application. The application code has to be downloaded before via the debug interface.
2. Select boot device from the fetched boot configuration at GPIO's
If there is none selected check for physical boot device with the sequence of:
 - a. For SCROM mask A:
parallel flash -> SPI flash -> external host -> jump to address 0x8000 0000
 - b. for SCROM mask B:
SPI flash -> NAND flash -> external host -> jump to address 0x8000 0000
3. Jump to boot device and check for valid content (magic number or boot parameters) or established connection if external host is selected.
In case of boot from SPI flash, apply the boot parameters either from EEPROM or from SPI flash and copy the secondary boot loader code from SPI flash into the internal SCRAM.
In case of SCROM mask A the EEPROM can be used to hold a secondary bootloader copied into internal SCRAM to support booting from NAND flash (see [\[30.\]](#)).
4. Jump to the defined start address (for SPI/NAND flash boot with default HW register settings) for code or secondary boot loader execution.

5. In case of SCROM mask B and unexpected errors (e.g. boot parameter table inconsistent or size of data to be loaded exceeds available SCRAM size) a jump to reset at address 0xFFFF 0000 will be executed.

The application code (in parallel flash, SPI flash or NAND flash) has to be downloaded before via the debug interface. The loaded software can then be used for more sophisticated purposes like download application code using SC peripherals (e.g. UART, USB, etc.) and flash programming.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

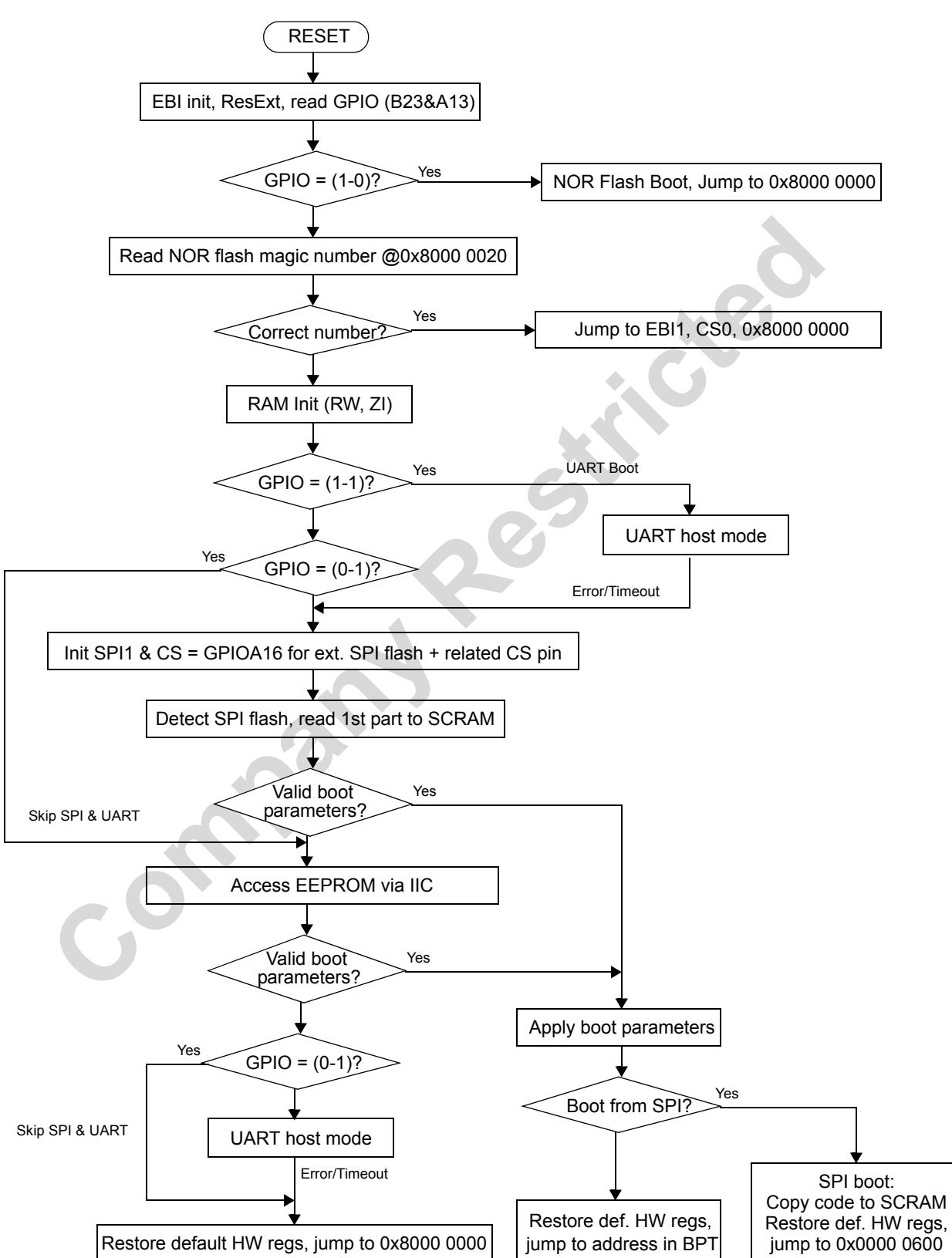


Fig 57. Boot sequence for the SCROM mask A

Complementary Restricted

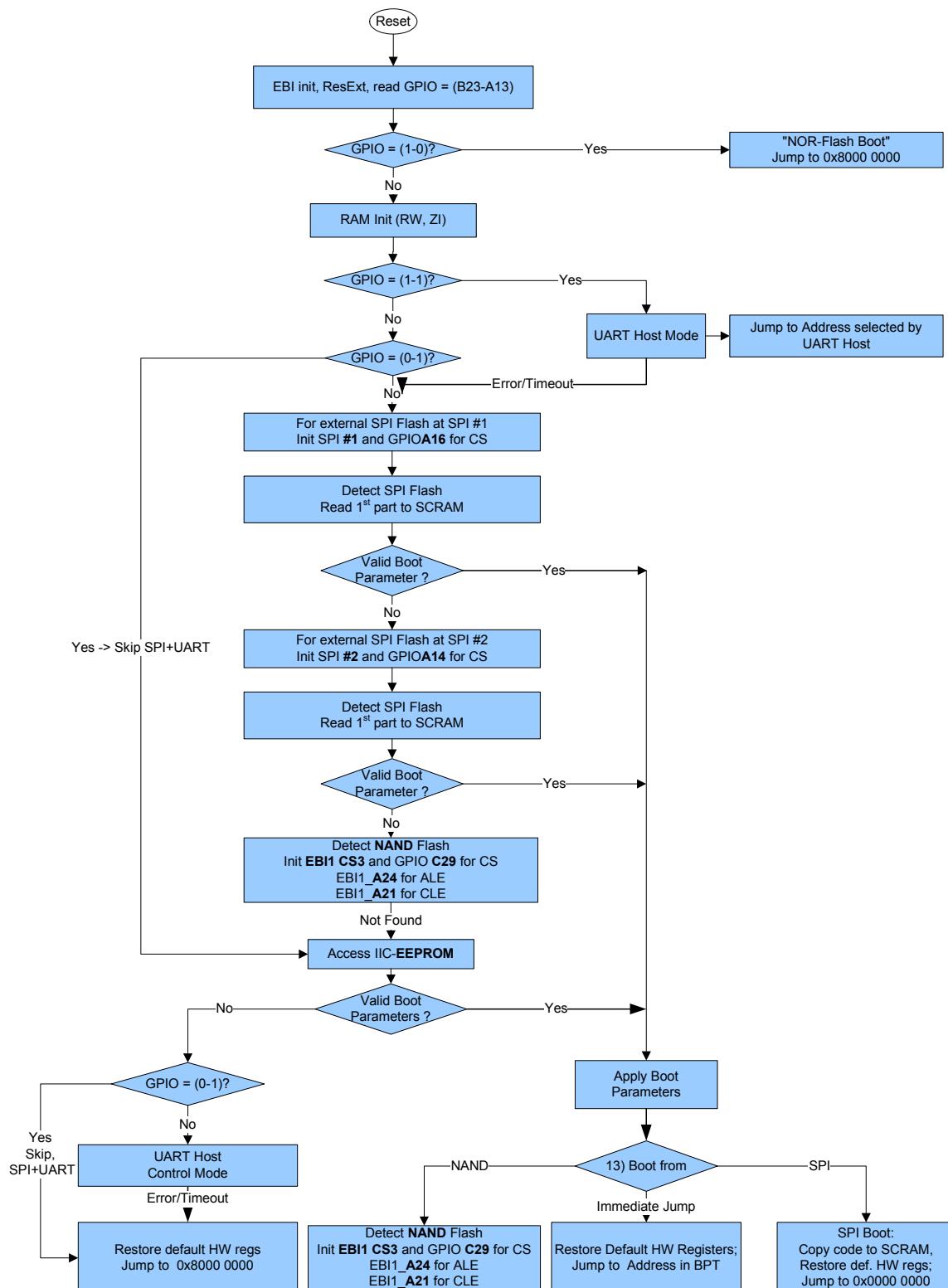


Fig 58. Boot sequence for the SCROM mask B

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.6.3 GPIO port configuration

The boot options defined by the GPIO port configuration are summarized in Table 160.

Table 160:GPIO boot options

GPIOB23	GPIOA13	Description
0	0	Default configuration, GPIOs are floating or tied to GND. Sequence is: <ul style="list-style-type: none"> • SPI-flash and EEPROM (IIC) will be checked for a valid boot parameter table (in case of SCROM mask B also NAND). • UART polling follows if no valid boot parameter table is found. • If UART polling leads to an error or a time out than a branch to the address 0x80000000 is executed.
1	0	Branch to address 0x8000 0000. Fast secure boot from NOR-Flash.
1	1	UART host mode
0	1	Skip SPI-Flash, NAND flash and UART check. Boot parameter table for boot (or secondary boot loader in case of SCROM mask A) expected in EEPROM.

9.6.4 Default HW register values restore by boot code

The HW register values restored before exiting the boot procedure are presented in Table 161.

Table 161:Default HW register values

Register name	Address	Default restored value
spi1_global	0xC200 2000	0x0000
spi1_con	0xC200 2004	0x0000 0E08
uart2_dll	0xC200 5000	0x0000 0001
uart2_dlm	0xC200 5004	0x0000 0000
uart2_lcr	0xC200 500C	0x0000 0000
uart2_fdiv_ctrl	0xC200 5C00	0x0000 0000
uart2_fdiv_m	0xC200 5C04	0x0000 0000
uart2_fdiv_n	0xC200 5C08	0x0000 0000
tim1_cr	0xC210 2000	0x0000 0000
tim1_rr	0xC210 2004	0x0000 0000
tim1_sr	0xC210 201C	0x0000 0000
tim1_pr	0xC210 2020	0x0000 0000
gpioa_or	0xC210 4004	0x0000 0000
gpioa_dr	0xC210 4008	0x0000 0000
gpiob_or	0xC210 4204	0x0400 0000
gpiob_dr	0xC210 4208	0x0400 0000
sysmux0	0xC220 400C	0x0005 0141
sysmux1	0xC220 4010	0x0140 0010
sysmux2	0xC220 4014	0x0000 0000
syspad0	0xC220 4034	0x1F5A 5296

Table 161:Default HW register values ...continued

Register name	Address	Default restored value
syspad1	0xC220 4038	0x043F 5761
syspad2	0xC220 403C	0xAFF FFFF
cgscccon	0xC200 0004	0x0011 005D
iicsts	0xC200 1004	0x6AC0
iicclkhi	0xC200 100C	0x0000
iicclklo	0xC200 1010	0x0000
iicholddat	0xC200 1018	0x00

9.6.5 Application information

The GPIOB23 and GPIOA13 are used for determining the boot options. It is expected that after chip (system) reset these GPIO's are terminated according to the wanted boot option. Later in the application SW these GPIO's can be reprogrammed and used for another function.

The magic number is 0x01234567. It is located for the NOR flash at address 0x80000020 in case of SCROM mask A or at the boot parameter table in case of SCROM mask B.

The boot parameter table in the SPI flash is starting at address 0x0. The user has to protect this area from overwriting by application SW. Practically the first sector needs to be protected.

The boot parameter table in the EEPROM is starting at address 0x0. Due to non fixed length of the boot parameter table the system configuration parameter in EEPROM are no longer fixed on an address.

9.6.5.1 Supported serial flash types by the boot loader

Any serial flash type which fulfills the JEDEC standard access scheme for read by the command code 0x03 is supported by the boot loader.

9.7 ADPCM -Adaptive Differential Pulse Code Modulation

The four ADPCM channels are strictly identical. Hereafter, only adp1 is described. adp2, adp3 and adp4 are identical to adp1.

9.7.1 Features

- Four independent channels are supported
- Four different bit rates are supported (16/24/32 and 40kbit/s)
- Three different input formats are selectable (A-law, u-law and 14-bit linear PCM)
- Build in mode for improved performance to bit errors (decoder channel only)

9.7.2 Block diagram

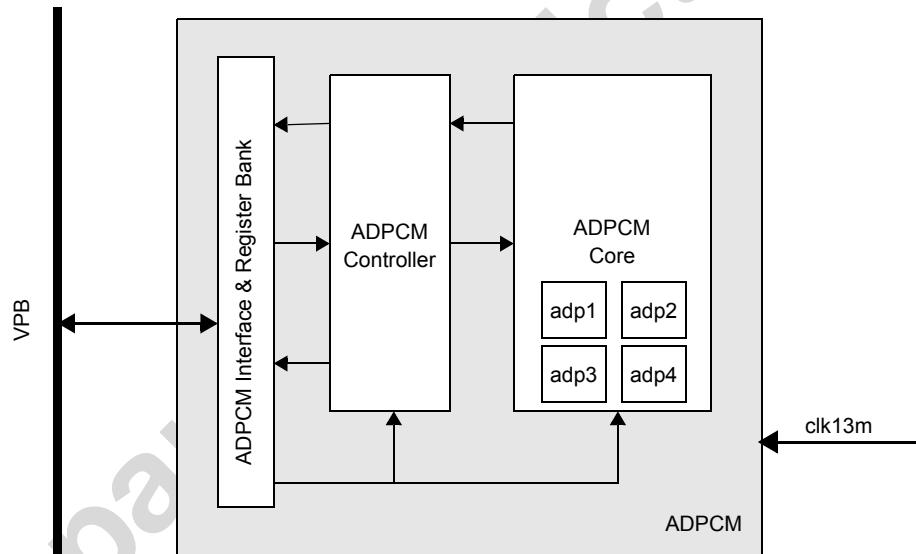


Fig 59. ADPCM accelerator block diagram

9.7.3 Software interface

Table 162: Register overview of the ADPCM accelerator

Symbol	Name	I/O	Reset
adp_status	ADPCM status register (covers all channels)	R	0x0000
adp1_enc_conf	ADPCM channel encoder configuration register	R/W	0x0000
adp1_enc_in	ADPCM channel encoder input register	R/W	0x0000
adp1_enc_out	ADPCM channel encoder output register	R	0x0000
adp1_dec_conf	ADPCM channel decoder configuration register	R/W	0x0000
adp1_dec_in	ADPCM channel decoder input register	R/W	0x0000
adp1_dec_out	ADPCM channel decoder output register	R	0x0000

Table 163: Register adp_status

Bit	Symbol	Access	Value [1]	Description
15 to 8	-	R	0*	reserved
7	ch4dec	R	0*	Indicates that channel 4 decoding result is available
6	ch4enc	R	0*	Indicates that channel 4 encoding result is available
5	ch3dec	R	0*	Indicates that channel 3 decoding result is available
4	ch3enc	R	0*	Indicates that channel 3 encoding result is available
3	ch2dec	R	0*	Indicates that channel 2 decoding result is available
2	ch2enc	R	0*	Indicates that channel 2 encoding result is available
1	ch1dec	R	0*	Indicates that channel 1 decoding result is available
0	ch1enc	R	0*	Indicates that channel 1 encoding result is available

[1] Status bits are automatically cleared after a read of the corresponding data result register.

Table 164: Register adp1_enc_conf / adp1_dec_conf

Bit	Symbol	Access	Value	Description
15 to 7	-	R	0*	reserved
6	sync_em	R/W	0*	sync. error mode
5	opt_em	R/W	0*	optimized error mode
4 to 3	bit rate	R/W		ADPCM Bit rate
			0b11	40 kbit/s
			0b10	32 kbit/s
			0b01	24 kbit/s
			0b00*	16 kbit/s
2 to 1	mode	R/W		ADPCM mode
			0b1x	14 bits linear format
			0b01	8 bits PCM G.711 format, A-law
			0b00*	8 bits PCM G.711 format, u-law
0	reset	R/W	0*	Reset channel to initial values before the next calculation; automatically cleared by hardware to logic '0' after completion of a calculation. [1]

[1] After enabling the block via register **cgugatesc.adpcmen**, a block reset is executed which brings all registers to initial state as defined in ITU-T G.726.r.

Table 165: Register adp1_enc_in

Bit	Symbol	Access	Value	Description
15 to 2	data	R/W	0*	<p>Data input register - data written to this register is passed to the encoder core for processing. Once processing is completed, adp_status.ch1enc bit will be set.</p> <p>adp1_enc_conf.mode = 0b0x : 8 bit PCM data in [15:8]. bits [7:2] are ignored.</p> <p>adp1_enc_conf.mode = 0b1x : 14 bit data formatted as 2's complement</p>
1 to 0	-	R	0*	reserved

[1] If at **adp1_enc_conf.mode** = 0b0x is selected than any value written to **adp1_enc_in[15:8]** is mirrored to **adp1_enc_in[7:0]**. If the register is read back after write the value written to **adp1_enc_in[15:8]** can be seen also at **adp1_enc_in[7:0]**.

Table 166: Register adp1_enc_out

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	data	R	0*	<p>Data output register - the calculation result is stored in this register. A read of this register will reset the adp_status.ch1enc bit.</p> <p>adp1_enc_conf.bitrate = 0b11 : all bits are used</p> <p>adp1_enc_conf.bitrate = 0b10 : 4bit data in [3:0], data[4] = 0.</p> <p>adp1_enc_conf.bitrate = 0b01 : 3bit data in [2:0], data[4:3] = 0</p> <p>adp1_enc_conf.bitrate = 0b00 : 2bit data in [1:0], data[4:2] = 0</p>

Table 167: Register adp1_dec_in

Bit	Symbol	Access	Value	Description
15 to 5	-	R	0*	reserved
4 to 0	data	R/W	0*	<p>Data input register - data written to this register is passed to the decoder core for processing. Once processing is completed, adp_status.ch1dec bit will be set.</p> <p>adp1_dec_conf.bitrate = 0b11 : all bits are used</p> <p>adp1_dec_conf.bitrate = 0b10 : 4bit data in [3:0], data[4] ignored.</p> <p>adp1_dec_conf.bitrate = 0b01 : 3bit data in [2:0], data[4:3] ignored.</p> <p>adp1_dec_conf.bitrate = 0b00 : 2bit data in [1:0], data[4:2] ignored.</p>

Table 168: Register adp1_dec_out

Bit	Symbol	Access	Value	Description
15 to 0	data	R	0*	<p>Data output register - the calculation result is stored in this register. A read of this register will reset the adp_status.ch1dec bit.</p> <p>adp1_dec_conf.mode = 0x : 8 bit PCM data in [15:8]. The same data is repeated in bits [7:0].</p> <p>adp1_dec_conf.mode = 1x : 14 bit data formatted as 2's complement in [15:2]. data[1:0] = 0.</p>

9.7.4 Functional description

The ADPCM block performs a hardware audio encoding and decoding according to ITU-T recommendation G.726.

The G.726 recommendation includes also ITU-T recommendation G.711. The ADPCM block can be used with or without G.711 recommendation, see ITU-T recommendation G.726 - Annex A.

The status register bits are set when a calculation has been completed and they will be automatically reset by hardware when the result of the calculation is read.

The status register **adp_status** contains the status of all encoding / decoding channels, in this way only one read cycle is necessary to know which results are available. This register can be read using different addresses, in this way all encoding / decoding blocks are identical: in the 1st register contains the status, in the 2nd register contains the setting, the 3rd register contains the input value and the 4th register contains the output value.

By means of the register **adp_enc_conf** and **adp_dec_conf** the synchronous channel reset (active high), the bit rate, and the mode can be set. The values are internally copied and are updated only when the ADPCM is not performing any calculations. In this way it is guaranteed that no data corruption can happen. The synchronous reset is performed in the next calculation, during which the reset initial values are used instead of the previous values. After that calculation the synchronous reset bit is automatically reset to logic '0' by hardware.

The configuration bit **opt_em** modifies the ADPCM parameters (adaptation speed) of the 32kbit/s bit rate mode. It can be used in presence of some transmission errors to reduce the noise caused by the errors. This bit is effective only in 32kbit/s.

The configuration bit **sync_em** freezes the ADPCM status and it can be used in presence of several transmission errors (sync. error mode). This bit is effective in all bit rates.

The registers **adp1_enc_in** and **adp1_dec_out** are MSB aligned, on the contrary the registers **adp1_enc_out** and **adp1_dec_in** are LSB aligned. Depending on the calculation mode and bit rate, the size of this register is changing. At the inputs the non-significant bits are masked; at the outputs the non-significant bits are set to logic '0', as the test vectors in ITU-T G.726 Appendix II Test Vectors.

9.7.5 Application information

The ADPCM needs 11 cycles of clock **clk13m** to perform a single calculation. If all calculations are finished the ADPCM goes into standby to avoid unnecessary power consumption. As soon as one of the encoding registers is written a new conversion cycle is started (if **cgugatesc.adpcmen** is set).

When an input value is written, an internal flag (one for each encoding and one for each decoding channel) will toggle and the ADPCM knows that it has to perform a calculation. If the user writes twice the same register before a calculation is done,

then the flag will toggle twice and the ADPCM will not perform any calculation.
Therefore it is important that the software will write a new value in the register only
when the last calculation is finished.

It is suggested to control the status register at the end of the read routine, in order to
check if another register needs to be read.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.8 AHB2VPB - AHB to VPB Bridge

9.8.1 Features

- The AHB2VPB performs the adaptation of the AMBA AHB protocol to the VPB protocol
- The AHB2VPB is fully transparent to the user
- 3 AHB2VPBs are implemented in order to lower bus load on each VPB bus
 - one VPB running with pclk1 equal to hclk (Multi-layer AHB bus clock)
 - two VPB running with pclk2, fixed clock of either 13, 26, 39 or 52 MHz
- Each AHB2VPB contains a one word write buffer to speed up Multi-layer AHB bus release
- No bus error from peripherals (perr) supported due to write buffer
- Most of accesses are performed in 3 pclk1/pclk2 clock cycles. However, some peripherals can extend these accesses with their prdy signal
- The AHB2VPB is a slave of the Multi-layer AHB bus and the only master of each VPB

9.8.2 Block diagram

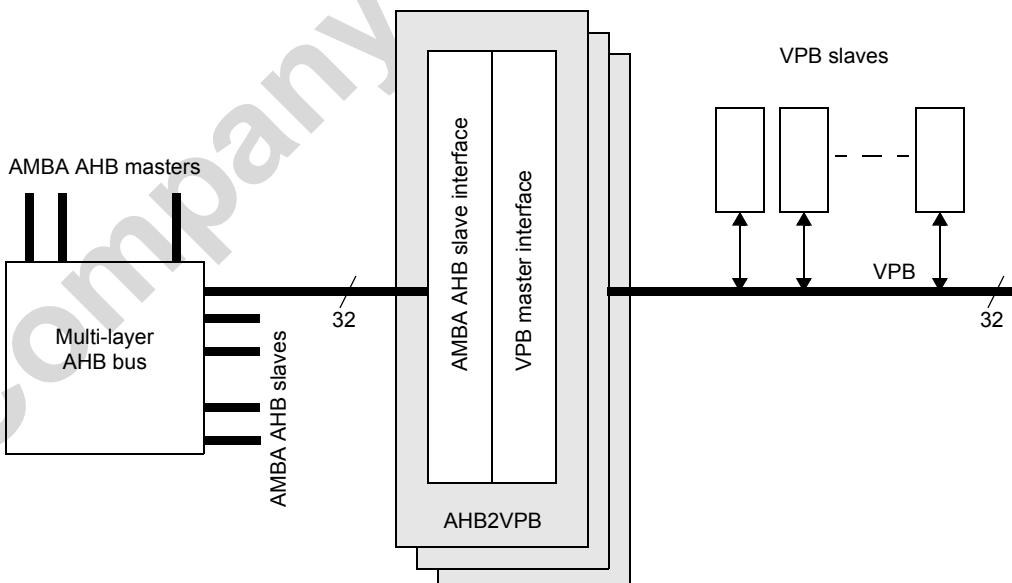


Fig 60. AHB2VPB block diagram

9.8.3 Functional description

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
Peripherals re-partition between VPB1, VPB2 and VPB3 is done according to SC memory configuration (see [Table 158](#)).

VPB1 runs at the same frequency as the AHB bus (typically $f_{hclk} = 104$ MHz) while both VPB2 and VPB3 run at a fixed frequency. This fixed frequency clock is derived from the fixed PLL by a programmable divider and is therefore independent from the clock of the Multi-layer AHB bus.

9.8.3.1 Read access

A read from a VPB slave via any of the AHB2VPB bridge takes 3 pclk1 or pclk2 clock cycles.

9.8.3.2 Write access

A write to a VPB slave via the AHB2VPB takes 3 pclk1 or pclk2 clock cycle. If the write buffer is empty, the Multi-layer AHB bus is released immediately. Otherwise, the effective write delay has to be taken into account. There is no restriction on write/read/write sequences. Due to the early release of Multi-layer AHB bus when write buffer is used, VPB bus error (perr signal) cannot be used.

It must be noted that due to write buffer in AHB2VPB bridge (in addition to write buffer in ARM926EJ-S), the effective write time can be delayed meanwhile the processor continue executing code. This can create issue for interrupt acknowledge in peripheral. To avoid this it is recommended to do a read in the same VPB bus (e.g. the same register which has been written) to force the processor to wait the effective write.

9.8.3.3 Access cycles on Multi-layer AHB bus

Due to the pipelined architecture of AHB, the following table count the used cycles excluding the address phase which take place during the end of the previous cycle.

[Table 169: Access cycles on Multi-layer AHB bus](#)

	VPB1 bus pclk1 = hclk	VPB2/VPB3 bus pclk2 <> hclk
Read	5 hclk	5 hclk + 4 pclk2
Write	write buffer empty write buffer full ^[1]	2 hclk 5 hclk + 4 pclk2

[1] Assuming write occur immediately after write who have filled the write buffer, otherwise, time can be any value between ‘write buffer empty’ and ‘write buffer full’.

9.9 AMU1 to AMU6 - AHB Monitoring Unit

9.9.1 Features

- 6 independent and identical AMU on following Multi-layer AHB bus slaves
 - AMU1 on SDI port #0
 - AMU2 on SDI port #1
 - AMU3 on SDI port #2
 - AMU4 on EBI1
 - AMU5 on SCRAM bank #0
 - AMU6 on SCRAM bank #1
- Each AMU contains a set of 3 memory areas which can be defined with additional triggering facilities
 - Read/write
 - Opcode/data
 - User/privileged mode
 - Master number
- Full report on Interrupt generated by each AMU
 - Aborted transfer type (size, burst and direction)
 - Aborted master
 - Aborted address

9.9.2 Block diagram

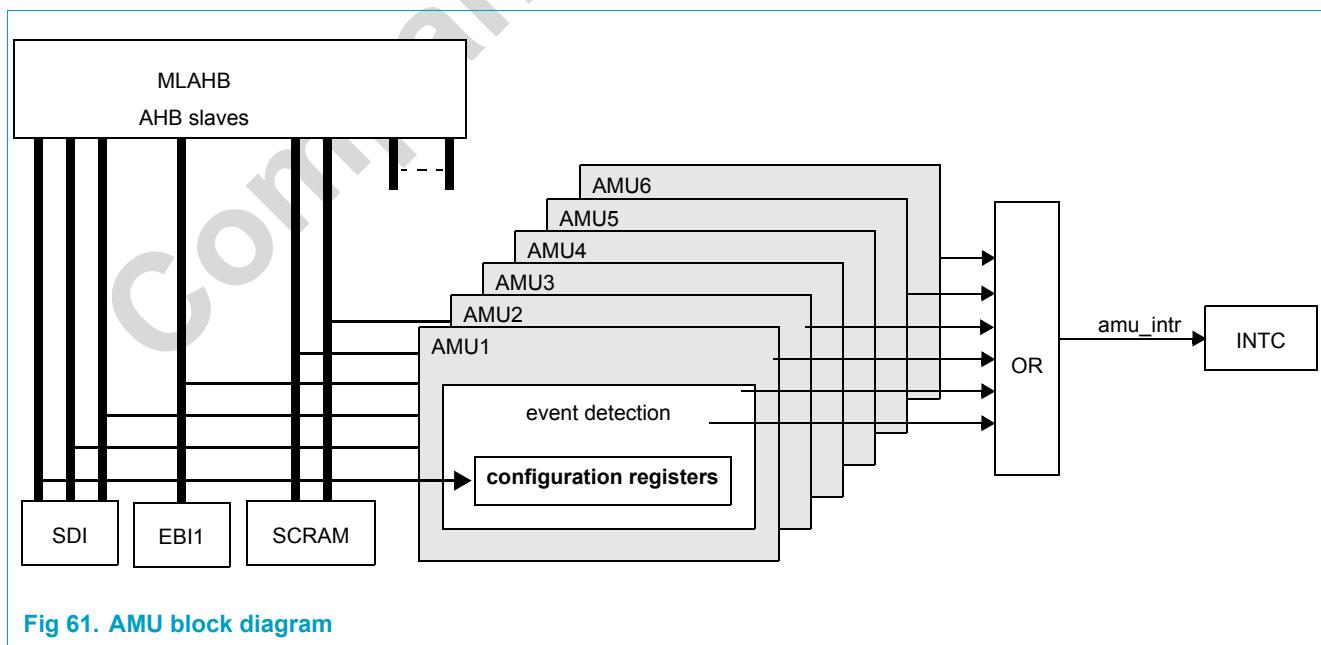


Fig 61. AMU block diagram

9.9.3 Software interface

Table 170: Register overview of AMU

Symbol	Name	I/O	reset
amu_cfg0 to 2	configuration for area 0 to 2	R/W	0x0000 0000
amu_start0 to 2	start address for area 0 to 2	R/W	0x0000 0000
amu_stop0 to 2	stop address for area 0 to 2	R/W	0x0000 0000
amu_area_status	area status register	R/C	0x0000
amu_bus_status	bus status register	R	0x0000 0000
amu_address_status	address status register	R	0x0000 0000

Table 171: Register amu_cfg0 to 2

Bit	Symbol	Access	Value	Description
31 to 18	-	R	0*	reserved
17 to 12	master[5:0]	R/W	0*	master - one bit for each Multi-layer AHB bus master [1]
			1	no detection on master[n] access
			1	detection on master[n] access [2]
11 to 8	hprotmask[3:0]	R/W	0*	each bit enable or disable the corresponding hprot bit
			0*	corresponding hprot bit value is not taken into account
			1	corresponding hprot bit value is taken for comparison
7 to 4	hprot[3:0]	R/W	0*	contain the AHB HPROT value to be compared; each bit is granted by corresponding hprotmask bit; because DMAU hprot[3:1] values are software programmable independently for each channel (See Table 156), this field can be used to define different detection for each DMAU channels:
			0*	hprot0 = 0: opcode fetch
			1	hprot0 = 1: data access
			0*	hprot1 = 0: user access
			1	hprot1 = 1: privileged access
			0*	hprot2 = 0: not buffer able
			1	hprot2 = 1: buffer able
			0*	hprot3 = 0: not cache able
			1	hprot3 = 1: cache able
3 to 2	-	R	0*	reserved
1	wr	R/W	0*	no detection on write access
			1	detection on write access
0	rd	R/W	0*	no detection on read access
			1	detection on read access

[1] Not all masters can access to the slave connected on AMU, please refer to Table 155.

[2] The access is detected when the following equation is true:

$$\begin{aligned}
 & (\text{AHB.hprot}[0] = \text{hprot}[0]) \text{ OR } (\text{hprotmask}[0] = 0) \text{ AND} \\
 & (\text{AHB.hprot}[1] = \text{hprot}[1]) \text{ OR } (\text{hprotmask}[1] = 0) \text{ AND} \\
 & (\text{AHB.hprot}[2] = \text{hprot}[2]) \text{ OR } (\text{hprotmask}[2] = 0) \text{ AND} \\
 & (\text{AHB.hprot}[3] = \text{hprot}[3]) \text{ OR } (\text{hprotmask}[3] = 0) \text{ AND} \\
 & ((\text{AHB.hwrite} = 1) \text{ AND } (\text{wr} = 1)) \text{ OR } ((\text{AHB.hwrite} = 0) \text{ AND } (\text{rd} = 1))
 \end{aligned}$$

Table 172: Register amu_start0 to 2

Bit	Symbol	Access	Value	Description
31 to 0	startadd[31:0] ^[1] ^[2]	R/W	0*	start address of area

[1] Only lowest bits are taken into account, number of effective bits depend on the related slave area size, see [Table 157](#).

[2] For EBI1, the AMU4 make no distinction between all EBI1_CS signals, so only bits 27 to 0 are relevant. This imply that AMU4 will trigger on the address present on the external EBI1 bus as soon one of the EBI1_CS signal is activated.

Table 173: Register amu_stop0 to 2

Bit	Symbol	Access	Value	Description
31 to 0	stopadd[31:0] ^[1] ^[2]	R/W	0*	stop address of area

[1] Only lowest bits are taken into account, number of effective bits depend on the related slave area size, see [Table 157](#).

[2] For EBI1, the AMU4 make no distinction between all EBI1_CS signals, so only bits 27 to 0 are relevant. This imply that AMU4 will trigger on the address present on the external EBI1 bus as soon one of the EBI1_CS signal is activated.

Table 174: Register amu_area_status

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2 to 0	area[2:0] ^[1]	R/C		each bit of this field indicates if an interrupt has been generated in the corresponding area; this register must be explicitly cleared.
			0*	Bit _N = 0 no detection in area N
			1	Bit _N = 1 detection in area N

[1] To avoid wrong status in case of successive detection, when at least a bit in **amu_area_status** register is set, updates of **amu_bus_status** and **amu_address_status** are frozen. This is done independently for each AMU.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 175: Register amu_bus_status^[2]

Bit	Symbol	Access	Value	Description
31 to 18	-	R	0*	reserved
17 to 12	master[5:0]	R	0*	master - one bit for each Multi-layer AHB bus master ^[1]
			1	no detection from master[n]
			1	detection occurred from master[n]
11	-	R	0*	reserved
10 to 8	hburst[2:0]	R		burst type of detected transfer
			0b000*	SINGLE = single transfer
			0b001	INCR = incrementing burst of unspecified length
			0b010	WRAP4 = 4-beat wrapping burst
			0b011	INCR4 = 4-beat incrementing burst
			0b100	WRAP8 = 8-beat wrapping burst
			0b101	INCR8 = 8-beat incrementing burst
			0b110	WRAP16 = 16-beat wrapping burst
			0b111	INCR16 = 16-beat incrementing burst
7 to 4	hprot[3:0]	R	0*	hprot value (see Table 171)
3 to 2	hsize[1:0]	R		size of detected transfer
			0b00*	byte
			0b01	half-word
			0b10	word
			0b11	reserved
1	wr	R	0*	no detection occurred during write
			1	detection occurred during write
0	rd	R	0*	no detection occurred during read
			1	detection occurred during read

[1] Not all masters can access the slave connected on AMU, please refer to [Table 155](#).

[2] To avoid wrong status in case of successive detection, when at least a bit in **amu_area_status** register is set, updates of **amu_bus_status** and **amu_address_status** are frozen. This is done independently for each AMU.

Table 176: Register amu_address_status^[3]

Bit	Symbol	Access	Value	Description
31 to 0	add[31:0] ^{[1][2]}	R	0*	detected address

[1] Only lowest bits are meaningful, number of effective bits depend on the related slave area size, see [Table 157](#).

[2] For EBI1, the AMU4 make no distinction between all EBI1_CS signals, so only bits 27 to 0 are relevant. This imply that AMU4 will trigger on the address present on the external EBI1 bus as soon one of the EBI1_CS signal is activated.

[3] To avoid wrong status in case of successive detection, when at least a bit in **amu_area_status** register is set, updates of **amu_bus_status** and **amu_address_status** are frozen. This is done independently for each AMU.

9.9.4 Functional description

See [13.] for details about AHB protocol .

The aim of the AMU unit is to detect wrong type of access to a particular address region for one or several AHB master(s). The AMBA AHB specification provides information about a bus access with HPROT[3:0] control signals. See [Table 156](#) for available HPROT[3:0] combinations. These signals indicate if the transfer is:

- An opcode fetch or data access
- A privileged mode access or User mode access.

The AHB master being able to manage byte, half-word or word transfers, the comparison between the AHB address bus during a transfer and the address range specified above is always done on bits 31 to 2, but selectively done on bits 0 and 1 depending on the AHB transfer size.

Accesses that trigger AMU are not avoided, but as an interrupt is generated, the software can know the source of the unwanted access. This is useful during system integration.

Interrupt is generated in each AMU when there is at least one bit set in **amu_area_status** register. **amu_intr** signal is generated when there is at least one AMU interrupt set. Interrupt request remains active until all bits in all **amu_area_status** registers have been cleared by SW.

9.9.4.1 System consideration

- Since AMU are multiple on several AHB slave that can be accessed by several masters, software need to check all AMUs in order to know which AMU is in cause.

9.9.5 Application information

CAUTION



The maximum stop address of AMU5 and AMU6 is 0x7FFF. Therefore it does not cover the complete range for monitoring of SCRAM0 and SCRAM1.

9.10 ARB - Arbiter

9.10.1 Features

- flexible priority scheme controllable by software
- 3-bit priority per master

9.10.2 Software interface

Table 177: Register overview of ARB

Symbol	Name	I/O	Reset
arb_cfg0 to 5	ARB configuration register	R/W	0x0000

Table 178: Register arb_cfg0 to 5

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2 to 0	mprio[2:0]	R/W		priority level of master n
			0b000*	lower priority
			...	
			0b111	higher priority

9.10.3 Functional description

Arbitration is required only when different masters want to access the same slave bus.

Each master is assigned a programmable priority coded with 3 bits. Arbitration is done according to the following rules:

1. the bus is granted to the highest priority master requesting the slave bus
2. if multiple masters with the same assigned priority try to access the same slave bus, then the slave bus is uniformly granted to each of these masters in a round-robin fashion
3. if a slave bus is granted to a master and if this master asserts its lock signal, then it keeps the slave bus until the lock signal is deasserted
4. if a master is not requesting any access to a slave bus, then the corresponding layer generates idle cycles

Priorities are programmed with registers **arb_cfg0 to 5** (one for each master, see [Table 155](#)).

9.11 DAIF - Digital to Analog Interface

9.11.1 Features

- Write data to APU at **clk13m**
- Read data from APU registers
- Codec serial data interface (two codecs are supported)
- Dedicated signals for RF, power and reset
- Interrupt generation for low battery and from dedicated analog blocks
- High priority transfer of BMP vector
- Additional features
 - 64 indirect registers read and write
 - 256 indirectly addressable RF registers when interfacing with DRF1902

9.11.2 Block diagram

The interface between the DPU and the APU is done with a serial link and some additional dedicated signals. The interface is composed of two blocks: DAIF on the DPU and ADIF on the APU (see [Figure 62](#)). The signals between the two processing units are described in [Table 179](#)

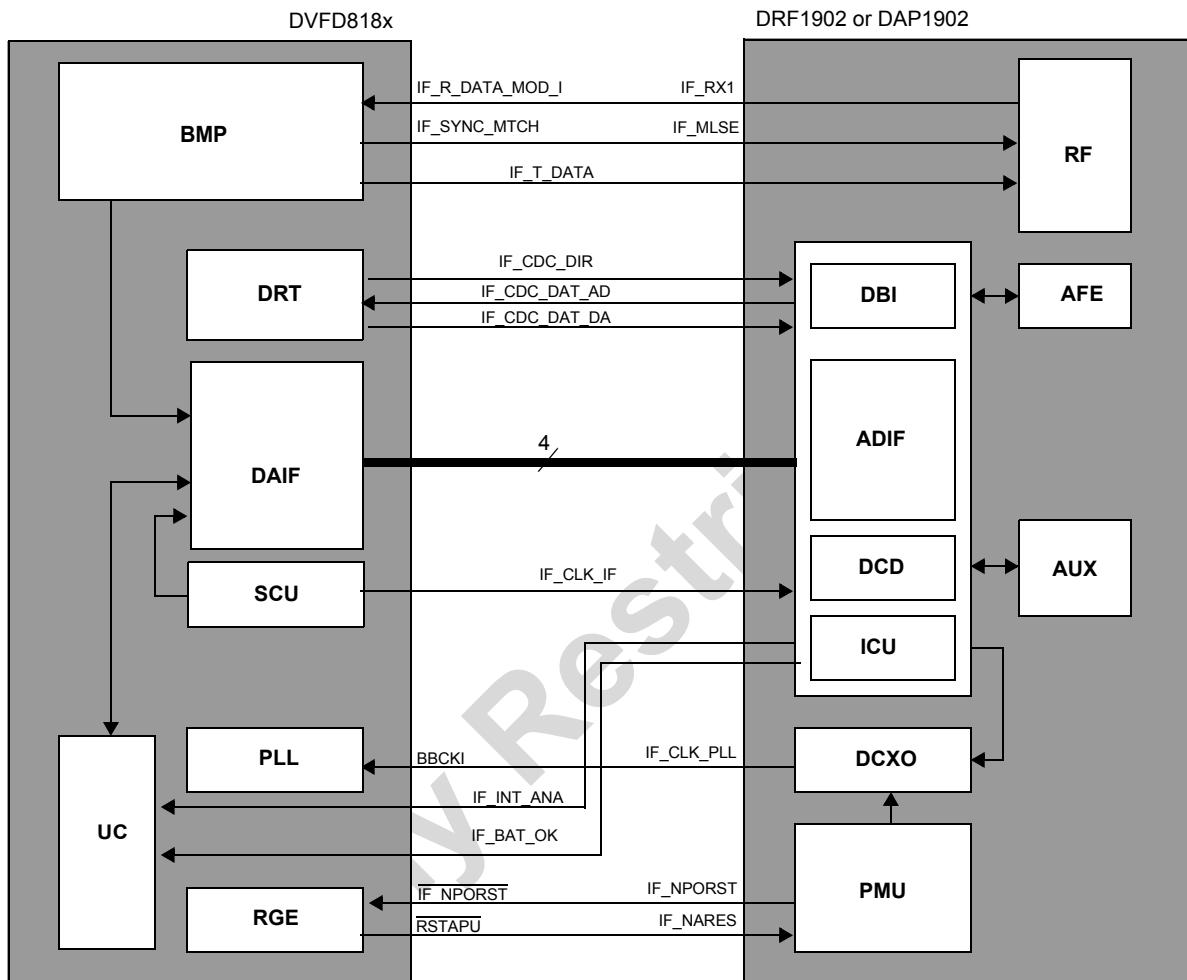


Fig 62. Digital - Analog interface block diagram for DVFD8185 and DVFD8187 with external APU

9.11.3 Hardware interface

Table 179:DAIF interface pin overview

PIN #	Name	I/O	Description
1	IF_BAT_OK	I	Output from low voltage detector logic. Interrupts the ARM with <i>int_pwr_fail</i> at both rising and falling edges.
2	IF_CDC_DAT_AD	I	Data signal from the Sigma-Delta converters (ATC) to the Digital Decimating Filters (DDF).
3	IF_CDC_DAT_DA	O	Data signal from the Digital Noise Shapers (DNS) to the DA converters (ARD).
4	IF_CDC_DIR	O	Synchronisation and direction control signal of the audio bit stream frames.
5	IF_CLK_IF	O	clock for data transmission (equal to clk13m)
7	BBCKI	I	PLL input clock from XTAL after pre-devider. Connects to IF_CLK_PLL on the APU
8	IF_DATA_AD	I	4 wire interface to control the APU.
9	IF_DATA_DA	O	
10	IF_EN_AD	I	
11	IF_EN_DA	O	

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 179:DAIF interface pin overview...continued

PIN #	Name	I/O	Description
12	IF_INT_TS_INT_ANA	I	Interrupt signal from APU (touchscreen, voice monitor, on-off, charge detect) - connects to int_ana on APU and DPU.
13	RSTAPU	O	There are different reset conditions generated on the DPU. This signal transfers this information to the APU. Connects to IF_NARES on the APU.
14	RSTPO or (IF_NPORST)	I	The power on reset block is located on the APU. This signal sends the power on reset to the WDRU on the DPU.
16:15	IF_R_DATA_MOD ^[1]	I	RF receive data signals from ABS (OM6115) or in-phase components from RFAPU
19	IF_T_DATA ^[1]	O	RF transmit data serial from BMP.

[1] Function not supported at DVFD8187

9.11.4 Software interface

For the register map and the register description consult datasheets of DAP1902 or DRF1902.

9.11.5 Functional description

The DAIF is a programmable interface between the DPU and the APU. The DAIF also carries power and reset signals, RF control and data signals and the clock from the crystal oscillator as reference for the system PLL's.

9.11.5.1 Serial data transfer

There are two types of data that have to be transferred from the DPU to the APU over the serial interface:

- BMP signals (from MCU and RFCU). There are several tasks (e.g. fast antenna diversity, RSSI measurement) that have to start at a given bit counter value. To initiate/finish these tasks, 8 free programmable signals are available. A list of all signals is given in [Table 180](#).
- Analog Control Registers. The register view for external APU (DAP1902 or DRF1902) should be taken from the corresponding datasheet.

Table 180: MCU and RFCU signal to control analog functions

BIT #	BMP vector usage on DRF1902		Function on APU
7	<i>start_ad</i>		Start AD conversion cycle
6	reserved		-
5	radio_en		Enables RF section
4	slot_control		slot timing control signal
3	sym_sync_time	<i>start_fad</i>	Symbol sync time
2	rssi_one_shot	<i>en_ref_clk</i>	RSSI one shot
1	reserved	<i>en_abs</i>	-
0	reserved	<i>en_mod</i>	-

Control Registers and BMP vector are serially sent to the APU (see [Figure 63](#)). The BMP vector has a higher priority because it is time critical. If the BMP programs an event to be sent to the APU during a register transfer, the register transfer is interrupted and the BMP vector is sent followed by a HW controlled retransmission of the register data. The registers can be accessed (read and write) by the system controller and are duplicated on the APU.

The time needed for an uninterrupted serial transfer of an analog control register is 20 periods of **clk13m** (e.g. 1.45 us for DECT). The time needed for a BMP vector serial transfer is 11 periods of **clk13m** (e.g. 0.80 us for DECT from bit counter position to signal activation). These values are valid if the **PLL_FIX** is on **cgufixcon pllfixen = '1'**.

A transfer is needed from the APU to the DPU for the results of an A/D conversion (for RSSI in **daif_rssio**, for VBAT, VANLI and AD0..AD3 in **daif_raado**). The transfer is initiated on completion of the A/D conversion. The results are stored in registers on the DPU that are readable by the system controller. When the result is available on the DPU, an interrupt (**int_ad**) is issued.

The used antenna is flagged in register **daif_rantheant_used**, which is transferred to the DPU.

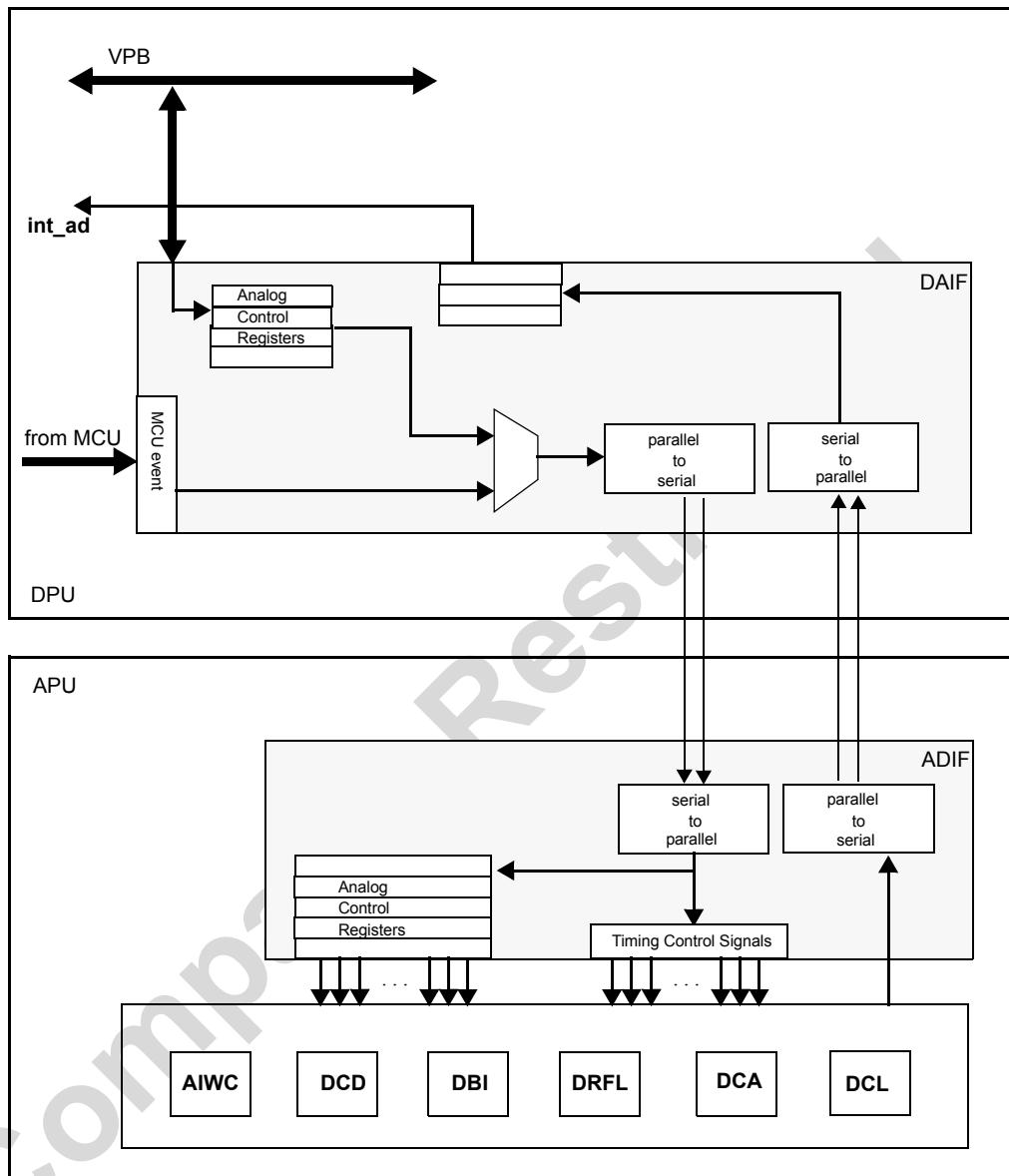


Fig 63. Control of the APU

9.11.5.2 APU register read

The system controller read access to a register on the APU which is not automatically updated is done via a write action to the register **daif_rapu** with the required address of the register on the APU. The register write action starts a read action from the requested address on the APU via the serial interface. The read register content and address is stored in the register **daif_rapu** and can be read out by the system controller after the transmission is finished. The bit **daif_rapu.new_value** indicates that a request has been executed.

Refer to DAP1902 or DRF1902 datasheet hardware specification section for additional details regarding access to the RF and Analog Register Space.

9.11.5.3 APU register write

Writing a DAIF register in the register within index range 0 to 40 and 42 to 62 will automatically result in an update of the corresponding register in the ADIF.

Two consecutive writes to the same register must be separated by at least 1.1 µs to guarantee a correct transfer of register contents and / or the corresponding transfer pending flag must be checked.

For DVFD8185 and DVFD8187 an additional register space is accessible using the WAPU register.

A write action in DAIF to the WAPU register will start a write in ADIF to the register with address WAPU[14:8] with data WAPU[7:0]; when a write is on-going the flag WAPU[WAPU_TRAN], i.e. WAPU[15], is set. A next write action to the WAPU register can only be started when WAPU[WAPU_TRAN] is reset.

Refer to DAP1902 or DRF1902 datasheet hardware specification section for additional details regarding access to the RF and Analog Register Space.

Since the address field of the WAPU register is 7 bits long, 128 registers can be addressed, but only 100 have been defined; registers 105 and 127 are reserved and cannot be used; registers from 117 to 127 are reserved for DAIF itself.

9.11.5.4 Reset states

Some of the reset states of the registers on the DPU differ from the ones on the RF19APU. To prevent unexpected results when doing a read-modify-write access of a DAIF register, it is recommended to synchronize the registers on the DPU with the contents of the RF19APU after reset.

The reset state of the DAP1902 and DRF1902 are defined at the detailed register view at their hardware specification.

9.11.6 Application information

Two consecutive writes to the same DAIF register should respect an interval of more than two transfer times.

The bit **cgusleepsc.daifslen** must be set to allow wake up in sleep mode from SVM (int_pwr_fail).

9.12 DMAU - Direct Memory Access Unit**9.12.1 Features**

- Thirty-two DMA request lines 1
- Twelve DMA channels; each channel can support a unidirectional transfer 2
- Two AHB master capable of simultaneous transfer 3
- Single DMA and burst DMA request signals; each peripheral connected to the DMAU controller can assert either a burst DMA request or a single DMA request; programming the DMAU controller sets the DMA burst size 4
- Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers 5
- Scatter or gather transfer is supported through the use of linked lists: source and destination areas do not need to occupy consecutive areas of memory 6
- Hardware DMA channel priority 7
- Incrementing or non-incrementing addressing for source and destination 8
- Programmable DMA burst size 9
- Internal 4-word FIFO per channel 10
- Supports 8, 16 and 32-bit wide transactions 11
- Combined DMA error and DMA terminal count interrupt requests 12
- Interrupt masking; the DMA error and DMA terminal count interrupt requests can be masked 13
- Raw interrupt status; the DMA error and DMA count raw interrupt status can be read prior to masking 14

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.12.2 Block diagram

The DMAU provides two AHB master and a AHB slave port plus the complete hardware for the DMA channels processing.

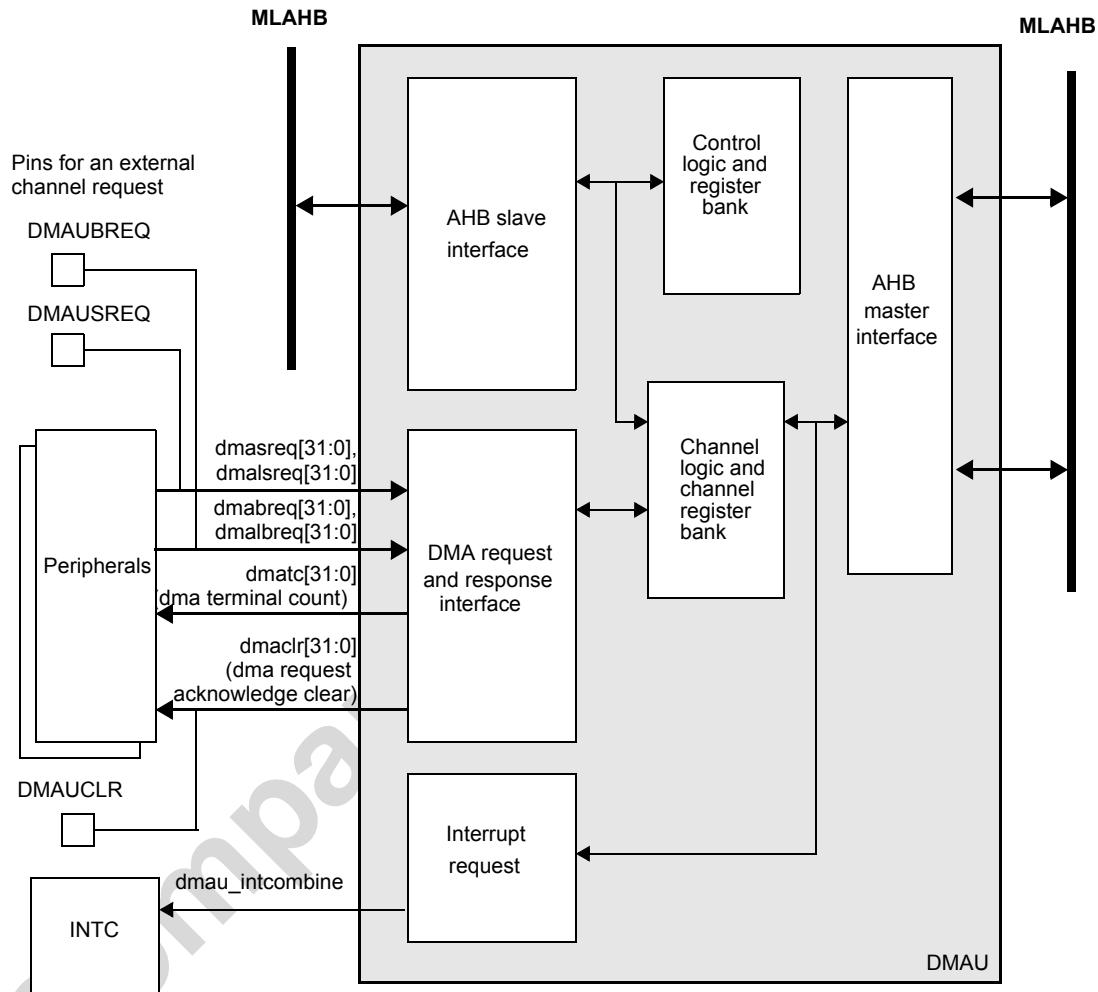


Fig 64. DMAU block diagram

9.12.3 Hardware interface

Table 181: DMAU pins overview

Symbol	Name	I/O	Description
DMAUSREQ	DMAUSREQ pin	I	external channel single request
DMAUBREQ	DMAUBREQ pin	I	external channel burst request
DMAUCLR	DMAUCLR pin	O	external channel request clear

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.12.4 Software interface

9.12.4.1 DMAU register overview

Table 182: Register overview of DMAU

Symbol	Name	I/O	Reset
dmau_int_status	DMAU interrupt status	R	0x00
dmau_int_tc_status	DMAU interrupt terminal count status	R	0x00
dmau_int_tc_clear	DMAU interrupt terminal count clear	W	0x00
dmau_int_error_status	DMAU interrupt error status	R	0x00
dmau_int_error_clear	DMAU interrupt error clear	W	0x00
dmau_raw_int_tc_status	DMAU raw interrupt terminal status	R	0x00
dmau_raw_int_error_sta tus	DMAU raw interrupt error status	R	0x00
dmau_enbld_chns	DMAU enable channel	R	0x00
dmau_soft_breq	DMAU software generated burst request	R/S	0x0000 0000
dmau_soft_sreq	DMAU software generated single request	R/S	0x0000 0000
dmau_configuration	DMAU configuration	R/W	0x00
dmau_sync	DMAU synchronization of the request signals	R/W	0x0000 0000
Same for all 12 channels (x = 0 to 11)			
dmau_cx_src_addr	DMAU channel x source address	R/W	0x0000 0000
dmau_cx_dest_addr	DMAU channel x destination address	R/W	0x0000 0000
dmau_cx_lli	DMAU channel x link list	R/W	0x0000 0000
dmau_cx_control	DMAU channel x control	R/W	0x0000 0000
dmau_cx_configuration	DMAU channel x configuration	R/W	0x0000 0000

9.12.4.2 Interrupt status register

The **dmau_int_status** register shows the status of the interrupts after masking. The request can be generated from either the error or terminal count interrupt requests.

Table 183: Register **dmau_int_status**

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	intstatus[11:0]	R		status of the DMA interrupts after masking
			0*	no request active on DMA channel i
			1	DMA channel i interrupt request active

9.12.4.3 Interrupt terminal count status register

The **dmau_int_tc_status** register indicates the status of the terminal count after masking.

This register must be used in conjunction with the **dmau_int_status** register when the interrupt request **dmau_intcombine** is activated in the interrupt controller INTC.

Table 184: Register dmau_int_tc_status

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	intstatus[11:0]	R		interrupt terminal count request status
			0*	terminal count not reached on DMA channel i
			1	terminal count reached on DMA channel i

9.12.4.4 Interrupt terminal count clear register

The **dmau_int_tc_clear** register is used to clear a terminal count interrupt request.

Table 185: Register dmau_int_tc_clear

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	inttcclear[11:0]	W		terminal count request clear
			0*	no effect
			1	clear corresponding bit i in the status register dmau_int_tc_status

9.12.4.5 Interrupt error status register

The **dmau_int_error_status** register indicates the status of the error request after masking.

This register must be used in conjunction with the **dmau_int_status** register when the interrupt request **dmau_intcombine** is activated in the interrupt controller INTC.

Table 186: Register dmau_int_error_status

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	interrstatus[11:0]	R		error interrupt status
			0*	no error on DMA channel i
			1	error on DMA channel i

9.12.4.6 Interrupt error clear register

The **dmau_int_error_clear** register is used to clear the error interrupt requests.

Table 187: Register dmau_int_error_clear

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	interrclr[11:0]	W	0*	interrupt error clear
			0*	no effect
			1	clear corresponding bit i in the status register dmau_int_error_status

9.12.4.7 Raw interrupt terminal count status register

The **dmau_raw_int_tc_status** register indicates which DMA channels are requesting a transfer complete (terminal count interrupt) prior to masking by **dmau_cx_configuration.itc**.

Table 188: Register dmau_raw_int_tc_status

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	rawinttcstatus[11:0]	R	0*	status of the terminal count interrupt prior to masking
			0*	terminal count not reached on DMA channel i
			1	terminal count reached on DMA channel i

9.12.4.8 Raw error interrupt status register

The **dmau_raw_int_error_status** register indicates which DMA channels are requesting an error interrupt prior to masking by **dmau_cx_configuration.ie**.

Table 189: Register dmau_raw_int_error_status

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	rawinterrrorstatus[11:0]	R	0*	status of the error interrupt prior to masking
			0*	no error on DMA channel i
			1	error on DMA channel i

9.12.4.9 Enabled channel register

The **dmau_enbld_chns** register indicates which DMA channels are enabled, as indicated by the enable bit in the **dmau_cx_configuration** register.

Each bit is cleared on completion of the DMA transfer.

Table 190: Register dmau_enbld_chns

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	enabledchannels [11:0]	R	0*	channel enable status
			0*	channel i not enabled
			1	channel i enabled

9.12.4.10 Software burst request register

The **dmau_soft_breq** register allows DMA burst requests to be generated by software.

A DMA request can be generated for each source by setting the corresponding bit in the register. Each bit is cleared when the transaction has completed. DMA requests can be generated from either a peripheral or the software request register. Writing zeroes to this register has no effect.

Reading the register indicates which sources are requesting burst DMA transfers.

Table 191: Register dmau_soft_breq

Bit	Symbol	Access	Value	Description
31 to 0	softbreq[31:0]	R/S	0*	software burst request

9.12.4.11 Software single request register

The **dmau_soft_sreq** register allows DMA single requests to be generated by software.

A DMA request can be generated for each source by setting the corresponding bit in the register. Each bit is cleared when the transaction has completed. DMA requests can be generated from either a peripheral or the software request register. Writing zeroes to this register has no effect.

Reading the register indicates which sources are requesting single DMA transfers.

Table 192: Register dmau_soft_sreq

Bit	Symbol	Access	Value	Description
31 to 0	softsreq[31:0]	R/S	0*	software single request

9.12.4.12 Configuration register

The **dmau_configuration** register is used to configure the operation of the DMAU controller.

The endianess of the each AHB master interfaces can be altered by writing to the endian bit of this register. The AHB master interfaces are set to Little-endian mode on reset. Although it is most of the time useless, it is possible to have different AHB endianess.

Table 193: Register dmau_configuration

Bit	Symbol	Access	Value	Description
31 to 3	-	R	0*	reserved
2	endian1	R/W		AHB2 endianess configuration
			0*	Little-endian mode
			1	Big-endian mode [1]
1	endian0	R/W		AHB1 endianess configuration
			0*	Little-endian mode
			1	Big-endian mode [1]
0	e	R/W		DMAU controller enable
			0*	disabled
			1	enabled

[1] Don't use

9.12.4.13 Synchronization register

The **dmau_sync** register is used to enable or disable synchronization logic for the DMA request signals. A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests.

Note: synchronization logic must be used when the peripheral generating the DMA request runs on a different clock than the DMAU controller. For peripherals running on the same clock as the DMAU controller disabling the synchronization logic improves the DMA request response time.

Table 194: Register **dmau_sync**

Bit	Symbol	Access	Value	Description
31 to 0	dmasync[31:0]	R/W		DMA synchronization logic for DMA request signals
			0*	synchronization logic enabled
			1	synchronization logic disabled

9.12.4.14 Channel registers

The channel registers are used to program a DMA channel. These registers consist of:

- Twelve **dmau_cx_src_addr** registers
- Twelve **dmau_cx_dest_addr** registers
- Twelve **dmau_cx_lli** registers
- Twelve **dmau_cx_control** registers
- Twelve **dmau_cx_configuration** registers.

Note: the notation cx means one of the twelve registers c0 to c11.

When performing scatter/gather DMA the first four registers are automatically updated.

Table 195: Register **dmau_cx_src_addr**

Bit	Symbol	Access	Value	Description
31 to 0	srcaddr[31:0]	R/W	0*	DMA source address

The twelve **dmau_cx_src_addr** registers contain the current source address (byte-aligned) of the data to be transferred.

Each register is programmed by the firmware before the appropriate channel is enabled. When the DMA channel is enabled this register is updated.

- The source address is incremented
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. It is intended to be read only when the channel is stopped, to indicate the source address of the last item read.

Note: the source and destination addresses must be aligned to the source and destination widths.

Table 196: Register dmau_cx_dest_addr

Bit	Symbol	Access	Value	Description
31 to 0	destaddr[31:0]	R/W	0*	DMA destination address

The twelve **dmau_cx_dest_addr** registers contain the current destination address (byte-aligned) of the data to be transferred.

Each register is programmed by the firmware before the appropriate channel is enabled. When the DMA channel is enabled this register is updated.

- As the destination address is incremented
- By following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. It is intended to be read only when the channel is stopped, to indicate the destination address of the last item read.

Table 197: Register dmau_cx_lli

Bit	Symbol	Access	Value	Description
31 to 2	lli[29:0]	R/W	0*	linked list item; bits [31:2] of the address for the next LLI; address bits [1:0] are 0
1	-	R/W	0*	reserved
0	lm	R/W		AHB master select for loading next LLI
			0*	AHB1
			1	AHB2

The twelve **dmau_cx_lli** registers contain a word aligned address of the next Linked List Item (LLI).

If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled after all DMA transfers associated with it are completed.

Note: programming this register when the DMA channel is enabled is not allowed and can result in an undefined state.

Note: as the LLI data structure contains 4 words, it is better to align it to a multiple of 4 words to optimize the loading.

Table 198: Register dmau_cx_control

Bit	Symbol	Access	Value	Description
31	i	R/W	0*	terminal count interrupt enable bit ; it controls whether or not the current LLI is expected to trigger the terminal count interrupt
				0*: no terminal count interrupt
				1: terminal count interrupt
30 to 28	prot[2:0]	R/W	0*	protection bits , see Section 9.12.4.15 “Protection and access informations” on page 247
27	di	R/W	0*	destination increment
				0*: no increment
				1: destination address is incremented after each transfer
26	si	R/W	0*	source increment
				0*: no increment
				1: source address is incremented after each transfer
25	d	R/W	0*	Destination AHB master select
				AHB1
				AHB2
24	s	R/W	0*	Source AHB master select
				AHB1
				AHB2
23 to 21	dwidth[2:0]	R/W	0*	destination transfer width ; transfers wider than the AHB master bus width are illegal; the source and destination widths can be different; the hardware automatically packs and unpacks the data as required (see Table 200)
20 to 18	swidth[2:0]	R/W	0*	source transfer width ; transfers wider than the AHB master bus width are illegal; the source and destination widths can be different; the hardware automatically packs and unpacks the data as required (see Table 200)
17 to 15	dbsize[2:0]	R/W	0*	destination burst size ; indicates the number of transfers per destination burst request; this value corresponds to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size (see Table 199)
14 to 12	sbsize[2:0]	R/W	0*	source burst size ; indicates the number of transfers per source burst; this value corresponds to the burst size of the source peripheral, or if the source is memory, to the memory boundary size (see Table 199)
11 to 0	transfersize[11:0]	W	0*	transfer size : indicates the number of transfers to perform to source (in term of source width) when the DMAU controller is the flow controller; if the source transfer width is lesser than that of destination, then the transfersize value multiplied by the source transfer width should be an integral multiple of the destination transfer width; if the DMAU controller is not the flow controller the transfer size value is not used and must be set to 0; reading this field when the transfer is started return the number of destination transfer remaining; this value must be used only for debugging purposes

The twelve **dmau_cx_control** registers contain DMA channel control information such as the transfer size, burst size, and transfer width.

Each register is programmed by the firmware before the appropriate channel is enabled.

The transfer-size value should be set before the channel is enabled. Transfer size is updated as data transfers are completed on the destination bus.

Reading the register when the channel is active does not provide useful information. It is intended to be read only when the channel is stopped.

When the channel is enabled, the register is updated following the linked list after the transfer of a block of data.

Table 199 shows the value of the dbsize or sbsize bits and the corresponding burst sizes.

Table 199:dbsize and sbsize burst transfer request sizes

Bit value of dbsize and sbsize	Source or destination burst transfer request size ^[1]
0b000	1
0b001	4
0b010	8
0b011	16
0b100	32
0b101	64
0b110	128
0b111	256

[1] All peripherals must be programmed with burst size of 1 except FCI which use a burst size of 8 and FIR which can use a burst size of 1 or 4

Table 200 shows the value of the swidth or dwidth bits and the corresponding width.

Table 200:swidth and dwidth values

Bit value of swidth and dwidth	Source or destination width
0b000	byte (8-bit)
0b001	halfword (16-bit)
0b010	word (32-bit)
0b011	reserved
0b100	reserved
0b101	reserved
0b110	reserved
0b111	reserved

9.12.4.15 Protection and access informations

AHB access information is provided to the source and destination peripherals when a transfer occurs. The transfer information is provided by programming the DMA channel (the protection bit of the **dmau_cx_control** register). These bits are programmed by software and peripherals can use this information if necessary. Three bits of information are provided. This information can be traced on the HDP.

9.12.4.16 Channel configuration registers

Table 201: Register dmau_cx_configuration

Bit	Symbol	Access	Value	Description
31 to 19	-	R	0*	reserved
18	h	R/W		halt ; this value can be used with the a and e bits to disable a DMA channel without data loss
			0*	allow DMA requests
			1	ignore further source DMA requests; the contents of the channels FIFO are drained
17	a	R		active ; this value can be used with the h and e bits to disable a DMA channel without data loss
			0*	there is no data in the FIFO of the channel
			1	the FIFO of the channel has data
16	l	R/W		lock
			0*	no locked transfer
			1	reserved, should not be used
15	itc	R/W		terminal count interrupt mask
			0*	mask out the terminal count interrupt of the relevant channel
			1	no mask
14	ie	R/W		interrupt error mask
			0*	mask out the error interrupt of the relevant channel
			1	no mask
13 to 11	flowcntrl[2:0]	R/W	0*	flow control and transfer type ; this value is used to indicate the flow controller and transfer type; the flow controller can be the DMAU controller, the source peripheral, or the destination peripheral; the transfer type can be either memory-to-memory, memory-to-peripheral, peripheral-to-memory or peripheral-to-peripheral (see Table 202)
10 to 6	destperiph[4:0]	R/W	0*	destination peripheral ; this value selects the DMA destination request peripheral; this field is ignored if the destination of the transfer is memory (see Table 203)
5 to 1	srcperiph[4:0]	R/W	0*	source peripheral ; this value selects the DMA source request peripheral; this field is ignored if the source of the transfer is memory (see Table 203)
0	e	R/W	0*	channel enable ; reading this bit indicates whether a channel is currently enabled or disabled: read 0 = channel disabled read 1 = channel enabled write 0 = the current AHB transfer (if one is in progress) completes and the channel is disabled; any data in the channels FIFO is lost write 1 = enable the channel

The twelve **dmau_cx_configuration** registers are used to configure the DMA channel.

The registers are not updated when a new LLI is requested.

The channel enable bit status can also be read in the **dmau_enbld_chns** register. 1
 Restarting the channel by setting the channel enable bit without initializing the 2
 channel is not allowed: the channel must be fully re-initialized. 3
 The channel is disabled and channel enable bit cleared, when the last LLI is reached 4
 or if a channel error is encountered. 5
 If a channel has to be disabled without losing data in a channel FIFO, the halt bit **h** 6
 must be set so that further DMA requests are ignored. The active bit **a** must then be 7
 polled until it is reset, indicating that there is no data left in the channels FIFO before 8
 clearing the channel enable bit. 9
 If a channel has to be disabled without losing data in a channel FIFO, the halt bit **h** 10
 must be set so that further DMA requests are ignored. The active bit **a** must then be 11
 polled until it is reset, indicating that there is no data left in the channels FIFO before 12
 clearing the channel enable bit. 13
Table 202 describes the bit values of the three flow control and transfer type bits. The 14
 flow controller manages the number of data to transfer. For most of the peripherals 15
 when the number of data is known by the software, the DMAU is the flow controller. 16
 When the number of data is only known by the peripheral (either source or 17
 destination), the peripheral is the flow controller. In this case the 18
dmau_cx_control.transfersize[11:0] is not used. When the peripheral indicates to 19
 the DMAU that the last data has been transferred, a terminal count interrupt is 20
 generated or the next item in the LLI is processed. 21

Table 202: Control and transfer type bits

Bits	Transfer type	Controller
0b000	memory to memory	DMAU
0b001	memory to peripheral	DMAU
0b010	peripheral to memory	DMAU
0b011	source peripheral to destination peripheral	DMAU
0b100	source peripheral to destination peripheral	destination peripheral ^[1]
0b101	memory to peripheral	peripheral ^[1]
0b110	peripheral to memory	peripheral ^[1]
0b111	source peripheral to destination peripheral	source peripheral ^[1]

[1] Only FCI, SPI and USIM can be used as flow controller, all other peripherals need to use DMAU as flow controller.

9.12.5 Functional description

For data transfers, the DMAU controller is connected to two AHB master buses, it is able to function as a master simultaneously on each bus. The DMAU controller also provides one AHB slave port for internal register access. Up to 32 DMA requestors are possible, but only 12 peripherals can be simultaneously active (DMA transfer on-going) at any given time. When a peripheral request is served by a DMAU channel, the corresponding data interrupt should be disabled in the peripheral or in the INTC. The interrupt **dmau_intcombine** should be enabled in INTC.

Table 203:ARM DMAU peripherals requesters

DMA request #	Device	Flow control signals	Flow controller
31 to 21	reserved	-	-
20	FIR	single, burst, last single and last burst ^[4]	peripheral
19	reserved	-	-
18	reserved	-	-
17	reserved	-	-
16	reserved	-	-
15	external	single or burst	DMAU
14	reserved	-	-
13	reserved	-	-
12	reserved	-	-
11	reserved	-	-
8	IIS	single and burst ^[1]	DMAU
7	reserved	-	-
5	SPI1	single and burst ^{[1][3]}	DMAU or peripheral
4	UART2 RX	single and burst ^[1]	DMAU
3	UART2 TX	single and burst ^[1]	DMAU
2	UART1 RX	single and burst ^[1]	DMAU
1	UART1 TX	single and burst ^[1]	DMAU

[1] When such peripheral is used as source, single is used. When such peripheral is used as destination, burst is used. Because burst size is programmed to 1 (see [Table 199](#)), in both case, 1 transfer is made for each request.

[2] As USIM operates as flow controller, the USIM **done** interrupt is used to indicates the end of transfer. The DMA must then be cleanly disabled to avoid loss of data (see [Section 9.12.6.5 "Disabling a DMA channel"](#)).

[3] When the SPI operates as flow controller, the SPI **eot** interrupt is used to indicates the end of transfer. The DMA must then be cleanly disabled to avoid loss of data (see [Section 9.12.6.5 "Disabling a DMA channel"](#)).

[4] As FCI or FIR operates as flow controller, they indicates to the DMAU the transfer of the last data in single or burst mode.

9.12.5.1 Latency

Each stream is supported by a dedicated 4 x 32-bits FIFO and hardware channel, including source and destination controllers. This allows better latency and simplifies the control logic.

9.12.5.2 Slave

All transactions on the AHB slave programming bus of the DMAU controller are 32-bit wide. This eliminates endian issues when programming the DMAU controller. A register block is used to program the DMAU controller using an AMBA AHB slave interface.

9.12.5.3 Master

The DMAU controller contains a two full AHB masters capable of dealing with all types of AHB transactions, including:

- Error response from slaves
- Locked transfers for source and destination of each stream
- Setting of protection bits for transfers on each stream.

An error during a DMA transfer is flagged directly by the peripheral by asserting an error response on the AHB bus during the transfer. The DMAU controller automatically disables the DMA stream after the current transfer has completed, and can optionally generate an error interrupt to the CPU. This error interrupt can be masked.

9.12.5.4 DMA request priority

DMA channel priority is fixed, with DMA channel 0 having the highest priority and DMA channel 11 having the lowest priority. If the DMAU controller is transferring data for a lower priority channel and then a higher priority channel goes active, it completes the number of transfers (up to the channel FIFO size, i.e. 4 x 32-bits, 8 x 16-bits or 16 x 8-bits) delegated to the master interface by the lower priority channel before transferring data for the higher priority channel.

The two lowest priority channels (channels 10 and 11) in the DMAU controller are designed so that they cannot saturate the AHB bus. If one of these lower priority channels goes active, the DMAU controller relinquishes control of the bus (for a bus cycle), after four transfers (irrespective of the size of transfer). This allows other AHB masters to access the bus.

It is recommended that memory-to-memory transactions use one of these low priority channels. Otherwise other (lower priority) AHB bus masters are prevented from accessing the bus during DMAU controller memory-to-memory transfer.

9.12.5.5 Data transaction

The DMAU controller allows peripheral-to-memory, memory-to-peripheral, peripheral-to-peripheral, and memory-to-memory transactions.

Each DMA stream is configured to provide unidirectional DMA transfers for a single source and destination. For example, a bi-directional serial port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral.

9.12.6 Application information

General system/software assumptions

- There should not be any write-operation to channel registers in an active channel, after the channel enable is made high. If any DMAU channel parameters are required to be reprogrammed, the reprogramming should be done after disabling the DMAU channel cleanly. 1
2
3
4
5
6
7
8
- If the source width is lesser than that of destination, then the **transfersize** value multiplied by the source width should be an integral multiple of the destination width. 9
10
11
- When the source peripheral is the flow controller, and the source width is less than the destination width, the number of transfers performed by the source peripheral (before asserting an last single/burst request or disabling the transfer) should be such that the number of transfers multiplied by the source width should be an integral multiple of the destination width. If the above is violated, data could get stuck and lost in the FIFO and the results are unpredictable. The transfer can be aborted by disabling the relevant DMAU channel. 12
13
14
15
16
17
18
- After the software disables a channel by clearing the bit **dmau_cx_configuration.e** [x = 0 to 11 (channel number)], it should re-enable the bit only after it has polled a 0 in the corresponding **dmau_enbld_chns** (enabled channels register) bit. This is because the actual disabling (indicated by **dmau_enbld_chns**) does not immediately happen with the clearing of **dmau_cx_configuration.e** bit. The latency of the ongoing AHB burst needs to be accommodated. 19
20
21
22
23
24
25
26
- The **dmau_cx_lll.lll** [x = 0 to 11 (channel number)] field should not indicate an address greater than 0xFFFFFFFF0. Otherwise, the four-word LLI burst will wrap over at 0x00000000 and the LLI-data-structure will not be in contiguous memory location. 27
28
29
30
- If **dmau_cx_control.transfersize** [x = 0 to 11 (channel number)] field is programmed as zero and the DMAU is the flow controller (in other flow-control modes, transfer-size field has got no meaning), then the channel does not initiate any transfers. It is the responsibility of the programmer to disable the channel by writing into the **dmau_cx_configuration.e** [x = 0 to 11 (channel number)] bit and re-program the channel again. 31
32
33
34
35
36
- The normal read-write tests should not be run on **dmau_cx_control** [x = 0 to 11 (channel number)] register, as the **transfersize** field is not a typical write and read-back register field. While writing, **transfersize** bit-field is like a control-register (in the sense that it determines how many transfers the DMAU should perform). However, during read-back, **transfersize** is expected to behave like a status-register. It is supposed to return the number of transfers remaining (in terms of source width). So when **transfersize** is read back, the number of destination-transfer-completed multiplied by a factor is returned. Hence the same physical-register is not being written-into and read-from. Thus normal write-and read-back tests are not applicable. 37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

- For destination flow controlled case (peripheral-to-peripheral transfer) with **dwidth < swidth**, the number of data bytes requested by the destination peripheral must be an integral multiple of **swidth** (expressed in bytes). If this is not ensured, the DMAU may fetch more data from the source peripheral than is absolutely required, resulting in data loss. 1
2
3
4
5
- At the end of accesses corresponding to low-priority channels (10 and 11), an idle cycle is inserted on the AHB bus to allow other masters access to the bus. This ensures that a low-priority channel does not hog the bus. It does, however, mean that the bus could be occupied by transactions corresponding to a low priority for up to 16 beats in the worst case. This applies to all transfer configurations, including memory-to-memory transfers. 6
7
8
9
10
11
12

9.12.6.1 Enabling the DMAU controller

To enable the DMAU controller, set the DMAU enable bit **dmau_configuration.e**. 13
14
15

9.12.6.2 Disabling the DMAU controller

To disable the DMAU controller, first read the **dmau_enbld_chns** register and ensure that all the DMA channels have been disabled, if not disable them, then disable the DMAU controller by resetting the DMAU enable bit **dmau_configuration.e**. 18
19
20
21

9.12.6.3 Enabling a DMA channel

To enable a DMA channel set the channel enable bit in the corresponding DMA channel configuration register. The channel must be fully initialized before it is enabled. Additionally, the enable bit of the DMAU controller must be set before any channel is enabled. 22
23
24
25
26
27

9.12.6.4 Halting a DMA channel

Set the halt bit in the relevant DMA channel configuration register. The current source request is serviced; any further source DMA requests are ignored until the halt bit is cleared. 28
29
30
31
32
33

9.12.6.5 Disabling a DMA channel

A DMA channel can be disabled in three ways: 34
35
36

1. Clear the channel enable bit in the corresponding channel configuration register; the current AHB transfer (if one is in progress) completes and the channel is disabled; any remaining data in the FIFOs is lost 37
38
39
2. Set the halt bit in the relevant channel configuration register; this causes any further DMA requests to be ignored; poll the active bit in the relevant channel configuration register until it reaches 0; this bit indicates whether there is any data in the channel, which has to be transferred; clear the channel enable bit in the relevant channel configuration register 40
41
42
43
44
3. Wait until the transfer completes. the channel is then automatically disabled. 45
46
47
48
49
50
51
52

9.12.6.6 Programming a DMA channel

1. Choose a free DMA channel with the priority needed. where DMA channel 0 has
the highest priority and DMA channel 11 the lowest priority
2. Clear any pending interrupts on the channel to be used by writing to the
dmau_int_tc_clear and **dmau_int_error_clear** registers; the previous channel
operation might have left interrupts active
3. Write the source address into the **dmau_cx_src_addr** register
4. Write the destination address into the **dmau_cx_dest_addr** register
5. Write the address of the next LLI into the **dmau_cx_lll** register; if the transfer
comprises of a single packet of data then 0 must be written into this register
6. Write the control information into the **dmau_cx_control** register
7. Write the channel configuration information into the **dmau_cx_configuration**
register; if the enable bit is set then the DMA channel is automatically enabled.

9.12.6.7 Setting a new DMA transfer

1. If the channel is not set aside for the DMA transaction:
 - a. Read the **dmau_enbld_chns** controller register and find out which channel
are inactive
 - b. Choose an inactive channel which has the required priority.
2. Program the DMAU controller.

9.12.6.8 Address generation

Address generation can be either incrementing or non-incrementing (address
wrapping is not supported).

Some devices, especially memories, disallow burst accesses across certain address
boundaries. The DMAU controller assumes that this is the case with any source or
destination area which is configured for incrementing addressing. This boundary is
assumed to be aligned with the specified burst size. For example, if the channel is set
for 16-transfer bursts to a 32-bit wide device then the boundary is 64-byte aligned
(that is address bits [5:0] equal 0). If a DMA burst is to cross one of these boundaries
then instead of a burst, that transfer is split into separate AHB transactions.

9.12.6.9 Word-aligned transfer across a boundary

The channel is configurated for 16-transfer bursts, each transfer 32-bits wide, to a
destination for which address incrementing is enabled. The start address for the
current burst is 0x0C000024, the next boundary (calculated from the burst size and
transfer width) is 0x0C000040.

The transfer is split into two AHB transactions:

- A 7-transfer burst starting at address 0x0C000024
- A 9-transfer burst starting at address 0x0C000040.

9.12.6.10 Peripheral to peripheral DMA flow

1. Program and enable the DMA channel
2. Wait for a source DMA request
3. The DMAU controller starts transferring data from the source peripheral to the destination peripheral through the DMAU FIFO when:
 4. The DMA request goes active
 5. Or the DMA stream has the highest pending priority
 6. Or the DMAU controller is the bus master of the AHB bus.
7. If an error occurs while transferring the data an error interrupt is generated, then finish
8. Decrement the transfer count if the DMAU controller is controlling the flow control
9. When the transfer has completed (indicated by the transfer count reaching 0 if the DMAU controller is performing flow control or by other internal signals when the peripheral is performing flow control) the following occurs:
 10. a. The DMAU controller asserts a signal to the source peripheral indicating that the last data has been transferred
 11. b. Further source DMA requests are ignored.
12. When the destination DMA request goes active and there is data in the DMAU controller FIFO, transfer data into the destination peripheral
13. If an error occurs while transferring the data, generate an error interrupt, disable the channel and finish
14. When the transfer has completed (indicated by the transfer count reaching 0 if the DMAU controller is performing flow control or by other internal signals when the peripheral is performing flow control) the following occurs:
 15. a. The DMAU controller asserts the dmact signal to the destination peripheral
 16. b. The **dmau_intcombine** interrupt is generated (this interrupt can be masked)
 17. c. If the **dmau_cx_lli** register is not 0, then reload the **dmau_cx_src_addr**, **dmau_cx_dest_addr**, **dmau_cx_lli** and **dmau_cx_control** registers and start again at step 2. Else if **dmau_cx_lli** is 0, disable the channel and finish.

9.12.6.11 Memory to memory DMA flow

1. Program and enable the DMA channel
2. Transfer data whenever the DMA channel has the highest pending priority and the DMAU controller gains bus mastership of the AHB bus
3. If an error occurs while transferring the data, generate an error interrupt, disable the DMA stream and finish
4. Decrement the transfer count
5. If the counter has reached zero:
 6. a. Generate a **dmau_intcombine** interrupt (the interrupt can be masked)
 7. b. If the **dmau_cx_lli** register is not 0, then reload the **dmau_cx_src_addr**, **dmau_cx_dest_addr**, **dmau_cx_lli** and **dmau_cx_control** registers and go back at step 2. However if **dmau_cx_lli** is 0, disable the channel.

9.12.6.12 Scatter/gather

Scatter/gather is supported through the use of linked lists: the source and destination data areas are defined by a series of Linked Lists Items (LLI) allocated anywhere in a readable memory location. Each LLI defines the transfer of one block of data, and then may loads a further LLI to continue the DMA operation. Source and destination areas do not need to occupy contiguous areas in memory. Each LLI can be set to generate an interrupt after completion and continue to next LLI (if next LLI is present) without waiting this interrupt to be handled by the SC.

The first Linked List Item (LLI) is programmed into the DMAU controller. The others LLIs have to be allocated to memory.

A LLI consists of four words organized as described:

- **dmau_cx_src_addr**
- **dmau_cx_dest_addr**
- **dmau_cx_llli**
- **dmau_cx_control.**

Where scatter/gather is not required the **dmau_cx_llli** register should be set to 0.

Note: the **dmau_cx_configuration** DMA channel configuration register is not part of the linked list item.

See [Figure 65](#) for an example of a LLI. A memory block has to be transferred to a peripheral. The addresses of each line of data are given (in hexadecimal) at the left-hand side of the figure. The LLIs describing the transfer are stored contiguously from address 0x20000.

	0x00200	0x00E00
0xA000		
0xB000		
0xC000		
0xD000		
0xE000		
0xF000		
0x10000		
0x11000		

Fig 65. DMAU example of LLI.

The first LLI, stored at 0x20000, defines the first block of data to be transferred, which
is the data stored between addresses 0x0A200 and 0x0AE00. It contains:

- Source start address 0x0A200
- Destination address set to the destination peripheral address
- Transfer width, word (32-bit)
- Transfer size, 3072 bytes (0xC00)
- Source and destination burst sizes, 16 transfers
- Next LLI address, 0x20010.

The second LLI, stored at 0x20010 describes the next block of data to be transferred.
It contains:

- Source start address 0x0B200
- Destination address set to the destination peripheral address
- Transfer width, word (32-bit)
- Transfer size, 3072 bytes (0xC00)
- Source and destination burst sizes, 16 transfers
- Next LLI address, 0x20020.

A chain of descriptors is built up, each one pointing to the next in the series. To
initialize the DMA stream, the first LLI (0x20000) is programmed into the DMAU
controller (The first LLI is not loaded automatically by the DMAU, programming of the
first descriptor in the DMAU is made by the SC, so first LLI does not necessarily need
to be stored in memory, although this is a classic way of work). When the first packet
of data has been transferred the next linked list item is automatically loaded.

The last LLI would be stored at 0x20070 and contain:

- Source start address 0x11200
- Destination address set to the destination peripheral address
- Transfer width, word (32-bit)
- Transfer size, 3072 bytes (0xC00)
- Source and destination burst sizes, 16 transfers
- Next LLI address, 0x0.

Since the next LLI address is set to zero, this is the last descriptor, and the DMA
channel is disabled after transferring the last item of data. The channel is probably set
to generate an interrupt at this point to indicate to the ARM processor that the
channel can be reprogrammed.

Programming the DMAU controller for scatter/gather DMA.

1. Write the LLIs for the complete DMA transfer to memory; the last LLI has its
linked list word pointer set to 0
2. Choose a free DMA channel with the priority needed; DMA channel 0 has the
highest priority and DMA channel 11 the lowest priority

3. Write the first linked list item, previously written to memory, to the relevant channel in the DMAU controller 1
4. Write the channel configuration information to the channel configuration register and set the channel enable bit; the DMAU controller then transfers the first and then subsequent packets of data as each linked list item is loaded 2
5. An interrupt at the end of each LLI is generated if **dmau_cx_control.i** is set - this interrupt is called terminal count interrupt - the **dmau_int_tc_clear** register has to be used to clear the corresponding interrupt - the DMAU continues to run if a new LLI has been reloaded. 3
6. 4
7. 5
8. 6
9. 7
10. 8

9.12.6.13 Interrupt request

Interrupt requests are generated when an AHB error is encountered, or at the end of a transfer (terminal count). Programming the relevant bits on the relevant **dmau_cx_control** and **dmau_cx_configuration** registers can mask the interrupts.

Interrupts status registers are provided prior to interrupt masking (**dmau_raw_int_tc_status**, **dmau_raw_int_error_status**), and after masking (**dmau_int_tc_status**, **dmau_int_error_status**).

The **dmau_int_status** register combines both **dmau_int_tc_status** and **dmau_int_error_status** register into a single register.

Writing to the **dmau_int_tc_clear** or the **dmau_int_error_clear** registers allows selective clearing of interrupts.

Combined terminal count and error interrupt sequence flow: the following procedure should be followed when the **dmau_intcombine** interrupt is asserted in the interrupt controller INTC:

1. Wait until the combined interrupt request from DMAU controller goes active 29
2. Assuming the interrupt is enabled in the interrupt controller and in the processor, the processor branches to the interrupt vector address and enters the interrupt service routine 30
3. Read the interrupt controller register **inthpai** and determine whether the source of the request was the DMAU controller 31
4. Read the **dmau_int_status** register to determine which channel generated the interrupt. If more than one request is active it is recommended that the highest priority channels must be checked first 32
5. Read the **dmau_int_tc_status** or the **dmau_int_error_status** register to determine whether an error occurred or the transfer ended (terminal count) 33
6. Clear the interrupt by setting the corresponding bit in the **dmau_int_tc_clear** or **dmau_int_error_clear** register. 34

Interrupt polling sequence: the DMAU controller interrupt request signal is either masked out, disabled in the interrupt controller or disabled in the processor. The following procedure must be followed when polling the DMAU controller:

1. Read the **dmau_int_status** register; if none of the bits are HIGH repeat this step, otherwise go to step 2; if more than one request is active it is recommended that the highest priority channels must be checked first 45

2. Read the **dmau_int_tc_status** or the **dmau_int_error_status** register to determine whether an error occurred or the transfer ended (terminal count)
 3. Service the interrupt request
 4. Clear the interrupt by setting the corresponding bit in the **dmau_int_tc_clear** or **dmau_int_error_clear** registers.
- 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Company Restricted

9.13 DRT - Digital Receive Transmit

9.13.1 Features

- Two full duplex voice channels are supported
- Hardware decimation and interpolation with 32kHz (DECT) sampling rate
- Processor interface at 32kHz (DECT)sampling rate
- Order of the interpolation filter can be selected (2nd or 3rd order)
- Timing synchronization to speech frame interrupt (int_fsi)

9.13.2 Software interface

Table 204: Register overview of the DRT

Symbol	Name	I/O	Reset
drt_global	DRT global control register	R/W	0x0000 0000
drt_cdc1_di12	CODEC1 Inbound buffer storing samples 1 and 2 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc1_di34	CODEC1 Inbound buffer storing samples 3 and 4 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc1_di5	CODEC1 Inbound buffer storing sample 5 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc1_do12	CODEC1 Outbound buffer storing samples 1 and 2 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_cdc1_do34	CODEC1 Outbound buffer storing samples 3 and 4 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_cdc1_do5	CODEC1 Outbound buffer storing sample 5 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_cdc2_di12	CODEC2 Inbound buffer storing samples 1 and 2 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc2_di34	CODEC2 Inbound buffer storing samples 3 and 4 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc2_di5	CODEC2 Inbound buffer storing sample 5 written by the digital decimation filter to be read by the system controller	R	0x0000 0000
drt_cdc2_do12	CODEC2 Outbound buffer storing samples 1 and 2 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_cdc2_do34	CODEC2 Outbound buffer storing samples 3 and 4 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_cdc2_do5	CODEC2 Outbound buffer storing sample 5 written by the system controller to be read by the digital noise shaper	R/W	0x0000 0000
drt_con	DRT Control Register	R/W	0x0000 0000
drt_codtr	CODEC Test Register defining Digital Dithering functions	R/W	0x0028 0500

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 205: Register drt_global

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	filt2_dis	R/W	0*	Disable filter 2
2	filt1_dis	R/W	0*	Disable filter 1
1	reset	R/W	0*	Block reset DRT ^[1]
0	drt_on	R/W	0*	DRT block enable

[1] Note that a block reset only should be done after the block has been turned via **cgugatesc.drt_en**. This bit is automatically reset by hardware once the reset sequence has completed. After finishing the block reset sequence, it is recommended to add a wait time of 15 ARM cycles (at least one **clk_drt**) before proceeding with DRT register writes.

Table 206: Register drt_cdc1_di12

Bit	Symbol	Access	Value	Description
31 to 16	cdc1_di2	R	0*	16 bit linear PCM sample 2 inbound of CODEC1
15 to 0	cdc1_di1	R	0*	16 bit linear PCM sample 1 inbound of CODEC1, i.e. this is the earliest sample written by the decimation filter during a 125 ms frame

Table 207: Register drt_cdc1_di34

Bit	Symbol	Access	Value	Description
31 to 16	cdc1_di4	R	0*	16 bit linear PCM sample 4 inbound of CODEC1, i.e. this is the latest sample written by the decimation filter during a 125 ms frame if drt_fac = 27
15 to 0	cdc1_di3	R	0*	16 bit linear PCM sample 3 inbound of CODEC1

Table 208: Register drt_cdc1_di5

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15 to 0	cdc1_di5	R	0*	16 bit linear PCM sample 5 inbound of CODEC1, i.e. this is the latest sample written by the decimation filter during a 125 us frame if drt_fac = 25

Table 209: Register drt_cdc1_do12

Bit	Symbol	Access	Value	Description
31 to 16	cdc1_do2	R/W	0*	16 bit linear PCM sample 2 outbound of CODEC1
15 to 0	cdc1_do1	R/W	0*	16 bit linear PCM sample 1 outbound of CODEC1, i.e. this is the first sample read by the noise shaper during in one 125 ms frame

Table 210: Register drt_cdc1_do34

Bit	Symbol	Access	Value	Description
31 to 16	cdc1_do4	R/W	0*	16 bit linear PCM sample 4 outbound, i.e. this is the latest sample read by the noise shaper in one 125 us frame if drt_fac=27
15 to 0	cdc1_do3	R/W	0*	16 bit linear PCM sample 3 outbound of CODEC1

Table 211: Register drt_cdc1_d05

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15 to 0	cdc1_d05	R/W	0*	16 bit linear PCM sample 5 outbound of CODEC1, i.e. this is the latest sample read by the noise shaper in one 125 us frame if drt_fac = 25

Table 212: Register drt_cdc2_di12

Bit	Symbol	Access	Value	Description
31 to 16	cdc2_di2	R	0*	16 bit linear PCM sample 2 inbound of CODEC2
15 to 0	cdc2_di1	R	0*	16 bit linear PCM sample 1 inbound of CODEC2, i.e. this is the earliest sample written by the decimation filter during a 125 ms frame

Table 213: Register drt_cdc2_di34

Bit	Symbol	Access	Value	Description
31 to 16	cdc2_di4	R	0*	16 bit linear PCM sample 4 inbound of CODEC2, i.e. this is the latest sample written by the decimation filter during a 125 ms frame if drt_fac = 27
15 to 0	cdc2_di3	R	0*	16 bit linear PCM sample 3 inbound of CODEC2

Table 214: Register drt_cdc2_di5

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15 to 0	cdc2_di5	R	0*	16 bit linear PCM sample 5 inbound of CODEC2, i.e. this is the latest sample written by the decimation filter during a 125 us frame if drt_fac = 25

Table 215: Register drt_cdc2_d012

Bit	Symbol	Access	Value	Description
31 to 16	cdc2_d02	R/W	0*	16 bit linear PCM sample 2 outbound of CODEC2
15 to 0	cdc2_d01	R/W	0*	16 bit linear PCM sample 1 outbound of CODEC2, i.e. this is the first sample read by the noise shaper during in one 125 ms frame

Table 216: Register drt_cdc2_d034

Bit	Symbol	Access	Value	Description
31 to 16	cdc2_d04	R/W	0*	16 bit linear PCM sample 4 outbound, i.e. this is the latest sample read by the noise shaper in one 125 us frame if drt_fac = 27
15 to 0	cdc2_d03	R/W	0*	16 bit linear PCM sample 3 outbound of CODEC2

Table 217: Register drt_cdc2_d05

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15 to 0	cdc2_d05	R/W	0*	16 bit linear PCM sample 5 outbound of CODEC2, i.e. this is the latest sample read by the noise shaper in one 125 us frame if drt_fac = 25

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 218: Register drt_con

Bit	Symbol	Access	Value	Description
31 to 9	-	R	0*	reserved
8 to 7	dither_pwr	R/W		Select the power of the dither: applied dither=+/- $(0:7)*2^{(7+dither_pwr)}$
			0b11	+/- $(0:7)*2^{10}$
			0b10	+/- $(0:7)*2^9$
			0b01	+/- $(0:7)*2^8$
			0b00*	+/- $(0:7)*2^7$
6	dns2_newdith	R/W		select new dither method for DNS2
			1	enabled
			0*	disabled
5	dns1_newdith	R/W		select new dither method for DNS1
			1	enabled
			0*	disabled
4 to 3	dns2_mode	R/W		select mode of noise shaper DNS2
			0b11	DNS2_PWM: generate a PWM signal with duty cycle cdc2_do1 (usable range is +/- 108, respectively +/-100 when drt_fac is set). Period is 125 us.
			0b10	DNS2_FS: generate a full-scale signal for input values >0, an ideal 0 for = 0 and a negative full-scale signal for input values <0
			0b01	DNS2_3RD: 3rd order noise shaper
			0b00*	DNS2_2ND: 2nd order noise shaper
2 to 1	dns1_mode	R/W		select mode of noise shaper DNS1
			0b11	DNS1_PWM: generate a PWM signal with duty cycle cdc1_do1 (usable range is +/- 108, respectively +/-100 when drt_fac is set). Period is 125 us.
			0b10	DNS1_FS: generate a full-scale signal for input values >0, an ideal 0 for = 0 and a negative full-scale signal for input values <0
			0b01	DNS1_3RD: 3rd order noise shaper selected
			0b00*	DNS1_2ND: 2nd order noise shaper selected
0	drt_fac	R/W		select decimation for DDF and interpolation factor for DNS
			1	set to 25
			0*	set to 27

Company Confidential

Table 219: Register drt_codtr

Bit	Symbol	Access	Value	Description
31 to 22	-	R	0*	reserved
21 to 14	dns2_dither	R/W	0b10100000*	Digital dither signal in DNS2
13 to 11	codec2_tst	R/W		codec test mode with multiplexed pins on GPIO
			0b111	PCM probe for CODEC 2
			0b110	4fs closed loop for CODEC 2
			0b101	4fs closed loop for CODEC 2
			0b100	4fs CODEC for CODEC 2
			0b011	1 bit closed loop for CODEC 2
			0b010	1 bit digital for CODEC 2
			0b001	1 bit analog for CODEC 2
			0b000*	normal operation
10 to 3	dns1_dither	R/W	0b10100000*	Digital dither signal in DNS1
2 to 0	codec1_tst	R/W		codec test mode with multiplexed pins on GPIO's
			0b111	PCM probe for CODEC 1
			0b110	4fs closed loop for CODEC 1
			0b101	4fs closed loop for CODEC 1
			0b100	4fs CODEC for CODEC 1
			0b011	1 bit closed loop for CODEC 1
			0b010	1 bit digital for CODEC 1
			0b001	1 bit analog for CODEC 1
			0b000*	normal operation

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.13.3 Functional description

9.13.3.1 General description

The DRT block consists of a digital decimation DDF and an interpolation or noise shaping filter DNS. The decimation filter (DDF) decimates the samples from the analog-to-digital converters to an intermediate frequency of 32 or 40 kHz for further processing. The digital interpolation filter or noise shaping filter (DNS) interpolates samples at the intermediate frequency of 32 or 40 kHz to the sampling rate required by the digital-to-analog-converters. The intermediate frequency of 32 or 40 kHz depends on the system frequency of the application.

For DECT and ISM applications, the system frequency is a multiple of 1.152 MHz and the AD/DA-converters operate at 864 kHz. The decimation factor is 27 and the intermediate frequency is 32 kHz.

The general description in this chapter describes a single voice path (analog-to-digital converter, DDF, DNS, digital-to-analog converter) only although there is a second one implemented. The required description of the registers for the second voice path can be found in [9.13.2](#).

9.13.3.2 Operation

The 16-bit linear PCM data from the digital decimation filter is written every 32 or 40 kHz into the corresponding intermediate transmit buffers ITB1..ITB5 and with the same rate the 14-bit linear PCM data is read by the digital noise shapers from the intermediate receive buffer IRB1..IRB5. Upon an int_fsi signal both the intermediate transmit buffers ITB1..5 are copied to the codec inbound buffers **cdc1_diX** and the codec outbound buffers **cdc1_doX** are copied to the intermediate receive buffers IRB1..5. The system controller must copy the data from all inbound buffers **cdc1_diX** to the memory and write data from the memory to all outbound buffers **cdc1_doX**. The system controller has 125 us time to do these operations otherwise the inbound buffers are overwritten or old data is sent to the DNS. The samples in buffers ITB1 and IRB1 are the oldest ones of a speech frame. The DRT control block synchronizes the conversion cycles of the DNS / DDF and the intermediate receive/transmit buffer selection to the speech frame marked by int_fsi. It also controls the serial data transfer between the DPU and the APU. The same process of operation applies to the **cdc2_diX** and **cdc2_doX** registers.

9.13.3.3 3rd order noise shaper

There is a 3rd order noise shaper implemented. To use it, register **drt_con.dns1_mode** should be set to '0b01' or **drt_con.dns2_mode** is set to '0b01'. It is recommended to set also the bits **drt_con.dns1_newdith** =1 or **drt_con.dns2_newdith** = 1 and **drt_con.dither_pwr** ='0b00' to avoid idle tones. The advantage of the 3rd order noise shaper is that noise level in the range of 4-20kHz is improved by up to 8dB (see [Table 220](#)). For signals larger than -3dBm, there is no improvement.

Table 220: Noise shaper performance comparison (measured with sine tone of 1kHz)

DNS input signal level [dBm0]	2nd order noise shaper				3rd order noise shaper		
	Measured signal [dBm]	S/N+THD 22Hz - 22kHz [dB]	S/N+THD CCITT [dBp]	Measured signal [dBm]	S/N+THD 22Hz - 22kHz [dB]	S/N+THD CCITT [dBp]	
+3.14	+1.69	-63	-69	+1.67	-63.8	-67.7	
0	-1.43	-65.6	-78	-1.44	-68.4	-73.4	
-6	-7.47	-63.8	-76.9	-7.47	-69.5	-79.7	
-12	-13.5	-58.8	-75	-13.5	-66.4	-77.8	
-20	-21.4	-50.6	-69.3	-21.4	-59.4	-71.5	
-40	-41.4	-31.6	-49.9	-41.4	-39.5	-51.3	
-60	-61.5	-11.7	-30.3	-61.5	-19.5	-32.0	
muted	noise only	noise floor at -73.2dBm	noise floor at -92dBmp	noise only	noise floor at -81.3dBm	noise floor at -92dBmp	

9.13.3.4 New dither

There is a new dither method built in based on adding white-noise AC-dither to cancel idle-tones. The strength can be chosen with the bit field **drt_con.dither_pwr**.

9.13.3.5 Full scale mode / PWM mode

For non-audio application of the D/A converters the DNS can be put into the fullscale mode (output is either plus or minus fullscale or zero) or a pulse-width-modulation mode. This is set with **drt_con.dns1_mode** or **.drt_con.dns2_mode**.

9.13.3.6 Power saving

When one of the two DRTs is not used, it can be disabled by setting **filt1_dis** or **filt2_dis**. This saves a few uA of current at VDDC supply rail.

9.13.4 Application information

With the bit **drt_global.reset** a software controlled reset of the DRT can be initiated.
The controlled reset leads to the same state of the DRT block as a power on reset.

Note that if the 32 kHz mode is active the registers **cdc1_di5**, **cdc2_di5**, **cdc1_do5** and **cdc2_do5** contain invalid data.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.14 EBI1 and EBI2 - External Bus Interfaces

There are two external bus interfaces in the DVFD818x. The two interfaces have identical functionality, but their configuration differs. They are available on the FMP and HMP. Due to the HMP supports less pins than the FMP the use cases for the EBI2 are limited to asynchronous mode only.

9.14.1 Features

- Connection to external parallel interface devices
- 6 independent chip selects with fixed address space
- Up to 256 MByte address range per chip select
- Support for 8, 16, and 32 bit (multiplexed synchronous mode) devices
- Byte enable signals for 16 bit and 32 bit devices
- Fully programmable asynchronous timings based on **hclk** cycles
 - individually selectable timings for read and write
 - 0 to 7 clock cycles for setup
 - 1 to 128 clock cycles for access cycle
 - 1 to 8 clock cycles for page access cycle
 - 0 to 7 clock cycles for hold
 - 1 to 15 clock cycles for read and write turnaround
- Page mode memory support
 - page size of 4, 8, 16 or 32
- Synchronous memory support up to **hclk** clock frequency (for devices sensitive to rising edge of the clock only)
 - access time and turnaround time configuration
 - **hclk**, **hclk/2**, **hclk/4** or **hclk/8** burst clock output
 - burst size of 4, 8, 16, 32
 - **WAIT** input
 - automatic CLKBURST power-down between accesses
- Synchronous write mode
- Configuration of address valid duration and position
- 8080 mode (**WE** and **OE**) and 6800 mode (**E** and **R/W**) control signals
- Multiplexed data/address mode (x16 and x32 width)
 - support for latched address mode
- Adaptation to word, halfword, and byte accesses to the external devices

CAUTION



At DVFD8185 and DVFD8187 there are less EBI pins available and therefore the supported function especially on EBI1 is very limited. See pinlist at [Table 7](#) for available pins and signal assignments for FMP at [Table 10](#) and for HMP at [Table 27](#). Have it in mind for this chapter due to the differences are not always emphasized.

9.14.2 Block diagram

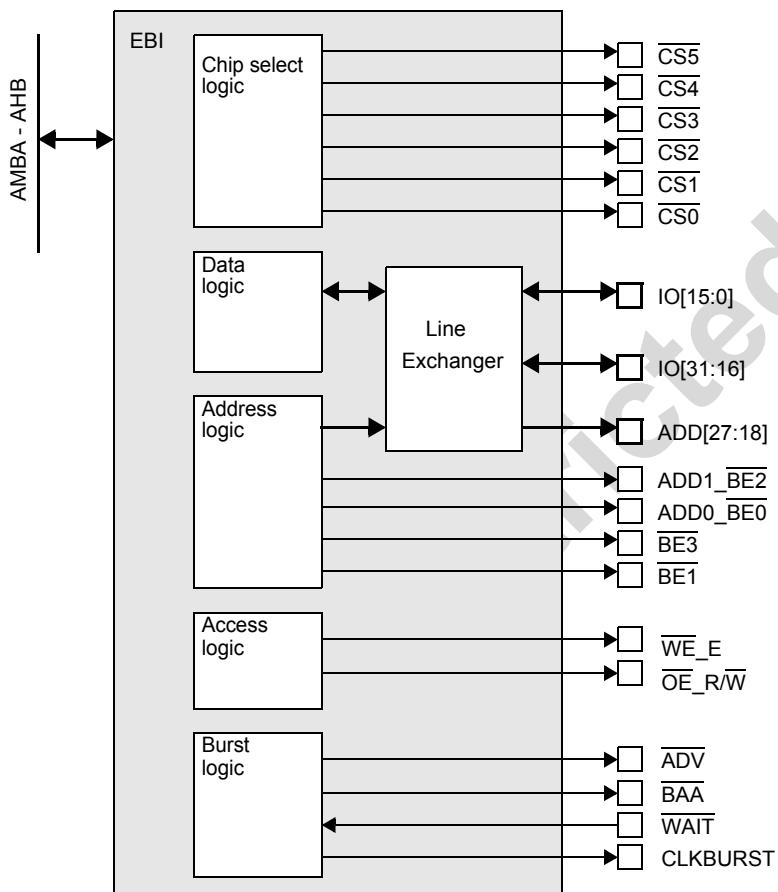


Fig 66. EBI block diagram

Note: For supported signals at DVFD818x please see sections FMP - Fast Memory Port and HMP - Hybrid Memory Port

9.14.3 Hardware Interface

Table 221:EBI pin overview

Pin	Name	I/O	Description
CS[5:0]	chip selects	O	chip select signals for external devices
IO[31:0]	external IO bus	I/O	bidirectional data and address lines for external devices
ADD[27:18]	external address bus	O	address lines for external devices
ADD1_BE2	external address / byte enable 2	O	address line for external devices or byte select for 32 bit devices
ADD0_BE0	external address / byte enable 0	O	address line for external devices or byte select for 16 or 32 bit devices
BE3	byte enable 3	O	byte select for 32 bits devices
BE1	byte enable 1	O	byte select for 16 or 32 bits devices
OE_R/W	output enable / read not write	O	output enable signal (8080 mode) read not write signal (6800 mode)
WE_E	write enable / enable	O	write enable signal (8080 mode) enable signal (6800 mode)
ADV	address valid	O	address valid
BAA	burst address advance	O	burst address advance for synchronous operation
CLKBURST	clock for burst	O	clock for burst
WAIT	wait	I	wait signal for synchronous operation

[1] For available pins at DVFD8185 and DVFD8187 look for FMP at [Table 10](#) and for HMP at [Table 27](#).

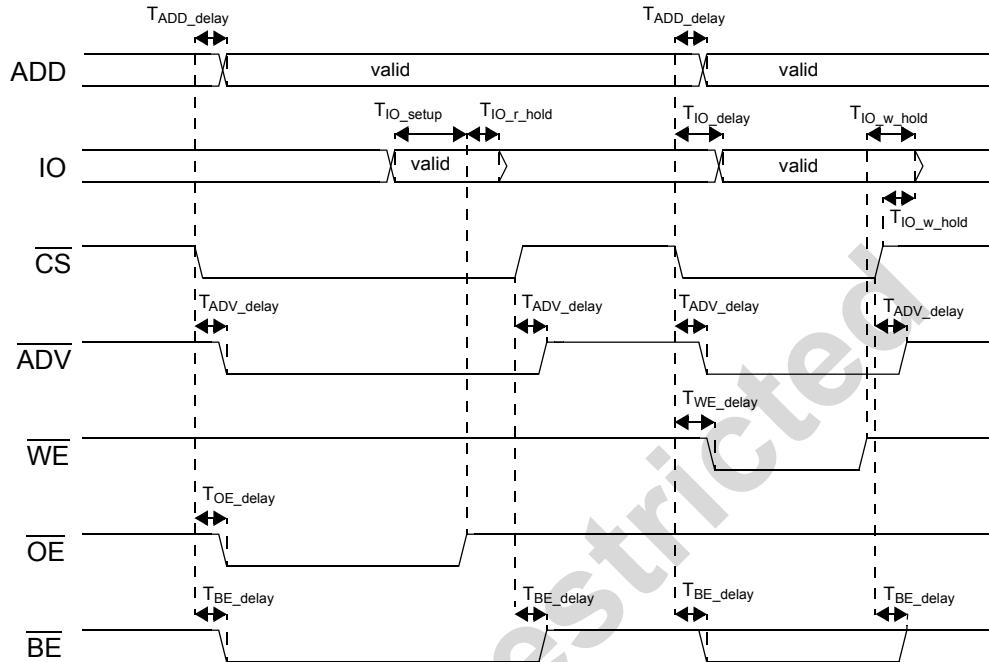
For all timing diagrams in this section the following notations are used:

- ADD is used for address bus on ADD[27:18] or on IO
- CS is used for either CS0, CS1, CS2, CS3, CS4, CS5
- BE is used either for ADD1_BE2, ADD0_BE0, BE3, or BE1
- OE and R/W are used for OE_R/W
- WE and E are used for WE_E

The EBI1 timing parameters are specified for 104 MHz (see [Table 626](#)) operation and 20 pF maximum load.

The EBI2 timing parameters are specified for 52 MHz operation and 30 pF maximum load.

The connected memory devices need to match the 70 Ω impedance on 30 cm maximum distance in order to reduce the signal overshoot because of signal reflection.



(1) The ADV is mentioned here because synchronous devices require the signal to be asserted when an asynchronous access is performed.

Fig 67. EBI asynchronous read and write and page read timing diagram

Table 222: AC characteristics of EBI1 asynchronous accesses

Symbol	Description	Min	Typ	Max	Unit
T_ADD_delay	ADD delay time from <u>CS</u> active	-	-	3	ns
T_IO_setup	IO to <u>OE</u> setup time	4	-	-	ns
T_IO_r_hold	IO hold time after <u>OE</u> inactive	0	-	-	ns
T_IO_delay	IO delay time from <u>CS</u> active	-	-	3	ns
T_IO_w_hold	IO hold time after <u>WE</u> inactive or <u>CS</u> inactive	1	-	-	ns
T_ADV_delay	ADV delay time from <u>CS</u> active and inactive	-	-	2	ns
T_WE_delay	<u>WE</u> delay time from <u>CS</u> active	-	-	2 [1]	ns
T_OE_delay	<u>OE</u> delay time from <u>CS</u> active	-	-	2 [1]	ns
T_BE_delay	<u>BE</u> delay time from <u>CS</u> active	-	-	2	ns

[1] These maximum delays depend also on the setting of the EBI registers. The specified time holds for **rs=0** and **ws=0**.

