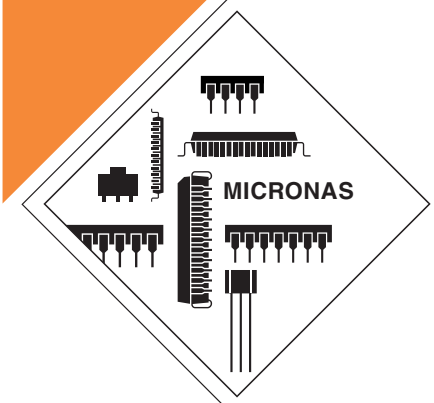


DATA SHEET

CDC 32xxG-C
Automotive Controller Family
User Manual

CDC 3205G-C
Automotive Controller
Specification

ARM POWERED®

Edition Feb. 10, 2005
6251-579-1DS

MICRONAS

Contents

Contents, continued

Contents, continued

Contents, continued

Contents, continued

Page Section Title

1. Introduction

Release Note: Revision bars indicate significant changes to the previous edition.

The device is a microcontroller for use in automotive applications. The on-chip CPU is an ARM® processor ARM7TDMI™ with 32-bit data and address bus, which supports Thumb™ format instructions.

The chip contains timer/counters, interrupt controller, multichannel AD converter, stepper motor and LCD driver, CAN interfaces and PWM outputs and a crystal clock multiplying PLL.

1.1. Features

Table 1–1: CDC32xxG-C Family Feature List

	This Device:					
Item	**CDC3205G-C MCM Flash**	**CDC3207G-C MCM Flash**	**CDC3217G-C MCM Flash**	**CDC3257G-C2 MCM Flash**	**CDC3272G-C Mask ROM**	**CDC3231G-C Mask ROM**
Core						
CPU	32-bit ARM7TDMI™					
CPU-active operation modes	DEEP SLOW, SLOW, FAST and PLL					
Power-saving operation modes (CPU inactive)	IDLE, WAKE and STANDBY					
CPU clock multiplication	PLL delivering up to 50 MHz					
EMI reduction mode	selectable in PLL mode					
Oscillators	4 to 5 MHz quartz and 32 kHz internal RC					
RAM, zero wait state, 32 bit wide	32 Kbyte			12 Kbyte	16 Kbyte	6 Kbyte
ROM	ROMless, ext. up to 4 M × 32/ 8 M × 16	512-Kbyte Flash (256 K × 16) top-boot conf.	1024-Kbyte Flash (512 K × 16) top-boot conf.	256-Kbyte Flash (128 K × 16) top-boot conf.	384 Kbyte (96 K × 32/ 192 K × 16)	128 Kbyte (32 K × 32/ 64 K × 16)
Boot ROM	8 Kbyte (special function ROM)					
Digital watchdog	✔					
Central clock divider	✔					
Interrupt controller expanding IRQ	40 inputs, 16 priority levels					26 inputs, 16 priority levels
Port interrupts including slope selection	6 inputs					5 inputs
Port wake-up inputs including slope/level selection	10 inputs					
Patch module	10 ROM locations					

Table 1–1: CDC32xxG-C Family Feature List

Item	This Device: CDC3205G-C MCM Flash	CDC3207G-C MCM Flash	CDC3217G-C MCM Flash	CDC3257G-C2 MCM Flash	CDC3272G-C Mask ROM	CDC3231G-C Mask ROM
Boot system	allows in-system downloading of external code to Flash memory via JTAG				-	
Device lock module	inhibits access to internal firmware, lock can be set by customer				-	
Analog						
Reset/Alarm	combined input for regulator input supervision					
Clock and supply supervision	✔					
10-bit ADC, charge balance type	16 channels (each selectable as digital input)					
ADC reference	VREF pin, P1.0 pin, P1.1 pin or VREFINT internal bandgap selectable					
Comparators	P06COMP with 1/2 AVDD reference, WAITCOMP with internal bandgap reference					
LCD	internal processing of all analog voltages for the LCD driver					
Communication						
DMA	3 DMA channels, one each for serving the graphics bus interface, SPI0 and SPI1					-
UART	2: UART0 and UART1					UART0
Synchronous serial peripheral interfaces	2: SPI0 and SPI1, DMA supported					
Full CAN modules V2.0B each with a 32-object RAM (LCAN000E)	4: CAN0, CAN1, CAN2 and CAN3			2: CAN0 and CAN1		1: CAN0
DIGITbus	1 master module					-
I^2C	2 master modules: I2C0 and I2C1					I2C0
Graphics bus interface	8-bit data bus, DMA supported, e.g., for connection of EPSON SED 1560 LCD controller					-
Input & Output						
Universal ports selectable as 4:1-mux LCD segment/back-plane lines or digital I/O ports	up to 52 I/O or 48 LCD segment lines (= 192 segments), individually configurable as I/O or LCD					up to 50 I/O or 46 LCD segment lines (= 184 segments)
Universal port slew rate	SW-selectable					
Stepper motor control modules with high-current ports	7 modules, 32 dI/dt-controlled ports					4 modules 23 dI/dt-controlled ports

Table 1–1: CDC32xxG-C Family Feature List

Item	This Device: CDC3205G-C MCM Flash	CDC3207G-C MCM Flash	CDC3217G-C MCM Flash	CDC3257G-C2 MCM Flash	CDC3272G-C Mask ROM	CDC3231G-C Mask ROM
PWM modules, each configurable as two 8-bit PWMs or one 16-bit PWM	6 modules: PWM0/1, PWM2/3, PWM4/5, PWM6/7, PWM8/9 and PWM10/11					5 modules: PWM0/1, PWM2/3, PWM4/5, PWM6/7, PWM8/9
Pulse/frequency modulator	2: PFM0 and PFM1					-
Audio module with auto-decay	✔					
SW-selectable clock outputs	2					
Polling/flash timer output	1 high-current port output operable in power-saving operation modes					
Timers & Counters						
16-bit free-running counters with capture/compare modules	CCC0 with 4 CAPCOM CCC1 with 2 CAPCOM					CCC0 with 4 CAPCOM
16-bit timers	1: T0					
8-bit timers	4: T1, T2, T3 and T4					
Real-time clock, delivering hours, minutes and seconds	✔					
Miscellaneous						
Scalable layout in CAN, RAM and ROM	-	✔				
Various HW options selectable at random	set by copy from user program storage during system start-up					
JTAG interface	allows Flash programming				✔	✔
On-chip debug aids	Embedded trace module, JTAG	JTAG				
Core bond-out	✔	-				
Supply voltage	3.5 to 5.5 V (limited I/O performance below 4.5 V)					
Case temperature range	0 °C to +70 °C	–40 °C to +105 °C				
Package						
Type	ceramic 257PGA	plastic 128QFP 0.5 mm pitch				
Bonded pins	256	128	128	128	126	111

1.2. Abbreviations

ADC	Analog-to-Digital Converter
AM	Audio Module
CAN	Controller Area Network Module
CAPCOM	Capture/Compare Module
CCC	Capture/Compare Counter
CPU	Central Processing Unit
DMA	Direct Memory Access Module
ERM	EMI Reduction Mode
ETM	Embedded Trace Module
I2C	I^2C Interface Module
LCD	Liquid Crystal Display Module
P06COMP	P0.6 Alarm Comparator
PWM	Pulse Width Modulator Module
SM	Stepper Motor Control Module
SPI	Serial Synchronous Peripheral Interface
T	Timer
UART	Universal Asynchronous Receiver Transmitter
WAITCOMP	Wait Comparator

1.3. Block Diagram

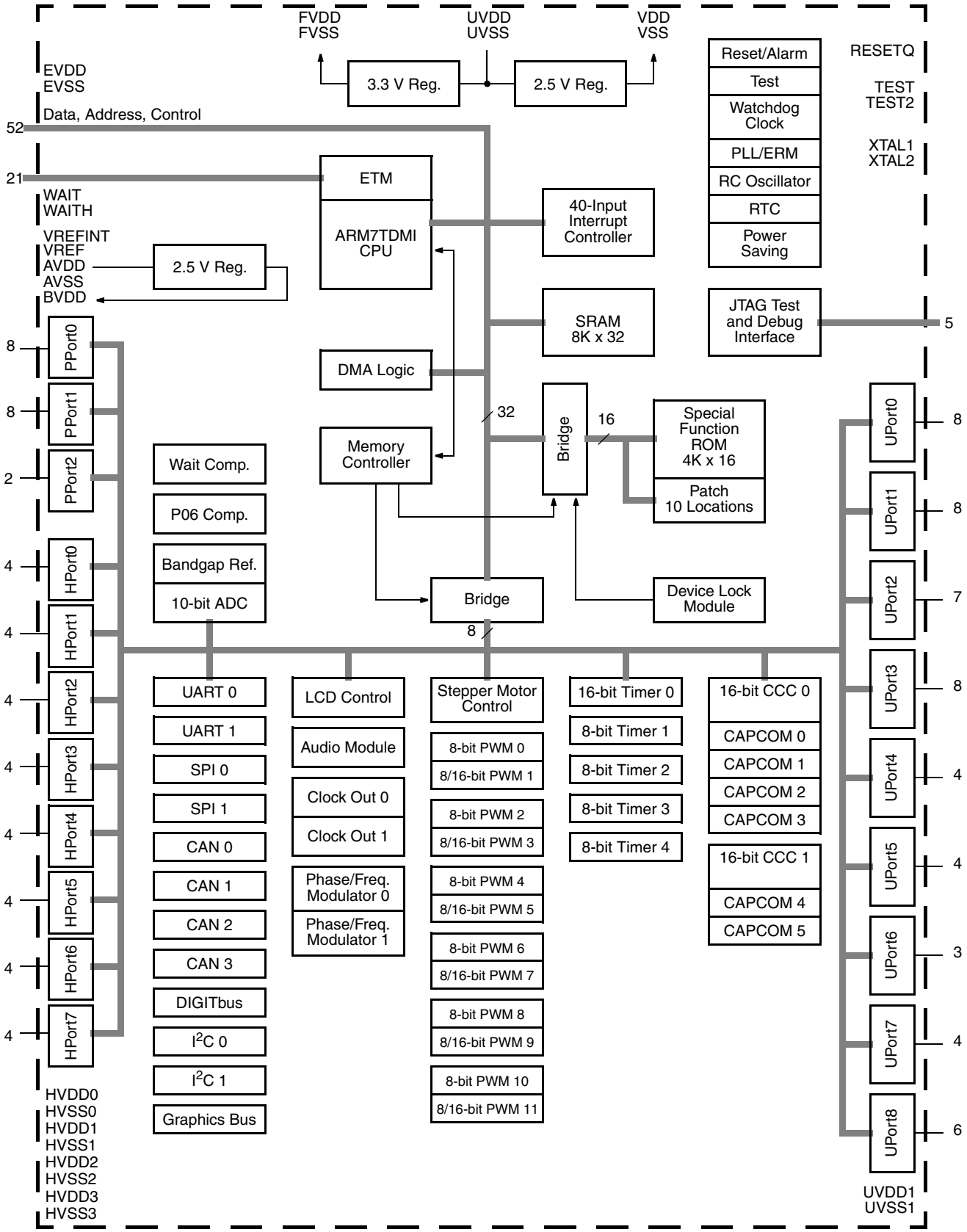


Fig. 1–1: CDC3205G-C block diagram

2. Packages and Pins

2.1. Pin Assignment

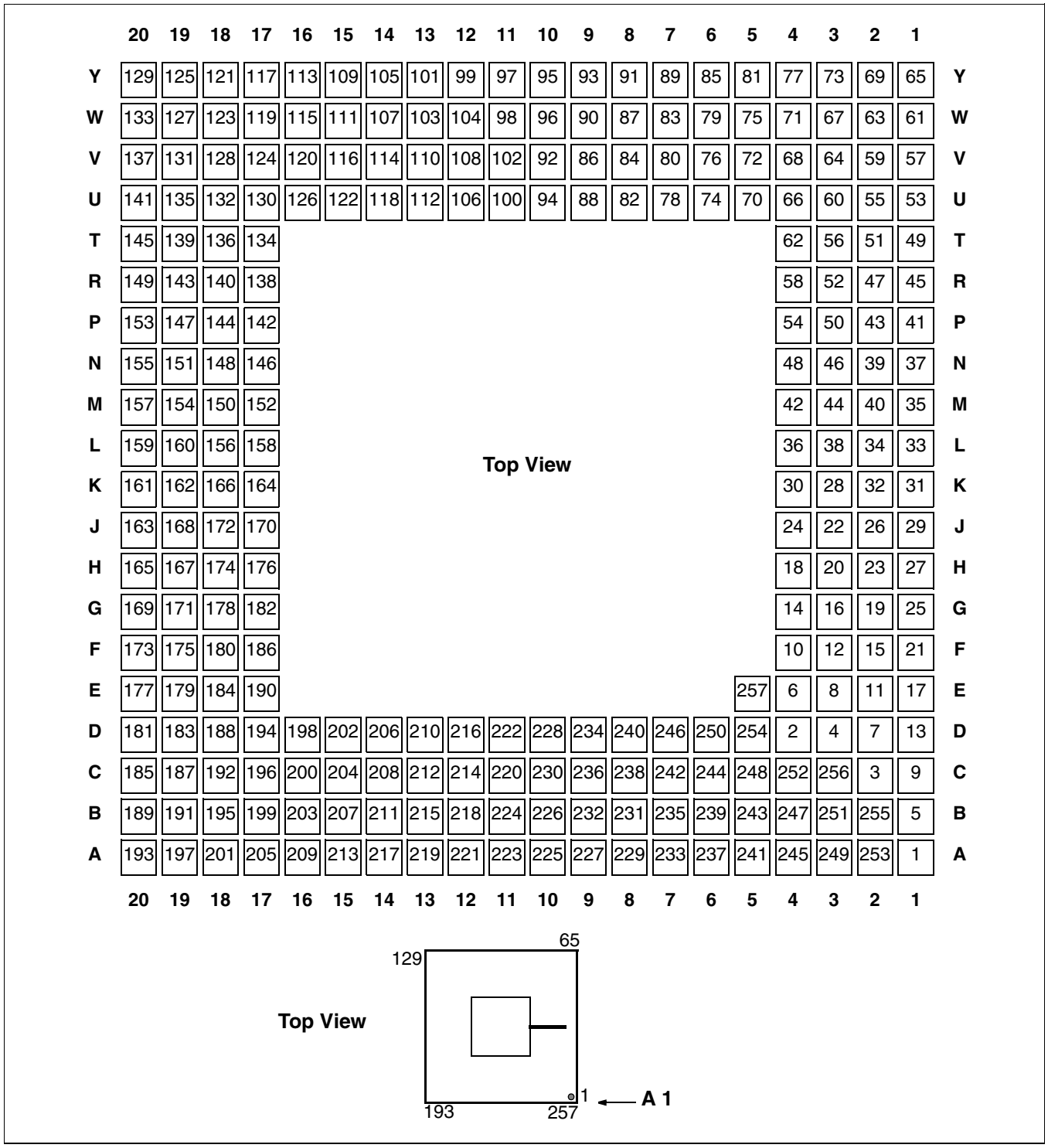


Fig. 2–1: Pin map of CPGA257 package

Table 2–1: Pin assignment for **CPGA257** package

CPGA257		PQFP128	Pin Functions			
Pin No.	Co-ord.	Pin No.	Basic Function	Port Special In	Port Special Out	LCD Mode
1	A1	4	U5.3/GD3	CC4-IN	CC4-OUT	SEG5.3
2	D4	5	U5.2/GD2	SDA1	SDA1	SEG5.2
3	C2	6	U5.1/GD1	SCL1	SCL1	SEG5.1
4	D3	7	U5.0/GD0		PFM0	SEG5.0
5	B1	8	U2.1	SDA0/CAN0-RX/WP6	SDA0	SEG2.1
6	E4	9	U2.0	SCL0	SCL0/CAN0-TX	SEG2.0
7	D2	10	U1.7	PINT0/WP0	PFM0	SEG1.7
8	E3	11	U1.6	PINT1	CO0/INTRES	SEG1.6
9	C1	12	U1.5	PINT2	CO0Q/CO1	SEG1.5
10	F4	13	TEST			
11	E2	14	RESETQ/ALARMQ			
12	F3	15	XTAL2			
13	D1	16	XTAL1			
14	G4	17	VSS			
15	F2	18	VDD			
16	G3	19	U1.4		ITSTOUT/AM-OUT	SEG1.4
17	E1	20	U1.3	WP3	MTO/AM-PWM	SEG1.3
18	H4	21	U1.2	MTI/ITSTIN	T0-OUT/INTRES	SEG1.2
19	G2	22	U1.1		T1-OUT	SEG1.1
20	H3	23	U1.0		T2-OUT	SEG1.0
21	F1	24	U0.7	WP4	T3-OUT	SEG0.7
22	J3	25	U0.6	CC3-IN	T4-OUT/CC3-OUT	SEG0.6
23	H2	26	U0.5	PINT4	CC3-OUT	SEG0.5
24	J4	27	U0.4	PINT5	CO1	SEG0.4
25	G1	28	U0.3		PWM0	SEG0.3
26	J2	29	U0.2		PWM1	SEG0.2
27	H1	30	U0.1		PWM2	SEG0.1
28	K3	31	U0.0		PWM3	SEG0.0
29	J1		D31			
30	K4		D30			
31	K1		D29			
32	K2		D28			
33	L1		D27			
34	L2		D26			
35	M1		D25			
36	L4		EVDD8			
37	N1		EVSS8			
38	L3		D24			
39	N2		D23			
40	M2		D22			
41	P1		D21			
42	M4		D20			
43	P2		D19			
44	M3		D18			
45	R1		D17			
46	N3		D16			
47	R2		D15			
48	N4		D7			
49	T1		D14			
50	P3		EVDD7			
51	T2		EVSS7			
52	R3		D6			
53	U1		D13			
54	P4		D5			
55	U2		D12			
56	T3		D4			
57	V1		D11			
58	R4		D3			
59	V2		D10			
60	U3		D2			
61	W1		D9			
62	T4		D1			
63	W2		D8			
64	V3		EVDD6			

Table 2–1: Pin assignment for **CPGA257** package

CPGA257		PQFP128	Pin Functions			
Pin No.	Co-ord.	Pin No.	Basic Function	Port Special In	Port Special Out	LCD Mode
65	Y1		EVSS6			
66	U4		D0			
67	W3		OEQ			
68	V4		CE0Q			
69	Y2		BWQ3			
70	U5		BWQ2			
71	W4		BWQ1			
72	V5		BWQ0			
73	Y3		EMUTRI			
74	U6		ABORT			
75	W5		EXTERN0			
76	V6		EXTERN1			
77	Y4	32	H7.3	SME-COMP3	SME1+/PWM4	
78	U7	33	H7.2	SME-COMP2	SME1-/PWM6	
79	W6	34	H7.1	SME-COMP1	SME2+/PWM8	
80	V7	35	H7.0	SME-COMP0	SME2-/PWM9	
81	Y5	36	HVDD2			
82	U8	37	HVSS2			
83	W7	38	H6.3		PWM8	
84	V8	39	H6.2		PWM9	
85	Y6	40	H6.1		PWM10	
86	V9	41	H6.0		PWM11	
87	W8	42	H5.3	SMD-COMP3	SMD1+	
88	U9	43	H5.2	SMD-COMP2	SMD1-	
89	Y7	44	HVDD0			
90	W9	45	HVSS0			
91	Y8	46	H5.1	SMD-COMP1	SMD2+	
92	V10	47	H5.0	SMD-COMP0	SMD2-	
93	Y9	48	H4.3	SMA-COMP3	SMA1+	
94	U10	49	H4.2	SMA-COMP2	SMA1-	
95	Y10	50	H4.1	SMA-COMP1	SMA2+	
96	W10	51	H4.0	SMA-COMP0	SMA2-	
97	Y11	52	H3.3	SMB-COMP3	SMB1+	
98	W11	53	H3.2	SMB-COMP2	SMB1-	
99	Y12	54	H3.1	SMB-COMP1	SMB2+	
100	U11	55	H3.0	SMB-COMP0	SMB2-	
101	Y13	56	H2.3	SMC-COMP3	SMC1+	
102	V11	57	H2.2	SMC-COMP2	SMC1-	
103	W13	58	HVDD1			
104	W12	59	HVSS1			
105	Y14	60	H2.1	SMC-COMP1	SMC2+	
106	U12	61	H2.0	SMC-COMP0	SMC2-	
107	W14	62	H1.3	SMF-COMP3	SMF1+	
108	V12	63	H1.2	SMF-COMP2	SMF1-	
109	Y15	64	H1.1	SMF-COMP1	SMF2+	
110	V13	65	H1.0	SMF-COMP0	SMF2-	
111	W15	66	HVDD3			
112	U13	67	HVSS3			
113	Y16	68	H0.3	SMG-COMP3	SMG1+/PWM1	
114	V14	69	H0.2	SMG-COMP2	SMG1-/PWM3/POL	
115	W16	70	H0.1	SMG-COMP1	SMG2+/PWM5	
116	V15	71	H0.0	SMG-COMP0	SMG2-/PWM7	
117	Y17		nTRST			
118	U14		ETDI			
119	W17		ETMS			
120	V16		ETCK			
121	Y18		ETDO			
122	U15		ABE			
123	W18		CE1Q			
124	V17		FBUSQ			
125	Y19		AMCS1			
126	U16		AICU2			
127	W19		AICU3			
128	V18		EVSS5			

Table 2–1: Pin assignment for **CPGA257** package

CPGA257		PQFP128	Pin Functions			
Pin No.	Co-ord.	Pin No.	Basic Function	Port Special In	Port Special Out	LCD Mode
129	Y20		EVDD5			
130	U17		AICU4			
131	V19		AICU5			
132	U18		AICU6			
133	W20		AICU7			
134	T17		A8			
135	U19		A18			
136	T18		A19			
137	V20		EVDD4			
138	R17		EVSS4			
139	T19		WEQ/RWQ			
140	R18		A9			
141	U20		A10			
142	P17		A11			
143	R19		A12			
144	P18		A13			
145	T20		A14			
146	N17		A15			
147	P19		EVDD3			
148	N18		EVSS3			
149	R20		A16			
150	M18		A17			
151	N19		A20			
152	M17		A21			
153	P20		A22			
154	M19		A23			
155	N20		AMCM21			
156	L18		AMCM22			
157	M20		EVDD2			
158	L17		EVSS2			
159	L20		AMCM23			
160	L19		SEQ			
161	K20		nMREQ			
162	K19		MAS0			
163	J20		MAS1			
164	K17		nRESET			
165	H20	72	P1.7	PINT5		
166	K18	73	P1.6	PINT4		
167	H19	74	P1.5	PINT3		
168	J19	75	P1.4	PINT2		
169	G20	76	P1.3	PINT1		
170	J17	77	P1.2	PINT0		
171	G19	78	P1.1	VREF1/WP2		
172	J18	79	P1.0	VREF0/WP1		
173	F20	80	VREF			
174	H18	81	VREFINT			
175	F19	82	AVDD			
176	H17	83	AVSS			
177	E20	84	BVDD			
178	G18	85	WAIT			
179	E19	86	WAITH			
180	F18	87	P0.7			
181	D20	88	P0.6	P06COMP		
182	G17	89	P0.5			
183	D19	90	P0.4			
184	E18	91	P0.3			
185	C20	92	P0.2			
186	F17	93	P0.1			
187	C19	94	P0.0	CC4-IN		
188	D18	95	P2.1			
189	B20	96	P2.0			
190	E17	97	U6.2		GWEQ	SEG6.2
191	B19	98	U6.1	CAN1-RX/WP7	GOEQ	SEG6.1
192	C18	99	U6.0		CAN1-TX	SEG6.0

Table 2–1: Pin assignment for **CPGA257** package

CPGA257		PQFP128	Pin Functions			
Pin No.	Co-ord.	Pin No.	Basic Function	Port Special In	Port Special Out	LCD Mode
193	A20	100	U8.5	CAN2-RX/PINT3/WP8	LCD-SYNC-OUT	SEG8.5
194	D17	101	U8.4	LCD-SYNC-IN	CAN2-TX	SEG8.4
195	B18	102	U8.3	CAN3-RX/WP9	LCD-CLK-OUT	SEG8.3
196	C17	103	U8.2	LCD-CLK-IN	CAN3-TX	SEG8.2
197	A19	104	U8.1		CC3-OUT	SEG8.1
198	D16	105	U8.0		CC4-OUT	SEG8.0
199	B17	106	U4.3	CAN0-RX/WP5	TO2	BP3
200	C16	107	U4.2		CAN0-TX	BP2
201	A18	108	U4.1	CC0-IN	SPI1-D-OUT	BP1
202	D15	109	U4.0	SPI1-D-IN	CC0-OUT	BP0
203	B16	110	U3.7	SPI1-CLK-IN	SPI1-CLK-OUT	SEG3.7
204	C15	111	U3.6		SPI0-D-OUT	SEG3.6
205	A17	112	U3.5	SPI0-D-IN	TO3	SEG3.5
206	D14	113	U3.4	SPI0-CLK-IN	SPI0-CLK-OUT	SEG3.4
207	B15	114	U3.3		CO0/TDO	SEG3.3
208	C14	115	U3.2	CC0-IN / TCK	CC0-OUT	SEG3.2
209	A16	116	U3.1	CC1-IN / TMS	CC1-OUT	SEG3.1
210	D13	117	U3.0	CC2-IN / TDI	CC2-OUT	SEG3.0
211	B14	118	TEST2			
212	C13	119	UVDD1			
213	A15	120	UVSS1			
214	C12		TRACEPKT0 / TBIT			
215	B13		TRACEPKT1 / nM0			
216	D12		TRACEPKT2 / nM1			
217	A14		TRACEPKT3 / nM2			
218	B12		TRACEPKT4 / nM3			
219	A13		TRACEPKT5 / nM4			
220	C11		EVDD1			
221	A12		EVSS1			
222	D11		TRACEPKT6 / LOCK			
223	A11		TRACEPKT7 / nEXEC			
224	B11		TRACEPKT8 / nOPC			
225	A10		TRACEPKT9 / nTRANS			
226	B10		TRACEPKT10 / A5			
227	A9		TRACEPKT11 / A6			
228	D10		TRACEPKT12 / RANGEOUT0			
229	A8		TRACEPKT13 / RANGEOUT1			
230	C10		TRACEPKT14 / A7			
231	B8		TRACEPKT15 / BREAKPT			
232	B9		PIPESTAT0 / nRW			
233	A7		PIPESTAT1 / A0			
234	D9		PIPESTAT2 / A1			
235	B7		EVDD0			
236	C9		EVSS0			
237	A6		TRACESYNC/A2			
238	C8		TRACECLK/A3			
239	B6		EXTTRIG/A4			
240	D8		FSYS			
241	A5		nWAIT			
242	C7		DBGACK			
243	B5		DBGRQ			
244	C6		UVDD			
245	A4		UVSS			
246	D7	121	U2.6	DIGIT-IN	DIGIT-OUT	SEG2.6
247	B4	122	U2.5	UART0-RX	CC1-OUT	SEG2.5
248	C5	123	U2.4	CC1-IN/DIGIT-IN	UART0-TX	SEG2.4
249	A3	124	U2.3	UART1-RX	CC2-OUT	SEG2.3
250	D6	125	U2.2	CC2-IN	UART1-TX	SEG2.2
251	B3	126	U7.7/GD7		CO0	SEG7.7
252	C4	127	U7.6/GD6		CO1	SEG7.6
253	A2	128	U7.5/GD5		LCK/PFM1	SEG7.5
254	D5	1	U7.4/GD4	CC5-IN	CC5-OUT	SEG7.4
255	B2	2	FVDD			
256	C3	3	FVSS			
257	E5		Extra insertion Pin: connect to system ground			

2.2. Package Outline Dimensions

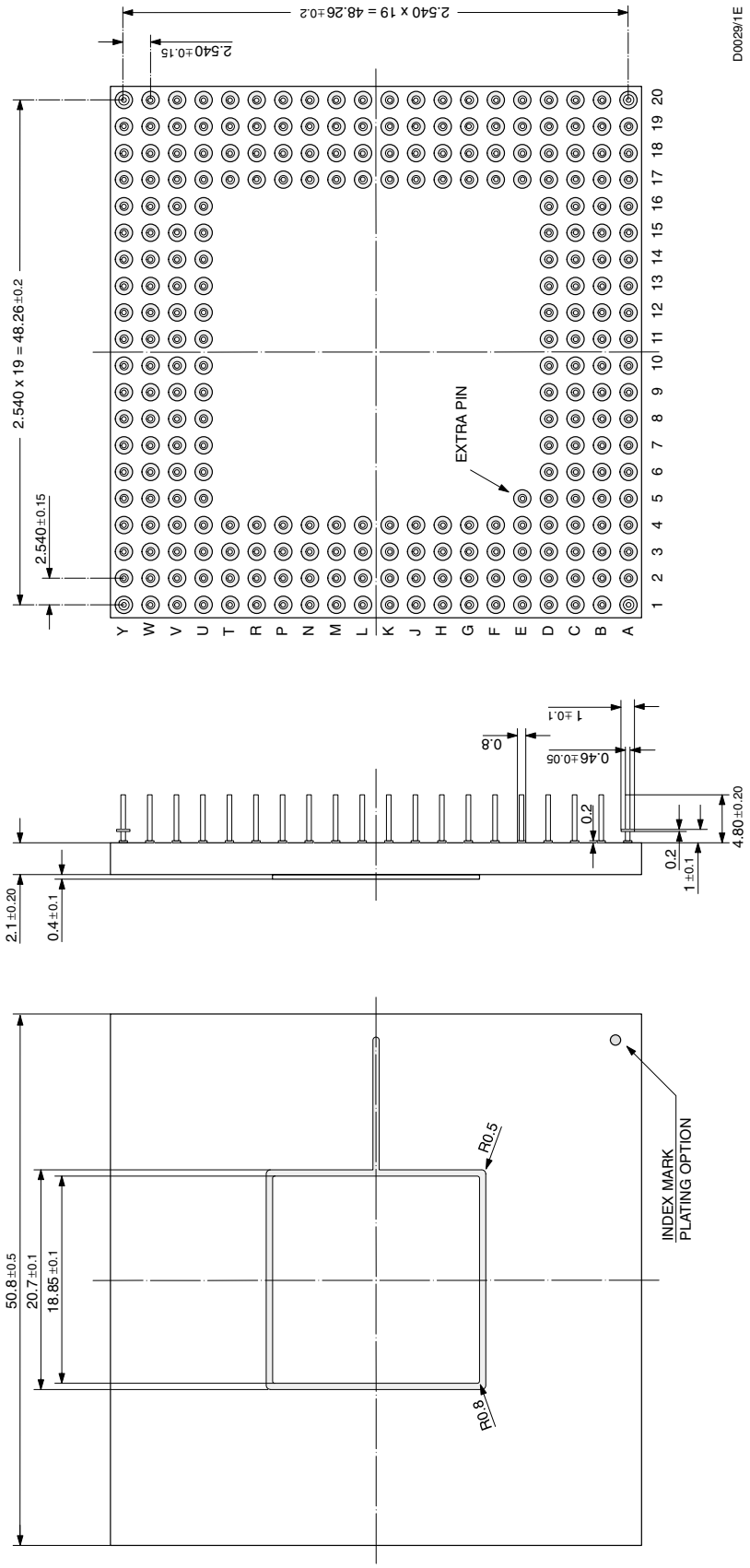


Fig. 2–2: CPGA257 ceramic pin grid array 257-pin (weight approx. 32 g. Dimensions in mm)

2.3. Multiple-Function Pins

2.3.1. U-Ports

Apart from their basic function (digital I/O), universal ports (with prefix “U”) have overlaid alternative functions
(see Table 2–1 on page 14).

How to enable basic function, special in and special out mode is explained in the functional description of the U-ports. How to enable LCD mode is explained in the functional descriptions of LCD module and U-ports.

2.3.2. H-Ports

Apart from their basic function (digital I/O), high current ports (prefix “H”) have overlaid alternative functions
(see Table 2–1 on page 14).

How to enable basic function, special in and special out mode is explained in the functional description of the H-ports.

2.3.3. Emulator Bus

In contrast to the PQFP128 standard package, the CPGA257 package has additional pins (emulator bus) which serve as memory interface, emulation JTAG interface or connection to an external emulation or trace hardware (trace bus).

The functionality of the memory interface and the trace bus is controlled by register CR. Refer to section “Control Word” for more information.

Some of the following pins are marked as being ARM or ETM signals. For details of the functionality please refer to ARM7TDMI data sheet (document number: ARM DDI 0029) or “Embedded Trace Macro Cell” (document number: ARM IHI 0014 and ARM DDI 0158).

2.4. Pin Function Description

A0 to A7 (ARM) 1)
A8 to A23 (ARM) 4)
These 24 lines are the original CPU addresses. Some are used for external memory access on the emulator bus. The function is controlled by register CR.

ABE (ARM) 1) 2)
This pin outputs the “address bus enable” signal of the ARM. It indicates that the CPU does not access the data and address bus when low. It is not possible to influence the CPU via this pin.

ABORT (ARM) 3)
This is an input which allows the memory system to tell the processor that a requested access is not allowed.

AICU2 to AICU7 4)
These pins correspond to the ARM address bus lines A2 to A7, but can be modified by the ICU. In the latter case, AICUx and Ax are not equal.

ALARMQ
This is the second input comparator level on the RESETQ pin.

AMCM21 to AMCM23 4)
These pins correspond to the ARM address bus lines A21 to A23 but can be modified by the memory controller. In the latter case, AMCMx and Ax are not equal.

AMCS1 4)
This pin corresponds to the ARM address bus line A1, but can be modified by the memory controller. In the latter case, AMCS1 and A1 are not equal.

AM-OUT
This is the output signal of the audio module.

AM-PWM
This is the output signal of the 8-bit PWM of the audio module. It is intended for testing only.

AVDD
This is the positive power supply for ADC, P06COMP, WAITCOMP and BVDD regulator. AVDD should be kept at UVDD ±0.5 V. It must be buffered by an external capacitor to analog ground

AVSS
This is the negative reference for the ADC and the negative power supply for ADC, P06COMP, WAITCOMP and PLL. Connect to analog ground.

BP0 to BP3
These pin functions serve as backplane drivers for a 4:1 multiplexed LCD.

BREAKPT (ARM) 3)
This is the input pin for the ARM BREAKPT signal in “full trace” mode. It allows external hardware to halt the execution of the processor for debugging purposes.

BVDD
This is the output of the internal 2.5 V regulator for the PLL. It must be buffered by an external capacitor to analog ground.

BWQ0 to BWQ3 4)
This is the byte write control signal to an external 32-bit memory.

CAN0-RX, CAN1-RX, CAN2-RX, CAN3-RX
These signals provide the input lines for the CAN0, CAN1, CAN2 and CAN3 modules.

CAN0-TX, CAN1-TX, CAN2-TX, CAN3-TX
These signals provide the output lines for the CAN0, CAN1, CAN2 and CAN3 modules.

CC0-IN, CC1-IN, CC2-IN, CC3-IN, CC4-IN, CC5-IN
These signals are the capture inputs of the CAPCOM0 to CAPCOM5 modules.

CC0-OUT, CC1-OUT, CC2-OUT, CC3-OUT, CC4-OUT, CC5-OUT
These signals are the compare outputs of the CAPCOM0 to CAPCOM5 modules.

CE0Q 4)
The “Chip Enable” output signal connects to external program memory’s CEQ pin. With CR.EFLA set, it serves to reduce program memory’s power consumption when CPU operates in slow mode. Active LOW.

CE1Q 4)
The “chip enable” output signal connects to external RAM or Boot ROM memory’s CEQ pin and reduces its power consumption when CPU operates in slow mode. Active LOW.

CO0, CO0Q, CO1
These signals provide frequency outputs. They are connected to internal prescaler and multiplexer. They can be hardwired by HW Option. Refer to section “Hardware Options” for setting the CO0/CO1 options and section “CPU and Clock System” setting the “clock out 0” selection register.

For testing purposes, it is possible to drive clocks and other signals of internal peripheral modules out of CO0 and CO1. Selection is done via register TST2.

D0 to D31 (ARM) 4)
These 32 signals are the original CPU bidirectional data bus lines. They provide the 32-bit data bus for use during data exchanges between the microprocessor and external memory or peripherals.

DBGACK (ARM)
This is the debug acknowledge output signal of the ARM. A high state indicates that ARM is in debug state.

DBGRQ (ARM) 3)
This is the “debug request input” of the ARM. It is a level-sensitive input, which when high causes ARM to enter debug state after executing the current instruction.

DIGIT-IN
This is the receive input line of the DIGITbus module.

DIGIT-OUT
This is the transmit output line of the DIGITbus module.

EMUTRI
This input signal allows to tristate (= high) the interface pins to external memory (A8 to A23, AMCS1, AICU2 to AICU7, AMCM21 to AMCM23, CE1Q, FBUSQ and WEQ/RWQ).

ETCK (ARM)
This pin is the ARM “test clock input” (TCK) of the emulation JTAG interface.

ETDI (ARM)
This pin is the ARM “test data input” (TDI) of the emulation JTAG interface.

ETDO (ARM)
This pin is the ARM “test data output” (TDO) of the emulation JTAG interface.

ETMS (ARM)
This pin is the ARM “test mode select” (TMS) input of the emulation JTAG interface.

EVDD0 to EVDD8
These nine lines form the positive power supply of the emulator and trace bus drivers. EVDD0 to EVDD8 may be connected to any voltage between 3 to 5.5 V. Normally they are connected to FVDD.

EVSS0 to EVSS8
These nine lines form the negative supply of the emulator bus and trace drivers. EVSS0 to EVSS8 have to be hardwired to system ground.

EXTERN0, EXTERN1 (ARM) 3)
These are inputs to the ICEBreaker logic of the ARM which allows breakpoints and/or watch points to be dependent on an external condition.

EXTTRIG (ETM) 2)
This is a trigger input to the ETM.

FBUSQ 4)
This signal is the reference for access to external synchronous memory. It is active for memory access only.

FSYS
This signal provides the system frequency clock f_{SYS}. It is the PLL output frequency if PLL is enabled.

FVDD
This is the output of the internal 3.3 V regulator for the external Flash chip. It must be buffered by an external capacitor to FVSS.

FVSS
This is the ground reference of the internal 3.3 V regulator for the external Flash chip.

GD0 to GD7
These eight graphics IC data lines provide an 8-bit DMA-controlled data link to an external IC.

GOEQ
This graphics IC read line provides the control signal for read accesses via the GD7 to GD0 bus. Active LOW.

GWEQ
This graphics IC write line provides the control signal for write accesses via the GD7 to GD0 bus. Active LOW.

H0.0 to H7.3
The high current ports are intended for use as digital I/O which can drive higher currents than the universal ports.

HVDD0 to HVDD3
The pins HVDD0 to HVDD3 are the positive power supply of the high current ports H0.0 to H7.3. HVDD0 to HVDD3 should be kept at UVDD ±0.5 V. Be careful to design the PCB traces for carrying the considerable operating current on these pins.

HVSS0 to HVSS3
The pins HVSS1 to HVSS3 are the negative power supply for the high-current ports H0.0 to H7.3. HVSS0 to HVSS3 have to be hardwired to system ground. Be careful to layout sufficient PCB traces for carrying the considerable operating current on these pins.

INTRES
Test output of internal reset signal. Only for testing and available only in test mode.

ITSTIN
Test input signal for interrupt controller. Only for testing and available only in test mode.

ITSTOUT
Test output signal of internal peripheral modules. Only for testing and available only in test mode.

LCD-CLK-IN
The clock input of the LCD module receives the clock of an optional external LCD master driver which is used to extend

the LCD driver capability. This input is active if the internal LCD module is configured as slave and the external LCD driver operates as master.

LCD-CLK-OUT
The clock output of the LCD module provides a clock signal to optional external LCD slave drivers if the internal LCD module is configured as master and the other LCD drivers are slaves.

LCD-SYNC-IN
The synchronization input of the LCD module receives the sync signal from an optional external LCD master driver. This input is active if the internal LCD module is configured as slave and the external LCD driver serves as master.

LCD-SYNC-OUT
The synchronization output of the LCD module provides a sync signal to optional external LCD slave drivers if the internal LCD module is configured as master and the other LCD drivers are slaves.

LCK
This output signal indicates that the PLL has locked.

LOCK (ARM) 1)
This is the LOCK output signal of the ARM indicating that the processor is performing a “locked” memory access when high.

MAS0, MAS1 (ARM) 1) 2)
These are ARM output signals used by the processor to indicate to the external memory system when a word transfer or a half-word or a byte length is required.

MTI
This is a test input line. It is intended for factory test only. The application should not use this signal.

MTO
This is a test output line. It is intended for factory test only. The application should not use this signal.

nEXEC (ARM) 1)
This is the “not executed” signal of the ARM indicating that the instruction in the execution unit is not being executed when high.

nM0 to nM4 (ARM) 1)
These pins output the “not processor mode” signal of the ARM.

nMREQ (ARM) 1) 2)
This pin outputs the “not memory request” signal of the ARM. The processor requires memory access during the following cycle when low.

nOPC (ARM) 1)
This pin outputs the “not op-code fetch” signal of the ARM. The processor is fetching an instruction from memory when low.

nRESET (ARM)
This pin outputs the “not reset” signal of the ARM. This pin is not an input.

nRW (ARM) 1)
This pin outputs the “not read/write” signal of the ARM. High indicates a processor write cycle, low a read cycle.

nTRANS (ARM) 1)
This pin outputs the “not memory translate” signal of the ARM. When low, it indicates that the processor is in user mode.

nTRST (ARM)
This pin is the “not test reset” signal of the ARM. It resets the boundary scan logic of the CPU when low. It is also the reset for the Emulation JTAG interface (not for the application JTAG interface).

nWAIT (ARM) 1) 2)
This pin outputs the “not wait” signal of the ARM. It is not possible to cause a wait via this pin.

OEQ 4)
The Output Enable signal connects to the OEQ pin of external memory for read access. Active LOW.

P0.0 to P0.7, P1.0 to 1.7 and P2.0 to P2.1
P0.0 to P1.7 are 16 analog ports that are the multiplexed input channels of the ADC. All analog ports P0.0 to P2.1 can also be used as digital input lines. The analog ports P1.2 to P1.7 can also be used as port interrupts.

P06COMP
Analog port P0.6 is additionally input to the P06 comparator.

PFM0, 1
These are the outputs of the PFM0 and PFM1 pulse frequency modulators.

PINT0 to PINT5
The port interrupt 0 to 5 inputs serve as inputs to the interrupt controller via the port interrupt module. HW option PM.PINT has to be set to determine which of the possible input pins are used as source of PINT0 to 5.

PIPESTAT0 to PIPESTAT2 (ETM) 2)
These signals indicate the pipeline status of the ETM.

POL
Output of the polling module.

PWM0 to PWM11
These are the outputs of the PWM module. Some of these PWM signals are directed to two pins.

RANGEOUT0, RANGEOUT1 (ARM) 1)
These pins output the “ICEBreaker rangeout” signals of the ARM. They indicate that ICEBreaker watch point register 0 or 1 has matched the conditions currently present on the address, data and control busses.

RESETQ
This bidirectional signal is used to initialize all modules and start program execution.

Two comparators distinguish three input levels:

– A low level resets all internal modules.

– A medium level activates all internal modules and starts program execution. An alarm signal is generated which can be directed to the interrupt controller.

– A high level keeps all internal modules active and cancels the alarm signal.

The RESETQ input signal must be held low for at least two clock cycles after VDD reaches operating voltage.

Internal reset sources output their reset request on the RESETQ pin via an internal open drain pull-down transistor. Thus RESETQ can be wire-ored with external reset sources. The internally limited pull-down current allows direct connection to large capacitors. The connection of such a capacitor (e.g. 10 nF) is recommended to reduce the capacitive influence of the neighboring XTAL2 pin.

RESETQ must be pulled up by an external pull-up resistor (e.g. 10 kΩ).

RWQ 4)
This is an interface signal to external memory.

SCL0 to SCL1
These are the serial clock lines of the I2C modules.

SDA0 to SDA1
These are the serial data lines of the I2C modules.

SEG0.0 to SEG8.5
These pin functions serve as segment drivers for a 4:1 multiplexed LCD.

SEQ (ARM) 1) 2)
This pin outputs the “sequential address” signal of the ARM. High indicates that the address of the next memory cycle will be related to that of the last memory access.

SMA to SMG
These lines are intended for driving stepper motors. They are the outputs of the SM. Two of these lines together with an external coil form an H-bridge. Thus each of the signals SMA to SMG can drive a two-phase bipolar stepper motor.

SMA-COMP0, 1, 2, 3 to SMG-COMP0, 1, 2, 3
These lines are comparator inputs that connect to one line each of the SMA to SMG lines. They serve to distinguish rotation from stand-still during zero detection in each stepper motor.

SPI0-CLK-IN, SPI1-CLK-IN
The serial synchronous peripheral interface clock input receives the bit clock from an external master, to shift data in or out of SPI0 resp. SPI1 in slave mode. This means that the external master controls the bit stream.

SPI0-CLK-OUT, SPI1-CLK-OUT
The serial synchronous peripheral interface clock output supplies the bit clock of SPI0 resp. SPI1 to an external slave, to shift data in or out of SPI0 resp. SPI1 in master mode. This means that the internal SPI controls the bit stream.

SPI0-D-IN, SPI1-D-IN
These are the data input lines of the SPI0 and SPI1 modules.

SPI0-D-OUT, SPI1-D-OUT
These are the data output lines of the SPI0 and SPI1 modules.

T0-OUT
The Timer 0 output is connected to the zero output of T0 by a divide-by-2 scaler. The scaler generates a 50% pulse duty factor.

T1-OUT to T4-OUT
These signals are connected to the overflow outputs of T1 to T4.

TBIT (ARM) 1)
This pin outputs the TBIT signal of the ARM. High indicates that the processor is executing the THUMB instruction set.

TCK (ARM)
This pin is the ARM “test clock” input of the application JTAG interface.

TDI (ARM)
This pin is the ARM “test data input” of the application JTAG interface.

TDO (ARM)
This pin is the ARM “test data output” of the application JTAG interface.

TEST, TEST2
Pins TEST and TEST2 define the source for the control word fetch during reset. Please refer to section “Core Logic” for detailed information.

TEST2 serves to enable the JTAG interface. Refer to section “JTAG Interface“ for detailed information.

For normal operation with internal code connect TEST and TEST2 to system ground or leave it floating (internal pull-down).

TMS (ARM)
This pin is the ARM “test mode select” input of the application JTAG interface.

TO2 and TO3
Test outputs.

TRACECLK (ETM) 2)
This is the output of the modified CLK signal of the ETM.

TRACEPKT0 to TRACEPKT15 (ETM) 2)
This is the trace packet port of the ETM.

TRACEPKT15 is pulled low to prevent floating, when full trace mode is enabled.

TRACESYNC (ETM) 2)
This is the synchronization signal from the ETM, indicating the start of a branch sequence on the trace packet port.

U0.0 to U8.5
Universal ports are intended for use as digital I/O or as LCD driver outputs.

UART0-RX, UART1-RX
These are the receive input lines of UART0 and UART1. Polarity of the signals is settable by HW options UA0 resp. UA1.

UART0-TX, UART1-TX
These are the data output lines of UART0 and UART1. Polarity of signals can be set by HW options UA0, resp. UA1.

UVDD, UVDD1
The pins UVDD and UVDD1 are the positive 5 V supply for the U-Port output stages, for the VDD regulator and the FVDD regulator (see Fig. 2–3 for external connection). It must be buffered by an external capacitor to UVSS, resp. UVSS1.

UVSS, UVSS1
The pins UVSS and UVSS1 are the negative power supply for the U-Port output stages, and the ground reference for the VDD and FVDD regulators. They have to be connected to system ground (see Fig. 2–3).

VDD
This is the output of the internal 2.5V regulator for the internal digital modules (see Fig. 2–3 for external connection). It must be buffered by an external capacitor to VSS.

VREF, VREF0, VREF1
These pins are selectable as positive reference inputs for the ADC. The voltage on these pins should be set to a level between 2.56 V and AVDD.

VREFINT
This pin is the positive reference output of the ADC. The voltage at this pin is generated internally (approx. 2.5 V) and

must be buffered by an external capacitor to AVSS. No DC load is allowed.

VSS
The pin VSS is the negative supply terminal of the internal digital modules (see Fig. 2–3 for external connection).

WAIT
This is the positive input to the WAIT comparator. The negative input is VREFINT. The comparator level can be adjusted by an external voltage divider.

WAITH
This is the output of the WAIT comparator. The hysteresis can be adjusted by an external feedback resistor to the voltage divider connected to the WAIT pin.

WEQ 4)
The output signal Write Enable connects to the external memory's WEQ pin and activates it for write access. Active LOW.

WP0 to WP9
The wake port inputs are inputs to the port wake module inside the power-saving module. They serve as wake ports during power-saving modes and as port interrupt inputs during CPU-active modes.

XTAL1
This is the quartz oscillator or clock input pin (see Fig. 2–3 for external connection).

XTAL2
This is the quartz oscillator output pin for two pin oscillator circuits (see Fig. 2–3 for external connection).

1) Trace Bus output. Active in analyzer mode.
2) Trace Bus output. Active in ETM mode.
3) Trace Bus input. Always active.
4) Memory interface signal. Tristate if EMUTRI is high.
Please refer to section "Memory Interface" (see Table 34–1 on page 250) for details on interfaces and Trace Bus modes.

2.5. External Components

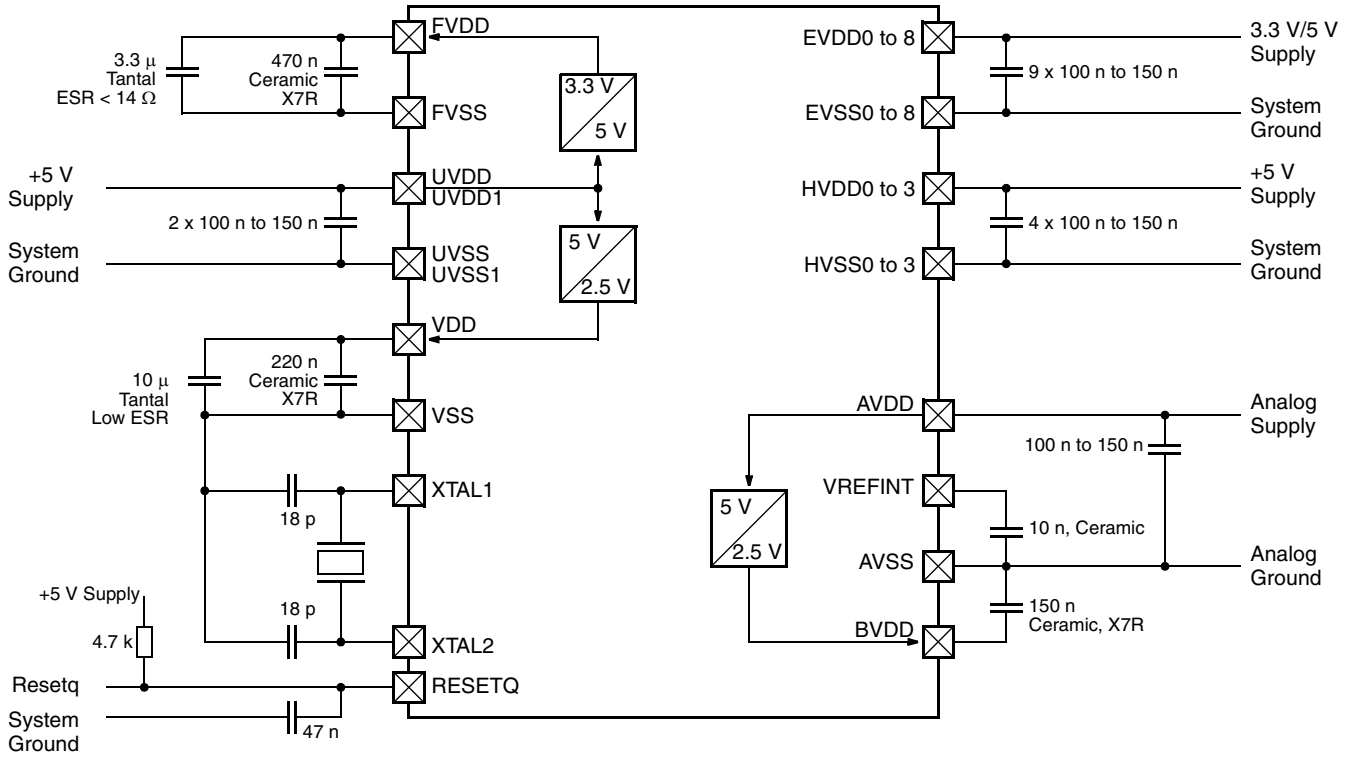


Fig. 2–3: CDC3205G-C: Recommended external supply and quartz connection.

To provide effective decoupling and to improve EMC behavior, the small decoupling capacitors must be located as close to the supply pins as possible. The self-inductance of these capacitors and the parasitic inductance and capacitance of the interconnecting traces determine the self-resonant frequency of the decoupling network. Too low a frequency will reduce decoupling effectiveness, will increase RF emissions and may adversely affect device operation.

XTAL1 and XTAL2 quartz connections are especially sensitive to capacitive coupling from other PC board signals. It is

strongly recommended to place quartz and oscillation capacitors as close to the pins as possible and to shield the XTAL1 and XTAL2 traces from other signals by embedding them in a VSS trace.

The RESETQ pin adjacent to XTAL2 should be supplied with a 47 nF capacitor, to prevent fast RESETQ transients from being coupled into XTAL2, to prevent XTAL2 from coupling into RESETQ, and to guarantee a time constant of ≥200 µs sufficient for proper wake reset functionality.

2.6. Pin Circuits

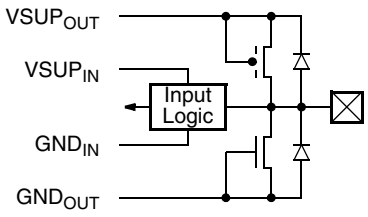


Fig. 2–4: Input pins

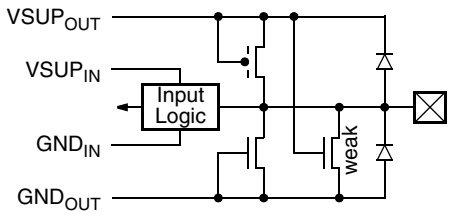


Fig. 2–5: Input pins with pull-down

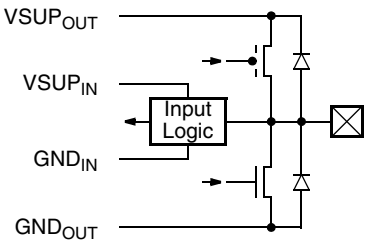


Fig. 2–6: Push-pull I/O pins

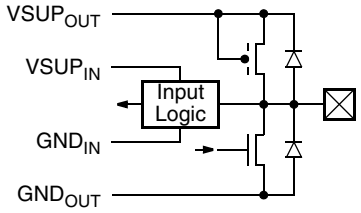


Fig. 2–7: Open-drain I/O

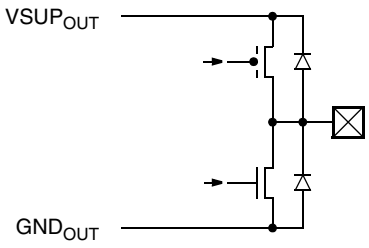


Fig. 2–8: Push-pull output pins

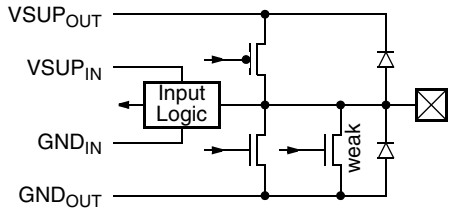


Fig. 2–9: Push-pull I/O pins with switchable pull-down

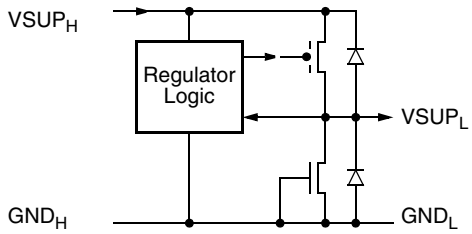


Fig. 2–10: Regulator pins

Table 2–2: I/O supply catalog

Pin Names	Figure	$VSUP_{OUT}$	GND_{OUT}	$VSUP_{IN}$	GND_{IN}
XTAL1, XTAL2	2–4	UVDD	UVSS	UVDD	UVSS
WAIT		AVDD	AVSS	AVDD	AVSS
TCK, TDI, TMS		EVDD	EVSS	EVDD	EVSS
EMUTRI, ABORT, EXTERN0, EXTERN1, DBGRQ	2–5				
TEST, TEST2		UVDD	UVSS	UVDD	UVSS
U-Ports	2–6				
H-Ports		HVDD	HVSS	HVDD	HVSS
P-Ports		AVDD	AVSS	AVDD	AVSS
A31 to A0, ABE, AICU7 to 2, AMCM21 to 23, AMCS1, BWQ0 to 3, CE0Q, CE1Q, D31 to D0, DBGACK, EXTTRIG, FBUSQ, FSYS, MAS0, MAS1, nMREQ, nRESET, nTRST, nWAIT, OEQ, PIPSTAT0 to 2, SEQ, TDO, TRACECLK, TRACEPKT0 to 14, TRACESYNC		EVDD	EVSS	EVDD	EVSS
RESETQ	2–7	UVDD	UVSS	UVDD	UVSS
WAITH	2–8	AVDD	AVSS		
TRACEPKT15	2–9	EVDD	EVSS	EVDD	EVSS

Table 2–3: Regulator pin supply catalog

Regulator	Figure	$VSUP_H$	GND_H	$VSUP_L$	GND_L
VDD	2–10	UVDD	UVSS	VDD	VSS
FVDD		UVDD	UVSS	FVDD	FVSS
BVDD		AVDD	AVSS	BVDD	AVSS

3. Electrical Data

3.1. Absolute Maximum Ratings

Stresses beyond those listed in the “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only. Functional operation of the device at these conditions is not implied. Exposure to absolute maximum ratings conditions for extended periods will affect device reliability.

This device contains circuitry to protect the inputs and outputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than absolute maximum-rated voltages to this high-impedance circuit.

Table 3–1: All voltages listed are referenced to ground (UV_{SS} = UV_{SS1} = HV_{SSn} = AV_{SS} = 0 V), except where noted. All grounds except VSS must be connected to a low-resistive ground plane close to the IC.

Symbol	Parameter	Pin Name	Min.	Max.	Unit
V_{SUP}	Main supply voltage Analog supply voltage SM supply voltage Flash port supply voltage	UVDD, UVDD1 AVDD HVDDn EVDDn	–0.3	6.0	V
V_{REG}	Flash supply voltage	FVDD	–0.3	4.0	V
	Core supply voltage PLL supply voltage	VDD BVDD	–0.3	3.0	V
I_{SUP}	Core supply current Main supply current	VDD, VSS, UVDD, UVDD1, UVSS, UVSS1	–100	100	mA
	Flash port supply current	EVDDn EVSSn	–100	100	mA
	Analog supply current	AVDD, AVSS	–20	20	mA
	SM supply current @T_{CASE} = 105 °C, duty factor = 0.71 [1)]	HVDDn HVSSn	–250	250	mA
	Flash supply current	FDD, FVSS	–50	50	mA
	PLL supply current	BVDD	–20	20	mA
V_{in}	Input voltage	U ports, XTAL, RESETQ, TEST, TEST2	UV_{SS} – 0.5	UV_{DD} + 0.7	V
		P ports, VREF	UV_{SS} – 0.5	AV_{DD} + 0.7	V
		H ports	HV_{SS} – 0.5	HV_{DD} + 0.7	V
		E ports	EV_{SS} – 0.5	EV_{DD} + 0.7	V
I_{in}	Input current	all inputs	0	2	mA
I_o	Output current	U ports, E ports, RESETQ, WAITH	–5	5	mA
		H ports	–60	60	mA
t_{oshsl}	Duration of short circuit to UVSS or UVDD, Port SLOW mode enabled	U ports, except in DP mode		indefinite	s
T_j	Junction temperature under bias		–45	115	°C

[1)] This condition represents the worst case load with regard to the intended application.

Table 3–1: All voltages listed are referenced to ground (UV_{SS} = UV_{SS1} = HV_{SSn} = AV_{SS} = 0 V), except where noted. All grounds except VSS must be connected to a low-resistive ground plane close to the IC.

Symbol	Parameter	Pin Name	Min.	Max.	Unit
T_s	Storage temperature		–45	125	°C
P_{max}	Maximum power dissipation			0.8	W

3.2. Recommended Operating Conditions

Do not insert the device into a live socket. Instead, apply power by switching on the external power supply.

Keep UV_{DD} = UV_{DD1} = AV_{DD} during all power-up and power-down sequences.

Failure to comply with the above recommendations will result in unpredictable behavior of the device and may result in device destruction.

Functional operation of the device beyond those indicated in the “Recommended Operating Conditions” of this specification is not implied, may result in unpredictable behavior of the device and may reduce reliability and lifetime.

Table 3–2: All voltages listed are referenced to ground (UV_{SS} = UV_{SS1} = HV_{SSn} = AV_{SS} = 0 V), except where noted. All grounds except VSS must be connected to a low-resistive ground plane close to the IC.

Symbol	Parameter	Pin Name	Min.	Typ.	Max.	Unit
V_{SUP}	Main supply voltage Analog supply voltage	UVDD = UVDD1 = AVDD	3.5	5	5.5	V
	Flash port supply voltage	EVDDn	3		5.5	V
HV_{SUP}	SM supply voltage	HVDDn	4.75	5	5.25	V
dV_{DD}	Ripple, peak-to-peak	UVDD AVDD BVDD FVDD VDD			200	mV
dV_{DD}/dt	Supply voltage up/down ramping rate	UVDD AVDD			20	V/µs
f_{XTAL}	XTAL clock frequency	XTAL1	4	4	5	MHz
f_{SYS}	CPU clock frequency, PLL on		For a list of available settings see Tables 4–6 and 4–7.			
f_{BUS}	Program storage clock frequency, PLL on					
V_{il} (see Table 2-2 for a list of input types and their supply voltages)	Automotive low input voltage	U ports H ports P ports			0.5 × xV_{DD}	V
	CMOS low input voltage	U ports, TEST, TEST2 H ports P ports			0.3 × xV_{DD}	V
	TTL low input voltage	E ports			0.8	V

Table 3–2: All voltages listed are referenced to ground (UV_{SS} = UV_{SS1} = HV_{SSn} = AV_{SS} = 0 V), except where noted. All grounds except VSS must be connected to a low-resistive ground plane close to the IC.

Symbol	Parameter	Pin Name	Min.	Typ.	Max.	Unit
V_{ih} (see Table 2-2 for a list of input types and their supply voltages)	Automotive high input voltage	U ports H ports P ports	0.86 × xV_{DD}			V
	CMOS high input voltage	U ports,TEST, TEST2 H ports P ports	0.7 × xV_{DD}			V
	TTL high input voltage	E ports	2.2			V
RV_{il}	Reset active input voltage	RESETQ			0.75	V
WRV_{il}	Reset active input voltage during power-saving modes and wake reset	RESETQ			0.4	V
RV_{im}	Reset inactive and alarm active input voltage	RESETQ	1.5		2.3	V
RV_{ih}	Reset inactive and alarm inactive input voltage	RESETQ	3.2			V
WRV_{ih}	Reset inactive input voltage during power-saving modes and wake reset	RESETQ	UV_{DD} – 0.4 V			V
V_{REFi}	Ext. ADC reference input voltage	VREF	2.56		AV_{DD}	V
PV_i	ADC port input voltage referenced to ext. VREF reference ADC port input voltage referenced to int. VREFINT reference	P ports	0 0		V_{REF} V_{REFINT}	V

3.3. Characteristics

Table 3–3: UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS} = 0 V, 3.5 V < AV_{DD} = UV_{DD} = UV_{DD1} < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
Package							
R_{thjc}	Thermal resistance from junction to case			15		K/W	

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
Supply Currents (CMOS levels on all inputs, i.e. $V_{il} = xV_{SS} \pm 0.3$ V and $V_{ih} = xV_{DD} \pm 0.3$ V, no loads on outputs)							
UI_{DDp}	UVDD PLL mode supply current, ETM off	UVDD + UVDD1			55 100	mA	f_{SYS} = 24 MHz f_{SYS} = 50 MHz
UI_{DDpe}	UVDD PLL mode supply current, ETM on	UVDD + UVDD1			65 110	mA	f_{SYS} = 24 MHz f_{SYS} = 50 MHz
UI_{DDf}	UVDD FAST mode supply current	UVDD + UVDD1			15	mA	all modules off [2]
UI_{DDs}	UVDD SLOW mode supply current	UVDD + UVDD1			1.3	mA	all modules off, [2] [6]
UI_{DDd}	UVDD DEEP SLOW mode supply current	UVDD + UVDD1			0.8	mA	all modules off, [6]
UI_{DDw}	UVDD WAKE mode supply current	UVDD + UVDD1	0	20	50	µA	RC and XTAL oscillators off
UI_{DDst}	UVDD STANDBY mode supply current	UVDD + UVDD1		35	75	µA	RC oscillator on, XTAL off
				60	100	µA	XTAL oscillator on, RC off [6]
UI_{DDi}	UVDD IDLE mode supply current	UVDD + UVDD1		50	775	µA	RC oscillator on, XTAL off
				75	800	µA	XTAL oscillator on, RC off [6]
AI_{DDa}	AVDD active supply current	AVDD		0.35	0.6	mA	ADC on, PLL off
					2	mA	ADC, buffer and PLL on
AI_{DDq}	Quiescent supply current	AVDD	0		10	µA	ADC and PLL off
EI_{DDq}		EVDDn	0		10	µA	no output activity
HI_{DDq}		Sum of all HVDDn	0		40	µA	no output activity, SM module off

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
Inputs							
V_{ilha}	Automotive input low to high threshold voltage	U ports H ports P ports	0.68 × xV_{DD}	0.76 × xV_{DD}	0.84 × xV_{DD}	V	4.5V < xV_{DD} < 5.5V, [3] (see Table 2-2 for a list of input types and their supply voltages)
V_{ihla}	Automotive input high to low threshold voltage		0.53 × xV_{DD}	0.61 × xV_{DD}	0.69 × xV_{DD}	V	
V_{ilha}-V_{ihla}	Automotive input hysteresis		0.1 × xV_{DD}	0.15 × xV_{DD}	0.2 × xV_{DD}	V	
V_{ilhc}	CMOS input low to high threshold voltage	U ports H ports P ports TEST, TEST2	0.5 × xV_{DD}	0.6 × xV_{DD}	0.7 × xV_{DD}	V	4.5V < xV_{DD} < 5.5V, [3] (see Table 2-2 for a list of input types and their supply voltages)
V_{ihlc}	CMOS input high to low threshold voltage		0.3 × xV_{DD}	0.4 × xV_{DD}	0.5 × xV_{DD}	V	
V_{ilhc}-V_{ihlc}	CMOS input hysteresis		0.1 × xV_{DD}	0.2 × xV_{DD}	0.3 × xV_{DD}	V	
I_i	Input leakage current	U ports H ports P ports P06, WAIT VREF E ports	–1 –10 –1 –0.2 –1 –1		1 10 1 0.2 1 1	µA	$0 < V_i < UV_{DD}$ $0 < V_i < HV_{DD}$ $0 < V_i < AV_{DD}$ $0 < V_i < AV_{DD}$ $0 < V_i < AV_{DD}$ $0 < V_i < EV_{DD}$
I_{pd}	Input pull-down current	TEST, TEST2 E ports	10 10	100 100	220 220	µA	$V_i = UV_{DD}$ $V_i = EV_{DD}$, when unused
I_{pu}	Input pull-up current	E-DB	–220	–100	–10	µA	V_i = 0, when unused
Outputs (4.5 V < $UV_{DD} = EV_{DD}$ < 5.5 V)							
V_{ol}	Port low output voltage	U ports, E ports RESETQ			0.4	V	I_o = 2 mA
		H ports	0.125		0.45	V	I_o = 27 mA I_o = 40 mA @ T_{CASE} = –40 °C I_o = 30 mA @ T_{CASE} = 25 °C
ΔV_{ol}	Difference of V_{ol} values within one SM module	H ports			50	mV	

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
V_{oh}	Port high output voltage	U ports E ports	UV_{DD} – 0.4 EV_{DD} – 0.4			V	I_o = –2 mA
		H ports	HV_{DD} – 0.55		HV_{DD} – 0.125	V	I_o = –27 mA I_o = –40 mA @ T_{CASE} = –40 °C I_o = –30 mA @ T_{CASE} = 25 °C
ΔV_{oh}	Difference of V_{oh} values within one SM module	H ports			50	mV	
dV_{oH}/dt	H-Port slew rate with inductive load	H ports	by first order approximation defined by the quotient of the inductive load current and the external capacitances on the port pin			V/ns	
LV_{ol}	LCD port zero output voltage	U ports	–0.05		0.05	V	no load
LV_{o1}	LCD port low output voltage	U ports	1 / 3 × UV_{DD} – 0.05		1 / 3 × UV_{DD} + 0.05	V	no load
LV_{o2}	LCD port high output voltage	U ports	2 / 3 × UV_{DD} – 0.05		2 / 3 × UV_{DD} + 0.05	V	no load
LV_{oh}	LCD port full output voltage	U ports	UV_{DD} – 0.05		UV_{DD} + 0.05	V	no load
ΣLI_{o1}	Internal LCD-low supply short circuit current	U ports	0.3		 –0.3	mA	pin short to 2 / 3 × UV_{DD} pin short to UV_{SS}
ΣLI_{o2}	Internal LCD-high supply short circuit current	U ports	0.3		 –0.3	mA	pin short to UV_{DD} pin short to 1 / 3 × UV_{DD}
I_{shf}	Port FAST short circuit current	U ports		14	23	mA	pin short to UV_{DD} or UV_{SS}, Port FAST mode
I_{shs}	Port SLOW short circuit current	U ports		3.7	5.5	mA	pin short to UV_{DD} or UV_{SS}, Port SLOW mode
I_{shsd}	Port SLOW short circuit current, DP mode	U ports		7.5	11	mA	pin short to UV_{DD}, Port SLOW and Double Pull-Down modes

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS} = 0$ V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
References and Comparators, AVDD Section							
V_{REFINT}	VREFINT generator reference output voltage	VREFINT	2.39		2.49	V	external load current < 1 µA
t_{REFINT}	VREFINT generator setup time after power-up on AV_{DD}, or on leaving SLOW or DEEP SLOW mode	VREFINT			500	µs	
V_{REFP06}	P06 comparator reference voltage	P06	0.49 × AV_{DD}		0.51 × AV_{DD}	V	
$P06V_{lh}$-$P06V_{hl}$	P06 comparator hysteresis, symmetrical to V_{REFP06}	P06	0.02 × AV_{DD}		0.05 × AV_{DD}	V	[3]
V_{REFW}	WAIT comparator reference voltage	WAIT	0.98 × V_{REFINT}		1.02 × V_{REFINT}	V	
VW_{ol}	WAIT comparator low output voltage	WAITH			0.4	V	4.5 V < xV_{DD} < 5.5 V I_o = 50 µA
VW_{oh}	WAIT comparator high output voltage	WAITH	AV_{DD} – 0.4 V			V	4.5 V < xV_{DD} < 5.5 V I_o = –50 µA
t_{ACDEL}	P06, WAIT comparator delay time	P06 WAIT			1	µs	overdrive = 50 mV
References and Comparators, UVDD Section							
V_{BG}	VBG generator reference output voltage		2.25	2.5	2.75	V	
V_{REFR}	RESET comparator reference voltage	RESETQ	0.45 × V_{BG}		0.45 × V_{BG}	V	
RV_{lh}-RV_{hl}	RESET comparator hysteresis, symmetrical to V_{REFR}	RESETQ	0.25		0.375	V	[3]
WRV_{ihl}	Reset active high to low threshold voltage during power-saving modes and wake reset	RESETQ	0.5		1.2	V	
V_{REFA}	ALARM comparator reference voltage	RESETQ	1.1 × V_{BG}		1.1 × V_{BG}	V	
AV_{lh}-AV_{hl}	ALARM comparator hysteresis, symmetrical to V_{REFA}	RESETQ	0.1		0.2	V	[3]
V_{REFPOR}	UV_{DD} power-on reset threshold	UVDD	1.125 × V_{BG}		1.125 × V_{BG}	V	

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
t_{UCDEL}	RESET, ALARM, comparator delay time	RESETQ			1	µs	overdrive = 50 mV
References and Comparators, HVDD Section							
V_{REFSM}	SM comparator reference voltage	H00, H04, H20, H24, H32, H40, H70	1 / 9 × HV_{DD} – 0.07		1 / 9 × HV_{DD} + 0.07	V	
t_{HCDEL}	SM comparator delay time	H00, H04, H20, H24, H32, H40, H70			100	ns	overdrive = 50 mV
VDD Regulator							
V_{DD_ro}	Regulator output voltage	VDD	1.02 × V_{BG} – 30 mV		1.02 × V_{BG} + 30 mV	V	DEEP SLOW mode
			1.02 × V_{BG} – 105mV		1.02 × V_{BG} – 0 mV	V	PLL mode, f_{SYS} = 50 MHz
I_{DD_rlim}	Regulator internal output current limit	VDD	120	275	460	mA	
I_{DD_rlimr}	Regulator internal output current limit during reset	VDD	24	55	92	mA	
t_{VDD_su}	Regulator setup time after power-up on UVDD	VDD		120	400	µs	FAST mode
BVDD Regulator							
BV_{DD_ro}	Regulator output voltage	BVDD	V_{REFINT} – 25 mV		V_{REFINT} + 25 mV	V	PLLC.PMF > 0
BI_{DD_rlim}	Regulator internal output current limit	BVDD	17	40	70	mA	PLLC.PMF > 0
t_{BVDD_su}	Regulator setup time after setting PLLC.PMF > 0	BVDD		70	230	µs	

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
FVDD Regulator							
$FV_{DD\_}ro$	Regulator output voltage	FVDD	V_{BG} × 1.34 – 40 mV		V_{BG} × 1.34 + 40 mV	V	no load
			V_{BG} × 1.34 – 170 mV		V_{BG} × 1.34 – 40 mV	V	I_{FVDD} = –40 mA
$FI_{DD\_}rl_{im}$	Regulator internal output current limit	FVDD	52	120	200	mA	
t_{FVDD_su}	Regulator setup time after power-up on UV_{DD}	FVDD		100	330	µs	I_{FVDD} = –40 mA
ADC (conversion reference V_{REFC} either equal to external reference V_{REF} or internal reference V_{REFINT})							
LSB	LSB value			V_{REFC} / 1024		V	
INL	Integral non-linearity: difference between the output of an actual ADC and the line best fitting the output function (best-fit line)		–2.5		2.5	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V
ZE	Zero error: difference between the output of an ideal and an actual ADC for zero input voltage		–1		1	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V
FSE	Full-scale error: difference between the output of an ideal and an actual ADC for full-scale input voltage		–1		1	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V
TUE	Total unadjusted error: maximum sum of integral non-linearity, zero error and full-scale error		–3.5		3.5	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V
QE	Quantization error: uncertainty because of ADC resolution		–0.5		0.5	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
AE	Absolute error: difference between the actual input voltage and the full-scale weighted equivalent of the binary output code, all error sources included		–4		4	LSB	2.4 V < V_{REFC} < AV_{DD}, 4.5 V < AV_{DD} < 5.5 V
R	Conversion range	P ports	AV_{SS}		V_{REFC}	V	2.4 V < V_{REFC} < AV_{DD}
A	Conversion result			INT (V_{in} / LSB)		hex	AV_{SS} < V_{in} < V_{REFC}
			000			hex	$V_{in} \leq AV_{SS}$
					3FF	hex	$V_{in} \geq V_{REFC}$
t_c	Conversion time		4			µs	TSAMP = 0, f_{IO} = 10 MHz
t_s	Sample time		2			µs	
C_i	Internal sampling capacitance during sampling period			7.5 2.5		pF	buffer off buffer on
R_i	Internal serial resistance during sampling period			7		kOhm	buffer off
PLL and ERM							
t_{SUPLL}	PLL locking time				100	µs	@ f_{XTAL} = 4 MHz, f_{SYS} = 16 MHz
dt_{PLL}	I/O clock uncertainty due to PLL jitter, short term and long term			±2.5		ns	ERM off
dt_{ERM}	I/O clock phase shift due to ERM action, short term and long term		0 0 0		7.5 12.5 20	ns	ERM on, WEAK setting ERM on, NORMAL setting ERM on, STRONG setting
RC Oscillator							
f_{RC}	Output frequency		20	35	50	kHz	
Clock Supervision							
f_{SUP}	Clock supervision threshold frequency	XTAL1	70		350	kHz	

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = UV_{SS1} = FV_{SS} = HV_{SSn} = EV_{SSn} = AV_{SS}$ = 0 V, 3.5 V < $AV_{DD} = UV_{DD} = UV_{DD1}$ < 5.5 V, 4.75 V < HV_{DDn} < 5.25 V, 3 V < EV_{DDn} < 5.5 V, T_{CASE} = 0 °C to +70 °C, f_{XTAL} = 5 MHz, external components according to Fig. 2–3 (unless otherwise noted)

Symbol	Parameter	Pin Na.	Min.	Typ. [1]	Max.	Unit	Test Conditions
SPI (Fig. 3–1, Fig. 3–2)							
t_{so}	Data out setup time from internal clock out	SPI-D-OUT			+5	ns	@ C_l = 30 pF, Port FAST mode, $UV_{DD} \geq$ 4.5 V
t_{ho}	Data out hold time from internal clock out	SPI-D-OUT	–5			ns	
t_{soci}	Data in setup time to internal clock out	SPI-D-IN			35	ns	
t_{hoci}	Data in hold time from internal clock out	SPI-D-IN			0	ns	
t_{soce}	Data out setup time from external clock in	SPI-D-OUT			$3.5 / f_{IO}$ + 35 [4]	ns	
t_{hoce}	Data out hold time from external clock in	SPI-D-OUT	$2.5 / f_{IO}$ + 15 [5]			ns	
t_{si}	Data in setup time to external clock in	SPI-D-IN			$1 / f_{IO}$ + 25 [4]	ns	
t_{hi}	Data in hold time from external clock in	SPI-D-IN			$1 / f_{IO}$ [4]	ns	
CAN							
t_{IOd}	Internal I/O delay time	CAN-RX, CAN-TX			35 [4]	ns	@ C_l = 30 pF, Port FAST mode, $UV_{DD} \geq$ 4.5 V
DIGITbus (Fig. 3–3)							
t_{btj}	Bit time jitter	DIGIT-OUT			±15	ns	rising edges, internal clock master, ERM on
t_{fed}	Falling edge delay from nominal pulse falling edge	DIGIT-OUT			$t_{BIT} / 64$ + 40 [4]	ns	@ C_l = 30 pF, Port FAST mode, $UV_{DD} \geq$ 4.5 V

[1] Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied, and are not 100% tested.

[2] Value may be exceeded with unusual hardware option setting
[3] Design value only, the actually observable hysteresis may be lower due to system activity and related supply noise
[4] When ERM is active, this time value is increased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[5] When ERM is active, this time value is decreased by 7.5 ns with WEAK ERM setting, by 12.5 ns with NORMAL ERM setting and by 20 ns with STRONG ERM setting.
[6] Measured with external clock. Add typically 120 µA for operation on quartz with SR0.XTAL = 0 (Oscillator RUN mode).

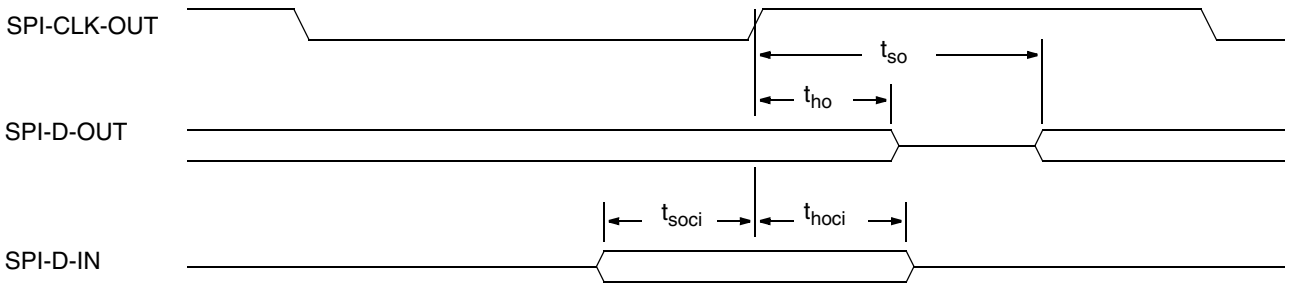


Fig. 3–1: SPI: Send and receive data with internal clock. Timing is valid for inverted clock too (data valid at positive edge).

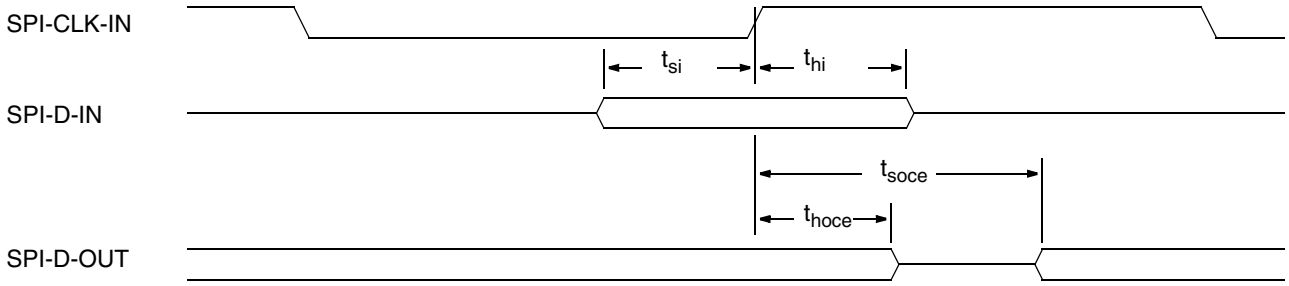
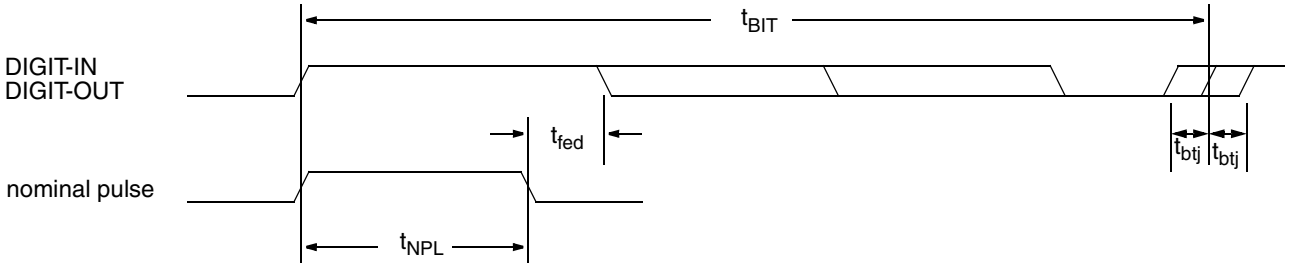


Fig. 3–2: SPI: Send and receive data with external clock. Timing is valid for inverted clock too (data valid at positive edge).



t_{NPL}: Nominal programmed pulse length. Depends on programmed phase, baud rate and transmitted sign (0, 1, T). Should be 1/4 for sign 0, 1/2 for sign 1 and 3/4 for sign T of t_{BIT}.

Fig. 3–3: DIGITbus I/O timing

3.4. Recommended Quartz Crystal Characteristics

Table 3–4: 3.5 V < UV_{DD} < 5.5 V, external components according to Fig. 2–3, unless otherwise noted

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
f_P	Parallel resonance frequency @ C_L = 12 pF	4		5	MHz	
R_1	Series resonance resistance for 50 ms oscillation start-up time and proper function @ C_L = 12 pF @ f_P = 4 MHz @ f_P = 5 MHz			340 380 210 270 240 280 140 180	Ohm	START-UP START-UP, 4.5 V < UV_{DD} < 5.5 V RUN RUN, 4.5 V < UV_{DD} < 5.5 V START-UP START-UP, 4.5 V < UV_{DD} < 5.5 V RUN RUN, 4.5 V < UV_{DD} < 5.5 V
C_{EXT}	External oscillation capacitances, connected to V_{SS}		18		pF	

4. CPU and Clock System

4.1. ARM7TDMI™ CPU

The CPU is an ARM7TDMI 32-bit RISC processor. This is a member of the Advanced RISC Machines (ARM) family. The ARM7TDMI is a 3-stage pipeline machine and supports a 4-Gbit address range. In addition to the 32-bit standard ARM instruction set, the ARM7TDMI supports the 16-bit Thumb instruction set which allows a higher code density. It includes a 64-bit result multiplier and a JTAG interface with an embedded debug module.

4.1.1. CPU States

The ARM7TDMI CPU allows operation in two states:

– ARM state: 32-bit instructions

– Thumb state: 16-bit instructions

After reset and exceptions, ARM state is active.

4.2. Operating Modes

To adapt to the large variety of CPU speed and current consumption requirements, the device offers a number of operating modes:

– CPU-active modes, where the CPU is clocked at selectable speeds

– Power-saving modes, where the CPU is kept reset and where only certain circuit portions are powered.

Fig. 4–1 shows how the various modes are accessed, in an operating modes state diagram.