Table 223: AC characteristics of EBI2 asynchronous accesses

Symbol	Description	Min	Typ	Max	Unit
T_ADD_delay	ADD delay time from <u>CS</u> active	-	-	3	ns
T_IO_setup	IO to <u>OE</u> setup time	8	-	-	ns
T_IO_r_hold	IO hold time after <u>OE</u> inactive	0	-	-	ns
T_IO_delay	IO delay time from <u>CS</u> active	-	-	5	ns
T_IO_w_hold	IO hold time after <u>WE</u> inactive or <u>CS</u> inactive	1	-	-	ns

Table 223: AC characteristics of EBI2 asynchronous accesses...continued

Symbol	Description	Min	Typ	Max	Unit
T _{WE_delay}	WE delay time from CS active	-	-	3 [1]	ns
T _{OE_delay}	OE delay time from CS active	-	-	3 [1]	ns
T _{BE_delay}	BE delay time from CS active	-	-	3	ns

[1] These maximum delays depend also on the setting of the EBI registers. The specified time holds for rs=0 and ws=0.

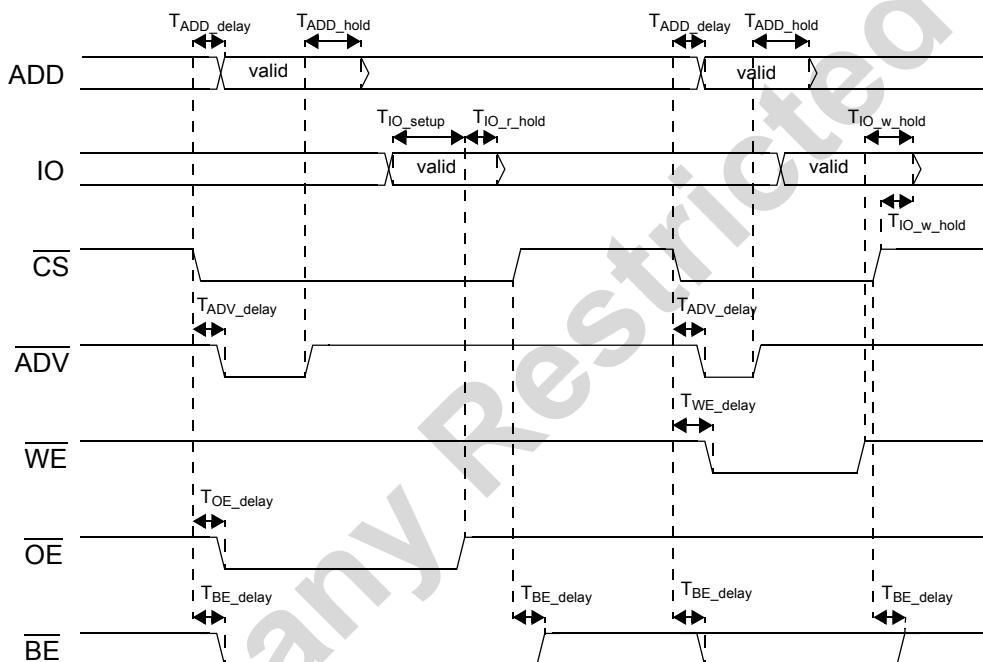


Fig 68. EBI asynchronous latched read and write access timing diagram

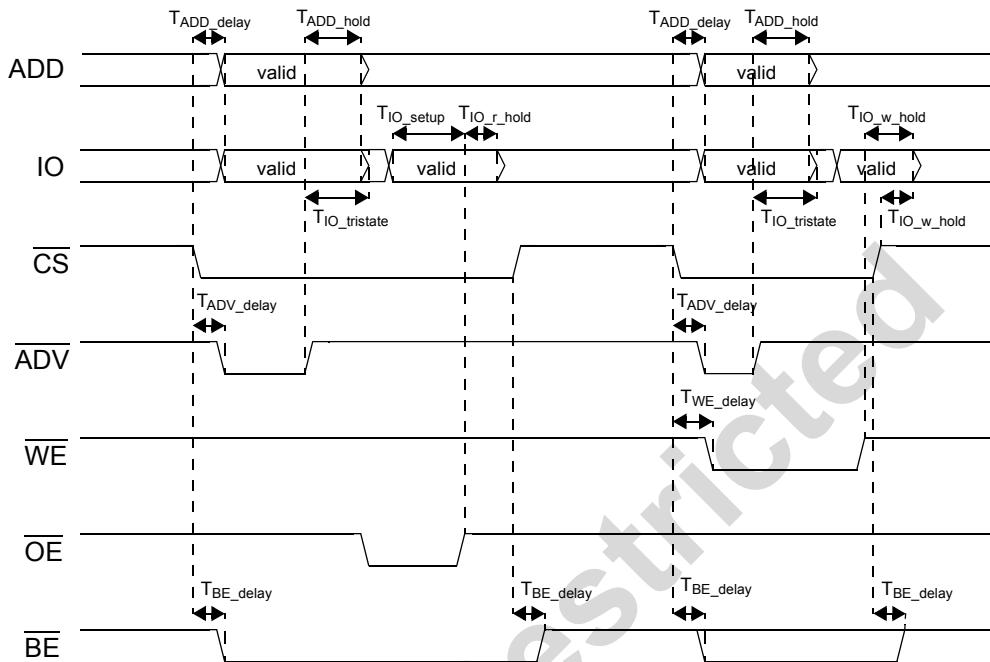
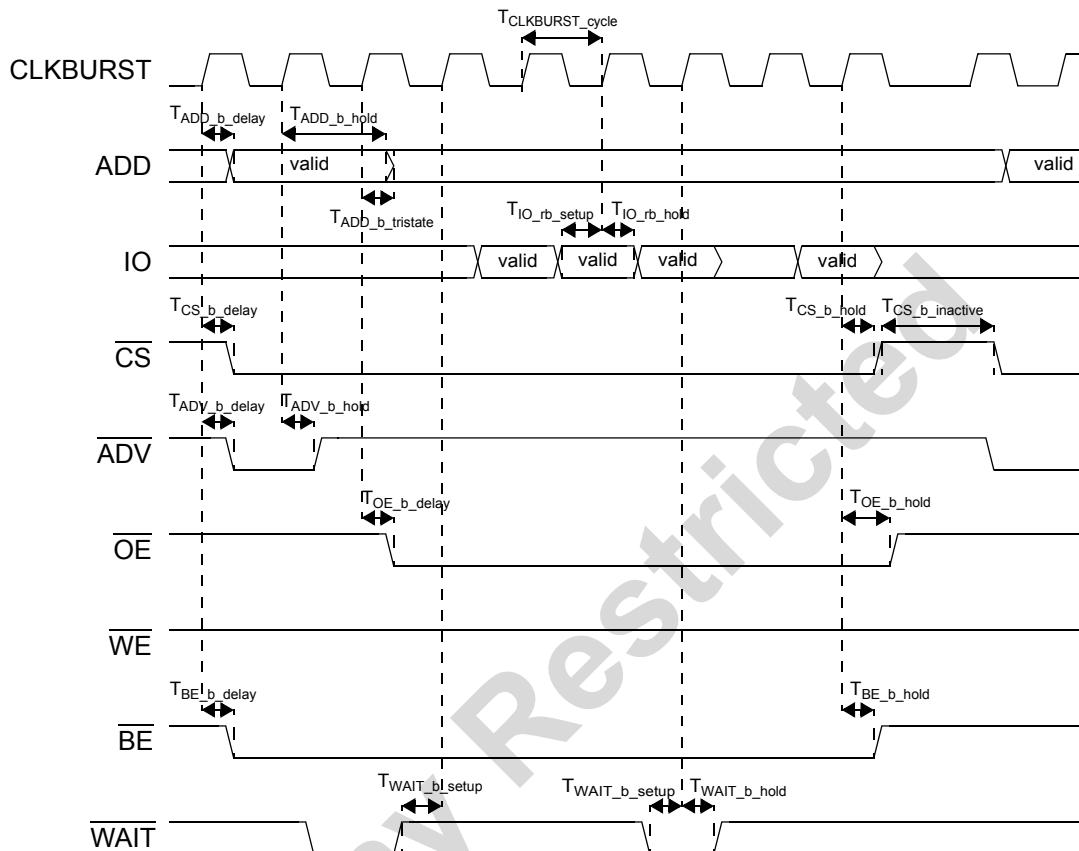


Fig 69. EBI asynchronous latched multiplexed read and write timing diagram

Table 224: AC characteristics of EBI1 latched address access

Symbol	Description	Min	Typ	Max	Unit
T_{ADD_hold}	ADD and IO hold time from \overline{ADV} rising edge	9 [1]	-	-	ns
$T_{IO_tristate}$	IO tristate after \overline{ADV} rising edge	-	-	16	ns

[1] Higher values can be achieved with **advcfg[1:0]** different than 0b01.



(1) The CLKBURST signal can be configured to be active between bursts with `always_clk_en = 1`.

Fig 70. EBI synchronous read timing diagram

Table 225: AC characteristics of EBI1 synchronous read

Symbol	Description	Min	Typ	Max	Unit
T _{CLKBURST_cycle}	CLKBURST cycle time	9	-	616	ns
T _{ADD_b_delay}	CLKBURST to ADD valid delay	-	-	6 [1]	ns
T _{ADD_b_hold}	ADD hold time after CLKBURST	9	-	-	ns
T _{ADD_b_tristate}	ADD transition to tristate after rising edge of CLKBURST	-	-	4 [1]	ns
T _{IO_rb_setup}	IO to CLKBURST setup time	2.5	-	-	ns
T _{IO_rb_hold}	IO hold time after CLKBURST	1	-	-	ns
T _{CS_b_delay}	CLKBURST to CS valid delay	-	-	3 [1]	ns
T _{CS_b_hold}	CLKBURST to CS valid hold	2 [4]	-	-	ns
T _{CS_b_inactive}	CS inactive time	8 [2]	-	-	ns
T _{ADV_b_delay}	CLKBURST to ADV to valid delay	-	-	5 [1]	ns
T _{ADV_b_hold}	ADV hold time after CLKBURST	2 [3][4]	-	-	ns
T _{OE_b_delay}	CLKBURST to OE valid delay	-	-	3 [1]	ns
T _{OE_b_hold}	CLKBURST to OE valid hold	2 [4]	-	-	ns
T _{BE_b_delay}	CLKBURST to BE valid delay	-	-	3 [1]	ns

Table 225: AC characteristics of EBI1 synchronous read...continued

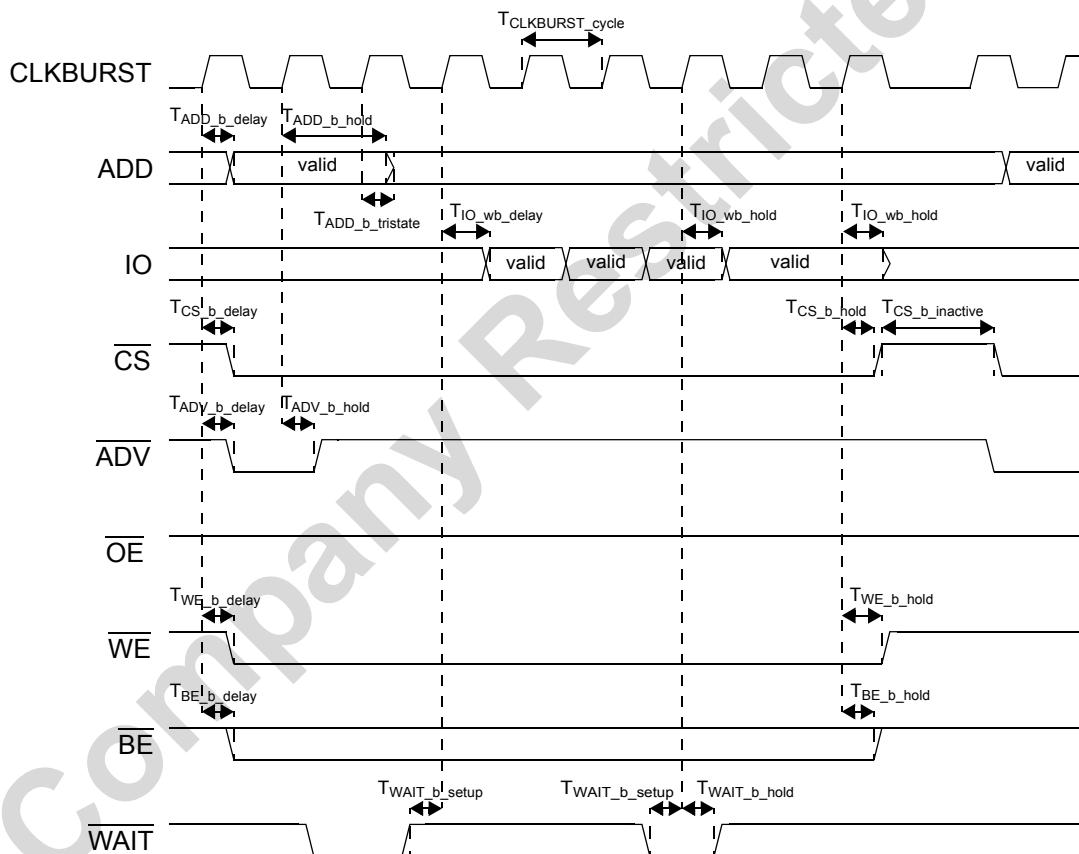
Symbol	Description	Min	Typ	Max	Unit
T _{BE_b_hold}	CLKBURST to BE valid hold	2 ^[4]	-	-	ns
T _{WAIT_b_setup}	WAIT to CLKBURST setup time	2.5	-	-	ns
T _{WAIT_b_hold}	WAIT hold time after CLKBURST	1	-	-	ns

[1] This delay timing is $T_{CLKBURST}/2$ longer when CLKBURST is divided with a factor larger than 1.

[2] The minimum time is also insured if the turnaround times are programmed to 0.

[3] $T_{ADV_b_hold}$ is 0.5 hclk cycles longer in case bit burstcfg.advext is set to 1

[4] To achieve 2ns minimum hold time the field **syscon0.ebi_delay_ctrl** should be set to 0b010. The delay is also applied to the IO lines.



(1) The CLKBURST signal can be configured to be active between bursts (**always_clk_en** = 1).

Fig 71. EBI synchronous write timing diagram

Table 226: AC characteristics of EBI1 synchronous write

Symbol	Description	Min	Typ	Max	Unit
T _{IO_wb_delay}	IO delay time after rising edge of CLKBURST	-	-	6 ^[1]	ns
T _{IO_wb_hold}	IO hold time after rising edge of CLKBURST	2	-	-	ns
T _{WE_b_delay}	CLKBURST to WE valid delay	-	-	3 ^[1]	ns

Table 226: AC characteristics of EBI1 synchronous write...continued

Symbol	Description	Min	Typ	Max	Unit
T _{WE_b_hold}	CLKBURST to \overline{WE} valid hold	2	-	-	ns
T _{BE_b_delay}	CLKBURST to BE valid delay	-	-	3 [1]	ns
T _{BE_b_hold}	CLKBURST to BE valid hold	2	-	-	ns

[1] This delay timing is $T_{CLKBURST}/2$ longer when CLKBURST is divided with a factor larger than 1.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.14.4 Software Interface

Table 227: Register overview of EBI

Symbol	Name	I/O	Reset
ebi_maincfg0	EBI main configuration register for chip select 0	R/W	0x0000 0001 [1]
ebi_readcfg0	EBI read timing configuration register for chip select 0	R/W	0x001F FFFF
ebi_writecfg0	EBI write timing configuration register for chip select 0	R/W	0x001F FFFF
ebi_burstcfg0	EBI burst and page configuration register for chip select 0	R/W	0x0000 00E0
ebi_maincfg1	EBI main configuration register for chip select 1	R/W	0x0000 0001
ebi_readcfg1	EBI read timing configuration register for chip select 1	R/W	0x001F FFFF
ebi_writecfg1	EBI write timing configuration register for chip select 1	R/W	0x001F FFFF
ebi_burstcfg1	EBI burst and page configuration register for chip select 1	R/W	0x0000 00E0
ebi_maincfg3	EBI main configuration register for chip select 3	R/W	0x0000 0001
ebi_readcfg3	EBI read timing configuration register for chip select 3	R/W	0x001F FFFF
ebi_writecfg3	EBI write timing configuration register for chip select 3	R/W	0x001F FFFF
ebi_burstcfg3	EBI burst and page configuration register for chip select 3	R/W	0x0000 00E0
ebi_always_clk_cfg	EBI clock configuration register	R/W	0x0000 0000
ebi_apb_err_cf	APB Error configuration register	R/W	0x0000 0003
ebi_apb_latenc_y_cfg	APB Read data latency configuration register	R/W	0x0000 0001
ebi_sw_resetcf	EBI SW reset register	R/W	0x0000 0000
ebi_power_dow_ncfg	EBI powerdown control register	R/W	0x0000 0000
ebi_ip_id	EBI1 IP identification register	R	0x2016 5000
	EBI2 IP identification register		0x2016 5200

[1] For EBI1, the reset value of this register can be altered by SCROM code execution during start-up phase

ebi_maincfg[3,1:0] registers have identical content. For this reason only **ebi_maincfg0** is detailed hereafter. The same holds for **ebi_readcfg[3,1:0]**, **ebi_writecfg[3,1:0]**, and **ebi_burstcfg[3,1:0]**.

Table 228: Register ebi_ip_id

Bit	Symbol	Access	Value	Description
31 to 16	module_id	R	0x2016*	module identification for EBI
15 to 12	major_rev	R	0x5*	major revision
11 to 8	minor_rev	R	0x0*	minor revision for EBI1
			0x2*	minor revision for EBI2
7 to 0	aperture	R	0x00*	aperture of register interface in 4kbyte blocks

Table 229: Register ebi_maincfg0

Bit	Symbol	Access	Value	Description
31 to 7	-	R	0*	reserved
9 to 8	adv_cfg[1:0] ^[1]	R/W		ADV length: length of pulse in asynchronous latched mode
			0x0*	4 cycles
			0x1	1 cycle
			0x2	2 cycles
			0x3	3 cycles
7	pol	R/W		enable polarity: active level of output signal \overline{WE}_E in 6800 mode
			0*	\overline{WE}_E pin active at high level
			1	\overline{WE}_E pin active at low level
6	be	R/W		byte enable operation
			0*	$\overline{BE}_0, \overline{BE}_1, \overline{BE}_2, \overline{BE}_3$ function as chip select \overline{BCS}
			1	$\overline{BE}_0, \overline{BE}_1, \overline{BE}_2, \overline{BE}_3$ function as write enable \overline{BWE}
5 to 3	mode[2:0]	R/W		operation mode
			0b000*	asynchronous mode (8080 compatible) when msize = 0b0x asynchronous latched mode (8080) when msize = 0b1x
			0b001	page mode
			0b010	synchronous read, asynchronous write when msize = 0b0x synchronous read, asynchronous latched write when msize = 0b1x
			0b011	asynchronous mode (6800 compatible)
			0b100	synchronous read and write
			others	reserved
2 to 0	msize[2:0]	R/W		device width
			0b000	8 bits device
			0b001*	16 bits device
			0b010	16 bits multiplexed device (x16 width) - this setting is only allowed when mode = 0b000, 0b010, 0b100 - other combinations are not supported
			0b011	32 bits multiplexed device (x32 width) - this setting is only allowed when mode = 0b000, 0b010, 0b100 - other combinations are not supported
			others	reserved

[1] **adv_cfg** should be programmed less than **ws+wc** and less than **rs+rc** to avoid unproper latching situation.

Table 230: Register ebi_readcfg0

Bit	Symbol	Access	Value	Description
31 to 22	-	R	0*	reserved
21 ^[6]	rbe	R/W	0*	read byte enable : determines the operation of the \overline{BE} signals for read accesses when ebi_maincfg.be is set to 0 ^[5]
			0*	the byte enable signals are only active for the bytes that are actually used by the EBI - this mode can be selected for those memories that support reading of data with a smaller word size than the memory width, resulting in some power saving
			1	all byte enable signals are active in read accesses - this mode has to be selected for memory devices that require all enable signals to be active during read accesses
20 to 17	rrt[3:0]	R/W	0b1111*	read to read turnaround time : determines the number of cycles ^[1] inserted at the end of a read operation before the next read on the same chip select occurs. The applied value is rrt+1. ^{[2][3][7]}
16 to 13	rt[3:0]	R/W	0b1111*	read turnaround time : determines the number of cycles ^[1] inserted at the end of a read operation (except if next access is a read on the same chip select) ^{[2][3]}
12 to 10	rs[2:0]	R/W	0b111*	read setup time : determines the number of cycles ^[1] required to setup a read operation between the assertion of \overline{CS} and the assertion of \overline{OE} ; in 6800 mode it is the number of cycles between the falling edge of \overline{CS} and the first edge of enable E
9 to 7	rh[2:0]	R/W	0b111*	read hold time : determines the number of cycles ^[1] inserted at the end of a read operation between \overline{OE} is released and the \overline{CS} is released; in 6800 mode it is the number of cycles between the second edge of enable E and the \overline{CS} rising edge
6 to 0	rc[6:0]	R/W	0b111111*	read access time : determines the number of cycles ^[1] required to perform a read access; this field configures the number of cycles that the \overline{OE} is active; in 6800 mode it is the number of cycles that the enable E is active; in synchronous mode it is the number of cycles between the rising of \overline{ADV} and the falling edge of \overline{BAA} ^[4]

[1] The number of clock cycles with a granularity of **hclk** for asynchronous access, and granularity of CLKBURST for synchronous access.

[2] For the turnaround times a programmed 0 and 1 mean 1 cycle.

[3] It is mandatory to program **rt** \geq **rrt**.

[4] For the read access time a programmed 0 means 128 cycles.

[5] When the field **ebi_maincfg.be** is set to 1, i.e. when the byte enable signals behave as write enable, then the byte enable outputs are de-asserted during read accesses.

[6] Function is not available at EBI1. Will return '0' when read. Is in conflict with CellularRAM 1.5 specification when devices with 16/32-bit data bus width are used where during burst read operation all byte select signals have to be active.

[7] The **rrt** should be ≥ 2

Table 231: Register ebi_writecfg0

Bit	Symbol	Access	Value	Description
31 to 21	-	R	0*	reserved
20 to 17	wwt[3:0]	R/W	0b1111*	write to write turnaround time: determines the number of cycles [1] inserted at the end of a write operation before the next write on the same chip select occurs. The applied value is wwt+1. [2][3][5]
16 to 13	wt[3:0]	R/W	0b1111*	write turnaround time: determines the number of cycles [1] inserted at the end of a write operation (except if next access is a write on the same chip select) [2][3]
12 to 10	ws[2:0]	R/W	0b111*	write setup time: determines the number of cycles [1] required to setup a write operation between the assertion of CS and the assertion of WE; in 6800 mode it is the number of cycles between the falling edge of CS and the first edge of enable E
9 to 7	wh[2:0]	R/W	0b111*	write hold time: determines the number of cycles [1] inserted at the end of a write operation between WE is released and the CS is released; in 6800 mode it is the number of cycles between the second edge of enable E and the CS rising edge
6 to 0	wc[6:0]	R/W	0b1111111*	write access time: determines the number of cycles [1] required to perform a write access; this field configures the number of cycles that the WE is active; in 6800 mode it is the number of cycles that the enable E is active; in synchronous mode it is the number of cycles between the rising of ADV and the falling edge of BAA [4]

- [1] The number of clock cycles with a granularity of hclk for asynchronous access, and granularity of CLKBURST for synchronous access.
 [2] For the turnaround times a programmed 0 and 1 mean 1 cycle.
 [3] It is mandatory to program wt >= wwt.
 [4] For the write access time a programmed 0 means 128 cycles.
 [5] The wwt should be >= 3.

Table 232: Register ebi_burstcfg0

Bit	Symbol	Access	Value	Description
31 to 17	-	R	0*	reserved
16	be_hold	R/W	0*	extension of BE pulse length by 0.5 CLKBURST cycles
			1 [4]	disabled
			1	enabled
15	we_hold	R/W	0*	extension of WE pulse length by 0.5 CLKBURST cycles
			1	disabled
			1	enabled
14	oe_hold	R/W	0*	extension of OE pulse length by 0.5 CLKBURST cycles
			1	disabled
			1	enabled
13	cs_hold	R/W	0*	extension of CS pulse length by 0.5 CLKBURST cycles
			1	disabled
			1	enabled
12	adv_hold	R/W	0*	extension of ADV pulse length by 0.5 CLKBURST cycles
			1	disabled
			1	enabled

Table 232: Register ebi_burstcfg0...continued

Bit	Symbol	Access	Value	Description
11 to 10	sync_cap_delay[1:0]	R/W		capture delay: delay in sampling data in synchronous read accesses - when this field is set to a value different than 0, then wait_en must be disabled (not supported)
			0b00*	0 cycles
			0b01	1 cycle after synchronous read access rc
			0b10	2 cycles after synchronous read access rc
			0b11	3 cycles after synchronous read access rc
9	-	R	0*	reserved
8	en	R/W		wait enable for synchronous mode
			0*	WAIT is not used; wait-states generation by WAIT is disabled
			1	WAIT is used; wait-states generation by WAIT is enabled
7 to 5	prc[2:0]	R/W	0b111*	page read access time: specifies the number of cycles between an address change on the external bus and a valid data output from the memory in page mode [2]
4 to 3	clkdiv[1:0]	R/W		burst clock divisor for synchronous mode: configures the clock divider [1]
			0b00*	hclk
			0b01	hclk divided by 2
			0b10	hclk divided by 4
			0b11	hclk divided by 8
2 to 0	size[2:0]	R/W		page size or burst size: indicates the page size of a page mode device or the burst size of a synchronous mode device - the size in bytes has to be correlated with device data width [3]
			0b000*	4-data
			0b001	8-data
			0b010	16-data
			0b011	32-data
			others	reserved

[1] The setting of **clk_div** for each of the EBI devices is overruled by the **always_clk_div** setting when **always_clk_en** is set.

[2] For the page read access time a programmed 0 means 8 cycles.

[3] The maximum page or burst size is 32 bytes independent on the data width

[4] Setting is ignored if **ebi_maincfg0.mode** = 0b010.

Table 233: Register ebi_always_clkcfg

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	stop_during_asy_nc	R/W		memory clock control during asynchronous accesses and related turnaround times
			0*	disabled. CLKBURST always running
			1	enabled. CLKBURST stopped during asynchronous accesses

Table 233: Register ebi_always_clkcfg...continued

Bit	Symbol	Access	Value	Description
2	always_clk_en	R/W	0*	memory clock enable
			disabled.	CLKBURST running only for synchronous memory accesses
			1	enabled. CLKBURST always running, also in-between memory accesses
1 to 0	always_clk_div[1: 0]	R/W		memory clock divider: this setting overrules the clk_div setting for each single device when always_clk_en is set
			0b00*	hclk
			0b01	hclk divided by 2
			0b10	hclk divided by 4
			0b11	hclk divided by 8

Table 234: Register ebi_apb_error_cfg

Bit	Symbol	Access	Value	Description
31 to 2	-	-	0*	To be ignored when read, written as 0's
1	wr_error_en	R/W		Enables/disables APB error generation for write transfers
			0	Disable write error generation
			1*	Enable write error generation
0	rd_error_en	R/W		Enables/disables APB error generation for read transfers
			0	Disable read error generation
			1*	Enable read error generation

Table 235: Register ebi_apb_latency_cfg

Bit	Symbol	Access	Value	Description
31 to 1	-	-	0*	To be ignored when read, written as 0's
0	nr_of_wait_state_s	R/W		APB latency control
			0	no additional wait state insertion; minimum read/write latency is 1 APB clock cycle
			1*	Insert one additional wait state; minimum read/write latency is 2 APB clock cycles

Table 236: Register ebi_sw_resetcfg

Bit	Symbol	Access	Value	Description
31 to 1	-	R	0*	reserved
0	sw_reset	R/W		SW reset: when enabled, initiates a SW reset of the EBI - this bit is automatically cleared by the HW after the reset has been performed - it is possible to poll this bit to check when it is cleared
			0*	reset disabled
			1	reset enabled

Table 237: Register ebi_power_downcfg

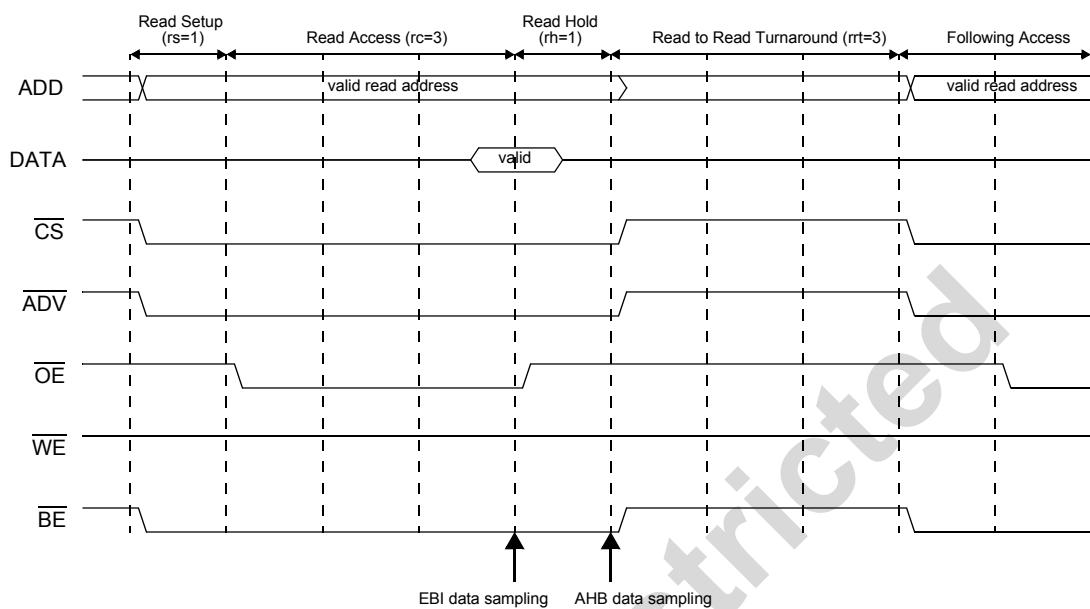
Bit	Symbol	Access	Value	Description
31 to 17	-	R	0*	reserved
16 to 1	time_out[15:0]	R/W		<p>timeout: this field defines the timeout value for a VPB access to a register in the AHB clock domain when the timeout counter expires, the pending VPB transaction are completed, but the result of the transaction is unknown (i.e., write data may or may not have been written; read data may or may not have been updated)</p> <p>0x0* disabled</p> <p>0x1 timeout duration the value of the field in hclk periods (max 65535 periods)</p>
0	power_down	R/W		<p>powerdown: when enabled, the EBI immediately stop being accessible in the AHB and VPB domains - written data are ignored and read accesses return zeroes</p> <p>0* disabled</p> <p>1 enabled</p>

9.14.5 Functional Description

9.14.5.1 Asynchronous mode

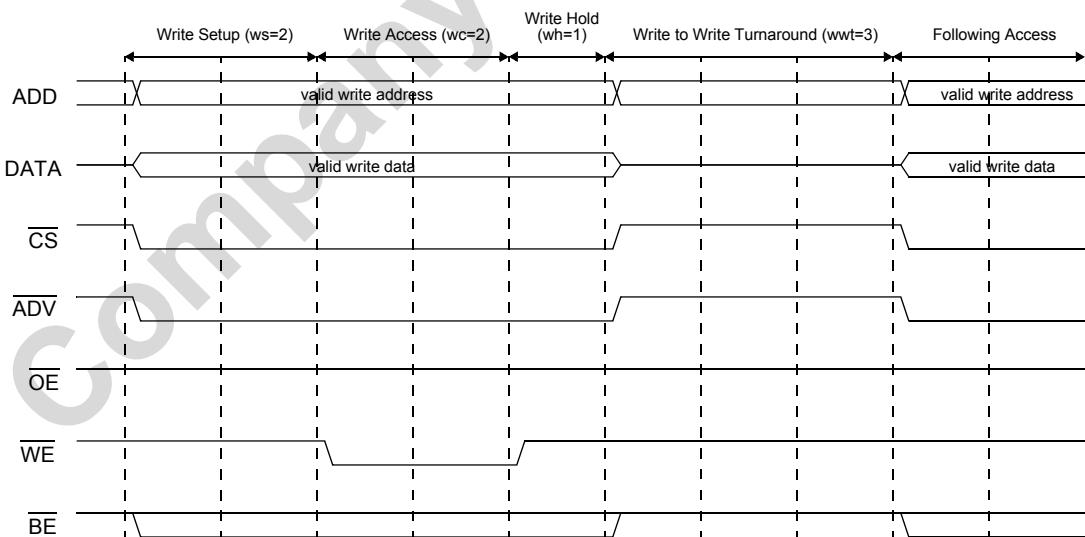
The asynchronous mode is used with asynchronous devices with \overline{OE} and \overline{WE} pins. Various device parameters, including device width, read timings (read setup time, read access time, read hold time and read turnaround time) and write timings (write setup time, write access time, write hold time and write turnaround time) can be programmed in the configuration registers for the corresponding device. Zero setup and hold time is also supported.

The CS signal goes inactive after every transfer is completed.



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The \overline{BE} signals in this diagram are drawn for \overline{BCS} function.

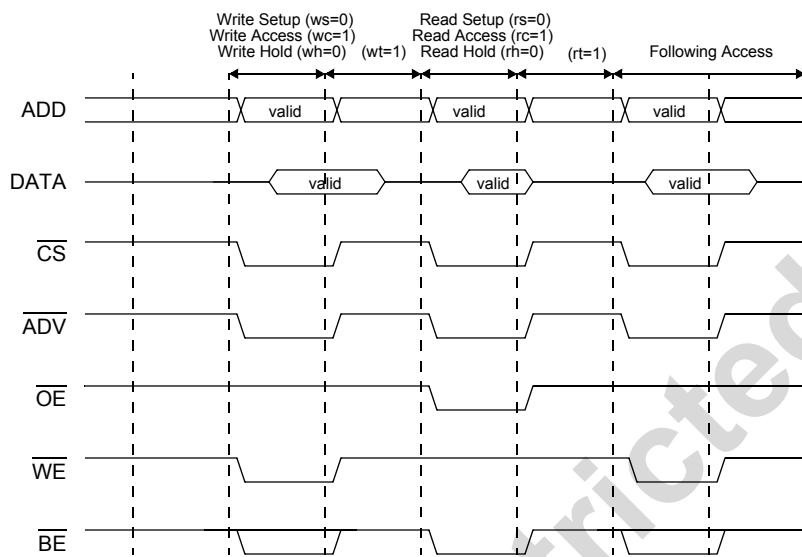
Fig 72. EBI cycle-based timings for asynchronous read access (non-multiplexed)



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The \overline{BE} signals in this diagram are drawn for \overline{BCS} function.

Fig 73. EBI cycle-based timings for asynchronous write access (non-multiplexed)

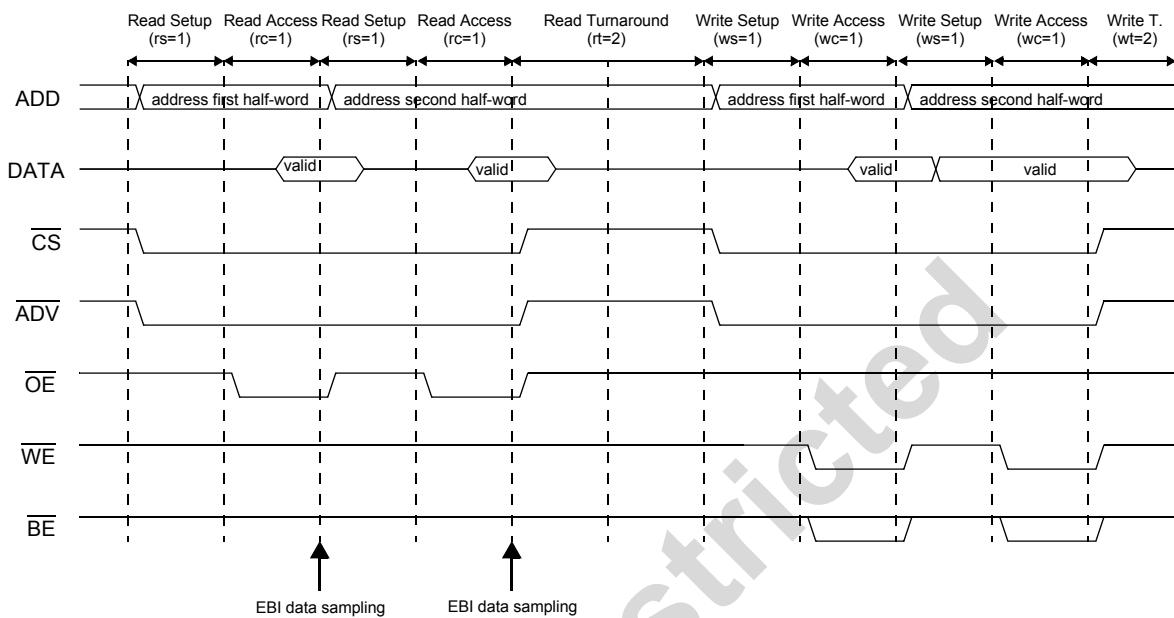
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
(2) The \overline{BE} signals in this diagram are drawn for \overline{BCS} function.

Fig 74. EBI cycle-based timings for short asynchronous read and write access

Figure 74 shows an example where zero setup and hold time is supported: this only works for 16-bit accesses to 16-bit devices (or when the access size is equivalent to the device size).



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read hold parameter is chosen $rh = 0$ in this diagram.
- (3) A similar diagram is performed when 32 bits are read from a 8-bit device (4 accesses) and when 16 bits are read from a 8-bit device (2 accesses).
- (4) The **BE** signals in this diagram are drawn for **BWE** function.

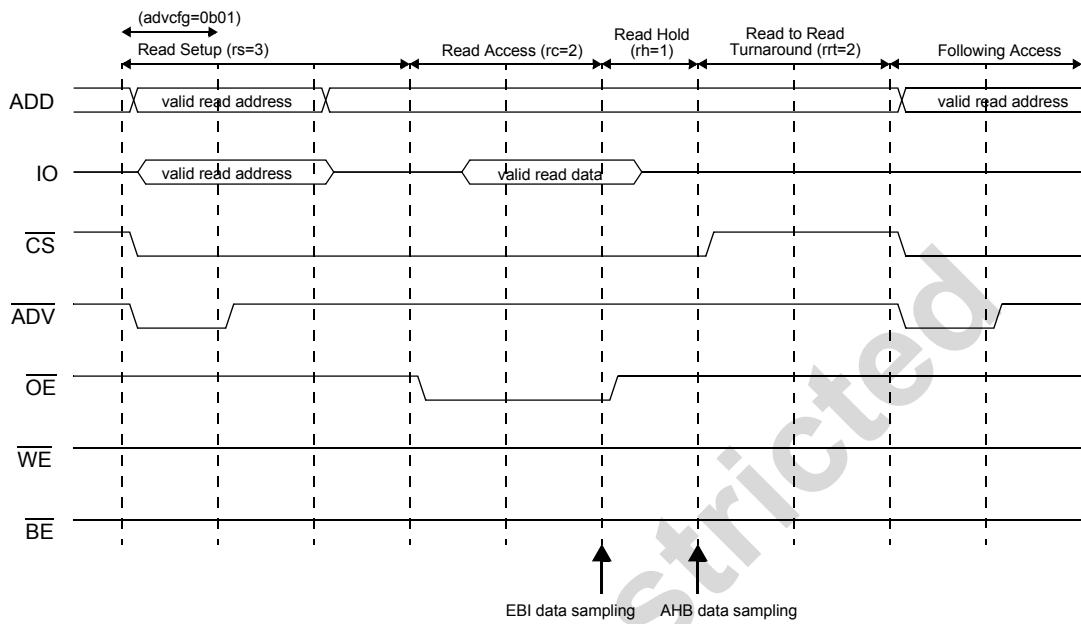
Fig 75. EBI cycle-based timings for asynchronous 32-bit read access to 16-bit device

Remark: For asynchronous 8-bit write accesses to a 16/32-bit device the **CS** and **ADV** remain low for subsequent accesses to achieve a higher bandwidth. Similar behaviour appears for asynchronous 16-bit write accesses to a 32-bit device.

9.14.5.2 Asynchronous latched address mode

This mode is used with asynchronous devices which latch the address data with the rising edge of the **ADV** signal. This kind of devices only requires a valid address around the **ADV** pulse. All timing parameters valid for the asynchronous mode are also valid for the asynchronous latched mode.

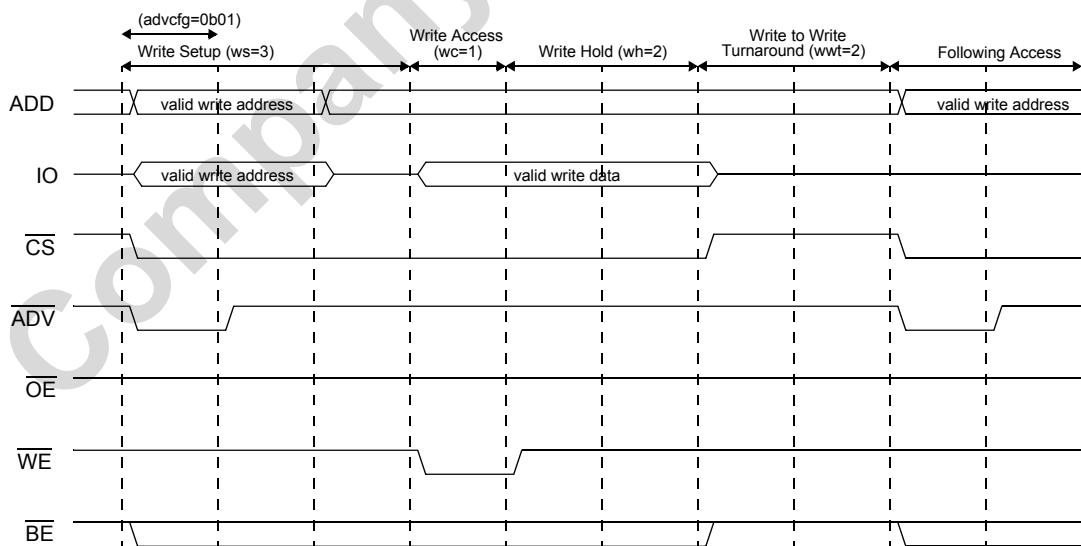
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



(1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.

(2) The \overline{BE} signals in this diagram are drawn for \overline{BWE} function.

Fig 76. EBI cycle-based timings for asynchronous latched read access



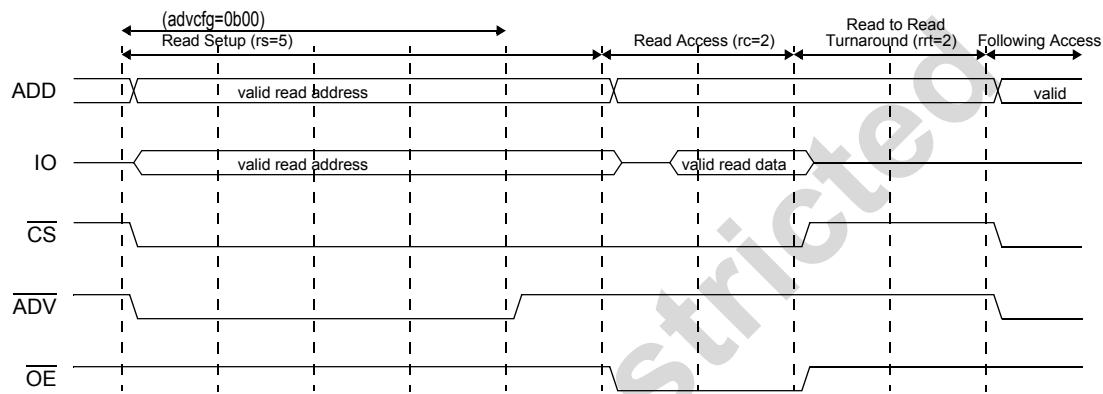
(1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.

(2) The \overline{BE} signals in this diagram are drawn for \overline{BCS} function.

Fig 77. EBI cycle-based timings for asynchronous latched write access

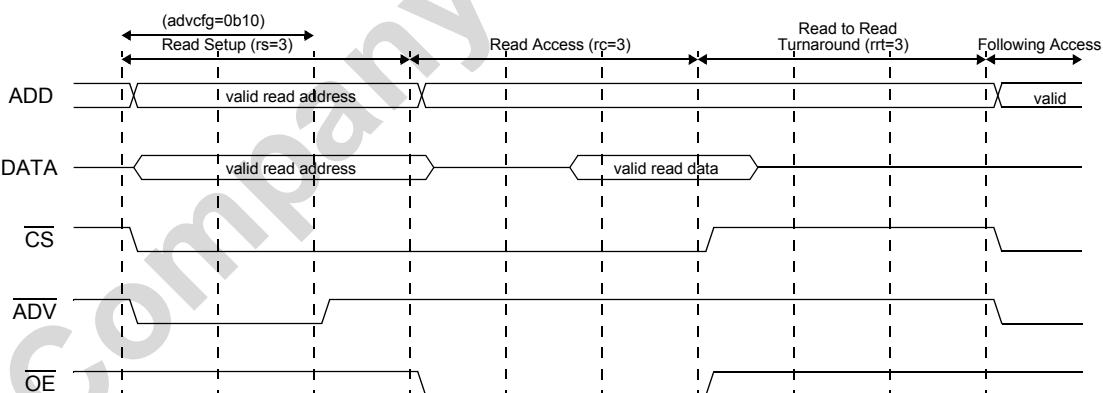
The asynchronous latched operation is always multiplexed (refer to [Section 9.14.5.7 “x16 multiplexed mode”](#) and [Section 9.14.5.8 “x32 multiplexed mode”](#)). The write data is driven on the bus one cycle after the address has been invalidated.

For asynchronous latched it is possible to define the duration of the address ADD valid and the length of the ADV pulse. The granularity is defined in multiples of the AHB clock in the **advcfg[1:0]** field.



(1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.

Fig 78. ADV pulse length setting $\text{advcfg} = 0b00$



(1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.

Fig 79. ADV pulse length setting $\text{advcfg} = 0b10$

9.14.5.3 Page mode

The page mode is used with asynchronous memories. The read access time indicates the initial access time and the page read access time indicates the access time for all following accesses.

A page mode access occurs when a read access is immediately followed by another read in the same page and the \overline{CS} has been active the entire time. \overline{CS} goes inactive if the addressed location is in a new page, if the transfer direction is changing to write or if the transfer is made on other device.

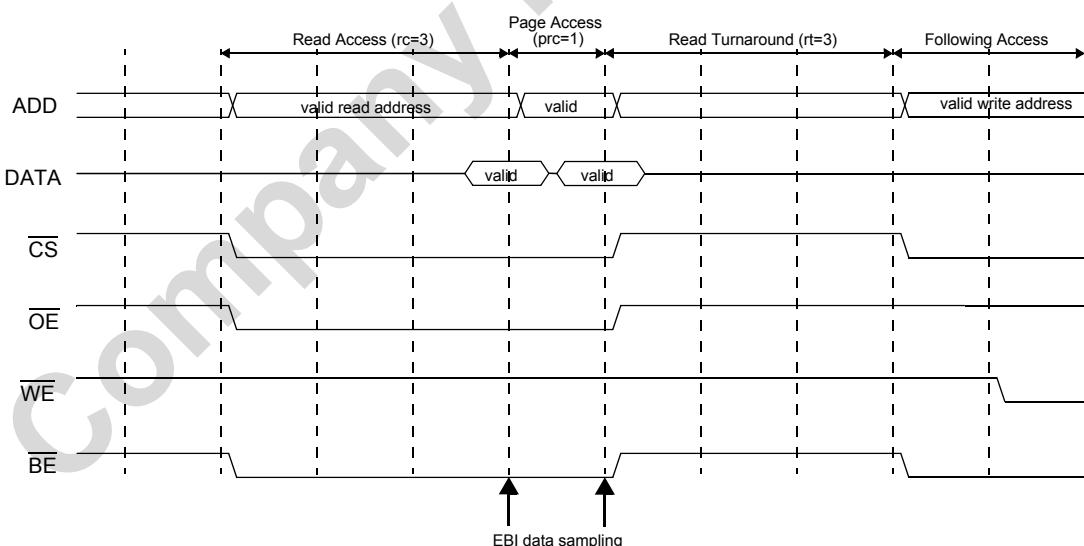
The configuration field **ebi_burstcfg.size** indicates how many data there are in a page. **size** can be configured to 4-data, 8-data, 16-data and 32-data. **Table 238** explains the valid memory address range within a page.

Table 238: Memory address valid by page^[1]

Size	Memory address bus
4-data	[0, 1, 2, 3], [4, 5, 6, 7], ...
8-data	[0, 1, 2, 3, 4, 5, 6, 7], [8, 9, A, B, C, D, E, F], ...
16-data	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F], ...
32-data	[0, 1, 2, 3, ..., E, F, 10, 11, ..., 1E, 1F], ...

[1] Pages are enclosed in brackets.

From the point of view of the SC, the addresses contained in a page depend also from the memory width. As an example, for **size** configured as 4-data with a memory width of 8 bits, then the valid AHB addresses range is from 0 to 3 or from 4 to 7, and so on. With a memory width of 16 bits, then the valid AHB addresses range is (0, 2, 4 and 6) or (8, A, C and E), and so on.



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read setup **rs** and read hold **rh** are ignored in page mode

Fig 80. EBI cycle-based timings for page read access

9.14.5.4 Synchronous mode

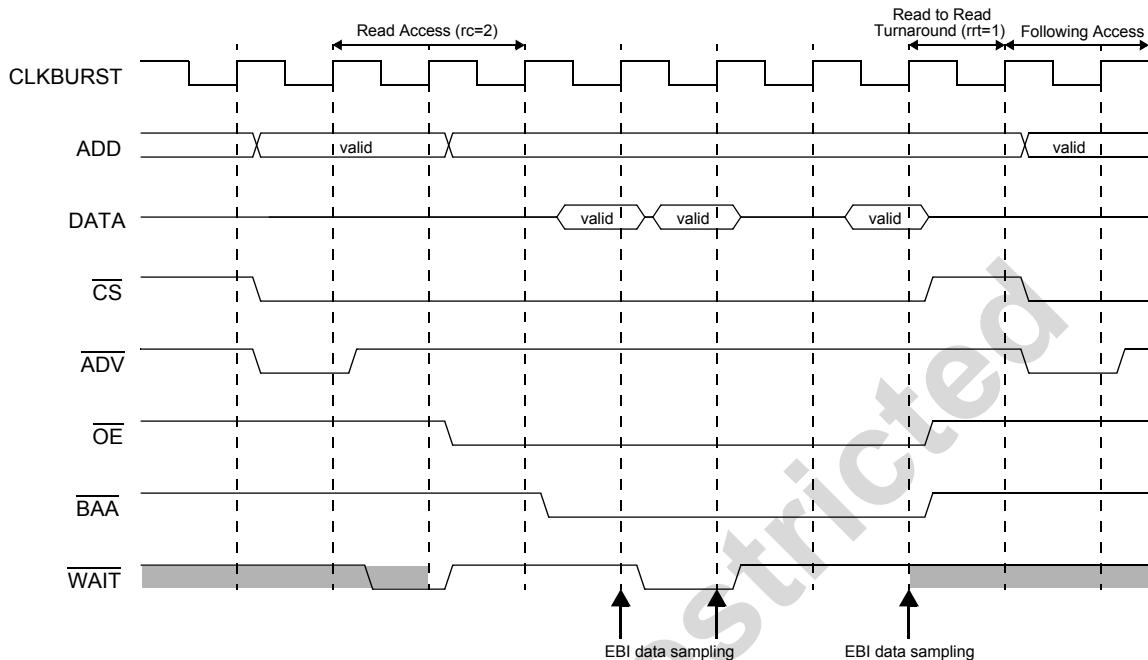
The synchronous mode is used with devices with transfer data based on a active clock edge. The synchronous mode allows high bandwidth data transfer, and is particularly useful for cache reload operations.

In this mode the read/write access time determines the number of cycles between the rising edge of ADV and the start of the bursd (falling edge of BAA). The read/write setup and read/write hold times are ignored. The addresses are always valid for two cycles, which means one cycle longer than the rising edge of ADV. This allows to support ADV latched addresses and clock latched addresses. The setting of **sync_cap_delay** further helps to fine-tune the capture delay is synchronous read accesses.

The **clkdiv** field indicates the clock frequency provided to the memory. The frequency depends on the internal **hclk**. Three configurations are available. For high speed memory it is possible to configure CLKBURST = hclk. For high AHB frequency or slow synchronous memory CLKBURST can be scaled down to hclk/2, hclk/4, or hclk/8. The CLKBURST clock can be enabled in-between bursts with **clken**.

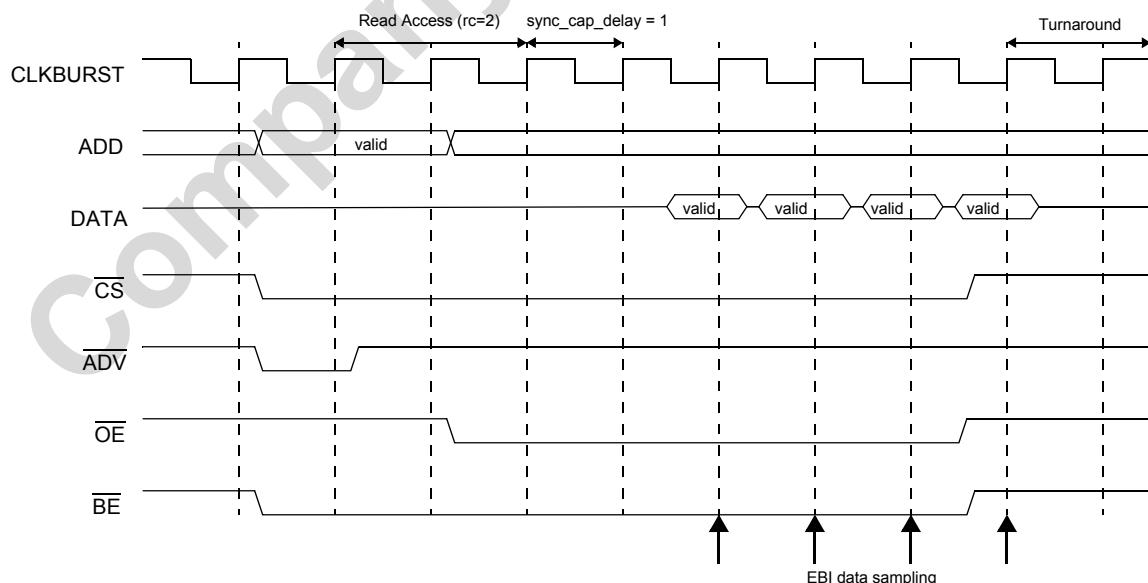
The **waiten** field indicates if the input signals WAIT is used by the EBI to control to the data flow. WAIT is sampled with the rising edge of CLKBURST and the memory device has to drive the signal active if it is not able to follow the speed of the EBI. If the memory drives WAIT low, access time is lengthened and becomes greater than programmed rc. WAIT is ignored at the beginning of the burst (as long as BAA is high). WAIT can be used to lengthen the first access if rc has been programmed to a low value.

The OE signal is always activated one cycle after the rising edge of ADV. The WE signal is always activated as long as CS is active. During synchronous write access CLKBURST, BAA, ADV, and WAIT have identical meaning than during synchronous read.



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read setup **rs** and read hold **rh** are ignored in synchronous mode.

Fig 81. EBI cycle-based timings for synchronous read access



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read setup **rs** and read hold **rh** are ignored in synchronous mode.

Fig 82. EBI cycle-based timings for synchronous read access with capture delay (WAIT not supported)

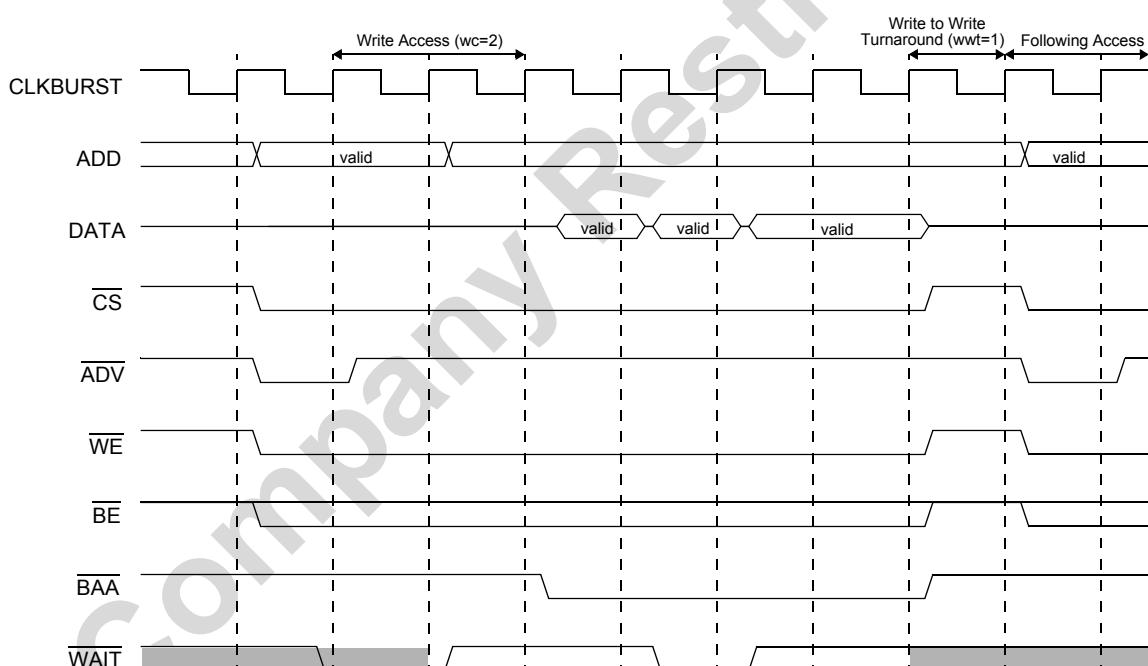
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The EBI supports all synchronous devices which are sensitive to the rising edge of the burst clock and assert the WAIT signal one cycle before the wait cycle.

The configuration field **ebi_burstcfg.size** indicates the maximum number of data which is requested by the EBI in a burst. **size** can be configured to 4-data, 8-data, 16-data, and 32-data. All burst accesses can be interrupted at any time by the EBI before the maximum burst size is reached. In such a case, the CS is deactivated after a reduced number of transfers. As a consequence, all burst sizes between 1 and **ebi_burstcfg.size** can be achieved.

Bursts of maximum size are only possible if they are aligned to the burst size itself. This means, that depending on the address of the first transfer, a burst can be split into two parts by the EBI: one burst lasts until the first **size** boundary, the second burst transfers the remaining data.

If the transfer size is smaller than the width of the connected memory (usage of BE as byte write enable), then there is no burst and only single accesses are performed.

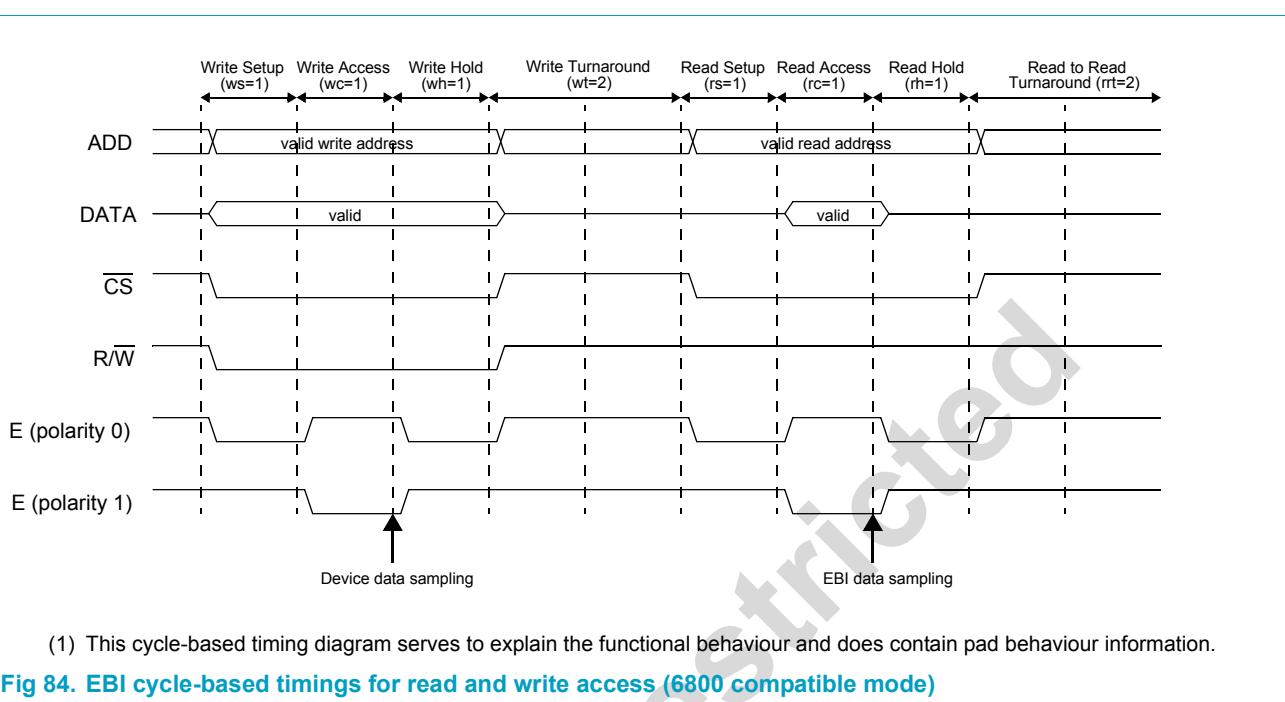


- (1) This cycle-based timing diagram serves to explain the functional behaviour and does not contain pad behaviour information.
- (2) The write setup ws and write hold wh are ignored in synchronous mode.

Fig 83. EBI cycle-based timings for synchronous write access

9.14.5.5 6800 compatible mode

The 6800 compatible mode is used to control asynchronous devices with E and R/W pins. All timing values (setup time, access time, hold time and turnaround time) are available in this mode. The polarity of the enable signal can be set in **ebi_maincfg** register. See [Figure 84](#) for the functional timing diagram.



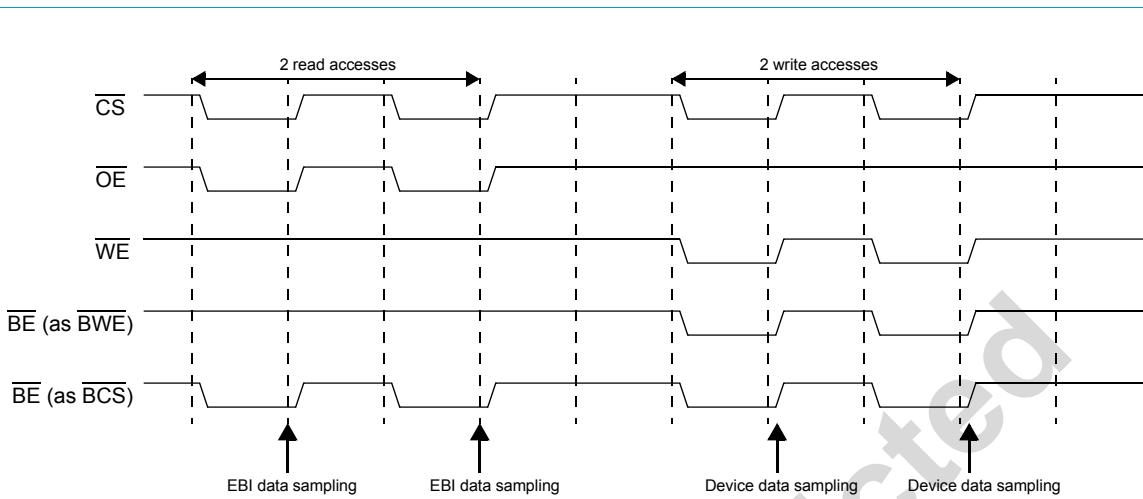
9.14.5.6 Byte enable behavior

The byte enable feature is used to perform byte accesses on 16-bit and 32-bit memories. There are four byte enable pins, two of which are multiplexed with the lowest addresses ADD[1:0]. The byte enables can operate as byte chip select BCS or as byte write enable BWE depending on the bit `ebi_maincfg.be`.

Table 239: Connection of external device byte enable signals

Device data bus width	EBI signal	Device signal
32 bits (multiplexed mode)	IO[0]	A0
	BE3	BE3
	BE1	BE1
	ADD1_BE2	BE2
	ADD0_BE0	BE0
16 bits (multiplexed mode)	IO[0]	A0
	BE1	BE1
	ADD0_BE0	BE0
16 bits	BE1	BE1
	ADD1_BE2	A0
	ADD0_BE0	BE0
8 bits	ADD0_BE0	A0

Figure 85 shows the behavior of the byte enables according to configuration BCS or BWE and according to access type. If a read access occurs and if BE is configured as BCS, then BE is asserted to a low level.



(1) This cycle-based timing diagram serves to explain the functional behaviour and does not contain pad behaviour information.

Fig 85. Byte enable functionality

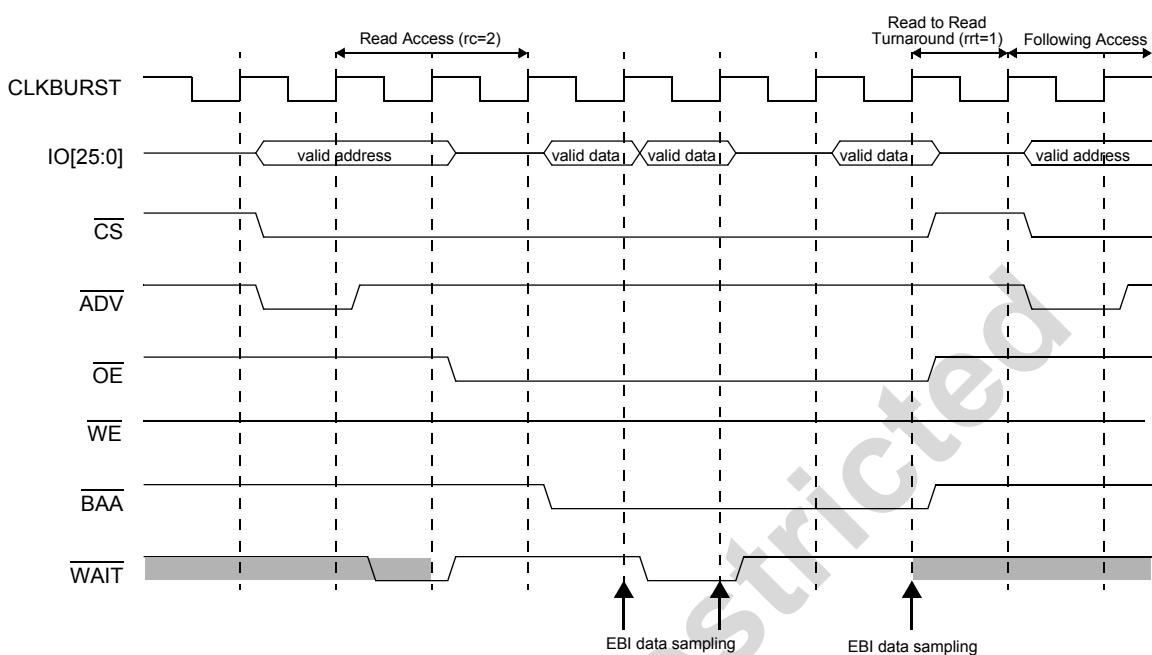
9.14.5.7 x16 multiplexed mode

The x16 multiplexed mode provides a reduction of the signals required to connect external devices. In this mode data and addresses are multiplexed on the same pins. This signals configuration is supported for asynchronous latched mode and for synchronous mode. The multiplexing which is required to perform this function is shown in [Table 240](#).

Table 240: Signal multiplexing matrix for x16 and x32

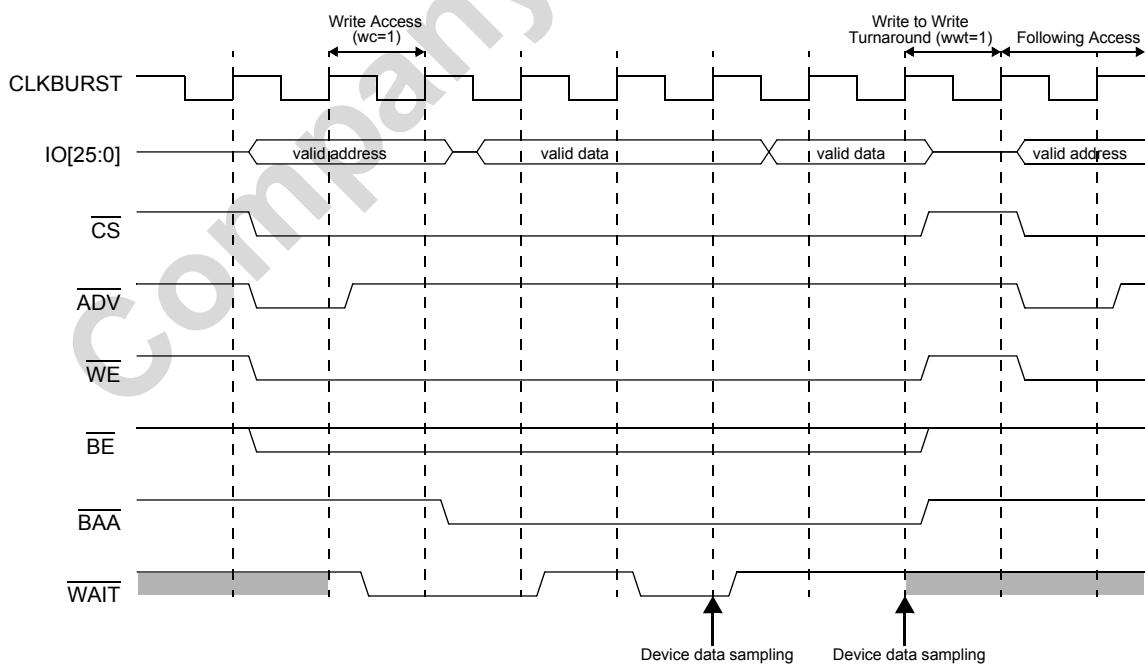
Name of EBI signal	# pins	No access	Function of signal in non multiplexed mode	Function of signal in x16 multiplexed mode	Function of signal in x32 multiplexed mode
ADD[27:18]	10	-	ADD[27:18]	-	-
IO[31:26]	6	-	ADD[17:12]	-	DATA[31:26]
IO[25:16]	10	-	ADD[11:2]	ADD[26:17]	ADD[27:18] / DATA[25:16]
IO[15:0]	16	-	DATA[15:0]	ADD[16:1] / DATA[15:0]	ADD[17:2] / DATA[15:0]
BE3	1	H	H	H	BE3
ADD1_BE2	1	H	ADD1	H	BE2
BE1	1	H	BE1	BE1	BE1
ADD0_BE0	1	H	ADD0_BE0	BE0	BE0

Please refer to [Section 9.14.5.2 “Asynchronous latched address mode”](#) to match the multiplexed operation with the asynchronous latched operation.



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read setup **rs** and read hold **rh** are ignored in synchronous mode.

Fig 86. EBI cycle-based timings for synchronous x16 multiplexed read access



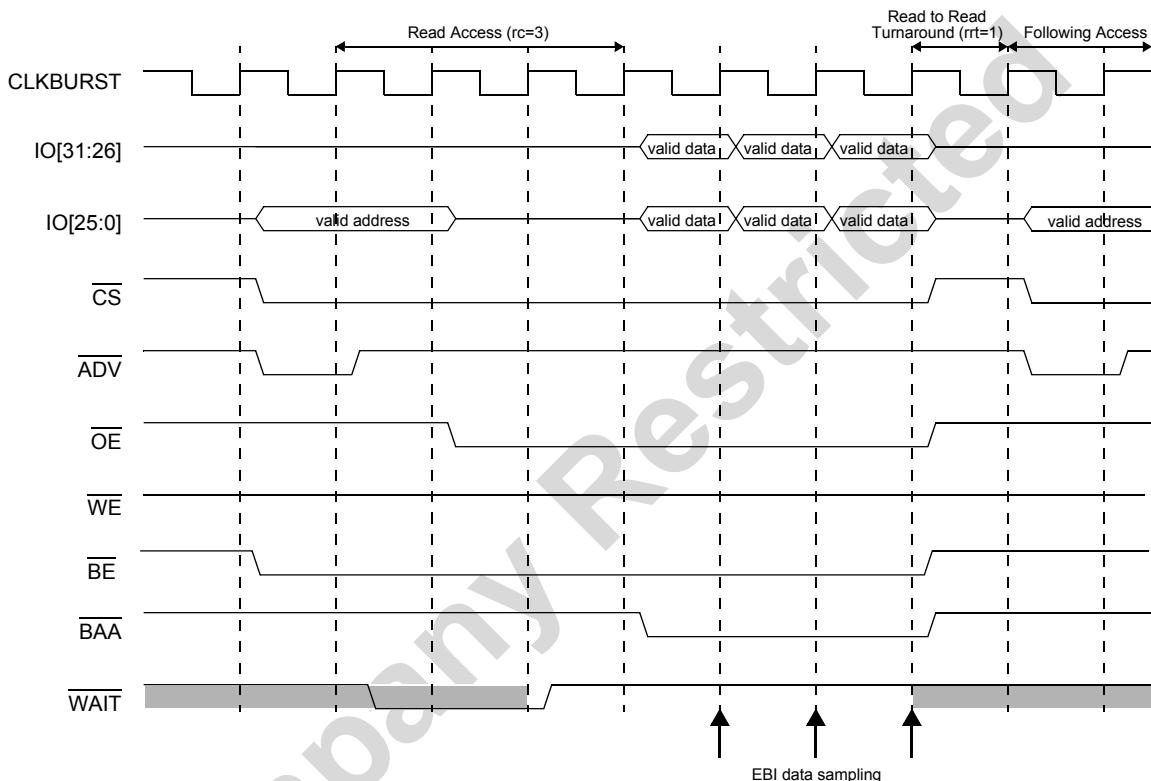
- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The write setup **ws** and write hold **wh** are ignored in synchronous mode.

Fig 87. EBI cycle-based timings for synchronous x16 multiplexed write access

Complementary document

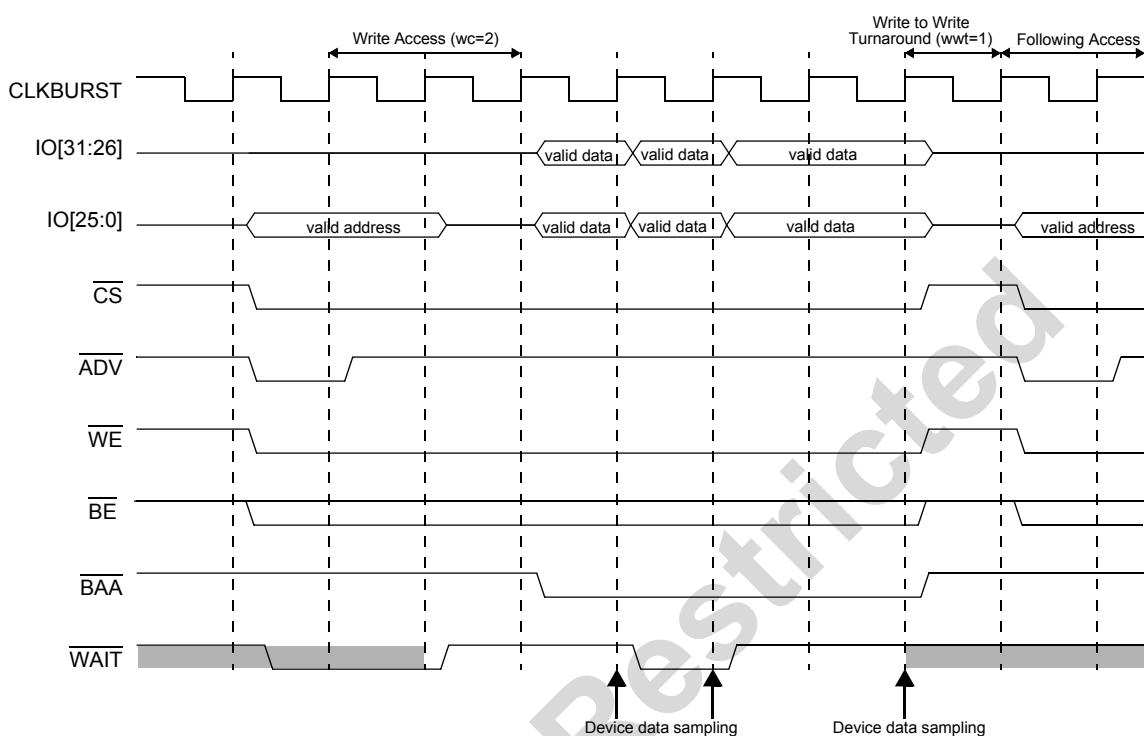
9.14.5.8 x32 multiplexed mode

The x32 multiplexed mode is similar to the x16 mode, but data is transferred with 32 bits width. The multiplexing which is required to perform this function is shown in Table 240.



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The read setup **rs** and read hold **rh** are ignored in synchronous mode.

Fig 88. EBI cycle-based timings for synchronous x32 multiplexed read access



- (1) This cycle-based timing diagram serves to explain the functional behaviour and does contain pad behaviour information.
- (2) The write setup **ws** and write hold **wh** are ignored in synchronous mode.

Fig 89. EBI cycle-based timings for synchronous x32 multiplexed write access

9.14.5.9 Turnaround time

for each EBI channel two turnaround times can be defined. The two counters are complementary having mutually exclusive application conditions. So **rrt** and **wwt** define the numbers of cycles inserted when the same type of access is performed on the same channel - **rt** and **wt** define the number of cycles inserted for any other type of cycle. This allows the user to fine tune the bus turnaround time when many devices of different nature are connected to the EBI.

The turnaround as mentioned in this section, is always meant to be part of the previous access. The consequence for this is that the effective bus tristate time depends on:

- the value programmed in **rrt/wwt/rt/wt**
- the access direction of the previous access
- the access direction of the next access
- (for synchronous operation) the speed defined for CLKBURST for the channel of the previous access
- internal latency cycles

The turnaround times which are programmed are the minimum guaranteed times. Because of internal latency cycles, the effective turnaround cycles can be longer than programmed.

In case **rrt/rt** and **wwt/wt** define different lengths of turnaround, the validity of the longer turnaround expires if there is an access with the shorter turnaround. For consistency of operation, it is mandatory to program **rt >= rrt** and **wt >= wwt**.

In synchronous mode, the CLKBURST during the turnaround time has the same frequency as for the previous access.

9.14.6 Application information

9.14.6.1 Connection tables

This section serves as reference for the connection to several types of devices.

Table 241: Example connections

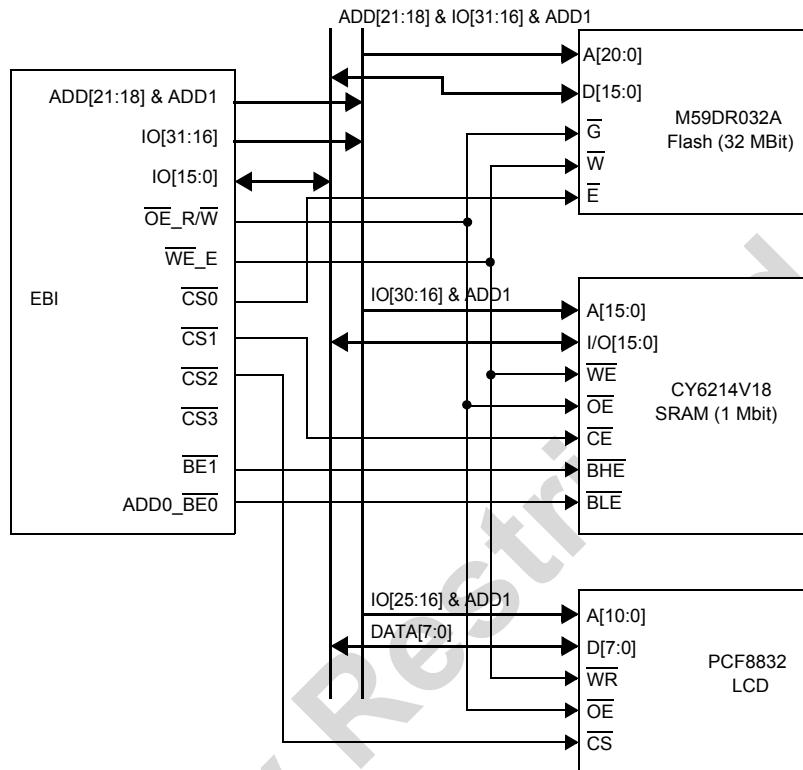
Name of signal on EBI interface	Inapac 16 Mbit non multiplexed	Inapac 16 Mbit x32 mode	CellularRAM 2M x 16 bit non multiplexed	CellularRAM 4M x 16 bit non multiplexed
ADD[26:18]	A[18:16]	tristate	A[20:17]	A[21:17]
IO[31:24]	A[15:8]	D[31:24]	A[16:9]	A[16:9]
IO[23:16]	A[7:0]	D[23:19] + AD[18:16]	A[8:1]	A[8:1]
IO[15:0]	D[15:0]	AD[15:0]	DQ[15:0]	DQ[15:0]
BE3	H	BE3	-	-
ADD1_BE2	A[-1]	BE2	A[0]	A[0]
BE1	BE1	BE1	UB	UB
ADD0_BE0	BE0	BE0	LB	LB

Table 242: Example connections

Name of signal on EBI interface	Intel 128M x32 mode	Intel 256M x32 mode	Intel 512M x32 mode
ADD[26:18]	tristate	tristate	tristate
IO[31:24]	DATA[31:24]	DATA[31:24]	DATA[31:24]
IO[23:16]	DATA[23:22] + AD[21:16]	DATA[23] + AD[22:16]	AD[23:16]
IO[15:0]	AD[15:0]	AD[15:0]	AD[15:0]
BE3	-	-	-
ADD1_BE2	-	-	-
BE1	-	-	-
ADD0_BE0	-	-	-

9.14.6.2 EBI register settings for an application example

Figure 90 shows an application example of the EBI with the **hclk** running at 52 MHz. The connected devices are a 16-bit Flash memory M59DR032A, a 16-bit SRAM CY6214V18 from with separate high byte/low byte enable (BHE/BLE), and an 8-bit LCD driver PCF8832.



(1) For this application example the following operating conditions apply: VDDM = 1.8 V and SC frequency = 52 MHz.

Fig 90. EBI application example

Register settings for CS0:

The flash memory must always be connected to CS0 in order to allow fast jumps from the on-chip SRAM to the external flash and vice versa. In the application example the flash memory M59DR032A from ST is used. In the following, register settings for **maincfg0**, **readcfg0**, **writecfg0** and **burstcfg0** are derived from this flash data sheet information.

- **mode[2:0]**: this flash allows page mode, **mode[2:0]** = 0b001
- **size[1:0]**: the page size of this memory is 4 data = 8 bytes; accordingly **size[1:0]** = 0b00
- **msize[1:0]**: the flash is a 16-bit device, therefore **msize[1:0]** = 0b01
- **wwt[3:0]**: one turnaround time has to be introduced, **wwt[3:0]** = 0b0001
- **wc[4:0]**: the data sheet specifies the minimal time the **HWR** signal is low as $t_{WLWH} = 50$ ns; three write access cycles have to be introduced **wc[4:0]** = 0b00011
- **ws[1:0]**: the data sheets specifies the write setup time $t_{ELWL} = 0$ ns; therefore **ws[1:0]** = 0b00
- **wh[1:0]**: the data sheets specifies the write hold time $t_{WHEH} = 0$ ns; therefore **wh[1:0]** = 0b00

- **prc[2:0]**: the page access time of the memory is $t_{page} = 35$ ns; two cycles have to be introduced **prc[2:0] = 0b0010** 1
- **rc[4:0]**: random read access time is $t_{ELQY} = 100$ ns translating into 6 cycles, **rc[4:0] = 0b00110** 2
- **rs[1:0]**: the read setup time is 0 ns; therefore **rs[1:0] = 0b00** 3
- **rh[1:0]**: the read hold time is 0 ns; therefore **rh[1:0] = 0b00** 4
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 5
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 6
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 7
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 8
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 9

Register settings for CS1:

In the application example the external SRAM (CY6214V18 from Cypress) is connected to CS1. In the following the register settings for **maincfg1**, **readcfg1**, **writecfg1** and **burstcfg1** are derived based on data sheet information.

- **mode[2:0]**: no page or synchronous mode capability, **mode[2:0] = 0b000** 15
- **size[1:0]**: not applicable 16
- **msize[1:0]**: the SRAM is a 16-bit device, therefore **msize[1:0] = 0b01** 18
- **wwt[3:0]**: one turnaround time has to be introduced, **wwt[3:0] = 0b0001** 19
- **wc[4:0]**: the data sheet specifies the minimal time the HWR signal is low as $t_{PWE} = 10$ ns; one write access cycles has to be introduced **wc[3:0] = 0b00001** 20
- **ws[1:0]**: the data sheets specifies the write setup time $t_{SA} = 0$ ns; therefore **ws[1:0] = 0b00** 23
- **wh[1:0]**: the data sheets specifies the write hold time as 0 ns; therefore **wh[1:0] = 0b00** 25
- **prc[2:0]**: not applicable 27
- **rrt[3:0]**: one turnaround time has to be introduced, **rrt[3:0] = 0b0001** 29
- **rc[4:0]**: read access time is $t_{RC} = 15$ ns translating into 1 cycle, **rc[4:0] = 0b00001** 30
- **rs[1:0]**: the read setup time is 0 ns; therefore **rs[1:0] = 0b00** 32
- **rh[1:0]**: the read hold time is 0 ns; therefore **rh[1:0] = 0b00** 33
- **byte enables**: the SRAM is a 16-bit device with separate low-byte/high-byte enable signals with two active low enables; therefore **be = 0**. 34

Register settings for CS2:

In the application example an LCD driver (PCF8832 from NXP Semiconductors) is connected to CS2. In the following the register settings for **maincfg2**, **readcfg2**, **writecfg2** and **burstcfg2** are derived based on data sheet information.

- **mode[2:0]**: no page or synchronous mode capability, **mode[2:0] = 0b000** 42
- **size[1:0]**: not applicable 43
- **msize[1:0]**: the LCD-driver is a 8-bit device, therefore **msize[1:0] = 0b00** 45
- **wwt[3:0]**: one turnaround time has to be introduced, **wwt[3:0] = 0b000** 46
- **wc[4:0]**: the data sheet specifies the minimal time the HWR signal is low as $t_{CCLW} = 20$ ns (80 type); two write access cycles has to be introduced **wc[4:0] = 0b00010** 48
- **wc[4:0]**: the data sheet specifies the minimal time the HWR signal is low as $t_{CCLW} = 20$ ns (80 type); two write access cycles has to be introduced **wc[4:0] = 0b00010** 49

- **ws[1:0]**: the data sheets specifies the address setup time $t_{SA} = 30$ ns; therefore $\text{ws}[1:0] = 0b10$
- **wh[1:0]**: the data sheets specifies the write hold time as $t_{DH} = 10$ ns; therefore $\text{wh}[1:0] = 0b01$
- **prc[2:0]**: not applicable
- **rrt[3:0]**: one turnaround time has to be introduced, $\text{rrt}[3:0] = 0b0001$
- **rc[4:0]**: read access time is $t_{ACC} = 70$ ns translating into 4 cycles, $\text{rc}[4:0] = 0b00100$
- **rs[1:0]**: the read setup time is 0 ns; therefore $\text{rs}[1:0] = 0b00$
- **rh[1:0]**: the read hold time is 0 ns; therefore $\text{rh}[1:0] = 0b00$
- **be**: not applicable.

9.14.6.3 Asynchronous mode connection

One 16-bit external memory device can be made of two 8-bit banks. The $\overline{\text{BE}}$ can be configured as the $\overline{\text{BWE}}$ or as the $\overline{\text{BCS}}$.

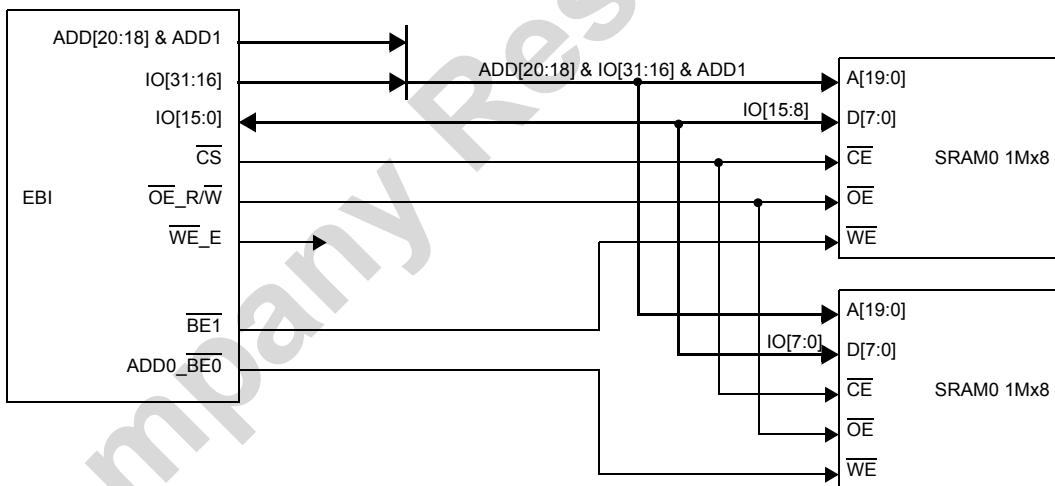


Fig 91. 1Mx16 SRAM with two banks ($\overline{\text{BE}}$ as $\overline{\text{BWE}}$)

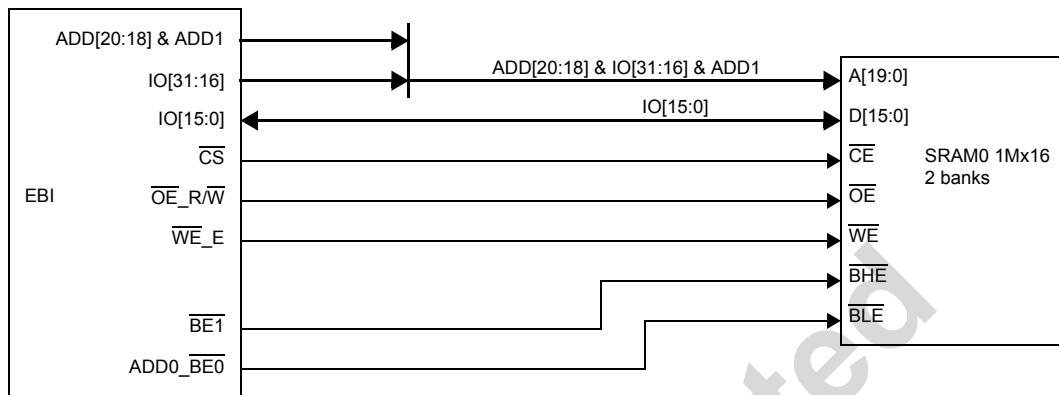
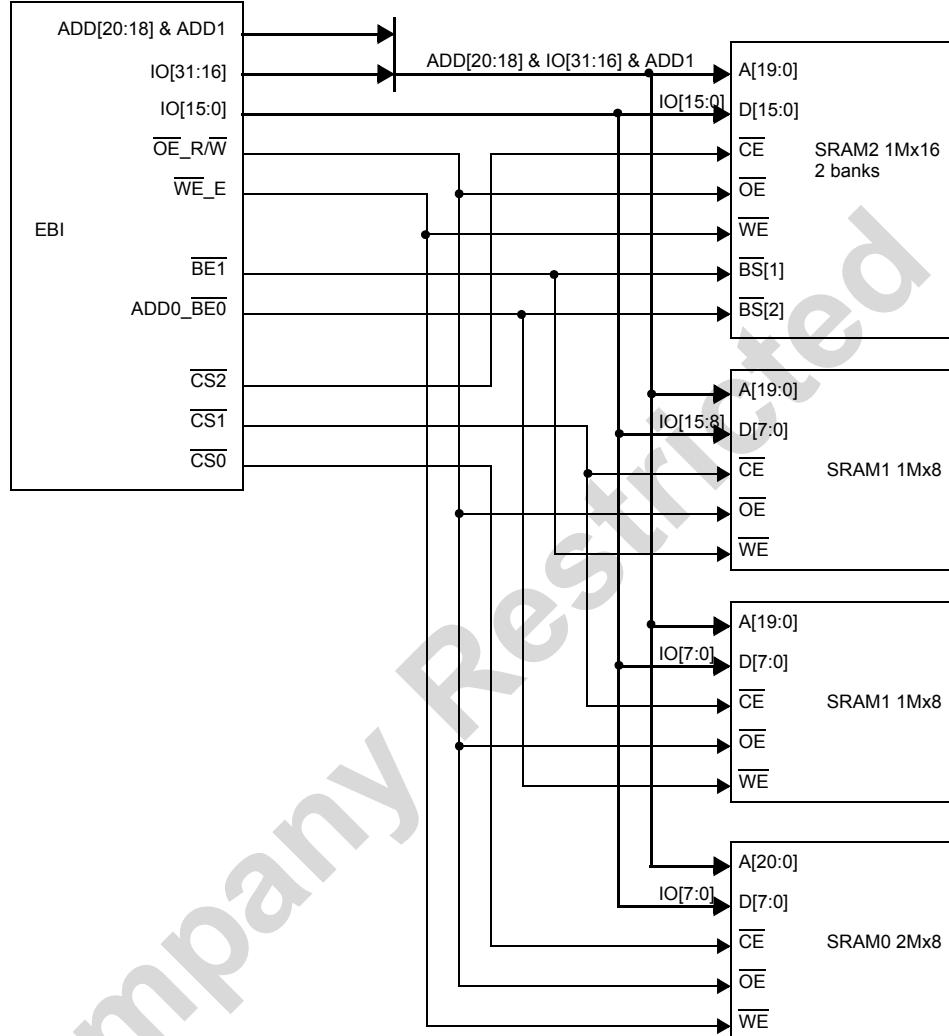


Fig 92. 1Mx16 SRAM with two banks (\overline{BE}_1 as \overline{BCS})

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



In this scheme there are three external memories: SRAM0 2Mx8, SRAM1 1Mx16 with two byte banks, and SRAM2 1Mx16 with byte enable support. Three chip selects CS0, CS1 and CS2 are used. The output enable can be shared for the three memories. The write enable can be shared for SRAM0 and SRAM2. Two byte write enables (BWE) are used for SRAM1. Two byte chip select (BCS) are used for SRAM2.

Fig 93. 2Mx8, 2x1Mx8 and 1Mx16 SRAM with shared OE, WE and BE

9.14.6.4 Synchronous memory connection

Company Approved

In this section the connection to the AM29BDS640G (Figure 94), to the 28F640W18 (Figure 95), and to the MT45W2MW16BFB (Figure 96) is shown. Note that the wait signal connection is required.

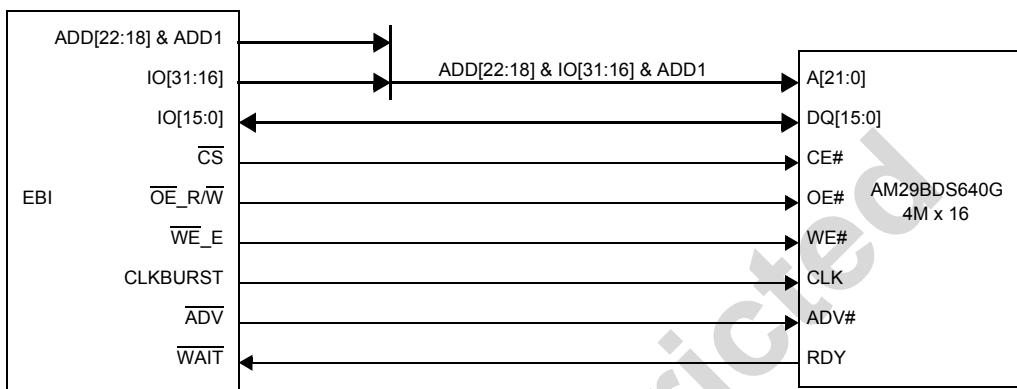


Fig 94. AM29BD640G synchronous Flash connection

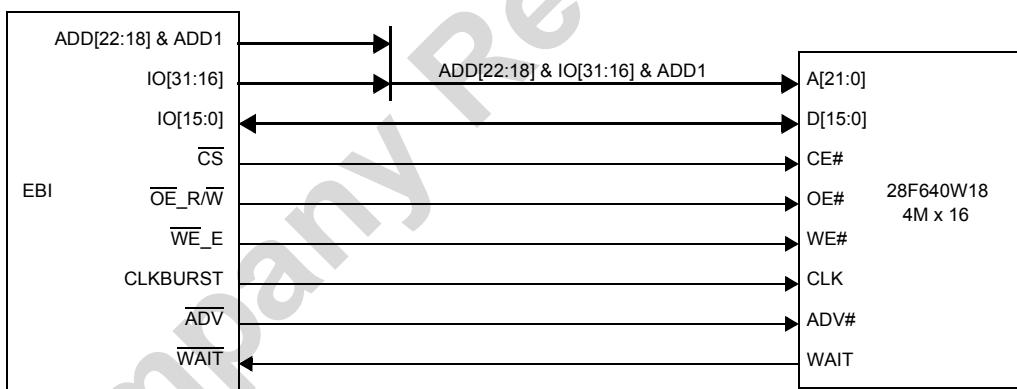


Fig 95. 28F640W18 synchronous Flash connection

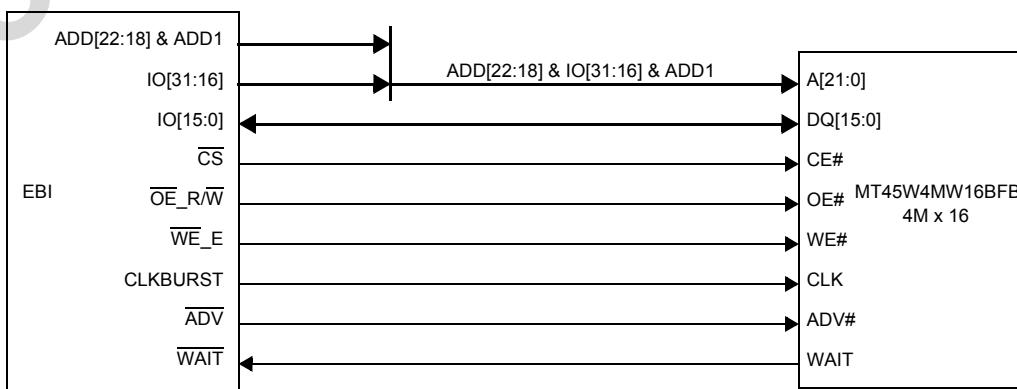


Fig 96. MT45W2MW16BFB synchronous RAM connection

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.14.6.5 Multiplexed device connection

This connection example shows the application with the Am29N128H device.

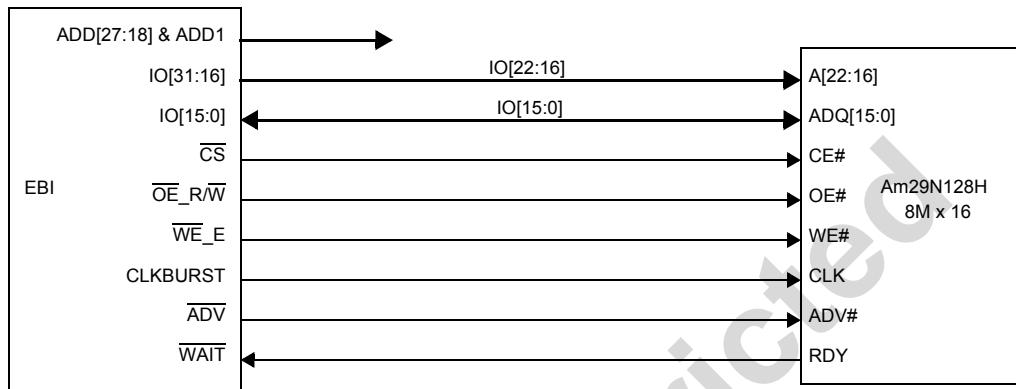


Fig 97. Am29N128H synchronous multiplexed Flash connection

9.14.6.6 6800 compatible mode connection

Some LCD drivers work in 6800 mode like the SED1565. The following diagram shows how to interface the EBI with a 8-bit parallel LCD driver. A[0] determines if the access corresponds to a command or a data transfer.

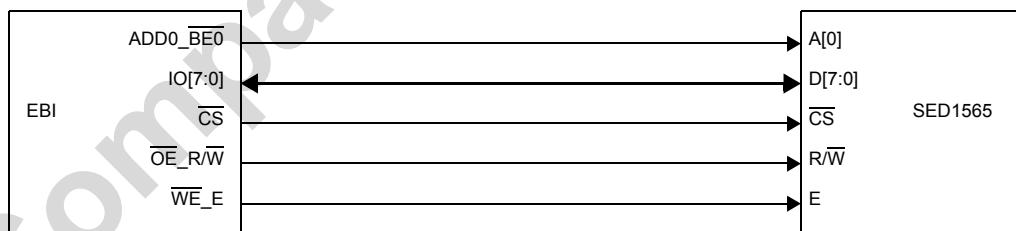


Fig 98. 6800 compatible mode connection

9.14.6.7 NAND-Flash connection

The EBI supports 8-bits and 16-bit NAND-Flash devices. However several control signals must be emulated with other pins, and the required error correction for must be made by SW.

Figure 99 shows an example connection with the Samsung K9F5616Q0B. CLE and ALE can be emulated with address pins, while the connection of the R/B signal is optional as the status can be read through data bus with a specific command. Alternative a GPIO can be used as chip enable signal of the NAND flash.

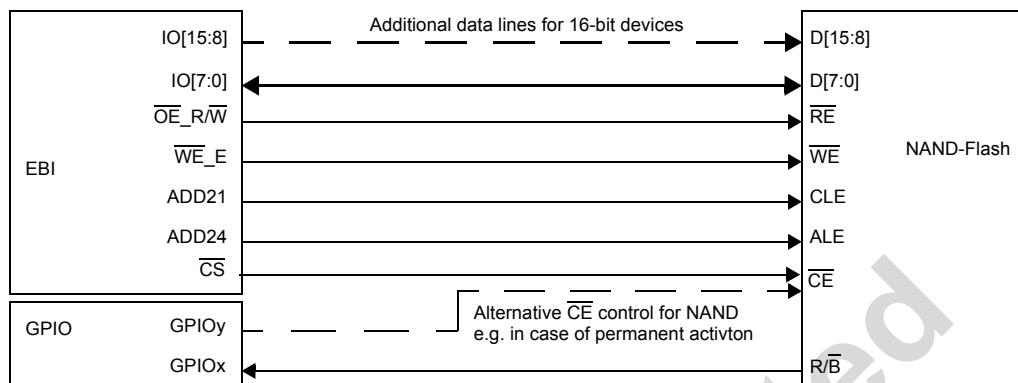


Fig 99. NAND-Flash device connection (recommended for DVFD818x Family)

Table 243:NAND-Flash addressing

Offset address	ADD18 = ALE	ADD20 = CLE	Comment
+0x0000 0000 to +0x0003 FFFF (e.g. can follow internal address in SCRAM to ease debugging)	0	0	data access
+0x0004 0000 to +0x0007 FFFF (recommended offset to ease debug = 0x0007 FFFF)	1	0	address access (write only)
+0x0010 0000 to +0x0013 FFFF (recommended offset to ease debug = 0x13 FFFF)	0	1	command access (write only)

[1] Other address values are not allowed because activation ALE and CLE at same time is not allowed by the NAND-Flash.

9.14.6.8 LCD connection example

The EBI supports 8-bits and 16-bit LCD devices. The Figure 100 shows an example of the connections scheme.

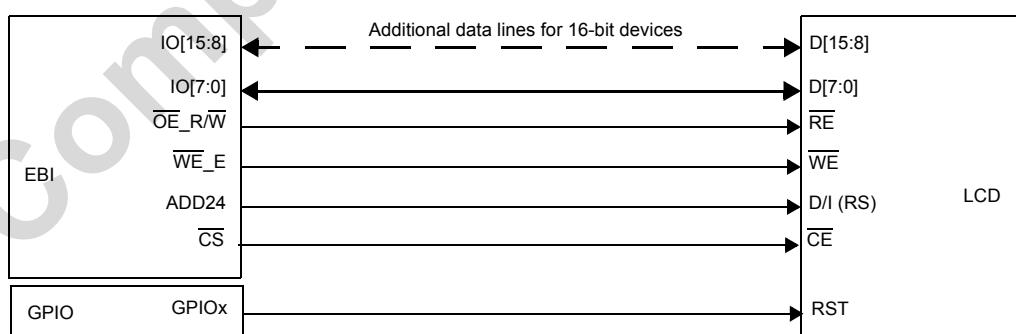


Fig 100.LCD device connection

Table 244:LCD addressing example

Offset address	ADD24 = D/I (RS)	Comment
+0x0000 0000 to +0x00FF FFFF	0	Consult the LCD datasheet for function on D/I or RS
+0x0100 0000 to +0x01FF FFFF	1	

9.14.6.9 Unwanted generation of memory clock at CLOCK_BURST_OUT

If the field **ebi_always_clkcfg.always_clk_en** is set to “0” than it is expected that the
memory clock is running only during a synchronous transfer and during the
subsequent turnaround time. However, after every synchronous transfer the clock
continues to run until either a reset or any asynchronous transfer starts at any of the
devices controlled by the EBI. However, there is still a rising and falling clock edge
present when the chip select goes low for the asynchronous transfer.

To avoid unwanted power consumption the SW should execute a dummy
asynchronous read immediately before entering any low-power state.

Company Restricted

9.15 ETM and ETB - Embedded Trace Macrocell and Buffer

Caution

ETM is not supported at DVFD8185 and DVFD8187.



9.15.1 Features

- ETM9 revision 2a compliant [14.]
- ETM v1.3 architecture
- 4, 8 or 16-bit trace data port ¹
- ARM ETM large implementation
 - FIFO depth 45 bytes
 - 8 pairs of address comparators
 - 8 data comparators with mask
 - 16 memory map decoders
 - 4 x 16-bit counters
 - 1 three-stage sequencer
 - 1 external input and 1 external output
- Full or half-rate clocking
- Demultiplexed mode with half-rate clocking
 - Two 4-bit trace data ports allowing ARM926EJ-S tracing with ETM external signalling rate four time lower than armclk
- Can use FIFO full stall on selected area
- Can use EmbeddedICE-RT module watch points for trace triggering
- Trace of Context ID change. See ARM926EJ-S reference manual for details [12.]

The ETB block offers the following features:

- dedicated on-chip 4k x 24-bit trace buffer for high speed systems

1. 8 or 16-bit trace port might reduce available features (e.g. ETN) due to additional pin needed for such modes

9.15.2 Block diagram

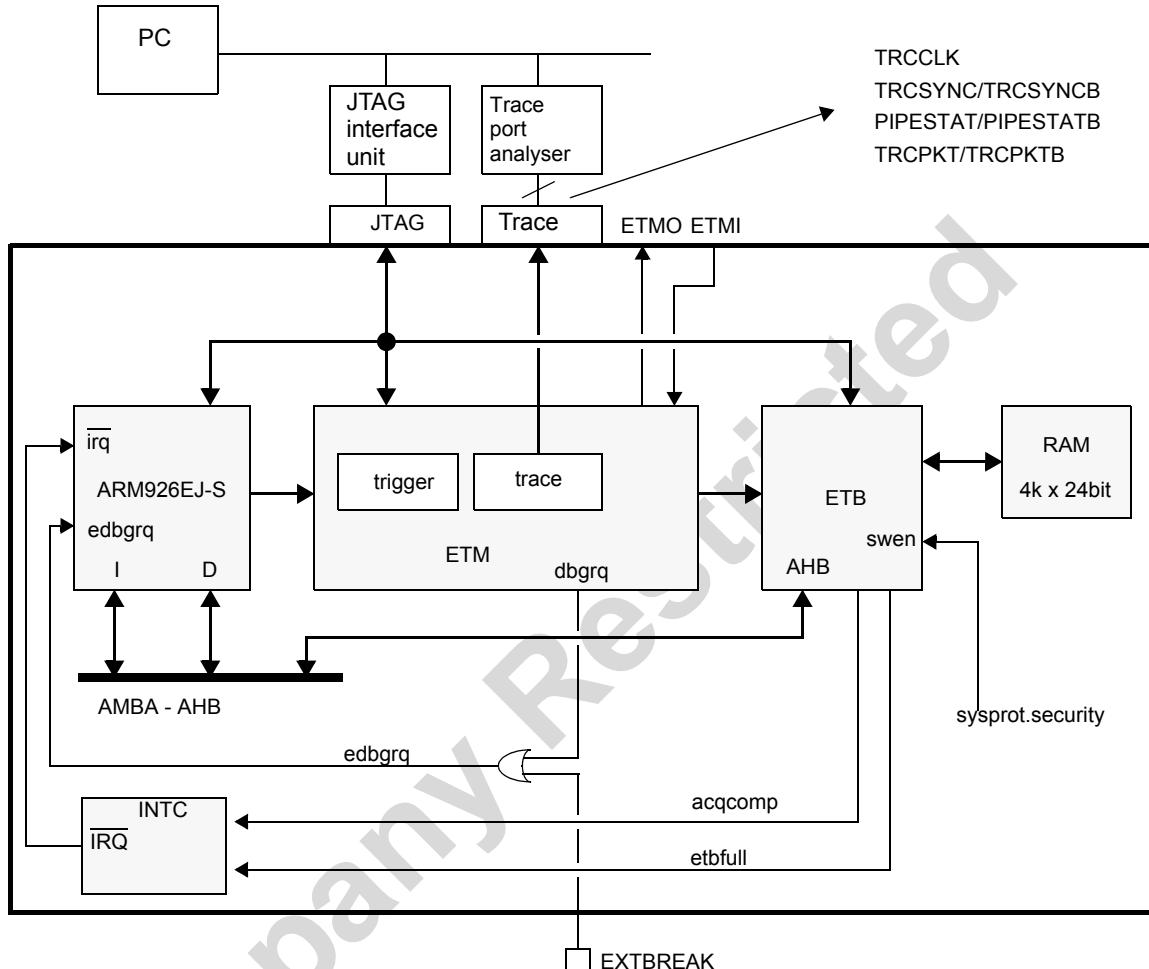


Fig 101.ETM / ETB block diagram

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.15.3 Hardware interface

Table 245: ETM signal overview^[1]

PIN	Name	I/O	Description
ETMTO	trigger output	O	ETM trigger output, high when trigger condition becomes true
ETMTI	trigger input	I	ETM trigger input, can be used instead of setting a trigger condition
TRCCLK	trace clock	O	ETM trace clock, has the same frequency as the ARM clock
PIPESTAT[2:0]	pipeline status	O	reflects the status of the ARM pipeline
TRCPKT[16:0]	trace packet port	O	carries the compressed trace information
TRCSYNC	synchronization	O	indicates the type of packet which is broadcasted; important for decompressing software
demux mode			
PIPESTATA[2:0]	pipeline status	O	reflects the status of the ARM pipeline.
TRCPKTA[3:0]	trace packet port	O	carries the compressed trace information.
TRCSYNCA	synchronization	O	indicates the type of packet which is broadcasted; important for decompressing software.
PIPESTATB[2:0]	pipeline status	O	reflects the status of the ARM pipeline.
TRCPKTB[3:0]	trace packet port	O	carries the compressed trace information.
TRCSYNCB	synchronization	O	indicates the type of packet which is broadcasted; important for decompressing software.

[1] ETM signals are part of HDP, see [Section 6.5](#) for more details

Company Confidential

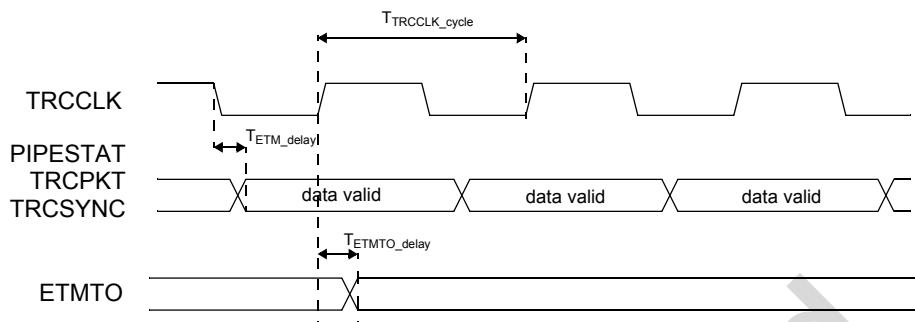


Fig 102.Timing diagrams for ETM full rate clocking

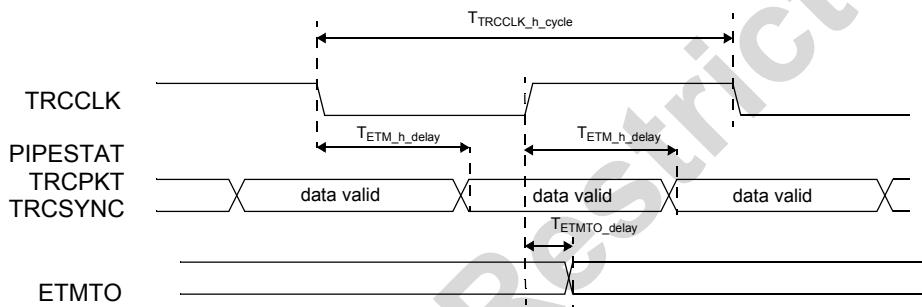


Fig 103.Timing diagrams for ETM half rate clocking

Table 246:AC characteristics of the ETM^[1]

Name	Parameter	Min	Typ	Max	Unit
Normal clocking					
T_{TRCCLK_cycle}	TRCCLK clock period	normal mode demux mode	4.8 [3] 9.6 [3]	T_{armclk}	- ns
T_{TRCCLK_duty}	TRCCLK clock duty cycle	-	50	-	%
T_{ETM_delay}	ETM trace signals valid after TRCCLK falling edge	-	-	2	ns
T_{ETMTO_delay}	ETMTO valid after TRCCLK rising edge	-	-	2	ns
Half-rate clocking					
$T_{TRCCLK_h_cycle}$	TRCCLK clock period	normal mode demux mode	9.6 [3] 19.2	$2 \cdot T_{armclk}$ $4 \cdot T_{armclk}$	- ns
$T_{TRCCLK_h_duty}$	TRCCLK clock duty cycle	-	50	-	%
$T_{ETM_h_delay}$	ETM trace signals valid after TRCCLK edge	-	-	$\frac{T_{TRCCLK}}{4} + 2$	ns
$T_{ETMTO_h_delay}$	ETMTO valid after TRCCLK edge	-	-	2	ns

[1] T_{armclk} correspond to ARM core clock cycle duration (e.g. 19 ns for 52 MHz ARM clock).

[2] The timings in this table are guaranteed by design, correlation, and structural testing. They are not specifically measured in production testing.

[3] T_{TRCCLK_cycle} and $T_{TRCCLK_h_cycle}$ below 15ns (65 MHz) are not guaranteed, in that case, only ETB or demux mode has to be used.

[4] ETMTI is treated asynchronously with respect to TRCCLK, hence not shown in timing diagrams

9.15.4 Functional description

The ETM allows real time tracing of the ARM926EJ-S core. Trace samples are either brought off-chip or stored into an on-chip trace buffer called ETB. The buffered data can then be accessed by the debugging tools using the JTAG interface as shown in [Figure 101](#). They can also be retrieved through the AHB bus.