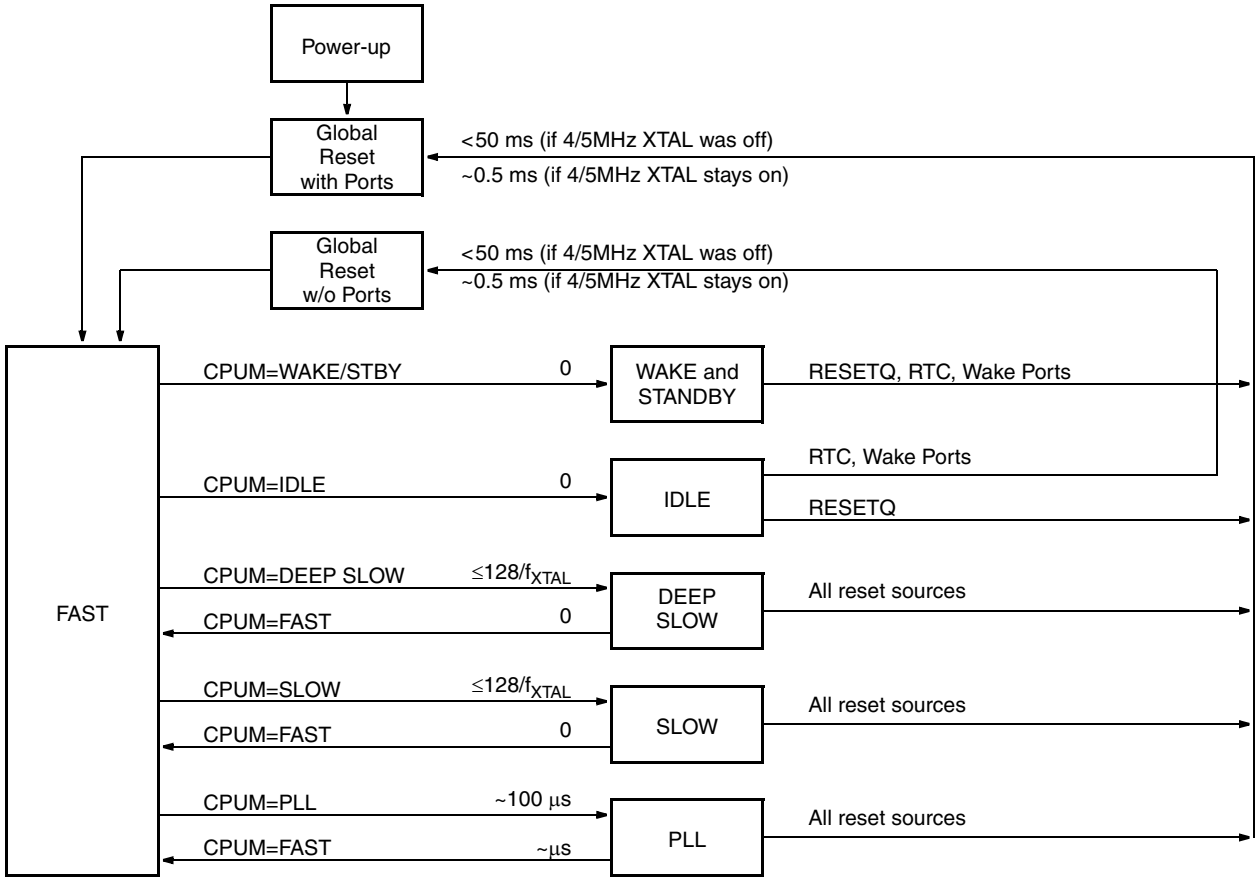


Fig. 4–1: Operating modes state diagram

4.2.1. CPU-Active Modes

The CPU can be operated in five different CPU-active modes (Table 4–1). Core modules that are also affected by CPU-active modes are:

1. Interrupt Controller with all internal and external interrupts

2. RAM, ROM/Flash and DMA

3. Watchdog

Table 4–1 shows the operability of the peripheral modules in the various CPU-active modes.

When switching between modes, neither interrupts (see Section 11.4.4. on page 96) nor DMA (see Section 22.3. on page 146) accesses are allowed, to prevent undefined behavior of the clock system.

4.2.1.1. FAST Mode

After reset the CPU is in FAST mode. The CPU clock and the I/O clock both equal the oscillator frequency f_{XTAL}.

Internal clock frequencies higher than f_{XTAL} are not available in this mode. Modules requiring $f_0 = 2f_{XTAL}$ for operation will not work properly, as f_0 is set to $f_1 = f_{XTAL}$.

Returning CPU from any other CPU-active mode to FAST mode is done by selecting the appropriate mode in the standby registers field SR1.CPUM (Table 4–3).

4.2.1.2. PLL Modes

To increase CPU performance, a PLL allows to multiply f_{XTAL}. The CPU will operate at this higher frequency f_{SYS} and its speed is automatically reduced only for accesses to slower modules (ROM/Flash, I/O).

Table 4–6 gives recommended settings for control registers and the various resulting operating frequencies for the PLL mode. These recommended settings achieve $f_0 = 2*f_{XTAL}$, $f_1 = f_{XTAL}$ and so forth, for unlimited operation of peripheral modules.

A PLL2 mode allows bypassing of the first stage of the divider chain. It allows a clock system with $f_{SYS} = n*f_{XTAL}$ and $f_0 = f_1 = f_{XTAL}$ for special applications, where the unlimited operation of peripheral circuits has to be sacrificed (see Table 4–7 for settings).

In both PLL modes, the EMI reduction module (ERM) can be operated, which reduces electromagnetic energy emission (see Section 4.5. on page 47).

Activating the PLL modes is done in FAST mode by the following routine:

1. For initialization, choose a pair of settings for the clock multiplication factor and the clock prescaler from Tables 4–6 or 4–7 and write them to PLLC.PMF and IOC.IOP.

2. When coming from SLOW or DEEP SLOW mode, allow t_{REFINT} to elapse for VREFINT and BVDD to set up. In all other cases, wait the time of t_{BVDD_su} for BVDD to set up.

3. Wait for at least t_{SUPLL} before checking PLLC.LCK to be set, to make sure that the PLL has locked.

4. Disable ICU and DMA, if active.

5. Enable PLL mode by writing 0x03, or PLL2 mode by writing 0x07 to SR1.CPUM (32-bit access only).

6. As the system frequency divider and the prescaler need some time to synchronize, the PLL mode is not active until PLLC.PLLM reads as 1.

7. At this point the ERM may be activated (see Section 4.5.3. on page 47) and ICU and DMA may be (re-)enabled.

Returning to FAST mode is done by the following routine:

1. Deactivate the ERM, if active (see Section 4.5.4. on page 47).

2. Disable ICU and DMA, if active.

3. Return to FAST mode by setting SR1.CPUM to 0x01 (32-bit access only).

4. Wait for PLLC.PLLM to read as 0.

5. Now the ICU and DMA may be (re-)enabled and the program may resume.

Attention: The PLL modes must be entered and left only via FAST mode. The registers PLLC.PMF and IOC.IOP may only be changed in FAST mode.

To reduce the power consumption in other CPU-active modes than PLL modes, the registers PLLC.PMF and IOC.IOP should be programmed to zero.

4.2.1.3. SLOW Mode

To considerably reduce power consumption, the user can reduce the internal CPU clock frequency to 1/128 of the normal f_{XTAL} value. In this SLOW mode, program execution is reduced to 1/128 of f_{XTAL}.

Some modules must not be operated during SLOW mode (e.g. CAN). Refer to module sections for details (see Table 4–1 on page 41).

Internal clock frequencies higher than f_{XTAL} are not available in this mode. Modules requiring $f_0 = 2*f_{XTAL}$ for operation will not work properly, as f_0 is set to $f_1 = f_{XTAL}$.

For switching between SLOW and FAST modes, use the following routine:

1. Disable ICU and DMA, if active.

2. Select the desired mode in the standby registers field CPUM (Table 4–3). The new f_{BUS} is effective immediately.

3. Now the ICU and DMA may be (re-)enabled and the program may resume (no waiting time).

4.2.1.4. DEEP SLOW Mode

To further reduce power consumption beyond SLOW mode, DEEP SLOW mode also disables most of the internal peripheral clocking system. Table 4–1 shows which peripheral modules can be operated in DEEP SLOW mode.

Only peripheral module clocks f_5 and slower are available from the divider chain. T0 can be operated only with this limitation.

For switching between DEEP SLOW and FAST modes, use the routine given for SLOW mode.

4.2.2. Power-Saving Modes

All power-saving modes are activated by the CPU. The complete core logic will immediately terminate operation and power will be removed. The result is a device current consumption that is greatly reduced, to the amount of leakage currents.

Table 4–1: Operability of modules in CPU-active modes

Module	PLL	PLL2	FAST	SLOW	DEEP SLOW
Core					
Digital Watchdog	✔	✔	✔	✔	✔
IRQ Interrupt Controller Unit	✔	✔	✔	✔	✔
FIQ Interrupt Logic	✔	✔	✔	✔	✔
Port Interrupts	✔	✔	✔	✔	✔
Port Wake Module	✔	✔	✔	✔	✔
Memory Patch Module	✔	✔	✔	✔	✔
Analog					
A/D Converter	✔	✔ 1)	✔ 1)	✔ 1)	
ALARM, P06 and WAIT Comparators	✔	✔	✔	✔	✔
LCD Module	✔	✔ 1)	✔ 1)	✔ 1)	✔ 3)
Communication					
DMA	✔	✔	✔	✔	✔
DMA Timer, GBus	✔	✔ 1)	✔ 1)	✔ 1)	✔ 3)
UART	✔	✔ 1)	✔ 1)	✔ 1)	
SPI	✔	✔ 1)	✔ 1)	✔ 1)	
CAN	✔	✔ 1)	✔ 1)	✔ 1) 4)	
DIGITbus	✔	✔ 1)	✔ 1)	✔ 1) 4)	✔ 3) 4)
I^2C	✔	✔	✔	✔	
Input & Output					
Ports	✔	✔	✔	✔	✔
Stepper Motor Module	✔	✔ 1)	✔ 1)		
PWM	✔	✔ 1)	✔ 1)	✔ 1)	
PFM	✔	✔ 1)	✔ 1)	✔ 1)	✔ 3)
Audio Module	✔	✔ 1)	✔ 1)		
Clock Outputs	✔	✔ 1)	✔ 1)	✔ 1)	✔ 3)
Timers & Counters					
Capture Compare Module	✔	✔ 1)	✔ 1)	✔ 2)	✔ 2) 3) 4)
Timers	✔	✔ 1)	✔ 1)	✔ 1)	✔ 3)
RTC/Polling Module	✔	✔	✔	✔	✔

1) Possibly affected by f_0 equaling f_1
2) Avoid write access to CCxI
3) Only clocks f5 and slower are available from Clock Divider
4) Don't access registers or CAN RAM

Table 4–1: Operability of modules in CPU-active modes

Module	PLL	PLL2	FAST	SLOW	DEEP SLOW
Miscellaneous					
JTAG	✔	✔	✔	✔	✔
Embedded Trace Module	✔	✔	✔	✔	✔

[1] Possibly affected by f_0 equaling f_1
[2] Avoid write access to CCxI
[3] Only clocks f5 and slower are available from Clock Divider
[4] Don't access registers or CAN RAM

However, a means to leave these modes has to be provided. As the CPU is no longer active, either an external or internal WAKE signal has to be generated. The external WAKE costs no device current, but to generate an internal WAKE requires an internal oscillator and a real-time clock (RTC) to run, which will cost a small amount of supply current.

Please note that inadvertently entering a power-saving mode, (e.g., by an external electrical overstress (EOS) condition, when no wake source has been configured previously as recovery path from this state), renders the device locked in this power-saving mode. Only a RESETQ pin reset or a complete power removal and reapplication recovers the device from this state. Sufficient external shielding measures must be taken to avoid this hazard.

Table 4–2: Power-saving modes and related functionality versus CPU-active modes

Operating Mode		Modules That Can Be Activated	SRAM, CAN-RAM	Port Registers	Available Wake Sources
Power-Saving Modes	WAKE	Port wake module	data lost	reset	Wake ports
	STANDBY	All WAKE mode modules plus: - 4M XTAL or 20..50k RC oscillator - Real-time clock - Polling module	data lost	reset	Wake ports and RTC
	IDLE	All STANDBY mode modules plus an auxiliary VDD regulator that keeps RAM and port registers alive. Other modules according to Table 4–1 are not supported.	data retained	state retained	Wake ports and RTC
CPU-Active Modes		In principle all, for limitations see Table 4–1	active	active	Wake sources usable as interrupt

4.2.2.1. WAKE Mode

WAKE mode is the most current-saving operation mode. All device circuits are stopped or powered down except the port wake module (Table 4–2).

The port wake module allows the CPU to configure up to ten fixed device ports (see the device pinout for details) as wake ports (WP).

To prepare for WAKE mode, the CPU has to switch off the RTC and to configure the desired wake port(s) (see chapter "Power Saving Module", sections "Port Wake Module" and "RTC Module").

To enter WAKE mode, the CPU selects WAKE/STBY in register SR1.CPUM.

The device will immediately enter WAKE mode by resetting all circuitry, stopping all clocks, and powering down all regulators and analog circuitry. As long as all wake port inputs are kept at CMOS input levels (V_{il} = xV_{SS} ± 0.3 V and V_{ih} = xV_{DD} ± 0.3 V), the supply currents will be minimal. The device may be kept in this state indefinitely.

To exit WAKE mode, the previously configured wake port has to switch. Immediately a "wake reset" sequence will be started internally that pulls the RESETQ pin low and releases it as soon as all internal reset sources have become inactive. See chapter "Core Logic" for details on internal reset sources. After reset, the CPU starts in FAST mode, as usual.

4.2.2.2. STANDBY Mode

STANDBY mode allows to configure an internal wake source that wakes after a preselected period. As clock sources, either a current-saving, but imprecise internal RC oscillator, or the precise, but more current-consuming XTAL oscillator are selectable. Beside the port wake module, these circuits and the RTC are kept alive (Table 4–2).

The RTC allows the CPU to select from one-second to one-day clocks (see section “Power Saving Module” for details) as wake signal. Beside serving as wake source, the CPU may use the RTC as real time clock that is not halted by resets.

A polling module, driven by a selectable RTC clock, may be configured to generate a polling pulse on H0.2 and sample the wake ports immediately after. Thus a periodical polling of wake ports may be achieved, with no continuous power consumption in external circuitry.

To prepare for STANDBY mode, the CPU has to configure the desired RTC wake clock (see chapter “Power Saving Module”, section “RTC Module”), in addition to the desired wake port(s), see chapter “Power Saving Module”, section “Port Wake Module”.

To enter STANDBY mode, the CPU selects WAKE/STBY in register SR1.CPUM.

The device will immediately enter STANDBY mode by resetting all circuitry, stopping the unused clocks, and powering down all regulators and remaining analog circuitry. As long as all wake port inputs are kept at CMOS input levels ($V_{il} = xV_{SS} \pm 0.3$ V and $V_{ih} = xV_{DD} \pm 0.3$ V), the supply currents will only amount to the requirement of the oscillator(s) and the slow-clocked RTC and polling modules.

To exit STANDBY mode, the previously configured wake source has to switch. Immediately a Wake Reset sequence will be started internally that pulls the RESETQ pin low and releases it as soon as all internal reset sources have become inactive. See chapter “Core Logic” for details on internal reset sources. After reset, the CPU starts in FAST mode, as usual.

4.2.2.3. IDLE Mode

IDLE mode allows usage of the same wake sources as STANDBY mode. But in contrast to WAKE and STANDBY modes, an auxiliary V_{DD} regulator allows to maintain some core functionality (Table 4–2):

– the internal SRAM and CAN-RAM keeps its programmed data

– all U-, P- and H-Port registers (see chapter “Ports”) keep their programmed state (LCD ports will output a low level).

Leakage currents in these additionally powered modules add to the device current consumption.

To prepare for IDLE mode, the CPU has to configure the desired RTC wake clock (see chapter “Power Saving Module”, section “RTC Module”), the desired wake port(s) (see chapter “Power Saving Module”, section “Port Wake Module”) and the port registers as desired.

To enter IDLE mode, the CPU selects IDLE in register SR1.CPUM.

The device will immediately enter IDLE mode by resetting all circuitry except the port registers, stopping the unused clocks, and powering down all main regulators and remaining analog circuitry, but not the auxiliary V_{DD} regulator. As long as all U-, P- and H-Ports, that are configured as inputs, are kept at CMOS input levels ($V_{il} = xV_{SS} \pm 0.3$ V and $V_{ih} = xV_{DD} \pm 0.3$ V), the supply currents will amount to the requirement of the oscillator(s), the RTC and polling modules and mere leakage currents flowing mainly in SRAM.

To exit IDLE mode, the previously configured wake source has to switch. Immediately a Wake Reset sequence will be started internally that pulls the RESETQ pin low and releases it as soon as all internal reset sources have become inactive, but port registers and SRAM will be exempt from being reset. See chapter “Core Logic” for details on internal reset sources. After reset, the CPU starts in FAST mode, as usual.

4.3. Clock System

The IC contains a quartz oscillator circuit that only requires external connection of a quartz and of two oscillation capacitors. Its start-up and run properties are controllable by SW. See section “Core Logic” for details.

The oscillator clock f_{XTAL} drives a clock system that supplies the various modules with its specific clock (Fig. 4–2).

A frequency multiplying PLL allows to select the system clock f_{SYS} to be higher than f_{XTAL} for high speed CPU and module operation.

A divider chain divides f_{IO} down to supply peripheral module clocks f_0 to f_{17}. Module clock selection is software defined in some cases, hardware or HW option defined in other cases. The module descriptions give details.

The standby register field SR1.CPUM selects the operating mode (Table 4–3).

Table 4–3: Operating mode selection and effect on clocks

SR1.CPUM			Operating Mode	f_{BUS}	f_{IO}	f_0	f_1
0	0	0	SLOW	f_{XTAL}/ 128	f_{XTAL}/ 128	f_{XTAL}	f_{XTAL}
0	0	1	FAST	f_{XTAL}	f_{XTAL}	f_{XTAL}	f_{XTAL}
0	1	0	DEEP SLOW	f_{XTAL}/ 128	f_{XTAL}/ 128	f0 to f4 = 0	
0	1	1	PLL	n f_{XTAL}	n/m f_{XTAL}	n/m f_{XTAL}	n/2m f_{XTAL}
1	0	0	WAKE /STBY	-	-	-	-
1	0	1	IDLE	-	-	-	-
1	1	0	rsvd	-	-	-	-
1	1	1	PLL2	n f_{XTAL}	n/m f_{XTAL}	n/m f_{XTAL}	n/m f_{XTAL}

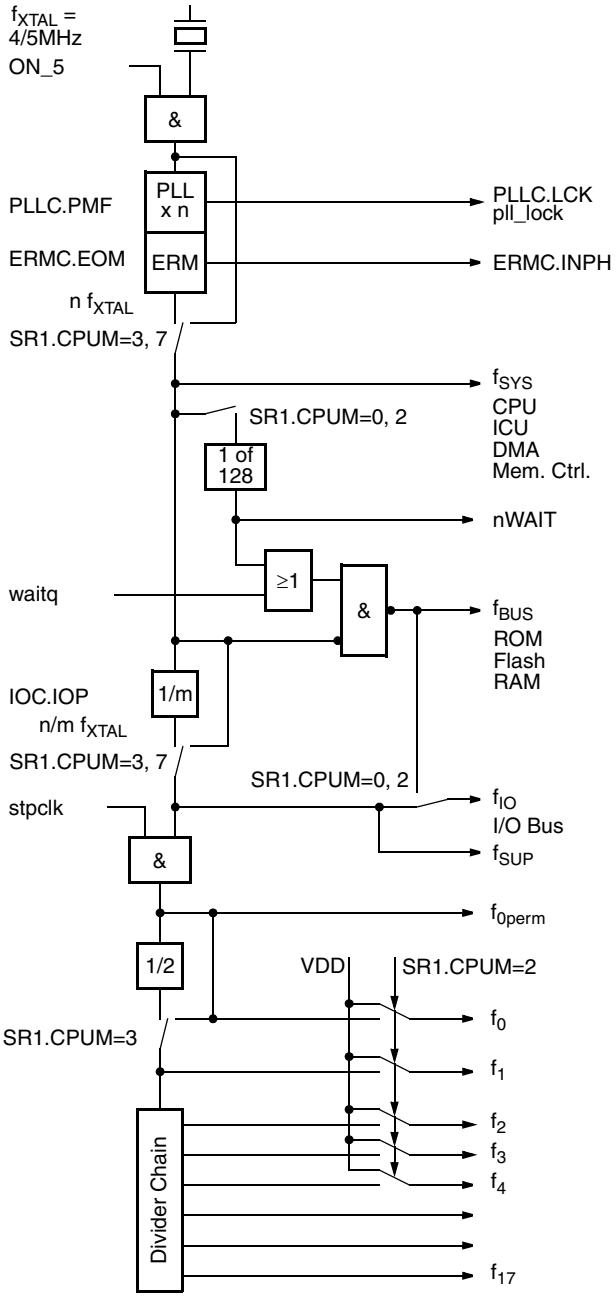


Fig. 4–2: Clock system

4.3.1. PLL

The PLL is composed of a phase comparator, a voltage-controlled oscillator (VCO), a frequency divider and an internal bypass. The phase comparator compares the input frequency f_{XTAL} with the output frequency of the frequency divider, f_{REF}. It outputs the voltage V_P, which is proportional to the phase difference of the two input frequencies. V_P controls the VCO which outputs the desired frequency. This frequency is fed back by the frequency divider to the phase comparator. The frequency divider divides the input frequency down to f_{REF} which ideally is the same as f_{XTAL}.

The phase-locked state of the PLL is signaled by a lock signal. It is available as flag PLLC.LCK. It may be routed to the

LCK special output by selection in fields ANAU.LE and ANAU.LS (UVDD Analog Section).

The block multiplies f_{XTAL} by n = PMF+1 to achieve f_{SYS}. PLL control register PLLC allows to set the desired value.

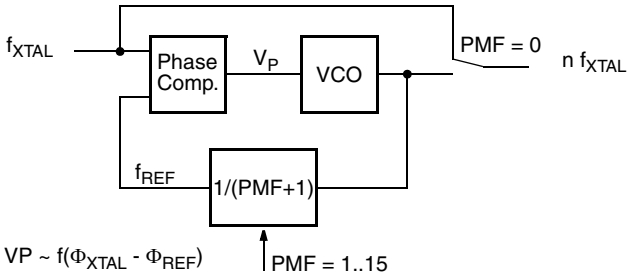


Fig. 4–3: PLL block diagram

4.3.2. I/O Clock Prescaler

This prescaler derives the clock for the peripheral modules (the I/O clock f_{IO}) from the system clock f_{SYS}. It divides f_{SYS} by an integer number m. The I/O clock control register IOC allows to set the desired value. m is recommended to be set equal to n/2.

4.3.3. Divider Chain and Clock Outputs CO0 and CO1

The peripheral module divider chain receives f_{SYS}/m and supplies the various modules with their specific clocks. Each stage of the divider chain divides its input clock by two. Thus only powers of two of the divider chain input clock are obtainable for the peripheral modules.

Table 4–3 shows the effect of CPU-active mode selection on the divider output frequencies f_0 through f_{17}. Note that in modes FAST, PLL and SLOW, with n = 2m (which is recommended), f_1 always equals f_{XTAL}. Thus f1 and all further subdivided clocks are unaffected by switching between these modes.

Section “HW Options” gives details about HW option-controlled clocks, their selection and their activation.

Note that specifying 1/1.5 and 1/2.5 prescaled clocks results in clock signals with 33% resp. 20% duty factor.

Two clock output signals, CO0 and CO1, provide external visibility of internal clocks (Figure 4–4). Clocks are selected by register CO0SEL.

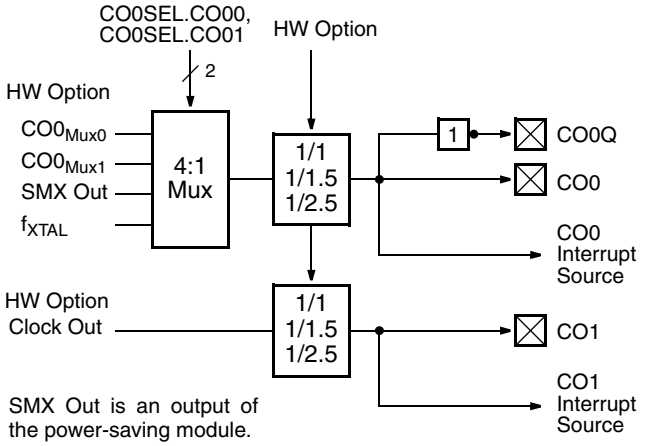


Fig. 4–4: Clock outputs diagram

Signal CO0 is the output of a prescaler and a 4-to-1 multiplexer. Prescaler and input for the multiplexer are selectable by HW options (see Table 4–4). The output selection of the multiplexer is done by register CO0SEL, bits CO01 and CO00. The outputs of the prescalers are fed not only to the ports, but may also serve as interrupt source. The U-Ports assigned to function as clock outputs (see Table 4–4) have to be configured as SPECIAL OUT.

The interrupt source output of this module is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

CO0 and CO1 are not affected by SLOW mode.

Table 4–4: HW options and ports

Signal	HW Options		Initialization	
	Item	Address	Item	Setting
CO0	CO0 Prescaler	CO00C	CO0 output	U1.6, U3.3 or U7.7 special out
	$CO0_{Mux0}$		CO0Q output	U1.5 special out 1)
	$CO0_{Mux1}$	CO01C		
CO1	CO1 Prescaler	CO1C	CO1 output	U0.4, U1.5 or U7.6 special out1)
	Clock Out			
1) HW Options register flag PM.U15 switches between CO0Q and CO1 at U1.5 special out.				

4.4. Memory Controller

The memory controller connects the CPU to the complex memory system. It controls the various types of access and wait states.

Features

- support of one synchronous and up to three different asynchronous memory areas
- different wait state values for sequential and nonsequential accesses to asynchronous memory
- allows 8, 16 and 32-bit memory accesses from CPU
- supports access to 8, 16 and 32-bit wide memory
- allows big or little endian memory access.

4.4.1. Principle of Operation

4.4.1.1. General

The memory controller contains a memory mapping unit (MMU) to control the mapping of the whole memory system, a wait state register (WSR) to initialize the various wait states, a final state machine and a wait state counter to control the various types of access and wait states.

4.4.1.2. Initialization

After reset, the CPU runs in FAST mode, the wait state counter is disabled due to SR1.CPUM=1 and no wait states for access to asynchronous memory are programmed. The access to the synchronous memory (I/O) works right after reset, independent from any setting by software. Endian mode is set by the control word via bit CR.ENDIAN.

First, initialize the wait state register WSR (cf. Table 4–6). Then the PLL or PLL2 mode may be enabled, if desired. Don't change the wait state register while in PLL mode, this may lead to a memory access with undetermined wait state count in the following cycle.

4.4.1.3. Operation

With proper SW initialization, the memory controller is also ready to access the asynchronous memory systems (ROM/Flash, RAM, Boot ROM).

The MMU decides from the address and the CR setting, which area in memory space contains a 8, 16 or 32-bit memory system. It preselects the different types of memory (ROM/Flash, RAM, Boot ROM and I/O-Pages) and computes the address for ROM/Flash minus 200000hex, if ROM/Flash should be mapped to base address 200000hex.

In presence of a patch module, the PATCHACC signal from the patch module signals to the MMU that the next access will be an access to the patch module instead to normal ROM/Flash.

The DMAACC signal from the DMA Module signals that the next address value will be driven by DMA and not by CPU. Due to the fact that DMA always reads/writes a location in ROM/Flash, RAM or Boot ROM and writes/reads to a location in the I/O area within the same bus cycle, the MMU also preselects the I/O area, and the Memory Controller times the whole DMA cycle to the restrictions of the slow synchronous I/O area.

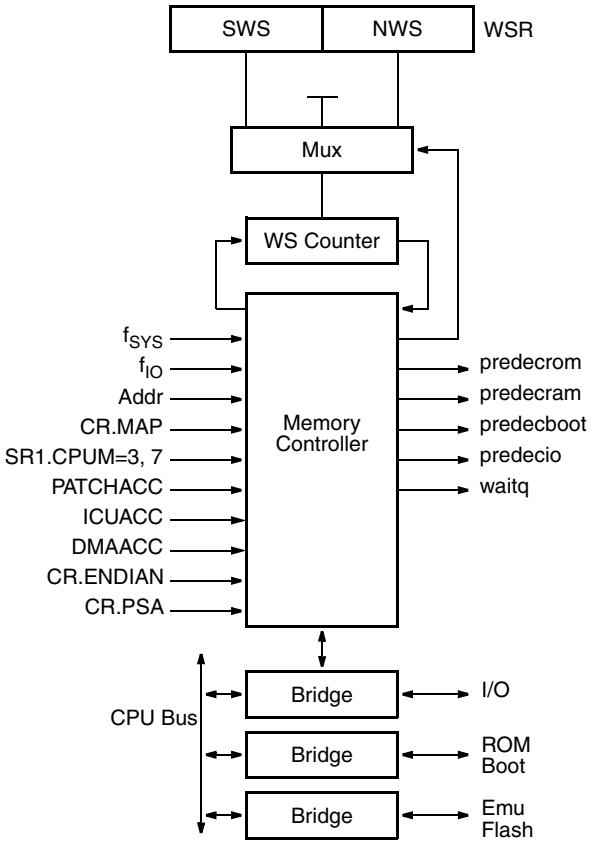


Fig. 4–5: Memory controller block diagram

4.4.1.4. Inactivation

Returning the memory controller to FAST mode (SR1.CPUM = 1) will immediately switch the CPU to FAST mode and change any further access to asynchronous memory to non-wait-state operation.

4.5. EMI Reduction Module (ERM)

The IC contains an EMI reduction module (ERM), which is capable of reducing electromagnetic radiation that might cause interference to other electronic equipment. The concept of this circuit uses precisely defined time offsets of the master clock phases as generated by the PLL. Thus, the module is only available in PLL modes.

All internal clock signals except f_{XTAL} are affected. Thus also the sampling time points of clocked inputs and outputs of all internal modules are modulated. To avoid a possible performance degradation of, e.g., communication modules in a user environment, the maximum possible delay of the sampling clock phase can be controlled with the parameters given in table 4–6. In critical applications, I/O sampling time point modulation and EMI reduction can thus be compromised. Section 4.7. gives application hints.

Features

- Strong suppression of electromagnetic radiation
- Precisely controlled effect on sampling time points of clocked I/O (ADC, CAN, UART, SPI etc.)
- All parameters fully controllable and reproducible
- Three operation modes for different purposes
- Works for clock frequencies of up to 48 MHz
- No degradation of CPU performance

4.5.1. Modes

The ERM has four modes of operation:

- Mode 0, ERM off.
- Mode 1 is intended for the low f_{SYS} frequencies of 8 MHz or 10 MHz with a clock multiplier of n = 2 (PLLC.PMF = 1). Mode 1 with a clock tolerance of 7 has a similar harmonics suppression as the formerly used EMI Reduction Module V3.1.
- Mode 2. This mode is primarily intended for the deactivation process of mode 3 but may also be used for operation. It performs best at high clock frequencies.
- Mode 3 gives best results at medium and high f_{SYS} frequencies. At medium frequencies it is even capable of a certain suppression of the fundamental.

In each of the three operation modes the parameters may be particularly chosen with the help of registers ERMC.SUP and ERMC.TOL, to achieve an optimum suppression while keeping phase modulation of f_{IO} as low as possible. In Table 4–6, three sets of parameters with maximum f_{IO} sampling delays of:

- $\leq$ 7.5 ns (weak),
- $\leq$ 12.5 ns (normal) and
- $\leq$ 20 ns (strong)

are given as examples. However, other individual settings are possible (see section 4.5.2.).

At f_{SYS} frequencies of 40 MHz and above only a limited EMI suppression is possible. At an f_{SYS} of 50 MHz the ERM must be switched off (mode 0).

4.5.2. Rules for Setting Parameters

Each f_{SYS} multiplier n and each mode requires its own set of parameters. For individual settings other than those given in Table 4–6 the rules are given below.

For mode 1 the limits are:

- The suppression strength has no effect and should be kept at 0.
- The clock tolerance must not exceed the values given in the columns for strong settings.

For modes 2 and 3 the limits are:

- Numbers must not exceed the values given in the columns for strong settings.
- The clock tolerance must be equal to or less than (suppression strength + 1) / 2.
- In mode 2 the suppression strength must not exceed 43.
- In mode 3 the sum of clock tolerance and suppression strength must not exceed 43.

4.5.3. Initialization

For operation of the ERM, the clock system has to be in one of the PLL modes. After Reset, the ERM is in mode 0. All internal registers are reset to their default values. Registers TOL, SUP and EOM are also reset by PLLC.PMF = 0.

The initialization must be done in the following order:

1. Set the clock tolerance (ERMC.TOL) to 1. Set the suppression strength (ERMC.SUP) to 1 (modes 2 and 3 only).

2. Select the desired mode (1...3) in ERMC.EOM according to Table 4–6 (steps 1. and 2. must not be done in one operation).

3. Select the desired suppression strength (ERMC.SUP).

4. Select the desired clock tolerance (ERMC.TOL).

4.5.4. Deactivation

To deactivate the ERM, the following sequence must be observed:

1. If in mode 3, enter mode 2 by writing 2 to field ERMC.EOM.

2. Set the clock tolerance parameter (ERMC.TOL) to 1 (steps 1. and 2. must not be done in one operation).

3. Set the suppression strength (ERMC.SUP) to 1 (modes 2 and 3 only).

4. Wait at least 8 f_{SYS} cycles (e.g. CPU NOPs) before checking the in-phase flag ERMC.INPH.

5. Wait for flag ERMC.INPH to be set (may take up to 80 f_{SYS} cycles), then clear the field ERMC.EOM to return to mode 0. This will turn off the ERM.

6. To reset the ERM, set SUP = TOL = 0 (same result achieved by setting PLLC.PMF = 0 when back in FAST mode).

4.6. Registers

PLLC PLL Control

	7	6	5	4	3	2	1	0	
r/w	ACT	LCK	PLLM	x	PMF				
	x	x	x	x	0	0	0	0	Res

ACT **PLL Active**
r1: PLL started (PMF > 0)
r0: PLL not activated (PMF = 0)

LCK **PLL Locked**
r1: PLL locked
r0: PLL not locked

PLLM **PLL Mode Acknowledge**
r1: The clock chain has switched to PLL mode
r0: Not PLL mode

PMF **PLL Multiplication Factor (Table 4–6)**
w0: PLL is switched off and internally bypassed. This is the standby mode for the PLL.
w15-1: Starts PLL with the corresponding frequency. If not active anyway, the VREFINT Generator and BVDD Regulator are enabled

PMF is a write only field. Don't modify PMF in PLL mode.

$$f_{SYS} = n \cdot f_{XTAL} = (PMF + 1) \cdot f_{XTAL}$$

Above formula relates to PLL mode.

ERMC ERM Control

	7	6	5	4	3	2	1	0	
r/w	TSEL								3
r/w	x	x	x	x	x	x	x	x	2
r/w	EOM		x	x	TOL				1
r/w	INPH	x	SUP						0
	0x00000000								Res

TSEL **Test Select**
r/w Factory use only.

EOM **ERM Operation Mode**
r/w3: Mode 3
r/w2: Mode 2
r/w1: Mode 1
r/w0: Off

TOL **Clock Tolerance**
r/w15-0: (see Table 4–6 and 4–7)

INPH **In Phase (during deactivation)**
r1: Phase is 0 or 1
r0: Phase > 1

SUP **Suppression Strength**
r/w63-0 (see Table 4–6 and 4–7)

IOC I/O Control

	7	6	5	4	3	2	1	0	
w	x	x	x	x	x	IOP			
	x	x	x	x	x	0	0	0	Res

IOP **I/O Clock Prescaler (Table 4–6)**
w IOP is a write only field.

$$f_0 = \frac{f_{SYS}}{m} = \frac{f_{SYS}}{IOP + 1} = \frac{PMF + 1}{IOP + 1} \cdot f_{XTAL}$$

Above formula relates to PLL mode.

WSR Wait State Register

	7	6	5	4	3	2	1	0	
w	NWS				SWS				
	0x00								Res

NWS **Nonsequential Wait State Bits**
w: Number of wait states at nonsequential memory access.

SWS **Sequential Wait State Bits**
w: Number of wait states at sequential access.

The WSR influences access to ROM, Flash and Boot ROM.

CO0SEL Clock Out 0 Selection

	7	6	5	4	3	2	1	0	
w	x	x	x	x	x	x	CO01	CO00	
	x	x	x	x	x	x	0	0	Res

CO00, CO01 **Clock Out Bit 0 and 1**
w: Clock selection

Table 4–5: CO00 and CO01 Usage

CO01	CO00	Selection
0	0	$CO0_{Mux0}$
0	1	$CO0_{Mux1}$
1	0	SMX Out (Power Saving Module)
1	1	f_{XTAL}

4.6.1. Recommended Settings

Tables 4–6 and 4–7 list settings available for the EMU device. **When emulating a specific target device (MCM or mask ROM), use the recommended settings of that device only**. Settings **differing** from these two lists shall not be used and may result in undefined behavior.

It is required not to operate I/O faster than Flash.

Suppression Strength (SUP) and Clock Tolerance (TOL) may be varied between zero and the values for strong settings according to the rules in Section 4.5.2. The given limits must not be exceeded

Table 4–6: PLL and ERM modes: Recommended settings and resulting operating frequencies (MHz)

f_{XTAL}	CPU		Program Storage		I/O		ERMC.EOM = 1						ERMC.EOM = 2 or 3					
							Weak		Normal		Strong		Weak		Normal		Strong	
	f_{SYS}	PLLC. PMF	f_{BUS}	WSR	f_{IO}= f_0	IOC. IOP	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL
4	8	1	8	0x00	8	0	0	4	0	7	0	11	4	2	7	4	11	6
	16	3	16	0x00		1	0	8	0	14	0	15	8	4	14	7	22	11
			8	0x11			0	8	0	14	0	15	8	4	14	7	22	11
	24	5	24	0x00		2	0	2	0	2	0	2	12	1	21	1	33	1
			12	0x11			0	12	0	15	0	15	12	6	21	11	31	12
			8	0x22			0	12	0	15	0	15	12	6	21	11	31	12
	32	7	16	0x11		3	0	12	0	12	0	12	16	8	28	12	31	12
			10.7	0x22			0	12	0	12	0	12	16	8	28	12	31	12
	40	9	20	0x11		4	0	6	0	6	0	6	21	6	35	6	37	6
			13.3	0x22			0	6	0	6	0	6	21	6	35	6	37	6
			10	0x33			0	6	0	6	0	6	21	6	35	6	37	6
	48	11	24	0x11		5	0	1	0	1	0	1	25	1	42	1	42	1
			16	0x22			0	1	0	1	0	1	25	1	42	1	42	1
			12	0x33			0	1	0	1	0	1	25	1	42	1	42	1
5	10	1	10	0x00	10	0	0	5	0	8	0	14	5	3	8	4	14	7
	20	3	20	0x00		1	0	10	0	13	0	13	10	5	17	6	28	6
			10	0x11			0	10	0	15	0	15	10	5	17	9	28	12
	30	5	15	0x11		2	0	14	0	14	0	14	15	8	26	12	31	12
			10	0x22			0	14	0	14	0	14	15	8	26	12	31	12
	40	7	20	0x11		3	0	6	0	6	0	6	21	6	35	6	37	6
			13.3	0x22			0	6	0	6	0	6	21	6	35	6	37	6
			10	0x33			0	6	0	6	0	6	21	6	35	6	37	6
	50	9	25	0x11		4	set ERMC.EOM = 0						set ERMC.EOM = 0					
			16.7	0x22														
			12.5	0x33														

Table 4–7: PLL2 and ERM Modes: Settings sacrificing unlimited operation of peripheral modules and resulting operating frequencies (MHz)

f_{XTAL}	CPU		Flash		I/O		ERMC.EOM = 1						ERMC.EOM = 2 or 3					
							Weak		Normal		Strong		Weak		Normal		Strong	
	f_{SYS}	PLLC. PMF	f_{BUS}	WSR	f_{IO}= f_0	IOC. IOP	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL	SUP	TOL
4	12	2	6	0x11	4	2	0	6	0	10	0	15	6	3	10	5	16	8
			12	0x00			0	5	0	5	0	5	6	2	10	2	16	2
	20	4	10	0x11		4	0	10	0	15	0	15	10	5	17	8	28	8
5	15	2	7.5	0x11	5	2	0	7	0	13	0	15	7	4	13	7	21	11

4.7. PLL/ERM Application Notes

4.7.1. PLL Jitter

The embedded PLL synchronizes every n-th f_{SYS} cycle to the externally applied f_{XTAL} signal. This synchronization smoothly tries to cancel out influences from power supply noise and f_{XTAL} fluctuations. Depending on the application, this permanent re-synchronization process is expected to introduce some nanoseconds of phase jitter to f_{IO}.

It is important to note that PLL jitter does not introduce a noticeable frequency error, because the phase stays locked to the f_{XTAL} reference and fluctuates, even over long times, only within the same tight limits. Even with a PLL induced f_{IO} jitter of ±5 ns, the maximum observable frequency error between two f_{IO} clocks
– spaced 1µs apart is (1 µs ± 2 × 5 ns) / 1 µs = ±1%,
– spaced 1ms apart is (1 ms ± 2 × 5ns) / 1 ms = ±10 ppm,
– spaced 1s apart is (1 s ± 2 × 5 ns) / 1 s = ±0.01 ppm,
and so forth.

4.7.2. ERM "Jitter"

The effect of the ERM on f_{IO} phase and frequency is similar to that of PLL jitter in that it adds limited phase modulation. However, this ERM-induced jitter is especially tailored to improve the electromagnetic emission properties of the device. Section 4.5.1. gives details on setting the maximum phase delay:
– 7.5 ns (weak) translates to ±3.75 ns of f_{IO} jitter,
– 12.5 ns (normal) translates to ±6.25 ns of f_{IO} jitter,
– 20 ns (strong) translates to ±10 ns of f_{IO} jitter.

From these figures it is evident, that ERM introduces a jitter that, in its extent, is comparable to PLL jitter. Both influences may be added to estimate the combined PLL/ERM effect on I/O module operation.

4.7.3. Influence of PLL/ERM on Module Operation

DIGITbus, SPI and I2C synchronize external devices to one master clock. Their operation is hardly impeded by PLL/ERM jitter.

Modules such as UART and CAN communicate with external fixed-frequency devices, and there is a maximum frequency offset between transmitting and receiving station, that can be tolerated without transmission error.

Viewed from the receiving station, a frequency offset of the transmitting station is tolerable, as long as over the length of a complete telegram, every bit can still be detected unambiguously. Once the tolerable frequency offset is exceeded, communication is fatally disturbed. This tolerable offset is dependent on the capability of the involved circuitry to detect and compensate for frequency offset.

In the further discussion, the clock tolerance TOL is defined as percentage offset of the actual from the nominal frequency

$$TOL = \frac{|f_{act} - f_{nom}|}{f_{nom}}$$

Note that each transmitting and receiving station are allowed this same tolerance from nominal:
$f_{trans}=f_{nom}\pm TOL$ and $f_{rec}=f_{nom}\pm TOL$
The resulting maximum offset between transmitter and receiver is thus 2 x TOL.

4.7.3.1. UART

Let us consider the tolerable frequency offset in the case of the UART.

The baud rate frequency is always the sample clock frequency, divided by 8. The maximum telegram length is 12 bit. A transmitter frequency offset is tolerable as long as 12 × 8 ± 3 receiver sample clocks equal 12 × 8 transmitter sample clocks, which gives a transmitter frequency of $f_{Trans} = f_{Rec}(12 \times 8 / (12 \times 8 \pm 3)) = f_{Rec} \pm 3.23\%$ and TOL = ± 1.61%.

PLL and ERM jitter claim a certain portion of this tolerable offset. The assumption is that both influences add up to ±15 ns of f_{IO} jitter, and that $f_{IO} = f_0 = 8$ MHz.

With the baud rate set to 500 kBd, f_{SAMPLE} equals 4 MHz. With this setting, PLL and ERM jitter consume
2 × 15 ns / 750 ns = 4%
of the tolerable transmitter frequency offset and reduce TOL to 1.55%.

With the baud rate set to 19.23 kBd, f_{SAMPLE} equals 153.84 kHz. With this setting, PLL and ERM jitter consume
2×15 ns / 19.5 µs = 0.15%
of the tolerable transmitter frequency offset and reduce TOL only slightly, to 1.605%.

4.7.3.2. CAN

The CAN module contains logic that re-synchronizes a receiver to a transmitter several times during a telegram. By these means, a receiver is able to adapt to the transmitter frequency within narrow limits.

Two situations have to be distinguished:

1. Bit stuffing guarantees a maximum of 10 bit periods between two consecutive re-synchronization edges. Therefore the accumulated phase error must be less than the programmed re-synchronization jump width (SJW). The limitation that this situation imposes on the maximum TOL can be expressed as:

$$2 \times TOL \leq \frac{t_{SJW}}{10 \times t_{Bit}}$$

or

$$TOL \leq \frac{t_{SJW}}{2 \times 10 \times t_{Bit}}$$

2. Another limit on the maximum TOL is set by the situation where the CAN logic must be able to correctly sample the first bit after an error frame. This is the 13th bit after the last re-synchronization. This limitation can be expressed as:

$$2 \times TOL \leq \frac{\min(t_{Phase\ Seg1}, t_{Phase\ Seg2})}{13 \times t_{Bit} - t_{Phase\ Seg2}}$$

or

$$TOL \leq \frac{\min(t_{Phase\ Seg1}, t_{Phase\ Seg2})}{2 \times (13 \times t_{Bit} - t_{Phase\ Seg2})}$$

Example (f_0 = 10MHz)

With the baud rate set to 1 MBd, t_{Bit} equals 1 µs and is divided into 10 time quants (t_Q = 100ns). t_{SJW} and t_{TSEG2} are programmed to 3 t_Q (= $t_{Phase\ Seg1}$ = $t_{Phase\ Seg2}$). 3 t_Q are reserved for the propagation delay segment. In the first case the maximum tolerance TOL is 1.5% (edge to edge):

$$TOL \leq \frac{3}{2 \times 10 \times 10} = 0{,}015$$

In the second case, TOL is 1.2% (edge to sample point):

$$TOL \leq \frac{3}{2 \times (13 \times 10 - 3)} = 0{,}012$$

The smaller value of the above (1.2%) is relevant.

Following the UART example, PLL/ERM jitter consumes up to 2×15 ns of 300 ns (SJW = 3 time quants). This makes 30 ns / 300 ns = 10% of this tolerance, thus reducing TOL to ±1.08%.

With the baud rate set to 500 kBd, t_{Bit} = 2 µs and t_Q = 200 ns, the maximum tolerance TOL of 1.2% is reduced by 2×15 ns / 600 ns = 5% to ±1.14%.

Example (f_0 = 8MHz)

With the baud rate set to 1 MBd, t_{Bit} equals 1 µs and is divided into 8 time quants (t_Q = 125 ns). t_{SJW} and t_{TSEG2} are programmed to 2 t_Q (= $t_{Phase\ Seg1}$ = $t_{Phase\ Seg2}$). 3 t_Q are reserved for the propagation delay segment. In the first case the maximum tolerance TOL is 1.25% (edge to edge):

$$TOL \leq \frac{2}{2 \times 10 \times 8} = 0{,}0125$$

In the second case, TOL is 0.98% (edge to sample point):

$$TOL \leq \frac{2}{2 \times (13 \times 8 - 2)} = 0{,}0098$$

The smaller value of the above (0.98%) is relevant.

Following the UART example, PLL/ERM jitter consumes up to 2×15 ns of 250 ns (SJW = 2 time quants). This gives 30 ns / 250 ns = 12% of this tolerance, thus reducing TOL to ±0.86%.

With the baud rate set to 500 kBd, t_{Bit} = 2 µs and t_Q = 250 ns. The maximum tolerance TOL of 0.98% is reduced by
2×15 ns / 500 ns = 6.0% to ±0.92%.

4.7.3.3. DIGITbus

The DIGITbus master synchronizes with external devices via the serial data line. The slave node recovers the transmission clock from the data signal via an own PLL. This PLL will lock to the long-term average frequency of the master, and the slave node sees PLL/ERM jitter as a short-term frequency offset.

Following the UART example, one can define the tolerable frequency offset:
Every bit starts with a rising edge and thus every bit has a re-synchronization point. The bit period (t_{Bit}) is divided into four equal-length parts. Falling edges happen nominally after 1/4, 2/4 or 3/4 of the bit period. After 4/4 of the bit period a rising edge indicates the beginning of the next bit. The DIGITbus logic tolerates a jitter of these edges up to ±1/8 of the bit period. Thus, a transmitter frequency offset is tolerable up to $f_{Trans} = f_{Rec}(1 \pm 1/8) = f_{Rec} \pm 12.5\%$ and TOL = ±6.25%.

Again following the UART example, ERM/PLL jitter influences this tolerable offset:
With the baud rate set to 31.25 kBd, 1/8 of the bit period is 4 µs. PLL/ERM jitter reduces the maximum tolerance TOL of ±6.25% by 2×15 ns / 4µs = 0.75% to ±6.2%.

4.7.3.4. SPI and I2C

Modules like SPI and I2C synchronize with external devices by the serial clock. Thus, no frequency offset between transmitting and receiving station can develop, and no adverse effects of PLL/ERM operation are expected.

5. Memory and Special Function ROM (SFR) System

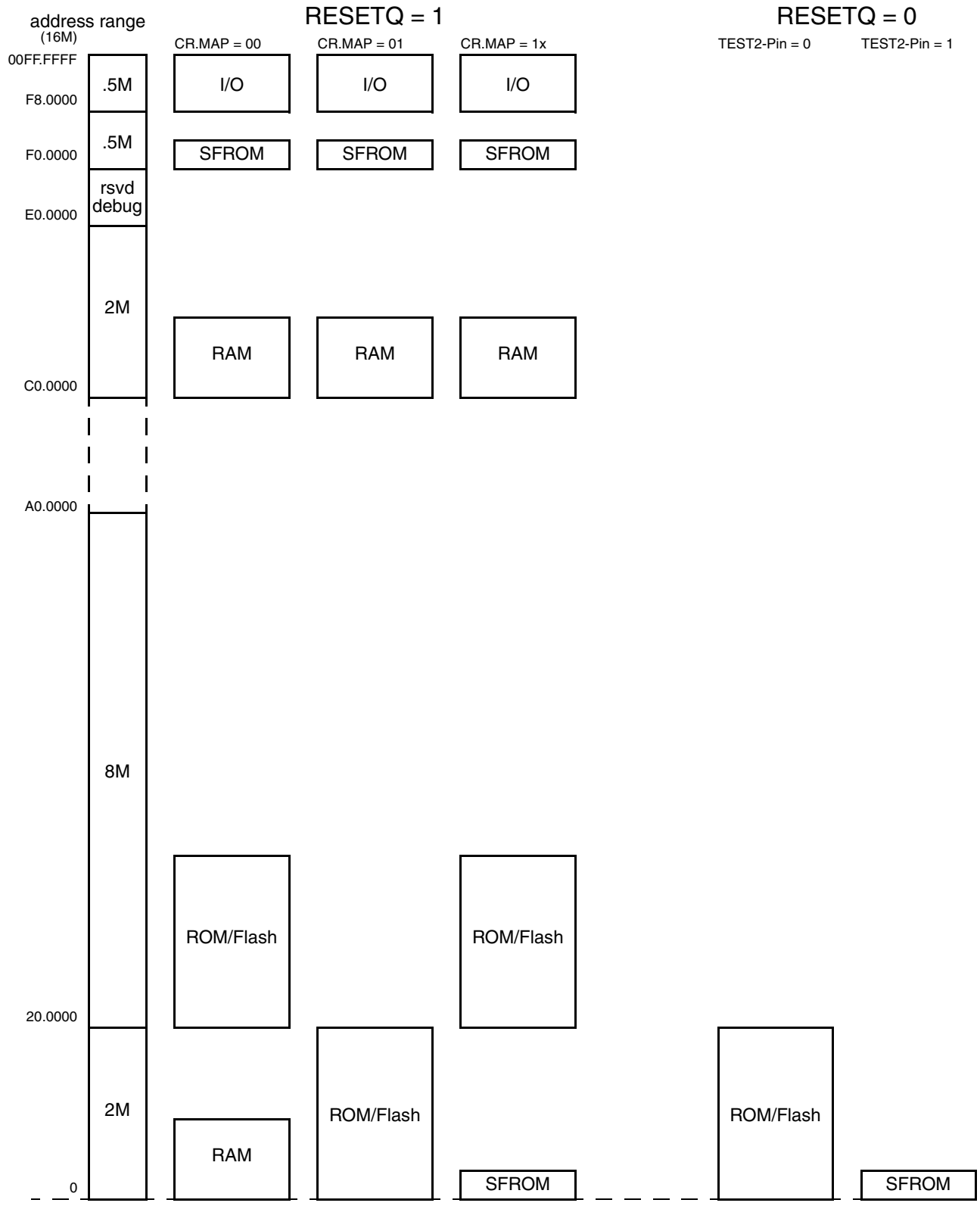


Fig. 5–1: Address map. Most common settings

5.1. RAM and ROM

On-chip RAM is composed of static RAM cells. It is protected against disturbances during reset as long as the specified operating voltages are available.

The 128PQFP multichip module also contains a 512 KB Flash EEPROM of the AMD Am29LV400BT type (top boot configuration). This device exhibits electrical byte program and sector erase functions. Refer to the AMD data sheet for details.

Future mask ROM derivatives may be specified to contain less or more internal RAM and ROM than this IC:

– ROM will grow upward from 0x0200000 to 0x09FFFFF (8 MByte) and can be remapped to physical address 0 (growing upward to 0x07FFFFF). It is 16 bit wide and is asynchronously accessed with wait states.

– RAM will always grow upward from physical address 0x0C00000 to 0x0DFFFFF (2 MByte) and can be mirrored to physical address 0. It is 32 bit wide and is asynchronously accessed without wait states.

– SFR will grow upward from physical address 0x0F00000 to 0x0F7FFFF (0.5 MByte) and can be mirrored to physical address 0.

– The I/O area will grow upward from physical address 0x0F80000 to 0x0FFFFFF (0.5 MByte). It is 16 bit wide and is asynchronously accessed with wait states.

Mirrored means the memory is accessible at both locations. Remapped means the memory is accessible at the new location only.

All parts (ROM, Flash and EMU) contain at least the I/O area, the RAM and the SFR.

5.1.1. Reserved Addresses

Reserved addresses (Table 5–1) are memory locations which define the behavior of internal HW or external systems. In our system the memory locations at address 0 and following have dedicated functions. The function of these memory locations depend on which kind of physical memory is mapped to these locations. As you can see in Fig. 5–1, ROM/Flash or SFR is mapped to location 0 at reset. Thus, a meaningful control word and reset vector can be obtained at start-up.

5.1.1.1. HW Options

Please refer to section “Hardware Options” for information on the layout of the HW options field and the corresponding registers in the I/O area. To activate the HW-option-related functions, the SW has to copy them to the corresponding locations in the I/O area.

But nevertheless these are HW options. It is not possible to modify them by SW in future ROM parts.

5.1.1.2. ROM ID

The ROM ID serves as identification of the corresponding application/SFR program. It will be read by an external system (test, debug) and does not influence internal HW.

The ROM ID contains a half-word sized hexadecimal value. The range is 0x0000 to 0xFFFF.

Table 5–1: Reserved addresses

Address	Size [byte]	Usage if mapped/mirrored to 0		
		RAM	ROM/ Flash	SFR
030 - 5F	48	General-purpose RAM. No HW-defined action.	HW options	
02C - 2F	4		Factory ID	
02A - 2B	2		reserved	
028 - 29	2		ROM ID	
024 - 27	4		Security Vector	ARM ID
020 - 23	4		Control word	
01C - 1F	4	FIQ (Fast Interrupt)		
018 - 1B	4	IRQ (Interrupt)		
014 - 17	4	Reserved		
010 - 13	4	Data Abort		
00C - 0F	4	Prefetch Abort		
008 - 0B	4	SWI (Software Interrupt)		
004 - 07	4	Undefined instruction		
000 - 03	4	Reset		

5.1.1.3. Factory ID

The factory ID contains a factory-defined code. It holds information about the HW version and other items. It can be read by an external system (test, debug) and does not influence internal HW. The SFR system may use this information and adapt its behavior according to the factory ID. The layout of the factory ID is not yet defined.

5.1.1.4. Security Vector (SV)

If the security vector is set (equal to 0x55AA55AA), the SFR does not enable the JTAG interface. In this way the application program may keep the JTAG access disabled.

An empty (erased) Flash ROM contains all ones. Hence the JTAG access is enabled.

Keep to the following sequence to reprogram a Flash ROM:

1. Clear the SV (i.e., program the SV address with 0x00000000 or any other value different from 0x55AA55AA).

2. Erase the Flash ROM (this sets the SV to 0xFFFFFFFF).

3. Program all of the Flash ROM, but skip the SV address (leave it at 0x0xFFFFFFFF).

4. Verify the Flash ROM.

5. If ok, set the SV (write 0x55AA55AA to the SV address). Otherwise return to step 2.

This procedure guarantees that the JTAG interface will be enabled after reset via SFR, if something has gone wrong during Flash ROM programming.

A correct application program should provide a separate way and interface to enable JTAG and to modify the Security Vector.

To allow easy JTAG access during development and debugging, leave the SV not set.

5.1.1.5. ARM ID

The ARM core can comprise a system control coprocessor (CP15), which contains among other things an ID Register. It allows the identification of the implemented processor. No CP15 has been implemented so far, but the ARM ID field may contain the same information.

5.1.1.6. Control Word

The control word defines the behavior of the HW during reset. Refer to section “Core Logic” for information on control word definition.

5.2. I/O Map

The I/O region (Table 5–2) is divided into the lower part (addresses 00F80000 to 0x00FBFFFF) which is connected to the 8 bit wide bus and into the upper part (addresses 00FC0000 to 0x00FFFFFF) which is connected to the 32-bit-wide bus. Please refer to section “Register Cross Reference Table” for detailed I/O register mapping.

Access to I/O modules which are connected to the 8-bit-wide bus is restricted: If not otherwise mentioned those modules must be accessed by byte access only. This memory area is organized in little endian format. If the CPU operates in big endian format, only byte access is recommended.

Table 5–2: I/O map

Address	Size [byte]	Access	Register
00FFFFFF 00FFFF00	256	32-bit, asynchronous, no wait states	IRQ and FIQ registers
00FFFEFF 00FFFE00	256		DMA registers
00FFFDFF 00FFFC00	512		Core registers
00FFFBFF 00FC0000	255 k		Free
00FBFFFF 00F90800	190 k	8-bit, synchronous, wait states	Free
00F907FF 00F90000	2 k		Registers
00F8FFFF 00F81200	59.5 k		Free
00F811FF 00F81000	512		CAN registers
00F80FFF 00F80000	4 k		CAN-RAM

5.3. Special Function ROM (SFR)

The job of the SFR is to enable the JTAG interface if necessary. Further actions as there are download, Flash ROM programming or debugging and monitoring have to be done via JTAG interface.

If TEST2 pin is held high during reset, the control word from the SFR is copied to the control register by HW. The SFR control word is configured to start program execution from the SFR, mirrored to location 0. The SFR control word disables JTAG.

5.3.1. Principle of operation

The first instruction of the SFR (the reset vector) loads the address of the next instruction in the original SFR (0xF00100, above the SFR HW options) into the program counter. This causes a jump from the mirrored SFR to the original SFR. The remaining part of the program is running in the original SFR and remapping of the memory does not influence correct operation of the SFR program.

If the TEST pin is detected low immediately after the SFR Firmware is started, the security vector in the Flash ROM is checked (Table 5–3). If the security vector is set in the application program, the application is started, otherwise the JTAG interface is enabled and the program stays in an endless loop to allow TAP access.

If the TEST pin is detected high immediately after the SFR firmware is started, additional functions may be invoked, selected by the digital levels at the analog input pins P0[1:0] (Table 5–3).
This mode is **not intended to be used within the application environment** but for test purposes only! It depends on the selected function if, what and how additional pins will be used for further operations. For notifications UART0 is used with 9600 Bd, 8 data bits, 2 stop bits, no parity and no kind of handshake:

– P0[1:0] = 0: Erase Flash:
  A blank check of the Flash ROM is executed to detect if it is erased. If the Flash ROM is not clear, it is erased and checked again, until detected as completely blank.
  When finished, “Flash is blank” is indicated via UART 0.
– P0[1:0] = 1: Run Test Program:
  For test purposes an endless program sequence is executed, using most parts of the chip internal modules. Make sure to run this program in the test environment only!
– P0[1:0] = 2: Check IDs:
  ROM and Factory ID are read out of the Flash ROM and indicated via UART 0. In addition to the 16-bit ROM ID (RID) the adjacent reserved 2 bytes are displayed as upper half word of the RID, e.g.:
  “RID: 0x0x56BC78AB
  FID: 0xDE9ACDFE”
– P0[1:0] = 3: Generate Checksum:
  A 32-bit check sum of the whole Flash ROM content is generated by summing up all data words, without considering carry-overs. The result is shown as hex value via UART 0, e.g.: “CHECKSUM: 0x3456BCDE”.

The watchdog is not triggered in the SFR program. Especially when the program is in the endless loop this would cause problems. But this endless loop will not be reached in ROM parts as long as the security vector is valid.

Table 5–3: Startup control by TEST pin and security vector (SV), TEST2 pin = 1

TEST pin	SV	QFP 128	EMU
0	set	Use application CW in flash to start the application in flash	Use application CW in emulation memory to start the application in emulation memory
0	not set	Enable application JTAG to allow debugging	
1	x	Decode further input selection to - run Test Program - erase Flash [1)] - generate checksum - indicate IDs	
[1)] Inadvertent execution of critical code additionally blocked by HW decoding of TEST pin state.			

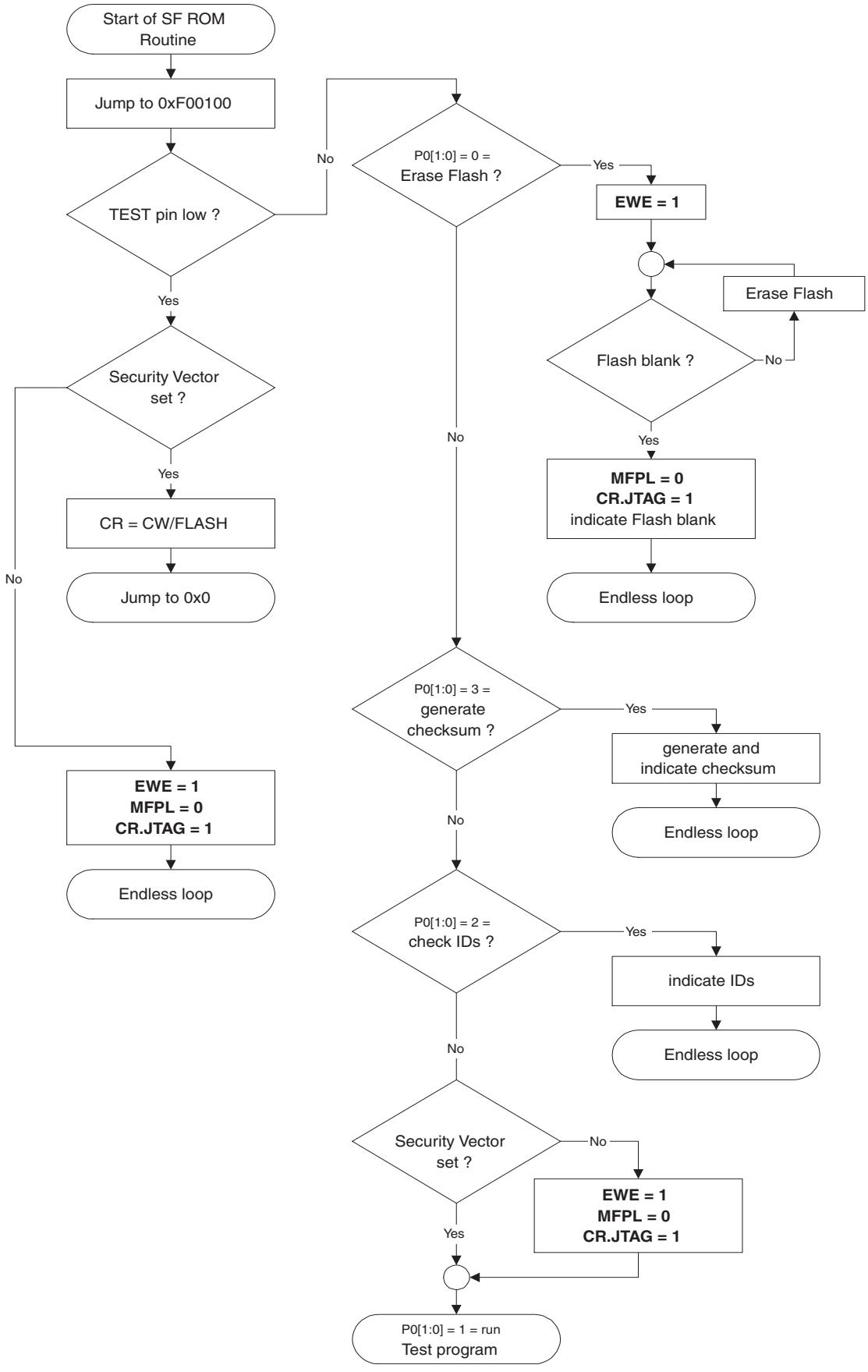


Fig. 5–2: Flow chart of special-function ROM routine

6. Core Logic

6.1. Control Word (CW)

A number of important system configuration properties are selectable during device start-up by means of a unique control word (CW).

6.1.1. Reset Active

At the end of the reset period, the device fetches this CW from address locations 0x20 to 0x23 of a source that is determined by the state of pins TEST and TEST2 and flag MFPLR.MFPL, see Table 6–1 for MCM and EMU parts, Table 6–2 for ROM parts.

Table 6–1: CW fetch in MCM parts (QFP128), and in EMU parts (CPGA257)

MCM	EMU	Necessary reset configuration		
"Control Word Fetch" desired from		**TEST2**	**TEST**	**MFPL**
Int. Flash	Ext. via EMU port	0	0	x
Int. Flash	Ext. via EMU port	0	1	1
Ext. via multifunction port				0 [1)]
Int. special-function ROM		1	x	x
[1)] Only available after a non-power-on RESET with MFPL = 0 set before				

As can be seen from Table 6–1, the device disables external access (through the Multifunction port) to internal code, as long as MFPLR.MFPL is 1 (= state after UVDD power-up). Setting it to 0 requires internal SW. By this means, an effective device lock mechanism is implemented that prevents unauthorized access to internal SW. See section 6.2. on the device lock module (DLM) for details.

In ROM parts, flag MFPLR.MFPL is available, but does not lock the multifunction port, so Table 6–1 reduces to Table 6–2.

Table 6–2: CW fetch in ROM parts (QFP128)

"Control Word Fetch" desired from	Necessary reset config. of pins	
	TEST2	**TEST**
Internal ROM	0	0
External via Multifunction port	0	1
Int. Special-Function ROM	1	x

6.1.2. Reset Inactive

When exiting reset, the CW is read and stored in the control register (CR) and the system will start up according to the configuration defined therein.

Normally the CW is fetched from the same memory that the system will start executing code from. Table 6–3 gives fixed CWs for a list of the most commonly used configurations.

Table 6–3: Some common system configurations and the corresponding CW setting

Part type	"Program Start" desired from	Additional desired properties	Necessary CW	
			31:16	**15:0**
EMU	ext. 32-bit sync SRAM (e.g. MT55L256L32F) on EMU port	Trace Bus ETM mode	0xFFBA	0x835F
		16-bit ROM/Flash emulation, Trace Bus ETM mode	0xFFBB	0x835F
		16-bit ROM/Flash emulation, Trace Bus Analyzer mode, Appl. JTAG released	0xFFBB	0xA3DF
	ext. 32-bit async auto-power-down Flash (2x Am29LV400BT) on EMU port	-	0xFFBA	0x675F
	ext. 16-bit async. auto-power-down Flash (Am29LV400BT) on EMU port	Trace Bus Analyzer mode	Don't care	0x2F5F
		Trace Bus ETM mode	Don't care	0x4F5F
MCM	int. 16-bit Flash (Am29LV400BT)	-	Don't care	0x7F5F
ROM	int. 16-bit ROM	-	Don't care	0x7F5F

For special purposes, the CW source and the program source may differ. For these purposes, a detailed description of the CW and CR function is given in the chapter “Control Register and Memory Interface”.

6.2. Device Lock Module (DLM)

Together with the corresponding application software, the DLM allows to establish protections against unwanted device accesses from outside the device and against unwanted flash data erase. It is composed of the

– multifunction port lock (MFPL), the

– EMU bus write enable control bit CR.EWE and the

– special-function ROM (SFR) system.

6.2.1. Multifunction Port Lock (MFPL)

To protect the memories contents against access from outside the device, the multifunction port, used as test bus, has to have a protection mechanism that is active by default. Together with appropriate hardware, the multifunction port lock (MFPL), controlled by a bit located in the multifunction port lock register (MFPLR), ensures a disabled multifunction port after POR (Power On RESET).

MFPLR Multifunction Port Lock Register

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	x	x	x	MFPL	
	x	x	x	x	x	x	x	1[1)]	Res

1) Set by POR or wake-up from power-saving modes.

MFPL	Multifunction Port Lock
r/w1:	Locked
r/w0:	Not locked

6.2.2. Emu Bus Write Enable (EWE)

In addition to MFPL, which protects the device memories against accesses from outside, with EMU bus write enable (EWE) inactive, the EMU bus is protected against write from anywhere, externally via test bus or internally via EMU or memory bus.
EWE is controlled by a bit in the control register which is cleared (inactive) by RESET. It prevents unauthorized and unintended Flash write and/or erase. EWE is located in the CR to be set easily by the control word generation of the emulation hardware. Code can be downloaded into the emulation RAM without having to enter a dedicated register write command or running a script file before. Write accesses by the debugger, e.g. to set break points, may take place immediately after RESET.
For details on the control register, please refer to chapter “Control Register and Memory Interface”.

With MFPL and CR.JTAG the device can be protected against external read and write accesses.
In addition, CR.EWE offers locking the Emu Bus/Flash memory separately against external and internal write and erase. With the reset status of EWE = 0 no Flash write is possible at first. EWE = 1 has to be programmed by software before.

Appropriate precautions within the application software are necessary, especially controlling DMA and PATCH activities, not to undermine a system lock.

6.2.3. Special Function ROM (SFR)

To get the CW from the multifunction port, MFPL has to be set to 0 before. This can be achieved either by a run through an appropriate part of the special-function ROM code or by a corresponding application software that prepares MFPL = 0. The necessary RESET to follow must not be a POR, because it resets MFPL to 1. Power-saving modes also set MFPL = 1.

For details on the SFR please refer to chapter “Memory and Special-Function ROM (SFR) System”.

6.3. Standby Registers

The standby registers SR0 to SR1 allow the user to switch on/off power or clock supply of single modules. With these flags it is possible to greatly influence power consumption and its related electromagnetic interference.

For details on enabling and disabling procedures and the standby state, please refer to the specific module descriptions.

SR0 Standby Register 0

	7	6	5	4	3	2	1	0	Offs
r/w	I2C1	I2C0	x	x	x	CAN3	CAN2	CAN1	3
r/w	TIM2	TIM3	TIM4	UART1	x	DGB	CCC1	x	2
r/w	LCD	x	PSLW	UART0	ADC	x	TIM1	XTAL	1
r/w	SM	x	x	x	SPI1	CAN0	CCC0	SPI0	0
	0x00000100								Res

I2C1 **I2C Module 1**
r/w1: Enabled
r/w0: Disabled

I2C0 **I2C Module 0**
r/w1: Enabled
r/w0: Disabled

CAN3 **CAN Module 3**
r/w1: Module active.
r/w0: Module off.

CAN2 **CAN Module 2**
r/w1: Module active.
r/w0: Module off.

CAN1 **CAN Module 1**
r/w1: Module active.
r/w0: Module off.

TIM2 **Timer 2**
r/w1: Module active.
r/w0: Module off.

TIM3 **Timer 3**
r/w1: Module active.
r/w0: Module off.

TIM4 **Timer 4**
r/w1: Module active.
r/w0: Module off.

UART1 **UART 1**
r/w1: Module active.
r/w0: Module off.

DGB **DIGITbus Master**
r/w1: Module active.
r/w0: Module off.

CCC1 **Capture Compare Counter 1**
r/w1: Module active.
r/w0: Module off.

LCD **LCD Module**
r/w1: Module active.
r/w0: Module off.

PSLW **Port Slow Mode**
r/w1: Slow mode.
r/w0: Fast mode.

UART0 **UART 0**
r/w1: Module active.
r/w0: Module off.

ADC **ADC Module**
r/w1: Module active.
r/w0: Module off.

TIM1 **Timer 1**
r/w1: Module active.
r/w0: Module off.

XTAL **Quartz Oscillator Mode**
r/w1: Start-Up Mode active (default after Reset).
r/w0: Run Mode active.

SM **Stepper Motor**
r/w1: Module active.
r/w0: Module off.

SPI1 **SPI 1**
r/w1: Module active.
r/w0: Module off.

CAN0 **CAN Module 0**
r/w1: Module active.
r/w0: Module off.

CCC0 **Capture Compare Counter 0**
r/w1: Module active.
r/w0: Module off.

SPI0 **SPI 0**
r/w1: Module active.
r/w0: Module off.

SR1 Standby Register 1

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	x	x	x	3
r/w	x	x	x	x	x	x	x	x	2
r/w	PFM1	PFM0	PWM11	PWM9	PWM7	PWM5	PWM3	PWM1	1
r/w	IRQ	FIQ	x	x	x	CPUM			0
	0x00000001								Res

PFM1 **Pulse Frequency Modulator 1**
r/w1: On
r/w0: Off

PFM0 **Pulse Frequency Modulator 0**
r/w1: On
r/w0: Off

PWM11 **Pulse Width Modulator 11**
r/w1: On
r/w0: Off

PWM9 **Pulse Width Modulator 9**
r/w1: On
r/w0: Off

PWM7 **Pulse Width Modulator 7**
r/w1: On
r/w0: Off

PWM5 **Pulse Width Modulator 5**
r/w1: On
r/w0: Off

PWM3 **Pulse Width Modulator 3**
r/w1: On
r/w0: Off

PWM1 **Pulse Width Modulator 1**
r/w1: On
r/w0: Off

IRQ **IRQ Interrupt Controller**
r/w1: Enabled
r/w0: Disabled

FIQ **FIQ Interrupt Controller**
r/w1: Enabled
r/w0: Disabled

CPUM **CPU Mode**
Clock selection for CPU and peripheral modules (Section “CPU and Clock System”). Change CPUM by 32-bit access only.

6.4. UVDD Analog Section

6.4.1. VBG Generator

The low-power VBG generator generates bias signals which are necessary for the operation of all “UVDD analog section” modules. Furthermore, it produces a reference voltage VBG, that is delivered to the VDD and FVDD regulators, the RESET and ALARM comparators and the UVDD supply supervision.

This module is permanently enabled except during power-saving modes.

6.4.2. VDD Regulator

The VDD regulator generates the 2.5 V VDD supply voltage for the internal core logic from the 5 V UVDD. It derives its reference from the VBG generator.

VDD must be buffered externally by a 220 nF ceramic capacitor in parallel with a 10 µF tantalum capacitor.

This module is permanently enabled except during power-saving modes. A certain set-up time has to elapse after power-up of UVDD for VDD to stabilize. During this time, the supply supervision (cf. 6.4.7.) generates a power-on reset.

An overload condition in the regulator (current or voltage drop-out) generates an immediate reset and is stored in flag ANAU.VE. The immediate overload signal may be routed to the LCK special output by selection in field ANAU.LS.

6.4.3. VDD Auxiliary Regulator

The low-power VDD auxiliary regulator generates a reduced supply voltage for the core logic from the 5 V UVDD.

This module is enabled only during IDLE mode, where no clocked operation is required.

6.4.4. FVDD Regulator

The FVDD regulator generates the 3.3 V FVDD supply voltage for the external Flash memory device from the 5 V UVDD. It derives its reference from the VBG generator.

FVDD must be buffered externally by a 470 nF ceramic capacitor in parallel with a 3.3 µF tantalum capacitor.

This module is permanently enabled except during power-saving modes. A certain set-up time has to elapse after power-up of UVDD for FVDD to stabilize. During this time, the supply supervision (cf. 6.4.7.) generates a power-on reset.
An overload condition in the regulator (current or voltage drop-out) generates an immediate “reset” and is stored in flag ANAU.FVE. The immediate overload signal may be routed to the LCK special output by selection in field ANAU.LS.

6.4.5. ALARM Comparator

The alarm comparator on the pin RESETQ allows the detection of a threshold higher than the reset threshold. An alarm interrupt can be triggered with the output of this comparator.

To obtain a result that is independent from UVDD, the level of pin RESETQ is compared to the VBG reference voltage. The comparator features a small built-in hysteresis. The output constitutes the RESETQ/ALARM interrupt source and must be enabled by setting flag ANAU.EAL. Please refer to section 6.5.1. for functional details.

The interrupt source output is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

The alarm interrupt is a level triggered interrupt. The interrupt is active as long as the voltage on pin RESETQ remains between the two thresholds of the ALARM and the RESET comparator.

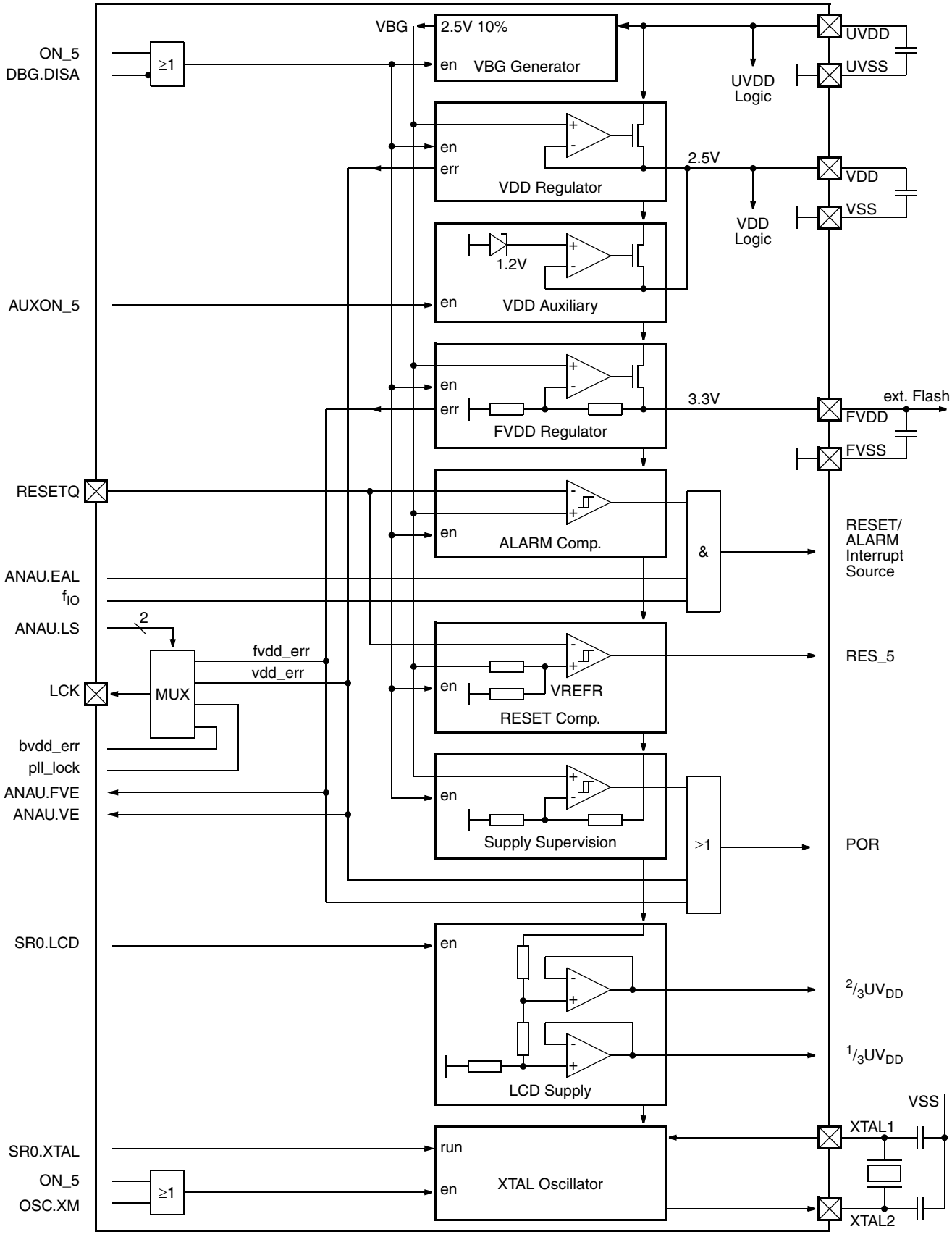


Fig. 6–1: UVDD analog section block diagram

6.4.6. RESET Comparator

As long as the reset comparator on the pin RESETQ detects the low level, the overall IC is reset except the PSM/RTC.

To obtain a result that is independent from UVDD, the level of pin RESETQ is compared to the scaled down VBG reference voltage. The comparator features a built-in hysteresis.

During power-saving modes, the comparator function is not available and is bypassed by a simple CMOS Schmitt input. Full CMOS input levels ($V_{il} = UV_{SS} \pm 0.3$ V and $V_{ih} = UV_{DD} \pm 0.3$ V) are thus required on this input in these modes.

Please refer to sections 6.5.2. and 6.5.3. for functional details.

6.4.7. Supply Supervision

When UVDD drops below a level VREFPOR of approx. 2.8 V, or when the internal VDD or FVDD regulators detect an overload condition, this module generates a power-on reset signal POR that is routed to the Reset Logic.

Refer to section 6.5.2.1. for more details.

This module is permanently active, except during power-saving modes.

6.4.8. XTAL Oscillator

The XTAL oscillator generates a 4 to 5 MHz reference signal from an external quartz resonator, cf. section “Electrical Characteristics” for quartz data.

A reset sets the module to START-UP mode, where, at the expense of a higher current consumption, marginal quartzes receive more drive to ease start-up of oscillation.

After start-up of the CPU program, register SR0.XTAL may be cleared by SW to set the XTAL oscillator to RUN mode, where current consumption is at its standard level.

For operation at UVDD levels between 3.5 V and 4.5 V, continuous operation of the module in START-UP mode may be necessary to guarantee sufficient drive to the connected quartz.

Switching between START-UP and RUN modes must not be done in CPU modes PLL or PLL2, as this might lead to unpredictable behavior of the clock system.

This module is permanently active, except during power-saving modes, where continued operation may be selected for STANDBY and IDLE modes in register OSC.XM.

6.4.9. UVDD Analog Registers

ANAU **Analog UVDD Register**

	7	6	5	4	3	2	1	0	
r/w	EAL	x	LS		LE	x	FVE	VE	
	0	-	0	0	0	-	0	0	Res

EAL **Enable RESET/ALARM Interrupt Source output**
r/w1: Enabled.
r/w0: Disabled.

LS **LCK Output Select**
w0: PLL Lock Signal.
w1: VDD Regulator Error.
w2: FVDD Regulator Error.
w3: BVDD Regulator Error.

LE **LCK Enable**
w1: LCK signal at pin.
w0: PFM1 output at pin.

FVE **FVDD Regulator Error Flag**
r1: Out of specification.
r0: Normal operation.
w1: Reset flag.
w0: No action.

VE **VDD Regulator Error Flag**
r1: Out of specification.
r0: Normal operation.
w1: Reset flag.
w0: No action.

6.5. Reset Logic

6.5.1. Alarm Function

The alarm comparator on the pin RESETQ allows the detection of a threshold higher than the reset threshold. An alarm interrupt can be triggered with the output of this comparator.

The intended use of this function is made, when a system uses a 5 V regulator with an unregulated input. In this case, the unregulated input, scaled down by a resistive divider, is fed to the RESETQ pin. With falling regulator input voltage this alarm interrupt is triggered first. Then the reset threshold is reached and the IC is reset before the regulator drops out.

The time interval between the occurrence of the alarm interrupt and the reset may be used to save process data to non-volatile memory. In addition, power-saving steps like turning off stepper motor drivers may be taken to increase the time interval until reset.

6.5.2. Internal Reset Sources

During CPU-active modes, this IC contains four internal circuits that are able to generate a system reset: watchdog, supply supervision, clock supervision and FHR flag.

During power-saving modes, internal resets are generated that are not routed to pin RESETQ. However, a wake-up from any power-saving mode is treated as a fifth internal reset source and does pull RESETQ low. See chapter “Power Saving Module” for details.

All these five internal resets are directed to the open-drain output of pin RESETQ. Thus a “wired or” combination with external reset sources is possible. The RESETQ pin is current-limited and therefore large external capacitances may be connected.

All internal reset sources initially set a reset request flag. This flag activates the pull-down transistor on the RESETQ pin. An internal reset extension timer starts, as soon as no internal reset source is active any more. It counts 2048 f_{XTAL} periods (for alternative settings refer to HW options register CR) and then resets the reset request flag, thus releasing the RESETQ pin.

As long as the reset comparator on the pin RESETQ detects the low level, the overall IC is reset, except the PSM/RTC.

The state of bits 6:0 in register CSW1, read directly after a system reset, allows to distinguish the cause of this last system reset (Table 6–5).

6.5.2.1. Supply Supervision

A UV_{DD} level below the supply supervision threshold VREFPOR, or an overload condition in the internal VDD or FVDD Regulators will permanently pull the pin RESETQ low and thus hold the IC in reset (see Fig. 6–3 on page 66). This reset source can be enabled/disabled by flag CMA in register CSW0 (see Section 6.5.2.2. on page 65).

6.5.2.2. Clock Supervision

The clock supervision monitors the frequency at the oscillator input XTAL1 and also the frequency f_{SUP} that is present at the input of the central clock divider (see Fig. 4–2).

Fig 6–2 shows how the clock supervision works:
Upon power-up the crystal oscillator starts to build up oscillation on pins XTAL1 and XTAL2. Initially, the internally available f_{XTAL}/f_{SUP} clock may show fluctuations and drop-outs. The internal reset extension timer starts counting f_{XTAL} clocks as soon as ≈20 uninterrupted clocks, having a certain minimum amplitude, are detected, but it is reset when a drop-out period of ≈5 µs appears. When the XTAL oscillation and thus f_{XTAL} and f_{SUP} have been stable over 2048 clocks, the reset extension timer may finish its travel and finally release pin RESETQ.

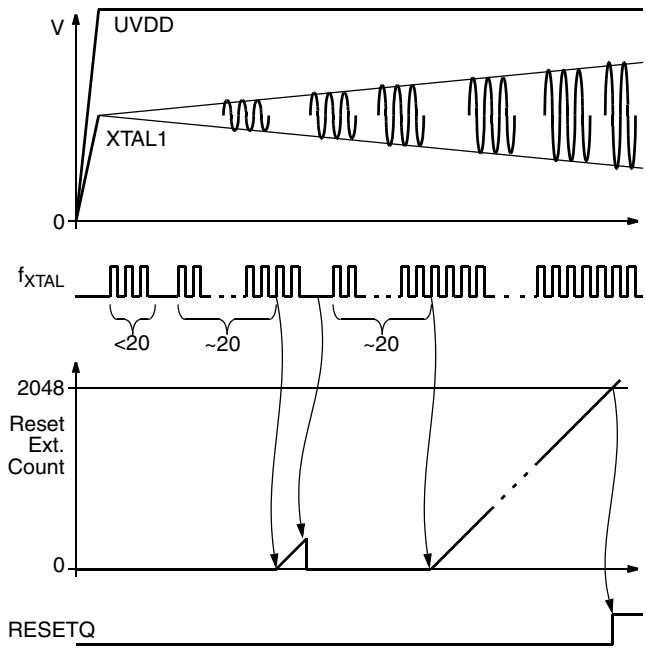


Fig. 6–2: Clock supervision: Principle of operation

Frequencies below the clock supervision threshold of approx. 200 kHz will permanently pull the pin RESETQ low and thus hold the IC in reset (see Fig. 6–3 on page 66). This reset source can be enabled/disabled by flag CMA in register CSW0.

Frequencies exceeding the specified IC frequency are not detected.

Clock and supply supervision are active after reset, but can be enabled/disabled by the clock-monitor-active flag CMA of register CSW0. Setting CSW0.CMA to 0 is recommended for test and evaluation purposes only.

The clock supervision is switched off during power-saving mode. Every time the clock supervision is switched off (VDD Regulator is off) and is switched on again, it outputs a reset signal and sets flag CSW1.CLM.

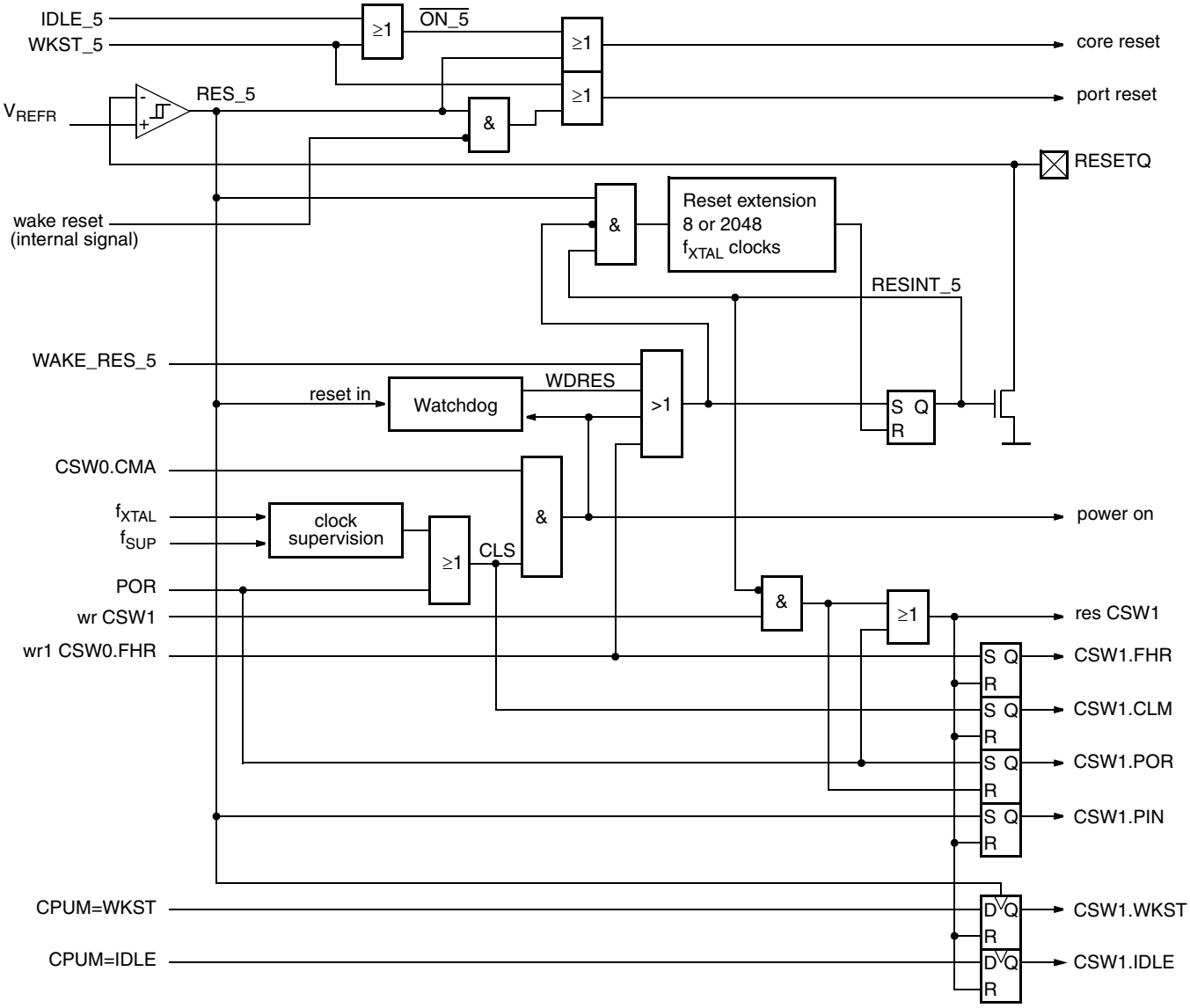


Fig. 6–3: Reset logic block diagram

6.5.2.3. Watchdog

The watchdog module serves to monitor undisturbed program execution. A failure of the program to retrigger the watchdog within a pre-selectable time will pull the pin RESETQ low and thus reset the IC (Fig. 6–3 and 6–4). This reset source is disabled by any POR and any WAKE-Reset. It is only enabled after a write access to location CSW1.

The watchdog (Fig. 6–4) contains a down-counter that generates a reset when it wraps from zero to 0xFF. It is reloaded with the content of the watchdog timer register, when, upon a write access to location CSW1, watchdog trigger registers 1 and 2 contain bit-complemented values. An IC reset resets the watchdog timer register to 0xFF, thus setting the watchdog to the maximum reset interval.

The three watchdog registers are programmed by writes to the same location CSW1. First write the desired watchdog timer value (only values between 1 and 255 are allowed):

$$\text{FAST and PLL modes: Value} = \frac{\text{Interval} \times f_{15}}{1} - 1$$

$$\text{SLOW modes: Value} = \frac{\text{Interval} \times f_{15}}{128} - 1$$

On further writes, to trigger a reload of the counter, alternatingly write a retrigger value (routed to trigger register 1) (not necessarily the former timer value) and its bit-complement (routed to trigger register 2) to CSW1. Failure to reload will result in a counter underflow and a watchdog reset.

Never change the retrigger value. Writing a wrong value to CSW1 immediately prohibits further reloading of the watchdog counter.

The flag CSW1.WDRES is set as soon as the watchdog counter wraps to 0xFF. Thus, it is true after a watchdog reset. Only a supply supervision reset or a write access to CSW1 clears it.

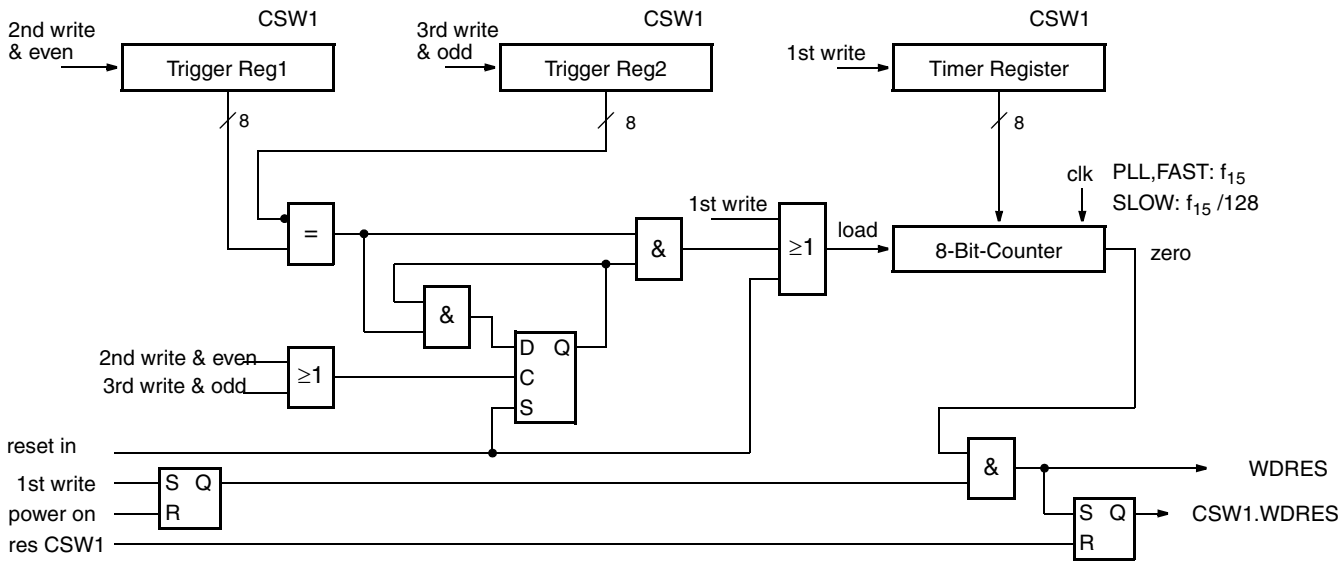


Fig. 6–4: Watchdog block diagram

6.5.2.4. Forced Hardware Reset

Setting flag CSW0.FHR immediately forces the RESETQ pin low. This allows the SW to restart the whole system by HW reset.

6.5.2.5. Wake-Up Reset

Entering one of the power-saving modes sets the corresponding flag in register CSW1, but does not pull RESETQ low. However, wake-up from one of the power-saving modes (signal WAKE_RES_5 in Fig. 6–3) does pull RESETQ low.

6.5.3. External Reset Sources

As long as the reset comparator on the pin RESETQ detects the low level, the overall IC is reset except the PSM/RTC. On this pin, external reset sources may be wire-ored with the IC internal reset sources, leading to a system-wide reset signal combining all system reset sources.

6.5.4. Summary of Module Reset States

After reset the IC modules are set to the reset state (Table 6–4)

Table 6–4: Status after reset

Module	Status
CPU	CPU Fast mode (f_{OSC}).
Interrupt Controller	Interrupts are disabled. Priority registers, request flip-flops and stack are cleared.
U-Ports	Normal mode. Output is tristate.
High current ports	Normal mode. Output is low.
LCD module	Registers are reset. No display.
Watchdog	Inactive after POR and Wake-up. SW may activate. Unaffected in the other cases.
Clock monitor	Active. SW may deactivate.
SRAM, CAN-RAM	If during reset, the power was off, content is undefined. If power was not removed, last state is contained.
PSM	If during reset, the power was off, content is undefined, except registers WSC, OSC, POL and SMX. If power was not removed, only registers OSC (flags RC, XK,XM), POL and SMX are reset.

6.5.5. Reset Registers

CSW0	Clock, Supply & Watchdog Register 0							
	7	6	5	4	3	2	1	0
w	FHR	x	x	x	x	x	x	CMA
	0	x	x	x	x	x	x	1 Res

This register controls the supply and cock supervision modules and allows to force a system reset.

FHR **Forced Hardware Reset**
w1: Reset forced
w0: no action
Enter an idle loop after setting this flag because, depending on the RESETQ external circuit, a number of opcodes will be executed before the reset becomes active.

CMA **Clock and Supply Monitor Active**
w1: Both Enabled.
w0: Both Disabled.
Can be written to zero only after supply or clock supervision reset and before first write access to register CSW1.

CSW1	Clock, Supply & Watchdog Register 1							
	7	6	5	4	3	2	1	0
r	TST	IDLE	WKST	FHR	CLM	PIN	POR	WDRES
	-	0	0	0	0	0	0	0 Res

The Reset state in the register frame above describes the state after a write to register CSW1.

TST **TEST Pin State**
r1: TEST is 1.
r0: TEST is 0.

IDLE **Wake Reset from IDLE Mode** (Tables 6–5, 6–6)

WKST **Wake Reset from WAKE or STANDBY Mode** (Tables 6–5, 6–6)

FHR **Forced Hardware Reset** (Tables 6–5, 6–6)

CLM **Clock Supervision Reset** (Tables 6–5, 6–6)

PIN **RESETQ Pin Reset** (Tables 6–5, 6–6)

POR **Supply Supervision Reset** (Tables 6–5, 6–6)

WDRES **Watchdog Reset** (Tables 6–5, 6–6)

Table 6–5: Source of last hardware reset (CPU active modes)

IDLE	WKST	FHR	CLM	PIN	POR	WDRES	Reset Source
0	0	0	0	1	0	0	external from RESETQ pin
0	0	0	0	1	0	1	internal watchdog
0	0	0	1	1	0	0	internal clock supervision
0	0	0	1	1	1	0	internal supply supervision
0	0	1	0	1	0	0	internal forced hardware
The flags FHR to WDRES sum up the source of all HW resets that occurred since the last write to register CSW1. Any write access to CSW1 resets all flags to 0.							

Table 6–6: Source of last hardware reset (power-saving modes)

IDLE	WKST	FHR	CLM	PIN	POR	WDRES	Reset Source
0	0	0	1	1	0	0	external from RESETQ pin
0	1	x	1	1	x	x	wake-up from WAKE/STBY
1	0	x	1	1	x	x	wake-up from IDLE

CSW1	Clock, Supply & Watchdog Register 1							
	7	6	5	4	3	2	1	0
w	Watchdog Time and Trigger Value							
	1	1	1	1	1	1	1	1 Res

This register controls the watchdog module. See section 6.5.2.3. for details.

6.6. Test Registers

Test registers are for manufacturing test only. They must not be written by the user with values other than their reset values (00h). They are valid independent of the TEST input state.

In all applications where a hardware reset may not occur over long times, it is good practice to force a software reset on these registers within appropriate intervals.

TST1	Test Register 1							
	7	6	5	4	3	2	1	0
w	For testing purposes only							
	0	0	0	0	0	0	0	0 Res

TST2 Test Register 2

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

TST3 Test Register 3

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

TST4 Test Register 4

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

TST5 Test Register 5

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

TSTAD2 Test Register AD2

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

TSTAD3 Test Register AD3

	7	6	5	4	3	2	1	0	
w	For testing purposes only								
	0	0	0	0	0	0	0	0	Res

7. Power Saving Module (PSM)

To further reduce the power consumption one of three power-saving modes (WAKE, STANDBY and IDLE) can be selected. Non-power-saving modes are the CPU-active modes (DEEP SLOW, SLOW, FAST and PLL). Most of the core logic is switched off in a power-saving mode. Only HW necessary for wake-up (and RAM and port logic in IDLE mode) is supplied.

Features

- Power reduction down to 50 µA possible
- Real-time clock module (RTC)
- Clock source: Built-in RC oscillator or XTAL
- Up to 10 edge and level triggered wake ports
- Wake sources: RTC module and/or wake ports
- Polling Module for cyclic scan of the wake ports
- Interrupt outputs of RTC module and port wake module

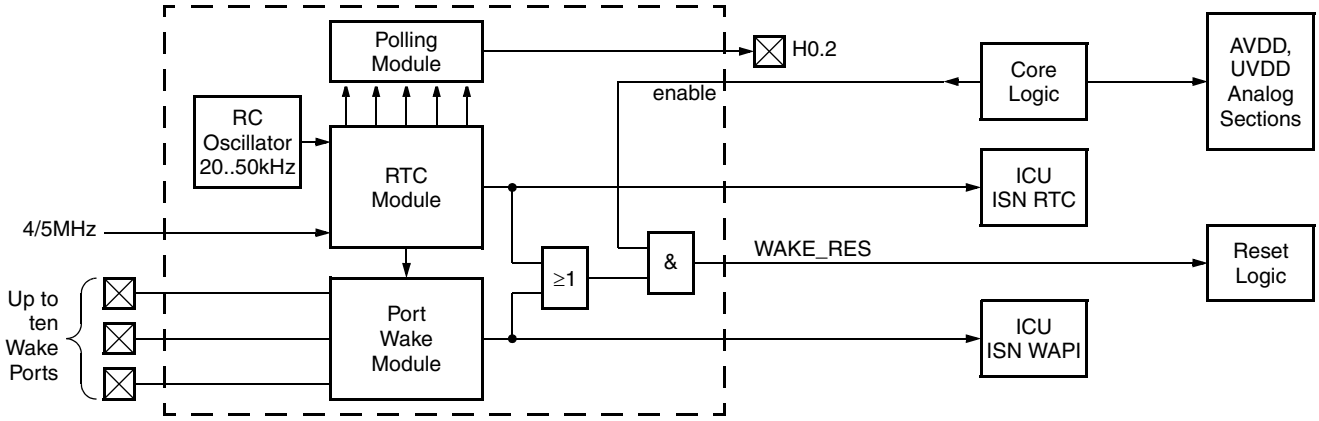


Fig. 7–1: Context diagram

The major task of the power saving module, as shown in figure 7–1, is to supply a wake-up signal (WAKE_RES) for the main system. This signal is necessary to get the IC out of a power-saving mode which otherwise cannot be left by any reset signal with exception of pin RESETQ. For that purpose it combines an RTC module for cyclic wake-up and a port wake module for event-driven wake-up.

The power saving module is active all the time, during power-saving modes as well as during non-power-saving modes (CPU active modes). The WAKE_RES output signal can be generated during power-saving modes only.

The RTC module has to provide the time of day accurately to the second and to generate an output signal which can be used to trigger an interrupt or a wake-up signal. The RTC module can be clocked by an external quartz oscillator or by a built-in RC oscillator.

The polling module cyclically outputs a high pulse of programmable duration at port H0.2. Some of the RTC module taps are connected to the polling module for deriving the pulse period and duration.

The port wake module merges several wake ports and outputs a signal that can be used to trigger an interrupt or a wake-up signal.

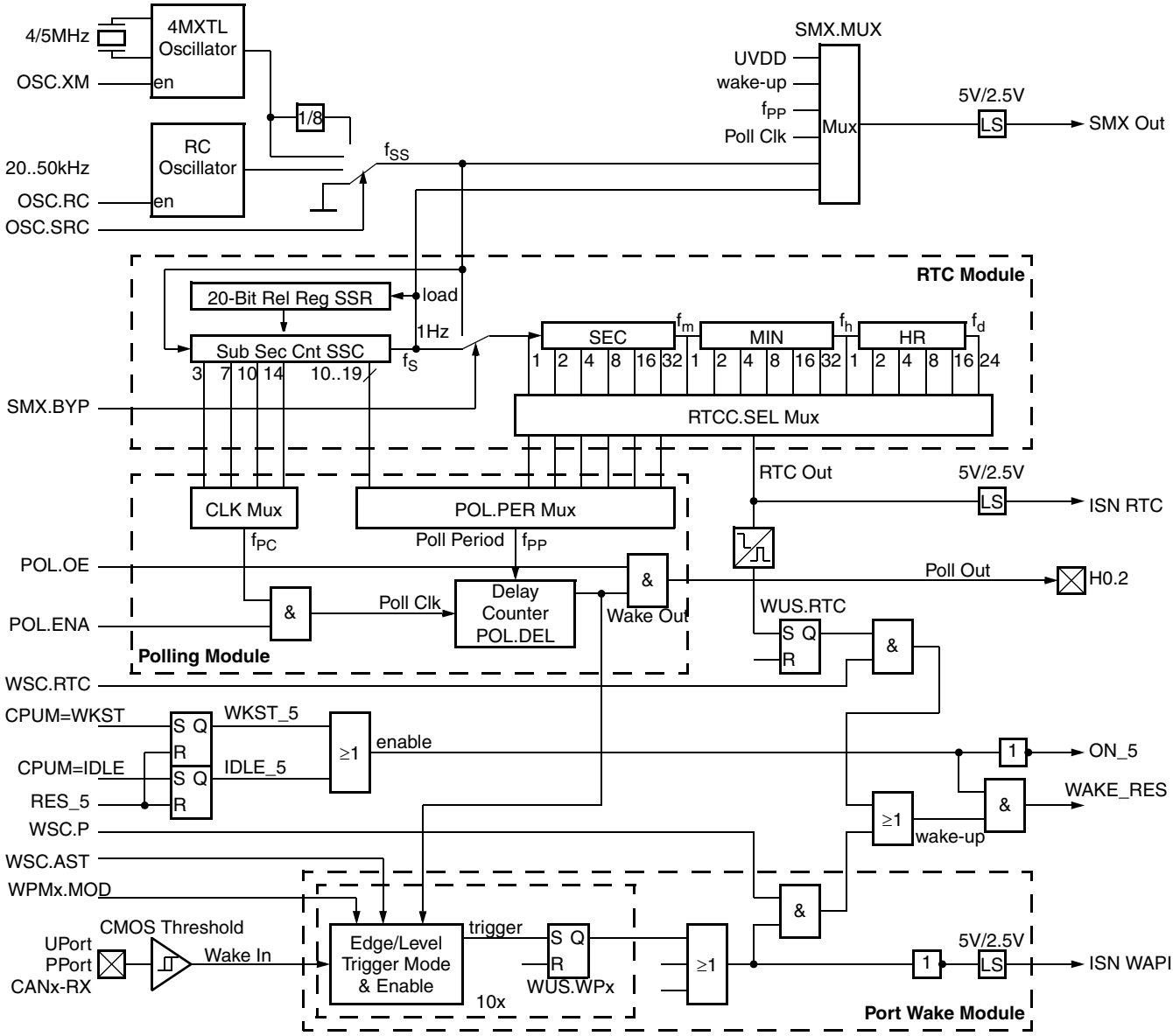


Fig. 7–2: Power-saving module functional block diagram

7.1. Functional Description

The power-saving logic combines all wake-up sources. It is completely supplied by UVDD. It contains an RC oscillator, a 20-bit sub-second counter, an RTC, multiplexers to select different taps of the sub second counter or of the RTC, a polling module and the logic for up to ten wake ports.

The RTC module output can generate an interrupt or a wake-up signal. All wake ports can generate a collective interrupt or a wake-up signal. The polling module can drive an H-Port and generate a strobe signal, which allows a wake port to trigger on a dedicated input level. Several internal clock signals can be output via CO0.

7.1.1. RTC Module

The RTC module is mainly composed of a sub-second counter (SSC) and its reload register (SSR), and the real-time counter's second, minute and hour counters. The input clock for the sub-second counter (f_{SS}) can be selected between the external quartz oscillator clock divided by 8 or an internal RC oscillator clock. The reload value SSR shall be selected so that the output signal of the SSC (f_S) corresponds with a one Hertz clock. The signal f_S is the reload signal for SSC, which is a down counter, as well as the input clock to the second counter (RTC.SEC). The second counter clocks the minute counter (RTC.MIN) which on his part clocks the hour counter (RTC.HR). All of them are up counters.

All stages of the three counters can be selected to generate an interrupt or a wake-up signal.

The RTC cannot be stopped during debug mode by flag CR.STPCLK=1.

7.1.2. Polling Module

The polling module periodically activates the output signal “wake out” which can be enabled by SW to drive port H0.2 (Poll Out). The rising edge of the polling period input (f_{PP}) defines the begin and the polling clock (f_{PC}) input together with the delay counter defines the duration of the high time. This can be used to cyclically flash a LED or to feed a current into an external circuit.

The falling edge of the wake out signal is used to sample the input level of all wake ports (WPx) which are configured for high or low level trigger mode. Those configured ports will set the corresponding WPx flag in register WUS with the falling edge of the strobe signal.

Please refer to figure 7–5 for timing details on the wake out and the strobe signals.

Due to the adjustment mechanism by the 20-bit reload register, the polling period is not always constant. Depending on the reload value, the polling period may vary between 0.5 and 1.5 nominal polling periods at the point of reloading.

The polling period has to be set equal or greater than four times the polling delay.

7.1.3. Port Wake Module

There is a trigger mode logic (level or edge sensitive) and a wake source flag for each wake port. The wake out input is a signal from the polling module. The falling edge generates a strobe pulse which is used to sample the level of the wake In input and set the corresponding wake source flag if necessary. An alternative strobe signal can be used (WSC.AST=1), if there is no RTC subsystem (with polling logic) implemented. The corresponding WPx flag in register WUS will be forced to high as long as the programmed condition (high or low level) is met at the wake port. Please see figure 7–6 for details. The selected strobe signal source is valid for all wake ports. Mixing of the strobe signal sources (polling and alternative) is not possible.

The wake flags of all wake ports are located in the wake-up source register WUS. The trigger events which can set the wake flags can be configured in the wake-up pin mode registers WPM0 to 8 either in field MOD0 or MOD1. Please refer to table 7–7 for details about allocation of mode registers and wake ports.

The output of each wake port is connected to an or gate, whose output can generate a wake port interrupt as well as a wake-up signal.

7.2. Registers

OSC Oscillator Source Register

	7	6	5	4	3	2	1	0	Offs
r/w	RC	XK	XM	x	LD	x	SRC		0
	1	x	1	No HW reset					Res

RC **RC oscillator**
r/w1: enable
r/w0: disable

XK **External 32kHz XTAL (not available)**
r/w1: enable
r/w0: disable
Write to zero for future compatibility.

XM **4MHz XTAL**
r/w1: always enabled
r/w0: disabled during power-saving modes

LD **Load SRC and SSC**
r: Always read as zero
w1: Immediately selects the oscillator source according to SRC and loads the register SSR to the SSC.
w0: No action

SRC **Oscillator Source Select**
r/w0: 4/5MHZ XTAL divided by 8 selected
r/w1: Don’t use, factory test only
r/w2: RC oscillator selected
r/w3: Ground (don’t use for compatibility reasons)
Writing to SRC does not select a new oscillator source immediately. This will be done automatically together with the next reload of the SSC. It can be forced immediately by writing a one to the flag LD with the same write access. A read access returns the current source select, not the requested. Thus read-modify-write instructions have to be used carefully.

SSR Sub Second Reload Register

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	x	x	x	3
r/w	x	x	x	x	Bit 19 to 16				2
r/w	Bit 15 to 8								1
r/w	Bit 7 to 0								0
	No HW reset								Res

For typical settings refer to table 7–1, the values 0 and 1 are not allowed. To avoid programming unexpected values, never write single bytes of the SSR on their own, but always all 3 bytes without being interrupted by SSC read. This is necessary, as for reading SSC the register hardware uses the same buffer registers as for writing SSR. Writing SSR does not load the SSC immediately. This will be done automatically together with the next reload of the SSC. Immediate loading of the SSC can also be forced by setting OSC.LD. Wait one f_{IO} cycle between write and read access to SSR because it lasts one bus cycle until a written value becomes valid.

SSC Sub Second Counter

	7	6	5	4	3	2	1	0	Offs
r	x	x	x	x	x	x	x	x	3
r	x	x	x	x	Bit 19 to 16				2
r	Bit 15 to 8								1
r	Bit 7 to 0								0
	No HW reset								Res

A read access to byte 0 of the SSC latches the bytes 1 and 2. This mechanism grants consistent read access to the SSC.

RTC Real Time Counter

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	x	x	x	3
r/w	x	x	x	HR					2
r/w	x	x	MIN						1
r/w	x	x	SEC						0
	No HW reset								Res

Reading SEC latches MIN and HR. Writing HR simultaneously loads MIN and SEC to the corresponding counters. This mechanism allows consistent read and write access. Since reading and writing use the same latches, do not mix these access types. Wait one f_{IO} cycle between write and read accesses to RTC because it lasts one bus cycle until a written value gets valid.

HR Hour Counter
r/w0 to 23:

MIN Minute Counter
r/w0 to 59:

SEC Second Counter
r/w0 to 59:

RTCC RTC Control Register

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	SEL					0
	No HW reset								Res

SEL Select RTC Output (Table 7–2)

WUS Wake-Up Source Register

	7	6	5	4	3	2	1	0	Offs
r/w	RTC	x	x	x	x	x	WP9	WP8	1
r/w	WP7	WP6	WP5	WP4	WP3	WP2	WP1	WP0	0
	No HW reset								Res

RTC Real Time Clock
r1: RTC was trigger source
r0: No trigger
w1: Clear
w0: No modification

WPx Wake Port x
r1: Wake Port was trigger source
r0: No trigger
w1: Clear
w0: No modification

For proper interrupt generation some peculiarities in operating this register have to be considered. All set bits must be cleared by writing back the whole pattern that was read before. Always read and clear (write back) the whole register (byte 0 first), even if only flags in byte 0 are in use. Every write access to byte 1 will produce an interrupt as long as WUS contains a one.

WPMx Wake Port x Mode Register

	7	6	5	4	3	2	1	0	Offs
r/w	x	MOD1			x	MOD0			0
	No HW reset								Res

MODy Trigger Mode (Table 7–3)
Trigger mode for wake port WPx+y. For assignment of wake port and mode field please refer to table 7–7.

POL Polling Register

	7	6	5	4	3	2	1	0	Offs
r/w	x	CLK		x	PER				1
r/w	ENA	OE	x	DEL					0
	0x00								Res

CLK Select Polling Clock (Table 7–5)

PER Select Polling Period (Table 7–4)

ENA Enable Polling Module
r/w1: enable
r/w0: disable

OE Enable Polling Output
r/w1: enable
r/w0: disable

DEL Select Polling Delay Time
r/w1 to 31: Delay time = DEL/f_{PC}
r/w0: Delay time = 32/f_{PC}
A write access to DEL immediately loads the 5-bit down counter. The delay time defines the duration of the Wake Out signal (Fig. 7–5).

WSC Wake Source Control

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	AST	RTC	P	0
	0x00 after UVDD power-up								Res

AST Alternative Strobe
r/w1: Alternative strobe input
r/w0: Wake out signal from Polling Logic

RTC RTC Wake-up Enable
r/w1: enable

r/w0: disable
Neither WUS.RTC nor the RTC interrupt source output are affected.

P **Port Wake-up Enable**
r/w1: enable
r/w0: disable
Neither WUS.WPx nor the WAPI interrupt source output are affected.

SMX **Signal Multiplexer Register**

	7	6	5	4	3	2	1	0	Offs
r/w	BYP	x	x	x	x	MUX			0
	0x00								Res

BYP **Bypass SSC**
r/w1: RTC is clocked by f_{SS} (test)
r/w0: RTC is clocked by f_S (1Hz)

MUX **Signal Multiplexer** (Table 7–6)
Defines a signal which is output as SMX Out.

Table 7–1: SSR: Typical settings for f_S = 1Hz

XTAL	1/8	Period	Reload Value	Step Resolution
4 MHz	500 kHz	2 µs	500.000	±1 ppm
5 MHz	625 kHz	1.6 µs	625.000	±0.8 ppm
20...50 kHz RC		30.5 µs	32.768	±16 ppm

Table 7–2: SEL usage

SEL	Tap#		Activation
0	Gnd		Never
1	second counter taps	1	Every second
2		2	Every 2 seconds
3		4	Every 4 seconds
4		8	At second 8, 16, 24, 32, 40, 48, 56
5		16	At second 0, 16, 32, 48
6		32	At second 0, 32
7	minute counter taps	1	Every minute
8		2	Every 2 minutes
9		4	Every 4 minutes
10		8	At minute 8, 16, 24, 32, 40, 48, 56
11		16	At minute 0, 16, 32, 48
12		32	At minute 0, 32
13	hour counter taps	1	Every hour
14		2	Every 2 hours
15		4	Every 4 hours
16		8	Every 8 hours
17		16	At hour 0, 16
18		24	Every day
19..31	Don’t use		
SEL = 4, 5, 6, 10, 11, 12 and 17 do not produce isochronous intervals.			

Table 7–3: MOD usage

MOD			Trigger Modes
2	**1**	**0**	
x	0	0	Disabled
0	0	1	Rising edge
0	1	0	Falling edge
0	1	1	Rising and falling edge
1	0	1	High level 1)
1	1	0	Low level 1)
1	1	1	Both levels (every strobe signal) 1)
1) In WAKE mode with alternative strobe only.			

Table 7–4: PER usage

PER	TAP#		f_{PP} (rising edge)
0	Sub Second Counter	10	$f_{SS}/2^{TAP\#}$
1		11	
2		12	
:		:	
9		19	
10	Second Counter	1	1 Hz
11		2	0.5 Hz
12		4	0.25 Hz
13		8	at second 4, 12, 20, 28, 36, 44, 52
14		16	at second 8, 24, 40, 56
15		32	at second 16, 48
Isochronous intervals can only be achieved by PER = 10, 11 or 12.			

Table 7–5: CLK usage

CLK	Sub Sec Tap#	f_{PC}
0	3	$f_{SS}/2^{TAP\#}$
1	7	
2	10	
3	14	

Table 7–6: MUX usage

MUX	Name	
0	UV_{DD}	logic high level
1	f_{SS}	SSC input (calibration)
2	f_S (1Hz)	SSC output (adjustment)
3	(wake-up)	Test (factory use only)
4	(wake-up)	
5	wake-up	
6	f_{PP} 1)	
7	Poll Clk	
1) If POL.PER =10 (1Hz selected), POL.ENA has to be enabled. No 1 Hz pulses can be observed at SMX out otherwise.		

Table 7–7: Wake ports

Name	Basic Funct.	WPMx	MODy	
WP0	U1.7	0	0	
WP1	P1.0		1	
WP2	P1.1	2	0	
WP3	U1.3		1	
WP4	U0.7	4	0	
WP5	U4.3		1	
WP6	U2.1	6	0	
WP7	U6.1		1	
WP8	U8.5	8	0	
WP9	U8.3		1	

7.3. Operation of Power Saving Module

Before entering a power-saving mode, the necessary wake-up sources have to be configured carefully. The reset/wake-up reason in register CSW1 and the wake-up source register WUS have to be cleared. Please see section “CPU and Clock System” for information on entering a power-saving mode.

7.3.1. Configuration of Wake Sources

7.3.1.1. Port Wake Module

If an external event driven wake-up is necessary, the port wake module has to be configured according to section 7.6. The register WUS has to be cleared. Flag WSC.P has to be set, enabling the port wake module output signal to generate a wake-up signal by setting signal WAKE_RES.

If a wake port is to be operated in polling mode (level triggered), configuration of RTC module and polling module is necessary as described in sections 7.4. and 7.5.

7.3.1.2. RTC Module

If a cyclic wake-up is necessary, the RTC module has to be configured according to section 7.4. Flag WUS.RTC has to be cleared. Flag WSC.RTC has to be set, enabling the WUS.RTC output signal to generate a wake-up signal by setting signal WAKE_RES.

7.3.2. Configuration of Interrupts

During CPU active modes, the RTC module and the port wake module can be operated as interrupt sources. The ICU and the corresponding ISNs have to be configured according to section “IRQ Interrupt Controller Unit”.

7.3.3. WAKE/STANDBY

With writing WAKE/STBY to SR1.CPUM, the VDD regulator is immediately switched off and a core and a port reset signal is generated.

A wake-up signal sets WAKE_RES to one, immediately pulling pin RESETQ to low. This sets register SR1.CPUM to FAST, disabling the WAKE_RES output of the power saving module and enables the VDD generator. When VDD is active again, the signal CLS is cleared and the reset extension is started. After the reset extension has finished, the pin RESETQ is released and the core and port reset signals (RES_5, RESPORT, RESCORE) are pulled low. The flag CSW1.WKST is set.

The CPU starts execution at the reset vector and can read the reset/wake-up reason in register CSW1. The wake-up source can be read in register WUS and should be cleared thereafter, otherwise wake port interrupts are not possible.

7.3.4. IDLE

With writing IDLE to SR1.CPUM, the VDD regulator is immediately switched off, the auxiliary VDD generator is switched on and a core reset signal is generated.

A wake-up signal sets WAKE_RES to one, immediately pulling pin RESETQ to low. This sets register SR1.CPUM to FAST, disabling the WAKE_RES output of the power saving module and enables the VDD generator and disables the auxiliary VDD generator. When VDD is active again, the signal CLS is cleared and the reset extension is started. After the reset extension has finished, the pin RESETQ is released and the core reset signal is pulled to low. The flag CSW1.IDLE is set.

The CPU starts execution at the reset vector and can read the reset/wake-up reason in register CSW1. The wake-up source can be read in register WUS and should be cleared thereafter, otherwise wake port interrupts are not possible.

7.3.5. Precautions

The SW has to guarantee the ability of wake-up by careful configuration of the wake logic.

Especially the necessary wake ports have to be configured as input. Verifying the necessary configuration before switching to a power-saving mode will enhance the safety.

Enabling wake-up by pin only is dangerous. Accidentally entering a power-saving mode or if a flag is modified by electrical over stress (EOS) in a power-saving mode, may lead to an IC which can’t be wakened.

From an inadvertently entered power-saving mode, with no wake source being enabled, neither the watchdog nor the clock supervision nor any other internal reset source, but only an external reset on pin RESETQ may recover the device.

As neither the VBG generator nor the RESET comparator are enabled during power-saving modes, proper CMOS input levels ($V_{il} = UV_{SS} \pm 0.3$ V and $V_{ih} = UV_{DD} \pm 0.3$ V) are required on pin RESETQ during a wake-up reset. The external circuitry must allow the device to establish WRV_{il} on that pin.

The specified power-saving mode current consumption values are only obtainable when CMOS input levels ($V_{il} = xV_{SS} \pm 0.3$ V and $V_{ih} = xV_{DD} \pm 0.3$ V) are applied to:

– in IDLE mode all H-, P- and U-Ports

– in WAKE/STANDBY modes all, not only the configured, wake ports.

If an RTC-/Polling module is not used, it is advisable to switch it off to reduce current consumption. Its outputs should be disabled.

7.3.6. Timing

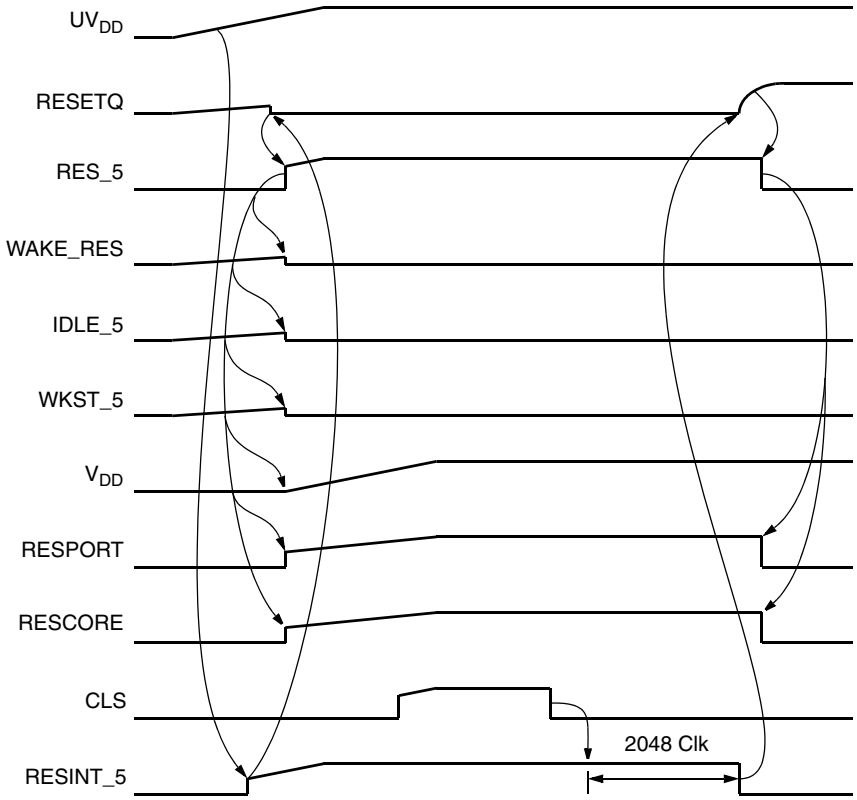


Fig. 7–3: Power-on reset

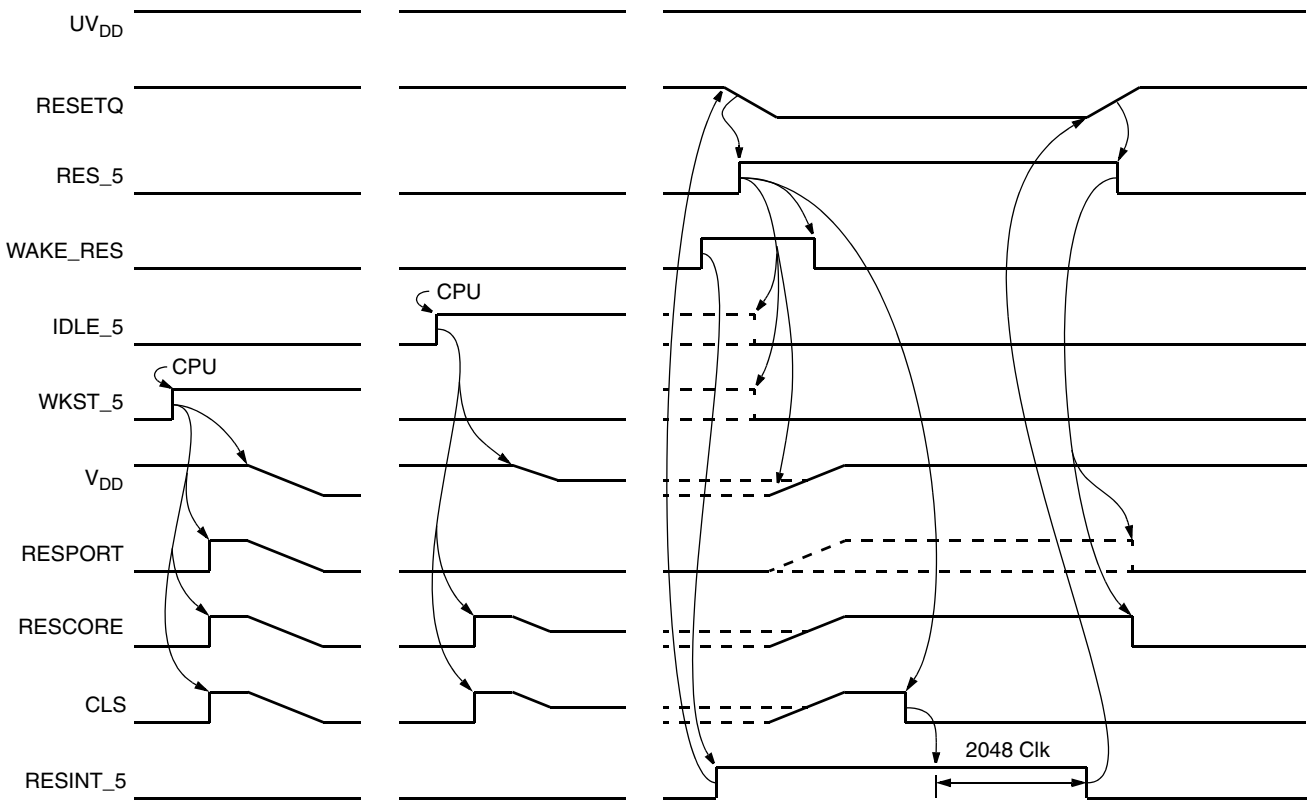


Fig. 7–4: Switching to WAKE/STANDBY and IDLE and wake-up

7.4. Operation of RTC Module

7.4.1. Reset

With the exception of the oscillators, which are enabled by every reset (OSC.RC = OSC.XM = 1), most parts of the logic will never be reset. This means that after every reset the unneeded oscillator has to be switched off. Further, the whole logic has to be initialized after a power-on reset.

An application that does not require RTC and polling module can inhibit the outputs of the modules by writing a 2 to OSC.SRC, a 0 to OSC.RC, a 1 to OSC.LD and a 0 to RTCC.SEL after every reset. This setting also minimizes the power consumption.

7.4.2. Oscillator Source and Sub-Second Counter

The sub-second counter (SSC) generates a 1 Hz output signal at underflow (0x00000 to 0xFFFFF). This signal loads the SSC with the content of the SSR register and switches the oscillator source select multiplexer to the desired oscillator according to the SRC field in the OSC register. This load signal can be forced by writing a one to flag LD in register OSC.

There are three necessities for when to change SSR and OSC.SRC:

– Starting the SSC for the first time (after power on)
Because OSC.SRC is not reset by HW, an oscillator source must be selected and enabled. The SSR has to be loaded with the reload value necessary for a 1 Hz output frequency. Writing the desired oscillator source in field SRC, enabling it if necessary and setting flag LD in register OSC immediately selects the new oscillator source and loads SSR to SSC.

– Changing the SSR
Due to temperature or other dependencies of the oscillator it may be necessary to adjust the reload value in the SSR register from time to time. This can be done within the RTC interrupt service routine. Write the new reload value to the SSR register. Make sure that this will be completed before the next underflow of the SSC, which simultaneously loads the SSC with the new SSR value and also switches the oscillator source with the beginning of the next second.

– Changing the oscillator source
Due to switching to a power-saving mode it may be necessary to change the oscillator source. This can be done within the RTC interrupt service routine. Select the desired oscillator source in OSC.SRC and write the corresponding reload value to the SSR register. Make sure that both will be completed before the next underflow of the SSC, which simultaneously loads the new SSR value to the SSC and switches to the new oscillator source with the beginning of the next second.

Precaution: Changing the oscillator source may cause a fragmentary clock pulse. This may result in wrong SSC and/

or RTC values and unwanted interrupt or wake pulses. Changing the oscillator source makes it necessary to:

1. Disable all output signals of the power saving module (ISN RTC, ISN WAPI, POL.OE=WSC.RTC=WSC.P=0).
2. Switch to the new oscillator source.
3. Initialize SSR, RTC and POL.
4. Clear WUS (WUS = 0xFFFF).
5. Clear the pending flags of the corresponding ISNs.
6. Enable the output signals and interrupts again.

7.4.3. Access to SSC and RTC

SSC and RTC are periodically altered by clock pulses. Even if the 32 kHz subsystem is clocked by the 4/5 MHz oscillator, a CPU access to SSC and RTC may be corrupted by a clock pulse. As this situation cannot be avoided, the SSC or the RTC register have to be read twice. If the difference between the two accesses is illogical, the read has to be repeated. After a write to register RTC it has to be read and compared to the desired value. If there is a difference, write, read and compare have to be repeated. Since the RTC is clocked not faster than in second distance, a read or write access to register RTC can be done in the RTC-ISR. Such an access is safe and guarantees a correct result as long as the RTC-ISR is finished before the next clock alters the RTC.

7.4.4. RTC Output Multiplexer

All the taps of the second, minute and hour counters are connected to a multiplexer (Table 7–2) and can be selected as output by register RTCC.SEL. The output of this multiplexer can generate an RTC interrupt as well as a wake-up signal.

7.4.5. RTC Interrupt

Beneath above described initialization, the ICU and the corresponding ISN has to be initialized.

The RTC has its own ISN and interrupt vector. Thus, further investigation about the interrupt source is not necessary. After an interrupt the flag WUS.RTC is set. The register WUS.RTC is not necessary for, and has not to be handled by the RTC-ISR.

Precaution: Please be aware that modifying RTC or RTCC may result in additional negative edges on RTC Out. If no measures are taken, these edges will generate unwanted interrupts.

It is recommended in this situation to temporarily disable the RTC interrupt. But, in order not to affect the intended interrupts, a 1 Hz clock pulse must be avoided during modification of RTC or RTCC.

A solution that takes care of the above-described situation, is reading SSC and modifying RTC or RTCC only, if sufficient time is available to intercept the unwanted interrupt before the next 1 Hz clock pulse occurs.

7.4.6. Inactivation

Disabling of the RTC module can be done by selecting the RC oscillator (OSC.SRC=2) and disabling this source (OSC.RC=0) by simultaneously writing a one to OSC.LD. Though selecting ground (OSC.SRC=3) would also disable the 32 kHz subsystem, this should be avoided to be compatible with future extensions. Setting RTCC.SEL to zero avoids unwanted interrupts after a non-power-on reset.

7.4.7. Signal Multiplexer

Different internal signals can be switched to SMX Out. The internal signal can be selected via register SMX.MUX. The possible signals are shown in table 7–6. Only the signals f_{SS} and f_S are of general interest, the remaining signals may be used, but are intended for testing purposes.

The signal f_{SS} is useful for measuring the quartz frequency and calculating the corresponding reload value for the sub-second counter with external equipment.

The signal f_S is useful for re-adjustment of the sub-second counter, with the help of, e.g., an XTAL-driven CAPCOM counter, when driven by the internal RC oscillator.

The bypass switch (SMX.BYP) allows to bypass the SSC and directly feed f_{SS} into the second counter. This feature is intended for testing purposes only and is to be written to zero.

7.5. Operation of Polling Module

7.5.1. Reset

The whole logic is reset by every reset, even wake-up from power-saving mode resets the polling module. This means that the logic has to be initialized after every reset.

7.5.2. Initialization and Start

The polling module needs the RTC module to be running, because the polling clock f_{PC} is derived from sub-second counter taps, and the polling period f_{PP} is derived from sub-second counter taps or second counter taps. See section 7.4. for RTC module initialization.

The enable input (POL.ENA) and the output (POL.OE) has to be disabled.

The port H0.2 has to be configured as normal, out, low for operation as polling output.

Select f_{PC} (POL.CLK) and f_{PP} (POL.PER). Enable input and output (POL.ENA, POL.OE) and load the delay counter reload register (POL.DEL) with a non-zero value.

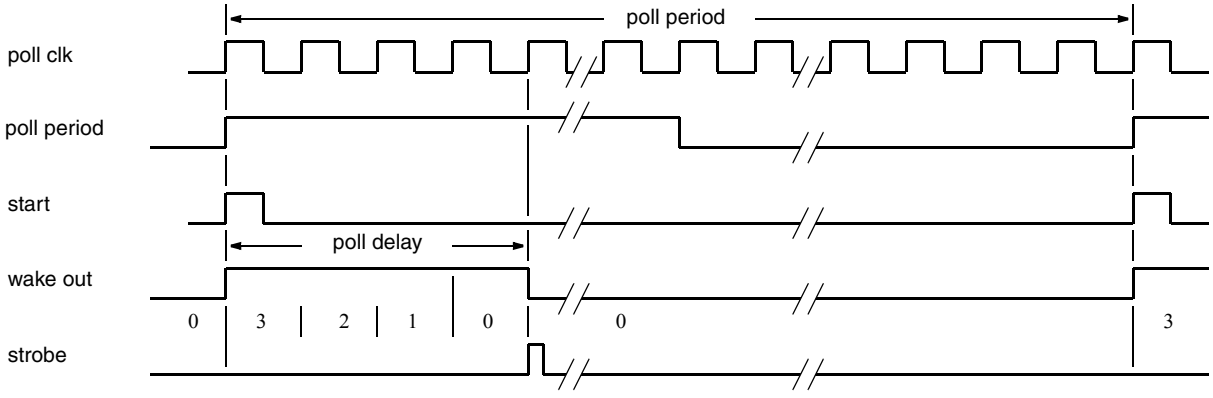


Fig. 7–5: Polling timing

7.5.3. Stop

Disable all inputs and outputs (POL.ENA=0, POL.OE=0).

7.5.4. Restart

Set POL.ENA and POL.OE to one. Load the delay counter reload register (POL.DEL).

7.6. Operation of Port Wake Module

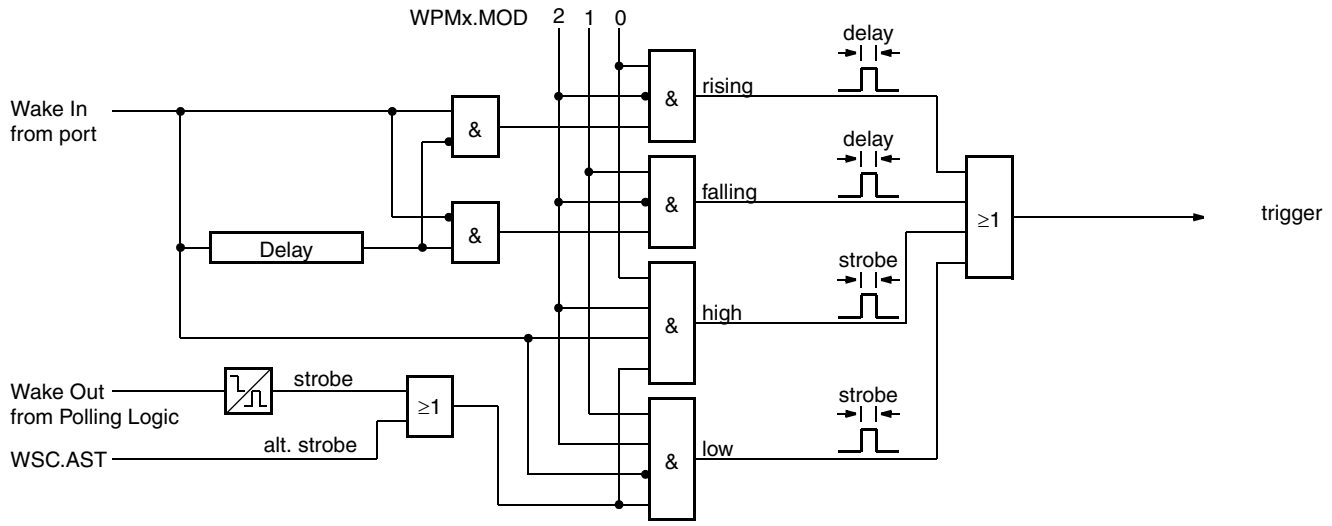


Fig. 7–6: Edge/level trigger logic

7.6.1. Reset

Neither the wake-up source register (WUS) nor the wake-up pin mode registers (WPMx) are reset to a defined value by any reset source.

7.6.2. Initialization

The corresponding ports must be configured normal in.

For every wake port which is to generate a wake-up signal, the trigger mode in the WPMx register has to be programmed. For every wake port not generating a wake-up signal, the trigger mode in the WPMx register has to be disabled. The source of the strobe signal has to be selected by flag WSC.AST if a level triggered mode is to be used. The whole register WUS has to be cleared.

7.6.3. Operation

The port wake module can be operated by polling, it can generate an interrupt (ISN WAPI) or it can be used to generate a wake-up signal to leave a power-saving mode.

To use a level triggered mode of a wake port the RTC module and the polling module have to be configured to provide

the necessary “wake out” signal. The signal “wake out” is necessary for a strobe pulse at the falling edge of “wake out”.

If no RTC/Polling module is available or should not be used, an alternative strobe signal can be used by setting flag WSC.AST to one. As long as WSC.AST is set and the programmed trigger level is applied to the corresponding pin, the flag WUS.WPx is set and cannot be reset by the SW. If more than one wake port is operated with the alternative strobe signal, WPMx.MOD has to be disabled before WUS.WPx can be cleared. Clearing WSC.AST only during the WUS clearing procedure does not help in this case. Interrupts of other wake ports can be lost if the high or low time is too short.

Set WSC.AST to 1 if a wake port is used as CAN bus wake-up, otherwise it is possible that a CAN telegram is always strobed at its high level. In this case the CAN telegram cannot arouse the system.

The selected strobe signal source is valid for all wake ports. Mixing of the strobe signal sources (polling and alternative) is not possible.

If only edge-triggered mode is used, the RTC module and the polling module are not necessary for correct operation of the port wake module.

7.6.4. Wake-up from Power-Saving Mode

In addition to the initialization described above, it is necessary to enable the port wake module as wake-up source by register WSC flag P.

After wake-up the reason can be read in register WUS. It should be cleared after reading, otherwise neither wake-up nor wake port interrupt via ISN WAPI is possible.

7.6.5. Wake Port Interrupt

Beneath above described initialization, the ICU and the corresponding ISN has to be initialized.

All wake ports are directed to a single ISN and interrupt vector. After an interrupt the source can be read in register WUS. It should be cleared after reading, otherwise no further wake port interrupts via ISN WAPI are possible.

Precautions

Parallel usage of a P-port as analog and wake port input is possible, but not recommended. In this case, the Schmitt Trigger input circuit is enabled. This is the reason why input levels other than AV_{SS} and AV_{DD} may cause quiescent currents in the Schmitt trigger circuit and thus lead to higher power consumption.

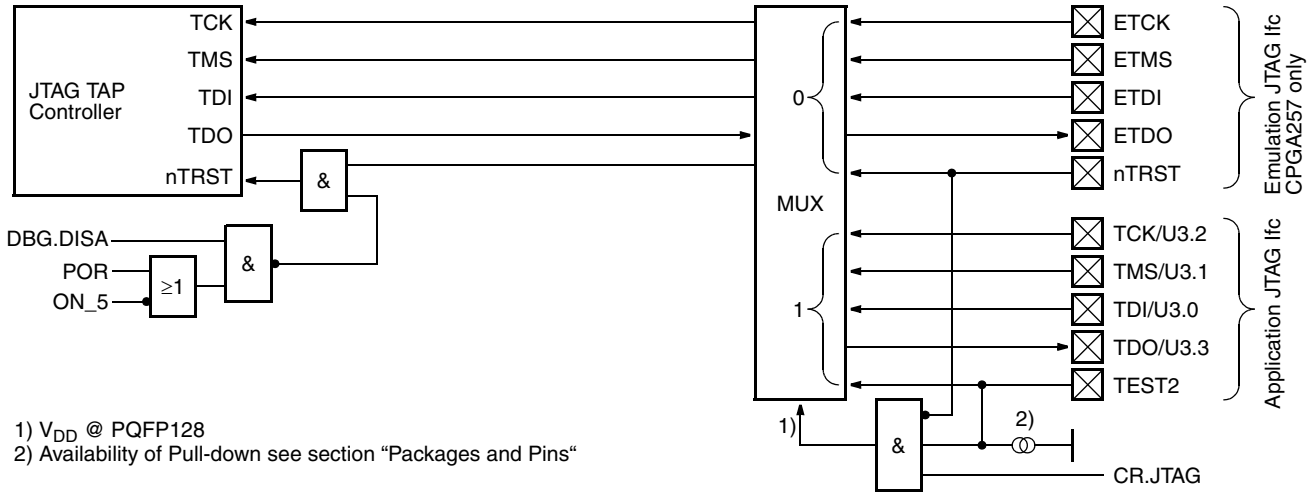
8. JTAG Interface

This module provides JTAG style access to five internal scan chains. These allow testing, debugging, EmbeddedICE and embedded trace module (ETM) programming. The scan chains are controlled by a JTAG style test access port (TAP) controller. For further details on operating the TAP controller, EmbeddedICE and ETM, please refer to the “ARM7TDMI Data Sheet” (Document Number: ARM DDI 0029), “Embedded Trace Macrocell Specification” (Document Number: ARM IHI 0014) and “ETM7 Technical Reference Manual” (Document Number: ARM DDI 0158).

Features

– 2 interfaces selectable
– Access to CPU periphery
– Access to EmbeddedICE
– Access to ETM (embedded trace module)

8.1. Functional Description



1) V_{DD} @ PQFP128
2) Availability of Pull-down see section “Packages and Pins“

Fig. 8–1: JTAG interface block diagram

The TAP controls the access to the scan chains. Scan chain 0 allows access to the entire periphery of the CPU. Scan chain 1 is a subset of the scan chain 0. Scan chain 2 allows programming of the EmbeddedICE debug module. Scan chain 3 is reserved for the boundary scan of the pads of the packaged device. Scan chain 6 allows programming of the ETM.

Table 8–1: Scan chains

Number	Size [Bit]	Function
0	105	ARM7 Macrocell
1	33	Part of scan chain 0
2	38	EmbeddedICE
3	-	reserved for boundary scan
6	40	ETM

Two interfaces can be selected to access the TAP controller. The selection has to be done by the control register flag CR.JTAG and the pins TEST2 and nTRST.

8.1.1. Application JTAG Interface

The application JTAG interface is connected to U-Ports U3.0 to U3.3 and to pin TEST2. The application JTAG interface is available if enabled and the external circuit layout allows it. It is enabled if the TEST2 pin is high, the nTRST pin is low and the flag CR.JTAG is set to one.

In detail, during reset, CR.JTAG is forced to zero and U-Port U3.3 (JTAG TDO) is tristate. If TEST2 pin is high and CR.JTAG is set during operation, U-Port U3.3 is forced to Port, Special, Output mode until programmed by SW. Otherwise the application JTAG interface could not be operated without internal SW support. To avoid conflicts between JTAG mode and SW control on this port bit, never mix these two modes in one application. The application SW must not initialize the involved U-Ports if flag CR.JTAG is set to one.

The flag CR.JTAG is modified by the Special Function ROM too, see section “Special Function ROM” for details.

8.1.2. Emulation JTAG Interface

The emulation JTAG interface is connected to dedicated pins of the emulation parts (CPGA257 package). Series parts (PQFP128 package) do not provide this interface. It is enabled as long as the application JTAG interface is disabled.

8.1.3. Boundary Scan

The boundary scan is not implemented in this IC.

8.1.4. Pin TEST2

Refer to section “Electrical Characteristics” for details. Besides JTAG, the pin TEST2 controls the behavior of the IC during reset. Refer to section “Core Logic” for further details.

8.1.5. External HW Requirements

The external circuit has to make sure that the TEST2 pin is driven to the required level during reset and switched to the JTAG probe after reset so that the pin can be controlled by the probe. This can be done by connecting a driver between JTAG probe and TEST2 pin and disabling the driver during reset. The reset state can be detected by sensing the RESETQ pin. The threshold voltage of the external logic must be higher than the minimum recommended reset inactive input voltage RV_{im} (1.5 V) of this IC. Use, for example, the 74HC14 Schmitt Trigger (not the HCT variant!). Figure 8–2 shows an example of the external circuit.

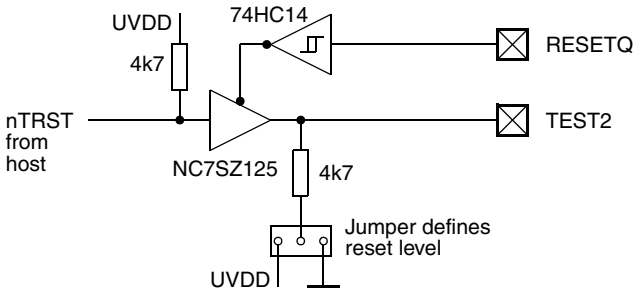


Fig. 8–2: TEST2 pin external circuit diagram for application JTAG interface

8.2. Registers

DBG	Debug Register								
	7	6	5	4	3	2	1	0	Offs
w	x	x	x	x	x	x	x	DISA	0
	0x01 (after UVDD power-up)								Res

DISA **Disable Voltage Generators, JTAG/ETM**

w1: Standard: Debug data lost during power-saving modes because supply is disabled.

w0: Survive mode: Regulators don’t shut off power during power-saving modes, and debug data survive wake reset.

8.3. External Circuit Layout

The emulation JTAG interface uses TTL level input comparators. The emulation JTAG inputs ETCK, ETMS, ETDI and nTRST need external pull-up resistors to EVDD. This has to be done in such a way that the TAP controller sees a logic one if the emulation JTAG interface is enabled but not driven.

The application JTAG interface shares its input and output pins with the I/O of U-Ports. The external circuit layout has to be done carefully in order to guarantee functionality of the JTAG interface. As long as the application JTAG interface is enabled and not driven, the TAP controller inputs TMS and TDI must see logic one level. If it is enabled and driven, the external application circuit shall not influence proper operation of the JTAG interface. The external host must be able to drive the levels at the inputs TCK, TMS and TDI to CMOS logic one and logic zero levels and it must be the only source of these signals. The TAP controller must be able to drive the output TDO to both CMOS levels, logic one and zero, and must be the only source of this signal.

Common JTAG tools expect to see pull-up resistors at nTRST (TEST2), TCK,TMS and TDI.

8.4. JTAG ID

The JTAG ID is not implemented in this IC.

The JTAG TAP controller contains a HW coded JTAG ID which can be read serially via the JTAG interface. The CPU can’t access this ID.

Bits 1 to 19 are manufacturer-defined. Bits 0 and 20 to 31 are ARM-defined.

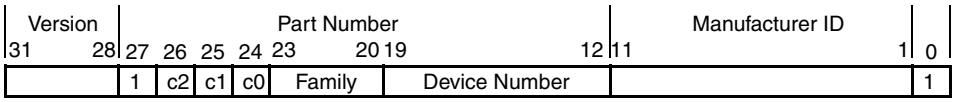


Fig. 8–3: JTAG ID format

Bit 31 to 28 Version
0: ARM core revision 0.
1: ARM core revision 1.
2: etc.

Bit 27
0: ARM core ID.
1: Non ARM core ID.

Bit 26 Capability bit 2
0: Standard part.
1: ‘E’ part.

Bit 25 Capability bit 1 (Reserved)

Bit 24 Capability bit 0
0: Hard macro.
1: Synthesizable.

Bit 23 to 20 Family
7: ARM7.
9: ARM9.
A: ARM10.
etc.

Bit 19 to 12 Device Number
Manufacturer device number 0 to 255.

Bit 11 to 1 Manufacturer ID
Manufacturer ID is the compressed JEDEC code (0x06C).

Bit 0 Marker
Fixed value.

The necessity of using bits 19 to 12 as manufacturer device number forces us to use the Non-ARM core ID format of the JTAG ID. This is the reason why some debug tools cannot use auto configuration, but must be configured by the user for the correct core and revision.
The part number shall be selected in a way that no two component types in the same package with TAP pins in the same location have the same part number.

9. Embedded Trace Module (ETM)

This module provides instruction and data trace capability. The ETM is controlled by a JTAG style test access port (TAP) controller. For further details on the installed Rev1A please refer to the “Embedded Trace Macrocell Specification” (Document Number: ARM IHI 0014) and the “ETM7 Technical Reference Manual” (Document Number: ARM DDI 0158).

Features

- Instruction trace
- Data trace
- Trace before, about, after trigger
- Trigger and filter capabilities
- Access to “Embedded Trace Module”
- Normal trace data format
- Full-rate and half-rate clocking
- 4/8/16-bit maximum port width

9.1. Functional Description

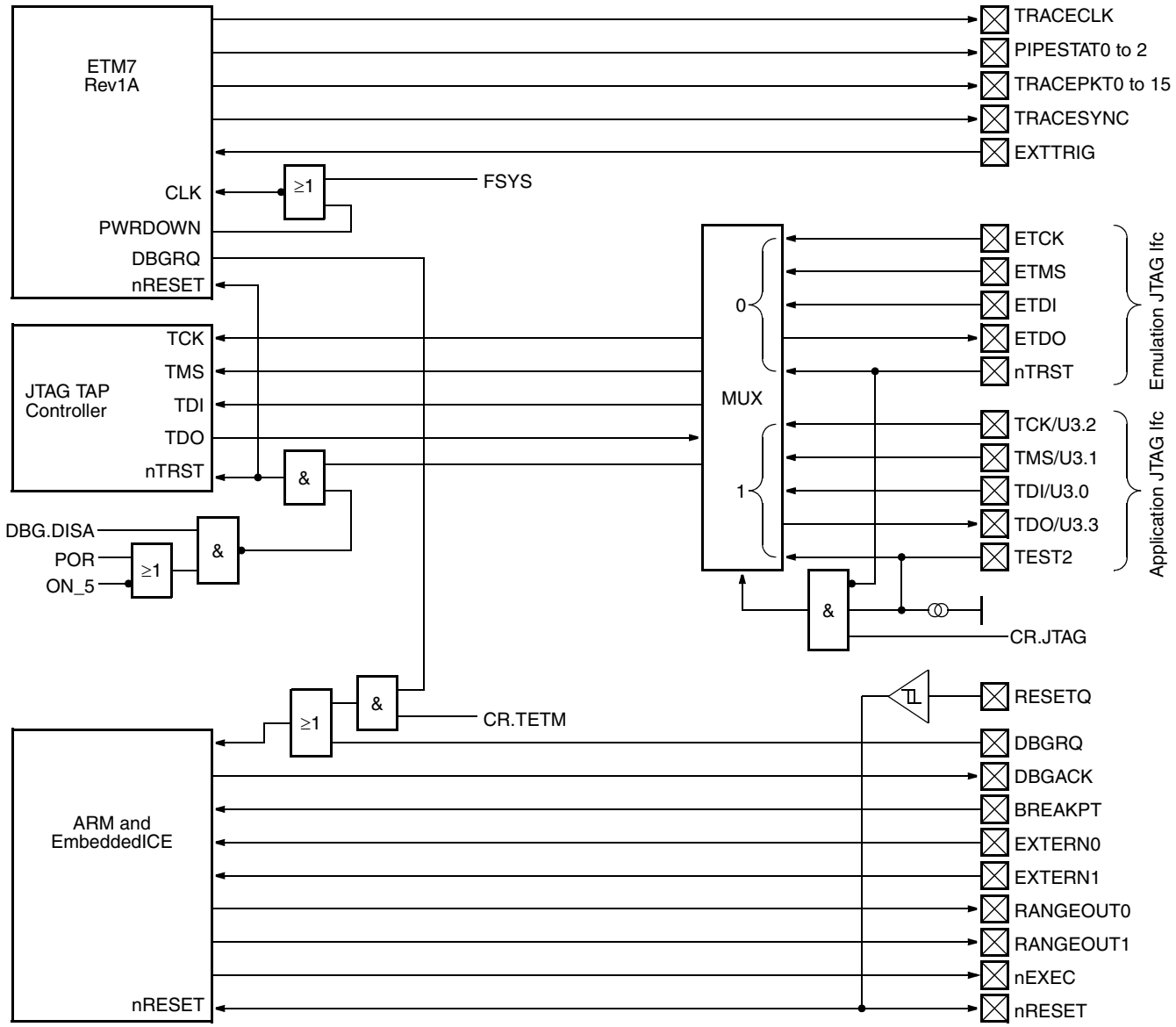


Fig. 9–1: ETM interface block diagram

The ETM is controlled via scan chain 6 of the JTAG interface.

The process of remapping or loading code to RAM and executing there, is a problem for the ETM because one address can contain different code (overlay). The solution is based on the requirement that the memory map into which overlays are loaded, exists in multiple places in the address space.

The memory controller of the IC decodes the 24 LSB address lines A0 to A23. This results in a memory map of 16 MByte. This memory map is repeated 256 times within the 32-bit ARM core address space of 4 GByte.

Thus it is possible to have one static image of the code being executed for the trace tool, with different possible overlays statically linked into the appropriate area of the address space.

Loading code into the RAM and executing it there means, copying the code into the RAM and then jumping to its overlay. The ETM sees the full 32-bit address and reports this jump to the trace tool which has the static image with a memory map for each configuration at different places of its address space. The memory controller sees the 24 lower address lines only, therefore the jump is directed to the correct location.

The supported trace features are listed in Table 9–1.

Table 9–1: Trace features

Features	Supported
Demultiplexed trace data format	-
Multiplexed trace data format	-
Normal trace data format	✔
Full-rate clocking	✔
Half-rate clocking	✔
Maximum port width	4/8/16-bit

10. Memory Patch Module V1.0

The memory patch module allows the user to modify data words within the ROM/Flash address space without changing the contents of the ROM/Flash itself. This function is useful if faulty parts of software or data are detected after the ROM code has been cast into mask ROM.

An application software in ROM has to be prepared to support patching. It loads addresses and code or data, e.g., from external non-volatile memory to program the appropriate registers of the module. After this kind of patch module initialization, code and/or data of the specified address locations will become replaced by the contents of the patch data registers upon address matches.

Single ROM locations are replaced directly. Longer, that is, faulty code sequences may be bypassed by introducing a jump-to-a-code sequence or a call of a subroutine in RAM. The program part in RAM ends with a return or jump to the appropriate address in ROM. With it, even complex software parts may be replaced.

Features

– patching of up to 10 different (32-bit) words in ROM address space

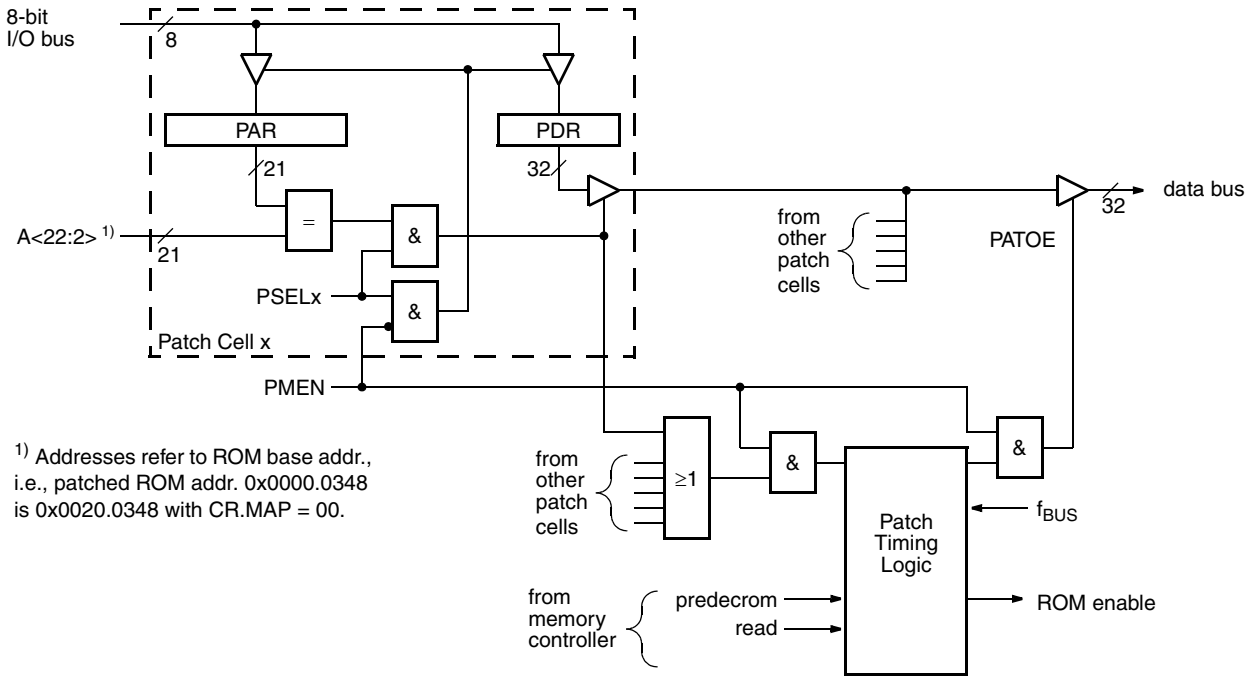


Fig. 10–1: Block diagram

10.1. Principle of Operation

10.1.1. General

The logic contains up to ten patch cells (see Fig. 10–1 on page 89), each consisting of a 21-bit compare register (patch address register (PAR)), a 21-bit address comparator, a patch select bit (PSELx) in the patch enable register (PER) and a 32-bit patch data register (PDR).

The current address information for a ROM access is fed to a bank of patch cells. In case of a match in one patch cell, and provided that the corresponding patch enable register bit is set, the module's logic disables the ROM data bus drivers and instead places the data information from the corresponding patch data register on the data bus.

10.1.2. Initialization

After reset bit PER.PMEN is reset to 0 and patch operation is disabled. All patch cell registers are in write mode and may be programmed.

At first, select a cell by setting the corresponding PSELx bit in register PER (PSELx = 1). Then feed the address into register PAR and the accessory 32-bit patch data into register PDR.

If desired, repeat the above sequence for further patch cells. Make sure to have only one PSELx bit set at a time, although more than one register may be selected and programmed with a single access. This is possible because of used subaddressing for the address and data registers, but is probably of very rare use indeed.

10.1.3. Patch Operation

To activate a number of properly initialized patch cells for ROM code patching, set all the corresponding PSELx bits in registers PER and set bit PER.PMEN to 1 at last.

The memory patch module will immediately start comparing the current address to the setting of the enabled patch cells. In case of a match the ROM data will be replaced by the corresponding patch cell data register setting.

If several PDRs have been initialized with different data but same PAR contents, the patch result is unpredictable.

Patched data can not be observed externally on the trace bus because the patch module would deliver the gating signal too late for the logic to respond accordingly. Use the ETM to trace instead.

10.1.4. Reconfiguration

To reconfigure the memory patch module, first set PER.PMEN to 0. The module will immediately terminate patch operation.

Then proceed as described in 10.1.2. on page 89.

10.2. Registers

PAR Patch Address Register

	7	6	5	4	3	2	1	0	Offs
w	x	x	x	x	x	x	x	x	3
w	x	A22 to A16							2
w	A15 to A8								1
w	A7 to A2						x	x	0
	0x00FFFFFF								Res

PDR Patch Data Register

	7	6	5	4	3	2	1	0	Offs
w	D31 to D24								3
w	D23 to D16								2
w	D15 to D8								1
w	D7 to D0								0
	0x00000000								Res

PER Patch Enable Register

	7	6	5	4	3	2	1	0	Offs
w	x	x	x	x	x	x	x	x	
w	x	x	x	x	x	x	x	x	
w	x	x	x	x	x	PSEL9	PSEL8	PSEL7	
w	PSEL6	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	PSEL0	PMEN	
	0x0000								Res

PSEL0 to 9 Patch Cell Select
w1: select cell for write or enabled for patch
w0: de-select cell for write or disable for patch
Before writing compare address or replace data of a patch cell, only one cell must be selected. In compare mode one or more patch cells can be selected.

PMEN Patch Mode Enable
w1: enable patch mode of all cells
w0: enable write mode of all cells

ICs which are derived from the Emulator IC may have less patch cells. In these cases use always cells upwards from cell 0.

11. IRQ Interrupt Controller Unit (ICU)

The interrupt controller unit (ICU) manages up to 63 interrupt sources. Each interrupt source has its own interrupt vector pointing to an interrupt service routine. One of 16 priorities can be assigned to each channel, or it can be disabled. The interrupt controller unit is connected to the nIRQ input of the CPU.

Features

- Expanding nIRQ input of ARM7TDMI
- Up to 63 interrupt sources (39 implemented)
- 16 priority levels
- HW vectoring
- Vector table switching
- HW prioritizing
- HW stacking of priority levels
- 2 cycles maximum from input to CPU nIRQ

11.1. Functional Description

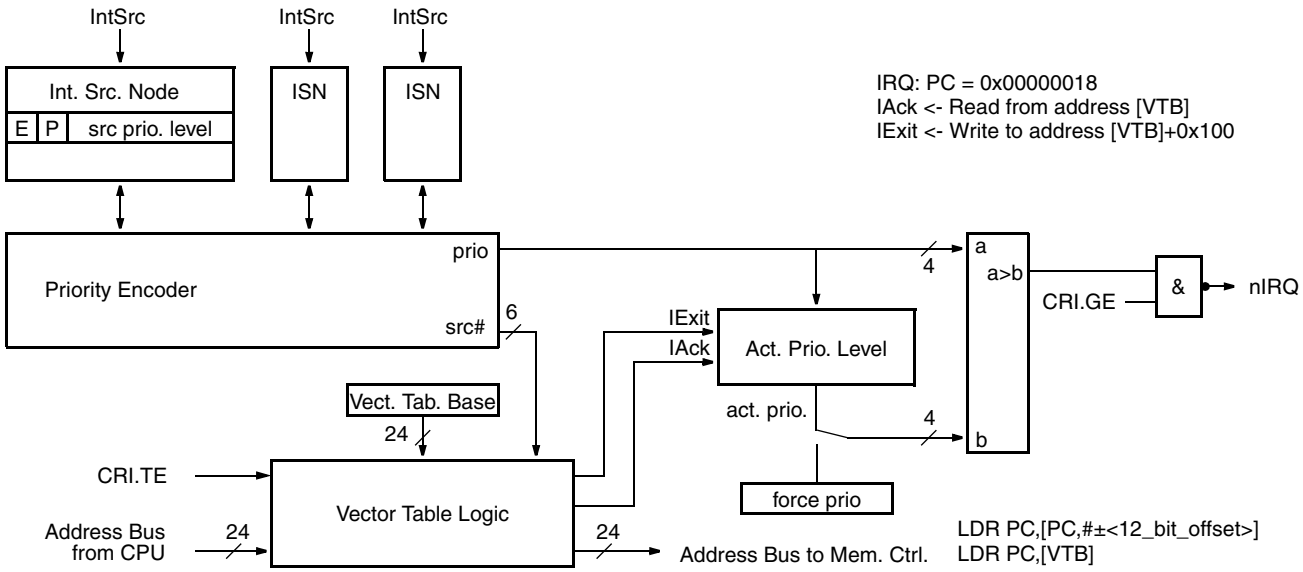


Fig. 11–1: Block diagram

The interrupt controller unit (ICU) is composed of an interrupt source node (ISN) for each interrupt source, of a priority encoder, of a vector table logic, of an active priority level logic and a comparator (Fig. 11–1).

Each falling edge of an interrupt source signals an interrupt request to its ISN and sets its pending flag "P" (Fig. 11–2). Apart from the flag P, each ISN consists of an enable flag (E), and a source priority register containing the priority of the corresponding interrupt source. As long as both flags (E and P) are true, the ISN outputs its priority. Otherwise it outputs the lowest priority (that is no priority).

The priority encoder outputs number and priority of the ISN with the highest active priority. If several ISNs with the same priority are active at the same time, the ISN with the lowest source number is selected, thus the ISNs are operated in a HW-defined order. The interrupt vector table contains the start addresses of the interrupt service routines (ISR). The vector table base register points to the first entry of the interrupt vector table. Thus the location of the interrupt vector table is programmable to any memory location. This allows easy switching between different tables.

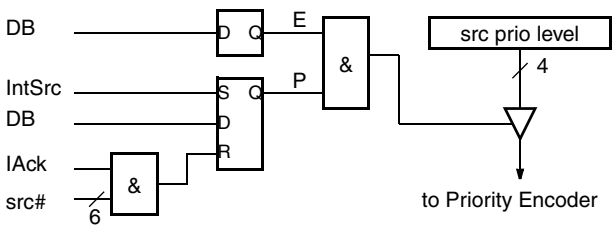


Fig. 11–2: ISN flags

The active priority level logic outputs the priority of the currently running task (lowest priority is the background task). The comparator activates its output, if this priority is lower than the priority output from the priority encoder.

If the ICU's output is enabled by the global enable flag (GE) the nIRQ input of the CPU is activated and held active until acknowledged. The IRQ is granted by the CPU as soon as the CPU internal IRQ flag (flag I in CPU register CPSR) is enabled by SW. In the meantime, between interrupt activation in the ICU and granting by the CPU, higher priority interrupt requests may be signaled to the ICU, raising the priority output of the priority encoder.

The SW has to read the address where the vector table base register points to ([VTB]) in order to get the start address of the ISR for the ISN with the currently highest active priority (Fig. 11–3). A data fetch from this location generates an internal interrupt acknowledge signal (IAck). With IAck the active priority level logic accepts the new priority and internally saves the priority of the interrupted task. IAck clears the P flag in the corresponding ISN and deactivates the nIRQ output. The outputs of the priority encoder (registers PEPRIO and PESRC) change their values immediately and show source number and priority of the next pending interrupt. The ICU is ready for new interrupts now.

Before leaving the interrupt service routine, the SW has to write to the address where VTB points to plus 0x100 ([VTB]+0x100). A write to this location generates an internal interrupt exit signal (IExit). With IExit the active priority level logic internally deletes the priority of the current task and outputs the priority of the interrupted task where the immediately following return instruction jumps to.

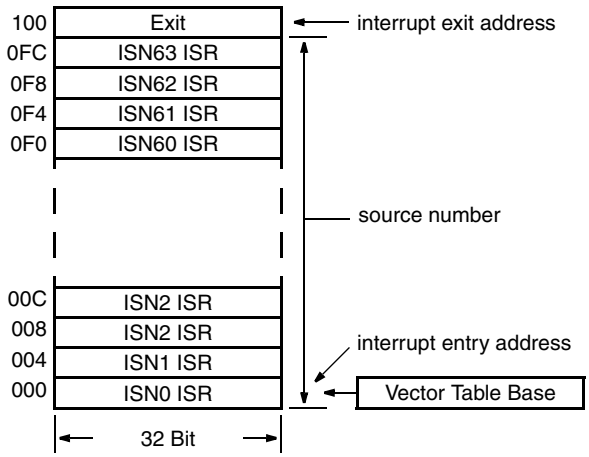


Fig. 11–3: Interrupt vector table

Each ISN has a dedicated source number. A maximum of 64 ISNs can be connected to the priority encoder. The priority encoder outputs source number 0 as long as all ISNs output priority 0 or no ISN is active or the comparator output is inactive. This is to guarantee a valid state of the priority encoder and return a valid start address even if no interrupt source is active. The corresponding vector is the first in the vector table (default vector). For this reason ISN0 can't be used for connecting an interrupt source, because ISN0 is not the only user of the corresponding interrupt vector. For example, reading the address location pointed to by the vector table base register while nIRQ is inactive, will return the ISR start address of ISN0.

In addition to the ISNs whose inputs are connected to a HW module, some ISNs are necessary whose inputs are not connected or unused. Those interrupts can be activated by SW solely (delayed interrupt).

The output of the active priority level logic may be forced by writing a higher priority to the forced priority register. This allows temporary raising of the priority of the currently running task. Writing the maximum priority to the forced priority register is another way to disable the ICU because no ISN can generate an IRQ. Raising of the priority in this way does not take effect as long as nIRQ is active.

The global enable (GE) signal at the output of the comparator disables the ICU output. It is impossible to inactivate an active nIRQ output by modifying an ISN, the GE flag or forcing the priority. Only IAck resets the nIRQ output.

The size of the ICU can be scaled in steps of 8 ISNs. This IC has 40 interrupt source nodes implemented (ISN0 to ISN39). Derived parts can contain 40, 32 (ISN0 to ISN31), 24 (ISN0 to ISN23) or less ISNs.

The pending flags P in the ISNs operate even when the ICU is disabled (CRI.GE = 0). To be exact, only the ICU output is disabled. This avoids further interrupts. Interrupted ISRs will be finished and the Act. Prio. Level stack will be handled properly if those ISRs generate IExit before returning.