Providing an on-chip buffer enables the trace data generated by the ETM (at the system clock rate) to be read by the debugger at a reduced clock rate.

When the ETB is used, trace samples are stored in a dedicated 4k x 24-bit trace buffer.

9.15.5 Application information

9.15.5.1 Using the ETM

All registers of the ETM are programmed via the JTAG interface. The interface is an extension of the ARM TAP controller and is assigned scan chain number 6.

[Figure 104](#) shows the ETM scan chain which consists of a 40-bit shift register comprising a 32-bit data field, a 7-bit address field and a read/write bit.



Fig 104.Arrangement of the ETM JTAG register

Typically, a user would not be concerned about the detailed register layout since a graphical user interface is provided by the debug/trace tool.

For detailed information about the register layout, the detailed description of all debug and trace features of the ETM and the physical interface to the trace port analyzer to be connected to the ETM, please refer to the ARM embedded macro cell specification [\[15.\]](#).

Please note that a typical ETM implementation consists only of a subset of all functions provided by the ARM ETM specification. Find below the description of the specific ETM9 setup.

[Table 247](#) shows all available registers of the ETM9.

Table 247: Register overview of the ETM9

Address	Name	I/O	reset
0b000 0000	ETM control register, controls the general operation of the ETM	R/W	0x0000 0401
0b000 0001	ETM configuration code register, reports the current configuration	R	0x50C9 9088
0b000 0010	ETM trigger event, defines the trigger event	W	0x0000 0000
0b000 0011	ETM memory map decode control	W	0x0000 0000
0b000 0100	ETM status register	R	0x0000 0000
0b000 0101	ETM system configuration	R	0x0000 01BA
0b000 0110	ETM trace start/stop resource control	W	0x0000 0000
0b000 0111	ETM trace enable control 2	W	0x0000 0000
0b000 1000	ETM trace enable event	W	0x0000 0000
0b000 1001	ETM trace enable control 1	W	0x0000 0000
0b000 1010	ETM FIFO full region	W	0x0000 0000
0b000 1011	ETM FIFO full level	W	0x0000 0000
0b000 1100	ETM view data event	W	0x0000 0000
0b000 1101	ETM view data control 1	W	0x0000 0000
0b000 1110	ETM view data control 2	W	0x0000 0000
0b000 1111	ETM view data control 3	W	0x0000 0000
0b001 0000 to 0b001 1111	ETM address comparators (16 registers)	W	0x0000 0000
0b010 0000 to 0b010 1111	ETM address access type (16 registers)	W	0x0000 0000
0b011 0000 to 0b011 0111	ETM data compare value (8 registers)	W	0x0000 0000
0b100 0000 to 0b100 0111	ETM data mask value (8 registers)	W	0x0000 0000
0b101 0000 to 0b101 0011	ETM initial counter value (4 registers)	W	0x0000 0000
0b101 0100 to 0b101 0111	ETM counter enable value (4 registers)	W	0x0000 0000
0b101 1000 to 0b101 1011	ETM counter reload value (4 registers)	W	0x0000 0000
0b101 1100 to 0b101 1111	ETM counter value (4 registers)	W	0x0000 0000
0b110 0000	ETM sequencer state transaction event (1 to 5), holds the next state transition for the sequencer states	W	0x0000 0000
0b110 1000	ETM external output	W	0x0000 0000

Compliance Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The ETM implementation is equipped with 16 specific memory map decoders which can be used for enabling or disabling the tracing of instructions resp. data. The memory map decoders work in a similar way as the address map decoders but are not programmable. They are defined according to [Table 248](#).

Table 248: Memory map decoders of the ETM

Map decoder number	Description	Type
0	SCRAM all banks	code/data
1	EBI1_CS0	code
2	EBI1_CS1	code/data
3	EBI1_CS2	code/data
4	EBI1_CS3	code/data
5	SDI port #0	code/data
6	EBI2_CS0	data
7	EBI2_CS1	data
8	EBI2_CS2	data
9	AHB peripherals	data
10	VPB1 peripherals	data
11	VPB2 peripherals	data
12	VPB3 peripherals	data
13	reserved	data
14	reserved	data
15	reserved	data

The memory map control register allowing multiple memory map to be selected is not used, only one set of memory map is available. Other debugging specific area can be defined by using one of the available address map decoder (inside ETM or EmbeddedICE-RT).

9.15.5.2 Using the ETB

Trace samples stored in the ETB trace RAM can be read back either through the JTAG interface or through the AHB bus interface. Similarly, ETB registers can be programmed either through the JTAG interface or through the AHB bus.

Both **acqcomp** and **etbfull** ETB output signals are connected to the interrupt controller so that the trace can be dumped by software if required.

For security reasons, the ETB AHB interface can be disabled by the **sysprot.security** bit located in SCON. When sysprot.security = 1 then reading the trace as well as JTAG is disabled.

[Table 249](#) shows all available registers of the ETB. As for the ETM, these registers are normally not used by the end user (even if accessible through AHB bus) but are programmed directly by the debug tools. Refer to ARM ETB specification for more details [16].

Table 249: Register overview of the ETB

Register number	Offset	Name	Description	I/O	reset
0	0x00	etb_id	Identification register	R	0x1B90002B
1	0x04	etb_ram_depth	RAM depth register	R	0x00001000
2	0x08	etb_ram_width	RAM width register	R	0x00000018
3	0x0C	etb_status	Status register	R	0x00000000
4	0x10	etb_ram_data	RAM data register	R	-
5	0x14	etb_ram_read_ptr	RAM read pointer register	R/W	0x00000000
6	0x18	etb_ram_write_ptr	RAM write pointer register	R/W	0x00000000
7	0x1C	etb_trig_cnt	Trigger counter register	R/W	0x00000000
8	0x20	etb_ctrl	Control register	R/W	0x00000000
9 to 127	0x24	-	reserved	-	-

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.16 EXTINT - External Interrupt

9.16.1 Features

External interrupts are provided to have special treatment (such as debounce or both edge detection) on external signals.

- 14 input channels
- Selectable polarity
- Programmable debounce from 0 to 56 ms by 8 ms steps
- Interrupt on one (rising or falling) or both edges of debounced input signal
- Bypass mode allow external interrupt to be level sensitive
- 3 interrupt outputs to INTC with separate enable for any external interrupt input sources
- Asynchronous stretching to detect short events (e.g. IrDA light pulse)
- Input multiplexer to select pad input or internal signal (e.g. USB vbus)
- Wake-up from sleep capabilities (internal logic clocked with clk32k)
- Share pad with some other peripheral in order to have wake-up capabilities

9.16.2 Block diagram

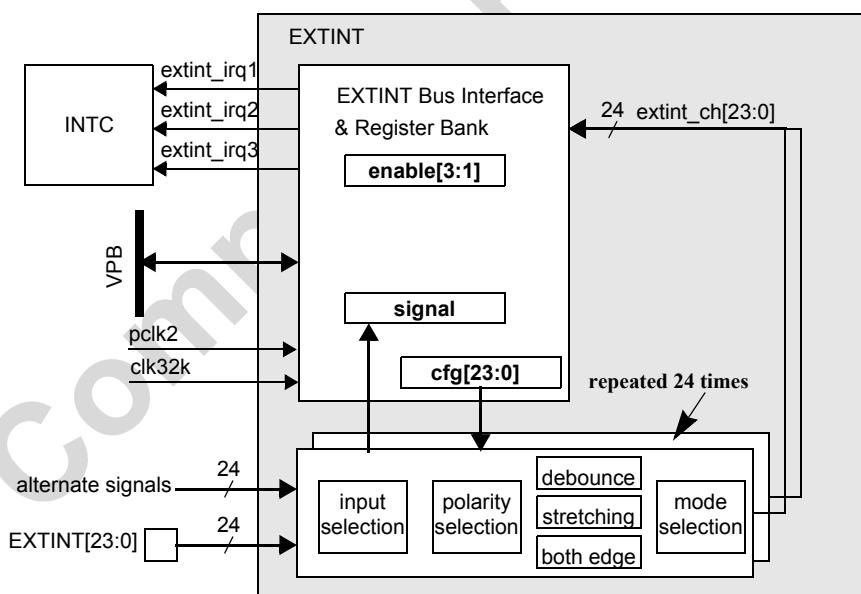


Fig 105.EXTINT block diagram

Note : At DVFD818x there is a reduced number of EXINT inputs available

9.16.3 Hardware interface

Table 250:EXTINT pin overview

PIN	Name	I/O	Description
EXTINT[23:0]	external interrupt	I	external interrupt signal

[1] At DVFD818x there is a reduced number of EXTINT inputs available

9.16.4 Software interface

Table 251:Register overview of EXTINT

Symbol	Name	I/O	Reset
extint_cfg1	EXTINT1 configuration register	R/W	0x00
extint_cfg2	EXTINT2 configuration register	R/W	0x00
extint_cfg5	EXTINT5 configuration register	R/W	0x00
extint_cfg7	EXTINT7 configuration register	R/W	0x00
extint_cfg10	EXTINT10 configuration register	R/W	0x00
extint_cfg12	EXTINT12 configuration register	R/W	0x00
extint_cfg13	EXTINT13 configuration register	R/W	0x00
extint_cfg15 ^[2]	EXTINT15 configuration register	R/W	0x00
extint_cfg16	EXTINT16 configuration register	R/W	0x00
extint_cfg17	EXTINT17 configuration register	R/W	0x00
extint_cfg18	EXTINT18 configuration register	R/W	0x00
extint_cfg19	EXTINT19 configuration register	R/W	0x00
extint_cfg21	EXTINT21 configuration register	R/W	0x00
extint_cfg22	EXTINT22 configuration register	R/W	0x00
extint_cfg23	EXTINT23 configuration register	R/W	0x00
extint_enable1	EXTINT interrupt1 enable	R/W	0x0000 0000
extint_enable2	EXTINT interrupt2 enable	R/W	0x0000 0000
extint_enable3	EXTINT interrupt3 enable	R/W	0x0000 0000
extint_status	EXTINT interrupt status	R/C	-[1]
extint_signal	EXTINT signal status	R	-

[1] Interrupt status reset value depend on pin state at reset

[2] Only the alternate input is supported

Table 252: Register extint_cfg0

Bit	Symbol	Access	Value	Description
7 to 6	mode[1:0]	R/W		select processing mode
			0b00*	Bypass mode (no stretching and no debounce) (default) The interrupt is level sensitive
			0b01	Stretching mode (no debounce) The interrupt is edge sensitive
			0b10	Debounce mode (no stretching) The interrupt is edge sensitive
			0b11	debounce and both edge detector mode The interrupt is edge sensitive
5 to 3	deb[2:0]	R/W		select de-bouncing duration (in number of 8 ms cycle duration)
			0b000*	no debounce (default value)
			0b001	1 cycle
			0b010	2 cycles
			0b011	3 cycles
			:	:
			0b111	7 cycles.
2	pol	R/W		polarity ; note: pol does not care if mode = 0b11 (both edge detector activated), but to avoid spurious detection after programming, it can be either set to 0 (if input is high, first edge to detect is falling) or 1 (if input is low, first edge to detect is rising).
			0*	normal polarity (falling edge/low level) (default value)
			1	reverse polarity (rising edge/high level)
1	sel	R/W		local input selection (see Table 256)
			0*	EXTINT0 pin selected (default value)
			1	alternate internal signal selected
0	-	R	0*	reserved

[1] Registers **extint_cfg1** to **extint_cfg23** have same meanings than **extint_cfg0**.

[2] Changing this register can generate level change in channel output and generate spurious interrupts.

Table 253: Register extint_enable1

Bit	Symbol	Access	Value	Description
31 to 24	-	R	0*	reserved
23 to 0	enable[23:0]	R/W		enable individuals external interrupt to extint_irq1 output signal ; each bit is assigned at an EXTINT input, respectively EXTINT[23:0] to enable[23:0]
			0*	interrupt masked
			1	interrupt enabled

[1] Registers **extint_enable2** and **extint_enable3** have same meanings than **extint_enable1**, but for extint_irq2 and extint_irq3 output signals.

[2] To avoid status contention, one external interrupt line can be enabled in only one enable register at a time.

Table 254: Register extint_status^[1]

Bit	Symbol	Access	Value	Description
31 to 24	-	R	0*	reserved
23 to 0	status[23:0]	R/C		extint interrupt status ; writing 0 clear corresponding bit; writing 1 has no effect on corresponding bit; status[23:0] is assigned respectively to EXTINT[23:0]
			0*	no interrupt
			1	interrupt pending Clearing this bit in bypass mode (extint_cfg.mode = 0b00) when extint_ch signal is still active will have no effect. This behavior is also known as level sensitivity

[1] This bits are set and cleared regardless to **extint_enable1**, **extint_enable2**, **extint_enable3** registers. It is under software responsibility to take into account only the bits assigned to current interrupt routine.

Table 255: Register extint_signal

Bit	Symbol	Access	Value	Description
31 to 24	-	R	0*	reserved
23 to 0	signal[23:0]	R	~*	reflect the state of the EXTINT[23:0] pins or alternate internal signal (see Table 256)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.16.5 Functional description

The EXTINT blocks serve to deal with interrupts from external sources. At DVFD8185 and DVFD8187 only 14 inputs are supported (see [Table 7](#)). The GPIO pin multiplexing is done externally to the EXTINT block. For details on how to select input multiplexing for GPIO pins, see [Section 6.4 “GPM - GPIO Pin Multiplexing” on page 37](#) and [Section 7.4 “SCON - System Configuration Block” on page 106](#). Alternatively to the connection to the pads, it is possible to choose up-to 24 internal sources according to [Table 256](#).

Table 256: Alternate internal routing for EXTINT signals

EXTINT line	Alternate input
EXTINT1	-
EXTINT2	-
EXTINT5	-
EXTINT7	-
EXTINT10	-
EXTINT12	-
EXTINT13	vbus
EXTINT14	-
EXTINT15	usb_need_clock
EXTINT16	-
EXTINT17	-
EXTINT18	-
EXTINT19	-
EXTINT21	-
EXTINT22	-
EXTINT23	-

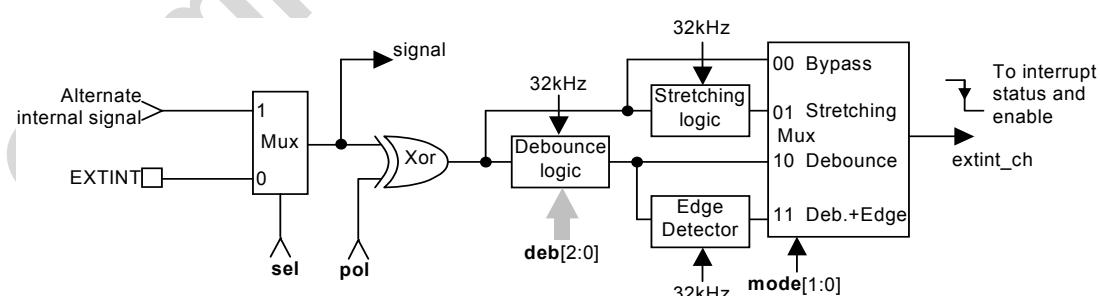


Fig 106.EXTINT channel

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.16.5.1 Debounce logic

Aim: to detect falling (**pol** = 0), rising (**pol** = 1) or both edge (**mode** = 0b11) on noisy signal.

The time-base used is **clk32k** divide by 256, which represent about 8 ms.

The logic checks that the input signal is stable for a programmable number of periods of the clock (from 1 to 7). When this condition is reached, the output changes. This is done on signal transition going to high level as well as low level. **deb** = 0 bypass the debounce logic (i.e. output = input).

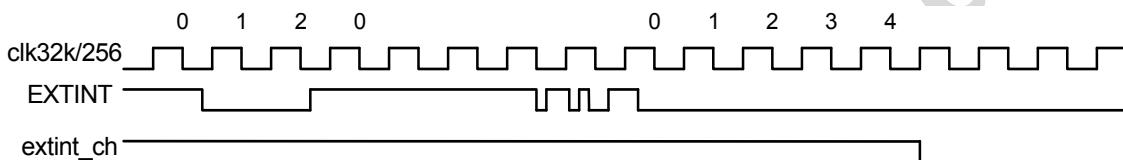


Fig 107.High to low timing diagram examples (N = 4)

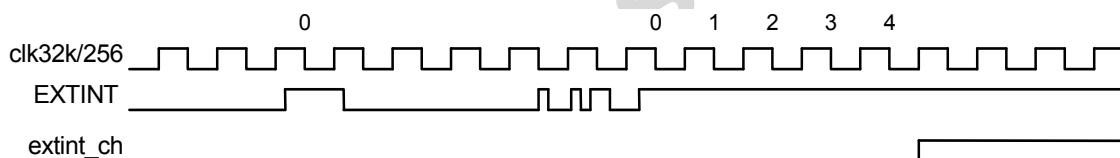


Fig 108.Low to high timing diagram examples (N = 4)

9.16.5.2 Stretching logic

Aim: to wake-up the system (from slow clock) with short pulses (e.g. 1.4 µs for SIR-IrDA or 8.7 µs for RS-232 115 kbits/s), asynchronous stretching logic ensure that short pulses are taken into account.

This logic react on active edge of input signal (regardless of the clock) and output a signal suitable to be taken into account by the status register.

Care must be taken because stretching logic is also sensitive to input glitches.

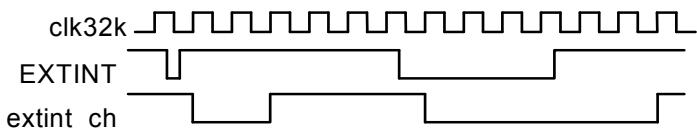


Fig 109.Stretching logic (pol = 0)

9.16.5.3 Both edge detector

Aim: to detect both edges on clean or on noisy signal. **pol** bit has no effect in this mode.

The initial state of the edge detector, considers that the signal at its input was HIGH before the edge detector is activated. So if the signal fed to the edge detector is LOW an interrupt is immediately generated if the edge detector is programmed on both edges or falling edge. If the signal fed to the edge detector is HIGH, no interrupt is immediately generated. **pol** may be used to avoid this first interrupt.

The purpose of this block is to have an interrupt on each edge of the input signal (e.g. open and close of the phone cover).

The interrupt is activated every edge of the input signal.

Input signal must have a sufficient duration to leave enough time for software latency between two events.

Short duration (less than $3 \times \text{clk32k}$) cause only one interrupt generated, so use with no-debounce is not recommended. Without debounce, signal input duration of $3 \times (\text{clk32k period}) = 91 \mu\text{s}$ ensure secure detection if software latency to acknowledge the interrupt in **extint_status** is less than $2 \times (\text{clk32k period}) = 61 \mu\text{s}$.

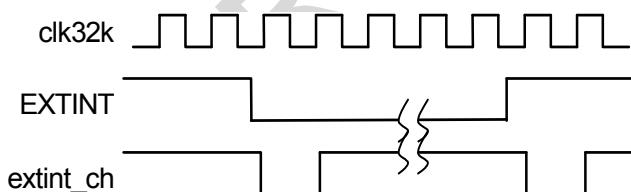


Fig 110.Both edge detector

9.16.5.4 Interrupt status:

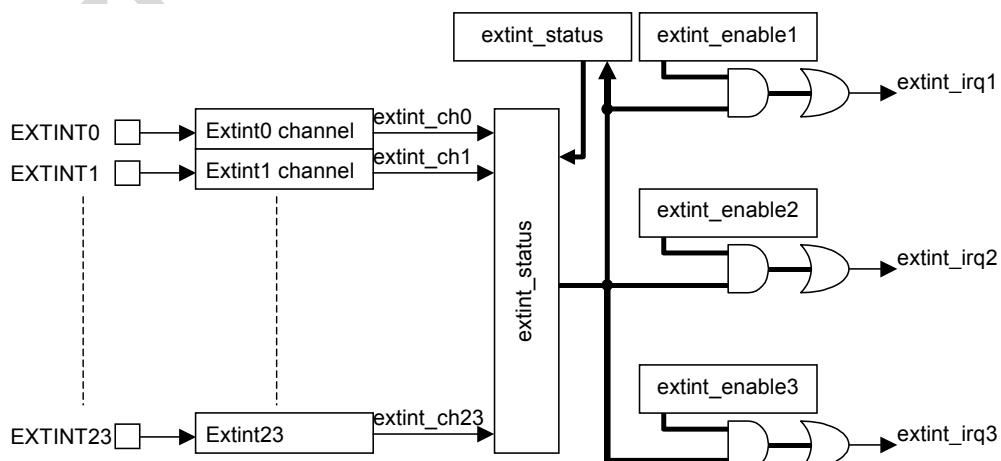


Fig 111.Status read/clear

extint_status return raw status for all external interrupt input. Software must mask unneeded bits (related to other extint_irq lines) before handling.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Company Restricted

9.16.6 Application information

9.16.6.1 Using the EXTINT for UART wake-up

Figure 112 shows an application for the EXTINT pins to support wake-up and sleep entry for the system. In this example an application running on an external host requests some transfer with the DVFD818x, and after that goes to sleep. Furthermore one wake-up request from the DVFD818x and one from the application are shown. Same functionality can be achieved with CTS.

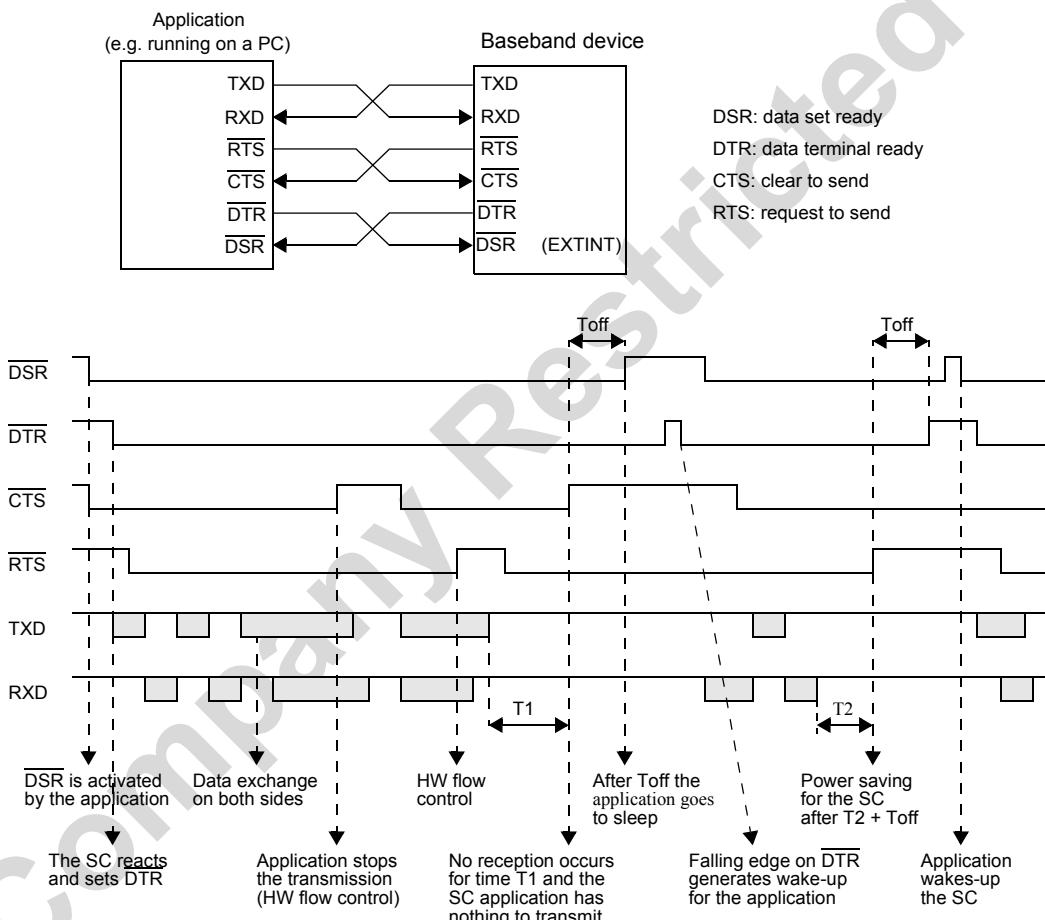


Fig 112.Example how to use EXTINT pins in order to provide wake-up from sleep functionality for the UART - drawn signals are from the DVFD818x Family point of view

9.17 FIR - Fast IrDA Controller

9.17.1 Features

- Supported standards:
 - IrDA Serial Infrared Physical Layer Specification (IrPHY) [19.]
 - IrDA Link Access Protocol (IrLAP) [20.]
- Supported infrared modes and baud rates:
 - serial infrared SIR: 9.6, 19.2, 38.4, 57.6 and 115.2 kbits/s
 - medium infrared MIR: 576 kbits/s and 1.152 Mbits/s
 - fast infrared FIR: 4 Mbits/s
- Half duplex infrared frame transmission and reception
- 16-bit CRC algorithm for SIR and MIR
- 32-bit CRC algorithm for FIR
- Bit and character stuffing
- Preamble, start and stop flag generation
- RZI and 4PPM modulation and demodulation
- 8 x 32-bit FIFO
- Transceiver interface:
 - Compliant to all IrDA transceivers
 - Configurable polarity of FIRTX and FIRRX signal

9.17.2 Block diagram

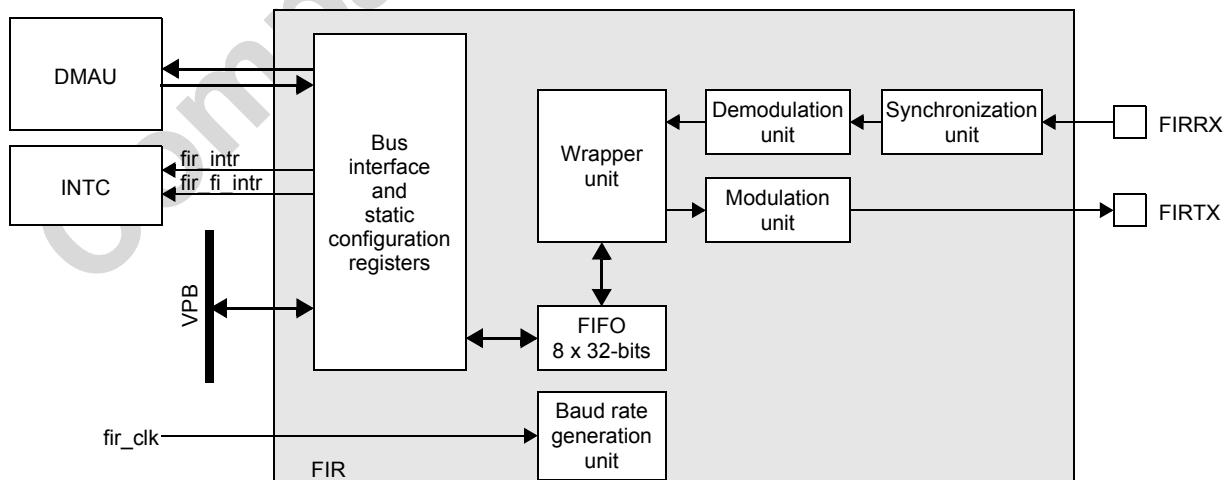


Fig 113.FIR block diagram

9.17.3 Hardware Interface

Table 257:FIR pin overview

PIN	Name	I/O	Description
FIRRX	Receive data	I	FIR reception from infrared transceiver
FIRTX	Transmit data	O	FIR transmission to infrared transceiver

9.17.4 Software Interface

Table 258:Register overview of FIR

Symbol	Name	I/O	Reset
fir_ip_id	ID register	R	0x0120 1101
fir_con	Control register	R/W	0x0000 0000
fir_conf	Configuration register	R/W	0x0002 0EA6
fir_para	Parameter register	R/W	0x0046 0000
fir_dv	Divider register	R/W	0x0000 0000
fir_stat	Status register	R	0x0000 0000
fir_tfs	Transmission frame size register	W	-
fir_rfs	Reception frame size register	R	0x0000 0000
fir_txb	Transmission buffer register	W	-
fir_rxb	Reception buffer register	R	0x0000 0000
fir_imsc	Interrupt mask control register	R/W	0x0000 0000
fir_ris	Raw interrupt status register	R	0x0000 0000
fir_mis	Masked interrupt status register	R	0x0000 0000
fir_icr	Interrupt clear register	W	-
fir_isr	Interrupt set register	W	-
fir_dma	DMA control register	R/W	0x0000 0000

Table 259:Register fir_ip_id

Bit	Symbol	Access	Value	Description
31 to 16	module_id	R	0x0120	module identification for FIR
15 to 12	major_rev	R	0x1	major revision
11 to 8	minor_rev	R	0x1	minor revision
7 to 0	aperture	R	0x01	aperture of register interface in 4kbyte blocks

Table 260:Register fir_con

Bit	Symbol	Access	Value	Description
31 to 1	-	R	0*	reserved
0	run	R/W		Enable FIR
			0*	FIR switches to the Inactive state
			1	FIR switches to the Listening state

Table 261: Register fir_conf

Bit	Symbol	Access	Value	Description
31 to 21	-	R	0*	reserved
20	poltx	R/W		Polarity of FIRTX pulses
			0*	active high
			1	active low
19	polrx	R/W		Polarity of FIRRX pulses
			0*	active low
			1	active high
18 to 16	bs[2:0]	R/W		Burst size
			0b000	1 word
			0b001	2 words Note: DMAU does not support a burst size of 2 words
			0b010*	4 words
			other	reserved
15 to 13	-	R	0*	reserved
12 to 0	ratv[12:0]	R/W	0x0EA6*	Reception Abort Timer Value A frame with a single pair of characters with a time gap greater than a defined value is considered as an invalid frame. The reception abort timer is programmed according to the following equation: $\text{ratv} = \frac{T_{\text{abort}} \times f_{\text{fir_clk}}}{128}$ Remark: 0x0EA6 corresponds to the IrDA T_{abort} default value of 10 ms with $f_{\text{fir_clk}} = 48$ MHz

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 262: Register fir_para

Bit	Symbol	Access	Value	Description
31 to 28	-	R	0*	reserved
27 to 16	mnrb[11:0]	R/W		Maximum number of received bytes Should be programmed accordingly to the negotiated Data Size (see [20.]). In the FIR mode the effective maximum number of received bytes is Data Size + 2 + 4 (Information + Address and Control + 4 CRC bytes)
	0x046*		70 bytes	
	0x086		134 bytes	
	0x106		262 bytes	
	0x206		518 bytes	
	0x406		1030 bytes	
	0x806		2054 bytes	
	other		reserved	
15 to 8	-	R	0*	reserved
7 to 2	abff[5:0]	R/W		Number of additional beginning flags Should be programmed accordingly to the negotiated Additional XBOF (SIR mode) or STA (MIR mode). See [20.] for details. This field is ignored in FIR mode. Allowed values are 0 to 48. Values above 48 are reserved
	0*		No additional Beginning Flags	
1 to 0	mode[1:0]	R/W		Infrared mode Should be programmed accordingly to the negotiated Baud Rate (see [20.])
	0b00*		SIR	
	0b01		MIR	
	0b10		FIR	
	0b11		reserved	

Table 263: Register fir_dv

Bit	Symbol	Access	Value	Description
31 to 27	-	R	0*	reserved
26 to 16	dec[10:0]	R/W	0*	fractional divider decrement value Should be programmed accordingly to Table 276 , Table 277 and Table 278 dec = 0 with inc > 0 is reserved dec = 0 and inc = 0 will generates en_pulse = fir_clk
15 to 8	inc[7:0]	R/W	0*	fractional divider increment value Should be programmed accordingly to Table 276 , Table 277 and Table 278 inc = 0 with dec > 0 is reserved inc = 0 and dec = 0 will generates en_pulse = fir_clk
7 to 0	n[7:0]	R/W	0*	integer divider denominator Should be programmed accordingly to Table 276 , Table 277 and Table 278

Table 264: Register fir_stat

Bit	Symbol	Access	Value	Description
31 to 2	-	R	0*	reserved
1	txs	R	0*	Transmission state
			1	FIR not in transmission state
			1	FIR in transmission state
0	rxs	R	0*	Reception state
			1	FIR not in reception state
			1	FIR in reception state

Table 265: Register fir_tfs

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	tfs[11:0]	W	0*	Transmission frame size Transmit frame size is $(tfs + 1)$ bytes Values above 2049 are reserved (2050 bytes transmit frame size) Remark: tfs represents the number of transmitted bytes excluding CRC (which is computed and appended by the FIR controller) equal to: Data Size + 2 (Information + Address and Control bytes). When using interrupts, amount of data written in fir_txb must be coherent with value in this register.

Table 266: Register fir_rfs

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 0	rfs[11:0]	R	0*	Reception frame size Reception frame size is rfs bytes Remark: rfs represents the number of received bytes equal to: <ul style="list-style-type: none"> • in SIR and MIR: Data Size + 2 + 2 (Information + Address and Control + 2 CRC bytes) • In FIR: Data Size + 2 + 4 (Information + Address and Control + 4 CRC bytes)

Table 267: Register fir_txb

Bit	Symbol	Access	Value	Description
31 to 0	txb[31:0]	W	0*	Transmit data First transmitted byte is on txb[7:0]

Table 268: Register fir_rxb

Bit	Symbol	Access	Value	Description
31 to 0	rx[31:0]	R	0*	Reception data First received byte is on rx[7:0]

Table 269: Register fir_imsc

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7	fd	R/W	0*	Frame Detected interrupt mask
			1	interrupt disabled
			1	interrupt enabled
6	fi	R/W	0*	Frame Invalid interrupt mask
			1	interrupt disabled
			1	interrupt enabled
5	sd	R/W	0*	Signal Detected interrupt mask
			1	interrupt disabled
			1	interrupt enabled
4	ft	R/W	0*	Frame Transmitted interrupt mask
			1	interrupt disabled
			1	interrupt enabled
3	breq	R/W	0*	BREQ interrupt mask
			1	interrupt disabled
			1	interrupt enabled
2	lbreq	R/W	0*	LBREQ interrupt mask
			1	interrupt disabled
			1	interrupt enabled
1	sreq	R/W	0*	SREQ interrupt mask
			1	interrupt disabled
			1	interrupt enabled
0	lsreq	R/W	0*	LSREQ interrupt mask
			1	interrupt disabled
			1	interrupt enabled

Company Confidential

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 270: Register fir_ris

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7	fd	R	0*	Frame Detected raw interrupt status
			0	No interrupt
			1	interrupt pending
6	fi	R	0*	Frame Invalid raw interrupt status
			0	No interrupt
			1	interrupt pending
5	sd	R	0*	Signal Detected raw interrupt status
			0	No interrupt
			1	interrupt pending
4	ft	R	0*	Frame Transmitted raw interrupt status
			0	No interrupt
			1	interrupt pending
3	breq	R	0*	BREQ raw interrupt status
			0	No interrupt
			1	interrupt pending
2	lbreq	R	0*	LBREQ raw interrupt status
			0	No interrupt
			1	interrupt pending
1	sreq	R	0*	SREQ raw interrupt status
			0	No interrupt
			1	interrupt pending
0	lsreq	R	0*	LSREQ raw interrupt status
			0	No interrupt
			1	interrupt pending

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 271: Register fir_mis

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7	fd	R	0*	Frame Detected masked interrupt status
			0	No interrupt
			1	interrupt pending
6	fi	R	0*	Frame Invalid masked interrupt status
			0	No interrupt
			1	interrupt pending
5	sd	R	0*	Signal Detected masked interrupt status
			0	No interrupt
			1	interrupt pending
4	ft	R	0*	Frame Transmitted masked interrupt status
			0	No interrupt
			1	interrupt pending
3	breq	R	0*	BREQ masked interrupt status
			0	No interrupt
			1	interrupt pending
2	lbreq	R	0*	LBREQ masked interrupt status
			0	No interrupt
			1	interrupt pending
1	sreq	R	0*	SREQ masked interrupt status
			0	No interrupt
			1	interrupt pending
0	lsreq	R	0*	LSREQ masked interrupt status
			0	No interrupt
			1	interrupt pending

Company Confidential

Table 272: Register fir_icr

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7	fd	W	0*	Frame Detected interrupt clear
			0*	No effect
			1	interrupt clear
6	fi	W	0*	Frame Invalid interrupt clear
			0*	No effect
			1	interrupt clear
5	sd	W	0*	Signal Detected interrupt clear
			0*	No effect
			1	interrupt clear
4	ft	W	0*	Frame Transmitted interrupt clear
			0*	No effect
			1	interrupt clear
3	breq	W	0*	BREQ interrupt clear
			0*	No effect
			1	interrupt clear
2	lbreq	W	0*	LBREQ interrupt clear
			0*	No effect
			1	interrupt clear
1	sreq	W	0*	SREQ interrupt clear
			0*	No effect
			1	interrupt clear
0	lsreq	W	0*	LSREQ interrupt clear
			0*	No effect
			1	interrupt clear

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 273: Register fir_isr

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7	fd	W	0*	Frame Detected interrupt set
			0*	No effect
			1	interrupt set
6	fi	W	0*	Frame Invalid interrupt set
			0*	No effect
			1	interrupt set
5	sd	W	0*	Signal Detected interrupt set
			0*	No effect
			1	interrupt set
4	ft	W	0*	Frame Transmitted interrupt set
			0*	No effect
			1	interrupt set
3	breq	W	0*	BREQ interrupt set
			0*	No effect
			1	interrupt set
2	lbreq	W	0*	LBREQ interrupt set
			0*	No effect
			1	interrupt set
1	sreq	W	0*	SREQ interrupt set
			0*	No effect
			1	interrupt set
0	lsreq	W	0*	LSREQ interrupt set
			0*	No effect
			1	interrupt set

Table 274: Register fir_dma

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	breq	R/W	0*	BREQ DMA enable
			0*	DMA disabled
			1	DMA enabled
2	lbreq	R/W	0*	LBREQ DMA enable
			0*	DMA disabled
			1	DMA enabled
1	sreq	R/W	0*	SREQ DMA enable
			0*	DMA disabled
			1	DMA enabled
0	lsreq	R/W	0*	LSREQ DMA enable
			0*	DMA disabled
			1	DMA enabled

9.17.5 Functional Description

9.17.5.1 Location of the IrDA controller within protocol stack layers

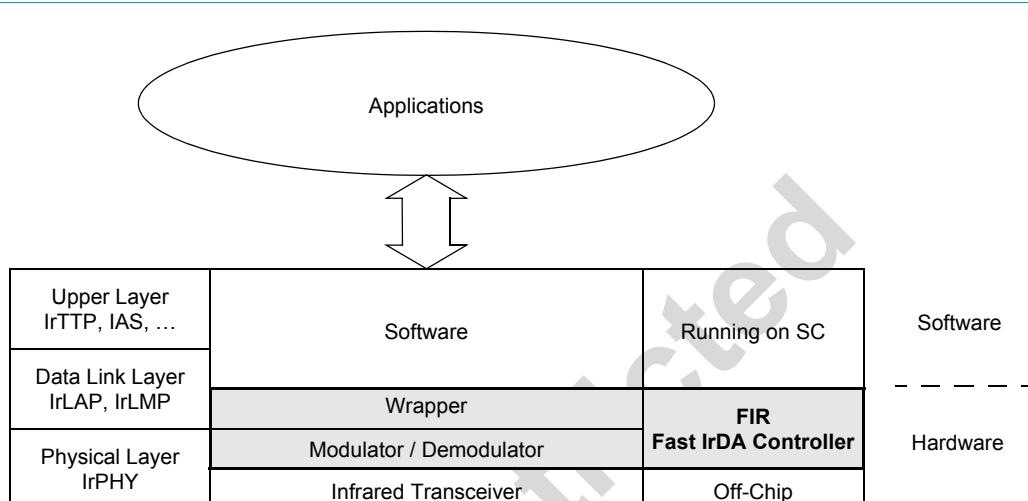


Fig 114.Location of the IrDA controller within protocol stack layers

9.17.5.2 Operating states

The Fast IrDA controller has four different states:

- Inactive state IAS
- Listening state LIS
- Reception state RXS
- Transmission state TXS

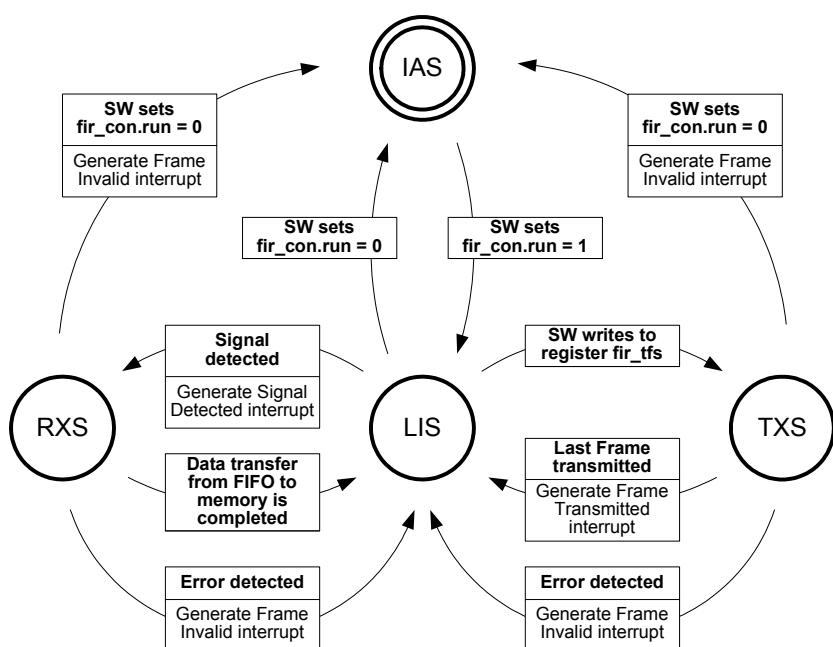


Fig 115.FIR Operating states

The Fast IrDA controller is in the Inactive state, if the bit **fir_con.run** is set to 0. As soon as the bit **fir_con.run** is set to 1, the Fast IrDA controller switches to the Listening state.

In the Listening state the Synchronization Unit is scanning for a rising edge of the active high FIRRX signal.

If a FIRRX signal is detected in the Listening state, the Fast IrDA controller switches to the Reception state, indicates this by generating a Signal Detected interrupt and by setting the bit **fir_stat.rxs**.

In the Reception state the controller can receive a frame. The controller changes back to the Listening state, when the received frame has been completely fetched from the FIFO. If the HW detects an invalid frame, then it aborts the reception, changes back to the Listening state and indicates this to the SW by a Frame Invalid interrupt.

If the SW sets the bit **fir_con.run** to 0 during the Reception state, then the Fast IrDA controller switches to the Inactive state and generates a Frame Invalid interrupt. If the SW indicates, that it wants to copy data to the FIFO in the Listening state, the Fast IrDA controller automatically switches to the Transmission state, indicated by bit **fir_stat.txs**.

In the Transmission state the controller can transmit a frame. The controller changes back to the Listening state, when the frame has been completely transmitted. If the HW detects a transmission error, then it aborts the transmission, changes back to the Listening state and indicates this to the SW by a Frame Invalid interrupt.

If the SW sets the bit **fir_con.run** to 0 during the Transmission state, then the Fast IrDA controller switches to the Inactive state and generates a Frame Invalid interrupt.

It is not possible to receive data from the IrDA transceiver in the Transmission state and it is also not possible to copy data into the FIFO in the Reception state.

9.17.5.3 FIFO Unit

Data can be transmitted via the Fast IrDA controller by writing to the Transmission Buffer register **fir_txb**. A received data byte can be read out from the Reception Buffer register **fir_rxb**. These buffer registers represents the head respectively the tail of the FIFO.

A 8-stage 32-bit shift register is used as buffer so that data can be transferred at full speed using DMA burst transfers. Since IrDA supports only half-duplex communication one buffer is used for both transmission and reception.

The Fast IrDA controller generates the following request signals to control the data transfer to and from the memory.

- Burst Request Signal **breq**:
Requests a transfer of a programmed burst number of words.
- Last Burst Request Signal **Ibreq**:
Requests a last burst transfer.
- Single Request Signal **sreq**:
Requests a transfer of a single word.
- Last Single Request Signal **Isreq**:
Requests a last single transfer.

These signals can either be used as interrupt requests or as DMA requests. They are reset on the occurrence of the Request Clear Signal, which is either set by SW via the register **fir_icr** or automatically generated by the DMAU.

The burst size is programmable via the **fir_conf.bs** field.

Note: To allow some latency in the DMAU transfer, the recommended burst size is 4 as the FIFO size is 8.

Note: The same request signals are used for transmission as well as for reception. The SW is responsible to provide a proper interrupt service or for a correct programming of the DMAU.

FIFO Unit transmission state

The SW indicates that it wants to transmit data by writing the frame size to the **fir_tfs** register. Then the Fast IrDA controller changes to the Transmission state and the FIFO Unit asserts burst requests using **breq** until the amount of data still to be transferred is less than or equal to the burst size **fir_conf.bs**. At this point, if the remaining data is equal to the burst size, then a last burst request is issued using **Ibreq**. Otherwise, single requests are issued on **sreq** until the last data item is ready, when **Isreq** is used.

Note: Since the size of the frame to be transmitted is not necessarily a multiple of 4, the last 32-bit word must be filled up with dummy bytes. The HW transmits only the valid bytes of the last word by means of the **fir_tfs** register.

The data is buffered in a 8-stage 32-bit shift register before it is processed by the Fast IrDA controller. The Fifo indicates to the Wrapper Unit, if there are still bytes of the frame in the buffer ready to be moved to the Wrapper Unit. If a FIFO underflow occurs before all bytes of a frame has been shifted into the FIFO, then a Frame Invalid interrupt is generated, the transmission is aborted and all pending bytes in the peripheral are discarded.

The next frame to be transmitted can be copied into the buffer, when all bytes of the current frame are completely transferred to the Wrapper Unit. The SW can write the size of the next frame into **fir_tfs** immediately after the last word of the current frame has been written to **fir_txb**.

FIFO Unit reception state

The received bytes of one frame are shifted from the Wrapper Unit into the FIFO, where the data is buffered.

The received bytes are counted by a 12-Bit Counter. The counter value can be read by SW via the **fir_rfs** register. If the number of the received bytes is greater than the maximum number of received bytes, which is defined by **fir_para.mnrb**, then the currently received frame becomes invalid, what is indicated by a Frame Invalid interrupt. A Frame Invalid interrupt is also generated, if a buffer overflow occurs.

The internal signal 'Frame Complete' indicates, that the whole data of the current received frame has been moved into the buffer. The bytes of this frame have to be moved out of the buffer by the SW before the next received frame will be shifted into the buffer. If this is not done, the next received frame is completely discarded.

Note: Between two frames there are always start and stop flags. Due to this flags there is enough time for the SW to flush the FIFO, even when the frames are sent back to back.

The buffered data is moved out of the buffer with the full bus speed using DMA burst transfers. The FIFO Unit asserts burst requests using **breq** until the amount of data still to be transferred is less than or equal to the burst size **fir_conf.bs**. At this point, if the remaining data is equal to the burst size, a burst request using **Ibreq** is issued. Otherwise, single requests are issued on **sreq** until the last word is ready, when **Isreq** is used.

Note: Since the size of the received frame is not necessarily a multiple of 4, the upper bytes of the last word can be invalid. The SW has to check for invalid bytes of the last word by means of the **fir_rfs** register.

Note: Along with the IrLAP bytes the CRC bytes are also transferred to the memory. The CRC bytes can be double-checked by the SW for the purpose of testing.

The occurrence of a Frame Invalid interrupt due to any reason during the reception indicates that the received data has become invalid and then the buffer content is cleared without sending further requests.

The Fast IrDA controller changes back to the Listening state, when the received frame has been completely read out of the buffer.

9.17.5.4 Interrupt summary

Table 275:Fast IrDA controller interrupt summary

INTC line	Interrupt	Description
fir_fi_intr	fi	In Reception state: The currently received frame is invalid. This can be due to: - a CRC error, - the reception of an invalid flag, - a frame abort, - the reception of an invalid symbol, - the reception of a too long frame, - a FIFO overflow, - an abort by SW (fir_con.run=0). In Transmission state: The current frame has not been sent completely. This can be due to: - a FIFO underflow, - an abort by SW (fir_con.run=0).
[1] fir_intr	fd	A frame has been detected during the Reception state
	Frame Detected	
	sd	A signal has been detected during the Listening state.
	Signal Detected	
	ft	A frame has been completely transmitted.
	Frame Transmitted	
	breq	A transfer of a programmed burst number of words from/to the memory is requested.
	Burst Request	This request can either be used as interrupt request or as DMA requests.
	lbreq	A last burst transfer from/to the memory is requested.
	Last Burst Request	This request can either be used as interrupt request or as DMA requests.
	sreq	A transfer of a word from/to the memory is requested.
	Single Request	This request can either be used as interrupt request or as DMA requests.
	lsreq	A last single transfer from/to the memory is requested.
	Last Single Request	This request can either be used as interrupt request or as DMA requests.

[1] **fir_fi_intr** must have a higher priority than **fir_intr**

9.17.5.5 Wrapper Unit

The Wrapper Unit serves to mark the beginning and end of the IrLAP frame and to check for the reliable transmission of data.

The wrapping schemes are specified in [20.]

Thus the Wrapper Unit appends beginning, ending flags and a Cyclic Redundancy Check (CRC) flag to the IrLAP frame, if the controller is transmitting data. Thereby it generates the TX frame, which drives the Modulation Unit.

If the controller is receiving data, the Wrapper Unit regains the IrLAP frame from the RX frame and examines the CRC.

The Wrapper Unit is active in the Transmission and Reception state. In the Listening state the Wrapper Unit is not used.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Wrapper unit transmission state

In Transmission state the Wrapper Unit builds the TX frame out of IrLAP frame, which should be transmitted.

The data is sent LSB first. The encoding method depends on the used infrared mode, which is determined by the **fir_para.mode** field.

- **Serial Infrared SIR**

- CRC Generation

The raw transmitted data is scanned from the least significant to the most significant bit of each byte. A 16-bit CRC-CCITT is computed for the IrLAP frame and is appended at the end of data.

The CRC is initialized to all ones. This allows detection of any missed or inserted zero bits at the beginning of a block. (Missed or inserted ones are still detected.) The one's complement of the CRC is transmitted rather than the CRC itself. This allows detection of slippage-type errors.

The CRC is sent MSB first. The CRC-CCITT polynomial is defined as follows:

$$\text{CRC}(x) = x^{16} + x^{12} + x^5 + 1$$

Note: the IrDA specification does not mention clearly which bit of the CRC has to be sent first. The FIR sends MSB first (industry standard).

- Character Stuffing

Information bytes that would otherwise be interpreted as flags or other control characters are transformed into non-flag/control characters prior to transmission. Therefore a Control Escape (CE) byte is defined as 0x7D. For each byte that encounters with the same value as a flag or a CE byte (0xC0, 0xC1 or 0x7D) the encoder performs the following:

- Inserts a Control Escape (CE) byte preceding the byte

- Complements bit 5 of the byte

- Beginning of Frame BOF and End of Frame EOF

The BOF and EOF flags enclose the frame. The BOF serves as a reference for the position of the payload data. The ending flag delimits the end of the CRC and marks the end of frame.

The first BOF before the data is obligatory and is defined as 0xC0. Additional consecutive XBOFs before the BOF are optional. An additional XBOF is defined as 0xFF. The number of additional XBOFs sent with each frame can be set by **fir_para.abf** field. The maximum number of additional flags is 48. The EOF flag is defined as 0xC1.

- Start and Stop Bits

Each TX byte is enclosed by a start and a stop bit. This is necessary to be compatible with IrDA controllers that use a UART. The start bit is defined as 0 and the stop bit is defined as 1.

- **Medium Infrared MIR**
 - CRC Generation
 - The CRC generation is the same as at SIR.
 - Bit Stuffing
 - A 'Zero' is inserted after five consecutive ones are transmitted in order to distinguish the flag from data. Zero insertion is done on every field except the flags.
 - Beginning Flag (STA) and Ending Flag (STO)
 - Finally two STAs and one STO flag are appended to the TX Data. 0x7E is used for the STA as well as for STO.
 - Additional consecutive beginning flags are optional. The number of additional beginning flags sent with each packet can be set by the **fir_para.abf** field. The maximum number of additional flags is 2.
 - **Fast Infrared FIR**
 - In the Fast Infrared mode the Wrapper Unit generates a 32-bit CRC. The preamble, start and stop flags in this mode are described in the Modulation Unit
 - CRC Generation
 - For a FIR transmission, a 32-bit CRC code is applied with the following CRC32 polynomial:
$$\text{CRC}(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$
 - The CRC32 calculated result for each packet is treated as four data bytes, and each byte is encoded in the same fashion as payload data. Payload data bytes are input to this calculation in LSB first format. The 32-bit CRC register is preset to all "1's" prior to calculation of the CRC on the transmit data stream.
- 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Wrapper unit reception state

In the Reception state the Wrapper Unit retrieves the IrLAP frame and the CRC bytes out of the RX frame. This decoding process is aborted, if an error is detected in the Demodulation Unit (e.g. a reception of an invalid symbol) or in the FIFO Unit (e.g. a FIFO overflow). After decoding the IrLAP and CRC bytes are shifted to the FIFO Unit.

- **Serial Infrared SIR**

The start and stop bits are removed from the RX frame in order to retrieve the RX bytes. If the stop bit is not 1, then a Frame Invalid interrupt is generated and the Wrapper Unit changes to the Listening state.

Note: There can be transmission pauses between stop and start bits.

By means of the BOF (0xC0) and EOF (0xC1) flags the Wrapper Unit recognizes the beginning and the end of a RX frame. The occurrence of the first BOF is indicated to the SW by a Frame Detected interrupt. Multiple flags are treated as a single flag.

Note: 0xFF is treated as any byte in the Reception state.

As soon as RX bytes are detected, the Wrapper Unit starts with decoding in order to retrieve the bytes of the IrLAP frame. Thereby incoming CE flags are discarded and the IrLAP and CRC bytes are moved to the FIFO.

The IrLAP frame and the received CRC field are also shifted into the CRC Unit, which calculates a new CRC on all the protected bits and the received CRC field. After receiving an EOF it compares the new CRC to the known constant 0xF0B8. In the case of a mismatch the received IrLAP frame is no more valid and a Frame Invalid interrupt is generated. After a successful CRC the internal signal 'Frame Complete' of the FIFO is set.

- **Medium Infrared MIR**

The decoding process is quite similar to SIR. The CRC is also calculated in the same fashion. The occurrence of the first STA is indicated to the SW by a Frame Detected interrupt.

A prematurely terminated frame is called an aborted frame. The frame can be aborted by blocking the IR transmission path in the middle of the frame, a random introduction of infrared noise, or intentional termination by the transmitter. Regardless what caused the aborted frame, the Wrapper Unit treats a frame as an aborted frame when seven or more consecutive ones (no optical signal) are received. The abort terminates the frame immediately without processing the CRC or an ending flag. Furthermore a Frame Invalid is generated.

- **Fast Infrared FIR**

The state machine in the FIR mode is quite simple due to the lack of beginning and ending flags. The calculated 32-bit CRC field is compared to the constant 0xDEBB20E3.

9.17.5.6 Modulation Unit

The Modulation Unit modulates the bits of the TX frame from the Wrapper Unit in order to create the FIRTX signal, which drives the off-chip IrDA infrared transceiver.

The used modulation schemes are Four Pulse Position Modulation (4PPM) for FIR and Return Zero Inverted (RZI) for SIR and MIR.

- The modulation schemes are specified in [19.]

The Modulation Unit is responsible for the modulation of the TX frames of the Wrapper Unit in order to generate the FIRTX signal. The bit **fir_conf.poltx** determines the polarity of the FIRTX signal. I.e. with bit **fir_conf.poltx** = 0 a low level of the FIRTX signal indicates a “LED off” state and a high level of the FIRTX signal indicates a “LED on” state and vice versa with **fir_conf.poltx** = 1.

The FIRTX signal is generated by means of the signals **en_symb** and **en_pulse**. These signals are generated by the Baud Rate Generation Unit and they determine the baud rate of the FIRTX signal.

If a frame is completely transmitted, then a Frame Transmitted interrupt is generated and the Fast IrDA controller changes back to the Listening state (TXS=0).

- **Serial Infrared SIR**

In this mode a RZI (Return-to-Zero-Inverted) modulation is used. This means that for a “0“ bit in the TX frame a high pulse is generated and for a “1“ bit in the TX frame no pulse is generated. For bit rates up to 115.2 kbit/s a pulse duration of 1.736 µs is used.

For example, the byte to be transmitted is 0x6C, then 0b01101100 is shifted into the Modulation Unit with LSB first.

- **Medium Infrared MIR**

This mode also uses RZI modulation, but the pulse length is 1/4 of a bit.

• Fast Infrared FIR

In this mode a Four Pulse Position Modulation (4PPM) is used to modulate the TX frame. Additionally a preamble PA, a start flag STA and a stop flag STO are added.

4PPM encoding is achieved by defining a data symbol duration (Dt) and subsequently subdividing Dt into four equal time slices called chips. A chip marks the time space when an infrared pulse can be sent. Within a data symbol only one pulse is allowed, thus there exist four different data symbols, which represent the four possible combinations of two bits.

The 4PPM data encoding scheme defines only the legal encoded payload data symbols. All other chip combinations are by definition illegal symbols for encoded payload data. Some of these illegal symbols are used in the definition of the preamble, start flag and stop flag fields because they are unambiguously not data.

With a bit rate of 4 Mbits/s the resulting data symbol duration Dt is 500 ns and the resulting chip duration Ct is 125 ns.

– Preamble Field Definition

The preamble field consists of exactly sixteen repeated transmissions of the following stream of symbols.

0b1000 0000 1010 1000

– Start Flag Definition

The start flag consists of exactly one transmission of the following stream of symbols.

0b0000 1100 0000 1100 0110 0000 0110 0000

Note: There is only one start flag in the FIR mode. Thus **fir_para.abf** is obsolete in this mode and will be ignored by the controller.

– Stop Flag Definition

The stop flag consists of exactly one transmission of the following stream of symbols.

0b0000 1100 0000 1100 0000 0110 0000 0110

9.17.5.7 Synchronization Unit

The Synchronization Unit detects and synchronizes the FIRRX signal from the off-chip IrDA infrared transceiver.

The FIRRX signal is sampled by the rising edge of the clock **fir_clk** for synchronization.

If the Synchronization Unit detects an activity of the FIRRX signal in the Listening state, then the Fast IrDA controller switches to the Reception state and sets the bit **fir_stat.rxs**. Furthermore a Signal Detected interrupt is generated.

If the Synchronization Unit detects no activity of the FIRRX signal for more than 10 ms in the Reception state, then the Fast IrDA controller switches to the Listening state and resets the bit **fir_stat.rxs**. Furthermore a Frame Invalid interrupt is generated. This behavior handles the case of a frame abort. If different value than the default 10 ms is needed, the reception abort timer has to be programmed via the **fir_conf.ratv** field.

9.17.5.8 Demodulation Unit

The Demodulation Unit demodulates the synchronized FIRRX signal in order to retrieve the RX frame.

The Demodulation Unit is active in the Reception state only. The Demodulation Unit is responsible for the demodulation of the synchronized active high FIRRX signal from the Synchronization Unit in order to obtain the RX frame.

The bit **fir_conf.polrx** must be set according to the polarity of the FIRRX signal. I.e. for a low level of the FIRRX signal indicating a “LED on” state and a high level of the FIRRX signal indicating a “LED off” state the bit **fir_conf.polrx** must be set to 0. Otherwise **fir_conf.polrx** must be set to 1.

- **Serial SIR and Medium Infrared MIR**

The pulses of the synchronized active high signal are extended in order to eliminate jitter influences. Then the signal is sampled with the signal **en_symb**.

The phase of **en_symb** is determined by the first rising edge of FIRRX signal and is readjusted with every following rising edge.

- **Fast Infrared FIR**

The preamble field PA of the synchronized FIRRX signal is used by the receiver to establish phase lock. For that the 8 MHz signal **en_pulse** is recovered from the FIRRX signal by means of a DPLL (digital phase locked loop). Then the incoming FIRRX signal is sampled with the recovered signal **en_pulse**.

During PA, the receiver begins to search for the start flag STA to establish symbol synchronization. If a STA is received correctly, then the receiver begins to demodulate the 4PPM modulated RX frame and generates a Frame Detected interrupt. The receiver continues to receive and demodulate until the stop flag STO is recognized. STO indicates the end of the frame.

9.17.5.9 Baud Rate Generation Unit

The Baud Rate Generation Unit creates the signals **en_symb** and **en_pulse**. The signal **en_symb** determines the symbol rate during transmission. It is also used to sample the FIRRX signal during reception in SIR or MIR mode. The signal **en_pulse** determines the pulse width during transmission.

The Baud Rate Generation Unit creates the signals **en_symb** and **en_pulse**.

During transmission the signal **en_symb** determines the symbol rate, which is defined as transmitted symbols per second. In the Reception state of SIR and MIR, FIRRX signal is sampled with **en_symb**. The signal **en_pulse** is used to create the pulses of the FIRTX signal during transmission.

The signal **en_symb** and the signal **en_pulse** are derived from the clock **fir_clk** (48 MHz) by using cascaded clock dividers.

Thus the resulting frequencies of **en_pulse** and **en_symb** are:

$$f_{en_pulse} = f_{fir_clk} \times \frac{inc}{inc + dec} \text{ and } f_{en_symb} = \frac{f_{en_pulse}}{n + 1}$$

The values **inc**, **dec** and **n** are set in **fir_dv** register.

The fractional divider causes jitter with a maximum of $\frac{1}{2 \times f_{fir_clk}}$ = 10.417 ns at SIR and MIR.

- The value is within the tolerances of [19.]

SIR mode

Since with each SIR symbol one bit is transmitted, the bit rate and the symbol rate are equal at SIR. Thus the Baud Rate Generation Unit has to create the following symbol rates f_{en_symb} : 9.6 kHz, 19.2 kHz, 38.4 kHz, 57.6 kHz and 115.2 kHz.

In the case of a SIR transmission a pulse duration of 1.736 us is used, thus the Baud Rate Generation Unit has to create a pulse rate of $f_{en_pulse} = 576$ kHz.

Table 276:clock setting for SIR mode

Bit rate [kbit/s]	f_{en_pulse} [kHz]	f_{en_symb} [kHz]	inc	dec	n
9.6	576	9.6	3	247	59
19.2	576	19.2	3	247	29
38.4	576	38.4	3	247	14
57.6	576	57.6	3	247	9
115.2	576	115.2	3	247	4

MIR mode

Since with each MIR symbol one bit is transmitted, the bit rate and the symbol rate are equal at MIR. Thus the Baud Rate Generation Unit has to create the following symbol rates f_{en_symb} : 576 kHz and 1.152 MHz.

In the case of a MIR transmission the pulse duration is a quarter of the symbol duration, thus the Baud Rate Generation Unit has to create a pulse rate of $f_{en_pulse} = 4 * f_{en_symb}$.

Table 277:clock setting for MIR mode

Bit rate [kbit/s]	f_{en_pulse} [kHz]	f_{en_symb} [kHz]	inc	dec	n
576	2304	576	6	119	3
1152	4608	1152	12	113	3

FIR mode

Since with each FIR symbol two bits are transmitted, the symbol rate is one half of the bit rate at FIR. Thus the Baud Rate Generation Unit has to create the symbol rate $f_{en_symb} = 2$ MHz.

In the case of a FIR transmission the pulse duration is a quarter of the symbol duration, thus the Baud Rate Generation Unit has to create a pulse rate of $f_{en_pulse} = 4 * f_{en_symb}$.

Table 278:clock setting for FIR mode

Bit rate [kbit/s]	f_{en_pulse} [kHz]	f_{en_symb} [kHz]	inc	dec	n
4000	8000	2000	1	5	3

9.17.6 Application information**9.17.6.1 Recommended Programming of DMAU Channel Control register**

- Transfer size
Since the DMA controller is not the flow controller the transfer size value is not used
- Fast IrDA controller burst size
The Fast IrDA controller burst size should be 4
Note: A Fast IrDA controller burst size of 1 is also supported
- Memory burst size
The memory burst size should be equal to the Fast IrDA controller burst size
- Fast IrDA controller transfer width
The Fast IrDA controller transfer width is 32-bit
- Memory transfer width
The memory transfer width is 32-bit
- Fast IrDA controller address increment
The Fast IrDA controller address is not incremented after a transfer
- Memory address increment
The memory address is incremented after each transfer

9.17.6.2 DMAU Procedure of Data Transfers between Peripheral and Memory	1
1. DMAU channel and DMAU requests in register fir_dma are disabled. If a Frame Detected interrupt occurs, then go to step 2. If the SW wants to transmit data, then go to step 10.	2
2. SW programs and enables the DMAU channel for reception.	3
3. SW enables the DMAU requests in register fir_dma .	4
4. Fast IrDA controller generates burst and/or single requests, which are served by the DMAU controller.	5
5. If the Fast IrDA controller detects an error, then it stops the requests and generates a Frame Invalid interrupt. The SW can discard the bytes of the frame, which are already in the memory. Go to step 9.	6
6. Fast IrDA controller generates last burst or last single request, which is served by the DMAU controller.	7
7. When the DMA transfer is completed, the terminal count interrupt is generated.	8
8. SW reads out the value of the fir_rfs register. Thereby the SW can verify, which bytes of the last word are invalid.	9
9. Data transfer is finished. SW disables the DMA channel and the DMA requests in register fir_dma . Go to step 1.	10
10. SW programs and enables the DMAU channel for transmission.	11
11. SW enables the DMAU requests in register fir_dma .	12
12. SW writes the size of the frame to be transmitted in the register fir_tfs .	13
13. Fast IrDA controller generates burst and/or single requests, which are served by the DMAU controller.	14
14. If the Fast IrDA controller detects an error, then it stops the requests and generates a Frame Invalid interrupt. Go to step 17.	15
15. Fast IrDA controller generates last burst or last single request, which is served by the DMAU controller.	16
16. When the DMA transfer is completed, the terminal count interrupt is generated.	17
17. Data transfer is finished. SW disables the DMAU channel and the DMAU requests in register fir_dma . Go to step 1.	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32
	33
	34
	35
	36
	37
	38
	39
	40
	41
	42
	43
	44
	45
	46
	47
	48
	49
	50
	51
	52
	53
	54

9.18 GPIO - General Purpose I/O

9.18.1 Features

- 96-bit general purpose parallel I/O port - every pin can be configured as input or output independently
Note : For DVFD8185 and DVFD8187 only 58 pins are supported (see [Table 7](#)).
• Three GPIO banks are available
 - GPIOA[31:0], pin multiplexing configured as GPIO or peripheral after reset
 - GPIOB[31:0], pin multiplexing configured as peripheral signal after reset
 - GPIOC[31:0], pin multiplexing configured as peripheral signal after reset
• Input always connected to pin allowing read-back of pin level even when GPIO is not selected by **SCON.sysmux**

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.18.2 Block diagram

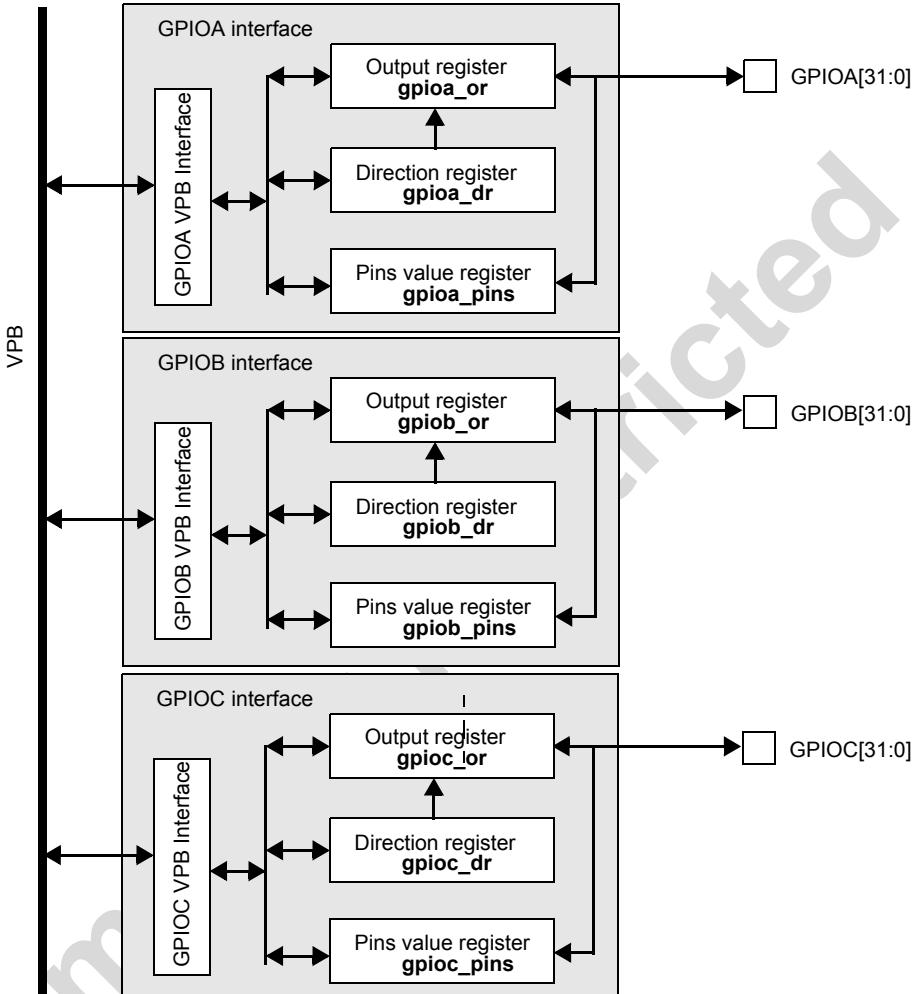


Fig 116.GPIO block diagram (GPIOA...C)

Note : For available pins look at pinlist

9.18.3 Hardware interface

Table 279:GPIO pin overview

PIN	Name	I/O	Description
GPIOA[31:0]	GPIO bank A	I/O	general purpose I/O pin (bank A)
GPIOB[31:0]	GPIO bank B	I/O	general purpose I/O pin (bank B)
GPIOC[31:0]	GPIO bank C	I/O	general purpose I/O pin (bank C)

Committed

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.18.4 Software interface

Table 280: Register overview of GPIO

Symbol	Name	I/O	Reset
gpioa_pins	GPIOA pins value register	R [1]	-
gpioa_or	GPIOA output register	R/W	0x0000 0000
gpioa_dr	GPIOA direction register	R/W	0x0000 0000
gpiob_pins	GPIOB pins value register	R [1]	-
gpiob_or	GPIOB output register	R/W	0x0400 0000
gpiob_dr	GPIOB direction register	R/W	0x0400 0000
gpioc_pins	GPIOC pins value register	R [1]	-
gpioc_or	GPIOC output register	R/W	0x0000 0000
gpioc_dr	GPIOC direction register	R/W	0x0000 0000

[1] Value of this register always reflect the state of the pin regardless **SCON.sysmux** settings

Table 281: Register gpioa_pins

Bit	Symbol	Access	Value	Description
31 to 0	pins[31:0]	R	~*	reflect the value of the pin, regardless of gpioa_dr and SCON.sysmux value

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 282: Register gpioa_or

Bit	Symbol	Access	Value	Description
31 to 0	or[31:0]	R/W	0x00000000*	output register ; value of gpioa_or is present on output if corresponding bit in gpioa_dr is equal to 1

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 283: Register gpioa_dr

Bit	Symbol	Access	Value	Description
31 to 0	dr[31:0]	R/W	0x00000000*	direction register
			0	corresponding GPIOA signal is configured as an input
			1	corresponding GPIOA signal is configured as an output

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 284: Register gpiob_pins

Bit	Symbol	Access	Value	Description
31 to 0	pins[31:0]	R	~*	reflect the pin value, regardless gpiob_dr and SCON.sysmux value

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 285: Register gpiob_or

Bit	Symbol	Access	Value	Description
31 to 0	or[31:0]	R/W	0x04000000*	output register ; value of gpiob_or is present on output if corresponding bit in gpiob_dr is equal to 1

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 286: Register gpiob_dr

Bit	Symbol	Access	Value	Description
31 to 0	dr[31:0]	R/W	0x04000000*	direction register
			0	corresponding GPIOB signal is configured as an input
			1	corresponding GPIOB signal is configured as an output

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 287: Register gpioc_pins

Bit	Symbol	Access	Value	Description
31 to 0	pins[31:0]	R	~*	reflect the pin value, regardless gpioc_dr and SCON.sysmux value

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 288: Register gpioc_or

Bit	Symbol	Access	Value	Description
31 to 0	or[31:0]	R/W	0x00000000*	output register ; value of gpioc_or is present on output if corresponding bit in gpioc_dr is equal to 1

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

Table 289: Register gpioc_dr

Bit	Symbol	Access	Value	Description
31 to 0	dr[31:0]	R/W	0x00000000*	direction register
			0	corresponding GPIOC signal is configured as an input
			1	corresponding GPIOC signal is configured as an output

[1] For available pins on DVFD8185 and DVFD8187 look at [Table 7](#).

9.18.5 Functional description

Each GPIO pin can be programmed individually as input or output (**gpio_dr** register). Pin value is read by **gpio_pins** register in either Input or Output mode. Pin value can be set by writing to **gpio_or**, the value is present on the pin if corresponding **gpio_dr** bit is equal to 1. Obviously, top level pin multiplexing must be taken into account to use GPIO signal. If an IIC pin is configured as GPIO output a high state on the pin can only be reached if an external pull up to VDDH is present.

9.19 IIC - I²C-bus Interface

9.19.1 Features

The I²C-bus interface allows the communication with other devices via the I²C-bus. For more information on the I²C-bus, the reader can consult the I²C-bus specification[3.]. The I²C-bus interface supports the following operation modes:

- Master Transmitter mode (MTx)
- Master Receiver mode (MRx)
- Slave Transmitter mode (STx)
- Slave Receiver mode (SRx)

In addition, the I²C-bus interface can operate according to the following baud rate standards:

- Standard mode (s-mode): the maximum baud rate is 100 kbit/s
- Fast mode (f-mode): the maximum baud rate is 400 kbit/s

In order to buffer the transmitted and received data, the I²C-bus interface contains:

- 64 bytes TX FIFO
- 8 bytes slave TX FIFO
- 64 bytes RX FIFO

9.19.2 Block diagram

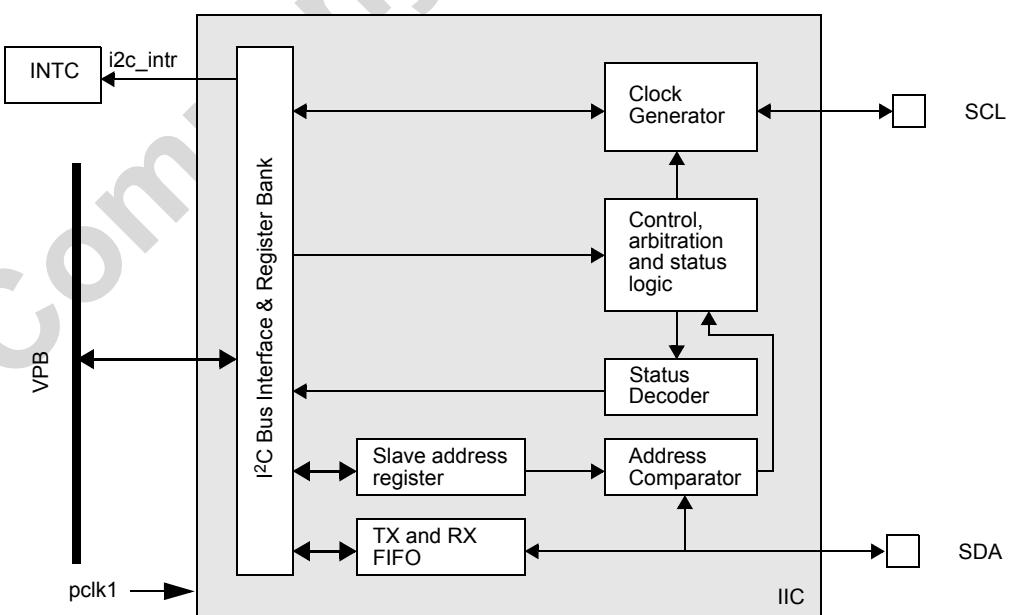


Fig 117.IIC interface block diagram

9.19.3 Hardware interface

Table 290:I²C-bus interface pin overview

PIN	Name	I/O	Description
SCL	I ² C-bus clock pin	I/O	serial clock line, open drain output
SDA	I ² C-bus data pin	I/O	serial data line, open drain output

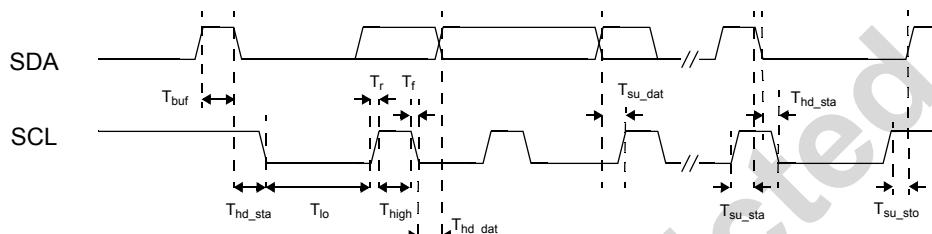


Fig 118.Timing diagram of I²C-bus interface in Master mode

Table 291:AC characteristics of the I²C-bus interface

Name	Parameter	Min	Typ	Max	UNIT
f_{scl}	SCL clock frequency	0	-	400	kHz
T_{buf}	bus free time between STOP and START	1.3	-	-	μs
T_{hd_sta}	hold time (repeated) START condition	0.6	-	-	μs
T_{low}	low period of SCL clock	1.3	-	-	μs
T_{high}	high period of SCL clock	0.6	-	-	μs
T_{su_sta}	set-up time for a repeated START condition	0.6	-	-	μs
T_{hd_dat}	data hold time [3]	300	-	-	ns
T_{su_dat}	data set-up time [3]	100	-	-	ns
T_r	rise time for SDA and SCL [1]	$20 + 0.1 \cdot C_b$	-	250	ns
T_f	fall time for SDA and SCL [1]	$20 + 0.1 \cdot C_b$	-	250	ns
T_{su_sto}	setup time for STOP condition [3]	0.6	-	-	μs
C_b	capacitive load on SDA and SCL	-	-	400	pF
R_{PU}	recommended pull-up resistor on SDA and SCL	$C_b = \text{up to } 10\text{pF}$	2.5	-	kΩ
		$C_b = 10 \text{ to } 100\text{pF}$	1.1	-	3.5
		$C_b = 100\text{pF to } 200\text{pF}$	1.1	-	1.7
I_{PU}	recommended pull-up current source on SDA and SCL	$C_b = 200\text{pF to } 400\text{pF}$	-	3	mA

[1] With recommended pull-up resistor or current source

[2] The timings in this table are guaranteed by design, correlation, and structural testing. They are not specifically measured in production testing.

[3] Timing are achieved with recommend settings in [Table 302](#) and [Table 303](#)

9.19.4 Software interface

Table 292: Register overview of I²C-bus interface

Symbol	Name	I/O	Reset
iicrx	IIC RX FIFO register	R	~
iictx	IIC TX FIFO register	W	0x0000
iicsts	IIC status register	R/C	0x6AX0
iicctl	IIC control register	R/W	0x00
iicclkhi	IIC clock high register	R/W	0x0000
iicclklo	IIC clock low register	R/W	0x0000
iicaddr	IIC slave address register	R/W	0x00
iicholddat	IIC data hold control register	W	0x00
iictxs	IIC slave TX FIFO register	W	0x00

Table 293: Register iicrx

Bit	Symbol	Access	Value	Description
7 to 0	data[7:0]	R	~*	receive FIFO register: this is the top of the RX FIFO; it is the oldest byte in the RX FIFO; the FIFO is flushed by an hard or soft reset (iicctl.sr); read empty FIFO (also immediately after reset) return undefined value

Table 294: Register iictx

Bit	Symbol	Access	Value	Description
15 to 10	-	W	0*	reserved
9	stop	W	0*	stop
			1	do not issue a STOP condition
			1	issue a STOP condition after transmitting this byte
8	start	W	0*	start
			1	do not issue a START condition
			1	issue a START condition before transmitting this byte
7 to 0	data[7:0]	W	0*	transmit FIFO register: this is the top of the TX FIFO; it is the newest byte in the TX FIFO; the TX FIFO is flushed by a hard reset, soft reset (iicctl.sr), or if an arbitration failure occurs (iicsts.afi); writes to a full FIFO causes the data is ignored; when operating as a master, the TX FIFO must be written for both write and read operations to transfer each byte; the field data[7:0] is ignored for master-receive operations; the master-receiver must write a dummy byte to the TX FIFO, for each byte it expects to receive in the RX FIFO; when the STOP is set or a START bit is set to cause a RESTART condition on a byte written to the TX FIFO as a master-receiver, then the byte read from the slave is not acknowledged; that is, the last byte of a master-receive operation is not acknowledged

Table 295: Register iicsts

Bit	Symbol	Access	Value	Description
15	-	R	0*	reserved
14	mast	R		master mode: The value of this bit is not relevant until first master or slave activity occurs on the I ² C-bus
			0	last active state was slave of the I ² C-bus
			1*	last active state was master of the I ² C-bus.
13	tfes	R		slave transmit FIFO empty
			0	slave TX FIFO is not empty
			1*	slave TX FIFO is empty; it is cleared when the slave TX FIFO contains valid data
12	tffs	R		slave transmit FIFO full
			0*	slave TX FIFO is not full
			1	slave TX FIFO is full
11	tfe	R		transmit FIFO empty
			0	TX FIFO contains valid data
			1*	TX FIFO is empty; it is cleared when the TX FIFO contains valid data
10	tff	R		transmit FIFO full
			0*	TX FIFO is not full
			1	TX FIFO is full
9	rfe	R		receive FIFO empty
			0	RX FIFO is not empty
			1*	RX FIFO is empty; it is cleared when the RX FIFO contains valid data
8	rff	R		receive FIFO full
			0*	RX FIFO is not full
			1	RX FIFO is full; if a byte arrives when the receive FIFO is full, the SCL is held low until the processor reads the RX FIFO and makes room for it
7	sda	R	~*	the current value of the SDA signal
6	scl	R	~*	the current value of the SCL signal
5	active	R	0*	indicates whether the bus is busy; this bit is set when a START condition has been detected; it is cleared when a STOP condition has been detected
4	drsi	R		slave data request; once a transmission is started, the transmitter must have data to transmit as long as it is not followed by a STOP condition or it will hold SCL low until more data is available; this bit is set when the slave transmitter is data-starved; if the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the processor writes another byte to transmit; this bit is cleared when a byte is written to the slave TX FIFO
			0*	slave transmitter does not need data
			1	slave transmitter needs data

Company Confidential

Table 295: Register iicsts...continued

Bit	Symbol	Access	Value	Description
3	drmi	R		master data request: once a transmission is started, the transmitter must have data to transmit as long as it is not followed by a STOP condition or it will hold SCL low until more data is available; this bit is set when the master transmitter is data-starved; if the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the processor writes another byte to transmit; this bit is cleared when a byte is written to the master TX FIFO
			0*	master transmitter does not need data
			1	master transmitter needs data
2	nai	R		no acknowledge: after every byte of data is sent, the transmitter expects an acknowledge from the receiver; this bit is set if the acknowledge is not received; it is cleared when a byte is written to the master TX FIFO
			0*	last transmission received an acknowledge
			1	last transmission did not receive an acknowledge
1	afi	R/W		arbitration failure: when transmitting, if the SDA input is low when the SDA output is high, then this I ² C-bus device has lost the arbitration to another device on the bus; this bit is cleared by writing a 1 at this location
			0*	no arbitration failure on the last transmission
			1	arbitration failure occurred on last transmission
0	tdi	R/W		transaction done: it is unaffected by slave transactions
			0*	transaction has not completed
			1	transaction completed successfully; this bit is cleared by writing a 1 at this location

Company Confidential

Table 296: Register iicctl

Bit	Symbol	Access	Value	Description
15 to 12	-	R	0*	reserved
11	stopie	R/W	0*	STOP condition received interrupt enable
			1	disable the STOP interrupt
			1	enable the STOP interrupt to indicate that a STOP has been received (applies to slave mode)
10	tffsie	R/W	0*	slave transmit FIFO not full interrupt enable
			1	disable the tffs interrupt
			1	enable the tffs interrupt to indicate that more data can be written to the slave transmit FIFO
9	-	R	0*	reserved
8	sr	WaC	0*	soft reset
			0*	no reset
			1	reset the I ² C-bus
7	tffie	R/W	0*	transmit FIFO not full interrupt enable
			1	disable the tff interrupt
			1	enable the tff interrupt to indicate that more data can be written to the transmit FIFO
6	rfdaie	R/W	0*	receive data available interrupt enable
			1	disable the rfe interrupt
			1	enable the rfe interrupt to indicate that data is available in the receive FIFO (i.e not empty)
5	daie	R/W	0*	receive FIFO full interrupt enable
			1	disable the rff interrupt
			1	enable the rff interrupt to indicate that the FIFO cannot accept any more data
4	drsie	R/W	0*	slave transmitter data request interrupt enable
			1	disable the drs interrupt
			1	enable the drs interrupt signalling that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low
3	drmie	R/W	0*	master transmit data request interrupt enable
			1	disable the drm interrupt
			1	enable the drm interrupt, which signals that the master transmitter has run out of data, has not issued a stop, and is holding the SCL line low

Company Confidential

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 296: Register iicctl...continued

Bit	Symbol	Access	Value	Description
2	naie	R/W		no acknowledge interrupt enable
			0*	disable the nai interrupt
			1	enable the nai interrupt signalling that transmitted byte was not acknowledged
1	afie	R/W		arbitration failure interrupt enable
			0*	disable the afi interrupt
			1	enable the afi interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device
0	tdie	R/W		transaction done interrupt enable
			0*	disable the tdi interrupt
			1	enable the tdi interrupt signalling that this device issued a stop condition

Table 297: Register iicclkhi

Bit	Symbol	Access	Value	Description
15 to 10	-	R	0*	reserved
9 to 0	clkdivhi[9:0]	R/W	0*	high period clock divider value defining the clock high duration in Master mode; the clock pclk1 is divided by this value to generate the high period of clock SCL. This time includes the time T_r on SCL in order to compensate for slow edges. $T_{high_o} = \frac{clkdivhi + 7}{f_{pclk1}}$

Remark: In slave mode, clock has to be set to maximum expected I2C-bus clock frequency (i.e. 100 or 400 KHz). See [Table 302](#) and [Table 303](#)

Table 298: Register iicclklo

Bit	Symbol	Access	Value	Description
15 to 10	-	R	0*	reserved
9 to 0	clkdivlo[9:0]	R/W	0*	low period clock divider value defining the clock low duration in Master mode; the clock pclk1 is divided by this value to generate the low period of clock SCL. This time includes the time T_f on SCL in order to compensate for slow edges. $T_{low_o} = \frac{clkdivlo + 1}{f_{pclk1}}$ $T_{cycle} = T_{high_o} + T_{low_o} = T_{high} + T_{low} + T_r + T_f$

Remark: In slave mode, clock has to be set to maximum expected I2C-bus clock frequency (i.e. 100 or 400 KHz). See [Table 302](#) and [Table 303](#)

Table 299: Register iicaddr

Bit	Symbol	Access	Value	Description
7	-	R	0*	reserved
6 to 0	saddr[6:0]	R/W	0*	slave address bits

Table 300: Register iicholddat

Bit	Symbol	Access	Value	Description
7	-	W	0*	reserved
6 to 0	holddat[6:0]	W	0*	data hold time control. See Table 302 and Table 303

Table 301: Register iictxs

Bit	Symbol	Access	Value	Description
7 to 0	data[7:0]	W	0*	slave transmit buffer register: it is the top of the slave TX FIFO; the top byte is the newest byte in the slave TX FIFO; the FIFO is flushed by a hard or soft reset (iicctl.sr)

9.19.5 Functional description

The I²C-bus consists of a data line SDA and a clock line SCL. The data transfer, clock generation, address recognition and bus arbitration of this interface are all controlled directly by hardware. The I²C-bus serial interface has complete autonomy in byte handling and operates in four modes: master transmit, master receive, slave transmit and slave receive.

These functions are controlled by the **iicctl** register. The interface has the capability to communicate in a multi-master environment. The I²C-bus data transfers are preceded or followed by some conditions (START, STOP, RESTART and ACKNOWLEDGE). The corresponding flags are accessible in the status register **iicsts**. The **iicrx** and **iictx** (and **iictxs** for Slave mode) registers are the top of the receive and transmit FIFOs and **iicadr** is the register that stores its own slave address. Slave address recognition is performed by hardware. See [3.] for a documentation of the I²C-bus protocol and for I²C-bus state encoding.

The interrupt signal **i2c_intr** indicates to the interrupt controller INTC if an interrupt is pending. The reason for the interrupt is encoded in the status register **iicsts**. There are several possible interrupt types: transfer completed, arbitration failure, missing acknowledge, need more data, TX FIFO has room for more data, or data has been received. Status register bits are ANDed with the enable bits of the control register **iicctl** and then ORed together to create the interrupt signal **i2c_intr**.

After a soft reset, FIFOs are flushed, **iicsts** register is cleared and all state-machines are set to idle. **drmi** is not set as a start has not been issued yet. If all interrupts are enabled, only the **tfe** (Master mode) and **tfs** (Slave mode) will occur because FIFO is empty.

The main clock of the I²C-bus interface (**pclk1**) is divided under the control of **iicclkhi.clkdivhi** and **iicclklo.clkdivlo** respectively. The I²C-bus clock frequencies are subdivided in two frequency ranges: 0 to 100 kbit/s for s-mode and 100 to 400 kbit/s for f-mode. Table 302 and Table 303 show the possible I²C-bus Master mode frequencies.

Table 302: I²C-bus clock frequencies settings (XTAL = 13.000MHz)

I ² C-bus clock frequency [2]		Register setting	pclk1					
			110.5 MHz [3]	104 MHz	52 MHz	39 MHz	26 MHz	13 MHz
100 kbit/s	clkdivhi	519	488	244	183	118	55	
	clkdivlo	578	544	268	199	134	67	
	holddat	72	68	33	24	16	8	
	Resulting fscl (kHz)	100.00	100.00	100.00	100.00	100.00	100.00	
400 kbit/s	clkdivhi	90	84	38	27	15	4	
	clkdivlo	179	168	84	63	42	21	
	holddat	45	42	21	16	11	6	
	Resulting fscl (kHz)	398.92	400.00	400.00	397.96	400.00	393.94	

[1] Values greater than 400 kbit/s (f-mode) respectively 100 kbit/s (s-mode) are outside the I²C-bus specification (excluding Hs mode which is not supported).

[2] In slave mode, setting must be done for maximum expected I²C-bus clock frequency

[3] These operating points are not guaranteed

Table 303:I²C-bus clock frequencies settings (XTAL = 10.368/13.824MHz)

I ² C-bus clock frequency [2]		pclk1					
100 kbit/s	clkdivhi	555	488	258	192	126	60
	clkdivlo	613	541	287	215	143	71
	holddat	76	67	35	26	17	8
	Resulting fscl (kHz)	99.92	99.98	99.99	99.93	99.81	99.45
400 kbit/s	clkdivhi	96	84	42	29	18	5
	clkdivlo	190	168	89	67	44	22
	holddat	47	42	23	17	12	6
	Resulting fscl (kHz)	399.67	398.77	397.81	398.77	394.97	394.97

[1] Values greater than 400 kbit/s (f-mode) respectively 100 kbit/s (s-mode) are outside the I²C-bus specification (excluding Hs mode which is not supported).

[2] In slave mode, setting must be done for maximum expected I²C-bus clock frequency

[3] These operating points are not guaranteed

9.19.6 Application information

- **Slave operation** (default after reset): during slave operations, the I²C-bus acknowledges then receive/send data if its address follow the start bit (sent by the master). For sending data in slave mode, you write all data in **iictxs**. Reception is made through **iicrx** register. In slave mode, clock setting must be done for maximum expected I²C-bus clock frequency.
- **Master operation**: for working as a master, software must write a data in **iictx** with **start** bit active. The I²C-bus works as a master until you write data in **iictx** with **stop** bit active. Reception is made through **iicrx** register. **Note:** master receive operation. As stated in [Table 294](#), master reception needs a dummy write in **iictx** register. Data are stored in the receive FIFO only on reception of next character (received data stays in shift register until written in the FIFO). This introduce a one character delay in the receive FIFO status update (**iicsts** bits **rfe** and **rff**). Nevertheless, the receive FIFO status is updated when issuing a STOP condition.
- **Bus arbitration**: the I²C-bus allows any bus master to start a transfer when the bus is idle. In a multi-master system, it is possible to have more than one master start a transfer at the same time. To arbitrate between masters, all I²C-bus masters must monitor the state of the bus while they are driving it. If a master is trying to put a 1 on the bus while another is driving a 0, the bus is low (wire-AND) and the master trying to put a 1 on the bus must abort its operation. The master driving a 0 continues its operation unaware that another master aborted a transfer. This situation is signalled to the aborted master by the **iicsts.afi** bit.

9.20 IIS - I²S-bus Interface

9.20.1 Features

- Support for MP3, AAC and WMA decoding
- I²S-bus output of 16-bit stereo samples, MSB justified data
- Wide range of output sampling frequencies: 8, 11.025, 12, 16, 22.05, 24, 32, 44.1 and 48 kHz
- Frequency error smaller than 0.04%
- 8 stages deep 32-bit FIFO
- Dedicated DMAU channel to reload the FIFO

9.20.2 Block diagram

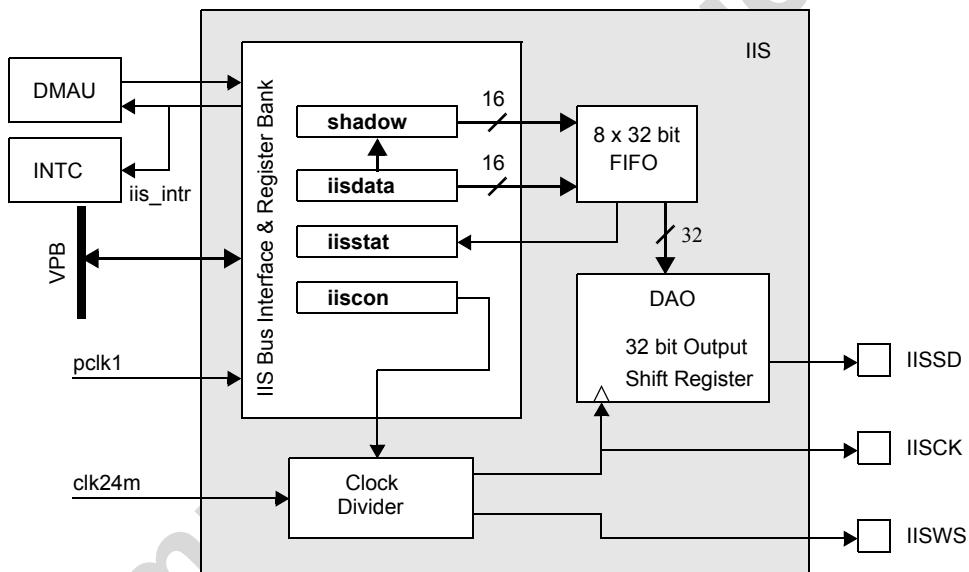


Fig 119.IIS block diagram

Complementary material
selected

9.20.3 Software interface

Table 304: Register overview of I²S-bus

Symbol	Name	I/O	Reset
iiscon	I ² S-bus configuration register	R/W	0x0400
iisstat	I ² S-bus status register	R/C	0x0040
iisdata	I ² S-bus data register for left and right channels	R/W	0x0000

Table 305: Register iiscon

Bit	Symbol	Access	Value	Description
15 to 11	-	R	0*	reserved
10 to 7	trilev[3:0]	R/W		defines FIFO reload activation level - an interrupt to INTC and a request to the DMAU is generated when the FIFO level drops below the value programmed in this field
			1	allowed levels
			2	
			3	
			4	
			5	
			6	
			7	
			8*	
			others	reserved
6	trigen	R/W	0*	disable FIFO level detection - this is equivalent to operate the IIS with a FIFO of length one
			1	enable FIFO level detection
5 to 2	fsc[3:0]	R/W	0b0000*	8 kHz
			0b0001	11.025 kHz
			0b0010	12 kHz
			0b0011	16 kHz
			0b0100	22.05 kHz
			0b0101	24 kHz
			0b0110	32 kHz
			0b0111	44.1 kHz
			0b1000	48 kHz
			others	reserved
1	mute	R/W	0*	mute disabled
			1	mute enabled, the IISSD output is forced to LOW as soon as the current right word has been shifted out - the shift register in the DAO is not reloaded
0	enable	R/W	0*	external interface is disabled - all outputs assume their reset levels after the current right word has been transferred - the interrupt generation is still activated and the DMAU can load the FIFO
			1	external interface is enabled - starts with the transmission of a left word

Table 306: Register iisstat

Bit	Symbol	Access	Value	Description
15 to 8	-	R	0*	reserved
7	underflow [1]	R/C	0*	FIFO underflow flag - does not generate any interrupt
			1	no underflow detected
			1	underflow detected - is set when the DAO attempts to read an empty FIFO
6	siorq [1]	R/C	1*	I/O request flag
5 to 4	-	R	0*	reserved
3 to 0	level[3:0]	R	0*	current FIFO level

[1] These flag have to be explicitly cleared by the user.

Table 307: Register iisdata

Bit	Symbol	Access	Value	Description
15 to 0	data[15:0]	R/W	0*	I²S-bus channel data : first left and then right channel has to be written - the read access to this register returns the last written halfword

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.20.4 Functional description

The I²S-bus interface consists of the VPB bus interface and a stereo digital audio output DAO. The bus interface contains all the registers accessible by the SC: **iiscon** to control the interface, **iisstat** to get the status of the interface, and one 16-bit data register **iisdata** to write the left and right data samples.

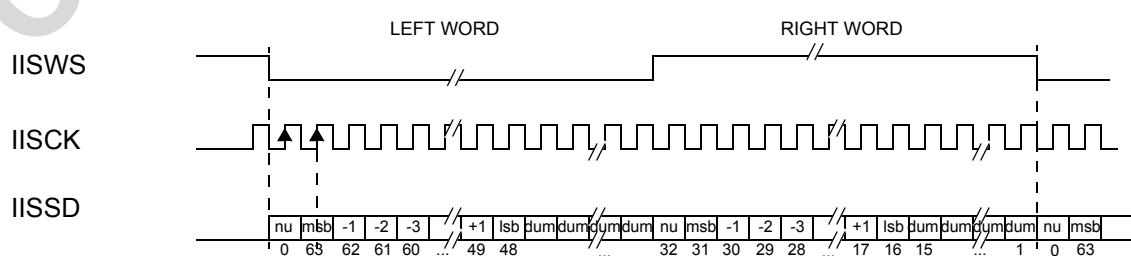
The audio data are transferred to the I²S-bus by means of the **iisdata** register. Left and right channel are written with two consecutive writes to this register. The first data which has to be written is the left channel, then to the right channel. Each second time **iisdata** is written, the two 16-bit data words are transferred to the 32-bit FIFO. The user has to take care that always and even number of half words are transferred.

Inside the I²S-bus, the data are stored in a 8 stage FIFO and are transferred to the DAO shift register. The FIFO level is incremented with every second access to **iisdata**, and decremented for each word transferred to the shift register. When the FIFO level is smaller than the level defined in **iiscon**, an interrupt is issued and **iisstat.siorq** is set. The IIS requests to reload the FIFO as long as the current level is smaller than the value programmed in **iiscon**. If the FIFO level is zero and the DAO requests reload, the flag **underflow** is set. The **underflow** condition itself does not generate an interrupt. The user can use this flag for debugging purposes. If the user does not react on the **underflow** condition, the previous value of the FIFO is shifted out again.

The data stored in the 32-bit shift register are shifted-out with IISCK, which is 64 times the sampling frequency. The left and right channel data are sent over the IISSD line one after the other. The IISWS word select indicates which channel is the left or the right channel.

Remark: When the I²S-bus is enabled, the first data sent corresponds to the last data hold in the 32-bit shift register during the previous use of the I²S-bus. After reset, this data is equal to 0.

The I²S-bus block supports a wide range of output sampling frequencies. The selection of the output rate can be selected in the **iiscon** register. The output frequency is derived from the clk24m input clock. The frequency error is less than 0.1%, because the length of bit 32 and 0 of IISSD and of the related IISCK period are changed according to [Table 308](#).



dum = dummy, nu = not used

Fig 120.I²S output formats

Table 308: Computation of exact output sampling frequency

fsc[3:0]	Selected sampling frequency [1]	Divider ratio [1]	Sampling frequency	Error in%	Length of bits #0 and #32 [1/24MHz] [2]	Length of other bits [1/24MHz] [2]
0000	8 kHz	3000	8 kHz	0	74	46
0001	11.025 kHz	2176	11.029 kHz	0.04	34	34
0010	12 kHz	2000	12 kHz	0	39	31
0011	16 kHz	1500	16 kHz	0	37	23
0100	22.05 kHz	1088	22.06 kHz	0.04	17	17
0101	24 kHz	1000	24 kHz	0	35	15
0110	32 kHz	750	32 kHz	0	34	11
0111	44.1 kHz	544	44.12 kHz	0.04	24	8
1000	48 kHz	500	48 kHz	0	33	7

[1] The sampling frequency is deduced from a 24 MHz reference clock by integer division.

[2] Each audio sampling period consists of 64 bits numbered from #0 to #63. Bits #0 and #32 may have different length than the others 62 bits.

9.20.5 Application information

9.20.5.1 General recommendations

The user may start to fill the FIFO before enabling the external interface in **iiscon** in order to avoid the setting of the **underflow** flag.

The switch of sampling frequency during the operation of the I²S-bus interface is not supported. If the user needs to change the sampling frequency, the **iiscon.enable** bit has to be cleared first.

The I²S-bus interface generates an interrupt after reset of the DVFD818x Family.

9.20.5.2 I²S-bus software architecture

It is proposed to develop two SW modules to manage the I²S-bus peripheral: a driver part to manage the hardware, and a handler part to provide a high level interface to the application like a media player.

The driver part shall use an SCRAM buffer (example 1024 bytes) for the I²S-bus DMA transfers. The DMAU is used in Ring mode (scatter/gather with two linked buffer), with an interrupt in middle and at the end of the buffer. Each time the DMA interrupt occurs, the driver asks the handler to fill the corresponding half DMA buffer.

To provide an appropriate end to an audio stream (at the end of a file or when the stream is stopped), the handler has to provide a last half DMA buffer filled with 0, to be sure that the last audio is silence. When the next interrupt occurs at the end of this last half DMA buffer, the handler asks the driver to disable the I²S-bus (**enable** = 0).

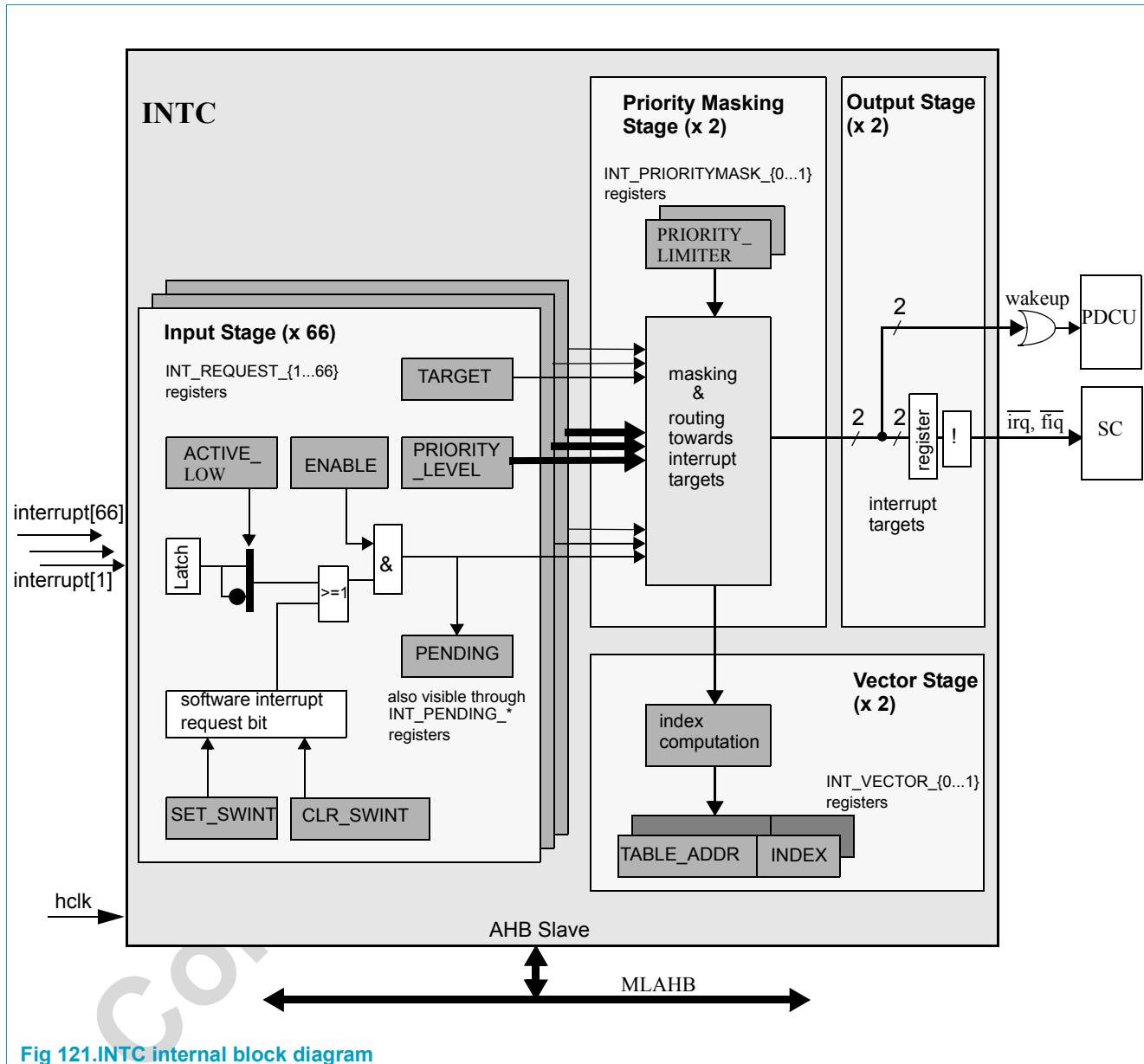
Note: the maximum added silence is $\frac{512}{4} \times \frac{1}{8 \text{ kHz}} = 16 \text{ ms}$

9.21 INTC - Interrupt Controller**9.21.1 Features**

- Interrupt controller for 66 interrupt inputs
- Generates fast (FIQ) and normal (IRQ) interrupts on the ARM926
- Control registers for each interrupt input allow
 - individual enables
 - 32 programmable priority levels for each interrupt
 - software controlled activation of interrupt (for debugging)
- Current serviced interrupt priority can be used to mask any lower priority interrupt
- All interrupts can be routed to either FIQ or IRQ
- All interrupts can be programmed to activate the wake-up from idle. The interrupts which can wake up the SC from Stop or Sleep mode are identified in [Table 325](#).

Company Restricted

9.21.2 Block diagram



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

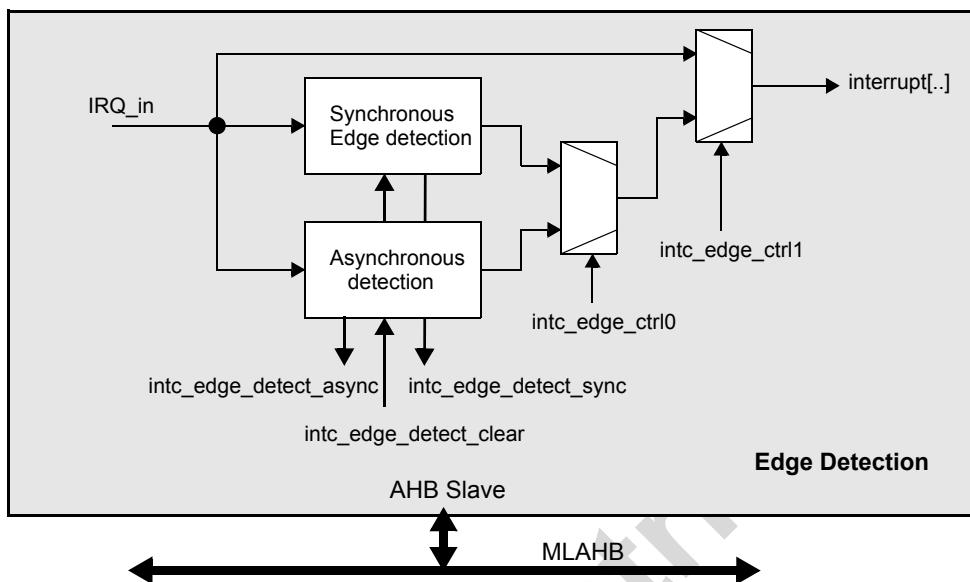


Fig 122.Edge detection circuit block diagram

9.21.3 Software interface

Table 309: Register overview of INTC

Symbol	Name	I/O	reset
intc_priomask_irq	interrupt IRQ target priority threshold register	R/W	-
intc_priomask_fiq	interrupt FIQ target priority threshold register	R/W	-
intc_vector_irq	interrupt IRQ target vector register	R/W	-
intc_vector_fiq	interrupt FIQ target vector register	R/W	-
intc_pending_1	status of interrupt requests 31...1 register	R	-
intc_pending_2	status of interrupt requests 63...32 register	R	-
intc_pending_3	status of interrupt requests 66...64 register	R	-
intc_features	interrupt controller features register	R	0x0001 1F42
intc_request[1...66]	interrupt request 1...66 configuration / software interrupt control registers	R/W	-
intc_edge_detect_sync	status of synchronous edge detection	R	0x0000 0000
intc_edge_detect_async	status of asynchronous edge detection	R	0x0000 0000
intc_edge_clear	edge detection clear register	W	-
intc_edge_ctrl0	edge detection control register 0	R/W	0x0000 0000
intc_edge_ctrl1	edge detection control register 1	R/W	0x0000 0000
intc_mod_id	interrupt controller module ID	R	0x1106 1000

Table 310: Register intc_priomask_irq

Bit	Symbol	Access	Value	Description
31 to 5	-	R	0*	reserved
4 to 0	prio_lim_irq[4:0]	R/W		priority limiter; this field determines a priority threshold that incoming interrupt requests must exceed to trigger irq requests towards processor and power management controller.

Table 311: Register intc_priomask_fiq

Bit	Symbol	Access	Value	Description
31 to 5	-	R	0*	reserved
4 to 0	prio_lim_fiq[4:0]	R/W		priority limiter; this field determines a priority threshold that incoming interrupt requests must exceed to trigger fiq requests towards processor and power management controller.

Table 312: Register intc_vector_irq [1]

Bit	Symbol	Access	Value	Description
31 to 11	table_addr[31:11]	R/W		table address; indicates the start address of a 2048 byte aligned interrupt vector table in memory (optional usage)
			0*	Interrupt vector table is at address 0
10 to 3	index[7:0]	R		interrupt index; indicates the number of the irq request to be served by the processor
			0*	no interrupt to be serviced
2 to 0	-	R	0*	reserved

[1] The intc_vector_irq register content has been designed so it can be used as a vector into a memory based table. For full details read the section in the functional description [on page 385](#).

Table 313: Register intc_vector_fiq [1]

Bit	Symbol	Access	Value	Description
31 to 11	table_addr[31:11]	R/W		table address; indicates the start address of a 2048 byte aligned interrupt vector table in memory (optional usage)
			0	Interrupt vector table is at address 0
10 to 3	index[7:0]	R		interrupt index; indicates the number of the fiq request to be served by the processor
			0	no interrupt to be serviced
2 to 0	-	R	0*	reserved

[1] The intc_vector_fiq register content has been designed so it can be used as a vector into a memory based table. For full details read the section in the functional description [on page 385](#).

Table 314: Register intc_pending_1

Bit	Symbol	Access	Value	Description
31 to 1	intc_pend_31_1	R	~*	interrupt pending; reflects the state of the interrupt[31:1] lines (if needed, converted to active high) OR'ed by the state of the local software interrupt request bit at the time the register is read. Note that the pending bits are also found in the intc_request[N] registers individually present for each interrupt request input
0	-	R	0*	reserved

Table 315: Register intc_pending_2

Bit	Symbol	Access	Value	Description
31 to 0	int_pend_63_32	R	~*	interrupt pending ; reflects the state of the interrupt[63:32] lines (if needed, converted to active high) OR'ed by the state of the local software interrupt request bit at the time the register is read. Note that the pending bits are also found in the intc_request[N] registers individually present for each interrupt request input

Table 316: Register intc_pending_3

Bit	Symbol	Access	Value	Description
31 to 3	-	R	0*	reserved
2 to 0	int_pend_66_64	R	~*	interrupt pending ; reflects the state of the interrupt[66:64] lines (if needed, converted to active high) OR'ed by the state of the local software interrupt request bit at the time the register is read. Note that the pending bits are also found in the intc_request[N] registers individually present for each interrupt request input

Table 317: Register intc_features

Bit	Symbol	Access	Value	Description
31 to 22	-	R	0*	reserved
21 to 16	targets	R	1	targets ; number of interrupt targets supported (minus one) Fixed value 1, meaning two targets (e.g. IRQ, FIQ)
15 to 8	priority	R	31	priority levels ; number of priority levels supported (minus one) Fixed value 31, meaning 32 priority levels
7 to 0	inputs	R	66	interrupt request inputs ; number of interrupt request inputs Fixed value 66

Table 318: Register(s) intc_request[N]

Bit	Symbol	Access	Value	Description
31	pending	R	0	pending ; reflects the state of the interrupt[N] line (if needed, converted to active high) OR'ed by the state of the local software interrupt request bit at the time the register is read. Note that the status of pending interrupts is also visible from the intc_pending_3 , intc_pending_2 and intc_pending_1 registers
			0	this interrupt is not pending
			1	this interrupt is pending
30	set_swint [1]	W	0	set software interrupt ; set software interrupt request
			1	no effect on the state of the local software interrupt request bit
29	clr_swint [1]	W	0	set the state of the local software interrupt request bit to '1'
			1	clear software interrupt ; clear software interrupt request
28	we_priority [2]	W	0	no effect on the state of the local software interrupt request bit
			1	clear the state of the local software interrupt request bit to '0'
			0	write enable priority ; write enable for the priority field
			1	priority_level is unchanged
				priority_level may be changed

Table 318: Register(s) intc_request[N]...continued

Bit	Symbol	Access	Value	Description
27	we_target [2]	W		write enable target;
			0	target is unchanged
			1	target may be changed
26	we_enable [2]	W		write enable enable;
			0	enable is unchanged
			1	enable may be changed
25	we_active_low [2]	W		write enable active low;
			0	active_low is unchanged
			1	active_low may be changed
24 to 18	-	R	-	reserved; should be written as 0
17	active_low	R/W		active low; sets the active state of the interrupt. Depending on the state of we_active_low , this field may contain a don't care value
			0*	the interrupt signal is interpreted as active high
			1	the interrupt signal is interpreted as active low
16	enable	R/W		enable; controls whether an interrupt request is enabled for further processing by the interrupt controller. Depending on the state of we_enable , this field may contain a don't care value
			0*	the interrupt request is discarded. It cannot cause an SC interrupt or wakeup request to PDCU
			1	the interrupt request may cause an SC interrupt or wakeup request to PDCU
15 to 9	-	R	-	reserved; should be written as 0
8	target	R/W		target; defines the interrupt target of an interrupt request. Depending on the state of we_target , this field may contain a don't care value
			0*	the interrupt request shall lead to an IRQ and wakeup to the PDCU
			1	the interrupt request shall lead to an FIQ and wakeup to the PDCU
7 to 5	-	R	-	reserved; should be written as 0
4 to 0	prio_level	R/W		priority level; determines the priority level of the interrupt request. Depending on the state of we_priority , this field may contain a don't care value
			0	interrupt is always ignored (same as enable = 0)
			1	lowest priority level
			others	intermediate priority levels
			31	highest priority level

[1] These bits allow software to trigger (and clear) an interrupt in the absence of the real interrupt request

[2] When attempting to change only some of **active_low**, **enable**, **target**, **prio_level**, it is possible (using **we_priority**, **we_target**, **we_enable**, **we_active_low**) to qualify which of the former fields are being changed. If fields are not being altered then those fields can be written with any value without changing the content. This simplifies, for example, enabling an interrupt, e.g. **we_enable**=1, **enable**=1, with all other bits are 0.