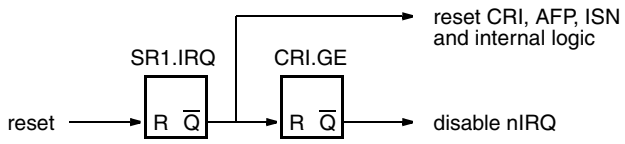


Fig. 11–4: Reset structure

Figure 11–4 shows the reset structure. Registers can't be written until the IRQ flag in the standby register SR1 is set. The pending flags P in the ISNs are not reset by the standby register. It can be operated by HW even while SR1.IRQ is zero. Reading and writing of the P flags is impossible unless SR1.IRQ is set to one.

Table 11–1: Interrupt assignment

ISN	Interrupt Source
0	Default vector, not connected
1	CC0OR
2	CC1OR
3	PINT0
4	PINT1
5	CAN0
6	SPI0
7	Timer 1
8	Timer 0
9	P06 COMP
10	RESET/ALARM
11	WAIT COMP
12	UART0
13	PINT2
14	WAPI
15	CC2OR
16	CC3OR
17	Timer 2
18	RTC
19	I2C0
20	Timer 3
21	SPI1
22	COMMRX (buffer not empty) ored with COMMTX (buffer empty)
23	PINT5
24	PINT3
25	DIGITbus
26	I2C1
27	CAN1
28	CC4OR
29	CC5OR
30	Timer 4
31	UART1

Table 11–1: Interrupt assignment

ISN	Interrupt Source
32	CAN3
33	CAN2
34	CC0COMP
35	CC1COMP
36	CC2COMP
37	CC3COMP
38	PINT4
39	GBus

11.2. Timing

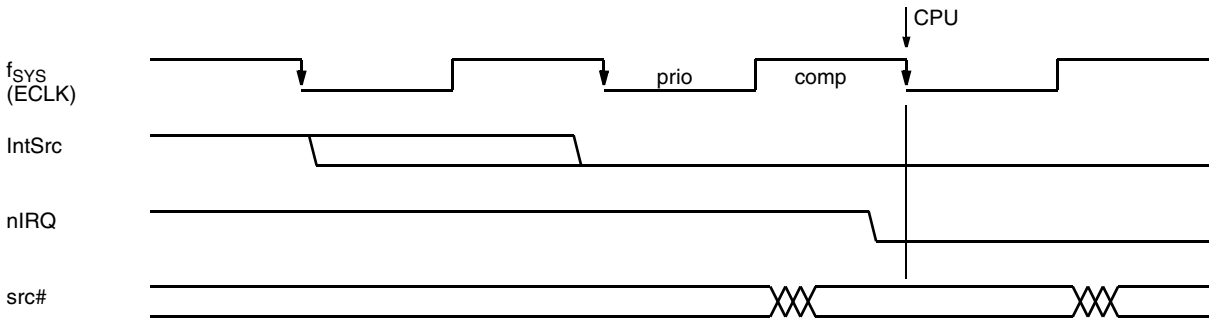


Fig. 11–5: Timing

The sample period of an incoming interrupt request lasts one cycle in the worst case. It is sampled by an ISN with the falling edge of f_{SYS}. Priority Encoder and comparator require another cycle. The CPU finally evaluates with the next falling edge of f_{SYS}.

This results in a maximum delay of 2 f_{SYS} cycles from request to CPU input.

11.3. Registers

CRI **Control Register IRQ**

	7	6	5	4	3	2	1	0	
r/w	GE	TE	x	x	x	x	x	x	
	0	0	x	x	x	x	x	x	Res

GE **Global Enable**
r/w1: Enable IRQ.
r/w0: Disable IRQ.
Disabling happens as soon as nIRQ is inactive. An active nIRQ will not be interrupted by writing a zero to GE.

TE **Table Enable**
r/w1: Enable.
r/w0: Disable.
The Vector Table Logic doesn't work if TE is disabled. Neither the correct ISR start address is returned nor the internal signals IAck and IExit are generated on accessing the dedicated memory location.

AFP **Actual and Forced Priority Register**

	7	6	5	4	3	2	1	0	
r/w	APRIO				FPRIO				
	0	0	0	0	0	0	0	0	Res

APRIO **Actual Priority**
r: (Table 11–3)
This field indicates the programmed priority of the actually running ISR. It is modified by HW only.

FPRIO **Forced Priority**
r/w: (Table 11–3)
Writing a value higher than the APRIO value to this location raises the priority of the actual running ISR. It doesn't change APRIO. Only ISRs with a priority higher than the forced priority are able to interrupt now.

It is necessary to first save the original FPRIO value before raising the own priority by overwriting FPRIO. The saved FPRIO value has to be restored before ISR exit.

PEPRIO **Priority Encoder Priority output**

	7	6	5	4	3	2	1	0	
r	x	x	x	x	Priority				
	x	x	x	x	0	0	0	0	Res

This register shows the priority of the highest pending and enabled interrupt source.

PESRC **Priority Encoder Source output**

	7	6	5	4	3	2	1	0	
r	x	x	Source						
	x	x	0	0	0	0	0	0	Res

This register shows the number of the highest pending and enabled interrupt source.

VTB	Vector Table Base								
	7	6	5	4	3	2	1	0	Offs
r/w	0	0	0	0	0	0	0	0	3
r/w	Address bit 23 to 16								2
r/w	Address bit 15 to 9							0	1
r/w	0	0	0	0	0	0	0	0	0
	0x00000000								Res

The register VTB has to be programmed with the memory base address of the interrupt vector table. The interrupt vector table has to start at an even page address (9 LSB are zero) and is not longer than one page (256 bytes). Apart from the start address of the interrupt vector table, VTB defines two addresses which perform HW actions when accessed and CRI.TE is set.

Every word read access to the location addressed by VTB deactivates the nIRQ output. If the comparator output is active, the internal signal IAck is activated, which returns the ISR start address of the ISN with the highest active priority, clears the corresponding P flag and saves the interrupted priority.

Every word write access to the location addressed by VTB plus 0x100 activates the internal signal IExit.

Accessing these locations ([VTB] and [VTB]+0x100) without generating IAck or IExit is possible when the vector table logic is disabled (CRI.TE = 0).

ISNx	Interrupt Source Node Register x							
	7	6	5	4	3	2	1	0
r/w	M	P	E	x	PRIO			
	0	x	0	x	0	0	0	0 Res

M **Modify Pending Flag** (Table 11–2)
w1: Modify Pending flag.
r/w0: Don’t modify Pending flag.
This flag is modified by SW only and always reads as 0. It allows modification of register ISNx without influence to flag P. Without this flag a HW modification of flag P could be corrupted by a simultaneous read-modify-write of register ISNx.

P **Pending** (Table 11–2)
r/w1: Interrupt is pending.
r/w0: No interrupt pending.
This flag can be modified by HW and SW. It is set by HW when the corresponding interrupt source input is activated. If this interrupt source node is enabled, this flag is cleared by HW as soon as the corresponding ISR is called.

Table 11–2: Pending flag access

M	P	Read	Write
0	0	Not pending	Don’t modify P
0	1	Pending	
1	0	Not possible	Clear P
1	1		Set P

E **Enable**
r/w1: Enable interrupt.
r/w0: Disable interrupt.
This flag is modified by SW only.

PRIO **Interrupt Source Node Priority**
This field is modified by SW only (Table 11–3).

Table 11–3: Priority encoding

PRIO				Priority number
3	2	1	0	
0	0	0	0	0 (No priority)
0	0	0	1	1 (Lowest priority)
0	0	1	0	2
:	:	:	:	:
1	1	1	0	14
1	1	1	1	15 (Highest priority)

11.4. Principle of Operation

11.4.1. Reset

Clearing standby register flag SR1.IRQ resets the ICU (see Fig. 11–4 on page 92). The registers are reset to their mentioned values (see Section 11.3. on page 94) and cannot be modified. The nIRQ output is inactive and the actual priority level logic is cleared.

11.4.2. Initialization

Proper configuration of the interrupt sources in the peripheral modules has to be made prior to initialization of the ICU.

Initialization is possible after the standby register flag SR1.IRQ has been written to one. Now the registers can be modified by SW. But no interrupt request is generated to the CPU.

Install the vector table beginning at an even page address (9 LSB are zero). Each entry has to be a 32-bit start address of an interrupt service routine. The vector table has to be located near (±4 kB) the load PC instruction. Write the start address of the vector table to the vector table base register VTB. Further access to register VTB is not necessary until you want to switch to another vector table at another location.

Set up the interrupt source node registers ISNx with the necessary priority and enable them. The pending flags have to be cleared, because they are not cleared by SR1.IRQ and are operative all the time. Clearing an active pending flag and enabling the corresponding ISN must not be done with a single instruction. This might lead to an unwanted (spurious) interrupt which is directed to the default vector. First clear P and then set E in two instructions. Interrupt sources which shall not generate interrupts must not be enabled and need no priority (PRIO=0), but can be operated by polling and resetting the pending flag P by SW.

11.4.3. Operation

The ICU is operable in all CPU modes. However, interrupts have to be disabled during CPU mode switching to prevent undefined clock system behavior.

Setting both flags CRI.GE and CRI.TE enables the ICU at last. When an interrupt occurs, execution starts at address 0x18. For proper operation of the ICU the jump to the interrupt service routine has to be done by the PC relative load PC instruction
LDR PC,[PC,#<12_bit_offset>],
where the operand [PC,#<12_bit_offset>] must point to the first entry of the vector table. Due to the 12_bit_offset the vector table has to be located within ±4 kB from the above instruction. Above instruction is called vectoring. There are two possibilities for the point of time, direct and delayed, when vectoring takes place.

11.4.3.1. Direct Vectoring

Above instruction is the first instruction which is executed when an interrupt occurs. The address 0x18 contains the PC relative load PC instruction.

11.4.3.2. Delayed Vectoring

Above instruction is delayed. The address 0x18 contains a jump to a short piece of code which does all that has to be done for every ISR (save LR, SPSR and working registers). After this common prefix, the jump to the appropriate ISR is launched by the PC relative load PC instruction.

11.4.4. CPU Mode Switching

A CPU mode switch (see Section 4.2. on page 39) has to be handled like critical code. Disable the ICU prior to the mode switch by setting CPSR flag I of the ARM core (see Section 11.5.7.1. on page 97) or by clearing flag CRI.GE (see Section 11.5.7.2. on page 98). The ICU can be enabled again after the CPU mode switch sequence has finished. If the mode switch has to be done within an ISR, wait with the generation of the signal IExit until the ICU has been enabled again.

11.4.5. Inactivation

An interrupt source can be disabled locally by clearing the enable flag E in the corresponding ISN register. Even a pending interrupt can be disabled this way. A disabled ISN does not participate in sending interrupt requests to the CPU.

All interrupt sources can be disabled globally by clearing the global enable flag CRI.GE. It is impossible to inactivate an active nIRQ output signal by clearing CRI.GE. An active nIRQ will be served and only further IRQs can be suppressed by setting the GE flag.

The pending flag P stays operative in both cases and may be polled by SW.

A zero in the standby register flag SR1.IRQ immediately resets registers and logic and forces the nIRQ output to become inactive.

11.5. Application Hints

11.5.1. Hardware Triggered Interrupts

Normally, the connected peripheral modules set the pending flag P. If the ISN is enabled (E=1) and the priority is not zero, an IRQ is generated. The P flag will be reset as soon as the corresponding interrupt service routine is called. It is not required and should be avoided to modify the P flag of those ISNs by SW.

11.5.2. Software Triggered Interrupts

Any ISN which is not used by the connected peripheral module can be used for generating IRQ interrupts by SW. It must be avoided that the interrupt source of this ISN also generates interrupt requests. Either the corresponding peripheral module has to be switched off, or its interrupt source output has to be disabled.

The ISN has to be enabled (E=1) and programmed to the desired priority (PRIO>0). Setting the pending flag P by SW generates an interrupt. This interrupt will be processed as soon as possible. When the CPU responds to the interrupt request and jumps to the corresponding ISR, the pending flag is cleared automatically.

11.5.2.1. Delayed Interrupt

Any ISN which is not used by the connected peripheral module can be used for implementing the delayed interrupt mechanism for an operating system. The ISN has to be enabled (E=1) and programmed to priority 1 (the lowest priority which can generate an interrupt). Setting the pending flag P by OS-SW within a higher priority interrupt service routine generates a delayed interrupt, which is processed after all higher priority interrupts are finished.

11.5.3. Polling

Polling means that the pending flag P is observed by SW. Set by the corresponding interrupt source, the SW recognizes the P flag to be set, calls the corresponding routine and clears the P flag. The ISN should be disabled (E=0), otherwise unwanted IRQs would be generated.

11.5.4. Operating Nested Interrupts

Nested interrupt service routines use common data resources. Every routine, which may have interrupted a lower priority routine, has to save common data resources upon interrupt entry and restore them before returning to the interrupted routine. This is efficiently done by an entry and an exit sequence which are enclosing the interrupt service routine.

11.5.4.1. Interrupt Entry Sequence

The IRQ disable flag I in the core register CPSR is set after an IRQ, thus disabling further IRQs. Before the interrupt is enabled again, the user has to take the following steps:

1. For direct vectoring: Jump to the corresponding interrupt service routine by loading the first element from the vector table into the program counter by an LDR instruction.

2. Save Link Register (R14), SPSR and working registers to stack.

3. For delayed vectoring: Jump to the corresponding interrupt service routine by loading the first element from the vector table into the program counter an LDR instruction.

4. Clear CPSR.I to re-enable IRQs.

Now the actual application ISR can start.

11.5.4.2. Interrupt Exit Sequence

Before returning, it is necessary to clear the interrupt cause. Upon exit from an ISR some actions have to be taken without being interrupted:

1. Set CPSR.I to disable further IRQs.

2. Restore link register (R14), SPSR and working registers from stack.

3. Generate the signal IExit by performing a word write by an STR instruction to the interrupt exit address at [VTB]+0x100.

4. Returning to the interrupted routine has to be done by an instruction, which simultaneously writes the PC (R15) and CPSR with the values in R14 and SPSR (e.g., SUBS PC,R14_irq,#4).

11.5.5. Default Vector

Any read access to vector table address [VTB] will deliver the default vector, but will not generate IAck as long as the comparator output is inactive or the priority output of the priority encoder is zero. Due to this the default vector ISR runs with the priority of the interrupted routine. This is the only ISR which could be interrupted by itself. As long as this default vector ISR is not programmed re-entrant, interrupts should not be re-enabled by clearing the I flag of the CPSR. No IExit shall be generated on interrupt exit by writing to [VTB]+0x100 because there was no IAck at interrupt entry.

Unintentional inactivation of an active comparator output signal can be caused by modifying the ISN which is the only source for the momentary active nIRQ output. This can be done by disabling (E=0), or clearing the P flag, or lowering the priority of this ISN. These actions may lead to a default vector interrupt.

11.5.6. Debugger

Unintentional access to vector table addresses [VTB] and [VTB]+0x100 can result in malfunction of the interrupt system (HW and SW). If it is necessary, for instance, to dump the vector table, there are two ways to do this without generation of IAck or IExit:

The first way is to clear the flag CRI.TE which controls the vector table logic. Clearing it disables HW actions on accessing above addresses. But ensure that no interrupts are possible while TE is disabled.

The second way is to access above addresses by byte or half word operations only. The HW actions are only generated by word access. Disabling interrupts is not required in the latter case.

11.5.7. Critical Code

Critical code is a sequence of instructions which must not be interrupted, because it modifies common data resources. Protection from being interrupted can be achieved by disabling interrupts during critical code. There are several ways of doing this:

11.5.7.1. ARM Core's Interrupt Disable Flag I and F

The ARM core itself provides the interrupt disable bits I and F in the program status register CPSR.

The control bits of the CPSR (I, F and others) can be SW altered only when the processor is in a privileged mode. ARM recommends to modify the CPSR by a read-modify-write instruction sequence in order to leave the reserved bits unchanged.

```
MRS     r0,cpsr
ORR     r0,r0,#I_Bit    ;disable interrupts
MSR     cpsr_c,r0
```

The interesting case is when an interrupt comes in during execution of the MSR instruction. The core commits to taking an interrupt before the instruction being executed completes. Therefore even though an MSR instruction may have written to the CPSR to disable interrupts, the interrupt will still be taken. A NOP between the MSR instruction and the first instruction of the critical code is not necessary. If an interrupt

occurs during an MSR instruction, it will return to the instruction immediately following the MSR.

11.5.7.2. Global Enable Flag GE

Protection of critical code can be achieved by disabling the nIRQ output with the global enable flag CRI.GE. The GE flag changes its value in the cycle after the data transfer of the store instruction. In this cycle the next instruction is in the execution stage of the CPU and will be executed. Due to this, one NOP is required between the store instruction, which clears CRI.GE, and the first instruction of the critical code.

11.5.7.3. Force Priority

To protect critical code, further IRQ interrupts can be disabled by writing the maximum priority to the forced priority register AFP. Modifying AFP works like clearing the GE flag. One NOP is required between the store instruction, which writes AFP, and the first instruction of the critical code.

11.5.7.4. Disabling via ISN

At last, critical code protection can be achieved by disabling all ISNs by clearing their enable flag E. The priority encoder is calculated at the beginning of a cycle. Due to this, changing an ISN register becomes effective only in the next cycle. Two NOPs are required between the store instruction, which clears the flag E, and the first instruction of the critical code.

11.5.8. Switching an Interrupt Vector

Interrupt service routines of an interrupt source can easily be changed by entering the start address of the new ISR at the corresponding entry of the interrupt vector table.

11.5.9. Switching the Vector Table

Switching between different vector tables is possible if they have been installed. Changing the vector table is simply done by writing the base address of the new vector table to register VTB. All subsequent interrupt service routines must relate to this vector table. Due to this, it is necessary for an ISR in such an environment to read the location of the current vector table from register VTB before accessing it.

Be careful when switching vector tables within an ISR. Interrupted ISRs could try to do the IExit from an outdated table.

12. FIQ Interrupt Logic

The FIQ interrupt logic selects one out of eight interrupt sources as the CPU's nFIQ input. An interrupt request is latched in a pending flag until it is cleared by SW. The output can be disabled.

Features

- Expanding nFIQ input of ARM7TDMI
- 1 of 8 selection
- IRQ or FIQ selectable

12.1. Functional Description

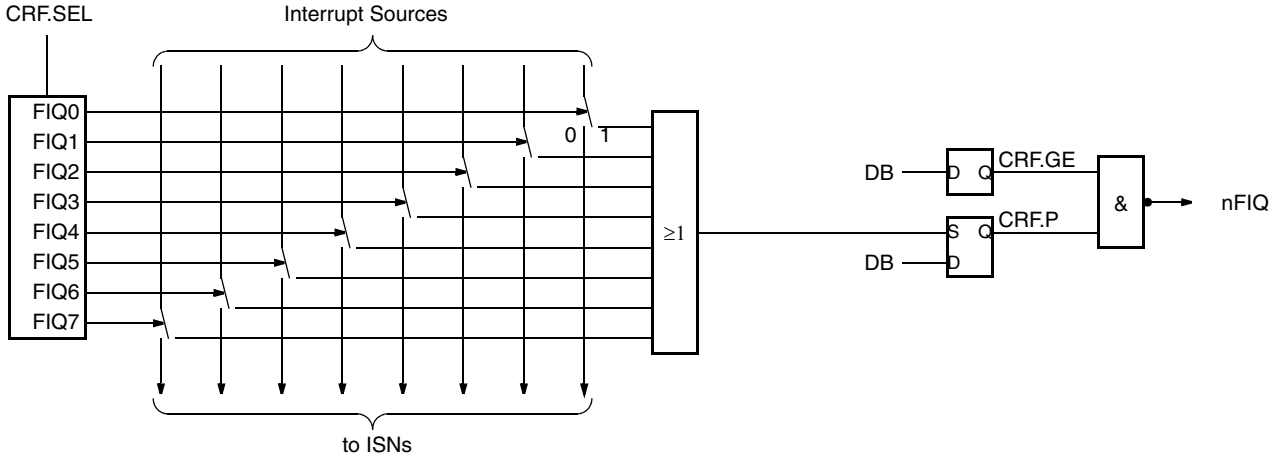


Fig. 12–1: Block diagram FIQ

At one time, only one interrupt source can be connected to the nFIQ input of the CPU. The interrupt source which is connected to the nFIQ, is disconnected from the corresponding ISN. This ISN can then be used by SW.

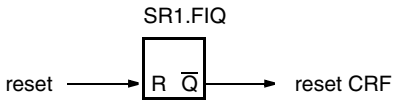


Fig. 12–2: Reset structure

Figure 12–2 shows the reset structure. Registers can't be written until the FIQ flag in the standby register SR1 is set.

12.2. Registers

PRF	Pending Register FIQ							
	7	6	5	4	3	2	1	0
r/w	x	x	x	x	x	x	x	P
	x	x	x	x	x	x	x	0 Res

P **FIQ Pending**
r/w1: Pending FIQ.
r/w0: No pending FIQ.

This flag is set by HW and SW. It must be cleared by SW before re-enabling FIQ by bit F in the core's CPSR register, or no further interrupt can occur.

CRF	Control Register FIQ							
	7	6	5	4	3	2	1	0
r/w	GE	x	x	x	SEL			
	0	x	x	x	0	0	0	0 Res

GE **Global Enable FIQ**
r/w1: Enable FIQ.
r/w0: Disable FIQ.
Disabling happens as soon as nFIQ is inactive. An active nFIQ will not be interrupted by clearing GE.

SEL **Select FIQ Source**
r/w: (Table 12–1)

Table 12–1: FIQ source selection

SEL				Switched to nFIQ		
3	2	1	0	Select	ISN	Interrupt Source
0	x	x	x	None		
1	0	0	0	FIQ0	6	SPI0
1	0	0	1	FIQ1	11	WAIT COMP
1	0	1	0	FIQ2	12	UART0
1	0	1	1	FIQ3	13	PINT2
1	1	0	0	FIQ4	15	CC2OR
1	1	0	1	FIQ5	17	Timer 2
1	1	1	0	FIQ6	19	I2C0
1	1	1	1	FIQ7	5	CAN0

12.3. Principle of Operation

12.3.1. Reset

Clearing standby register flag SR1.FIQ resets the FIQ interrupt logic (see Fig. 12–2 on page 99). The registers are reset to their mentioned values (see Section 12.2. on page 99) and cannot be modified. The nFIQ output is inactive.

12.3.2. Initialization

Proper configuration of the interrupt sources in the peripheral modules has to be made prior to initialization of the FIQ interrupt logic.

Initialization is possible after the standby register flag SR1.FIQ has been set. Now the registers can be modified by SW. But no interrupt request is generated to the CPU.

The FIQ interrupt logic is operable in all CPU speed modes.

12.3.3. Operation

Setting flag CRF.GE enables the FIQ interrupt logic. When an interrupt occurs, execution starts from address 0x1C.

12.3.4. Inactivation

The FIQ interrupt logic can be disabled by clearing the global enable flag CRF.GE. An active nFIQ will be served, however, and only future FIQs will be suppressed by clearing the GE flag.

Clearing the standby register flag SR1.FIQ immediately resets registers and logic and forces the nFIQ output to become inactive.

13. Port Interrupts

Port interrupts are the interface of the interrupt controller to the external world. Six U-port pins and, alternatively, six P-port pins are connected to the module via their special input lines (Fig. 13–1). HW-Option-programmable multiplexers define which port signal is actually connected to the trigger mode logic (Table 13–1). The P-ports are actually analog input ports, thus Schmitt triggers are enabled if the P-ports are selected as port interrupts. The input sampling frequency is f_{0perm}, which is not disabled by CPU SLOW or DEEP SLOW modes.

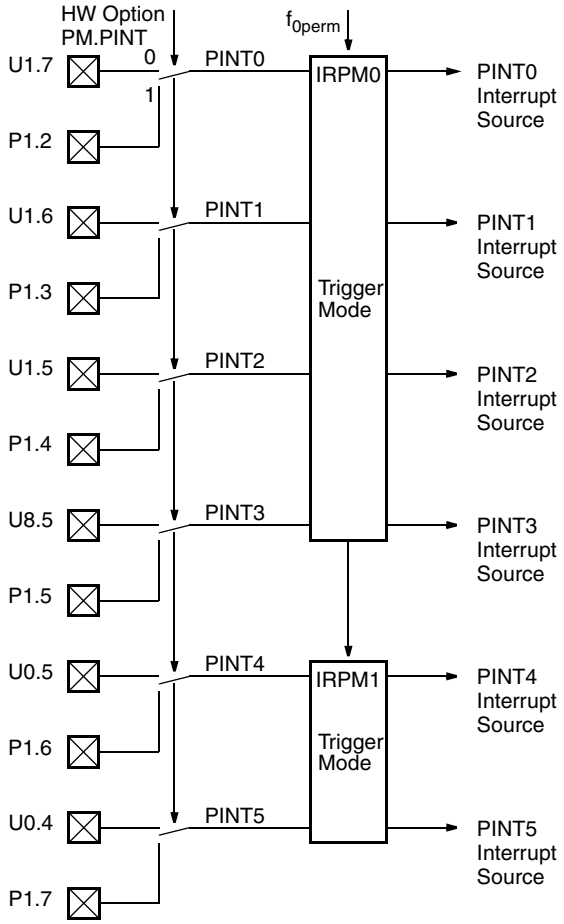


Fig. 13–1: Port interrupts

The user can define the trigger mode for each port interrupt by the interrupt port mode register.

The trigger mode defines on which edge of the interrupt source signal the Interrupt Controller is triggered. The triggering of the interrupt controller is shown in figure 13–2.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

Precautions

Parallel usage of a P-port as analog and port interrupt input is possible but not recommended. In this case the Schmitt trigger input circuit is enabled. This is the reason why input levels other than AV_{DD} and AV_{SS} may cause quiescent currents in the Schmitt trigger circuit and thus lead to higher power consumption.

Table 13–1: Module-specific settings

Module Name	HW Options		Initialization	
	Item	Address	Item	Setting
PINT0	Port Multiplexers	PM.PINT	PINT0	U1.7 special in
				P1.2
PINT1			PINT1	U1.6 special in
				P1.3
PINT2			PINT2	U1.5 special in
				P1.4
PINT3			PINT3	U8.5 special in
				P1.5
PINT4			PINT4	U0.5 special in
				P1.6
PINT5			PINT5	U0.4 special in
				P1.7

IRPM0 Interrupt Port Mode Register 0

	7	6	5	4	3	2	1	0	
r/w	PIT3		PIT2		PIT1		PIT0		
	0	0	0	0	0	0	0	0	Res

IRPM1 Interrupt Port Mode Register 1

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	PIT5		PIT4		
	x	x	x	x	0	0	0	0	Res

PITn Port interrupt trigger number n
This field defines the trigger behavior of the associated port interrupt (Table 13–2).

Table 13–2: PITn usage

PITn	Trigger Mode
0h	Interrupt source is disabled
1h	Rising edge
2h	Falling edge
3h	Rising and falling edges

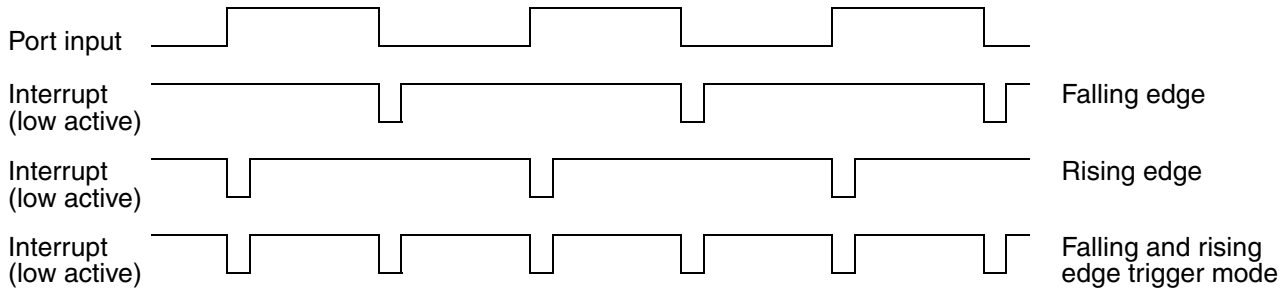


Fig. 13–2: Interrupt timing

14. Ports

This chapter describes the P-, U- and H-Ports.

The analog input ports, P0 and P1, serve as input for the analog-to-digital converter and may be used as digital inputs.

P2 may be used as digital input, only.

The universal ports U0 to U8 serve as digital I/O and can be configured as LCD drivers.

The high current ports H0 to H7 serve as digital I/O and can be configured as stepper motor drivers.

14.1. Analog Input Port

The 16-pin analog input port is composed of ports P0 and P1. All port pins can be configured as digital input. P0.6 is connected to a comparator, which may be selected as interrupt source. P1.2 to P1.7 can be used as port interrupts. The 2-pin port P2 solely serves as digital input.

Features

- 16-pin analog input multiplexer.
- 18 pins configurable as digital input ports.
- Schmitt hysteresis digital input buffer, CMOS level (2.5 V) or automotive level (3.3 V) selectable.
- 6 pins configurable as port interrupts.
- 2 pins usable as wake-up input.

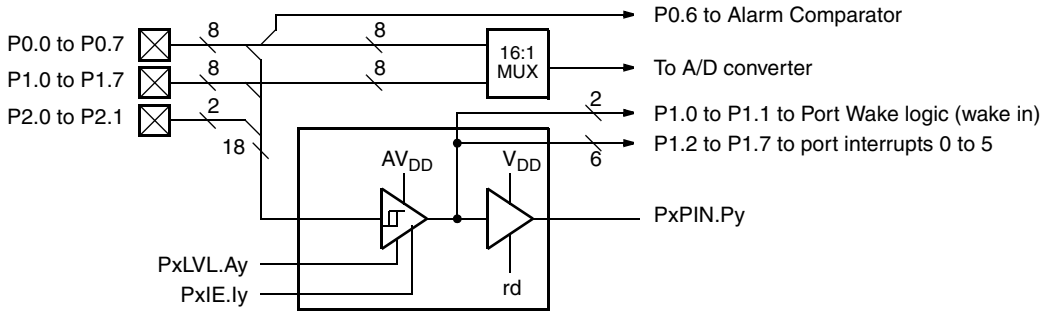


Fig. 14–1: P-Ports with input multiplexer and P0.6 alarm comparator

P0 and P1 analog input lines are connected to a multiplexer. The output of this multiplexer is connected to the 10-bit A/D converter.

Port P0.6 is, in addition, the input of the P0.6 alarm comparator, described in the chapter on the analog section.

P0 and P1 pins may alternatively, P2 may exclusively be used as digital input if enabled by setting the individual pin's enable flag PxIE.Iy. CMOS or automotive Schmitt trigger input level may individually be selected by writing registers PxLVL. The digital value of the input pins is obtained by reading registers PxPIN. Disabled inputs read as 1. Pins should either be used as analog or digital inputs, not both at the same time.

Six of the analog input pins (P1.2 to P1.7) may be used as port interrupt input if selected by HW Option PM.PINT (see sections "Port Interrupts" and "HW Options" for more details). Configure as digital input for this operation.

Ports P1.0 and P1.1 may be used as wake-up inputs (see section "Power Saving Module" for details), hence their digital input enable flags (P1IE.I0 and I1) reset to 1 (input buffer enabled).

PxPIN Port x Pin Register

	7	6	5	4	3	2	1	0	
r	P7	P6	P5	P4	P3	P2	P1	P0	
	1	1	1	1	1	1	1	1	Res

P0 to 7 **Pin Data 0 to 7**
r: Read Pin state

PxLVL Port x Input Level Register

	7	6	5	4	3	2	1	0	
r/w	A7	A6	A5	A4	A3	A2	A1	A0	
	0	0	0	0	0	0	0	0	Res

A0 to 7 **Automotive Flag 0 to 7**
r/w1: Schmitt trigger input level is Automotive
r/w0: Schmitt trigger input level is CMOS

P0IE	Port 0 Input Enable Register							
	7	6	5	4	3	2	1	0
r/w	I7	I6	I5	I4	I3	I2	I1	I0
	0	0	0	0	0	0	0	0 Res

I0 to 7 **Digital Input Enable 0 to 7**
r/w1: Enable input buffer
r/w0: Disable input buffer

P1IE	Port 1 Input Enable Register							
	7	6	5	4	3	2	1	0
r/w	I7	I6	I5	I4	I3	I2	I1	I0
	0	0	0	0	0	0	1	1 Res

I0 to 7 **Digital Input Enable 0 to 7**
r/w1: Enable input buffer
r/w0: Disable input buffer

P2IE	Port 2 Input Enable Register							
	7	6	5	4	3	2	1	0
r/w	x	x	x	x	x	x	I1	I0
							0	0 Res

I0 to 1 **Digital Input Enable 0 to 1**
r/w1: Enable input buffer
r/w0: Disable input buffer

14.2. Universal Ports U0 to U8

Universal ports are pin-configurable as SW I/O port, special input/output port (SI/SO) to special internal hardware modules or direct drive of 4:1 multiplexed LCD segment and backplane lines (LCD port).

The output drivers feature a current fold-back characteristic to allow shorting the output and to improve EMI performance.

Features

- Pin-configurable as I/O or Special Port or LCD driver.
- LCD mode: 1-to-4 multiplex, 5 V supply.
- PORT mode: push-pull or open-drain output.
- Two output current fold-back characteristics selectable
- Schmitt hysteresis input buffer, CMOS level (2.5 V) or automotive level (3.3 V) selectable.

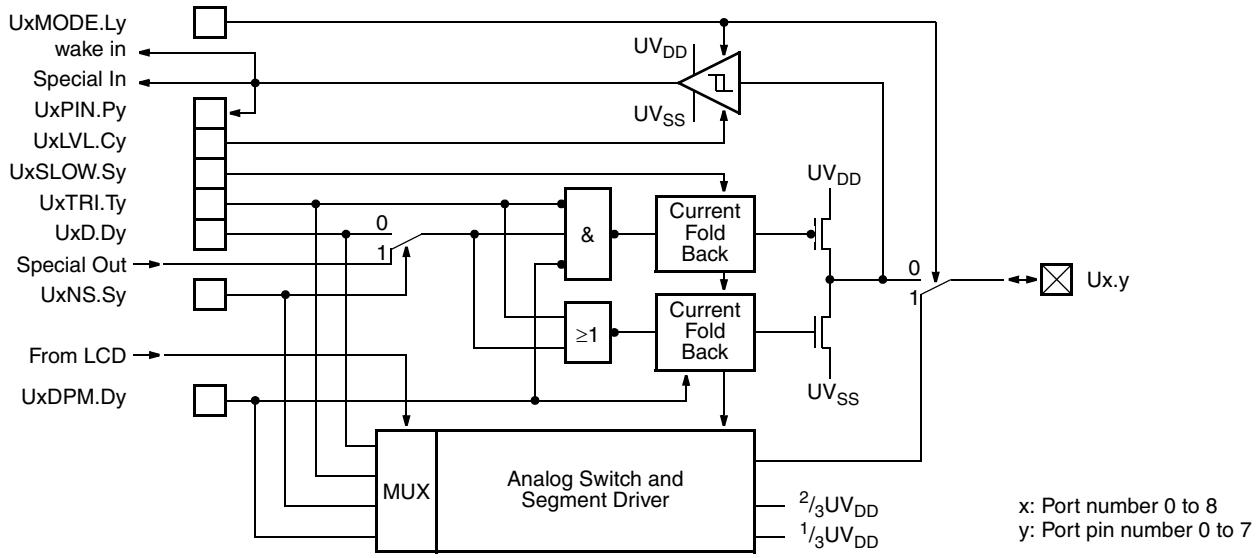


Fig. 14–2: Universal port pin circuit diagram

The universal port pins can be configured for several basic operating modes (Table 14–1)

Table 14–1: Universal port basic operating modes

Modes		Function
Port Mode	Normal Input	The SW uses the port as digital input.
	Special Input	The port input is additionally connected to specific hardware modules.
	Normal Output	The SW uses the port as latched digital tristateable output.
	Special Output	The output signals of specific hardware modules are directly port output source.
LCD Mode		The port pin serves as backplane/segment driver for a 4:1 multiplexed LC display

See the chapter on pinning for information about specific hardware module connections to individual port pins for special input and special output purposes.

Universal port control is distributed among eight registers. Four of these registers have duplicate functions for port and LCD mode. All register bits corresponding to one U-Port pin are controlled by the same bus bit.

In both LCD and Port modes, the SLOW mode may be defined for each individual U-Port pin. It reduces the current drive capability of the output stage. Set flag SR0.PSLW to enable this operation mode.

After reset, all universal ports are in port, normal, tristate, CMOS input level condition. SLOW mode is disabled.

14.2.1. Port Mode

For port mode, the respective UxMODE register bit has to be cleared for mode selection.

The function of the seven remaining registers is given in Table 14–2.

Table 14–2: Register functions in port mode

Register	Function
UxD	r/w data register
UxTRI	enable/disable output
UxNS	select data register or specific hardware module as output source
UxDPM	select push-pull or open-drain, double drive mode for output drivers
UxSLOW	enable/disable port slow mode for output drivers
UxLVL	select CMOS or automotive Schmitt trigger input level
UxPIN	read pin state

In port mode, the special input path is always operative. This allows manipulating the input signal to the specific hardware module through normal output operations by software.

Because register UxPIN allows reading the pin level also in special output mode, the output state of the specific hardware module may be read by the CPU.

14.2.2. LCD Mode

For LCD Mode, the respective UxMODE register bit has to be set for mode selection.

The function of the seven remaining registers is given in Table 14–3.

Table 14–3: Register functions in LCD mode

Register	Function
UxD	r/w phase 0 segment line data
UxTRI	r/w phase 1 segment line data
UxNS	r/w phase 2 segment line data
UxDPM	r/w phase 3 segment line data
UxSLOW	enable/disable port slow mode for output drivers
UxLVL	no function
UxPIN	no function

By writing segment line data registers, only a master is changed. Any write to global register ULCDLD will transfer all master settings to the respective slaves and thereby change the LC display in one instant.

Registers UxD, UxTRI, UxNS and UxDPM compose a word-aligned 32-bit register and may be accessed by one 32-bit operation.

The output sequence timing on backplane and segment output ports in LCD mode is controlled by the LCD module. Please refer to section LCD module for information about operation of this module.

As generation of the backplane port output sequence is fully done by the LCD module, no segment line data setting is necessary for these ports.

14.2.3. Port Fast and Slow Modes

Once individual port pins have been enabled for port slow mode by setting registers UxSLOW, set flag SR0.PSLW to simultaneously enter this mode in all respective ports.

All U-ports exhibit two operating regions in the DC output characteristic (see Fig. 14–3). Near zero output voltage, the internal driver transistors operate non-limited, to offer a linear, low on-resistance. With larger output voltages, however, the output current folds back to a a limited value. This measure helps to fight supply current transients and related EMI noise during port switching.

In the fold-back region, port fast mode and port slow mode select two different current limits I_{shf} and I_{shs}. Port slow mode sets a limit where the output may even be shorted continuously to either supply rail. Thus, wired-or configurations may be realized. The external load resistance should be greater than 5 kOhms in port slow mode.

For actually switching to port slow mode, both registers UxSLOW and SR0.PSLW have to be set. In all other cases, port fast mode is selected.

It is recommended to place all LCD ports in Port Slow mode.

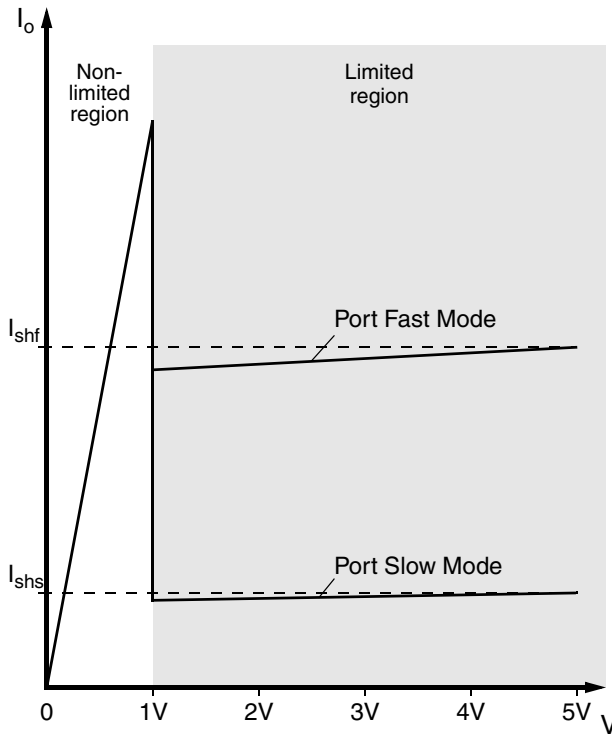


Fig. 14–3: Typical U-Port pull-down DC output characteristic (pull-up characteristic is complementary).

14.3. Universal Port Registers

Eight U-Port registers are basically available for 9 U-Ports U0 to U8, each. Because some U-Ports are less than 8 pins wide, not all of the described bits are available for every port. Furthermore, the respective device's pinning may require a reduction in available U-Ports. See the respective pinning table for details.

The general U-port register model is given below.

UxMODE Universal Port Mode Register

	7	6	5	4	3	2	1	0	
r/w	L7	L6	L5	L4	L3	L2	L1	L0	
	0	0	0	0	0	0	0	0	Res

L0 to 7 Port Mode Flag
Select the mode of the corresponding port pins.
r/w1: Port pin is in LCD mode.
r/w0: Port pin is in Port mode.

UxD Universal Port x Data / Segment 0 Register

	7	6	5	4	3	2	1	0	
r/w	D7	D6	D5	D4	D3	D2	D1	D0	Port
r/w	SG7_0	SG6_0	SG5_0	SG4_0	SG3_0	SG2_0	SG1_0	SG0_0	LCD
	0	0	0	0	0	0	0	0	Res

D0 to 7 Data Latch
w: Write latch.
r: Read latch.

SG0_0 to 7_3 Segment Data Latch
w: Write latch.
r: Read latch.
In LCD mode, U-Port registers UxD, UxTRI, UxNS and UxDPM store LCD segment information. Segment register bits UxY.SGm_n contain the information for segment line m during phase n, which controls segment m_n. Thus, register bits UxD.SG0_0, UxTRI.SG0_1, UxNS.SG0_2 and UxPIN.SG0_3 contain the complete information for segment line 0 in U-Port x.

Please refer to pin assignment and description for segment/ pin number assignment. Information about the usage of the LCD segment field will be found at the functional description of the LCD module.

UxTRI Universal Port x Tristate / Segment 1 Register

	7	6	5	4	3	2	1	0	
r/w	T7	T6	T5	T4	T3	T2	T1	T0	Port
r/w	SG7_1	SG6_1	SG5_1	SG4_1	SG3_1	SG2_1	SG1_1	SG0_1	LCD
	1	1	1	1	1	1	1	1	Res

T0 to 7 Output Tristate Flag 0 to 7
r/w1: Output driver is disabled (tristate)
r/w0: Output driver is enabled

UxNS Universal Port x Normal-Special / Segment 2 Register

	7	6	5	4	3	2	1	0	
r/w	S7	S6	S5	S4	S3	S2	S1	S0	Port
r/w	SG7_2	SG6_2	SG5_2	SG4_2	SG3_2	SG2_2	SG1_2	SG0_2	LCD
	0	0	0	0	0	0	0	0	Res

S0 to 7 Normal/Special Mode Flag 0 to 7
r/w1: Special mode. Special hardware drives pin.
r/w0: Normal mode. Data latch drives pin.

UxDPM Universal Port x Double Pull-Down Mode / Segment 3 Register

	7	6	5	4	3	2	1	0	
r/w	D7	D6	D5	D4	D3	D2	D1	D0	Port
r/w	SG7_3	SG6_3	SG5_3	SG4_3	SG3_3	SG2_3	SG1_3	SG0_3	LCD
	0	0	0	0	0	0	0	0	Res

D0 to 7 Double Pull-Down Mode
r/w1: Output driver is pull-down, I_{shs} (Port Slow mode) doubled.
r/w0: Standard.

All U-port pins may be switched into a double pull-down mode (DPM) by setting the appropriate DPMx flag, where

– the short circuit current I_{shs} is doubled (with Port Slow Mode enabled for these ports, and SR0.PSLW set to 1)

– the output configuration is pull-down, not the standard push-pull.

By this means, these ports may be configured to operate as connection to a wired-or, single-wire bus (e.g., DIGITbus or I^2C) with external pull-up resistor.

UxSLOW Universal Port x Slow Mode Register

	7	6	5	4	3	2	1	0	
r/w	S7	S6	S5	S4	S3	S2	S1	S0	
	0	0	0	0	0	0	0	0	Res

S0 to 7 Slow Flag 0 to 7
r/w1: Output driver is in Port Slow mode
r/w0: Output driver is in Port Fast mode

UxLVL Universal Port x Input Level Register

	7	6	5	4	3	2	1	0	
r/w	A7	A6	A5	A4	A3	A2	A1	A0	
	0	0	0	0	0	0	0	0	Res

A0 to 7 Automotive Flag 0 to 7
r/w1: Schmitt trigger input level is Automotive
r/w0: Schmitt trigger input level is CMOS

UxPIN	Universal Port x Pin Register							
	7	6	5	4	3	2	1	0
r	P7	P6	P5	P4	P3	P2	P1	P0
	x	x	x	x	x	x	x	x

Res

P0 to 7 **Pin Data 0 to 7**
r: Read Pin state.

ULCDLD	Universal Port LCD Load Register							
	7	6	5	4	3	2	1	0
w	LCDSLV	x	x	x	x	x	x	x
	0	0	0	0	0	0	0	0

Res

LCDSLV **LCD Module is Slave**
Select the mode of the LCD module.
w1: LCD module is slave.
w0: LCD module is master.

A write access to this memory location simultaneously loads all segment information of all U-Ports in LCD mode to the display. The flag LCDSLV is available only in LCD mode.

14.3.1. Special Register Layout of U-Port 4

U4.0 to U4.3 provide backplane signals in LCD Mode. To operate any ports as LCD segment driver it is necessary to switch all these ports to LCD mode. This has to be done by setting flags U4MODE.L0 through U4MODE.L3.

As backplane ports U4.0 to U4.3 require no segment data setting, SG0_0 through SG3_3 bits are not available in U4 registers.

14.4. High Current Ports H0 to H7

High current ports 0 to 7 are used to drive coils of stepper motors. All ports are 4 pins wide to facilitate control of individual stepper motors. The H-ports are similar to universal ports but as the name says, they can drive higher currents. H-ports can be operated via software just like universal ports (port mode). Their special out inputs are connected to the stepper motor module or to PWM outputs.

Features

- Pin-configurable as I/O or Special Port driver
- 30 mA output current
- Schmitt hysteresis input buffer, CMOS level (2.5 V) or automotive level (3.3 V) selectable.
- Reduced slew rate of current and voltage for driving resistive, capacitive or inductive loads.

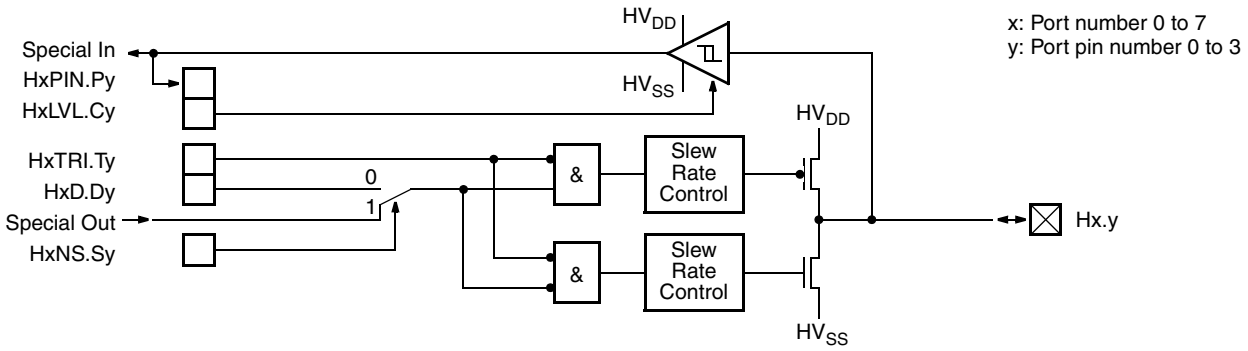


Fig. 14–4: High current port pin circuit diagram

The H-port pins can be configured for several basic operating modes (Table 14–4)

Table 14–4: High current port basic operating modes

Mode	Function
Normal Input	The SW uses the port as digital input.
Special Input	The port input is additionally connected to specific hardware modules.
Normal Output	The SW uses the port as latched digital tristateable output.
Special Output	The output signals of specific hardware modules are directly port output source.

See the chapter on pinning for information on specific hardware module connections to individual port pins for special input and special output purposes.

H-port control is distributed among five registers. All register bits corresponding to one H-port pin are controlled by the same bus bit.

After reset, all H-ports are in normal, output, low, CMOS input level condition.

The function of the five registers is given in Table 14–5.

Table 14–5: Register functions

Register	Function
HxD	r/w Data register
HxTRI	enable/disable output
HxNS	select Data register or specific hardware module as output source
HxLVL	select CMOS or Automotive Schmitt trigger input level
HxPIN	read pin state

The special input path is always operative. This allows manipulating the input signal to the specific hardware module through normal output operations by software.

Because register HxPIN allows reading the pin level also in special output mode, the output state of the specific hardware module may be read by the CPU.

Two high current ports together with a coil, build an H-bridge. Two H-bridges are necessary to operate a stepper motor.

The n-channel and the p-channel transistor of the output driver are controlled separately, to eliminate crossover currents.

The reset output levels of the ports are low to avoid floating coils.

14.5. High Current Port Registers

Five H-port registers are basically available for 8 H-ports H0 to H7, each. But the respective device's pinning may require a reduction in available H-ports. See the respective pinning table for details.

The general H-port register model is given below.

HxD High Current Port x Data Register

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	D3	D2	D1	D0	
	x	x	x	x	0	0	0	0	Res

D0 to 3 Data Latch
w: Write latch.
r: Read latch.

HxTRI High Current Port x Tristate Register

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	T3	T2	T1	T0	
	x	x	x	x	0	0	0	0	Res

T0 to 3 Output Tristate Flag 0 to 3
r/w1: Output driver is disabled (tristate)
r/w0: Output driver is enabled

HxNS High Current Port x Normal/Special Register

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	S3	S2	S1	S0	
	x	x	x	x	0	0	0	0	Res

S0 to 3 Normal/Special Mode Flag 0 to 3
r/w1: Special Mode. Special hardware drives pin.
r/w0: Normal Mode. Data latch drives pin.

HxLVL High Current Port x Input Level Register

	7	6	5	4	3	2	1	0	
r/w	x	x	x	x	A3	A2	A1	A0	
	x	x	x	x	0	0	0	0	Res

A0 to 3 Automotive Flag 0 to 3
r/w1: Schmitt trigger input level is Automotive
r/w0: Schmitt trigger input level is CMOS

HxPIN High Current Port x Pin Register

	7	6	5	4	3	2	1	0	
r	x	x	x	x	P3	P2	P1	P0	
	x	x	x	x	0	0	0	0	Res

P0 to 3 Pin Data 0 to 3
r: Read Pin state.

15. AVDD Analog Section

The analog section operates from the AVDD supply pin and comprises the PLL/ERM module, the ADC, the P06 and the WAIT comparators. In addition, it contains support circuits like the VREFINT generator, the BVDD regulator and the necessary biasing circuits. Fig. 15–1 gives an overview.

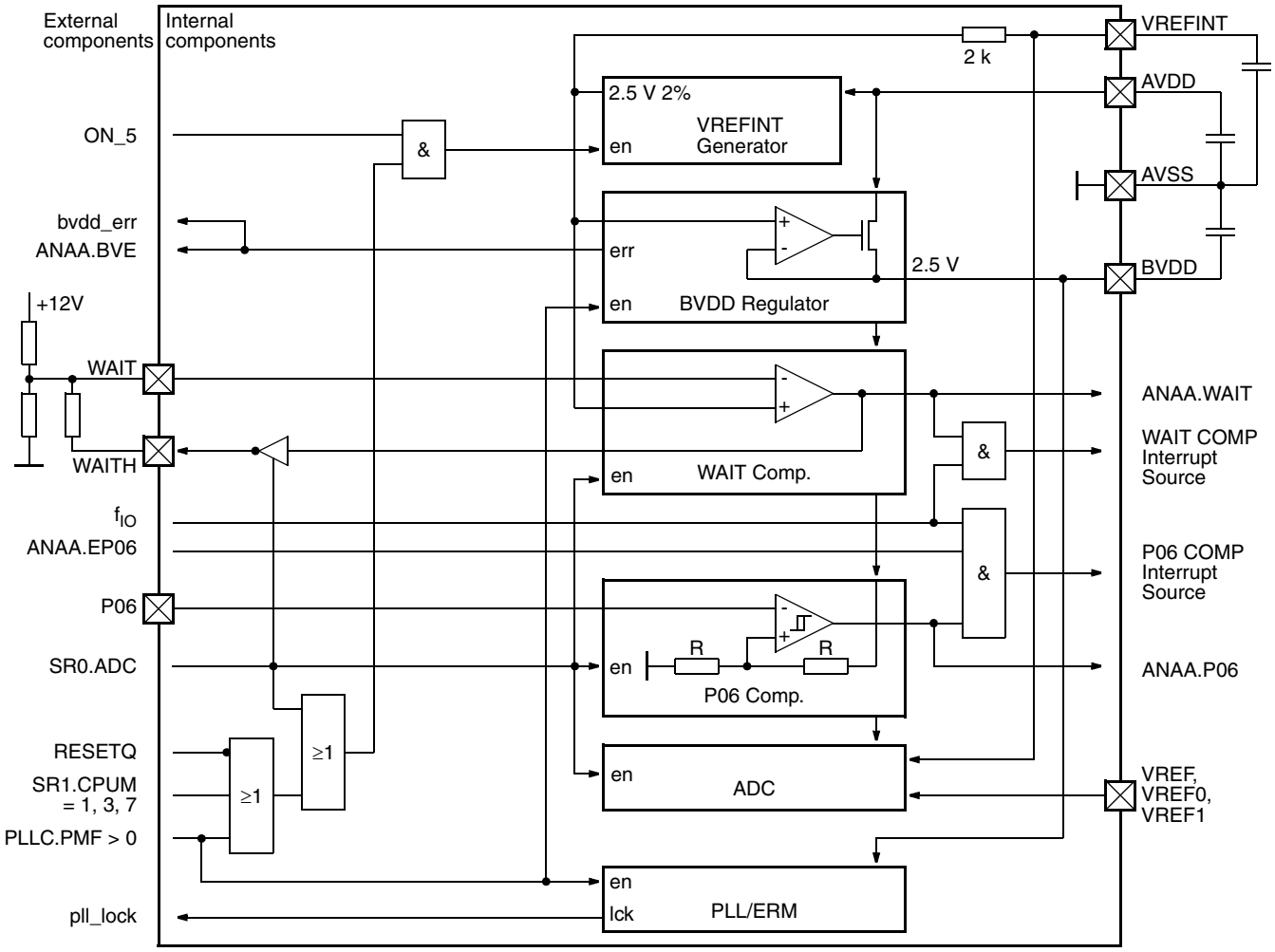


Fig. 15–1: AVDD section

Table 15–1: Activation of AVDD analog section modules

Operation Mode	VREFINT Gen.	PLL/ERM and BVDD Regulator	ADC, P06, WAIT
RESET	on	off	off
FAST, PLL and PLL2	on	on if PLLC.PMF > 0	on if SR0.ADC=1
SLOW and DEEP SLOW	on if PLLC.PMF > 0 or SR0.ADC=1		
WAKE, STANDBY, IDLE	off	off	off

15.1. VREFINT Generator

The VREFINT generator generates bias signals which are necessary for the operation of all analog-section modules. Furthermore, it produces a tightly controlled reference voltage VREFINT, that is delivered to the BVDD regulator and the WAIT comparator. Via a decoupling resistor it is also routed to the VREFINT pin.

The VREFINT pin voltage, which has to be buffered externally by a 10 nF ceramic capacitor, is input to the ADC as alternative, internally generated, reference voltage.

This module is permanently enabled during reset, in the CPU modes FAST, PLL and PLL2, and whenever SR0.ADC or PLLC.PMF is not 0. A certain set-up time has to elapse after enabling the module for VREFINT to stabilize.

No resistive load must be connected to the VREFINT pin.

15.2. BVDD Regulator

The BVDD Regulator generates the 2.5 V BVDD supply voltage for the internal PLL/ERM module from the 5 V AVDD. It derives its reference from the VREFINT generator.

BVDD must be buffered externally by a 150 nF ceramic capacitor.

This module is permanently enabled whenever PLLC.PMF is not 0. A certain set-up time has to elapse after enable for BVDD to stabilize.

An overload condition in the regulator (current or voltage drop-out) is stored in flag ANAA.BVE. The immediate overload signal may be routed to the LCK special output by selection in field ANAU.LS (UVDD analog section).

15.3. Wait Comparator

The level on pin WAIT is compared to the internal reference VREFINT. The state of the comparator output is available as flag ANAA.WAIT and as WAIT comparator interrupt source.

Furthermore, the output is available on pin WAITH, so that the hysteresis of this comparator can be set with an external positive-feedback resistor (100 kOhms min.).

After reset, the module is off (zero standby current). The module is enabled by setting flag SR0.ADC, together with the P0.6 Comparator and the ADC. If the VREFINT generator is powered up as well (cf. Table 15–1), the user has to assure that the necessary VREFINT set-up time has elapsed, before using comparator results (flag and interrupt).

The interrupt source output is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

The WAIT comparator interrupt source toggles with f_{IO}, to generate interrupts as long as the level on pin WAIT is lower than the internal reference.

15.4. P0.6 Comparator

The level on port P0.6 is compared to AVDD/2. The comparator features a small built-in hysteresis. The state of the comparator output is available as flag ANAA.P06 and as P0.6 Comparator interrupt source.

After reset, the module is off (zero standby current). The module is enabled by setting flag SR0.ADC, together with the WAIT comparator and the ADC. If the VREFINT generator is powered up as well (cf. Table 15–1), the user has to assure that the necessary VREFINT set-up time has elapsed, before using comparator results (flag and interrupt).

The interrupt source output, which must be enabled by setting flag ANAA.EP06, is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

The P0.6 comparator interrupt source toggles with f_{IO}, to generate interrupts as long as the level on pin P0.6 is lower than the internal reference.

15.5. PLL/ERM

The PLL and ERM modules are operated on the internally generated 2.5 V BVDD supply voltage.

For details on operating this module please refer to section “CPU and Clock System”.

15.6. A/D Converter (ADC)

The analog-to-digital converter allows the conversion of an analog voltage ranging from AVSS to either one of three external references VREF, VREF0, VREF1 (2.5 to 5 V) or the internal reference VREFINT (≈2.5 V), to a 10-bit digital value. A multiplexer connects one of 16 analog input ports to the ADC. A sample and hold circuit holds the analog voltage during conversion. The duration of the sampling time is programmable.

Features

- 10-bit resolution.
- Successive approximation, charge balance type.
- 16-channel input multiplexer.
- Input buffering for high ohmic sources selectable.
- Sample and hold circuit.
- 4/8/16/32 µs conversion selectable for optimum throughput/accuracy balance.
- 2.5 V internal reference (VREFINT) or 2.5 to 5 V external references (VREF, VREF0, VREF1) selectable
- Zero standby current

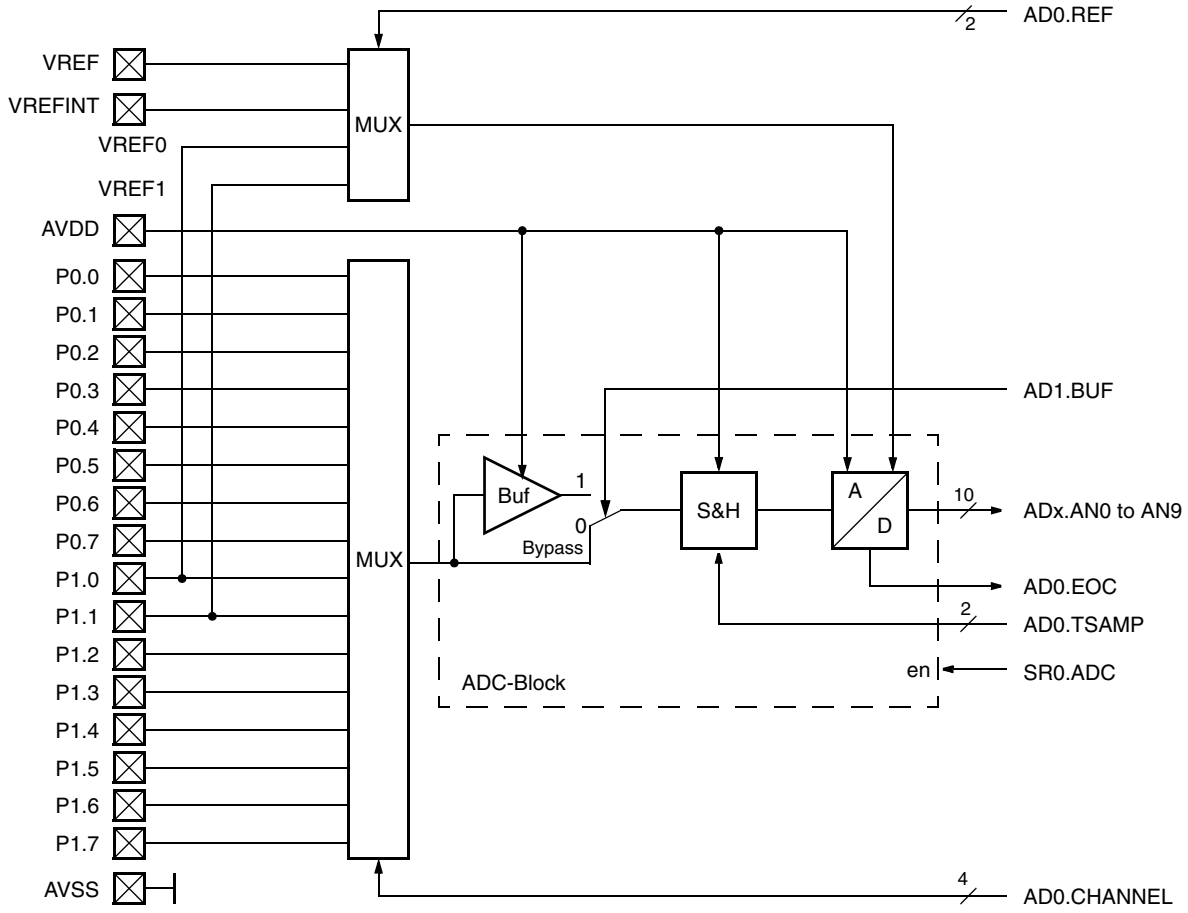


Fig. 15–2: ADC block diagram

15.6.1. Principle of Operation

After reset, the module is off (zero standby current). The module is enabled by setting flag SR0.ADC. The user has to assure that the necessary VREFINT-set-up time has elapsed.

Before starting a conversion, select input-buffer usage or bypass with flag AD1.BUF. Note that the input buffer requires

a 1 µs setup time before usage. When the buffer is never used, leave flag AD1.BUF cleared. When the buffer is always used, leave this flag set. When toggling buffer usage, set this flag at least 1 µs before starting a conversion.

Before starting a conversion, check flag AD0.EOC to be set.

A conversion is started by a write access to register AD0, selecting sample time (AD0.TSAM), reference source (AD0.REF) and input channel (AD0.CHANNEL).

Sampling starts one f_0 clock cycle after completion of the write access to AD0. Flag AD0.EOC signals the end of conversion. The 10-bit result is stored in the registers AD1 (8 MSB) and AD0.

The conversion time depends on f_0 and the programmed sample time (Table 15–2).

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

15.6.1.1. Conversion Law

The result of A/D conversion is described by the following formula:

$$DV = INT\left(\frac{U_{In}}{1LSB}\right) \quad \text{where} \quad 1LSB = \frac{U_{Ref}}{1024}$$

DV = Digital Value; INT = Integer part of the result

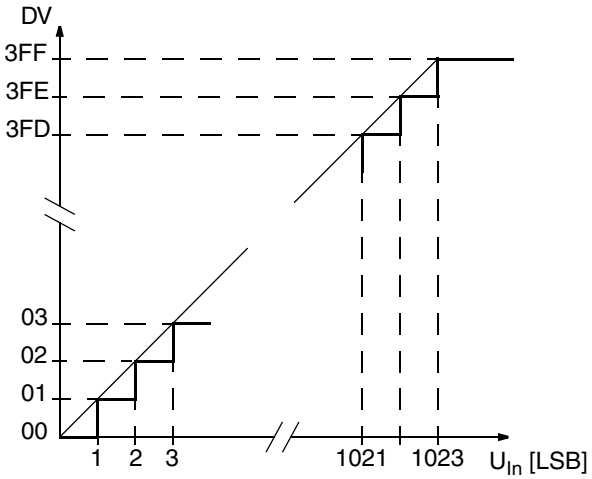


Fig. 15–3: Characteristic curve

The voltage on the reference-input pins VREF, VREF0 and VREF1 may be set to any level in the range from AV_{SS} to AV_{DD}. However, accuracy is only specified in the range from 2.56 V (1 LSB = 0.25 mV) to 5.12 V (1 LSB = 0.5 mV).

15.6.1.2. Measurement Errors

The result of the conversion mirrors the voltage potential of the sampling capacitance (typically 8 pF) at the end of the sampling time. This capacitance has to be charged by the source through the source impedance within the sampling-time period. To avoid measurement errors, system design has to make sure that at the end of this sampling period, the voltage on the sampling capacitance is within ±0.1 LSB from the source voltage.

Measurement errors may occur, when the voltage of high-impedance sources has to be measured:

– To reduce these errors, the sampling time may be increased by programming the field AD0.TSAMP.

– In cases where high-impedance sources are only rarely sampled, a 100 nF capacitor from the input to AVSS is a sufficient measure to ensure that the voltage on the sampling capacitance reaches the full source voltage, even with the shortest sampling time.

– In some high-impedance applications a charge-pumping effect may noticeably influence the measurement result: Charge pumping from a high-potential to a low-potential source will occur when such two sources are measured alternatingly. This results in a current that appears as flowing from the high-potential source through the IC into the low-potential source. This current explains from the fact that during the respective sampling period the high-potential source always charges the sampling capacitance, while the low-potential source always discharges it. Usage of the input buffer (AD1.BUF) substantially reduces this effect.

15.7. Registers

AD0 ADC Register 0

	7	6	5	4	3	2	1	0	
r	EOC	x	x	x	x	TEST	AN1	AN0	
w	TSAMP		REF		CHANNEL				
	0	0	0	0	0	0	0	0	Res

AD1 ADC Register 1

	7	6	5	4	3	2	1	0	
r	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	
w	x	x	x	x	x	x	x	BUF	
								0	Res

EOC End of Conversion
r1: End of conversion
r0: Busy
EOC is reset by a write access to the register AD0. EOC must be true before starting the a conversion. EOC is true after enabling the module by setting SR0.ADC.

TSAMP Sampling Time
TSAMP adjusts the sample conversion times.

Table 15–2: TSAMP usage: Sample and conversion time

TSAMP	t_{Sample}	$t_{Conversion}$
0H	$20/f_0$	$40/f_0$
1H	$60/f_0$	$80/f_0$
2H	$140/f_0$	$160/f_0$
3H	$300/f_0$	$320/f_0$

REF Conversion Reference
w0: External reference from VREF pin used
w1: Internal reference on VREFINT pin used
w2: External reference from VREF0 pin used
w3: External reference from VREF1 pin used

CHANNEL Channel of Input Multiplexer
CHANNEL selects from which pin of port P0 or P1 the conversion is done. The MSB of CHANNEL is bit 3.

Table 15–3: CHANNEL usage: ADC input selection

CHANNEL	Port Pin
0 to 7	P0.0 to P0.7
8 to 15	P1.0 to P1.7

AN 9 to 0 Analog Value Bit 9 to 0
The 10-bit data format is positive integer, i.e., 000H for lowest and 3FFH for highest possible input signal. The 8 MSB can be read from register AD1. The two LSB can be read from register AD0. The result is available until a new conversion is started.

BUF Input Buffer Usage
w1: Buffer used
w0: Buffer bypassed

TEST for factory use only

ANAA Analog AVDD Register

	7	6	5	4	3	2	1	0	
r/w	EP06	P06	WAIT	x	x	x	x	BVE	
	0							0	Res

EP06 Enable P06 Comparator Interrupt Source output
r/w1: Enabled.
r/w0: Disabled.

P06 P06 Comparator Output
r1: P0.6 is lower than AVDD/2.
r0: P0.6 is higher than AVDD/2.

WAIT WAIT Comparator Output
r1: WAIT is lower than VREFINT.
r0: WAIT is higher than VREFINT.

BVE BVDD Regulator Error Flag
r1: Out of specification.
r0: Normal operation.
w1: Reset flag.
w0: No action.

16. Timers (TIMER)

Five general-purpose timers are implemented. T0 is a 16-bit timer, T1 to T4 are 8-bit timers.

16.1. Timer T0

Timer T0 is a 16-bit auto reload down counter. Its function is to deliver a timing reference signal to the ICU, to output a frequency signal or to produce time stamps.

Features

- 16-bit auto reload counter
- Time value readable
- Interrupt source output
- Frequency output

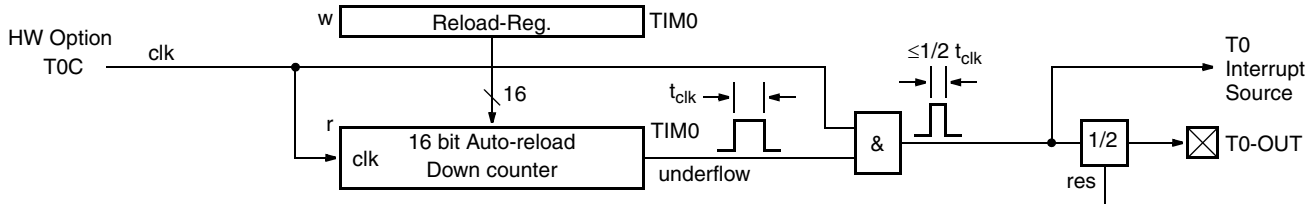


Fig. 16–1: Timer T0 block diagram

16.1.1. Principle of Operation

16.1.1.1. General

The timer's 16-bit down-counter is clocked by the input clock and counts down to zero. One clock count after reaching zero, it generates an output pulse, reloads with the content of the TIM0 reload register and restarts its travel.

For the effect of CPU clock modes on the operation of this module refer to section "CPU and Clock System" (see Table 4–1 on page 41).

16.1.1.2. Operation

The clock input frequency is settable by HW option (see Table 16–1 on page 118).

Prior to entering active mode, proper SW initialization of the U-ports assigned to function as T0-OUT outputs has to be made (Table 16–1). The ports have to be configured special out. Refer to "Ports" for details.

T0 is always active (no standby mode). After reset, the timer starts counting with reload value 0xFFFF generating a maximum period output signal.

A new time value is loaded by writing to the 16-bit register TIM0, high byte first. Upon writing the low byte, the reload register is set to the new 16-bit value, the counter is reset, and immediately starts down-counting with the new value.

After reaching zero, on wrapping to 0xFFFF, the counter generates a reload signal, which can be used to trigger an interrupt. The same signal is connected to a divide-by-two scaler to generate the output signal T0-OUT with a pulse duty factor of 50%.

The interrupt source output of this module is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section "Interrupt Controller" for the actually selectable sources and how to select them.

The state of the down-counter is readable by reading the 16-bit register TIM0, low byte first. Upon reading the low byte, the high byte is saved to a temporary latch, which is then accessed during the subsequent high byte read. Thus, for time stamp applications, read consistency between low and high byte is guaranteed.

16.1.1.3. Precautions

Use 8-bit load/store operations to access Timer 0 register rather than 16-bit access.

A load with a new value within a time period of < $t_{clk}/2$ before a scheduled interrupt source output signal, can no longer cancel this signal. It will appear at the interrupt source output anyway (See fig. 16–2 for details).

Thus, after loading a new time value, wait at least $t_{clk}/2$ before resetting the pending flag P and enabling the interrupt channel.

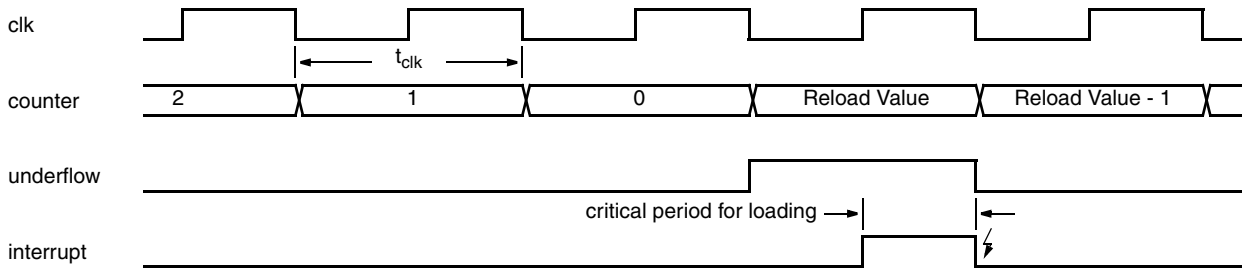
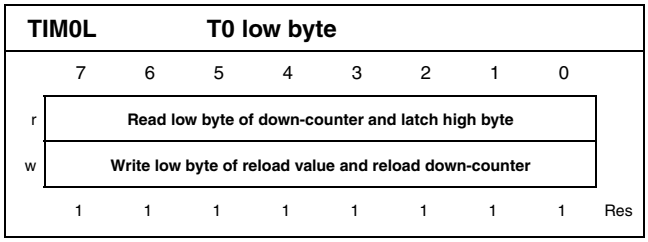


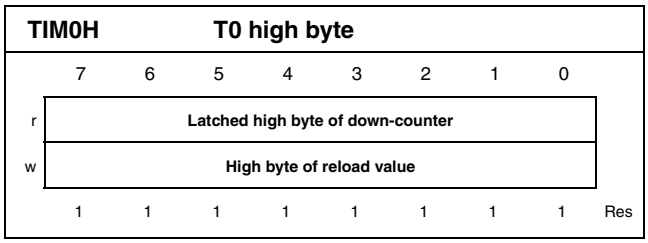
Fig. 16–2: Timer0 timing

Table 16–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
T0	Input clock	T0C	T0-OUT output	U1.2 special out	

16.1.2. Registers





TIM0 has to be read low byte first and written high byte first.

Table 16–2: Reload register programming

Reload value	Output interrupt source frequency is divided by	Output T0-OUT is divided by
0x0000	1	2
0x0001	2	4
0x0002	3	6
:	:	:
0xFFFF	65536	131072

16.2. Timer T1 to T4

Timer T1 to T4 are 8-bit auto reload down counters. They serve to deliver timing reference signals to the ICU or to output frequency signals.

Table 16–3 describes implementation-specific HW option addresses and enable flags of T1 to T4.

Features

- 8-bit auto reload counter
- Interrupt source output
- Frequency output

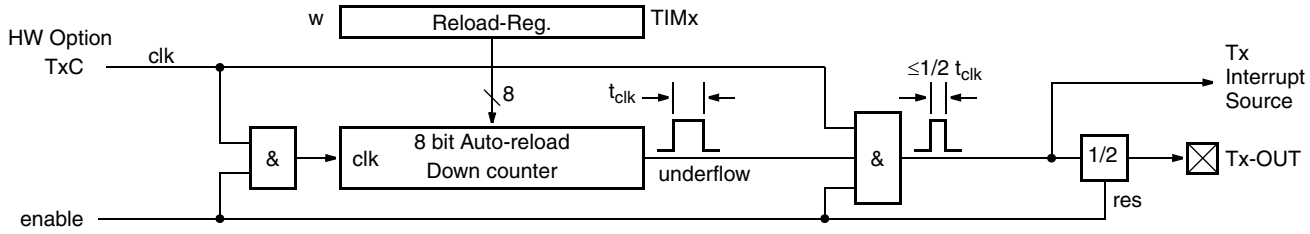


Fig. 16–3: Timer T1 to T4 block diagram

16.2.1. Principle of Operation

16.2.1.1. General

The timer's 8-bit down-counter is clocked by the input clock and counts down to zero. One clock count after reaching zero, it generates an output pulse, reloads with the content of the TIMx reload register and restarts its travel.

For the effect of CPU clock modes on the operation of this module refer to section "CPU and Clock System" (see Table 4–1 on page 41).

16.2.1.2. Operation

The clock input frequencies can be set with HW options (see Table 16–3 on page 120). After reset, the 8-bit timer is in standby (inactive) mode.

Prior to entering active mode, proper SW initialization of the U-Ports assigned to function as Tx-OUT outputs has to be made (Table 16–3). The ports have to be configured as special out. Refer to "Ports" for details.

To initialize a timer, reload register TIMx can be set to the desired time value already in standby mode.

For entering active mode, set the corresponding enable bit in the standby registers (see Table 16–3 on page 120). The timer will immediately start counting down from the time value present in register TIMx.

During active mode, a new time value is loaded by simply writing to register TIMx. Upon writing, the counter is reset, and immediately starts counting down from the new time value.

After reaching zero, on wrapping to 0xFF, the counter generates a reload signal, which can be used to trigger an interrupt. The same signal is connected to a divide by two scaler to generate the output signal Tx-OUT with a pulse duty factor of 50%.

The interrupt source output of this module may be but must not be connected to the interrupt controller. Please refer to section interrupt controller.

Returning Tx to standby mode by resetting its respective enable bit will halt its counter and will set its outputs LOW. The register TIMx remains unchanged.

The state of the down-counter is not readable.

16.2.1.3. Precautions

A load with a new value within a time period of $< t_{clk} / 2$ before a scheduled interrupt source output signal, can no longer cancel this signal. It will appear at the interrupt source output anyway (See Fig. 16–4 for details).

Furthermore, disabling the timer within a time period of $< t_{clk} / 2$ before a scheduled interrupt source output signal, will immediately generate an extra interrupt source output signal. Re-enabling the timer afterwards will lead to generation of the previously scheduled interrupt source output signal, because it was stored internally. This latter interrupt source output signal will be generated even if a new time value is loaded during the inactive time.

Thus after configuring and (re-)enabling the timer, or after loading a new time value during active mode, wait at least $t_{clk} / 2$ before resetting the pending flag P and (re-)enabling the interrupt channel.

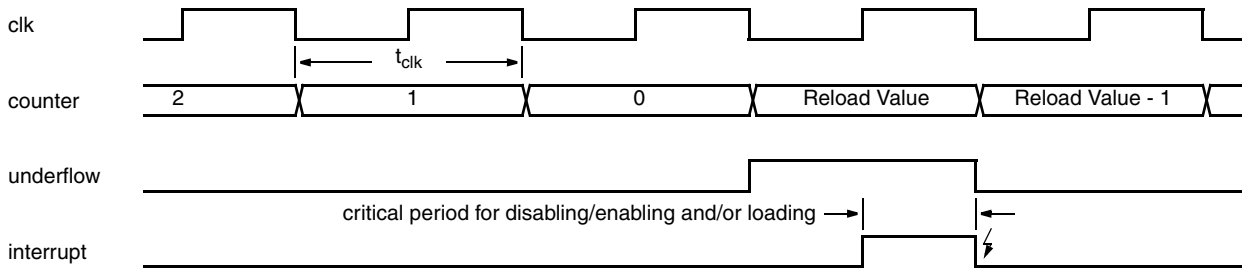


Fig. 16–4: Timer1 to 4 timing

Table 16–3: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
T1	Input clock	T1C	T1-OUT output	U1.1 special out	SR0.TIM1
T2	Input clock	T2C	T2-OUT output	U1.0 special out	SR0.TIM2
T3	Input clock	T3C	T3-OUT output	U0.7 special out	SR0.TIM3
T4	Input clock	T4C	T4-OUT output	U0.6 special out, PM.U06 = 0	SR0.TIM4

16.2.2. Registers

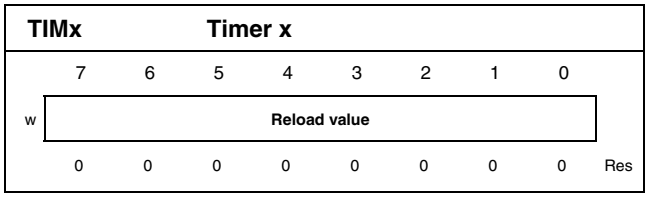


Table 16–4: Reload register programming

Reload value	Output interrupt source frequency is divided by	Output Tn-OUT is divided by
0x00	1	2
0x01	2	4
0x02	3	6
:	:	:
0xFF	256	512

17. Pulse Width Modulator (PWM)

A PWM is an auto-reload down-counter with fixed reload interval. It serves as a generator of a frequency signal with variable pulse width or, with an external low-pass filter, as a digital-to-analog converter.

This module is combined of two independently operable 8-bit PWMs which can be combined to a single 16-bit PWM.

The number of PWMs implemented is given in table 17–1. The “x” in register names distinguishes the module number and can be 1, 3, 5, 7, 9, 11.

Features

– Two 8-bit or one 16-bit pulse width modulator

– Wide range of HW-option-selectable cycle frequencies

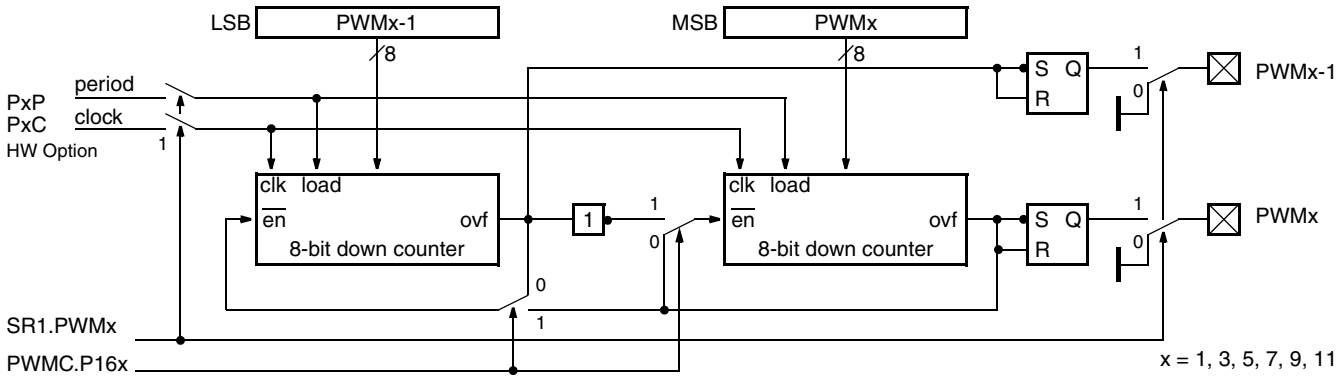


Fig. 17–1: PWM block diagram

17.1. Principle of Operation

17.1.1. General

A PWM’s down-counter is clocked by its input clock and counts down to zero. Reaching zero, it stops and sets the output to LOW. A period input pulse reloads the counter with the content of the PWM register, restarts it and sets the output to HIGH.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

17.1.2. Hardware settings

The clock and period input frequencies can be set by HW option (Table 17–1). There is one common source for both 8-bit PWMs, one for clock and one for period, thus clock and period are not independently selectable for the two 8-bit PWMs. For full resolution, a clock-to-period frequency ratio of 256 (65536 in 16-bit mode) is recommended. Should other ratios be used, make sure that the combination of clock, period and pulse width setting allow the PWM to generate an output signal with a LOW transition.

Some of the PWM outputs share pins with outputs of other modules. The output multiplexer is controlled by HW option (Table 17–1).

17.1.3. Initialization

Prior to entering active mode, proper SW initialization of the H-Ports and U-Ports assigned to function as PWMx outputs has to be made (Table 17–1). The ports have to be configured special out. Refer to “Ports” for details.

It has to be decided which PWM module shall work as one 16-bit or as two 8-bit PWMs. Selection has to be done via the PWM control register PWMC as long as the PWM module is disabled.

17.1.4. Operation

After reset, a PWM is in standby mode (inactive) and the output signal PWMx is LOW.

For entering active mode, select the desired mode (8-/16-bit mode) and then set the respective enable bit (Table 17–1). Then write the desired pulse width value to register PWMx (write low byte first in 16-bit mode). Each PWM will start producing its output signal immediately after the next subsequent input pulse on its period input.

During active mode, a new pulse width value is set by simply writing to the register PWMx. Upon the next subsequent input pulse on its period input the PWM will start producing an output signal with the new pulse width value, starting with a HIGH level.

Table 17–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
PWM1	Clock and period	P1C, P1P	PWM0	U0.3 special out	SR1.PWM1
	H0.3 SMG/PWM1 output multiplexer	PM.H0	PWM1	U0.2 and/or H0.3 special out	
PWM3	Clock and period	P3C, P3P	PWM2	U0.1special out	SR1.PWM3
	H0.2 SMG/PWM3 output multiplexer	PM.H0	PWM3	U0.0 and/or H0.2 special out	
PWM5	Clock and period	P5C, P5P			SR1.PWM5
	H7.3 SME/PWM4 output multiplexer	PM.H7	PWM4	H7.3 special out	
	H0.1 SMG/PWM5 output multiplexer	PM.H0	PWM5	H0.1 special out	
PWM7	Clock and period	P7C, P7P			SR1.PWM7
	H7.2 SME/PWM6 output multiplexer	PM.H7	PWM6	H7.2 special out	
	H0.0 SMG/PWM7 output multiplexer	PM.H0	PWM7	H0.0 special out	
PWM9	Clock and period	P9C, P9P			SR1.PWM9
	H7.1 SME/PWM8 output multiplexer	PM.H7	PWM8	H7.1 and/or H6.3 special out	
	H7.0 SME/PWM9 output multiplexer	PM.H7	PWM9	H7.0 and/or H6.2 special out	
PWM11	Clock and period	P11C, P11P	PWM10	H6.1 special out	SR1.PWM11
			PWM11	H6.0 special out	

Returning a PWM to standby mode by resetting its respective enable flag will immediately set its output LOW.