Table 319: Register intc_edge_detect_sync

Bit	Symbol	Access	Value	Description
31 to 25	-	R	0*	reserved
24	int_pwr_fail_sstat	R	0*	status of the synchronous edge detection for power fail interrupt
23	int_fsi_ip_sstat	R	0*	status of the synchronous edge detection for IPINT speech frame interrupt
22	int_fsi_sstat	R	0*	status of the synchronous edge detection for TBU speech frame interrupt
21	int_mcu_irq1_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP MCU low priority interrupt 1
20	int_mcu_irq0_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP MCU low priority interrupt 0
19	int_rcu_irq_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP RCU low priority interrupt
18	int_tcu_irq_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP TCU low priority interrupt
17	int_sif_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP serial interface interrupt
16	int_sync_det_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP sync detected interrupt
15	int_rfcu_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP RFCU interrupt

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 319: Register intc_edge_detect_sync...continued

Bit	Symbol	Access	Value	Description
14	int_mcu_fiq1_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP MCU high priority interrupt 1
13	int_bcnt_s1_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP short bit counter interrupt1
12	int_bcnt_s0_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP short bitcounter interrupt0
11	int_rbuf_full_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP receive buffer full interrupt
10	int_tbuf_emp_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP transmit buffer empty interrupt
9	int_mcu_fiq0_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP MCU high priority interrupt 0
8	int_rcu_fiq_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP RCU high priority interrupt
7	int_tcu_fiq_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP TCU high priority interrupt
6	int_bcnt1_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP bitcounter interrupt 1
5	int_bcnt0_sstat ^{[1][2]}	R	0*	status of the synchronous edge detection for BMP bitcounter interrupt 0
4	int_ad_sstat	R	0*	status of the synchronous edge detection for AD conversion complete interrupt
3	int_bmptime_sstat ^[1]	R	0*	status of the synchronous edge detection for BMP timer interrupt
2	int_dsp1_sstat	R	0*	status of the synchronous edge detection for DSP request 1 via TMU
1	int_dsp0_sstat	R	0*	status of the synchronous edge detection for DSP request 0 via TMU
0	int_ana_sstat	R	0*	status of the synchronous edge detection for APU interrupt

[1] Not supported at DVFD8187

[2] Optional as additional OS timer at DVFD8187 in DSP group application SW

Table 320: Register intc_edge_detect_async

Bit	Symbol	Access	Value	Description
31 to 25	-	R	0*	reserved
24	int_pwr_fail_astat	R	0*	status of the asynchronous detection for power fail interrupt
23	int_fsi_ip_astat	R	0*	status of the asynchronous detection for IPINT speech frame interrupt
22	int_fsi_astat	R	0*	status of the asynchronous detection for TBU speech frame interrupt
21	int_mcu_irq1_astat ^[1]	R	0*	status of the asynchronous detection for BMP MCU low priority interrupt 1
20	int_mcu_irq0_astat ^[1]	R	0*	status of the asynchronous detection for BMP MCU low priority interrupt 0
19	int_rcu_irq_astat ^[1]	R	0*	status of the asynchronous detection for BMP RCU low priority interrupt
18	int_tcu_irq_astat ^[1]	R	0*	status of the asynchronous detection for BMP TCU low priority interrupt
17	int_sif_astat ^[1]	R	0*	status of the asynchronous detection for BMP serial interface interrupt
16	int_sync_det_astat ^[1]	R	0*	status of the asynchronous detection for BMP sync detected interrupt
15	int_rfcl_astat ^[1]	R	0*	status of the asynchronous detection for BMP RFCU interrupt
14	int_mcu_fiq1_astat ^[1]	R	0*	status of the asynchronous detection for BMP MCU high priority interrupt 1
13	int_bcnt_s1_astat ^[1]	R	0*	status of the asynchronous detection for BMP short bit counter interrupt1
12	int_bcnt_s0_astat ^[1]	R	0*	status of the asynchronous detection for BMP short bitcounter interrupt0
11	int_rbuf_full_astat ^[1]	R	0*	status of the asynchronous detection for BMP receive buffer full interrupt
10	int_tbuf_emp_astat ^[1]	R	0*	status of the asynchronous detection for BMP transmit buffer empty interrupt
9	int_mcu_fiq0_astat ^[1]	R	0*	status of the asynchronous detection for BMP MCU high priority interrupt 0
8	int_rcu_fiq_astat ^[1]	R	0*	status of the asynchronous detection for BMP RCU high priority interrupt
7	int_tcu_fiq_astat ^[1]	R	0*	status of the asynchronous detection for BMP TCU high priority interrupt
6	int_bcnt1_astat ^[1]	R	0*	status of the asynchronous detection for BMP bitcounter interrupt 1
5	int_bcnt0_astat ^{[1][2]}	R	0*	status of the asynchronous detection for BMP bitcounter interrupt 0
4	int_ad_astat	R	0*	status of the asynchronous detection for AD conversion complete interrupt
3	int_bmptime_astat ^[1]	R	0*	status of the asynchronous detection for BMP timer interrupt
2	int_dsp1_astat	R	0*	status of the asynchronous detection for DSP request 1 via TMU
1	int_dsp0_astat	R	0*	status of the asynchronous detection for DSP request 0 via TMU
0	int_ana_astat	R	0*	status of the asynchronous detection for APU interrupt

[1] Not supported at DVFD8187

[2] Optional as additional OS timer at DVFD8187 in DSP group application SW

Table 321: Register intc_edge_detect_clear

Bit	Symbol	Access	Value	Description
31 to 25	-	W	-	reserved
24 to 0	edge_detect_clear[24:0]	W	-	clear; Writing a '1' to a bit will clear the corresponding bit in the registers intc_edge_detect_sync and intc_edge_detect_async

Table 322: Register intc_edge_detect_ctrl0

Bit	Symbol	Access	Value	Description
31 to 25	-	-	0*	reserved
24	int_pwr_fail_edsel	R/W		edge detection circuit selection for power fail interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
23	int_fsi_ip_edsel	R/W		edge detection circuit selection for IPINT speech frame interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
22	int_fsi_edsel	R/W		edge detection circuit selection for TBU speech frame interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
21	int_mcu_irq1_edsel ^[1]	R/W		edge detection circuit selection for BMP MCU low priority interrupt 1
			1	asynchronous detection is used
			0*	synchronous edge detection is used
20	int_mcu_irq0_edsel ^[1]	R/W		edge detection circuit selection for BMP MCU low priority interrupt 0
			1	asynchronous detection is used
			0*	synchronous edge detection is used
19	int_rcu_irq_edsel ^[1]	R/W		edge detection circuit selection for BMP RCU low priority interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
18	int_tcu_irq_edsel ^[1]	R/W		edge detection circuit selection for BMP TCU low priority interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
17	int_sif_edsel ^[1]	R/W		edge detection circuit selection for BMP serial interface interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
16	int_sync_det_edsel ^[1]	R/W		edge detection circuit selection for BMP sync detected interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
15	int_rfcu_edsel ^[1]	R/W		edge detection circuit selection for BMP RFCU interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
14	int_mcu_fiq1_edsel ^[1]	R/W		edge detection circuit selection for BMP MCU high priority interrupt 1
			1	asynchronous detection is used
			0*	synchronous edge detection is used

Table 322: Register intc_edge_detect_ctrl0...continued

Bit	Symbol	Access	Value	Description
13	int_bcnt_s1_edsel ^[1]	R/W		edge detection circuit selection for BMP short bit counter interrupt1
			1	asynchronous detection is used
			0*	synchronous edge detection is used
12	int_bcnt_s0_edsel ^[1]	R/W		edge detection circuit selection for BMP short bitcounter interrupt0
			1	asynchronous detection is used
			0*	synchronous edge detection is used
11	int_rbuf_full_edsel ^[1]	R/W		edge detection circuit selection for BMP receive buffer full interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
10	int_tbuf_emp_edsel ^[1]	R/W		edge detection circuit selection for BMP transmit buffer empty interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
9	int_mcu_fiq0_edsel ^[1]	R/W		edge detection circuit selection for BMP MCU high priority interrupt 0
			1	asynchronous detection is used
			0*	synchronous edge detection is used
8	int_rcu_fiq_edsel ^[1]	R/W		edge detection circuit selection for BMP RCU high priority interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
7	int_tcu_fiq_edsel ^[1]	R/W		edge detection circuit selection for BMP TCU high priority interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
6	int_bcnt1_edsel ^[1]	R/W		edge detection circuit selection for BMP bitcounter interrupt 1
			1	asynchronous detection is used
			0*	synchronous edge detection is used
5	int_bcnt0_edsel ^{[1][2]}	R/W		edge detection circuit selection for BMP bitcounter interrupt 0
			1	asynchronous detection is used
			0*	synchronous edge detection is used

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 322: Register intc_edge_detect_ctrl0...continued

Bit	Symbol	Access	Value	Description
4	int_ad_edsel	R/W		edge detection circuit selection for AD conversion complete interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
3	int_bmptime_edsel ^[1]	R/W		edge detection circuit selection for BMP timer interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used
2	int_dsp1_edsel	R/W		edge detection circuit selection for DSP request 1 via TMU
			1	asynchronous detection is used
			0*	synchronous edge detection is used
1	int_dsp0_edsel	R/W		edge detection circuit selection for DSP request 0 via TMU
			1	asynchronous detection is used
			0*	synchronous edge detection is used
0	int_ana_edsel	R/W		edge detection circuit selection for APU interrupt
			1	asynchronous detection is used
			0*	synchronous edge detection is used

[1] Not supported at DVFD8187

[2] Optional as additional OS timer at DVFD8187 in DSP group application SW

Table 323: Register intc_edge_detect_ctrl1

Bit	Symbol	Access	Value	Description
31 to 25	-	-	0*	reserved
24	int_pwr_fail_ssel	R/W		power fail interrupt request is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
23	int_fsi_ip_ssel	R/W		IPINT speech frame interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
22	int_fsi_ssel	R/W		TBU speech frame interrupt
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
21	int_mcu_irq1_ssel ^[1]	R/W		BMP MCU low priority interrupt 1 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
20	int_mcu_irq0_ssel ^[1]	R/W		BMP MCU low priority interrupt 0 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
19	int_rcu_irq_ssel ^[1]	R/W		status of the asynchronous edge detection for BMP RCU low priority interrupt
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)

Table 323: Register intc_edge_detect_ctrl1 ...continued

Bit	Symbol	Access	Value	Description
18	int_tcu_irq_ssel ^[1]	R/W		BMP TCU low priority interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
17	int_sif_ssel ^[1]	R/W		BMP serial interface interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
16	int_sync_det_ssel ^[1]	R/W		BMP sync detected interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
15	int_rcu_ssel ^[1]	R/W		BMP RFCU interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
14	int_mcu_fiq1_ssel ^[1]	R/W		BMP MCU high priority interrupt 1 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
13	int_bcnt_s1_ssel ^[1]	R/W		BMP short bit counter interrupt1 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
12	int_bcnt_s0_ssel ^[1]	R/W		BMP short bitcounter interrupt0 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
11	int_rbuf_full_ssel ^[1]	R/W		BMP receive buffer full interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
10	int_tbuf_emp_ssel ^[1]	R/W		BMP transmit buffer empty interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
9	int_mcu_fiq0_ssel ^[1]	R/W		BMP MCU high priority interrupt 0 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
8	int_rcu_fiq_ssel ^[1]	R/W		BMP RCU high priority interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
7	int_tcu_fiq_ssel ^[1]	R/W		BMP TCU high priority interrupt is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
6	int_bcnt1_ssel ^[1]	R/W		BMP bitcounter interrupt 1 is generated by
			1	the original interrupt signal (raw source)
			0*	by the edge detection circuit (see Table 322)
5	int_bcnt0_ssel ^{[1][2]}	R/W		BMP bitcounter interrupt 0 is generated by
			1	the original interrupt signal (raw source)

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 323: Register intc_edge_detect_ctrl1...continued

Bit	Symbol	Access	Value	Description
4	int_ad_ssel	R/W	0*	by the edge detection circuit (see Table 322)
			1	status of the asynchronous edge detection for AD conversion complete interrupt
			0*	the original interrupt signal (raw source)
3	int_bmptime_ssel ^[1]	R/W	0*	by the edge detection circuit (see Table 322)
			1	status of the asynchronous edge detection for BMP timer interrupt
			0*	the original interrupt signal (raw source)
2	int_dsp1_ssel	R/W	0*	by the edge detection circuit (see Table 322)
			1	status of the asynchronous edge detection for DSP request 1 via TMU
			0*	the original interrupt signal (raw source)
1	int_dsp0_ssel	R/W	0*	by the edge detection circuit (see Table 322)
			1	status of the asynchronous edge detection for DSP request 0 via TMU
			0*	the original interrupt signal (raw source)
0	int_ana_ssel	R/W	0*	by the edge detection circuit (see Table 322)
			1	status of the asynchronous edge detection for APU interrupt
			0*	the original interrupt signal (raw source)

[1] Not supported at DVFD8187

[2] Optional as additional OS timer at DVFD8187 in DSP group application SW

Table 324: Register intc_mod_id

Bit	Symbol	Access	Value	Description
31 to 16	id_num	R	0x1106*	identification number ; unique module identifier
15 to 12	major_rev	R	1*	major revision ; major revision of module implementation. Note that there is no guaranteed software compatibility between different major revisions of the same module
11 to 8	minor_rev	R	0*	minor revision ; minor revision of module implementation, starting at 0. There is software compatibility between minor revisions
7 to 0	aperture	R	0*	aperture size ; the value indicates a single 4 KByte address space is required for this block - N.B. can be ignored by software.

9.21.4 Functional description

Table 325: Overview of the SC interrupt sources

INTC interrupt number	Interrupt signal	Active level	Connected to	Wake-up from sleep capability
interrupt 66	int_pwr_fail	high	power fail interrupt	yes
interrupt 65	int_fsi_ip	high	IPINT speech frame interrupt	no
interrupt 64	int_fsi	high	BMPTBU speech frame interrupt	no
interrupt 63	int_mcu_irq1 [6]	high	BMP MCU low priority interrupt 1	no
interrupt 62	int_mcu_irq0 [6]	high	BMP MCU low priority interrupt 0	no
interrupt 61	int_rcu_irq [6]	high	BMP RCU low priority interrupt	no
interrupt 60	int_tcu_irq [6]	high	BMP TCU low priority interrupt	no
interrupt 59	int_sif [6]	high	BMP serial interface interrupt	no
interrupt 58	int_sync_det [6]	high	BMP sync detected interrupt	no
interrupt 57	int_rfcu	high	BMP RFCU interrupt	no
interrupt 56	int_mcu_fiq1 [6]	high	BMP MCU high priority interrupt 1	no
interrupt 55	int_bcnt_s1 [6]	high	BMP short bit counter interrupt1	no
interrupt 54	int_bcnt_s0 [6]	high	BMP short bitcounter interrupt0	no
interrupt 53	int_rbuf_full [6]	high	BMP receive buffer full interrupt	no
interrupt 52	int_tbuf_emp [6]	high	BMP transmit buffer empty interrupt	no
interrupt 51	int_mcu_fiq0 [6]	high	BMP MCU high priority interrupt 0	no
interrupt 50	int_rcu_fiq [6]	high	BMP RCU high priority interrupt	no
interrupt 49	int_tcu_fiq [6]	high	BMP TCU high priority interrupt	no
interrupt 48	pdcu_intr	high	wake-up interrupt of PDCU	no [1]
interrupt 47	fir_intr	high	FIR interrupt	no
interrupt 46	fir_fi_intr	high	FIR Frame Invalid Interrupt	no
interrupt 45	amu_intr	high	AMU interrupt	no
interrupt 44	int_bcnt1 [6]	high	BMP bitcounter interrupt 1	no
interrupt 43	int_bcnt0 [6][7]	high	BMP bitcounter interrupt 0	no
interrupt 42	etb_etbfull	high	ETB full interrupt	no
interrupt 41	etb_acqcomp	high	ETB acquisition complete interrupt	no
interrupt 40	reserved	-	-	-
interrupt 39	reserved	-	-	-
interrupt 38	reserved	-	-	-
interrupt 37	fint0_sc_irq	high	TBU interrupt	yes

Compliance

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 325: Overview of the SC interrupt sources...continued

INTC interrupt number	Interrupt signal	Active level	Connected to	Wake-up from sleep capability
interrupt 36	reserved	-	-	-
interrupt 35	int_ad	high	AD conversion complete interrupt	no
interrupt 34	int_bmptime ^[6]	high	BMP timer interrupt	yes
interrupt 33	reserved	-	-	-
interrupt 32	reserved	-	-	-
interrupt 31	reserved	-	-	-
interrupt 30	usim_intr	high	USIM interrupt	no
interrupt 29	dmau_intcombine	high	DMAU combined interrupt	no
interrupt 28	i2c_intr	high	IIC receive/transmit interrupt	no
interrupt 27	fci_int0	high	FCI interrupt	no
interrupt 26	spi1_tirq	high	SPI1 interrupt	no ^[2]
interrupt 25	spi2_tirq	high	SPI2 interrupt	no ^[2]
interrupt 24	uart1_tirq	high	UART1 interrupt	no ^[3]
interrupt 23	uart2_tirq	high	UART2 interrupt	no ^[3]
interrupt 22	etn1_intr	high	ETN1 interrupt	no
interrupt 21	etn2_intr	high	ETN2 interrupt	no
interrupt 20	int_dsp1	high	DSP request 1 via TMU	no
interrupt 19	int_dsp0	high	DSP request 0 via TMU	no
interrupt 18	sctu1_intr	high	SCTU1 timer interrupt	yes
interrupt 17	sctu2_intr	high	SCTU2 timer interrupt	yes
interrupt 16	kbs_intr	high	KBS interrupt	yes
interrupt 15	arm9_commrx	high	ARM commrx interrupt	no
interrupt 14	arm9_commtx	high	ARM commtx interrupt	no
interrupt 13	usb_irq	high	USB IRQ interrupt	no ^[4]
interrupt 12	usb_fiq	high	USB FIQ interrupt	no ^[4]
interrupt 11	reserved	-	-	-
interrupt 10	reserved	-	-	-
interrupt 9	reserved	-	-	-
interrupt 8	reserved	-	-	-
interrupt 7	reserved	-	-	-
interrupt 6	iis_intr	high	IIS interrupt	no
interrupt 5	reserved	-	-	-
interrupt 4	int_ana	high	interrupt from APU	yes
interrupt 3	extint_irq3	low	external interrupt 3	yes
interrupt 2	extint_irq2	low	external interrupt 2	yes
interrupt 1	extint_irq1	low	external interrupt 1	yes
interrupt 0	-	-	Ghost interrupt ^[5]	-

[1] PDCU wake-up from sleep triggers this interrupt

[2] Can wake-up from sleep by using EXTINT on SPIx SCSx signal. See EXTINT description Section 9.16

[3] Can wake-up from sleep by using EXTINT on RXD, CTS or DSR (DSR is only for UART1). See EXTINT description Section 9.16

- | | | |
|-----|--|-------------|
| [4] | USB wake-up from sleep is done by using EXTINT. See EXTINT description in Section 9.16 | 1 |
| [5] | In the absence of interrupts the INTC vector will point to the base address of the vector table (this appears to indicate interrupt[0] which does not physically exist). For this to happen during an interrupt service routine an interrupt must have previously been generated which has since disappeared (this is abnormal behavior). To cover this case properly (without handling this as an exception) a legal (but otherwise empty) interrupt service routine should be pointed to with the vector found at this address | 2
3
4 |
| [6] | Not supported at DVFD8187 | 5 |
| [7] | Optional as additional OS timer at DVFD8187 in DSP group application SW | 6
7 |

The interrupt controller receives the interrupt requests from the peripherals. After priority sorting it triggers an interrupt to the ARM core either on **fiq_n** (fast interrupt request) or **irq_n** (normal interrupt request) inputs. In the Interrupt Service Routines (ISR) for IRQ and FIQ, the ARM core should read the **intc_vector_irq** or **intc_vector_fiq** registers in the INTC (respectively) to know which interrupt is active. Each interrupt is configured by programming its interrupt control register **intc_request[n]** storing the priority level and the type of interrupt (active high or low). To enable an interrupt, set the bit **intc_request[n].enable**, and choose its target (IRQ or FIQ) by programming **intc_request[n].target**. The priorities of all enabled and active interrupts are sorted and compared in the priority detection logic with the current level stored in **intc_priomask_irq** (or **intc_priomask_fiq**). The highest priority interrupt activates its chosen target (FIQ or IRQ).

Other debug facilities are available in **intc_request[n]**. For example it is possible to:

- test the RTOS interrupt handling without using a device specific ISR
- do software emulation of an interrupt requesting device, including interrupts

The registers **intc_pending_1**, **intc_pending_2** and **intc_pending_3** are more orientated to debug, allowing an overview of all pending interrupts.

The remaining registers **intc_features** and **intc_mod_id** are read-only registers allowing a generic ISR to work with different versions of this interrupt controller.

The FIQ interrupt has the advantage of using a dedicated register bank for registers **R8_FIQ** to **R14_FIQ**. Note that FIQ interrupts may not be vectored.

ARM architectural aspects of IRQ/FIQ: The IRQ can be used for any normal interrupt. It gets two banked registers **R13_IRQ** and **R14_IRQ** when it is activated. All used registers should be saved on the stack. As soon as the core enters the FIQ interrupt routine, the enable flags I and F are set in the status register **cpsr** (ARM internal register) disabling further interrupts. An IRQ interrupt is handled similarly except that only the I flag is set.

Wakeup: INTC has a purely combinatorial path to the wakeup output. This means that any block producing an interrupt while the SC and INTC are not clocked will wake-up the system controller thus bringing it out of the Idle, Stop, Sleep or Off modes. This concerns a limited number of special blocks (as indicated in [Table 325](#)). The wake-up output of the INTC indicates to the PDCU that the clocks should be re-enabled, then the ARM core can serve the corresponding interrupt (IRQ or FIQ according to the **target** setting).

9.21.5 Application information

9.21.5.1 Priority handling in interrupt handler

In order to allow an interrupt to be interrupted only by another source which has a higher priority level, it is required to update the intc_piomask_irq registers. This is not done by the HW.

Pseudo-code example:

```

void IRQ_handler()
{
U32 interrupt_nb;
U32 priority_mask_save;

interrupt_nb = IRQ_INDEX(); /* function which return the value of intc_vector_irq.index field */

/* set priority mask for nesting, save it before */
priority_mask_save = intc_piomask_irq;
intc_piomask_irq = intc_ctrl[interrupt_nb] & 0x0000000F;

EnableIRQ(); /* Function that re-enable ARM irq interrupt line */

***** Interrupt handling here ****/
switch (interrupt_nb)
{
    case xxxx:
        /* interrupt source xxxx is cleared */
}

/* restore previous priority mask */
intc_piomask_irq = priority_mask_save;
}

```

Remark: intc_piomask_irq has to be initialized to 0 to allow first interrupt with priority set to 1 to be fed to the SC

Remark: Same mechanism can be used for FIQ if more than one source is used (using more than one source on FIQ is not recommended as optimization on FIQ routine will then be low)

Remark: The vector stage provides one vector register per interrupt target. If the priority condition is true for a multitude of input stages, then the input stage with the highest index is taken.

9.21.5.2 Edge detection circuit

The synchronous edge detection circuit should be selected for the interrupt sources mentioned in [Table 322](#) and [Table 323](#) in active and idle mode while the asynchronous detection circuit should be selected in stop, sleep or PSM mode (due to INTC runs here with 32kHz and e.g. BMP interrupts have a width of one bitclock only).

Only positive pulses are processed by the edge detection circuit.

9.21.5.3 Reserved interrupts

The reserved interrupt sources can be used as SW interrupts.

9.21.5.4 Additional sources to wake up from stop, sleep or sleep with psm

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54**CAUTION**

Beside the interrupts indicated in [Table 325](#) as wake up sources any other block able to raise an interrupt can wake up the system if enabled. It is not recommended to take use of it because these blocks operate than outside the specification.

Company Restricted

9.22 IPINT - IOM/PCM Interface

The IPINT provides the interface between the host processor and an external PCM/IOM highway.

9.22.1 Features

- IOM-2 and PCM mode are supported (single and double clock per data bit)
- 8/12/24/32 slots per frame formats are supported (up to 12 slots can be used)
- Linear, log a-law and u-law data formats.
- Master and slave mode operation is supported
- Long, short and IOM frame sync generation.
- Host processor access to IPINT data bus for signal processing

9.22.2 Block diagram

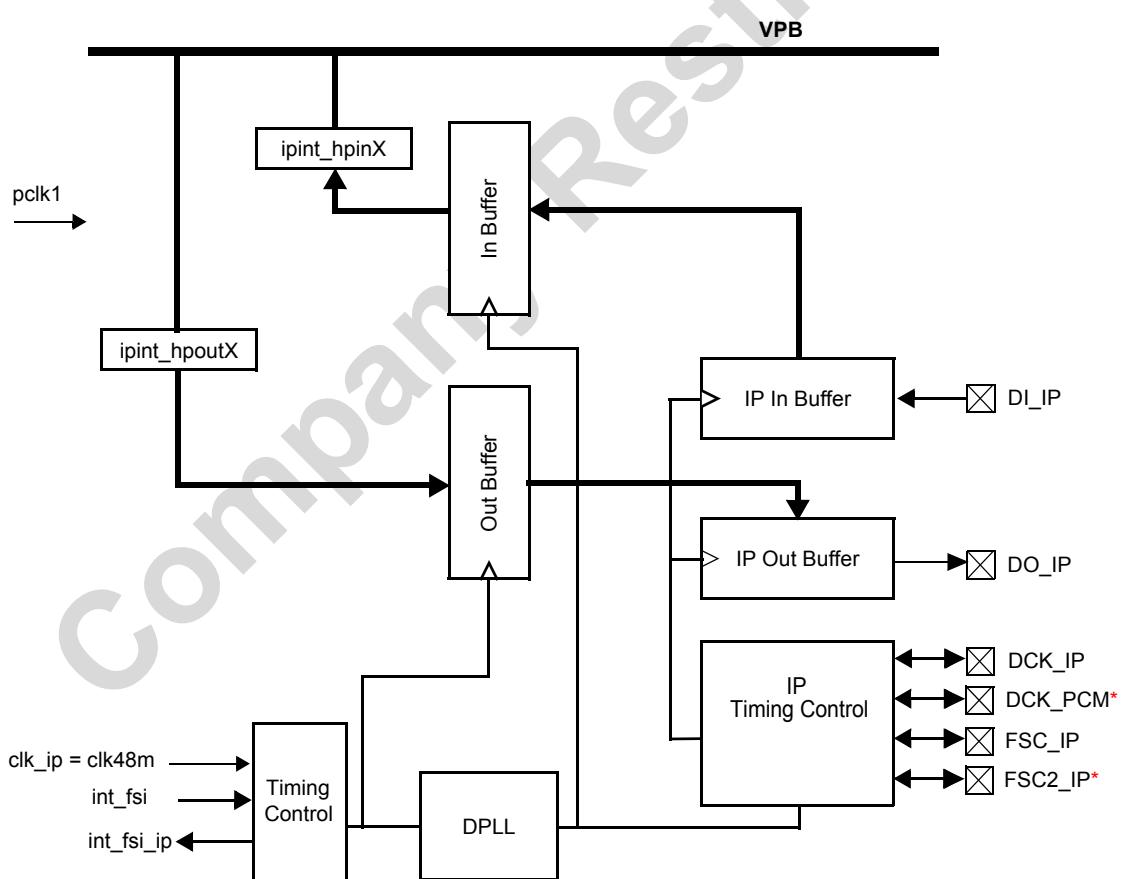


Fig 123. IPINT block diagram

Note: Pins marked with “*” are not available at DVFD818x

9.22.3 Hardware interface

Table 326: IPINT- interface pin overview

PIN	Name	I/O	Description
DO_IP	Data out	O	PCM/IOM data output
DI_IP	Data in	I	PCM/IOM data input
DCK_IP	IOM clock	I/O	Data clock (input in slave mode, output in master mode)
FSC_IP	Frame sync	I/O	8kHz frame sync signal (input in slave mode, output in master mode).

AC Characteristics PCM Mode

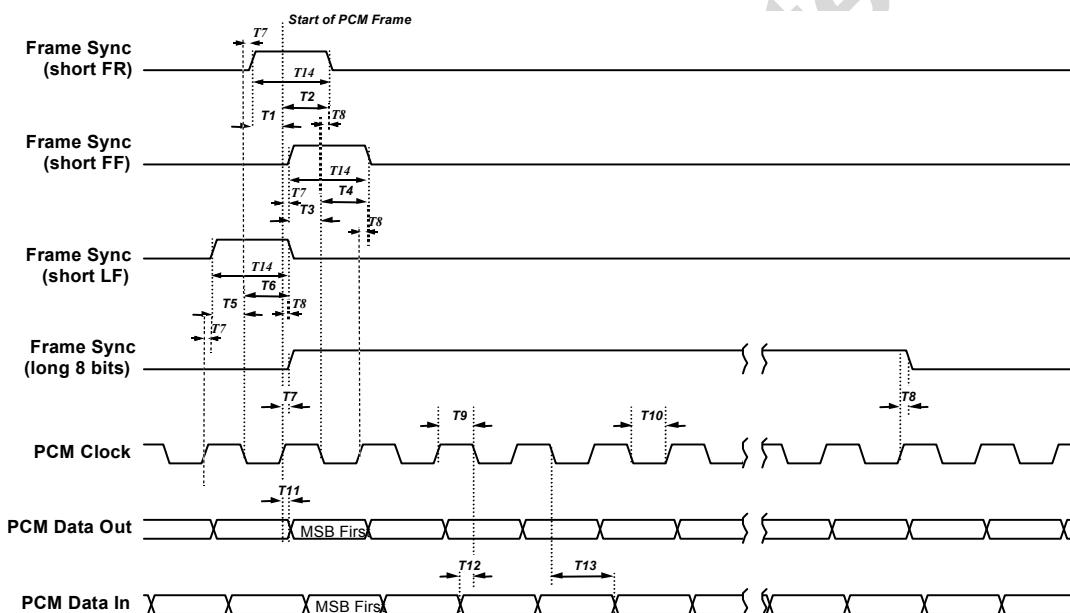


Fig 124.PCM/IOM port detailed external timing diagram for single clocked data (PCM mode)

Table 327: AC characteristics of the IPINT-bus interface in PCM master mode (see Figure 124)

Name	Parameter	Min	Typ	Max	UNIT
T _{IPINT}	IPINT clock time base		1/clk_ipint		ns
T7	Frame Sync to clock rising edge delay		T10-T _{IPINT}	20	ns
T8	Frame Sync to clock falling edge hold time		T9-T _{IPINT}	20	ns
T9	PCM clock high time		see Table 347		ns
T10	PCM clock low time		see Table 347		ns
T11	PCM data out valid delay	0	60		ns
T12	PCM data in set-up time	20	0		ns
T13	PCM data in hold time	50	0		ns

[1] FR denotes the first rising clock edge in the frame

[2] FF denotes the first falling clock edge in the frame

[3] LF denotes the last falling clock edge in the frame

Table 328:AC characteristics of the IPINT-bus interface in PCM slave mode (see Figure 124)

Name	Parameter	Min	Typ	Max	UNIT
T1	Frame Sync (short FR) to Clock rising edge setup time	80			ns
T2	Frame Sync (short FR) to Clock rising edge hold time	50			ns
T3	Frame Sync (short FF) to Clock falling edge setup time	80			ns
T4	Frame Sync (short FF) to Clock falling edge hold time	50			ns
T5	Frame Sync (short LF) to Clock falling edge setup time	80			ns
T6	Frame Sync (short LF) to Clock falling edge hold time	50			ns
T9	PCM Clock high time	80			ns
T10	PCM clock low time	80			ns
T11	PCM data out valid delay	0		60	ns
T12	PCM data in set-up time	20			ns
T13	PCM data in hold time	50			ns
T14	Frame Sync (short FR, FF or LF) high time		T9+T10		ns

[1] *FR* denotes the first rising clock edge in the frame[2] *FF* denotes the first falling clock edge in the frame[3] *LF* denotes the last falling clock edge in the frame

9.22.3.1 AC Characteristics IOM mode

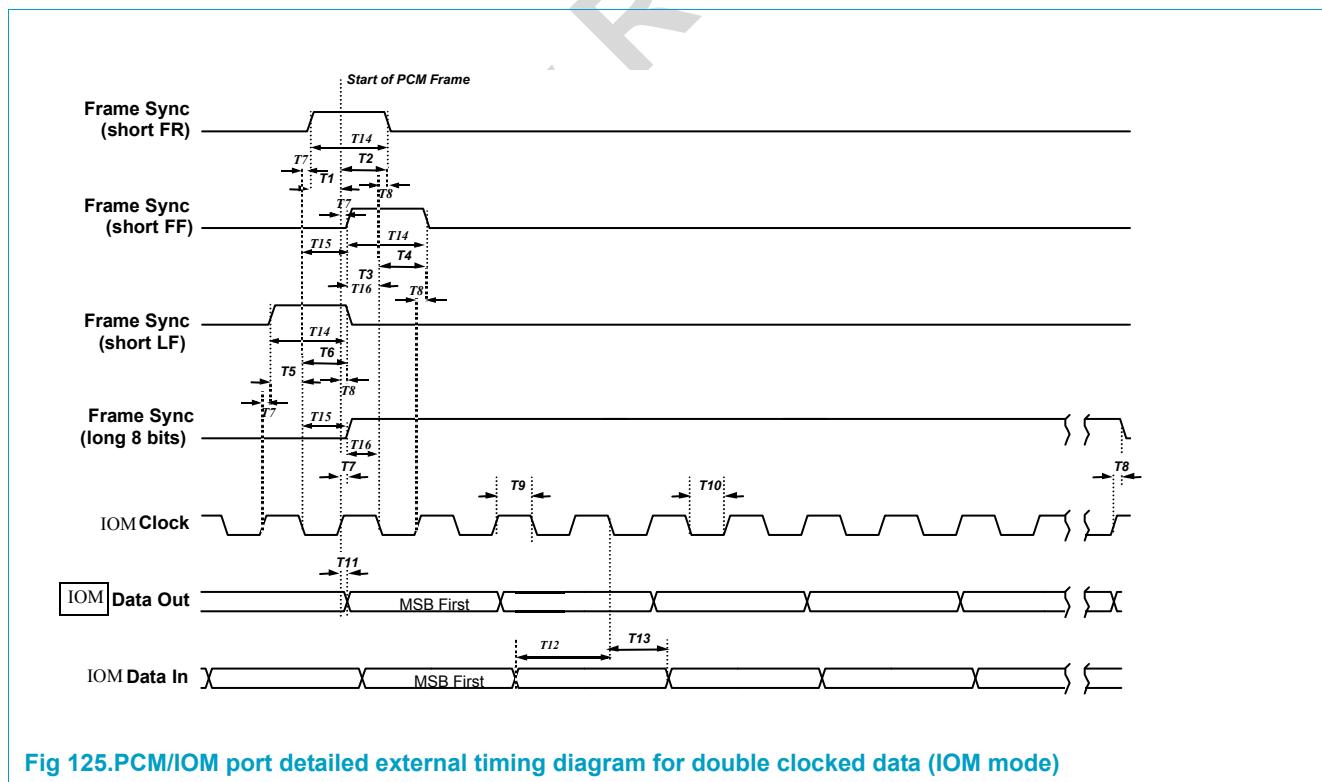
**Fig 125.PCM/IOM port detailed external timing diagram for double clocked data (IOM mode)**

Table 329:AC characteristics of the IPINT-bus interface in IOM master mode (see Figure 125)

Name	Parameter	Min	Typ	Max	UNIT
T7	Frame Sync to clock rising edge delay	-20		20	ns
T8	Frame Sync to clock falling edge hold time	-20		20	ns
T9	IOM clock high time			see Table 347	ns
T10	IOM clock low time			see Table 347	ns
T11	IOM data out valid delay	50 ^[4] 15 ^[5]		90 ^[4] 55 ^[5]	ns
T12	IOM data in set-up time	20	0		ns
T13	IOM data in hold time	50	0		ns

[1] *FR* denotes the first rising clock edge in the frame[2] *FF* denotes the first falling clock edge in the frame[3] *LF* denotes the last falling clock edge in the frame[4] If *ipint_ctrl0.freq* <> 111[5] If *ipint_ctrl0.freq* = 111**Table 330:AC characteristics of the IPINT-bus interface in IOM slave mode (see Figure 125)**

Name	Parameter	Min	Typ	Max	UNIT
T1	Frame Sync (short FR) to Clock rising edge setup time	70			ns
T2	Frame Sync (short FR) to Clock rising edge hold time	40			ns
T3	Frame Sync (short FF) to Clock falling edge setup time	70			ns
T4	Frame Sync (short FF) to Clock falling edge hold time	40			ns
T5	Frame Sync (short LF) to Clock falling edge setup time	70			ns
T6	Frame Sync (short LF) to Clock falling edge hold time	40			ns
T9	PCM Clock high time	80			ns
T10	PCM clock low time	80			ns
T11	PCM data out valid delay	0		30	ns
T12	PCM data in set-up time	30			ns
T13	PCM data in hold time	45			ns
T14	Frame Sync (short FR, FF or LF) high time			T9+T10	ns
T15	Clock to Frame Sync (long, short FF) hold time	30			
T16	Clock to Frame Sync (long, short FF) setup time	30			

[1] *FR* denotes the first rising clock edge in the frame[2] *FF* denotes the first falling clock edge in the frame[3] *LF* denotes the last falling clock edge in the frame

9.22.4 Software interface

All registers are accessible via the ARM Peripheral Bus (VPB), with a 16-bits data interface. The IPINT uses linear 16-bit 2's complement, 14-bit 2's complement, linear 13-bit 2's complement, and PCM 8-bit a-law/u-law.

Table 331: Register overview of the IPINT

Symbol	Name	I/O	Reset
ipint_global	IPINT global register	R/W	0x0000
ipint_ctrl0	PCM/IOM port control (0)	R/W	0x0000
ipint_ctrl1	PCM/IOM port control (1)	R/W	0x0000
ipint_hpout0	PCM/IOM port data out - slot 0 and slot 1	R/W	0x0000
ipint_hpout1	PCM/IOM port data out - slot 2 and slot 3	R/W	0x0000
ipint_hpout2	PCM/IOM port data out - slot 4 and slot 5	R/W	0x0000
ipint_hpout3	PCM/IOM port data out - slot 6 and slot 7	R/W	0x0000
ipint_hpout4	PCM/IOM port data out - slot 8 and slot 9	R/W	0x0000
ipint_hpout5	PCM/IOM port data out - slot 10 and slot 11	R/W	0x0000
ipint_hpin0	PCM/IOM port data in - slot 0 and slot 1	R	0x0000
ipint_hpin1	PCM/IOM port data in - slot 2 and slot 3	R	0x0000
ipint_hpin2	PCM/IOM port data in - slot 4 and slot 5	R	0x0000
ipint_hpin3	PCM/IOM port data in - slot 6 and slot 7	R	0x0000
ipint_hpin4	PCM/IOM port data in - slot 8 and slot 9	R	0x0000
ipint_hpin5	PCM/IOM port data in - slot 10 and slot 11	R	0x0000

[1] Please note that for subsequent write accesses within short time to the same register, restrictions apply as described in Section 9.8.

Table 332: Register ipint_global

Bit	Symbol	Access	Value	Description
15 to 1	-	R	0*	reserved
0	iom_sint_en	R/W	0*	IOM slave interrupt enable. For IOM slave mode i.e. to enable the int_fsi_ip this bit must be set.

Table 333: Register ipint_cntl0

Bit	Symbol	Access	Value	Description
15	reserved			must be set to '0'
14	master	R/W		PCM/IOM master mode ^[1]
			1	IPINT is master on the PCM/IOM port, DCK_IP and FSC_IP are outputs.
13	intmode	R/W		IPINT is slaved to the PCM/IOM port timing, DCK_IP and FSC_IP are inputs.
			0*	select interrupt mode
12	dckp_en			int_fsi_ip is generated only if at least 1 slot is enabled (see Table 334). Also ipint_global.iom_sint_en should be set to '1'.
			1	int_fsi_ip is generated every 8kHz. In master mode or if ipint_global.iom_sint_en = 0 the int_fsi_ip is equal to the int_fsi.
11	loop	R/W		enable DCK_PCM clock output a half the rate of DCK_IP
10	iock	R/W		Loop-back mode
			1	DO_IP is internally connected to DI_IP. Data is not re clocked.
			0*	no loop-back selected.
9 to 8	iod	R/W		Type of FSC_IP and DCK_IP output port.
			1	FSC_IP and DCK_IP use push-pull.
			0	FSC_IP and DCK_IP use open-drain.
7 to 6	fstyp	R/W		Type of PCM/IOM data output port.
			0b11	DO_IP uses push-pull and is always driven
			0b10	DO_IP uses push-pull and is set to tri-state outside transmission
			0b01	DO_IP uses open-drain and is set to tri-state outside transmission
			0b00*	PCM/IOM port output disabled, DO_IP set to tri-state
5 to 3	freq	R/W		Shape of frame synchronization signal.
			0b11	Long frame-sync over the first slot (8 bit)
			0b10	Short frame-sync LF enclosing the last falling clock edge
			0b01	Short frame-sync FF enclosing the first falling clock edge
			0b00*	Short frame-sync FR enclosing the first rising clock edge
2 to 0	-	R		Port frequency selection.
			0b111	4.096 MHz (2 clock cycles per bit), 256 bits clocked per frame.
			0b110	1.536 MHz (2 clock cycles per bit), 96 bits clocked per frame.
			0b101	768kHz (2 clock cycles per bit), 48 bits clocked per frame.
			0b100	512kHz (2 clock cycles per bit), 32 bits clocked per frame.
			0b011	2.048 MHz, 256 bits clocked per frame.
			0b010	1.536 MHz, 192 bits clocked per frame.
			0b001	768kHz, 96 bits clocked per frame.
			0b000*	512kHz, 64 bits clocked per frame of 125 µs.
2 to 0				must be set to '0'

[1] After enabling the IPINT block or when switching from master to slave mode, the first data bit in the first frame might not be transmitted correctly.

Table 334: Register ipint_ctrl1

Bit	Symbol	Access	Value	Description
15 to 12	fsc2_del	R/W		Frame sync signal 2 enabling and delay selection
			0b1111	FSC2_IP frame sync signal has 15 slots delay to FSC_IP signal
		
			0b0001	FSC2_IP frame sync signal has 1 slot delay to FSC_IP signal
			0b0000*	FSC2_IP output disabled, FSC2_IP signal set to '0'
11	enslt11	R/W	0*	Enable PCM/IOM slot 11 (bits 88 to 95 in the frame)
10	enslt10	R/W	0*	Enable PCM/IOM slot 10 (bits 80 to 87 in the frame)
9	enslt9	R/W	0*	Enable PCM/IOM slot 9 (bits 72 to 89 in the frame)
8	enslt8	R/W	0*	Enable PCM/IOM slot 8 (bits 64 to 71 in the frame)
7	enslt7	R/W	0*	Enable PCM/IOM slot 7 (bits 56 to 63 in the frame)
6	enslt6	R/W	0*	Enable PCM/IOM slot 6 (bits 48 to 55 in the frame)
5	enslt5	R/W	0*	Enable PCM/IOM slot 5 (bits 40 to 47 in the frame)
4	enslt4	R/W	0*	Enable PCM/IOM slot 4 (bits 32 to 39 in the frame)
3	enslt3	R/W	0*	Enable PCM/IOM slot 3 (bits 24 to 31 in the frame)
2	enslt2	R/W	0*	Enable PCM/IOM slot 2 (bits 16 to 23 in the frame)
1	enslt1	R/W	0*	Enable PCM/IOM slot 1 (bits 8 to 15 in the frame)
0	enslt0	R/W		Enable PCM/IOM slot 0 (bits 0 to 7 in the frame)
			1	enabled
			0*	disabled

[1] In case of IPINT operation mode with FREQ=0b100 (IOM operation mode with 512kHz clock) and IOD = 0b01 or 0b10 (DO_IP is high-z outside transmission) the slot 3 is active too (even if not enabled) whenever one ore more of the slots 0 to 2 is/are enabled.

Table 335: Register ipint_hpout0

Bit	Symbol	Access	Value	Description
15 to 8	outslot0	R/W	0*	Slot0 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot1	R/W	0*	Slot1 data sent from the IPINT to the PCM/IOM port

Table 336: Register ipint_hpout1

Bit	Symbol	Access	Value	Description
15 to 8	outslot2	R/W	0*	Slot2 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot3	R/W	0*	Slot3 data sent from the IPINT to the PCM/IOM port

Table 337: Register ipint_hpout2

Bit	Symbol	Access	Value	Description
15 to 8	outslot4	R/W	0*	Slot4 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot5	R/W	0*	Slot5 data sent from the IPINT to the PCM/IOM port

Table 338: Register ipint_hpout3

Bit	Symbol	Access	Value	Description
15 to 8	outslot6	R/W	0*	Slot6 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot7	R/W	0*	Slot7 data sent from the IPINT to the PCM/IOM port

Table 339: Register ipint_hpout4

Bit	Symbol	Access	Value	Description
15 to 8	outslot8	R/W	0*	Slot8 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot9	R/W	0*	Slot9 data sent from the IPINT to the PCM/IOM port

Table 340: Register ipint_hpout5

Bit	Symbol	Access	Value	Description
15 to 8	outslot10	R/W	0*	Slot10 data sent from the IPINT to the PCM/IOM port
7 to 0	outslot11	R/W	0*	Slot11 data sent from the IPINT to the PCM/IOM port

Table 341: Register ipint_hpin0

Bit	Symbol	Access	Value	Description
15 to 8	inslot0	R/W	0*	Slot0 data received from the IPINT from the PCM/IOM port, left data 0 for IIS (MSBs)
7 to 0	inslot1	R/W	0*	Slot1 data received from the IPINT from the PCM/IOM port, left data 0 for IIS (LSBs)

Table 342: Register ipint_hpin1

Bit	Symbol	Access	Value	Description
15 to 8	inslot2	R/W	0*	Slot2 data received from the IPINT from the PCM/IOM port, right data 0 for IIS (MSBs)
7 to 0	inslot3	R/W	0*	Slot3 data received from the IPINT from the PCM/IOM port, right data 0 for IIS (LSBs)

Table 343: Register ipint_hpin2

Bit	Symbol	Access	Value	Description
15 to 8	inslot4	R/W	0*	Slot4 data received from the IPINT from the PCM/IOM port, left data 1 for IIS (MSBs)
7 to 0	inslot5	R/W	0*	Slot5 data received from the IPINT from the PCM/IOM port, left data 1 for IIS (LSBs)

Table 344: Register ipint_hpin3

Bit	Symbol	Access	Value	Description
15 to 8	inslot6	R/W	0*	Slot6 data received from the IPINT from the PCM/IOM port, right data 1 for IIS (MSBs)
7 to 0	inslot7	R/W	0*	Slot7 data received from the IPINT from the PCM/IOM port, right data 1 for IIS (LSBs)

Table 345: Register ipint_hpin4

Bit	Symbol	Access	Value	Description
15 to 8	inslot8	R/W	0*	Slot8 data received from the IPINT from the PCM/IOM port, left data 2 for IIS (MSBs)
7 to 0	inslot9	R/W	0*	Slot9 data received from the IPINT from the PCM/IOM port, left data 2 for IIS (LSBs)

Table 346: Register ipint_hpin5

Bit	Symbol	Access	Value	Description
15 to 8	inslot10	R/W	0*	Slot10 data received from the IPINT from the PCM/IOM port, right data 2 for IIS (MSBs)
7 to 0	inslot11	R/W	0*	Slot10 data received from the IPINT from the PCM/IOM port, right data 2 for IIS (LSBs)

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.22.5 Functional description

Figure 123 'IPINT overview' shows the data flow inside the IPINT, which is detailed in this chapter.

The IPINT provides an interrupt, int_fsi_ip, to the ARM processor to allow voice processing. It is generated when all new data are available in the associated registers.

The int_fsi_ip has a delay of 20.83μs. The delay allows for DPLL corrections without impairing the received data integrity.

All transfers are synchronized to **clk_ip**, hence no violation is possible and data integrity is guaranteed.

9.22.5.1 Data flow and timings, PCM/IOM

In the receive direction, the first 96 bits of each IP frame are clocked in by the IPINT interface and shifted into the IPInBuffer (see Figure 126). There they are held until they are copied into the INBuffer with the start of the next IP frame. From there, data are forwarded to the **ipint_hpout** registers. An interrupt, int_fsi_ip, to the Host Processor is generated, where the Host Processor may read and process PCM/IOM data from the **ipint_hpout** register.

In the transmit direction the int_fsi_ip interrupt to the Host Processor is generated allowing it to manually write data into the **ipint_hpout** register. With the next int_fsi_ip pulse, the data in the **ipint_hpout** register are latched into the OutBuffer. From there, they are transferred to the IPOutBuffer shift register with the start of the next IP frame and clocked serially out of the IPINT.

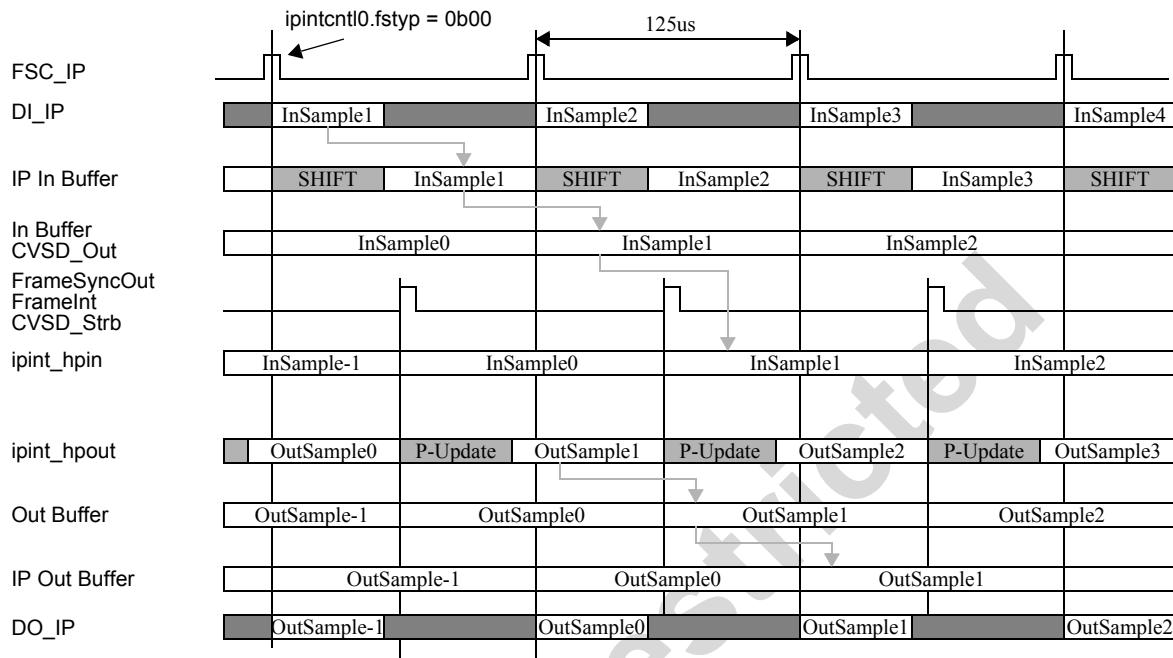


Fig 126. PCM/IOM data flow timing

The IPINT control register configures the PCM/IOM port, including operation, master/slave mode, shape of synchronization signal, port frequency selection, PCM/IOM data length.

PCM/IOM Data formats: There are maximum 12 x 8-bit slots in one PCM/IOM frame of 125 us. Each of these slots in the PCM/IOM frame may be enabled individually. The bits in HPIN registers corresponding to disabled slots are set to '0'.

DO_IP is set to '0' or high impedance during the bit positions in the frame corresponding to disabled slots.

Timing of the IPINT

1. General Description

The IPINT may operate as timing master or slave on the PCM/IOM bus.

When being configured as PCM/IOM slave, it is synchronized to the FSC_IP signal received from the PCM/IOM bus. The signal int_fsi_ip is used to synchronize other blocks on the chip to the PCM/IOM timing.

In the PCM/IOM master configuration, the IPINT is synchronized to an 8kHz frame pulse int_fsi coming from the BMP. This frame sync pulse is handled as an asynchronous input. In master mode, the basic clock timings are determined as follows (parameter namings refer to the timings given in [Figure 124](#) and [Figure 125](#)).

Note that after enabling the IPINT block or when switching from master to slave mode, the first data bit in the first frame might not be transmitted correctly.

Table 347: IPINT master clock characteristics

intcntl0.freq / Port frequency in MHz	T9 and T10 if clk_ip = 48MHz	
	MIN. [ns]	MAX. [ns]
000 / 0.512	958	979
001 / 0.768	646	666
010 / 1.536	312	333
011 / 2.048	229	250
100 / 0.512	958	979
101 / 0.768	646	666
110 / 1.536	312	333
111 / 4.096	104	125

[1] MIX. / MAX. spread is mainly resulting from a maximum cycle-to-cycle jitter due to the implemented clock generation principle. The application has to ensure that a connected slave device can support the resulting worst case lengths.

9.22.5.2 Synchronization

1. Master Mode

In PCM/IOM master mode, the PCM/IOM timing is derived from the 8kHz int_fsi signal of the timing interface. The 8kHz int_fsi_ip is synchronous to this signal.

A DPLL synchronized to the int_fsi signal generates the timing for the PCM/IOM interface itself, i.e. the DCK_IP and FSC_IP signals. The FSC_IP signal aligns with a phase shift of 60 degree with respect to the int_fsi. This ensures that DPLL phase corrections do not impact data integrity (see [Figure 126](#) Data flow timing diagram).

The int_fsi signal received from the BMP core might have a deviation of +/-25ppm with respect to the local clock. Phase corrections in the BMP are performed on a 13.824 MHz clock basis resulting in phase jumps of the int_fsi signal of 72 ns.

2. Slave Mode

In PCM/IOM slave mode, the timing of int_fsi_ip from a DPLL synchronized with a 60 degree phase offset to the FSC_IP signal. To do phase corrections, the DPLL is allowed to add or skip one **clk_ip** cycle per PCM/IOM frame resulting in a lock range of 166 ppm for **clk_ip** = 48MHz.

The int_fsi input signal is disregarded.

3. Interrupt gating

When **ipint_ctrl0.intmode** is set to '1' the int_fsi_ip is generated at 8kHz intervals but only when the block is active and one of the enabled buffers is empty. This behavior is the same in master and in slave mode. When **ipint_ctrl0.intmode** is set to '0' int_fsi_ip is generated at every 8kHz interval independent of the received data.

9.22.5.3 Standby Mode

When the IPINT block is switched off, int_fsi_ip is still present and equals int_fsi.

9.22.6 Application information

The IPINT may work in master mode (i.e. bus timing is provided by IPINT) or slave mode (IPINT is synchronized to the timing received from the bus). Synchronization is also exchanged with the BMP.

The IPINT handles the 96 bits of PCM/IOM frame. Processing 96 bit in a PCM/IOM frame allows for example accessing 12 channels (slots) to 8 bit devices. The availability of IP data is signalled to the host processor by a single interrupt, int_fsi_ip, every PCM/IOM frame. For frames longer than 96 bits all bits are ignored after the 96th bit.

The IPINT supports the physical data formats for PCM (one clock cycle per bit) or IOM (two clock cycles per bit). Any potentially required data structuring (e.g. monitor channel for IOM) has to be performed by the processor.

The connection between the IPINT and the different blocks and channels is handled by the Host Processor by copying data between the IPINT registers and the different registers associated with each block and channel.

There is an option to switch the block into loop-back mode to allow for easier in-circuit testing on board-level.

The bit rates support in IOM/PCM master and slave mode by the IPINT are presented in [Table 348](#).

Table 348:IPINT Bit rates

IOM	PCM
256kbps ^[1]	512kbps
384kbps	768kbps
768kbps	1.536Mbps
2.048Mbps	2.048Mbps

[1] See footnote of [Table 334](#).

9.23 KBS - Keyboard Scanner

9.23.1 Features

The KBS implements all the logic necessary to interface a rectangle matrix keyboard with up to 64 keys (see [Figure 129 on page 406](#)) to the system controller. The KBS is characterized by the following features:

- Up to 64 keys with multiple key detection
- Usable for any rectangle matrix keyboards up to 8x8
 - Number of active column must be less or equal than number of active rows.
- Minimum power consumption, no key scan until at least one key is pressed
- Programmable key debounce times for keyboard type adjustment; debouncing is performed for key pressure as well as for key release
- Power-on key detection capabilities (i.e. key pressed during and after reset is correctly handled)
- Interrupt on key press and key release

9.23.2 Block diagram

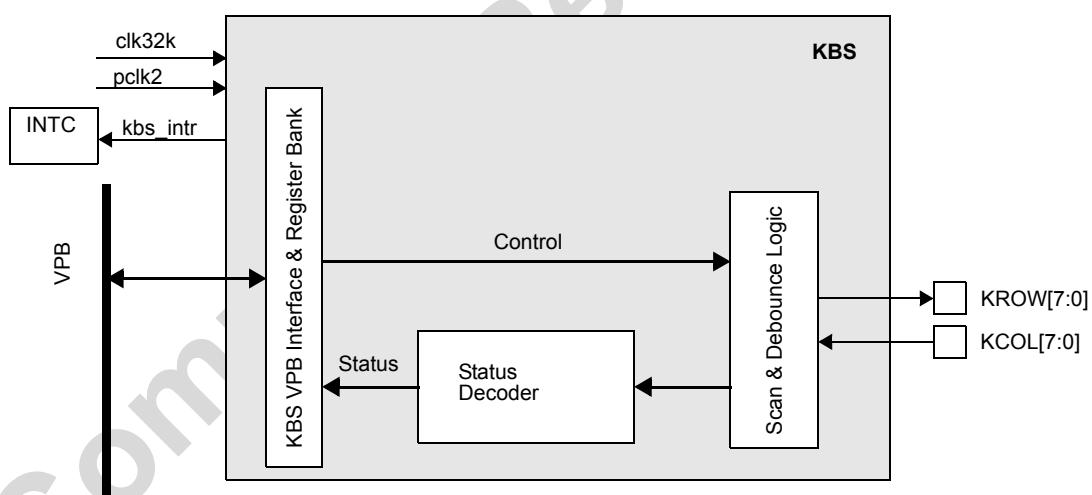


Fig 127.KBS block diagram

9.23.3 Hardware interface

Table 349:KBS pin overview

PIN	Name	I/O	Description
KCOL[7:0]	column	I	column signal, equipped with internal pull-down
KROW[7:0]	row	O	row signal, can drive '1' or be pulled-down (to be enabled by SW)

9.23.4 Software interface

Table 350: Register overview of KBS

Symbol	Name	I/O	Reset
kbs_deb	Keypad debounce counter register	R/W	0x05
kbs_state_cond	Keypad IO control state condition register	R	0x00
kbs_it	Keypad interrupt register	R/W	0x01
kbs_fast_test	Keypad fast_tst register	R/W	0x00
kbs_matrix_dim	Keypad matrix_dim register	R/W	0x06
kbs_data0	Keypad data0 register	R	0x00
kbs_data1	Keypad data1 register	R	0x00
kbs_data2	Keypad data2 register	R	0x00
kbs_data3	Keypad data3 register	R	0x00
kbs_data4	Keypad data4 register	R	0x00
kbs_data5	Keypad data5 register	R	0x00
kbs_data6	Keypad data6 register	R	0x00
kbs_data7	Keypad data7 register	R	0x00

Table 351: Register kbs_deb

Bit	Symbol	Access	Value	Description
7 to 0	kbs_deb[7:0]	R/W	0x05*	Keypad Debounce Counter: debounce duration in full matrix scan periods. A matrix scan takes (kbs_matrix_dim x T_{clk32k}). A key press is recognized by the software if it lasts longer than the value specified in this register. Value 0 must not be used in this register. See Table 357 for details

Table 352: Register kbs_state_cond

Bit	Symbol	Access	Value	Description
7 to 3	-	R	0*	reserved
2 to 0	key_state[2:0]	R	0*	Keypad IO Control State Condition Register: current state of internal state machine. Used for test purpose only.

Table 353: Register kbs_it

Bit	Symbol	Access	Value	Description
7 to 1	-	R	0*	reserved
0	irqn	R/W ^[1]		Keypad Interrupt Register: state of the interrupt signal. The interrupts follow a standard handshake protocol. When a key is pressed or released the interrupt signal is set active, then the software responds by an acknowledge (by writing to this register), thus releasing the interrupt.
			0	interrupt is pending
			1*	no interrupt is pending

[1] Writing any value in this register will clear the interrupt (irqn=1). Even in polling mode this bit must be acknowledged after a keypress.

Table 354: Register kbs_fast_test

Bit	Symbol	Access	Value	Description
7 to 2	-	R	0*	reserved
1	kbs_fast_test1	R/W	0*	Use pclk2 instead of clk32k. Reserved for test purposes
0	kbs_fast_test0	R/W	0*	Force scan_once state. Reserved for test purposes

Table 355: Register kbs_matrix_dim

Bit	Symbol	Access	Value	Description
7 to 4	-	R	0*	reserved
3 to 0	kbs_matrix_dim[3:0]	R/W	6*	Keypad Matrix dimension Register: number of active rows (and maximum number of column). Values below 1 or above 8 are illegal. Number of used column must be less or equal than number of active rows.

Table 356: Register kbs_data0 to 7^[1]

Bit	Symbol	Access	Value	Description
7	kbs_key_r[7]	R	0*	Keypad DATA Register: keypress data stored after programmable de-bouncing period.
			0*	Column 7 not pressed in row n (n from 0 to 7)
			1	Column 7 pressed in row n (n from 0 to 7)
6	kbs_key_r[6]	R	0*	same meaning than kbs_key_r[7] , but for column 6
5	kbs_key_r[5]	R	0*	same meaning than kbs_key_r[7] , but for column 5
4	kbs_key_r[4]	R	0*	same meaning than kbs_key_r[7] , but for column 4
3	kbs_key_r[3]	R	0*	same meaning than kbs_key_r[7] , but for column 3
2	kbs_key_r[2]	R	0*	same meaning than kbs_key_r[7] , but for column 2
1	kbs_key_r[1]	R	0*	same meaning than kbs_key_r[7] , but for column 1
0	kbs_key_r[0]	R	0*	same meaning than kbs_key_r[7] , but for column 0

[1] This register is an image of KCOL[7:0] of the keypad captured during row scanning, **kbs_data0** for ROW0, **kbs_data1** for ROW1, etc...

9.23.5 Functional description

The KBS delivers a number of registers (8 in max configuration) which represents the active row and column position of the pressed key within the rectangle switch matrix; the key code is available as soon as a key pressure is judged valid by the debounce logic; simultaneously the interrupt pending flag is set and the KBS interrupt request is asserted

The status flag and KBS interrupt request is also asserted when a key release is detected; the current state is indicated by the key code in the KBS status register; by this means the SC knows whether a key is still pressed

The KBS is able to detect multiple keys pressed or released at the same time. The principle of the button detection used in this block is the matrix shown below, in which all rows are individually scanned.

For this purpose each time an event on one key is detected software need to know which column and which row for each key.

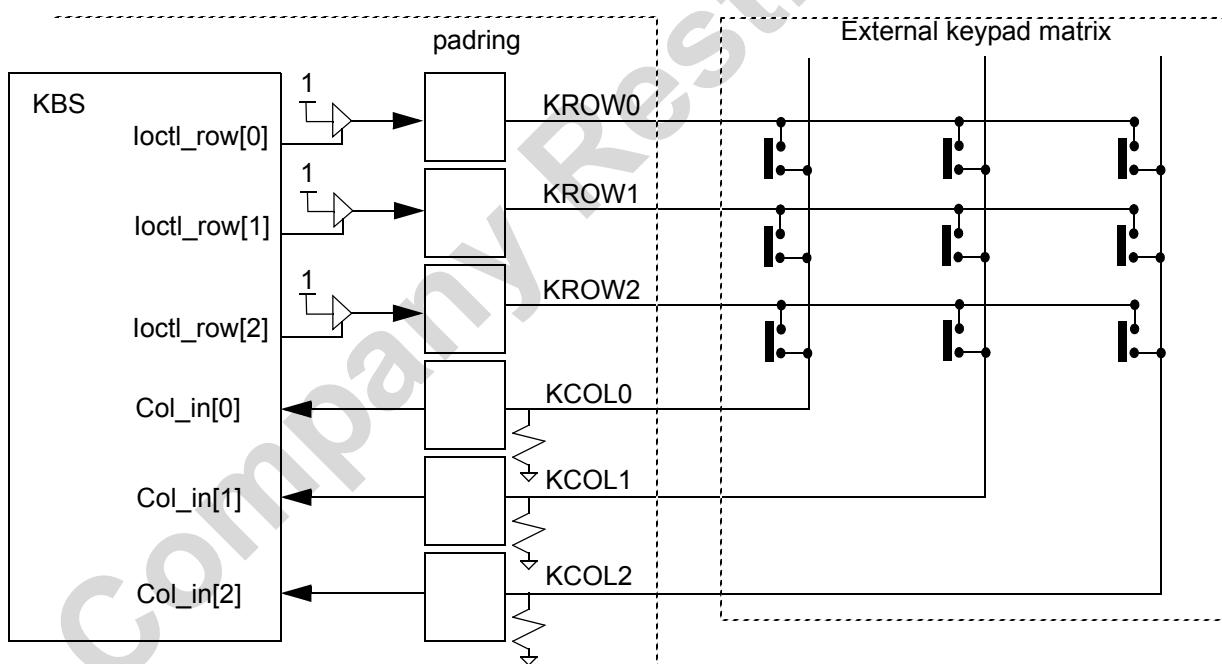


Fig 128.External connection (example has a simple 3x3 keypad matrix)

When a key-press is recognized, data is stored after a programmable number of clock periods, and an interrupt is generated. The key press is therefore recognized if it lasts longer than value specified in the **kbs_deb**.

If the interrupt was due to a key press/release, then the column and row register is read to determine which key was press/release. Writing clears the interrupt to the Keypad Interrupt register.

At reset rows are all driven allowing any keypress to be detected on the column pins. Once this is the case, a scan is performed by driving the row pins, one at a time, in sequence, de-bounced and corresponding keypad status bits are updated according

to any key which may have been pressed. If one or more keys are pressed at reset, no key interrupt is generated, however when a key activity occur after reset (key release or another key press), an interrupt is generated.

It is possible to optimize the function to the actual matrix size using register **kbs_matrix_dim**, which, once set in an application should not be changed.

Latency between key(s) activity and interrupt depend on **kbs_deb** as well of **kbs_matrix_dim**.

Table 357:KBS latency (from key(s) activity to interrupt)

example values with: kbs_deb = 5; kbs_matrix_dim = 6

min	max	unit
first key(s) press after KBS in idle state, assumed no bounce on key(s)		
3 + kbs_deb × kbs_matrix_dim	4 + kbs_deb × kbs_matrix_dim	clk32k
example: 1.007	example: 1.038	ms
other key(s) activity, including last key(s) release		
1 + kbs_deb × kbs_matrix_dim	1 + (kbs_deb + 1) × kbs_matrix_dim	clk32k
example: 0.946	example: 1.129	ms

Once a new key (matrix) status is detected, it can be read from **kbs_data0** to **kbs_data7**.

The remaining two registers are not used in a typical application. Register **kbs_state_cond** allows visibility on the internal state of the state machine. Register **kbs_fast_test** facilitates fast simulation of the performance by running the block at the VPB frequency and not 32kHz and also allowing forcing the state machine.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.23.6 Application information

Figure 129 shows the wiring of a rectangle matrix keyboard with 36 keys. The pull downs of the keyboard column and row pads have to be configured by software in register **syspad**. Due to during the scanning mode only one row is driven at a time all used rows should have the internal pull downs enabled for safe operation.

For different keypad size, **kbs_matrix_dim** must be used to adapt the matrix. Unused column which are not selected as pins through GPIO muxing is forced to 0 by the muxing logic.

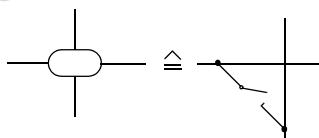
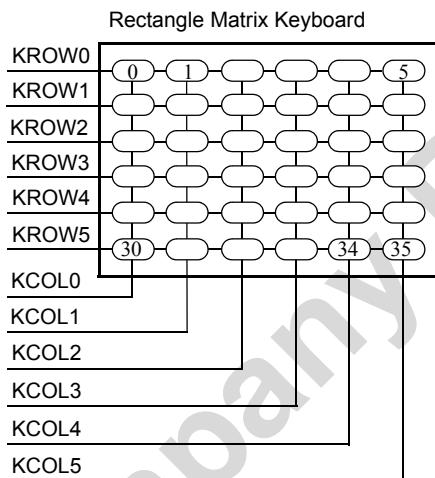
key0 status is stored in **kbs_data[0].kbs_key_r[0]**

key1 status is stored in **kbs_data[0].kbs_key_r[1]**

...

key34 status is stored in **kbs_data[5].kbs_key_r[4]**

key35 status is stored in **kbs_data[5].kbs_key_r[5]**



Connection of the matrix rows and columns via switch

Fig 129.Connecting a rectangle matrix keyboard with 36 keys

While the keypad scanner can always detect two simultaneous key presses, it is not always possible to uniquely identify three or more keys, depending on the configuration. An example is shown below.

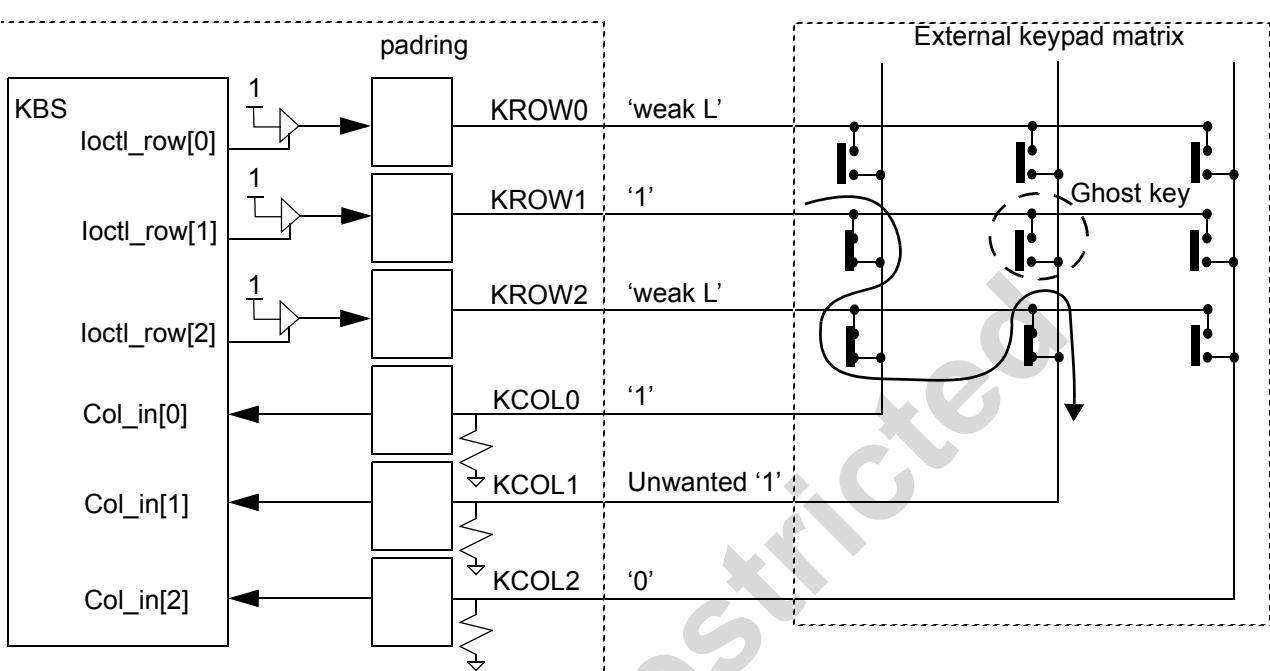


Fig 130.External connection showing an illegal triple key combination (if simple switches are used)

In the above example, KBS is trying to detect keypresses on KROW1. As such, KROW1 is driven high. KBS sees KCOL1 as high, indicating that the center switch connected to KCOL1 is pressed (which is not the case). In general this means that keypad matrix wiring must be done carefully.

A solution which resolves this perfectly (but adds external components thus cost) is simply to place a diode in series with each switch contact so that a logic 1 cannot come up the switch but only go down.

Without using diodes it is necessary to limit legal multiple key combinations to those where all keys are in unique rows (or equally in unique columns).

9.23.6.1 KBS interrupt handling

Keypad interrupts are handled as follows. The SC

- Tests registers **kbs_data0** to **kbs_data7** to determine which key has been pressed (or if all keys have been released)
- Writes to the **kbs_it** register to set IRQ inactive, thus clearing the keypad interrupt source.

9.23.6.2 Big KBS matrix

The KCOL6/7 and KROW6/7 key board scanner pins can be used only if the TAPMUX control defined by the JSEL[1:0] pins is 0b11 (mode3 with all TAPCx in a chain).

9.24 PWM2/PWM3 - Pulse Width Modulators

9.24.1 Features

- Pulse width modulator mode
 - Frequency from 199 Hz to 6.5 MHz (for $\text{clk13mm} = 13\text{MHz}$)
 - Duty-cycle from near 0% to almost 100%
- Alerter mode
 - Alerter frequency from 3 Hz to 101 kHz (for $\text{clk13mm} = 13\text{MHz}$)
 - Adjustable volume by duty-cycle trimming
- Pulse density Modulation mode
- All outputs can be combined with one or two other outputs

9.24.2 Block diagram

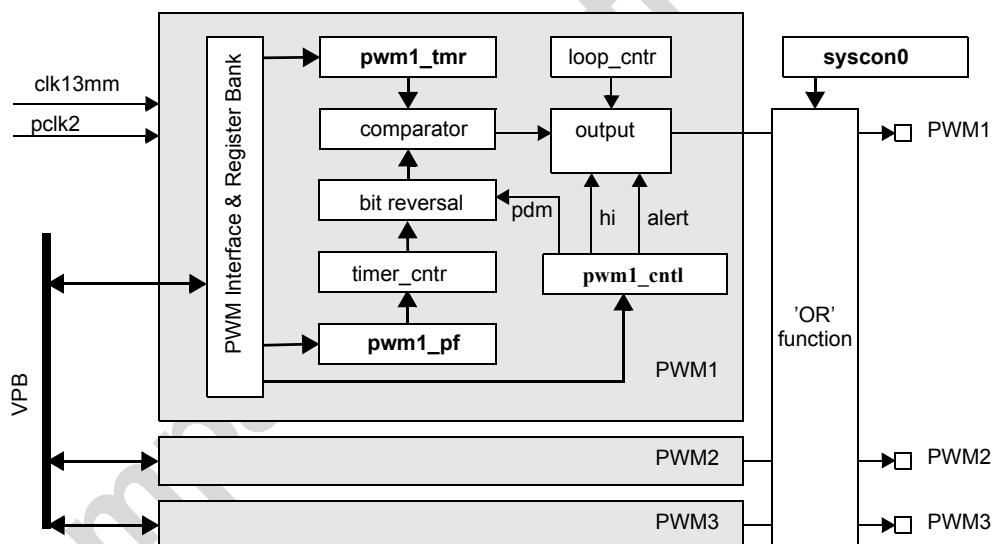


Fig 131.PWM block diagram

Note: PWM1 pin is not available at DVFD818x

9.24.3 Hardware interface

Table 358: PWM pin overview

PIN	Name	I/O	Description
PWM2	PWM 2 out	O	PWM 2 output
PWM3	PWM 3 out	O	PWM 3 output

9.24.4 Software interface

Table 359: Register overview of PWM2/PWM3

Symbol	Name	I/O	Reset
pwm_pf	PWM pulse frequency register	R/W	0x0000
pwm_tmr	PWM timer value register	R/W	0x0000
pwm_cntl	PWM control register	R/W	0x0000

Table 360: Register pwm2_pf/pwm3_pf

Bit	Symbol	Access	Value	Description
15 to 0	pf[15:0]	R/W	0*	period of the PWM output - must be in range 1 to 65535 - the reset value pf = 0 is overridden by tmr = 0 which forces the PWM output to 0

Table 361: Register pwm2_tmr / pwm3_tmr

Bit	Symbol	Access	Value	Description
15 to 0	tmr[15:0]	R/W	0*	high time of the PWM output - must be in range 0 to 65535 special values: 0 forces PWM output to 0 or tmr > pf forces PWM output to 1

Table 362: Register pwm2_cntl/pwm3_cntl

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2	pdm	R/W	0*	PWM mode
			1	PDM mode
1	alert	R/W	0*	PWM mode
			1	alerter mode, output generates alternatively 32 normal pulses and 32 inverted pulses
0	hi	R/W	0*	PWM is running
			1	PWM output is forced to 1

9.24.5 Functional description

The counter timer_cntr and loop_cntr are reset any time **pwm_pf** or **pwm_ctl** are written.

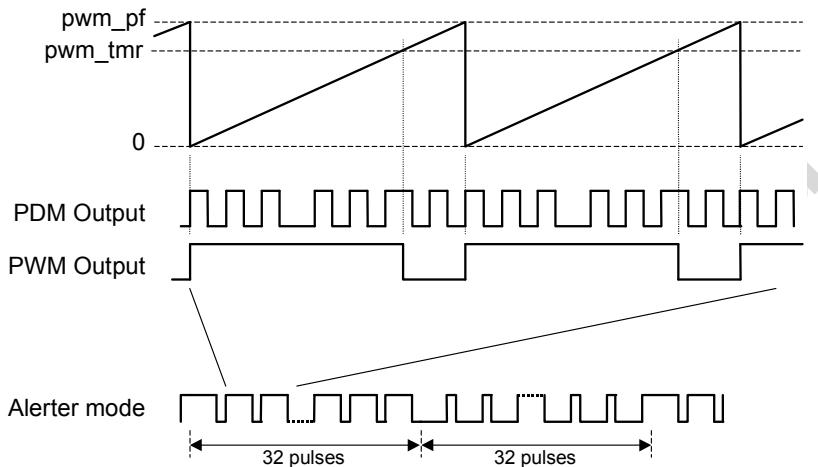


Fig 132.PWM/PDM/alster principle

9.24.5.1 PWM mode

The counter timer_cntr increments at $f_{clk13mm}$ until the value of **pwm_pf** is reached. The duty cycle of the PWM output is defined by **pwm_tmr**.

9.24.5.2 PDM mode

The counter timer_cntr increments at $f_{clk13mm}$ until the value of **pwm_pf** is reached. The duty cycle of the PWM output is defined by $\text{bit_reverse}(\text{timer_cntr}) < \text{pwm_tmr}$. $\text{bit_reverse}(x)$ is a bit swapping function, bit 15 → bit 0, bit 14 → bit 1, and so on. This results in the same output average voltage as in the PWM mode, but the pulses are spread over time.

9.24.5.3 Alerter mode

Same as PWM mode, but output signal is alternatively normal or inverted every 32 output pulse. This is used to generate audible tone with volume control. Because carrier frequency is 64 times above needed alerter frequency, external filtering is easy to do. The volume is minimal when the difference of average amplitude (low pass filtering) between the two 32 pulses burst are minimum (both duty cycle equal → duty cycle set to 50%). The volume is maximal when the difference of average amplitude (low pass filtering) between the two 32 pulses burst are maximum (both duty cycle are at the maximum/minimum value → duty cycle close to 0 or 100%).

9.24.6 Application information

The frequency calculations are based on a $f_{clk13mm}$ of 13MHz (see also [Table 46](#)).

9.24.6.1 PWM mode

PWM frequency = $\frac{f_{clk13mm}}{pwm_pf+1}$ with $1 \leq pwm_pf \leq 65\,535$. From 199 Hz to 6.5 MHz

Duty cycle = $\frac{pwm_tmr}{pwm_pf+1}$ with $1 \leq pwm_tmr \leq pwm_pf$. From near 0 to almost 100%

PWM example for a frequency of 1 kHz, duty cycle 30%:

- $pwm_pf = 12999$
- $pwm_tmr = 3900$
- $pwm_cntl = 0x0000$

9.24.6.2 PDM mode

PWM frequency = $\frac{f_{clk13mm}}{pwm_pf+1}$ with $1 \leq pwm_pf \leq 65\,535$. From 199 Hz to 6.5 MHz

Duty cycle = $bit_reverse(timer_cntr) < pwm_tmr$ with
 $bit_reverse(MSB\ of\ pwm_pf) \leq pwm_tmr \leq 65536 - 2 * bit_reverse(MSB\ of\ pwm_pf)$.
From near 0 to almost 100%

PDM example for a frequency of 1 kHz, overall duty cycle 30%:

- $pwm_pf = 12999$
- $pwm_tmr = 15600$ (is the $bit_reverse$ of $0.3 * 13000 = 3900$)
The pmw_tmr should be in the range of 4 to 65528.
The minimum value can be calculated by $bit_reverse$ of the MSB of the pwm_pf of 12999 = 4.
The maximum value can be calculated by $65536 - 2 * 4 = 65528$.
- $pwm_cntl = 0x0004$

9.24.6.3 Alerter mode

Alerter frequency = $\frac{f_{clk13mm}}{64 \times (pwm_pf+1)}$. From 3 Hz to 101 kHz

Alerter volume = $\frac{pwm_tmr}{pwm_pf+1}$ with $0 \leq pwm_tmr \leq pwm_pf$.

Volume = 0 or volume = 1 give maximum output level. Volume = 0.5 is the minimum output level.

Alerter example for a frequency of 1 kHz:

- $pwm_pf = 202$ (resulting PWM carrier frequency is 64 kHz)
- $pwm_cntl = 0x0002$

Volume is maximum when $pwm_tmr = 1$ or 201 (carrier duty cycle respectively 0.5 or 99.5%). Volume is minimum when $pwm_tmr = 101$ (duty cycle around 50%).

9.24.6.4 Operation during sleep mode

The PWM can be enabled in sleep mode by setting the bit **cgusleepsc.pwmslen**.
The SW has to adjust the configuration of the PWM to the **clk32k** input clock frequency instead of **clk13mm**.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.25 SCTU1/SCTU2 - SC Timer Units

9.25.1 Features

- Two 16-bit timers with four compare channels
- Each timer contains a time base consisting of a 16-bit counter and a 8-bit pre-scaler clocked by clk13mm
- Each of the four channels contains a compare function which can be enabled/disabled
- Each channel can generate an interrupt if enabled

9.25.2 Block diagram

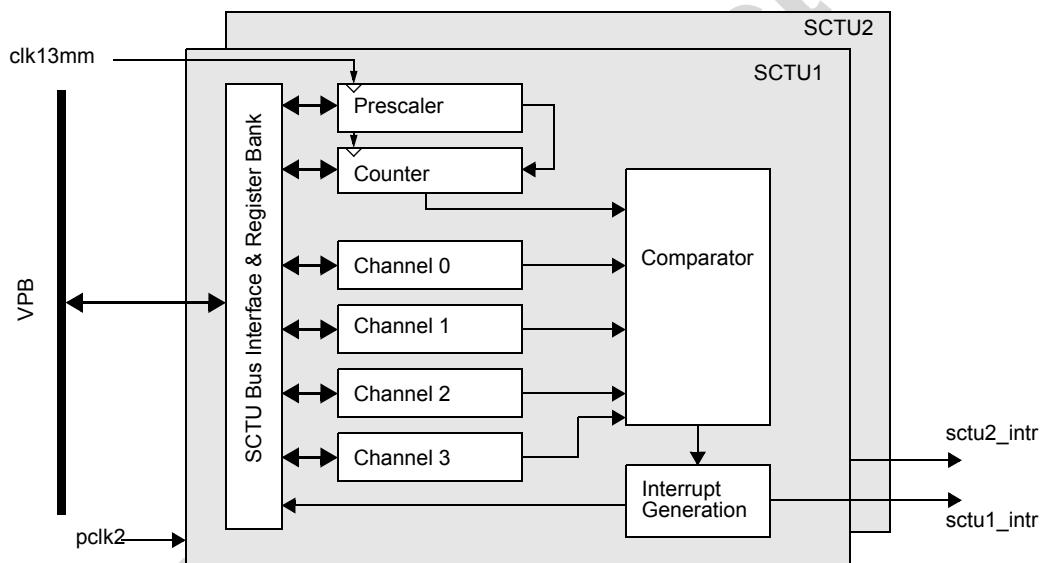


Fig 133. SCTU block diagram

9.25.3 Software interface

Table 363: Register overview of SCTU

Symbol	Name	I/O	Reset
tim1cr/tim2cr	timer control register (tim1cr for SCTU1/tim2cr for SCTU2)	R/W	0x0000
tim1sr/tim2sr	timer status register	R/C	0x00
tim1rr/tim2rr	timer reload register	R/W	0x0000
tim1wr/tim2wr	timer counter work register	R	0x0000
tim1pr/tim2pr	timer pre-scaler reload register	R/W	0x00
tim1c0/tim2c0	timer channel 0 register	R/W	0x0000
tim1c1/tim2c1	timer channel 1 register	R/W	0x0000
tim1c2/tim2c2	timer channel 2 register	R/W	0x0000
tim1c3/tim2c3	timer channel 3 register	R/W	0x0000

Table 364: Register tim1cr/tim2cr

Bit	Symbol	Access	Value	Description	
15 to 10	-	R	0*	reserved	1 2 3 4 5 6 7
9	ei3	R/W	0*	channel interrupt enable	8 9 10 11
			1	disabled	12 13 14
			1	enabled	15 16 17 18
8	cm3	R/W	0*	Channel mode selection	19 20 21
			1	disabled	22 23 24
			1	enabled	25 26 27
7	ei2	R/W	0*	channel interrupt enable	28 29 30
			1	disabled	31 32 33
			1	enabled	34 35 36
6	cm2	R/W	0*	Channel mode selection	37 38 39
			1	disabled	40 41 42
			1	enabled	43 44 45
5	ei1	R/W	0*	channel interrupt enable	46 47 48
			1	disabled	49 50 51
			1	enabled	52 53 54
4	cm1	R/W	0*	Channel mode selection	55 56 57
			1	disabled	58 59 60
			1	enabled	61 62 63
3	ei0	R/W	0*	channel interrupt enable	64 65 66
			1	disabled	67 68 69
			1	enabled	70 71 72
2	cm0	R/W	0*	Channel mode selection	73 74 75
			1	disabled	76 77 78
			1	enabled	79 80 81
1	etov	R/W	0*	timer overflow interrupt enable	82 83 84
			1	disabled	85 86 87
			1	enabled	88 89 90
0	run	R/W	0*	timer enable	91 92 93
			1	timer pre-scaler stopped and register value held	94 95 96
			1	pre-scaler and counter are loaded and pre-scaler is incremented	97 98 99

Table 365: Register tim1sr/tim2sr

Bit	Symbol	Access	Value	Description
7 to 5	-	R	0*	reserved
4	c3	R/C	0*	channel flag
			0*	no event
			1	channel match occurred
3	c2	R/C	0*	channel flag
			0*	no event
			1	channel match occurred
2	c1	R/C	0*	channel flag
			0*	no event
			1	channel match occurred
1	c0	R/C	0*	channel flag
			0*	no event
			1	channel match occurred
0	tov	R/C	0*	timer overflow flag
			0*	no overflow
			1	overflow occurred

Table 366: Register tim1rr/tim2rr

Bit	Symbol	Access	Value	Description
15 to 0	timrr[15:0]	R/W	0*	SCTU reload register

Table 367: Register tim1wr/tim2wr

Bit	Symbol	Access	Value	Description
15 to 0	timwr[15:0]	R	0*	SCTU counter work register

Table 368: Register tim1pr/tim2pr

Bit	Symbol	Access	Value	Description
7 to 0	timpr[7:0]	R/W	0*	SCTU pre-scaler reload register

Table 369: Register tim1c0/tim2c0

Bit	Symbol	Access	Value	Description
15 to 0	timc0[15:0]	R/W	0*	SCTU channel 0 register

The register layout of **tim1c1**, **tim1c2**, **tim1c3** and **tim2c1**, **tim2c2** and **tim2c3** are identical to **tim1c0/tim2c0**.

9.25.4 Functional description

The SCTU is frozen while the SC is in Debug mode. This allows to maintain system timer (like OS ticks) not affected by breakpoints.

9.25.4.1 Time base

The time base contains an 8-bit pre-scaler and a 16-bit counter. The pre-scaler is clocked by a fixed clock **clk13mm** (see Table 46). It is incremented at each clock cycle. On pre-scaler overflow the pre-scaler reload value **timpr** is loaded into the pre-scaler. The 16-bit counter register is incremented at each pre-scaler overflow. If an overflow of the counter occurs, the **tov** flag in the status register **timsr** is set and the value of the counter reload register **timrr** is loaded into the counter register. In addition the SCTU1/2 timer interrupt is generated if the **etov** enable bit in the control register **timcr** is set.

The **tov** bit can be cleared by writing a zero to its bit location in the **timsr**.

The timer does not stop after an overflow or any other event. Clearing **run** bit must be used to stop the timer.

9.25.4.2 Timer activation and deactivation

The timer is activated/deactivated in setting/resetting the **run** bit in the control register **timcr**. If the timer is activated, the pre-scaler and counter are reloaded. The pre-scaler starts to increment with the next clock cycle.

9.25.4.3 Channel function

Each channel consists of a register (**timc0**, **timc1**, **timc2** and **timc3**) and an equality comparator. For each of the four channels a compare function can be enabled and disabled in the control register **timcr**. When the value of the counter register is equal to the value loaded in the channel register, the (**c0**, **c1**, **c2** and **c3**) flag in the status register **timsr** is set. If the (**ei0**, **ei1**, **ei2** and **ei3**) enable bit in the control register **timcr** is set, the SCTU1/2 timer interrupt is generated. The channel status bits (**c0**, **c1**, **c2** and **c3**) can be cleared by writing a zero to the corresponding bit in the **timsr**.

Note: when a comparator match occurred, the counter continues counting and is not reloaded.

9.25.5 Application information

The SCTU can be enabled in sleep mode by setting the bit **cgusleepsc.sctuslen**. The SW has to adjust the configuration of the SCTU to the **clk32k** input clock frequency instead of **clk13mm**.

9.26 SDI - SDRAM Interface**9.26.1 Features**

- High performance SDRAM controller with 3 AHB data ports 1
- SDRAM clock equal to AHB bus clock (up to 104 MHz) 2
- Supports 64-, 128-, 256-, 512-Mbit, single data rate standard SDRAM (3.3V signalling) and Mobile-SDRAM (1.8 V signalling) 3
- 8 or 16-bit off-chip data bus 4
- Programmable address mapping scheme 5
- Programmable SDRAM timing arcs 6
- Programmable SDRAM burst size of 4 or 8 data 7
- Programmable power management features (mobile SDRAM features supported) 8
- Programmable refresh engine 9
- Programmable arbitration between the 3 data ports 10
- round robin with priority 11
- programmable CPU preemption 12
- programmable transaction age priority increment 13
- transaction statistics priority increment 14
- Data coherence checking performed at 1kbyte address boundaries 15

9.26.2 Block diagram

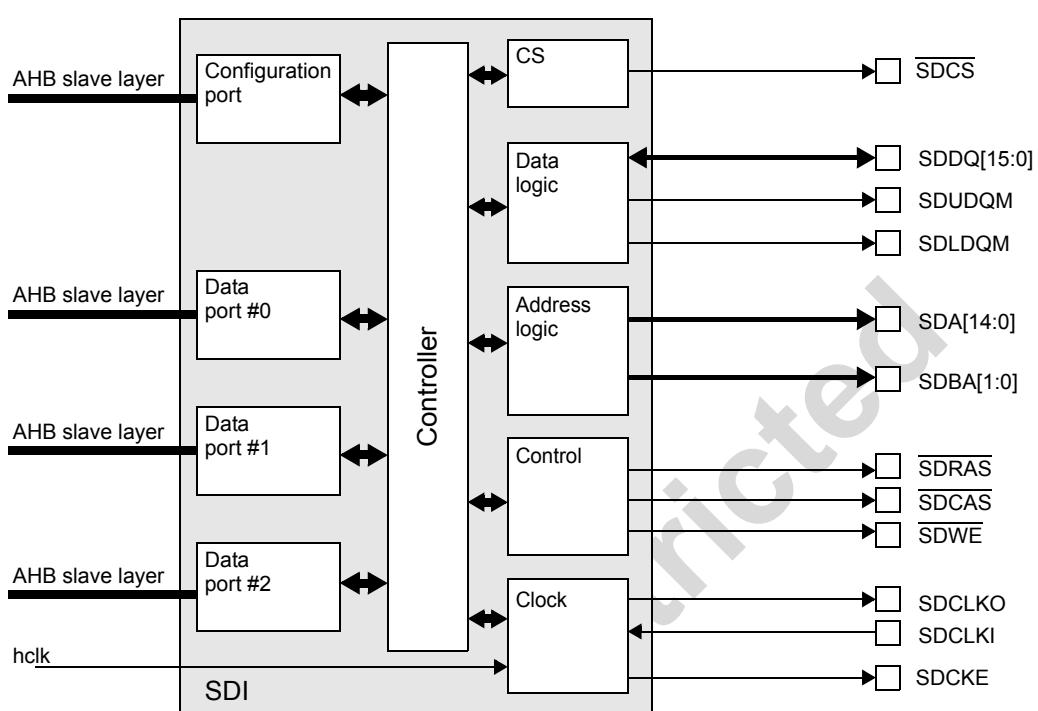


Fig 134.SDI block diagram

Note: The DVFD818x does support SDA[12:0] only

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.26.3 Hardware interface

Table 370: SDI pin overview

Pin	Name	I/O	Description
SDCS	Chip select	O	SDRAM chip select
SDA[12:0]	Address bus	O	
SDBA[1:0]	Bank select	O	
SDRAS	RAS	O	Row address strobe
SDCAS	CAS	O	Column address strobe
SDWE	Write enable	O	
SDDQ[15:0]	Data bus	I/O	
SDUDQM	Data mask enable	O	Upper byte data enable
SDLDQM	Data mask enable	O	Lower byte data enable
SDCLKO ^[1]	Clock output	O	Clock output to be connected to the SDRAM
SDCLKI ^[1]	Clock input	I	Clock input compensation coming from the SDRAM clock input
SDCKE	CKE	O	Clock enable

[1] SDCLKO and SDCLKI must be tied together close to the SDRAM device in order to compensate PCB delays.

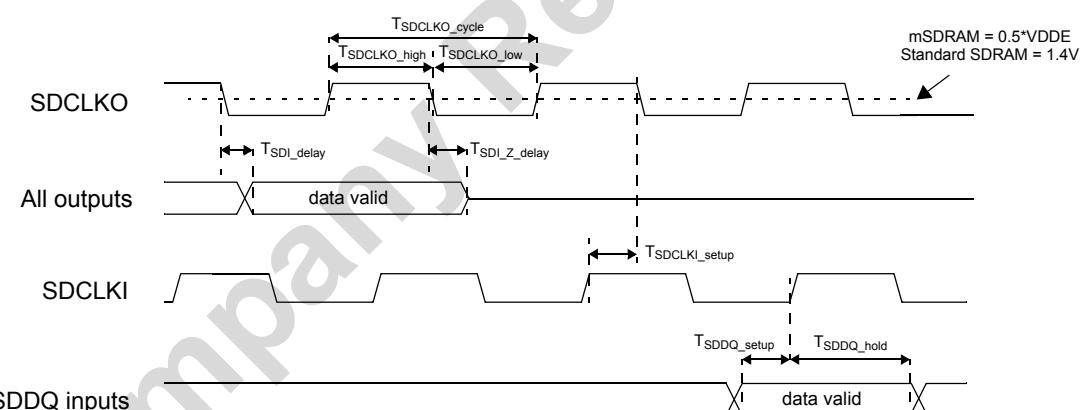


Fig 135.Timing diagrams for SDI (sdi_cfg2.capture_nedge = 0)

Table 371:AC characteristics of the SDI (sdi_cfg2.capture_nedge = 0)

Name	Parameter	Min	Typ	Max	Unit
T_{SDCLKO_cycle}	SDCLKO clock period (=hclk)	9.6	T_{hclk}	77	ns
T_{SDCLKO_high}	SDCLKO high period ^[2]	45	-	55	%
	SDCLKO high period ^[3]			43	
	SDCLKO high period ^[4]			45	
T_{SDCLKO_low}	SDCLKO low period	45	-	55	%
T_{SDI_delay}	Output valid after SDCLKO falling edge	0	-	2	ns
$T_{SDI_Z_delay}$	Output Hi-Z after SDCLKO falling edge	-	-	2	ns
T_{SDCLKI_setup}	SDCLKI rising edge setup time before SDCLKO falling edge	2	-	-	ns
T_{SDDQ_setup}	SDDQ setup time before SDCLKI rising edge	2	-	-	ns
T_{SDDQ_hold}	SDDQ hold time after SDCLKI rising edge	1	-	-	ns

- [1] The timings in this table are guaranteed by design, correlation, and structural testing. They are not specifically measured in production. 1
 [2] Conditions are hclk=armclk/2, hclk ≤ 70MHz, VDDH=3.2V, **sysesh.esh_hmp=1** 2
 [3] Conditions are hclk=armclk/2, hclk = 100MHz, VDDH=3.2V, **sysesh.esh_hmp=1** 3
 [4] Conditions are hclk=armclk/2, hclk = 100MHz, VDDH=3.2V, **sysesh.esh_hmp=0**. This operating point is not guaranteed. 4

9.26.4 Software interface

Table 372: Registers overview of SDI

Symbol	Name	I/O	Reset
sdi_ip_id	IP identification	R	0x2022 0500
sdi_cfg1	SDRAM interface configuration 1	R/W	0x0000 0000
sdi_cfg2	SDRAM interface configuration 2	R/W	0x0000 0000
sdi_pwr_ctrl	Power savings parameters	R/W	0x0000 0000
sdi_rfrsh1	Refresh parameters 1	R/W	0x00FF 0040
sdi_rfrsh2	Refresh parameters 2	R/W	0x0000 0000
sdi_tim1	Timings parameters 1	R/W	0xFFFF FFFF
sdi_tim2	Timings parameters 2	R/W	0xFFFF FFFF
sdi_mode	SDRAM mode register	R/W	0x0000 0022
sdi_ext_mode	SDRAM extended mode register	R/W	0x0000 0000
sdi_sched0	Data port #0 scheduler	R/W	0x0000 0000
sdi_sched1	Data port #1 scheduler	R/W	0x0000 0000
sdi_sched2	Data port #2 scheduler	R/W	0x0000 0000
sdi_wtag0	Data port #0 write tag enable	R/W	0x0000 0007
sdi_wtag1	Data port #1 write tag enable	R/W	0x0000 0007
sdi_wtag2	Data port #2 write tag enable	R/W	0x0000 0007
sdi_tam	Tag address mask	R/W	0x0000 0000
sdi_sw_ctrl	Software control	R/W	0x0000 0006
sdi_stat	Status register	R	0x0011 001F

Table 373: Register sdi_ip_id

Bit	Symbol	Access	Value	Description
31 to 16	module_id	R	0x2022	SDI module identification number
15 to 8	version	R	0x05	Version number
7 to 0	revision	R	0x00	Revision number

Table 374: Register sdi_cfg1

Bit	Symbol	Access	Value	Description
31 to 25	-	R	0*	reserved
24	pc_sdram	R/W	0*	PC SDRAM compliant device: indicates whether or not the device is a PC type device (in which case, some extra functionality are removed from the JEDEC specification).
			0*	not PC SDRAM compliant (burst terminates with BURST TERMINATE)
			1	PC SDRAM compliant (burst terminate with PRECHARGE)
23 to 21	-	R	0*	reserved

Table 374: Register sdi_cfg1...continued

Bit	Symbol	Access	Value	Description
20	ranked	R/W		Rank configuration: indicates whether or not devices are ranked (2 devices in parallel to increase memory size)
			0*	not ranked
			1	ranked - not supported in DVFD818x Family
19 to 18	-	R	0*	reserved
17 to 16	extract[1:0]	R/W		Address extraction type (see page 431)
			0b00*	{ rank, bank[1], bank[0], row, col } - lowest power
			0b01	{ rank, bank[1], row, bank[0], col }
			0b10	{ rank, row, bank[1], bank[0], col } - best performances
			0b11	{ row, rank, bank[1], bank[0], col } - not supported in DVFD818x Family
15 to 14	-	R	0*	reserved
13 to 12	bank[1:0]	R/W		Bank address pins: indicates the number of bank address pins on the SDRAM
			0b00*	0 pin corresponding to 1 bank
			0b01	1 pin corresponding to 2 banks
			0b10	2 pins corresponding to 4 banks
			0b11	reserved
11	-	R	0*	reserved
10 to 8	row[2:0]	R/W		Row address width
			0b000*	9 bits
			0b001	10 bits
			0b010	11 bits
			0b011	12 bits
			0b100	13 bits
			0b101	14 bits - not supported in DVFD818x Family
			0b110	15 bits - not supported in DVFD818x Family
			0b111	16 bits - not supported in DVFD818x Family
7 to 6	-	R	0*	reserved
5 to 4	col[1:0]	R/W		Column address width
			0b00*	8 bits
			0b01	9 bits
			0b10	10 bits
			0b11	11 bits
3 to 2	-	R	0*	reserved
1 to 0	width[1:0]	R/W		SDRAM data bus width
			0b00*	8 bits
			0b01	16 bits
			0b10	32 bits - not supported in DVFD818x Family
			0b11	64 bits - not supported in DVFD818x Family

Table 375: Register sdi_cfg2

Bit	Symbol	Access	Value	Description
31 to 10	-	R	0*	reserved
9	capture_nededge	R/W	0*	SDRAM read capture edge: Controls read data capture at the rising or falling edge of SDCLKO. The propagation delays associated with going off-chip and the clock frequency determine which setting to choose. See Section 9.26.6.8 "Capture setting" .
			1	data internally captured at the falling edge of SDCLKO
			1	data captured at the rising edge of SDCLKO
8	capture	R/W	0*	SDRAM read capture cycle: indicates the number of additional cycles requires for successful read data capture. The propagation delays associated with going off-chip and the clock frequency determine in which cycle data should be captured. See 9.26.6
			1	no additional cycle external delay (lower frequency devices)
			1	1 additional cycle external delay (higher frequency devices)
7 to 3	-	R	0*	reserved
2	nspec_rd2	R/W	0*	speculative reading disable for port #2: indicates whether or not the SDI do speculative reading. See page 442
			0*	speculative reading enabled
			1	speculative reading disabled
1	nspec_rd1	R/W	0*	speculative reading disable for port #1: indicates whether or not the SDI do speculative reading. See page 442
			0*	speculative reading enabled
			1	speculative reading disabled
0	nspec_rd0	R/W	0*	speculative reading disable for port #0: indicates whether or not the SDI do speculative reading. See page 442
			0*	speculative reading enabled
			1	speculative reading disabled

Table 376: Register sdi_pwr_ctrl

Bit	Symbol	Access	Value	Description
31 to 25	-	R	0*	reserved
24	clkout_en	R/W	0*	clock enable control
			0*	SDCLKO forced to 0
			1	SDCLKO enabled
23 to 20	-	R	0*	reserved
19 to 16	pwrx[3:0]	R/W	0*	Idle cycles before entering SELF REFRESH: indicates how many SDRAM clock cycles (SDCLKO) should pass while the SDI is idle before putting the SDRAM in self refresh mode. If pwrx is greater than 0, then self refresh mode is entered after (pwrx + pwry SDCLKO) clock cycles.
			0*	disabled
			m	m * 16 clock cycles
15 to 12	-	R	0*	reserved

Table 376: Register sdi_pwr_ctrl...continued

Bit	Symbol	Access	Value	Description
11 to 8	pwr[3:0]	R/W		Idle cycles before asserting power savings features: indicates how many SDRAM clock cycles (SDCLKO) should pass while the SDI is idle before issuing the power saving features as indicated by the pwr[1:0] parameter
			0*	disabled
			n	n * 16 clock cycles
7 to 2	-	R	0*	reserved
1 to 0	pwr[1:0]	R/W		Idle time power savings features: indicates which power savings feature to implement after the pwr[3:0] counter has reached its terminal count
			0b00*	disabled
			0b01	close all pages (PRECHARGE ALL)
			0b10	enter power down mode (de-asserts SDCKE) ^[1]
			0b11	close all pages and enter power down mode

[1] An autorefresh cycle forces a wakeup from PowerDown, but PowerDown mode will not be entered after refresh cycle has finished again.

Table 377: Register sdi_rfrsh1

Bit	Symbol	Access	Value	Description
31 to 26	-	R	0*	reserved
25	holdsref	R/W		Hold self-refresh mode: When this bit is set, the controller will be kept in selfrefresh when as selfrefresh command is given. The controller will exit this mode again when the bit is cleared. This feature can be used for power saving features. When a frequency change is required, then the controller can be put in hold self-refresh mode. Since the data ports are stalled in this mode it is safe to change the register settings to match with the new frequency.
			0*	self-refresh mode is not held
			1	self-refresh mode is held until this bit is cleared

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 377: Register sdi_rfrsh1...continued

Bit	Symbol	Access	Value	Description
24	renable	R/W		Refresh enable: enable the occurrence of a row refresh. A row refresh should not occur during the first 200us after CKE was asserted while powering up or recovering from deep power down.
			0*	auto-refresh generation disabled
			1	auto refresh generation enabled
23 to 16	-	R	0xFF*	reserved. Should always be written with 0xFF
15 to 12	-	R	0*	reserved
11 to 0	rcnt[11:0]	R/W	0x040	Refresh count: indicates the maximum number of clock cycles before a Auto Refresh occurs. An Auto Refresh should occur no later than a determined amount of time. The equation used to determine this value rcnt[11:0] is as follows: RP = page (rows) refresh period (refer to SDRAM datasheet, usually: 15.6 µs) DP = number of SDRAM pages (rows per bank) (refer to SDRAM datasheet) qsize = refresh queue size (See Table 378) $T_{SDCLKO_cycle} = \text{period of SDCLKO (note: } T_{SDCLKO} = T_{hclk})$ $\cdot cnt = \frac{DP \cdot RP}{(DP + qsize) \cdot T_{SDCLKO_cycle}}$

Table 378: Register sdi_rfrsh2

Bit	Symbol	Access	Value	Description
31 to 16	rpost[15:0]	R/W	0*	Refresh post amount: indicates the maximum number of refresh commands to issue when the refresh post interrupt signal is asserted and refresh posting interrupts are enabled. Please refer to the SDRAM vendors data sheet for information regarding the maximum number of refreshes postable. This feature is not available at DVFD818x Family
15 to 11	-	R	0*	reserved
10 to 8	qsize[2:0]	R/W	0*	Refresh queue size: indicates the maximum number of refreshes to queue before actually doing a refresh (1 to 8)
			n	number refresh minus 1 (n = 0 means 1 refresh)
7 to 5	-	R	0*	reserved

Table 378: Register sdi_rfrsh2...continued

Bit	Symbol	Access	Value	Description
4	posthalt	R/W		Refresh post interrupt halt: enables / disables the refresh posting interrupt capabilities of the refresh engine. If the SDI is servicing a refresh interrupt and the command pipeline is no longer IDLE, then refresh posting will halt. If this feature is disabled, then the refresh posting will continue until (rpost[15:0] + qsize[2:0]) number of refreshes has been posted regardless if the command pipeline is BUSY.
			0*	disabled
			1	enabled - not available at DVFD818x Family
3 to 1	-	R	0*	reserved
0	postenable	R/W		Refresh post interrupt enable: enables / disables the refresh posting interrupt capabilities of the refresh engine
			0*	disabled
			1	enabled - not available at DVFD818x Family

Table 379: Register sdi_tim1

Bit	Symbol	Access	Value	Description
31 to 28	-	R	0xF*	reserved
27 to 24	twr[3:0]	R/W	0xF*	Write recovery period
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
23 to 20	tmr[3:0]	R/W	0xF*	'Load Mode Register' command to command
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
19 to 16	tpd[3:0]	R/W	0xF*	'Precharge' command to power-down
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
15 to 12	trrd[3:0]	R/W	0xF*	Active bank A to active bank B delay
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
11 to 8	trcd[3:0]	R/W	0xF*	Active to read or write delay
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
7 to 4	tras[3:0]	R/W	0xF*	Active to 'Precharge' command
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))
3 to 0	trp[3:0]	R/W	0xF*	'Precharge' command period
			n	number of clock cycles (n = 0 means 16 clock cycles, n > 0 mean n clock cycle(s))

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 380: Register sdi_tim2

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0xFFFF*	reserved
15 to 8	txsr[7:0]	R/W	0xFF*	Exit self refresh to active command
			n	number of clock cycles minus 1 (n = 0 means 1 clock cycle)
7 to 0	trc[7:0]	R/W	0xFF*	Auto refresh, active command period
			n	number of clock cycles minus 1 (n = 0 means 1 clock cycle)

Table 381: Register sdi_mode

Bit	Symbol	Access	Value	Description
31 to 12	-	R	0*	reserved
11 to 7	mode[4:0]	R/W		Operating mode
			0*	SDR - Single Data Rate
			others	reserved
6 to 4	latency[2:0]	R/W		CAS latency
			0b010*	2 cycles
			0b011	3 cycles
			others	reserved
3	type	R/W		Burst type
			0*	sequential
			others	reserved
2 to 0	burst[2:0]	R/W		Burst length
			2*	4 data
			3	8 data
			others	reserved

Company Confidential

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 382: Register sdi_ext_mode

Bit	Symbol	Access	Value	Description
31 to 16	-	R	0*	reserved
15 to 7	xmode[8:0]	R/W		Extended operating mode , not used at DVFD818x Family
			0*	This field should be always written with 0
6 to 5	ds	R/W		Driver Strength of the SDRAM outputs. This value should be set according to the application's requirements (see datasheet of the used SDRAM).
			0b00*	full strength
			0b01	half strength
			0b10	quarter strength
			0b11	reserved
4 to 3	tcr[1:0]	R/W		Temperature Compensated Self Refresh (TCSR) : specifies the refresh period during self refresh depending on SDRAM case temperature
			0b00*	70 °C
			0b01	45 °C
			0b10	15 °C
			0b11	85 °C
2 to 0	pasr[2:0]	R/W		Partial Array Self Refresh (PASR) : self refresh enabled banks
			0b000*	all banks
			0b001	bank 0, bank 2
			0b010	bank 0
			0b101	50% bank 0
			0b110	25% bank 0
			others	reserved

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 383: Register sdi_sched0 to 2

Bit	Symbol	Access	Value	Description
31 to 16	age[15:0]	R/W	0x0000*	Transaction age time-out count: indicates the amount of time in number of clock cycles before increasing the ports priority
			0x0000*	lowest
			0xFFFF	highest
15 to 13	-	R	0*	reserved
12	age_type	R/W	0*	Transaction age tracking type: specifies how the transaction age count is initiated and terminated
			0	begin age tracking when while requesting but not granted
			1	begin age tracking while not granted
11 to 9	-	R	0*	reserved
8	age_enable	R/W	0*	Transaction age tracking enable: enables the ports ability to preempt current transactions by systematically increasing the ports priority as the pending transactions age increases. To avoid blocking, this feature should be enabled for all ports.
			0*	disabled
			1	enabled
7 to 6	-	R	0*	reserved
5 to 4	priority[1:0]	R/W	0*	Port priority: an increased port's priority positively influences access privileges.
			0b00*	lowest
			0b11	highest
3 to 1	-	R	0*	reserved
0	port_type	R/W	0*	Scheduler mask: specifies the data transaction type. Latency critical ports should be of type CPU as they may preempt already in-progress transfers.
			0*	normal (higher latency)
			1	CPU (lower latency)

Table 384: Register sdi_wtag0 to 2

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3	-	R	0*	reserved
2	port_2	R/W	0	Port 2 read matching port X posted write tag enable: when enabled, reads originating from port #2 may tag matching posted writes in the port in question (port #0 for sdi_wtag0 , port #1 for sdi_wtag1 , etc...)
			1*	disabled
			1*	enabled
1	port_1	R/W	1*	same for port #1
0	port_0	R/W	1*	same for port #0

Table 385: Register sdi_tam

Bit	Symbol	Access	Value	Description
31 to 8	-	R	0*	reserved
7 to 0	mask[7:0]	R/W	0*	Tag address mask: when a bit in mask[7:0] is 1, the corresponding port address bit is not considered during coherence checking mask[7] - address[31] ... mask[0] - address[24] Note: only mask of bits 31 to 24 is possible

Table 386: Register sdi_sw_ctrl

Bit	Symbol	Access	Value	Description
31 to 4	-	R	0*	reserved
3 to 0	swcr[3:0]	R/W		Software command:
			0x0	NOP No operation
			0x1	PRECHARGE ALL This will precharge all banks
			0x2	SELF REFRESH A SELF REFRESH command is issued to the SDRAM. Once in self refresh mode, any pending request will automatically “wake up” the SDRAM and the transfer begins (except in hold self-refresh mode, see Section 9.26.6.6 “SDRAM in hold self-refresh mode”). All pending refreshes in the refresh queue is ignored and the refresh engine will reset itself upon exiting SELF REFRESH mode.
			0x3	AUTO REFRESH An AUTO REFRESH is sent to the SDRAM
			0x4	ACTIVE POWER DOWN (without precharge) issues the POWER DOWN command to the SDRAM hence putting the SDRAM in active power down mode
			0x5	PRECHARGE POWER DOWN precharges all banks (if necessary) and issues the POWER DOWN command to the SDRAM hence putting the SDRAM in PRECHARGE POWER DOWN mode
			0x6*	DEEP POWER DOWN Precharges all banks (if necessary) and issues the DEEP POWER DOWN command to the SDRAM hence putting the SDRAM in DEEP POWER DOWN mode
			0x7	EXIT POWER DOWN Exit power down mode by asserting the SDRAM clock enable pin
			0x8	LOAD MODE REGISTER The load mode register command will send the appropriate LMR command to the SDRAM. The information sent to the SDRAM is pre-determined by information contained in sdi_mode (Table 381)
			0x9	LOAD EXTENDED MODE REGISTER The load extended mode register command will send the appropriate LXMR command to the SDRAM. The information sent to the SDRAM is pre-determined by information contained in sdi_ext_mode (Table 382)
			0xA to 0xF	reserved

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 387: Register sdi_stat

Bit	Symbol	Access	Value	Description
31 to 21	-	R	0*	reserved
20	srs	R		Self Refresh Status:
			0	SDRAM memory is not in self refresh
			1*	SDRAM memory is in self refresh
19 to 0	-	R	0x1001F*	reserved

9.26.5 Functional description

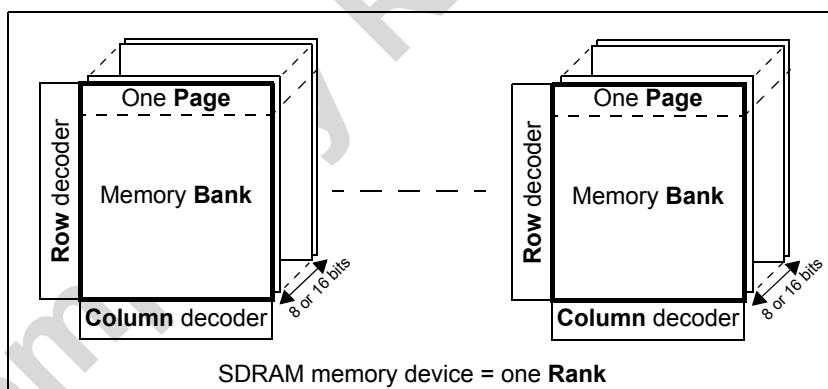
9.26.5.1 Configuration port

The SDI has one configuration port which is located together with all AHB configuration register on a dedicated Multi-layer AHB bus slave (See 9.2 and 9.3). All register described in 9.26.4 are accessed through the configuration port.

9.26.5.2 Data ports

The SDI has three data ports which are used to access the content of the SDRAM device.

9.26.5.3 General SDRAM information

**Fig 136. SDRAM overview**

SDRAM data (8 or 16-bit) are grouped to form a page (256 to 2048 data) decoded by the column address (8 to 11 bits). All pages (rows) are grouped to form a bank (512 to 8192 pages). All banks are grouped to form an SDRAM device (1 to 4 banks) which is also referenced as a rank when multiple SDRAM device are used.

Example: 128 Mbit SDRAM has 4 banks with 4096 rows with 512 times 16-bit data

Mobile-SDRAM: This is an extension of SDRAM specification with additional power saving features like:

- TCSR; Temperature Compensated Self Refresh. This feature allow to set different Self Refresh rate depending on SDRAM die temperature. Reducing Self Refresh rate is possible at lower temperature and will reduce SDRAM power consumption. Some SDRAM has built in temperature sensor and the temperature tuning is then automatically done.

- PASR: Partial Array Self Refresh. This feature allow to specify which bank or part of bank to be refresh in Self Refresh mode. Not refresh data are lost. This will reduce SDRAM power consumption.
 - DS: Drive Strength. This feature allow to change the SDRAM signals drive strength in order to reduce SDRAM power consumption when wire load is low.
 - DPD: Deep Power Down. This will totally stop the SDRAM and reduce the power consumption to only few microamps. All data are lost. Entering into this mode is done by issuing the DEEP POWER DOWN command in **sdi_sw_ctrl** register. Exiting from deep power down is similar than first power on sequence (See SDRAM datasheet for more information).

9.26.5.4 Addressing modes

each of the 3 data ports of the SDI is capable of addressing 256 Mbytes (2 Gbits). All ports point to the same physical SDRAM device.

The physical address is divided into several pieces: column address, row address, bank address and rank select. Rank select is not used in DVFD818x Family since there is only one chip select.

In order to extract this information from the complete address, several parameters must be known: column width, row width, number of banks, ranked configuration and SDRAM width. These parameters are located in **sdi_cfg** register and affect how the column address, row address, bank address and rank select are extracted.

Remark: From SDI point of view, having 1 x 16-bit wide or 2 x 8-bit wide SDRAMs makes no difference since, in this case, both SDRAM use the same chip select.

9.26.5.5 Programmable address extraction

Address extraction is the operation to convert system address (AHB) in column/row/bank/rank signals to the SDRAM device.

Figure 137 illustrates the different address extraction schemes that are provided by the SDI.

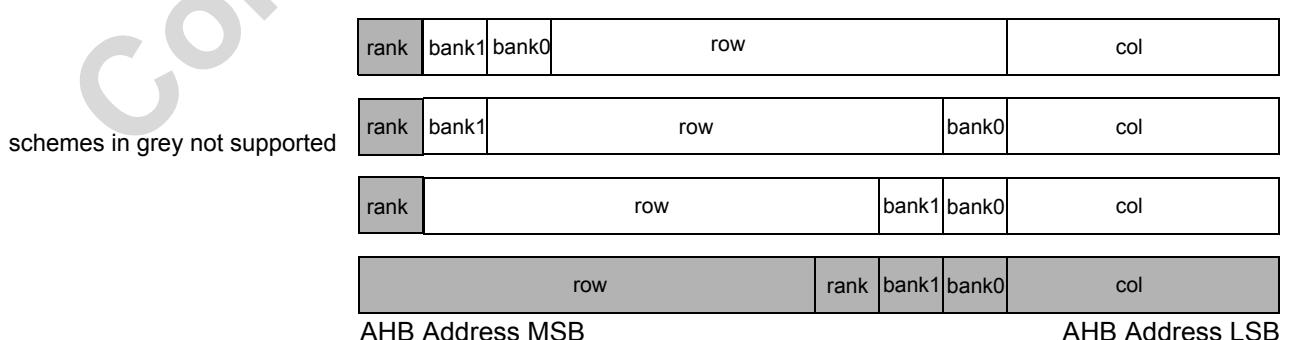


Fig 137. Address extraction schemes

Rank / bank address placement can increase performance by making a single page appear larger than normal. This is achieved by placing the rank / bank address pins just below the row address pins where they may toggle more frequently. Instead of opening a new page in the same bank (which takes `trcd` cycles where no read in this

bank is possible), the SDI will open the new page in the next bank (which can be done in parallel to reads in another page). So, with 4 banks per module, it is possible to make a single page appear 4 times as large in a single rank configuration (DVFD818x Family support only one rank).

However, the power consumption is the lowest when only the minimum of banks are open. So, placing bank address pins below the row pins may increase slightly the power consumption.

9.26.5.6 Transfer size

The SDI has three data ports in order to enhance performances by interleaving access from different AHB masters while another AHB masters are waiting data from the SDRAM device on other SDI data ports. The SDI supports byte, half-word and word accesses to/from either one 16-bit wide or two 8-bit wide SDRAM devices. Accesses with a size that exceeds the SDRAM width (for example, 32-bit read with only 1 x 16-bit wide SDRAM) are handled internally by the SDI.

SDRAMs have a pre-defined burst size that is programmed during the SDI setup and that is typically 4 or 8 elements (one element is 8-bit for an 8-bit wide SDRAM and 16-bit for a 16-bit wide SDRAM). Whenever a READ or WRITE is broadcasted to the SDRAM, the SDRAM returns (or accepts) the pre-defined number of elements.

Transactions which require more elements than the pre-defined SDRAM burst size are divided into SDRAM burst size multiples by the SDI. This mechanism supports low latency port preemption by higher priority ports as well as efficient data port transaction handling.

If a transaction is smaller than the SDRAM burst size, the transaction completes thus terminating the remainder of the SDRAM burst (with use of SDUDQM and SDLDQM signals to qualify useful data), allowing other pending transfers access.

The SDI is capable of full bank interleaving when possible. Bank interleaving means to start request data access on a bank while ongoing data transfer is currently done on another bank. Bank interleaving ensures a maximum throughput to bandwidth ratio by issuing, whenever possible, PRECHARGE and ACTIVE commands during ongoing SDRAM read or write transfers.

Remark: in case a data port receives a large number of write transfers in succession the command FIFO in the SDI may become full. At this time, the ongoing write data transfers is waited until there is space in the command FIFO.

9.26.5.7 Arbitration between data ports

the port arbitration is handled by 2 independent arbitration units: command arbiter and interleave arbiter.

The command arbiter is responsible for the scheduling of refreshes, configuration accesses, data port accesses, and power saving requests to the command execution unit.

The interleave arbiter is responsible for the scheduling of PRECHARGE and ACTIVE commands from requesting data ports whenever interleaving is possible.

Each arbiter is constructed with a priority encoder based architecture where each data port has 10 bits of priority assigned. Each entry of the priority encoder has an underlying priority assigned: the lower the port number, the higher its underlying priority.

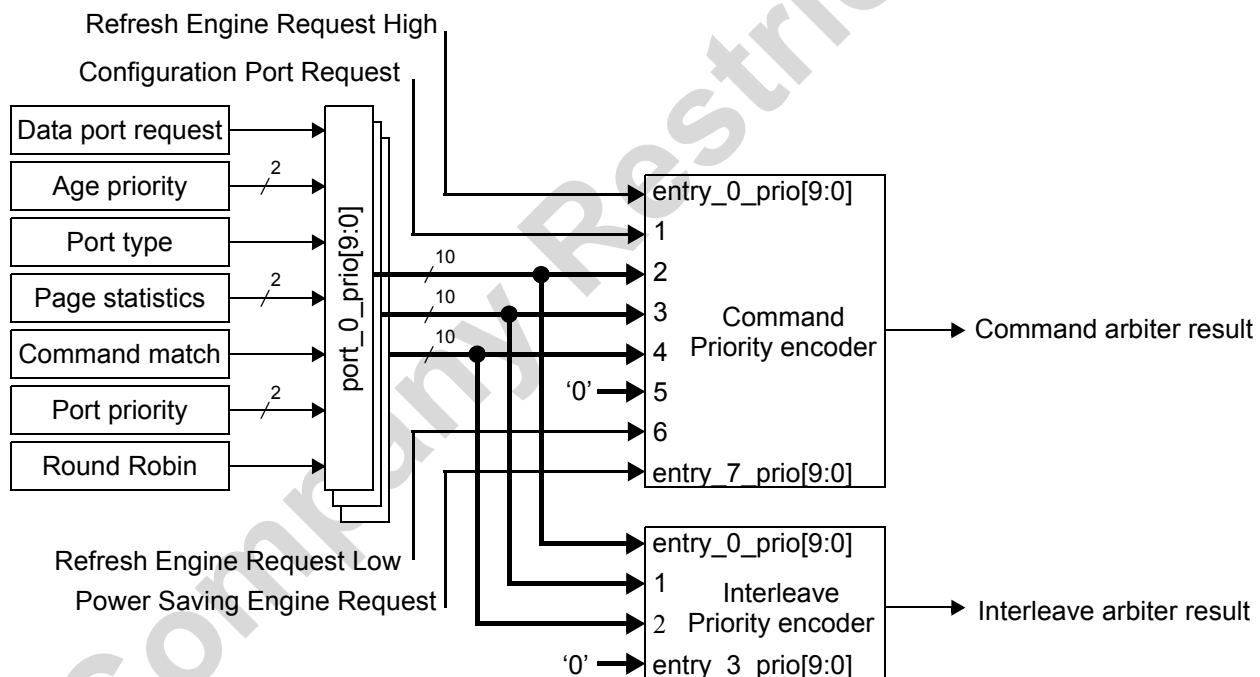


Fig 138. Arbitration overview

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The 10-bit priority mechanism is divided into categories. This is described in Table 388.

Table 388: Data port priority encoding

port_X_pri[9:0] ^[1]	Description	Reference
[9]	port request	
[8]	programmable transaction age count bit [1]	• “Programmable Transaction Age Count” on page 435
[7]	programmable port type	• “Programmable Port Type” on page 436
[6]	port page statistics	• “Port Page Statistics” on page 436
[5]		
[4]	command match	• “Command Match” on page 435
[3]	programmable transaction age count bit [0]	• “Programmable Transaction Age Count” on page 435
[2]		
[1]	programmable port priority	• “Programmable Port Priority” on page 435
[0]	round robin	• “Round Robin” on page 435

[1] X = 0, 1 or 2

Table below depicts the priority encoder entry assignments for the command arbiter. The command arbiter contains eight entries of which three are data ports

Table 389: Command arbiter priority encoder entry assignments

Entry#	Underlying Priority	entry_Y_pri[9:0] ^[1]	Requestor
0	Highest	R 1 1 1 1 1 1 1 1 1	Refresh Engine Request High
1		R 1 1 1 1 1 1 1 1 1	Configuration Port
2		port_0_pri[9:0]	Data port #0
3		port_1_pri[9:0]	Data port #1
4		port_2_pri[9:0]	Data port #2
5		0 0 0 0 0 0 0 0 0 0	unused data port
6		R 0 0 0 0 0 0 0 0 0	Refresh Engine Request Low
7		R 0 0 0 0 0 0 0 0 0	Power Saving Engine

[1] Y = 0 to 7

Table below depicts the priority encoder entry assignments for the interleave arbiter. The interleave arbiter contains four entries of which three are data ports

Table 390: Interleave arbiter priority encoder entry assignments

Entry#	Underlying Priority	entry_Y_pri[9:0] ^[1]	Requestor
0	Highest	port_0_pri[9:0]	Data port #0
1		port_1_pri[9:0]	Data port #1
2		port_2_pri[9:0]	Data port #2
3		0 0 0 0 0 0 0 0 0 0	unused data port

[1] Y = 0 to 3

- **Round Robin**

The round robin feature ensures that all port requesters that have equal priority will get uniform access. That is, entry_Y_pri[8:1] is equal for all Y where {Y = 2, 3, 4}. Bit entry_Y_pri[0] where {Y = 2, 3, 4} is set or clear dynamically to have uniform access priority.

- **Programmable Port Priority**

Programmable port priorities influence a ports probability of obtaining a grant. Two priority bits are allocated for each port to indicate its base priority. The higher the value the higher the port's base priority becomes.

- **Command Match**

Read transfers typically have an associated latency defined as CAS Latency. That is, whenever a read command is sent to the SDRAM device, 2 or 3 cycles elapse before these devices return valid read data. A four element read burst requires up to 7 SDRAM cycles worst case (3 cycles of CAS Latency and 4 for the return data). Write commands cannot be posted to the SDRAM device until after all read data is returned (cycle 8). However, read commands can be posted in cycle 4 as defined by the JEDEC standard. This results in a continuous stream of read data and better SDRAM utilization as the CAS Latency for the subsequent read command is hidden by the previous read command. The scheduler can increase utilization by increasing a data port's priority whenever the previously scheduled command was a read and the current data port's command is also a read. [Table 391](#) indicates the possible values which is assigned to a port's command match priority. The higher the value the higher the port's command match priority becomes.

[Table 391: Command Match Priority Level](#)

Read Match	Priority
Yes	1
No	0

- **Programmable Transaction Age Count**

To prevent data port starvation, a transaction age counter is employed for each data port. This mechanism allows lower priority ports to preempt higher priority ports if its transaction has been pending for a prolonged period of time. The age count consists of two counters. The first counts a pre-defined number of clock cycles ([age\[15:0\]](#) in [Table 383](#)) at which point it increments a two bit age unit counter when reaching terminal count. Once the port is granted the two bit counter is reset to zero. Table below indicates the possible values which may be assigned to a port's transaction age priority. The higher the value the higher the port's transaction age priority becomes.

[Table 392: Transaction Age Priority Levels](#)

Age Count	Priority	
	Age[1]	Age[0]
The transaction has waited a maximum of four age units	1	1
The transaction has waited three age units	1	0
The transaction has waited two age units	0	1
The transaction has waited one age unit	0	0

• Port Page Statistics

In order to better utilize open pages within the SDRAM, page statistic information may be used by both arbiters to influence scheduling. Each data port provides the current transaction address to a page table lookup module. This module returns the result of the lookup to the arbiter modules and data port. This statistical information includes whether the bank in question is active and if the page in question is a page hit or page miss. This two bit result is used to increase the port's priority. Table below indicates the possible values which are assigned to a ports page statistic priority. The higher the value the higher the port's page statistic priority becomes.

Table 393:Page Statistics Priority Levels

Lookup		Priority	
Bank	Page	Page[1]	Page[0]
active	hit	1	0
precharged	don't care	0	1
active	miss	0	0

• Programmable Port Type

CPUs are typically latency critical and require immediate access to SDRAM. A single priority bit is allocated for each port to indicate its latency characteristics. In order to guarantee low latency it is necessary to preempt a port whose transaction is already in progress. Once the low latency port has been granted access it will not relinquish its access privileges unless its transfer is completed or it is preempted. Any of the following conditions allow a low latency port to be preempted:

- A requesting port's two bit age status is greater than 1
- A configuration port is requesting
- A high priority refresh is requesting
- Another low latency port is requesting and its priority value has surpassed that of the current low latency port

Table below indicates the possible values which may be assigned to a port's latency priority. The higher the value the higher the port's latency priority becomes.

Table 394:Latency Priority Level

Lowest Latency	Priority
Guaranteed	1
Not Guaranteed	0

The overall priority encoding scheme is described below using pseudo language.

```
1 if (there is a high-priority refresh request)           then [1] grant request to refresh
2 else
3 if (the configuration port makes a request)          then [2] grant request to this port
4 else
5 if (transaction age counter is enabled)
6   AND this port has waited more than 2 age units
7   AND it has the lowest entry number amongst such ports)
8 else
9 if (this port is a low-latency CPU port
10  AND it has the lowest entry number amongst such ports)
11 else
12 if (this port has a page hit
13  AND it has the lowest entry number amongst such ports)
14 else
15 if (the required bank is already pre-charged
16  AND it has the lowest entry number amongst such ports)
17 else
18 if (the burst direction matches the previous burst (read / write)
19  AND this port has the lowest entry number amongst such ports)
20 else
21 if (transaction age counter is enabled
22  AND this port has waited more than one age unit
23  AND it has waited for the longest amount of time amongst all the ports
24  AND it has the lowest entry number amongst such ports)
25 else
26 if (this port has the lowest entry number amongst such ports)
27 else
28 if (this port is requesting
29  AND it hasn't been served for the longest time (round-robin))
30 else
31 if (there is a low priority refresh request)
32 else
33 (default request from the power saving engine)
34 end of arbitration
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
```

9.26.5.8 Data Coherency Between Multiple ports

Memory transfer requests at a single data port are processed and executed in the order that they are received. The SDI does not re-order the commands on a single data port. From this point of view data coherency between memory bus agents that connect to a single port on the controller is guaranteed.

However, those memory transfer requests, that are made by different data ports, may not be serviced in the order that they appeared. The order in which these requests are serviced depends on the state of the SDRAM device and how the internal arbiter is programmed. In order to yield a high performance, each data port buffers the write data before it is sent to SDRAM. The size of the write buffers is 16 words.

Note that the SDI begins requesting access to the SDRAM when enough write data has been in the buffer to complete a single SDRAM burst.

Assume that the ARM starts a read operation via port #0 that targets a memory address for which (already for quite some time) write data sits in the write data FIFO of port #1. In order for the ARM read operation to provide coherent data, first the relevant part of the write data FIFO in port #1 needs to be flushed to SDRAM before the ARM read operation should continue. The Coherence checking implementation is described below.

Coherence Checking Implementation

This section describes the implementation techniques used in SDI for maintaining data coherence between multiple ports. In this implementation address checking is based on the AHB transaction and is done with 1 KByte granularity.

- Known Conditions:

- AHB commands on a single port are strongly ordered. This means the write buffer in a certain AHB bus interface will always be flushed before processing a subsequent read transaction on the same port. For this reason there is no requirement to compare the read command with pending write commands in the same port.
- Every write transfer is assigned with a tag value by a tag counter. The tag value signifies the order of accepting the write transfers.

- When to flush:

- If starting address of AHB read transaction (incl. INCR of unspecified length) is in same 1KByte region than the address of write data in the write buffer of another port.

Note: Address regions of 1KByte are defined as address 0x0 upto 0x3FF, 0x400 upto 0x7FF, 0x800 upto 0xBFF, etc.

- Only write data that belong to AHB transactions that are fully posted on the write port the moment the read command is sampled are considered for coherence checking. Any write data in the write buffers belonging to AHB transactions that are not finished are not considered for coherency. Even if it might be to an address in memory that matches the read on another port.
- What will happen in case the read address matches (with 1KByte granularity) writes in two or more write buffers see fourth bullet below

• What to flush:

- On a write buffer address hit, the complete AHB write transaction that has the hit gets flushed. This is accomplished using the tag.
- Any write data belonging to a write transfer, that had completed prior to the tagged write transfer, gets flushed before the tagged write transfer data, to maintain a strongly ordered architecture.
- Any subsequent data in the write data buffer does not get flushed. This data is identified by its higher tag value.

• Arbitration between ports in case of a hit:

This section describes the arbitration of ports when one or more read transfers have matching

write transfers in the write buffers of other ports.

- As each tagged write transfer is being processed, the port with the write transfer inherits the priority of the tagging read port for the duration of the write. Thus the write transfers that are required to be flushed before a higher priority read transfer are flushed first.
- The write transfers that match with a lower priority read transfer, wait until the lower priority read transfer becomes the highest priority transfer.
- In case, if the port with the matching write transfer has a higher priority than the tagging port with the read transfer, then the priority of the tagging port is not inherited by the port with the write transfer.
- All the matching write transfers from all the ports are completed before a read transfer happens. In case if more than one ports need to be flushed, the arbitration between those is done based upon the other criteria such as port order.
- In case if there are no matching write transfers in any of the write buffers, it does not have any impact on read latency whereas in the case of a hit, the penalty on read latency is that (minimal part of) the write data buffers of interest get flushed first, and the read request has to go through arbitration again.
- Flushing write buffers for the sake of coherency may have a minimum impact on overall memory bandwidth.

9.26.5.9 Programmable Refresh Features

The Refresh Engine implemented in the SDI is designed to supply an AUTO REFRESH command at the appropriate time interval. To add in flexibility the refresh engine is capable of queuing refresh requests. Queuing of refresh commands may lower the probability of data port preemption by the refresh engine and allow refreshes to occur at more opportunistic times such as when the SDI is idle.

- **Normal Refresh Operation**

The refresh engine accomplishes the refresh procedure by means of requesting access to the SDI execute engine. There are two types of requests that can be generated; low priority and high priority. The low priority request is generated whenever a refresh transaction is pending in the refresh engine queue. The high priority request is generated when the refresh queue reaches a pre-programmed number of pending refreshes as indicated by **sdi_rfrsh2** register on [Table 378](#). Lower priority requests are granted access to the execute engine whenever higher priority ports are not requesting as explained in [Section 9.26.5.7 "Arbitration between data ports" on page 433](#).

The average auto-refresh interval is programmable via the **rcnt** and **qsize** fields of the **sdi_rfrsh2** ([Table 378](#)) and **sdi_rfrsh1** ([Table 377](#)) registers respectively. An auto refresh request is queued periodically every **rcnt** clock cycles. While the number of queued requests is smaller than or equal to **qsize**, these requests have low priority. Whenever the number of queued requests exceeds **qsize**, the requests get high priority.

When programming a non-zero value to **qsize**, the average auto refresh interval will decrease, because a smaller value must be programmed to **rcnt**, to guarantee timely refreshing under worst case conditions. See the expression in [Table 377](#).

In an application that occupies the full SDRAM bandwidth, with no idle time at all, the refresh queue is filled up to **qsize** continuously, making all refresh requests high priority. This application will not benefit from this feature but rather will suffer from the smaller refresh interval.

Note that auto-refresh generation is disabled at power-on. It should be enabled after the initial power-up delay (normally 200us) has expired, using the reenable bit in **sdi_rfrsh1**.

9.26.5.10 Programmable Power-saving Features

Power Saving Engine

The SDI is designed to provide flexibility in performance and power consumption. The SDI implement the following programmable parameters (See [Table 376](#)):

- **pwr[1:0]:** Power saving mode:

Table 395: Power Saving Modes

pwr[1:0]	Mode	Description
0b00	disabled	Power savings features are disabled
0b01	precharge-all	When enabled, this option will close all open pages by issuing a PRECHARGE ALL command after the SDI has been idle for pwrx[3:0] cycles. The SDCKE pin will still be asserted for quick transfer response
0b10	active power down	When enabled, this option will de-assert the clock enable SDCKE pin. The SDI will remain in Clock Suspend mode until a port requests access. If a refresh request occurs all pages will close and the SDI will issue the refresh at which point the power count procedure begins again. It requires an additional cycle to Power-Up the SDRAM devices as defined in the JEDEC SDRAM Specification for enabling SDCKE
0b11	precharge + power down	When enabled, this option will close all open pages by issuing a PRECHARGE ALL command followed by de-assertion of the SDCKE. This achieves a powered down mode where the SDI will remain in power down until a port requests access. If a refresh request occurs all pages will close and the SDI will issue the refresh at which point the power count procedure begins again. It requires an additional cycle to Power-Up the SDRAM devices as defined in the JEDEC SDRAM Specification for enabling SDCKE

- **pwrx[3:0]** selects the number of SDCLKO cycles to wait in the idle state before implementing the features described in **pwr[1:0]**.
- **pwry[3:0]** selects the number of SDCLKO cycles to wait in the idle state before issuing the SELF REFRESH command to the SDRAM device. If **pwrx[3:0]** is greater than zero then the SDI waits (**pwrx+pwry**) SDCLKO cycles before issuing the SELF REFRESH command.

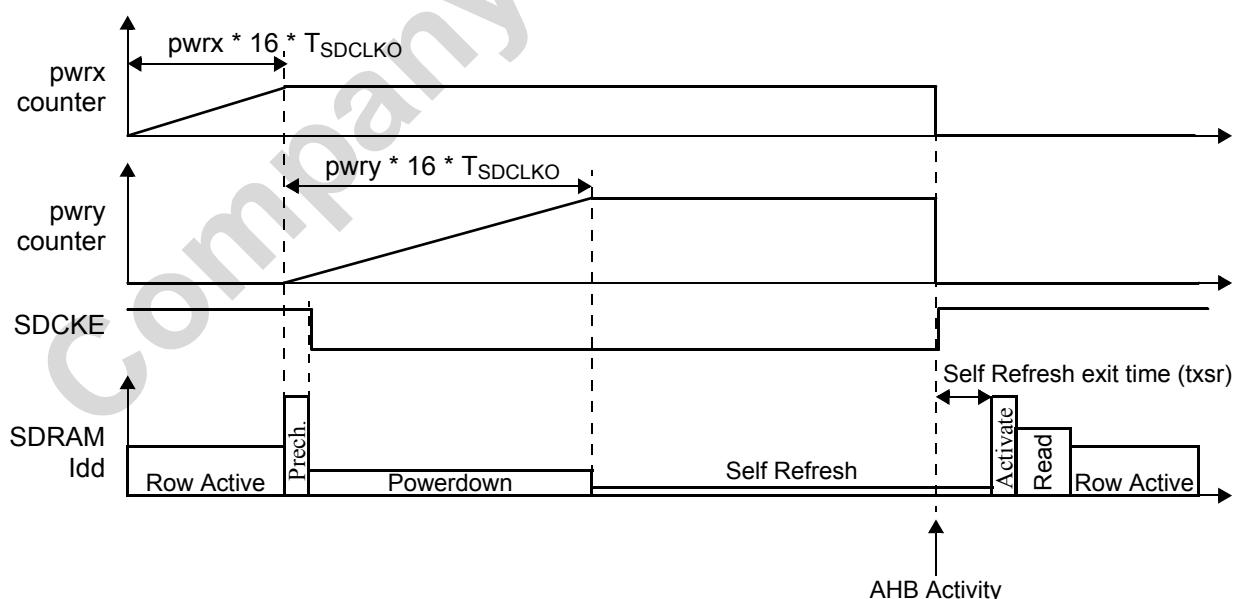


Fig 139.Example of power saving sequences (pwr[1:0] = 0b11), no Auto-Refresh while in Powerdown

The user is capable of selecting several combinations of the methods described above hence satisfying most power/performance issues. These parameters are controllable via the **sdi_pwr_ctrl** register on [Table 376](#). The SDI will automatically

issue a EXIT POWER DOWN command when there is a pending request from a data port or the configuration port and complete the transaction. Once the transaction is complete the power count procedure begins again.

Remark: PASR and DPD are never issued automatically and is the responsibility to the software to use it in order to save more power in specific system state.

9.26.5.11 Bank interleaving feature in SDRAM interface

The SDI is capable of full bank interleaving when possible. Bank interleaving ensures a maximum throughput to bandwidth ratio by issuing PRECHARGE and ACTIVE commands in parallel with read/write transfers whenever possible.

9.26.5.12 Speculative reading

In order to improve the performance with bus read bursts that are larger than the SDRAM burst length, a data port whose read command is sent to SDRAM, automatically requests for the next read data, unless another transfer is offered at the data port.

When there is little data port activity, and mainly small bursts or even single transfers, this might lead to unnecessary SDRAM accesses, causing additional power consumption.

To avoid this from happening, this speculative reading can be disabled per data port.

9.26.6 Application information**9.26.6.1 SDRAM and SDI setup**

The SDI must be set-up after power up or reset. The order in writing to the various Configuration registers is very important for proper SDI operation. Description below shows the order to correctly set-up the SDI. SDI set-up is done through configuration port.

- Various control register(s) - External configuration like pin multiplexing, system clocks must be correctly set before SDI and SDRAM configuration
- Software control register - Send the exit power down command. More information can be found in [Table 386 “Register sdi_sw_ctrl” on page 429](#)
- The software should then wait for a specified time typically 200us (refer to the SDRAM vendor’s data sheet for this information) before continuing.
- SDI parameters register - SDI configuration information should be written. Refer to [Table 374 “Register sdi_cfg1” on page 420](#) and [Table 375 “Register sdi_cfg2” on page 422](#) for more detail. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Power savings parameter register - Power saving and refresh information should be written. [Table 376 “Register sdi_pwr_ctrl” on page 422](#) provides more insight to power saving/refresh setup. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Refresh parameter register 1 - Refresh cycle count information should be written. Auto-refresh generation should be enabled. [Table 377 “Register sdi_rfrsh1” on page 423](#) provides more insight to refresh setup. See also [Section 9.26.6.9 “SDI setting example” on page 447](#).
- Refresh parameter register 2 - Refresh posting and refresh queue information should be written. [Table 378 “Register sdi_rfrsh2” on page 424](#) provides more insight to refresh setup. See also [Section 9.26.6.9 “SDI setting example” on page 447](#).

Remark: Enabling refresh posting before proper setup via mode register load may result in undesirable SDRAM behavior.

- Timing parameters register 1 - SDI timing arc information should be written. Refer to [Table 379 “Register sdi_tim1” on page 425](#) for more detail. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Timing Parameters register 2 - SDI timing arc information should be written. Refer to [Table 380 “Register sdi_tim2” on page 426](#) for more detail. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Scheduler registers - The scheduler information should be written. Refer to [Table 383 “Register sdi_sched0 to 2” on page 428](#) for more detail. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Write tag information registers - The write tag information should be written. Refer to [Table 384 “Register sdi_wtag0 to 2” on page 428](#) for more detail. See also [Section 9.26.6.9 “SDI setting example” on page 447](#)
- Software control register - Send the power-up information through the software control register such as mode register load for the SDRAM device. This may require several writes to the register (refer to the SDRAM vendor’s data sheet for

proper power-up information). More information can be found in [Table 386 “Register sdi_sw_ctrl” on page 429](#). Informations on how to write SDRAM mode and extended mode registers are provided [9.26.6.2 on page 444](#).

9.26.6.2 SDRAM mode register write

In order to load the SDRAM internal mode register the LOAD MODE register command has to be sent to the software control register (**sdi_sw_ctrl**). Refer to [Table 386 “Register sdi_sw_ctrl” on page 429](#) for more detail. The SDI will then send the appropriate LMR command to the SDRAM.

If the SDRAM requires a PRECHARGE and REFRESH command before actually loading the mode register the procedure below must be used:

- Give PRECHARGE ALL command by writing into **sdi_sw_ctrl**.
- Give 8 times an AUTO REFRESH command into **sdi_sw_ctrl**.
- Give LOAD MODE REGISTER command by writing into **sdi_sw_ctrl**.

9.26.6.3 SDRAM extended mode register write

In order to load the SDRAM internal extended mode register the EXTENDED LOAD MODE register command has to be sent to the software control register (**sdi_sw_ctrl**). The SDI will then send the appropriate LEMR command to the SDRAM

- Give LOAD EXTENDED MODE REGISTER command by writing into **sdi_sw_ctrl**.

9.26.6.4 SDRAM into SELF REFRESH mode

Before setting the SDRAM in self-refresh mode, all banks must be precharged and some AUTO REFRESH may be required. After the command execution, self-refresh operation continues as long as SDCKE remains low. Description below shows what we have to write through the software control pseudo-register to get the SDRAM into self-refresh mode.

- Give LOAD EXTENDED MODE REGISTER command by writing into **sdi_sw_ctrl**.
- Give SELF REFRESH command by writing into **sdi_sw_ctrl**.

9.26.6.5 SDRAM out of self refresh mode

Any pending request to the SDRAM will automatically “wake-up” the SDRAM and the transfer begins.

9.26.6.6 SDRAM in hold self-refresh mode

This feature can be used when the frequency of the SDRAM needs to be changed during normal operation e.g power reduction. To safely change the registers to match the new frequency it is needed that the traffic at the SDRAM is stopped.

When bit **holdsref** in the **sdi_rfrsh1** register is set this mode is activated. As soon as a selfrefresh command is applied to the memory then the controller will stay in self-refresh mode. The controller can exit selfrefresh mode via resetting bit **holdsref**.

To uses the hold self-refresh the following should be programmed via the configuration port.

1. Activate hold self refresh by setting bit **holdsref** the **sdi_rfrsh1**
2. Give SELF REFRESH command by writing into **sdi_sw_ctrl**. Do not issue any other command while **holdsref** bit is set
3. You may need to wait a short period of time until you go to step 3 of about 20-30 clocks.
4. Read the Controller Status Register bit(20) SRS "Self Refresh Status"
5. If the read of CSR from step 3 returns:
 - a. 1'b0 then the auto refresh likely snuck in thus waking the device up. Start over by Going to step 2
 - b. 1'b1 so the memory is in Self Refresh and the auto refreshes should not "sneak" in.
6. Change various SDI and CGU settings
7. disable hold self refresh clearing bit **holdsref** in the **sdi_rfrsh1**. SDI will go out of self-refresh by any pending request

9.26.6.7 SDRAM in DEEP POWER DOWN MODE

To enter DEEP POWER DOWN MODE the following steps should be programmed:

1. Disable autorefresh by clearing bit **sdi_refresh1.renable**
2. Give DEEP POWER DOWN command by writing into **sdi_sw_ctrl**.

9.26.6.8 Capture setting

The **capture** and **capture_nedge** settings allow to capture data coming from SDRAM at different time. This is required to cope with timings at high frequency.

Table 396: Capture setting

capture	capture_nedge	Timing constraints
0	0	Can be used only for low SDCLKO frequency Delay between SDCLKO rising edge and SDCLKI rising edge must be less than: $\frac{T_{SDCLKO_cycle}}{2} - 2\text{ns}$
0	1	Can be used in most cases Delay between SDCLKO falling edge and SDCLKI rising edge must be less than: $\frac{T_{SDCLKO_cycle}}{2} - 2\text{ns}$
1	0	Can be used with bad SDRAM timings Delay between SDCLKO rising edge and SDCLKI rising edge must be less than: $T_{SDCLKO_cycle} + \frac{T_{SDCLKO_cycle}}{2} - 2\text{ns}$
1	1	Can be used with bad SDRAM timings Delay between SDCLKO falling edge and SDCLKI rising edge must be less than: $T_{SDCLKO_cycle} + \frac{T_{SDCLKO_cycle}}{2} - 2\text{ns}$

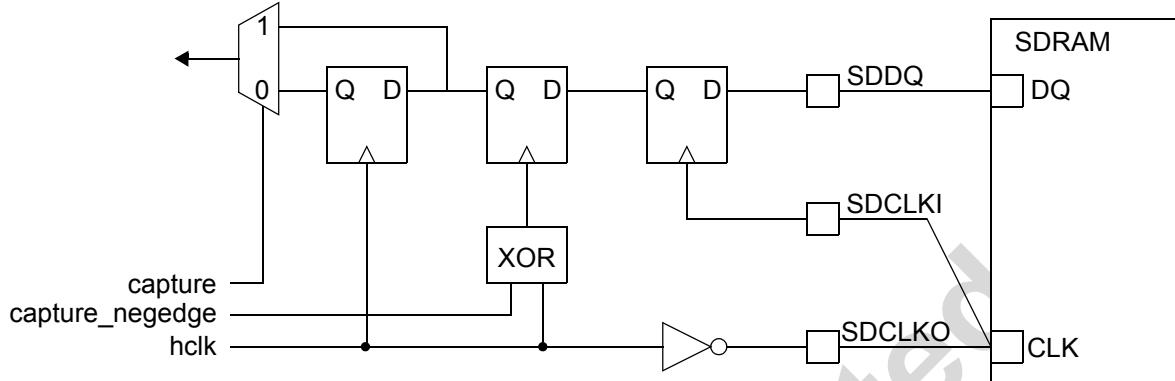


Fig 140.Simplified view of data input sampling

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.26.6.9 SDI setting example

This section propose a setting that can be used for the following configuration:

- SAMSUNG mobile SDRAM
 - Part number: K4M28163PD-R(B)G/S1L
 - 8M x 16-bit (128Mbit)
 - 105 MHz (CL=3) operation
 - 4 banks
 - 1, 2, 4, 8 and full page burst length
 - PASR, TCSR and DS features
 - 64ms refresh period (4096 cycle)
- 104MHz AHB clock

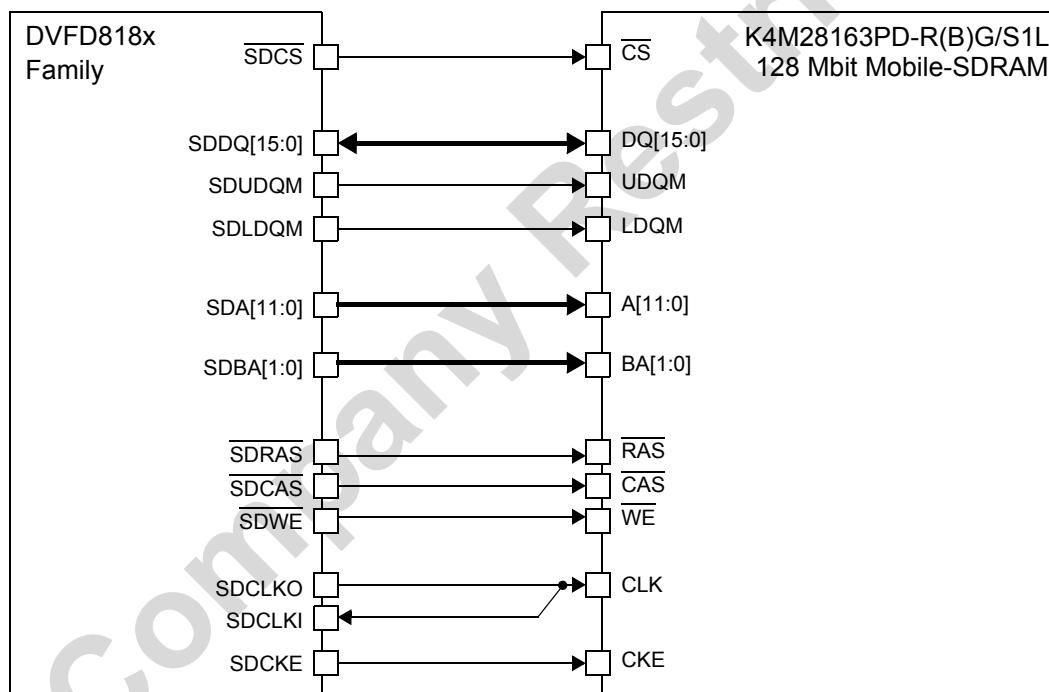


Fig 141.DVFD818x Family to SDRAM connection example

Table 397: Register setting example

Name	Value	Comment
sdi_cfg1	0x0002 2311	address extraction for best performances; 4 banks; 12 row address; 9 column address; 16-bit SDRAM
sdi_cfg2	0x0000 0100	capture on rising edge; 1 additional capture cycle; speculative read enabled on all ports
sdi_pwr_ctrl	0x0100 0000	SDCLKO enabled; no automatic power saving control; can be customized to target performance/consumption requirements; power saving commands can still be issued through sdi_sw_ctrl register.
sdi_rfrsh1	0x0000 0659	refresh of 4096 pages every 64ms with a queue depth of 1
sdi_rfrsh2	0x0000 0000	posting feature not available in DVFD818x Family
sdi_tim1	0x0112 2373	2 clock cycles write recovery time; 2 clock cycles load mode register command to command; 28.5ns precharge to power down (3 clock cycles); 19ns bank to bank delay (2 clock cycles); 28.5ns active to read or write (3 clock cycles); 60ns active to precharge delay (7 clock cycles); 28.5ns precharge command delay (3 clock cycles)
sdi_tim2	0x0000 0C0A	120ns exit self refresh to command (13 clock cycles); 105ns auto refresh cycle time (11 clock cycles)
sdi_mode	0x0000 0033	CAS Latency = 3; sequential burst; 8 data burst
sdi_ext_mode	0x0000 0030	85 °C temperature; self refresh enabled on all banks
sdi_sched0	0xFFFF 0001	no age tracking; default port priority; low latency
sdi_sched1	0xFFFF 0000	no age tracking; default port priority; normal latency
sdi_sched2	0xFFFF 0000	no age tracking; default port priority; normal latency
sdi_wtag0 to 2	0x0000 000F	full read matching enabled on all ports
sdi_tam	0x0000 00FC	AHB address bits A[31:26] masked for coherence checking

9.26.6.10 SDI performance

The following recommendations can be used for best performance:

- SDRAM clock duty cycle

The following schemes should be applied for best SDRAM duty cycle with the priority of:

- Use armclk = 2* ahbclk
- If armclk = ahbclk, than use for **cgucccon.msc** even values

If HMP operates at 3V the high speed performance of the pads can be selected. This will improve the duty cycle further and also reduces rise and fall times but with overshoot. It is recommended to use for 100MHz SDRAM clock a 125MHz or 133MHz SDRAM device for safe operation.

- Bandwidth

A burst transfer size of 32Bytes gives best performance both for data and code fetch (e.g. multiple load/store of 8 registers for data transfers).

- Maximum data write transfer bandwidth is >90% of the ideal bus bandwidth (armclk = 2*ahbclk, refresh cycles every 15.6us included, exclusive SDRAM access, 8registers used in a loop with store multiple command).

- Maximum data read bandwidth is >50% of the ideal bus bandwidth
(armclk = 2*ahbclk, refresh cycles every 15.6us included, exclusive SDRAM access, 8registers used in a loop with load multiple command). 1
- Maximum code fetch bandwidth is >50% of the ideal bus bandwidth
(armclk = 2*ahbclk, refresh cycles every 15.6us included, exclusive SDRAM access, cache load with 8 word burst). 2
- The limitation comes not from SDI block but the AHB inserts a non sequential cycle at the end of an 8word burst which leads to discontinued data transfer on the SDRAM interface. 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54

Company Restricted

9.27 SPI1 - Serial Peripheral Interfaces

The SC is equipped with a Serial Peripheral Interface (SPI). SPI is designed to be compatible to a large range of devices, therefore implement many functions.

9.27.1 Features

- SPI modes 0 to 3
- Half duplex synchronous transfers
- Master/slave operation
- DMA support for data transmit and receive
- 1 to 16-bit word length
- Choice for LSB/MSB first
- 64×16 -bit input or output FIFO
- Master operation supported between 0.05 and 26 MBit/s, slave operation up to 13 MBit/s
- Timed interrupt to support interrupt of known duration
- Automatic insertion of address bit (required by some LCD having C/D mixed with data)
- Chip select input for slave operation
- Load output function
- Busy input function
- DMA time-out interrupt to allow detection of end of reception

9.27.2 Block diagram

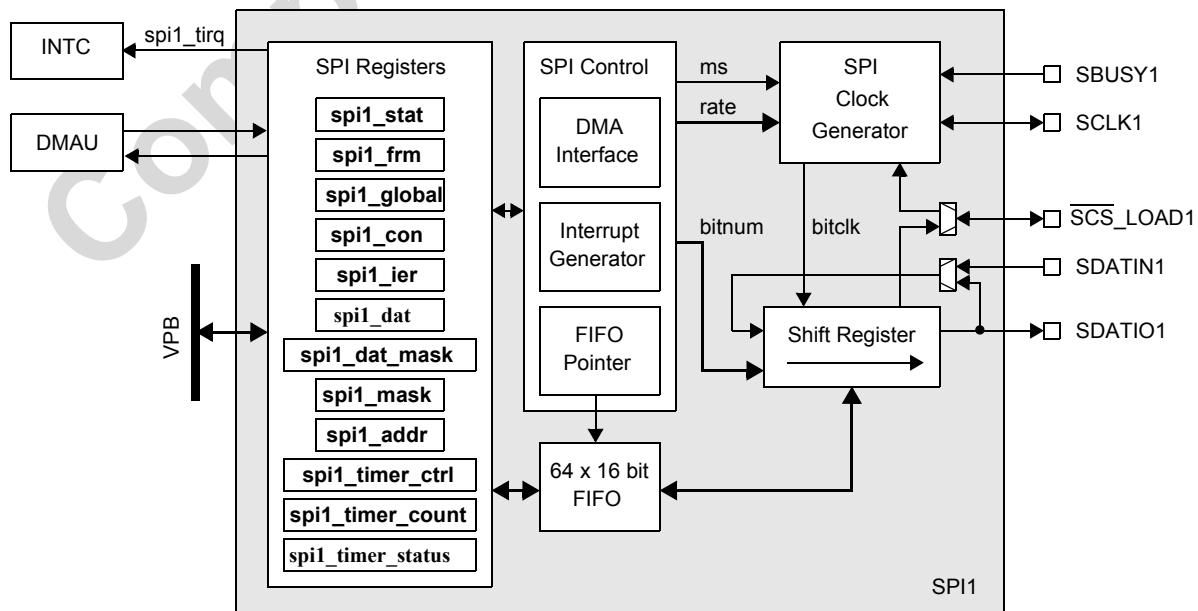


Fig 142.SPI1 block diagram

9.27.3 Hardware interface

Table 398: SPI1 interface pin overview

PIN	Name	I/O	Description
SDATIN1	serial data input	I	SPI serial input
SDATIO1	serial data output	I/O	SPI serial input and output
SCLK1	serial clock	I/O	SPI serial clock
SBUSY1	serial busy	I	SPI busy, also referred as hold
SCS_LOAD1	serial chip select	I/O	SPI slave chip select input, master load data output and load

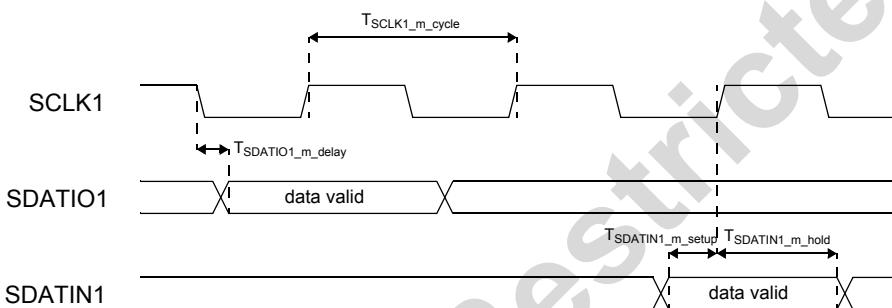


Fig 143.Timing diagrams for SPI mode 0 (master)

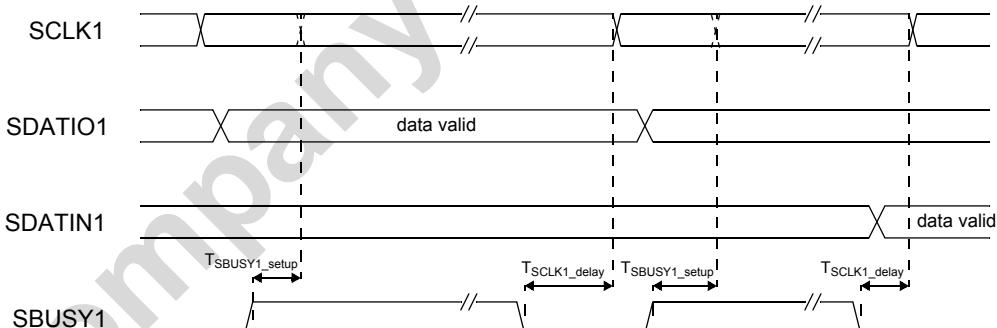


Fig 144.Timing diagrams for the SBUSY signals for SPI mode 0 (master)

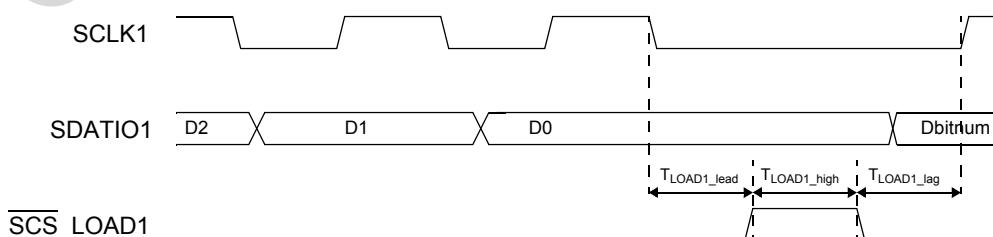


Fig 145.Load signal timing diagram in SPI mode 0 (master)

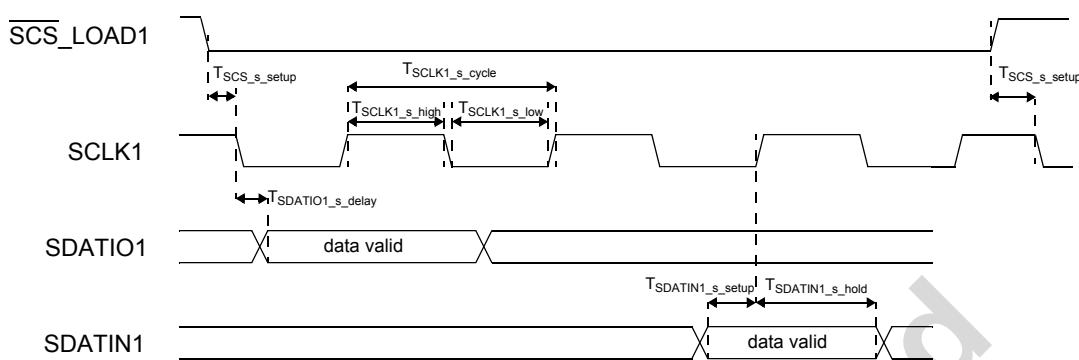


Fig 146.Timing diagrams for SPI mode 0 (slave)

Table 399:AC characteristics of the SPI [1]

Name	Parameter	Min	Typ	Max	Unit
$T_{SCLK1_m_cycle}$	SCLK1 clock period (Master mode, see Table 413 and Table 414)	$2 T_{pclk1}$	-	$256 T_{pclk1}$	ns
$T_{SCLK1_s_cycle}$	SCLK1 clock period (Slave mode, see Table 413 and Table 414)	$8 T_{pclk1}$	-	-	ns
$T_{SCLK1_s_high}$	SCLK1 high level duration (Slave mode)	$1.5 T_{pclk1}$	-	-	ns
$T_{SCLK1_s_low}$	SCLK1 low level duration (Slave mode)	$1.5 T_{pclk1}$	-	-	ns
$T_{SCLK1_m_duty}$	SCLK1 clock duty cycle	-	50	-	%
$T_{SDATIO1_m_delay}$	SDATIO1 data valid after SCLK1 falling edge (Master mode)	-	-	10	ns
$T_{SDATIO1_s_delay}$	SDATIO1 data valid after SCLK1 falling edge (Slave mode)	-	-	$2 T_{pclk1}$	ns
$T_{SDATIN1_m_setup}$	SDATIN1 setup time before SCLK1 rising edge (Master mode)	2	-	-	ns
$T_{SDATIN1_m_hold}$	SDATIN1 hold time after SCKL1 rising edge (Master mode)	2	-	-	ns
$T_{SDATIN1_s_setup}$	SDATIN1 setup time before SCLK1 rising edge (Slave mode)	2	-	-	ns
$T_{SDATIN1_s_hold}$	SDATIN1 hold time after SCKL1 rising edge (Slave mode)	T_{pclk1}	-	-	ns
T_{SBUSY1_setup}	SBUSY1 setup time before SCLK1 edge (in order to not have this edge)	20	-	T_{pclk1}	ns
T_{SCLK1_delay}	SCLK1 delay after SBUSY1 falling edge	-	-	$\frac{T_{sclk1_cycle}}{2}$	ns
$T_{SCS_s_setup}$	$\overline{SCS_LOAD1}$ setup before SCLK active edge (Slave mode)	$2 T_{pclk1}$	-	-	ns
T_{LOAD1_lead}	$\overline{SCS_LOAD1}$ lead time after SCKL1 falling edge	-	$\frac{T_{sclk1_cycle}}{2}$	-	ns
T_{LOAD1_high}	$\overline{SCS_LOAD1}$ high time	-	$\frac{T_{sclk1_cycle}}{2}$	-	ns
T_{LOAD1_lag}	$\overline{SCS_LOAD1}$ lag time before SCLK1 rising edge	-	$\frac{T_{sclk1_cycle}}{2}$	-	ns

[1] T_{pclk1} correspond to AHB/VPB clock cycle duration (e.g. 19 ns for 52 MHz AHB/VPB clock).

[2] The timings in this table are guaranteed by design, correlation, and structural testing. They are not specifically measured in production testing.

9.27.4 Software interface

Table 400: Register overview of the SPI1

Symbol	Name	I/O	reset
spi1_global	SPI global control register	R/W	0x0000
spi1_con	SPI control register	R/W	0x0000 0E08
spi1_frm	SPI frame count register	R/W	0x0000
spi1_iер	SPI interrupt enable register	R/W	0x0000
spi1_stat	SPI status register	R/W	0x0001
spi1_dat	SPI data buffer	R/W	0x0000
spi1_dat_mask	SPI data buffer for using Mask mode	W	0x0000
spi1_mask	SPI mask register	R/W	0x0000
spi1_addr	SPI address register	R/W	0x0000
spi1_timer_ctrl	SPI timer control register	R/W	0x0002
spi1_timer_count	SPI timer counter register	R/W	0x0000
spi1_timer_status	SPI timer status register	R/W	0x0000

Table 401: Register spi1_global

Bit	Symbol	Access	Value	Description
15 to 2	-	R	0*	reserved
1	rst	R/W	0*	reset of the SPI1 excluding the register interface
			1	interface is not reset
			1	interface is reset - is only active if enable bit is set as well
0	enable	R/W	0*	enable of the SPI1 excluding the register interface
			1	interface disabled
				interface enabled

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 402: Register spi1_con

Bit	Symbol	Access	Value	Description
31 to 24	-	R	0*	reserved
23	unidir	R/W	0*	the SPI1 operates with the bidirectional data line SDATIO1
			1	the SPI1 operates with unidirectional input and output pins SDATIN1 and SDATIO1
22	bhalt	R/W	0*	SBUSY1 pin is ignored in master operation
			1	transfer is halted if SBUSY1 is active in master operation
21	bpol	R/W	0*	SBUSY1 is active low
			1	SBUSY1 is active high
20	shft	R/W	0*	no data shift
			1	one bit shift left for data written to spi1_dat_mask
19	msb	R/W	0*	MSB first
			1	LSB first
18	ldins	R/W	0*	load pulse is not inserted. <u>SCS_LOAD</u> is forced low in master operation.
			1	load pulse is inserted in master operation (see Figure 145)
17 to 16	mode[1:0]	R/W	SPI mode	
			0b00*	SPI mode 0
			0b01	SPI mode 1
			0b10	SPI mode 2
			0b11	SPI mode 3
15	rxtx	R/W	0*	data is shifted into the SPI1 (receive) ^[1]
			1	data is shifted out the SPI1 (transmit)
14	thr	R/W	for receive (rxtx = 0):	
			0*	FIFO threshold enabled, threshold = 1 entry in FIFO
			1	FIFO threshold enabled, threshold = 56 entries in FIFO
			for transmit (rxtx = 1):	
			0	FIFO threshold disabled
			1	FIFO threshold enabled, threshold = 8 entries in FIFO
13	shift_off	R/W	0*	enables the generation of clock pulses on SCLK1
			1	disables the generation of clock pulses on SCLK1
12 to 9	bitnum[3:0]	R/W	0b0111*	number of bits to be transmitted or received in one block transfer (transmit or receive operation - the number of shifted bits is bitnum + 1)
8	cs_en	R/W	0*	<u>SCS_LOAD1</u> enabled as chip select in Slave mode
			1	<u>SCS_LOAD1</u> not relevant in Slave mode (assumed always active)
7	ms	R/W	0*	SPI1 is operating as a slave
			1	SPI1 is operating as a master
6 to 0	rate[6:0]	R/W	0b000 1000*	transfer rate - please refer to Equation 4 - this field determines the transfer rate in master mode only - this field has to be programmed also in slave mode to generate bitclk

[1] In order to avoid unexpected DMA request due to FIFO, SPI must be disabled (**spi_global.enable**=0) before changing **rxtx** bit from 1 to 0 (i.e. changing from transmit to receive), then SPI can be enabled again (**spi_global.enable**=1).

Table 403: Register spi1_frm

Bit	Symbol	Access	Value	Description
15 to 0	spif[15:0]	R/W	0*	SPI frame count ; specifies the number of SPI frames which have to be transferred (one frame covers the number of bits specified in field bitnum in register spi1_con); writing a zero to this field, clears the shiftact status flag

[1] The value read back from this register does not reflect the internal status of the spi frame counter. The register should be used for write only.

Table 404: Register spi1_iер

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2	intcs	R/W	0*	SCS_LOAD1 level change interrupt enable
			1	disabled
			1	enabled (only allowed in Slave mode)
1	inteot	R/W	0*	end of transfer interrupt enable
			1	disabled
			1	enabled
0	intthr	R/W	0*	FIFO threshold interrupt enable
			1	disabled
			1	enabled

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 405: Register spi1_stat

Bit	Symbol	Access	Value	Description
15 to 9	-	R	0*	reserved
8	intclr	W	0*	SPI1 interrupt clear - setting this bit clears the SPI1 interrupt and clears the eof flag - writing a zero to this bit has no effect
7	eot ^{[1][2]}	R/W		end of transfer interrupt flag
			0*	end of transfer not reached
			1	end of transfer reached - this bit is either cleared by setting intclr or by writing '0'
6	busylev ^[2]	R/W	~*	SBUSY1 level; this bit shows the level of the SBUSY1 input
5	csl	R/W		edge on <u>SCS_LOAD1</u> that caused the interrupt if css = 1
			0*	falling edge
			1	rising edge
4	css ^{[1][2]}	R/W		chip select state change interrupt flag - this bit needs to be cleared by writing '0'
			0*	level of <u>SCS_LOAD1</u> has not changed
			1	level of <u>SCS_LOAD1</u> has changed in slave mode
3	shiftact ^[2]	R/W	0*	no SPI transfer is ongoing
			1	a SPI data transfer is ongoing
2	bf ^[2]	R/W		FIFO full flag
			0*	FIFO is not full
			1	FIFO is full
1	thr ^{[1][2]}	R/W		FIFO threshold interrupt flag - this bit needs to be cleared by writing '0'
			0*	for rxtx = 0 number of entries in the FIFO is below the threshold
			1	number of entries in the FIFO is equal or above the threshold for rxtx = 1
			0	number of entries in the FIFO is above the threshold
			1	number of entries in the FIFO is equal or below the threshold
0	be ^[2]	R/W		FIFO empty flag
			0	FIFO is not empty
			1*	FIFO is empty

[1] These flags trigger the SPI1 interrupt.

[2] These flags can be forced to get active by the SW - this may trigger the interrupt depending on the current state of the SPI1. As the interrupt condition is not necessarily given by the HW, the flag might get inactive after 2 **pclk1** cycles. However the interrupt remains active.

Table 406: Register spi1_dat

Bit	Symbol	Access	Value	Description
15 to 0	spid[15:0]	R/W	0*	SPI data bits when spi1_dat is read, an entry in the FIFO is accessed when spi1_dat is written, an entry in the FIFO is written; when a data size of less than 16 bits is selected in bitnum , the user has to right-justify data - received data are automatically right-justified

Table 407: Register spi1_dat_mask

Bit	Symbol	Access	Value	Description
15 to 0	spid[15:0]	W	0*	SPI data bits with masking feature - when spi1_dat_mask is written, an entry in the FIFO is written - the data are processed by the mask and by the shift function; when a data size of less than 16 bits is selected in bitnum , the user has to right-justify the data

Table 408: Register spi1_mask

Bit	Symbol	Access	Value	Description
15 to 0	mask[15:0]	R/W	0*	mask field for SPI data bits with masking feature - the mask function is an AND
			0*	bit is cleared
			1	bit is unchanged compared to spi_dat_mask

Table 409: Register spi1_addr

Bit	Symbol	Access	Value	Description
15 to 0	addr[15:0]	R/W	0*	address bit to be added to data bits ; this register may not be used when 16-bit transfers are programmed - bitnum[3:0] = 0b1111

Table 410: Register spi1_timer_ctrl

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2	tirqe	R/W	0*	timed interrupt disabled
			1	timed interrupt enabled ^[1]
1	pirqe	R/W	0	SPI1 status interrupt input disabled
			1*	SPI1 status interrupt input enabled ^[1]
0	mode	R/W	0*	timed Interrupt mode
			1	DMA Time-out mode

[1] Peripheral interrupt and timed interrupt are ORed together when both are enabled. When both **tirqe** and **pirqe** are enabled, **tirqstat** may be used to know if **spi1_tirq** interrupt is from peripheral or from timed interrupt logic

Table 411: Register spi1_timer_count

Bit	Symbol	Access	Value	Description
15 to 0	count[15:0]	R/W	0*	timed interrupt period : writing a value to this register, starts the timed interrupt counter from 0

Table 412: Register spi1_timer_status

Bit	Symbol	Access	Value	Description
15	tirqstat	R/C	0*	timed IRQ status - indicates that the conditions are met to generate an interrupt - write 0 to this bit to clear it
			0*	conditions not met, no timed IRQ pending
			1	conditions met, timed IRQ pending
14 to 7	-	R	0*	reserved
6 to 0	fifodepth[6:0]	R	0*	shows the number of entries currently stored in the FIFO - this field works both in transmit and receive directions; this field is mainly relevant for debugging purposes

9.27.5 Functional description

9.27.5.1 General description

The SPI1 is a 3-wire serial interface designed to interface with a large range of serial memories (SPI mode 0 to 3 compatible slave devices) or with external SPI master devices. The 3-wire serial interface is consisting of a serial data output SDATIO1, a serial data input SDATIN1, a serial clock signal SCLK1 and a chip select input SCS_LOAD1. Both single-master and slave operations are supported by the interface. Alternatively to the two unidirectional data lines, it is possible to program SDATIO1 to be a bidirectional line.

In Master mode, the SCLK1 pin is used as output of the master clock. The master clock is generated by the internal SPI clock generator. The SDATIN1 pin and the SDATIO1 pin are the SPI data I/O-lines. The SCS_LOAD1 pin has a function if the **spi1_con.Idins** bit is set. For each slave device connected to the SPI master, a chip select signal must be generated with a GPIO signal.

In Slave mode, the SCLK1 pin is input for the master clock signal that is generated by the external master. The SDATIN1 pin and the SDATIO1 pin are the SPI data I/O-lines. The chip select input SCS_LOAD1 is used by the external master to select and activate the SPI1. As long as the SCS_LOAD1 signal is HIGH in Slave mode, no data is read from the SDATIN1 and the SDATIO1 data line is in high-impedance state.

In order to use the interface the **spi1_global.enable** bit must be set. With the **spi1_global.rst** bit a SW controlled reset of the SPI interface can be initiated. The reset is only executed if bit **enable** is set as well.

SPI mode 0 to 3 (please refer to [Figure 147](#)) are supported for both master and slave operation. The integrated FIFO allows continuous data transfers up to a programmed number of SPI frames. The frame length can be configured between 1 and 16 bits.

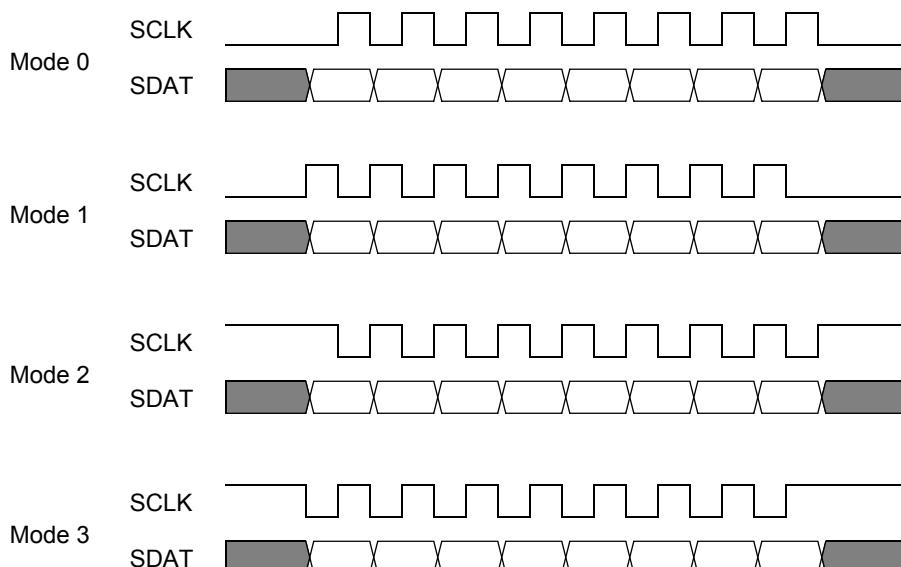


Fig 147.SPI mode 0 to 3 overview

Figure 148 and Figure 149 show some two examples of SPI transfer protocols where a master is requesting a specified data transfer, and then the slave starts to shift out the requested data.

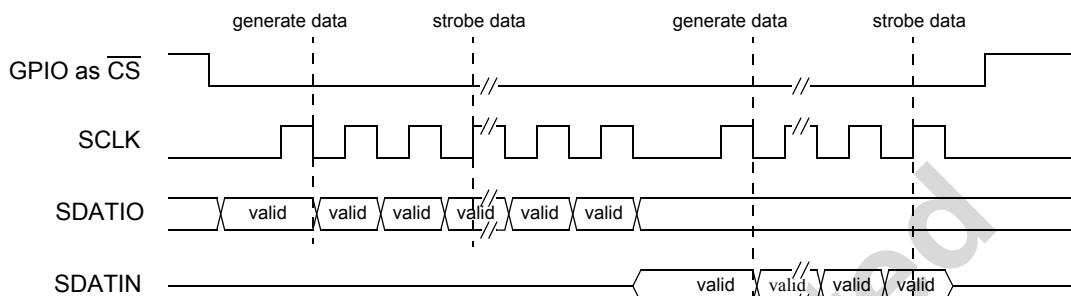


Fig 148.Sample SPI mode 0 master protocol

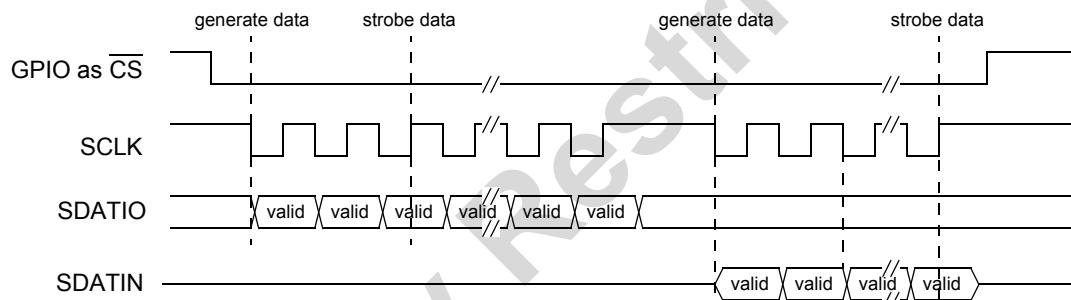


Fig 149.Sample SPI mode 3 master protocol

9.27.5.2 Baud rate definition

The baud rate of the SPI1 is defined by the formula:

$$\text{sclk} = \frac{\text{pclk1}}{(2 \times \text{rate}) + 2} \quad (4)$$

Table 413:SPI transfer rate (XTAL = 13MHz)

Rate selection [1]	Clock division factor	SPI transfer rate [Mbits/s] at pclk1 frequency of				
Rate[6:0]	Decimal	13 MHz	26 MHz	39 MHz	52 MHz	104 MHz
0b0000000	2	6.5 [3]	13 [3]	19.5 [3]	26 [3]	-
0b0000001	4	3.25 [3]	6.5 [3]	9.75 [3]	13 [3]	26 [3]
0b0000010	6	2.17 [3]	4.33 [3]	6.5 [3]	8.67 [3]	17.33 [3]
0b0000011	8	1.625	3.25	4.875	6.5	13
-	-	-	-	-	-	-
0b0011001	52	0.25	0.5	0.75	1	2
-	-	-	-	-	-	-
0b1111111	256	0.051	0.102	0.152	0.203	0.406

[1] The rate field is only relevant for master operation, but has an impact on **bitclk** (timed IRQ/DMA time-out clock).

[2] In slave mode, changing **pclk1** is possible at any time, but maximum slave rate must be kept within these limits.

[3] These rates are not supported in slave operation.

Table 414: SPI transfer rate (XTAL = 10.368/13.824MHz)

Rate selection [1]	Clock division factor	SPI transfer rate [Mbits/s] at pclk1 frequency of					
Rate[6:0]	Decimal	13.824 MHz	27.648 MHz	41.472 MHz	96.768 MHz	103.680 MHz	
0b0000000	2	6.912 [3]	13.824 [3]	20.736 [3]	-	-	
0b0000001	4	3.456 [3]	6.912 [3]	10.368 [3]	24.19 [3]	25.92 [3]	
0b0000010	6	2.17 [3]	4.608 [3]	6.912 [3]	16.13 [3]	17.28 [3]	
0b0000011	8	1.728	3.456	5.184	12.10	12.96	
-	-	-	-	-	-	-	
0b0011001	52	0.2513	0.5027	0.75	1.86	1.99	
-	-	-	-	-	-	-	
0b1111111	256	0.054	0.108	0.152	0.378	0.405	

[1] The rate field is only relevant for master operation, but has an impact on **bitclk** (timed IRQ/DMA time-out clock).

[2] In slave mode, changing **pclk1** is possible at any time, but maximum slave rate must be kept within these limits.

[3] These rates are not supported in slave operation.

9.27.5.3 Slave operation

When the SPI1 device operates in Slave mode, it has to be selected by the master by applying a low level on the **SCS_LOAD1** pin. An interrupt can be generated accordingly by setting the bit **intcs** bit. **intcs** has only effect for Slave mode **ms** = 0. If an interrupt is generated that is caused by a status change on the **SCS_LOAD1** pin, then the status bit **css** is set: in this case the status bit **csi** indicates whether the interrupt was triggered by a falling edge or a rising edge. **csi** has only a meaning while **css** is set.

9.27.5.4 Single frame transfers

Transfers of a single SPI frame can be executed when the **spi1_frm** is set to zero and the FIFO is empty. Data coming from or going to the external device can be accessed via the **spi1_dat** register. This register consists of a separate read and write register. The read part of the register connects to the parallel output of the data shift register; the write part of the register connects to the parallel input of the data shift register. The clock of the shift register is based on the output of the SPI clock generator (Master mode) or by the clock signal on the **SCLK1** pin (Slave mode). The bits of the **spi1_dat** write register are shifted out on **SDATIO1** or the bits on **SDATIN1** are shifted into the shift register (depending on **rxtx** bit).

Writing data in Master mode is done by writing the data to be sent to **spi1_dat** register, which is copied to the shift register and the data is clocked out on **SDATIO1**. **spi1_dat** may be written with new data while the previous data is being shifted out. An end of transfer interrupt is always generated after **bitnum** + 1 **SCLK1** clock cycles if enabled.

To read data when master, the SPI clock pulses can be generated by doing a dummy read or dummy write to **spi1_dat**. The next read operation takes place after **bitnum** + 1 SPI clock cycles and read the actual received data from **spi1_dat**.

A data read sequence can be stopped by setting the **shift_off** bit in the **spi1_con** register. If the SC reads the last data in **spi1_dat** while **shift_off** has been set and no new data has been written to **spi1_dat**, no new sequence of SPI clock pulses is generated. The SCLK1 **shift_off** operation however is not necessary if the SPI1 is already turned off when reading the **spi1_dat** register for the last time.

Writing data in Slave mode is done by writing the data to be sent to **spi1_dat** register. If the shift register is empty because the previous transmit data has been sent out or it is the very first write since the slave was selected, the contents of the **spi1_dat** write register are copied to the shift register. This data is transmitted on the next sequence of SPI clock pulses generated by the external master. **spi1_dat** may be written with new data while the previous data is being shifted out. An end of transfer interrupt is always generated after **bitnum** + 1 SCLK1 clock cycles if enabled.

The data on SDATIN1 are shifted in. So, when the slave has no data to send, still data can be received if the slave is selected. After **bitnum** + 1 clock cycles, the contents of the shift register are copied to the **spi1_dat** read register. At the same time the contents of the **spi1_dat** write register is copied to the shift register again.

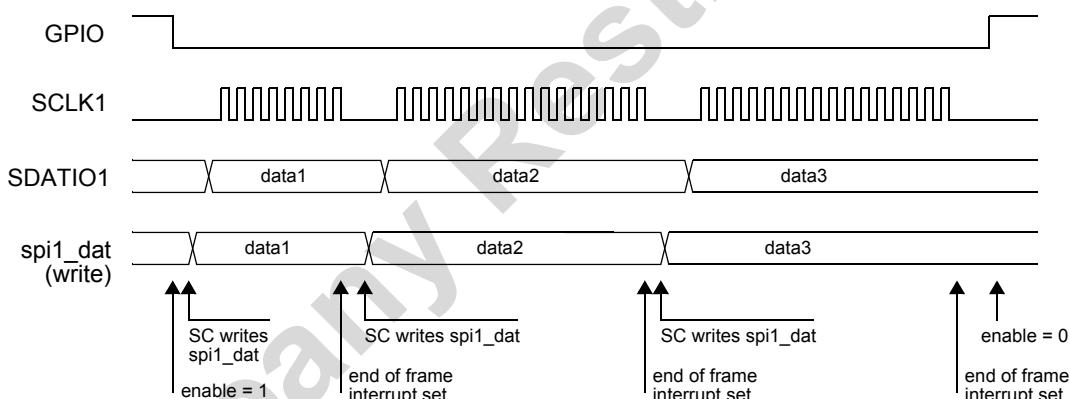


Fig 150.SPI write data frame timing in Master mode

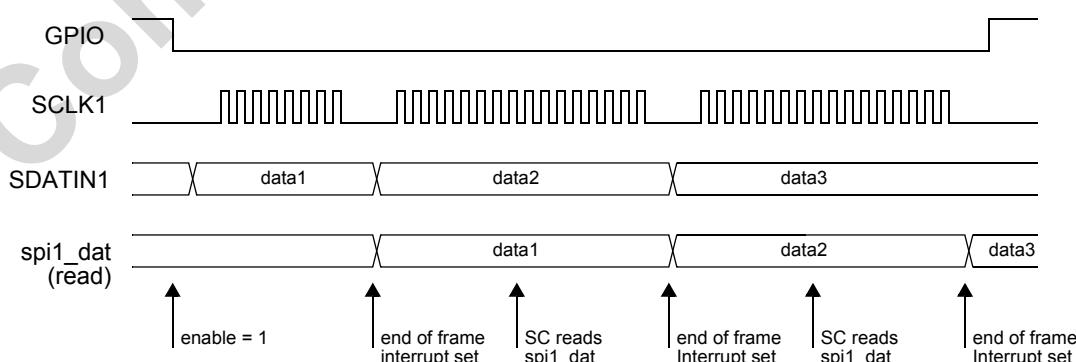


Fig 151.SPI read data frame timing in Master mode

9.27.5.5 Block transfers

A block transfer on the SPIs can be used to transfer e.g. complete pages from/to an external memory. The **spi1_frm** register contains the number of consecutive SPI frames to transfer. The threshold interrupt should be enabled if the number of frames exceeds the FIFO size. The FIFO pointer controls the read or write actions to the FIFO and generates an interrupt request if the number of entries falls below the threshold in Transmit mode or rises above the threshold in Receive mode.

In master operation, the data transfer is only stopped if the receive FIFO is full during Receive mode or the transmit FIFO is empty during Transmit mode. After reading or writing new data, the transfer continues until the **spi1_frm** reaches zero. Then the end of transfer interrupt is asserted.

In slave operation, the user must make sure that no underrun condition (FIFO empty) in transmission and no overrun condition (FIFO full) in reception occurs. The SPI has no means to stop the data transfer in slave operation, hence the master should adapt his transfer speed to the slave.

A block transfer can be initiated as follows.

- Check that the FIFO is empty and no transfer is running (**spi1_stat**)
- Define the SPI frame length (**bitnum**)
- Load the **spi1_frm** register with the number of SPI frames to be executed
- Enable the threshold interrupt if required.

The master can initiate the transfer as follows.

- Read the **spi1_dat** register if a block should be transferred from the external slave to the master
- Write the data word of the first SPI frame to the **spi1_dat** register if a block should be transferred from the chip to the external device - this frame is also included in the value of **spi1_frm**.

Every time a threshold interrupt is asserted the SC has to read or write data from **spi1_dat** to maintain a continuous data transfer on the interface. After the complete block is transferred, the end of transfer interrupt is asserted. If there are still entries in the FIFO, they should be read by the SW.

The status of the block transfer can be checked by reading the **shiftact** bit. This bit is set when the first frame is being transferred. The bit is cleared when all frames defined in **spi1_frm** have been transmitted. Writing a value of zero to **spi1_frm** clears the **shiftact** status flag.

9.27.5.6 DMA mode

During reception (either Master or Slave mode), DMA request are generated if FIFO is not empty.

During transmit (either master or Slave mode), DMA request are generated if FIFO is not full.

In DMA Master Receive mode, in a same way than in Interrupt mode, a first dummy read or write in **spi1_dat** is needed to start the reception (i.e. start the clock generation). After that, DMA transfer request are generated when the FIFO is not empty.

FIFO threshold interrupt must be disabled, and for block transfer, either DMAU terminal count or SPI inteto interrupt can be used.

Note: if **inteto** interrupt is used, DMAU must be configured so that the peripheral is the flow controller (see [Table 202](#)).

Note: in reception, **inteto** interrupt may be asserted while some data remain stored inside the DMAU buffers (see [Table 203](#) 3).

9.27.5.7 Mask and Shift mode

The Mask mode enables the user to mask defined bits in the data field to be transmitted. This is used for frames which have an additional bit for the address. For sending a data in Mask mode both **spi1_mask** and **spi1_addr** registers must be set first. The data has to be written in **spi1_dat_mask** register, instead of **spi1_dat** which bypasses the mask circuitry. The Mask mode can be useful to write commands and data without re-programming the SPI registers.

The Shift mode allows to shift the data written into **spi1_dat_mask** left by one bit. The Shift mode is not possible for data written into **spi1_dat**. When switching between LSB and MSB first, all bits (data and address) are impacted: the address bits are considered as MSB. The Shift mode allows to have the address bit on LSB instead of MSB.

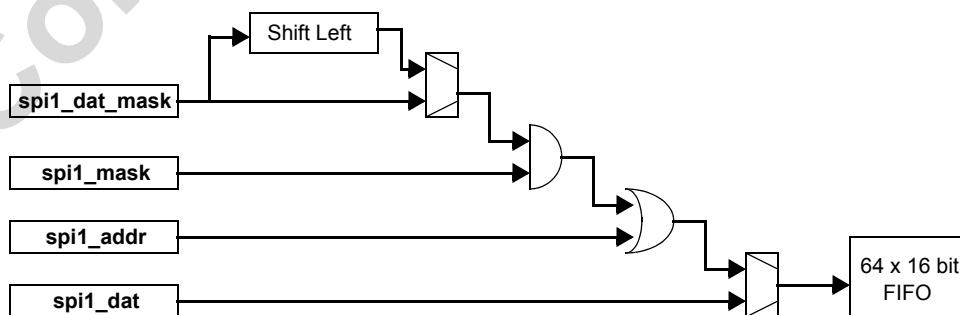


Fig 152. Mask function block diagram

9.27.5.8 Timed interrupt and DMA Time-out modes

The interrupt generator of the SPI1 handles several interrupt sources like the chip select state change interrupt **css**, the FIFO full interrupt **bf**, the FIFO empty interrupt **be**, and the FIFO threshold interrupt **thr**. In addition to these interrupt sources, the SPI may as well generate a timed interrupt based on the SPI1 internal **bitclk** signal (same frequency than SCLK, but generated continuously in Master or Slave mode)

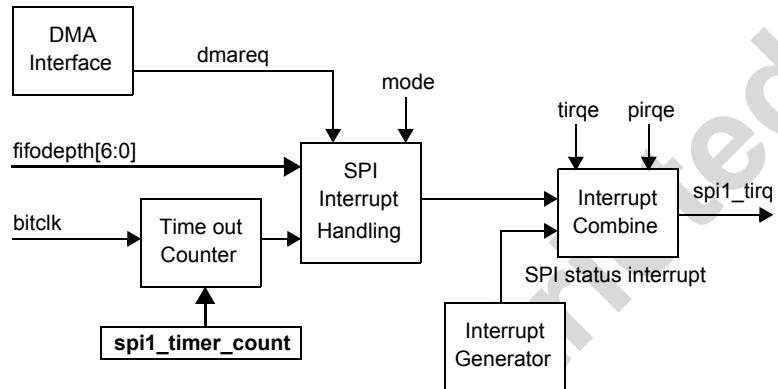


Fig 153.Timed interrupt and DMA time-out block diagram

Timed Interrupt mode

- When **spi1_timer_count** is written, the time-out counter starts from 0
- The time-out counter re-starts from 0 when the value written in **spi1_timer_counter** is reached, generating an output pulse
- If the fifodepth higher than 0 during the rising edge of the counter output pulse, a timed interrupt is generated
- When the SW clears the **tirqstat** bit, the SPI1 interrupt is cleared regardless of counter value.

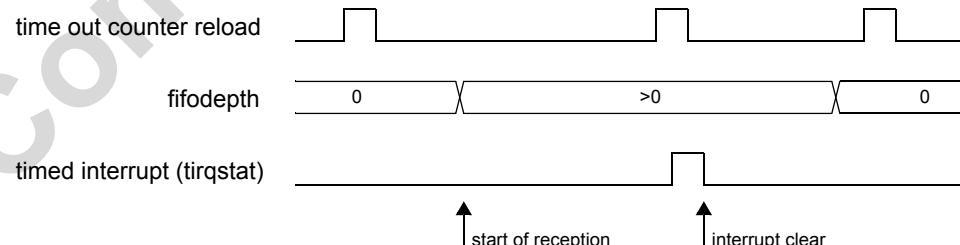
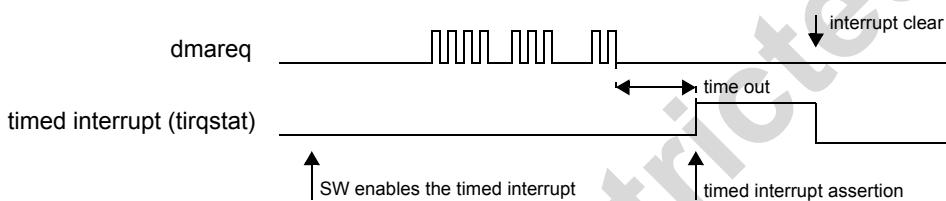


Fig 154.Timed Interrupt mode timing diagram

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

DMA Time-out mode

- When **spi1_timer_count** is written, the time-out counter is enabled, but does not start incrementing
- The time-out counter is launched by the first **dmareq**
- Every rising edge of **dmareq** clears the time out counter
- When the time-out counter reaches the value in **spi1_timer_count**, the timed interrupt is set, and the time-out counter is disabled
- When the SW clears the **tirqstat** bit, the SPI1 interrupt output is cleared regardless of the level of **dmareq**.

**Fig 155.DMA Time-out mode timing diagram****9.27.5.9 Load and busy signals**

This LOAD signal may be necessary for some slave devices to validate or load data. It may be generated after each word including the last word of a transmission.

The SBUSY1 signal may be necessary for some slave devices which need some time to program parts of its memory. During the busy period, the SPI1 stops the generation of SCLK1 pulses. When enabled (**bhalt** = 1), SBUSY1 at active level stalls all master transfers and may be active also before SCS active. When SBUSY1 become active the current transfer is stall regardless the state of the transfer (i.e. the transfer can be stopped in a middle of a word).

9.27.6 Application information

9.27.6.1 List of compatible devices (not a comprehensive view)

Table 415: Devices compatible with the SPI1

Part number	Company	Size [Mbit]	Comment
Serial flash memories			
AT45DB161	Atmel	16	SPI mode 3 recommended
AT45DB321	Atmel	32	SPI mode 3 recommended
FM93C46A	Fairchild	0.001	microwire bus
MMC cards in SPI mode			
SDMB-32	SanDisk	32	SPI mode 0
SDMB-64	SanDisk	64	SPI mode 0
LCD drivers			
BU98500CH-3BW	Rohm	-	D/C selection by 9th bit; 4 MHz maximum operation
FM radio IC			
TEA5767	NXP Semiconductors	-	32 bits serial word, no CS; W/R signal set by GPIO; 1 MHz maximum operation
WLAN SiP			
BGW211	NXP Semiconductors	-	-

[1] Many more external components are compatible with the SPI but only the ones listed in Table 415 are proven to be compatible.

9.27.6.2 Reset support

Devices which feature a reset input can be supported by the connection to a GPIO line.

9.27.6.3 Single-master multiple-slave support

Multiple slave support is given. In this case, multiple chip select output pins are needed to select between the multiple slaves. The application may use GPIO signal for this purpose, making sure that only one chip select output is active at a time.

9.27.6.4 Transfer examples (Simplified view)

Interrupt and DMA request behavior are identical in Master or Slave mode.

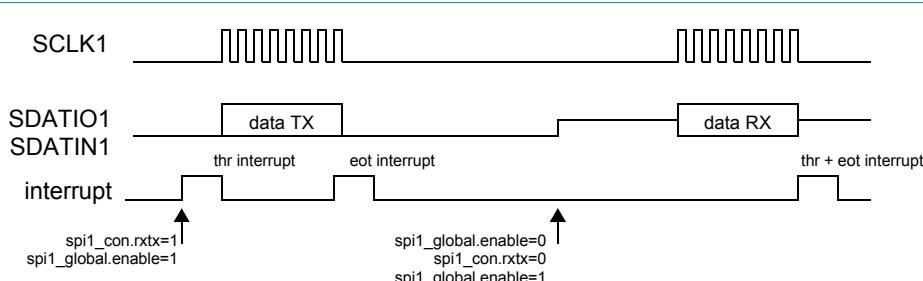


Fig 156.Single transfer, Interrupt mode

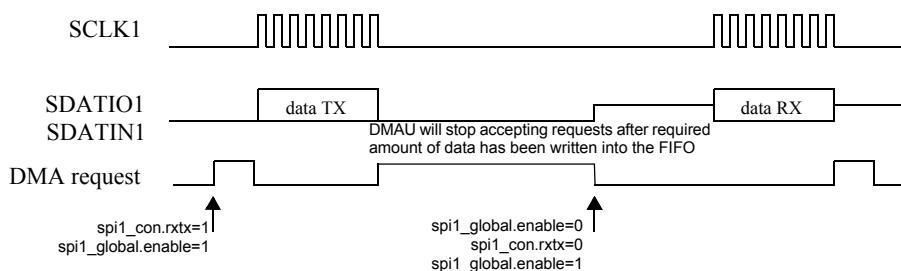


Fig 157.Single transfer, DMA mode

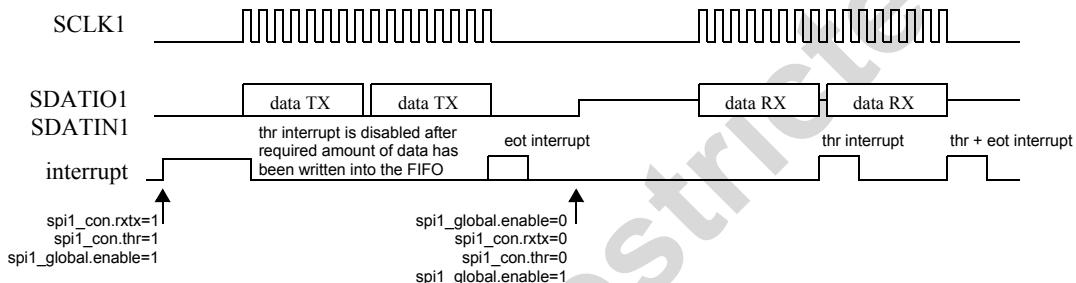


Fig 158.Block transfer, Interrupt mode

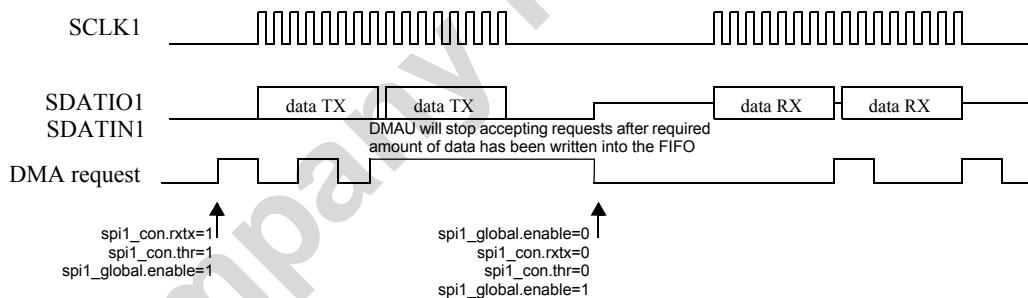


Fig 159.Block transfer, DMA mode

Mask and Shift mode examples

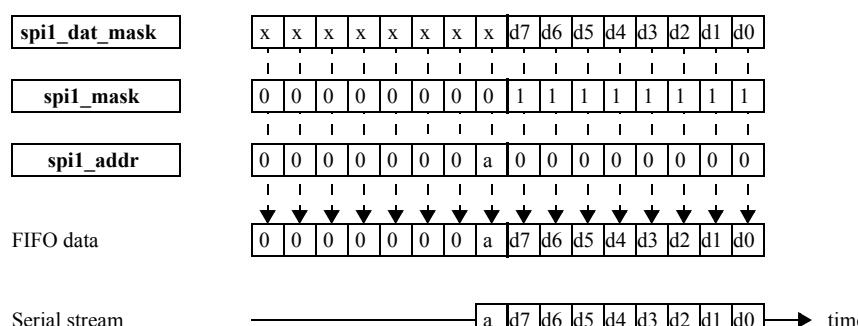


Fig 160.SPI mask example 1: 8-bit data - address bit in 9th position - MSB first - DMAU programmed for 8-bit transfer

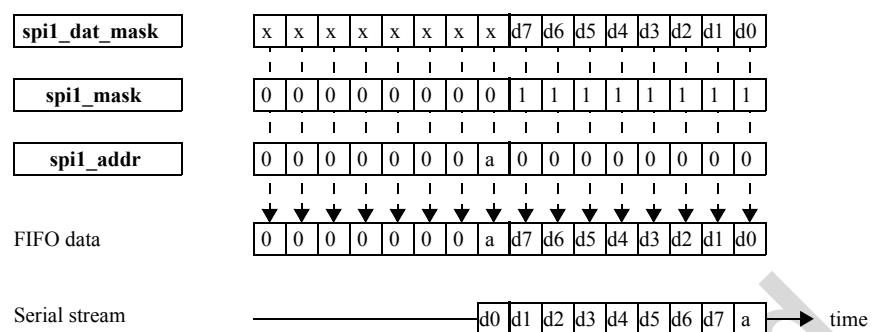


Fig 161.SPI mask example 1: 8-bit data - address bit in 9th position - LSB first - DMAU programmed for 8-bit transfer

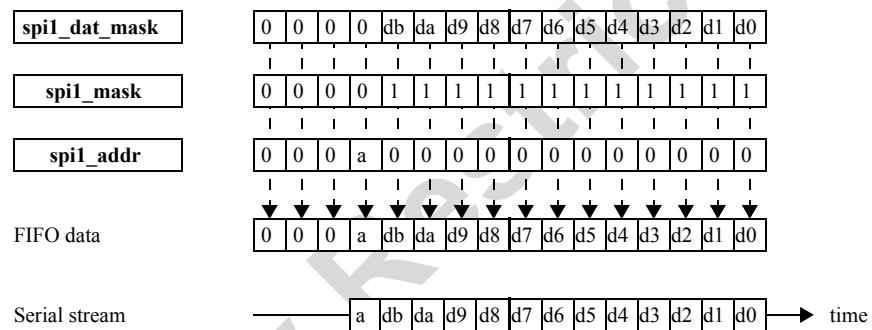


Fig 162.SPI mask example 3: 12-bit data - address bit in 13th position - MSB first - DMAU programmed for 16-bit transfer

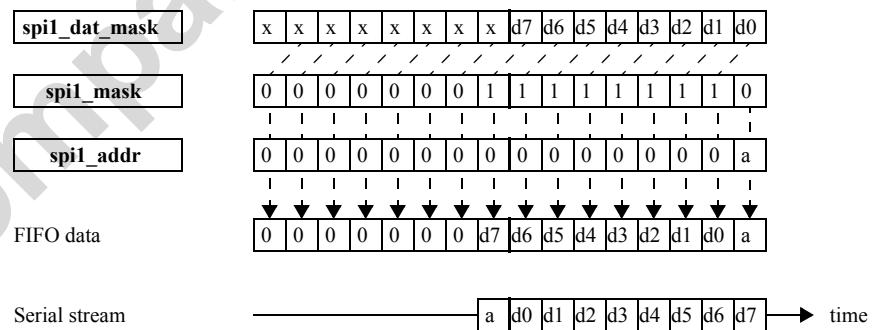


Fig 163.SPI mask example 4: 8-bit data - address bit in 1st position - LSB first - Shift mode - DMAU programmed for 8-bit transfer

9.27.6.5 Atmel mode 0

Atmel serial memories have a special behavior in mode 0: there is an inactive bit period between a transmit and the receive. Atmel mode 0 can be performed with the SPI mode 0 if user programs n + 1 bits to transmit (8 × 8 bits + 1 dummy bit for example) then the reception does not need any special care (including use of DMA).

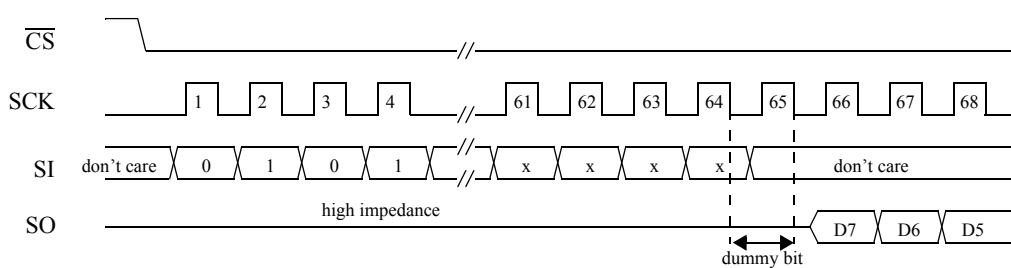


Fig 164. Atmel mode 0 timings (from AT45DB321 specification)

9.27.6.6 ATMEL mode 3

Atmel mode 3 does not have any special behavior.

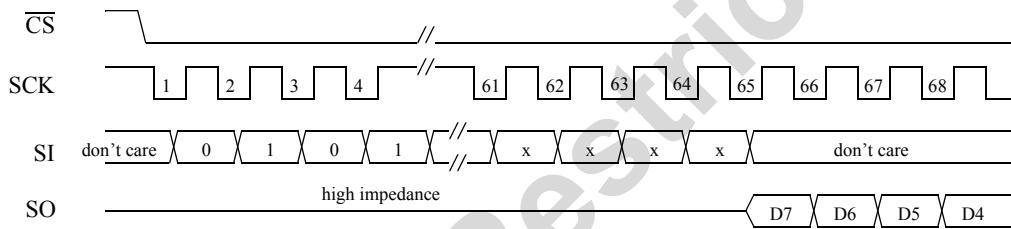


Fig 165. Atmel mode 3 timings (from AT45DB321 specification)

9.27.6.7 Microwire mode support

Microwire mode requires a dummy bit on DO output. Typical solution is to program the first reception to 9 bits, then use 8 bits (with or without DMA) for subsequent data.

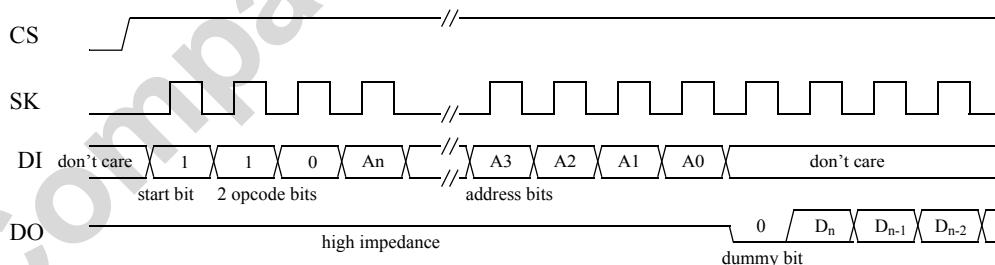


Fig 166. Microwire read timings (from Fairchild FM93C46A specification)

9.28 UART1/UART2 - RS232 Interfaces

9.28.1 Features

Two UARTs are provided in order to allow the ARM access to two RS232 ports. This allows implementation of data services, transfer of program code into flash EEPROM and software debug facilities.

To access the UART registers, the UART must be powered up.

- Fully compatible with industry standard 16C550 and 16C450 from various manufacturers
- RX and TX 64 byte FIFO reduces CPU interrupts
- Full double buffering
- Modem control signals include CTS, RTS, (and DSR, DTR on UART1 only)
Note that on DVFD818x Family the RTS is not supported for UART2.
- Automatic baud rate selection
- Manual or automatic RTS/CTS smart hardware flow control
- Programmable serial characteristics:
 - Baud rate generation (50 to 3.25M baud)
 - 5, 6, 7 or 8-bit characters
 - Even, odd or no-parity bit generation and detection
 - 1, 1.5 or 2 stop bit generation
- Independent control of transmit, receive, line status, data set interrupts and FIFOs
- Full status-reporting capabilities
- SIR-IrDA encoder/decoder optional for both UARTs (2400 to 115k baud)
- Separate DMA signalling for RX and TX
- Timed interrupt to spread receive interrupt on known duration
- DMA time-out interrupt to allow detection of end of reception

9.28.2 Block diagram

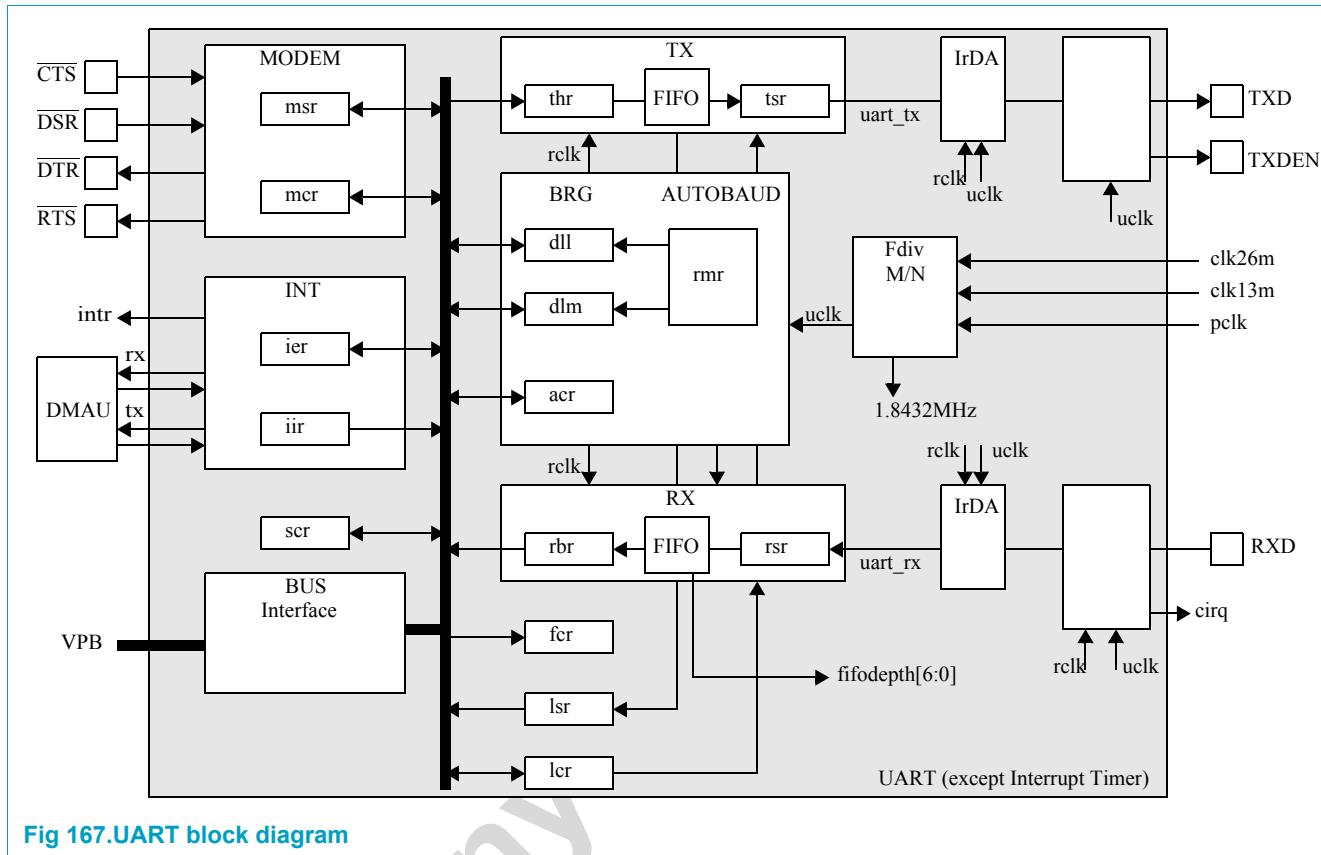


Fig 167.UART block diagram

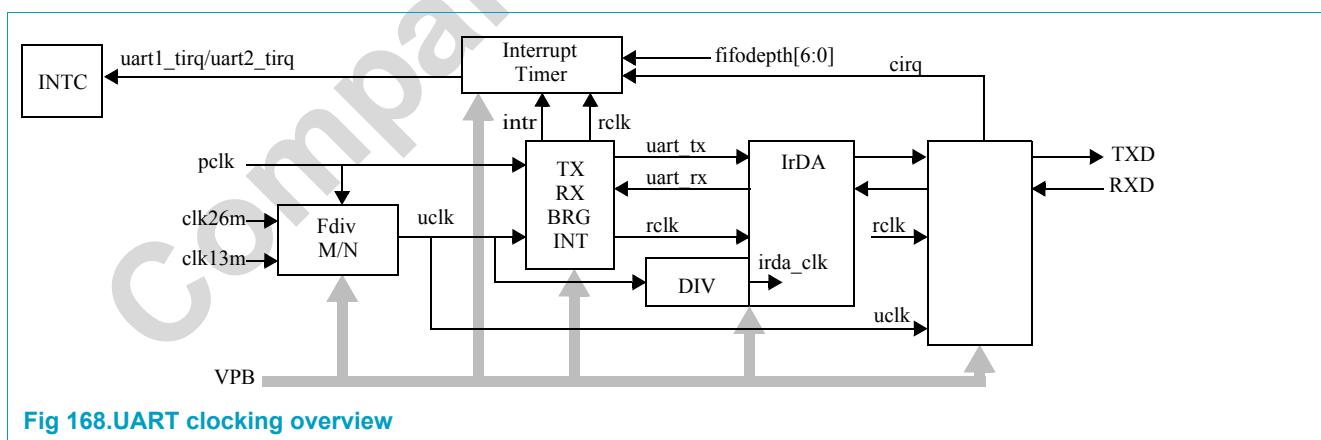


Fig 168.UART clocking overview

9.28.3 Hardware interface

Table 416:UART1 pin overview

PIN	Name	I/O	Description
DTR1	data terminal ready	O	This pin can be set low by writing logic 1 to uart_mcr.dtr bit; this signal is cleared (high) by writing logic 0 to the dtr bit or whenever a master reset occurs; when active (low), the DTR pin indicates that the UART is ready to receive data
RTS1	request to send	O	This pin is set low by writing logic 1 to uart_mcr.rts ; the RTS pin is reset high by writing logic 0 to uart_mcr.rts or by master reset; a low on the RTS pin indicates that the UART has data ready to transmit; used if automatic flow control enabled
CTS1	clear to send	I	The complement of this pin is reflected in the uart_msr.cts bit; a change of state of the CTS pin, since the previous reading of the uart_msr , causes the setting of uart_msr.dcts ; used if automatic flow control enabled; this change is a source for a priority level 4 interrupt, if enabled (uart_iер.modem_ie = 1)
DSR1	data set ready	I	The complement of this pin is reflected in uart_msr.dsrr bit; a change of state of the DSR pin, since the previous reading of the uart_msr , causes the setting of uart_msr.ddsrr ; this change is a source for a priority level 4 interrupt, if enabled (uart_iер.modem_ie = 1)
TXD1	serial data output	O	This is the serial data output from the UART; a mark (1) is logic 1 (high) and space (0) is logic 0 (low); TXD is held in the mark condition when the transmitter is disabled, master reset is true, the transmitter register is empty, or when in the Loop mode. Behavior is different in IrDA mode.
RXD1	serial data input	I	The serial data input moves information from the communication line or modem to the UART receiver circuits; a mark (1) is high, and a space (0) is low; data on serial data input is disabled when operating in the Loop mode. Behavior is different in IrDA mode.

Table 417:UART2 pin overview

PIN	Name	I/O	Description
CTS2	clear to send	I	see Table 416
TXD2	serial data output	O	see Table 416
RXD2	serial data input	I	see Table 416

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.28.4 Software interface

Table 418: Register overview of UART1 / UART2

Symbol	Name	I/O	Reset
uart_rbr ^[1]	receive buffer register	R	0x00
uart_thr ^[1]	transmit holding register	W	-
uart_iер ^[1]	interrupt enable	R/W	0x00
uart_iir	interrupt identification	R	0x01
uart_fcr	FIFO control	W	0x00
uart_lcr	line control	R/W	0x00
uart_mcr	modem control	R/W	0x00
uart_lsr	line status	RaC	0x60
uart_msr	modem status	RaC	0x00
uart_scr ^[1]	scratch pad	R/W	0x00
uart_dll ^[2]	divisor latch LSB	R/W	0x01
uart_dlm ^[2]	divisor latch MSB	R/W	0x00
uart_acr ^[2]	autobaud control	R/W	0x00
uart_timer_ctrl	control behavior of timed IRQ/DMA time-out	R/W	0x02
uart_timer_count	to set timed period of time-out duration	R/W	0x00
uart_timer_status	to know if interrupt is pending	R/W	0x00
uart_irda	to control IrDA signal generation/decoding	R/W	0x00
uart_fdiv_ctrl	fractional divider control	R/W	0x00
uart_fdiv_m	fractional divider M value	R/W	0x0000
uart_fdiv_n	fractional divider N value	R/W	0x0000

[1] Access only with `uart_lcr.dlab` = 0 (see `uart_lcr` description in [Table 425](#)).

[2] Access only with `uart_lcr.dlab` = 1 (see `uart_lcr` description in [Table 425](#)).

9.28.5 UART software interface

Three types of internal registers are used in the UART (control, status and data).

- The control registers are the bit rate select register (divisor latch LSB and divisor latch MSB), line control register, interrupt enable register and the modem control register
- The status registers are the line status registers and the modem status register
- The data registers are the receiver buffer register and the transmitter holding register.

The address, read, and write inputs are used in conjunction with the Divisor Latch Access Bit (**uart_lcr.dlab**) to select the register to be written or read.

The transmitter holding register and receiver buffer register are data registers that hold from five to eight bits of data. If less than eight data bits are transmitted, data is right justified to the LSB.

Bit 0 of a data word is always the first serial bit received and transmitted.

The UART data registers are double-buffered so that read and write operations may be performed when the UART is performing the parallel-to-serial or serial-to-parallel conversion.

The control registers **uart_lcr**, **uart_iер**, **uart_dll** and **uart_dlm**, **uart_mcr** and **uart_fcr** program the serial channel of the UART. These control words define the character length, number of stop bits, parity, baud rate and modem interface. While control-registers can be written in any order, the **uart_iер** should be written last because it controls the interrupt enables. Once the serial channel is programmed and operational, these registers can be updated any time the UART serial channel is not transmitting or receiving data.

It is also important to notice that even if the **uart_iер[3:0]** bits are set to 0, if interruption conditions are met, the interruptions are memorized. This could be very interesting for the software to prevent interruption loss if some of them have to be briefly masked. However, some spurious interrupts may appear the first time the interrupts of the UART are enabled.

For example: if the UART is initialized in Polling mode, if you transmit some characters and wait for a transmission complete, and then, if you enable the TX interrupt (**uart_iер.thre_ie = 1**); The UART will send a TX interrupt because when the UART was working in Polling mode an interrupt condition was met (**thre** empty). This pending interrupt can't be cleared.

Table 419: Register uart1_rbr/uart2_rbr

Bit	Symbol	Access	Value	Description
7 to 0	rbr[7:0]	R	0*	the RBR is the top byte of the Rx FIFO; the top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface; in 450 mode, received data is passed from the Receive Shift Register (RSR) to the top byte of the RBR, essentially producing a 1 byte Rx FIFO; the LSB (bit 0) represents the 'oldest' received data bit; if the character received is less than 8 bits, the unused MSBs are padded with zeroes

Table 420: Register uart1_thr/uart2_thr

Bit	Symbol	Access	Value	Description
7 to 0	thr[7:0]	W	~*	the THR is the top byte of the Tx FIFO; the top byte is the newest character in the Tx FIFO and can be written via the bus interface; in '450 mode, data is passed from the THR to the Transmit Shift Register (TSR), essentially producing a 1 byte Tx FIFO; the LSB represents the first bit to transmit

Table 421: Register uart1_iер/uart2_iер

Bit	Symbol	Access	Value	Description
7 to 4	-	R	0*	reserved
3	modem_ie	R/W	0*	modem status interrupt enable ; enables the modem interrupt; the status of this interrupt can be read from uart_msr[3:0]
			1	disable modem interrupt
			0*	enable modem interrupt
2	rls_ie	R/W	0*	Rx line status interrupt enable ; enables the Rx line status interrupts; the status of this interrupt can be read from uart_lsr[4:1]
			1	disable the Rx line status interrupts
			0*	enable the Rx line status interrupts
1	thre_ie	R/W	0*	THRE interrupt enable ; enables the transmitter holding register empty interrupt; the status of this interrupt can be read from uart_lsr.thre
			1	disable the THRE interrupt
			0*	enable the THRE interrupt
0	rda_ie	R/W	0*	RDA interrupt enable ; enables the receive data available interrupt; the status of this interrupt can be read from uart_lsr.rdr ; in FIFO mode, rda_ie also controls the receive time-out interrupt
			1	disable the RDA interrupt
			0*	enable the RDA interrupt

The interrupt enable register (**uart_iер**) is used to independently enable the serial channel interrupt sources, which activate the interrupt (intr) output. All interrupts are disabled by resetting **uart_iер[3:0]** of the interrupt enable register. Setting the appropriate bits of the **uart_iер** high enables interrupts. Disabling the interrupt system inhibits the Interrupt identification register and the active (high) intr output. All other system functions operate in their normal manner, including the setting of the line status and modem status registers.

Table 422: Register uart1_iir/uart2_iir

Bit	Symbol	Access	Value	Description
7 to 6	fifoe	R		FIFO enable ; these bits are equivalent to uart_fcr.fifo
			0b00*	if uart_fcr fifoe = 0
			0b11	if uart_fcr fifoe = 1
5 to 4	-	R	0*	reserved
3 to 1	iid[2:0]	R		interrupt identification (see Table 423); all other combinations of uart_iir[3:1] not listed above are reserved (0b100, 0b101 and 0b111)
			0b011	1 Receive Line Status (RLS)
			0b010	2a Receive Data Available (RDA)
			0b110	2b Character Time-out Indicator (CTI)
			0b001	3 THRE interrupt
			0b000*	4 modem interrupt
0	ipend	R		interrupt pending (see Table 423); note that ipend is active low; the pending interrupt can be determined by evaluating uart_iir[3:1]
			0	at least one interrupt is pending
			1*	no pending interrupts

Table 423: Interrupt identification handling

IIR Bit				Interrupt set and reset functions			
3	2	1	0	Priority level	Interrupt type	Interrupt source	Interrupt reset control
0	0	0	1	-	none	none	-
0	1	1	0	highest	Rx Line Stat (RLS)	oe, pe, fe, or bi	uart_lsr read
0	1	0	0	second	Rx Data Available (RDA)	receiver data available for 450 or 550 mode, or trigger level reached for FIFO mode	uart_rbr read or FIFO drops below trigger Lvl
1	1	0	0	second	Character Time-out Indication (CTI)	minimum of one character in the RCVR FIFO and no character input or removed during a time period depending on how many characters are in FIFO (numchar) and what the trigger level is set at. value = 3.5 to 4.5 character times remark: the exact time is $[wordlength \times 7 - 2] \times 8 + [(triglevel - numchar) \times 8 + 1] \times rclk$ period	uart_rbr read
0	0	1	0	third	THRE	thre	read of uart_iir or uart_thr write
0	0	0	0	fourth	modem status	cts and dsr	uart_msr read

Interrupts are handled as described in [Table 423](#). Given the status of IIR[3:0] an interrupt handler routine can access how the interrupt occurred and how to clear the active interrupt.

The RLS interrupt (**uart_iir[3:1]** = 0b011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the Rx input: overrun error (**oe**), parity error (**pe**), framing error (**fe**) and break interrupt (**bi**). The Rx error condition that set the interrupt can be observed via **uart_isr[4:1]**. The interrupt is cleared upon an **uart_isr** read.

The **rda** interrupt (**uart_iir[3:1]** = 0b010) shares the second level priority with the CTI interrupt (**uart_iir[3:1]** = 0b110). In 450 mode, the **rda** is set when the **uart_rbr** contains a character and is reset upon an **uart_rbr** read. In 550 mode, the RDA is activated when the Rx FIFO reaches the trigger level defined in **uart_fcr.rxttrig** and is reset when the Rx FIFO depth falls below the trigger level. When the **rda** interrupt goes active in 550 mode, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (**uart_iir[3:1]** = 0b110) is a second level interrupt and is set when the Rx FIFO contains at least one character and no Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any Rx FIFO activity (read or write of **uart_rbr**) will clear the interrupt. This interrupt is intended to flush the **uart_rbr** after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 165 character message and the trigger level was 16 characters, the CPU would receive 10 RDA interrupts, resulting in the transfer of 160 characters, and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

The **thre** interrupt (**uart_iir[3:1]** = 0b001) is a third level interrupt and is activated when the **uart_thr** FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the **uart_thr** FIFO a chance to fill up with data to eliminate many **thre** interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever **thre** = 1 and there have not been at least two characters in the **uart_thr** at one time since the last **thre** = 1 event. This delay is provided to give the CPU time to write data to **uart_thr** without a **thre** interrupt to decode and service. A **thre** interrupt is set immediately if the **uart_thr** FIFO has held two or more characters at one time and currently, the **uart_thr** is empty. The **thre** interrupt is reset when a **uart_thr** write occurs or a read of the **uart_iir** occurs and the **thre** is the active highest interrupt (**uart_IIR[3:1]** = 0b001).

The modem interrupt (**uart_iir[3:1]** = 0b000) is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DSR or CTS. The source of the modem interrupt can be determined by examining **uart_msr[5:3]**. A **uart_msr** read clear the modem interrupt.

The **uart_fcr** controls the operation of the Rx and Tx FIFOs.

Table 424: Register uart1_fcr/uart2_fcr

Bit	Symbol	Access	Value	Description
7 to 6	rxtrig[1:0]	W		Rx trigger level select ; these two bits determine how many received FIFO characters must be written before INTR output pin are activated
			0b00*	trilevel 0 = 1 byte
			0b01	trilevel 1 = 16 bytes
			0b10	trilevel 2 = 32 bytes
			0b11	trilevel 3 = 56 bytes
5 to 3	-	R	0*	reserved
2	txfiforst	WaC	0*	Tx FIFO reset ; writing a logic 1 will clear all bytes in Tx FIFO and reset the pointer logic; this bit is self-clearing
1	rxfiforst	WaC	0*	Rx FIFO reset ; writing a logic 1 will clear all bytes in Rx FIFO and reset the pointer logic; this bit is self-clearing
0	fifoenable	W	0*	FIFO enable ; active high enable for both Rx and Tx FIFOs and uart_fcr[7:1] access; any transition on this bit will automatically clear the FIFOs; when this bit is logic 0, both FIFOs are disabled and a '450 mode is in effect

Company Reference Manual

The **uart_lcr** determines the format of the data character that is to be transmitted or received.