The 8-bit PWM output PWMx-1 is not usable in 16-bit mode.

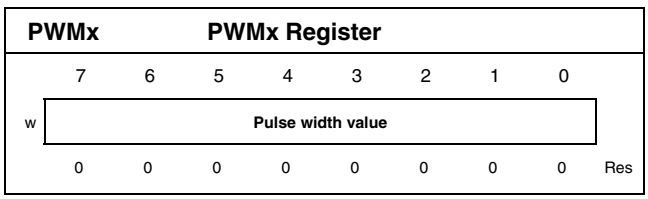
The state of the down-counters and the PWM registers is not readable.

Due to EMI reduction the start of a period is delayed for different PWMs (Table 17–2).

Table 17–2: Module delay

Module Number	Delay
PWM 0, 4, 8	0
PWM 1, 5, 9	$1/f_0$
PWM 2, 6, 10	$2/f_0$
PWM 3, 7, 11	$3/f_0$

17.2. Registers



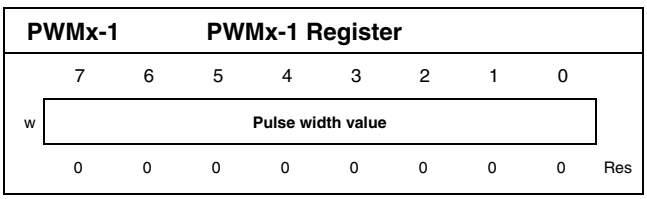


Table 17–3: 8-bit mode pulse width programming

Pulse width value	Pulse duty factor
0x00	0% (Output is permanently low)
0x01	1/256
0x02	2/256
:	:
0xFE	254/256
0xFF	100% (Output is permanently high) [1])
[1]) Pulse duty factor 255/256 is not selectable.	

Table 17–4: 16-bit mode pulse width programming

Pulse width value	Pulse duty factor
0x0000	0% (Output is permanently low)
0x0001	1/65536
0x0002	2/65536
:	:
0xFFFE	65534/65536
0xFFFF	100% (Output is permanently high) [1])
[1]) Pulse duty factor 65535/65536 is not selectable.	

PWMC PWM Control Register

	7	6	5	4	3	2	1	0	
w	x	x	P1611	P169	P167	P165	P163	P161	
	x	x	0	0	0	0	0	0	Res

P16x PWM 16 Mode of Module x
w1: 16-bit mode.
w0: 8-bit mode.

18. Pulse Frequency Modulator (PFM)

The PFM generates a signal with variable frequency and variable pulse width. Together with external elements, it may serve to generate a negative voltage for LCD elements.

Features

- Pulse width and period separately controllable
- Pulse width and period counters operate with HW option selectable clock
- Output polarity selectable
- Standby mode

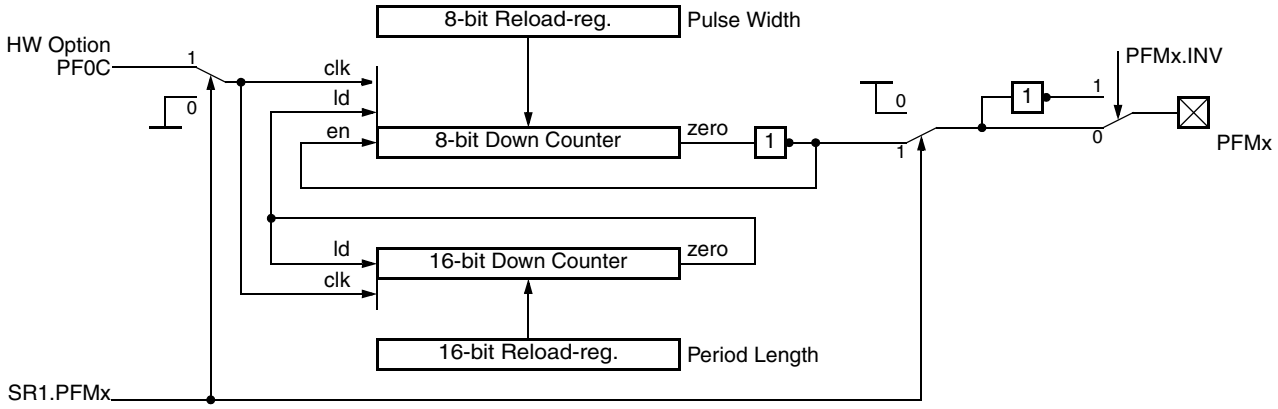


Fig. 18–1: PFM block diagram

18.1. Principle of Operation

18.1.1. General

The pulse width and the period counter start synchronously with down-counting. As long as the pulse width counter is running, its zero output is LOW. When this counter reaches zero, it stops counting and sets the zero output to HIGH. When the period counter reaches zero, it reloads both counters, which starts a new count cycle. The zero output of the pulse width counter can be driven out directly or inverted via pin PFMx.

The module is operable in PLL, FAST and SLOW mode. As long as PF0C is available it is also operable in DEEP SLOW mode. See also chapter “CPU and Clock System” for further details.

18.1.2. Hardware Settings

The clock input frequency PF0C can be set by HW option (see Table 18–1).

Table 18–1: Module-specific settings

HW Options		Initialization		Enable Bit
Item	**Address**	**Item**	**Setting**	
Input clock	PF0C	PFM0	U5.0 and/or U1.7 special out	SR1.PFM0
Input clock	PF0C	PFM1	U7.5 special out 1)	SR1.PFM1
1) Make sure that flag LE in register ANAU is set to zero, otherwise an internal test signal is output at this pin.				

18.1.3. Initialization

Prior to entering active mode, proper SW initialization of the U-Ports assigned to function as PFMx output has to be made (Table 18–1). The ports have to be configured as special out. Refer to “Ports” for details.

18.1.4. Operation

After reset the PFM is in standby mode (inactive) and the output signal is LOW.

To prepare for active mode, write new values, if needed, for the pulse width and the period length to the respective PFMx register and select output inversion, if necessary, with flag INV. For entering active mode set the enable bit SR1.PFMx.

Changing the PFMx register setting during active mode, is simply done by writing a 32-bit word to this register. After the register has been updated, the PFM will produce an output signal with the new pulse width and period length starting with the next subsequent load signal of the period counter.

For data consistency, when using 8-bit and 16-bit writes, new values will only become valid after a write to the pulse-width register (byte 2 in the PFMx register).

Returning the PFM to standby mode by clearing its enable bit SR1.PFMx will immediately set its output to INV and disable the clock input. The content of the PFMx register is not affected by standby mode.

The state of the counters and the reload registers is not readable.

18.2. Registers

PFMx				Pulse Width and Period Register					
	7	6	5	4	3	2	1	0	Offs
w	INV	x	x	x	x	x	x	x	3
w	Pulse Width								2
w	Period Length (High Byte)								1
w	Period Length (Low Byte)								0
	0x00								Res

INV **Invert Output Signal**
r/w1: inverted
r/w0: direct

The pulse width counter zero output HIGH time is calculated by

$$t_{HIGH} = \frac{\text{Pulse Width}}{F_{PF0C}}$$

and the duration of the period time by

$$t_{Period} = \frac{\text{Period Length}}{F_{PF0C}}$$

Therefore, the pulse width counter zero output LOW time is

$$t_{LOW} = t_{Period} - t_{HIGH}$$

Table 18–2 shows the relation between the pulse width and the period length and its effect on the PFMx output.

Table 18–2: Pulse width to period length relation

Pulse Width	Period Length	INV	PFMx output
0	x	0	Always low
> 0	≤ Pulse Width		Always high
> 0	> Pulse Width		High pulses
0	x	1	Always high
> 0	≤ Pulse Width		Always low
> 0	> Pulse Width		Low pulses

19. Capture Compare Module (CAPCOM)

The IC contains two capture compare modules (CAPCOM).

A CAPCOM is a complex relative timer. It comprises a free-running 16-bit capture compare counter (CCC) and a number of subunits (SU). The timer value can be read by SW.

A SU is able to capture the relative time of an external event input and to generate an output signal when the CCC passes a predefined timer value. Three types of interrupts enable interaction with SW. Special functionality provides an interface to the asynchronous external world.

- 16-bit free-running counter with read-out.
- 16-bit capture register.
- 16-bit compare register.
- Input trigger on rising, falling or both edges.
- Output action: toggle, low or high level.
- Three different interrupt sources: overflow, input, compare
- Designed for interface to asynchronous external events

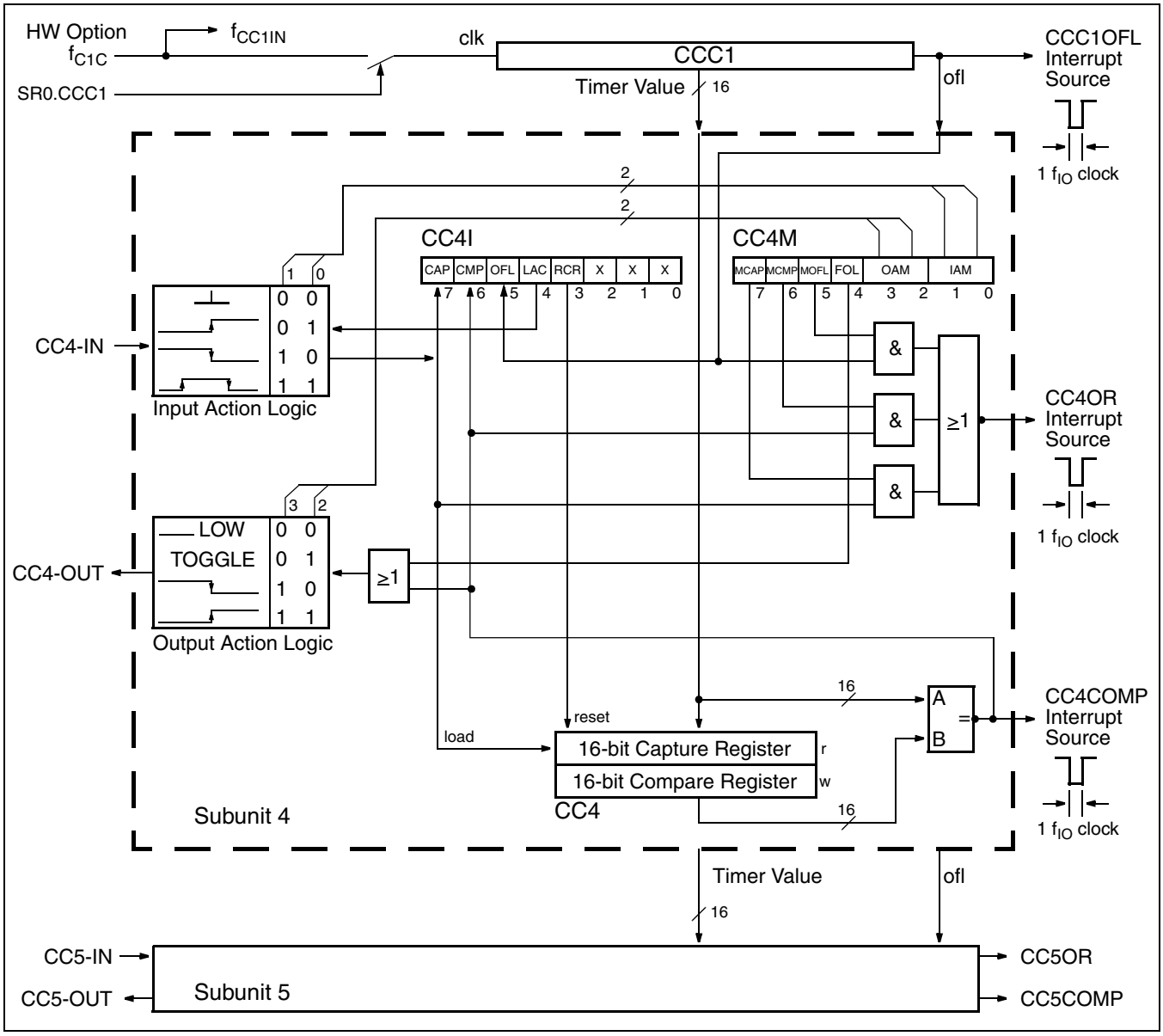


Fig. 19–1: CAPCOM module 1 block diagram

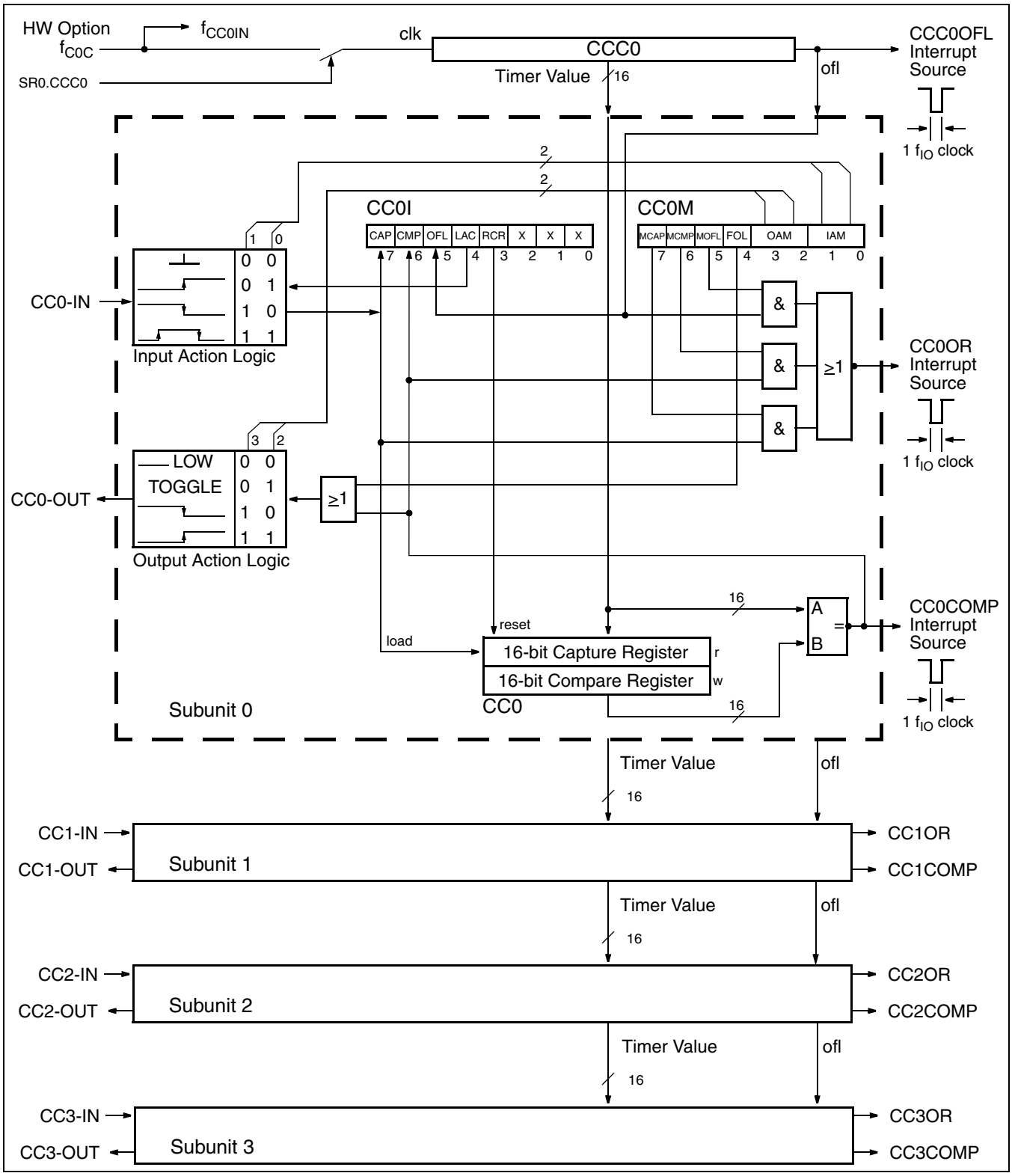


Fig. 19–2: CAPCOM module 0 block diagram

19.1. Principle of Operation

19.1.1. General

The capture compare module (CAPCOM, Fig. 19–1, 19–2) contains one common free-running 16-bit counter (CCC) and a number of capture and compare subunits (SU). More details are given in Tables 19–1 and 19–2. The timer value can be read by SW from 16-bit register CCC. The CCC provides an interrupt on overflow.

Each SU is able to capture the CCC value at a point of time given by an external input event processed by an input action logic.

A SU can also change an output line level via an output action logic at a point of time given by the CCC value.

Thus, a SU contains a 16-bit capture register CCx to store the input event CCC value, a 16-bit compare register CCx to program the output action CCC value, an 8-bit interrupt register CCxI and an 8-bit mode register CCxM. Two types of interrupts per SU enable interaction with SW.

The interrupt source output of this module is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

19.1.2. Hardware Settings

The CCC0 and CCC1 clock frequency must be set via HW option (Table 19–1 and 19–2). Some SUs use several ports. They can be selected via HW option port multiplexer (PM). Refer to “HW Options” for setting them.

19.1.3. Initialization

After system reset the CCC and all SUs are in standby mode (inactive).

In standby mode, the CCC is reset to value 0x0000. Capture and compare registers CCx are reset (comp. reg to 0xFFFF, capt. reg to 0x0). No information processing will take place, e.g., update of interrupt flags. However, the values of registers CCxI and CCxM are only reset by system reset, not by standby mode. Thus it is possible to program all mode bits in standby mode and a predetermined start-up out of standby mode is guaranteed. The flags CCxI.CAP, CCxI.CMP and CCxI.OFL are read-only during standby mode.

Prior to entering active mode, proper SW configuration of the U-Ports, assigned to function as input capture inputs and output action outputs has to be made (Table 19–1, 19–2). The output action ports have to be configured as special out and the Input Capture ports as special in. Refer to “Ports” for details.

19.1.3.1. Subunit

For a proper setup the SW has to program the following SU control bits in registers CCxI and CCxM: interrupt mask (MSK), force output logic (FOL, 0 recommended), output action mode (OAM), input action mode (IAM), reset capture register (RCR, 0 recommended), and lock after capture (LAC). Refer to section 19.2. for details.

Please note that the compare register CCx is reset in standby mode. It can only be programmed in active mode.

Table 19–1: Unit 0 specific settings

Sub-unit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SU0			CC0-OUT	U3.2, U4.0 special out	SR0. CCC0
	Input	PM. CACO	CC0-IN	U3.2, U4.1 special in	
SU1			CC1-OUT	U3.1, U2.5 special out	
	Input	PM. CACO	CC1-IN	U3.1, U2.4 special in	
SU2			CC2-OUT	U3.0, U2.3 special out	
	Input	PM. CACO	CC2-IN	U3.0, U2.2 special in	
SU3	Output	PM. U06	CC3-OUT	U0.5, U0.6, U8.1 special out	
			CC3-IN	U0.6 special in	
SU0, SU1, SU2, SU3	Clock	C0C			

Table 19–2: Unit 1 specific settings

Sub-unit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SU4			CC4-OUT	U5.3, U8.0 special out	SR0. CCC1
	Input	PM. CC4I	CC4-IN	U5.3, P0.0 special in	
SU5			CC5-OUT	U7.4 special out	
			CC5-IN	U7.4 special in	
SU4, SU5	Clock	C1C			

19.1.4. Operation of CCC

For entering active mode of the entire CAPCOM module set the enable bit (Table 19–1 and 19–2).

The CCC will immediately start up-counting with the selected clock frequency and will deliver this 16-bit value to the SUs.

The state of the counter is readable by reading the 16-bit register CCC, low byte first. Upon reading the low byte, the high byte is saved to a temporary latch, which is then accessed during the subsequent high byte read. Thus, for time stamp applications, read consistency between low and high byte is guaranteed.

The CCC is free-running and will overflow from time to time. This will cause the generation of an overflow interrupt event. The interrupt (CCCxOFL) is directly fed to the interrupt controller and also to all SUs where further processing takes place.

19.1.5. Operation of Subunit

19.1.5.1. Compare and Output Action

To activate a SUs compare logic the respective 16-bit compare register CCx has to be programmed, low byte first. The compare action will be locked until the high byte write is completed. As soon as CCx setting and CCC value match, the following actions are triggered:

– The flag CMP in the CCxI register is set.

– The CCxCOMP interrupt source is triggered.

– The CCxOR interrupt source is triggered if activated.

– The output action logic is triggered.
Four different reactions are selectable for the output action signal: according to field CCxM.OAM (Table 19–3) the equal state will lead to a high or low level, toggling or inactivity on this output.
Another way to control the output action is bit CCxM.FOL. For instance, rise-mode and force will set the output pin to high level, fall-mode and force to low level. This forcing is static, i.e., it will be permanently active and may override compare events. Thus it is recommended to set and reset shortly after that, i.e., to pulse the bit with SW. Toggle mode of the output action logic and forcing leads to a burst with clock-frequency and is not recommended.

19.1.5.2. Capture and Input Action

The input action logic operates independently of the output action logic and is triggered by an external input in a way defined by field CCxM.IAM. Following Table 19–4 it can completely ignore events, trigger on rising or falling edge or on both edges. When triggered, the following actions take place:

– Flag CCxI.CAP is set.

– The CCxOR interrupt source is triggered if activated.

– The 16-bit capture register CCx stores the current CCC value, i.e., the “time” of the external event. Read CCx low byte first. Further capture and input action will be locked until the subsequent high byte read is completed. Thus a coherent result is ensured, no matter how much time has elapsed between the two reads.

Some applications suffer from fast input bursts and a lot of capture events and interrupts in consequence. If the SW cannot handle such a rate of interrupts, this could evoke stack overflow and system crash. To prevent such fatal situations the lock after capture (LAC) mode is implemented. If bit CCxI.LAC is set, only one capture event will pass. After this event has triggered a capture, the input action logic will lock until it is unlocked again by writing an arbitrary value to register CCxM. Make sure that this write only restores the desired setting of this register.

Programming the input action logic while an input transition occurs may result in an unexpected triggering. This may overwrite the capture register, lock the input action logic if in LAC mode and generate an interrupt. Make sure that SW is prepared to handle such a situation.

For testing purposes, a permanent reset (0xFFFF) may be forced on capture register CCx by setting bit CCxI.RCR. Make sure that the reset is only temporary.

19.1.5.3. Interrupts

Each SU supplies two internal interrupt events:

1. Input capture event and

2. Comparator equal state.

In addition to the above-mentioned two interrupt events, the CCC overflow interrupt event sets flag CCxI.OFL in each SU. Thus, three interrupt events are available in each SU. As previously explained, interrupt events will set the corresponding flags in register CCxI. Those three interrupt events are masked with their mask bits in register CCxM and passed to a logical or. The result (CCxOR) is fed to the interrupt controller as a first interrupt source. In addition, the comparator equal (CCxCOMP) interrupt is directly passed to the interrupt controller as second interrupt source. Thus a SU offers four types of interrupts: CCC overflow (maskable ored), input capture event (maskable ored) and comparator equal state (maskable ored and non-maskable direct).

All interrupt sources act independently, parallel interrupts are possible. The interrupt flags enable SW to determine the interrupt source and to take the appropriate action. Before returning from the interrupt routine the corresponding interrupt flag should thus be cleared by writing a 1 to the corresponding bit location in register CCxI.

The interrupts generated by internal logic (CCC overflow and comparator equal) will trigger in a predetermined and known way. But as explained in 19.1.5.2. erroneous input signals may cause some difficulties concerning the input capture input as well as interrupt handling. To overcome possible problems the input capture Interrupt flag CCxI.CAP is double-buffered. If a second or even more input capture interrupt events occur before the interrupt flag is cleared (i.e. SW was not able to keep track), the flag goes to a third state. Two consecutive writes to this bit in register CCxI are then necessary to clear the flag. This enables SW to detect such a multiple interrupt situation and eventually to discard the capture register value, which always relates to the latest input capture event and interrupt.

The internal CAPCOM module control logic always runs on the clock divider chain f_0 frequency, regardless of CPU clock mode. Avoid write accesses to the CCxI register in CPU slow mode since the logic would interpret one CPU access as many consecutive accesses. This may yield to unexpected results concerning the functionality of the interrupt flags. The following procedure should be followed to handle the capture interrupt flag CAP:

1. SW responds to a CAPCOM interrupt, switching to CPU Fast or PLL mode if necessary and determining that the

source is a capture interrupt (CAP flag =1).

2. The interrupt service routine is processed.

3. Just before returning to main program, the service routine acknowledges the interrupt by writing a 1 to flag CAP.

4. The service routine reads CAP again. If it is reset, the routine can return to main program as usual. If it is still set, an external capture event overrun has happened. Appropriate actions may be taken (i.e., discarding the capture register value etc.).

5. go to 3.

19.1.6. Inactivation

The CAPCOM module is inactivated and returned to standby mode (power down mode) by setting the enable bit to 0. Section 19.1.3. applies.

CCxI and CCxM are only reset by system reset, not by standby mode.

19.1.7. Precautions

The CCxI register must not be written in CPU Slow mode (see Section 19.1.5.3. on page 130). Read-Modify-Write operations on single flags of register CCxI must be avoided. Unwanted clearing of other flags of this register may be the result otherwise.

19.2. Registers

The CAPCOM counter and the Capture/Compare registers have to be read/written low byte first to avoid inconsistencies. The memory controller accesses multiple byte quantities low byte first. Thus the 16-bit CAPCOM counter and the 16-bit capture/compare registers can be accessed 16 bit wide.

CCCyL CAPCOM Counter low byte

	7	6	5	4	3	2	1	0	
r	Read low byte and lock CCC								
	0	0	0	0	0	0	0	0	Res

CCCyH CAPCOM Counter high byte

	7	6	5	4	3	2	1	0	
r	Read high byte and unlock CCC								
	0	0	0	0	0	0	0	0	Res

CCxM CAPCOM x Mode Register

	7	6	5	4	3	2	1	0	
r/w	MCAP	MCMP	MOFL	FOL	OAM		IAM		
	0	0	0	0	0	0	0	0	Res

MCAP **Mask CAP Flag**
r/w1: Enable.
r/w0: Disable.

MCMP **Mask CMP Flag**
r/w1: Enable.
r/w0: Disable.

MOFL **Mask OFL Flag**
r/w1: Enable.
r/w0: Disable.

FOL **Force Output Action Logic**
r/w1: Force Output Action logic.
r/w0: Release Output Action logic.
This flag is static. As long as FOL is true, neither comparator can trigger, nor SW can force the output action logic by writing another “one”. After forcing it is recommended to clear FOL unless output action logic should not be locked.

OAM **Output Action Mode**
r/w: Defines behavior of Output Action logic.

Table 19–3: OAM usage

Bit 3 2	Output Action Logic Modes
0 0	Disabled, ignore trigger, output low level.
0 1	Toggle output.
1 0	Output low level.
1 1	Output high level.

IAM **Input Action Mode**
r/w: Defines behavior of Input Action logic.

Table 19–4: IAM usage

Bit 1 0	Input Action Logic Modes
0 0	Disabled, don’t trigger.
0 1	Trigger on rising edge.
1 0	Trigger on falling edge.
1 1	Trigger on rising and falling edge.

CCxI CAPCOM x Interrupt Register

	7	6	5	4	3	2	1	0	
r/w	CAP	CMP	OFL	LAC	RCR	x	x	x	
	0	0	0	0	0	0	0	0	Res

CAP **Capture Event**
r1: Event.
r0: No Event.
w1: Clear flag.
w0: No change.

CMP **Compare Event**
r1: Event.
r0: No Event.
w1: Clear flag.
w0: No change.

OFL **Overflow Event**
r1: Event.
r0: No Event.
w1: Clear flag.
w0: No change.

LAC **Lock After Capture**
r/w1: Enable.
r/w0: Disable.
Refer to section 19.1.5.2.

RCR **Reset Capture Register**
r/w1: Reset capture register permanently to 0xFFFF.
r/w0: Release capture register.

CCxL **CAPCOM x Capture/Compare Register low byte**

	7	6	5	4	3	2	1	0	
r	Read low byte of capture register and lock it.								
w	Write low byte of compare register and lock it.								
	1	1	1	1	1	1	1	1	Res

CCxH **CAPCOM x Capture/Compare Register high byte**

	7	6	5	4	3	2	1	0	
r	Read high byte of capture register and unlock it.								
w	Write high byte of compare register and unlock it.								
	1	1	1	1	1	1	1	1	Res

20. Stepper Motor Module (SMM)

The SMM serves as a controller for air-cored movements or stepper motors that are directly coupled in H-bridge formation to H-Ports. Upon CPU programming it creates all waveforms necessary to position the drive pointer as desired.

The number of motors that are controllable by subunits (control units) of the module is given in Table 20–4.

Features

- Multichannel pulse-width modulated output
- Outputs offset for improved EMC properties
- Four quadrant operation
- 8-bit resolution
- HW option selectable output cycle frequency

20.1. Functional Description

20.1.1. General

An 8-bit, free-running counter FRC (Fig. 20–2) operates on the f_{SM} input clock (generally 4 MHz) and creates an 8-bit counter word that is fed to a number of control units SMx.

A control unit (Fig. 20–2) contains 8-bit sine and cosine compare registers. One comparator each is associated with these registers and creates a compare signal when register content and FRC word are equal. An output flip-flop associated with each comparator is set when the FRC word is zero and reset by the respective compare signal. A delay stage associated with each control unit delays the flip-flop output signals by a fixed number of f_{SM} cycles to achieve non-synchronism between the output signals of the various control units, thus achieving an improved EMC behavior of the SMV (cf. Fig. 20–1). According to the setting of a quadrant register associated with each control unit, each of a unit's two output signals is multiplexed to signals SMxn+ and SMxn- so as to properly control 2 individual H-Ports that form an H-bridge together with the connected motor coil. By these means, a control unit supplies two H-bridges with signals SMx1+, SMx1-, SMx2+ and SMx2- to function as variable pulse width modulator outputs with selectable polarity.

Summing up: when the compare registers are set to the sine and cosine value of a desired rotor angle and the quadrant register is set to the desired quadrant, an air cored movement or a stepper motor connected to the unit's 4 H-Ports will carry the proper average coil currents of proper polarity so that its rotor will assume the desired rotary angle.

Three registers control readjustment of a rotor to a new angle. Sine, cosine and unit/quadrant registers serve as temporary storage of new sine, cosine, related quadrant and unit selection values. A scheduler logic times the synchronous downloading of the three buffered words to the respective unit's sine, cosine and quadrant registers, so as to avoid inconsistencies among them. A busy bit may be read out, signaling completion of the downloading.

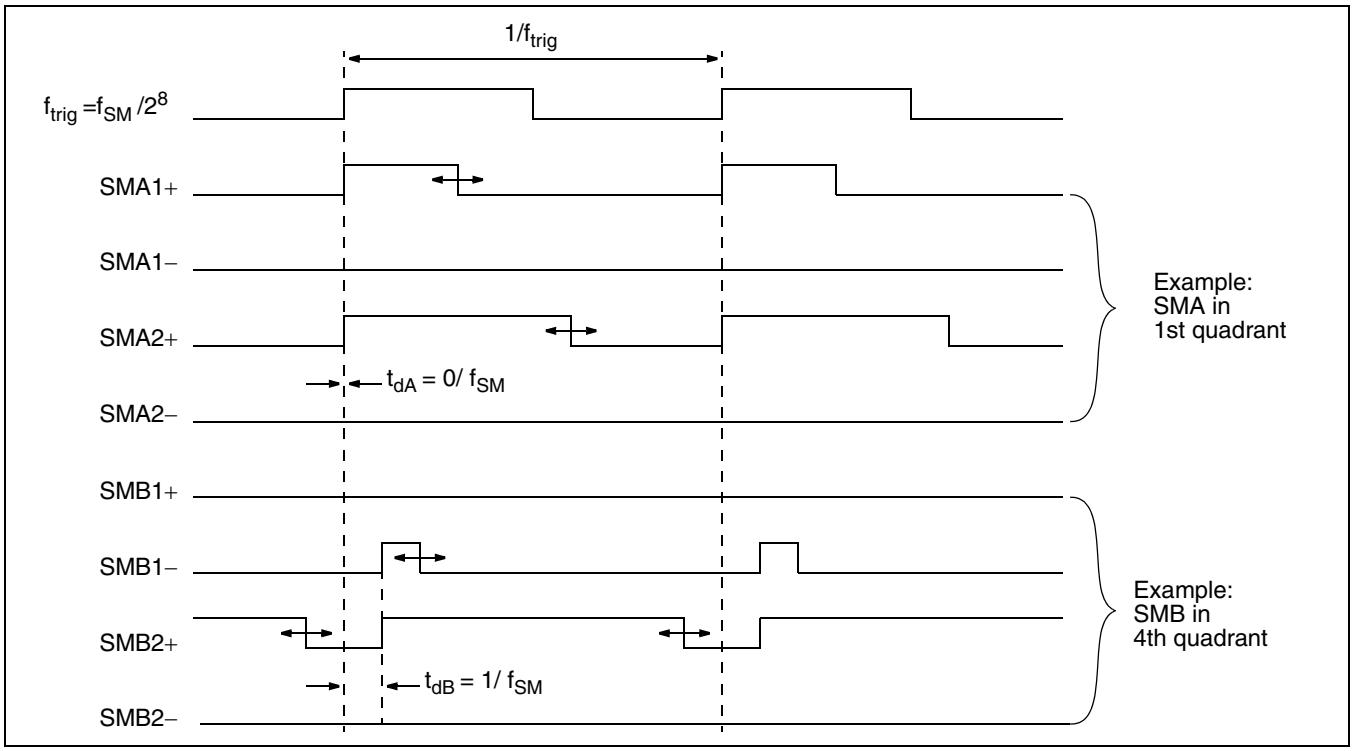


Fig. 20–1: Timing diagram of output signals

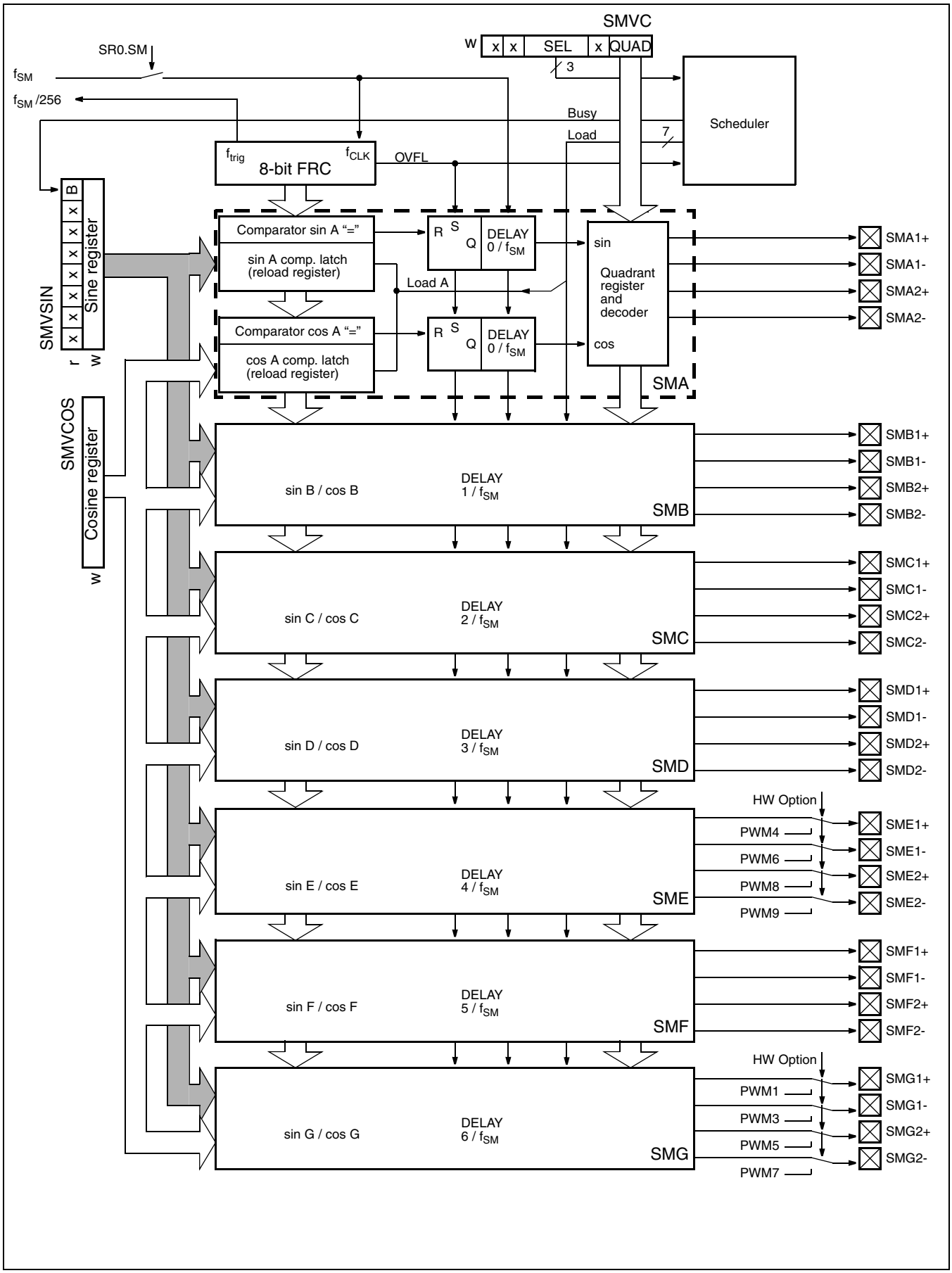


Fig. 20–2: Block diagram of output generation circuit

20.2. Registers

SMVC SMM Control Register

	7	6	5	4	3	2	1	0
w	x	x	SEL			x	QUAD	
	x	x	0	0	0	x	0	0 Res

SEL **Control unit Selection field (Table 20–1)**

QUAD **Quadrant selection field (Table 20–2)**

Table 20–1: SEL usage

SEL	selected control unit
000	SMA
001	SMB
010	SMC
011	SMD
100	SME
101	SMF
110	SMG
111	Not permitted

Table 20–2: QUAD setting and resulting control unit output signal function

QUAD	Control unit output signal function			
	SMx1+	SMx1-	SMx2+	SMx2-
00	sine	VSS	cosine	VSS
01	sine	VSS	VSS	cosine
10	VSS	sine	VSS	cosine
11	VSS	sine	cosine	VSS

SMVSIN SMM Sine Register

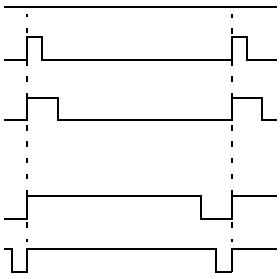
	7	6	5	4	3	2	1	0
r	x	x	x	x	x	x	x	BUSY
w	8bit Sine Value							
	0	0	0	0	0	0	0	0 Res

SMVCOS SMM Cosine Register

	7	6	5	4	3	2	1	0
w	8bit Cosine Value							
	0	0	0	0	0	0	0	0 Res

BUSY **Scheduler Busy Flag**
r0: Scheduler not busy
r1: Scheduler busy, do not write registers SMVC, SMVCOS, SMVSIN

Table 20–3: Usage of SMVSIN and SMVCOS registers

Value	Duty factor	Pulse Diagram
0x00	0/256 (continuously low)	
0x01	1/256	
0x02	2/256	
:	:	
0xFE	254/256	
0xFF	255/256 [1])	

[1]) 256/256 (continuously high) is not available.

20.3. Principle of Operation

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

20.3.1. Hardware settings

Prior to entering active mode, the f_{SM} input clock has to be set by HW option (see Table 20–4 on page 136). A frequency value of 4 MHz is recommended, resulting in a pulse width modulator cycle frequency of 4 MHz / 256.

Some H-Ports may receive the output signals either of the SMM module or of PWM modules as an alternative. Refer to Table 20–4 for the necessary settings.

Refer to section “HW Options” for details.

20.3.2. Initialization

Prior to entering active mode, proper SW initialization of the H-Ports assigned to function as H-bridge outputs SMxn+ and SMxn- has to be made (Table 20–4). The H-Ports have to be configured special out. Refer to “Ports” for details.

Table 20–4: Unit-specific settings

Contr. Unit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SMA			SMAn+/- outputs	H4.0 to H4.3 special out	SR0.SM
SMB			SMBn+/- outputs	H3.0 to H3.3 special out	
SMC			SMCn+/- outputs	H2.0 to H2.3 special out	
SMD			SMDn+/- outputs	H5.0 to H5.3 special out	
SME	SME/PWM selection	PM.H7	SMEn+/- outputs	H7.0 to H7.3 special out	
SMF			SMFn+/- outputs	H1.0 to H1.3 special out	
SMG	SMG/PWM selection	PM.H0	SMGn+/- outputs	H0.0 to H0.3 special out	
All	Input clock selection	SM			

20.3.3. Operation

After reset, the SMM is in standby mode (inactive). The output lines to the H-Ports are low.

For entering active mode, set bit SR0.SM. The FRC will immediately start counting but the control units’ output lines will still be low.

20.3.3.1. Generating Output

After entering active mode, the SMM’s control units are ready to receive sine, cosine and quadrant values.

First load the unit/quadrant information to register SMVC, then the cosine value to register SMVCOS and at last the sine value to register SMVSIN. Upon writing SMVSIN, the scheduler logic will set flag SMVSIN.BUSY and load the buffered values to the respective unit’s sine, cosine and quadrant registers on the next zero transition of the FRC, after a maximum of 256 f_{SM} input clock cycles. After completing the download, flag BUSY is reset and the respective unit will immediately start producing the output signals with the desired timing (see Table 20–3) on the proper pins (see Table 20–2).

The above procedure for loading values to a first unit is repeated for all others. Make sure that the BUSY flag is 0 before rewriting registers SMVC, SMVCOS and SMVSIN.

20.3.4. Inactivation

Returning the SMM module to standby mode by resetting bit SR0.SM will immediately halt the FRC, return all output signals to 0 and reset all internal registers.

20.4. Rotor Zero Position Detection (RZPD)

In addition to the above descriptions, this module supports the rotor zero position detection by supplying motor blockage information.

The rotor zero position detection capability is protected by a patent belonging to Siemens VDO Automotive (SV), and may only be used with SV’s prior approval.

20.4.1. Functional Description

Each control unit contains circuitry to detect an induced voltage resulting from the rotation of the connected motor’s rotor (Fig. 20–3). A comparator compares the input voltage from one of the unit’s H-Ports to 1/9th of the supply voltage. A capture logic opens a capture window and samples the comparator output. The capture result signal supplies a rotor blockage information necessary for the rotor zero position detection in all cases where the CPU has lost track of the display angle of a pointer that is driven by the motor via a mechanical transmission.

Furthermore, the rotor zero position detection clock has to be set by HW option (see Table 20–5 on page 137).

20.4.2. Registers

SMVCMP SMM Comparator Register

	7	6	5	4	3	2	1	0	
r/w	x	ACRF	ACRD	ACRB	ACRG	ACRE	ACRC	ACRA	
	x	0	0	0	0	0	0	0	Res

ACRA to G **Analog Comparator Control and Result for SMA to SMG**
r0: Capture result: no induced voltage detected
r1: Capture result: induced voltage detected
w0: Stop capture and clear result flag
w1: Start capture

SMVMUX **SMM Multiplex Register**

	7	6	5	4	3	2	1	0	
r/w	E		x	x	D		A		1
r/w	B		C		F		G		0
	0x0000								Res

A to G **Multiplexer for SMA to SMG**
r/w0 to 3: Select SMx-COMP input 0 to 3
Never switch a multiplexer during the detection process.

20.4.3. Principle of Operation

The RZPD can only be operated together with the stepper motor module. Switching SR0.SM connects/disconnects the comparators from supply and resets all registers.

During rotor zero position detection one of a unit's H-Ports (Table 20–4) has temporarily to be operated as input to an internal analog comparator. Reconfigure this port as special input and select this port by register SMVMUX.x. Refer to "Ports" for details.

Reading of the induced voltage at the measured motor winding is started by setting the questioned unit's control bit SMVCMP.ACRx to 1. The respective analog comparator's output will now be sampled. Once three consecutive '1' samples (spaced $1/f_{RZP}$) - indicating a sufficient analog comparator input voltage - are received, a 1 may be read from the questioned unit's result flag SMVCMP.ACRx, indicating that the rotor zero position detection is under way.

Resetting the questioned unit's control bit SMVCMP.ACRx to 0 stops the sampling and resets the result flag. When after a restart of the above sampling procedure and after a sufficiently long capture period still no 1 was read from the questioned unit's result flag SMVCMP.ACRx, this indicates that rotor zero position detection is complete.

Parallel rotor zero position detection on all control units is permitted.

After completion of rotor zero position detection, reconfigure the comparator input port as special out.

Table 20–5: RZPD-specific settings

Contr. Unit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SMA			SMA-COMP inputs	H4.0 to H4.3 special in	SR0.SM
SMB			SMB-COMP inputs	H3.0 to H3.3 special in	
SMC			SMC-COMP inputs	H2.0 to H2.3 special in	
SMD			SMD-COMP inputs	H5.0 to H5.3 special in	
SME			SME-COMP inputs	H7.0 to H7.3 special in	
SMF			SMF-COMP inputs	H1.0 to H1.3 special in	
SMG			SMG-COMP inputs	H0.0 to H0.3 special in	
All	Rotor Zero Position Detection Clock	RZPC			

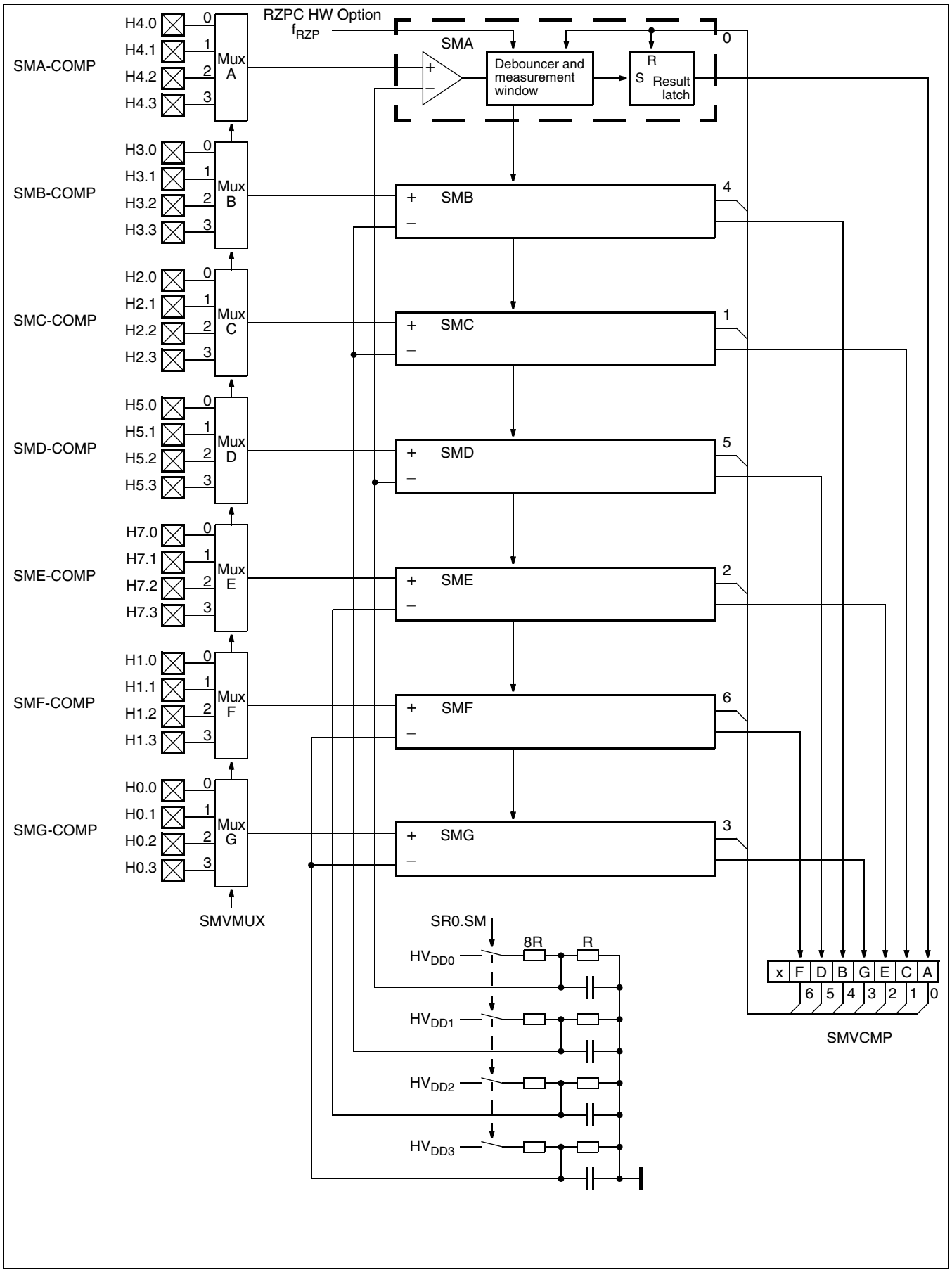


Fig. 20–3: Block diagram of rotor zero position detection circuit

21. LCD Module

The liquid crystal display (LCD) module is designed to directly drive a 1:4 multiplexed liquid crystal display. It generates all signals necessary to drive 4 backplane and 48 segment lines which are output via U-Ports in LCD mode. Up to 192 segments or pixels can be controlled if all U-Ports are designated as segment outputs.

In addition, the module provides functions that enable the user to cascade it with external expansion ICs providing more segment lines. It can be operated as master or slave in such an extended system.

Features

- 1:4 multiplex
- 5 V supply
- Maximum of 192 segments
- Cascadable with external expansion ICs
- 0.3 mA buffered 1/3 and 2/3 voltage divider
- Zero standby current
- 200 µA no load active current
- Frame frequency HW Option selectable

21.1. Principle of Operation

21.1.1. General

Each LCD pixel or segment which is controlled by the LCD module is located at the crossing point of a segment line and a backplane line. The LCD module co-ordinates the output sequences of backplane and segment lines (see Fig. 21–3 on page 141).

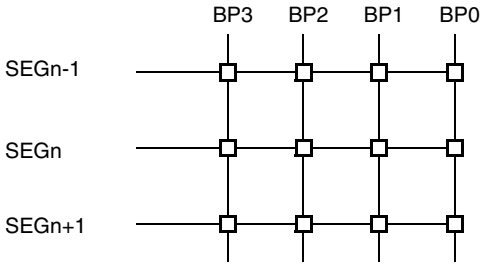


Fig. 21–1: Segments and backplanes

A segment pin can drive 4 different voltage levels (UVSS, 1/3 UVDD, 2/3 UVDD, UVDD) in LCD mode. The output of each segment pin is controlled by the corresponding segment bits of the registers UxD, UxTRI, UxNS and UxDPM (further called segment registers). Each such register contains one bit (of a 4 bit segment field) for each of its port pins. Each segment bit (0 to 3) of a segment field corresponds to a backplane line (BP0 to BP3). If the segment bit, corresponding with the backplane line BPx is true, then the segment at the crossing of the two lines is on (black).

The LCD module does not contain a display ROM translating character information into segment code. The advantage is that arbitrary characters or displays can be generated just by changing the program code. Segment information is directly entered by writing to the corresponding segment bit. It is validated (loaded to all corresponding slave registers) for all segment U-Ports simultaneously by a write access to register ULCDLD.

Two internal voltage sources provide the U-Port circuits and the backplane generator with the voltage levels 1/3 UVDD and 2/3 UVDD. These levels are generated by a buffered resistor divider.

21.1.2. Hardware settings

The LCD frame frequency is settable by HW option LC. The resulting frame frequency is the selected input frequency, divided by 120. It should be in the range from 50 to 200 Hz.

For best electromagnetic interference results it is recommended to operate all segment and backplane U-Ports in Port Slow mode. Refer to “Ports” for more details and to “HW-Options” for setting the corresponding HW options. Set flag PSLW in register SR0 to HIGH to enable Port Slow mode.

21.1.3. Initialization

After reset, the LCD module is in standby mode (inactive) and all U-Ports are in Port mode, non-conducting.

All U-Ports designated to function as backplane or segment outputs are to be set to LCD mode. Refer to “Ports” for more details. This will set these U-Ports to output LOW state.

After reset the content of the segment registers is undefined. It must be set by writing the desired segment information to the segment registers and by validating it by a write access to register ULCDLD (write 0x00 for master mode, 0xFF for slave mode), before the LCD module is enabled.

21.1.4. Operation

For entering active mode, set flag LCD in register SR0. Each segment and backplane U-Port will immediately start producing its LCD output signal according to the segment information provided during initialization.

During active mode, a new segment information is entered by simply writing the desired segment information to the segment registers and by validating it by a write access to register ULCDLD (write 0x00 while in master mode, 0xFF while in slave mode). Each segment and backplane U-Port will immediately start producing an LCD output signal according to the new segment information.

Returning the LCD module to standby mode by resetting flag LCD in register SR0 will immediately return all segment and backplane U-Ports to the output LOW state.

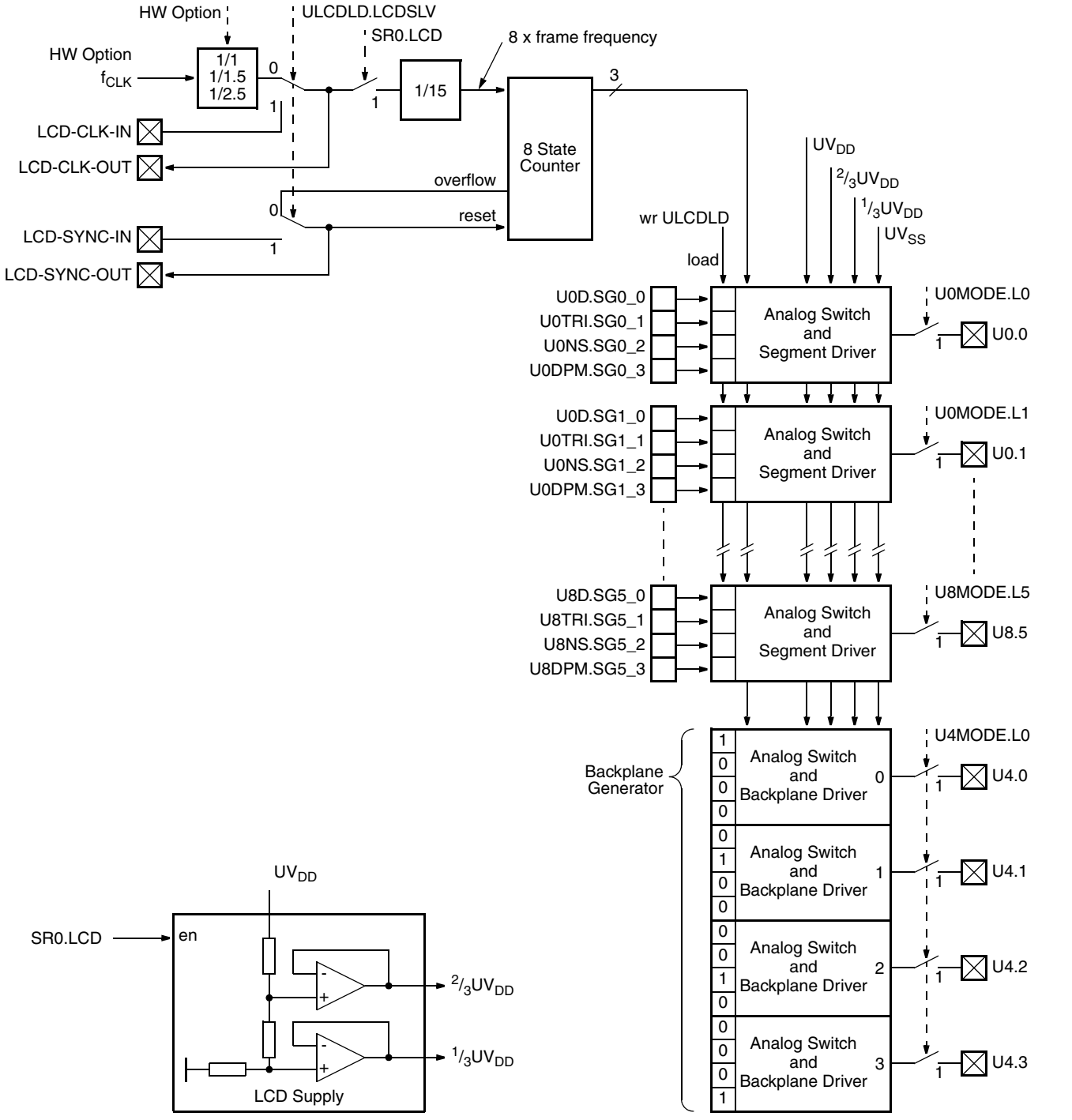


Fig. 21–2: Block diagram

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41). During power-saving mode the LCD logic is switched off, the port outputs are held at low level but the port register contents will be preserved.

21.1.5. Cascading of LCD Driver Modules

For expansion purposes, the LCD module may be cascaded with external LCD driver ICs. Master or slave mode is selectable for the LCD module while in standby. Special signals provide phase and frequency synchronism for the LCD frame among the cascaded ICs.

For master mode, set flag LCDSLV in register ULCDLD LOW. The module always directs signal LCD-SYNC-OUT to pins U8.5 and LCD-CLK-OUT to pins U8.3. They connect to external slave ICs’ SYNC-IN and CLK-IN inputs for synchronization.

For slave mode, set flag LCDSLV in register ULCDLD HIGH. Configure pins U8.4 and U8.2 to receive signals LCD-SYNC-IN and LCD-CLK-IN from an external master IC’s SYNC-

OUT and CLK-OUT outputs. These signals will then substitute the LCD module's own HW option frame frequency settings.

Starting up and shutting down such an expanded system is described in section 21.3.

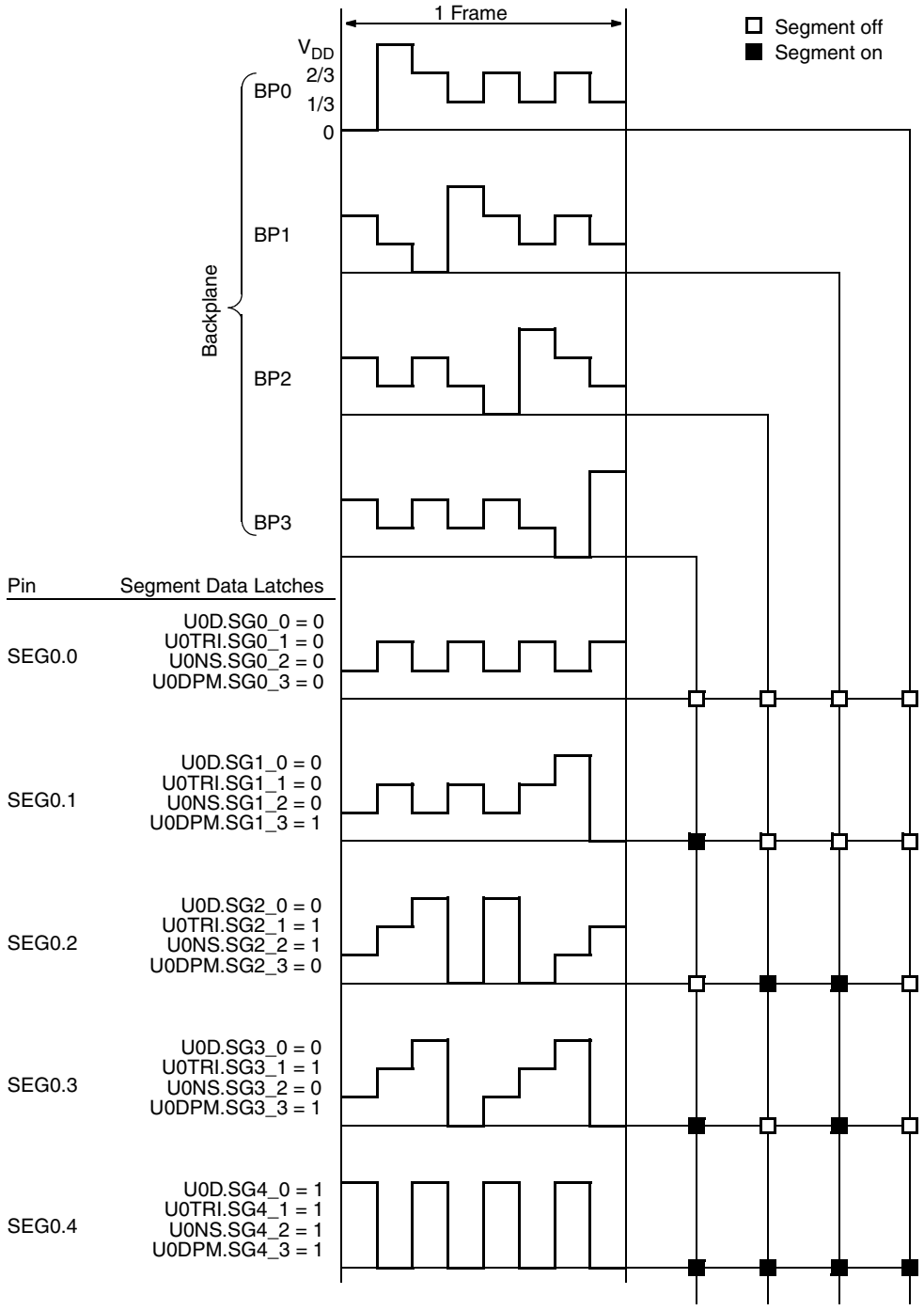


Fig. 21–3: Frame timing diagram

A segment at a crossing of backplane and segment lines turns black when the backplane driver outputs a full swing and the segment driver outputs a full swing of opposite polarity at the same time.

21.2. Registers

Please refer to section “Universal Port Registers” for details on segment register layout.

ULCDLD	Universal Port LCD Load Register							
	7	6	5	4	3	2	1	0
w	LCDSLV	x	x	x	x	x	x	x
	0	0	0	0	0	0	0	0 Res

LCDSLV LCD Module is Slave
Select the mode of the LCD module.
w1: LCD module is slave.
w0: LCD module is master.

A write access to this memory location simultaneously loads all segment information of all U-Ports in LCD mode to the display.

21.2.1. Special Register Layout of U-Port 4

U4.0 to U4.3 provide backplane signals in LCD Mode. To operate any ports as LCD segment driver it is necessary to switch all these ports to LCD mode. This has to be done by setting flags U4MODE.L0 through U4MODE.L3.

As backplane ports U4.0 to U4.3 require no segment data setting, SG0_0 through SG3_3 bits are not available in U4 registers.

21.3. Application Hints for Cascading LCD Modules

21.3.1. Power On and Start Up Procedure

1. The SW in master and slave configures the corresponding IC.

Table 21–1: Suggested sequence

Master	Slave
Load LCD display register.	Load LCD display register.
Clear flag LCDSLV.	Set flag LCDSLV.
LCD-CLK-OUT, and LCD-SYNC-OUT: Configure universal ports as Special Out Ports.	LCD-CLK-IN, and LCD-SYNC-IN: Configure universal ports as Special In Ports.

2. Optionally the slave signals to the master via handshake link or an inter-processor interface (IPI) that it is ready to display.

3. The slave continuously scans the inputs LCD-CLK-IN and LCD-SYNC-IN for the bit combination “01” (SW debouncing required).

4. The master LCD module is switched on. LCD-CLK-OUT and LCD-SYNC-OUT switch to “01”.

5. The slave CPU detects the bit combination “01” and immediately switches on the slave LCD module. The slave LCD now generates a display.

Note: During the time that the slave needs to detect the bit combination “01”, master and slave operate asynchronously. Suggestion: limit time to approximately 100 to 200 ms.

6. The LCD modules now operate in controlled synchronization.

21.3.2. Operation

In order to obtain optimum synchronization of LCD switchover, a change of display must be coordinated between master and slave (preferably via IPI) in such a way that the time lag between write accesses to ULCDLD of the master and of the slave is kept as small as possible. Suggestion: Lower ms range or customer specification.

21.3.3. Power Off Procedure

1. (Optional) The processor which decides that the display is to be switched off signals this to the other via IPI.

2. The slave continuously scans the inputs LCD-CLK-IN and LCD-SYNC-IN for the bit combination “11” (SW debouncing required).

3. The master LCD module is switched off. LCD-CLK-OUT and LCD-SYNC-OUT switch to “11”.

4. The slave CPU detects the bit combination “11” and immediately switches off the slave LCD module.

Note: Keep time delay as short as when switching on.

5. All LCD ports output a low signal now. The LCD display is now inactive.

22. DMA Controller

The DMA controller allows transferring data fields between internal memory and either an external IC via U-Ports (G-Bus), or an SPI module, with minimum CPU interaction.

DMA transfers can be triggered by the interrupt source output of the corresponding module (self-timed), a dedicated DMA timer output or a port interrupt.

The G-Bus is intended to support the operation of external LCD driver ICs (e.g., SED1560 by Epson):
The DMA module copies 8-bit pixel data bytes by direct memory access (DMA) to the external IC's graphic RAM with help of that IC's internal autoincrement address counter, and without CPU interaction. Other off-chip registers, allowing control of the display behavior (blinking, scrolling, etc.), have to be addressed by CPU operations.

In SPI mode, the DMA module copies data bytes by direct memory access (DMA) to the SPIxD data register, self-timed or under timing of the DMA timer and without CPU interaction, to construct long serial data transfer sequences. It frees the CPU of repeatedly reloading data, e.g. under interrupt control.

Features

- 3 DMA channels:
  direct 8-bit data read or write between memory and U-Ports U5 and U7 (G-Bus),
  direct 8-bit data read or write between memory and SPI0,
  direct 8-bit data read or write between memory and SPI1
- 256 byte maximum DMA block size
- one byte DMA block alignment
- CPU cycle steal
- Interrupt on DMA sequence finished

22.1. Functions

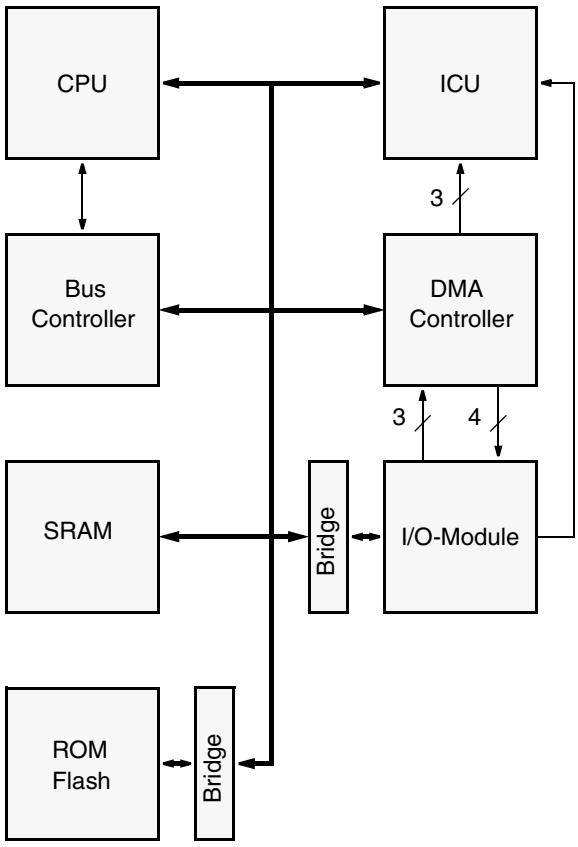


Fig. 22–1: System block diagram

The DMA Controller transfers bytes (8 bit) between I/O modules and memory. One transfer is called a DMA cycle. The transfer of a block of bytes is called a DMA sequence.

The DMA Controller contains one DMA channel logic for each DMA channel, the priority encoder, the control logic, the DMA vector base register, address and cycle count buffer, and the bus interface (see Fig. 22–2 on page 144).

The DMA vector base register points to the beginning of the DMA table which is filled with a DMA vector for each DMA channel. Location zero contains the default vector and is not assigned to any DMA channel. Each DMA vector is composed of a 24-bit source/destination address and an 8-bit cycle counter value (see Fig. 22–5 on page 145).

A DMA cycle is divided into a sequence of three steps:

1. Output Address of the DMA vector and read source/destination address and cycle counter.

2. Output Address of the DMA vector and write back incremented source/destination address and decremented cycle counter.

3. Output source/destination address and write/read data to/from I/O module.

Each step is one bus access which holds the CPU (cycle stealing). The DMA controller generates the necessary control signals for above bus accesses.

An I/O module requests a DMA cycle via its interrupt source output which is connected to the DMA request input (DREQ) of the corresponding DMA channel logic. The DMA interrupt output (DINT) is connected to the ICU instead, where it indicates the end of a DMA sequence (see Fig. 22–3 on page 145). The signal DINT is connected to the G-Bus logic too, where it sets a flag indicating the end of the DMA sequence (see Fig. 22–4 on page 145).

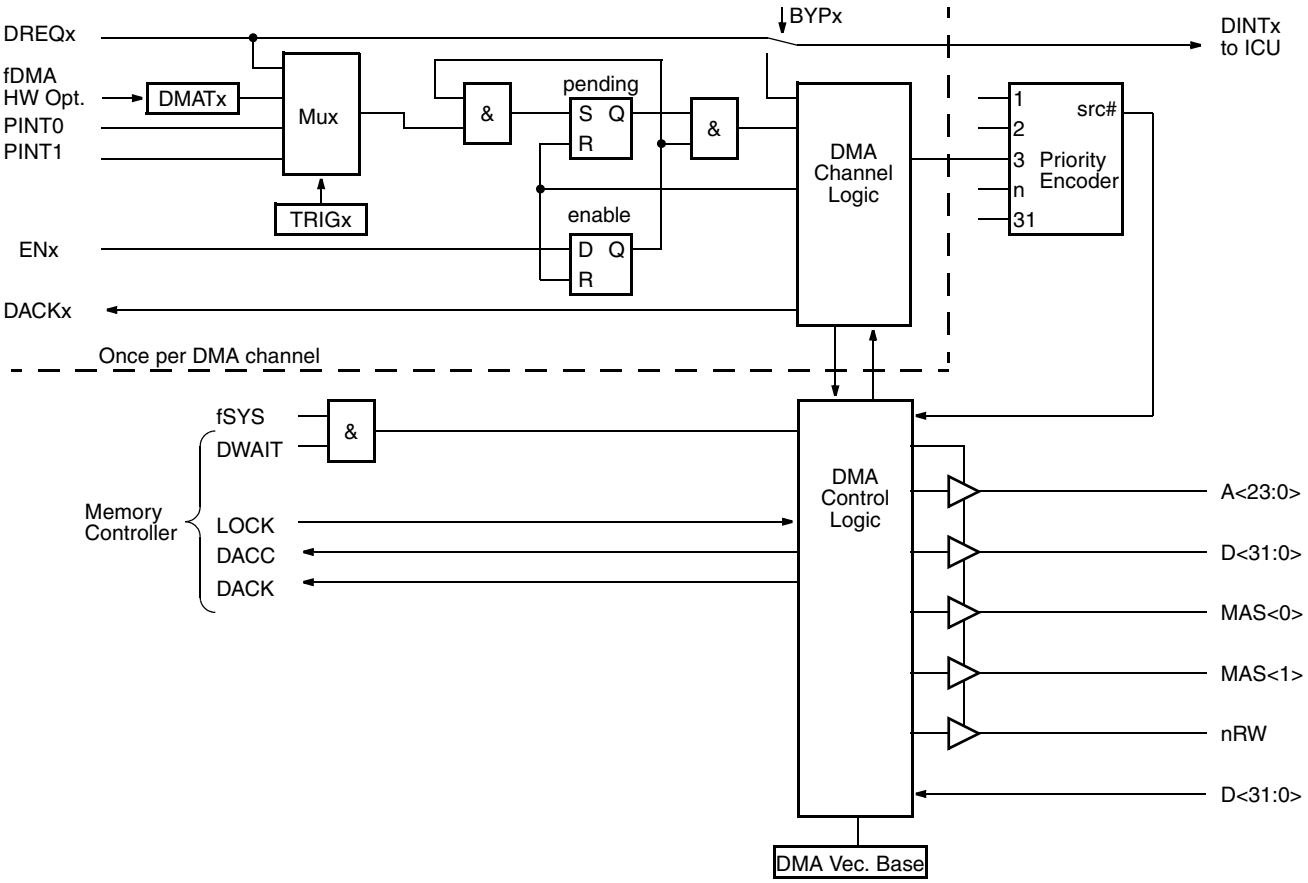


Fig. 22–2: DMA controller

The DMA channel logic contains an input multiplexer which selects one of four possible DMA request sources (see Table 22–2 on page 146). The output of this multiplexer sets a pending flag which is automatically reset when the DMA cycle is finished. An enable flag (EN) masks the pending flag output to the priority encoder. A bypass flag (BYP) allows to redirect DREQ to DINT and thus generate no DMA request but an interrupt.

The priority encoder assigns each DMA channel a fixed unique priority (Table 22–1). This is necessary when more than one DMA channel signals a DMA request at the same time. The priority encoder outputs the source number with the highest priority.

The control logic controls the above described three steps of bus accesses and generates the DMA acknowledge signal (DACKx) which indicates to the requesting module that the DMA transfer has finished.

Table 22–1: DMA channels

Priority (=Channel No.)	**I/O-Module**	**Register Name**
0	Default	
1	U-Port	GD
2	SPI 0	SPI0D
3	SPI 1	SPI1D

There are two fundamentally different modes to operate DMA sequences.

– Self-timed mode describes the situation where the corresponding I/O module requests a DMA transfer when it is ready. In this case the I/O module starts the DMA module when there is something to transfer and the DMA controller starts the I/O module after the transfer is finished. This is the fastest possible way to transfer information via DMA. Trying to get it faster enforces the danger that one of the communication partners is not ready.

– Externally triggered mode describes the situation where a third party requests a DMA transfer. This can be a DMA timer or a port interrupt. The SW design has to guarantee that the I/O module as well as the DMA controller can do their work between two consecutive DMA requests.

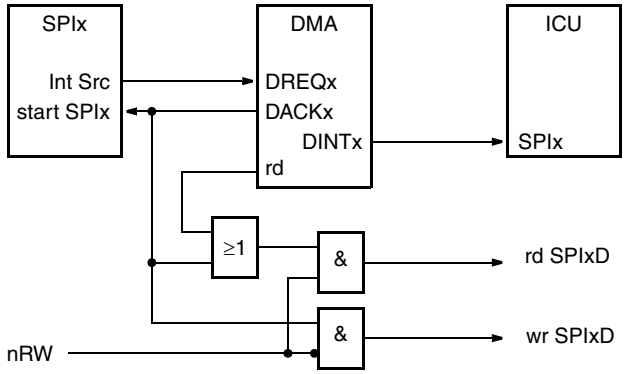


Fig. 22–3: DMA SPI interaction

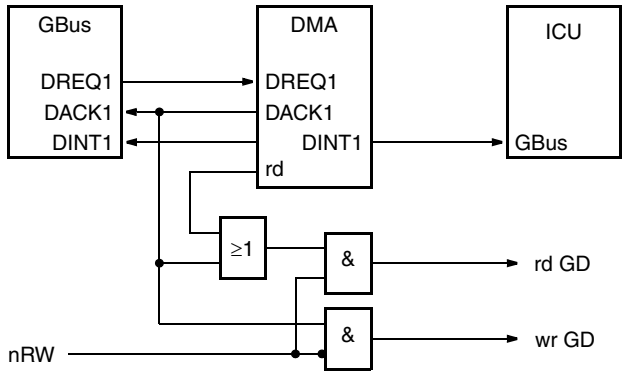


Fig. 22–4: DMA port interaction

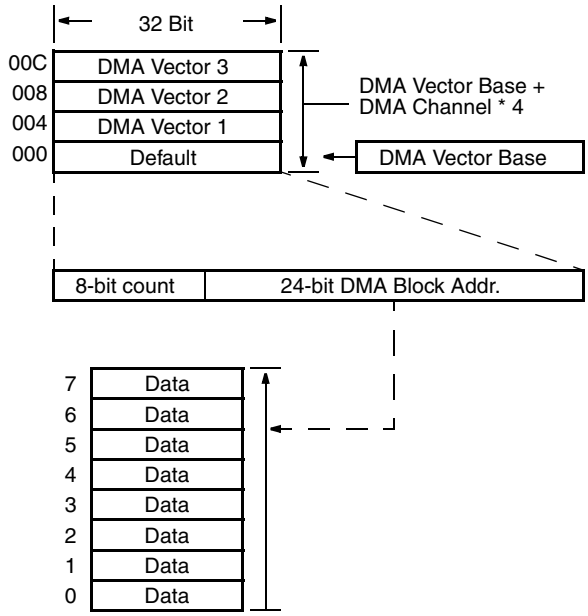


Fig. 22–5: DMA vector table

22.2. Registers

The DMA registers can be read or written 32-bit wide, asynchronous and without wait states.

DVB DMA Vector Base

	7	6	5	4	3	2	1	0	Offs
r/w	0	0	0	0	0	0	0	0	3
r/w	A23 to A16								2
r/w	A15 to A8								1
r/w	A7	0	0	0	0	0	0	0	0
	0x0000								Res

DST DMA Status Register

	7	6	5	4	3	2	1	0	Offs
r/w	DE	x	x	SRC					0
	0x00								Res

DE **DMA Enable**
r/w1: enable DMA controller
r/w0: disable DMA controller
Enables the DMA controller clock (fSYS) and the clock for all DMA Timer (fDMA). Before setting to 0, make sure that all individual DMA channels are terminated.

SRC **Priority source output**
r31-0: The number of the highest pending and enabled DMA request.

DCxM DMA Channel x Mode Register

	7	6	5	4	3	2	1	0	Offs
r/w	P	DMAT			TRIG				1
r/w	EN	x	x	x	BYP	DIR	MAS		0
	0x0000								Res

P **DMA Pending**
r1: DMA transfer pending
r0: No DMA transfer pending
w1: No action
w0: Clear P

DMAT **DMA Timer**
r/w7-0: DMA timing, equation:

$$f_{DMAT} = \frac{f_{DMA}}{2^{DMAT+1}}$$

TRIG **Trigger Source**
r/w15-0: (see Table 22–2)

Table 22–2: DMA trigger sources

TRIG				Source
3	2	1	0	
x	x	0	0	DMA request from I/O-module
x	x	0	1	DMATx
x	x	1	0	PINT0
x	x	1	1	PINT1

EN **Enable DMA channel**
r1: DMA sequence active
r0: DMA sequence finished
w1: enable DMA channel
w0: disable DMA channel

BYP **Bypass Interrupt**
r/w1: don't bypass DMA-Request to ICU
r/w0: bypass DMA-Request to ICU.

DIR **DMA Direction**
r/w1: write to I/O-module
r/w0: read from I/O-module

MAS **Memory Access Size**
r/w3: reserved
r/w2: 32-bit (not supported)
r/w1: 16-bit (not supported)
r/w0: 8-bit

22.3. Principle of Operation

The DMA Controller is operable in all CPU modes. However, it has to be disabled (see 22.3.9.) during CPU mode switching to prevent undefined clock system behavior. For correct CPU mode switching, follow the sequence given section 4.2.

22.3.1. Initialization of the DMA Controller

The DMA vector table has to be installed starting at a 128 byte aligned address. See figure 22–5 for DMA vector layout. Write the start address of the DMA vector table as a 32 bit address to the DMA Vector Base register (DVB) and note that only bits 7 to 23 may be modified. The other bits are forced to zero. Enable the DMA controller by setting flag DE in the DMA Status register (DST).

The input frequency fDMA for all DMA timer can be selected by the register DMAC in the HW Options field.

22.3.2. Initialization of a DMA Channel

All steps necessary to initialize the involved I/O module have to be taken according to the description in the respective chapter.

Write the appropriate values to the DMA Channel Mode register (DCxM). Select the trigger source by field TRIG, program the DMA timer by field DMAT if necessary, select transfer direction (DIR) and size (MAS) and set BYP to one.

22.3.3. Self-Timed DMA Write to I/O Operation

Flag DIR in register DCxM must contain a one for writing to an I/O module.

Write the source address (24 bit), pointing to the first plus one element, and the block size (8 bit) to the corresponding DMA vector table entry.

Start the DMA sequence by writing the first element to be transferred to the data register of the corresponding I/O module and enable the DMA channel.

22.3.4. Self-Timed DMA Read from I/O Operation

Flag DIR in register DCxM must contain a zero for reading from an I/O module.

Write the destination address (24 bit), pointing to the first element, and the block size (8 bit) to the corresponding DMA vector table entry.

Start an SPI DMA sequence by writing to register SPIxD of the corresponding SPI module. The data of this write may be omitted. Then enable the DMA channel.

Start a graphic bus DMA sequence by reading from register GD. The data of this read may be omitted. Then enable the DMA channel.

22.3.5. Externally Triggered DMA Operation

The procedure is the same as with the self-timed operation with some distinctions.

In both cases (read/write) the SW initiates the first action in the peripheral module. Enable the DMA channel after this module has finished its work. Otherwise a DMA cycle may happen too early, transferring invalid data.

It is possible to do externally triggered DMA transfers without the SW initiating the first action. In this case, the maximum block size is limited to 255 byte because the count value in the DMA vector has to be programmed with block size plus one. In a write case (Fig. 22–7), the sequence starts with data D1. In a read case (Fig. 22–8), the first DMA cycle reads invalid data D0. The first element of the transferred block has to be omitted in the latter case.

22.3.6. End of DMA Sequence

The end of the DMA sequence is indicated by the enable flag (EN=0) and an interrupt which calls the ISR of the corresponding I/O module.

The address field of the corresponding DMA vector points to the next element after the last transferred element. The counter field is at zero.

22.3.7. Enabling of a DMA Channel

Setting the flag EN to one enables the DMA channel. Make sure that there is no pending DMA request at that point of time. Clearing an active pending flag P and enabling the corresponding DMA channel must not be done with a single instruction. This might lead to an unwanted DMA cycle. First clear P and then set EN in two instructions.

22.3.8. Stalling and Termination of a DMA Sequence

Clearing the flag DCxM.EN stalls a DMA sequence. Further DMA requests of the disabled DMA channel set the pending flag DCxM.P. Setting the flag DCxM.EN again enables the DMA channel thus continuing the DMA sequence. This mechanism has to be operated carefully because data may be lost during a stall period.

A final termination of a DMA sequence can be achieved by first disabling the DMA channel (DCxM.EN=0) and the source of the DMA requests and secondly clearing the pending flag (DCxM.P=0). Do not forget to clear DCxM.BYP if further DMA requests are to generate an interrupt.

22.3.9. Disabling the DMA Controller

First terminate all DMA channels (see 22.3.8.) and then clear flag DST.DE. Do not forget to clear DCxM.BYP if further DMA requests are to generate an interrupt.