Table 425: Register uart1_lcr/uart2_lcr

Bit	Symbol	Access	Value	Description
7	dlab	R/W		divisor latch access bit (dlab); dlab = 1 is necessary to access following registers: uart_dll, uart_dlm and uart_acr
			0*	disable access to divisor latches (default value)
			1	enable access to divisor latches
6	break	R/W		break control; output pin TXD is forced to logic 0 when break = 1
			0*	disable break transmission (default value)
			1	enable break transmission
5	stick	R/W		stick parity; when parity is enabled (parity = 1), stick = 1 causes the transmission and reception of a parity bit to be in the opposite state from the value of even bit; this forces parity to a known state and allows the receiver to check the parity bit in a known state
			0*	normal parity generation (default value)
			1	forced state parity generation
4	even	R/W		even parity select (used if parity = 1)
			0*	odd parity (default value)
			1	even parity
3	parity	R/W		parity enable
			0*	disable parity generation and checking (default value)
			1	enable parity generation and checking
2	stop	R/W		stop bit select
			0*	1 stop bit (default value)
			1	2 stop bits (1.5 if uart_lcr.length[1:0] = 00 : 5 bits length)
1 to 0	length[1:0]	R/W		character length select
			0b00*	wordlength = 5 bits (default value)
			0b01	wordlength = 6 bits
			0b10	wordlength = 7 bits
			0b11	wordlength = 8 bits

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The **uart_mcr** enables the modem Loopback mode, Auto RTS/CTS mode and controls the modem output.

Table 426: Register uart1_mcr/uart2_mcr

Bit	Symbol	Access	Value	Description
7 to 6	-	R	0*	reserved
5	auto	R/W		<p>RTS/CTS Auto mode. In this mode, the UART controls <u>RTS</u> and <u>CTS</u> directly. In Rx side, when the receive FIFO threshold is reached the interrupt is generated. However, the <u>RTS</u> signal is automatically de-asserted (set high) when another threshold level is reached, 8 (or 7) bytes later. It remains high until receive FIFO is emptied, by reading the received data. This give to the software time to respond to the interrupt without slowing the transfer. If threshold is 56 bytes, the next threshold must be 63 (and not 64) to have provision for one subsequent byte after <u>RTS</u> assertion (see Figure 169). In transmit direction, while <u>CTS</u> is low the transmitter continues until the TX FIFO is emptied. If CTS goes high, the transmitter finishes sending the current byte and then stops sending. The next byte in the TX FIFO is sent when <u>CTS</u> returns low (see Figure 170).</p>
			0*	Normal mode
			1	Auto mode
4	loop	R/W		<p>Loopback mode select. The modem Loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD, has no effect on loopback and output pin, TXD, is held in marking state. The two modem inputs (<u>CTS</u> and <u>DSR</u>) are disconnected externally. Externally, the two modem outputs (<u>RTS</u>, <u>DTR</u>) are set inactive. Internally, the two modem outputs are connected to the two modem inputs. As a result of these connections, the uart_msr is driven by the lower two bits of the uart_mcr (rather than the two modem inputs in Normal mode). This permits modem status interrupts to be generated in Loopback mode by writing the lower two bits of uart_mcr.</p>
			0*	disable modem Loopback mode
			1	enable modem Loopback mode
3 to 2	-	R	0*	reserved
1	rts ^[3]	R/W ^[2]		<p>RTS control; source for modem output pin <u>RTS</u>; this bit contains rts and is inverted at the output to generate <u>RTS</u></p>
			0*	<u>RTS</u> output high (inactive)
			1	<u>RTS</u> output low (active)
0	dtr	R/W ^[2]		<p>DTR control^[1]; source for modem output pin, <u>DTR</u>; this bit contains dtr and is inverted at the output to generate <u>DTR</u></p>
			0*	<u>DTR</u> output high (inactive)
			1	<u>DTR</u> output low (active)

[1] On UART1 only

[2] This bit is write only when loopback is selected (**uart_mcr.loop** = 1). Value can be read back in corresponding **uart_msr** bits (**cts** for **rts** and **dsr** for **dtr**).

[3] Not supported on DVFD8185 and DVFD8187 for UART2

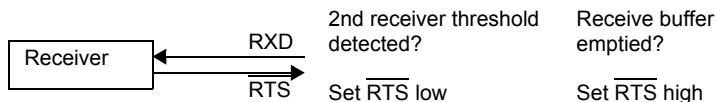


Fig 169.RX auto RTS/CTS

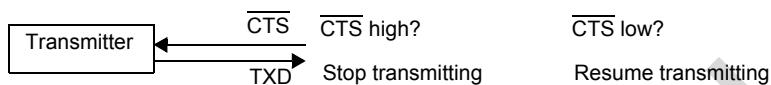


Fig 170.TX auto RTS/CTS

The **uart_lsr** is a read-only register that provides status information on the Tx and Rx blocks.

Table 427: Register **uart1_lsr/uart2_lsr**

Bit	Symbol	Access	Value	Description
7	rxfe	RAC		error in Rx FIFO ^[1]
			0*	uart_rbr contains no Rx errors or uart_fcr.fifo_e = 0
			1	uart_rbr contains at least one Rx error
6	temt	R		transmitter empty ; temt is set when both uart_thr and uart_ts्र are empty; temt is cleared when either the uart_tsր or the uart_thr contain valid data
			0	uart_thr and/or the uart_tsր contain valid data
			1*	uart_thr and the uart_tsր are empty
5	thre	R		transmitter holding register empty ^[2] ; thre is set immediately upon detection of an empty uart_thr and is cleared on a uart_thr write
			0	uart_thr contains valid data
			1*	uart_thr is empty
4	bi	RAC		break interrupt ^[3] ; when RXD is held in the spacing state (all 0s) for one full character transmission (start, data, parity and stop), a break interrupt occurs; once the break condition has been detected, the receiver goes idle until RXD goes to marking state (all 1s); an uart_lsr read clears bi
			0*	break interrupt status is non-active
			1	break interrupt status is active
3	fe	RAC		framing error ^[4] ; when the stop bit of a received character is a logic 0, a framing error occurs; an uart_lsr read clears fe ; upon detection of a framing error, the Rx will attempt to re-synchronize to the data and assume that the bad stop bit is actually an early start bit
			0*	framing error status is non-active
			1	framing error status is active
2	pe	RAC		parity error ^[5] ; when the parity bit of a received character is in the wrong state, a parity error occurs; an uart_lsr read clears pe
			0*	parity error status is non-active
			1	parity error status is active

Table 427: Register uart1_lsr/uart2_lsr...continued

Bit	Symbol	Access	Value	Description
1	oe	RAC		overrun error ^[6] ; the overrun error condition is set as soon as it occurs; an uart_lsr read clears oe
			0*	overrun error status is non-active
			1	overrun error status is active
0	rdr	R		receiver data ready ^[7]
			0*	uart_rbr is empty
			1	uart_rbr contains valid data

- [1] In 450 mode (**uart_fcr.fifoe** = 0), **rxfe** is always 0. In 550 mode (**uart_fcr.fifoe** = 1), **rxfe** is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the **uart_rbr**. This bit is cleared when the **uart_lsr** register is read and there are no subsequent errors in the FIFO.
- [2] In 450 mode (**uart_fcr.fifoe** = 0), the **uart_thr** is a 1 byte FIFO. In 550 mode (**uart_fcr.fifoe** = 1), the **uart_thr** is a 64 bytes FIFO.
- [3] The time of break detection is dependent on **uart_fcr.fifoe**. In 450 mode (**uart_fcr.fifoe** = 0), the break interrupt is associated with the character that has just been received in the **uart_rsr**. In 550 mode (**uart_fcr.fifoe** = 1), the break interrupt is associated with the character being read from the **uart_rbr** FIFO.
- [4] The time of the framing error detection is dependent on **uart_fcr.fifoe**. In 450 mode (**uart_fcr.fifoe** = 0), a framing error is associated with the character that has just been received in the RSR. In 550 mode (**uart_fcr.fifoe** = 1), a framing error is associated with the character being read from the **uart_rbr** FIFO.
- [5] Time of parity error detection is dependent on **uart_fcr.fifoe**. In 450 mode (**uart_fcr.fifoe** = 0), a parity error is associated with the character that has been received by the **uart_rsr**. In 550 mode (**uart_fcr.fifoe** = 1), a parity error is associated with the character being read from the **uart_rbr** FIFO.
- [6] In 450 mode (**uart_fcr.fifoe** = 0), **oe** is set when **uart_rsr** has a new character assembled and the **uart_rbr** has not been read. In this case, the **uart_rbr** is overwritten with **uart_rsr** and previous contents of **uart_rbr** is lost. In 550 mode (**uart_fcr.fifoe** = 1), OE is set when **uart_rsr** has a new character assembled and the **uart_rbr** FIFO is full. In this case, the **uart_rbr** FIFO will not be over-written and the character in the **uart_rsr** is lost. Note that this is opposite the 450 mode operation.
- [7] In 450 mode (**uart_fcr.fifoe** = 0), **uart_rdr** is set when **uart_rbr** is loaded with a new character and cleared after an **uart_rbr** read operation. In 550 mode (**uart_fcr.fifoe** = 1), **uart_rdr** is set when the **uart_rbr** holds an unread character and is cleared when the **uart_rbr** FIFO is empty.

The **uart_msr** is a read-only register that provides status information on the input of modem signals.

Table 428: Register uart1_msr/uart2_msr

Bit	Symbol	Access	Value	Description
7 to 6	-	R	0*	reserved
5	dsr	R	~*	Data Set Ready state (dsr) ^[1] ; complement of input signal \overline{DSR} ; if the serial channel is in Loop mode (uart_mcr.loop = 1), dsr reflects the value of uart_mcr.dtr
4	cts	R	~*	Clear To send State (cts) ; complement of input signal \overline{CTS} ; if the serial channel is in Loop mode (uart_mcr.loop = 1), cts reflects the value of uart_mcr.rts
3 to 2	-	R	0*	reserved
1	ddsr	RAC		delta dsr ^[1] ; set upon state change of input \overline{DSR} ; cleared after an uart_msr read
			0*	no change detected on modem input, \overline{DSR}
			1	state change detected on modem input, \overline{DSR}
0	dcts	RAC		delta cts ; set upon state change of input \overline{CTS} ; cleared after an uart_msr read
			0*	no change detected on modem input, \overline{CTS}
			1	state change detected on modem input, \overline{CTS}

[1] On UART1 only

The **uart_msr** provides status of the modem-input lines from the modem or peripheral devices. The **uart_msr** allows to read the serial channel modem signal inputs, in addition to the current status information, two bits of the **uart_msr** indicate whether the modem inputs have changed since the last reading of the **uart_msr**. The delta status bits are set high when control input from the modem changed state, and reset low when software reads the **uart_msr**.

The modem-input lines are **CTS**, **DSR** (on UART1 only). **uart_msr.cts** and **uart_msr.dsr** are status indications of these lines. A status bit equal to 1 indicates the input is low. A status bit equal to 0 indicates the input is high. If the modem status interrupt in the Interrupt enable register is enabled (**uart_iер.modem_ie**), an interrupt is generated whenever **dcts** or **ddsr** is set to a one. The **uart_msr** is a priority 4 interrupt. The contents of the modem status register are described in [Table 428](#). Reading the **uart_msr** register will clear the delta status indications but has no effect on the other status bits. For **uart_lsr** and **uart_msr**, the setting of status bits is inhibited during status register read operations. If a status condition is generated during a read operation, the status bit is not set until the trailing edge of the read. If a status bit is already set during a read operation, and the same status condition occurs, that status bit is cleared at the trailing edge of the read instead of being set again.

The **uart_scr** has no effect on the UART operation. This register can be written and/or read at user's discretion. Access is shared with **uart_acr**. **uart_scr** is accessed only when **dlab** = 0 otherwise **uart_acr** is accessed.

Table 429: Register uart1_scr/uart2_scr

Bit	Symbol	Access	Value	Description
7 to 0	scr[7:0]	R/W	0*	scratch pad data

Table 430: Register uart1_dll/uart2_dll

Bit	Symbol	Access	Value	Description
7 to 0	dll[7:0]	R/W	1*	divisor latch LSB bits; value altered after autobaud detection

Table 431: Register uart1_dlm/uart2_dlm

Bit	Symbol	Access	Value	Description
7 to 0	dlm[7:0]	R/W	0*	divisor latch MSB bits; value altered after autobaud detection

Table 432: Baud rates

uclk	7.3728 MHz		14.7456 MHz		29.4912 MHz		32 MHz		52 MHz	
	Baud rate	Divisor	Error	Divisor	Error	Divisor	Error	Divisor	Error	Divisor
50	9216	0.000%	18432	0.000%	36864	0.000%	-	-	-	-
75	6144	0.000%	12288	0.000%	24576	0.000%	-	-	-	-
110	4189	0.002%	8378	0.002%	16756	0.002%	-	-	-	-
134.5	3426	0.001%	6852	0.001%	13704	0.001%	-	-	-	-
150	3072	0.000%	6144	0.000%	12288	0.000%	-	-	-	-
300	1536	0.000%	3072	0.000%	6144	0.000%	-	-	-	-
600	768	0.000%	1536	0.000%	3072	0.000%	-	-	-	-

Table 432: Baud rates...continued

uclk	7.3728 MHz		14.7456 MHz		29.4912 MHz		32 MHz		52 MHz	
Baud rate	Divisor	Error	Divisor	Error	Divisor	Error	Divisor	Error	Divisor	Error
1200	384	0.000%	768	0.000%	1536	0.000%	-	-	-	-
1800	256	0.000%	512	0.000%	1024	0.000%	-	-	-	-
2000	230	0.174%	461	-0.043%	922	-0.043%	-	-	-	-
2400	192	0.000%	384	0.000%	768	0.000%	-	-	-	-
3600	128	0.000%	256	0.000%	512	0.000%	-	-	-	-
4800	96	0.000%	192	0.000%	384	0.000%	-	-	-	-
7200	64	0.000%	128	0.000%	256	0.000%	-	-	-	-
9600	48	0.000%	96	0.000%	192	0.000%	-	-	-	-
19200	24	0.000%	48	0.000%	96	0.000%	-	-	-	-
38400	12	0.000%	24	0.000%	48	0.000%	-	-	-	-
57600	8	0.000%	16	0.000%	32	0.000%	-	-	-	-
115200	4	0.000%	8	0.000%	16	0.000%	-	-	-	-
230400	2	0.000%	4	0.000%	8	0.000%	-	-	-	-
460800	1	0.000%	2	0.000%	4	0.000%	-	-	-	-
921600	-	-	1	0.000%	2	0.000%	-	-	-	-
1 843 200	-	-	-	-	1	0.000%	-	-	-	-
2 000 000	-	-	-	-	-	-	1	0.000%	-	-
3 250 000	-	-	-	-	-	-	-	-	1	0.000%

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 433: Register uart1_acr / uart2_acr

Bit	Symbol	Access	Value	Description
7	-	R/W	0*	reserved
6	auto_restart	R/W		automatic restart mode
			0*	no automatic restart in case of autobaud timeout
			1	automatic autobaud restart at the next falling edge in case of timeout (identical to a write with start=1) - uart_rmr count start on next falling edge on RXD
5	uato	R/C		autobaud time-out status
			0*	no autobaud time-out has occurred
			1	autobaud counter uart_rmr has rolled over during autobaud measurement - an interrupt is generated if uato_ie is set - autobaud mechanism is either stopped (uart_dlm and uart_dll are unchanged) or restarted depending of auto_restart value
4	uato_ie	R/W		autobaud timeout interrupt enable
			0*	interrupt disabled
			1	interrupt enabled
3	ueoa	R/C		end of autobaud status
			0*	autobaud not finished or unsuccessfully finished
			1	autobaud ended successfully - an interrupt is generated if ueoa_ie is set
2	ueoa_ie	R/W		end of autobaud interrupt enable
			0*	interrupt disabled
			1	interrupt enabled
1	mode	R/W		autobaud measurement mode
			0*	measurement is made between two subsequent falling edges of RXD - the uart_rmr is 14-bit wide
			1	measurement is made between falling and rising edge of RXD - in this case, the uart_rmr value is doubled when transferred to uart_dll and uart_dlm and is 15-bit wide
0	acr	R/W		autobaud flag - this bit is automatically reset after autobaud is complete - the manual clearing of this bit before autobaud end, stops the measurement and uart_dll and uart_dlm are unchanged
			0*	autobaud stopped
			1	start the autobaud process

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.28.6 Frac-N software interface

Table 434: Register uart1_fdiv_ctrl/uart2_fdiv_ctrl

Bit	Symbol	Access	Value	Description
7	fdive	R/W		fractional divider enable
			0*	frac-N not used (bypass)
			1	frac-N used
6 to 2	-	R	0*	reserved
1 to 0	clksel[1:0]	R/W		fractional divider input clock select (fclk)
			0b00*	fclk = pclk1; pclk1 frequency switching in CGU interfere with UART baud rate; maximum uclk frequency is limited to pclk1 frequency
			0b01 [1]	fclk = clk13m; pclk1 frequency switching in CGU allowed without care; maximum uclk frequency is limited to 13 MHz (XTAL = 13MHz) and 13.824MHz (XTAL = 10.368 or 13.824MHz) with this selection
			0b10 [2]	fclk = clk26m; pclk1 frequency switching in CGU allowed without care; maximum uclk frequency is limited to 26 MHz (XTAL = 13MHz) and 27.648MHz (XTAL = 10.368 or 13.824MHz) with this selection
			0b11	reserved

[1] For properly operation **pclk1 ≥ clk13m** is required

[2] For properly operation **pclk1 ≥ clk26m** is required

Table 435: Register uart1_fdiv_m/uart2_fdiv_m

Bit	Symbol	Access	Value	Description
15 to 0	fdivm[15:0]	R/W	0*	m value of fractional divider

Table 436: Register uart1_fdiv_n/uart2_fdiv_n

Bit	Symbol	Access	Value	Description
15 to 0	fdivn[15:0]	R/W	0*	n value of fractional divider

The resulting UART baud rate is calculated by the equation

$$\text{baudrate} = \frac{\text{fdivn}}{\text{fdivm}} \times \frac{\text{fclk}}{16}$$

Table 437: Fractional-n useful values for a system clock of clk13m = 13MHz^[1]

fclk ^[2] (MHz)	uclk (MHz)	Max baud rate (kbits/s)	fdivm	fdivn
104	52	3250	0xBE6E	0x5F37
104	32	2000	0x5F37	0x1D4C
104	29.4912	1843.2	0x5F37	0x1B00
52	52	3250	0x5F37	0x5F37
52	32	2000	0x5F37	0x3A98
52	29.4912	1843.2	0x5F37	0x3600
52	14.7456	921.6	0x5F37	0x1B00
52	7.3728	460.8	0x5F37	0x0D80
52	3.6864	230.4	0x5F37	0x06C0
52	1.8432	115.2	0x5F37	0x0360
26^[3]	14.7456	921.6	0x5F37	0x3600
26	7.3728	460.8	0x5F37	0x1B00
26	3.6864	230.4	0x5F37	0x0D80
26	1.8432	115.2	0x5F37	0x06C0
13^[4]	7.3728	460.8	0x5F37	0x3600
13	3.6864	230.4	0x5F37	0x1B00
13	1.8432	115.2	0x5F37	0x0D80

[1] Recommend values are in bold.

[2] See [Table 434](#).

[3] Configuration should be used if fclk = clk26m is selected (see [Table 434](#))

[4] Configuration should be used if fclk = clk13m is selected (see [Table 434](#))

Table 438: Fractional-n useful values for a system clock of clk13m = 13.824MHz^[1]

fclk ^[2] (MHz)	uclk (MHz)	Max baud rate (kbits/s)	fdivm	fdivn
103.68	51.84	3240	0xBE6E	0x5F37
103.68	32	2000	0x5F37	0x1D63
103.68	29.4912	1843.2	0x5F37	0x1B15
55.296	55.296	3456	0x5F37	0x5F37
55.296	32	2000	0x5F37	0x3719
55.296	29.4912	1843.2	0x5F37	0x32C8
55.296	14.7456	921.6	0x5F37	0x1964
55.296	7.3728	460.8	0x5F37	0x0CB2
55.296	3.6864	230.4	0x5F37	0x0659
55.296	1.8432	115.2	0x5F37	0x032C
27.648^[3]	14.7456	921.6	0x5F37	0x32C8
27.648	7.3728	460.8	0x5F37	0x1964
27.648	3.6864	230.4	0x5F37	0x0CB2
27.648	1.8432	115.2	0x5F37	0x0659
13.824^[4]	7.3728	460.8	0x5F37	0x32C8
13.824	3.6864	230.4	0x5F37	0x1964
13.824	1.8432	115.2	0x5F37	0x0CB2

[1] Recommend values are in bold.

[2] See [Table 434](#).

[3] Configuration should be used if fclk = clk26m is selected (see [Table 434](#))

[4] Configuration should be used if fclk = clk13m is selected (see [Table 434](#))

9.28.7 IrDA software interface

Table 439: Register uart1_irda/uart2_irda

Bit	Symbol	Access	Value	Description
7 to 6	-	R/W	0*	reserved
5 to 3	irdaclkdiv[2:0]	R/W		IrDA clock divider ; other values reserved irda_clk must be fixed at 1.8432 MHz and is derived from uclk
			0b000*	1 for uclk = 1.8432 MHz
			0b001	2 for uclk = 3.6864 MHz
			0b010	4 for uclk = 7.3728 MHz
			0b011	8 for uclk = 14.7456 MHz
			0b100	16 for uclk = 29.4912 MHz
2	invert	R/W		receive data
			0*	not inverted
			1	inverted
1	width	R/W		transmit pulse width
			0*	variable width ($\frac{3}{16}$ bit)
			1	fix 1.6 μ s
0	irdae	R/W	0*	normal UART connection
			1	IrDA enable for UART

Table 440: IrDA clock settings

uclk (MHz)	Max UART rate (kbits/s) ^[1]	IrDA clk divider
1.8432	115.2	1
3.6864	230.4	2
7.3728	460.8	4
14.7456	921.6	8
29.4912	1843.2	16

[1] SIR-IrDA is limited to 115.2 kbits/s

9.28.8 Timed IRQ/DMA time-out software interface

Table 441: Register uart1_timer_ctrl/uart2_timer_ctrl

Bit	Symbol	Access	Value	Description
15 to 3	-	R	0*	reserved
2	tirqe	R/W	0*	timed IRQ enable
			1	timed IRQ disabled
			1	timed IRQ enabled ^[1]
1	pirqe	R/W	0	peripheral IRQ input enable
			0	UART data/status interrupt disabled
			1*	UART data/status interrupt enabled ^[1]
0	mode	R/W	0*	timed IRQ
			1	DMA time-out

[1] Enabled peripheral interrupt and timed interrupt are ORed together when more than one are enabled. When more than one **tirqe** and **pirqe** are enabled, **tirqstat** must be used to know if **uart1_tirq/uart2_tirq** interrupt is from peripheral from timed interrupt logic

Table 442: Register uart1_timer_count/uart2_timer_count

Bit	Symbol	Access	Value	Description
15 to 0	count[15:0]	R/W	0*	timed interrupt period ; writing to this register start counting from 0

Table 443: Register uart1_timer_status/uart2_timer_status

Bit	Symbol	Access	Value	Description
15	tirqstat	R/C	0*	timed IRQ status - indicates that the conditions are met to generate an interrupt - write 0 to this bit to clear it
			0*	conditions not met, no timed IRQ pending
			1	conditions met, timed IRQ pending
14 to 7	-	R	0*	reserved
6 to 0	fifodepth[6:0]	R	0*	receive FIFO depth value (for debug only)

9.28.9 UART functional description

The device integrates two UARTs, providing the ARM SC with two external serial interfaces.

The UART serves as input/output serial data interface, performing serial-to-parallel conversion on data characters received from peripheral devices or modems, and parallel-to-serial conversion on data characters transmitted by the ARM SC. In the FIFO mode, FIFOs are enabled permitting 64 bytes to be stored in both Transmit and Receive mode. Receive FIFO also provides three bits per byte of error status. The ARM SC can read the complete status of the UART at any time during functional operation. The information obtained includes the type and condition of the transfer operations being performed, and error conditions involving parity, overrun, framing or break interrupt.

9.28.9.1 Divisor latches

The UART serial channel contains a programmable Baud Rate Generator (BRG) that divides the UART clock by any divisor from 1 to 216-1. The output of the baud rate generator (NBAUDOUT) is referred to as **rclk**. The frequency of **rclk** is 16 times the data rate. The desired divisor number is calculated through use of the following equation:

$$\text{Divisor} = \frac{\text{uclk}}{(\text{baud rate} \times 16)} \quad (5)$$

The DLL and DLM registers together form a 16-bit divisor where DLL contains the lower 8 bits of the divisor and DLM contains the higher 8 bits of the divisor. These divisor latch register must be loaded during initialization. Upon loading either of the Divisor Latches, a 16-bit baud counter is immediately loaded. This prevents long counts on initial load. A 'h0000 value is treated like a 'h0001 value as division by zero is not allowed.

These registers are automatically loaded by result of the autobaud calculation.

UART is clocked by VPB pclk1 divided by Frac-N, which gives uclk.

IrDA is clocked by uclk divided by N (1.8432 MHz) and **rclk** (depend on baud rate).

9.28.9.2 DMA requests

DMAREQ_RX signal is activated when there is at least one character in the receiver FIFO or receiver holding register.

DMAREQ_TX signal is activated whenever there is room for a character in transmit FIFO (as indicated by the thre interrupt).

DMA requests are activated regardless the interrupt setting in **uart_iер**, but to avoid useless interruptions, when the reception (and/or the transmission) is made under DMA control, the bit **uart_iер.rda_ie** (and/or respectively **uart_ie.thre_ie**) must be cleared.

9.28.9.3 Autobaud

The measurement provides support for implementing autobauding procedure based on the 'AT'-protocol.

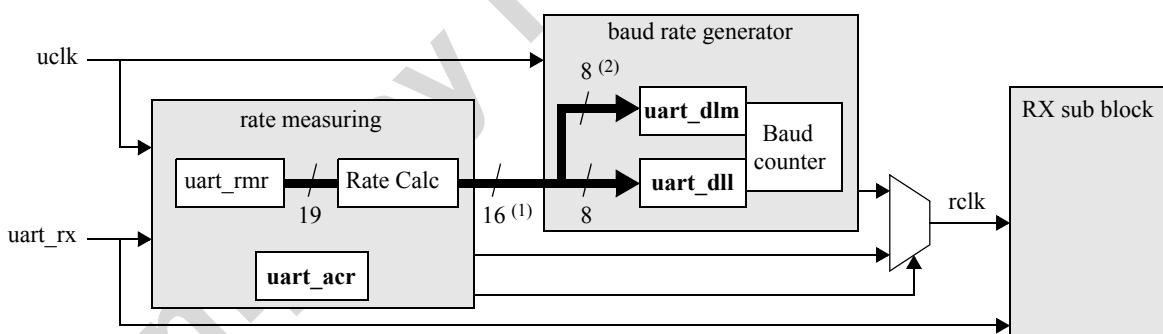
It is assumed that the baud rate may change in the command mode in between two successive 'AT'-commands only.

If the **start** bit is set, the receive baud rate is computed by measuring on the next received character, the length of the start bit and the character LSB (**mode** = 0) or the start bit only (**mode** = 1). After successful autobaud, the registers **uart_dlm** and **uart_dll** are updated and the behavior of the baud generator is modified accordingly. Interrupt can be generated at the end of successful autobaud or when the measurement counter **uart_rmr** overflows.

In case of overflow, an automatic restart function can be enabled to immediately restart a measurement on next character. This is useful when the first character follows a break state on RXD line (assuming break state duration is greater than time-out value in Table 444).

In case of possible short break duration (Less than Table 444), customer must use **mode=1** and check by software (polling or interrupt at end of autobaud) if reported baud rate is acceptable or if measure was corrupted by a short break duration.

The following diagram coarsely shows the flow diagram modification in the UART block diagram.



(1) relevant bits: 14-bit if **mode** = 0, 15-bit if **mode** = 1, unused MSB are 0

(2) relevant bits: 6-bit if **mode** = 0, 7-bit if **mode** = 1, unused MSB are 0

Fig 171. Autobaud rate measurement

uart_rmr = Autobaud rate measurement register, the 19 bits counter used to measure the input rate. This register is not accessible by the SC.
Rate Calculation is given by the formula:

For **mode** = 0:

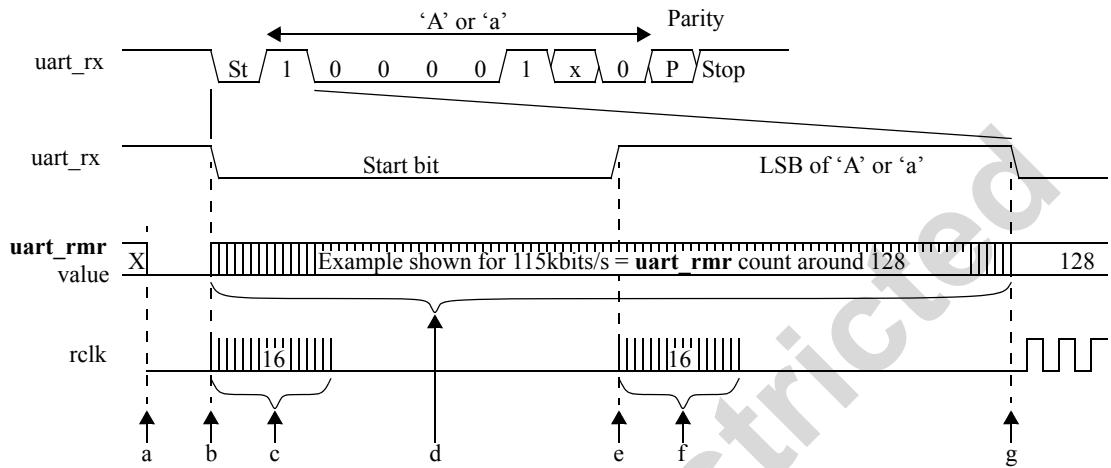
$$(uart_{dlm} \times 256) + uart_{dll} = \frac{uart_{rmr} + 16}{32} \quad (6)$$

For **mode** = 1:

(7)

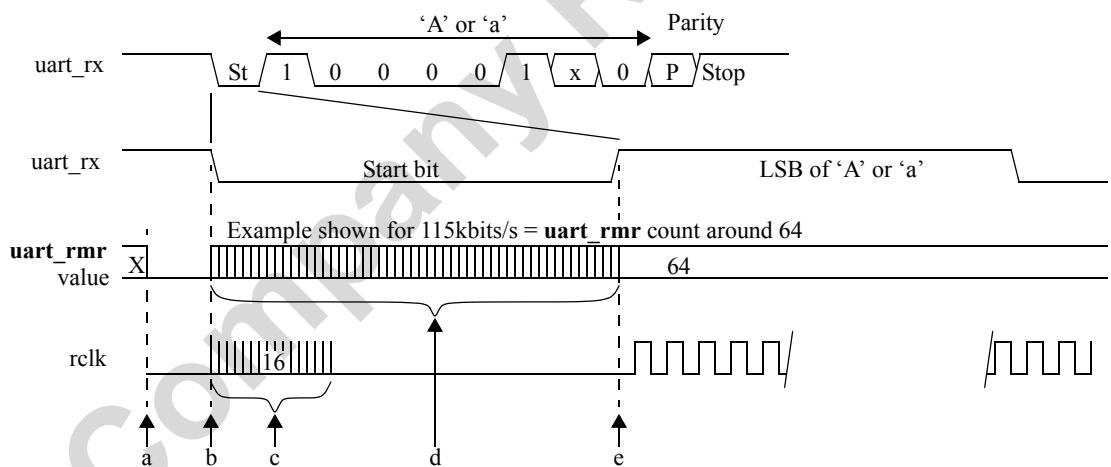
$$(uart_dlm \times 256) + uart_dll = \frac{(uart_rmr \times 2) + 16}{32}$$

The following figures shows typical examples:



(1) See text below for description of state a, b, c, d, e, f and g.

Fig 172.Autobaud example for mode = 0 (with uclk = 7.3728 MHz)



(1) See text below for description of state a, b, c, d, and e.

Fig 173.Autobaud example for mode = 1 (with uclk = 7.3728 MHz)

When the software is expecting an 'AT' command, it configures the UART with the expected character format and sets the **start** bit. The initial values in the divisor latches (**uart_dll** and **uart_dlm**) don't care.

Because of the 'A' or 'a' ASCII coding ('A'= 0x41, 'a' =0x61), the start bit and the LSB of the expected character are delimited by two falling edges.

When the **start** bit is set, the following operations are run by the hardware (in case of timeout with **auto_restart** = 1, the autobaud operation is resumed on state a):

- a. On **start** bit setting, **uart_rmr** is reset and the uart receive sub-block state machine is reset.
The **rclk** is switched to the rate measuring mechanism, which is at low level.
- b. Falling edge on RXD trigger the c and d
- c. During the receipt of the start bit, 16 pulses are generated on **rclk** with the frequency of the UART input clock (**uclk**, which corresponds to the highest baud rate). This action store the start bit in the RX sub-block state machine.
- d. During the receipt of the start bit (**and** the character LSB for **mode = 0**), **uart_rmr** is incremented with the UART input clock (**uclk**).
- e. **mode = 0**: Rising edge on RXD trigger the f (**uart_rmr** continue to be incremented)
mode = 1: On rising edge on RXD, **uart_rmr** is frozen.
Then the baud generator divisor (**uart_dll**, **uart_dlm**) is forced with the 15 bits value of [Equation 7 on page 492](#). Unused bit **uart_dlm[7]** is cleared.
With Figure 173 example, the loaded value is $\frac{(64 \times 2) + 16}{32} = 4,5$, automatically down rounded to 0x0004 corresponding to 115.2kbits/s. The divisor loading initialize the baud counter, so, the baud generator now delivers immediately the computed **rclk** corresponding to 16 times the RXD input baud rate. **rclk** is switched to the baud rate generator to be able to get the remaining bits of the 'A' (or 'a') character at the right baud rate. The **start** bit is reset and the **ueoa** is set (generates an interrupt if **ueoa_ie** is set).
- f. Only in **mode = 0**, during the receipt of the character LSB, 16 pulses are generated on **rclk** with the frequency of the UART input clock (**uclk**). This action stores the LSB bit in the RX sub-block state machine.
- g. Only in **mode = 0**, on the falling edge indicating the end of the character LSB, **uart_rmr** is frozen.
Then the baud generator divisor (**uart_dll**, **uart_dlm**) is forced with the 14 bits value of [Equation 6 on page 492](#). Unused bits **uart_dlm[7:6]** are cleared.
With Figure 172 example, the loaded value is $\frac{128 + 16}{32} = 4,5$, automatically down rounded to 0x0004 corresponding to 115.2kbits/s. The divisor loading initialize the baud counter, so, the baud generator now delivers immediately the computed **rclk** corresponding to 16 times the RXD input baud rate. **rclk** is switched to the baud rate generator to be able to get the remaining bits of the 'A' (or 'a') character at the right baud rate. The **start** bit is reset and the **ueoa** is set (generates an interrupt if **ueoa_ie** is set).

The size of the **uart_rmr** counter (19 bits) allows estimating the lowest baud rate and the timeout value:

Table 444: Lowest autobaud rate / Timeout values

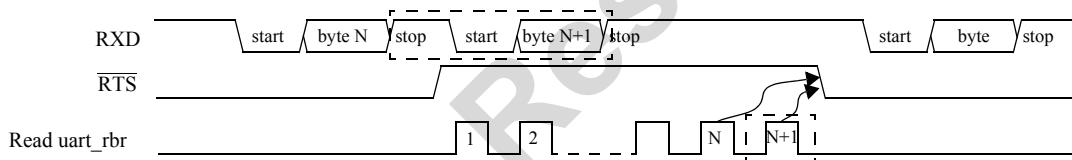
uclk (MHz)	Lowest Autobaud rate		Timeout value
	mode = 0	mode = 1	
7.3628	28 bits/s	14 bits/s	71.2 ms
14.7257	56 bits/s	28 bits/s	35.6 ms
29.4513	112 bits/s	56 bits/s	17.8 ms

The software has to correlate the received characters with the sequence expected for the 'AT'-command and to determine the character format (Parity, data length, number of stop bits).

For instance, having configured the UART with expected format 8N1, the software is able to distinguish the received data format among the formats 8N1, 7N2, 7E1, 7O1.

9.28.9.4 Auto-RTS

Auto-RTS data flow control originates in the receiver block and is linked to the programmed receiver FIFO trigger level. When the receiver FIFO level reaches a trigger level of 9, 24, 40, or 63, RTS is deasserted. The sending device may send an additional byte after the deassertion of RTS (assuming the sending device has another byte to send) because it may not recognize the change of RTS until after it has begun sending the additional byte. RTS is automatically reasserted once the receiver FIFO is emptied by reading the `uart_rbr` register. The assertion enables the sending device to continue transmitting data. Use of a higher threshold than the receive data available threshold avoids RTS assertion if user software is able to start emptying FIFO before this next threshold is reached. These optimize data transfer rates by best use of receive FIFO.

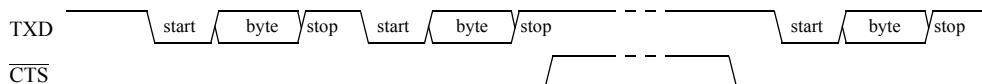


- (1) N = receiver FIFO trigger level + 8 (or + 7 if level = 56).
- (2) The two blocks in dashed lines cover the case where an additional byte is sent as described in auto-RTS.

Fig 174. RTS functional timing

9.28.9.5 Auto-CTS

The transmitter circuitry checks CTS before sending the next data byte. When CTS is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, CTS must be released before the middle of the last stop bit that is currently being sent. The auto-CTS function reduces interrupts to the host system. When flow control is enabled, CTS changes state and do not trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS and without software flow control, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.



- (1) When CTS is low, the transmitter keeps sending serial data out.
- (2) When CTS goes high before the middle of the last stop bit of the current byte, the transmitter finishes sending the current byte but it does not send the next byte.
- (3) When CTS goes from high to low, the transmitter begins sending data again.

Fig 175. CTS functional timing

9.28.9.6 Enabling auto-RTS and auto-CTS

The Auto-RTS and Auto-CTS modes of operation are activated by setting **uart_mcr.auto** to 1. The receiver FIFO trigger level can be set to 1, 16, 32, or 56 bytes. Corresponding RTS threshold is 9, 24, 40, or 63 bytes

Table 445:RTS/CTS flow configuration

uart_mcr. auto	uart_mcr. rts	Flow configuration
1	1	auto-RTS and auto-CTS enabled (autoflow control enabled)
1	0	auto-CTS only enabled; RTS forced to inactive state
0	x	auto-RTS and Auto-CTS disabled; manual handling of RTS and CTS

9.28.9.7 FIFO Interrupt mode operation

See also detailed description of **uart_iir** on [Table 422](#).

The following receive interrupts will occur when the receive FIFO and receive interrupts are enabled. All interrupts reflect the byte at the top of the FIFO. The interrupt descriptions are in order of their priority. (The first description is the highest priority.)

1. **uart_iir[3:0] = 0x1** indicates that there are no interrupts pending
 2. **uart_iir[3:0] = 0x6** (receive line status interrupt) indicates that the byte at the top of the FIFO has some sort of error in it (**oe**, **pe**, **fe**, or **bi**); reading the **uart_lsr** clears this interrupt; value of **uart_lsr** also tells the user which one of the errors is in that byte
 3. **uart_iir[3:0] = 0x4** (receive data available interrupt) indicates that data is available in the receive FIFO. This occurs when the receiver FIFO is filled with data at or above the trigger level. Reading from the **uart_rbr** clears this interrupt when the amount of data in FIFO is less than the trigger level. The **uart_lsr.rdr** bit, when it is 1, says that the data in the top byte in the receive FIFO is available. When reading empties the FIFO, **uart_lsr.rdr** is reset
- The character time-out indication interrupt (**uart_iir[3:0] = 0xC**) description is found in the following section; it has the same priority as the receiver data available interrupt (**uart_iir[3:0] = 0x4**)
- a. If the following conditions exist, a character time-out indication interrupt will occur: at least one character in the FIFO, last serial character was received more than 3.5 to 4.5 continuous character times ago (if two stop bits are programmed, the second one is included in the time delay) and the last read of the FIFO was more than 3.5 to 4.5 continuous character times ago; at

- 300 baud and 12-bit characters (start + 8 bits + parity + 2 stop), the character time-out indication interrupt causes a latency of 160 ms maximum from received character to interrupt issued
- b. Character times are calculated by using the **rclk** clock signal (the delay is proportional to the baud rate)
 - c. The FIFO level change timer is reset after the software reads the receive FIFO or after a new character is received when there has been no character time-out indication interrupt
 - d. A character time-out indication Interrupt is cleared and the timer is reset when the software reads a character from the RCVR FIFO.
4. **uart_iir[3:0] = 0x2** (**thre** interrupt) indicates that the transit holding register is empty; this interrupt is cleared by either writing a byte to the **uart_thr** or by reading the **uart_iir**
 5. **uart_iir[3:0] = 0x0** (modem status interrupt) indicates that there has been some change in the status of the modem; reading the **uart_msr**, which also tells the user what type of status change occurred, clears this interrupt.

Transmit interrupts will occur as follows when the transmitter and transmit FIFO interrupts are enabled (**uart_fcr.fifoe** = 1, **uart_ier.thre_ie** = 1).

1. The transmit FIFO empty indications is delayed one character time minus the last stop bit time whenever: **thre** = 1 and there have not been a minimum of two bytes at the same time in transmit FIFO, since the last **thre** = 1
2. When the transmitter FIFO is empty, the transmitter holding register interrupt (**uart_iir** = 02) occurs. The interrupt is cleared as soon as the transmitter holding register is written to or the **uart_iir** is read. When **uart_fcr.fifoe** is enabled, an interrupt does not occur immediately if enabled (**uart_ier.thre_ie** = 1). The first transmit interrupt occurs due to the conditions stated in 1 and 2 above only after data has first been entered into the transmit FIFO. Receive FIFO trigger level and character trigger change level interrupts have the same priority as the transmitter FIFO empty.

9.28.9.8 FIFO Polled mode operation

Resetting **uart_ier.rda_ie**, **uart_ier.thre_ie**, **uart_ier.rls_ie**, **uart_ier.modem_ie** or all to zero, with **uart_fcr.fifoe** = 1, puts the UART into the FIFO Polled mode. Since the receiver and transmitter are controlled separately, either or both can be in the Polled mode. In the FIFO Polled mode, there is no time-out condition indicated or trigger level reached. The receive and transmit FIFOs still have the capability of holding characters, however.

9.28.10 Timed IRQ/DMA time-out functional description

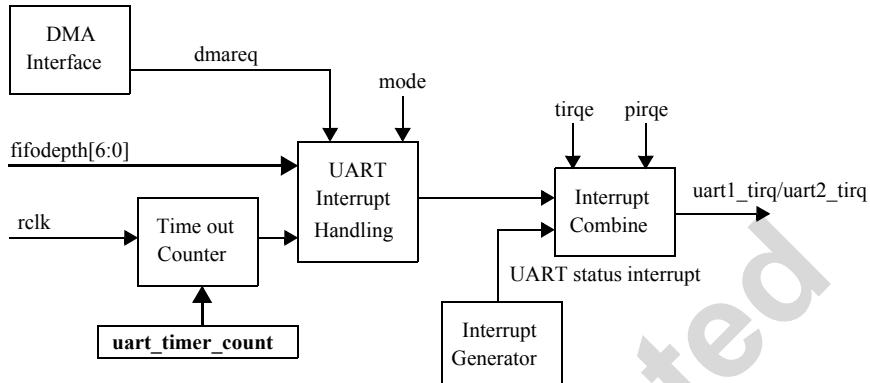


Fig 176.Timed interrupt and DMA time-out block diagram

9.28.10.1 TIMER mode

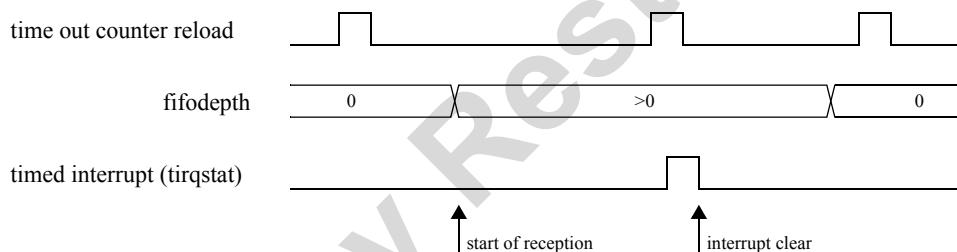
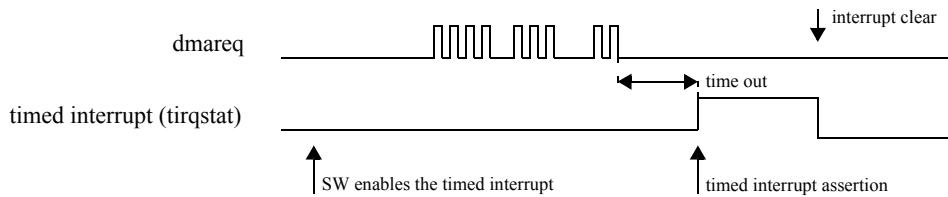


Fig 177.Timed Interrupt mode timing diagram

- When **uart_timer_count** value is written, counter start from 0
- Counter continuously count from 0, with automatic reload to 0 when count value is reached
- When counter roll over count value, an output pulse is generated
- If the **fifodepth** is not 0 during the rising edge of the counter output, a **tirq** output is generated and memorized
- When software clear **tirqstat** bit, the **tirq** output is cleared, regardless counter output neither **fifodepth**
- Software then serve peripheral data by usual way

Note: data IRQ must be disabled in the peripheral, but error interrupt can be enabled.

9.28.10.2 DMA Time-out mode**Fig 178.DMA Time-out mode timing diagram**

- When **uart_timer_count** value is written, counter start from 0, time-out go to idle; (software has armed the time-out)
- Time-out is launch by the first **dmareq**
- Every rising edge on **dmareq** clear the counter
- When counter roll over count value, an output pulse is generated
- Time-out go to idle
- **tirq** output is generated and memorized
- When software clear **tirqstat** bit, the **tirq** output is cleared, regardless **dmareq** level.

Counter timing are based on **rclk** ($\frac{1}{16}$ th of bit duration). i.e. 16-bit counter allow timing from 0 to 4095 bits duration (about 370 characters of start + 8-bits + parity + stop).

9.28.11 IrDA functional description

The IC incorporates two IrDA encoder/decoder. An IrDA encoder/decoder may be used in conjunction with UART1 and/or UART2. When enabled the TXD1 or TXD2 pin becomes transmit data interfaces to an IrDA LED, and the RXD1 or RXD2 pin becomes the receiver interface with an IrDA detector. The software interface is via the UART1 or UART2 register-set, FIFOs and interrupts. A single IrDA register selects: IrDA mode, transmit pulse width and receive data polarity.

The IrDA physical layer specification describes a half duplex, point to point, 0 to at least 1 meter, 2.4 to 115.2 kbit/s, infrared data link. Use with other speed can cause unpredictable results.

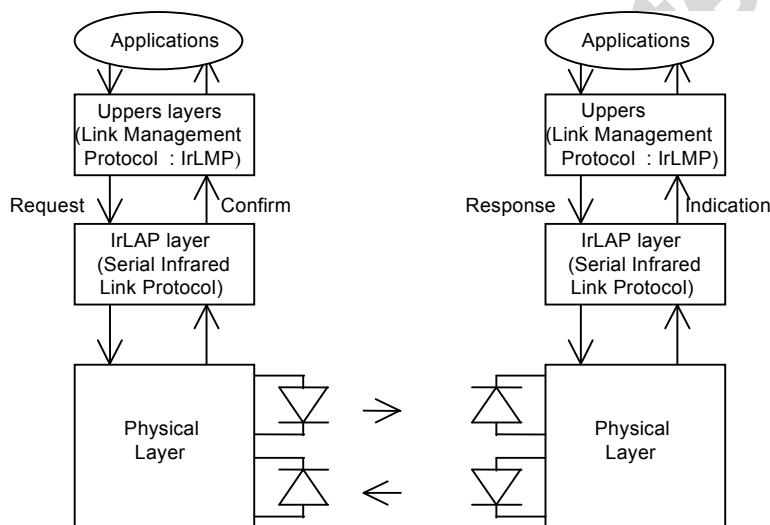


Fig 179.Structure of IrDA protocol

9.28.11.1 Physical layer

The IrDA physical hardware consists of an encoder/decoder (which performs the IR transmit encoder and IR receiver decoder), and the IR transducer (which consist of the IR led driver for transmitting and the receiver/detector). The encoder/decoder interfaces directly to the UART.

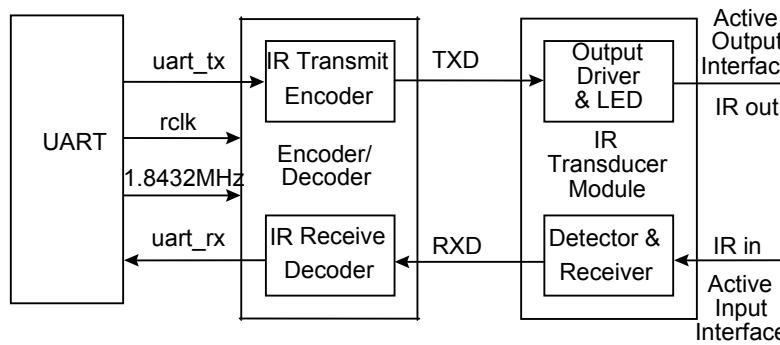


Fig 180.Transducer module interface

The UART must be programmed with 1 start bit (logical 0), 8 data bits, no parity and 1 stop bit (logical 1). The data are encoded such that a logical 0 is represented by a IR light pulse, and a logical 1 is represented by no pulse.

9.28.11.2 Encoding

An IR light pulse is defined as occupying a minimum of 1.6 μ s to a maximum of $3\frac{1}{16}$ th-bit length. $3\frac{1}{16}$ th-bit length is inversely proportional to the bit rate of the data, i.e. the slower the data rate, the longer the pulse is. Consequently, at lower data rate the pulse is very long (78 μ s at 2400 bits/s), and since the IR led current is about 300 mA, the power consummation increases dramatically. Therefore, for portable equipment 1.6 μ s pulses are usual.

This pulse stream forms the input to the driver for the IR emitter, which converts the electrical pulses to IR energy.

The start and stop bits are encoded/decoded exactly like data bits.

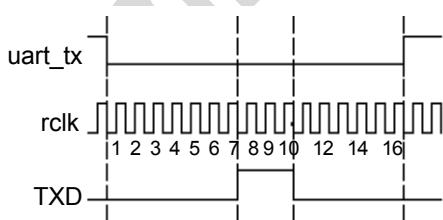


Fig 181.Encoding scheme (3/16 bit rate pulse)

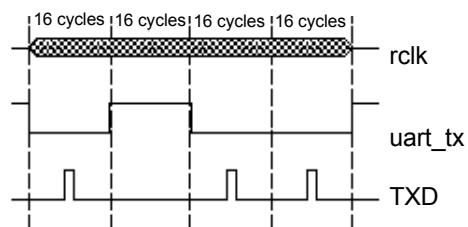


Fig 182.Macro view

9.28.11.3 Decoding

The decoding function performs the opposite of the encoder, and restores the received signal for the UART. Every high to low transition of the IR_RXD line signifies the arrival of a pulse. This pulse is stretched to accommodate 1 bit time. Every pulse received is translated into a logical 0 on the RXD line for the UART.

The decoder is made less sensitive to noise by checking that the input signal remains low for two successive edges of the 1.8432 MHz clock, before detecting a receive data bit.

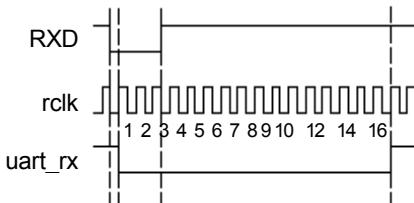


Fig 183.Decoding scheme

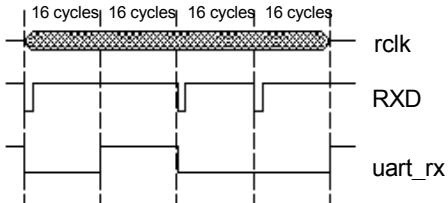


Fig 184.Macro view

Since some IR receivers provide a positive pulse, the decoder is able to invert the polarity of its input.

Table 446: Signalling rate and pulse duration specification

Signalling rate (kbits/s)	Rate tolerance% of rate	Pulse duration (us)		
		Minimum	Nominal	Maximum
2.4	± 0.87	1.41	78.13	88.55
9.6	± 0.87	1.41	19.53	22.13
19.2	± 0.87	1.41	9.77	11.07
38.4	± 0.87	1.41	4.88	5.96
57.6	± 0.87	1.41	3.26	4.34
115.2	± 0.87	1.41	1.63	2.23

9.28.11.4 Protocol

The infrared data association has defined and adopted a link protocol for the serial infrared link. IrDAs infrared link access protocol, or IrLAP, is derived from an existing asynchronous data communications standard (an adaptation of HDLC.) An essential element of IrLAP is the relationship between a primary and one or more secondary stations. A data link involves at least two participating stations. IrLAP provides two roles for participating stations.

Primary (commanding) stations and secondary (responding) stations. The primary station has responsibility for the data link. All transmissions over a data link go to, or from, the primary station. IrLAP communication links can be point-to-point or point-to-multipoint. There is always one and only one primary station; all other stations must be secondary stations. Any station capable can contend to play the primary station role. The role of primary station is determined dynamically at link-connection establishment and remains until the connection is closed. The exception is that there is a method provided for a primary and secondary on a point to point link to exchange roles without closing the connection. Framing, procedures and data link operation IrLAP uses most of the standard types of frame defined by the HDLC standards. The frames are classified by function as follows: unnumbered or U frames, supervisory or S frames and information or I frames. U frames are used for such functions as establishing and removing connections and discovery of other

station device, addresses, etc. I frames are used to transfer information from one station to another. S frames are used to assist in the transfer of information, they may be used to specifically acknowledge receipt of I frames (I frames can implicitly acknowledge other I frames also) and to convey ready and busy conditions.

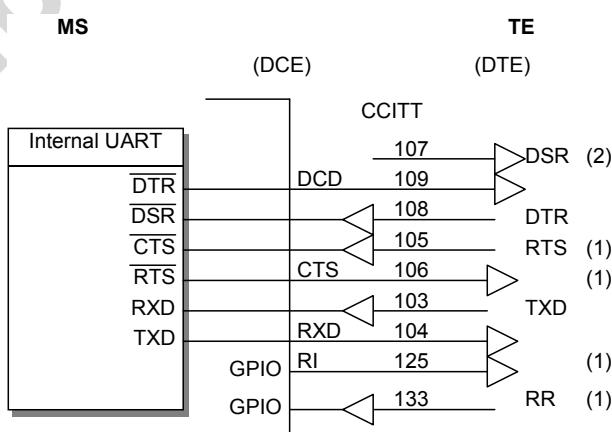
IrLAP also describes procedures that support link initialization, device address discovery, connection start-up (including link data rate negotiation), information exchange, disconnection, link shutdown, and device address conflict resolution.

A link operates essentially as follows. A device will want to connect to another device (either by automatic detection via the discovery and sniffing capability of IrLAP, or via direct user request). After obeying the media access rules the initiator will send connection request information at 9.6 kbits/s to the other device, this data will include information such as its address and its other capabilities (e.g., data rate, etc.). The responding device will assume the secondary role and after obeying, the media access rules return information that contains its address and capabilities. The primary and secondary will then change the data rate and other link parameters to the common set defined by the capabilities described in the information transfer. The primary will then send data to the secondary confirming the link data rate and capabilities. The two devices are now connected and the data is transferred between primary and secondary under the complete control of the primary. Rules are defined which ensure that the secondary and primary are both able to efficiently transfer data. IrDA Infrared Link Management Protocol (IrLMP)

9.28.12 Application information

9.28.12.1 Connecting

Note: the UART control pins was designed for modem usage. Therefore, it is suggested that the following alternate name of the pins be used. DTR used for DCD and DCD used for DTR.



(1) May be required by the application.

(2) No longer used.

Fig 185.Connecting UART with a DTE

9.28.12.2 CTS_n selection

If CTS_n pin is used, gpio muxing need to be configured before initialising the uart (due to internal muxing the bit uart_msr.dcts becomes '1' after reset).

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.29 USB - Universal Serial Bus**9.29.1 Features**

The USB peripheral is a serial interface fully compliant to [22.]. It allows the connection of the DVFD818x Family as a slave to an external host (like a PC) in order to transfer large amount of data. The following features are supported:

- 12 Mbit/s baud rate only (full-speed)
- Integrated analog transceiver (only 2 interface wires)
- On-chip 48 MHz clock generation
- Number of endpoints
 - one bidirectional control endpoint (1 input and 1 output) with 8-byte buffer in each direction
 - 4 bulk/interrupt endpoints (2 inputs and 2 outputs) with 16-byte buffer each
 - 2 bulk/interrupt endpoints (1 input and 1 output) with double buffer of 64 bytes each
 - 2 isochronous endpoints (1 input and 1 output) with double buffer of 294 bytes each
 - 2 × 788 byte RAM accessible in words
- 2 interrupts to the INTC
- USB bus error detection and recovery
- Power management features: suspend mode, wake-up from suspend
- SoftConnect™
- USB power detection capability (via GPIO EXTINT) - hot-plug: supports plugging during USB operation
- GoodLink™

9.29.2 Block diagram

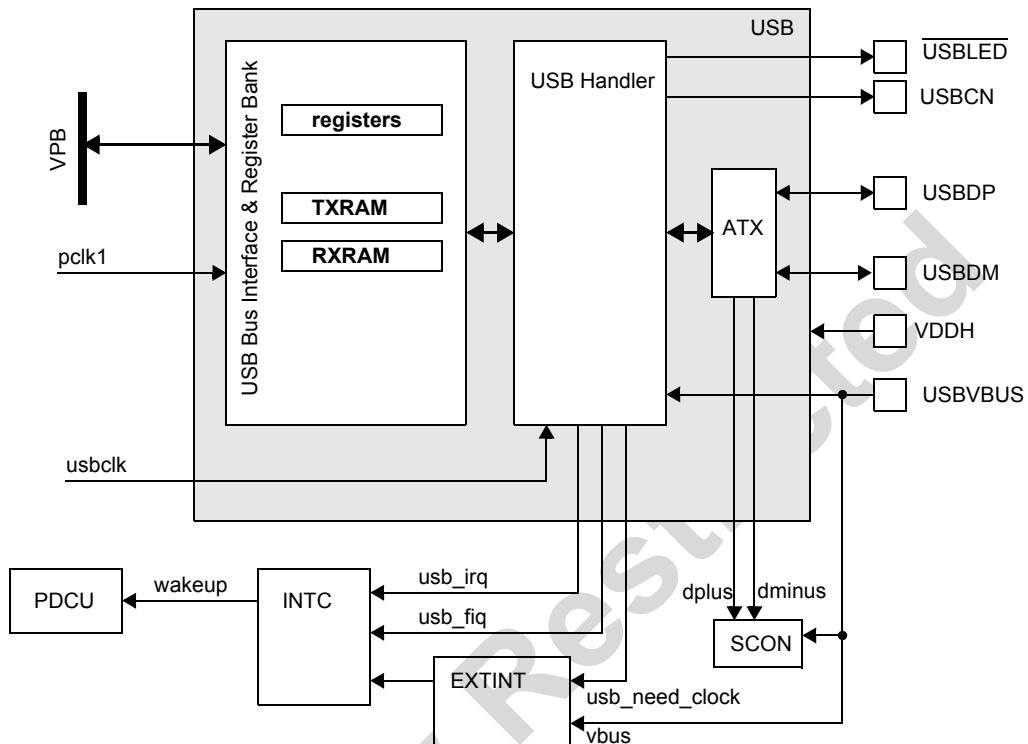


Fig 186.USB block diagram

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.29.3 Hardware interface

Table 447:USB pin overview

PIN	Name	I/O	Description
USBDP	USB data plus	I/O	positive part of the differential USB data signal
USBDM	USB data minus	I/O	negative part of the differential USB data signal
USBCN	USB connect	O	external pull-up control for SoftConnect™
USBLED	USB GoodLink™	O	USB GoodLink™ Indicator. Low level means external LED lighted

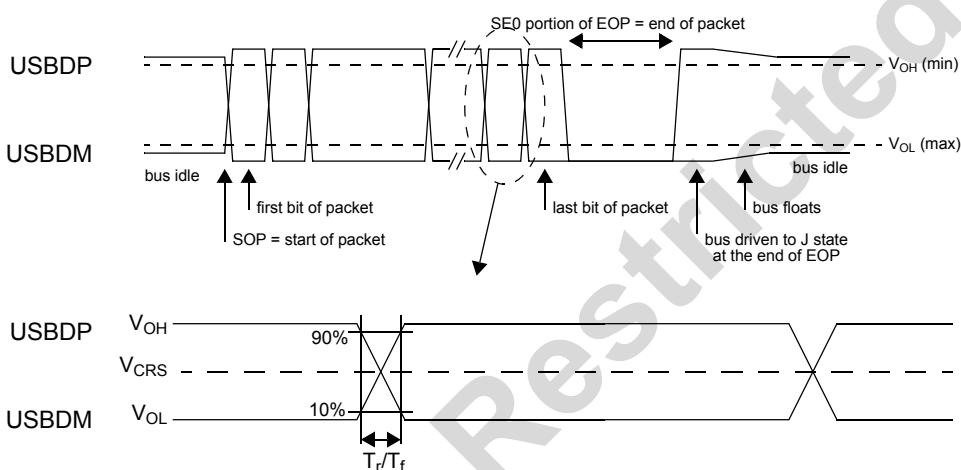


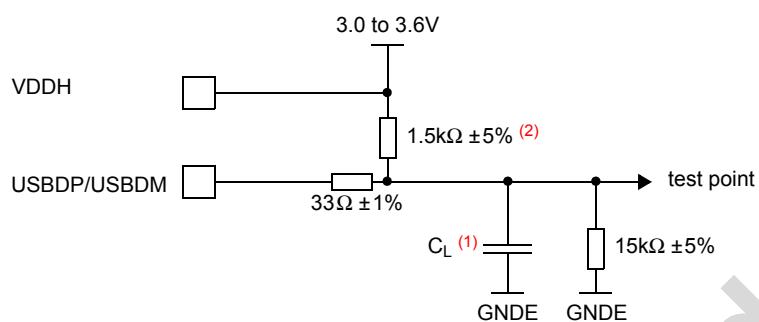
Fig 187.USB timing diagrams

Table 448:AC characteristics of USB interface (USBDP and USBDM signals) [1]

Test circuit, see [Figure 188](#)

Name	Parameter	Conditions	Min	Typ	Max	Unit
T _r	rise time	C _L = 50 pF; 10 to 90% of V _{OH} - V _{OL}	4	-	20	ns
T _f	fall time	C _L = 50 pF; 90 to 10% of V _{OH} - V _{OL}	4	-	20	ns
RFM	differential rise/fall time matching (T _r /T _f)	Excluding the first transition from Idle state	90	-	111.1	%
V _{CRS}	output signal crossover voltage	Excluding the first transition from Idle state	1.3	-	2.0	V

[1] The timings in this table are guaranteed by design, correlation, and structural testing. They are not specifically measured in production testing.



(1) $C_L = 50 \text{ pF}$

(2) Present only in case of USBDP

Fig 188. USBDP, USBDM AC characteristic test load

Remark: As stated in USB specification [22.], USB signals must withstand continuous short-circuit for at least 24 hour and must not be damaged even when transmitting 50% of the time and receiving 50% of the time (obviously, data transmission will be corrupted during short-circuit).

Short-circuit is intended to occur at USB appliance connector or inside USB cable, so the resistors implemented on USBDP/USBDM signals are fully part of the protection.

DVFD818x Family supports short-circuit of D+ and/or D- to VBUS, GND or the cable shield. See [Section 9.29.6.2 “External circuitry” on page 529](#) for USB connector connection.

9.29.4 Software interface

Table 449: Register overview of USB

Symbol	Name	I/O	Reset
usbintstat	interrupt status register	R	0x0000 0000
usbinten	interrupt enable register	R/W	0x0000
usbintclr	interrupt clear register	W	0x0000
usbintset	interrupt set register	W	0x0000
usbcmcmdcode	command code register (for USB handler)	W	0x0000 0000
usbcmddata	command data register (for USB handler)	R	0x00
usbrxdata	receive data register	R	0x0000 0000
usbttxdata	transmit data register	W	0x0000 0000
usbrxpckl	receive packet length register	R	0x0000
usbtxpckl	transmit packet length register	R/W	0x0000
usbcon	control register	R/W	0x00
usbfiqsel	usb_fiq interrupt selection register	W	0x00

Table 450: Register usbintstat^[1]

Bit	Symbol	Access	Value	Description
31 to 20	-	R	0*	reserved
19	ep7bf	R	0*	IN endpoint 7 buffer full = indicates that endpoint 7 has valid data = any packet which has not been transferred yet is considered valid ^[2]
18	ep5bf	R	0*	IN endpoint 5 buffer full = indicates that endpoint 5 has valid data = any packet which has not been transferred yet is considered valid ^[2]
17	ep3bf	R	0*	IN endpoint 3 buffer full = indicates that endpoint 3 has valid data = any packet which has not been transferred yet is considered valid ^[2]
16	ep1bf	R	0*	IN endpoint 1 buffer full = indicates that endpoint 1 has valid data - any packet which has not been transferred yet is considered valid ^[2]
15 to 14	-	R	0*	reserved
13	intstat[13]	R	0*	end_pck_in_int : end of packet for IN transfer reached
12	intstat[12]	R	0*	end_pck_out_int : end of packet for OUT transfer reached
11	intstat[11]	R	0*	cmd_data_full_int : command data register is full ^[3]
10	intstat[10]	R	0*	cmd_code_empty_int : command code register is empty - a new command can be issued ^[4]
9	intstat[9]	R	0*	device_stat : device status interrupt ^[5]
8	intstat[8]	R	0*	endpoint7_int : endpoint 7 interrupt (IN) ^[6]
7	intstat[7]	R	0*	endpoint6_int : endpoint 6 interrupt (OUT) ^[6]
6	intstat[6]	R	0*	endpoint5_int : endpoint 5 interrupt (IN) ^[6]
5	intstat[5]	R	0*	endpoint4_int : endpoint 4 interrupt (OUT) ^[6]
4	intstat[4]	R	0*	endpoint3_int : endpoint 3 interrupt (IN) ^[6]

Table 450: Register usbintstat^[1]...continued

Bit	Symbol	Access	Value	Description
3	intstat[3]	R	0*	endpoint2_int : endpoint 2 interrupt (OUT) ^[6]
2	intstat[2]	R	0*	endpoint1_int : endpoint 1 interrupt (IN) ^[6]
1	intstat[1]	R	0*	endpoint0_int : endpoint 0 interrupt (OUT) ^[6]
0	intstat[0]	R	0*	frame_int : indicates the start of a USB frame - occurs once every millisecond

- [1] Bits 13 to 0 in this register are set when the corresponding interrupt has occurred - the values in this register are not dependent of the value inside **usbinten**.
- [2] These bits are status information and do not generate an interrupt.
- [3] The **cmd_data_full_int** bit gets active 7 pclk1 cycles after issuing a read command to the USB handler.
- [4] The **cmd_code_empty_int** bit has little practical significance for the DVFD818x Family, as the maximum rate of VPB accesses is $\frac{104 \text{ MHz}}{3} = (34,7) \text{ MHz}$ - this interrupt can be disabled.
- [5] **device_stat** can be set by the interrupt conditions from the device status command (see Table 474).
- [6] These are interrupts from the endpoints. OUT endpoints generate interrupts when their buffer is full and hence means that the SC has to transfer the data from the buffer. IN endpoints generate interrupts according to the setting in the Set mode command (**inton_nak_xx** fields), and they signal that the SC has to fill the corresponding IN endpoint buffer with data.

Table 451: Register usbinten

Bit	Symbol	Access	Value	Description
15 to 14	-	R	0*	reserved
13	inten[13]	R/W	0*	end_pck_in_int : end of packet for IN transfer reached
12	inten[12]	R/W	0*	end_pck_out_int : end of packet for OUT transfer reached
11	inten[11]	R/W	0*	cmd_data_full_int : command data register is full
10	inten[10]	R/W	0*	cmd_code_empty_int : command code register is empty - a new command can be issued
9	inten[9]	R/W	0*	device_stat : device status interrupt
8	inten[8]	R/W	0*	endpoint7_int : endpoint 7 interrupt
7	inten[7]	R/W	0*	endpoint6_int : endpoint 6 interrupt
6	inten[6]	R/W	0*	endpoint5_int : endpoint 5 interrupt
5	inten[5]	R/W	0*	endpoint4_int : endpoint 4 interrupt
4	inten[4]	R/W	0*	endpoint3_int : endpoint 3 interrupt
3	inten[3]	R/W	0*	endpoint2_int : endpoint 2 interrupt
2	inten[2]	R/W	0*	endpoint1_int : endpoint 1 interrupt
1	inten[1]	R/W	0*	endpoint0_int : endpoint 0 interrupt
0	inten[0]	R/W	0*	frame_int : indicates the start of a USB frame - occurs once every millisecond

- [1] When these bits are cleared, then the corresponding source does not generate a **usb_irq** interrupt - the bits inside **usbintstat** are still set. When these bits are set, the corresponding interrupt source can generate a **usb_irq** interrupt.

Table 452: Register usbintclr

Bit	Symbol	Access	Value	Description
15 to 14	-	R	0*	reserved
13	intclr[13]	W	0*	end_pck_in_int : end of packet for IN transfer reached
12	intclr[12]	W	0*	end_pck_out_int : end of packet for OUT transfer reached
11	intclr[11]	W	0*	cmd_data_full_int : command data register is full
10	intclr[10]	W	0*	cmd_code_empty_int : command code register is empty - a new command can be issued
9	intclr[9]	W	0*	device_stat : device status interrupt
8	intclr[8]	W	0*	endpoint7_int : endpoint 7 interrupt
7	intclr[7]	W	0*	endpoint6_int : endpoint 6 interrupt
6	intclr[6]	W	0*	endpoint5_int : endpoint 5 interrupt
5	intclr[5]	W	0*	endpoint4_int : endpoint 4 interrupt
4	intclr[4]	W	0*	endpoint3_int : endpoint 3 interrupt
3	intclr[3]	W	0*	endpoint2_int : endpoint 2 interrupt
2	intclr[2]	W	0*	endpoint1_int : endpoint 1 interrupt
1	intclr[1]	W	0*	endpoint0_int : endpoint 0 interrupt
0	intclr[0]	W	0*	frame_int : indicates the start of a USB frame - occurs once every millisecond

- [1] Setting a bit in this register causes the clearing of the corresponding bit inside **usbintstat**. If the corresponding bit is set in **usbinten**, then a USB interrupt **usb_irq** is cleared - writing a zero into these bits has no effect.

Table 453: Register usbintset

Bit	Symbol	Access	Value	Description
15 to 14	-	R	0*	reserved
13	intset[13]	W	0*	end_pck_in_int : end of packet for IN transfer reached
12	intset[12]	W	0*	end_pck_out_int : end of packet for OUT transfer reached
11	intset[11]	W	0*	cmd_data_full_int : command data register is full
10	intset[10]	W	0*	cmd_code_empty_int : command code register is empty - a new command can be issued
9	intset[9]	W	0*	device_stat : device status interrupt
8	intset[8]	W	0*	endpoint7_int : endpoint 7 interrupt
7	intset[7]	W	0*	endpoint6_int : endpoint 6 interrupt
6	intset[6]	W	0*	endpoint5_int : endpoint 5 interrupt
5	intset[5]	W	0*	endpoint4_int : endpoint 4 interrupt
4	intset[4]	W	0*	endpoint3_int : endpoint 3 interrupt
3	intset[3]	W	0*	endpoint2_int : endpoint 2 interrupt
2	intset[2]	W	0*	endpoint1_int : endpoint 1 interrupt
1	intset[1]	W	0*	endpoint0_int : endpoint 0 interrupt
0	intset[0]	W	0*	frame_int : indicates the start of a USB frame - occurs once every millisecond

- [1] This register is used for debugging purposes.
[2] Setting a bit in this register causes the corresponding bit inside **usbintstat** to go active. If the corresponding bit is set in **usbinten**, then a USB interrupt **usb_irq** is generated - writing a zero into these bits has no effect.

Table 454: Register usbcmcmdcode

Bit	Symbol	Access	Value	Description
31 to 24	-	W	0*	reserved
23 to 16	cmdcode[7:0]	W	0*	defines a USB device or a USB endpoint command according to Table 464 - after executing the command, this register is empty and the cmd_code_empty_int is generated
15 to 8	cmd_ph	W	0x00*	command phase
			0x01	Write data phase
			0x02	Read data phase
			0x05	Command data phase
			other	reserved
7 to 0	-	W	0*	reserved

Table 455: Register usbcmddata

Bit	Symbol	Access	Value	Description
7 to 0	cmddata[7:0]	R	0*	carries the data retrieved from the USB handler after executing a command - when this register is full, the interrupt cmd_data_full_int is generated

Table 456: Register usbrxdata

Bit	Symbol	Access	Value	Description
31 to 0	rxdata[31:0]	R	0*	used for OUT transactions - data from the RXRAM is fetched and put into this register at a maximum rate of 2 pclk1 cycles per transfer - no interrupt is generated; this register is structured in byte according to little-endian convention

Table 457: Register usbttxdata

Bit	Symbol	Access	Value	Description
31 to 0	txdata[31:0]	W	0*	used for IN transactions - data written into this registers, is subsequently transferred to the TXRAM at a maximum rate once every pclk1 cycle - no interrupt is generated; this register is structured in byte according to little-endian convention

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 458: Register usbrxpckl

Bit	Symbol	Access	Value	Description
15 to 11	-	R	0*	reserved
10	valid	R	0*	data validity. Non-isochronous end point will not raise an interrupt when an erroneous data packet is received. For isochronous endpoint, data transfer will happen even if an erroneous packet is received.
				data is not valid
				data is valid
9 to 0	rxpckl[9:0]	R	0*	receive packet length: states the number of bytes available in RXRAM for the current packet - the register is decremented with every word which is read by the SC - the SC can use this field to determine the amount of bytes to be received - when this value reaches zero, the end of packet interrupt end_pck_out_int is generated after 3 pclk1 clock cycles Note: After programming of usbcon , reading this register must never occur in the very next access.

Table 459: Register usbtxpckl

Bit	Symbol	Access	Value	Description
15 to 10	-	R	0*	reserved
9 to 0	txpckl[9:0]	R/W	0*	transmit packet length: states the number of bytes to be transferred to TXRAM for the current packet - the register is decremented with every word which is written by the SC - the SC can use this field to determine the amount of bytes which still have to be transmitted - when this value reaches zero, the end of packet interrupt end_pck_in_int is generated after 3 pclk1 clock cycles Remark: To send an empty packet, it is necessary to write at least one dummy byte in the FIFO after having specified zero as length (usbtxpckl = 0)

Table 460: Register usbcon

Bit	Symbol	Access	Value	Description
7 to 6	-	R	0*	reserved
5 to 2	logep[3:0]	R/W	0b0000* to 0b0100	defines the active logical endpoint number integrated logical endpoints; others: reserved
1				write data enable 0* disables data writing to endpoint 1 enables data writing to endpoint
0	read	R/W	0* 1	read data enable 0* disables data reading from endpoint 1 enables data reading from endpoint

Table 461: Register usbfiqsel

Bit	Symbol	Access	Value	Description
7 to 3	-	-	~*	reserved
2 ^[1]	bulk_in	W	0*	disables endpoint7_int to be the interrupt source for usb_fiq
			1	causes endpoint7_int to be the interrupt source for usb_fiq
1 ^[1]	bulk_out	W	0*	disables endpoint6_int to be the interrupt source for usb_fiq
			1	causes endpoint6_int to be the interrupt source for usb_fiq
0 ^[1]	frame	W	0*	disables frame_int to be the interrupt source for usb_fiq
			1	causes frame_int to be the interrupt source for usb_fiq

[1] Only one of these bits can be active at a time. Setting more than one bit or clearing all of them removes the generation of the **usb_fiq**.

9.29.5 Functional description**9.29.5.1 General notes**

The USB block consists of a VPB bus interface and a USB device handler. The SC sends commands to the USB handler through the bus interface. Registers allow the management of the interrupts, the access to the transmit and receive buffers of the endpoints and the control of the interface.

To access the registers of the USB, both **pclk1** and **usbcclk** (48 MHz) have to be enabled in the CGU. As the USB block has two clock domains (**pclk1** and **usbcclk**), in order to avoid synchronization failure, the **pclk1** frequency should be higher than 16 MHz (equal to worst case USB clock frequency). Accesses to the block before the settling time has expired are not prevented and the result is undefined.

9.29.5.2 Data flow from the Host to the Device

The USB ATX receives the bi-directional D+ and D- lines of the USB bus. It will put this data in the unidirectional interface between ATX and USB block.

The USB block protocol engine receives this serial data and converts it into a parallel data stream. The parallel data is sent to the RXRAM interface which in turn will transfer the data to the endpoint buffer. The endpoint buffer is implemented as an SRAM based FIFO. Data is written to the buffers with the header showing how many bytes are valid in the buffer.

For non-isochronous endpoints when a full data packet is received without any errors, the endpoint generates a request for data transfer from its FIFO by generating an interrupt to the system.

Isochronous endpoint will have one packet of data to be transferred in every frame. This requires the data transfer has to be synchronised to the USB frame rather than packet arrival. The 1 kHz free running clock re synchronized on the incoming SOF tokens will generate an interrupt every millisecond.

The data transfer follows the little endian format. The first byte received from the USB bus will be available in the LS byte of the receive data register.

9.29.5.3 Data flow from the Device to the Host

For data transfer from an endpoint to the host, the host will send an IN token to that endpoint. If the FIFO corresponding to the endpoint is empty, the device will return a NAK and will generate an interrupt (assuming the interrupt on NAK is enabled). On this interrupt the processor fills a packet of data in the endpoint FIFO. The next IN token that comes (after filling this packet) will transfer this packet to the host.

The data transfer follows the little endian format. The first byte sent on the USB bus will be the LS byte of the transmit data register.

Remark: USB is a host controlled protocol, i.e., irrespective of whether the data transfer is from the host to the device or from the device to the host, the transfer sequence is always initiated by the host. During data transfer from the device to the host, the host sends an IN token to the device, following which the device responds with the data.

9.29.5.4 Interrupt Based Transfer

Slave data transfer is done through the interrupt issued from the USB block to the SC.

Reception of a valid (error-free) data packet in any of the OUT non-isochronous endpoint buffer generates an interrupt. Upon receiving the interrupt, the software can read the data using receive length and data registers. When there is no empty buffer (for a given non-isochronous OUT endpoint), any data arrival generates an interrupt only if Interrupt On NAK feature for that endpoint type is enabled and existing interrupt is cleared.

Similarly, when a packet is successfully transferred to the host from any IN non-isochronous endpoint buffer, an interrupt is generated. When there is no data available in any of the buffers (for a given non-isochronous IN endpoint), a data request generates an interrupt only if Interrupt On NAK feature for that endpoint type is enabled and existing interrupt is cleared. Upon receiving the interrupt, the software can load any data to be sent using transmit length and data registers.

9.29.5.5 Isochronous Transfer

Isochronous endpoints are double buffered and the buffer toggling will happen only on frame boundaries i.e., at every 1 ms. ‘Clear Buffer’ and ‘Validate Buffer’ do not cause the buffer to toggle.

For OUT isochronous endpoints, the data will always be written irrespective of the buffer status.

For IN isochronous endpoints, the data available in the buffer will be sent only if the buffer is validated; otherwise, an empty packet will be sent.

There will not be any interrupt generated specific to isochronous endpoints other than the frame interrupt. It is assumed that the Isochronous pipe is open at the reception of a request “Set Interface (alternate setting > 0)”. This request is sent to the interface to which the isochronous endpoint belongs. This means that the device is expecting the first isochronous transfer within the millisecond.

9.29.5.6 Automatic Stall Feature

USB block implement a hardware STALL mechanism (See [22.] for details). Automatic STALL will occur in case there is extra IN or OUT token following the Status stage in the following control transactions:

- Data stage consists of INs, the status is a single OUT transaction with empty packet sent by the host.
- Data stage consists of OUTs, the status is a single IN transaction, for which device responds with empty packet.
- Setup stage followed by a Status stage consisting of an IN transaction, for which device responds with empty packet.

and STALL will not occur in the following situations:

- Data stage consists of OUTs, the status is a single IN transaction, for which device responds with non-empty packet.

- Setup stage followed by a Status stage consisting of an IN transaction, for which device responds with non-empty packet.

9.29.5.7 Endpoint descriptions

The USB peripheral features 10 endpoints of different kinds (refer to [22.] for the description of an endpoint). All of the endpoints make use of 2 integrated SRAMs (TXRAM and RXRAM) of 788 bytes each in order to insure operation according to the USB specification.

Table 462:USB endpoints

Logical endpoint	Physical endpoint	Type	Direction	Buffer size (bytes)	Double buffer
0	0	control	OUT	8	no
	1	control	IN	8	no
1	2	bulk/interrupt	OUT	16	no
	3	bulk/interrupt	IN	16	no
2	4	bulk/interrupt	OUT	16	no
	5	bulk/interrupt	IN	16	no
3	6	bulk/interrupt	OUT	64	yes
	7	bulk/interrupt	IN	64	yes
4	8	isochronous	OUT	294	yes
	9	isochronous	IN	294	yes

9.29.5.8 Interrupts to the SC

The USB block generates two interrupts to the SC: **usb_irq** and **usb_fiq**.

usb_fiq can be configured in register **usbfiqsel** to have one of the following interrupt sources: frame_int, endpoint6_int or endpoint7_int (bulk/interrupt endpoint interrupts). Only one can be active at a time.

usb_irq can have multiple sources according to register **usbintstat** and should receive lower priority by the SW. For bits **usbintstat[9:1]** the SC has to find the reason of the interrupt in the USB handler.

The endpoint buffer full interrupts (for the IN endpoints) show that data in the buffer has not been yet transferred by the USB host. The SC should check whether there is still a valid packet in the buffer before writing again.

The isochronous endpoints 8 and 9 do not have a status bit in **usbintstat**, as they should be operated based on the frame clock. The SW has to read the buffer once every millisecond. If in the previous frame no transfer happened, the SW reads an empty packet.

9.29.5.9 Data Transfers

When the software wants to read the data from an endpoint buffer it should set the **usbcon.read** bit and should program the logical endpoint number. The control logic will first fetch the packet length to the **usbrxpckl** register. Also the hardware fills the **usbrxdata** register with the first word of the packet from the RXRAM. The software can now start reading the **usbrxdata** register. When the end of packet is reached the

usbcon.read bit will be automatically reset by the control logic, **usbintstat.instat[12]** bit is set and the **usb_irq** interrupt is asserted (if **usbinten.inten[12]** =1).
 If the software resets the **usbcon.read** bit during a packet transfer, the control logic stops to transfer data from the RXRAM. In this case the data will remain in the RXRAM. When the read enable (**usbcon.read**) is set again for this endpoint, data will be read from the beginning of the packet.

For writing data to an endpoint buffer, the **usbcon.write** bit should be made high and software should write to the **usbtxpckl** register the number of bytes it is going to send in the packet. The software can then write data continuously into the **usbtxdatal** register. When the number of bytes programmed in the **usbtxpckl** register are transferred, the control logic resets the **usbcon.write** bit, **usbintstat.instat[13]** bit is set and the **usb_irq** interrupt is asserted (if **usbinten.inten[13]** =1).
 If the software resets the **usbcon.write** bit during the transfer of a packet, the transfer has to start again from the beginning.

Both **usbcon.read** and **usbcon.write** bits can be high at the same time for the same logical endpoint. The interleaved read and write operation is possible. In this case read has priority over write. This feature gives the flexibility to stop a write transfer for some time and do a read transfer and again come back to the same write transfer.

9.29.5.10 Power management

Following USB block states are described hereafter: connected, suspend and disconnected.

- Connected: the block is clocked with both clocks, can accept and send data and generate interrupts at any time
- Suspend: the block is momentarily not used by the bus, and the 48 MHz clock can be turned off - the SC cannot force the USB to change to this state - this state can be left on request of the host or the SC
- Disconnected: the block is not connected to a USB cable
(in the DVFD818x Family this state is not possible).

Suspend state

The USB block enters the suspend state automatically when the bus has been constantly idle for more than 3 ms, generating a device status interrupt. The **usb_need_clock** signal goes low after 2 ms. The falling edge of **usb_need_clock** generates an interrupt, and signals that **usbclk** can be turned off.

The **usb_need_clock** signal is low when the USB is in suspend state. It is connected to EXTINT alternate channel (see [Table 256](#)). A rising edge on **usb_need_clock** sets the EXTINT interrupt flag and triggers a wake-up in the PDCU if the DVFD818x is in sleep state. **usb_need_clock** signal is only valid when VDDH is present, **usb_need_clock** indication has to be correlated with VBUS indication through SCON.

Operation is resumed (transition to connected state) when any non-idle bus signalling is detected or the SC generates a remote wake-up by clearing the **suspend** bit. This operation initiates a resume signal on the bus.

Disconnected state

The USB block enters the disconnected state when there is no **vbus** signal.

SC wake-up**Table 463: Inter-connection of SC modes and USB peripheral states**

SC mode	USB peripheral state		
	Connected	Suspend	Disconnected
active	the peripheral enters connect state: - when the bus is connected to the DVFD818x, - when any non-idle signalling is done on the bus, - when the SC clears the suspend bit	the peripheral enters suspend state when the bus is idle for more than 3 ms	When there is no vbus
idle	like active mode - any of the above conditions generates an interrupt to the core - the core goes to active	like active mode - an interrupt is generated - the core goes to active	
stop	like active mode - any of the above conditions generates an interrupt to the core - the core goes to active	like active mode - an interrupt is generated - the core goes to active	
sleep	like active mode - any of the above conditions generates an interrupt to the core - the core goes to active after the wake-up sequence	like active mode - an interrupt is generated - the core goes to active after the wake-up sequence	

SoftConnect™

This feature allows to pretend that DVFD818x is not a ready USB slave device. As soon as the DVFD818x is ready to accept USB activity, it attaches the pull-up resistor in order to signal to the host that a new peripheral device is available. The high-level on USBDP indicates that the USB slave is ready for reception.

After the SoftConnect™ feature has been enabled, the USB device of the DVFD818x gets enumerated.

In order to use this feature, a voltage derived from VBUS scaled to 3.3 V (VDDH) is connected to both USB **vbus** signal and SCON. The SC can use the **connect** bit in the device status command to control USBCN. The reset level 0 of **connect** implies that per default the SoftConnect™ is inactive, the USBCN pin is low, and that the DVFD818x is not connected to the bus (this is also necessary because of the time required by the Power-on reset sequence of the DVFD818x).

GoodLink™

Indication of a good USB connection is provided at pin **USBLED** through GoodLink™ technology. During enumeration, the LED indicator will blink on momentarily. When the USB block has been successfully enumerated (the device address is set), the LED indicator will remain permanently on. Upon each successful packet transfer (with ACK) to and from the DVFD818x, the LED will blink off for 100 ms. During 'suspend' state the LED will remain off.

9.29.5.11 Data flow considerations

The USB protocol is based on the transfer of packets. A packet consists of a synchronization field and a packed field. There are 4 categories of packets: **token** packets (define the endpoint to receive a packet), **data** packets (for data transfer), **handshake** packets (used to report the status of a data transaction) and **special** packets.

Data packets can have a payload varying from 0 to 1023 bytes and a 16-bit CRC (cyclic redundancy check). The handshake packets indicate the quality of the reception - following qualities are defined:

- ACK: handshake packet indicating a positive acknowledgement. Packet has been accepted
- NAK: handshake packet indicating a negative acknowledgement. Device is not ready to receive or transmit the packet
- STALL: handshake packet indicating that an error on the USB bus has occurred, or that a control pipe request is not supported.

Data flow from the USB host to the DVFD818x for non-isochronous endpoints

The USB handler converts the incoming serial data into byte streams. They are saved with the corresponding endpoint number and byte length in the RXRAM buffer. The handling of control, interrupts and bulk endpoints as well as the endpoint management is triggered by a dedicated interrupt **usb_irq**, which signals any corresponding event requiring attention of the system controller. It should read the data in the buffer for the required number of bytes.

Data flow from the USB host to the DVFD818x for isochronous endpoints

The handling of isochronous data has to be synchronized to the USB frame timing as at most one isochronous packet is transferred per frame for one endpoint. The interrupt **usb_fiq** represents a 1 kHz signal synchronous to the frame start used to trigger the isochronous data transfer to/from the isochronous input/output FIFO. **usbcpcpk1.valid** indicates if the data in the buffer are valid.

Data flow from the DVFD818x to the USB host for all endpoints

The system controller writes the data to be sent to the transmit register. The data are saved in the transmit buffer with a header. When the host initiates the IN transaction for the corresponding endpoint, data will be passed to the USB handler.

9.29.5.12 USB handler commands

The SC controls the USB handler and retrieves information from it by means of the **usbcmdcode** and **usbcmddata** registers. The access sequence always contains one command phase (with access to usbcmdcode), and zero or more data phases (with accesses to **usbcmdcode** or **usbcmddata**). In case of a write a second word has to be written to **usbcmdcode**. In case of a read one or more words have to be written into **usbcmdcode** and accordingly read accesses to **usbcmddata** may be performed.

Table 464:USB device commands

Command	Access sequence phase				Comment
	Command	Write data	Read data		
	usbcmcmdcode	usbcmcmdcode	usbcmcmdcode	usbcmddata	
device commands					
Set address	00'D0'05'00	00'B'01'00	-	-	Write 1 byte
Configure device	00'D8'05'00	00'B'01'00	-	-	Write 1 byte
Set mode	00'F3'05'00	00'B'01'00	-	-	Write 1 byte
Read current frame number	00'F5'05'00	-	00'F5'02'00	Read 1 byte	1 or 2 bytes
Read test register	00'FD'05'00	-	00'FD'02'00	Read 1 byte	2 bytes
Get device status	00'FE'05'00	-	00'FE'02'00	Read 1 byte	-
Set device status	00'FE'05'00	00'B'01'00	-	-	Write 1 byte
Get error code	00'FF'05'00	-	00'FF'02'00	Read 1 byte	-
Read error status	00'FB'05'00	-	00'FB'02'00	Read 1 byte	-
Read interrupt register	00'F4'05'00	-	00'F4'02'00	Read 1 byte	up to 5 bytes
endpoint commands					
Select endpoint	00'EP'05'00	-	(00'EP'02'00)	(Read 1 byte)	optional read
Select endpoint/clear interrupt ^[1]	00'EPP'05'00	-	00'EPP'02'00	Read 1 byte	-
Set endpoint status	00'EPP'05'00	00'B'01'00	-	-	Write 1 byte
selected endpoint commands					
Clear buffer	00'F2'05'00	-	(00'F2'02'00)	(Read 1 byte)	optional read
Validate buffer	00'FA'05'00	-	-	-	no data

[1] Not valid for Isochronous endpoints

[2] The notation B for the write data phase command means byte.

[3] The notation EP for the command phase means physical endpoint number (from 0 to 0x9).

[4] The notation EPP for the command phase means physical endpoint number plus 0x40 (from 0x40 to 0x49).

Table 465: Command set address (write 1 byte)

Bit	Symbol	Access	Value	Description
7	enable	W		USB handler enable
			0*	disabled
			1	enabled
6 to 0	address	W	0*	defines the USB device address (address for USB data packets)

Table 466: Command configure device (write 1 byte)

Bit	Symbol	Access	Value	Description
7 to 1	-	W	0*	reserved
0	configured	W	0*	disabled all non-control endpoints
			1	enable all non-control endpoints; Device is configured. This bit is set after the set configuration command from host is executed. USBLED signal is asserted.

Table 467: Command set mode (write 1 byte)

Bit	Symbol	Access	Value	Description
7 to 5	-	W	0*	reserved
4	inton_nak_ao	W	0*	Interrupt on NAK for bulk OUT endpoint
			1	only successful OUT transactions generate an interrupt
			1	both successful and NAKed OUT transactions generate interrupts
3	inton_nak_ai	W	0*	Interrupt on NAK for bulk IN endpoint
			1	only successful IN transactions generate an interrupt
			1	both successful and NAKed IN transactions generate interrupts
2	inton_nak_co	W	0*	Interrupt on NAK for control OUT endpoint
			1	only successful OUT transactions generate an interrupt
			1	both successful and NAKed OUT transactions generate interrupts
1	inton_nak_ci	W	0*	Interrupt on NAK for control IN endpoint
			1	only successful IN transactions generate an interrupt
			1	both successful and NAKed IN transactions generate interrupts
0	needclkon	W	0*	signal usb_need_clock changes depending on USB handler state
			1	signal usb_need_clock is always 1 - clock is not stopped during suspend

Table 468: Command read interrupt register (read first byte)

Bit	Symbol	Access	Value	Description
7	ep7	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
6	ep6	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
5	ep5	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
4	ep4	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
3	ep3	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
2	ep2	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
1	ep1	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command
0	ep0	R	0*	no interrupt
			1	corresponding endpoint interrupt is active - can be cleared by issuing the corresponding select endpoint/clear interrupt command

Table 469: Command read interrupt register (read second byte)

Bit	Symbol	Access	Value	Description
7 to 3	-	R	0*	reserved
2	device_status	R	0*	no device status change
			1	device status has changed - this bit is cleared by issuing the get device status command
1 to 0	-	R	0*	reserved

Table 470: Command read current frame number (read first byte)

Bit	Symbol	Access	Value	Description
7 to 0	frame[7:0]	R	0*	contains the least significant part of the frame number of the last successfully received SOF (start of frame) - the full frame number is 11 bits wide - in case the user is only interested in the lower 8 bits of the frame number only the first byte needs to be read; note: in case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF - in case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device

Table 471: Command read current frame number (optional read second byte)

Bit	Symbol	Access	Value	Description
7 to 3	-	R	0*	reserved
2 to 0	frame[10:8]	R	0*	contains the most significant part of the frame number of the last successfully received SOF (start of frame) - the full frame number is 11 bits wide - in case the user is only interested in the lower 8 bits of the frame number only the first byte needs to be read; note: in case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF - in case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device

Table 472: Command read chip identification (read first byte)

Bit	Symbol	Access	Value	Description
7 to 0	chip_id[7:0]	R	0xBE*	contains the least significant part of chip identification - the full chip identification is 16 bits wide

Table 473: Command read chip identification (read second byte)

Bit	Symbol	Access	Value	Description
7 to 0	chip_id[15:8]	R	0x18*	contains the most significant part of chip identification - the full chip identification is 16 bits wide

Table 474: Command set device status (write 1 byte)

Bit	Symbol	Access	Value	Description
7 to 5	-	R	0*	reserved
4	bus_reset	R	0*	indicates that the device received a bus reset - this bit is cleared after read; on a bus reset, the device goes into its default state and responds to address 0; with interrupt
3	suspend_c	R	0*	indicates that the suspend bit has changed, because of one of the following reasons: <ul style="list-style-type: none">• the device enters the suspend state• the device is disconnected• the device receives resume signalling from USB host this bit is cleared after read; with interrupt
2	suspend	R	0*	defines the suspend state of the device - read behavior; without interrupt
			0*	there has been USB activity in the last 3 ms
			1	there has not been any activity on the USB port for more than 3 ms
		W	0	write behavior
			0	when this bit is 1 and the SC writes a 0, the USB block generates a remote (USB host) wake-up if connect = 1; writing a 0 has no effect when it is already in this state or when no USB device is connected
			1	has no effect
1	connect_c	R	0*	indicates that the USB host has been disconnected (vbus disappeared); this bit is cleared after read; with interrupt
0	connect	R/W	0*	sets the current SoftConnect™ status of the device; without interrupt
			0*	SoftConnect™ is inactive, USBCN = 0
			1	SoftConnect™ is active, USBCN = 1

Table 475: Command get device status (read 1 byte)

Bit	Symbol	Access	Value	Description
7 to 5	-	R	0*	reserved
4	bus_reset	R	0*	[1]
3	suspend_c	R	0*	[1]
2	suspend	R	0*	[1]
1	connect_c	R	0*	[1]
0	connect	R	0*	[1]

[1] Bit definition as in command set device status (Table 474)

Table 476: Command get error code (read 1 byte)

Bit	Symbol	Access	Value	Description
7 to 5	-	R	0*	reserved
4	active	R/C	0*	the error code is not active
			1	the error code is active
3 to 0	code[3:0]	R		returns the value of the last occurred error:
			0b0000*	no error
			0b0001	PID encoding error
			0b0010	unknown PID
			0b0011	unexpected packet
			0b0100	error in token CRC
			0b0101	error in data CRC
			0b0110	time-out error
			0b0111	babble (transmission not finished within the boundary of a frame or transmission during SOF)
			0b1000	error in end of packet
			0b1001	sent/received no acknowledge
			0b1010	sent stall
			0b1011	buffer overrun error
			0b1100	sent empty packet (isochronous endpoints only)
			0b1101	bit-stuffing error
			0b1110	error in sync
			0b1111	wrong toggle bit in data PID, ignored data

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 477: Command select endpoint (optional read 1 byte)

Bit	Symbol	Access	Value	Description
7	-	R	0*	reserved
6	b_2_full	R		double buffered endpoints: buffer 2 status. The additional information will help to ensure that the last packet for the IN endpoint has been successfully received by the host.
			0*	empty
			1	full
5	b_1_full	R		double buffered endpoints: buffer 1 status. The additional information will help to ensure that the last packet for the IN endpoint has been successfully received by the host.
			0*	empty
			1	full
4	naked	R		all endpoints except iso; this bit is set when a NAK is sent and the inton_nak feature is enabled - this bit is reset after the device has sent an ACK after an OUT packet or when the device has seen an ACK after sending an IN packet
			0*	The device has not sent a NAK
			1	the USB block has sent a NAK to the host, if the host sends an OUT packet to a filled OUT buffer and if the host sends an IN token to an empty IN buffer
3	overwritten	R		endpoints control; the value of this bit is cleared by the select endpoint clear interrupt command
			0*	the previously received packet was not been overwritten by a setup packet
			1	the previously received packet was overwritten by a setup packet
2	setup	R		endpoints control; the value of this bit is updated after each successfully received packet - the value of this bit is cleared by the select endpoint/clear interrupt command
			0*	the last received packet for the selected endpoint was not a setup packet
			1	the last received packet for the selected endpoint was a setup packet
1	stall	R		all endpoints except iso
			0*	the selected endpoint is not stalled
			1	the selected endpoint is stalled
0	full/empty	R		all endpoints; the buffer of the selected endpoint is full/empty
			0*	for IN endpoints, the next write buffer is empty
			1	for OUT endpoints, the next read buffer is full

[1] This command initializes an internal pointer to the start of the selected buffer - this command can be optionally followed by a data read which returns some information on the packet in the buffer (the bits in this register).

Table 478: Command select endpoint/clear interrupt (read 1 byte)

Bit	Symbol	Access	Value	Description
7	-	R	0*	reserved
6	b2_full	R	0*	[2]
5	b1_full	R	0*	[2]
4	naked	R	0*	[2]
3	overwritten	R	0*	[2]
2	setup	R	0*	[2]
1	stall	R	0*	[2]
0	full/empty	R	0*	[2]

[1] This command distinguishes itself from the select endpoint as it clears the interrupt associated to the selected endpoint - in case of control OUT endpoint, the **setup** and the **overwritten** bits get cleared too.

[2] Bit definition as in command select endpoint ([Table 477](#))

Table 479: Command set endpoint status (write 1 byte)

Bit	Symbol	Access	Value	Description
7	condstall	W	0*	endpoints control OUT
			1	stall control OUT endpoint unless the packet setup bit is set
6	rate	W	0*	interrupt endpoint is in Toggle mode; endpoints interrupt IN
			1	interrupt endpoint is in rate Feedback mode
5	disable	W	0*	endpoints interrupt bulk iso
			1	the endpoint is disabled - after a bus reset each endpoint is disabled
4 to 1	-	W	0*	reserved
0	stall	W	0*	the endpoint is un-stalled - clearing this bit initializes the endpoint; endpoints control interrupt bulk
			1	the endpoint is stalled

[1] A stalled control endpoint is automatically un-stalled when it receives a setup token, regardless of the content of the packet. If the endpoint should stay in its stalled state, the SC can re-stall it. When a stalled endpoint is un-stalled, either by the set endpoint status command or by receiving a setup token, it is also re-initialized. This flushes the buffer: in case of an OUT buffer it waits for a DATA 0 PID, in case of an IN buffer it writes a DATA 0 PID. When an endpoint is stalled by the set endpoint status command it is also re-initialized.

Table 480: Command clear buffer (optional read 1 byte)

Bit	Symbol	Access	Value	Description
7 to 1	-	R	0*	reserved
0	overwritten	R	0*	endpoints control OUT
			1	the previously received packet was overwritten by a setup packet - this bit can be cleared with the select endpoint/clear interrupt command for control OUT 0

[1] When a packet sent by the host has been received successfully, an internal endpoint buffer full flag is set. All subsequent packets will be refused by returning a NAK. When the SC has read the data, it should free the buffer with the clear buffer command. When the buffer is cleared new packets will be accepted. A buffer cannot be cleared when its packet **overwritten** bit is set.

Table 481:Command validate buffer (no data phase)

Bit	Symbol	Access	Value	Description
7 to 0	-	-	~*	-

- [1] When the SC has written data into an IN buffer (even in case of empty packet), it should set the buffer full flag by the validate buffer command. This indicates that the data in the buffer is valid and can be sent to the host when the next IN token is received. A control IN buffer cannot be validated when the packet **overwritten** bit of its corresponding OUT buffer is set.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

9.29.6 Application information**9.29.6.1 Approximation of the bulk buffer reload time**

In order to estimate the time needed by the ARM to transfer 64 bytes we take the assumption that 16 read cycles and 16 write cycles are necessary. Each access to the VPB bus takes 3 periods, and each access to the SCRAM 1 cycle. The ARM code making the transfer is placed in SCRAM (or even better in D\$ cache or ITCM). The SC interrupt handler, register access and acknowledge are not taken into account. No load multiple/store multiple is possible. This gives:

$$T_{ARM} = 5 \text{ ns} \times 16 \times (1 + 3 + 1 + 1) = 2.4 \mu\text{s}.$$

The transfer on the USB bus (including 13 bytes overhead for header/CRC) takes:

$$T_{USB} = (64 + 13) \times 8 \times 83 \text{ ns} = 51 \mu\text{s}$$

So the SC load time represents less than 5% of the USB transmission time.

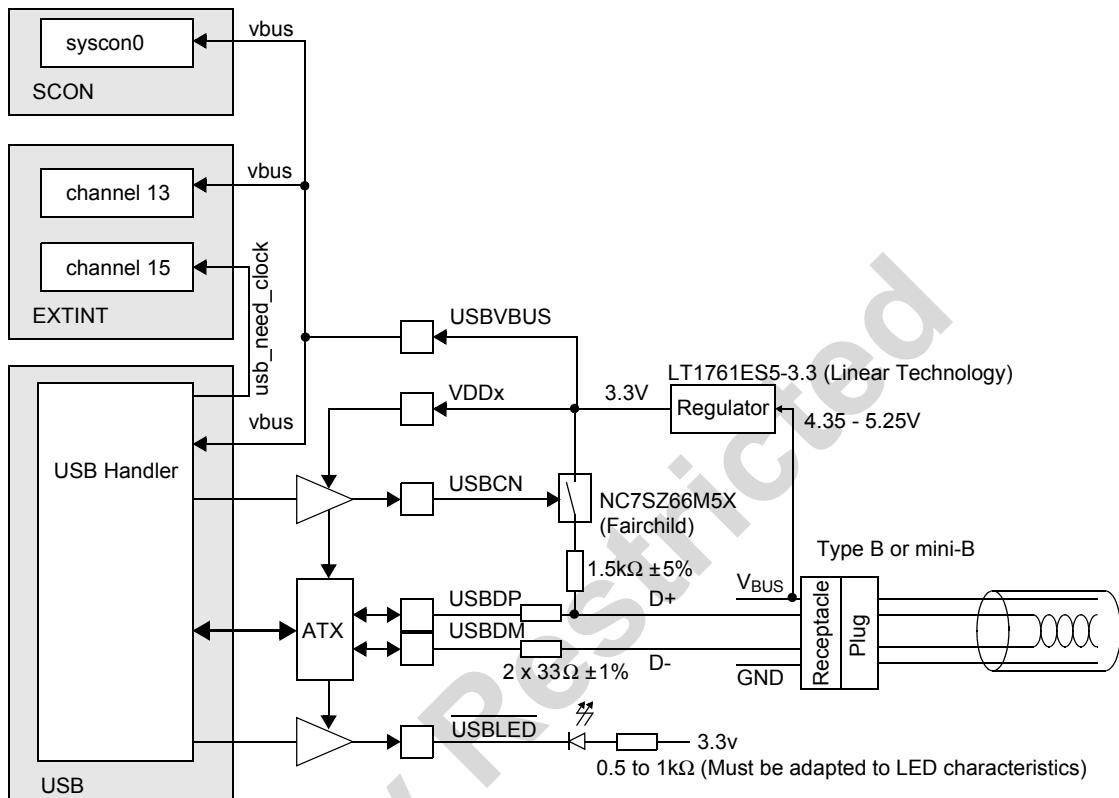
9.29.6.2 External circuitry

The external circuitry which is recommended for the USB block is drawn in [Figure 189](#). The VBUS voltage from the USB cable is nominally at 5 V at the source. The user has to make sure that all voltages applied to the USB port of the DVFD818x Family are in the range between 2.8 and 3.3 V. External circuitry like drawn in the figure should make sure that this range is maintained.

The USBCN pin shall drive a 1.5 k Ω external resistor which pulls up the USBDP line for the SoftConnect™ feature for the 12 Mbit/s USB transfer rate.

Two external 33 Ω serial resistors must be added on USBDP and USBDM for impedance matching.

If the user requires detection of connection and disconnection of the USB cable, a GPIO or EXTINT pin can be used. The connection of a GPIO EXTINT pin working at V_{DDH} is proposed.



(1) Additional ESD protections not shown

Fig 189.Typical external circuitry for USB block (de coupling capacitors not shown)

Note: `VDDx` is `VDDH` for DVFD818x

9.29.6.3 Debouncing of `usb_need_clock`

In order to reduce the wake-up time from suspend, it is recommended to disable or reduce to minimum the debouncing for EXTINT15.

9.29.6.4 Register access

The following procedures for access to the registers should be respected:

- poll the `cmd_code_empty_int` bit to know when a command has been processed
- poll the `cmd_data_full_int` bit to know when the data to be read is available

Complementary document

10. BMP - Burst Mode Processor

CAUTION



The BMP is not available at DVFD8187!

10.1 Introduction

This chapter describes the function of the BMP first at a high level and then in more detail.

Tasks of the BMP are:

- support bit and slot formatting
- support error control coding and decoding
- generate timing references
- control RF timing
- Symbol recovery

The BMP of the DVFD8185 is fully under software control. Hence, the slot and frame format is highly flexible and can be adjusted to the need of the user's system. The BMP supports digital cordless systems for DECT, Bluetooth and for the ISM band in the U.S. It can be altered during operation to cover different requirements during different operation modes.

The BMP is switched on and off by the register bit BMP_ON in the BMP global register **bmp_global**.

With the bit BLRES_BMP within the global control register **bmp_global** a software controlled reset of the burst mode processor can be initiated.

10.1.1 Packet formats for DECT

The following slot formats are supported:

- DECT half slots
- DECT full slots
- DECT long slots
- DECT double slots
- Zero blind slot
- Proprietary slot formats

Preamble	Syncword	A-Field	B-Field	X-Field	Z-Field
16 / 32	16	64	0 / 80 / 320 / 640 / 800	4	0 / 4
Preamble	Syncword	A-Field	B-Field	CRC32	X-Field
16 / 32	16	64	640	32	4

Fig 190.DECT Packets

10.1.2 Bit rates

The bit rate of the air interface can be selected to be either 115.2 kbit/s, 288 kbit/s, 384 kbit/s, 576 kbit/s, 1000 kbit/s or 1152 kbit/s. The latter one is for DECT, the other bit rates are for proprietary protocols which may need different slot and frame formats.

10.2 Block diagram

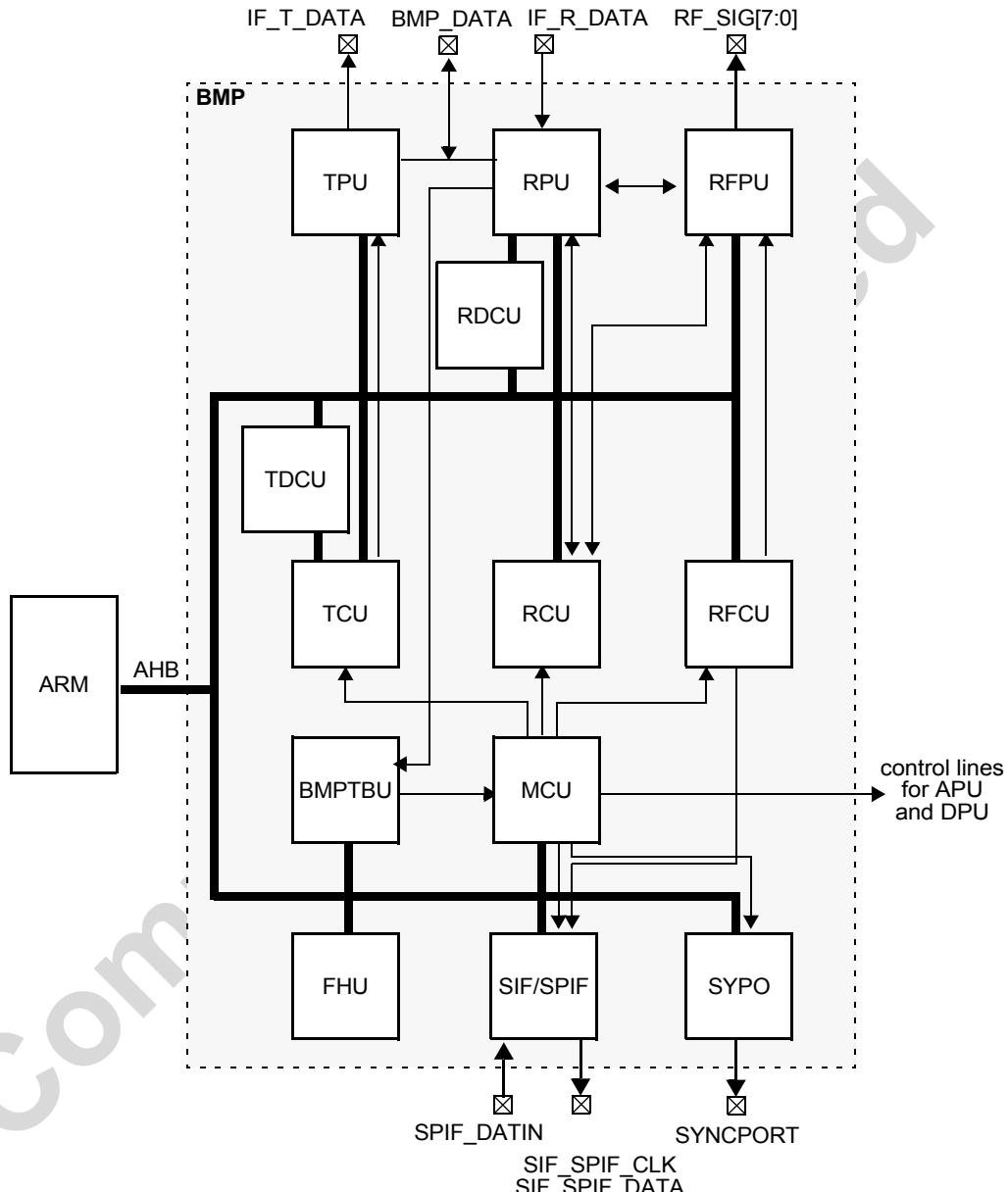


Fig 191. Architecture of Burst Mode Processor BMP

IF_R_DATA and IF_T_DATA in Figure 191 are pins on the interface to the APU.

Figure 191 gives an overview of the architecture of the Burst Mode Processor BMP. It is fully under software control and can therefore easily be adjusted to the various needs of prospective systems. Slot and frame formats may be changed during runtime. The thin lines in Figure 191 show the dataflow within the BMP.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

The Burst Mode Processor Time Base Unit BMPTBU generates the system reference timing, i.e. it contains the bit- and frame counters.	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
The Main Control Unit MCU is the master of the BMP. It controls all other Control Units (TCU, RCU, RFCU), the analog blocks of the IC, and takes over various other control functions. The timing of the MCU is controlled by the BMPTBU.	
The RF Control Unit RFCU generates timing critical signals that are similar for each packet. A typical example is the transmit power amplifier control: For each transmit slot it is enabled a few bits before the packet actually starts and disabled again at the end of the packet.	
The Transmit Control Unit TCU generates control signals for the Transmit Peripheral Unit TPU. It determines the transmit packet length, which bits are used for CRC calculation, which bits are scrambled, and which bits are encrypted. The TCU is under software control and is started by the Main Control Unit. Hence, it may be started at any bit- and frame counter value.	
The Transmit Data Cache Unit TDCU is the HW machine to control the TDATA cache area, for fetching of data for the transmit path.	
The Transmit data is written to the Transmit Peripheral Unit TPU by the system controller via the Advanced System Bus MLAHB. This data is processed in the TPU, for example CRC checks may be added and certain bit sequences may be scrambled and/or encrypted.	
The Receive Control Unit RCU generates control signals for the Receive Peripheral Unit RPU. It determines the receive packet length, which bits are used for CRC checks, which bits are to be de scrambled, which bits are to be decrypted and optionally which bits are to be FEC decoded. The RCU is under software control and may either be started by the MCU or by the RPU if a synchronization word and - optionally - an identity was detected.	
The Receive Data Cache Unit RDCU is the HW machine to control the RDATA cache area, for storing of data for the receive path.	
Data received and demodulated by the RF module is fed into the Receive Peripheral Unit RPU which recovers the bitclock, descrambles and decrypts data if required, detects the synchronization word and optionally the identity, checks CRCs, does error correction and buffers the data in registers from where it is picked up by the system controller.	
The serial interface SIF/SPIF is used to program the synthesizer. The programming speed as well as the length of the packet is programmable. With this flexibility it may also be used to transmit timing data for the RF module. Data of the RF module can be read by the SPIF interface only (e.g. RSSI value).	
Task of the synchronization port is to synchronize clusters of DECT base stations to mutually increase traffic capacity of adjacent systems by aligning guard bands (Class 1 device) and to support synchronous handovers between the base stations (Class 2 device). The SYPO is compliant to the more stringent Class 2 specification.	