22.4. Timing Diagrams

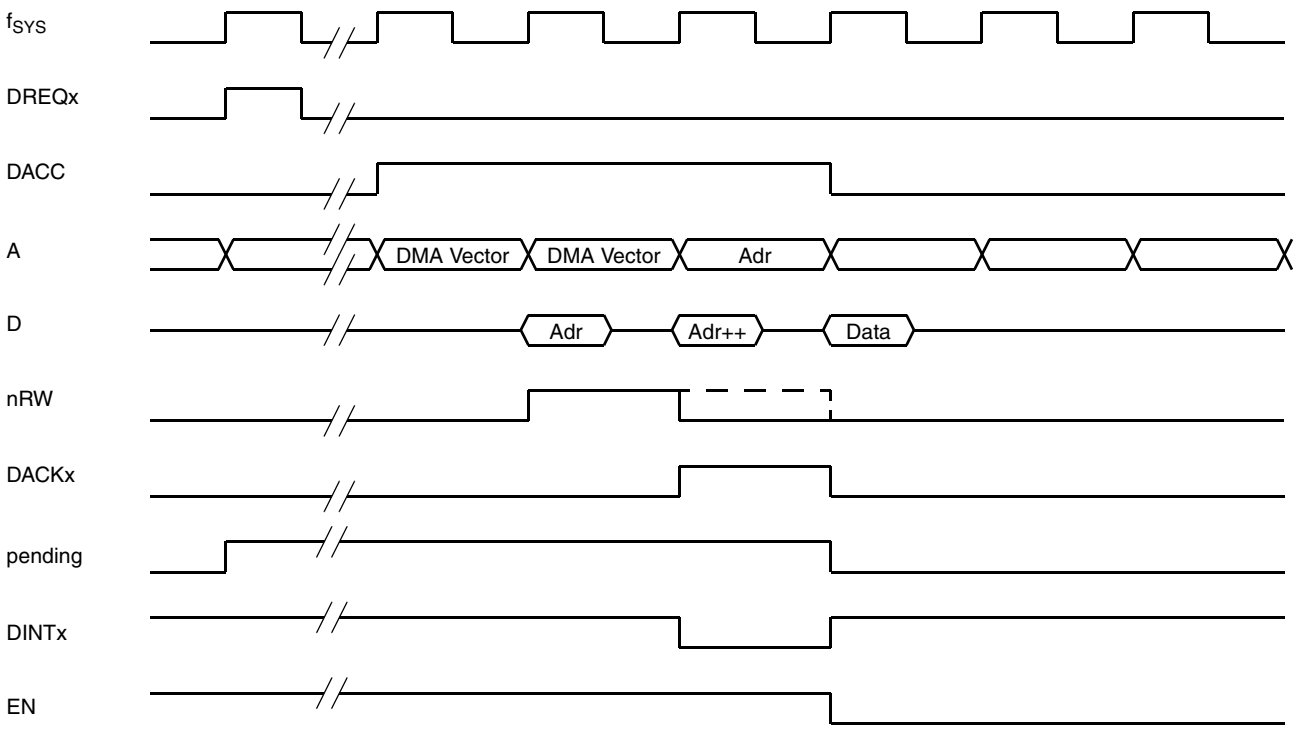


Fig. 22–6: DMA cycle timing

22.4.1. DMA Sequences SPI

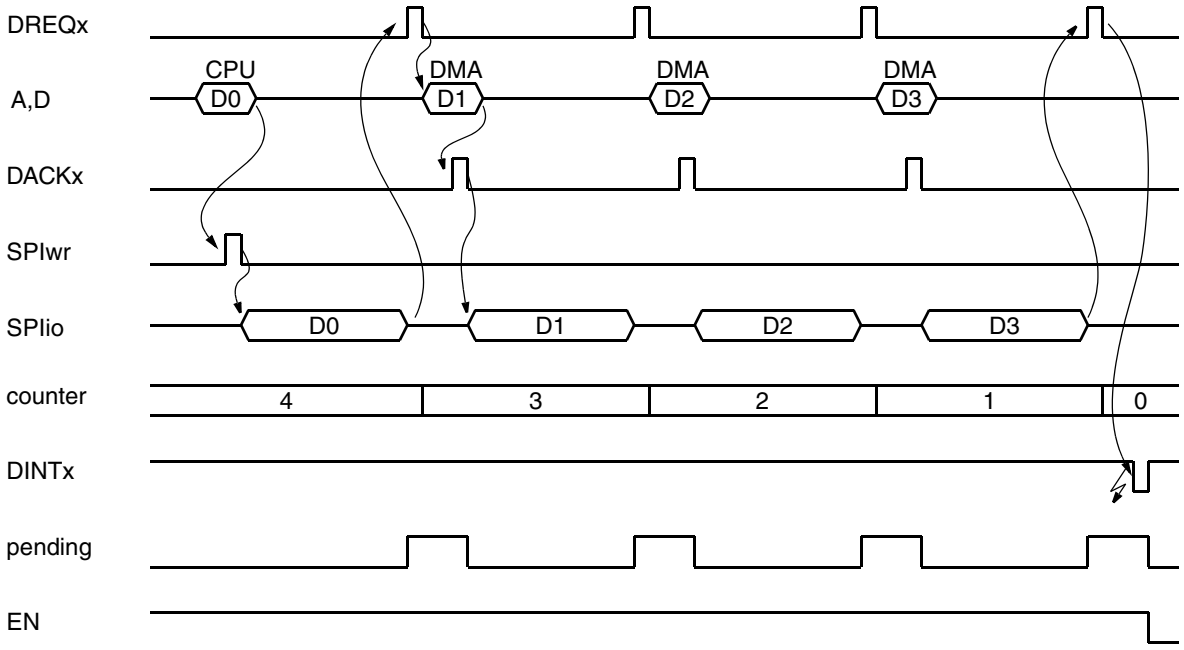


Fig. 22–7: SPI write sequence (serial out)

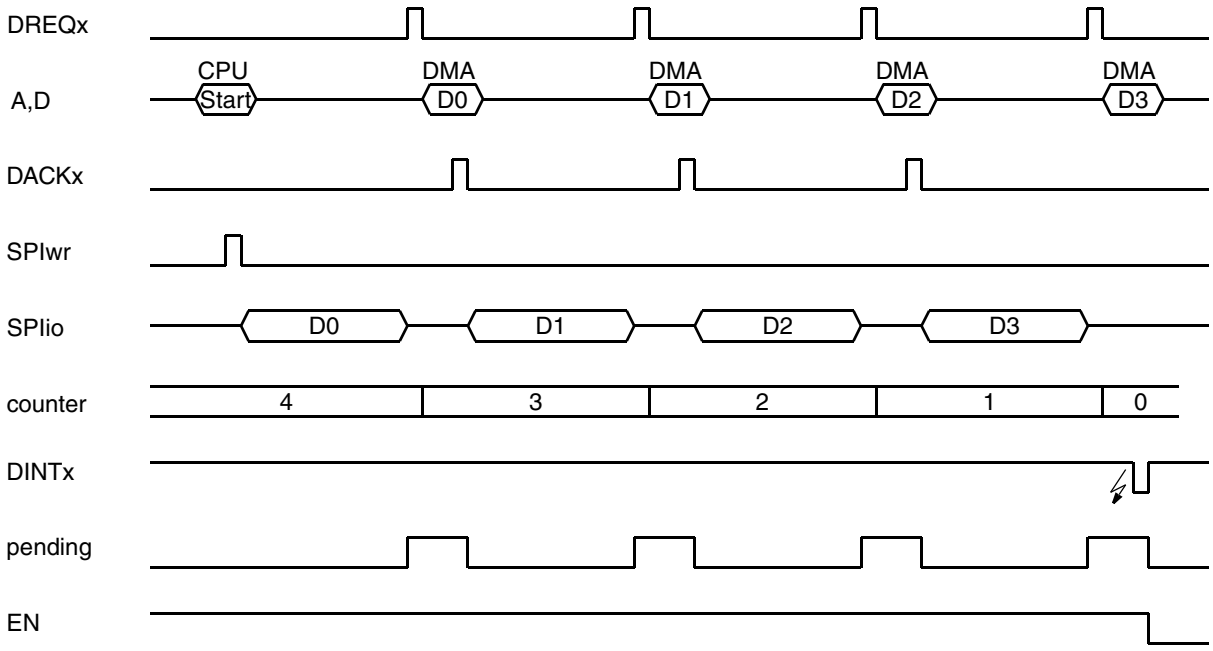


Fig. 22–8: SPI read sequence (serial in)

22.4.2. DMA Sequences Graphic Bus Interface

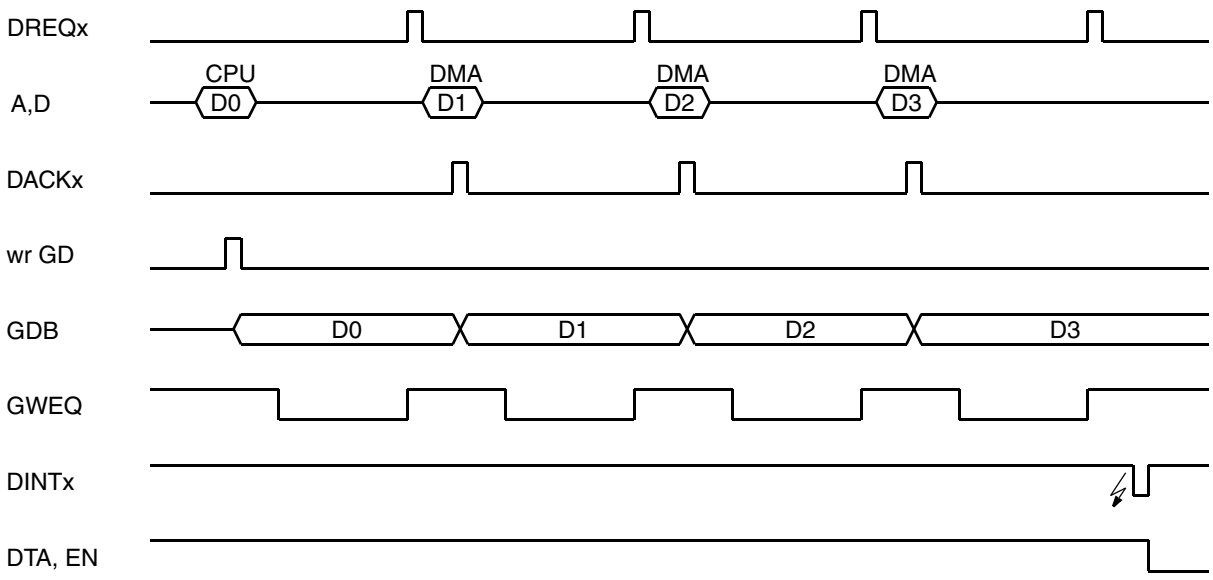


Fig. 22–9: Graphic bus write sequence (parallel out)

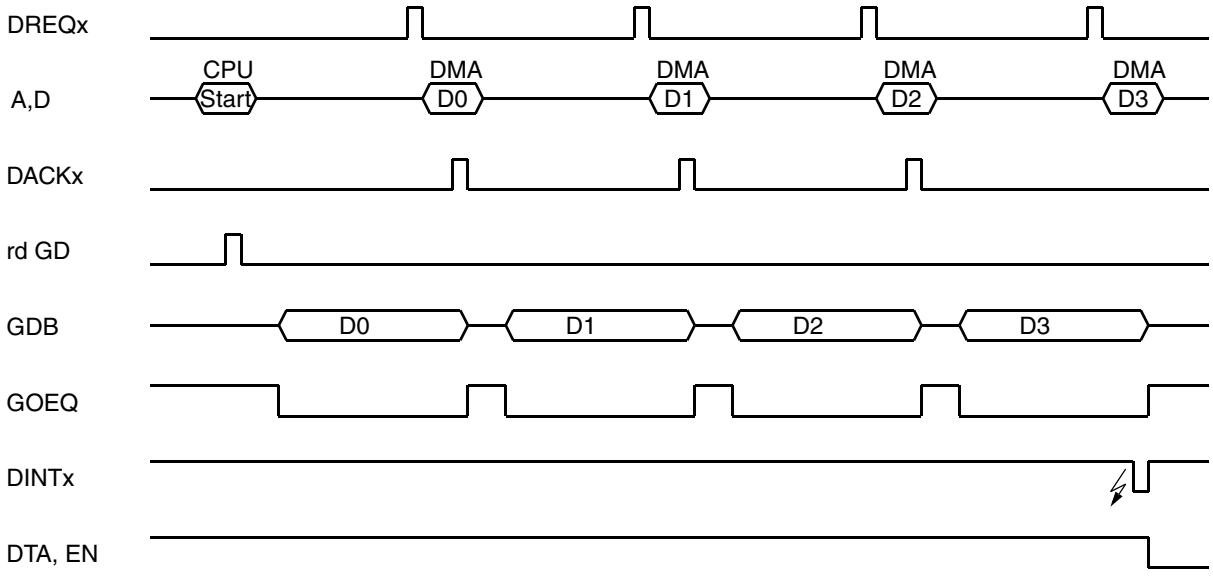


Fig. 22–10: Graphic bus read sequence (parallel in)

The final DMA request pulse clears the DMA Transfer Active (DTA) flag, in addition to generating an interrupt.

23. Graphic Bus Interface

The graphic bus interface (GB) is intended to support the operation of external LCD driver ICs (e.g., SED1560 by Epson).

Features

- DMA read/write to external device
- CPU read/write to external device
- Read/write timing generation
- Read/write control signals generation

23.1. Functions

The DMA module copies 8-bit pixel data bytes by direct memory access (DMA) to the external IC's graphic RAM with help of that IC's internal autoincrement address counter, and without CPU interaction. Other off-chip registers, allowing control of the display behavior (blinking, scrolling, etc.), have to be written and read by CPU operations.

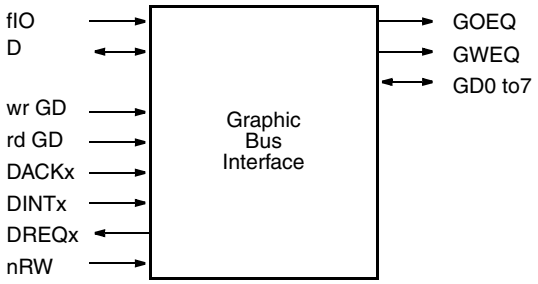


Fig. 23–1: Port bus block diagram

The necessary timing is done autonomously by the GB logic. Any U-Port may be used as address output port operated by the CPU. Please refer to section DMA for information on GB DMA interaction.

The register GD provides the data interface for the GB. Writing to GD outputs the data byte at U5.0 to U5.3 (low nibble) and U7.4 to U7.7 (high nibble). Reading from GD inputs a data byte from above pins. The assignment to external signals is shown in table 23–1.

Table 23–1: Port assignment

Port	Name	
U5.0	GDB0	External data bus
:	:	
U7.7	GDB7	
1)	GADB	External address bus
U6.2	GWEQ	External write signal
U6.1	GOEQ	External read signal
1) Any U-Port may be used as address output port.		

23.2. GB Registers

GD Graphic Bus Data Register

	7	6	5	4	3	2	1	0	Offs
r/w	Data								0
	0x00								Res

A write access to this register generates the DACK signal and writes to registers UxD. A read access to this register generates the DACK signal and reads from registers UxPIN.

GC Graphic Bus Control Register

	7	6	5	4	3	2	1	0	Offs
r/w	TIM				E	BSY	SEQ	DTA	0
	0x00								Res

TIM **GB Timer**

w15-1: GB timing, equation:

$$t_{GB} = \frac{2^{TIM+1}}{f_{IO}}$$

w0: GB logic is disabled, clock input is disabled

E **Enable**
r/w1: Enable timing generation
r/w0: Disable timing generation

BSY **Busy**
r1: GB timing is active
r0: GB timing is not active
Every DACKx signal or access to GD sets this flag and every DREQx signal clears it again.

SEQ **DMA Sequence**
r1: DMA sequence is active
r0: DMA sequence is not active
Every DACKx or access to GD signal sets this flag and the DINTx signal clears it again.

DTA **DMA Transfer Active**
r1: DMA sequence started
r0: DMA sequence is finished
w1: Set DTA
w0: No action
This flag indicates the end of a DMA sequence. It has to be set by SW before a DMA sequence is started. It is cleared by signal DINTx.

23.3. Principle of Operation

23.3.1. Initialization

Table 23–2 shows the necessary settings of the port configuration registers.

Table 23–2: Port configurations

Register	Setting	Mode
U5MODE, U7MODE, U6MODE	0x00	Port mode
U5NS, U7NS	0x00	Normal
U6NS	0x06	Special
U5TRI, U7TRI, U6TRI	0x00	Out

Enable the timing generation by setting flag E in register GC. Enable the clock input and select the desired timing of the control signals GOEQ and GWEQ in the field GC.TIM. The minimum high time of the control signals is one fIO cycle.

23.3.2. Data transfer

Data to/from an external device can be transferred directly by CPU access or, especially for bigger amounts of data and with help of the external device's autoincrement address counter, a DMA sequence can be started. Make sure not to start a GB transfer unless the flags DTA, SEQ and BSY are zero.

23.3.2.1. DMA Write Sequence

After initialization of the corresponding DMA channel, set flag DTA to show others that a DMA sequence was initiated but not finished and write the first value to be transferred via the GB to the register GD. The DMA Controller writes the remaining bytes to register GD and generates an interrupt when finished. DTA low marks the end of the DMA sequence.

23.3.2.2. DMA Read Sequence

After initialization of the corresponding DMA channel, set flag DTA to show others that a DMA sequence was initiated but not finished and read the register GD. The DMA Controller reads the remaining bytes from register GD and generates an interrupt when finished. DTA low marks the end of the DMA sequence.

23.3.2.3. CPU Write Access

Writing the byte to register GD is sufficient. The end of the transfer is indicated by flag BSY.

23.3.2.4. CPU Read Access

The read access must be initiated by a dummy read access to register GD. After BSY is low the desired byte can be read from register GD. This last step automatically initiates the next read timing of the GB logic. If this is not desired, because GOEQ stays active until the next access to GD, after BSY becomes low, first disable the GB timing generation by clearing flag E in register GC and then read register GD.

23.3.3. Inactivation

Inactivation is easily done by writing GC.TIM to zero. Make sure not to switch off the GB as long as a transfer is active (DTA or SEQ or BUSY are set).

23.3.4. Precautions

A write to register GD alters the universal ports data latches U5D and U7D even if the GB is disabled (GC.TIM = 0).

23.3.5. Timings

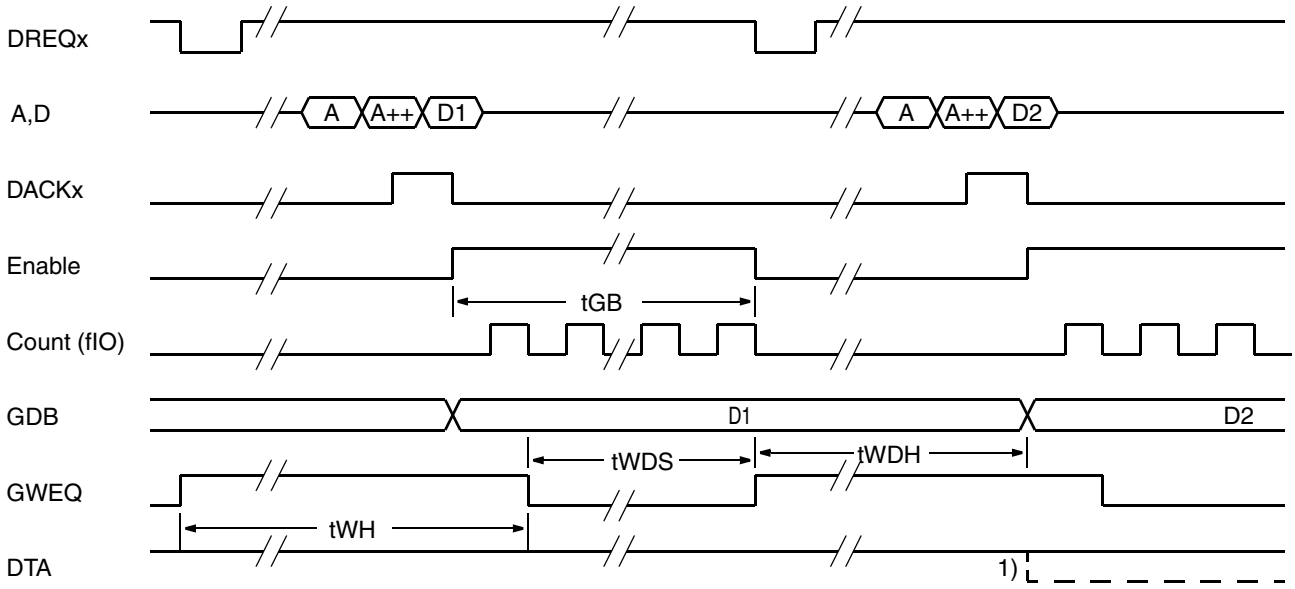


Fig. 23–2: DMA write (parallel out)

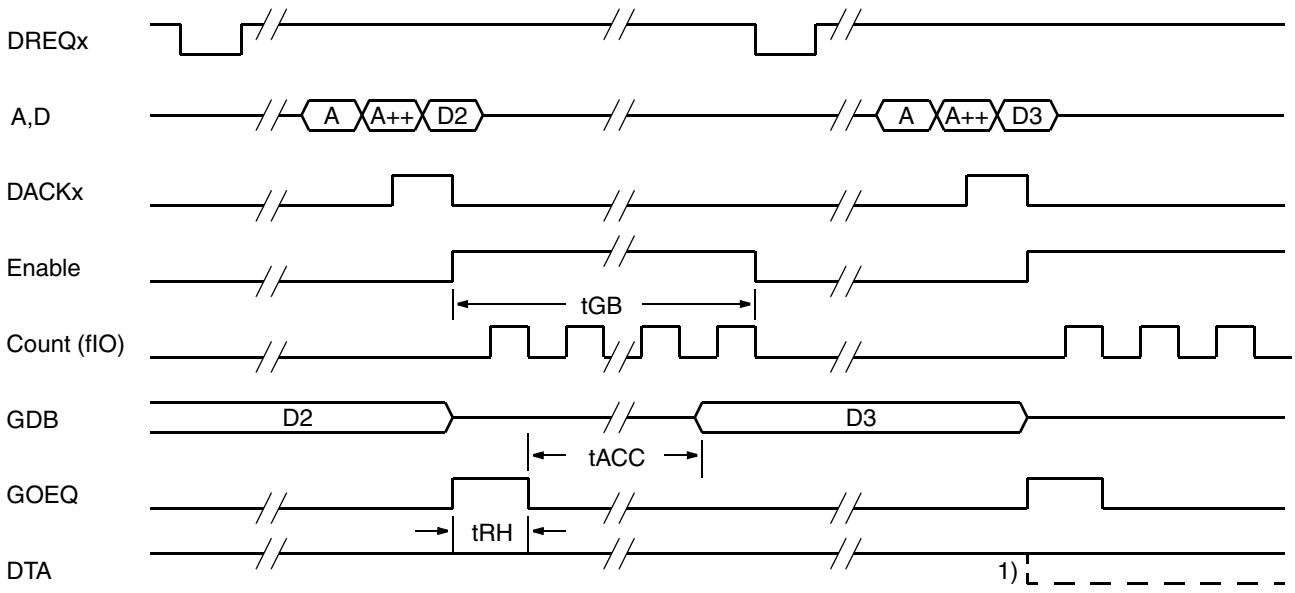


Fig. 23–3: DMA read (parallel in)

1) DTA at the end of the last DMA cycle.

tWDS: Write data setup time
tWDH: Write data hold time
tWH: DMA write high time
tRH: DMA read high time
tACC: Read access time
tGB: GB time

DACKx can be replaced by write to GD or read from GD if direct CPU access is desired. The signals “enable” and “count” are internal signals.

24. Serial Synchronous Peripheral Interface (SPI)

A SPI module provides a serial input and output link to external hardware. An eight or nine bit data frame can be transmitted in synchronism to an internally or externally generated clock.

The SPI module can be operated via direct access or via DMA.

The number of SPIs implemented is given in Table 24–1. The “x” in register names distinguishes the module number.

Features

- 8 or 9-bit frames
- Internal or external clock
- Programmable data valid edge
- Programmable clock polarity
- Three internal clock sources programmable
- Input deglitcher for clock and data
- DMA interface

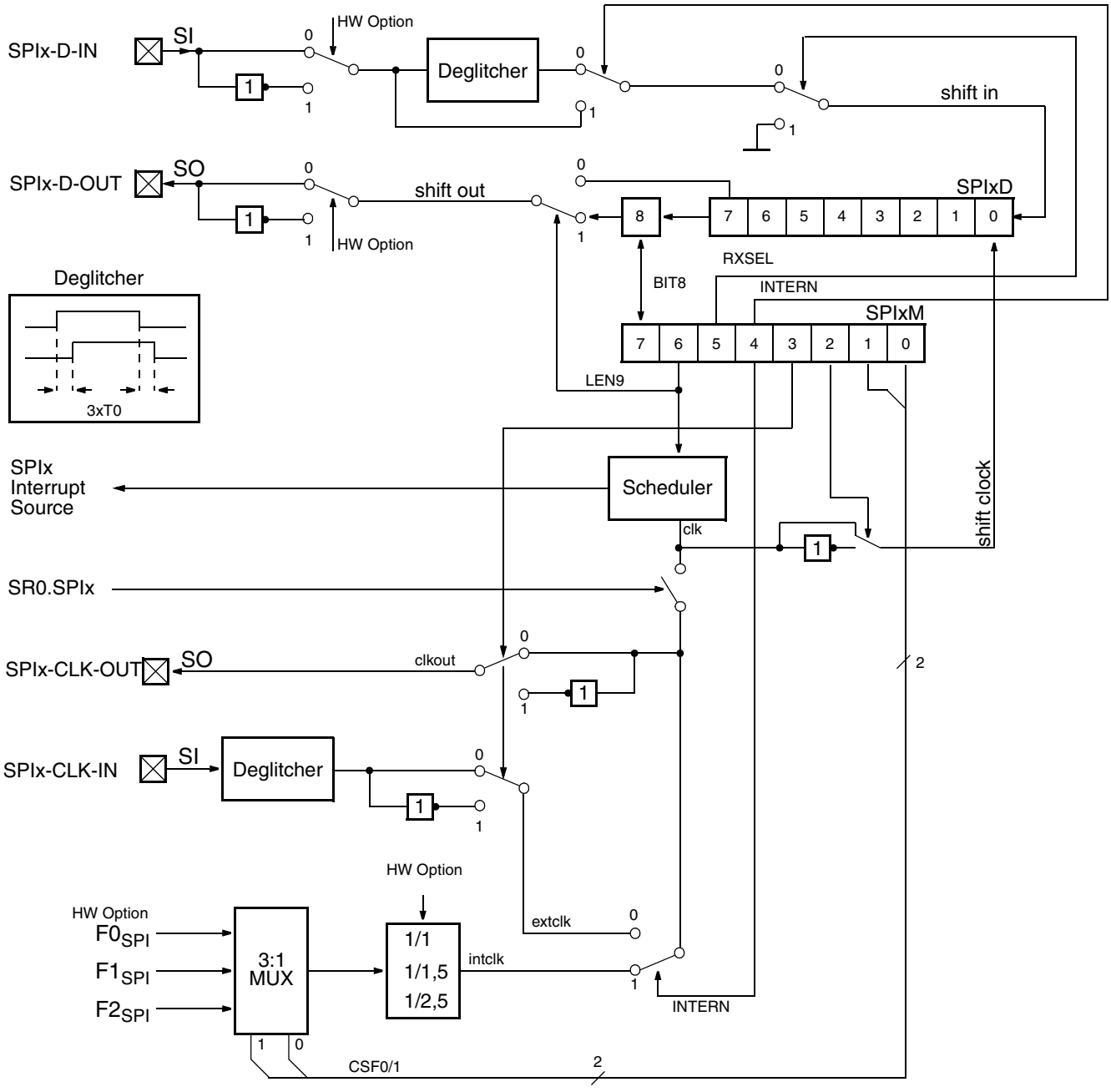


Fig. 24–1: Block diagram

24.1. Principle of Operation

24.1.1. General

An SPI serves as an 8 or 9-bit wide input/output shift register. Either an internally or an externally generated clock can be used to shift data in and out.

The input SPIx-D-IN is connected to the LSB of the shift register. The output of the shift register is connected to output signal SPIx-D-OUT. Thus each time a frame is transmitted by shifting bits out, bits are shifted in simultaneously and vice versa. Deglitchers in the data and clock input paths are active only in external clock mode. The input and output can be inverted by HW Option.

If the deglitcher is active, input changes polarity after three consecutive samples have shown the same new polarity. Thus, a delay of three oscillator clock cycles is introduced. This feature imposes a limit on the maximum transmission frequency.

The interrupt is generated after the last bit is clocked out. The interrupt source output of this module is routed to the Interrupt Controller logic. But this does not necessarily select it as input to the Interrupt Controller. Check section “Interrupt Controller” for the actually selectable sources and how to select them.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

24.1.2. Hardware settings

Clock frequency settings and the polarity of the data connections of the SPIs are settable by HW Options (Table 24–1). Refer to “HW Options” for setting them.

24.1.3. Initialization

After reset, a SPI is in standby mode (inactive).

Prior to entering active mode, proper SW configuration of the U-Ports assigned to function as data in- or outputs and clock in- or outputs has to be made (Table 24–1). Refer to “Ports” for details.

For entering active mode of a SPI, set the respective enable bit (Table 24–1).

Prior to operation, the desired clock frequency and telegram length have to be selected.

24.1.3.1. Clock Source

The SPI can be operated as clock master, using an internally generated clock, or as clock slave, using an externally generated clock.

The flag INTERN must be set in the SPIxM Mode register to operate the SPI as clock master. There are several options for selection of the internal clock. Each input of a 3-to-1 multiplexer can be programmed by HW Options to a different frequency. These three input frequencies F0SPI, F1SPI and F2SPI are used for all SPIs. The output of the 3-to-1 multiplexer is programmed by way of clock selection field (CSF) in register SPIxM. This clock can be used as shift clock directly, inverted and divided by 1.5 or 2.5. The shift clock is output by signal SPIx-CLK-OUT. f_0 can be selected as maximum clock speed in this operation mode.

Table 24–1: Module specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
All SPIs	F0SPI clock	SP0C			
	F1SPI clock	SP1C			
	F2SPI clock	SP2C			
SPI0	D in inversion	SP0C	SPI0-D-IN input	U3.5 special in	SR0. SPI0
	D out inversion	SP0C	SPI0-D-OUT output	U3.6 special out	
	Prescaler	SMC	SPI0-CLK-IN input	U3.4 special in	
			SPI0-CLK-OUT output	U3.4 special out	
SPI1	D in inversion	SP1C	SPI1-D-IN input	U4.0 special in	SR0. SPI1
	D out inversion	SP1C	SPI1-D-OUT output	U4.1 special out	
	Prescaler	SMC	SPI1-CLK-IN input	U3.7 special in	
			SPI1-CLK-OUT output	U3.7 special out	

If flag INTERN is zero, the SPI operates as clock slave and an externally generated clock is used. The external clock is input by signal SPIx-CLK-IN. This clock must not exceed 1.1 MHz.

The polarity and the sampling edge of the clock is defined by field SCLK in register SPIxM.

24.1.3.2. Telegram Length

Flag LEN9 in register SPIxM defines the length of a transferred frame. The ninth bit of the shift register is read or written at the location of flag BIT8 in register SPIxM.

24.1.4. Operation

24.1.4.1. Transmit Mode

Transmission is initiated by a write access to data register SPIxD. The SPI will immediately begin transmitting the selected number of data bits out from its shift register, in synchronism with the selected clock. A write access during a transmission is ignored. The frame is transmitted MSB first. In nine-bit mode flag BIT8 is MSB of the shift register (Fig. 24–2 to 24–5). At the end of the frame, an interrupt source signal is generated which may be selected to trigger an interrupt.

24.1.4.2. Receive Mode

The receive mode must be activated by a write access to register SPIxD. The SPI will immediately begin clocking in the selected number of data bits into its shift register, in synchronism with the selected clock. At the end of the frame, an interrupt source signal is generated which may be selected to trigger an interrupt.

24.1.4.3. DMA

Please refer to section “DMA” for information on the operation of the SPI in DMA mode.

24.1.5. Inactivation

Returning a SPI module to standby mode by resetting its respective enable bit (Table 24–1) will immediately terminate any running receive or transmit operation and will reset all internal registers.

24.1.6. Precautions

A single wire bus is easiest implemented by a wired-or configuration of the SPIx-D-OUT output port and the open drain output of the external transmitter:
simply configure the SPIx-D-OUT output port in Port Slow mode, always operate it in Port Special Output mode and connect it directly to the external open drain output. An external pull-up resistor is not necessary in this configuration because the SPIx-D-OUT output port supplies the necessary pull-up drive.
If the SPIx-D-OUT output port has to be operated in Port Fast mode, this simple scheme is not possible, because the pull-down action of the external open drain output may exceed the absolute maximum current rating of the SPIx-D-OUT output port. A discrete external wired-or is recommended for this situation.

During operation, make sure that the external clock does not start until after SPIxD has been written, otherwise correct data transfer is not be guaranteed.

24.2. Registers

The following registers are available once for SPI0 and SPI1 each.

SPIxD **SPI x Data Register**

	7	6	5	4	3	2	1	0	
r/w	Bit 7 to 0 of Rx/Tx Data								
	0	0	0	0	0	0	0	0	Res

SPIxM **SPI x Mode Register**

	7	6	5	4	3	2	1	0	
r/w	BIT8	LEN9	RXSEL	INTERN	SCLK		CSF		
	0	0	0	0	0	0	0	0	Res

BIT8 **Bit 8 of Rx/Tx Data**
r/w: Rx/Tx data bit.
In 8 bit mode (LEN9 = 0) this bit is undefined when read.

LEN9 **Frame Length 9 Bit Selection**
r/w0: 8 bit mode.
r/w1: 9 bit mode.

RXSEL **Receive Selection**
r/w0: Input active.
r/w1: Low level at input.

INTERN **Internal/External Clock Selection**
r/w0: Use external clock.
r/w1: Use internal clock.

SCLK **Sample Clock**
r/w: Clock polarity and edge of data sampling. (Table 24–2)

Table 24–2: SCLK usage

SCLK		Clock Polarity	Sampling Edge	See Fig.
1	0			
0	0	low	falling	24–3
0	1		rising	24–5
1	0	high	rising	24–2
1	1		falling	24–4

CSF **Clock Selection Field**
w: Source of internal clock (Table 24–3)

Table 24–3: CSF usage

CSF		Source of internal clock
1	0	
0	0	F0SPI
0	1	F1SPI
1	x	F2SPI

24.3. Timing

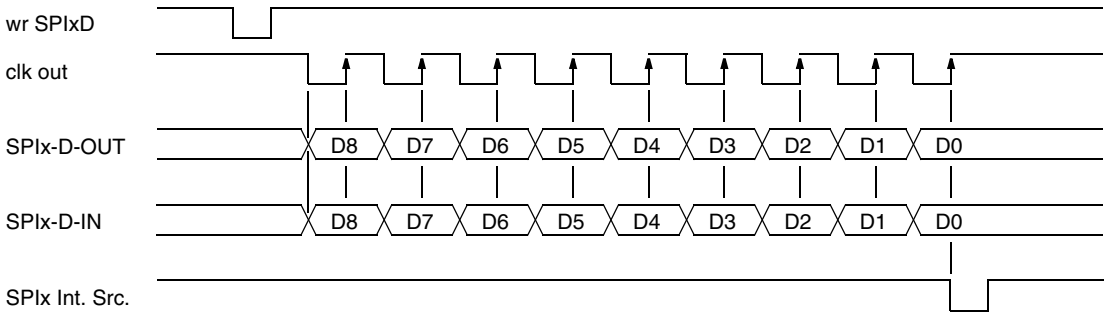


Fig. 24–2: Nine-bit frame. Data valid at rising edge. Clock inactive high

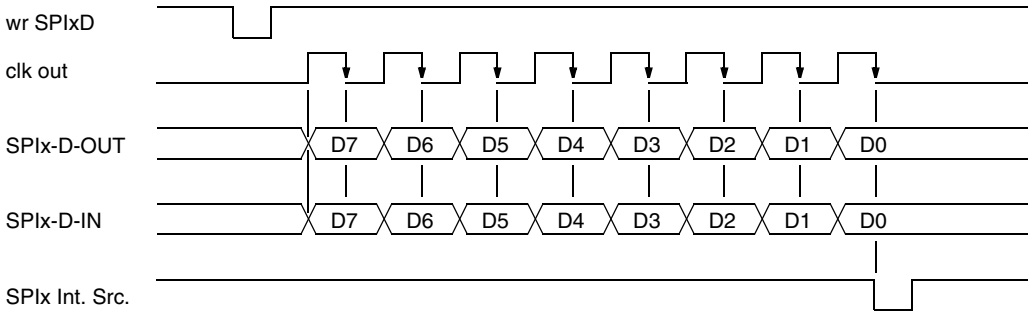


Fig. 24–3: Eight-bit frame. Data valid at falling edge. Clock inactive low

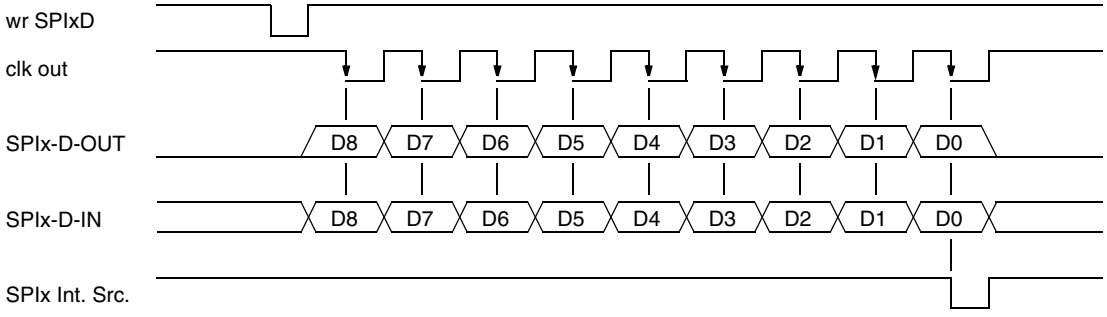


Fig. 24–4: Nine-bit frame. Data valid at falling edge. Clock inactive high

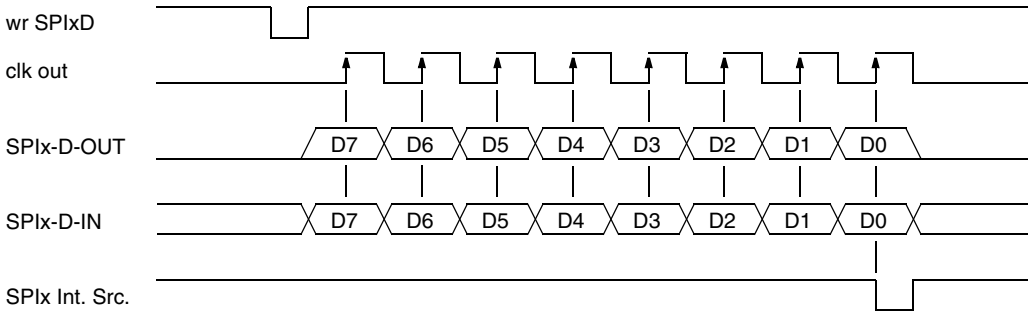


Fig. 24–5: Eight-bit frame. Data valid at rising edge. Clock inactive low

25. Universal Asynchronous Receiver Transmitter (UART)

A UART provides a serial receiver/transmitter. A 7-bit or 8-bit telegram can be transferred asynchronously with or without a parity bit and with one or two stop bits. A 13-bit baud rate generator allows a wide variety of baud rates. A two-word receive FIFO unburdens the SW. Incoming telegrams are compared with a register value. Interrupts can be triggered on transmission complete, reception complete, compare and break.

The number of UARTs implemented is given in Table 25–1. The “x” in register names distinguishes the module number.

Features

- Full duplex.
- 7-bit or 8-bit frames.
- Parity: None, odd or even.
- One or two stop bits.
- Receive compare register.
- Two word receive FIFO.
- 13-bit baud rate generator.

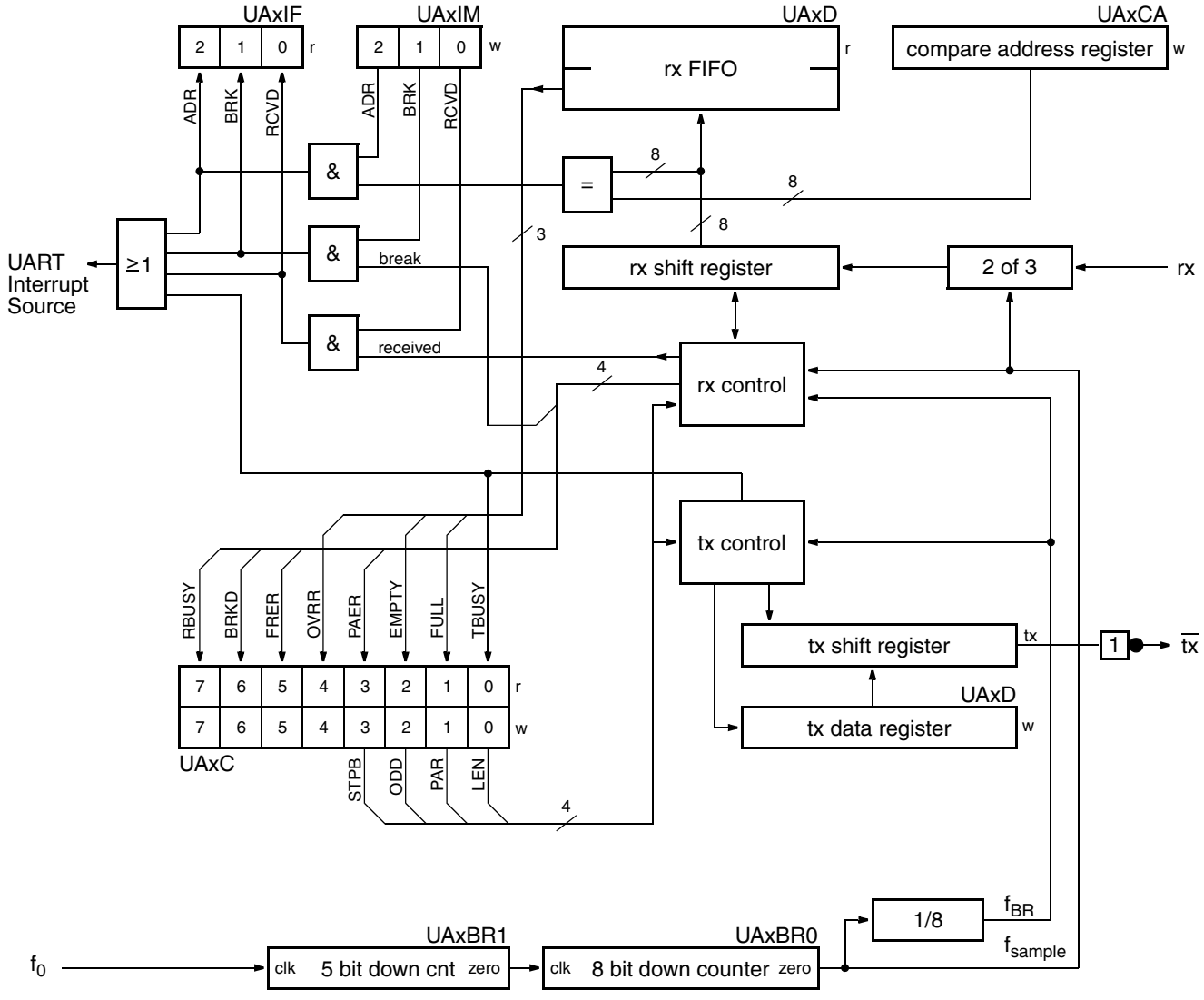


Fig. 25–1: Block diagram

25.1. Principle of Operation

25.1.1. General

A UART module contains a receive shift register that serves to receive a telegram via its RX input. A FIFO is affixed to it that stores two previously received telegrams.

A transmit shift register serves to transmit a telegram via its TX output.

Other features include a receive compare function, flexible interrupt generation and handling, and a set of control, error and status flags that facilitate management of the UART by SW.

The interrupt source output of this module is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “interrupt controller” for the actually selectable sources and how to select them.

A programmable baud rate generator generates the required bit clock frequency.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

A UART module is only capable to receive telegrams that differ by no more than ± 2.5% from its own baud rate setting.

25.1.2. Hardware settings

The polarity of most RX and TX connections of the UART is settable by HW Options (See table 25–1 and figure 25–2). Refer to “HW Options” for setting them.

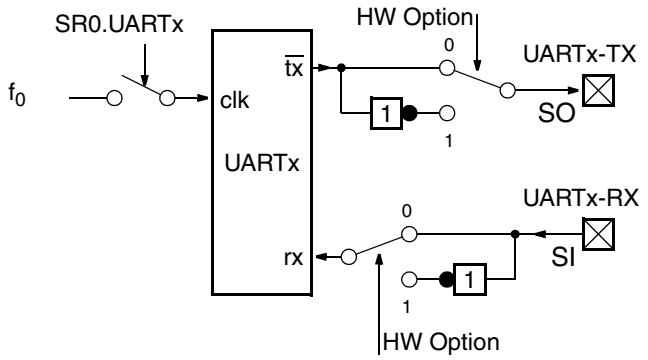


Fig. 25–2: Context diagram

25.1.3.2. Telegram Format

The format of a telegram is configured in the control and status register UAxC. A telegram starts with a start bit, followed by the data field. The data field consists of 7 or 8 data bit. There can be a parity bit after the data field. The telegram is finished by one or two stop bits (see Table 25–3 on page 164).

25.1.3. Initialization

After reset, a UART is in standby mode (inactive).

Prior to entering active mode, proper SW configuration of the U-Ports assigned to function as RX input and TX output has to be made (Table 25–1). The RX port has to be configured as special in and the TX port has to be configured as special out. Refer to “Ports” for details.

For entering active mode of a UART, set the respective enable bit (Table 25–1).

Prior to operation, the desired baud rate, telegram format, compare address and interrupt source configuration have to be done.

25.1.3.1. Baud Rate Generator

The receive and transmit baud rate is internally generated. The baud rate registers UAxBR0 (low byte) and UAxBR1 (high byte) serve to enter the desired 13-bit setting. Write UAxBR0 first, UAxBR1 last.

The baud rate generator is a 13-bit down-counter which is clocked by f_0. It generates the sample frequency:

$$f_{sample} = \frac{f_0}{\text{Value of Baud Rate Registers} + 1}$$

Its output frequency f_{sample} is divided by eight to generate the baud rate (bit/second).

$$BR = \frac{f_0}{(\text{Value of Baud Rate Registers} + 1) \times 8} = \frac{f_{sample}}{8}$$

$$\text{Value of Baud Rate Registers} = \frac{f_0}{BR \times 8} - 1$$

Table 25–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
UART0	RX inversion	UA0	UART0-RX input	U2.5 special in	SR0.UART0
	TX inversion		UART0-TX output	U2.4 special out	
UART1	RX inversion	UA1	UART1-RX input	U2.3 special in	SR0.UART1
	TX inversion		UART1-TX output	U2.2 special out	

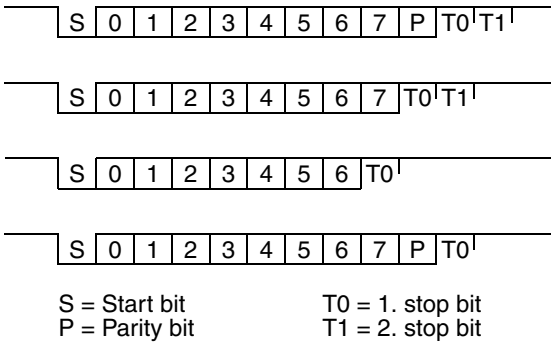


Fig. 25–3: Examples of telegram formats

The level of the start bit is always opposite to the neutral level. The level of the stop bits is always the same as the neutral level. If a parity bit is programmed, odd or even parity can be selected.

Table 25–2: Definition of parity bit

Parity Flag	Number of Ones	Parity Bit
odd	odd	0
odd	even	1
even	odd	1
even	even	0

As a general rule, the parity bit completes the number of ones in the data field to the selected parity.

25.1.3.3. Compare Address

The content of the Compare Address register UAxCA is compared with each received telegram. On a match, the interrupt flag ADR is set and the interrupt source signal is triggered.

The MSB of register UAxCA must be set to zero if transmission of a seven-bit data field is configured in register UAxC.

25.1.3.4. Interrupt

Four signals can trigger the UART interrupt source output. Three of them set their own flags in the interrupt flag register UAxIF and can be enabled by setting bits in the interrupt mask register UAxIM.

1. When the flag TBUSY in register UAxC is set to zero, the interrupt source output is triggered. This indicates that a transmission is finished and the transmit buffer is empty. There is neither an interrupt flag to indicate this event, nor a mask flag to disable this interrupt.

2. RCVD is generated by the receive control logic at the end of each received telegram even if the FIFO is full. This signal is enabled by setting the corresponding bit in register UAxIM.

3. BRK is generated by the receive control logic each time a break is detected. This signal is enabled by setting the corresponding bit in register UAxIM.

4. ADR is generated by the address comparator. This signal is enabled by setting the corresponding bit in register UAxIM.

BRK and ADR also set flags in the interrupt flag register UAxIF when enabled. The first RCVD interrupt, when the FIFO has been empty before, sets a flag in UAxIF too. Even if all interrupts are enabled in register UAxIM, the interrupt source output is triggered only once within a telegram. UAxIF flags remain valid until the end of the next telegram. ADR is not generated and the ADR flag is not set if a frame or parity error was detected in the corresponding telegram.

25.1.4. Operation

With proper HW configuration and SW initialization, a UART module is ready to transmit and receive telegrams in the selected format.

25.1.4.1. Transmit

A write access to UART Data register UAxD immediately loads the transmit shift register and starts transmission with sending the start bit. The flag TBUSY in register UAxC is set.

At the end of transmission the interrupt source signal is triggered and the flag TBUSY is reset.

To avoid data corruption, ensure that flag TBUSY is LOW before writing to UAxD

25.1.4.2. Receive

A first negative edge of a telegram on the RX line of a UART starts a receive cycle and sets the flag RBUSY in UAxC. After reception of the last bit of the telegram, the telegram content, together with its status information, is transferred to the receive FIFO and an interrupt is generated. RBUSY is resetted. Telegram data are available in register UAxD, telegram status in register UAxC.

During reception, the following checks are performed according to the register UAxC setting:

1. A parity error is detected if the parity of the received telegram does not match the programmed parity. The flag PAER in register UAxC is set in this case. Differing telegram length settings in register UAxC and receiver may also cause parity errors.

2. A frame error is detected if the level of start or stop bits violate the transmission rule. The flag FRER in register UAxC is set in this case.

3. A break condition is detected if the receive input remains low for one complete telegram duration. When a break starts during telegram, this condition must extend over another telegram length to be properly detected. This event sets the flag BRKD in register UAxC and can trigger the interrupt source output if enabled. After a break, the receive input must be high for at least 1/4 of the bit length before a new telegram can be received.

Telegrams of an external RS232 interface are correctly received, even if they are transmitted without gaps (the start bit immediately follows the stop bit of the preceding telegram).

25.1.4.3. Receive FIFO

The receive FIFO is able to buffer the data fields of two consecutive telegrams. But not only the data field of a telegram is double buffered, the related information is double buffered too. The flags PAER, FRER and BRKD in register UAxC apply to a certain telegram and are thus double buffered.

The receive FIFO is full if two telegrams were received but the SW did not yet read register UAxD. If there is a third telegram, it is not written to the FIFO and its data are lost. The flags EMPTY, FULL and OVRR show the status of the FIFO. EMPTY indicates that there is no entry in the FIFO. FULL will be set with the second entry in the receive FIFO and indicates that there is no more entry free. OVRR indicates that there was a third telegram which could not be written to the FIFO.

Status flags are readable as long as the corresponding data field was not read from register UAxD. As soon as a FIFO entry is read out, the status flags of this entry are lost. They are overwritten by the flags of the second entry. SW first has to read the flags and then the corresponding FIFO entry.

The flags PAER, FRER and BRKD apply to a certain telegram and are only valid if there is at least one entry in the FIFO (EMPTY = 0). The flags EMPTY, FULL and OVRR apply to the FIFO and are valid all the time.

25.1.5. Inactivation

Returning a UART module to standby mode by resetting its respective enable bit (Table 25–1) will immediately terminate any running receive or transmit operation and will reset all internal registers.

25.2. Timing

The duration of a telegram results from the total telegram length in bits (L_{TG}) (see Table 25–3 on page 164) and the baud rate (BR).

$$t_{TG} = \frac{L_{TG}}{BR}$$

The incoming signal is sampled with the sample frequency and filtered by a 2 of 3 majority filter. A falling edge at the output of the majority filter starts the receive timing frame for the telegram. An individual bit is sampled with the fifth sample clock pulse within that timing frame (cf. Fig. 25–4 and 25–5). If a bit was the last bit of its telegram, reception of a new telegram can start immediately after this sample. With a receive telegram, interrupt source is triggered and flags are set just after the sample of the last stop bit. With a transmit telegram, interrupt source is triggered and BUSY reset after the nominal end of the last stop bit.

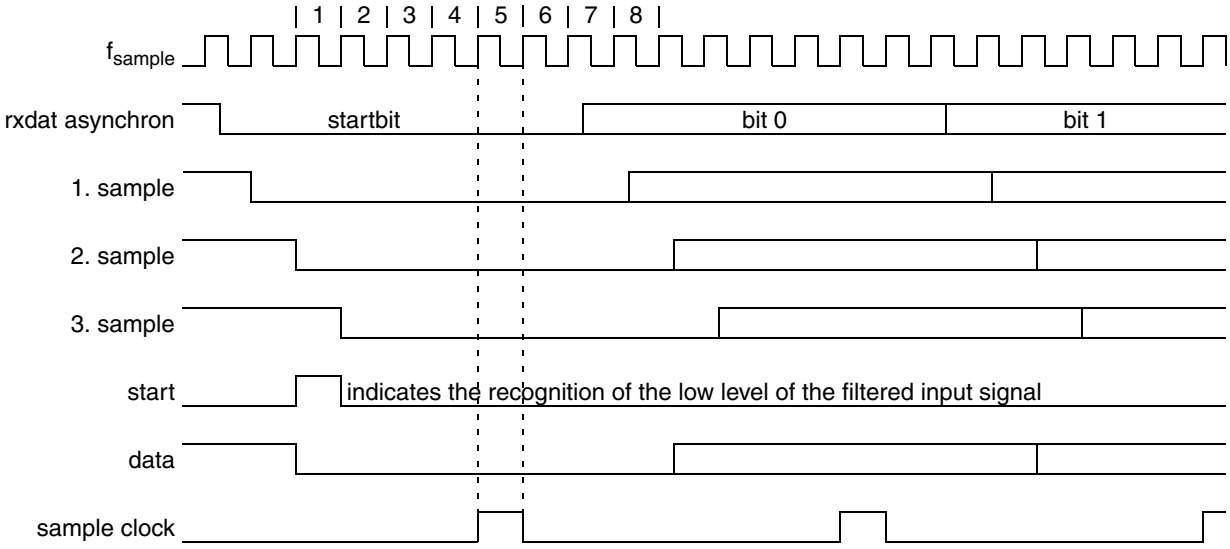


Fig. 25–4: Start of telegram

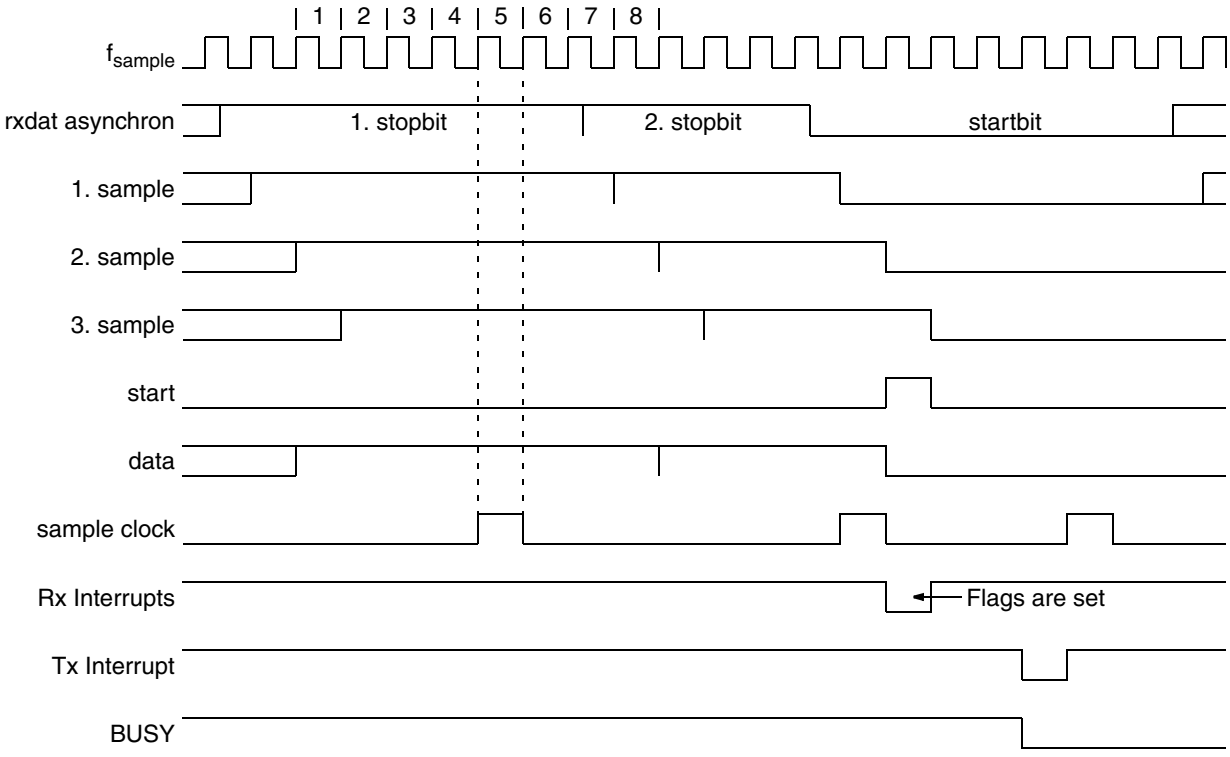


Fig. 25–5: End of telegram

25.3. Registers

UAxD UART x Data Register

	7	6	5	4	3	2	1	0	
r	Receive register								
w	Transmit register								
	x	x	x	x	x	x	x	x	Res

UAxC UART x Control and Status Register

	7	6	5	4	3	2	1	0	
r	RBUSY	BRKD	FRER	OVRR	PAER	EMPTY	FULL	TBUSY	
	0	x	x	0	x	1	0	0	Res
w	x	x	x	x	STPB	ODD	PAR	LEN	
	x	x	x	x	0	0	0	0	Res

RBUSY Receiver Busy
r0: Not busy.
r1: Busy.

BRKD Break Detected
r0: No break.
r1: Break.

FRER Frame Error Detected
r0: No error.
r1: Error.

OVRR Overrun Detected
r0: No overrun.
r1: Overrun.

PAER Parity Error Detected
r0: No error.
r1: Error.

EMPTY Rx FIFO Empty
r0: Not empty.
r1: Empty.
There is at least one entry present if EMPTY is zero. PAER, FRER and BRKD are not valid if EMPTY is set.

FULL Rx FIFO Full
r0: Not full.
r1: Full.

TBUSY Transmitter Busy
r0: Not busy.
r1: Busy.
Do not write to register UAxD as long as BUSY is true.

STPB Stop Bits
w0: One stop bit.
w1: Two stop bits.

ODD Odd Parity
w0: Even parity.
w1: Odd parity.

PAR Parity On
w0: No parity.
w1: Parity on.

LEN Length of Frame
w0: 7bit frame.
w1: 8bit frame.

Table 25–3: Telegram format and length

LEN	PAR	STPB	Format	L_{TG}
0	0	0	S, 7D, T0	9
0	0	1	S, 7D, T0, T1	10
0	1	0	S, 7D, P, T0	10
0	1	1	S, 7D, P, T0, T1	11
1	0	0	S, 8D, T0	10
1	0	1	S, 8D, T0, T1	11
1	1	0	S, 8D, P, T0	11
1	1	1	S, 8D, P, T0, T1	12

UAxBR0 UART x Baud Rate Register low byte

	7	6	5	4	3	2	1	0	
w	Bit 7 to 0 of Baud Rate								
	0	0	0	0	0	0	0	0	Res

UAxBR1 UART x Baud Rate Register high byte

	7	6	5	4	3	2	1	0	
w	x	x	x	Bit 12 to 8 of Baud Rate					
	-	-	-	0	0	0	0	0	Res

The Baud Rate Registers UAxBR0 and UAxBR1 have to be written low byte first to avoid inconsistencies. UAxBR0 is the low byte.

Valid entries in the Baud Rate Registers range from 1 to 8191. Don't operate the baud rate generator with its reset value zero.

UAxCA UART x Compare Address Register

	7	6	5	4	3	2	1	0	
w	Bit 7 to 0 of address								
	0	0	0	0	0	0	0	0	Res

UAxIM **UART x Interrupt Mask Register**

	7	6	5	4	3	2	1	0	
w	x	x	x	x	x	ADR	BRK	RCVD	
	-	-	-	-	-	0	0	0	Res

ADR **Mask Compare Address Detected**
w0: Disable interrupt.
w1: Enable interrupt.

BRK **Mask Break Detected**
w0: Disable interrupt.
w1: Enable interrupt.

RCVD **Mask Received a Telegram**
w0: Disable interrupt.
w1: Enable interrupt.

UAxIF **UART x Interrupt Flag Register**

	7	6	5	4	3	2	1	0	
r	Test	Test	Test	Test	Test	ADR	BRK	RCVD	
	-	-	-	-	-	x	0	0	Res

Test **Reserved for test (do not use)**

ADR **Compare Address Detected**
r0: No Interrupt.
r1: Interrupt pending.

BRK **Break Detected**
r0: No Interrupt.
r1: Interrupt pending.

RCVD **Received a Telegram**
r0: No Interrupt.
r1: Interrupt pending.

26. I²C-Bus Master Interface

The IC contains two independent I²C-bus master interface units (I²C), 0 and 1. These are pure master systems, multi-master busses are not realizable. The units contain read and write buffers with interrupt logic which makes automatic and software-independent operation possible for most types of I²C telegrams. Because of the internal clock pre-scaler, telegram clock rate does not depend on the system clock rate.

Features

- I²C master module
- 5-byte transmit FIFO
- 3-byte receive FIFO
- Progammable input deglitcher

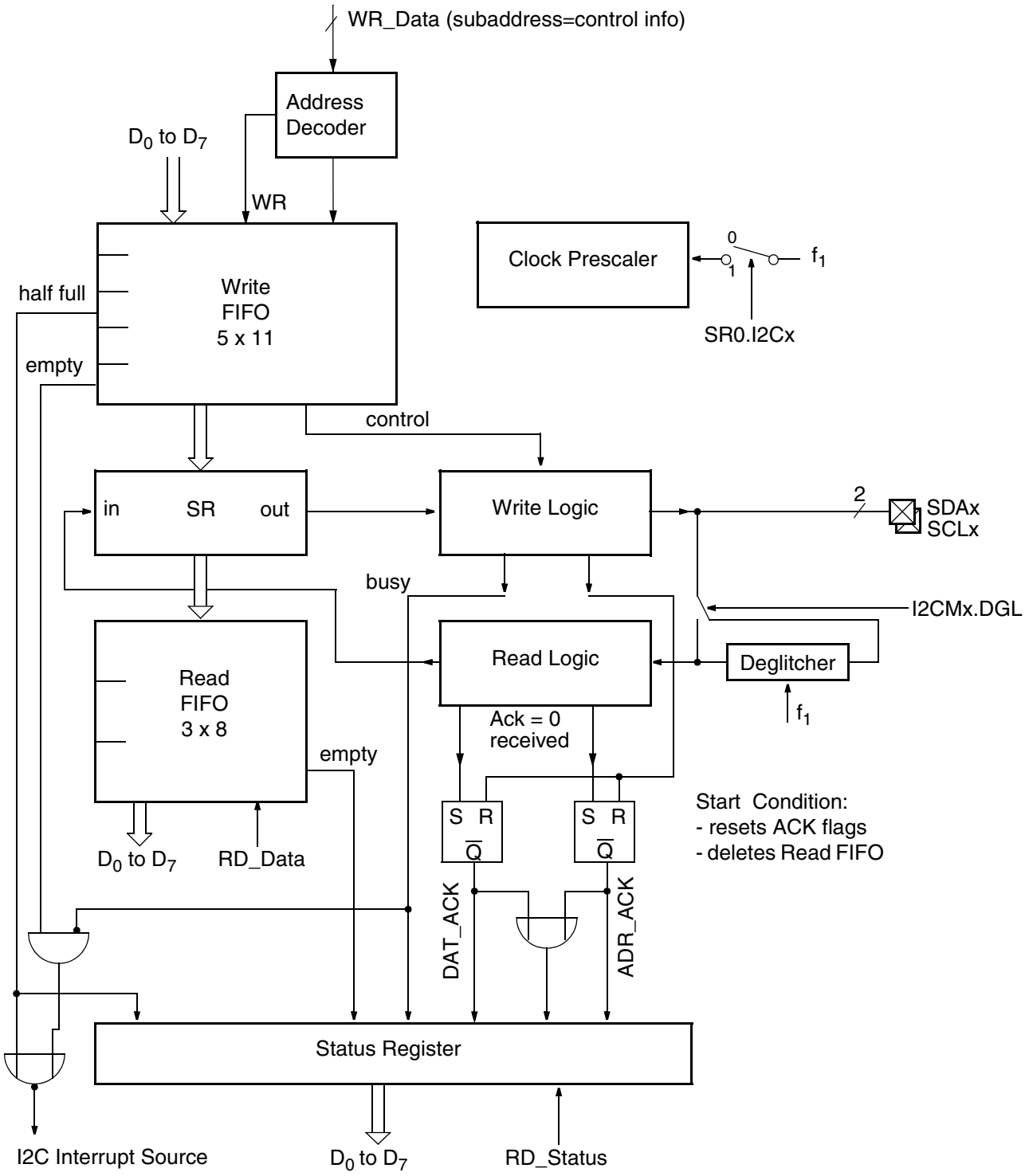


Fig. 26–1: Block diagram of I²C-bus master interface

26.1. Principle of Operation

26.1.1. General

26.1.2. Hardware Settings

Since the telegram clock rate is register programmable there is no HW option for the I²C-Bus Master Interface.

26.1.3. Initialization

After system reset the I²C is in standby mode, i.e. the block internal clock is halted and all registers are set to their reset value. By default, the input deglitcher is on, limiting the obtainable bit rate to 208.3 kbit/s (see Table 26–2).

In standby mode the clock is halted. Programming of the I²C registers is possible and the Write-FIFO can be filled.

Prior to operation, proper SW configuration of the U-Ports assigned to function as I²C Double Pull-down port has to be made. See table 26–1 and section "Ports" for details.

The bit rate and the desired input deglitcher configuration has to be set up in register I2CMx in order to get into an active and useful mode. All other registers serve I²C data I/O purposes.

Table 26–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
I2C0	U2.0 CAN0/SCL0 output multiplexer	PM.U20	SCL0	U2.0 special out, double pull-down mode	SR0.I2C0
	U2.1 CAN0/SDA0 output multiplexer		SDA0	U2.1 special out, double pull-down mode	
I2C1			SCL1	U5.1 special out, double pull-down mode	SR0.I2C1
			SDA1	U5.2 special out, double pull-down mode	

26.1.4. Operation

A complete telegram is assembled by the software out of individual sections. Each section contains 8-bit data. This data is written into one of the six possible write registers. Depending on the chosen address, a certain part of an I²C-bus cycle is generated: start, data, stop, with or without acknowledge. By means of corresponding calling sequences it is therefore possible to join even very long telegrams (e.g. long data files for auto increment addressing of I²C slaves).

The software interface contains a 5-word-deep Write-FIFO for the control-data registers as well as a 3-word-deep Read-FIFO for the received data. Thus most of the I²C telegrams can be transmitted to the hardware without the software having to wait for empty space in the FIFO.

All address and data fields appearing on the bus are constantly monitored and written into the Read-FIFO. The software can then check these data in comparison with the scheduled data.

Every reception of a start or restart condition immediately empties the Read-FIFO. The Read-FIFO stops if it is full. It's not overwritten, further received data are lost.

If a read instruction is handled, the interface must send the data word 0xFF so that the responding slave can insert its data. In this case the Read-FIFO contains the read-in data.

If telegrams longer than 3 bytes (1 address, 2 data bytes) are received, the software must check the filling condition of the Write-FIFO and, if necessary, fill it up and read out the Read-FIFO. A variety of status flags is available for this purpose:

– The 'half full' flag I2CRSx.WFH is set if the Write-FIFO is filled with exactly three bytes.

– The 'empty' flag I2CRSx.RFE is set if there is no more data available in the Read-FIFO.

– The 'busy' flag I2CRSx.BUSY is activated by writing any byte to any one of the Write registers. It stays active until the I²C-bus activities are stopped after the stop condition generation.

Moreover the ACK-bit is recorded separately on the bus lines for the address and the data fields. However, the interface itself can set the address ACK=0. In any case the two ACK flags show the actual bus condition. These flags will be reset with the next I²C start condition.

There is one data acknowledge (DACK) flag available. It indicates the level of the last received ACK bit. It will be cleared to zero with the reception of a zero and it will be set to one with the reception of a one within the acknowledge field of a data byte. Thus, after the stop condition, it indicates whether the last of the data bytes was acknowledged or not.

The bus activity starts immediately after the first write to the Write-FIFO. The transmission can be synchronized by an artificial extension of the low phase of the clock line. Transmission is not continued until the state of the clock line is high once again. Thus an I²C slave device can adjust the transmission rate to its own abilities.

The figures 26–2, 26–3 and 26–4 show the basic principle of I²C telegram transmission as a quick reference. Refer to the official I²C documentation for more details.

26.1.4.1. Interrupt Generation and Operation

The transmission of telegrams generates two classes of interrupts.

– End-of-Telegram interrupt

– Write-FIFO-Half-Full interrupt

The End-of-Telegram interrupt is generated if the Write-FIFO is empty and the stop condition is completed. The flag BUSY is zero in this case.

A Write-FIFO-Half-Full interrupt is generated by the Write-FIFO each time the entry number reduces from 4 to 3 ('half-full' state, WFH flag set), but not before at least two entries have been transferred from the Write-FIFO to the SR. Reaching the fill level of 3 entries during re-filling the Write-FIFO may (but must not) generate an interrupt too.

Those interrupts occurring during re-filling the Write-FIFO could be a source of problems. They would be served as soon as the momentary ISR is left. This situation would confuse the ISR seriously, because though the flag WFH is zero, the Write-FIFO is filled above the half-full level.

The Write-FIFO-Half-Full interrupt is essential as start of a refill routine. This routine has to guarantee that at the end the fill state is definitely > 3 entries. In this way prepared, again dropping to 3 entries safely triggers the next Write-FIFO-Half-Full interrupt, starting the next refill routine.

The strategy to obtain a fill state > 3, is to fill up to 'half-full' state (WFH flag set), and then to add 2 entries above that. During this refill routine, all other interrupts have to be disabled. At the end of this refill routine, to avoid ambiguities, any I²C interrupt requests received in meantime, have to be cleared.

26.1.4.2. Example of Operation

The software has to work in the following sequence (ACK=1) to read a 16-bit word from an I²C device address 0x10 (on condition that the bus is not active):

-write 0x21 to	I2CWS0x
-write 0xFF to	I2CWD0x
-write 0xFF to	I2CWP1x
-read RFE bit from	I2CRSx
-read dev. address from	I2CRDx
-read RFE bit from	I2CRSx
-read 1st databyte from	I2CRDx
-read RFE bit from	I2CRSx
-read 2nd databyte from	I2CRDx

The value 0x21 in the first step results from the device address in the 7 MSBs and the R/W-bit (read=1) in the LSB. If the telegrams are longer, the software has to ensure that neither the Write-FIFO nor the Read-FIFO can overflow.

To write data to this device:

-write 0x20 to	I2CWS0x
-write 1st databyte to	I2CWD0x
-write 2nd databyte to	I2CWP0x

26.1.5. Inactivation

Since the described block is an I²C master, all I²C bus activity stops if the end of a telegram is reached. I²C slaves cannot start any bus activity on their own. However, the block internal clock is always running at full speed of I²C clock (4 or 5 MHz), independent of the bit rate divider setting. The standby mode is therefore intended for the lowest possible power consumption.

Switching off the I²C module by the corresponding enable flag in the standby register may result in the output signal SDAx being drawn to zero for the time the I²C module is off. To avoid this situation, disable the corresponding port pin as special output during operation pausing and allow the port to return to high level.

26.1.6. Precautions

Switching off the I²C module by the corresponding enable flag in the standby register, and then re-enabling it, may result in the last transmitted byte being transmitted again. To avoid this situation, disable the corresponding port pin as special output during operation pausing. After re-enabling the module by setting the corresponding enable flag, wait at least the transmission time of one byte before re-enabling this special output.

For the effect of CPU clock modes on the operation of this module refer to section "CPU and Clock System" (see Table 4–1 on page 41).

Note: The I²C block uses U-Ports as connection to the outside world. This implies that neither logic output low level switching specs nor logic input value specs of the official I²C specification document are literally met. Refer to section "Ports" for the actual spec values of this implementation.

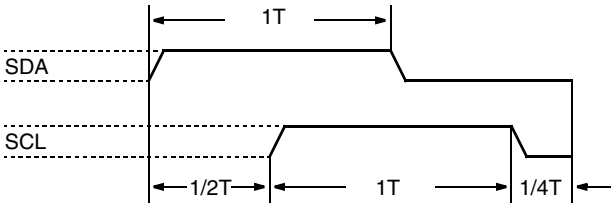


Fig. 26–2: Start or restart condition I²C-bus

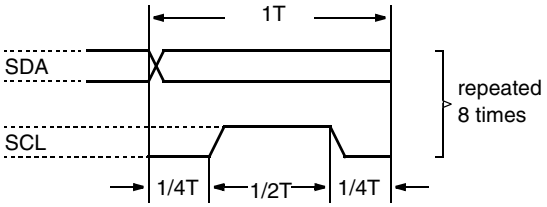


Fig. 26–3: Single bit on I²C-bus

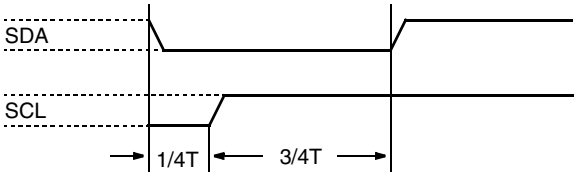


Fig. 26–4: Stop condition I²C-bus

26.2. Registers

I2CWS0x	I2C Write Start Register 0						
7	6	5	4	3	2	1	0
w	I2C Address						
0x00							Res

Writing this register moves I^2C start condition, I^2C Address and ACK=1 (no acknowledge) into the Write FIFO.

I2CWS1x	I2C Write Start Register 1						
7	6	5	4	3	2	1	0
w	I2C Address						
0x00							Res

Writing this register moves I^2C start condition, I^2C Address and ACK=0 (acknowledge) into the Write FIFO.

I2CWD0x	I2C Write Data Register 0						
7	6	5	4	3	2	1	0
w	I2C Data						
0x00							Res

Writing this register moves I^2C Data and ACK=1 (no acknowledge) into the Write FIFO.

I2CWD1x	I2C Write Data Register 1						
7	6	5	4	3	2	1	0
w	I2C Data						
0x00							Res

Writing this register moves I^2C Data and ACK=0 (acknowledge) into the Write FIFO.

I2CWP0x	I2C Write Stop Register 0						
7	6	5	4	3	2	1	0
w	I2C Data						
0x00							Res

Writing this register moves I^2C Data, ACK=1 (no acknowledge) and I^2C stop condition into the Write FIFO.

I2CWP1x	I2C Write Stop Register 1						
7	6	5	4	3	2	1	0
w	I2C Data						
0x00							Res

Writing this register moves I^2C Data, ACK=0 (acknowledge) and I^2C stop condition into the Write FIFO.

I2CRDx	I2C Read Data Register						
7	6	5	4	3	2	1	0
r	I2C Data						
0x00							Res

Reading this register returns the content of the Read FIFO.

I2CRSx	I2C Read Status Register							
7	6	5	4	3	2	1	0	
r	x	OACK	AACK	DACK	BUSY	WFH	RFE	x
0	0	0	0	0	0	0	0	Res

OACK **"OR"ed Acknowledge**
r: AACK OR DACK.

AACK **Address Acknowledge**
r1: Not acknowledged (received a one)
r0: Acknowledged (received a zero)

DACK **Data Acknowledge**
r1: Not acknowledged (received a one)
r0: Acknowledged (received a zero)

BUSY **Busy**
r1: I^2C Master Interface is busy.
r0: I^2C Master Interface is not busy.

WFH **Write-FIFO Half Full**
r1: Write-FIFO contains exactly 3 bytes.
r0: Write-FIFO contains more or less than 3 bytes.

RFE **Read-FIFO Empty**
r1: Read-FIFO is empty.
r0: Read-FIFO is not empty.

I2CMx	I2C Mode Register							
7	6	5	4	3	2	1	0	
w	DGL	SPEED						
1	0x02							Res

DGL **Input Deglitcher**
w1: Deglitcher is active.
w0: Deglitcher is bypassed.
If the deglitcher is active, the maximum bit rate is limited. SPEED must be programmed to 6 at least. The maximum bit rate may be further reduced by the bus load.

SPEED **Speed Select** (Table 26–2)
w: I^2C Bit Rate = f_1 / (4 × SPEED).

Table 26–2: SPEED usage: I^2C bit rates

SPEED	f_1 Division by	Bit Rate @ f_1 = 5 MHz	Bit Rate @ f_1 = 4 MHz
0	128 × 4 (!)	**9.8 kbit/s**	**7.8 kbit/s**
1 [1)]	1 × 4	**1.25 Mbit/s**	**1.00 Mbit/s**
2 [1)]	2 × 4	**625 kbit/s**	**500 kbit/s**
3 [1)]	3 × 4	**416.7 kbit/s**	**333.3 kbit/s**
4 [1)]	4 × 4	**312.5 kbit/s**	**250 kbit/s**
5 [1)]	5 × 4	**250 kbit/s**	**200 kbit/s**
6	6 × 4	**208.3 kbit/s**	**166.7 kbit/s**
7	7 × 4	**178.6 kbit/s**	**142.9 kbit/s**
...		**...**	
127	127 × 4	**9.8 kbit/s**	**7.9 kbit/s**
1) These bit rates may only be set with a bypassed input deglitcher (I2CMx.DGL=0)			

27. CAN Manual

This manual describes the user interface of the CAN module. For further information about the CAN bus, please refer to the CAN specification 2.0B from Bosch.

Features

- Bus controller according to CAN licence specification 1992, 2.0B
- Supports standard and extended telegrams
- FullCAN: up to 32 Rx and Tx telegrams
- Variable number of receive buffers
- Programmable acceptance filter
  Single, group or all telegrams received.
- Time stamp for each telegram
- Overwrite mode programmable for each telegram
- Programmable baud rate. Max. 1 MBd @ 8 MHz
- Sleep mode

The CAN interface is a VLSI module which enables coupling to a serial bus in compliance with CAN specification 2.0B. It controls the receiving and sending of telegrams, searches for Tx telegrams and interrupts and carries out acceptance filtering. It supports transmission of telegrams with standard (11-bit) and extended (29-bit) addresses.

The CAN interface can be configured as BasicCAN or FullCAN. It enables several active receive and transmit telegrams and supports the remote transmission request. The number of telegrams which can be handled depends mainly on the size of the communication RAM (16 byte per telegram), the system clock and the transmission speed. A maximum of 254 telegrams can be handled.

A mask register makes it possible to receive different groups of telegram addresses with different receive telegrams. Transmitting or receiving of a telegram as well as the occurrence of an error can trigger an interrupt.

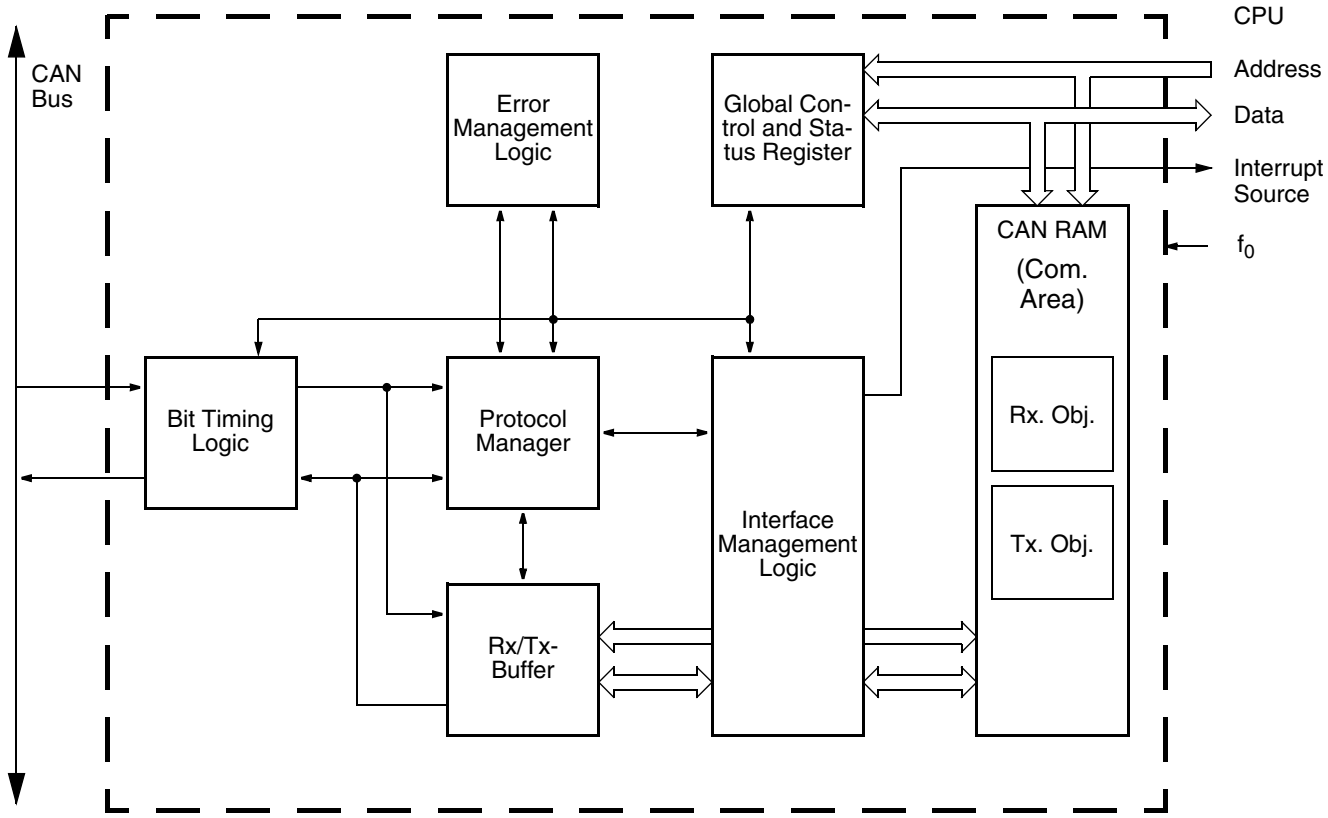


Fig. 27–1: Block diagram of the CAN bus interface

27.1. Abbreviations

BI	CAN Bus Interface
BTL	Bit Timing Logic
CAN	Controller Area Network
CA	Communication Area
CO	Communication Object
CM	Communication Mode
CRC	Cyclic Redundancy Code
DLC	Data Length Code
EoCA	End of CA
Ext. ID	Extended Identifier
Ext. Tg	Extended Telegram
GCS	Global Control and Status Register
ID	Identifier
IML	Interface Management Logic
Rx. Obj.	Receive Object
RxTg	Receive Telegram
Std. ID	Standard Identifier
Std. Tg	Standard Telegram
TD	Telegram Descriptor
Tg	Telegram
TQ	Time Quantum
Tx. Obj.	Transmit Object
TxTg	Transmit Telegram

27.2. Functional Description

27.2.1. HW Description

The CAN bus interface consists of the following components:

Bit Timing Logic: Scans the bus and synchronizes the CAN bus controller to the bus signal.

Protocol Manager: The PM monitors or generates the composition of a telegram and performs the arbitration, the CRC and the bit stuffing. It controls the data flow between Rx/Tx buffer and CAN bus. It also drives the error management logic.

Error Management Logic: Adds up the error messages received from the protocol manager and generates error messages when particular values are exceeded. Guarantees the error limitation as per the CAN Spec. V2.0B.

Interface Management Logic: The IML scans the Communication Area (CA) in the CAN-RAM for transmit telegrams. As soon as it finds one, it enters it into the Rx/Tx buffer and reports it to the protocol manager as ready for transmission. If a telegram is received, the IML carries out the acceptance filtering, i.e., scans the CA, taking into account the identifier mask register in the GCS, for a Tg with the appropriate address. After correct reception, it copies the Tg from the Rx/Tx buffer to the CA. The IML also reports to the CPU the valid transfer of a telegram or given errors per interrupt.

The interrupt source output of this module is routed to the interrupt controller logic. But this does not necessarily select it as input to the interrupt controller. Check section “interrupt controller” for the actually selectable sources and how to select them.

Rx/Tx buffer: This is used to buffer a full telegram (ID, DLC, data) during sending and receiving.

Global Control and Status Register: The GCS contains registers for the configuration of the BI. It also contains error and status flags and an identifier mask. The error counter and the capture timer can be read from the GCS.

Receive Object: The BI enters received telegrams into a matching Rx-Object. It can be retrieved from the application.

Transmit Object: The application enters data into the Tx-Object and reports it ready for transmission. The BI sends the telegram as soon as the bus traffic allows.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

27.2.2. Memory Map

From the CAN bus interface the user sees two storage areas in the user RAM area. The BI is configured with the global control and status registers (GCS). It also indicates the status here. The communication area (CA) contains the Rx and Tx telegrams.

The communication area lies in the CAN-RAM. The end of the communication area is fixed by the first control byte of an object whose 3 MSBs contain only ones (Communication Mode = 7 = EoCA). The area after this is available to the user.

The CA consists of communication objects (COs). A CO consists of 6 bytes telegram descriptor (TD), 8 data bytes and the time stamp which is 2 bytes long. The TD contains the address (ID) and the length of a telegram (DLC) as well as control bits which are needed for access to the CO and for the transmission of a telegram.

In the BasicCAN and the FullCAN versions, all the communication objects have the same, maximum size of 16 bytes. Unassigned storage locations in the data area of a CO can be freely used.

The maximum number of COs is limited by the time which the CAN interface has to search for an identifier in the communication area.

27.2.3. Global Control and Status Registers (GCS)

The GCS registers can be used to determine the behavior of the CAN interface. As well as flags for the interrupts, halt and sleep modes, they also contain interrupt index, ID mask, bus timing, error status, output control registers, baud rate prescalers, Tx and Rx error counters as well as the capture timer.