10.3 Hardware Interface

Table 482:BMP signal overview (function not available at DVFD8187)

PIN	Name	I/O	Description
BMP data signal			
IF_T_DATA	RF transmit data	O	RF transmit data serial from BMP
IF_R_DATA	RF receive data	I	RF receive data signal from ABS
RF_SIG[2] / BMP_DATA ^[1]	RF timing output / RF receive/transmit data	O / I/O	RF timing output [2] / digital receive/transmit data from/to RF module
SYNCPORT	synchronization port	I/O	DECT synchronization port
SYNC_MTCH	Correlator sync match event	O	Correlator sync match event indication output
Radio programming signal			
RF_SIG[7:0]	RF timing output	O	RF timing output
SIF_SPIF_CLK	Serial interface clock	O	RF programming bus clock
SIF_SPIF_DATA	Serial interface data out	O	RF programming bus data output
SPIF_DATIN ^[2]	Serial interface data in	I	RF programming bus data input

[1] Used as receive/transmit signal when operating with RF19

[2] Used only when operating with RF19

Table 483:Clocks generated by the BMP clock generation unit

Clock	freq. DECT	freq. ISM	freq. 2.4GHz	enable	phase correction	Description
clk_drt	3.456MHz	3.456MHz	3.456MHz	drt_global.dr_t_on	yes	CODEC clock
clk_fr	2.7648kHz	2.7648kHz	2.7648kHz	-	yes	frame clock
clk_bit	1.152MHz	0.1152MHz	0.576MHz	-	yes	bit clock
clk_o	13.824MHz	1.3824MHz	6.912MHz	BMP_ON	no	12 times oversampled bit clock
clk_mem	gated bmp_hclk	gated bmp_hclk	gated bmp_hclk	en_clk_mem	no	clock for command word array
clk_regs	gated bmp_hclk	gated bmp_hclk	gated bmp_hclk	en_clk_regs	no	clock for register access
clk_m	gated clk_bit	gated clk_bit	gated clk_bit	m_event, rfcu_event	yes	clocks the MCU
clk_t	gated clk_bit	gated clk_bit	gated clk_bit	en_clk_t	yes	clocks the TCU
clk_r	gated clk_o	gated clk_o	gated clk_o	en_clk_r	no, sync to receive signal (DPLL)	clocks the RCU (controlled by R_pll or correlator)

Table 483: Clocks generated by the BMP clock generation unit...continued

Clock	freq. DECT	freq. ISM	freq. 2.4GHz	enable	phase correction	Description
clk_r_sync	gated clk_o	gated clk_o	gated clk_o	en_clk_r, id_win, sync_win	no, sync to receive signal (DPLL)	clock for sync word and id detection blocks
clk_c	gated clk_o	gated clk_o	gated clk_o	en_clk_c	no	clocks the cipher engine
clk_sif	gated clk_bmp	gated clk_bmp	gated clk_bmp	en_clk_sif	no	clocks the SIF / SPIF

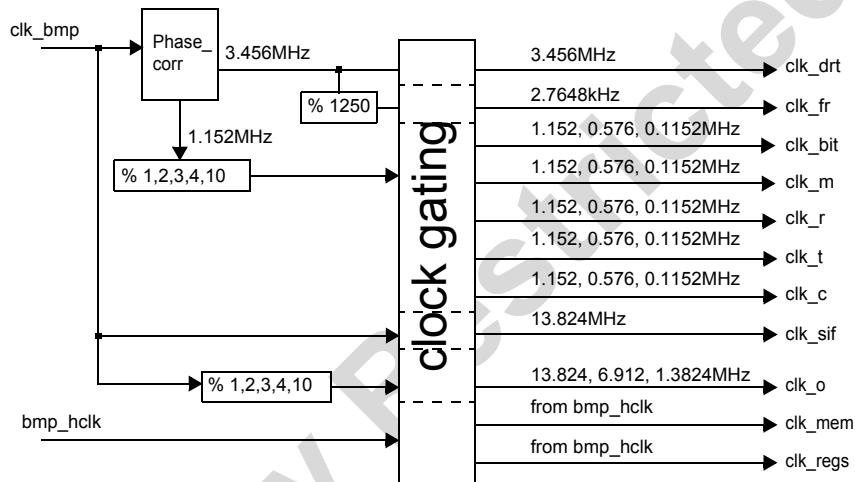


Fig 192.Clock domains BMP (DECT/ISM)

10.4 Software interface

Table 484: Register overview of BMP

Name	Description	I/O	reset
bmp_global	BMP global control register	R/W	0x0000 0020
bmp_bcnt_ref	BCNT reference values	R/W	0x0000 0000
bmp_bcnt_s_ref	BCNT_S reference values	R/W	0x0000 0000
bmp_bcnt_val	Bit counter value	R	0x0000 0000
bmp_ref	Reference value for system controller walk-up	R/W	0xFFFF FFFF
bmp_cnt_mod	Modulo values bit counters	R/W	0x0FFF 3FFF
bmp_com_addr0	Command word array address offsets 0	R/W	0x0000 0000
bmp_com_addr1	Command word array address offsets 1	R/W	0x0000 0000
bmp_com_addr2	Command word array address offsets 2	R/W	0x0000 0000
bmp_com_addr3	Command word array address offsets 3	R/W	0x0000 0000
bmp_cp_x_time	X_time_buf copy signals	R/W	0x0000 0000
bmp_crc_con0	CRC control register 0	R/W	0x0000 0000
bmp_crc_con1	CRC control register 1	R/W	0x0000 0000
bmp_crc_con2	CRC control register 2	R/W	0x0000 FFFF
bmp_crc_con3	32bit CRC control register	R/W	0x4C1 1DB7

Table 484: Register overview of BMP...continued

Name	Description	I/O	reset
bmp_crc_con4	32bit CRC control register	R/W	0x0000 0005
bmp_crc_con5	32bit CRC control register	R/W	0xFFFF FFFF
bmp_enc_dat0	Cipher engine initialization data 0	W	0x0000 0000
bmp_enc_dat1	Cipher engine initialization data 1	W	0x0000 0000
bmp_enc_dat2	Cipher engine initialization data 2	W	0x0000 0000
bmp_enc_dat3	Cipher engine initialization data 3	W	0x0000 0000
bmp_enc_dat4	Cipher engine initialization data 4	R/W	0x0000 0000
bmp_enc_dat5	Cipher engine initialization data 5	R/W	0x0000 0000
bmp_enc_dat6	Cipher engine initialization data 6	R/W	0x0000 0000
bmp_fcnt_mod	Frame counter modulo value	R/W	0xFFFF FFFF
bmp_fcnt_val	Frame counter value	R/W	0x0000 0000
bmp_m_pointer	MCU command array pointer	R/W	0x0000 0000
bmp_max_error	Maximum measurable phase error	R/W	0x0001 FFFF
bmp_phase_con	Phase Correction Control	R/W	0x0000 0000
bmp_phase_error	Phase Error	R/W	0x0000 0000
bmp_phase_offset	Phase Offset	R/W	0x0000 0000
bmp_r_buf_con	Length of r_buf	R/W	0x0000 0000
bmp_r_buf0	Receive data buffer 0	R	0x0000 0000
bmp_r_buf1	Receive data buffer 1	R	0x0000 0000
bmp_r_pointer	RCU command array pointer	R/W	0x0000 0000
bmp_rf_pointer	RFCU command array pointer	R/W	0x0000 0000
bmp_rfpu_con	RFPU control register	R/W	0x0000 0000
bmp_rpu_con0	Receive Peripheral Units control register 0	R/W	0x0000 0000
bmp_rpu_con1	Receive Peripheral Units control register 1	R/W	0x0000 0000
bmp_rpu_pat	Receive Peripheral Units pattern register	R/W	0x0540 0000
bmp_rpu_stat0	Receive Peripheral Units status register 0	R/W	0x0000 0000
bmp_rpu_stat1	Receive Peripheral Units status register 1	R	0x0000 0000
bmp_rpu_stat2	Receive Peripheral Units status register 2 32bit CRC DUMP register	R	0x0000 0000
bmp_rpu_stat3	Receive Peripheral Units status register 1 (for correlator)	R	0x0002 0000
bmp_scnt_mod	Modulo values of slot counters Scnt0 and Scnt1	R/W	0x0000 0000
bmp_scnt_val	Current values of slot counters Scnt0 and Scnt1	R/W	0x0000 0000
bmp_sif_buf0	Serial interface buffer 0 (shared with bmp_spif_adr)	R/W	0x0000 0000
bmp_sif_buf1	Serial interface buffer 1 (shared with bmp_spif_datout)	R/W	0x0000 0000
bmp_sif_buf2	Serial interface buffer 2 (shared with bmp_spif_datin)	R/W	0x0000 0000
bmp_sif_con	Serial interface control register (shared with bmp_spif_con)	R/W	0x0000 0000
bmp_spif_adr	Serial peripheral interface address register (shared with bmp_sif_buf0)	R/W	0x0000 0000
bmp_spif_con	Serial peripheral interface control register (shared with bmp_sif_con)	R/W	0x0000 0000
bmp_spif_datout	Serial peripheral interface data out register (shared with bmp_sif_buf1)	R/W	0x0000 0000

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 484: Register overview of BMP...continued

Name	Description	I/O	reset
bmp_spif_datin	Serial peripheral interface data in register (shared with bmp_sif_buf2)	R	0x0000 0000
bmp_start_time	Receive Control Unit start time register	R	0x0000 0000
bmp_symrec_con0	Symbolrecovery control register 0 (shared with bmp_sync_pat0)	R/W	0x0000 0000
bmp_symrec_con1	Symbolrecovery control register 1 (shared with bmp_sync_pat1)	R/W	0x0000 0000
bmp_sync_pat0	Synchronization pattern 0 (shared with bmp_symrec_con0)	R/W	0x0000 0000
bmp_sync_pat1	Synchronization pattern 1 (shared with bmp_symrec_con1)	R/W	0x0000 0000
bmp_sypo_con	Synchronization port control register	R/W	0x0000 0000
bmp_sypo_ref_cnt	SYNCPORT reference value	R	0x0000 0000
bmp_t_buf_len	Length of T_BUF	R/W	0x0000 0000
bmp_t_buf0	Transmit data buffer 0	R/W	0x0000 0000
bmp_t_buf1	Transmit data buffer 1	R/W	0x0000 0000
bmp_t_pointer	TCU command array pointer	R/W	0x0000 0000
bmp_tpu_con	Transmit Peripheral Unit Control register	R/W	0x0000 0000
bmp_com_add_rda	BMP RAM RDATA cache, START and END addresses ta	R/W	0x0000 0000
bmp_com_add_tda	BMP RAM TDATA cache, START and END addresses ta	R/W	0x0000 0000
bmp_rdata_pointer	BMP RAM RDATA cache pointer	R/W	0x0000 0000
bmp_tdata_pointer	BMP RAM TDATA cache pointer	R/W	0x0000 0000
bmp_ram	Command word array (2kByte)	R/W	random

[1] The reset value might changes for the different versions of DVFD8185.

Table 485: bmp_global - Burst mode processor global control register

Bit	Symbol	Access	Value	Description
31:12	-	R	0*	reserved, must be set to '0'
11	DPLL_SYMREC	R/W	1	Symbol recovery unit selection Correlator and the shared registers bmp_symrec_con0, bmp_symrec_con1, bmp_spif_adr, bmp_spif_con, bmp_spif_datout, bmp_spif_datin are selected. SPIF_CLK is routed to pin SIF_SPIF_CLK SPIF_DATA is routed to pin SIF_SPIF_DATA signal int_spif is routed to int_sif
			0*	DPLL or IQ demodulator and the shared registers bmp_sync_pat0, bmp_sync_pat1, bmp_sif_buf0, bmp_sif_buf1, bmp_sif_buf2, bmp_sif_con are selected. SIF_CLK is routed to pin SIF_SPIF_CLK SIF_DATA is routed to pin SIF_SPIF_DATA signal int_sif_origin is routed to int_sif
10:9	-	R	0*	reserved, must be set to '0'
8	-	R/W	0*	reserved, must be set to '0'
7	SEL_TBU2	R/W	0*	Selects time base unit mode, to be used in combination with SEL_TBU2, [SEL_TBU2: SEL_TBU] [2]

Table 485: bmp_global - Burst mode processor global control register...continued

Bit	Symbol	Access	Value	Description
6 ETCH	AUTO_HW_DATA_F	R/W		data fetch activation (see Section 10.5)
			1	active
			0*	inactive
5:3	BITCLOCK_SEL	R/W		clk_bit frequency selection
			111	reserved
			110	reserved
			101	reserved
			100*	1.152 MHz if [SEL_TBU2:SEL_TBU] = 0b00
			011	0.576 MHz ^[3]
			010	0.384 MHz ^[3]
			001	0.288 MHz ^[3]
			000	0.1152 MHz ^[3]
2	SEL_TBU	R/W	0*	Selects time base unit mode, to be used in combination with SEL_TBU2, [SEL_TBU2:SEL_TBU] ^[2]
1	BLRES_BMP	R/W	0*	Block reset Burst Mode Processor
0	BMP_ON	R/W	0*	Burst Mode Processor on
[1] Note that a block reset via BLRES_BMP should be done only after the block has been turned on via BMP_ON.				
[2] [SEL_TBU2:SEL_TBU] 0b00: DECT/ISM mode, default 0b01: reserved 0b10: reserved 0b11: reserved				
[3] Only if [SEL_TBU2:SEL_TBU] = 0b00				
[4] The 24 rate oversampling can only be used in combination with a clk_bmp of 27.648 MHz				

Table 486: bmp_bcnt_ref - BCNT reference values

Bit	Symbol	Access	Value	Description
31:16	BCNT_REF1	R/W	0*	Bcnt reference value 1
15:0	BCNT_REF0	R/W	0*	Bcnt reference value 0

[1] Only the lower 14 bits are valid

Table 487: bmp_bcnt_s_ref - BCNT_S reference values

Bit	Symbol	Access	Value	Description
31:16	BCNT_S_REF1	R/W	0*	Bcnt reference value 1
15:0	BCNT_S_REF0	R/W	0*	Bcnt reference value 0

[1] Only the lower 12 bits are valid

Table 488: bmp_bcnt_val - Bit counter value

Bit	Symbol	Access	Value	Description
31:14	reserved	R	0*	must be set to 0
13:0	BCNT_VAL	R	0*	Current value of bit counter Bcnt

Table 489:bmp_ref - BMP timer reference value

Bit	Symbol	Access	Value	Description
31:0	BMP_REF	R/W	all 1*	timer reference value

Table 490:bmp_cnt_mod - Modulo values bit counters

Bit	Symbol	Access	Value	Description
31:28	-	R	0*	reserved
27:16	BCNT_S_MOD	R/W	all 1*	Modulo of short bit counter Bcnt_s
15:14	-	R	0*	reserved
13:0	BCNT_MOD	R/W	all 1*	Modulo of bit counter Bcnt

[1] Only the lower 14 bits are valid for BCNT_MOD and the lower 12 bits for BCNT_S_MOD

Table 491:bmp_com_add0 - Command word array address offset 0

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
23:17	-	R	0*	reserved
24 &15:8	M_COM_END	R/W	0*	M_com array end address offset (9 bits)
16 & 7:0	M_COM_STA	R/W	0*	M_com array start address offset (9 bits)

[1] The actual address of entry 0 of:
M_com is: 0xC180 1000 + M_COM_STA * 4

Table 492:bmp_com_add1 - Command word array address offset 1

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
23:17	-	R	0*	reserved
24 &15:8	RF_COM_END	R/W	0*	RF_com array end address offset (9 bits)
16 & 7:0	RF_COM_STA	R/W	0*	RF_com array start address offset (9 bits)

[1] The actual address of entry 0 of:
RF_com is: 0xC180 1000 + RF_COM_STA * 4

Table 493:bmp_com_add2 - Command word array address offset 2

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
23:17	-	R	0*	reserved
24 &15:8	T_COM_END	R/W	0*	T_com array end address offset (9 bits)
16 & 7:0	T_COM_STA	R/W	0*	T_com array start address offset (9 bits)

[1] The actual address of entry 0 of:
T_com is: 0xC180 1000 + T_COM_STA * 4

Table 494:bmp_com_add3 - Command word array address offset 3

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
23:17	-	R	0*	reserved
24 & 15:8	R_COM_END	R/W	0*	R_com array end address offset (9 bits)
16 & 7:0	R_COM_STA	R/W	0*	R_com array start address offset (9 bits)

[1] The actual address of entry 0 of:
R_com is: 0xC180 1000 + R_COM_STA * 4

Table 495:bmp_cp_x_time - Copy signals for X_time_bufs

Bit	Symbol	Access	Value	Description
31:4	-	R	all 0*	reserved
3	FORCE_M_EV	R/W	0*	Force an m_event signal. Automatic reset.
2	FORCE_RF_EV	R/W	0*	Force an rf_event signal. Automatic reset
1	FORCE_R_EV	R/W	0*	Force an r_event_signal. Automatic reset
0	FORCE_T_EV	R/W	0*	Force an t_event signal. Automatic reset

[1] The corresponding x_POINTER registers are incremented when the CP_x_TIME bits are set.

Table 496:bmp_crc_con0 - CRC control register 0

Bit	Symbol	Access	Value	Description
31:18	-	R	0*	reserved
17	CRC_LEN0	R/W		Determines number of CRC check bits
			1	CRC is 16 bits wide
			0*	CRC is 8 bits wide
16:0	CRC_POL0	R/W	0*	Determines CRC polynomial 0

[1] See Notes of [Table 497](#) for recommended CRC polynomials.

Table 497:bmp_crc_con1 - CRC control register 1

Bit	Symbol	Access	Value	Description
31:18	-	R	0*	reserved
17	CRC_LEN1	R/W		Determines number of CRC check bits
			1	CRC is 16 bits wide
			0*	CRC is 8 bits wide
16:0	CRC_POL1	R/W	0*	Determines CRC polynomial 1

- [1] For a (24,16) code the polynomial $x^8+x^5+x^2+x+1$, i.e. $\text{CRC_POL}_x = 0\ 0000\ 0001\ 0010\ 0111$, has the best error detection performance.
[2] For a (40,32) code the polynomial $x^8+x^5+x^2+1$, i.e. $\text{CRC_POL}_x = 0\ 0000\ 0001\ 0010\ 0101$, has the best error detection performance.

Table 498:bmp_crc_con2 - CRC control register 2

Bit	Symbol	Access	Value	Description
31:17	-	R	0*	reserved
16	DIS_INV	R/W	0*	disable last bit inversion of calculated CRC
15:0	PRE_CRC	R/W	all 1*	preset value of CRC register

Table 499:bmp_crc_con3 - CRC control register 3 (32bit CRC)

Bit	Symbol	Access	Value	Description
31:0	CRC_POL2[31:0]	R/W	0x4C11D B7*	32 LSBs of 33bit polynomial CRC register

Table 500:bmp_crc_con4 - CRC control register (32bit CRC)

Bit	Symbol	Access	Value	Description
31:4	-	R	0*	reserved
3	DIS_ONES_COMP	R/W		disable one's complement of calculated CRC-32
			1	one's complement is disabled
			0*	one's complement is calculated
2:1	CRC_LEN2[1:0]	R/W		Determines number of CRC check bits
			11	reserved
			10*	CRC is 32 bits wide
			01	CRC is 16 bits wide
			00	CRC is 8 bits wide
0	CRC_POL2[32]	R/W	1*	MSB of 33bit polynomial CRC register

- [1] DIS_ONES_COMP functions in exactly the same way on both the RX and TX paths. That is when enabled it in effect inverts all the bits of the calculated CRC result for transmission and at the receive side it causes the CRC bits to be considered as inverted data bits. This results in a final checksum or crc dump of all zeros. The alternative would be to not consider the polarity of the received CRC and then the expected crc dump would be 0xC704DD7B. This has been confirmed by sending a CRC with DIS_ONES_COMP enabled and receiving the CRC with DIS_ONES_COMP disabled.

Table 501:bmp_crc_con5 - 32bit CRC control register

Bit	Symbol	Access	Value	Description
31:0	PRE_CRC2[31:0]	R/W	all 1*	preset value of CRC-32 register

Table 502:bmp_enc_dat0 - Cipher engine initialization data 0

Bit	Symbol	Access	Value	Description
31:0	IV0	W	0*	initialization vector bits [31:0]

Table 503:bmp_enc_dat1 - Cipher engine initialization data 1

Bit	Symbol	Access	Value	Description
31:0	IV1	W	0*	initialization vector bits [63:32]

Table 504:bmp_enc_dat2 - Cipher engine initialization data 2

Bit	Symbol	Access	Value	Description
31:0	CK0	W	0*	encryption key bits [31:0]

Table 505: bmp_enc_dat3 - Cipher engine initialization data 3

Bit	Symbol	Access	Value	Description
31:0	CK1	W	0*	encryption key bits [63:32]

Table 506: bmp_enc_dat4 - Cipher engine initialization data 4

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13	CT_DECODE	R/W		Decodes the succeeding two bits after start_x_enc to determine if A-field must be encrypted
			1	Bits a ₀ and a ₁ in A-field are decoded to enable encryption in A-field if both bits are '0' (i.e. a CT message is present)
			0*	No decoding of bits in the A-field will take place. Encryption starts immediately. Default for ISM systems.
12	ENCR_EN	R/W		Encryption enable
			1	Encryption engine will start on the next rising edge of start_ini_enc / start_t_enc / start_r_dec
			0*	Encryption engine disabled independent from start_ini_enc / start_t_enc / start_r_dec
11	PRECLOCK	R/W		Preclocking of Encryption engine enable
			1	Encryption engine is preclocked with SEQ_LEN clocks
			0*	Encryption engine is not preclocked
10:0	SEQ_LEN	R/W	0*	Determines encryption sequence length and the number of preclocking cycles ^[1]

- [1] The sequence length must be set to
 a) 120 for DECT half slots
 b) 360 for DECT full slots
 c) 840 for DECT double slots
 d) any value for proprietary protocols with e.g. the B-field encrypted

Table 507: bmp_enc_dat5 - Cipher engine initialization data 5

Bit	Symbol	Access	Value	Description
31:0	-	R	0*	reserved

Table 508: bmp_enc_dat6 - Cipher engine initialization data 6

Bit	Symbol	Access	Value	Description
31:0	-	R	0*	reserved

Table 509: bmp_fcnt_mod - Frame counter modulo value

Bit	Symbol	Access	Value	Description
31:0	FCNT_MOD	R/W	all 1*	Modulobits [26:1] of the master real time frame counter used for cipher engine initialization of frame counter Fcnt

Table 510: bmp_fcnt_val - Frame counter value

Bit	Symbol	Access	Value	Description
31:0	FCNT_VAL	R/W	0*	Current frame counter value

Table 511: bmp_m_pointer - MCU command array pointer

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	M_POINTER	R/W	0*	Current value of M_point in MCU

[1] A write to this register has effect latest at the transition to the next bit.

Table 512: bmp_max_error - Maximum measurable phase error

Bit	Symbol	Access	Value	Description
31:17	-	R	0*	reserved
16:0	MAX_ERROR	R/W	all 1*	Absolute value of maximum measurable phase error with a resolution of one 13.824 MHz cycle, i.e. ~ 72 ns. If this value is exceeded no phase correction will be performed.

Table 513: bmp_phase_con - Phase correction control register

Bit	Symbol	Access	Value	Description
31:5	-	R	0*	reserved
4:3	CORR_SPEED	R/W		Selects phase adjustment speed [1]
			11	72 ns per 868 ns (1.152 MHz cycle)
			10	8 x 72 ns (13.824 MHz cycle) per 125 us (int_fsi cycle)
			01	4 x 72 ns (13.824 MHz cycle) per 125 us (int_fsi cycle)
			00*	2 x 72 ns (13.824 MHz cycle) per 125 us (int_fsi cycle)
2:1	SYNC_SOURCE_SE L	R/W		Selects synchronization source
			11	SYNCPORT, i.e. external timing reference is SYNCPORT pin.
			10	IOM, i.e. external timing reference is FSC_IOM pin.
			01	RCU, i.e. external timing reference are received(via air) data packets.
			00*	None. No phase correction performed.
0	MEAS_ONLY	R/W		Phase measurement control
			1	PHASE_ERR is measured but no phase correction is performed.
			0*	PHASE_ERR is measured and phase correction is performed.

[1] Phase corrections are executed at the beginning of a speech frame (125 us, i.e. one int_fsi cycle), i.e. right after the int_fsi pulse.

Table 514: bmp_phase_error - Phase error register

Bit	Symbol	Access	Value	Description
31:20	-	R	0*	reserved
19	PHASE_CORR_IN_PROG	R	0*	Status of phase error measurement. If '1' then phase correction is still in progress.
18	MAX_PHASE_ERR_E_XC	R	0*	Status of phase error measurement. If '1' then out of range and phase measurement state machine goes into idle state. Automatically reset to '0' at the start of a phase measurement.

Table 514: bmp_phase_error - Phase error register ...continued

Bit	Symbol	Access	Value	Description
17	PHASE_ERROR_S	R/W		Sign of phase error between internal and external timing reference
			1	negative sign, i.e. internal clock too slow
			0*	positive sign, i.e. internal clock too fast
16:0	PHASE_ERROR	R/W	0*	Absolute value of phase error

- [1] A write to this register has effect latest at the transition to the next bit.
 [2] Note that if a phase correction is currently being performed a phase error measurement can not be done simultaneously.
 [3] This register is meant as status register only. Note that a write operation from the system controller is possible but might disturb BMP firmware operation.

Table 515: bmp_phase_offset - Phase offset register

Bit	Symbol	Access	Value	Description
31:18	-	R	0*	reserved
17	PHASE_OFFSET_S	R/W		Sign of phase error offset
			1	negative sign, i.e. internal clock too slow
			0*	positive sign, i.e. internal clock too fast
16:0	PHASE_OFFSET	R/W	0*	Absolute value of phase error offset

Table 516: bmp_r_buf_con - Length of R_BUF

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8	CP_R_BUF_LEN	R/W	0*	Copies content of R_BUF_LEN into an internal register of R_buf. Automatic reset by hardware.
7:0	R_BUF_LEN	R/W	0*	When the bits R_BUF[R_BUF_LEN:0] are received the next int_rbuf_full is generated. [1]

- [1] Only the lower 6 bits are valid

Table 517: bmp_r_buf0 - Receive data buffer 0

Bit	Symbol	Access	Value	Description
31:0	R_BUF0	R	0*	Receive buffer 0. The MSB was received first

Table 518: bmp_r_buf1 - Receive data buffer 1

Bit	Symbol	Access	Value	Description
31:0	R_BUF1	R	0*	Receive buffer 1. The MSB was received first

Table 519: bmp_r_pointer - RCU command array pointer

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	R_POINTER	R/W	0*	Current value of R_point in RCU

- [1] A write to this register has effect latest at the transition to the next bit.

Table 520:bmp_rf_pointer - RFCU command array pointer

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	RF_POINTER	R/W	0*	Current value of RF_point in RFCU

[1] A write to this register has effect latest at the transition to the next bit.

Table 521:bmp_rfpu_con - RFPUs control register

Bit	Symbol	Access	Value	Description
31:17	-	R	0*	reserved
16	REF_CLK_SEL	R/W		Selects how REF_CLK is controlled
			1	during en_ref_clk ='1' REF_CLK is enabled with scl_ctrl
			0*	REF_CLK is enabled with en_ref_clk
15	FAD_SWAP	R/W		Selection of antenna control signals at RF_SIG[0] (ant_sel0) and RF_SIG[1] (ant_sel1)
			1	swapped
			0*	default
14	IF_SLC_CTRL_DIS	R/W	0*	Disables (set to '0') if_scl_ctrl sent to APU
13:12	SLC_CTRL_SEL	R/W		Selects which slice control signal is switched to RF_SIG[RF_SIG_NUM].
			11	scl_ctrl_eop from R_eop
			10	rcu_sig from RCU
			01	scl_ctrl from RPU
			00*	as programmed by RFCU (rf_sig_rfcu[x])
11:9	RF_SIG_NUM	R/W		Determines to which RF_SIG the scl_ctrl signal is connected
			111	scl_ctrl is connected to RF_SIG[7]
			110	scl_ctrl is connected to RF_SIG[6]
			101	scl_ctrl is connected to RF_SIG[5]
			100	scl_ctrl is connected to RF_SIG[4]
			011	scl_ctrl is connected to RF_SIG[3]
			010	scl_ctrl is connected to RF_SIG[2]
			001	reserved
8	FAD_INV	R/W		Polarity of antenna control signals RF_SIG[0] (ant_sel0) and RF_SIG[1] (ant_sel1)
			1	inverted
			0*	as programmed by MCU
7:0	RF_SIG_CON	R/W	0*	RF_SIG[7:0] state when BMP_ON in bmp_global is set to 0

Table 522: bmp_rpu_con0 - Receive Peripheral Unit control register 0

Bit	Symbol	Access	Value	Description
31:30	THR_SYNC_H	R/W		Determines the additional number (to the value set in THR_SYNC_L) of errors accepted to generate sync_ok
			11	12 more errors accepted
			10	8 more errors accepted
			01	4 more errors accepted
			00*	no more errors accepted
29	RDATA_SEL	R/W		Select between traditional and enhanced RF interface
			1	R_DATA value is input from digital pin BMP_DATA muxed on RF_SIG[2] (enhanced RF interface)
			0*	R_DATA value is input from analog pin R_DATA (traditional RF interface)
28	SEL_R_FILTER	R/W		R_FILTER method selection
			1	majority voting filter method selected (when bit SEL_R_FILTER2 in bmp_rpu_con1[27] = 0)
			0*	single clock spike filter method selected
27	RDATA_INV	R/W		IF_R_DATA signal polarity
			1	IF_R_DATA is inverted
			0*	IF_R_DATA is not inverted
26:23	ID_LEN	R/W	0*	Length of ID pattern [ID_LEN:0]
22:13	ID_COUNT	R/W	0*	Maximum value of counter Id_count
12:11	THR_ID_L	R/W		Determines the numbers of errors accepted to generate id_ok
			11	3 errors accepted
			10	2 errors accepted
			01	1 error accepted
			00*	no errors accepted
10	COND_SEL	R/W		Selects condition to start RCU
			1	sync_ok AND id_ok must be found
			0*	sync_ok must be found
9:8	THR_SYNC_L	R/W		Determines the number of errors accepted to generate sync_ok
			11	3 errors accepted
			10	2 errors accepted
			01	1 error accepted
			00*	no errors accepted
7:2	SYNC_LEN	R/W	0*	Syncword has length [SYNC_LEN:0]
1	FREEZE_CONST	R/W		Selects time constant for freeze operation (MCU event 29)
			1	time constant = 8(when bit FREEZE_CONST2 in bmp_rpu_con1[26]=0)
			0*	time constant = infinite
0	R_DATA_SOURCE	R/W		selects which data source to use
			1	source demodulator
			0*	source R_DATA / BMP_DATA (RF_SIG[2])

Table 523: bmp_rpu_con1 - Receive Peripheral Unit control register 1

Bit	Symbol	Access	Value	Description
31:30	-	R	0*	reserved
29	SEL_R_RATE1	R/W	0*	R_DATA rate selection ^[1]
28	SEL_DUTY_DET	R/W		selects duty cycle detection method in r_pll
			1	duty cycle detection enabled
			0*	duty cycle detection disabled
27	SEL_R_FILTER2	R/W		R_FILTER method selection when bit SEL_R_FILTER in bmp_rpu_con0[28] = 1
			1	triangular filter method selected
			0*	majority voting filter method selected
26	FREEZE_CONST2	R/W		Selects time constant for freeze operation (MCU event 29) when bit FREEZE_CONST in bmp_rpu_con0[1] = 1
			1	time constant = 16
			0*	time constant = 8
25	SEL_R_RATE0	R/W	0*	R_DATA rate selection ^[1]
24:23	THR_ID_H	R/W		Determines the additional number (to the value set in THR_ID_L) of errors accepted to generate id_ok
			11	12 more errors accepted
			10	8 more errors accepted
			01	4 more errors accepted
			00*	no more errors accepted
22	EOP_RPLL_SYNC	R/W	0*	If set to '1' the signal slice_ctrl_eop is latched by the edge of the recovered clock clk_r. When enabled, EOP_S_SEL(0) (bmp_rpu_pat[22]) has to be '0'
21:15	EOP_REENB_DLY	R/W	0*	delay counter value to reenable the end of preamble detection a delay of zero disables the reenable counter
14:11	EOP_START_DLY	R/W	0*	start delay counter value for end of preamble detection
10	CP_R_BUF	R/W	0*	forces a copy from R_BUF into R_BUF1/R_BUF0, Automatic reset.
9	R_DIR	R/W		selects shift direction
			1	first received is the LSB in R_BUF and R_sync
			0*	last received is the LSB in R_BUF and R_sync
8:2	PRE_R_SCR	R/W	0*	Receive scrambler preset value [6:0]
1	SEL_R_SCR	R/W		selects scrambler polynomial to be used
			1	7-bit scrambler
			0*	5-bit scrambler with inversion mechanism
0	RES_R_XCRC	R/W	0*	Resets the receive XCRC block. Automatic reset.

[1] [SEL_R_RATE1:SEL_R_RATE0]
 0b00: standard, default
 0b01: half rate
 0b10: EDR double rate, enabled by command en_r_edr in bmp_rcu_com register
 0b11 : EDR triple rate, enabled by command en_r_edr in bmp_rcu_com register

Table 524:bmp_rpu_pat - Receive Peripheral Unit pattern register

Bit	Symbol	Access	Value	Description
31:27	-	R	0*	reserved
26:22	EOP_S_SEL	R/W	0b10101*	used samples per bit selection for end of preamble detection EOP_S_SEL[4] represents the latest sample in the line and is fixed to '1'
21:16	EOP_PAT	R/W	0*	Bit pattern used for end of preamble detection EOP_PAT[0] represents the latest received bit
15:0	ID_PAT	R/W	0*	Bit pattern used for ID detection

Table 525:bmp_rpu_stat0 - Receive Peripheral Unit status register

Bit	Symbol	Access	Value	Description
31:11	-	R	0*	reserved
10	EOP_STAT	R/W	0*	End of preamble found This bit is set to 1 whenever an end of preamble word was found. It must be reset by the system controller [2]
9	ID_STAT	R/W	0*	Identity found This bit is set to 1 whenever an ID word was found. It must be reset by the system controller [2]
8	SYNC_STAT	R/W	0*	Synchronization word found This bit is set to 1 whenever a synchronization word was found and en_sync_det was active. It must be reset by the ARM System Controller [2]
7:0	PREAM_DUMP [3]	R	0*	Dump of last 8 bits before synchronization word

[1] A write to this register has effect latest at the transition to the next bit.

[2] Any write to this register causes a reset of SYNC_STAT, ID_STAT and EOP_STAT, i.e. the pattern does not matter. Note that after closing the sync window, a second interrupt will occur.

[3] Not usable at DVFD8185 with DRF1902

Table 526:bmp_rpu_stat1 - Receive Peripheral Unit status register 1

Bit	Symbol	Access	Value	Description
31:16	CRC_DUMP	R	0*	Dump of 16 bit CRC shift register [1]
7:4	-	R	0*	reserved
3:0	XCRC_DUMP	R	0*	Dump of 4 bit XCRC shift register

[1] Depending on the setting of CRC_LENx in bmp_crc_conx either all 16 bits or only the lower 8 bits of CRC_DUMP are valid.

Table 527:bmp_rpu_stat2 - 32bit CRC Receive Peripheral Unit status register

Bit	Symbol	Access	Value	Description
31:0	CRC_DUMP2[31:0]	R	0*	Dump of 32 bit CRC calculation result [1]

[1] Depending on the setting of CRC_LEN2[1:0] in bmp_crc_con4 either all 32 bits or only the lower 16 or 8 bits of CRC_DUMP2 are valid.

Table 528: bmp_rpu_stat3 - Receive Peripheral Unit status register 3 (for correlator)

Bit	Symbol	Access	Value	Description
31:23	reserved	R	0*	
21 ^[1]	SYNC_MTCH	R	0*	
20:17	RXBITCLK_ST	R	1*	Value can vary, but is not relevant for operation.
16:1	RXSYNC_WRD	R	0*	
0	RXSYNC_LTCH	R	0*	

[1] Status is also available at **bmp_rpu_stat0.sync_stat**

Table 529: bmp_scnt_mod - Modulo values of slot counters Scnt0 and Scnt1

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:8	SCNT_MOD1	R/W	0*	Modulo value of Scnt1 ^[1]
7:0	SCNT_MOD0	R/W	0*	Modulo value of Scnt0 ^[1]

[1] Only the lower 5 bits are valid.

Table 530: bmp_scnt_val - Current values of slot counters Scnt0 and Scnt1

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:8	SCNT_VAL1	R/W	0*	Current value of slot counter Scnt1 ^[2]
7:0	SCNT_VAL0	R/W	0*	Current value of slot counter Scnt0 ^[2]

[1] This register may not be written within the vicinity (+/- 2 bit) of frame boundaries. A write to this register has effect latest at the transition to the next bit.

[2] Only the lower 5 bits are valid.

Table 531: bmp_sif_buf0 - Serial Interface buffer

Bit	Symbol	Access	Value	Description
31:0	SIF_BUF0	R/W	0*	Serial interface buffer 0

[1] Register shares the same address with register **bmp_spif_adr**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 532: bmp_sif_buf1 - Serial Interface buffer

Bit	Symbol	Access	Value	Description
31:0	SIF_BUF1	R/W	0*	Serial interface buffer 1

[1] Register shares the same address with register **bmp_spif_dataout**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 533: bmp_sif_buf2 - Serial Interface buffer

Bit	Symbol	Access	Value	Description
31:0	SIF_BUF2	R/W	0*	Serial interface buffer 2

[1] Register shares the same address with register **bmp_spif_datain**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 534:bmp_sif_con - Serial Interface Control

Bit	Symbol	Access	Value	Description
31:15	-	R	0*	reserved
14	SIF_MUX_EN	R/W		Define signal to be routed at SIF_SPIF_DATA pin
			1	rf_sig[2] signal is multiplexed to the SIF_SPIF_DATA pin
			0*	sif_data_out signal is multiplexed to the SIF_SPIF_DATA pin
13	SIF_CLK_DEL	R/W		Inserts delay to sif_clk compared to sif_data.
			1	sif_clk edges are delayed by 1 clk_bmp cycle compared to sif_data edges.
			0*	No delay is inserted.
12	SIF_OFF_MODE	R/W		Selects behavior of SIF_SPIF_DATA pin in function of SIF_OFF_POL.
			1	Any modification of the SIF_OFF_POL bit will take place after the next transfer (the new state for SIFF_OFF_POL is latched with the rising edge of start_sif).
			0*	Any change of bit SIF_OFF_POL is immediately visible on pin SIF_SPIF_DATA.
11	SIF_DIR	R/W		Determines which bit of the SIF_BUFS is shifted out first
			1	Bit 0 in bmp_sif_buf0 is shifted out first, i.e. right shift is executed
			0*	Bit SIF_LENGTH - 1 is shifted out first, i.e. a left shift is executed
10	SIF_OFF_POL	R/W		Determines polarity of SIF_SPIF_DATA when no data are shifted out
			1	SIF_SPIF_DATA goes to 1
			0*	SIF_SPIF_DATA goes to 0
9	SIF_CLK_POL	R/W		Determines polarity of SIF_SPIF_DATA
			1	SIF_SPIF_DATA change on negative edge of SIF_SPIF_CLK. (positive edge of SIF_SPIF_CLK can be used to latch the signal SIF_SPIF_DATA)
			0*	SIF_SPIF_DATA change on positive edge of SIF_SPIF_CLK. (negative edge of SIF_SPIF_CLK can be used to latch the signal SIF_SPIF_DATA)
8:7	SIF_SPEED	R/W		Determines clock speed of serial interface
			11	sif_clk/2 MHz
			10	sif_clk/4 MHz
			01	sif_clk/6 MHz
			00*	sif_clk/12 MHz
6:0	SIF_LENGTH	R/W	0*	Length of packet transmitted after start_sif rising edge. This value must be smaller or equal 96. If 0 is programmed no output signal is generated.

[1] Register shares the same address with register **bmp_spif_con**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 535: bmp_spif_adr - Serial peripheral interface address register

Bit	Symbol	Access	Value	Description
31:13	reserved	R/W	0*	Should be set to '0'
12	SPIF_STC	R/W		Start transfer control
			1	A read access is started by the BMP signal sif_start. A write access is started by the BMP signal sif_start AND the register bmp_spif_datout has been written before.
			0*	A read access is started immediately after the write to this register. A write access is started immediately after writing to the register bmp_spif_datout
11	reserved	R/W	0*	Should be set to '0'
10	SPIF_INC	W		Auto-increment address flag. Used in indirect write state only.
			1	auto-increment is forced (more than one data word)
			0*	no auto-increment
9	SPIF_IND	W		Direct / indirect access flag
			1	indirect access
			0*	direct access
8	SPIF_RW	W		Read / write access flag
			1	Read access
			0*	Write access
7:0	SPIF_ADR	W	0*	Address where the data should be written to or read from.

[1] Register shares the same address with register **bmp_sif_buf0**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 536: bmp_spif_con - Serial peripheral interface control register

Bit	Symbol	Access	Value	Description
31:11	reserved	R/W	0*	Should be set to '0'
10	SPIF_OFF_POL	R/W		Determines polarity of SIF_SPIF_DATA pin when no data are shifted out (SPIF inactive)
			1	SIF_SPIF_DATA pin goes to 1
			0*	SIF_SPIF_DATA pin goes to 0
9:0	reserved	R/W	0*	Should be set to '0'

[1] Register shares the same address with register **bmp_sif_con**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 537: bmp_spif_datout - Serial peripheral interface data out register

Bit	Symbol	Access	Value	Description
31:8	reserved	R/W	0*	Should be set to '0'
7:0	SPIF_DATOUT	R/W	0x00*	Serial peripheral interface data out buffer

[1] Register shares the same address with register **bmp_sif_buf1**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 538: bmp_spif_datin - Serial peripheral interface data in register

Bit	Symbol	Access	Value	Description
31:16	reserved	R	0*	
15	SPIF_BUSY	R		SPIF busy flag. It shows if any transaction is ongoing.
			1	SPIF is busy or a task is pending
			0*	SPIF is in idle
14	SPIF_ERR	R		SPIF error flag. It shows if an error occurs during any transaction.
			1	An error occurred. If there was one more access to the register while a task was pending or running.
			0*	No error occurred
13:8	reserved	R	0*	
7:0	SPIF_DATIN	R	0*	Serial peripheral interface data in buffer

[1] Register shares the same address with register **bmp_sif_buf2**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 539: bmp_start_time -Receive Control Unit start time register

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13:0	START_TIME	R	0*	Bit counter value when RCU was started last

[1] In case **bmp_global.dpll_symrec** is set, the START_TIME has an offset of +1 to the RCU start position.

Table 540: bmp_symrec_con0 - Symbolrecovery control register 0 (for correlator)

Bit	Symbol	Access	Value	Description
31	PHASE_LOAD [2]	R/W		Phase load enable
			1	Phase value in field bmp_symrec_con0.phase_abs is loaded
			0*	
30:27	PHASE_ABS [2]	R/W	0b0000*	Phase value
26	reserved	R/W	0*	Should be set to '0'
25	FREQ_10M_13M [2]	R/W		System frequency selection
			1	System frequency is 10.368MHz
			0*	System frequency is 13.824MHz
24:20	STHD	R/W	0*	Symbol threshold. Determines the number of bits matching with SYNCWRD to generate sync_mtch (sync found) at the received sync word. The Recommended values are: 0xD: Implies that SyncMatch will be declared for 13 or more correct bits (Locked Mode) 0xF: Implies that SyncMatch will be declared for 15 or more correct bits (Acquire Mode)

Table 540:bmp_symrec_con0 - Symbolrecovery control register 0 (for correlator)...continued

Bit	Symbol	Access	Value	Description
19:18	CORR_PH	R/W		Correlator (slice control) phase offset selection in relation to the middle of the last bit of the sync word in the oversampled clock periods
			0b11	-1 oversampled clock periods phase shift
			0b10	-2 oversampled clock periods phase shift
			0b01	+1 oversampled clock periods phase shift
			0b00*	no phase shift
17:16	reserved	R/W	0*	Should be set to '0'
15:0	SYNCWRD	R/W	0*	Synchronization word. Upon down-link, the SYNC will be transmitted in the manner it was defined in this register, whereas upon up-link it will be inverted (logical NOT). See also bmp_symrec_con1.s_inv_dis .

[1] Register shares the same address with register **bmp_sync_pat0**. The selection is made with bit **bmp_global.dpll_symrec**.

[2] Function should not be used. Reset values should be kept.

Table 541:bmp_symrec_con1 - Symbolrecovery control register 1 (for correlator)

Bit	Symbol	Access	Value	Description
31:15	reserved	R/W	0*	Should be set to '0'
14	DECT_MHS	R/W		Correlator operation mode selection
			1	Correlator is operating in DECT mode
			0*	Correlator is operating in EDCT mode
13:10	reserved	R/W	0*	Should be set to '0'
9	RX_DATA_INV	R/W		Receive data inversion enable
			1	Receive data is inverted
			0*	Normal
8	S_INV_DIS	R/W		Sync invers disable for receive path
			1	SYNC is received normally
			0*	SYNC is received inverted
7:0	CTHD	R/W	0*	Chips threshold (Chips are defined as the sub-symbol samples of the incoming data). The ange is 16*12=192chips. The recommended values are: 0x9C (156 samples match) for FP or PP in TDD (locked to FP) 0xAB (171 samples match) for PP in acquire

[1] Register shares the same address with register **bmp_sync_pat1**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 542:bmp_sync_pat0 - Synchronization pattern 0

Bit	Symbol	Access	Value	Description
31:0	SYNC_PAT0	R/W	0*	Synchronization pattern bits [31:0]

[1] Register shares the same address with register **bmp_symrec_con0**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 543: bmp_sync_pat1 - Synchronization pattern 1

Bit	Symbol	Access	Value	Description
31:0	SYNC_PAT1	R/W	0*	Synchronization pattern bits [63:32]

[1] Register shares the same address with register **bmp_symrec_con1**. The selection is made with bit **bmp_global.dpll_symrec**.

Table 544: bmp_sypo_con - Synchronization port control register

Bit	Symbol	Access	Value	Description
31:1	-	R	0*	reserved
0	SYPO_MASTER_SEL	R/W		Selects synchronization port master mode
			1	master mode
			0*	slave mode

Table 545: bmp_sypo_ref_cnt - Synchronization port reference value

Bit	Symbol	Access	Value	Description
31:20	-	R	0*	reserved
19:18	PULSE_LEN	R		Pulse length of wide pulse at SYPO pin ^[1]
			11	reserved
			10	4.5 ms
			01	3.4 ms
			00*	2.3 ms
17:0	SYPO_REF_CNT	R	0*	Synchronization port reference value

[1] These bits are only valid if SYPO_MASTER_SEL is set to 0, ie. SYPO is in slave mode.

Table 546: bmp_t_buf_len - Length of T_BUF

Bit	Symbol	Access	Value	Description
31:8	-	R	0*	reserved
7:0	T_BUF_LEN	R/W	0*	The bits T_BUF[T_BUF_LEN:0] are transmitted

[1] Only the lower 6 bits are valid

Table 547: bmp_t_buf0 - Transmit data buffer 0

Bit	Symbol	Access	Value	Description
31:0	T_BUFO	R/W	0*	Transmit buffer 0

Table 548: bmp_t_buf1 - Transmit data buffer 1

Bit	Symbol	Access	Value	Description
31:0	T_BUF1	R/W	0*	Transmit buffer 1

Table 549: bmp_t_pointer - TCU command array pointer

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	T_POINTER	R/W	0*	Current value of T_point in TCU

[1] A write to this register has effect latest at the transition to the next bit.

Table 550: bmp_tpu_con - Transmit Peripheral Unit control register

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15	TRDATA_MUX_SEL	R/W		Source selection to control the pad direction of the BMP_DATA pin if selected in GPM (see Table 13)
			1	The pin BMP_DATA becomes an output if signal rf_sig2=1
			0*	The pin BMP_DATA becomes an output if jmp_global.dpll_symrec=1 and tcu is running.
14	SEL_T_RATE1	R/W	0*	T_DATA rate selection [1]
13	SEL_T_RATE0	R/W	0*	T_DATA rate selection [1]
12	TDATA_INACTIVE_V ALUE	R/W		T_DATA value when TCU inactive
			1	When inactive, T_DATA is fed with constant '1'
			0*	When inactive, T_DATA is fed with constant '0'
11	TDATA_INV	R/W		IF_T_DATA signal polarity
			1	IF_T_DATA is inverted
			0*	IF_T_DATA is not inverted
10	RES_T_XCRC	R/W	0*	Reset XCRC generator T_xcrc. Automatic reset by hardware.
9	SEL_T_SCR	R/W		selects scrambler polynomial to be used
			1	7-bit scrambler
			0*	5-bit scrambler with inversion mechanism
8:2	PRE_T_SCR	R/W	0*	Transmit scrambler preset value [6:0]
1	T_DIR	R/W		selects shift direction
			1	first sent out is the LSB in T_BUF
			0*	last sent out is the LSB in T_BUF
0	CP_T_BUF	R/W	0*	If set T_BUF1/T_BUF0 is copied to T_BUF. Automatic reset.

[1] [SEL_T_RATE1:SEL_T_RATE0]
 0b00: standard, default
 0b01: half rate
 0b10: EDR double rate, enabled by command en_t_edr in **jmp_tcu_com** register
 0b11: EDR triple rate, enabled by command en_t_edr in **jmp_tcu_com** register

Table 551: bmp_com_add_rdata - address configuration for RDATA cache

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
24:16	RDATA_COM_END	R/W	0*	BMP RAM RDATA cache END addresses
15:9	-		0*	reserved
8:0	RDATA_COM_STA	R/W	0*	BMP RAM RDATA cache START addresses

Table 552: bmp_com_add_tdata - address configuration for TDATA cache

Bit	Symbol	Access	Value	Description
31:25	-	R	0*	reserved
24:16	TDATA_COM_END	R/W	0*	BMP RAM TDATA cache END addresses
15:9	-	R	0*	reserved
8:0	TDATA_COM_STA	R/W	0*	BMP RAM TDATA cache START addresses

Table 553: bmp_rdata_pointer - RDATA cache pointer offset

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	RDATA_PTR	R/W	0*	offset from RDATA_COM_STA

Table 554: bmp_tdata_pointer - TDATA cache pointer offset

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8:0	TDATA_PTR	R/W	0*	offset from TDATA_COM_STA

Table 555: bmp_m_com_XX - Main Control Unit command word

Bit	Symbol	Access	Value	Description
31	int_mcu_fiq0	R/W	0*	fast interrupt request 1 from MCU [1]
30	int_mcu_irq0	R/W	0*	interrupt request 1 from MCU [1]
29	int_mcu_fiq1	R/W	0*	fast interrupt request 2 from MCU [1]
28	int_mcu_irq1	R/W	0*	interrupt request 2 from MCU [1]
27	go_rcu	R/W	0*	starts RCU under MCU control
26	sync_ref	R/W	0*	phase correction reference signal to Phase_corr
25	sypo_ref	R/W	0*	reference signal for SYPO
24	en_sync_det	R/W	0*	enables synchronization word detection
	corr_sync_time			Used at correlator input stage. Enables the time window for correlator “peak search” prior to expected sync match event. The recommendation is to enable it 1 bit clock after bmp_rfcu_com.sym_sync_time and end together with bmp_rfcu_com.sym_sync_time .
23	en_p_meas	R/W	0*	enables a phase error measurement [2][3]
			1	upon the next valid condition a phase measurement is performed.
			0*	no phase error measurement is performed
22	reserved	R/W	0*	
21	start_ini_enc_dt	R/W	0*	start initialisation and preclocking of encryption / decryption engine for DECT
20	start_tcu	R/W	0*	start TCU
19	start_rfcu	R/W	0*	start RFCU
18	start_ad	R/W	0*	start AD conversion cycle on APU->AIO
17:14	reserved	R/W	0*	reserved
13: 0	m_time[13:0]	R/W	0*	time for MCU event to become active

[1] Any ‘1’ written here will cause an interrupt. There is no need to reset the interrupt line to 0.

[2] Note that if a phase correction is currently being performed another phase error measurement can not be done simultaneously.

[3] When using the go_rcu instead of the automatic sync_det the bcnt value is not copied to the register START_TIME.

Table 556:bmp_rfcu_com_xx - RF Control Unit command word

Bit	Symbol	Access	Value	Description	
31	int_rfcu	R/W	0*	interrupt from RFCU [1]	1
30	freeze	R/W		freezes clock recovery R_pll	2
			1	R_pll time constant set to 8 / 16 / infinite	3
			0*	R_pll time constant set to 4	4
29	sel_t_car	R/W		select for data or toggle mode of t_data	5
			1	t_data carries a 0101 pattern at clk_o to force the signal on the	6
				transmit FIR filter output to the average value, i.e. transmit carrier	7
			0*	frequency	8
28	stop_rfcu	R/W	0*	t_data carries the transmit data	9
27	en_r_pll	R/W	0*	stops RFCU	10
26	slc_rfcu	R/W	0*	enables receive data stream	11
25	start_sif	R/W	0*	control of bit slice filter (->RPU->APU)	12
24	en_abs	R/W	0*	starts SIF serial interface	13
23	en_mod	R/W	0*	enable bit slicer and RSSI (->APU)	14
22	en_ref_clk	R/W	0*	enable modulator (GFSK/IQ) (->APU)	15
21	en_peak_det	R/W	0*	enables reference clock output (->APU)	16
20	start_fad	R/W	0*	enables RSSI measurement and determines peak-detector	17
	sym_sync_time			window (->APU)	18
				start fast antenna diversity (->APU)	19
				Used at correlator input stage. Enables the time window defining	20
				the correlation stage where sync is expected to occur. Should be	21
				set just prior to reception of sync word.	22
19	rf_sig_rfcu[7]	R/W	0*		23
18	rf_sig_rfcu[6]	R/W	0*	RF timing control line 7	24
17	rf_sig_rfcu[5]	R/W	0*	RF timing control line 6	25
16	rf_sig_rfcu[4]	R/W	0*	RF timing control line 5	26
15	rf_sig_rfcu[3]	R/W	0*	RF timing control line 4	27
14	rf_sig_rfcu[2]	R/W	0*	RF timing control line 3	28
13	rf_sig_rfcu[1] -> ant_sel1	R/W	0*	RF timing control line 2	29
12	rf_sig_rfcu[0] -> ant_sel0	R/W	0*	RF timing control line 1, controls antenna 1, (->APU)	30
11: 0	rfcu_time[11: 0]	R/W	0*	RF timing control line 0, controls antenna 0, (->APU)	31
				time when control event becomes valid	32

[1] Any '1' written here will cause an interrupt. There is no need to reset the interrupt line to 0. This interrupt can be used as both FIQ and IRQ by enabling the appropriate interrupt source in the INTC

Table 557:bmp_tcu_com_xx - Transmit Control Unit command word

Bit	Symbol	Access	Value	Description	
31	int_tcu_FIQ	R/W	0*	fast interrupt request from TCU [1]	1
30	int_tcu_IRQ	R/W	0*	interrupt request from TCU [1]	2
29	dis_p_corr	R/W	0*	disable phase correction circuitry	3
28	stop_tcu	R/W	0*	stop TCU	4

Table 557:bmp_tcu_com_xx - Transmit Control Unit command word...continued

Bit	Symbol	Access	Value	Description
27	en_t_xcrc	R/W		enable XCRC calculation
			1	XCRC calculation enabled
			0*	XCRC calculation disabled
26	shift_t_xcrc	R/W		enable shift out process of XCRC
			1	XCRC is being shifted out. If this signal is set high the X-field is repeated until signal is low again. ^[2]
			0*	Nothing shifted. XCRC block retains its content.
25:24	reserved	R/W	0*	
23	en_t_scr	R/W	0*	enables scrambler
22	reserved	R/W	0*	
21	sel_t_crc	R/W		select CRC polynomial
			1	CRC_CON1 selected
			0*	CRC_CON0 selected
20	en_t_crc	R/W	0*	enables CRC generator
19	dis_t_enc_dt	R/W		disable DECT encryption
			1	Encryption disabled. Key stream still being generated.
			0*	Encryption enabled (if previously started and initialized by start_ini_enc and start_t_enc)
18	start_t_enc_dt	R/W	0*	start the encryption process. Reference signal at the beginning of the A-field (for the DECT case).
17	en_t_buf	R/W	0*	enables transmit buffer
16	reserved	R/W	0*	
15	sel_t_crc2	R/W		select CRC-32 polynomial
			1	CRC_CON3/4 selected
			0*	CRC_CON0/1 selected
14:12	-	R	0*	reserved
11: 0	tcu_time[11: 0]	R/W	0*	time when control event becomes valid

[1] Any '1' written here will cause an interrupt. There is no need to reset the interrupt line to 0.

[2] This function is to be used to add the Z-field, i.e. for normal DECT operation this signal must be high for 8 bits.

Table 558:bmp_rcu_com_xx - Receive Control Unit command word

Bit	Symbol	Access	Value	Description
31	int_rcu_FIQ	R/W	0*	fast interrupt request from RCU ^[1]
			0*	interrupt request from RCU ^[1]
			0*	RCU timing control line
28	stop_rcu	R/W	0*	stops RCU
				enable XCRC calculation
			1	XCRC calculation enabled
26:25	reserved	R/W	0*	
			0*	XCRC calculation disabled
			0*	enables descrambler

Table 558: bmp_rcu_com_xx - Receive Control Unit command word...continued

Bit	Symbol	Access	Value	Description
23	reserved	R	0*	
22	sel_r_crc	R/W		select CRC polynomial
			1	CRC_CON1 selected
			0*	CRC_CON0 selected
21	en_r_crc	R/W	0*	enables CRC check
20	start_r_dec_dt	R/W	0*	Starts the decryption process. Reference signal at the beginning of the A-field (for the DECT case).
19	dis_r_dec_dt	R/W		disable DECT decryption
			1	Decryption disabled. Key stream still being generated.
			0*	Decryption enabled (if previously started and initialized by start_ini_enc and start_r_dec)
18	en_r_buf	R/W	0*	enables receive buffer
17	sel_r_crc2	R/W		select CRC-32 polynomial
			1	CRC_CON3/4 selected
			0*	CRC_CON1/2 selected
16:12	reserved	R/W	0*	reserved
11:0	rcu_time[11:0]	R/W	0*	time when control event becomes valid

[1] Any '1' written here will cause an interrupt. There is no need to reset the interrupt line to 0.

10.5 Application information

10.5.1 Auto hardware fetch feature

If the auto hardware data fetch feature is enabled (see register **bmp_global.auto_hw_data_fetch =1**) the first command in a region of the bmp ram assigned to one of the control engines (TDCU or RDCU) needs to be pre-loaded to the engine when the ARM writes the command to the bmp ram. The preload procedure is that SW should execute a write, read, write sequence for the first command!

The RPU and RCU operate only if **clk_r** is running which is gated by the RFCU control signal **en_r_pll**. In case of **bmp_global.dpll_symrec =1** the **en_r_pll** is not required to be set.

10.5.2 Delay on signal SYNC_MTCH

Due to internal resynchronisation of the correlator signals an additional delay of 1 bit clock is applied when sync match is true at correlator stage and the event seen at pin **SYNC_MTCH**.

10.5.3 Synchronisation port operation modes limitation

The synchronisation port of the DVFD8185 can only be used in master mode due to pin GPIOA5 is output always if SYNCPORT is selected in register **sysmux0.gpioa5c**.

11. ETN1 -10/100 Ethernet MAC

11.1 Features

Some of the features of the 10/100 Ethernet MAC are:

- MAC sublayer of IEEE std 802.3
- Operate in 10 Mbps and in 100 Mbps mode
- Half-duplex and Full-duplex mode
- Attachment of external PHY chip through Reduced MII (RMII) interface
- DMA and FIFO managers with scatter/gather DMA and FIFOs of frame descriptors
- Memory traffic optimized by buffering and pre fetching
- Receive filtering includes perfect address matching, a hash table imperfect filter
- MAC address hash values of receiver packet
- Wake-on-LAN power management support allows system wake-up: using the receive filters or a magic frame detection filter
- Optional support for Quality of Service (QoS) using two transmit queues: a low priority and a high-priority transmit queue
- VLAN support
- Separate MLAHB interfaces for status/control and DMA traffic
- IEEE 802.3/clause 31 flow control, both receive and transmit

11.2 Block diagram

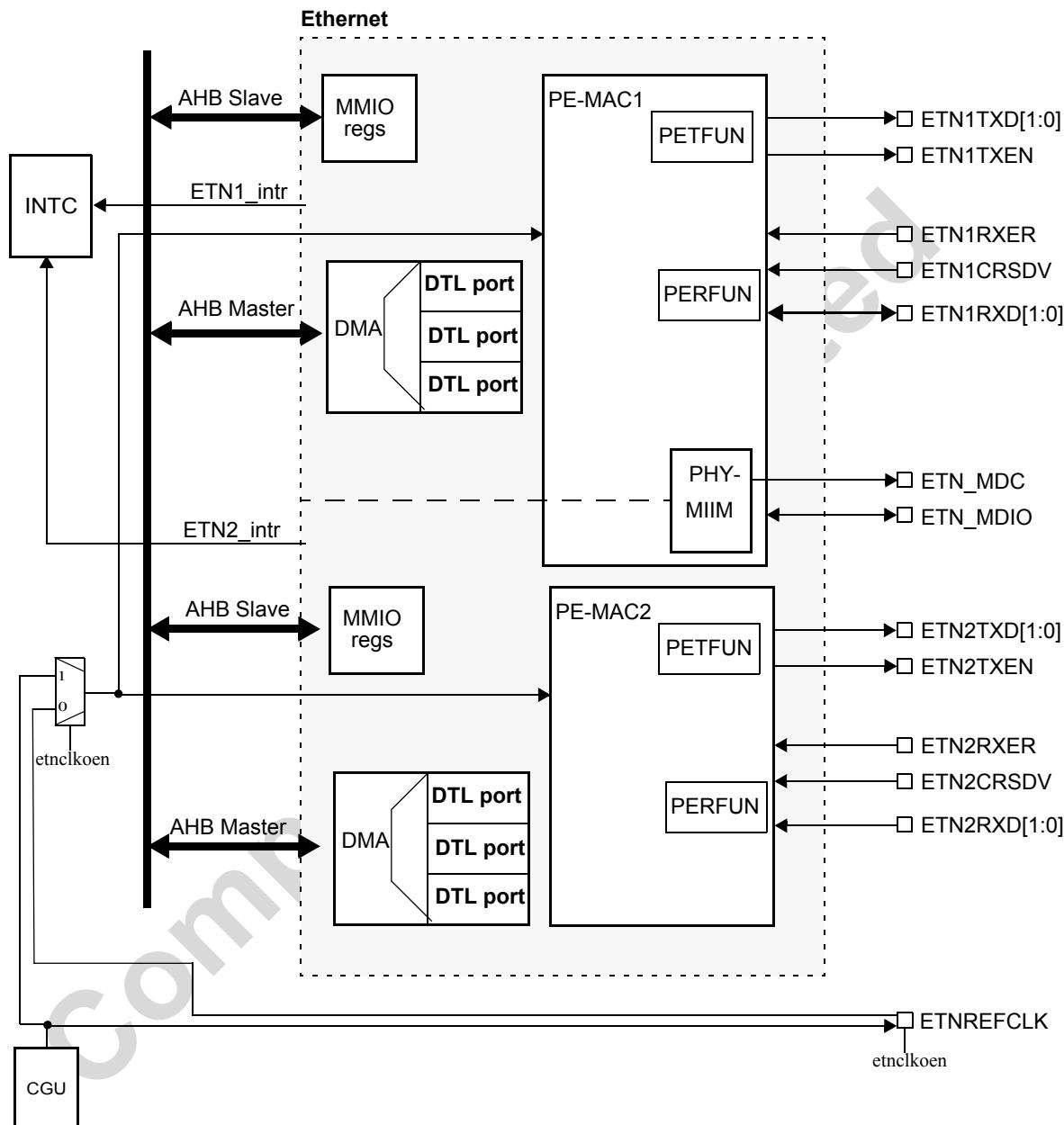


Fig 193.ETN1/2 block diagram

Note: At DVFD818x only ETN1 is available

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

11.3 Hardware Interface

Table 559: ETM signal overview

PIN	Name	I/O	Description
ETN general signal			
ETNREFCLK	reference clock	O or I ^[1]	Synchronous clock reference for receive, transmit and control interface for MAC and external PHY
ETNMDC ^[3]	reference clock	O	PHY management clock
ETNMDIO ^[3]	reference clock	I/O/Z ^[2]	PHY data input/output
ETN1 MAC controller			
ETN1TXD[1:0]	transmit data	O	transmit data to external PHY
ETN1TXEN	transmit enable	O	ETN1TXEN indicates that the MAC is presenting bits on ETN1TXD[1:0] for transmission
ETN1RXER	receive error	I	ETN1RXER is asserted to indicate that an error (e.g. a coding error or other error that a PHY is capable of detecting) was detected somewhere in the frame presently being transferred from the PHY
ETN1CRSDV	carrier sense / data valid	I	ETN1CRSDV will be asserted by the PHY when the receive medium is nonidle. The data on ETN1RXD[1:0] is considered valid once ETN1CRSDV is asserted.
ETN1RXD[1:0]	receive data	I	Receive Data from external PHY

[1] ETNREFCLK should be feed in from an external clock source (see Figure 14)

[2] I/O state must be tristate if it is not used (see Section 11.6.1)

[3] Only ETN1-PHY-MIIM is connected to the signals.

Table 560: AC characteristics of the RMII

Name	Parameter	Min	Typ	Max	Unit
ETNREFCLK	Reference clock frequency	-	50	-	MHz
	Reference clock duty cycle	35	-	65	%
T _{setup}	ETNTXD[1:0], ETNTXEN, ETNRXD[1:0], ETNCRSDV, ETNRXER data setup to rising edge of reference clock	4	-	-	ns
T _{hold}	ETNTXD[1:0], ETNTXEN, ETNRXD[1:0], ETNCRSDV, ETNRXER data hold from rising edge of reference clock	2	-	-	ns

[1] Output drivers shall be capable of meeting the output requirements while driving a 25pF or greater load. This loading accommodates over 12 inches of PCB trace and input capacitance of the receiving device.

Table 561: AC characteristics of the MDC/MDIO control interface

Name	Parameter	Min	Typ	Max	Unit
ETNMDC	Ethernet PHY management clock minimum cycle time	400	-	-	ns
T _{setup}	ETNMDIO data setup to MDC rising edge	10	-	-	ns
T _{hold}	ETNMDIO data hold from MDC rising edge	10	-	-	ns

11.4 Software Interface

11.4.1 Register map

Table 562 lists the registers and the address of the register. When reading or writing the value of a register via the MMIO interface the host registers module in the ETN1 takes care of the clock domain crossing between the register's clock domain and the MMIO clock domain.

After a hard reset or a soft reset via the RegReset bit of the **etn1_command** register all bits in all registers are reset to 1.b0 unless stated otherwise in Section 11.4.2.

Some registers will have unused bits which will return a 1.b0 on a read via the MMIO interface. Writing to unused register bits of an otherwise writable register will not have side effects.

The register map consists of registers in the PE-MAC core and registers around the core for controlling DMA transfers, flow control and filtering.

Reading from reserved addresses or reserved bits leads to unpredictable data. Writing to reserved addresses or reserved bits has no effect.

Table 562: Register map of the Ethernet MAC controller

Name	R/W	Size [1]	Description
ETN1			
PE-MAC core registers			
etn1_mac1	R/W	W	MAC configuration register 1
etn1_mac2	R/W	W	MAC configuration register 2
etn1_ipgt	R/W	W	Back-to-Back Inter-Packet-Gap register
etn1_ipgr	R/W	W	Non Back-to-Back Inter-Packet-Gap register
etn1_clrt	R/W	W	Collision window / Retry register
etn1_maxf	R/W	W	Maximum frame register
etn1_supp	R/W	W	PHY support register
etn1_test	R/W	W	Test register
etn1_mcfg^[3]	R/W	W	MII Mgmt configuration register
etn1_mcmd^[3]	R/W	W	MII Mgmt command register
etn1_madr^[3]	R/W	W	MII Mgmt address register
etn1_mwtd^[3]	W	W	MII Mgmt write data register
etn1_mrdd^[3]	R	W	MII Mgmt read data register
etn1_mind^[3]	R	W	MII Mgmt indicators register
etn1_sa0	R/W	W	Station address 0 register
etn1_sa1	R/W	W	Station address 1 register
etn1_sa2	R/W	W	Station address 2 register

Table 562: Register map of the Ethernet MAC controller...continued

Name	R/W	Size [1]	Description	
Control registers				
etn1_command	R/W	W	Command register	1
etn1_status	R	W	Status register	2
etn1_rxdescriptor	R/W	W	Receive descriptor base address register	3
etn1_rxstatus	R/W	W	Receive status base address register	4
etn1_rxdescriptornumber	R/W	W	Receive number of descriptors register	5
etn1_rxproduceindex	R	W	Receive produce indexregister	6
etn1_rxconsumeindex	R/W	W	Receive consume indexregister	7
etn1_txdescriptor	R/W	W	Non real-time transmit descriptor base address register	8
etn1_txstatus	R/W	W	Non real-time transmit status base address register	9
etn1_txdescriptornumber	R/W	W	Non real-time transmit number of descriptors register	10
etn1_txproduceindex	R/W	W	Non real-time transmit produce index register	11
etn1_txconsumeindex	R	W	Non real-time transmit consume index register	12
etn1_txrtdescriptor	R/W	W	Real-time transmit descriptor base address register	13
etn1_txrtstatus	R/W	W	Real-time transmit status base address register	14
etn1_txrtdescriptornumber	R/W	W	Real-time transmit number of descriptors register	15
etn1_txrtproduceindex	R/W	W	Real-time transmit produce index register	16
etn1_txrtconsumeindex	R	W	Real-time transmit consume index register	17
etn1_qostimeout	R/W	W	Transmit quality of service time-out register	18
etn1_tsv0	R	W	Transmit status vector 0 register	19
etn1_tsv1	R	W	Transmit status vector 1 register	20
etn1_rsv	R	W	Receive status vector register	21
etn1_flowcontrolcounter	R/W	W	Flow control counter register	22
etn1_flowcontrolstatus	R	W	Flow control status register	23
Rx filter registers				
etn1_rxfilterctrl	R/W	W	Receive filter control register	24
etn1_rxfilterwolstatus	R	W	Receive filter WoL status register	25
etn1_rxfilterwolclear	W	W	Receive filter WoL clear register	26
etn1_hashfilterl	R/W	W	Hash filter LSBs register	27
etn1_hashfilterh	R/W	W	Hash filter MSBs register	28
Standard registers				
etn1_intstatus	R	W	Interrupt status register	29
etn1_intenable	R/W	W	Interrupt enable register	30
etn1_intclear	W	W	Interrupt clear register	31
etn1_intset	W	W	Interrupt set register	32
etn1_powerdown	R/W	W	Power-down register	33
etn1_moduleid	R	W	Module ID register	34

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit

11.4.2 Register definitions

Table 563: MAC configuration register 1 - etn1_mac1

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15	SOFT_RESET ^[1]	R/W	1*	Setting this bit will put all modules within the PE-MAC in reset except the Host Interface (PE-MAC core registers).
14	SIM_RESET ^[2]	R/W	0*	Setting this bit will cause a reset to the random number generator within the Transmit Function
13:12	-	R	0*	reserved
11	RESET_PEMCS_RX	R/W	0*	Setting this bit will put the MAC Control Sublayer (pause frame logic) and the receive function in PE-MAC in reset.
10	RESET_PERFUN	R/W	0*	Setting this bit will put the Receive Function logic in reset.
9	RESET_PEMCS_TX	R/W	0*	Setting this bit will put the MAC Control Sublayer (pause frame logic) and the transmit function in PE-MAC in reset.
8	RESET_PTFUN	R/W	0*	Setting this bit will put the Transmit Function logic in reset.
7:5	-	R	0*	reserved
4	LOOPBACK	R/W	0*	Setting this bit will cause the MAC Transmit interface to be loop backed to the MAC Receive interface. Clearing this bit results in normal operation.
3	TX_FLOW CONTR OL	R/W	0*	When enabled, PAUSE Flow Control frames are allowed to be transmitted. When disabled, Flow Control frames are blocked.
2	RX_FLOW CONTR OL	R/W	0*	When enabled, the MAC acts upon received PAUSE Flow Control frames. When disabled, received PAUSE Flow Control frames are ignored.
1	PASS_ALL_RECEIV E_FRAMES	R/W	0*	When enabled, the MAC will pass all receive frame regardless of type (normal vs. Control ^[3]) to the receive buffers.
0	RECEIVE_ENABLE	R/W	0*	Set this to allow receive frames to be received.

[1] SOFT_RESET needs to be cleared after a hardware reset

[2] only for simulation purpose

[3] e.g. a PAUSE frame

Table 564: MAC configuration register 2- etn1_mac2

Bit	Symbol	Access	Value	Description
31:15	-	R	0*	reserved
14	EXCESS_DEFER_R	R/W	0*	When enabled the MAC will defer to carrier indefinitely as per the Standard. When disabled, the MAC will abort when the excessive deferral limit is reached and provide feedback to the host system.
13	BACK_PRESSURE_NO_BACKOFF_FF	R/W	0*	When enabled, the MAC after incidentally causing a collision during back pressure will immediately retransmit without backoff reducing the chance of further collisions and ensuring transmit packets get sent.
12	NO_BACKOFF	R/W	0*	When enabled, the MAC will immediately retransmit following a collision rather than using the Binary Exponential Backoff algorithm as specified in the Standard.
11:10	-	R	0*	reserved
9	LONG_PREAMBLE_ENFORCEMENT	R/W	0*	When enabled, the MAC only allows receive packets which contain preamble fields less than 12 bytes in length. When disabled, the MAC allows any length preamble as per the Standard.
8	PURE_PREAMBLE_ENFORCEMENT	R/W	0*	When enabled, the MAC will verify the content of the preamble to ensure it contains 0x55 and is error-free. A packet with errored preamble is discarded.
7	AUTO_DETECT_PAD_ENABLE	R/W	0*	Set this bit to cause the MAC to automatically detect the type of frame, either tagged or un-tagged, by comparing the two octets following the source address with 0x8100 (VLAN Protocol ID) and pad accordingly. ^[1]
6	VLAN_PAD_ENABLE	R/W	0*	Set this bit to cause the MAC to pad all short frames to 64bytes and append a valid CRC.
5	PAD/CRC_ENABLE	R/W	0*	Set this bit to have the MAC pad all short frames. Clear this bit if frames presented to the MAC have a valid length. This bit is used in conjunction with AUTO_DETECT_PAD_ENABLE and VLAN_PAD_ENABLE.
4	CRC_ENABLE	R/W	0*	Set this bit to append a CRC to every frame whether padding was required or not. Must be set if PAD/CRC_ENABLE is set. Clear this bit if frames presented to the MAC contain a CRC.
3	DELAYED_CRC	R/W	0*	Set this bit to delay the CRC generation by four bytes to skip proprietary header information
2	HUGE_FRAME_ENABLE	R/W	0*	When enabled frames of any length are transmitted and received.
1	FRAME_LENGTH_CHECKING	R/W	0*	When enabled, both transmit and receive frame lengths are compared to the Length/Type field. If the Length/Type field represents a length then the check is performed. Mismatches are reported on the Transmit/Receive Statistics Vector.
0	FULL_DUPLEX	R/W	0*	When enabled, MAC operates in Full-Duplex mode. Disable for Half-Duplex operation.

[1] This bit is ignored if PAD/CRC ENABLE is cleared.

Table 565: Back-toBack Inter-Packet-Gap register- etn1_ipgt

Bit	Symbol	Access	Value	Description
31:7	-	R	0*	reserved
6:0	BACK_TO_BAC K_INTER_PACK ET_GAP	R/W	0*	<p>This is a programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet, to the beginning of the next.</p> <p>In Full-Duplex mode, the register value should be the desired period in nibble times minus 3^[1].</p> <p>In Half-Duplex mode, the register value should be the desired period in nibble times minus 6^[2].</p>

[1] In Full-Duplex the recommended setting is 0x15, which represents the minimum InterPacketGap (IPG) of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s).

[2] In Half-Duplex the recommended setting is 0x12, which also represents the minimum IPG of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s)

Table 566: Non Back-toBack Inter-Packet-Gap register- etn1_ipgr

Bit	Symbol	Access	Value	Description
31:15	-	R	0*	reserved
14:8	NON_BACK_TO _BACK_INTER_ PACKET_GAP_ PART1	R/W	0*	<p>This is a programmable field representing the optional carrierSense window referenced in IEEE 802.3/ 4.2.3.2.1 'Carrier Deference'. If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x0 to IPGR2.</p>
7	-	R	0*	reserved
6:0	NON_BACK_TO _BACK_INTER_ PACKET_GAP_ PART2	R/W	0*	<p>This is a programmable field representing the Non- Back-to-Back Inter-Packet-Gap. Default is 0x12 (18d), which represents the minimum IPG of 0.96 µs (in 100 Mb/s) or 9.6 µs (in 10 Mb/s).</p>

Table 567: Collision window / Retry register- etn1_clrt

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13:8	COLLISION_WI NDOW	R/W	0x37*	<p>This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Since the collision window starts at the beginning of transmission, the preamble and SFD is included. Its default of 0x37 corresponds to the count of frame bytes at the end of the window</p>
7:4	-		0*	reserved
3:0	RETRANSMISSI ON_MAXIMUM	R/W	0xF*	<p>This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The Standard specifies the attemptLimit to be 0xF.</p>

Table 568: Maximum Frame register- etn1_maxf

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	MAXIMUM_FRA ME_LENGTH	R/W	0x0600*	This field resets to 0x0600, which represents a maximum receive frame of 1536 octets. An untagged maximum size Ethernet frame is 1518 octets. A tagged frame adds four octets for a total of 1522 octets. If a shorter maximum length restriction is desired, program this 16-bit field.

Table 569: PHY Support register - etn1_supp

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:13	-	R	0*	reserved
12	PHY_MODE	R/W	1*	This bit configures the Serial MII logic with the connecting SMII device type. Set this bit when connecting to a SMII PHY. Clear this bit when connecting to a SMII MAC. When MAC is selected, the SMII will operate at 100 Mb/s, Full Duplex.
11	RESET_PERMII	R/W	0*	This bit resets the Reduced MII logic.
10:9	-	R	0*	reserved
8	SPEED	R/W	0*	This bit configures the Reduced MII logic for the current operating speed. When set, 100 Mb/s mode is selected. When cleared, 10 Mb/s mode is selected.
7	RESET_PE_100 X[1]	R/W	0*	This bit resets the 100Mb part of the MAC. The module contains the 4B/5B symbol encipher/decipher logic.
6	FORCE QUIET [1]	R/W	0*	When enabled, transmit data is quieted which allows the contents of the cipher to be output. When cleared, normal operation is enabled. Effects 100Mb module only.
5	NO_CIPHER[1]	R/W	0*	When enabled, the raw transmit 5B symbols are transmitted without ciphering. When disabled, normal ciphering occurs. Effects the 100Mb mode only.
4	DISABLE_LINK_ FAIL[1]	R/W	0*	When enabled, the 330ms Link Fail timer is disabled allowing for shorter simulations. Removes the 330 ms link-up time before reception of streams is allowed. When cleared, normal operation occurs. Effects 100Mb module only.
3	RESET_PE10T[1]	R/W	0*	This bit resets the 10Mb part of the MAC, which converts MII nibble streams to the serial bit stream of 10Mb transceivers.
2	-	R	0*	reserved
1	ENABLE_JABBE R_PROTECTIO N	R/W	0*	This bit enables the Jabber Protection logic within the 10Mb module in ENDEC mode. Jabber is the condition where a transmitter is stuck on for longer than 50ms preventing other stations from transmitting. Effects 10Mb mode only.
0	BIT_MODE	R/W	0*	When '1' - MAC is in 10BASE-T ENDEC mode which changes decodes (such as Excess Defer) to be based on the bit clock rather than the nibble clock.

[1] only for debugging and simulation purpose

Table 570: Test register - etn1_test

Bit	Symbol	Access	Value	Description
31:3	-	R	0*	reserved
2	TEST_BACK_PR ESSURE	R/W	0*	Setting this bit will cause the MAC to assert back pressure on the link. Back pressure causes preamble to be transmitted, raising carrier sense. A transmit packet from the system will be sent during back pressure.
1	TEST_PAUSE ^[1]	R/W	0*	This bit causes the MAC Control sublayer to inhibit transmissions, just as if a PAUSE Receive Control frame with a non-zero pause time parameter was received.
0	SHORTCUT_PA USE_QUANTA ^[1]	R/W	0*	This bit reduces the effective PAUSE Quanta from 64 byte-times to 1 byte-time.

[1] only for debugging and simulation purpose

Table 571: MII Mgmt Configuration register - etn1_mcfg

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15	RESET_MII_MG MT	R/W	0*	This bit resets the MII Management module.
14:5	-	R	0*	reserved
4:2	CLOCK_SELEC T ^[2]	R/W	0*	This field is used by the clock divide logic in creating the MII Management Clock (MDC) which IEEE 802.3u defines to be no faster than 2.5 MHz. Some PHYs support clock rates up to 12.5 MHz.
1	SUPPRESS_PR EAMBLE	R/W	0*	Setting this bit will cause the MII Management module to perform read/write cycles without the 32-bit preamble field.
0	SCAN_INCREM ENT	R/W	0*	Setting this bit will cause the MII Management module to perform read cycles across a range of PHYs. When set, the MII Management module will perform read cycles from address 1 through the value set in etn1_madr.PHY_ADDR[4:0] .

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_mcfg is not used

[2] Clock encoding value is:
 00x = clock divided by 4
 010 = clock divided by 6
 011 = clock divided by 8
 100 = clock divided by 10
 101 = clock divided by 14
 110 = clock divided by 20
 111 = clock divided by 28

Table 572: MII Mgmt Command register - etn1_mcmd

Bit	Symbol	Access	Value	Description
31:2	-	R	0*	reserved
1	SCAN	R/W	0*	This bit causes the MII Management module to perform Read cycles continuously. This is useful for monitoring Link Fail for example.
0	READ	R/W	0*	This bit causes the MII Management module to perform a single Read cycle. The Read data is returned in Register etn1_mrdd (MII Mgmt Read Data).

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_mcnd is not used

Table 573:MII Mgmt Address register - etn1_madr

Bit	Symbol	Access	Value	Description
31:13	-	R	0*	reserved
12:8	PHY_ADDR	R/W	0*	This field represents the 5-bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is reserved).
7:5	-	R	0*	reserved
4:0	REG_ADDR	R/W	0*	This field represents the 5-bit Register Address field of MII Mgmt interface. Up to 32 registers can be accessed.

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_madr is not used

Table 574:MII Mgmt Write Data register - etn1_mwtd

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	WRITE_DATA	W	0	When written, a MII Mgmt write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from register etn1_madr

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_mwtd is not used

Table 575:MII Mgmt Read Data register - etn1_mrdd

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	READ_DATA	R	0*	Following a MII Mgmt Read Cycle, the 16-bit data can be read from this location.

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_mrdd is not used

Table 576:MII Mgmt Indicators register - etn1_mind

Bit	Symbol	Access	Value	Description
31:4	-	R	0*	reserved
3	MII_LINK_FAIL	R	0*	When '1' is returned - indicates MII link fail has occurred.
2	NOT_VALID	R	0*	When '1' is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.
1	SCANNING	R	0*	When '1' is returned - indicates a scan operation (continuous MII Mgmt Read cycles) is in progress.
0	BUSY	R	0*	When '1' is returned - indicates MII Mgmt module is currently performing an MII Mgmt Read or Write cycle.

[1] Only ETN1-PHY-MIIM is connected to the MIIM interface. etn2_mind is not used

Table 577:Station Address 0 register - etn1_sa0

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:8	STATION_ADR1	R/W	0*	This field holds the first octet of the station address.
7:0	STATION_ADR2	R/W	0*	This field holds the second octet of the station address.

[1] The station address (=MAC address) is used for perfect address filtering and for sending pause control frames.

Table 578:Station Address 1 register - etn1_sa1

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:8	STATION_ADR3	R/W	0*	This field holds the third octet of the station address.
7:0	STATION_ADR4	R/W	0*	This field holds the fourth octet of the station address.

[1] The station address is used for perfect address filtering and for sending pause control frames.

Table 579:Station Address 2 register - etn1_sa2

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:8	STATION_ADR5	R/W	0*	This field holds the fifth octet of the station address.
7:0	STATION_ADR6	R/W	0*	This field holds the sixth octet of the station address.

[1] The station address is used for perfect address filtering and for sending pause control frames.

Table 580:Command register - etn1_command

Bit	Symbol	Access	Value	Description
31:11	-	R	0*	reserved
10	FullDuplex	R/W	0*	When set to "1" indicates full duplex operation
9	RMII ^[2]	R/W	1	enable RMII mode;
			0*	interface disabled
8	Tx_Flow_Control	R/W	0*	Enable IEEE 802.3 / clause 31 flow control sending pause frames in full duplex and continuous preamble in half duplex
7	PassRxFilter	R/W	0*	When set to "1" disables receive filtering i.e. all frames received are written to memory
6	PassRuntFrame	R/W	0*	When set to "1" passes runt frames smaller than 64 bytes to memory unless they have a CRC error. If "0" runt frames are filtered out.
5	RxReset	W	0	If writing "1" reset the receive data path
4	TxReset	W	0	If writing "1" reset the transmit data path
3	RegReset	W	0	If writing "1" reset all data paths and the host registers. PE-MAC needs to be reset separately.
2	TxRtEnable	R/W	0*	Enable real-time transmit channel
1	TxEnable	R/W	0*	Enable non real-time transmit channel
0	RxEnable	R/W	0*	Enable receive

[1] All bits can be written and read. The Tx/TxRt/RxReset bits are write only, reading will return 1'b0.

[2] ETN1 using RMII interface, therefore this bit must be set

Table 581:Status register - etn1_status

Bit	Symbol	Access	Value	Description
31:3	-	R	0*	reserved
2	TxRtStatus	R	0*	If 1 real-time transmit channel is active, if 0 the channel is inactive
1	TxStatus	R	0*	If 1 non real-time transmit channel is active, if 0 the channel is inactive
0	RxStatus	R	0*	If 1 receive channel is active, if 0 the channel is inactive

- [1] The values represent the status of the three channels/data paths. In case the status is 1 the channel is active meaning it:
 - is enabled and the Rx/TxRt/TxEnable bit is set in the **etn1_command** register or it just got disabled while still transmitting or receiving a frame
 - and for transmit channels the transmit queue is not empty i.e. ProduceIndex != ConsumeIndex
 - and for the receive channel the receive queue is not full i.e. ProduceIndex != ConsumeIndex - 1
- A status transitions from active to inactive will also happen:
 - if the channel is disabled by resetting the Rx/Tx/ TxRtEnable bit in the **etn1_command** register and
 - if the channel status and data have been committed to memory.

Table 582:Receive descriptor base address register - etn1_rxdescriptor

Bit	Symbol	Access	Value	Description
31:2	RxDescriptor	R/W	0*	MSBs of receive descriptor base address
1:0	-	R	0*	Fixed to 2.b00

- [1] The register contains the lowest address in the array of descriptors.

Table 583:Receive status base address register - etn1_rxstatus

Bit	Symbol	Access	Value	Description
31:3	RxStatus	R/W	0*	MSBs of receive status base address
2:0	-	R	0*	Fixed to 3'b000

Table 584:Receive number of descriptors register - etn1_rxdescriptornumber

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	RxDescriptorNumber	R/W	0*	Number of descriptors in the descriptor array for which RxDescriptor is the base address

- [1] The number of descriptors should match the number of status elements. The register uses minus one encoding i.e. if the array has 8 status elements, the value in the register should be 7.

Table 585:Receive produce index register - etn1_rxproduceindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	RxProduceIndex	R	0*	Index of the descriptor that is going to be filled next by the receive data path

- [1] After a frame has been received the hardware increments the index. The value will be wrapped to 0 once the RxDescriptorNumber has been reached. If the RxProduceIndex equals (RxConsumeIndex - 1) the array is full and any further frames being received will cause a buffer overrun error.

Table 586:Receive consume index register - etn1_rxconsumeindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	RxConsumeIndex	R/W	0*	Index of the descriptor that is going to be processed next by the receive software

[1] The receive array is empty as long as RxProduceIndex equals RxConsumeIndex. As soon as the array is not empty software can process the frame pointed to by RxConsumeIndex. After a frame has been processed by software, software should increment the RxConsumeIndex while wrapping to 0 once the RxDescriptorNumber has been reached. If the RxProduceIndex equals RxConsumeIndex - 1 the array is full and any further frames being received will cause a buffer overrun error.

Table 587:Non real-time transmit descriptor base address register - etn1_txdescriptor

Bit	Symbol	Access	Value	Description
31:2	TxDescr	R/W	0*	MSBs of non real-time descriptor base address
1:0	-	R	0*	Fixed to 2'b00

Table 588:Non real-time transmit status base address register - etn1_txstatus

Bit	Symbol	Access	Value	Description
31:2	TxStatus	R/W	0*	MSBs of non real-time transmit status base address
1:0	-	R	0*	Fixed to 2'b00

Table 589:Non real-time transmit number of descriptors register - etn1_txdescriptornumber

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxDescrNum	R/W	0*	Number of descriptors in the descriptor array for which TxDescr is the base address.

[1] The number of descriptors should match the number of statuses. The register uses minus one encoding i.e. if the array has 8 status elements, the value in the register should be 7.

Table 590:Non real-time transmit produce index register - etn1_txproduceindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxProduc	R/W	0*	Index of the descriptor that is going to be filled next by the non real-time transmit software driver

[1] The transmit descriptor array is empty as long as TxProducIndex equals TxConsumeIndex. As soon as the array is not empty the non real-time transmit hardware will start transmitting frames if enabled. After a frame has been processed by software, software should increment the TxProducIndex while wrapping to 0 once the TxDescriptorNumber has been reached. If the TxProducIndex equals TxConsumeIndex - 1 the descriptor array is full and software should stop producing new descriptors until hardware has transmitted some frames and updated the TxConsumeIndex

Table 591:Non-real-time transmit consume index register - etn1_txconsumeindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxConsumeInde	R/W	0*	Index of the descriptor that is going to be transmitted next by the non real-time transmit software

- [1] After a frame has been transmitted hardware increments the index while wrapping to 0 once the TxDescriptorNumber has been reached. If the TxConsumeIndex equals TxProduceIndex the descriptor array is empty and the transmit channel will stop transmitting until software produces new descriptors.

Table 592: Real-time transmit descriptor base address register - etn1_txrtdescriptor

Bit	Symbol	Access	Value	Description
31:2	TxRtDescriptor	R/W	0*	MSBs of real-time descriptor base address
1:0	-	R	0*	Fixed to 2'b00

Table 593: Real-time transmit status base address register - etn1_txrtstatus

Bit	Symbol	Access	Value	Description
31:2	TxRtStatus	R/W	0*	MSBs of real-time transmit status base address
1:0	-	R	0*	Fixed to 2'b00

Table 594: Real-time transmit number of descriptors register - etn1_txrtdescriptornumber

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxRtDescriptorNumber	R/W	0*	Number of descriptors in the descriptor array for which TxRtDescriptor is the base address.

- [1] The number of descriptors should match the number of statuses. The register uses minus one encoding i.e. if the array has 8 status elements, the value in the register should be 7.

Table 595: Real-time transmit produce index register - etn1_txrtproduceindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxRtProduceIndex	R/W	0*	Index of the descriptor that is going to be filled next by the real-time transmit software driver

- [1] The transmit descriptor array is empty as long as TxRtProduceIndex equals TxRtConsumeIndex. As soon as the array is not empty the non real-time transmit hardware will start transmitting frames if enabled. After a frame has been processed by software, software should increment the TxProduceIndex while wrapping to 0 once the TxRtDescriptorNumber has been reached. If the TxRtProduceIndex equals TxRtConsumeIndex - 1 the descriptor array is full and software should stop producing new descriptors until hardware has transmitted some frames and updated the TxRtConsumeIndex

Table 596: Real-time transmit consume index register - etn1_txrtconsumeindex

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	TxRtConsumeIndex	R/W	0*	Index of the descriptor that is going to be transmitted next by the real-time transmit software

- [1] After a frame has been transmitted the hardware increments the index. The value will be wrapped to 0 once the TxRtDescriptorNumber has been reached. If the TxRtConsumeIndex equals TxRtProduceIndex the descriptor array is empty and the transmit channel will stop transmitting until software produces new descriptors.

Table 597: Transmit quality of service time-out register - etn1_qostimeout

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	QoSTimeout	R/W	0*	Specifies the maximum number of clock cycles a low priority transmission has to wait for transmission. If the time-out counter expires the low priority transmission will get the highest priority. QoSTimeout is specified in units of 64 times the transmit clock cycle.

Table 598: Transmit status vector 0 register - etn1_tsv0

Bit	Symbol	Access	Value	Description
31	VLAN	R	0*	Frame's length/type field contained 0x8100 which is the VLAN protocol identifier
30	Backpressure	R	0*	Carrier-sense method back pressure was previously applied
29	PAUSE	R	0*	The frame was a control frame with a valid PAUSE opcode
28	Control_frame	R	0*	The frame was a control frame
27-12	Total_bytes	R	0*	The total number of bytes transferred including collided attempts
11	Underrun	R	0*	Host side caused buffer underrun
10	Giant	R	0*	Byte count in frame was greater than etn1_maxf[15:0]
9	Late_Collision	R	0*	Collision occurred beyond collision window, 512 bit times
8	Max_Collision	R	0*	Packet was aborted due to exceeding of maximum allowed number of collisions
7	Excessive_Defer	R	0*	Packet was deferred in excess of 6071 nibble times in 100Mb/s or 24287 bit times in 10Mb/s mode
6	Packet_Defer	R	0*	Packet was deferred for at least one attempt, but less than an excessive defer
5	Broadcast	R	0*	Packet's destination was a broadcast address
4	Multicast	R	0*	Packet's destination was a multicast address
3	Done	R	0*	Transmission of packet was completed
2	Length_out_of_range	R	0*	Indicates that frame type/length field was larger than 1500 bytes
1	Length_check_error	R	0*	Indicates the frame length field does not match the actual number of data items and is not a type field
0	CRC_error	R	0*	The attached CRC in the packet did not match the internal generated CRC

Table 599: Transmit status vector 1 register - etn1_tsv1

Bit	Symbol	Access	Value	Description
31:20	-	R	0*	reserved
19:16	Transmit_collision_count	R	0*	Number of collisions current packet incurred during transmission attempts. The maximum number of collisions (16) cannot be represented.
15:0	Transmit_byte_count	R	0*	The total number of bytes in the frame not counting the collided bytes

Table 600:Receive status vector register - etn1_rsv

Bit	Symbol	Access	Value	Description
31	-	R	0*	reserved
30	VLAN	R	0*	Frame.s length/type field contained 0x8100 which is the VLAN protocol identifier
29	Unsupported_Op_code	R	0*	Current frame was recognized as a Control Frame but contains an unknown opcode
28	Pause	R	0*	The frame was a control frame with a valid PAUSE opcode
27	Control_frame	R	0*	The frame was a control frame
26	Dribble_Nibble	R	0*	Indicates that after the end of packet another 1-7 bits were received. A single nibble, called dribble nibble, is formed but not sent out
25	Broadcast	R	0*	Packet's destination was a broadcast address
24	Multicast	R	0*	Packet's destination was a multicast address
23	Receive_OK	R	0*	Packet had valid CRC and no symbol errors
22	Length_out_of_range	R	0*	Indicates that frame type/length field was larger than 1518 bytes
21	Length_check_error	R	0*	Indicates the frame length field does not match the actual number of data items and is not a type field
20	CRC_error	R	0*	The attached CRC in the packet did not match the internal generated CRC
19	Receive_code_violation	R	0*	Indicates that RMII data does not represent a valid receive code when mrixer asserts during the data phase of a frame
18	Carrier_event_previous_seen	R	0*	Indicates that at some time since the last receive statistics, a carrier event was detected
17	RXDV_event_previous_seen	R	0*	Indicates that the last receive event seen was not long enough to be a valid packet
16	Packet_previous_ignored	R	0*	Indicates that a packet since the last RSV was dropped
15:0	Received_byte_count	R	0*	Indicates length of received frame

Table 601:Flow control counter register - etn1_flowcontrolcounter

Bit	Symbol	Access	Value	Description
31:16	PauseTimer	R/W	0*	In full-duplex mode the PauseTimer specifies the value that is inserted into the pause timer field of a pause flow control frame. In half duplex mode the PauseTimer specifies the number of back pressure cycles.
15:0	MirrorCounter	R/W	0*	In full duplex mode the MirrorCounter specifies the number of cycles before re-issuing the Pause control frame. In half duplex mode: If the MirrorCounter is not 0 the number of backpressure cycles is specified by the value of the PauseTimer. If the MirrorCounter is 0 there will be backpressure cycles until TX flow control is disabled.

Table 602: Flow control status register - etn1_flowcontrolstatus

Bit	Symbol	Access	Value	Description
31:16	-	R	0*	reserved
15:0	MirrorCounterCur rent	R	0*	In full duplex mode this register represent the current value of the data path's mirror counter which counts up to the value of the MirrorCounter bits from the etn1_flowcontrolcounter register. In half duplex mode the register counts until it reaches the value of the PauseTimer bits in the etn1_flowcontrolcounter register.

Table 603: Receive filter control register - etn1_rxfilterctrl

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13	RxFilterEnWoL	R/W	0*	When set to one the result of the perfect address matching filter and the imperfect hash filter will generate a WoL interrupt in case of a match.
12	MagicPacketEn WoL	R/W	0*	When set to one the result of the magic packet filter will generate a WoL interrupt in case of a match.
11:6	-		0*	reserved
5	AcceptPerfectEn	R/W	0*	When set to "1" the frames with a DA address identical to the station address are accepted.
4	AcceptMulticastH ashEn	R/W	0*	When set to "1" multicast frames that pass the imperfect hash filter are accepted.
3	AcceptUnicastHa sh_En	R/W	0*	When set to "1" unicast frames that pass the imperfect hash filter are accepted.
2	AcceptMulticastE n	R/W	0*	When set to "1" all multicast frames are accepted.
1	AcceptBroadcast En	R/W	0*	When set to "1" all broadcast frames are accepted.
0	AcceptUnicastEn	R/W	0*	When set to "1" all unicast frames are accepted.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 604:Receive filter WoL status register - etn1_rxfilterwolstatus

Bit	Symbol	Access	Value	Description
31:9	-	R	0*	reserved
8	MagicPacketWoL	R	0*	When the value is “1” the magic packet filter caused WoL.
7	RxFilterWoL	R	0*	When the value is “1” the receive filter caused WoL.
6	-	R	0*	reserved
5	AcceptPerfectWoL	R	0*	When the value is “1” the perfect address matching filter caused WoL.
4	AcceptMulticastHashWoL	R	0*	When the value is “1” a multicast frame that pass the imperfect hash filter caused WoL.
3	AcceptunicasthashwoL	R	0*	When the value is “1” a unicast frame that pass the imperfect hash filter caused WoL.
2	AcceptMulticastWoL	R	0*	When the value is “1” a multicast frame caused WoL.
1	AcceptBroadcastWoL	R	0*	When the value is “1” a broadcast frame caused WoL.
0	AcceptUnicastWoL	R	0*	When the value is “1” a unicast frames caused WoL.

Table 605:Receive filter WoL clear register - etn1_rxfilterwolclear

Bit	Symbol	Access	Value	Description
31:9	-	W	0	reserved
8	MagicPacketWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
7	RxFilterWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
6	reserved	W	0	unused
5	AcceptPerfectWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
4	AcceptMulticastHashWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
3	AcceptUnicastHashWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
2	AcceptMulticastWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
1	AcceptBroadcastWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.
0	AcceptUnicastWoLClr	W	0	Writing “1” clears the corresponding status bit in the etn1_rxfilterwolstatus register.

Table 606:Hash filter table LSBs register - etn1_hashfilter

Bit	Symbol	Access	Value	Description
31:0	HshFilterL	R/W	0	Bit 31:0 of the imperfect filter hash table for receive filtering.

Table 607: Hash filter table MSBs register - etn1_hashfilterh

Bit	Symbol	Access	Value	Description
31:0	HashFilterH	R/W	0	Bit 63:32 of the imperfect filter hash table for receive filtering.

Table 608: Interrupt status register - etn1_intstatus

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13	WakeupInt	R	0*	Interrupt triggered by a Walk-up event detected by the receive filter
12	SoftInt	R	0*	Interrupt triggered by software writing a 1 in the etn1_intset .Softinterrupt register
11	TxRtDoneInt	R	0*	Interrupt triggered in case a real-time descriptor has been transmitted while the Control.Interrupt bit in the descriptor was set.
10	TxRtFinishedInt	R	0*	Interrupt triggered when all real-time descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex
9	TxRtErrorInt	R	0*	Interrupt trigger on real-time transmit errors: LateCollision, ExcessiveCollision and ExcessiveDefer, NoDescriptor or Underrun
8	TxRtUnderrunInt	R	0*	Interrupt set on a fatal underrun error in the real-time transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set in case of a non fatal underrun error.
7	TxDoneInt	R	0*	Interrupt triggered in case a non real-time descriptor has been transmitted while the Control.Interrupt bit in the descriptor was set.
6	TxFinishedInt	R	0*	Interrupt triggered when all non real-time descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex
5	TxErrorInt	R	0*	Interrupt trigger on non real-time transmit errors: Late- Collision, ExcessiveCollision and ExcessiveDefer, NoDescriptor or Underrun
4	TxUnderrunInt	R	0*	Interrupt set on a fatal underrun error in the non real-time transmit queue. The fatal interrupt should be resolved by a Tx soft-reset. The bit is not set in case of a non fatal underrun error.
3	RxDoneInt	R	0*	Interrupt triggered in case a receive descriptor has been processed while the Control.Interrupt bit in the descriptor was set.
2	RxFinishedInt	R	0*	Interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex - 1 (with wrap around taken into account)
1	RxErrorInt	R	0*	Interrupt trigger on receive errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError or NoDescriptor or Overrun
0	RxOverrunInt	R	0*	Interrupt set on a fatal overrun error in the receive queue. The fatal interrupt should be resolved by a Rx soft-reset. The bit is not set in case of a non fatal overrun error.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 609:Interrupt enable register - etn1_intenable

Bit	Symbol	Access	Value	Description
31:14	-	R	0*	reserved
13	WakeupIntEn	R/W	0*	Enable for interrupt triggered by a Walk-up event detected by the receive filter
12	SoftIntEn	R/W	0*	Enable for interrupt triggered by software writing a 1 in the register etn1_intset.SoftIntSet
11	TxRtDoneIntEn	R/W	0*	Enable for interrupt triggered in case a real-time descriptor has been transmitted while the Control.Interrupt bit in the descriptor was set.
10	TxRtFinishedIntEn	R/W	0*	Enable for interrupt triggered when all real-time descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex
9	TxRtErrorIntEn	R/W	0*	Enable for interrupt trigger on real-time transmit errors
8	TxRtUnderrunIntEn	R/W	0*	Enable for interrupt trigger on real-time transmit buffer or descriptor underrun situations
7	TxDoneIntEn	R/W	0*	Enable for interrupt triggered in case a non real-time descriptor has been transmitted while the Control.Interrupt bit in the descriptor was set.
6	TxFinishedIntEn	R/W	0*	Enable for interrupt triggered when all non real-time descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex
5	TxErrorIntEn	R/W	0*	Enable for interrupt trigger on non real-time transmit errors
4	TxUnderrunIntEn	R/W	0*	Enable for interrupt trigger on non real-time transmit buffer or descriptor underrun situations
3	RxDoneIntEn	R/W	0*	Enable for interrupt triggered in case a receive descriptor has been processed while the Control.Interrupt bit in the descriptor was set.
2	RxFinishedIntEn	R/W	0*	Enable for interrupt triggered when all receive descriptors have been processed i.e. on the transition to the situation where ProduceIndex == ConsumeIndex - 1 (with wrap around taken into account)
1	RxErrorIntEn	R/W	0*	Enable for interrupt trigger on receive errors
0	RxOverrunIntEn	R/W	0*	Enable for interrupt trigger on receive buffer overrun or descriptor underrun situations

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 610:Interrupt clear register - etn1_intclear

Bit	Symbol	Access	Value	Description
31:14	-	W	0	reserved
13	WakeupIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
12	SoftIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
11	TxRtDoneIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
10	TxRtFinishedIntC lr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
9	TxRtErrorIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
8	TxRtUnderrunInt Clr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
7	TxDoneIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
6	TxFinishedIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
5	TxErrorIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
4	TxUnderrunIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
3	RxDoneIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
2	RxFinishedIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
1	RxErrorIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .
0	RxOverrunIntClr	W	0	Writing a 1.b1 clears corresponding status bit in interrupt status register etn1_intstatus .

[1] The interrupt clear register is write-only. Writing a 1 to bit of the register clears the corresponding bit in the status register. Writing a 1.b0 will not affect the interrupt status.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Table 611: Interrupt set register - etn1_intset

Bit	Symbol	Access	Value	Description
31:14	-	W	0	reserved
13	WakeupIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
12	SoftIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
11	TxRtDoneIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
10	TxRtFinishedIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
9	TxRtErrorIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
8	TxRtUnderrunIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
7	TxDoneIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
6	TxFinishedIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
5	TxErrorIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
4	TxUnderrunIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
3	RxDoneIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
2	RxFinishedIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
1	RxErrorIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .
0	RxOverrunIntSet	W	0	Writing a 1.b1 sets the corresponding status bit in interrupt status register etn1_intstatus .

[1] The interrupt set register is write-only. Writing a 1 to bit of the register sets the corresponding bit in the status register. Writing a 1.b0 will not affect the interrupt status.

Table 612: Power-down register - etn1_powerdown

Bit	Symbol	Access	Value	Description
31	PowerDown	R/W	0*	If true all MMIO accesses will return a read/write error except accesses to the PowerDown register
30:0	reserved		0*	unused

[1] In case the Ethernet core is in a power-down state in which one or more clocks except the MMIO clock are switched off (e.g. because no PHY is connected) then software should set the PowerDown bit in the **etn1_powerdown** register in order to prevent deadlock (e.g. due to speculative read operation of the CPU). Setting the bit will return an error on all read and write accesses on the MMIO interface except for accesses to the **etn1_powerdown** register.

Table 613:Module ID register - etn1_moduleid

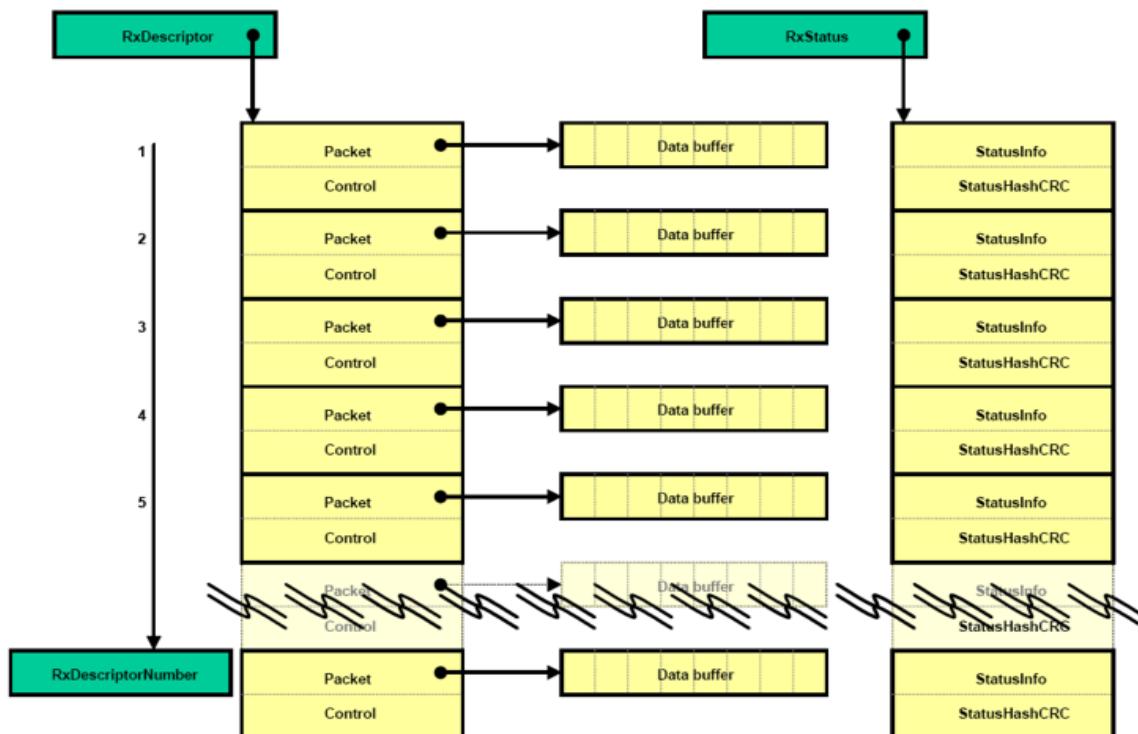
Bit	Symbol	Access	Value	Description
31:16	ModuleID	R	0x3902*	Unique 16-bit module identification. Value will be 0x3902
15:12	MajRev	R	2*	Major design revision number. The actual value depends on the hardware version
11:8	MinRev	R	1*	Minor design revision number. The actual value depends on the hardware version.
7:0	ApertureSize	R	0*	Represents the aperture of the software registers in the MMIO register space. Aperture size is 4Kb corresponding to a value 0 in this field.

11.4.3 Descriptor and status formats

Each Ethernet frame can consists of multiple fragments each fragment corresponds to a single descriptor. The DMA managers in the ETN1scatter (for receive) and gather (for transmit) multiple fragments into a single Ethernet frame.

11.4.3.1 Receive descriptors and statuses

Figure 194 depicts the layout of the receive descriptors in memory.

**Fig 194.Receive descriptor memory layout**

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

In memory the receive descriptors are stored in an array. The lowest address of the array is stored in the **etn1_rxdescriptor** register which should be aligned on a 8 byte address boundary. The number of descriptors in the array is stored in the **etn1_rxdescriptornumber** register of the ETN1. Parallel to the descriptors there is an array of status elements. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the **etn1_rxstatus** register which must be aligned on an 8 byte address boundary. During operation i.e. when the receive data path is enabled the **etn1_rxdescriptor**, **etn1_rxstatus** and **etn1_rxdescriptornumber** registers should not be modified.

The register view of the ETN1 has two registers **etn1_rxconsumeindex** and **etn1_rxproduceindex** - both counters starting at 0 and wrapping at **etn1_rxdescriptornumber** - which indicate the number of descriptors that have been processed. The **etn1_rxproduceindex** contains the index of the descriptor that is going to be filled with the next frame being received. The **etn1_rxconsumeindex** is programmed by software and is the index of the next descriptor that the software receive driver is going to process. In case **etn1_rxproduceindex==etn1_rxconsumeindex** it means the receive buffer is empty. In case **etn1_rxproduceindex == etn1_rxconsumeindex-1** it means the receive buffer is full and newly received data would generate an overflow unless the software driver would free up some descriptors.

Each receive descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes two words (8 bytes) in memory. Each receive descriptor consists of a pointer to the data buffer for storing receive data (Packet) and a control word (Control). The Packet field has a zero address offset, the control field has a 4 byte address offset with respect to the descriptor address as defined in [Table 614](#).

Table 614:Receive descriptor fields

Address offset	Name	R/W	Size [1]	Description
0x0	Packet	R	W	Base address of the data buffer for storing receive data
0x4	Control	R	W	Control information, see Table 615

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit

The data buffer pointer (Packet) is a word aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in [Table 615](#).

Table 615:Receive descriptor control word

Bit	Symbol	Description
31	Interrupt ^[1]	If true generate an RxDone interrupt when the data in this frame or frame fragment and the associated status information has been committed to memory.
30:11	-	reserved
10:0	Size ^{[2][3]}	Size in bytes of the data buffer. This is the size of the buffer reserved by the device driver for a frame or frame fragment i.e. the byte size of the buffer pointed to by the Packet field.

[1] Interrupt-bit must be set. If it is not set, then the RxProduceIndex will not be updated.

[2] The size is -1 encoded e.g. if the buffer is 8 bytes the size field should be equal to 7

[3] The buffer size must be a multiple of 32 bytes

Table 616 lists the fields in the receive status elements from the status array.

Table 616:Receive status fields

Address offset	Name	R/W	Size [1]	Description
0x0	StatusInfo	R	W	Receive status return flags, see Table 618
0x4	StatusHashCRC	R	W	The concatenation of the destination address hash CRC and the source address hash CRC

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit

Each receive status consists of two words. The StatusHashCRC word contains a concatenation of the two 9-bit hash CRCs calculated from the destination and source addresses contained in the received frame. After detecting the destination and source addresses, StatusHashCRC is calculated once, then held for every fragment of the same frame.

The concatenation of the two CRCs is shown in [Table 617](#):

Table 617:Receive Status hash CRC word

Bit	Symbol	Description
31:25	reserved	unused
24:16	DAHashCRC	Hash CRC calculated from the destination address
15:9	reserved	unused
8:0	SAHashCRC	Hash CRC calculated from the source address

The StatusInfo word contains flags returned by the PE-MAC and flags generated by the receive data path reflecting the status of the reception. [Table 618](#) lists the bit definitions in the StatusInfo word.

Table 618:Receive Status information word

Bit	Symbol	Description
31	Error	An error occurred during reception of this frame. This is a logic OR of AlignmentError, RangeError, LengthError, SymbolError, CRCError, Overrun.
30	LastFrag	When set to 1, it indicates this descriptor is for the last fragment of a frame. If the frame consists of a single fragment, this bit is also set to 1.
29	NoDescriptor	No new Rx descriptor is available and the frame is too long for the buffer size in the current receive descriptor.
28	Overrun	Receive overrun. The adapter can not accept the data stream.
27	AlignmentError	An alignment error is flagged when dribble bits are detected and also a CRC error is detected. This is in accordance with IEEE std. 802.3/clause 4.3.2.
26	RangeError ^[1]	The received packet exceeds max packet size.
25	LengthError	The frame length field value in the frame does not match the actual data byte-length and specifies a valid length.
24	SymbolError	The PHY reports a bit error over the RMII during reception.
23	CRCError	CRC error
22	Broadcast	The received frame is of type broadcast.
21	Multicast	A multicast frame is received.
20	FailFilter	Indicates this frame has failed the Rx filter. These frames are not supposed to pass to memory. But because the limitation of buffer FIFO in size, part of this frame may already passed to memory. Once the frame was found failed the Rx filter, the remaining of the frame will be discarded without passing to the memory. However, if Command. PassRxFilter is set, the whole frame will be passed to memory.
19	VLAN	Indicates a VLAN frame.
18	ControlFrame	Indicates this is a control frame for flow control, either a pause frame or a frame with an unsupported opcode.
17:11	-	reserved
10:0	EntryLevel	Size in bytes of the actual data transferred into one fragment buffer. In other words, this is the size of the frame or fragment as actually written by the DMA manager for one descriptor. This may be different from the Control.Size bits that indicate the size of the buffer allocated by the device driver. Size is -1 encoded e.g. if the buffer has 8 bytes the EntryLevel value should be 7.

[1] A valid frame with a Length/Type field larger than **etn1_maxf** (default value: 1536 bytes) will produce a RangeError. In this case the driver software needs to ignore this error

For multi fragment frames the value of the AlignmentError/RangeError/LengthError/SymbolError/ CRCError bits in all but the last fragment in the frame will be 0; likewise the value of the FailFilter/Multicast/ Broadcast/VLAN/ControlFrame bits is undefined. The status of the last fragment in the frame will copy the value for these bits from PE-MAC. All fragment status elements will have a valid LastFrag/ EntryLevel/Error/Overrun/NoDescriptor bits.

11.4.3.2 Transmit descriptor and status

Figure 195 depicts the layout of the transmit descriptors in memory. The layout and format of real-time (when present) and non real-time descriptors is identical.

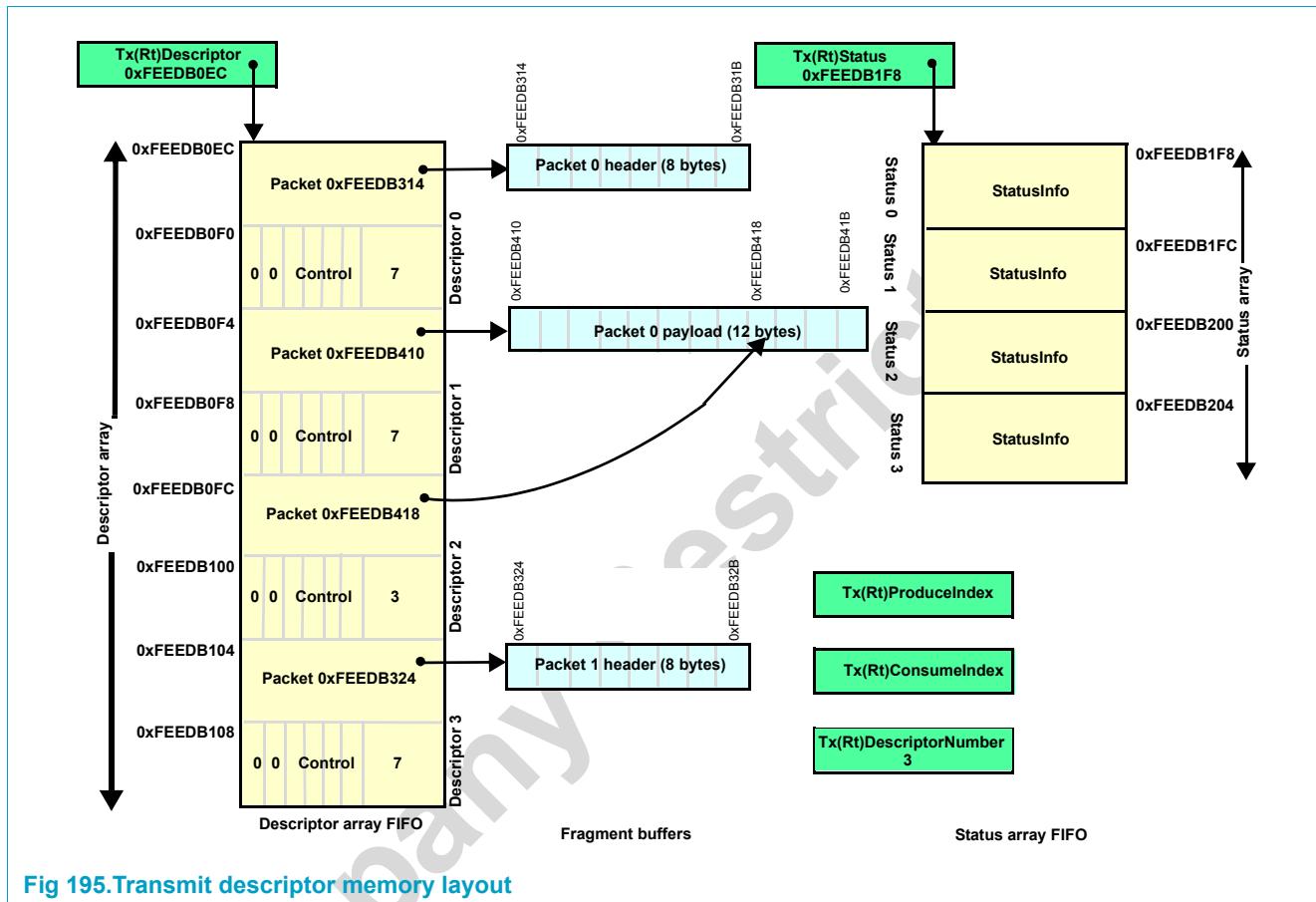


Fig 195. Transmit descriptor memory layout

For each of the two transmit channels the transmit descriptors are stored in an array in memory. The lowest address of the non real-time transmit descriptor array is stored in the **etn1_txdescriptor** register of the ETN1. Likewise for real-time transmissions the descriptor array base address is stored in the **etn1_txrtdescriptor** register. The number of descriptors in the array is stored in the **etn1_tx(rt)descriptornumber** register of the ETN1 using a minus one encoding style i.e. if the array has 8 elements the register value should be 7. Parallel to the descriptors there is an array of statuses. For each element of the descriptor array there is an associated status field in the status array. The base address of the status array is stored in the **etn1_tx(rt)status** register of the ETN1. During operation i.e. when the transmit data path is enabled the **etn1_tx(rt)descriptor**, **etn1_tx(rt)status** and **etn1_tx(rt)descriptornumber** registers should not be modified. The base address of the descriptor array as stored in the **etn1_tx(rt)descriptor** register should be aligned on a 8-byte address boundary. The status array base address as stored in the **etn1_tx(rt)status** register needs to be aligned on an 4-byte address boundary.

The register view of the ETN1 has two registers **etn1_txconsumeindex** and **etn1_txproduceindex** - both counters starting at 0 and wrapping at **etn1_txdescriptornumber** - which indicate the number of descriptors that have been

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

processed in the non real-time transmit channel. The **etn1txrtconsumeindex** and **etn1_txrtproduceindex** do the same for the real-time channel. The **etn1_tx(rt)produceindex** contains the index of the next descriptor that is going to be filled by the software driver. The **etn1_tx(rt)consumeindex** contains the index of the next descriptor going to be transmitted by the hardware. In case **etn1_tx(rt)produceindex == etn1_tx(rt)consumeindex** it means the transmit buffer is empty. In case **etn1_tx(rt)produceindex == etn1_tx(rt)consumeindex-1** it means the transmit buffer is full and the software driver cannot add new descriptors unless the hardware has transmitted some frames and freed up some descriptors.

Each transmit descriptor takes two word locations (8 bytes) in memory. Likewise each status field takes one words (4 bytes) in memory. Each transmit descriptor consists of a pointer to the data buffer containing transmit data (Packet) and a control word (Control). The Packet field has a zero address offset, whereas the control field has a 4 byte address offset, see [Table 619](#).

Table 619: Transmit descriptor fields

Address offset	Name	R/W	Size [1]	Description
0x0	Packet	R/W	W	Base address of the data buffer for containing transmit data
0x4	Control	R/W	W	Control information, see Table 620

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit

The data buffer pointer (Packet) is a word aligned address value containing the base address of the data buffer. The definition of the control word bits is listed in [Table 620](#).

Table 620: Transmit descriptor control word

Bit	Symbol	Description
31	Interrupt ^[1]	If true generate an Tx(Rt)Done interrupt when the data in this frame or frame fragment has been sent and the associated status information has been committed to memory.
30	Last ^[1]	If true indicates that this is the descriptor for the last fragment in the receive frame. If false the fragment from the next descriptor should be appended.
29	CRC	If true append a hardware CRC to the frame.
28	Pad	If true pad short frames to 64 bytes.
27	Huge	If true enables huge frame. When false prevents transmission of more than etn1_maxf , when true allows unlimited frame sizes.
26	Override	Per frame override. If true bits 30:27 will override the defaults from the PE-MAC internal registers. If false Control bits 30:27 will be ignored and the default values from PE-MAC will be used.
25:11	-	reserved
10:0	Size	Size in bytes of the data buffer. This is the size of the frame or fragment as it needs to be fetched by the DMA manager. In most cases it will be equal to the byte size of the data buffer pointed to by the Packet field of the descriptor. Size is -1 encoded e.g. a buffer of 8 bytes is encoded as the Size value 7.

[1] Interrupt-bit must be set together with the Last-bit. If it is not set, then the TxConsumeIndex will not be updated.

Table 621 shows the one field transmit status.

Table 621: Transmit descriptor fields

Address offset	Name	R/W	Size [1]	Description
0x0	StatusInfo	R	W	Transmit status return flags, see Table 622

[1] Size is either in byte (B) 8-bit, in half-word (H) 16-bit, or in word (W) 32-bit

The transmit status consists of one word which is the StatusInfo word. It contains flags returned by the PEMAC and flags generated by the transmit data path reflecting the status of the transmission. Table 622 lists the bit definitions in the StatusInfo word.

Table 622: Transmit Status information word

Bit	Symbol	Description
31	Error	An error occurred during transmission. This is a logic OR of Underrun, LateCollision, ExcessiveCollision and ExcessiveDefer.
30	NoDescriptor	The Tx stream is interrupted, because a descriptor is not available.
29	Underrun	A Tx underrun occurred due to the adapter not producing transmit data.
28	LateCollision	An Out of window Collision was seen, causing packet abort.
27	ExcessiveCollision	Indicates this packet exceeded the maximum collision limit and was aborted.
26	ExcessiveDefer	This packet incurred deferral beyond the maximum deferral limit and was aborted.
25	Defer	This packet incurred deferral, because the medium was occupied. This is not an error unless excessive deferral occurs.
24:21	CollisionCount	The number of collisions this packet incurred, up to the Retransmission Maximum.
20:0	reserved	unused

For multi fragment frames the value of the LateCollision/ExcessiveCollision/ExcessiveDefer/Defer/CollisionCount bits in all but the last fragment in the frame will be 0. The status of the last fragment in the frame will copy the value for these bits from PE-MAC. All fragment statuses will have valid Error/ NoDescriptor/Underrun bits.

11.5 Basic functionality

The Ethernet core can transmit and receive Ethernet packets from an off-chip Ethernet PHY interfacing through the RMII interface. Typically during system boot the device driver will initialize the Ethernet core and start up link auto-negotiate in the PHY via the MIIM interface. Software initialization of the Ethernet core should include initialization of the descriptor and status arrays as well as the receiver fragment buffers. To transmit a packet the software driver has to set up a descriptor to point to the packet data buffer before transferring the packet to hardware by incrementing the **etn1_txproduceindex** register. After transmission hardware will increment the TxConsumeIndex and hardware will optionally generate an interrupt. The hardware will receive packets from the PHY and apply filtering as configured by the software

driver. While receiving a packet the hardware will read a descriptor from memory to find the location of the associated receiver data buffer. Receive data is written in the data buffer and receive status is returned in the receive descriptor status word. Optionally an interrupt can be generated to notify software a packet has been received.

11.6 Functional Description

This section defines the functions of the ETN1 DMA capable 10/100 Ethernet MAC core. After introducing the DMA concepts of the ETN1 and a description of the basic transmit and receive functions this section elaborates on the advanced features like flow control, receive filtering etc.

11.6.1 MMIO interface

All MMIO write accesses to registers will be posted except for accesses to the **etn1_intset**, **etn1_intclear** and **etn1_intenable** registers. MMIO write operations will be executed in order.

In case the PowerDown bit of the **etn1_powerdown** register is set all MMIO read and write accesses will return a read or write error except for accesses to the **etn1_powerdown** register

11.6.2 Interrupts

An interrupt logic block raises and masks interrupts and keeps track of the cause of interrupts. The interrupt block sends a single interrupt request signal to the **INTC** block.

The SC interrupt service routine must read the **etn1_intstatus** register to locate the origin of the interrupt. All interrupt status can be set by software writing to the **etn1_intset** register and cleared by writing to the **etn1_intclear** register.

For more information on the source of an interrupt refer to [Section 11.6.5.8 “Transmit triggers interrupts”](#) and [Section 11.6.6.7 “Receive triggers interrupts”](#).

11.6.3 Direct Memory Accesses

11.6.3.1 Descriptor FIFOs

The ETN1 Ethernet module includes three high performance DMA managers for dual-Txqueue configurations.

The DMA managers make it possible to transfer frames directly to and from memory with little support from the processor and without the need to trigger an interrupt for each frame.

The DMA managers work with FIFOs of frame descriptors and statuses that are stored in host memory. The descriptors and statuses act as an interface between the Ethernet hardware module and the device driver software. There is one descriptor FIFO for receive frames and there are two descriptor FIFOs for transmit frames, one for real-time transmit traffic and one for non-real-time transmit traffic. By separating the areas in memory where the device driver and Ethernet module each carry out write operations, it's easy to maintain memory coherency and to make the descriptors

'cache safe', i.e. cache memory can be used to store descriptors. Using caching and buffering for frame descriptors, the memory traffic and memory bandwidth utilization of descriptors can be kept small.

Each frame descriptor contains two 32-bit fields: the first field is a pointer to a data buffer containing a frame or a fragment, whereas the second field is a control word related to that pointed frame or fragment. The software driver should write the base addresses of the descriptor and status arrays in the **etn1_txdescriptor/txrtdescriptor/rxdescriptor** and **txstatus/txrtstatus/rxstatus** registers of the ETN1. The number of descriptors/statuses in each array should be written in the **etn1_txdescriptornumber/txrtdescriptornumber/ rxdescriptornumber** registers of the ETN1. The number of descriptors in an array should correspond to the number of status elements in the associated status array.

Transmit descriptor arrays, receive descriptor arrays and transmit status arrays need to be aligned on a 4 byte address boundary, while receive status arrays need to be aligned on a 8 byte address boundary.

11.6.3.2 Ownership of descriptors

Both device driver software and Ethernet hardware can read and write the descriptor FIFOs simultaneously to produce and consume descriptors. A descriptor is either owned by the device driver or it is owned by the Ethernet hardware. Only the owner of a descriptor reads or writes its value. Typically, the sequence of use and ownership of descriptors/statuses is as follows: a descriptor is owned and set up by the device driver; ownership of the descriptor/status is passed by the device driver to the Ethernet module, which reads the descriptor and writes information to the status field; the Ethernet module passes ownership of the descriptor back to the device driver, which uses the status information and then recycles the descriptor to be used for another frame. Software must pre-allocate the arrays used to implement the FIFOs.

Software can hand over ownership of descriptors and statuses to the hardware by incrementing the **etn1_txproduceindex/txrtproduceindex/rxconsumeindex** registers. Hardware hands over descriptors and status to software by updating the **etn1_txconsumeindex/txrtconsumeindex/rxproduceindex** registers.

After handing over a descriptor to the receive and transmit DMA hardware device driver software should not modify the descriptor or reclaim the descriptor by decrementing the **etn1_txproduceindex/txrt-produceindex/rxconsumeindex** registers because descriptors may have been prefetched by the hardware. In this case the device driver software will have to wait until the frame has been transmitted or the device driver has to soft-reset the transmit and/or receive data paths which will also reset the descriptor FIFOs

11.6.3.3 Sequential order with wrap-around

Descriptors are read from and statuses are written to the arrays in sequential order with wrap-around. Sequential order means that when the Ethernet module has finished reading/writing a descriptor/status, the next descriptor/status is read/written at the next higher, adjacent memory address. Wrap around means that when the Ethernet module has finished reading/writing the last descriptor/status of the array (with the highest memory address), the next descriptor/status is read/written at the base address of the array.

11.6.3.4 Full and empty state of FIFOs

The descriptor FIFOs can be empty, partially full or full. A FIFO is empty when all descriptors are owned by the producer. A FIFO is partially full if both producer and consumer own part of the descriptors and both are busy processing those descriptors. A FIFO is full when all descriptors (except one) are owned by the consumer, so that the producer has no more room to process frames. Ownership of descriptors is indicated with the use of a consume index and a produce index. The produce index is the first element of the array owned by the producer. It is also the index of the array element that is next going to be used by the producer of frames (it may already be busy using it and subsequent elements). The consume index is the first element of the array that is owned by the consumer. It is also the number of the array element next to be consumed by the consumer of frames (it and subsequent elements may already be in the process of being consumed). If the consume index and the produce index are equal, the FIFO is empty and all array elements are owned by the producer. If the consume index equals the produce index plus one, then the array is full and all array elements (except the one at the produce index) are owned by the consumer. With a full FIFO, still one array element is kept empty, to be able to easily distinguish the full or empty state by looking at the value of the produce index and consume index. An array must have at least 2 elements to be able to indicate a full FIFO with a produce index of value 0 and a consume index of value 1. The wrap around of the arrays is taken into account when determining if a FIFO is full, so a produce index that indicates the last element in the array and a consume index that indicates the first element in the array, also means the fifo is full. When the produce index and the consume index are unequal and the consume index is not the produce index plus one (with wrap around taken into account), then the FIFO is partially full and both the consumer and producer own enough descriptors to be able to operate actively on the FIFO.

11.6.3.5 Frame fragments

For maximum flexibility in frame storage, frames can be split up into multiple frame fragments with fragments located in different places in memory. In this case one descriptor is used for each frame fragment. So, a descriptor can point to a single frame or to a fragment of a frame. By using fragments we can do scatter/gather DMA: transmit frames are gathered from multiple fragments in memory and receive frames can be scattered to multiple fragments in memory.

By stringing together fragments it is possible to create large frames from small memory areas. Another use of fragments is to be able to locate a frame header and frame payload in different places and to concatenate them without copy operations in the device driver.

For transmissions the Last bit in the descriptor **Control** field indicates if the fragment is the last in a frame; for receive frames the Last bit in the **StatusInfo** field of the status words indicates if the fragment is the last in the frame. If the Last bit is 0 the next descriptor belongs to the same Ethernet frame, If the Last bit is 1 the next descriptor is a new Ethernet frame.

11.6.4 Initialization

After reset the ETN1 software driver needs to initialize the ETN1 hardware. During initialization

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47

48
49
50
51
52
53
54

the software needs to:

- remove soft reset condition from PE-MAC
- configure the PHY via the MIIM interface of the MAC
- select RMII mode
- configure the transmit and receive DMA engines
- configure the host registers (**etn1_mac1**, **etn1_mac2** etc.) in the PE-MAC
- enable the receive and transmit data paths

Depending on the PHY the software needs to initialize registers in the PHY via the MII Management interface of the ETN1 (PHY2 connected to ETN2 needs to be configured via ETN1). The software can read and write PHY registers by programming the **etn1_mcfg**, **etn1_mcdrv**, **etn1_madr** registers of the MAC. Write data should be written to the **etn1_mwtd** register of the ETN1; read data and status information can be read from the **etn1_mrdd**, **etn1_mind** registers of the ETN1.

During initialization software need to select RMII mode by programming the **etn1_command** register. After initialization the RMII mode should not be modified.

Before switching to RMII mode the default soft reset (**etn1_mac1** register bit 15) has to be deasserted. The **ETNREFCLK** is necessary during this operation.

Transmit and receive DMA engines should be initialized by the device driver allocating the descriptor and status arrays in memory. Real-time transmit, non real-time transmit and receive have their own dedicated descriptor and status arrays. The base addresses of these arrays needs to be programmed in the **etn1_txdescriptor/txstatus**, **etn1_txrtdescriptor/txrtstatus** and **etn1_rxdescriptor/rxstatus** registers. The number of descriptors in an array should match the number of statuses in an array.

Please note the number of descriptors in the descriptor arrays needs to be written to the **etn1_txdescriptornumber/etn1txrtdescriptornumber/etn1_rxdescriptornumber** registers using a -1 encoding i.e. the value in the registers is the number of descriptors minus one e.g. if the descriptor array has 4 descriptors the value of the number of descriptors register should be 3.

After setting up the descriptor arrays frame buffers need to be allocated for the receive descriptors before enabling the receive data path. The Packet field of the receive descriptors needs to be filled with the base address of the frame buffer of that descriptor. Amongst others the Control field in the receive descriptor needs to contain the size of the data buffer using -1 encoding.

The receive data path has a configurable filtering function for discarding/ignoring specific Ethernet frames. The filtering function should also be configured during initialization.

After a power-on reset the soft reset bit in the PE-MAC will be asserted. Before enabling the ETN1 the soft reset condition has to be removed.

Enabling of the receive function is located in two places. The receive DMA manager needs to be enabled and the receive data path of PE-MAC needs to be enabled. To prevent overflow in the receive DMA engine the receive DMA engine should be

enabled by setting the RxEnable bit in the **etn1_command** register before enabling the receive data path in the PE-MAC core by setting the RECEIVE_ENABLE bit in the **etn1_mac1** register.

The non real-time and real-time transmit DMA engine can be enabled any time by setting the TxEnable and TxRtEnable bits in the **etn1_command** register.

Before enabling the data paths several options can be programmed in PE-MAC like automatic flow control, transmit to receive loop-back for verification, full/half duplex modes etc.

Base addresses of FIFOs and FIFO sizes cannot be modified without resetting the receive and transmit data paths.

11.6.5 Transmit process

11.6.5.1 Overview

This section outlines the transmission process. The ETN1 has two transmit data paths which can be configured as real-time/non real-time or low priority/high priority. For more information on (non) real-time and low/high priority transmission please refer to [Section 11.6.9](#). The following subsections the prefix TxRt refers to the real-time/low priority transmit data path and the prefix Tx refers to the non real-time/high priority transmit data path.

11.6.5.2 Device driver sets up descriptors and data

Before setting up one or more descriptors for transmission the device driver should select if the frame should go to the real-time or non real-time FIFO. Real-time traffic or low priority QoS traffic should go to the real-time Tx descriptor FIFO while non real-time or high priority QoS traffic should go to the nonreal-time Tx descriptor FIFO. If the selected descriptor FIFO is full the device driver should wait for the FIFO to become not full before writing the descriptor in the FIFO. If the selected FIFO is not full the device driver should use the **etn1_txproduceindex** descriptor from the array pointed to by **etn1_txdescriptor** (or the **2_txrtproduceindex** descriptor from the **etn1_txrtdescriptor** array for real-time/low priority QoS).

The Packet pointer in the descriptor is set to point to a data frame or frame fragment to be transmitted. The Size field in the **Command** field of the descriptor should be set to the number of bytes in the fragment buffer, -1 encoded. Additional control information can be indicated in the **Control** field in the descriptor (bits Interrupt, Last, CRC, Pad).

After writing the descriptor the descriptor needs to be handed over to the hardware by incrementing (and possibly wrapping) the **etn1_txproduceindex** or **etn1_txrtproduceindex** registers.

If transmit data path is disabled the device driver should not forget to enable the (non) real-time transmit data path by setting the TxEnable or TxRtEnable bit in the **etn1_command** register.

In case of a multi-fragment transmission only the last fragment must set the Last bit in the descriptor to 1. To trigger an interrupt when the frame has been transmitted and transmission status has been committed to memory, set the Interrupt bit in the

descriptor **Control** field to 1. To have the hardware add a CRC in the frame sequence control field of this Ethernet frame, set the CRC bit in the descriptor. This should be done if the CRC has not already been added by software. To enable automatic padding of small frames to the minimum required frame size, set the Pad bit in the Control field of the descriptor to 1. In typical applications bits CRC and Pad are both set to 1.

The device driver can set up interrupts using the **etn1_intenable** register to wait for a signal of completion from the hardware or can periodically inspect (poll) the progress of transmission. It can also add new frames at the end of the descriptor fifo, while hardware consumes descriptors at the start of the FIFO.

The device driver can stop the transmit process by resetting **etn1_command.TxEnable** and **etn1_command.TxRtEnable** to 0. The transmission will not stop immediately; frames already being transmitted will be transmitted completely and the status will be committed to memory before deactivating the data path. The status of the transmit data path can be monitored by the device driver reading the **etn1_txrtstatus/txstatus** bits in the **etn1_status** register.

As soon as the (non) real-time transmit data path is enabled and the corresponding Tx(Rt)ConsumeIndex and Tx(Rt)ProduceIndex are not equal (i.e. the hardware still needs to process frames from the descriptor FIFO) the Tx(Rt)Status bit in the **etn1_status** register will return to 1 (active).

11.6.5.3 Tx(Rt) DMA manager reads Tx(Rt) descriptor arrays

When the TxEnable bit (TxRtEnable bit for real-time traffic) is set the Tx DMA manager reads the descriptors from memory using DTL MMBD block transfers at the address determined by TxDescriptor and TxConsumeIndex, or, for real-time traffic, TxRtDescriptor and TxRtConsumeIndex. The block size of the DTL MMBD block transfer is determined by the total number of descriptors owned by the hardware Tx(Rt)ProduceIndex - Tx(Rt)ConsumeIndex. Block transferring descriptors maximizes prefetching options on the DTL adapter and minimizes memory load. Read data returned from the DTL MMBD descriptor read command is consumed per descriptor and only if needed.

While issuing the descriptor MMBD read commands the Tx(Rt) DMA manager also issues DTL MMBD commands for writing the transmission status without actually writing the status information.

11.6.5.4 Tx(Rt) DMA manager transmits data

After reading the descriptor the transmit DMA engine issues a DTL MMBD read command to read the associated frame data from memory and transmits the frame. After transfer completion the Tx DMA manager writes status information back to the StatusInfo and StatusHashCRC words of the status. The value of the Tx(Rt)ConsumeIndex is only updated after status information has been committed to memory which is checked by DTL interface. The Tx DMA manager continues to transmit frames until the descriptor FIFO is empty. If the transmit FIFO is empty the Tx(Rt)Status bit in the **etn1_status** register will return to 0 (inactive). If the descriptor FIFO is empty the Ethernet hardware will set the Tx(Rt)FinishedInt bit of the **etn1_intstatus** register. The transmit data path will still be enabled.

The Tx DMA manager inspect the Last bit of the descriptor **Control** field when loading the descriptor. If the Last bit is 0, this indicates that the frame consists of multiple fragments. The Tx DMA manager gathers all the fragments from the host memory visiting a string of frame descriptors, appends the fragments and sends them out as one Ethernet frame on the Ethernet connection. When the Tx DMA manager finds a descriptor with the Last bit in the Control field set to 1, this indicates the last fragment of the frame and thus the end of the Ethernet frame is found.

11.6.5.5 Update ConsumelIndex

Each time the Tx(Rt) DMA manager commits a status word to memory it completes the transmission of a descriptor and it increments the Tx(Rt)ConsumelIndex (taking wrap around into account) to hand the descriptor back to the device driver software. Software can re-use the descriptor for new transmissions after hardware has handed it back.

The device driver software can keep track of the progress of the DMA manager by reading the **etn1_tx(rt)consumeIndex** register to see how far along the transmit process is. When the Tx descriptor FIFOs get emptied completely, the TxConsumelIndex and TxRTConsumelIndex retain their last value.

11.6.5.6 Write transmission status

After the frame has been transmitted over the RMII bus and the status has been committed to memory by writing it across the DTL status write interface, the StatusInfo and StatusHashCRC words of the frame descriptor are updated by the DMA manager.

If the descriptor is for the last fragment of a frame (or for the whole frame if there are no fragments), then depending on the success or failure of the frame transmission, error flags (Error, LateCollision, ExcessiveCollision, Underrun, ExcessiveDefer, Defer) are set in the status. The CollisionCount field is set to the number of collisions the frame incurred, up to the Retransmission Maximum programmed in the **etn1_clrt** register of the Ethernet MAC Core.

Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx(Rt) DMA manager. Even if the descriptor is for a frame fragment other than the last fragment, the error flags are returned on the DTL status write interface. If the ETN1 detects a transmission error during transmission of a (multi-fragment) frame the rest of the transmit data all remaining fragments of the frame is still read on the transmit data DTL read interface. After an error the remaining transmit data is discarded by the ETN1. In case of errors during transmission of a multi fragment frame the error statuses will be repeated until the last fragment of the frame. Statuses for all but the last fragment in the frame will be written as soon as the data in the frame has been accepted by the Tx(Rt) DMA manager. The status for the last fragment in the frame will only be written after the transmission has completed on the Ethernet connection.

The status of the last frame transmission can also be inspected by reading the **etn1_tsv0** and **etn1_tsv1** registers. These registers do not report statuses on a fragment basis and do not store information of previously sent frames.

11.6.5.7 Transmission error handling

When an error occurs during the transmit process, the Tx(Rt) DMA manager will report the error via the transmission Status written in the Status FIFO and the **etn1_intstatus** interrupt status register.

The transmission can generate several types of errors: LateCollision, ExcessiveCollision, ExcessiveDefer, Underrun, NoDescriptor. All have corresponding bits in the transmission Status. On top of the separate bits in the Status LateCollision, ExcessiveCollision, ExcessiveDefer are ORed together into the Error bit of the Status. Errors are also propagated to the **etn1_intstatus** register; the Tx(Rt)Error bit in the **etn1_intstatus** register is set in case of a LateCollision, ExcessiveCollision, ExcessiveDefer, NoDescriptor error; Underrun errors are reported in the Tx(Rt)Underrun bit of the **etn1_intstatus** register.

Underrun errors can have three causes:

- the next fragment in a multi fragment transmission is not available. This is a non fatal error. A NoDescriptor status will be returned on the previous fragment and the **etn1_intstatus.Tx(Rt)Error** bit will be set.
- the transmission fragment data is not available on the DTL interface while the ETN1 already started sending the frame. This is a non fatal error. An Underrun status will be returned on transfer and **etn1_intstatus.Tx(Rt)Error** bit will be set.
- the flow of transmission statuses stalls and a new status has to be written while a previous status still waits to be transferred across the DTL MMBD transmission status write interface. This is a fatal error which can only be resolved by soft resetting the HW.

The first and second situations are non fatal and the device driver has to resend the frame or have upper SW layers resend the frame. In the third case the HW is in an undefined state and needs to be soft reset by setting the **etn1_command.TxReset** bit.

After reporting a LateCollision, ExcessiveCollision, ExcessiveDefer or Underrun error the transmission of the erroneous frame will be aborted, remaining transmission data and frame fragments will be discarded and transmission will continue with the next frame in the descriptor array.

11.6.5.8 Transmit triggers interrupts

The transmit data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set the Tx DMA will set the Tx(Rt)DoneInt bit in the **etn1_intstatus** register after sending the fragment and committing the associated transmission status to memory. Even if a descriptor (fragment) is not the last in a multi fragment frame the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor FIFO is empty while the Ethernet hardware is enabled the hardware will set the Tx(Rt)FinishedInt bit of the **etn1_intstatus** register.
- In case the DTL interface does not consume the transmission statuses at a sufficiently high bandwidth the transmission may underrun in which case the Tx(Rt)Underrun bit will be set in the **etn1_intstatus** register. This is a fatal error which requires a soft reset of the transmission queue.

- In case of a transmission error (LateCollision, ExcessiveCollision and ExcessiveDefer) or a multi fragment frame where the device driver did provide the initial fragments but did not provide the rest of the fragments (NoDescriptor) or in case of a non fatal overrun the hardware will set the Tx(Rt)ErrorInt bit of the **etn1_intstatus** register.

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the **etn1_intenable** register. Enabling or disabling does not affect the **etn1_intstatus** register contents, only the propagation of the interrupt status to the interrupt request signal (ETN1_intr).

11.6.6 Receive process

This section outlines the reception process including the activities in the device driver software.

11.6.6.1 Device driver sets up descriptors

After initializing the receive descriptor and status arrays as defined in [Section 11.6.4](#) to receive frames from the Ethernet connection the receive data path should be enabled in the **etn1_mac1** register and the Control register.

During initialization each Packet pointer in the descriptors is set to point to a data fragment buffer. The size of the buffer is stored in the Size bits of the Control field of the descriptor. Additionally the Control field in the descriptor has an Interrupt bit. The Interrupt bit allows generation of an interrupt after a fragment buffer has been filled and its status has been committed to memory.

After the initialization and enabling of the receive data path all descriptors are owned by the receive hardware and should not be modified by the software unless hardware hands over the descriptor by incrementing the RxProduceIndex indicating a frame has been received. The device driver is allowed to modify the descriptors after a reset of the receive data path.

11.6.6.2 Rx DMA manager reads Rx descriptor arrays

When the RxEnable bit in the **etn1_command** register is set the Rx DMA manager reads the descriptors from memory using DTL MMBD block transfers at the address determined by **etn1_rxdescriptor** and **etn1_rxproduceindex**. The ETN1 will start reading descriptors even before actual receive data arrives on the RMII interface (descriptor prefetching). The block size of the descriptor read DTL MMBD block transfer is determined by the total number of descriptors owned by the hardware **etn1_rxconsumeindex** - **etn1_rxproduceindex** - 1. Block transferring of descriptors maximizes prefetching options on the DTL adapter and minimizes memory load. Read data returned from the DTL MMBD descriptor read command is consumed per descriptor and only if needed.

11.6.6.3 Rx DMA manager receives data

After reading the descriptor the receive DMA engine waits for the PE-MAC to return receive data from the RMII interface that passes the receive filter. Receive frames that do not match the filtering criteria are not passed to memory. For more information on filtering refer to [Section 11.6.12](#). Once a frame passes the receive filter the data is written in the descriptors fragment buffer via the DTL MMBD data interface. The

RxDMA does not write beyond the size of the buffer. In case of a receive frame is
1
larger than a descriptor's fragment buffer the frame will be written to multiple
2
fragment buffers of consecutive descriptors. In case of a multi fragment reception all
3
but the last fragment in the frame will return a status where the Last bit is set to 0.
4
Only the last fragment of a frame will have set the Last bit in the status. If a fragment
5
buffer is the last of a frame the buffer may not be filled completely. The first receive
6
data of the next frame will be written to the next descriptor's fragment buffer.
7

After receiving a fragment the Rx DMA manager writes status information back to the
8
StatusInfo and StatusHashCRC words of the status. The ETN1 writes the fill level of a
9
descriptor's fragment buffer in the EntryLevel field of the Status word. The value of
10
the RxProduceIndex is only updated after the fragment data and the fragment status
11
information has been committed to memory. The Rx DMA manager continues to
12
receive frames until the descriptor FIFO is full. If the descriptor FIFO is full the
13
Ethernet hardware will set the RxFinishedInt bit of the **etn1_intstatus** register. The
14
receive data path will still be enabled. If the receive FIFO is full any new receive data
15
will generate an overflow error and interrupt.
16

11.6.6.4 Update ProduceIndex

Each time the Rx DMA manager commits a fragment data and the associated status
20
word to memory it completes the reception of a descriptor and it increments the
21
etn1_rxproduceindex (taking wrap around into account) to hand the descriptor back
22
to the device driver software. Software can re-use the descriptor for new receptions
23
by handing it back to hardware when the receive data has been processed.
24

The device driver software can keep track of the progress of the DMA manager by
25
reading the **etn1_rxproduceindex** register to see how far along the receive process
26
is. When the Rx descriptor FIFO get emptied completely, the **etn1_rxproduceindex**
27
retains its last value.
28

11.6.6.5 Write reception status

After the frame has been received from the RMII bus the StatusInfo and
32
StatusHashCRC words of the frame descriptor are updated by the DMA manager.
33

If the descriptor is for the last fragment of a frame (or for the whole frame if there are
35
no fragments), then depending on the success or failure of the frame reception, error
36
flags (Error, NoDescriptor, Overrun, AlignmentError, RangeError, LengthError,
37
SymbolError, CRCError) are set in the status. The EntryLevel field is set to the
38
number of bytes actually written to the fragment buffer, -1 encoded. For fragments not
39
being the last in the frame the EntryLevel will match the size of the buffer. The hash
40
CRCs of the destination and source addresses of a packet are calculated once for all
41
the fragments belonging to the same packet and then stored in every
42
StatusHashCRC word of the statuses associated with the corresponding fragments. If
43
the reception reports an error any remaining data in the receive frame is discarded
44
and the Last bit will be set in the receive status field so the error flags in all but the last
45
fragment of a frame will always be 0.
46

The status of the last receive frame can also be inspected by reading the **etn1_rsv**
47
register. The register does not report statuses on a fragment basis and does not store
48
information of previously received frames.
49

11.6.6.6 Reception error handling

When an error occurs during the receive process, the Rx DMA manager will report the error via the receive Status written in the Status FIFO and the **etn1_intstatus** interrupt status register.

The receive process can generate several types of errors: AlignmentError, RangeError, LengthError, SymbolError, CRCError, Overrun, NoDescriptor. All have corresponding bits in the receive Status. On top of the separate bits in the Status AlignmentError, RangeError, LengthError, SymbolError, CRCError are ORed together into the Error bit of the Status. Errors are also propagated to the **etn1_intstatus** register; the RxError bit in the **etn1_intstatus** register is set in case of a AlignmentError, RangeError, LengthError, SymbolError, CRCError, NoDescriptor error; non fatal overrun errors are reported in the RxError bit of the **etn1_intstatus** register; fatal Overrun errors are report in the RxOverrun bit of the **etn1_intstatus** register. On fatal overrun errors the Rx data path needs to be soft rest by setting the **etn1_command.RxReset** bit.

Overrun errors can have three causes:

- in case of a multi-fragment reception the next descriptor may be missing. In this case the NoDescriptor field is set in the status word of the previous descriptor and the RxError in the **etn1_intstatus** register is set. This error is non fatal.
- the data flow on the receiver data interface stalls corrupting the packet. In this case the overrun bit in the status word is set and the RxError bit in the **etn1_intstatus** register is set. This error is non fatal.
- the flow of reception statuses stalls and a new status has to be written while a previous status still waits to be transferred across the DTL MMBD receive status write interface. This error will corrupt the HW state and requires the HW to be soft reset. The error is detected and sets the Overrun bit in the **etn1_intstatus** register.

The first overrun situation will result in an incomplete frame with a NoDescriptor status and the **etn1_intstatus.RxError** bit set. SW should discard the partially received frame. In the second overrun situation the frame data will be corrupt which results in the Overrun status bit being set in the Status word while the IntError interrupt bit is set. In the third case receive errors cannot be reported in the receiver Status arrays which corrupts the HW state; the errors will still be reported in the **etn1_intstatus** register's Overrun bit and the **etn1_command.RxReset** bit should be used to soft reset the HW.

11.6.6.7 Receive triggers interrupts

The receive data path can generate four different interrupt types:

- If the Interrupt bit in the descriptor Control field is set the Rx DMA will set the RxDoneInt bit in the **etn1_intstatus** register after receiving a fragment and committing the associated data and status to memory. Even if a descriptor (fragment) is not the last in a multi fragment frame the Interrupt bit in the descriptor can be used to generate an interrupt.
- If the descriptor FIFO is full while the Ethernet hardware is enabled the hardware will set the RxFinishedInt bit of the **etn1_intstatus** register.

- In case the DTL interface does not consume receive statuses at a sufficiently high bandwidth the receive status process may overrun in which case the RxOverrun bit will be set in the **etn1_intstatus** register. 1
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 2
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 3
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 4
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 5
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 6
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 7
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 8
- In case of a receive error (AlignmentError, RangeError, LengthError, SymbolError, CRCError) or a multi fragment frame where the device driver did provide descriptors for the initial fragments but did not provide the descriptors for the rest of the fragments or if a non fatal data Overrun occurred the hardware will set the RxErrorInt bit of the **etn1_intstatus** register. 9

All of the above interrupts can be enabled and disabled by setting or resetting the corresponding bits in the **etn1_intenable** register. Enabling or disabling does not affect the **etn1_intstatus** register contents, only the propagation of the interrupt status to the interrupt request signal. 10

11.6.6.8 Device driver processes receive data

As a response to status (e.g. RxDoneInt) interrupts or polling of the **etn1_rxproduceindex** the device driver can read the descriptors that have been handed over to it by the hardware (**etn1_rxproduceindex - etn1_rxconsumeindex**). The device driver should inspect the status words in the status FIFO to check for multi fragment receptions and receive errors. 11

The device driver can forward receive data and status to upper software layers. After processing of data and status the descriptors, statuses and data buffers may be recycled and handed back to hardware by incrementing the **etn1_rxconsumeindex**. 12

11.6.7 Transmission Retry

If a collision on the Ethernet occurs, it usually takes place during the collision window spanning the first 64 bytes of a frame. If collision is detected the ETN1 will retry the transmission. For this purpose the first 64 bytes of a frame are buffered by the ETN1, so that this data can be used during the retry. A transmission retry within the first 64 bytes in a frame is fully transparent to the application and device driver software. 13

When a collision occurs outside of the 64 bytes collision window, a LateCollision error is triggered and the transmission is aborted. After a LateCollision error the remaining data in the transmit frame will be discarded. The ETN1 will set the Error and LateCollision bits in the frame's status fields. The Tx(Rt)Error bit in the **etn1_intstatus** register will be set. If the corresponding bit in the **etn1_intenable** register is set the Tx(Rt)Error bit in the **etn1_intstatus** register will trigger an interrupt. 14

The RETRANSMISSION_MAXIMUM field of the **etn1_clrt** register can be used to configure the maximum number of retries before aborting the transmission. 15

11.6.8 Status hash CRC calculations

For each received frame, the ETN1 is able to detect the destination address and source address and from them calculate the corresponding hash CRCs. 16

The destination address and source address hash CRCs being written in the StatusHashCRC word are the nine most significant bits of the 32-bit CRCs as calculated by the CRC calculator. 17

11.6.9 Transmission modes

11.6.9.1 Overview

The ETN1 hardware has two transmission data paths: Tx and TxRt. These transmission data paths can be switched in two modes:

- Quality of Service mode: In this mode the TxRt transmission data path handles low priority transmission, the Tx data path handles high priority transmissions.
- normal (non-QoS) mode: In this mode the TxRt transmission data path is unused for transmission, while the Tx data path handles all transmissions.

Each transmit data-path has it's own associated descriptor and status array in memory and it's own DMA manager for transferring data to and from memory. An ETN1 internal arbiter decided which of the transmit DMA managers should be allowed to transmit a frame, based on the mode of operation and the priority.

11.6.9.2 Normal (non-QoS) mode

In normal mode the Tx transmit data path transmits all the frames available, while the TxRt data path remains idle (**etn1_command.TxRtEnable** is set to 0).

11.6.9.3 Quality of Service transmission mode

The two transmit descriptor/status arrays and data paths can be used to implement a generic quality of service (QoS) mechanism for transmit frames.

The QoS mechanism distinguishes two priority queues: a queue with low priority and a queue with high priority. The Tx descriptor fifo (using TxDescriptor and TxStatus) and Tx DMA manager implement the high priority queue and the TxRt descriptor FIFO (using TxRtDescriptor and TxRtStatus) and TxRt DMA manager implement the low-priority queue.

To implement QoS software should enter transmit frames that have a high quality of service requirement into the Tx (high priority) descriptor array, other frames should be entered into the TxRt (low priority) descriptor array. If there are any frames in the high priority queue, they are sent out first before any frames that are waiting in the low priority queue. Frames in the low priority queue are only sent if the high priority queue is empty.

To prevent starvation of frames in the low priority queue, **etn1_qostimeout** register defines the maximum number of transmit clock cycles that low priority frame waits at the head of the low-priority queue. An internal time-out counter starts counting transmit clock cycles, starting from 0 as soon as a low priority frame reaches the transmission arbiter. If the low-priority frame is still waiting to be transmitted when the counter reaches the **etn1_qostimeout** register's value then the arbiter will promote the priority of only that low-priority frame over the high priority queue. After sending the low-priority frame the low-priority queue will be set back to the low priority. The waiting time of a frame in the low-priority queue can be larger than the **etn1_qostimeout** register's value, if it has to wait to reach the head of the queue.

11.6.10 IEEE 802.3/clause 31 flow control

11.6.10.1 Overview

For full duplex connections the ETN1 supports IEEE 802.3/clause 31 flow control using pause frames. This type of flow control may be used in full-duplex point-to-point connections. Flow control allows a receiver to stall a transmitter e.g. when the receive buffers are (almost) full. For this purpose the receiving side sends a pause frame to the transmitting side.

11.6.10.2 Receive flow control

In full-duplex mode the ETN1 will suspend its transmissions when the ETN1 receives a pause frame. Rx flow control initiated by the receiving side of the ETN1 transmission. It is enabled using **etn1_mac1** configuration register by setting the RX_FLOW_CONTROL bit to 1. If the RX_FLOW_CONTROL bit is zero, then the ETN1 ignores received pause control frames. When a pause frame is received on the Rx side of the ETN1, transmission on the Tx side will be interrupted after the currently transmitting frame has completed for an amount of time as indicated in the received pause frame. The transmit data path of ETN1 will stop transmitting data for the number of 512-bit slot times encoded in the pause-timer field of the received pause control frame.

By default the received pause control frames are not forwarded to the device driver. To forward the receive flow control frames to the device driver set the PASS_ALL RECEIVE_FRAMES bit in the **etn1_mac1** configuration register.

11.6.10.3 Tx flow control

In case device drivers need to stall the receive data e.g. because software buffers are full, the ETN1 can transmit pause control frames. Tx flow control needs to be initiated by the device driver software; there is no IEEE 802.3/31 flow control initiated by the hardware, such as the DMA managers.

With software flow control the device driver can detect a situation in which the process of receiving frames needs to be interrupted by sending Tx pause frames out. Note that due to Ethernet delays still a few frames can be received until the flow control takes effect and the receive stream stops.

If the MAC is operating in full-duplex mode, then setting the TxFlowControl bit of the **etn1_command** register will start a pause frame transmission. The value inserted into the pause-timer value field of transmitted pause frames is programmed via the PauseTimer[15:0] bits in the **etn1_flowcontrolcounter** register (see [Table 601](#)). When the TxFlowControl bit is deasserted, another pause frame having a pause-timer value of 0x0000 is automatically sent to abort flow control and resume transmission.

When the flow control has to last a long time, a sequence of pause frames must be transmitted. This is supported with a mirror counter mechanism. To enable mirror counting, write a value unequal to 0 to the **etn1_flowcontrolcounter.MirrorCounter** bits in the **etn1_flowcontrolcounter** register. When the TxFlowControl bit is asserted, a pause frame is transmitted. After sending the pause frame an internal mirror counter is initialized at zero. The internal mirror counter starts incrementing one

every 512 bit-slot times. When the internal mirror counter reaches the MirrorCounter value another pause frame is transmitted with pause-timer value equal to the PauseTimer field from the **etn1_flowcontrolcounter** register and the internal mirror counter is reset to zero and restarts count. The register

etn1_flowcontrolcounter.MirrorCounter is usually set to a smaller value than register PauseTimer[15:0] to ensure an early expiration of the mirror counter to be able to send a new pause frame before the transmission on the other side can resumes. By keep sending pause frames before the transmitting side finishes counting the pause timer, the pause can be extended as long as TxFlowControl is asserted. To disable the mirror counter mechanism, write the value 0 to MirrorCounter field in the **etn1_flowcontrolcounter** register. When using the mirror counter mechanism to account for time-of-flight delays, frame transmission time, queuing delays, crystal frequency tolerances and response time delays the MirrorCounter should be programmed conservatively typically about 80% of the PauseTimer value.

In case the software device driver sets the MirrorCounter field of the **etn1_flowcontrolcounter** register to zero the ETN1 will only send one pause control frame. After sending the pause frame an internal pause counter is initialized at zero; the internal pause counter is incremented by one every 512 bit-slot times. Once the internal pause counter reaches the value of the **etn1_flowcontrolcounter.PauseTimer** register the ETN1 will reset the TxFlowControl bit in the **etn1_command** register. The software device driver can poll the TxFlow- Control bit to detect when the pause completes.

The value of the internal counter in the flow control module can be read out via the **etn1_flowcontrolstatus** register. In case the MirrorCounter is non zero the **etn1_flowcontrolstatus** register will return the value of the internal mirror counter; if the MirrorCounter is zero the **etn1_flowcontrolstatus** register will return the value of the internal pause counter value.

The device driver is allowed to dynamically modify the MirrorCounter register value and switch between zero MirrorCounter and non zero MirrorCounter modes.

Transmit flow control is enabled via the TX_FLOW_CONTROL bit in the **etn1_mac1** configuration register. If the TX_FLOW_CONTROL bit is zero, then the MAC will not transmit pause control frames, and software must not initiate pause frame transmissions and the TxFlowControl bit in the **etn1_command** register should be zero.

11.6.11 Half-duplex mode back pressure

When in half-duplex mode the ETN1 can generate back pressure to stall receive packets by sending continuous preamble that basically jams any other transmissions on the Ethernet medium. When the Ethernet module operates in half duplex mode, asserting the TxFlowControl bit in the **etn1_command** register will result in applying continuous preamble on the Ethernet wire until TxFlow-Control is deasserted.

If the medium is idle, the ETN1 begins transmitting out preamble, which raises carrier sense causing all other stations to defer. In the event the transmitting of preamble causes a collision, the back pressure “rides through” the collision. The colliding station backs off and then defers to the back pressure. If during back pressure, the user wishes to send a frame, the back pressure is interrupted, the frame sent and

then the back pressure resumed. If TxFlowControl is asserted for longer than 3.3 ms in 10 Mbps mode or 0.33 ms in 100 Mbps mode, back pressure will cease sending preamble for several byte times to avoid the jabber limit.

11.6.12 Receive filtering

11.6.12.1 Overview

The Ethernet module has the capability to filter out receive frames analyzing the Ethernet destination address in the frame. This capability greatly reduces the load on the system controller, because the many Ethernet frames that are addressed to other stations, would otherwise need to be inspected and rejected by the device driver software. Address filtering can be implemented using the perfect address filter or the (imperfect) hash filter. The latter produces a 6 bits hash code which can be used as an index into a 64 entry programmable hash table. [Figure 196](#) depicts a functional view on the receive filter.

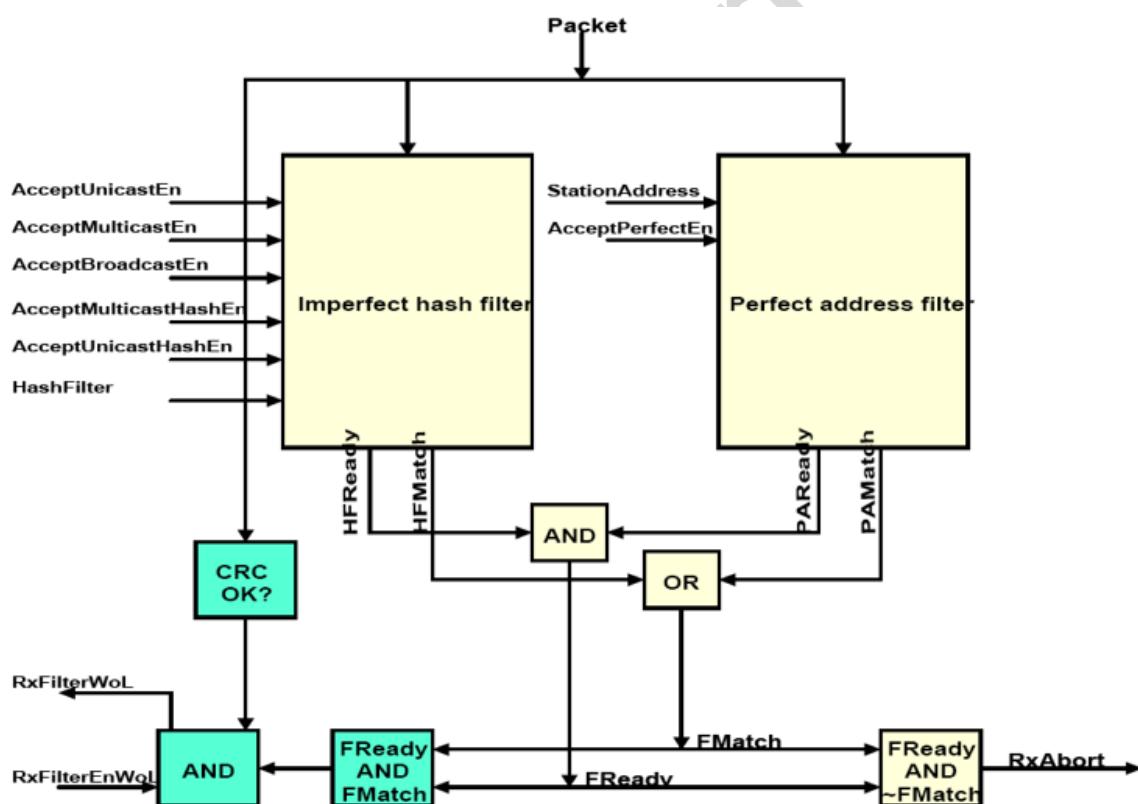


Fig 196. Receive filter block diagram

On the top of the diagram the Ethernet receive frame enters the filters. Each filter is controlled by signals from the software view; each filter produces a “Ready” output and a “Match” output. If “Ready” is 0 then the Match value is “don’t care”; if a filter finishes filtering then it will assert its Ready output; if the filter finds a matching frame

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

it will assert the Match output along with the Ready output. The results of the filters are combined by logic functions into a single RxAbort output. If the RxAbort output is asserted a frame need not be received.

The blocks in the left-hand side of the diagram are used for WoL and are discussed in [Section 11.6.15](#).

In order to reduce the memory traffic the receive data path of the Ethernet MAC has a FIFO buffer for 68 bytes. The Ethernet MAC will only start writing a frame to memory after 68 byte delays. If the RxAbort signal is asserted during the initial 68 bytes of the frame, the frame can be discarded and removed from the buffer and will not be stored to memory at all. If the RxAbort signal is asserted after the initial 68 bytes in a frame, part of the frame is already written to memory and the Ethernet MAC will stop writing further data in the frame to memory; the FailFilter bit in the status word of the frame will be set to indicate the software device driver can discard the frame immediately.

11.6.12.2 Unicast, broadcast and multicast

Generic filtering based on the type of frame (unicast, multicast or broadcast) can be programmed using the AcceptUnicastEn, AcceptMulticastEn and AcceptBroadcastEn bits from the **etn1_rxfilterctrl** register. Setting the AcceptUnicast, AcceptMulticast and AcceptBroadcast bits causes all frames of types unicast, multicast and broadcast, respectively, to be accepted, ignoring the Ethernet destination address in the frame. To program promiscuous mode, i.e. to accept all frames, set all 3 bits to 1.

11.6.12.3 Perfect address match

When a frame with a unicast destination address is received, a perfect filter compares the destination address with the 6-byte station address (programmed in the station address registers **etn1_sa0**, **sa1**, **sa2**.) If the AcceptPerfectEn bit in the **etn1_rxfilterctrl** register is set to 1, and the address matches, the frame is accepted.

11.6.12.4 Imperfect hash filtering

An imperfect filter is available, based on a hash mechanism. This filter applies a hash function to the destination address and uses the hash to access a table that indicates if the frame should be accepted. The advantage of this type of filter is that a small table can cover any possible address. The disadvantage is that the filtering is imperfect, i.e. sometimes frames are accepted that should have been discarded.

The standard Ethernet cyclic redundancy check (CRC) function is calculated from the 6-byte destination address in the Ethernet frame (this CRC is calculated anyway as part of calculating the CRC of the whole frame), then bits [28:23] out of the 32 bits CRC result are taken to form the hash. The 6-bit hash is used to access the hash table: it is used as an index in the 64-bits HashFilter register that has been programmed with accept values. If the selected accept value is 1, the frame is accepted.

The device driver can initialize the hash filter table by writing to the registers **etn1_hashfilterl** and **etn1_hashfilterh**. **etn1_hashfilterl** contains bits 0 through 31 of the table and **etn1_hashfilterh** contains bit 32 through 63 of the table. So, hash value 0 corresponds to bit 0 of the **etn1_hashfilterl** register and hash value 63 corresponds to bit 31 of the **etn1_hashfilterh** register.

The imperfect hash filter can be applied to multicast addresses, by setting the AcceptMulticastHashEn bit in the RxFilter register to 1.

The same imperfect hash filter that is available for multicast addresses can also be used for unicast addresses. This is useful to be able to respond to a multitude of unicast addresses without enabling all unicast addresses. The hash filter can be applied to unicast addresses, by setting the AcceptUnicast_HashEn bit in the **etn1_rxfilterctrl** register to 1.

11.6.12.5 Enabling and disabling filtering

The filters as defined in the sections above can be bypassed by setting the PassRxFilter bit in the **etn1_command** register. When the PassRxFilter bit is set all receive frames will be passed to memory. In this case the device driver software has to implement all filtering functionality in software.

Setting the PassRxFilter bit does not affect the runt frame filtering as defined in the next section.

11.6.12.6 Runt frames

A frame with less than 64 bytes (or 68 bytes for VLAN frames) is shorter than the minimum Ethernet frame size and therefore considered erroneous; they might be collision fragments. The receive data path automatically filters and discards these runt frames without writing them to memory and using a receive descriptor.

When a runt frame has a correct CRC there is a possibility that it is intended to be useful. The device driver can receive the runt frames with correct CRC by setting the PassRuntFrame bit of the **etn1_command** register to 1.

11.6.13 Transmission padding and CRC

In case a frame of less than 60 bytes or 64 bytes for VLAN frames the ETN1 can pad the frame to 64 or 68 bytes including a 4 bytes CRC Frame Check Sequence (FCS). Padding is affected by the value of the AUTO_DETECT_PAD_ENABLE, VLAN_PAD_ENABLE and PAD/CRC_ENABLE bits of the **etn1_mac2** configuration register as well as the Override and Pad bits from the transmit descriptor control word.

The effective pad enable (EPADEN) is equal to the PAD/CRC ENABLE bit from the **etn1_mac2** register if the Override bit in the descriptor is 0. If the Override bit is 1 then EPADEN will be taken from the descriptor Pad bit. Likewise the effective CRC enable (ECRCE) equals CRC_ENABLE if the Override bit is 0 otherwise it equal the CRC bit from the descriptor.

If padding is required and enabled a CRC will always be appended to the padded frames. A CRC will only be appended to the non padded frames if ECRCE is set. If EPADEN is 0 the frame will not be padded and no CRC will be added unless ECRCE is set.

If EPADEN is 1 then small frames will be padded and a CRC will always be added to the padded frames. In this case if AUTO_DETECT_PAD_ENABLE and VLAN_PAD_ENABLE are both 0, then the frames will be padded to 60 bytes and a CRC will be added creating 64 bytes frames; if AUTO_DETECT_PAD_ENABLE is 1

while VLAN_PAD_ENABLE is 0, VLAN frames will be padded to 64 bytes, non VLAN frames will be padded to 60 bytes and a CRC will be added to padded frames creating 64 or 68 bytes padded frames.

Table 623:Pad operation

AUTO_DETECT T_PAD_ENAB LE	VLAN_PAD_E NABLE	PAD/CRC_EN ABLE (EPADEN)	Operation
x	x	0	Frames will not be padded
0	0	1	Short frames will be padded to 60bytes, CRC appended
x	1	1	Short frames will be padded to 64bytes, CRC appended
1	0	1	Automatic padding according to the protocol ID

[1] "x" mean that the bit is ignored

11.6.14 Huge frames and frame length checking

The HUGE_FRAME_ENABLE bit in the **etn1_mac2** configuration register can be set to 1 to enable transmission and reception of frames of any length. Huge frame transmission can be enabled on a per frame basis by setting the Override and Huge bits in the transmit descriptor **Control** word.

When enabling huge frames the ETN1 will not check frame lengths and report frame length errors (RangeError and LengthError). If huge frames are enabled the received byte count in the **etn1_rsv** register may be invalid because the frame may exceed the maximum size; the EntryLevel fields from the receive status arrays will be valid.

The ETN1 will check frame lengths by comparing the length/type field of the frame to the actual number of bytes in the frame and report a LengthError by setting the corresponding bit in the receive **StatusInfo** word.

The **etn1_maxf** register in the ETN1 allows the device driver to specify the maximum number of bytes in a frame. The ETN1 will compare the actual receive frame to the **etn1_maxf** value and report a RangeError in the receive **StatusInfo** word if the frame is larger.

11.6.15 Wake-up on LAN

11.6.15.1 Overview

The Ethernet module supports power management with remote wake-up over LAN. The host system of the Ethernet module can be powered down, even including part of the ETN1 Ethernet module itself, while the Ethernet module continues to listen to packets on the LAN. Appropriately formed packets can be received and recognized by the Ethernet module and used to trigger the host system to wake up from its power-down state.

Wake-up of the system takes effect through an interrupt. When a wake-up event is detected, the WakeupInt bit in the **etn1_intstatus** register is set. The interrupt status will trigger an interrupt if the corresponding WakeupIntEn bit in the **etn1_intenable** register is set.

While in a power-down state the packet that generates a Wake-up on LAN event is lost.

There are two ways in which Ethernet packets can trigger wake-up events: generic Wake-up on LAN and Magic Packet. The generic Wake-up on LAN functionality is based on the filtering as defined in [Section 11.6.12](#); magic packet filtering uses an additional filter for magic packet detection. In both cases a WoL event is only triggered if the triggering packet has a valid CRC. [Figure 196](#) illustrates the wake-up functionality in a block diagram.

The **etn1_rxfilterwolstatus** register can be read by the software to inspect the reason for a Wake-up event. Before going to power-down the power management software should clear the register by writing the **etn1_rxfilterwolclear** register.

NOTE: when entering in power-down mode by switching off the clocks, a receive frame might be not entirely stored into the Rx buffer. In this situation, after turning on the clocks again, the next receive frame is corrupted due to the data of the previous frame being added in front of the last received frame. At the moment, the software drivers have to reset the receive data path just after turning on the gated clocks again. Moreover, another solution is using the power down bit to reset the receive buffer pointers after re-asserting the gated clocks.

The following subsections describe the two WoL mechanisms.

11.6.15.2 Filtering for WoL

The receive filter functionality as defined in [Section 11.6.12](#) can be used to generate WoL events. If the RxFilterEnWoL bit of the **etn1_rxfilterctrl** register is set the receive filter will set the WakeupInt bit of the **etn1_intstatus** register if a frame is received that passes the filter. The interrupt will only be generated if the CRC of the frame is correct.

In case of a wake-up generated by the receive filter the RxFilterWoL bit in the **etn1_rxfilterwolstatus** register will be set along with the origin of the filter match which will be set in the AcceptPerfectWoL, Accept-MulticastHashWoL, AcceptUnicastHashWoL, AcceptMulticastWoL, AcceptBroadcastWoL, AcceptUnicastWoL bits of the same register. Software can reset these bits writing a 1 to the corresponding bits of the **etn1_rxfilterwolclear** register.

11.6.15.3 Magic Packet WoL

The Ethernet module supports wake-up using AMD's Magic Packet technology (see "Magic Packet technology", Advanced Micro Devices). A Magic Packet is a specially formed packet solely intended for wake-up purposes. This packet can be received, analyzed and recognized by the Ethernet module and used to trigger a wake-up event.

A Magic Packet is a packet that contains in its data portion the station address repeated 16 times with no breaks or interruptions, preceded by 6 Magic Packet synchronization bytes with the value 0xFF. Other data may be surrounding the Magic Packet pattern in the data portion of the packet. The whole packet must be a well-formed Ethernet frame.

The magic packet detection unit analyzes the Ethernet packets, extracts the packet address and checks the payload for the Magic Packet pattern. The address from the packet is used for matching the pattern (not the address in the **etn1_sa0/1/2** registers.) A magic packet only sets the wake-up interrupt status bit if the packet passes the receive filter as illustrated in [Figure 196](#): the result of the receive filter is ANDed with the magic packet filter result to produce the result.

Magic Packet filtering is enabled by setting the MagicPacketEnWoL bit of the **etn1_rxfilterctrl** register. Note that when doing Magic Packet WoL the RxFilterEnWoL bit in the **etn1_rxfilterctrl** register should be 0. Setting the RxFilterEnWoL bit to 1 would accept all packets for a matching address, not just the Magic Packets i.e. WoL using Magic Packets is more strict.

When a magic packet is detected, apart from the WakeupInt bit in the **etn1_intstatus** register, the MagicPacketWoL bit is set in the **etn1_rxfilterwolstatus** register. Software can reset the bit writing a 1 to the corresponding bit of the **etn1_rxfilterwolclear** register.

11.6.16 Packet Engines Core status vectors

The transmit status vector and the receive status vector as delivered by the Ethernet MAC core are available in registers **etn1_tsv0**, **tsv1** and **etn1_rsv**. These registers are normally of limited use, because the communication between driver software and Ethernet module takes place primarily through the frame descriptors. However, for debug purposes these transmit and receive status vectors are made visible. The values in these registers are simple copies of the transmit and receive status vectors as produced by the ETN1 block. They are valid as long as the status vectors of the ETN1 core are valid and should typically only be read when the transmit and receive processes are halted.

The **etn1_tsv0**, **tsv1** and **etn1_rsv** registers are defined in [Section 11.4.2](#).

11.7 Power Management

The Ethernet core does support power management by means of clock switching. Basically all DTL clocks in the Ethernet core can be switched off.

The Ethernet MAC supports two power management modes:

- sleep mode: the SoC is active while most clocks in the Ethernet MAC are switched off except the MMIO clock; the CPU is active
- off mode: all clocks of the Ethernet MAC are switched off

In sleep mode a WoL event can wake-up the Ethernet core, but it will not wakeup the SC controller (see [Table 325](#)).

11.7.1 Sleep mode

The Ethernet core can be put in sleep mode if the receive and transmit DMA managers are disabled and inactive. By setting the PowerDown bit in the **etn1_powerdown** register software should prevent accesses to clock domains that have been switched off. If an external PHY is connected a WoL can trigger an interrupt. Clocks should only be disabled after software has set the PowerDown bit

The Ethernet MAC should also be put into sleep mode by setting the PowerDown bit if no external PHY is connected.

To enter sleep mode software should:

- disable both transmit DMA managers and the receive DMA manager by writing the **etn1_command** register
- wait for the transmit and receive data paths to be inactive by waiting for a Finished interrupt or by polling the Status register.
- set the PowerDown bit by writing to the **etn1_powerdown** register
- disable one or more clocks except for the MMIO clock

If no PHY is connected software can directly set the PowerDown bit and disable the clocks.

To exit sleep mode software should:

- enable the clocks
- reset the PowerDown bit
- reenable the receive and transmit data paths

11.7.2 Off mode

In Off mode all clocks in the Ethernet core are switched off and WoL will not work.

The Off mode is entered under software control. The software should first enter sleep mode and then disable the **ETNREFCLK** in the CGU. To exit Off mode software should re-enable the **ETNREFCLK** clock before following the wake-up step from [Section 11.7.1](#).

11.8 Application information

11.8.1 Reset

All registers in the ETN1 have a reset value of 0 unless specified differently in [Section 11.4.2](#). Refer to [Section 11.6.4](#) for a correct HW initialization of ETN1 after reset.

Parts of the ETN1 design can be soft reset by setting bits in the **etn1_command** register and the **etn1_mac1** configuration register.

The **etn1_mac1** register has six different reset bits:

- SOFT_RESET: Setting this bit will put all modules in the PE-MAC in reset except for the registers **etn1_mac1** -- **etn1_sa2**. The value after a power-on reset assertion is 1 i.e. the soft reset needs to be cleared after a hardware reset.
- RESET_PEMCS/Rx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the receive function in the PE-MAC. The value after a system reset assertion is 0.
- RESET_PERFUN: Setting this bit will reset the receive function in the PE-MAC. The value after a sys_rst_an assertion is 0.

- RESET_PEMCS/Tx: Setting this bit will reset the MAC Control Sublayer (pause frame logic) and the transmit function in the PE-MAC. The value after a sys_rst_an assertion is 0.
- RESET_PETFUN: Setting this bit will reset the transmit function of the PE-MAC. The value after a sys_rst_an assertion is 0.

The above reset bits must be cleared by software.

The RESET_PERMII bit in the **etn1_supp** register allows soft resetting of the RMII logic. The reset must be cleared by software.

The **etn1_command** register (see also [Table 580 on page 572](#)) has three different reset bits:

- TxReset: Writing a “1” to the TxReset bit will reset the transmit data path, excluding the PE-MAC portions, including all (read-only) registers in the transmit data path and as well as the **etn1_txproduceindex** register in the host registers module. Soft resetting the transmit data path will abort all DTL transactions of the transmit data path. The reset bit will be cleared autonomously by the ETN1. Soft resetting the Tx data path will clear the TxStatus and TxRtStatus bits in the Status register.
- RxReset: Writing a “1” to the RxReset bit will reset the receive data path, excluding the PE-MAC portions, including all (read-only) registers in the receive data path as well as the **etn1_rxconsumeindex** register in the host registers module. Soft resetting the receive data path will abort all DTL transactions of the receive data path. The reset bit will be cleared autonomously by the ETN1. Soft resetting the Rx data path will clear the RxStatus bit in the Status register.
- RegReset: Resets all of the data paths and registers in the host registers module, excluding the registers in the PE-MAC. Soft resetting the registers will also abort all DTL transactions of the transmit and receive data path. The reset bit will be cleared autonomously by the ETN1.

To do a full soft reset of the ETN1 device driver software has to:

- set the SOFT_RESET bit in the **etn1_mac1** register to 1
- set the RegReset bit in the **etn1_command** register, this bit clears automatically
- re initialize the PE-MAC core registers (**etn1_mac1** -- **etn1_sa2**)
- reset the SOFT_RESET bit in the **etn1_mac1** register to 0

To reset the just transmit data path the device driver software has to:

- set the RESET_PEMCS/Tx. bit in the **etn1_mac1** register to 1
- disable the Tx DMA managers by setting the Tx(Rt)Enable bits in the **etn1_command** register to 0
- set the TxReset bit in the **etn1_command** register, this bit clears automatically
- reset the RESET_PEMCS/Tx. bit in the **etn1_mac1** register to 0
- reset the RESET_PEMCS/Tx. bit in the **etn1_mac1** register to 0

To reset just the receive data path the device driver software has to:

- disable the receive function by resetting the RECEIVE_ENABLE bit in the **etn1_mac1** configuration register and resetting of the RxEnable bit of the **etn1_command** register.

1

2

3

- set the RESET_PEMCS/Rx. bit in the **etn1_mac1** register to 1

4

- set the RxReset bit in the **etn1_command** register, this bit clears automatically

5

- reset the RESET_PEMCS/Rx. bit in the **etn1_mac1** register to 0

6

7

A soft reset of the transmit data paths will generate a DTL abort assertion on the transmit descriptor read, status write and data read interfaces to be able to re initialize the DTL interface.

8

9

10

11

A soft reset of the receive data path will generate a DTL abort assertion on the receive descriptor read, status write and data write interfaces to be able to re initialize the DTL interface.

12

13

14

15

11.8.2 Duplex modes

The ETN1 can operate in full duplex and half duplex mode. Half or full duplex mode needs to be configured by the device driver software during initialization of the ETN1.

16

17

In case of a full duplex connection the FullDuplex bit of the **etn1_command** register needs to be set to 1 and the FULL_DUPLEX bit of the **etn1_mac2** configuration register needs to be set to 1; for half duplex the same bits needs to be set to 0.

18

19

11.8.3 Transmit example

Figure 197 illustrates the transmit process in an example transmitting a frame header of 8 bytes and a frame payload of 12 bytes.

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

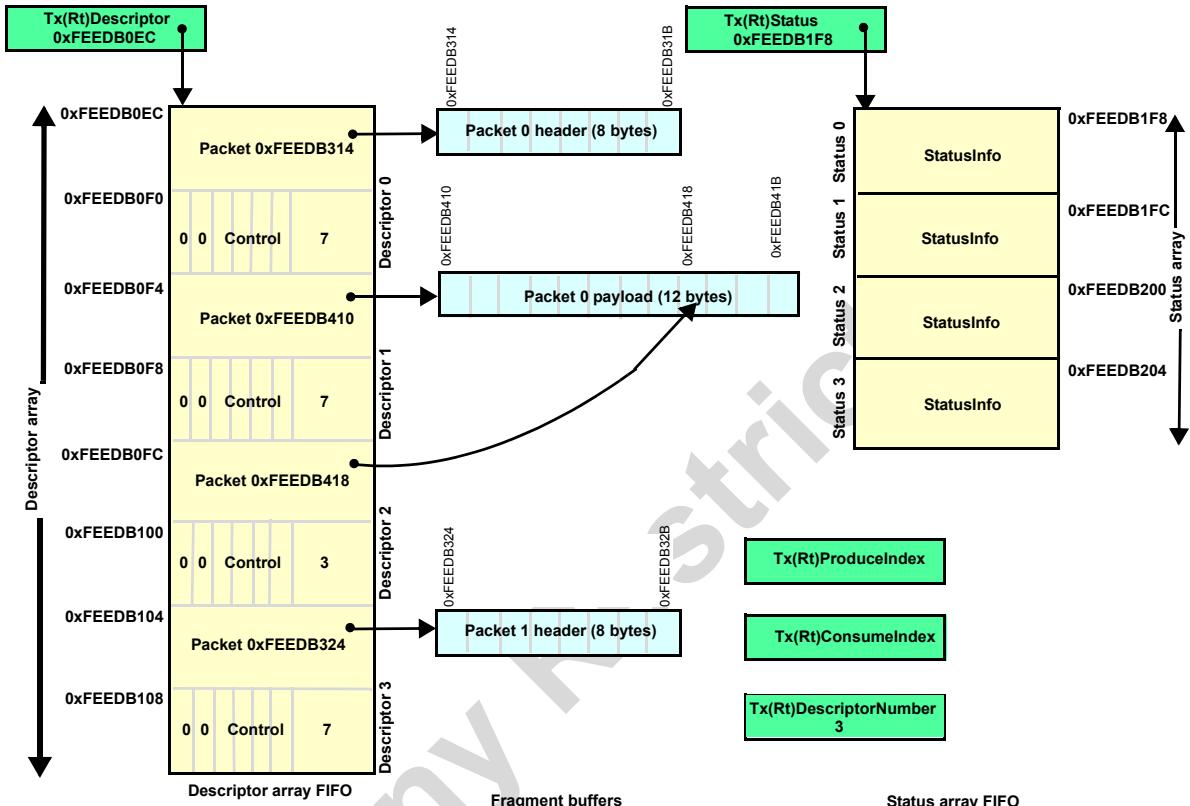


Fig 197. Transmit example memory and registers

After reset the values of the ETN1 DMA registers will be zero. During initialization the device driver will allocate the descriptor and status array in memory. In this example an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address boundary. Since the number of descriptors should match the number of statuses the status array consists of four elements; the array is 4x1x4 bytes and aligned on a 4 byte address boundary. The device driver writes the base address of the descriptor array (0xFEEDB0EC) in the **etn1_tx(rt)descriptor** register and the base address of the status array (0xFEEDB1F8) in the **etn1_tx(rt)status** register. The device driver writes the number of descriptors and statuses (4) in the **etn1_tx(rt)descriptornumber** register. The descriptors and statuses in the arrays need not be initialized, yet

Initialization may already enable the transmit data path by setting the Tx(Rt)Enable bit in the **etn1_command** register. In case the transmit data path is enabled while there are no further frames to send the ETN1 will set the Tx(Rt)FinishedInt interrupt flag. To reduce the processor interrupt load some interrupts can be disabled by setting the relevant bits in the **etn1_intenable** register of the ETN1.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Now suppose application software wants to transmit a frame of 12 bytes using a
TCP/IP protocol. The TCP/IP stack will add a header to the frame. For the ETN1 the
frame header need not be in memory in front of the payload data. The device driver
can program a Tx gather DMA to collect header and payload data. To do so, the
device driver will program the first descriptor to point at the frame header; the “last”
flag in the descriptor will be set to false/0 to indicate a multi-fragment transmission.
The device driver will program the next descriptor to point at the actual payload data.
The maximum size of a payload buffer is 2KB so a single descriptor suffices to
describe the payload buffer. For the sake of the example though the payload is
distributed across two descriptors. After the first descriptor in the array describing the
header the second descriptor in the descriptor array describes the initial 8 bytes of
the payload; the third descriptor in the array describes the remaining 4 bytes of the
frame. In the third descriptor the “last” bit in the Control word is set to true/1 to
indicate it is the last descriptor in the frame. In this example the Interrupt bit in the
descriptor Control field is set in the last fragment of the frame trigger an interrupt after
the transmission completed. The Size field in the descriptor’s Control word is set to
the number of bytes in the fragment buffer, -1 encoded.

After setting up the descriptors for the transaction the device driver increments the
etn1_tx(rt)produceindex register by 3 since three descriptors have been
programmed. If the transmit data path was not enabled during initialization the device
driver needs to enable the data path now.

If the transmit data path is enabled the ETN1 will start transmitting the frame as soon
as it detects the **etn1_tx(rt)produceindex** is not equal to **etn1_tx(rt)consumeindex**
- both were zero after reset. The ETN1 Tx(Rt) DMA will start reading the descriptors
from memory issuing a single DTL MMBD command with the base address from the
etn1_tx(rt)descriptor register and a block size value:

3 descriptors * 2words per descriptor - 1 = 5 (block_size is -1 encoded).
The memory system will return the descriptors and the ETN1 will accept them one by
one while issuing the DTL MMBD read commands for reading the transmit data
fragments. The commands will have the address from the Packet field in the
descriptor and a block size equal to the Size field in the descriptor.

As soon as transmission read data is returned by the MMBD interface the ETN1 will
try to start transmission on the Ethernet connection via the RMII interface. DTL
MMBD command for reading fragment buffers are issued in parallel to be able to
prefetch transmission data.

While issuing the descriptor MMBD read commands the Tx(Rt) DMA manager also
issues DTL MMBD commands for writing the transmission status without actually
writing the status information. The status write command address will be taken from
the **etn1_tx(rt)status register**; the block size value will be:

3 statuses * 1 word - 1 = 2 (block_size is -1 encoded).
After transmitting each fragment of the frame the Tx(Rt) DMA will write the status of
the fragment’s transmission across the status DTL MMBD interface. Statuses for all
but the last fragment in the frame will be written as soon as the data in the frame has
been accepted by the Tx(Rt) DMA manager. The status for the last fragment in the
frame will only be written after the transmission has completed on the Ethernet
connection.

After the Tx(Rt) DMA manager has committed the status write information to memory, the Tx(Rt)ConsumeIndex is updated and the interrupt flags are forwarded to the etn1_intstatus register.

Since the Interrupt bit in the descriptor of the last fragment is set, after committing status of the last fragment to memory the ETN1 will trigger a Tx(Rt)DoneInt interrupt which triggers the device driver to inspect the status information.

In this example the device driver cannot add new descriptors as long as the ETN1 has incremented the Tx(Rt)ConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet, see [Section 11.6.3.4](#)). Only after committing the status for the first fragment to memory and updating the Tx(Rt)ConsumeIndex to 1 the device driver can program the next (the fourth) descriptor. The fourth descriptor can already be programmed before completely transmitting the first frame.

Each byte transferred on the MMBD data read interface is transmitted across the RMII interface as four times 2data bits (clock frequency 50 MHz). On the RMII interface the ETN1 hardware adds the preamble, frame delimiter leader and the CRC trailer if hardware CRC is enabled. Once transmission on the RMII interface commences the transmission cannot be interrupted without generating an underrun error which is why descriptors and data read command are issued as soon as possible and pipelined.

In 10Mb/s mode data will only be transmitted once every 10 clock cycles; an clock gate disables the 50MHz clock for the 9 cycles.

11.8.4 Receive example

[Figure 198](#) illustrates the receive process in an example receiving a frame of 16 bytes.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

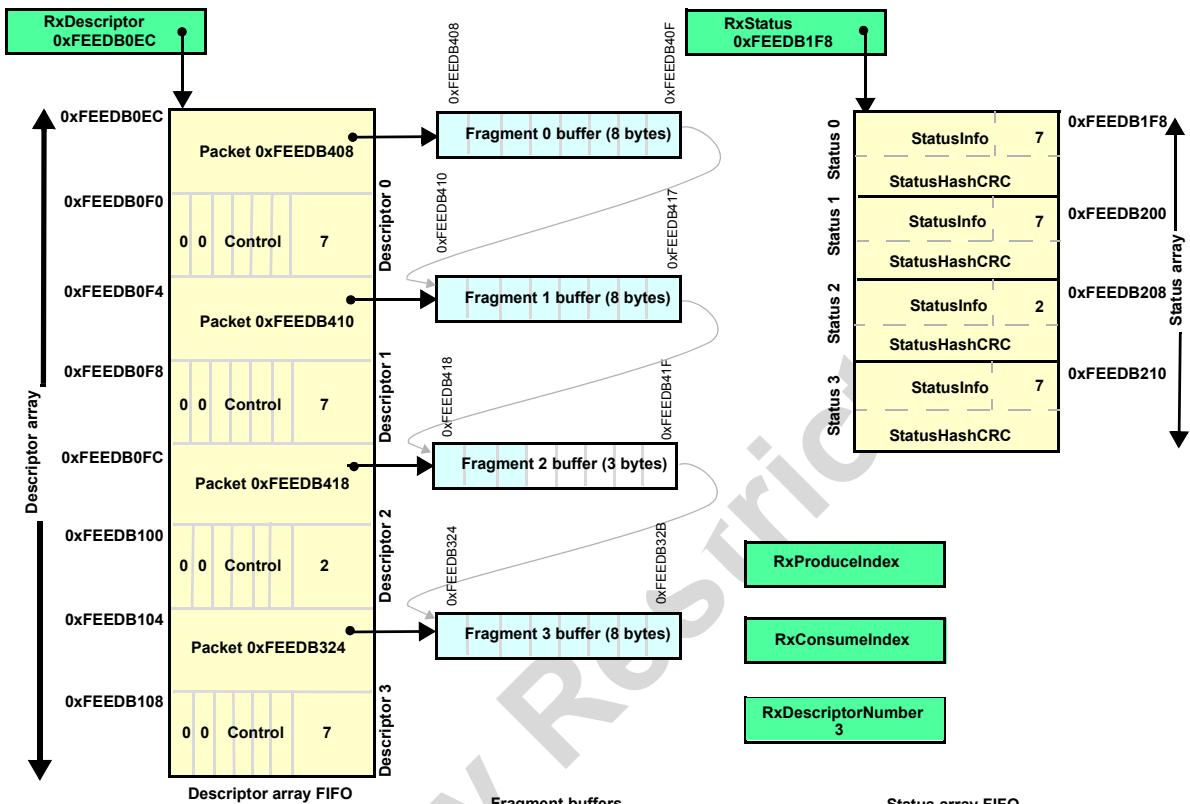


Fig 198.Receive example memory and registers

After reset the values of the ETN1 DMA registers will be zero. During initialization the device driver will allocate the descriptor and status array in memory. In this example an array of four descriptors is allocated; the array is 4x2x4 bytes and aligned on a 4 byte address boundary. Since the number of descriptors should match the number of statuses the status array consists of four elements; the array is 4x2x4 bytes and aligned on a 8 byte address boundary. The device driver writes the base address of the descriptor array (0xFEEDB0EC) in the **etn1_rxdescriptor** register and the base address of the status array (0xFEEDB1F8) in the **etn1_rxstatus** register. The device driver writes the number of descriptors and statuses (4) in the **etn1_rxdescriptornumber** register. The descriptors and statuses in the arrays need not be initialized, yet.

After allocating the descriptors a fragment buffer needs to be allocated for each of the descriptors. Each fragment buffer can be between 0 and 2K bytes. The base address of the fragment buffer is stored in the Packet field of the descriptors. The number of bytes in the fragment buffer is stored in the Size field of the descriptor Control word. The Interrupt field in the Control word of the descriptor can be set to generate an interrupt as soon as the descriptor has been filled by the receive process. In this example the fragment buffers are 8 bytes so the value of the Size field in the Control word of the descriptor is set to 7. Note that in this example the fragment buffers are actually a continuous memory space; even when a frame is distributed over multiple

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

fragments most of the time it will be in a linear, continuous memory space; only when the descriptors wrap at the end of the descriptor array the frame will not be in a continuous memory space.

The device driver should enable the receive process by writing a 1 to the RxEnable bit of the **etn1_command** register after which the MAC needs to be enabled by writing a 1 to the RECEIVE_ENABLE bit of the **etn1_mac1** configuration register. The ETN1 will now start receiving Ethernet frames. To reduce the processor interrupt load some interrupts can be disabled by setting the relevant bits in the **etn1_intenable** register of the ETN1.

After enabling the Rx DMA manager the Rx DMA manager will start issuing descriptor read commands. In this example the number of descriptors is 4. Initially the **etn1_rxproduceindex** and **etn1_rxconsumeindex** are 0. Since the descriptor array is considered full if **etn1_rxproduceindex == etn1_rxconsumeindex - 1**, the Rx DMA manager can only read (**etn1_rxconsumeindex - etn1_rxproduceindex - 1 =** 3 descriptors; note the wrapping. The Rx DMA manager reads the descriptors by issuing a DTL MMBD read command on the descriptor read interface; the start address will be 0xFFEEDBDEC (RxDescriptor) and the block size will be: 3 descriptors * 2 words per descriptor -1 = 5 (block_size is -1 encoded).

While issuing the descriptor MMBD read commands the Rx DMA manager also issues DTL MMBD commands for writing the transmission status without actually writing the status information. The status write command address will be taken from the **etn1_rxstatus** register; the block_size value will be 3 statuses * 1 double words - 1 = 2 (block_size is -1 encoded).

After enabling the receive function in the MAC the ETN1 will start receiving data from the RMII interface starting at the next frame i.e. if the receive function is enable will the RMII interface is halfway through sending a frame the frame will be discarded and reception will start at the next frame. The ETN1 will strip the preamble and start of frame delimiter from the frame. If the frame passes the receive filtering the Rx DMA manager will start writing the frame to the first fragment buffer.

Suppose the frame is 19 bytes then this frame will be distributed over three fragment buffers. After writing the initial 8 bytes in the first fragment buffer the status for the first fragment buffer will be written and the Rx DMA will continue filling the second fragment buffer. Since this is a multi-fragment receive, the status of the first fragment will have a 0 for the Last bit in the StatusInfo word; the EntryLevel field will be set to 7 (8, -1 encoded). After writing the 8 bytes in the second fragment the Rx DMA will continue writing the third fragment. The status of the second fragment will be like the status of the first fragment: Last = 0, EntryLevel = 7. After writing the three bytes in the third fragment buffer the end of the frame has been reached and the status of the third fragment is written. The third fragment.s status will have the Last bit set to 1 and the EntryLevel equal to 2 (3, -1 encoded).

The next frame received from the RMII interface will be written to the fourth fragment buffer i.e. five bytes of the third buffer will be unused.

After the Rx DMA manager has committed the receive data and status data to memory, the RxProduceIndex is updated and the interrupt flags are forwarded to the **etn1_intstatus** register.

After committing status of the fragments to memory the ETN1 will trigger a RxDoneInt interrupt which triggers the device driver to inspect the status information. In this example all descriptors have the Interrupt bit set in the Control word i.e. all descriptors will generate an interrupt after committing data and status to memory.

In this example the receive function of the ETN1 cannot read new descriptors as long as the device driver does not increment the RxConsumeIndex because the descriptor array is full (even though one descriptor is not programmed yet, see [Section 11.6.3.4 "Full and empty state of FIFOs"](#)). Only after the device driver has forwarded the receive data to application software and after the device driver has updated the RxConsumeIndex by incrementing it by 3, the ETN1 continue reading descriptors and receive data. The device driver will probably increment the RxConsumeIndex by 3 since the driver will forward the complete frame consisting of three fragments to the application and hence free up three descriptors at the same time.

The ETN1 removes preamble and frame start delimiter from the RMII data and checks the CRC. To limit the buffer NoDescriptor error probability the ETN1 buffers two descriptors. After executing the tag/tag acknowledge protocol open the data and status write DTL interfaces the RxProduceIndex is updated. The software device driver will process the receive data after which the device driver will update the RxConsumeIndex.

11.8.5 Cache coherency

Because the device driver can produce descriptors with write-only operations and consume the status fields with read-only operations, this can be made 'cache safe' and descriptors can be packed together in cache blocks and cached, statuses can be packed together too. The device driver needs to take care of cache coherency, if cache coherency is not enforced by a snooping cache in the host processor.

If the device driver doesn't own all of the statuses in a cache block that contains multiple statuses, then the Ethernet hardware may be writing to a status field in memory while that status is also included in a cache block loaded in the host processor's cache, causing the values for that status in the host cache to be stale. This can be solved by invalidating these cache blocks and causing them to be read again from the host memory whenever the device driver receives ownership of these statuses i.e. when the **etn1_tx(rt)consumeindex** or **etn1_rxproduceindex** are updated.

Before updating the **etn1_tx(rt)produceindex** or **etn1_rxconsumeindex** the device driver needs to make sure the associated descriptors are written back from the cache to the memory, so that the new descriptor values become visible for the Ethernet hardware. When flushing these cache blocks, care must be taken not to modify the fields of descriptors that are already owned by the Ethernet hardware.

Alternatively, the device driver can use non-cached memory traffic for the descriptors and statuses. In that case there is no need to worry about cache coherency, at the expense of a higher amount of memory traffic for the descriptors and statuses.

11.8.6 Statistics counters

Generally, in Ethernet applications many counters need to be maintained that count Ethernet traffic statistics. There are a number of standards specifying such counters, such as IEEE std 802.3 / clause 30. Other standards are RFC 2665 and RFC 2233.

The ETN1 hardware block does not offer counters in hardware. But with the help of the StatusInfo field in frame statuses many of the important statistics events listed in the standards can be counted by software.

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

12. Limiting Values

Table 624: Limiting values in accordance with the Absolute Maximum Rating System (IEC60134) [21.] [1][2]

Symbol	Parameter	Min	Max	Unit
VDD _{LOW}	supply voltage for the digital die (VDDC, VDDACGU, VDDCGU)	-0.5	+1.5	V
VDD _{DIGITAL}	supply voltage for DPU pads (VDDL)	-0.5	+2.45	V
VDD _{INT}	supply voltage for the die interface (DAIF on DPU and ADIF on APU)	-0.5	+3.3	V
VDD _{HIGH1}	supply voltage for high voltage pads (VDDH)	-0.5	+3.8	V
VDD _{HIGH2}	supply voltage for USIM, and USB pads (VDDH)	-0.5	+4.6	V
VDD _{Dual_voltage}	supply voltage for dual voltage pads (VDDM)	-0.5	+5.2	V
VDD _{SIGNAL_HIGH}	supply voltage into any signal pin	-0.5	VDDE + 0.5 [5]	V
P _{TOT} [3][4]	total maximum power dissipation	-	tbd.	W
T _{AMB}	operating ambient temperature	0	+70	°C
T _{JUNC} [4]	operating junction temperature		125	°C
Θ _{JA}	thermal resistance of the package from junction to ambient			K/W
			32.5 [6]	
T _{stg}	storage ambient temperature	-65	+150	°C

[1] Stresses above those listed may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other condition above those indicated in this objective specification is not implied.

[2] This product includes circuits specifically designed for the protection of internal devices from damaging effects of excessive static charge. Nonetheless it is suggested that conventional precautions should be taken to avoid applying any voltage larger than the rated maxima.

[3] The total power dissipation may not exceed the value P_{TOT} calculated according the following formula (e.g. Θ_{JA} = 33 K/W):

$$P_{TOT} = \frac{T_{JUNC} - T_{AMB}}{\Theta_{JA}}$$

[4] For operating at elevated temperatures, the device must be derated based on 125 °C junction temperature.

[5] VDDE can be either VDDL, VDDH or VDDM depending on the voltage domain the pad is connected.

[6] with air flow of zero m/s

Note: Stresses above those listed under Absolute Maximum Ratings may cause permanent device failure. Functionality at or above limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

13. DC Characteristics

13.1 Supply Voltage

The operation voltage ranges are given in Table 625. All voltages can be independently set if the constraints given in table are maintained. External independent decoupling for all power domains is strongly advised. There are two pad voltage ranges: while VDDL targets ICs with 1.8 V signalling, VDDM target standard V ICs, VDDM are used for the dual voltage pads and target IC's with either 1.8V or 2.8V signalling.

There are no particular requirements on power sequencing between core and pads supplies. This means that pads supplies (VDDL, VDDH and VDDM) can be turned-off even if core voltage is still on. However, in order to avoid IO pin leakage, all input pins have to be LOW when the corresponding pad supply is turned-off.

VDDC, VDDCGU, and VDDACGU have to be powered at the same time.

Table 625: Supply voltages ranges

Symbol	Description	Conditions	Min	Typ	Max	Unit
VDDC [1]	DPU core power supply	whole range	1.15 [2]	1.20	1.32	V
VDDCGU [1]	DPU CGU power supply		1.15	1.20	1.32	V
VDDACGU [1]	DPU CGU PLL power supply		1.15	1.20	1.32	V
VDDINT [1]	Die interface (DAIF) power supply	at DVFD818x	1.65	1.80	1.95	V
VDDL [1]	DPU low voltage pad		1.65	1.80	1.95	V
VDDH [1]	DPU high voltage pad power supply		3.00 [8]	3.30	3.60	V
VDDH [1][3]	DPU USB pad power supply		3.00	3.30	3.60	V
VDDM [4] /	DPU HMP pad supply voltage	Mobil memory level Standard memory level	1.65 2.50	1.80 2.80	1.95 [7] 3.10 [6]	V
VDDM [5]	DPU FMP pad supply voltage	Mobil memory level Standard memory level	1.65 2.50	1.80 2.80	1.95 [7] 3.10 [6]	V
VDDM [12]	DPU muxed memory port pads supply voltage	Mobil memory level Standard memory level	1.65 2.50	1.80 2.80	1.95 [7] 3.10 [6]	V

[1] All voltages can be switched-off independently (with the exception of VDDC / VDDCGU / VDDACGU) without damaging the circuits or generating internal cross-currents.

[2] VDDC can be reduced to minimum 0.90 V +/- 5% in sleep state if DSP subsystem is switched off.

[3] The DVFD818x Family is compliant to the USB specification in the given voltage range.

[4] Supply rail for pads of HMP[41:0]. The pads FMP[57:56] are muxed with HMP[41:40] and therefore if selected in GPM also supplied by VDDM.

[5] Supply rail for pads of FMP[57:0].

[6] The pads are validated to operate up to 3.3V. The operation up to 3.6V will lead to higher bouncing.

[7] For low/high speed configuration of the pads see [Table 93](#).

[8] The pads are validated to operate down to 2.75V

[Table 626](#) gives the maximum supported frequency with respect to the core power supply voltage.

Table 626:DPU core power supply voltage and maximum core frequency

Symbol	Description	Maximum clock frequency	Min	Typ	Max	Unit
VDDC_op2 [2]	Maximum performance operating mode	armclk	221 MHz	1.20	1.25	1.32
		hclk/pclk1	110.5 MHz			
		pclk2	52 MHz			
		dcclk/diclk	125 MHz			
		fc_clk	104 MHz			

Table 626:DPU core power supply voltage and maximum core frequency...continued

Symbol	Description	Maximum clock frequency	Min	Typ	Max	Unit
VDDC_op1	Typical operating mode	armclk	208 MHz	1.15	1.20	V
		hclk/pclk1	104 MHz			
		pclk2	52 MHz			
		dcclk/diclk	120 MHz			
		fc_clk	104 MHz			
VDDC_op0	Minimum power consumption operating mode	armclk	104 MHz	1.03	1.07	V
		hclk/pclk1	52 MHz			
		pclk2	26 MHz			
		dcclk/diclk	52 MHz			
		fc_clk	52 MHz			
VDDC_sleep	Sleep state	Reference clock stopped ^[1]	0.86	0.90	1.32	V

[1] Reference clock used depend on BBCKISEL level, see section [7.1.5](#)

[2] These operating points are not guaranteed

13.2 Power Dissipation

Table 627: Current consumption

Symbol	Description	Conditions	Min	Typ	Max	Unit
I _{VDDC_MAX}	current consumption with DSP and SC cores running: <ul style="list-style-type: none">• both at 100% worst case• no pad activity	dcclk, diclk = 104 MHz armclk = 208 MHz hclk, pclk1 = 104 MHz	-	160	258	mA
I _{VDDC_OP_OP}	current consumption with DSP and SC cores running: <ul style="list-style-type: none">• typical DECT application• no pad activity	dcclk, diclk = 104 MHz armclk = 208 MHz hclk, pclk1 = 104 MHz	-	tbd	-	mA
I _{VDDC_IDLE}	current consumption with DSP and SC cores in idle mode: <ul style="list-style-type: none">• all peripherals clocked and active• no pad activity	dcclk, diclk = 104 MHz armclk = 208 MHz hclk, pclk1 = 104 MHz	-	90	-	mA
I _{VDDC_STOP}	current consumption with DSP and SC cores in stop state: <ul style="list-style-type: none">• only CGU and TBU active		-	2.5	-	mA
I _{VDDC_SLEEP}	current consumption with DSP and SC cores in sleep state [1]	VDDC = 0.9 V and 25 °C temperature	-	tbd	-	µA
		VDDC = 1.2 V and 25 °C temperature	-	800	-	µA
		VDDC = 1.3 V and 25 °C temperature	-	tbd	-	µA
I _{VDDACGU_OP}	analog circuitry current consumption on VDDACGU	all PLL outputs at 312 MHz	-	-	1.4	mA
I _{VDDCGU_OP}	digital DPUCGU current consumption on VDDCGU	all PLL outputs at 312 MHz	-	-	3.6	mA
I _{VDDL_DPU_SLEEP}	pad current consumption on VDDL in sleep state		-	tbd	-	µA
I _{VDDH_SLEEP}	pad current consumption on VDDH in sleep state		-	tbd	-	µA
I _{VDDH_OP}	pad current consumption on VDDH with typical application		-	-	100	mA
I _{VDDH_SLEEP}	pad current consumption on VDDH in sleep state		-	tbd	-	µA
I _{VDDH_SLEEP}	pad current consumption on VDDH in sleep state		-	tbd	-	µA

[1] This parameter strongly depends on the leakage current of the device.

[2] Minimum conditions are 1.15 V and temperature -40 °C, slow process, typical conditions are 1.20 V and temperature 25 °C, nominal process, maximum conditions are 1.30 V and temperature 85 °C, fast process, analog voltage for both typical and maximum is 2.75 V.

13.3 Input Characteristics for Digital Pins

Table 628: DC input voltage characteristics [1]

Symbol	Description	Conditions	Min	Typ	Max	Unit
Inputs of pads:						
IH, IHM, IHMD, IHMD_nbs, IHMU, IHMU_nbs, TBHMP, TBHM_dualV, TBHMP_dualV, TBS1HP, TBS1PD, TBSH, TBSHP, TBSHPW1, TBSHPW2						
V _{IL}	low-level input voltage		0	-	0.3 × VDDE	V
V _{IH}	high-level input voltage		0.7 × VDDE	-	VDDE	V
V _{HYS} [2]	hysteresis voltage		0.4	-	600	mV
V _{HYS} [3]	hysteresis voltage		0.55	-	850	mV
V _{HYS} [4]	hysteresis voltage		0.1 × VDDE	-	-	V
Input of pad USB						
Single-ended receiver						
V _{IL_USB}	low-level input voltage		-0.5	-	0.8	V
V _{IH_USB}	high-level input voltage		2.0	-	VDDE+0.5	V
V _{HYS_USB}	hysteresis voltage for single-ended receivers		0.3	-	-	V
Differential receiver						
V _{DI_USB}	differential input sensitivity	(USBDP - USBDM)	0.2	-	-	V
V _{CMR_USB}	differential common mode range	include V _{DI_USB} range	0.8	-	2.5	V
Inputs of pads IICSDA, IICSCL						
V _{IL_IIC}	low-level input voltage		-0.5	-	0.3 × VDDE	V
V _{IH_IIC}	high-level input voltage		0.7 × VDDE	-	VDDE+0.5	V
V _{HYS_IIC}	hysteresis voltage		0.1 × VDDE	-	-	V

[1] VDDE can be either VDDL, VDDH or VDDM depending on the voltage domain the pad is connected.

[2] Valid for pads IH, IHM, IHMD, IHMD_nbs, IHMU, IHMU_nbs, TBHMP, TBHM_dualV, TBHMP_dualV if VDDE = 1.8V

[3] Valid for pads IH, IHM, IHMD, IHMD_nbs, IHMU, IHMU_nbs, TBHMP, TBHM_dualV, TBHMP_dualV if VDDE = 2.8V

[4] Valid for pads TBS1HP, TBS1PD, TBSH, TBSHPW1 and TBSHPW2

Table 629: DC input current characteristics [1][2]

Symbol	Description	Conditions	Min	Typ	Max	Unit
For pads IH,IHM, IHMD, IHMD_nbs, IHMU, IHMU_nbs, TBHMP, TBHM_dualV, TBHMP_dualV, TBS1HP, TBS1PD, TBSH, TBSHP, TBSHPW1, TBSHPW2						
I _{IL}	low-level sink current	V _{in} = 0	-	-	1	µA
For pads IH,IHM, IHMD, IHMD_nbs, IHMU, IHMU_nbs, TBHMP, TBHM_dualV, TBHMP_dualV, TBS1HP, TBS1PD, TBSH, TBSHP, TBSHPW1, TBSHPW2						
I _{IH}	high-level sink current without pull-down	V _{in} = VDDE	-	-	1	µA
For pads IHMU, IHMU_nbs, TBHM_dualV, TBHMP, TBHMP_dualV, TBSH						
I _{PU}	pull-up current	V _{in} = 0, VDDE = 1.8V	35	65	110	µA
		V _{in} = 0, VDDE = 2.8V	100	180	290	
For pads TBS1HP, TBS1PD, TBSHP, TBSHPW1, TBSHPW2						
I _{PU}	pull-up current	V _{in} = 0	25	50	80	µA
For pads IHMD, IHMD_nbs, TBHM_dualV, TBHMP, TBHMP_dualV, TBSH						
I _{PD}	pull-down current	V _{in} = VDDE, VDDE=1.8V	30	75	140	µA
		V _{in} = VDDE, VDDE=2.8V	100	220	400	
For pads TBS1HP, TBS1PD, TBSHP, TBSHPW1, TBSHPW2						
I _{PD}	pull-down current	V _{in} = VDDE	25	50	85	µA
For pads TBS1HP, TBS1PD, TBSHP, TBSHPW1, TBSHPW2						
I _{Repeater}	Repeater leakage current	V _{IO} = VDDE			1	µA
		V _{IO} = 2/3 × VDDE			55	µA
		V _{IO} = 1/3 × VDDE			60	µA
		V _{IO} = 0V			1	µA
For pad TBSHUX (USIM)						
R _{PU_USIMIO}	pull-up resistor		-	20	-	kΩ

[1] VDDE can be either VDDL or VDDH depending on the voltage domain the pad is connected.

[2] - For I/O pads with programmable inputs:

- a) I_{IH}, I_{IL} apply when programmed in plain input or repeater mode
- b) I_{PD}, I_{IL} apply when programmed in pull-down mode
- For I/O pads with pull-up inputs (i.e. IHU), I_{PU}, I_{IH} apply
- For I/O pads with pull-down inputs (i.e. IHD), I_{PD}, I_{IL} apply

13.4 Output Characteristics for Digital Pins

Table 630: DC output characteristics

Symbol	Description	Conditions	Min	Typ	Max	Unit
Pads on VDDL:						
TBHMP, TOM, TOM_fast						
V _{OL1}	low-level output voltage	I _{OL1} output current	-	-	0.4	V
V _{OH1}	high-level output voltage	I _{OH1} output current	VDDL - 0.4	-	-	V
I _{OL1}	low-level output current	V _{OL} = 0.4 V	6.0	-	-	mA
I _{OH1}	high-level output current	V _{OH} = VDDL - 0.4 V	5.5	-	-	mA
Pads on VDDINT (DPU):						
TBSH						
V _{OL2}	low-level output voltage	I _{OL2} output current	-	-	0.4	V
V _{OH2}	high-level output voltage	I _{OH2} output current	VDDINT - 0.4	-	-	V
I _{OL2}	low-level output current	V _{OL} = 0.4 V	7.0 ^[1]	-	-	mA
I _{OH2}	high-level output current	V _{OH} = VDDINT - 0.4 V	6.5 ^[1]	-	-	mA
Pads on VDDH:						
TBS1HP, TBS1PD, TBSHP, TBSHPW1, TBSHPW2						
V _{OL3}	low-level output voltage	I _{OL3} output current	-	-	0.4	V
V _{OH3}	high-level output voltage	I _{OH3} output current	VDDH - 0.4	-	-	V
I _{OL3}	low-level output current	V _{OL} = 0.4 V	3.0	-	-	mA
I _{OH3}	high-level output current	V _{OH} = VDDH - 0.4 V	6.5	-	-	mA
IIC SCL						
I _{OL_IICSL}	low-level output current	V _{OL_IICSL} = 0.4 V	3	-	-	mA
IIC SDA						
I _{OL_IICSDA}	low-level output current	V _{OL_IICSDA} = 0.4 V	3	-	-	mA
Pads on VDDH: USB						
V _{OL_USB}	low-level output voltage	R _{pu} of 1.5 kΩ to VDDE	-	-	0.3	V
V _{OH_USB}	high-level output voltage	R _{pd} of 15 kΩ to GNDE	2.8	-	-	V
Z _{DRV}	driver output resistance	Steady state drive	33	-	44	Ω
PADS on VDDM						
TBHMP_dualV, TBHM_dualV, TOM_dualV						
I _{OL5/6/7}	low-level output current	V _{OLM} = 0.3V, VDDE=1.8V	6	-	-	mA
		V _{OLM} = 0.3V, VDDE=2.8V	7			
I _{OH5/6/7}	high-level output current	V _{OHM} = V _{DDM} - 0.3V, VDDE=1.8V	5.5	-	-	mA
		V _{OHM} = V _{DDM} - 0.3V, VDDE=2.8V	6.5			

[1] Driving strength for VDDINT = 2.5V

Table 631: Electrostatic protection characteristics

Symbol	Description	Conditions	Min	Typ	Max	Unit
For all other pads						
V _{ESD}	electrostatic protection HBM [1]	leakage < 1 µA	-	-	2	kV
	electrostatic protection MM [2]		-	-	200	V

[1] Human Body Model: ANSI/ESD/ESD-S5.2-1994, Standard for ESD sensitivity testing, Human Body Model — Component level; of the Electrostatic Discharge Association, Rome, NY, USA.

[2] Machine Model: ANSI/ESD/ESD-S5.2-1996, Standard for ESD sensitivity testing, Machine Model — Component level; of the Electrostatic Discharge Association, Rome, NY, USA.

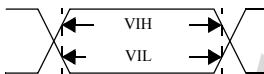
14. AC Characteristics of Digital Pins

The AC timings for the individual pins are to be found in the related sections within this document.

14.1 Input Characteristics

All AC parameters mentioned in this specification are tested based on the following input waveform specification.

All inputs

**Fig 199.AC input waveform****Table 632:Input rise and fall times**

Symbol	Description	Min	Typ	Max	Unit
T _{IN_RISE}	input rise time for UART, KBS, GPIO and WDRU	-	6	40	ns
T _{IN_FALL}	input fall time for UART, KBS, GPIO and WDRU	-	6	40	ns
T _{IN_RISE_FAST}	input rise time for JTAG, EBI and SPI ^[1]	-	6	10	ns
T _{IN_FALL_FAST}	input fall time for JTAG, EBI and SPI ^[1]	-	6	10	ns
T _{IN_RISE_I2C}	input rise time for I ² C-bus	20	-	300	ns
T _{IN_FALL_I2C}	input fall time for I ² C-bus	20	-	300	ns

[1] The tighter constraints for these peripherals are required in order to achieve the specified speed and in order to be compliant to all devices specified in this document.

Table 633:Input pin capacitance

Symbol	Description	Min	Typ	Max	Unit
C _{INPUT}	capacitance on pad types IH (at VDDL), IHD, IHD_nbs, IHU, IHU_nbs, TBHP	-	-	2.8	pF
	capacitance on pad types TBS1H, TBS1HP, TBS1PD	-	-	3.0	pF
	capacitance on pad types IH (at VDDINT), TBSH, TBSHP	-	-	3.3	pF
	capacitance on pad types TBHM, TBHMD, TBHMP	-	-	1.15	pF
	capacitance on pad types TBHMP_dualV, TBHM_dualIV, TOM_dualV	-	-	2.1	pF
	capacitance for pad type IICSDA	-	-	1.6	pF
	capacitance for pad type APIO	-	-	1.6	pF
	capacitance for pad type USB	-	10.8	11	pF
	capacitance for pad type LSIB ^[1]	-	3.0	3.5	pF

[1] The impedance of the LSIB is largely determined by the capacitance with respect to silicon substrate. Other parameters for the LSIB are described in the CGU chapter.

[2] This is the pad capacitance. The part of the package (tbd.) needs to be added.

Table 634:AC input voltage characteristics

Symbol	Description	Min	Typ	Max	Unit
PADs on VDDH:					
TBHM_dualV, TBHMP_dualV, TOM_dualV					
V _{IL(AC)}	low level input voltage	-	-	0.1 VDDE	V
V _{IH(AC)}	high level input voltage	0.9 VDDE	-	-	V

Company Restricted

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

14.2 Output Characteristics

The slew rate controlled pads mentioned in this specification feature in addition guaranteed transition times in a wider load range, reduced maximum signal slope, and smooth transition characteristic.

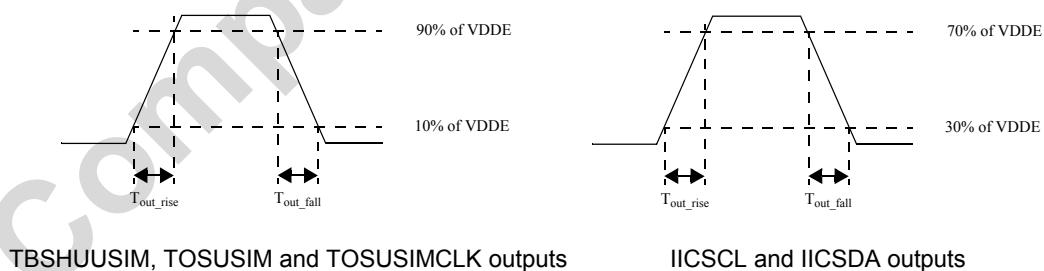
Table 635: Output rise and fall times [1]

Symbol	Description	Conditions	Min	Typ	Max	Unit	
Pads on VDDL:							
TBHMP, TOM (ESH=1)							
T _{OUT_RISE} ^[7]	output rise slew rate	Z = 75 Ω, C _{load} = 20pF	1.9	-	3.7	ns	
T _{OUT_FALL} ^[8]	output fall slew rate	Z = 75 Ω, C _{load} = 20pF	1.9	-	3.7	ns	
TOM_fast (ESH=0):							
T _{OUT_RISE} ^[7]	output rise slew rate	Z = 75 Ω	0.6	-	1.5	ns	
T _{OUT_FALL} ^[8]	output fall slew rate	Z = 75 Ω	0.6	-	1.5	ns	
Pads on VDDH: [2]							
TBSHP, TBSHPW1, TBSHPW2							
T _{OUT_RISE}	output rise slew rate	Z = 50 Ω	2.7	-	5.4	ns	
T _{OUT_FALL}	output fall slew rate	Z = 50 Ω	2.4	-	5.2	ns	
TBS1HP, TBS1PD							
T _{OUT_RISE}	output rise slew rate	Z = 50 Ω	1.0	-	2.1	ns	
T _{OUT_FALL}	output fall slew rate	Z = 50 Ω	1.0	-	3.1	ns	
IIC SCL, IIC SDA [5] [6]							
T _{OUT_RISE_IIC}	output rise time	10 pF (resistive pull-up)	21	-	250	ns	
		100 pF (resistive pull-up)	30				
		100 pF (current source pull-up)	30	-	250	ns	
		400 pF (current source pull-up)	60				
T _{OUT_FALL_IIC}	output fall time	10 pF (resistive pull-up)	21	-	250	ns	
		100 pF (resistive pull-up)	30				
		100 pF (current source pull-up)	30	-	250	ns	
		400 pF (current source pull-up)	60				
Pads on VDDH:							
USB							
Full-speed transmitting mode [4]							
T _{OUT_RISE_USB}	output rise time		4.0	-	20	ns	
T _{OUT_FALL_USB}	output fall time		4.0	-	20	ns	
F _{RFM_USB}	rise / fall time matching	(T _{OUT_RISE_USB} / T _{OUT_FALL_USB})	90	-	111.1	%	
V _{CRS_USB}	Output signal crossover voltage		1.3	-	2.0	V	
Low-speed transmitting mode							
T _{OUT_RISE_USBL}	output rise time [3]		75	-	300	ns	
T _{OUT_FALL_USBL}	output fall time [3]		75	-	300	ns	
F _{RFM_USBL}	rise / fall time matching	(T _{OUT_RISE_USBL} / T _{OUT_FALL_USBL})	80	-	125	%	

Table 635: Output rise and fall times [1]...continued

Symbol	Description	Conditions	Min	Typ	Max	Unit
V_{CRS_USBL}	Output signal crossover voltage		1.3	-	2.0	V
Pads on VDDM:						
TBHMMP_dualV, TBHM_dualV, TOM_dualV						
$T_{OUT_RISE}^{[7]}$	output rise time	$VDDE=1.8V (SSO=4)$ $VDDE=2.8V (SSO=4)$ $VDDE=1.8V (SSO=10), C_{load} = 20pF$	0.6 0.8 1.9	1.5	3.7	ns
$T_{OUT_FALL}^{[8]}$	output fall time	$VDDE=1.8V (SSO=4)$ $VDDE=2.8V (SSO=4)$ $VDDE=1.8V (SSO=10), C_{load} = 20pF$	0.6 0.8 1.9	1.5	3.7	ns

- [1] Minimum conditions are $VDDC = 1.3$ V, max $VDDE$ value, and temperature -40 °C, fast process,
Typical conditions are $VDDC = 1.20$ V, typical $VDDE$ value, and temperature 25 °C, nominal process,
Maximum conditions are $VDDC = 1.15$ V, minimum $VDDE$ value, and temperature 85 °C, slow process.
- [2] The transition time between 20% to 80% of $VDDE$. Refer to figure 201 and 202 for parameter definitions.
- [3] For Upstream port: Load Capacitance $CL = 50pF$ and $R_{pd} = 15k\Omega$ on USBDP and USBDM to external ground.
For Downstream port: Load Capacitance $CL = 200$ to $600pF$ on USBDP and USBDM, with $R_{pu} = 1.5k \Omega$ on USBDM to external supply ($VDDH$). Refer to figure 203 and 204 for parameter definitions
- [4] Measured between V_{OL_USB} and V_{OH_USB} , with a load capacitance of 50 pF. Refer to figure 203 and 204 for parameter definitions
- [5] Transition time between 30% and 70% of pad supply voltage, refer to figure 200
- [6] The rise time in standard/fast mode depends only upon the external pull-up resistive/current source that can be adjusted to meet the minimum rise time requirement.
- [7] L-to-H (10% $VDDE$ to 90% $VDDE$), load as defined in Figure 205
- [8] H-to-L (90% $VDDE$ to 10% $VDDE$), load as defined in Figure 205

**Fig 200. Output transition times for I2C**

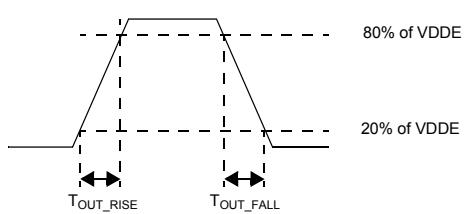
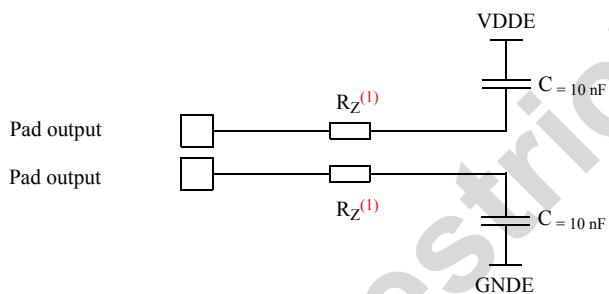


Fig 201. Output timing measurement condition



(1) $R_z = 70 \Omega$ for TBHMP, TOM, TOM_fast and TBSH pads, and 50Ω for TBSHP, TBSHPW1, TBSHPW2, TBS1HP and TBS1PD

Fig 202. Load setup for output timing measures for TBH^Mx, TOM, TBSH^x, TBS1H^x and TBS1P^x

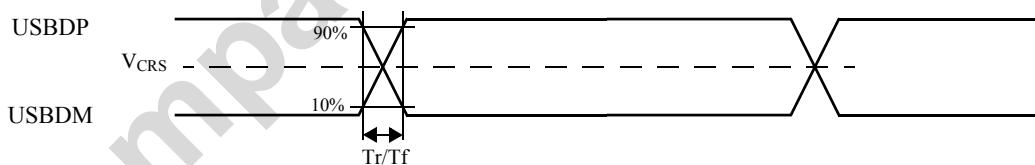
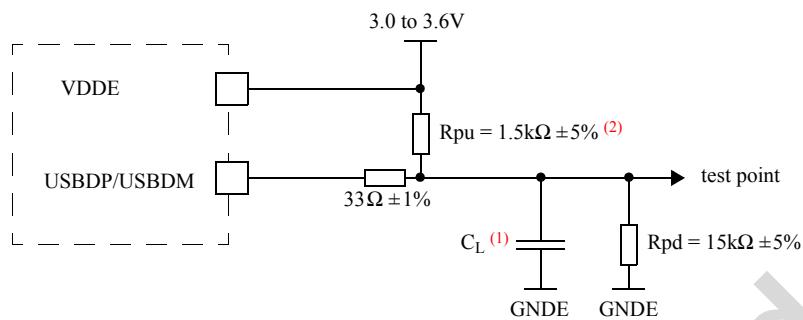


Fig 203. USB timing diagrams

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54



(1) $C_L = 50\text{pF}$ (full-speed mode, minimum or maximum rating)
 $C_L = 50\text{pF}$ to 600pF (low-speed mode, minimum or maximum rating)

(2) Full-speed mode: connected to USBDP, low-speed mode: connected to USBDM

Fig 204. USBDP, USBDM AC characteristic test load

Note: VDDE is VDDH

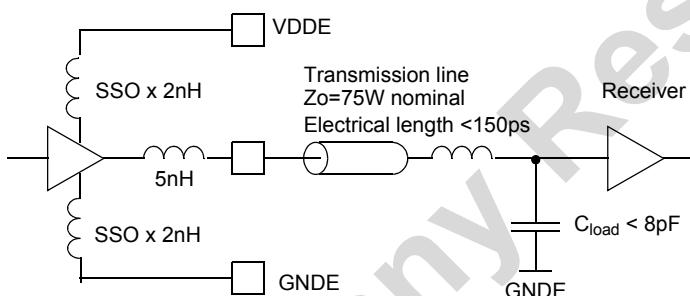


Fig 205. Load setup for measuring AC output timing of VDDM pads