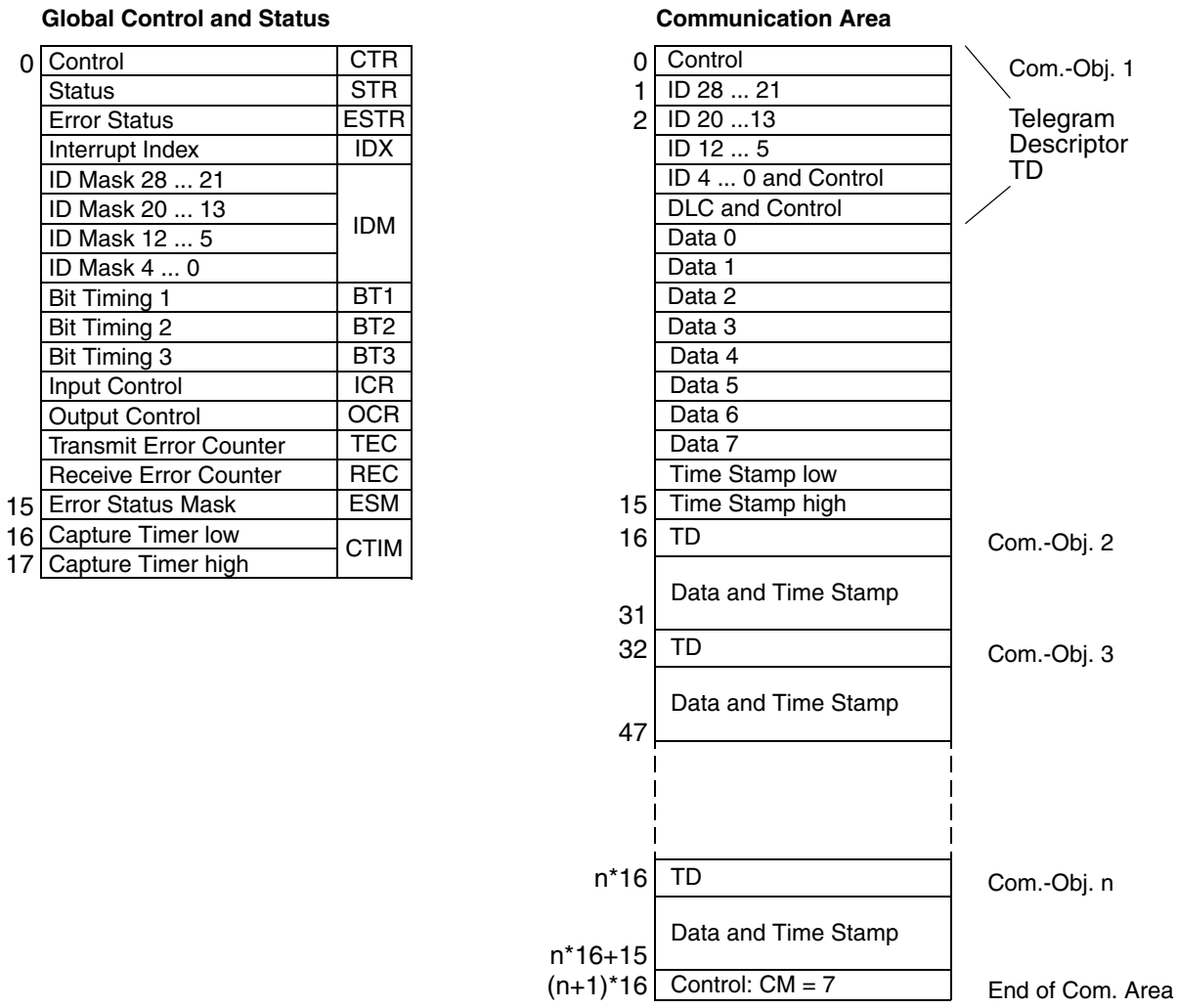


Fig. 27–2: Memory allocation

Access modes:
r: read
w: write
i: init (BI halted)
w0: clear
w1: set

CANxCTR Control Register

	7	6	5	4	3	2	1	0	
r/w	HLT	SLP	GRSC	EIE	GRIE	GTIE	BOST	rsvd	
	1	0	0	0	0	0	0	x	Res

HLT Halt
r/w0: Run.
r/w1: Halt.
Switches the CAN interface into the halt mode. Transmissions which have been started are brought to an end. The halt acknowledge is indicated in the status register (HACK). Re-initialization can be carried out in the halt mode (HACK is set). After this, the halt flag must be deleted again. After a reset, HLT is set.

If HLT is set during a Tx-Tg and this has to be repeated (error or no acknowledge), the BI stops yet. The corresponding TxCO is still reserved, however, and can no longer be operated from BI. Therefore, when HLT is set, the CA should always be re-initialized if the last Tx-Tg has not been correctly transmitted (Status Transfer Flag is still deleted).

If HLT is set during the BI is in Bus-Off mode, the BI stops after Bus-Off mode is finished. Flag BOFF is cleared then and receive and transmit error counters are reset to zero.

SLP Sleep
r/w0: Run.
r/w1: Sleep.
The BI goes into the sleep mode when the sleep flag is set and a started Tg is terminated. The sleep mode is finished as

soon as a dominant bus level is detected, or the sleep flag is deleted.

GRSC Global Rescan
r0: Don't rescan.
r1: Rescan
w0: Unaffected
w1: Set.
The microprocessor can set this flag in order to initiate a transmit telegram search at the beginning of the communication area. The BI resets the bit. The BI also sets the GRSC flag if the flag RSC has been set in a telegram descriptor of a Tx-Tg just operated, and thereby initiates a rescan. If the microprocessor writes a zero, nothing happens.

EIE Error-Interrupt-Enable
r/w0: Disabled.
r/w1: Enabled.

GRIE Global Rx-Interrupt-Enable
r/w0: Disabled.
r/w1: Enabled.

GTIE Global Tx-Interrupt-Enable
r/w0: Disabled.
r/w1: Enabled.

BOST Bus-Off Stop Select
r/w0: Don't stop when leaving Bus-Off mode.
r/w1: Stop when leaving Bus-Off mode.
The flag HLT is set by the BI after leaving the Bus-Off recovery sequence. The SW has to restart the CAN module in this case after re-initialisation. Consider the flag HACK even in this case.

CANxSTR Status Register

	7	6	5	4	3	2	1	0
r	HACK	BOFF	EPAS	ERS	rsvd	rsvd	rsvd	rsvd
	1	0	0	0	x	x	x	x

Res

HACK Halt-Acknowledge
r0: Running.
r1: Halted.
Is set by the BI when it enters the halt mode. It is deleted again when the halt mode is exited.

BOFF Bus-Off
r0: Bus active.
r1: Bus off.
With this flag the BI indicates whether the node is still actively participating in the bus. If the transmit error counter reaches a value of > 255 (overflow), the node is separated from the bus and the flag is set. The bus-off mode is left after the bus-off recovery sequence. The flag CANxCTR.BOST defines the behavior after leaving bus-off mode.

EPAS Error-Passive
r0: Error active.
r1: Error passive.
With this flag the BI indicates whether the node is still participating in the bus with active error frames. If an error counter has reached a value > 127, the node only transmits passive error frames and the flag is set.

ERS Error-Status
r0: No Errors.
r1: Errors.
This flag is set when the BI detects an error and the appropriate error flag is not masked in the error status mask register. It is set even if an error counter is greater than 96. It means that a bit has been set in the error status register. As soon as all the flags in the error status register are either deleted or masked, ERS is also deleted.

As long as a bit is set in the CANxESTR and not masked, the ERS bit is also set in the status register. If EIE has been set in the control register, an interrupt is triggered too; i.e., the value 254 is entered in the register CANxIDX as soon as it is free, and the interrupt source output is triggered.

To erase a bit in the CANxESTR the user must write a one at the appropriate place. Places at which he writes a zero will not be changed. Because it makes sense to erase only those bits which have previously been read, only the value which has been read has to be re-written.

CANxESTR Error Status Register

	7	6	5	4	3	2	1	0
r/w	GDM	CTOV	ECNT	BIT	STF	CRC	FRM	ACK
	0	0	0	0	0	0	0	0

Res

Read-modify-write operations on single flags of this register must be avoided. Unwanted clearing of other flags of this register may be the result otherwise.

GDM Good Morning
r0: No wake-up.
r1: Wake-up.
w0: Unaffected.
w1: Clear.
Is set by the BI when it is aroused from the sleep mode by a dominant bus level. The user must delete it.

CTOV Capture Time Overflow
r0: No overflow.
r1: Overflow.
w0: Unaffected.
w1: Clear.
Is set by the BI when the capture timer (CTIM) overflows. The user must delete it.

ECNT Error Counter Level
r0: No error counter.
r1: Error counter.
w0: Unaffected.
w1: Clear.
Is set by the BI as soon as the transmit error counter or the receive error counter exceeds a limit value. The user must delete it.

BIT Bit Error
r0: No bit error.
r1: Bit error.
w0: Unaffected.
w1: Clear.
Is set by the BI when a transmitted bit is not the same as the bit received. The user must delete the flag.

STF Stuff Error
r0: No stuff error.
r1: Stuff error.
w0: Unaffected.
w1: Clear.
Is set by the BI when 6 identical bits are received successively in one Tg. The user must delete it.

CRC CRC Error
r0: No CRC error.

r1: CRC error.
w0: Unaffected.
w1: Clear.
Is set by the BI when the CRC received does not coincide with the CRC calculated. The user must delete it.

FRM Form Error
r0: No form error.
r1: Form error.
w0: Unaffected.
w1: Clear.
Is set by the BI when an incorrect bit is received in a field with specified bit level (start of frame, end of frame, ...). The user must delete it.

ACK Acknowledge Error
r0: No acknowledge error.
r1: Acknowledge error.
w0: Unaffected.
w1: Clear.
Is set by the BI when there is no acknowledge for a transmitted Tg. The user must delete it.

CANxIDX Interrupt Index Register

	7	6	5	4	3	2	1	0	
r/w	Interrupt Index								
	1	1	1	1	1	1	1	1	Res

The interrupt index indicates the source of the interrupt. If a transmission was the cause of an interrupt, the interrupt index points to the corresponding telegram descriptor (CANxIDX = 0..253). If an error was responsible for the interrupt, the interrupt index designates the error status register (CANxIDX = 254). After dealing with the interrupt, the user must eliminate the cause of the interrupt and set the interrupt index to minus one (255 = EMPTY). As soon as CANxIDX is empty, the BI can enter a new index and initiate an interrupt. An interrupt can only be initiated when CANxIDX contains the value 255.

CANxIDM Identifier Mask Register

	7	6	5	4	3	2	1	0	
r/w	Identifier Mask Bits 4 to 0					x	x	x	3
r/w	Identifier Mask Bits 12 to 5								2
r/w	Identifier Mask Bits 20 to 13								1
r/w	Identifier Mask Bits 28 to 21								0
	0	0	0	0	0	0	0	0	Res

r/w0: Don't care.
r/w1: Compare.
The identifier mask register is 29 bits long; the MSB is in the MSB position in the lowest byte address. The CANxIDM defines a mask for the acceptance of address groups. Only the permitted bits are used for comparison with a received identifier. Whether the mask is used can be determined individually for each receive object.

CANxBT1 Bit Timing Register 1

	7	6	5	4	3	2	1	0	
r/w	MSAM	SYN	BPR						
	0	0	0	0	0	0	0	0	Res

MSAM Multi Sample
r/w0: Bus level is determined only once per bit.
r/w1: Bus level is determined three times per bit.

SYN Sync On
r/w0: Synchronization with falling edges only.
r/w1: Synchronization with rising edges too.

BPR Baud Rate Pre-scaler
r/w: Prescaler value.
The baud rate prescaler sets the length of a time quantum for the bit timing logic.

$t_Q = (BPR + 1) / f_0$.

With the 6-bit counter it is possible to extend t_Q by a factor of 1...64. Values from 0 to 63 are allowed.

0: $t_Q = 1 / f_0$
1: $t_Q = 2 / f_0$
2: $t_Q = 3 / f_0$
3: $t_Q = 4 / f_0$ etc.

CANxBT2 Bit Timing Register 2

	7	6	5	4	3	2	1	0	
r/w	rsvd	TSEG2			TSEG1				
	0	0	0	0	0	0	0	0	Res

TSEG2 Time Segment 2
r/i: TSEG2 value.
TSEG2 determines the number of time quanta after the sample point. Permitted entries: 1...7 (result in 2...8 TQ).

TSEG1 Time Segment 1
r/i: TSEG1 value.
TSEG1 determines the number of time quanta before the sample point. Permitted entries: 2...15 (result in 3...16 TQ).

CANxBT3 Bit Timing Register 3

	7	6	5	4	3	2	1	0	
r/w	rsvd	rsvd	rsvd	rsvd	rsvd	SJW			
	x	x	x	x	x	0	0	0	Res

SJW Synchronization Jump Width
r/i: SJW value.
SJW defines by how many TQs a bit may be lengthened or shortened because of resynchronization. Permitted entries: 1...4 (result in 1...4 TQ). Only the values 1, 2, 3 and 4 are allowed, other values result in unpredictable behavior.

CANxICR Input Control Register

	7	6	5	4	3	2	1	0	
r/w	rsvd	rsvd	rsvd	rsvd	rsvd	XREF	REF1	REF0	
	x	x	x	x	x	x	0	0	Res

XREF **External Reference (not avaliable)**
r/w0: The internal reference is used.
r/w1: The external reference is used where available (write to zero for future compatibility).

REF1 **Use Reference for RxD1**
r/w0: RxD is used as inverted input signal.
r/w1: Supply voltage is used as inverted input signal.

REF0 **Use Reference for RxD0**
r/w0: RxD is used as input signal.
r/w1: Ground is used as input signal.

CANxOCR Output Control Register

	7	6	5	4	3	2	1	0	
r/w	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	ITX	
	x	x	x	x	x	x	x	0	Res

ITX **Inverted transmission**
r/w0: Tx output is not inverted.
r/w1: output is inverted.

CANxTEC Transmit Error Counter

	7	6	5	4	3	2	1	0	
r	Counter Bit 7 to 0								
	0	0	0	0	0	0	0	0	Res

CANxREC Receive Error Counter

	7	6	5	4	3	2	1	0	
r	x	Counter Bit 6 to 0							
	x	0	0	0	0	0	0	0	Res

CANxESM Error Status Mask Register

	7	6	5	4	3	2	1	0	
r/w	EGDM	ECTV	EECT	EBIT	ESTF	ECRC	EFRM	EACK	
	1	1	1	1	1	1	1	1	Res

Every flag of the CANxESTR can be enabled/disabled generating an interrupt by modifying the corresponding flag in register CANxESM.

EGDM **Enable Good Morning**
r/w0: Disable.
r/w1: Enable.

ECTV **Enable Capture Time Overflow**
r/w0: Disable.
r/w1: Enable.

EECT **Enable Error Counter Level**
r/w0: Disable.
r/w1: Enable.

EBIT **Enable Bit Error**
r/w0: Disable.
r/w1: Enable.

ESTF **Enable Stuff Error**
r/w0: Disable.
r/w1: Enable.

ECRC **Enable CRC Error**
r/w0: Disable.
r/w1: Enable.

EFRM **Enable Form Error**
r/w0: Disable.
r/w1: Enable.

EACK **Enable Acknowledge Error**
r/w0: Disable.
r/w1: Enable.

CANxCTIM Capture Timer

	7	6	5	4	3	2	1	0	
r	Timer Bit 15 to 8								1
r	Timer Bit 7 to 0								0
	0	0	0	0	0	0	0	0	Res

The capture timer is incremented with a clock pulse derived from the CAN bus. Because it can only be read byte-wise, the low byte must be read first. The corresponding high byte is latched at the same time. When CANxCTIM overflows, the flag CTOV in the error status register is set. The capture timer will not be incremented during CAN module sleep mode (SLP = 1).

27.2.4. Communication Area (CA)

The CA is located in the CAN-RAM. It consists of communication objects, each of which is 16 bytes long. The CA begins at address 0 of the CAN-RAM with the first byte of a CO. It ends with the first byte of a CO which contains ones in its 3 MSBs (communication mode = 7 = EoCA). The following bytes can be used by the application. If the CAN-RAM is filled completely with COs, there is no place left and no need to mark the end of CA.

Every telegram which this node is to receive or transmit is represented by a CO. As well as the data and the time stamp, this also contains a header, the telegram descriptor (TD), in which the attributes of the communication object are stored.

The COs are entered in order of priority into the CA. This starts with the highest priority (the lowest identifier). The identifier defines the priority of a telegram. If the first eleven bits of an extended telegram are the same as the identifier of a standard telegram, the telegram with the standard identifier has higher priority.

27.2.4.1. Telegram Descriptor (TD)

The telegram descriptor is 6 bytes (TD0 to TD5) long and forms the beginning of a CO. Telegrams with standard and extended identifiers have different TDs. They differ only in the length of the identifiers. 18 bits therefore are not allocated in the TD of a standard telegram. They cannot be used by the application because they are overwritten by the reception of a telegram.

Extended Addr. Format (EXF is set)

	7	6	5	4	3	2	1	0
0	CM			RSC	MID	OW	rsvd	LCK
1	28			ID				21
2	20			ID				13
3	12			ID				5
4	4		ID			0	EXF	RSR	ACC
5	DLC				TIE	RIE	SR	TS

Standard Addr. Format (EXF is deleted)

	7	6	5	4	3	2	1	0
0	CM			RSC	MID	OW	rsvd	LCK
1	28			ID				21
2	20	ID	18	don't use				
3	don't use							
4	don't use					EXF	RSR	ACC
5	DLC				TIE	RIE	SR	TS

Fig. 27–3: Extended and standard TD map

Forms of access:
r: read
w: write
i: init (BI halted or CM = inactive)
w0: clear
w1: set

CM **Communication Mode**
r/i: Mode.
CM defines the type of telegram.

0: Inactive Inactive. No participation in the bus traffic.
1: Send Send data.
2: Receive Receive data.
3: Fetch Fetch data via remote frame.
4: Provide Have data fetched via remote frame.
5: Rx-All Receive every telegram.
6: rsvd Don't use (provis. EoCA).
7: EoCA End of Communication Area.

As long as the CO is inactive (CM = 0) or locked (LCK = TRUE), the BI accesses the first byte of the CO only by reading. All other bytes are neither read nor written. The inactive mode is suitable therefore for re-configuration of a CO online; i.e., while the node is taking part in the bus traffic.

RSC **Rescan**
r/w0: Don't rescan.
r/w1: Rescan.

If the rescan bit has been set in a transmit object just processed, the search for active Tx objects is started at the beginning of the communication area. Otherwise, the search continues at this transmit object until the end of the CA is reached. From there, the system jumps back to the beginning of the CA.

MID **Mask Identifier**
r/w0: Don't mask.
r/w1: Mask.
If MID is deleted, the identifier received is compared bit-by-bit with the identifier from the telegram descriptor, i.e., the entire identifier must be the same so that the telegram received is transferred into this CO. If MID is set, only bits which are allowed in the ID mask register of the GCS are used for the comparison.

OW **Overwrite**
r/w0: Don't overwrite.
r/w1: Overwrite.
When OW is set, the com. object may be overwritten even if the application has not yet fetched the contents (TS set). The BI must of course obtain right of access (LCK deleted).

LCK **Lock**
r/w0: BI has right of access.
r/w1: BI does not has right of access.
Lock determines the right of access for the BI.

ID **Identifier**
r/i: Identifier.
The ID contains the address of the telegram. 11 bits in the standard mode or 29 bits in the extended mode.

ACC **Access**
r/w0: CPU does not have right of access.
r/w1: CPU has right of access.
Access determines the right of access for the CPU. The CPU should not modify this flag after initialization. In operation mode only the BI modifies it and the CPU reads it.

RSR **Remote Send Request**
r/w0: Remote telegram received.
r/w1: Corresponding data transmitted.
In the provide mode, RSR signals a send request from outside; in the fetch mode it means that a remote telegram is being sent. It is set by the BI if a remote telegram has been received. It is deleted as soon as the corresponding data telegram has been transmitted.

EXF **Extended Format**
r/w0: Standard.
r/w1: Extended.
In order to send/receive telegrams with extended address format, this flag must be switched on. For standard telegrams it is deleted.

DLC **Data Length Code**
r/w: Data length.
The DLC defines the number of data bytes transmitted. Only telegrams with 0 to max. 8 data bytes are transmitted. If the DLC of a TxTg contains a value >8, the entered DLC and exactly 8 bytes will be transmitted. In the case of RxTgs the received DLC, and therefore also values > 8 will be entered by BI.

TIE **Tx Interrupt Enable**
r/w0: Disable.
r/w1: Enable.
Masks the Tx interrupt for this com. object.

RIE **Rx Interrupt Enable**
r/w0: Disable.
r/w1: Enable.
Masks the Rx interrupt for this communication object.

SR **Send Request**
r0: Successful transmission.
r/w1: Send request.
With SR, the microprocessor issues a send request. Both the microprocessor and the BI write the SR flag. If the microprocessor writes a one, the telegram is sent. The BI deletes the SR flag after successful transmission.

TS **Transfer Status**
r/w0: Ready for Transfer.
r/w1: Successful transfer.
The TS flag is set by BI after a successful transfer and is deleted by the microprocessor after a com. object has been processed.

27.2.4.2. Data Field

The data field consists of 8 Byte. They are filled with telegram data according to the DLC. Unused data bytes (DLC less than 8) can be used by the user.

27.2.4.3. Time Stamp

TIMST **Time Stamp**
r: Counter value.
The last two bytes in the CO are used for the time stamp.

At each SoF (Start of Frame) the free-running 16-bit counter CANxCTIM is loaded into a register. When the telegram has been correctly transmitted, this register is copied to the two time stamp bytes of the corresponding CO.

	Data 5
	Data 6
	Data 7
14	Time Stamp low
15	Time Stamp high

Fig. 27–4: Time stamp

27.3. Application Notes

27.3.1. Initialization

After reset, a CAN Module is in standby mode (inactive).

Prior to entering active mode, proper SW configuration of the U-Ports assigned to function as RX input and TX output has to be made (Table 27–1). The RX port has to be configured as special in and the TX port has to be configured as special out. Refer to “Ports” for details.

For entering active mode of a CAN, set the respective enable bit (Table 27–1).

In the initialization phase, a configuration of the CAN node takes place. The mode of operation of the BTL and the bus coupling is set. The communication area is created in the CAN-RAM. The different telegrams are specified in it.

The CAN node must be halted (HACK = TRUE) to carry out the initialization. After a reset, the flags HLT and HACK are set and initialization can take place. If initialization is required on-line, the flag HLT must be set. However, the BI must terminate any current transmission before it comes to a halt. For the user this means that he must wait until HACK has been set. If HLT is deleted after initialization, then BI begins to participate in the bus traffic and to scan the CA for tasks.

During initialization, the error status register (CANxESTR) and the interrupt index (CANxIDX) should be deleted, otherwise no interrupts can be initiated. The error status mask register default value after reset is not masked.

If telegrams with different identifiers are to be received in a single CO, the identifier mask register must be initialized. This defines which bit of the ID received must be the same as the ID in the CO.

Bit timing registers 1, 2 and 3 and the output control registers 1 and 2 must be initialized in all cases.

Table 27–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
CAN0	CAN0-RX input multiplexer	PM.U20	CAN0-RX	U2.1 or U4.3 special in	SR0.CAN0
	CAN0-TX output multiplexer		CAN0-TX	U2.0 or U4.2 special out	
CAN1			CAN1-RX	U6.1 special in	SR0.CAN1
			CAN1-TX	U6.0 special out	
CAN2			CAN2-RX	U8.5 special in	SR0.CAN2
			CAN2-TX	U8.4 special out	
CAN3			CAN3-RX	U8.3 special in	SR0.CAN3
			CAN3-TX	U8.2 special out	

The CA must be created in the CAN-RAM. The different COs are created one after the other starting at the address 0. It is important at this point that the three MSBs have been set in the first byte after the last CO, i.e., at an address divisible by 16 (CM = End of CA). This is not necessary if the CAN-RAM is completely filled with COs.

Communication mode (CM), identifier, data length code, extended format flag (EXF) and remote send request flag must be initialized in each CO. Lock flag (LCK) must be deleted and access flag (ACC) must be set in order that the BI may also view this CO. Transfer status flag (TS) must be deleted so that interrupts are not initiated erroneously.

27.3.2. Handling the COs

27.3.2.1. Principles

If the user wishes to access a CO, then he must lock out the BI from access to it. Also the BI reserves access for itself to one CO. In this case the user may not have access. When scanning the CA, the BI ignores inactive or locked COs; i.e., it reads only the first byte and then jumps to the next CO.

Reservations Procedure
If the user would like to access a com. object, then he must first set LCK. Then he must read ACC. If it is TRUE, he has right of access. After the operation he must delete LCK.

```
LCK = TRUE;
if (ACC == TRUE)
    {
    /* CPU has right of access */
    }
LCK = FALSE;
_____________ or _____________
LCK = TRUE;
while (ACC == FALSE)
    {
    /* wait until BI is ready */
    }
/* CPU has right of access */
LCK = FALSE;
```

Fig. 27–5: Access to a CO by the user

When the BI is accessing a com. object, it first deletes ACC and then reads LCK. If LCK is FALSE, it has right of access.

```
ACC = FALSE;
if (LCK == FALSE)
    {
    /* BI has right of access */
    }
ACC = TRUE;
```

Fig. 27–6: Access to a CO by the BI

The BI does not wait at a CO until it becomes free.

The BI scans the CA from beginning to end. After a TxTg has been transmitted, the next TxTg entered is reported ready to send.

It makes sense to enter the COs in the CA in order of their priority. The priority is determined by the ID. The lowest ID has the highest priority. If the first bits of an extended ID are identical with a standard ID, the standard ID has higher priority. The CO with the highest priority is at the beginning of the CA. This ensures that Tx-Tgs with high priority are transmitted first when a rescan is initiated.

27.3.2.2. Configuration

A CO may be configured only in the inactive and/or locked mode or when HACK has been set. Otherwise it can lead to access conflicts between the user and BI.

The communication mode (CM) is determined in the configuration phase. The identifiers are also entered. The flag EXF must not be overlooked. The flag RSR and DLC determine whether and how many data bytes will be transmitted in the telegram. The interrupts can be permitted. In case of a receive telegram it is necessary under certain circumstances to set the flags MID and OW. In case of a transmit telegram, the flag RSC must be adjusted.

27.3.2.3. Transmit Telegram

CM = Send

A transmit telegram is used to send data. How many data bytes will be sent is fixed in the DLC. The data is entered directly after the TD. Unused data bytes can be freely used by the user. If after the transmission of this telegram the user would like the next Tx-Tg in the CA to be sent, he deletes the RSC flag. If he sets the RSC, then the transmit search starts again at the beginning of the CA. The RSR flag has to be deleted.

The set SR flag tells BI that this telegram is to be sent; SR can be likened to a postage stamp. The TS flag must be deleted before the CO is released with the deletion of LCK.

If the BI finds a CO whose SR flag has been set, it reserves this (ACC = FALSE) and reports it ”ready to send”. It will be transmitted as soon as no higher-priority telegrams occupy the bus. After successful transmission, it deletes the flag SR and sets TS. The setting of ACC re-releases the CO. Whether an interrupt will be triggered depends on whether CANxIDX in the GCS contains the value minus one (255) and transmit interrupts are permitted.

The user should now reserve the CO, reset the flag TS and delete CANxIDX so that other interrupts can also be reported. Should he wish to send further data, he can now enter this.

27.3.2.4. Receive Telegram

CM = Receive

With a receive telegram, data is received. If the EXF flag and the unmasked bits of the identifier of a received telegram are the same as those of a receive CO, the telegram will be copied to the CO. ID, DLC and data bytes are overwritten by the received ID, DLC and data. Only as many data bytes as the received DLC specify will be overwritten (max. 8). The DLC actually received will be entered. A permitted receive CO is only used when TS has been deleted or OW has been set.

Once a telegram has been received and copied to a CO, the flag TS is set. An interrupt will also be initiated if receive interrupts are permitted and CANxIDX contains the value minus one (255).

If the user detects the reception of a telegram (TS set), he must reserve the CO. Then he can read the data and, before releasing the CO again, delete TS.

27.3.2.5. Receive All Telegrams

CM = Rx-All

If, while searching for an RX-CO, the BI comes across a free Rx-All-CO, the received telegram will be entered here without regard to ID and EXF.

Rx-All-COs should be applied at the end of the CA.

27.3.2.6. Fetch Telegram

CM = Fetch

A fetch CO is used to request data from another node. This is done by sending a telegram with the identifier of the desired data. The remote transmission request flag is set in this Tg. No data is therefore sent with it. If another node has the desired data available, this is transmitted with the same ID as soon as bus traffic allows.

In this mode, only the reception of the data telegram can trigger an interrupt.

The sequence of a fetch cycle is represented for the user in pseudo-code.

```
if (TS == FALSE && SR == FALSE)    /* CO is empty */
  {
  LCK = TRUE;                 /* claim CO */
  /* wait until BI released this CO */
  while (ACC == FALSE) {/* do anything else */}
  SR = TRUE;                 /* send this Tg */
  TS = FALSE;
  LCK = FALSE;                /* release CO */
  }
```

The BI now transmits the telegram with the RTR flag set. The other node receives the Tg, provides the data and returns the telegram with RTR flag deleted. After the reply telegram has been received, the BI sets the flag TS. The user waits for the data.

```
/* wait for answer */
while (TS == FALSE) {/* do anything else */}
LCK = TRUE;           /* claim CO */
/* wait until BI released this CO */
while (ACC == FALSE) {/* do anything else */}
/* copy data */
TS = FALSE;
LCK = FALSE;          /* release CO */
```

Instead of waiting for the answer, it is also possible for notification to be given by a receive interrupt.

27.3.2.7. Provide Telegram

CM = Provide

A provide CO is used to prepare data for fetching. It is the counterpart of a fetch CO. In a provide CO the RSR flag is cleared. It will be set and deleted by the BI. The data can be prepared in two ways:

In the first case, the user does not become active until a remote frame has been received (Rx interrupt or polling from RSR). After the CO has then been reserved, the data is written, the SR flag is set and the CO is released. The BI ensures then that the data is transferred back.

In the second case, the data has already been entered, SR has been set and TS deleted before the request. When the remote frame is received, the user does not need to become active. Also, no Rx interrupt will be initiated. The data is simply fetched. In this case the requesting RTR telegram must contain the correct DLC because, with an RTR telegram too, a received DLC overwrites the local DLC.

In both cases a Tx interrupt can occur after the data telegram has been transmitted.

27.3.2.8. Data Length Code

The data length code is 4 bits long. It can therefore contain values between 0 and 15. In principle, no more than 8 bytes can be transmitted. Empty data telegrams (DLC = 0) are also possible.

If a telegram with a DLC greater than 8 is received, this value will be written into the DLC of the CO, but exactly 8 bytes of data will be copied.

If the DLC of a Tx-CO contains a value greater than 8, this DLC will be transmitted, but only 8 bytes of data.

27.3.2.9. Overwrite Mode

The BI normally processes a CO only when the transfer status TS has been deleted; i.e. the user has processed the CO since the last transmission. In the case of COs with which telegrams are received, the TS flag can be by-passed. If overwrite (OW) is permitted, the BI may overwrite a previously received telegram. When accessing data therefore, the user always receives the most up-to-date data.

27.3.3. Interrupts

All interrupts are enabled or disabled by the global interrupt enable flags, GTIE for Tx interrupts, GRIE for Rx interrupts and EIE for error interrupts in the CANxCTR register. Each error interrupt can also be masked individually in the Error Status Mask register. A Tx interrupt can be enabled in the corresponding CO with the Tx interrupt enable flag TIE. An Rx interrupt can be enabled in the corresponding CO with the Rx interrupt enable flag RIE.

An interrupt can only be initiated when the interrupt index CANxIDX is empty (minus one). To initiate an interrupt, the BI enters the number (0...253) of the appropriate CO in the CANxIDX. When an error interrupt is involved, the number 254 is entered.

The BI attempts to initiate an interrupt immediately after successful transfer. If this does not work (CANxIDX not empty), the interrupt is pending (also error interrupt).

The BI permanently scans the CA. If, while doing so, it finds a CO whose interrupt condition is satisfied (e.g. TIE and TS are set), it generates an interrupt. This means that interrupts not yet reported will not be reported in the sequence of their occurrence, but in the sequence in which they are discovered later.

The interrupt service routine of the user must read the CANxIDX. The interrupt source is stored here. If CANxIDX points to a CO (0...253), the user must reserve this. After this, he must first delete TS so that this CO does not initiate an interrupt again. Only then he may release CANxIDX (CANxIDX = 255) so that the BI can enter further interrupts.

27.3.4. Rescan

The normal transmit strategy searches for the next transmit CO in the CA. If all the transmit COs are ready to send, they are processed one after the other. This is a democratic strategy.

If higher-priority TxTgs are reported in the meantime, these are not processed until the complete list has been finished. With rescan, the search for Tx telegrams is started again at the beginning of the CA. By this means the user can force the normal strategy to be interrupted and a search to be made first of all for higher-priority TxTgs. A transmit CO already reported will of course be transmitted first.

The rescan requirement can be achieved dynamically, when a transmit CO is reported, by setting the global rescan flag GRSC.

It is also possible to configure a rescan strategy statically. Each Tx-CO has the rescan flag RSC. If it is set, the system starts from the beginning with the transmit search after this CO has been processed. It is possible, for instance, to set RSC in the low-priority Tx-COs. Each time a low-priority Tx-CO has been handled, the search continues for higher-priority objects.

The user must ensure that each Tx-CO is processed.

27.3.5. Time Stamp

The time stamp of a CO shows the user how much time has elapsed since the transmission of the object. For this purpose, he compares the time stamp with the capture timer CANxCTIM. Because the time stamp contains the value of the CANxCTIM at the time of the start of transmission, the difference is proportional to the time which has elapsed.

The time stamp mechanism also enables network-wide synchronization. A master transmits a Tg. All nodes note the transmission time (local time). Then the master transmits its own (global) transmission time. The difference between local and global time shows by how much one's own clock (timer) is wrong.

27.3.6. Errors

In the error status register (CANxESTR) error messages and status data are collected which can generate an error interrupt. As long as a flag is set in the CANxESTR and not masked in the CANxESM, the flag ERS is also set in the status register. This means that the value 254 is written in CANxIDX and an interrupt is generated when EIE has been set.

An error interrupt is deleted by first deleting CANxESTR and then releasing CANxIDX.

The 5 flags BIT, STF, CRC, FRM and ACK originate from the protocol manager. The flag GDM (Good Morning) is not an error flag. GDM is set when the BI is aroused from the sleep mode by a dominant bus level.

The flag ECNT (error counter level) indicates that an error counter has exceeded a limit value. It is set when the transmit error counter exceeds the values 95, 127 and 255 or the receive error counter exceeds the values 95 and 127.

When the BI is in the Bus-Off mode, it no longer actively participates in the bus traffic. Nor does it receive telegrams, but continues to observe the bus. As soon as the BI has detected 128 x 11 successive recessive bits, it either reverts to the error-active mode if flag BOST is zero, or it sets the flag HLT and enters the HALT mode if flag BOST is set. At the same time the error counters are cleared.

A Bus-Off sequence triggers two interrupts, if the error interrupt is enabled. The first interrupt (ECNT=TRUE) indicates that the transmit error counter has exceeded the value 255. This means that the module is in the Bus-Off mode now (BOFF=TRUE). The receive error counter is used to count the reception of 128 x 11 successive recessive bits in the Bus-Off mode. This is the reason for the second interrupt (ECNT=TRUE), which indicates that the receive error counter has exceeded the value 95 (warning level). The second interrupt can be ignored in Bus-Off mode. The error interrupt can be disabled during Bus-Off mode to avoid this second interrupt.

27.3.7. Layout of the CA

The CA contains all COs beginning with the lowest identifier. The three MSBs must be set in the byte after the last CO (End of CA).

If the BI has received an identifier complete, it starts at the beginning of the CA with the search for an appropriate Rx-CO. If a rescan is initiated, the BI also starts from the beginning with the transmit search.

27.3.7.1. Buffers

Several successive receive COs may be allocated with the same identifier. The BI stores a received Tg in the first free Rx-CO. Using this mechanism it is possible to construct a receive buffer. If RIE is set in the last CO, the CPU is not informed until the buffer is full.

27.3.7.2. Basic/Full CAN

For a Basic CAN application, a single Tx-CO will be used. All outgoing telegrams will be transmitted with this. The user must receive all Rx-Tgs and must himself decide whether he needs it (acceptance filtering). For this case it is possible to use an Rx-All-CO. But it is necessary to ensure that this can be processed before the next Tg arrives.

For this reason, it is a good idea to employ 2 or 3 Rx-All-COs as buffers after the Tx-CO.

In the case of a FullCAN application, the built-in acceptance filtering can be used and a dedicated CO can be set up for each desired Rx-Tg and Tx-Tg.

If the CAN-RAM is not big enough, mixed strategies are also possible. The acceptance filtering, of course, burdens the CPU with communication tasks.

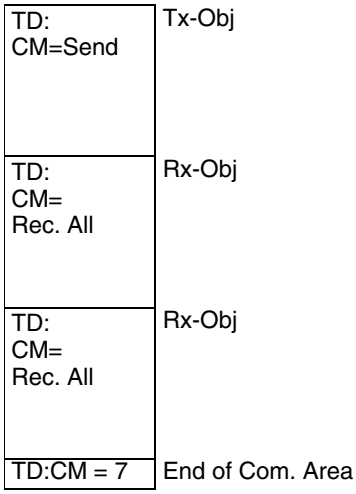


Fig. 27–7: Example: CA of a BasicCAN with 2 Rx-buffers

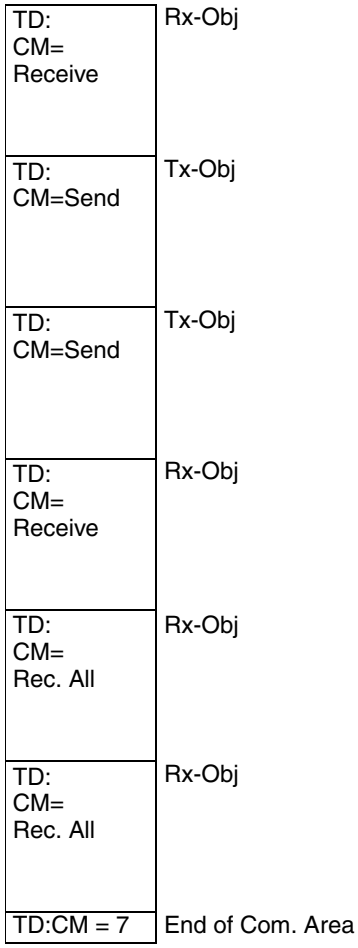


Fig. 27–8: Example: CA of a FullCAN with 2 Rx-objects, 2 Tx-objects, and 2 Rx-buffers

TD: CM=Send	Tx-Obj
TD: CM= Rec. All	Rx-Obj
TD: CM= Rec. All	Rx-Obj
TD: CM= Rec. All	Rx-Obj
TD: CM= Rec. All	Rx-Obj
TD:CM = 7	End of Com. Area

Fig. 27–9: Example: CA of a basicCAN with 4 Rx-buffers

27.3.7.3. Bus Monitor

With some Rx-All-COs it is possible to construct a user-friendly bus monitor. The CPU merely has to observe whether anything has been received. The contents of the CO must be stored. The transmission time can be calculated from the time stamp.

27.3.7.4. Maximum number of COs

The maximum number of COs depends on the size of the CAN-RAM, the baud rate, the system clock, the BI and the CPU accesses to the CAN-RAM.

– The BI can handle a maximum of 254 objects. The limiting factor is the 8-bit register CANxIDX in the GCS. CANxIDX can contain 256 different values. The values 255 (empty) and 254 (error) are reserved. The remaining values 0...253 can indicate 254 objects.

– The maximum number of COs is, of course, limited to a greater extent by the size of the CAN-RAM. The BI can only access the CAN-RAM. Therefore the CA can only be applied there.

  16 bytes are reserved for each CO. One extra byte for coding EoCA after the last CO must not be forgotten. The CAN-RAM area after the EoCA is freely available to the user. No EoCA is necessary if the CAN-RAM is filled completely with COs.

  A maximum number of 32 COs is possible in a CAN-RAM of 512 bytes.

$$\text{Max. Number CO} = \frac{\text{CAN RAM Size}}{16}$$

– The next limiting factor can be calculated from the baud rate and system clock. After the BI has received an identifier, it must be possible for it to scan the entire CA before the telegram comes to an end.

$$\text{Max. Number CO} = \frac{t_{CA\ SCAN}}{t_{CO\ SCAN}}$$

$$t_{CO\ SCAN} = \frac{9}{f_0}$$

$$t_{CA\ SCAN} = 28\ t_{Bit}$$

$$t_{Bit} = (3 + TSEG1 + TSEG2)\ t_Q$$

$$t_Q = \frac{BPR + 1}{f_0}$$

$t_{CA\ Scan}$ is the time from having received an ID to the end of a minimum telegram (11-bit ID, no data), which is at the BI's disposal to scan the CA.

$t_{CO\ Scan}$ is the worst case time needed by the BI to process an object (A value of 6 I/O cycles is a more realistic size than 9).

With an input frequency of 8 MHz and a baud rate ($1/t_{bit}$) of 1 MBd, the BI could handle 24 COs. Naturally, this value needs to be rounded off.

– The value thus calculated is further limited, however, by the CPU accesses to the CAN-RAM. Each I/O cycle required by the CPU to write or read data in the CAN-RAM is missing from the BI. The BI is halted by CPU accesses. This reduces the time which the BI has to scan the CA. Where there is a reduced CPU clock, in particular, the user should have only limited access to the CAN-RAM.

  With the ARM CPU accessing CAN RAM, it is easy to block the BI's CAN-RAM access over a long time. The ARM CPU can make a memory access with each I/O cycle, leaving nearly no I/O cycles to the BI.

  In the above example (8 MHz, 1 MBd), $t_{CA\ Scan}$ lasts 224 I/O cycles (28×8). The BI needs 144 I/O cycles to scan 16 COs leaving 80 I/O cycles to the CPU to process a telegram. It is not necessary to process more than one telegram during transmission of one telegram. As long as the COs are managed via interrupt, 80 I/O cycles should be more than enough to read or write a CO.

  In this worst case scenario the BI needs 288 I/O cycles to scan 32 COs. This is possible at an input frequency of 8 MHz and up to baud rate of 500 kBd. In a more realistic estimation (average $t_{CO\ Scan} = 6$) the BI needs 192 I/O cycles to scan 32 COs leaving 32 I/O cycles to the CPU to process a telegram. This means 1 MBd is possible even with 32 COs, as long as the COs are managed via interrupt only.

  Due to this, care has to be taken when using free CAN-RAM (after EoCA). It is not possible here, to make an assumption about how many accesses a non-CAN routine makes to its data storage.

27.4. Bit Timing Logic

In the bit timing logic the transmission speed (baud rate) and the sample point within one bit will be configured. By shifting the sample point it is possible to take account of the signal propagation delay in different buses. Furthermore, the nature of the sampling and the bit synchronization can also be defined.

27.4.1. Baud Rate Pre-scaler

The baud rate pre-scaler is a 6-bit counter. It divides the system clock down by the factor 1...64. The output is the clock for the bit timing logic. This clock TQ_{CLK} defines the time quantum (t_Q). The time quantum is the smallest time unit into which a bit is subdivided.

27.4.2. Bit Timing

A bit duration consists of a programmable number of TQ_{CLK} cycles. The cycles are split up into the segments SYNCSEG, TSEG1 and TSEG2.

27.4.2.1. Bit Timing Definition

Sync.Seg.
It is expected that a bit will begin in the synchronization segment. If the bit level changes, the resynchronization ensures that the edge lies inside this segment. The sync.seg is always one time quantum long.

Prop.Seg.
This part of a bit is necessary to compensate for delay times of the network. It is twice the sum of the signal propagation delay on the bus plus input comparator delay plus output driver delay.

Phase Seg.
Phase segments 1 and 2 are necessary to compensate phase differences. They can be lengthened or shortened by resynchronization.

Sample Point
The bus level is read at this point and interpreted as a received bit.

TSEG1
The CAN implementation combines propagation delay segment and phase segment 1 to form time segment TSEG1.

TSEG2
TSEG2 corresponds to phase segment 2.

SJW
The synchronization jump width gives the maximum number of time quanta by which a bit may be lengthened or shortened by resynchronization.

$$t_{Bit} = t_{SYNCSEG} + t_{TSEG1} + t_{TSEG2}$$

$$t_Q = \frac{BPR + 1}{f_0}$$

$$t_{SYNCSEG} = 1\ t_Q$$

$$t_{TSEG1} = (TSEG1 + 1)\ t_Q$$

$$t_{TSEG2} = (TSEG2 + 1)\ t_Q$$

$$t_{SJW} = SJW\ t_Q$$

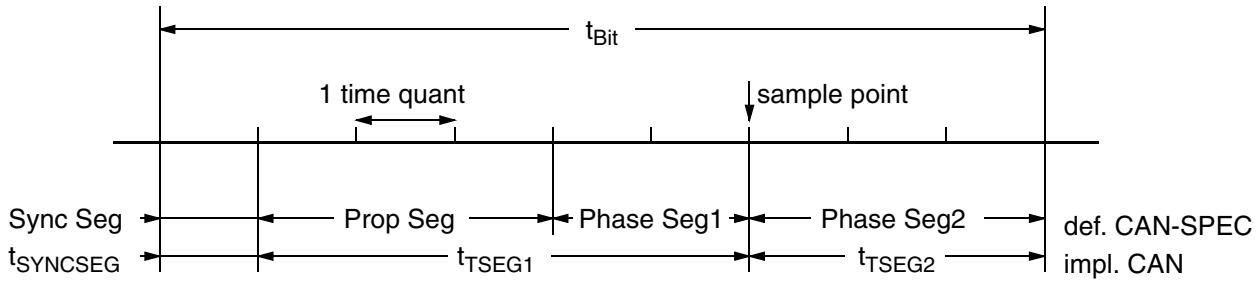


Fig. 27–10: Bit timing definition

The baud rate is then calculated as follows:

$$BR = \frac{1}{t_{Bit}}$$

$$t_{Bit} = \frac{(BPR + 1)\ (3 + TSEG1 + TSEG2)}{f_0}$$

$$BR = \frac{f_0}{(BPR + 1)\ (3 + TSEG1 + TSEG2)}$$

27.4.2.2. Bit Timing Configuration

Certain boundary conditions need to be observed when programming the bit timing registers. The correct location of the sample point is especially important with maximum bus length and at high baud rate.

$t_{TSEG2} \geq 2\, t_Q$ = Information Processing Time

$t_{TSEG2} \geq t_{SJW}$

$t_{TSEG1} \geq 3\, t_Q$

$t_{TSEG1} \geq t_{TSEG2}$

$t_{TSEG1} \geq t_{PROP} + t_{SJW}$

The information processing time is the internal processing time. After reception of a bit (sample point) this time is needed to calculate the next bit for transmission.

With a baud rate of 1 MBd a bit should be at least 8 t_Q long.

In the case of a triple sample mode (MSAM = 1), the following boundary condition must also be observed:

$t_{TSEG1} \geq t_{PROP} + t_{SJW} + 2t_Q$

The triple sample mode offers better immunity to interference signals. In the single sample mode a higher transmission speed is possible.

For high baud rates and maximum bus length, neither SYN nor MSAM may be switched on. Bosch advises against both adjustment facilities. When an input filter matched to the baud rate or a bus driver is used, the triple sample mode is not necessary. If SYN is set, synchronization will also be made with the soft edge (dominant to recessive) and this will mean higher demands being imposed on the clock tolerances.

27.4.2.3. Synchronization

The BTL carries out synchronization at an edge (change of the bus level) in order to compensate for phase shifts between the oscillators of the different CAN nodes.

27.4.2.4. Hard Synchronization

Hard synchronization is carried out at the start of a telegram. The BTL ensures that the first negative edge is in the sync. seg.

27.4.2.5. Resynchronization

Resynchronization takes place during the transmission of a telegram. If the BTL detects an edge outside the sync. seg., it can lengthen or shorten the bit. If it detects the edge during TSEG1, t_{TSEG1} is lengthened. If it detects the edge during TSEG2, t_{TSEG2} is shortened. In this way, it ensures that the edges lie in the sync. seg. T_{SJW} is the maximum time a bit can be lengthened or shortened.

Two forms of resynchronization are possible. In normal operation, synchronization is carried out only with the negative edge (recessive to dominant). At low transmission speeds, synchronization can also be carried out with the rising edge (SYN = 1).

27.5. Bus Coupling

The bus coupling describes the connection of the internal signals rx (receive line) and tx (transmit line) to the pins to the CAN bus.

The output pins are push/pull drivers for TLL levels. The input pins are also designed for TTL levels.

Integrated transceivers (Siliconix Si9200, Philips 82C250 etc.) are available for physical coupling in the high-speed range in compliance with ISO/DIS 11898.

For a laboratory system a “minimum bus” can be constructed by means of a wire-Or circuit.

To utilize the advantages of differential signal transmission, an analogue comparator is necessary.

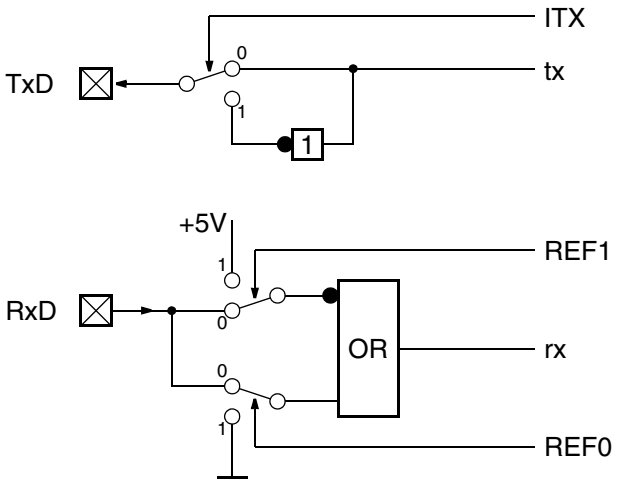


Fig. 27–11: Bus coupling

Table 27–2: Logical level transmitting

ITX	tx	TxD	Bus Level	Remarks
0	0	0	Dominant	direct
0	1	1	Recessive	
1	0	1	Recessive	inverted
1	1	0	Dominant	

Table 27–3: Logical level receiving

REF1	REF0	RxD	rx	Bus Level	Remarks
0	0	x	1	Don't work	
0	1	0	1	Recessive	inverted
0	1	1	0	Dominant	
1	0	0	0	Dominant	direct
1	0	1	1	Recessive	
1	1	x	0	Don't work	

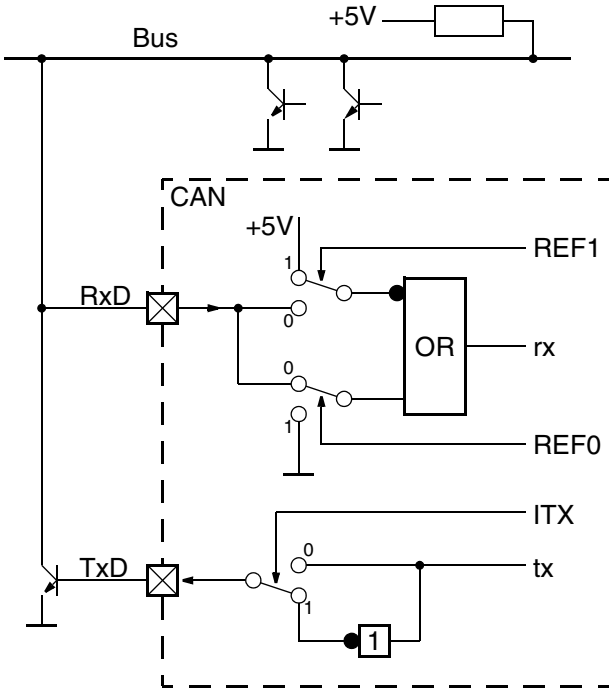


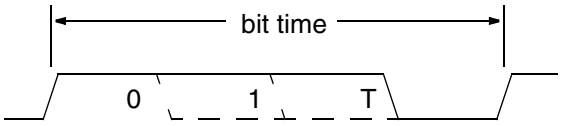
Fig. 27–12: Minimum bus

28. DIGITbus System Description

28.1. Bus Signal and Protocol

The DIGITbus is a single line serial master-slave-bus that allows clock recovery from the sign stream. Data on the bus are represented by a pulse width modulated signal. There are three different signs:

“0”: 25% High Time
“1”: 50% High Time
“T”: 75% High Time



A permanent high bus (100% High Time) means the bus is passive high. The bus is active if there are consecutive “T” signs, ones or zeros.

A permanent low bus (0% High Time) is interpreted as bus reset or failure indicator. Reasons may be shorts or opens or even a low level forced by a bus node to indicate an internal failure or reset condition.

The sign “T” is used to provide a system wide clock for the bus nodes and to separate the address and data fields and consecutive telegrams.

A telegram normally consists of an address and a data field separated by one “T”. These fields may be as long as necessary. Thus the length of an address or data field may carry information. The end is marked by a “T”. The end of a telegram is marked by two “T” signs.

One system implementation may be confined to certain address and/or data field lengths, thus reducing the hardware or software requirements.

The transmitter of an address has to guarantee that the address is preceded by four “T” signs at least.

An isolated data field is not possible. Each non “T” sequence, which is preceded by two or more consecutive “T” signs must be interpreted as an address. An address field is valid after the reception of the following “T”. The minimum address length is one bit. The minimum data field length is zero bit.

Telegrams with more than one data field are allowed, too. For instance TTTTAAATDDDDTDDDDDTT is a valid telegram format on the DIGITbus.

A telegram consisting of an address only is possible, too. The length of the data field is zero in this case.

A data field is preceded by an address field and separated from this by a single “T”. It is followed by one “T” sign. After reception of two “T” signs the telegram is finished and valid.

In the idle phase (no information exchange) of the bus traffic, only the bus clock is transmitted.

After the reception of two consecutive “T” signs all bus nodes have to be prepared to receive a new telegram starting with an address field. They are ready to send an address after the reception of four consecutive “T” signs.

The modification of a “T” sign to a zero or one is done by pulling the bus line to low (dominant state) at the right time. This is done by a master sending an address or a data bit or by a slave sending a data bit.

In case of reading data from a slave, the master first sends the address. After receiving the address the slave waits one “T” sign and then modifies the following “T” signs to zeros and ones which the master can recognize.

Slaves do not have the possibility to become active on the bus if they want to communicate a local event or if they need data from a master. It is a polling bus. Only a master is able to send an address. The master has to scan the slaves for their data. But it is possible to transfer data from one slave directly to another slave. The master has to transmit an address for which one slave is the source and the second slave is the destination. Telegrams on the bus are broadcast. Each bus node may receive them.

28.2. Other Features

There are two possibilities for a slave to signal a local event to the master. They are called wake-up and bus reset.

28.2.1. Wake-up

If the DIGITbus is passive high (permanent high level for more than one bit period) a slave may pull the bus line to low level. This will awake the master who has to store this event in a flag, to start the bus clock and to scan the bus for the source of this event. The minimum low time of the reset pulse is 1/16 of the nominal bit time (1/baud rate).

28.2.2. Bus Reset

The rising edge of a bit or bus clock is only controlled by the bus node which generates the bus clock (clock master). No other bus node may hold down the bus line at that moment. When the clock master releases the bus line at the end of a bit, he must watch the bus line. If the bus level does not rise after at least 1/2 bit time, this must be interpreted as a protocol violation. Delay of 1/2 of a bit time is the latest moment for a master. He can indicate this protocol violation if the rising edge is delayed 1/8 bit time. Slaves may use this mechanism to signal an exception to the master. They must pull down the bus for at least 2 bit times. After such an event normal communication may be impossible until the PLL of bus nodes have synchronized again.

28.2.3. Phase Correction

On a physical bus the signal edges may be delayed by the bus load. An extra delay may be added by different trigger edges. The bus nodes see the edges at different times. This causes them to pull the bus line delayed. To compensate this effect, the phase correction mechanism allows the bus node to adjust their internal counters.

The master sends a special address to which the slave answers with a single zero. The master measures the time between the rising and the falling edge. With this value he can calculate a phase correction value and transmit it to the slave. The slave may use it to adjust his internal counter.

The phase correction has to be done for each bus node separately.

28.2.4. Abort Transmission

The Abort Transmission feature is an option that allows the implementation of some kind of rip cord with the DIGITbus. On an alarm event, the SW of the sending master bus node may break the current telegram and send another telegram instead. The reception of an address/data field can not be stopped. The transmission of the alarm telegram is delayed until after the end of the reception in this case. Only the actual sending bus node can abort the transmission.

28.3. Standard Functions

The following standard functions have to be included in every DIGITbus implementation.

28.3.1. Send Bus Clock

The Bus Clock is the sequence of “T” signs on the DIGITbus. The rising edges of the bus signal are of constant distance. Only one bus node may generate this Bus Clock even in a multimaster system. All bus nodes use this stream of “T” signs to generate telegrams. The bus clock generator knows two states. “Active Bus” means the transmission of the Bus Clock. “Passive Bus” means permanent high bus level. “Passive Bus” may be a low-power mode.

28.3.2. Receive Bus Clock

Bus nodes which do not generate the bus clock need an internal clock for their operation. They may use a separate clock source or derive their clock from the bus clock by a PLL. Bus nodes which use their own clock sources nevertheless have to synchronize on the bus clock if they want to transmit or receive data.

28.3.3. Send Address

The address is the first bit field in a telegram. Only a master may send this field. The sender must guarantee, that at least two consecutive “T” signs have been visible on the DIGITbus before sending this field. Therefore, he has to send four “T” signs. If one of those four transmitted “T” signs is disturbed, only one of the separated telegrams is corrupted for a receiver. Sending of an address requires synchronization on the bus clock and, in case of a multimaster system, collision detection and arbitration capability.

28.3.4. Receive Address

Every slave and all multimaster-capable bus nodes must be able to receive an address. For a receiver, a valid address field must be preceded by two consecutive “T” signs. To verify a received address, it is not sufficient to compare the value. The length of the address must be correct too because of the arbitrary length of the address field.

28.3.5. Send Data

Every master must and some slaves are able to send a data field. A data field is preceded by an address or data field and one “T” sign.

28.3.6. Receive Data

Every master must and some slaves are able to receive a data field. A data field is preceded by an address or data field and one “T” sign. It is a good idea to verify the length of a received data field if possible. But variable length data fields are possible, too.

28.3.7. Collision Detection

Collision detection together with arbitration is necessary in multimaster systems. It is necessary to avoid the disturbance of telegrams if two masters try to send a telegram at the same time. As long as both transmit the same sign (one or zero) at the same time, they don’t detect a collision. If one master is sending a one and the other is sending a zero, a zero will be seen at the bus. In this case the master whose one was modified to the zero stops immediately sending. He should receive this telegram.

The sender has to arbitrate his part of the telegram.
Write telegram: TTTTT**AAAATDDDDT**TTTT
Read telegram: TTTTT**AAAAT**DDDDTTTTT
The separator (T sign) after an address or data field is object of arbitration too.

In a single master system arbitration loss has to be managed as a bus error.

28.4. Optional Functions

The following optional functions may be designed into a certain DIGITbus implementation.

28.4.1. Abort Transmission

A master who is controlling the transmission of a telegram can abort the sending of the address and data field. After four “T” signs after the last bit he can send another, more urgent telegram. If he is receiving a data field from a slave, he must wait until the slave has finished the data field. Then he can insert a new telegram.

28.4.2. Measure Pulse Width

The capability to measure the pulse width of a high pulse at the DIGITbus may be used for a phase correction by some bus nodes. The bus node who generates the bus clock, sends a data read telegram to another bus node. The other bus node answers with a data field which consists of a single zero. The pulse width of this zero is measured by the master. With this value he can calculate a phase correction value and transmit it to this bus member, which may adjust its time slots to the system dependencies.

28.4.3. Correct Phase

Bus nodes which do not generate the bus clock may use the above described procedure to adjust their phase. They have to answer to a special address with sending back a zero. Afterwards they will receive with another special address a correction value. With this value they can adjust the point where they pull the bus line to modify a “T” to a one or a zero.

28.4.4. Generate Wake-up

If the DIGITbus is passive high (no bus clock, always high level), the clock master may be wake up by pulling the bus level to low (dominant state) for 1/16 bit time at least. All nodes without the clock master may be able to do that.

28.4.5. Receive Wake-up

If there is a low pulse of at least 1/64 bit time on a passive high DIGITbus, the clock master must start to transmit the bus clock by sending “T” signs. All Masters with a bus clock generation unit must be able to do so in a system who uses this feature.

28.4.6. Generate Reset

During active DIGITbus a slave may be allowed to pull down the bus line longer than up to the end of the actual bit time (2 bit times at least). The rising edge at the end of the bit will be delayed in this case. This will disturb the bus clock for all bus nodes.

28.4.7. Receive Reset

The clock master is generating the rising edge at the end of a bit time. He will detect the above described reset condition and set a flag if the rising edge is delayed for at least 1/8 of the bit time.

29. DIGITbus Master Module

The DIGITbus is a single-line serial master-slave-bus that allows clock recovery from the sign stream. The address and data field are of arbitrary length.

The DIGITbus master module is a HW module for connecting a single-chip controller to the DIGITbus. It generates the bus clock and manages short telegrams autonomously. Transmission and reception of long telegrams is supported by a FIFO each. The DIGITbus master may be used in a single or in a multimaster bus system.

Features

- Single master in a single-master system.
- Clock master in a multimaster system.
- Passive master in a multimaster system.
- Bus clock generation.
- Receive and transmit a telegram with address and data field.
- Transmit FIFO and receive FIFO.
- Collision detection and arbitration.
- Abort transmission.
- Sleep mode.
- Bus monitor mode.
- Measure pulse width for phase correction.
- Phase correction.
- Receive wake-up and bus reset signal.
- Register interface to the CPU.

29.1. Context

Apart from reset and clock line, the interface to the CPU consists of registers connected to the internal address and data bus. An output signal may be connected to the interrupt controller.

A modified universal port builds the output logic which is connected with its special input and output to the DIGITbus master. This provides an easy way for the SW to hold the bus line permanently low or high, or to investigate bus level directly, without support of DIGITbus Master HW.

An open drain output instead of a push/pull output is necessary for the universal port to build a single-line wired and bus.

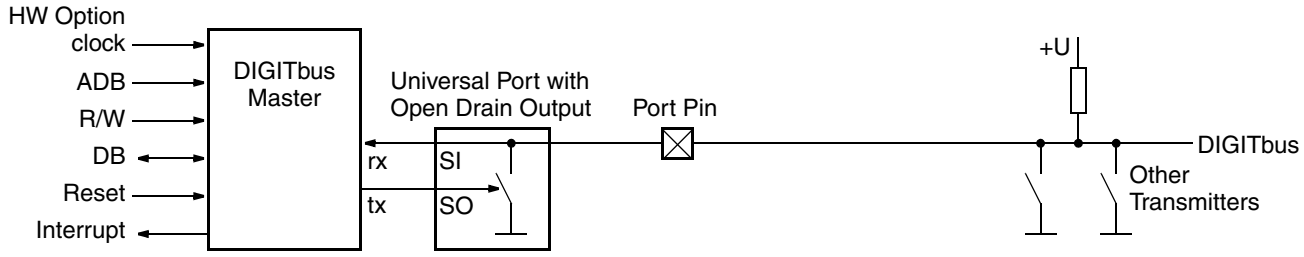


Fig. 29–1: Context diagram, single pin bus

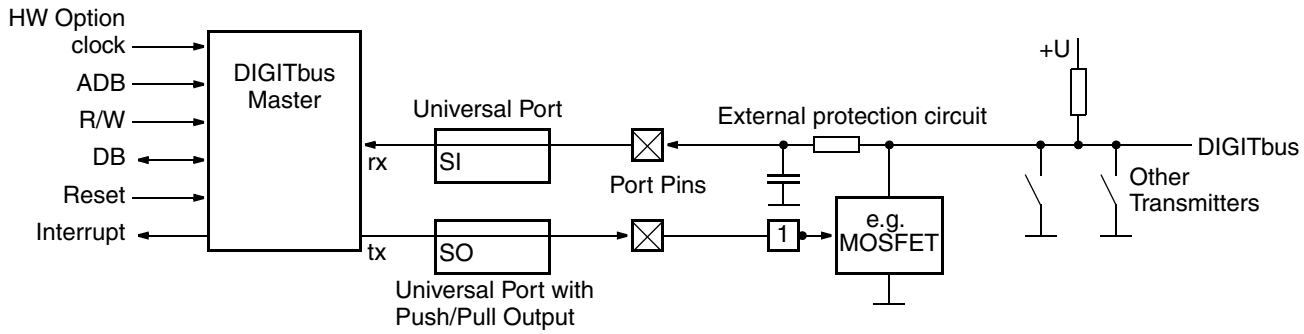


Fig. 29–2: Context diagram, double pin bus

29.2. Functional Description

29.2.1. 3-bit Prescaler

The programmable 3-bit Prescaler supplies the module with clock signals. It scales down the HW option selectable clock by factor 1, 2, 3 to 8 (see Table 29–2 on page 196). The output is 64 times the bus clock. The desired input frequency from the clock divider is hardware-programmable.

29.2.2. Internal Clocks

In low power mode the clock supply of the whole module with exception of the receive bit logic can be stopped. The receive bit logic needs a clock in low power mode too, because it must filter and watch the bus line for a wake-up signal.

29.2.3. Transmit T

The transmit T logic sends a continuous stream of T-signs if active. It outputs a permanent high if it is inactive.

29.2.4. Transmit Bit

Depending on the input signals the transmit bit logic modifies the T-signs to ones or zeros.

A phase correction can be done by adjusting the start time of a transmit bit sequence.

Other bus behavior than sending zeros, ones or T-signs may be forced by the SW using the universal port in normal mode directly. The bus line may be released or pulled low.

29.2.5. Receive Bit

The receive bit logic samples the bus level at a frequency of 64 times of the bus clock. It filters the input signal and decodes the input stream to supply the receive telegram logic with the logical bus signals (0, 1 and T) and the receive clock. Additional it measures the pulse width of each non T-sign. It creates a bus reset signal if the active bus is hold down beyond the end of a bit time. It creates a wake-up signal if there is a low level on the passive high bus.

29.2.6. Send Telegram

The send telegram logic will be enabled by the transmit FIFO and the receive telegram logic if four consecutive T-signs were received. It supports the transmit bit logic with the transmit bit sequence. If it recognizes the beginning of a new field, it waits one bit time (separator T-sign).

29.2.7. Receive Telegram

The receive telegram logic traces the bus and indicates the state to the status register and other related modules. The received bit field is written to the receive FIFO. The receive telegram logic is active all the time. Even if the module is transmitting a telegram, all bits must also be received in a multimaster system, because arbitration may be lost. Reception of own telegrams can be disabled (in a single-master system).

29.2.8. Collision Detection

The collision detection logic compares each incoming with the actual outgoing bit. A difference is signaled to the send telegram logic. If the module is transmitting, the send telegram logic is stopped immediately and the transmit FIFO and shift register are flushed.

29.2.9. Transmit FIFO

The transmit FIFO has five entry addresses. One for the field length of address or data field, one for a address byte, one for a data byte, one for more address bytes and one for more data bytes. The field length has to be written once before the corresponding field is entered into the FIFO unless the field length is not a multiple of 8.

An entry into the address register is inserted into the bus clock after the reception of 4 consecutive T-signs. An entry into the data register is inserted into the bus clock after the reception of a non T-sign and one T-sign. Thus it is possible to append a second data field (maybe acknowledge) after the reception of a telegram.

The transmit FIFO may be flushed to abort a transmission. It is also flushed if the transmit telegram logic is active and a collision is detected.

29.2.10. Receive FIFO

The receive FIFO will be filled from the receive shift register. It has two exit addresses. One for the field length and field type and one for the bit field. The field length has to be read before the corresponding field is taken from the FIFO. The receive FIFO will be frozen if it is full. The receive shift register will be overwritten.

29.2.11. Interrupt

Several flags of the status registers are connected by a logical-or to the interrupt source signal. The interrupt output can be masked by a flag in the control register.

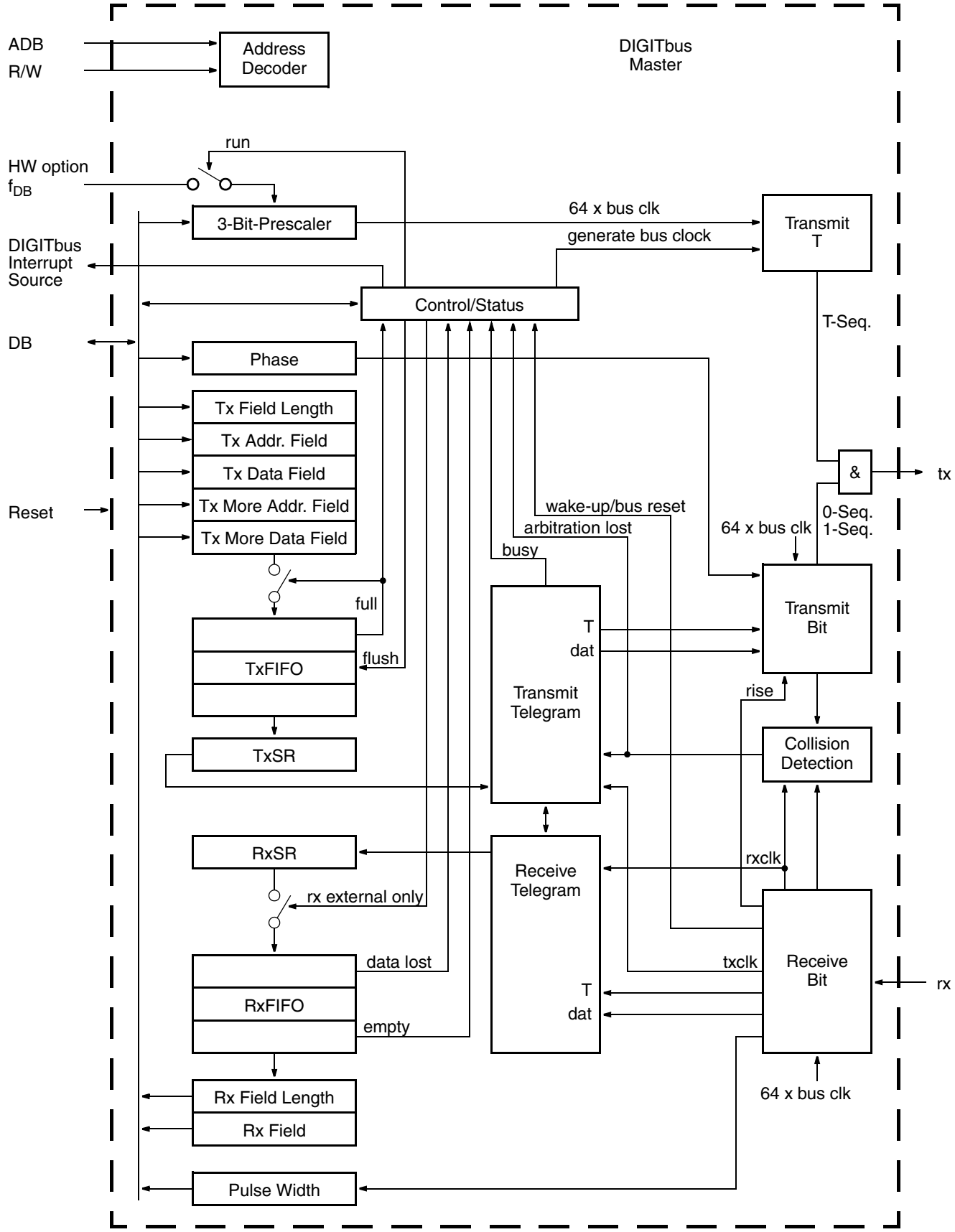


Fig. 29–3: Block diagram

29.3. Registers

The register mnemonic prefix “DG” stands for DIGITbus.

Table 29–1: Register mapping

Addr. Offs.	Mnem.	readable	writable
0	DGC0	Control 0	
1	DGC1	Control 1	
2	DGS0	Status 0	
3	DGRTMD	Rx Length	Tx More Data
4	DGTL	Tx Length	
5	DGS1TA	Status 1	Tx Addr.
6	DGTD	reserved	Tx Data
7	DGRTMA	Rx Field	Tx More Addr.

An “x” in a writable bit location means that this flag is reserved. The user has to write a zero to this location for further compatibility. An “x” in a readable bit location means that this flag is reserved. A read from this location results in an undefined value.

DGC0 Control Register 0

	7	6	5	4	3	2	1	0	
r/w	RUN	GBC	ACT	RXO	X	PSC 2 to 0			
	0	0	0	0	x	0	0	0	Res

RUN **Run**
r/w1: Module clock is active.
r/w0: Module is not clocked.
The module is absolute inactive if RUN is zero. Other flags are not functional then.

GBC **Generate Bus Clock**
r/w1: Module generates bus clock
r/w0: No bus clock

ACT **Activate**
r/w1: Module is active (reception and transmission).
r/w0: Module is sleeping (low power mode).
Only the receive bit logic is active in low power mode.

RXO **Receive External Only**
r/w1: Don’t receive own telegrams.
r/w0: Receive all.

PSC **Prescaler**
r/w: Scaling value

Table 29–2: Clock prescaler

PSC	Divide by	Bus Clock in kHz		
		f_{DB} = 4 MHz	f_{DB} = 5 MHz	f_{DB} = 10 MHz
0x0	1	62.5	78.1	156.25
0x1	2	31.25	39.1	78.1
0x2	3	20.8	26.0	52.1
0x3	4	15.6	19.5	39.1
0x4	5	12.5	15.6	31.25
0x5	6	10.4	13.0	26.0
0x6	7	8.9	11.2	22.3
0x7	8	7.8	9.8	19.5

Note: With an input clock of 5 MHz, the bus clock frequency of 31.25 kHz and its derivatives (16, 8, 4, 2, 1 kHz) cannot be achieved. Thus, with a 5 MHz quartz the DIGITbus should be operated in PLL mode.

DGC1 Control Register 1

	7	6	5	4	3	2	1	0	
r/w	INTE	ENEM	ENOF	x	PHASE				
	0	0	0	x	0	0	0	0	Res

INTE **Enable Interrupt**
r/w1: Enable interrupt
r/w0: Disable interrupt

ENEM **Enable Not Empty Interrupt**
r/w1: Enable
r/w0: Disable

ENOF **Enable Not Full Interrupt**
r/w1: Enable
r/w0: Disable

PHASE **Phase Correction Field**
r/w: Transmit phase.
The start of the transmit frame can be selected in increments of 1/64 of a total bit time related to the rising edge. Values between 0 and 15 are possible, but only the interval from 0 to 9 results in correct behavior.

Set PHASE to 2 if the DIGITbus is operated as clock master (GBC = 1). This is necessary to compensate for internal delay of 2 clocks. Refer to section 29.4.10. for further information on phase correction.

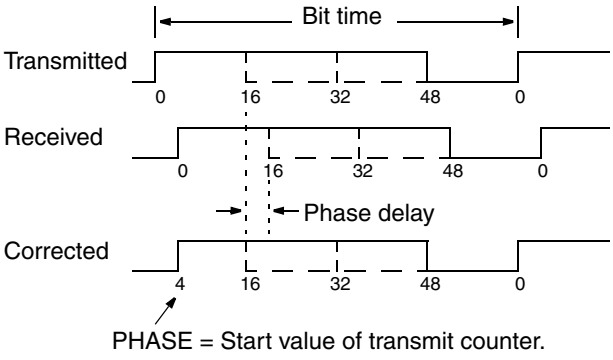


Fig. 29–4: Phase correction

DGS0 Status Register 0

	7	6	5	4	3	2	1	0
w	x	x	x	TGV	PV	ERR	x	ARB
r	RDL	NEM	NOF					
	x	0	1	0	0	0	x	0 Res

RDL Receive Data Lost
r1: Data lost
r0: No data lost
This flag is set if the receive FIFO is full and the shift register tries to store its contents to the FIFO because a new bit arrives. In this case the FIFO is frozen but the shift register is overwritten. It must be interpreted and cleared by the user. It is cleared by reading an entry from the FIFO.

NEM Rx FIFO is Not Empty
r1: There is at least one entry to read.
r0: Empty.
(see Fig. 29–5 on page 197)

NOF Tx FIFO is Not Full
r1: There is at least one entry free.
r0: Full.
It generates only an interrupt in the moment when the limit is passed. It does not generate interrupts when the FIFO is empty (see Fig. 29–5 on page 197).

TGV Telegram Valid
r1: Telegram valid
r0: Telegram not valid
w0: Clear flag
This flag will be set if two consecutive T-signs were received. It is reset by the HW if a non-T-sign is received. It can be cleared by the user if the related telegram is evaluated.

PV Protocol Violation
r1: Wake-up if bus is passive high.
Bus reset if bus is active.
r0: No trouble
w0: Clear flag
It must be interpreted and cleared by the user. It is set when the receive bit logic enters or leaves state passive high or when it enters the state passive low.

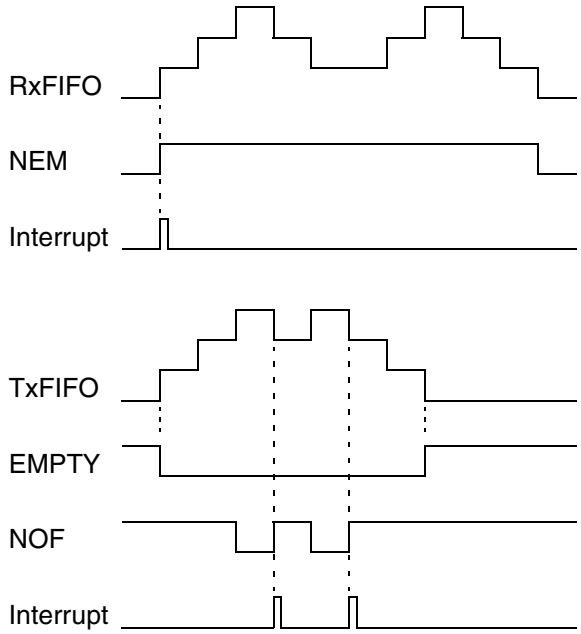


Fig. 29–5: Rx- and TxFIFO timing

ERR Error
r1: Fatal error.
r0: No error
w0: Clear flag
The HW sets this flag either if a dominant level is transmitted and a recessive level is detected (collision error), or if there was a wrong edge within a received bit. If a collision error is detected during transmission, the flag ARB will also be set and transmission stops immediately. This flag has to be cleared by the user.

ARB Arbitration Lost
r1: Arbitration lost.
r0: No arbitration loss.
w0: Clear flag
This flag will be set if a collision is detected during transmission. It must be cleared by the user. The transmit buffer is flushed if ARB is true. It is impossible to write to the transmit FIFO as long as ARB is true. Wait until flag TGV is true before reloading TxFIFO. This is automatically done if ARB is evaluated within the TGV interrupt subroutine only.

The flags RDL, NEM, NOF, TGV, and PV trigger the interrupt source signal (see Section 29.4.8. on page 201).

DGS1TA Status 1 & Tx Address Register

	7	6	5	4	3	2	1	0
w	Transmit Address							
r	STATE		PW5 to 0					
	0	1	0	0	0	0	0	0 Res

The first byte of an address field must be written to DGS1TA.

STATE **Bus State**
r: State of receive bit logic.

Table 29–3: Receiver states

STATE	Bus
0 0	Passive low
0 1	Passive high
1 0	Active low
1 1	Active high

PW **Pulse Width**
r: Pulse width
The pulse width of the most recent non-T-sign is stored in this register. It is measured in increments of 1/64 of the bus clock period.

DGRTMD **Rx Length & Tx More Data Register**

	7	6	5	4	3	2	1	0	
w	Transmit More Data								
r	RDL	NEM	FTYP	EOFLD	x	LEN2 to 0			
	0	0	x	x	x	x	x	x	Res

More bytes of a data field must be written to DGRTMD.

The read part of register DGRTMD is associated with the front entry in the receive FIFO (the receive field DGRTMA). It has to be read and interpreted before the corresponding FIFO entry.

RDL **Receive Data Lost**
r1: Data lost
r0: No data lost
The flag RDL from the status register DGS0 is mirrored here. It is cleared by a read access to register DGRTMA.

NEM **Receive FIFO is Not Empty**
r1: There is at least one entry.
r0: Empty
The flag NEM from the status register DGS0 is mirrored here. FTYP, EOFLD, LEN and register DGRTMA are not valid if NEM is false.

FTYP **Field Type**
r1: Address field
r0: Data field

EOFLD **End of Field**
r1: Last byte of a field
r0: Not last byte of a field
If EOFLD is set, the corresponding FIFO entry is the last part of the actual field. The next entry, if there is one, belongs to a new field.

LEN **Length of Field**
r: Length of valid data bit
The three-bit length does not limit the overall length of the corresponding field. The length field defines how many bits of the front entry of the receive FIFO carry valid bits. They are right aligned (Table 29–4). The real length of the field is unlimited. The user must count the bytes he fetched from the FIFO to calculate the real field length.

Table 29–4: LEN usage, receive and transmit length

	LEN 2 1 0	Valid Bit Numbers 7 6 5 4 3 2 1 0
1	0 0 1	_ _ _ _ _ _ _ **x**
2	0 1 0	_ _ _ _ _ _ **x x**
3	0 1 1	_ _ _ _ _ **x x x**
4	1 0 0	_ _ _ _ **x x x x**
5	1 0 1	_ _ _ **x x x x x**
6	1 1 0	_ _ **x x x x x x**
7	1 1 1	_ **x x x x x x x**
0	0 0 0	**x x x x x x x x**

The examples in Table 29–5 illustrate the interpretation of register DGRTMD. They are valid for an address field (FTYP = 1) or a data field (FTYP = 0).

Table 29–5: DGRTMD interpretation examples

LEN	EOFLD	
6	1	Last byte of a field. The six right most bits belong to the field.
0	0	A byte of a field. All bits belong to the field. At least one byte follows.
0	1	Last byte of a field. Eight bits belong to the field.
≠0	0	Impossible.

DGRTMA **Rx Field & Tx More Address Register**

	7	6	5	4	3	2	1	0	
w	Transmit More Address								
r	Receive Field								
	x	x	x	x	x	x	x	x	Res

More bytes of an address field must be written to DGRTMA.

The bytes of a received field must be read from register DGRTMA. The meaning of this field (address or data) is defined by the flag FTYP.

Received bytes of a bit field are right-aligned. The last byte of a long bit field (with the LSB) may be filled partially. To get the whole bit field right-aligned, it is necessary to shift all preceding bytes to the right.

A read access to this register takes the top entry of the receive FIFO. Both registers DGRTMA and DGRTMD are overwritten by the next FIFO entry as result of a read access.

DGTL	Transmit Length Register								
	7	6	5	4	3	2	1	0	
w	x	FLUSH	x	x	x	LEN2 to 0			
	x	0	x	x	x	0	0	0	Res
r	BUSY	EMPTY	x	x	x	x	x	x	
	0	1	x	x	x	x	x	x	Res

The transmit length register is associated with the whole field (address or data) which will be written into the transmit FIFO. It has to be written before the first entry of the field.

BUSY **Transmitter is Busy**
r1: Busy.
r0: Idle.
This flag is true as long as there is an entry in the TxFIFO or transmission is not completed. It is set with the first entry into the TxFIFO and reset after the transmission of the first T-sign after a telegram.

FLUSH **Flush Tx FIFO**
w1: Empty Tx FIFO and abort transmission.
w0: No action.
This flag will be reset by the HW autonomously. After FLUSH wait until EMPTY or TGV becomes true before rewriting TxFIFO. Setting of FLUSH clears TGV at the same time.

EMPTY **Tx FIFO is Empty**
r1: No transmit telegram in FIFO.
r0: Transmit telegram in FIFO.

LEN **Length of Field**
w: Length of address or data field.
These three bits correspond to the first byte of a bit field. They define how many bits of this byte carry valid information and should be transmitted (see Table 29–4 on page 198). DGTL must be written before the first byte of the actual bit field is written to the FIFO. It has only to be written once for each bit field. The overall length of the bit field is not limited.

DGTD	Transmit Data Register								
	7	6	5	4	3	2	1	0	
w	Transmit Data								
	x	x	x	x	x	x	x	x	Res

The first byte of a data field must be written to DGTD.

The first byte of a bit field (with the MSB) which is entered into DGS1TA or DGTD, may be partially filled. In the following bytes, all bits must contain valid data.

29.4. Principle of Operation

29.4.1. Reset

The module reset signal resets all registers and internal HW. The standby bit in a standby register does the same.

Setting flag RUN in register DGC0 resets all internal HW and registers with exception of registers DGC0, DGC1, DGS0 and DGS1TA. These registers are accessible all the time, they are not reset by any setting of the DIGITbus Master flags.

Internal HW are reset to an inactive state (not transmitting, not receiving). Internal counters are reset to zero. FIFOs and shift registers are empty. Internal representations of the bus line are reset to passive bus level (high).

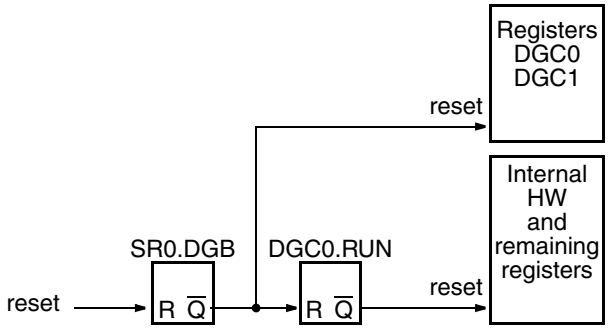


Fig. 29–6: Reset structure

29.4.2. Hardware Settings

The DIGITbus clock and the source of the DIGIT-IN input signal can be set by HW Options (See table 29–6). Refer to section “HW Options” for setting them.

29.4.3. Initialization

Prior to operation, proper SW configuration of the U-Ports assigned to module DIGITbus as input and as double pull-down output has to be made. Be aware that it is possible to have DIGITbus input and output on the same port pin for easy design of a serial wired-or bus (single-pin bus, Figure 29–1), or to have input and output on separate port pins which allows a design of an input protection circuit (double pin bus, Figure 29–2). See table 29–6 and section "Ports" for details.

After reset and after setting flag DGB in standby register SR0, the DIGITbus master is inactive. The global enable flag RUN must be set together with the appropriate prescaler entry PSC, to activate the module.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

29.4.3.1. Clock Master

The flag GBC (generate bus clock) must be set, if the DIGITbus master should generate the bus clock. The module acts now as clock master of the connected DIGITbus system. It outputs a stream of T-signs.

Table 29–6: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
DIGIT-bus	Clock	DC	DIGIT-OUT	U2.6 special out, double pull-down mode (single pin bus) or special out (double pin bus)	SR0. DGB
	Input	PM.CACO	DIGIT-IN	U2.4 (special in) or U2.6 (special out)	

29.4.3.2. Receiver/Transmitter

Setting the flag ACT activates the receive and transmit logic. From now on all telegrams are received in the receive FIFO. Writing to the transmit FIFO initiates transmission of a telegram.

The bus clock (T-signs) must be activated some time before the first telegram is transmitted. This is necessary, because other modules may use a PLL for generating the internal clock from the bus clock. No telegram shall be transmitted before all modules have locked on the bus clock.

29.4.3.3. Single Master System

In a single-master system (no collision possible), you can suppress reception of transmitted telegrams by setting flag RXO (receive external only). This unburdens the CPU from clearing the receive FIFO of those telegrams.

29.4.3.4. Multimaster System

In a multimaster system it is necessary that each transmitted telegram is received too, because arbitration may be lost and then the transmitter becomes a receiver. If arbitration was not lost, the receive FIFO must be read to empty it. The flag RXO has to be cleared in a multimaster system.

29.4.4. Transmission

Transmission is initiated by writing a telegram into the transmit FIFO.

If the field length is not a multiple of 8 bit, the total field length modulo 8 has to be written to register DGTL. This must be done once for each field and before any entry to registers DGS1TA, DGTD, DGRTMA or DGRTMD. If the total field length is a multiple of 8 it is not necessary to write the field length to register DGTL.

The first entry of a field (address or data) has to be written right-aligned to register DGS1TA (address) or DGTD (data). Further entries of the same field, if it is longer than 8 bits, have to be written to DGRTMA (more address) or DGRTMD (more data). A telegram is transmitted MSB first, hence fields have to be written to transmit FIFO MSB first.

Table 29–7: Operating modes

RUN	GBC	ACT	RXO	Remarks
0	x	x	x	Standby mode
1	0	x	x	Passive master. External bus clock generation is necessary.
1	1	x	x	Clock master
1	0	0	x	Sleep mode
1	x	1	x	Active mode
1	x	1	0	Receive all. (Recommended in multi master system)
1	x	1	1	Receive external only. (Recommended in single master system)

A new address field is transmitted if there are at least 4 consecutive T-signs on the bus. A new data field is transmitted if there was exactly one T-sign. If the last bit of a field was transmitted and there are no more entries in the transmit FIFO, the transmitter stops sending. After reception of two consecutive T-signs the telegram valid flag TGV is set. This is the signal for the SW to evaluate whether transmission was correct or whether an arbitration loss or an error canceled transmission (flags ARB, PV and ERR). In the latter case SW must initiate retransmission.

A telegram has been transmitted correctly, if ARB and ERR are false and EMPTY is true.

Transmission starts with the first entry in the transmit FIFO. Consecutive fields should be entered before the transmission of the preceding field is finished. Take care about possible interrupts.

29.4.4.1. Transmit FIFO

SW must ascertain that there is an empty entry in the transmit FIFO before writing to it. Flag NOF (not full) indicates that there is at least one entry free. Flag EMPTY indicates complete emptiness of transmit FIFO. After reset, FLUSH or ARB wait until flag TGV is true before rewriting TxFIFO.

Short telegrams can completely be buffered in the FIFO. Managing long telegrams is a SW job. The SW must buffer long telegrams and write the parts in time. The transmit FIFO is intended to unburden the CPU from immediately reaction on an NOF interrupt. If an entry becomes free, the SW has time to write, as long as it needs to transmit two FIFO entries and the contents of the transmit shift register. This time must not necessarily be the duration for sending 24 bit. Maybe only one bit of each remaining FIFO entry has to be sent.

The transmit FIFO is not intended for telegram tracking. Only one transmit telegram at a time must be entered.

29.4.5. Reception

Every non-T-sign is shifted into the receive shift register. If it is full or if a T-sign is received, the shift register is stored into the receive FIFO. This is done until the receive FIFO is full.

In this case, the FIFO is frozen, but the shift register continues operation. The flag RDL indicates the latter case.

If the shift register is stored to the receive FIFO because a T-sign was received, the corresponding flag EOFLD is set, indicating that this is the last entry of a field.

The corresponding flag FTYP is modified at the same time. If two or more consecutive T signs were received in front of the actual field, it is set, indicating that this field has to be interpreted as an address field. If only one T-sign has been received in front of the actual field, it is cleared, indicating that it has to be interpreted as a data field.

The flag TGV is set if two consecutive T-signs were received. This is the moment to read status flags and Receive FIFO. The flags PV and ERR have to be interpreted. Even if an error occurred, the Receive FIFO must be emptied by reading it because every telegram or fragment is stored there. Otherwise reception of the next telegram may overflow the receive FIFO, which is indicated by flag RDL.

Every time you want to read DGRTMA, it is ingenious to read DGRTMD first, because DGRTMD and DGRTMA are overwritten with a read access to DGRTMA.

29.4.5.1. Receive FIFO

The receive FIFO contains entries as long as flag NEM is true.

Short telegrams can be buffered completely in the receive FIFO. SW must buffer long telegrams and read parts of it in time.

29.4.6. Sleep Mode

Only the receive bit logic is active in sleep mode. Neither transmission nor reception of telegrams is possible.

A wake-up (passive high to low edge) is signaled by flag PV.

The DIGITbus master is not automatically activated by a wake-up. This has to be done by SW. The flag PV can be used to trigger an interrupt.

Switching to Sleep Mode while a telegram is transmitted can cause problems. Hence make sure, that bus clock generation is switched off only if bus is idle (T-signs).

29.4.7. Abort Transmission

Writing a one to flag FLUSH aborts the transmission of a telegram after completion of the actual transmitted bit, if the DIGITbus master is the transmitter. The transmit FIFO is emptied and another, more urgent telegram can be transmitted. Transmission of the new telegram starts, as soon as 4 consecutive T-signs were received after the aborted telegram.

Flag TGV is cleared with a FLUSH. This is the reason why TGV is set (and interrupt is triggered if enabled) after reception of 2 T-signs, even if no telegram was aborted by FLUSH because it happened during transmission of T-signs.

Resetting of flag TGV is the reason why an aborted address field is marked as data field (FTYP = 0) in the RxFIFO.

It is not possible to abort a telegram or a field which is transmitted by another bus node.

29.4.8. Interrupt

Five flags (RDL, NEM, NOF, TGV, PV) are connected to the interrupt source output by an or operation. This output can be enabled globally by flag INTE. The interrupt generation of two flags (NEM, NOF) can be enabled locally by flags ENEM and ENOF. A rising edge of a flag triggers the interrupt source output.

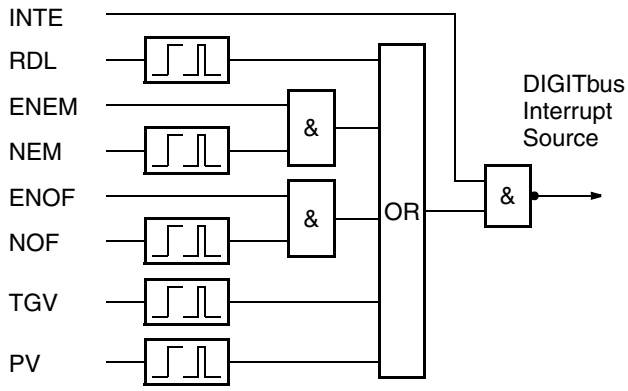


Fig. 29–7: Interrupt Sources

29.4.9. Measure Pulse Width

The pulse width (high time) of every non-T-sign is stored with the falling edge of the bus signal in status register DGS1TA in the field PW. T-signs do not affect PW. It must be read before the falling edge of the next non-T-sign.

29.4.10. Correct Phase

The rising edge of the bus signal can be delayed by inner (sampling and filter) or outer (bus load) influences. This delayed rising edge resets a 6-bit transmit counter in the transmit bit logic. The transmit counter pushes the bus line low when it reaches 15 (transmitting 0) or 31 (transmitting 1). It releases the bus line when it reaches 55.

The transmit counter is reset to a value which contains two zeros at the most significant position and the four PHASE bits of the control register DGC1 at the least significant position. This allows an adjustment of the transmitted non-T-signs between 0 and 15/64 of the whole bit length.

29.4.11. Error

The setting of flag ERR may have one of the following causes:

– Wrong baud rate of DIGITbus Master or other bus nodes.

– Wrong port configuration of DIGITbus Master.

– Disturbances on bus line.

– HW of DIGITbus Master is damaged.

29.4.12. Precautions

Do not access DIGITbus registers in CPU Slow and Deep Slow mode. This can cause interrupts.

If f_{XTAL} is 5 MHz, a bus clock of 31.25 kHz is possible only in PLL mode (Table 29–2).

29.5. Timings

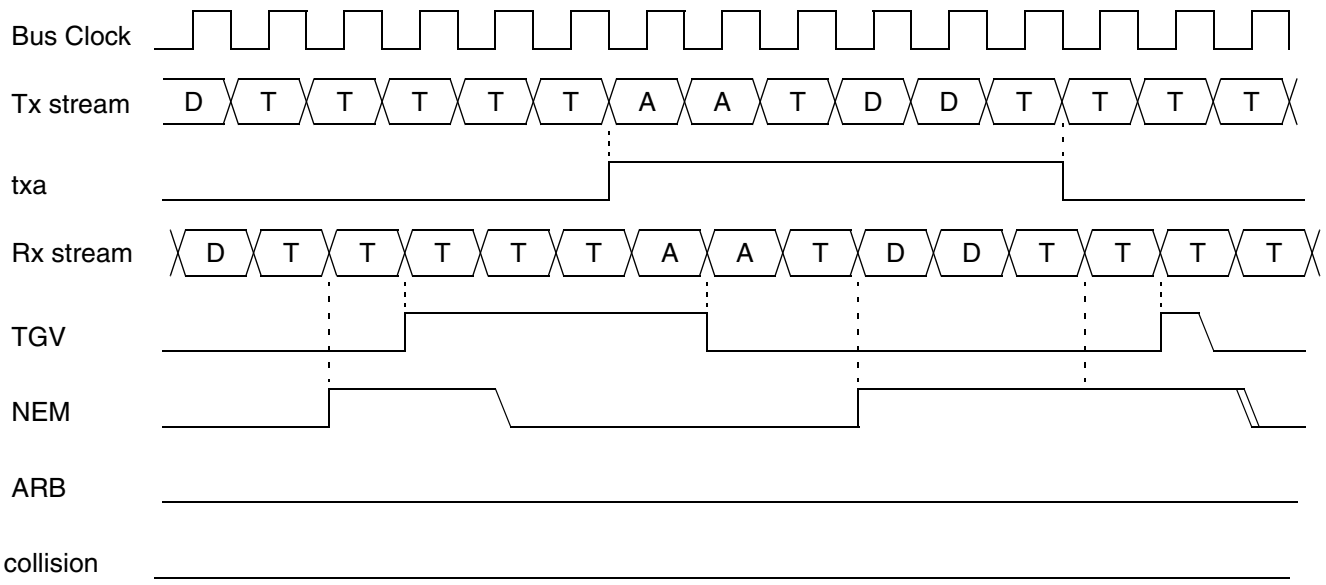


Fig. 29–8: Tx timing

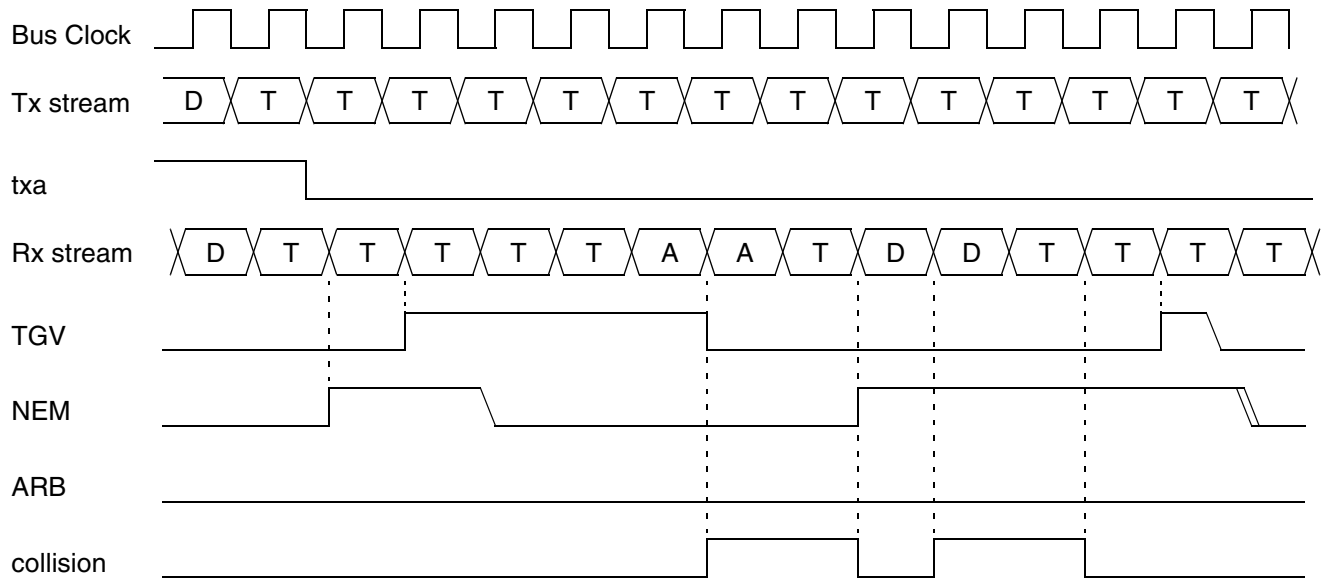


Fig. 29–9: Rx timing

30. Audio Module (AM)

The audio module (AM) provides a gong output signal that may be used to drive a speaker circuit.

The output signal is a square wave signal with selectable gong frequency.

The gong signal amplitude is defined by the pulse width of a PWM signal. An internal accumulator is selectable to automatically decrease this pulse width and thus the gong amplitude following an exponential function.

Features

- Programmable gong frequency
- Programmable gong duration
- Programmable initial amplitude
- Gong can be stopped and retriggered
- Generation of an exponentially decreasing gong amplitude function without CPU interaction

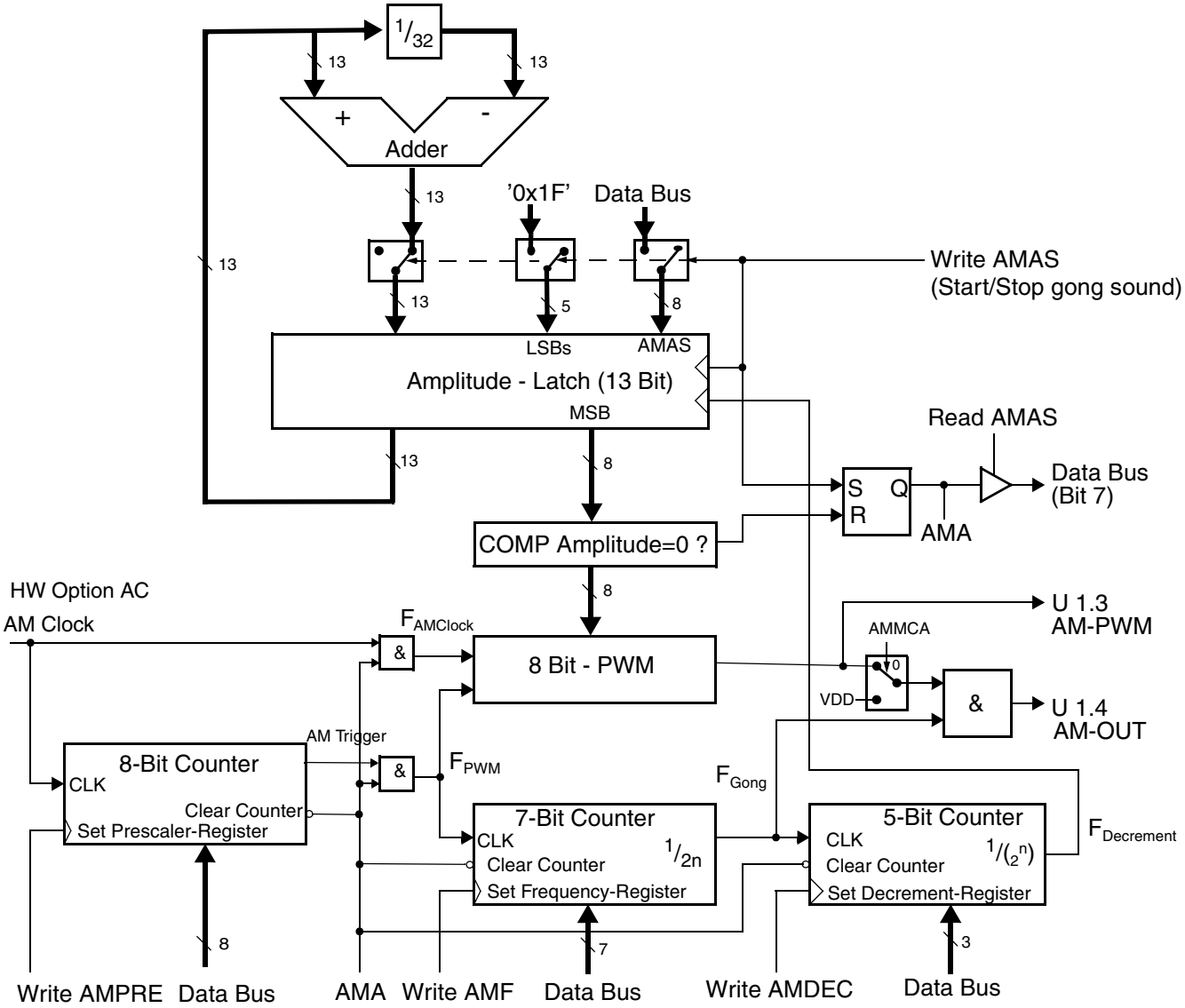


Fig. 30–1: Block diagram of the audio module

30.1. Functional Description

The audio module output frequency is defined by the following equations:

Frequency:

$$F_{Gong} = \frac{F_{AMTrigger}}{2(AMF+1)} = \frac{F_{\mathrm{AMClock}}}{2(AMF+1)(AMPRE+1)}$$

Amplitude:

$$\mathrm{Ampl.} \cong \frac{(\mathrm{AMAS}+1)}{(\mathrm{AMPRE}+1)}$$

where the maximum amplitude is 1 if AMAS is equal to or bigger than AMPRE. In the latter case the amplitude remains constant until the decay mechanism has decreased AMAS below AMPRE.

Duration:

$$t_d \cong \frac{\ln 0{,}5 - \ln \mathrm{AMAS}}{\ln\left(1-\frac{1}{32}\right)} \cdot \frac{2^{\mathrm{GDF}}}{F_{\mathrm{Gong}}}$$

AMF, AMPRE, AMAS and GDF are register values and described later.

The initial gong sound amplitude is set by writing the Audio Module Amplitude & Status Register (AMAS), this write also starts the gong sound. An active audio module is indicated by the read-only audio module active bit (AMA) in the AMAS.

Every 1st..32nd cycle of the gong sound frequency (depending on the gong duration factor (AMDEC.GDF) a new amplitude value is calculated ($F_{Decrement}$). The falling edge of the amplitude decrement frequency $F_{Decrement}$ is latching the output of the adder into the amplitude latch (13 Bit) and simultaneously the 8 MSBs into the PWM.

During the first low cycle of F_{Gong} following the active $F_{Decrement}$ edge the PWM is already running with the newly calculated amplitude, but takes effect at the output not until the next high cycle of F_{Gong}. F_{Gong} is modulating the PWM-output to generate the gong sound frequency, while the decreasing PWM-value generates an exponential decreasing amplitude.

As soon as the 8 MSBs of the amplitude latch are reaching zero, the AMA will be reset, which deactivates the audio module.

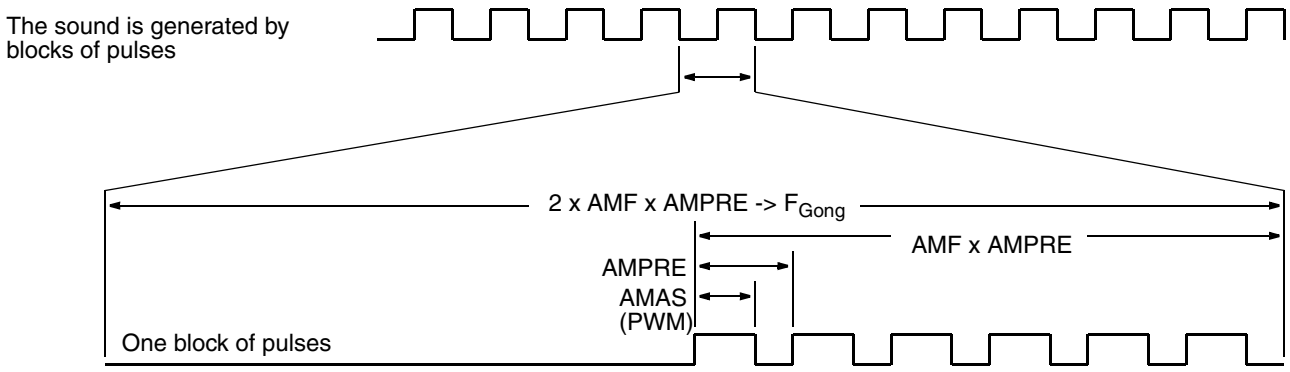


Fig. 30–2: Sound generation

30.1.1. Hardware Settings

The AM clock frequency $F_{AMClock}$ is set by HW option AC.

30.1.2. Initialization

Prior to entering active mode, proper SW initialization of the ports has to be made. The ports have to be configured special out. Refer to “Ports” for details.

Three audio module registers have to be set before the gong sound can be started: the gong prescaler (AMPRE), the gong sound frequency (AMF) and the gong duration factor (AMDEC.GDF) register.

For the effect of CPU clock modes on the operation of this module refer to section “CPU and Clock System” (see Table 4–1 on page 41).

30.1.3. Start Gong

The gong sound is started by writing the initial amplitude value into AMAS. Simultaneously with the write to AMAS the Flag Audio Module Active (AMA) is set, which enables the $F_{AMClock}$-input.

30.1.4. Restart Gong

It is possible to restart the gong sound simply by writing a new initial amplitude value to AMAS independent of the former initial value or the current value of the register. (Note:

The current amplitude value cannot be read out). The new gong sound will start immediately with a low cycle of F_{Gong}.

30.1.5. Stop Gong

The gong sound will stop automatically as soon as the amplitude value in AMAS reaches zero. This will reset the AMA, which indicates the inactive audio module.

To stop the gong sound, just write 0x00 into AMAS. The gong sound then will stop immediately with the writing of 0x00 (also indicated by AMA).

A continuous tone will never stop automatically. It has to be stopped by writing 0x00 into AMAS.

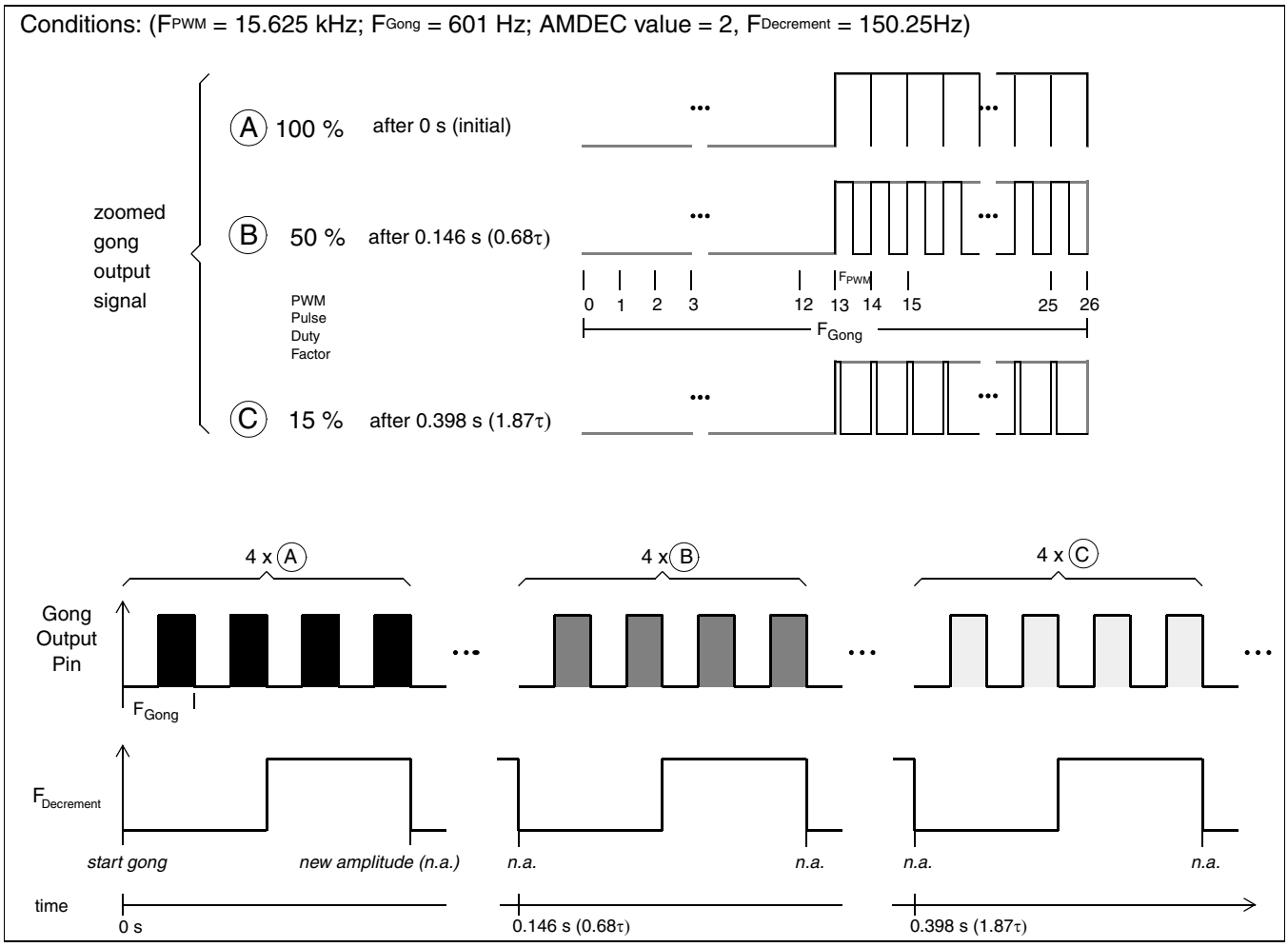


Fig. 30–3: Example sections of the audio module output signal

30.1.6. Decay of Sound

The decay characteristic used for this gong sound is described by the following exponential function:

$$A_n \cong A_0 \, (1 - {}^1/_{32})^n$$

with A_0 = initial amplitude (AMAS)
A_n = amplitude after n $F_{Decrement}$ cycles
n = int (t * $F_{Decrement}$)
= number of decrement cycles

Following the above formula, n can be expressed as

$$n \cong \frac{\ln A_n - \ln A_0}{\ln\left(1 - \frac{1}{32}\right)}$$

Each $F_{Decrement}$ cycle the amplitude is decreased by 1/32. $F_{Decrement}$ is determined by the value of GDF in the register AMDEC and by F_{Gong}:

$$F_{Decrement} = \frac{F_{GONG}}{2^{GDF}} \qquad GDF = 0 \ldots 5$$

With GDF settings of 6 and 7 the gong sound amplitude update frequency $F_{Decrement}$ is zero (continuous tone).

The time constant τ of the above exponential function is defined as the time interval within which the amplitude A is decreasing to 36.8%.

Given

$$0,368 = \left(1 - \frac{1}{32}\right)^{n_\tau}$$

the number n_τ of $F_{Decrement}$ cycles needed to reduce the initial amplitude to 36.8% is

$$n_\tau \cong 32$$

With an initial amplitude of 0xFF the total time $t_{255->0}$ needed to reach zero amplitude in the 8 Bit - AMAS is $n = 193$ $F_{Decrement}$ cycles, which is approximately 6τ.

With an initial amplitude lower than 0xFF the gong sound duration is shorter.

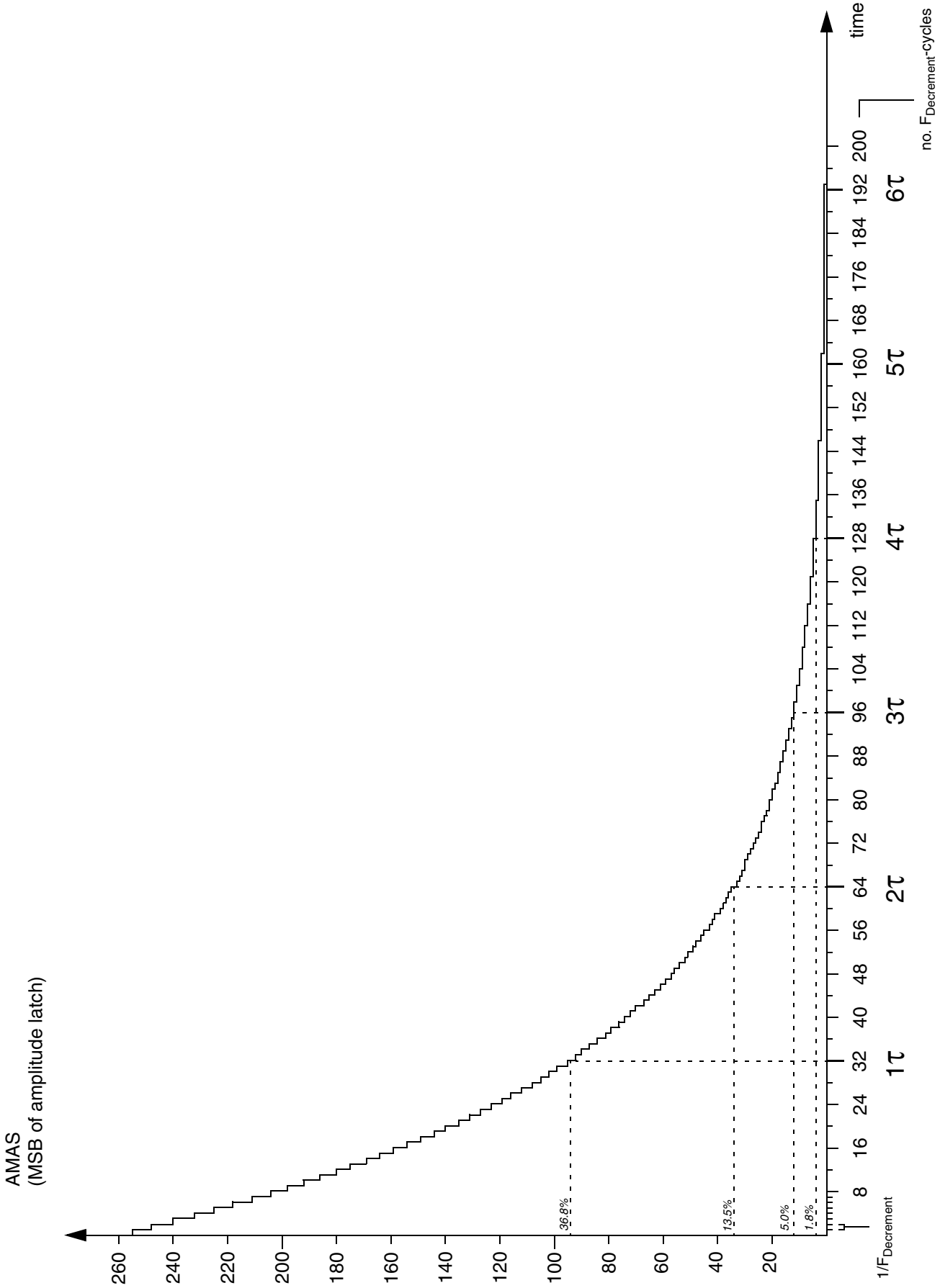
That means that τ is correlating with $F_{Decrement}$. The higher $F_{Decrement}$, the shorter τ is.

The total time from a start amplitude A_0 to an end amplitude A_n is approximately calculated according to following formula.

$$t_{A_0 \to A_n} \cong \frac{\ln A_n - \ln A_0}{\ln\left(1 - \frac{1}{32}\right)} \cdot \frac{2^{GDF}}{F_{GONG}}$$

To sum up, it can be said that the total duration of the gong sound depends on F_{Gong}, set with AMF and AMPRE, the setting of the gong duration factor GDF and the setting of the initial amplitude AMAS.

.



Decay of Sound - Function

30.2. Registers

AMAS	Audio Module Amplitude and Status Register								
	7	6	5	4	3	2	1	0	Note
w	Initial Amplitude								
r	AMA	x	x	x	x	x	x	x	
	0	x	x	x	x	x	x	x	Res

Initial Amplitude

A write access to this register starts or stops the gong sound, while the value written is the initial amplitude. Writing the value 0x0 into this register during an active gong sound deactivates the gong sound immediately, while writing a value > 0x0 is restarting the gong sound immediately with the new Initial Amplitude.

wnn: (Re-)Start gong sound with initial amplitude.
w00: Stop gong sound.

AMA Audio Module Active Flag

This flag indicates an active Audio Module generating a gong sound.

r1: Audio Module is active.
r0: Audio Module is not active.

AMF	Audio Module Frequency Register								
	7	6	5	4	3	2	1	0	Note
w	x	Sound Frequency							
	-	0	0	0	0	0	0	0	Res

With this register the gong sound frequency is programmed. The PWM frequency is divided by twice the register value increased by one.

The value which has to be written, resp. the resulting gong sound frequency is calculated with:

$$AMF = \frac{F_{AMTrigger}}{2F_{GONG}} - 1$$

It is possible to write a new gong sound frequency during an active audio module (AMA = '1').

AMDEC	Audio Module Decrement Register								
	7	6	5	4	3	2	1	0	Note
w	AMMCA	x	x	x	x	GDF			
	0	-	-	-	-	0	0	0	Res

AMMCA Audio Module Maximum Constant Amplitude Flag

w1: Activate the AMMCA mode.
w0: Deactivate the AMMCA mode.

With the flag AMMCA the audio module maximum constant amplitude mode is selected. If this flag is set, the gong sound with the maximum, not decreasing amplitude is available at the audio module output pin. The only difference between this tone and a 'normal' gong sound is the constant, not decreasing amplitude. The handling of this tone (i.e. start, stop, frequency, duration) is the same. The tone is started by writing an initial value to AMAS, but this value will only influence the duration of the tone, not its amplitude.

GDF Gong sound Duration Factor

This register sets the gong sound duration in dependence of F_{Gong}. With GDF=0 the amplitude will be decreased every F_{Gong} - cycle, values 1 to 5 will result in an amplitude update frequency of F_{Gong} / 2 to F_{Gong} / 32 according to this equation:

$$F_{Decrement} = \frac{F_{GONG}}{2^{GDF}} \qquad GDF = 0 \ldots 5$$

A value of 6 or 7 disables decrease of the amplitude, so a continuous tone with the initial amplitude will be generated ($F_{Decrement}$ = 0). To stop the continuous tone write a 0x00 to AMAS or change the gong sound duration factor to let the tone decay. It is possible to change GDF during an active gong sound (AMA = '1').

Table 30–1: Definition of GDF

GDF	gong sound duration factor
0x0	1
0x1	2
0x2	4
0x3	8
0x4	16
0x5	32
0x6	continuous tone
0x7	

AMPRE	Audio Module Prescaler								
	7	6	5	4	3	2	1	0	Note
w	Prescale Value								
	1	1	1	1	1	1	1	1	Res

AMPRE defines the frequency of the trigger input of the Audio Module. The AM clock input is divided by the Prescale Value plus one to derive the trigger frequency F_{PWM}.

$$\mathrm{AMPRE} = \frac{F_{\mathrm{AMClock}}}{F_{\mathrm{AMTrigger}}} - 1$$

AMPRE must be greater than zero.

31. Hardware Options

31.1. Functional Description

Hardware Options are available in several areas to adapt the IC function to the host system requirements:

– clock signal selection for most of the peripheral modules from f_0 to $f_0/2^{17}$ plus some internal signals (see Table 31–3 on page 210)

– special out signal selection for some U- and H-ports

– Rx/Tx polarity selection for SPI and UART modules

Hardware option setting requires two steps:

1. selection is done by programming dedicated address locations in the HW options field (see Section 31.2. on page 209) with the desired options' code (see Section 31.3. on page 210).

2. activation is done by copying the HW options field to the corresponding HW options registers (see Section 31.3. on page 210) at least once after each reset.

In EMU and MCM devices all HW options are SW progammable.

In mask ROM derivatives the clock options are hard-wired according to the HW options field of the ROM code hex file. Those options can only be altered by changing a production mask.

To ensure compatible option settings in MCM and mask ROM derivatives, when run with the same ROM code, it is mandatory to always write the HW options field to the HW option registers directly after reset.

31.2. Listing of Dedicated Addresses of the Hardware Options Field

Please refer to section "Memory and Special Function ROM System" for the dedicated start address of the HW options field.

Table 31–1: HW Options Field

Offs.	Mne.	Options
0x00	T0C	Timer 0 Clock
0x01	T1C	Timer 1 Clock
0x02	T2C	Timer 2 Clock
0x03	T3C	Timer 3 Clock
0x04	T4C	Timer 4 Clock
0x05	CO00C	Clock Out 0: Mux0 Pre. & Clock
0x06	CO01C	Clock Out 0: Mux1 Clock
0x07	RZPC	Rotor Zero Position Detection Clock
0x08	DMAC	DMA Timer Clock
0x09	CO1C	Clock Out 1: Pre. & Clock
0x0A	C0C	CAPCOM Counter 0 Clock
0x0B	C1C	CAPCOM Counter 1 Clock
0x0C	DC	DIGITbus Clock
0x0D	LC	LCD Pre. & Clock
0x0E	AC	AM Clock

Table 31–1: HW Options Field

Offs.	Mne.	Options
0x0F	PF0C	PFM 0, 1 Clock
0x10	SMC	SM, SPI0, SPI1 Pre. & SM Clock
0x11	SP0C	SPI0 I/O & F0SPI Clock
0x12	SP1C	SPI1 I/O & F1SPI Clock
0x13	SP2C	F2SPI Clock
0x14	P9C	PWM 8, 9 Clock
0x15	P9P	PWM 8, 9 Period
0x16	P11C	PWM 10, 11 Clock
0x17	P11P	PWM 10, 11 Period
0x18	P1C	PWM 0, 1 Clock
0x19	P1P	PWM 0, 1 Period
0x1A	P3C	PWM 2, 3 Clock
0x1B	P3P	PWM 2, 3 Period
0x1C	P5C	PWM 4, 5 Clock
0x1D	P5P	PWM 4, 5 Period
0x1E	P7C	PWM 6, 7 Clock
0x1F	P7P	PWM 6, 7 Period

Table 31–1: HW Options Field

Offs.	Mne.	Options
0x20		
0x21		
0x22		
0x23		
0x24		
0x25		
0x26		
0x27		
0x28		
0x29	PM	Port Mux
0x2A		
0x2B		
0x2C	UA0	UART0 I/O
0x2D	UA1	UART1 I/O
0x2E		
0x2F		

31.3. HW Options Registers and Code

The mapping of the HW options registers corresponds exactly to the HW options field in the section above. The order of the HW options registers description in this section does not correspond to the order of the HW options field.

The emulator IC allow SW programming of the whole registers. Future mask ROM derivatives do not allow writing other clock option values than defined in the HW options field.

The clock options may be programmed to values according to table 31–3 on page 210.

Some of the clocks may be prescaled by a programmable value. Refer to table 31–2 for possible values.

Table 31–2: Clock prescaler

PRE		Prescale Value
1	0	
x	0	direct
0	1	1/1.5
1	1	1/2.5

Table 31–3: Clock option selection code

Clock Option Number	Clock Signal	Selection Code
f0	f_0	xxx0.0000
f1	f_1	xxx0.0001
f2	$f_1/2^1$	xxx0.0010
f3	$f_1/2^2$	xxx0.0011
f4	$f_1/2^3$	xxx0.0100
f5	$f_1/2^4$	xxx0.0101
f6	$f_1/2^5$	xxx0.0110
f7	$f_1/2^6$	xxx0.0111
f8	$f_1/2^7$	xxx0.1000
f9	$f_1/2^8$	xxx0.1001
f10	$f_1/2^9$	xxx0.1010
f11	$f_1/2^{10}$	xxx0.1011
f12	$f_1/2^{11}$	xxx0.1100
f13	$f_1/2^{12}$	xxx0.1101
f14	$f_1/2^{13}$	xxx0.1110
f15	$f_1/2^{14}$	xxx0.1111
f16	$f_1/2^{15}$	xxx1.0000
f17	$f_1/2^{16}$	xxx1.0001
f18	V_{SS}	xxx1.0010
f19	T0-OUT	xxx1.0011
f20	V_{SS}	xxx1.0100
f21	fSM	xxx1.0101
f22 [1])	$fSM/2^8$	xxx1.0110
f23	fCC0IN	xxx1.0111
f24	fCC1IN	xxx1.1000
f25, 26, 27	V_{SS}	xxx1.1001 ...
f28	$f_1/2^1$	xxx1.1100
f29, 30	V_{SS}	xxx1.1101 ...
f31	$f_1/2^9$	xxx1.1111

If the leading “x” in the clock sampling table are not used for the purpose of coding other options, they must be replaced by zeros.
1) Clock option f22 is only available if the stepper motor module has been enabled by the standby bit.

31.3.1. Timers

T0C Timer 0 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f1 to f31				
	x	x	x	0x01				Res

T1C Timer 1 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

T2C Timer 2 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

T3C Timer 3 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

T4C Timer 4 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

31.3.2. PWMs

The high pulse width of the trigger period must be greater than the high pulse width of the clock the PWM is provided with.

P1C PWM 0, 1 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P1P PWM 0, 1 Period

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

P3C PWM 2, 3 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P3P PWM 2, 3 Period

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

P5C PWM 4, 5 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P5P PWM 4, 5 Period

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

P7C PWM 6, 7 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P7P PWM 6, 7 Period

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

P9C PWM 8, 9 Clock

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P9P PWM 8, 9 Period

	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

P11C	PWM 10, 11 Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

P11P	PWM 10, 11 Period							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x08				Res

31.3.3. CAPCOMs

C0C	CAPCOM Counter 0 Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

C1C	CAPCOM Counter 1 Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

31.3.4. DIGITbus

DC	DIGITbus Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

31.3.5. DMA

DMAC	DMA Timer Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

31.3.6. PFM

PF0C	PFM 0, 1 Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

31.3.7. Clock Out

CO00C	Clock Out 0: Mux0 Pre. & Clock							
	7	6	5	4	3	2	1	0
w	x	PRE		Clock Options f0 to f31 (all)				
	x	0x0		0x11				Res

PRE **Prescaler** (Table 31–2)

CO01C	Clock Out 0: Mux1 Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

RZPC	Rotor Zero Position Detection Clock							
	7	6	5	4	3	2	1	0
w	x	x	x	Clock Options f0 to f31 (all)				
	x	x	x	0x0				Res

CO1C	Clock Out 1: Pre. & Clock							
	7	6	5	4	3	2	1	0
w	x	PRE		Clock Options f0 to f31 (all)				
	x	0x0		0x11				Res

PRE **Prescaler** (Table 31–2)

31.3.8. LCD

LC	LCD Pre. & Clock							
	7	6	5	4	3	2	1	0
w	x	PRE		Clock Options f0 to f31 (all)				
	x	0x0		0x03				Res

PRE **Prescaler** (Table 31–2)

31.3.9. Stepper Motor and SPIs

SMC	SM, SPI0, SPI1 Pre. & SM Clock							
	7	6	5	4	3	2	1	0
w	x	PRE		Clock Options f0 to f31 (all)				
	x	0x0		0x0				Res

PRE **Prescaler** (Table 31–2)

The field PRE of register SMC defines the SPI0 and SPI1 prescaler setting too.

SP0C				SPI0 I/O & $F0_{SPI}$ Clock			
7	6	5	4	3	2	1	0
w SPI0OUT	SPI0IN	x	Clock Options f0 to f31 (all)				
0	0	x	0x0				Res

SPI0OUT **SPI0 Data Output Inverter**
w1: Inverted.
w0: Direct.

SPI0IN **SPI0 Data Input Inverter**
w1: Inverted.
w0: Direct.

The clock is prescaled by SMC.PRE.

SP1C				SPI1 I/O & $F1_{SPI}$ Clock			
7	6	5	4	3	2	1	0
w SPI1OUT	SPI1IN	x	Clock Options f0 to f31 (all)				
0	0	x	0x0				Res

SPI1OUT **SPI1 Data Output Inverter**
w1: Inverted.
w0: Direct.

SPI1IN **SPI1 Data Input Inverter**
w1: Inverted.
w0: Direct.

The clock is prescaled by SMC.PRE.

SP2C				$F2_{SPI}$ Clock			
7	6	5	4	3	2	1	0
w x	x	x	Clock Options f0 to f31 (all)				
x	x	x	0x0				Res

The clock is prescaled by SMC.PRE.

31.3.10. Audio Module

AC				AM Clock			
7	6	5	4	3	2	1	0
w x	x	x	Clock Options f0 to f31 (all)				
x	x	x	0x0				Res

31.3.11. Port Multiplexers

PM				Port Mux			
7	6	5	4	3	2	1	0
w U15	CC4I	CACO	U20	U06	H7	H0	PINT
0	0	0	1	0	0	0	0 Res

U15 **U-Port 1.5 Output Select**
w1: CO1.
w0: CO0Q.

CC4I **CAPCOM4-IN Select**
w1: Input from P0.0.
w0: Input from U5.3.

CACO **CAPCOM0, 1, 2-IN and DIGIT-IN Select**
w1/0: Table 31–4.

Table 31–4: PM.CACO Usage

Signal	PM.CACO = 0	PM.CACO = 1
CAPCOM0-IN	U3.2	U4.1
CAPCOM1-IN	U3.1	U2.4
CAPCOM2-IN	U3.0	U2.2
DIGIT-IN	U2.4	U2.6

U06 **U-Port 0.6 Output Select**
w1: CC3-OUT.
w0: T4-OUT.

U20 **U-Port 2.0 Output Select**
w1/0: Table 31–5.

Table 31–5: PM.U20 Usage

Signal	PM.U20 = 0	PM.U20 = 1
CAN0-TX	U2.0 and U4.2	U4.2
CAN0-RX	U2.1	U4.3
SCL0	not usable	U2.0
SDA0	not usable	U2.1

H7 **H-Port 7 Output Select**
w1: PWM9, 8, 6, 4.
w0: SME.

H0 **H-Port 0 Output Select**
w1: PWM7, 5, 3, 1.
w0: SMG.

PINT **Port Interrupt Select**
w1: Input from P1.2 to P1.7.
w0: Input from U1.7, U1.6, U1.5, U0.7, U0.5, U0.4.

31.3.12. UARTs

UA0				UART0 I/O				
7	6	5	4	3	2	1	0	
w U0TX	U0RX	x	x	x	x	x	x	
0	0	x	x	x	x	x	x	Res

U0TX **UART0 Tx Inversion Select**
w1: Inverted.
w0: Direct.

U0RX **UART0 Rx Inversion Select**
w1: Inverted.
w0: Direct.

UA1 **UART1 I/O**

	7	6	5	4	3	2	1	0	
w	U1TX	U1RX	x	x	x	x	x	x	
	0	0	x	x	x	x	x	x	Res

U1TX **UART1 Tx Inversion Select**
w1: Inverted.
w0: Direct.

U1RX **UART1 Rx Inversion Select**
w1: Inverted.
w0: Direct.

32. Register Cross Reference Table

32.1. 8-Bit I/O Region

Table 32–1: Base address 0x00F80000

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0xFFC					4 CAN reserved	CAN RAM
0x800						
0x7FC					CAN 3	
0x600						
0x5FC					CAN 2	
0x400						
0x3FC					CAN 1	
0x200						
0x1FC					CAN 0	
0x000						

Table 32–2: Base address 0x00F81000

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x1FC					4 CAN reserved	CAN register
0x100						
0x0FC					CAN3	
0x0D4						
0x0D0			CTIM			
0x0CC	ESM	REC	TEC	OCR		
0x0C8	ICR	BT3	BT2	BT1		
0x0C4	IDM					
0x0C0	IDX	ESTR	STR	CTR		
0x0BC					CAN2	
0x094						
0x090			CTIM			
0x08C	ESM	REC	TEC	OCR		
0x088	ICR	BT3	BT2	BT1		
0x084	IDM					
0x080	IDX	ESTR	STR	CTR		
0x07C					CAN1	
0x054						
0x050			CTIM			
0x04C	ESM	REC	TEC	OCR		
0x048	ICR	BT3	BT2	BT1		
0x044	IDM					
0x040	IDX	ESTR	STR	CTR		
0x03C					CAN0	
0x014						
0x010			CTIM			
0x00C	ESM	REC	TEC	OCR		
0x008	ICR	BT3	BT2	BT1		
0x004	IDM					
0x000	IDX	ESTR	STR	CTR		

Table 32–3: Base address 0x00F90000 (formerly 1F00)

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC	TST2	TST1	TST3	TST4		Test
0x0F8	TST5		TSTAD3	TSTAD2		
0x0F4	DGRTMA	DGTD	DGS1TA	DGTL		DIGITBus
0x0F0	DGRTMD	DGS0	DGC1	DGC0		
0x0EC					64 byte	
0x0B0						
0x0AC				ANAA		ADC
0x0A8			AD1	AD0		
0x0A4		UA0IF	UA0CA	UA0IM		UART0
0x0A0	UA0BR1	UA0BR0	UA0C	UA0D		
0x09C					32 byte	
0x080						
0x07C			CCC0H	CCC0L		CAPCOM0
0x078	CC3H	CC3L	CC3I	CC3M	CC3	
0x074	CC2H	CC2L	CC2I	CC2M	CC2	
0x070	CC1H	CC1L	CC1I	CC1M	CC1	
0x06C	CC0H	CC0L	CC0I	CC0M	CC0	
0x068					8 byte	
0x064						
0x060			DBG	CSW1		Core Logic
0x05C	SMVMUX		SMVCMP	SMVCOS		Stepper Motor Module VDO
0x058	SMVSIN	SMVC				
0x054	TIM4	TIM3	TIM2	TIM1		Timer
0x050						
0x04C	TIM0H	TIM0L			Timer0	
0x048			CCC1H	CCC1L		CAPCOM1
0x044	CC5H	CC5L	CC5I	CC5M	CC5	
0x040	CC4H	CC4L	CC4I	CC4M	CC4	
0x03C					16 byte	
0x030						
0x02C	AMDEC	AMF	AMAS	AMPRE		Audio Module
0x028	IRPM1	IRPM0				Port Interrupt
0x024					8 byte	
0x020						
0x01C		UA1IF	UA1CA	UA1IM		UART1
0x018	UA1BR1	UA1BR0	UA1C	UA1D		
0x014				CO0SEL		Core Logic
0x010	SPI1M	SPI1D	SPI0M	SPI0D		SPI
0x00C	SR1					Core Logic
0x008	SR0					
0x004				ANAU		
0x000				CSW0		

Table 32–4: Base address 0x00F90100 (formerly 1E00)

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC					16 byte	HW Options
0x0F0						
0x0EC			UA1	UA0		
0x0E8			PM			
0x0E4						
0x0E0						
0x0DC	P7P	P7C	P5P	P5C		
0x0D8	P3P	P3C	P1P	P1C		
0x0D4	P11P	P11C	P9P	P9C		
0x0D0	SP2C	SP1C	SP0C	SMC		
0x0CC	PF0C	AC	LC	DC		
0x0C8	C1C	C0C	CO1C	DMAC		
0x0C4	RZPC	CO01C	CO00C	T4C		
0x0C0	T3C	T2C	T1C	T0C		
0x0BC					96 byte	
0x060						
0x05C						PFM
0x058						
0x054	PFM1					
0x050	PFM0					
0x04C	PWMC					PWM
0x048	PWM11	PWM10	PWM9	PWM8		
0x044	PWM7	PWM6	PWM5	PWM4		
0x040	PWM3	PWM2	PWM1	PWM0		
0x03C					32 byte	
0x020						
0x01C					I2C1	I2C
0x018	I2CM1					
0x014	I2CRS1	I2CRD1	I2CWP11	I2CWP01		
0x010	I2CWD11	I2CWD01	I2CWS11	I2CWS01		
0x00C					I2C0	
0x008	I2CM0					
0x004	I2CRS0	I2CRD0	I2CWP10	I2CWP00		
0x000	I2CWD10	I2CWD00	I2CWS10	I2CWS00		

Table 32–5: Base address 0x00F90400

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC				HxPIN	H-Port7	H-Ports
0x0F8	HxLVL	HxNS	HxTRI	HxD		
0x0F4				HxPIN	H-Port6	
0x0F0	HxLVL	HxNS	HxTRI	HxD		
0x0EC				HxPIN	H-Port5	
0x0E8	HxLVL	HxNS	HxTRI	HxD		
0x0E4				HxPIN	H-Port4	
0x0E0	HxLVL	HxNS	HxTRI	HxD		
0x0DC				HxPIN	H-Port3	
0x0D8	HxLVL	HxNS	HxTRI	HxD		
0x0D4				HxPIN	H-Port2	
0x0D0	HxLVL	HxNS	HxTRI	HxD		
0x0CC				HxPIN	H-Port1	
0x0C8	HxLVL	HxNS	HxTRI	HxD		
0x0C4				HxPIN	H-Port0	
0x0C0	HxLVL	HxNS	HxTRI	HxD		
0x0BC						P-Ports
0x0B8	P2LVL		P2IE	P2PIN	P-Port 2	
0x0B4	P1LVL		P1IE	P1PIN	P-Port1	
0x0B0	P0LVL		P0IE	P0PIN	P-Port 0	
0x0AC					reserved	U-Ports
0x090						
0x084	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 8	
0x080	UxDPM	UxNS	UxTRI	UxD		
0x074	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 7	
0x070	UxDPM	UxNS	UxTRI	UxD		
0x064	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 6	
0x060	UxDPM	UxNS	UxTRI	UxD		
0x054	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 5	
0x050	UxDPM	UxNS	UxTRI	UxD		
0x044	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 4	
0x040	UxDPM	UxNS	UxTRI	UxD		
0x034	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 3	
0x030	UxDPM	UxNS	UxTRI	UxD		
0x024	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 2	
0x020	UxDPM	UxNS	UxTRI	UxD		
0x014	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 1	
0x010	UxDPM	UxNS	UxTRI	UxD		
0x004	UxMODE	UxPIN	UxLVL	UxSLOW	U-Port 0	
0x000	UxDPM	UxNS	UxTRI	UxD		

Table 32–6: Base address 0x00F90500

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC					128 Bytes	reserved
0x080						
0x07C				SMX		Power Saving
0x078			POL		Polling	
0x074				RTCC	RTC	
0x070				OSC		
0x06C						
0x068				WSC		
0x064				WPM8	Wake Ports mode	
0x060	WPM6	WPM4	WPM2	WPM0		
0x05C	RTC				RTC	
0x058	SSC					
0x054	SSR					
0x050			WUS		Wake-up source	
0x04C					reserved	GBus
0x048						
0x044				GC		
0x040				GD		
0x03C			MDL		Memory Ctrl.	Core Logic
0x030				MFPLR	DLM	
0x02C				WSR	Clock, PLL, ERM	
0x028				IOC		
0x024	ERMC					
0x020				PLLC		
0x01C					reserved	LCD
0x014						
0x010				ULCDLD		
0x00C						Patch
0x008	PER					
0x004	PDR					
0x000	PAR					

32.2. 32-Bit I/O Region

Table 32–7: Base address 0x00FFFD00

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC					252 bytes reserved	Core Logic
0x004						
0x000	CR				Control Register	

Table 32–8: Base address 0x00FFFE00

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC					rsvd Channel 4 to 31	DMA
0x020						
0x018			DC3M		Channel 3	
0x010			DC2M		Channel 2	
0x008			DC1M		Channel 1	
0x004				DST	Control	
0x000	DVB					

Table 32–9: Base address 0x00FFFF00

Offs.	Byte Address				Remarks	
	3	2	1	0		Module
0x0FC					12 bytes reserved	IRQ and FIQ Interrupt Controller
0x0F4						
0x0F0			CRF	PRF	FIQ registers	
0x0EC					40 bytes reserved	
0x0C8						
0x0C4	VTB				IRQ registers	
0x0C0	PESRC	PEPRIO	AFP	CRI		
0x0BC					128 bytes reserved	
0x040						
0x03C					Interrupt source nodes	
0x028						
0x024	ISN39	ISN38	ISN37	ISN36		
:	:	:	:	:		
0x004	ISN7	ISN6	ISN5	ISN4		
0x000	ISN3	ISN2	ISN1	ISN0		

33. Register Quick Reference

The I/O area is organized in little endian format, thus the LSB, independent of the flag CR.ENDIAN setting, is always stored at the low address.

Table 33–1: Analog section (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
AD0	ADC Register 0	0x0A8	r: EOC, x, x, x, x, TEST, AN1, AN0 w: TSAMP, REF, CHANNEL 0 0 0 0 0 0 0 0 Res	15.7.
AD1	ADC Register 1	0x0A9	r: AN9, AN8, AN7, AN6, AN5, AN4, AN3, AN2 w: x, x, x, x, x, x, x, BUF 0 Res	
ANAA	Analog AVDD Register	0x0AC	r/w: EP06, P06, WAIT, x, x, x, x, BVE 0 0 Res	

Table 33–2: Analog input ports (base addr. 0xF90400)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
P0PIN	Port x Pin Register	0x0B0	r: P7, P6, P5, P4, P3, P2, P1, P0 1 1 1 1 1 1 1 1 Res	14.1.
P1PIN		0x0B4		
P2PIN		0x0B8		
P0IE	Port x Input Enable Register	0x0B1	r/w: I7, I6, I5, I4, I3, I2, I1, I0 0 0 0 0 0 0 0 0 Res	
P1IE		0x0B5	r/w: I7, I6, I5, I4, I3, I2, I1, I0 0 0 0 0 0 0 1 1 Res	
P2IE		0x0B9	r/w: x, x, x, x, x, x, I1, I0 0 0 Res	
P0LVL	Port x Level Register	0x0B3	r/w: A7, A6, A5, A4, A3, A2, A1, A0 0 0 0 0 0 0 0 0 Res	
P1LVL		0x0B7		
P2LVL		0x0BB		

Table 33–3: Audio module (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
AMPRE	Audio Module Prescaler	0x02C	w: **Prescale Value** 1 1 1 1 1 1 1 1 Res	30.2.
AMAS	Audio Module Amplitude & Status Register	0x02D	w: **Initial Amplitude** r: **AMA** **x** **x** **x** **x** **x** **x** **x** 0 x x x x x x x Res	
AMF	Audio Module Frequency Register	0x02E	w: **x** **Sound Frequency** - 0 0 0 0 0 0 0 Res	
AMDEC	Audio Module Decrement Register	0x02F	w: **AMMCA** **x** **x** **x** **x** **GDF** 0 - - - - 0 0 0 Res	

Table 33–4: Capture-compare-unit 0 (base addr. 0xF90000)

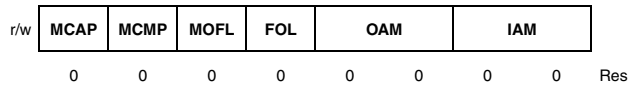
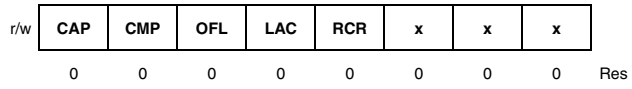
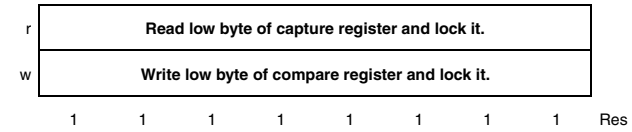
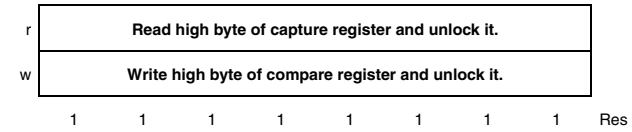
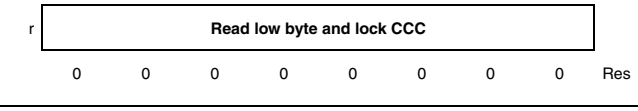
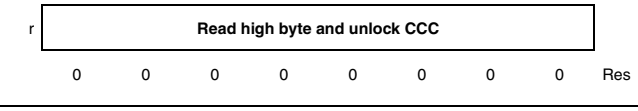
Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
CC0M	CAPCOM 0 Mode Register	0x06C		19.2.
CC1M		0x070		
CC2M		0x074		
CC3M		0x078		
CC0I	CAPCOM 0 Interrupt Register	0x06D		
CC1I		0x071		
CC2I		0x075		
CC3I		0x079		
CC0L	CAPCOM 0 Capture/ Compare Register low byte	0x06E		
CC1L		0x072		
CC2L		0x076		
CC3L		0x07A		
CC0H	CAPCOM 0 Capture/ Compare Register high byte	0x06F		
CC1H		0x073		
CC2H		0x077		
CC3H		0x07B		
CCC0L	CAPCOM Counter 0 low byte	0x07C		
CCC0H	CAPCOM Counter 0 high byte	0x07D		

Table 33–5: Capture-compare-unit 1 (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration	Section
			7 6 5 4 3 2 1 0	
CC4M	CAPCOM Mode Register	0x040	r/w: MCAP, MCMP, MOFL, FOL, OAM, IAM 0 0 0 0 0 0 0 0 Res	19.2.
CC5M		0x044		
CC4I	CAPCOM Interrupt Register	0x041	r/w: CAP, CMP, OFL, LAC, RCR, x, x, x 0 0 0 0 0 0 0 0 Res	
CC5I		0x045		
CC4L	CAPCOM Capture/ Compare Register low byte	0x042	r: Read low byte of capture register and lock it. w: Write low byte of compare register and lock it. 1 1 1 1 1 1 1 1 Res	
CC5L		0x046		
CC4H	CAPCOM Capture/ Compare Register high byte	0x043	r: Read high byte of capture register and unlock it. w: Write high byte of compare register and unlock it. 1 1 1 1 1 1 1 1 Res	
CC5H		0x047		
CCC1L	CAPCOM Counter 1 low byte	0x048	r: Read low byte and lock CCC 0 0 0 0 0 0 0 0 Res	
CCC1H	CAPCOM Counter 1 high byte	0x049	r: Read high byte and unlock CCC 0 0 0 0 0 0 0 0 Res	

Table 33–6: Controller area network registers (base addr. 0xF81000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN0CTR	Control Register	0x000	r/w HLT SLP GRSC EIE GRIE GTIE BOST rsvd 1 0 0 0 0 0 0 x Res	27.2.
CAN1CTR		0x040		
CAN2CTR		0x080		
CAN3CTR		0x0C0		
CAN0STR	Status Register	0x001	r HACK BOFF EPAS ERS rsvd rsvd rsvd rsvd 1 0 0 0 x x x x Res	
CAN1STR		0x041		
CAN2STR		0x081		
CAN3STR		0x0C1		
CAN0ESTR	Error Status Register	0x002	r/w GDM CTOV ECNT BIT STF CRC FRM ACK 0 0 0 0 0 0 0 0 Res	
CAN1ESTR		0x042		
CAN2ESTR		0x082		
CAN3ESTR		0x0C2		
CAN0IDX	Interrupt Index Register	0x003	r/w Interrupt Index 1 1 1 1 1 1 1 1 Res	
CAN1IDX		0x043		
CAN2IDX		0x083		
CAN3IDX		0x0C3		
CAN0IDM	Identifier Mask Register	0x004	r/w Identifier Mask Bits 4 to 0 x x x 3 r/w Identifier Mask Bits 12 to 5 2 r/w Identifier Mask Bits 20 to 13 1 r/w Identifier Mask Bits 28 to 21 0 0 0 0 0 0 0 0 0 Res	
CAN1IDM		0x044		
CAN2IDM		0x084		
CAN3IDM		0x0C4		
CAN0BT1	Bit Timing Register 1	0x008	r/w MSAM SYN BPR 0 0 0 0 0 0 0 0 Res	
CAN1BT1		0x048		
CAN2BT1		0x088		
CAN3BT1		0x0C8		
CAN0BT2	Bit Timing Register 2	0x009	r/w rsvd TSEG2 TSEG1 0 0 0 0 0 0 0 0 Res	
CAN1BT2		0x049		
CAN2BT2		0x089		
CAN3BT2		0x0C9		

Table 33–6: Controller area network registers (base addr. 0xF81000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN0BT3	Bit Timing Register 3	0x00A	r/w: rsvd rsvd rsvd rsvd rsvd SJW x x x x x 0 0 0 Res	27.2.
CAN1BT3		0x04A		
CAN2BT3		0x08A		
CAN3BT3		0x0CA		
CAN0ICR	Input Control Register	0x00B	r/w: rsvd rsvd rsvd rsvd rsvd XREF REF1 REF0 x x x x x x 0 0 Res	
CAN1ICR		0x04B		
CAN2ICR		0x08B		
CAN3ICR		0x0CB		
CAN0OCR	Output Control Register	0x00C	r/w: rsvd rsvd rsvd rsvd rsvd rsvd rsvd ITX x x x x x x x 0 Res	
CAN1OCR		0x04C		
CAN2OCR		0x08C		
CAN3OCR		0x0CC		
CAN0TEC	Transmit Error Counter	0x00D	r: Counter Bit 7 to 0 0 0 0 0 0 0 0 0 Res	
CAN1TEC		0x04D		
CAN2TEC		0x08D		
CAN3TEC		0x0CD		
CAN0REC	Receive Error Counter	0x00E	r: x Counter Bit 6 to 0 x 0 0 0 0 0 0 0 Res	
CAN1REC		0x04E		
CAN2REC		0x08E		
CAN3REC		0x0CE		
CAN0ESM	Error Status Mask Register	0x00F	r/w: EGDM ECTV EECT EBIT ESTF ECRC EFRM EACK 1 1 1 1 1 1 1 1 Res	
CAN1ESM		0x04F		
CAN2ESM		0x08F		
CAN3ESM		0x0CF		
CAN0CTIM	Capture Timer	0x010	r: Timer Bit 15 to 8 (1) r: Timer Bit 7 to 0 (0) 0 0 0 0 0 0 0 0 Res	
CAN1CTIM		0x050		
CAN2CTIM		0x090		
CAN3CTIM		0x0D0		

Table 33–7: Core logic 32-bit (base addr. 0xFFFD00)

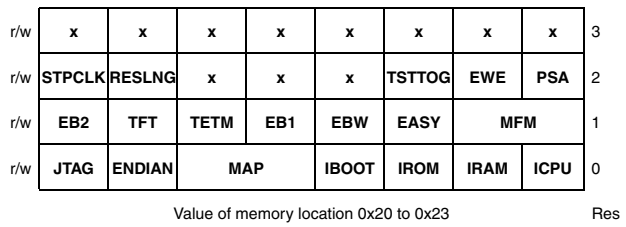
Mnemonic	Register Name	Offs.	Register Configuration	Section
CR	Control Register	0x000		6.1.

Table 33–8: Core Logic 8-bit (base addr. 0xF90000)

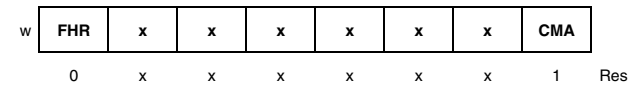
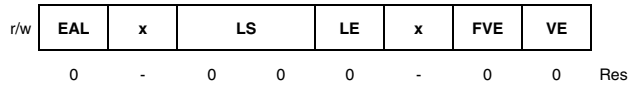
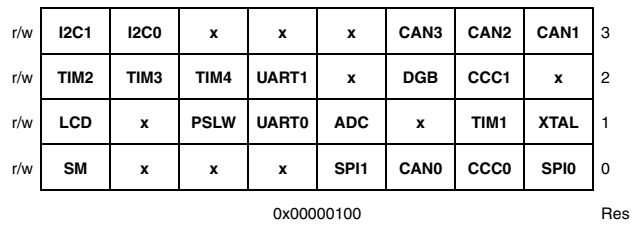
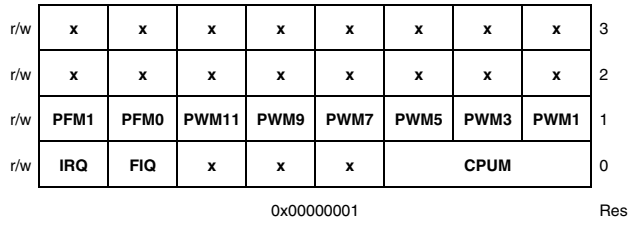
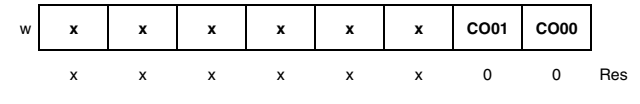
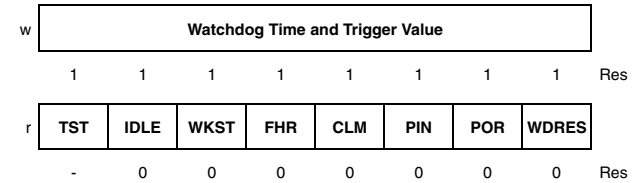
Mnemonic	Register Name	Offs.	Register Configuration	Section
CSW0	Clock, Supply and Watchdog Register 0	0x000		6.
ANAU	Analog UVDD Register	0x004		
SR0	Standby Register 0	0x008		
SR1	Standby Register 1	0x00C		
CO0SEL	Clock Out 0 Selection	0x014		
CSW1	Clock, Supply and Watchdog Register 1	0x060		

Table 33–9: Core logic 8-bit (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
PLLC	PLL Control	0x020	r/w: ACT | LCK | PLLM | x | PMF x x x x 0 0 0 0 Res	6.
ERMC	ERM Control	0x024	r/w: TSEL 3 r/w: x x x x x x x x 2 r/w: EOM | x | x | TOL 1 r/w: INPH | x | SUP 0 0x00000000 Res	
IOC	I/O Control	0x028	w: x x x x x | IOP x x x x x 0 0 0 Res	
WSR	Wait State Register	0x02C	w: NWS | SWS 0x00 Res	
MFPLR	Multi Function Port Lock Register	0x030	r/w: x x x x x x x | MFPL x x x x x x x 1[1)] Res	
MDL	Memory Delay	0x03C	w: x x x x | MDDL 1 w: x x x x | MCDL 0 0x0000 Res	34.2.

Table 33–10: DIGITbus (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
DGC0	Control Register 0	0x0F0	r/w: RUN | GBC | ACT | RXO | X | PSC 2 to 0 0 0 0 0 x 0 0 0 Res	29.3.
DGC1	Control Register 1	0x0F1	r/w: INTE | ENEM | ENOF | x | PHASE 0 0 0 x 0 0 0 0 Res	
DGS0	Status Register 0	0x0F2	w: x | x | x | TGV | PV | ERR | x | ARB r: RDL | NEM | NOF x 0 1 0 0 0 x 0 Res	
DGRTMD	Rx Length & Tx More Data Register	0x0F3	w: Transmit More Data r: RDL | NEM | FTYP | EOFLD | x | LEN2 to 0 0 0 x x x x x x Res	
DGTL	Tx Length Register	0x0F4	w: x | FLUSH | x | x | x | LEN2 to 0 x 0 x x x 0 0 0 Res r: BUSY | EMPTY | x | x | x | x | x | x 0 1 x x x x x x Res	
DGS1TA	Status 1 & Tx Address Register	0x0F5	w: Transmit Address r: STATE | PW5 to 0 0 1 0 0 0 0 0 0 Res	
DGTD	Tx Data Register	0x0F6	w: Transmit Data x x x x x x x x Res	
DGRTMA	Rx Field & Tx More Address Register	0x0F7	w: Transmit More Address r: Receive Field x x x x x x x x Res	

Table 33–11: DMA (base addr. 0xFFFE00)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
DVB	DMA Vector Base	0x000	r/w 0 0 0 0 0 0 0 0 3 r/w A23 to A16 2 r/w A15 to A8 1 r/w A7 0 0 0 0 0 0 0 0 0x0000 Res	22.2.
DST	DMA Status	0x004	r/w DE x x SRC 0 0x00 Res	
DC1M	DMA Channel x Mode	0x008	r/w P DMAT TRIG 1 r/w EN x x x BYP DIR MAS 0 0x0000 Res	
DC2M		0x010		
DC3M		0x018		

Table 33–12: FIQ interrupt logic (base addr. 0xFFFF00)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
PRF	Pending Register FIQ	0x0F0	r/w x x x x x x x P x x x x x x x 0 Res	12.2.
CRF	Control Register FIQ	0x0F1	r/w GE x x x SEL 0 x x x 0 0 0 0 Res	

Table 33–13: Graphic bus interface (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
GD	Graphic Bus Data Register	0x040	r/w Data 0 0x00 Res	23.2.
GC	Graphic Bus Control Register	0x044	r/w TIM E BSY SEQ DTA 0 0x00 Res	

Table 33–14: Hardware options registers (base addr. 0xF90100)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
T0C	Timer 0 Clock	0x0C0	w: x x x Clock Options f1 to f31 x x x 0x01 Res	31.3.
T1C	Timer 1 to 4 Clock	0x0C1	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
T2C		0x0C2		
T3C		0x0C3		
T4C		0x0C4		
CO00C	Clock Out0: Mux0 Pre. & Clock	0x0C5	w: x PRE Clock Options f0 to f31 (all) x 0x0 0x11 Res	
CO01C	Clock Out0: Mux1 to Mux3 Clock	0x0C6	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
RZPC	Rotor Zero Position Detection Clock	0x0C7	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
DMAC	DMA Timer Clock	0x0C8	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
CO1C	Clock Out1: Pre. & Clock	0x0C9	w: x PRE Clock Options f0 to f31 (all) x 0x0 0x11 Res	
C0C	CAPCOM Counter Clocks	0x0CA	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
C1C		0x0CB		
DC	DIGITbus Clock	0x0CC		
LC	LCD Pre. & Clock	0x0CD	w: x PRE Clock Options f0 to f31 (all) x 0x0 0x03 Res	
AC	AM Clock	0x0CE	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	
PF0C	PFM 0 Clock	0x0CF	w: x x x Clock Options f0 to f31 (all) x x x 0x0 Res	

Table 33–14: Hardware options registers (base addr. 0xF90100)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
SMC	SM, SPI0, SPI1 Pre. & SM Clock	0x0D0	w: x | PRE | Clock Options f0 to f31 (all) x 0x0 0x0 Res	31.3.
SP0C	SPI0 I/O & $F0_{SPI}$ Clock	0x0D1	w: SPI0OUT | SPI0IN | x | Clock Options f0 to f31 (all) 0 0 x 0x0 Res	
SP1C	SPI1 I/O & $F1_{SPI}$ Clock	0x0D2	w: SPI1OUT | SPI1IN | x | Clock Options f0 to f31 (all) 0 0 x 0x0 Res	
SP2C	$F2_{SPI}$ Clock	0x0D3	w: x | x | x | Clock Options f0 to f31 (all) x x x 0x0 Res	
P9C	PWM Clock	0x0D4	w: x | x | x | Clock Options f0 to f31 (all) x x x 0x0 Res	
P11C		0x0D6		
P1C		0x0D8		
P3C		0x0DA		
P5C		0x0DC		
P7C		0x0DE		
P9P	PWM Period	0x0D5	w: x | x | x | Clock Options f0 to f31 (all) x x x 0x08 Res	
P11P		0x0D7		
P1P		0x0D9		
P3P		0x0DB		
P5P		0x0DD		
P7P		0x0DF		
PM	Port Multiplexer	0x0E9	w: U15 | CC4I | CACO | U20 | U06 | H7 | H0 | PINT 0 0 0 1 0 0 0 0 Res	
UA0	UARTs	0x0EC	w: U0TX | U0RX | x | x | x | x | x | x 0 0 x x x x x x Res	
UA1		0x0ED	w: U1TX | U1RX | x | x | x | x | x | x 0 0 x x x x x x Res	

Table 33–15: High-current ports (base addr. 0xF90400)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
H0D	High Current Port Data Register	0x0C0		14.5.
H1D		0x0C8		
H2D		0x0D0		
H3D		0x0D8		
H4D		0x0E0		
H5D		0x0E8		
H6D		0x0F0		
H7D		0x0F8		
H0TRI	High Current Port Tristate Register	0x0C1		
H1TRI		0x0C9		
H2TRI		0x0D1		
H3TRI		0x0D9		
H4TRI		0x0E1		
H5TRI		0x0E9		
H6TRI		0x0F1		
H7TRI		0x0F9		
H0NS	High Current Port Normal/Special Register	0x0C2		
H1NS		0x0CA		
H2NS		0x0D2		
H3NS		0x0DA		
H4NS		0x0E2		
H5NS		0x0EA		
H6NS		0x0F2		
H7NS		0x0FA		

Table 33–15: High-current ports (base addr. 0xF90400)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
H0LVL	High Current Port Level Register	0x0C3	r/w: x x x x A3 A2 A1 A0 x x x x 0 0 0 0 Res	14.5.
H1LVL		0x0CB		
H2LVL		0x0D3		
H3LVL		0x0DB		
H4LVL		0x0E3		
H5LVL		0x0EB		
H6LVL		0x0F3		
H7LVL		0x0FB		
H0PIN	High Current Port Pin Register	0x0C4	r: x x x x P3 P2 P1 P0 x x x x 0 0 0 0 Res	
H1PIN		0x0CC		
H2PIN		0x0D4		
H3PIN		0x0DC		
H4PIN		0x0E4		
H5PIN		0x0EC		
H6PIN		0x0F4		
H7PIN		0x0FC		

Table 33–16: I2C-bus master interfaces (base addr. 0xF90100)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
I2CWS00	I2C Write Start Register 0	0x000	w: I2C Address; 0x00; Res	26.2.
I2CWS01		0x010		
I2CWS10	I2C Write Start Register 1	0x001	w: I2C Address; 0x00; Res	
I2CWS11		0x011		
I2CWD00	I2C Write Data Register 0	0x002	w: I2C Data; 0x00; Res	
I2CWD01		0x012		
I2CWD10	I2C Write Data Register 1	0x003	w: I2C Data; 0x00; Res	
I2CWD11		0x013		
I2CWP00	I2C Write Stop Register 0	0x004	w: I2C Data; 0x00; Res	
I2CWP01		0x014		
I2CWP10	I2C Write Stop Register 1	0x005	w: I2C Data; 0x00; Res	
I2CWP11		0x015		
I2CRD0	I2C Read Data Register	0x006	r: I2C Data; 0x00; Res	
I2CRD1		0x016		
I2CRS0	I2C Read Status Register	0x007	r: x, OACK, AACK, DACK, BUSY, WFH, RFE, x; 0 0 0 0 0 0 0 0; Res	
I2CRS1		0x017		
I2CM0	I2C Mode Register	0x00B	w: DGL, SPEED; 1 0x02; Res	
I2CM1		0x01B		

Table 33–17: Interrupt controller unit (base addr. 0xFFFF00)

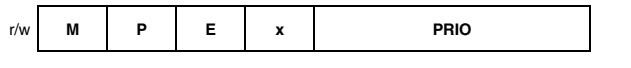
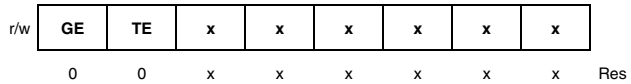
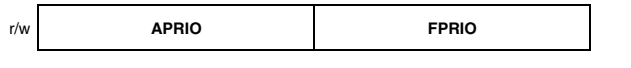
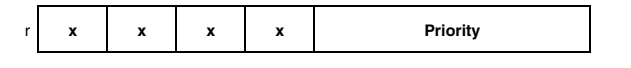
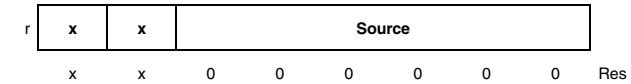
Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
ISN0	Interrupt Source Node Register 0	0x000		11.3.
:	:	:		
ISN39	Interrupt Source Node Register 39	0x027		
CRI	Control Register IRQ	0x0C0		
AFP	Actual and Forced Priority Register	0x0C1		
PEPRIO	Priority Encoder Priority output	0x0C2		
PESRC	Priority Encoder Source output	0x0C3		
VTB	Vector Table Base	0x0C4		

Table 33–18: JTAG (base addr. 0xF90000)

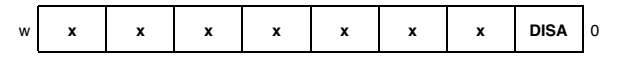
Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
DBG	Debug Register	0x061		8.2.

Table 33–19: LCD (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration	Section
			7 6 5 4 3 2 1 0	
ULCDLD	Universal Port LCD Load Register	0x010	w: LCDSLV x x x x x x x 0 0 0 0 0 0 0 0 Res	21.2.

Table 33–20: Patch module (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration	Section
			7 6 5 4 3 2 1 0	
PAR	Patch Address Register	0x000	w: x x x x x x x x 3 w: x A22 to A16 2 w: A15 to A8 1 w: A7 to A2 x x 0 0x00FFFFFF Res	10.2.
PDR	Patch Data Register	0x004	w: D31 to D24 3 w: D23 to D16 2 w: D15 to D8 1 w: D7 to D0 0 0x00000000 Res	
PER	Patch Enable Register	0x008	w: x x x x x x x x w: x x x x x x x x w: x x x x x PSEL9 PSEL8 PSEL7 w: PSEL6 PSEL5 PSEL4 PSEL3 PSEL2 PSEL1 PSEL0 PMEN 0x0000 Res	

Table 33–21: Port interrupts (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
IRPM0	Interrupt Port Mode Register 0	0x02A	r/w: PIT3 PIT2 PIT1 PIT0 0 0 0 0 0 0 0 0 Res	13.
IRPM1	Interrupt Port Mode Register 1	0x02B	r/w: x x x x PIT5 PIT4 x x x x 0 0 0 0 Res	

Table 33–22: Power-saving module (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
WUS	Wake-Up Source Register	0x050	r/w: RTC x x x x x WP9 WP8 1 r/w: WP7 WP6 WP5 WP4 WP3 WP2 WP1 WP0 0 No HW reset Res	7.2.
SSR	Sub Second Reload Register	0x054	r/w: x x x x x x x x 3 r/w: x x x x Bit 19 to 16 2 r/w: Bit 15 to 8 1 r/w: Bit 7 to 0 0 No HW reset Res	
SSC	Sub Second Counter	0x058	r: x x x x x x x x 3 r: x x x x Bit 19 to 16 2 r: Bit 15 to 8 1 r: Bit 7 to 0 0 No HW reset Res	
RTC	Real Time Counter	0x05C	r/w: x x x x x x x x 3 r/w: x x x HR 2 r/w: x x MIN 1 r/w: x x SEC 0 No HW reset Res	

Table 33–22: Power-saving module (base addr. 0xF90500)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
WPM0	Wake Port Mode Register	0x060	r/w: x | MOD1 | x | MOD0 — 0 No HW reset — Res	7.2.
WPM2		0x061		
WPM4		0x062		
WPM6		0x063		
WPM8		0x064		
WSC	Wake Source Control	0x068	r/w: x | x | x | x | x | AST | RTC | P — 0 0x00 after UVDD power-up — Res	
OSC	Oscillator Source Register	0x070	r/w: RC | XK | XM | x | LD | x | SRC — 0 1 | x | 1 — No HW reset — Res	
RTCC	RTC Control Register	0x074	r/w: x | x | x | SEL — 0 No HW reset — Res	
POL	Polling Register	0x078	r/w: x | CLK | x | PER — 1 r/w: ENA | OE | x | DEL — 0 0x00 — Res	
SMX	Signal Multiplexer Register	0x07C	r/w: BYP | x | x | x | x | MUX — 0 0x00 — Res	

Table 33–23: Pulse frequency modulator (base addr. 0xF90100)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
PFM0	Pulse Width and Period Length Register	0x050	w: INV | x | x | x | x | x | x | x — 3 w: Pulse Width — 2 w: Period Length (High Byte) — 1 w: Period Length (Low Byte) — 0 0x00 — Res	18.2.
PFM1		0x054		

Table 33–24: Pulse width modulator (base addr. 0xF90100)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
PWM0	PWM Register	0x040	w: Pulse width value 0 0 0 0 0 0 0 0 Res	17.2.
PWM1		0x041		
PWM2		0x042		
PWM3		0x043		
PWM4		0x044		
PWM5		0x045		
PWM6		0x046		
PWM7		0x047		
PWM8		0x048		
PWM9		0x049		
PWM10		0x04A		
PWM11		0x04B		
PWMC	PWM Control Register	0x04F	w: x x P1611 P169 P167 P165 P163 P161 x x 0 0 0 0 0 0 Res	

Table 33–25: Serial synchronous peripheral interfaces (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
SPI0D	SPI Data Register	0x010	r/w: Bit 7 to 0 of Rx/Tx Data 0 0 0 0 0 0 0 0 Res	24.2.
SPI1D		0x012		
SPI0M	SPI Mode Register	0x011	r/w: BIT8 LEN9 RXSEL INTERN SCLK CSF 0 0 0 0 0 0 0 0 Res	
SPI1M		0x013		

Table 33–26: Stepper motor VDO (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
SMVC	Stepper Motor VDO, Control Register	0x05A	w: x x SEL x QUAD x x 0 0 0 x 0 0 Res	20.2.
SMVSIN	Stepper Motor VDO, Sine Register	0x05B	r: x x x x x x x BUSY w: 8bit Sine Value 0 0 0 0 0 0 0 0 Res	
SMVCOS	Stepper Motor VDO, Cosine Register	0x05C	w: 8bit Cosine Value 0 0 0 0 0 0 0 0 Res	
SMVCMP	Stepper Motor VDO, Back-Up Comparator Register	0x05D	r/w: x ACRF ACRD ACRB ACRG ACRE ACRC ACRA x 0 0 0 0 0 0 0 Res	
SMVMUX	Stepper Motor VDO, Multiplexer Register	0x05E	r/w: E x x D A 1 r/w: B C F G 0 0x0000 Res	

Table 33–27: Test registers (base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
TSTAD2	Test Register AD2	0x0F8	w: For testing purposes only 0 0 0 0 0 0 0 0 Res	6.6.
TSTAD3	Test Register AD3	0x0F9		
TST5	Test Register 5	0x0FB		
TST4	Test Register 4	0x0FC		
TST3	Test Register 3	0x0FD		
TST1	Test Register 1	0x0FE		
TST2	Test Register 2	0x0FF		

Table 33–28: Timer (base addr. 0xF90000)

Mnemonic	Register Name	Offs	Register Configuration 7 6 5 4 3 2 1 0	Section
TIM0L	Timer 0 low byte	0x04E	r: Read low byte of down-counter and latch high byte w: Write low byte of reload value and reload down-counter 1 1 1 1 1 1 1 1 Res	16.
TIM0H	Timer 0 high byte	0x04F	r: Latched high byte of down-counter w: High byte of reload value 1 1 1 1 1 1 1 1 Res	
TIM1	Timer 1 Register	0x054	w: Reload value 0 0 0 0 0 0 0 0 Res	
TIM2	Timer 2 Register	0x055		
TIM3	Timer 3 Register	0x056		
TIM4	Timer 4 Register	0x057		

Table 33–29: Universal asynchronous receiver transmitters (Base addr. 0xF90000)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
UA0D	UART Data Register	0x0A0	r: Receive register w: Transmit register x x x x x x x x Res	25.3.
UA1D		0x018		
UA0C	UART Control and Status Register	0x0A1	r: RBUSY BRKD FRER OVRR PAER EMPTY FULL TBUSY 0 x x 0 x 1 0 0 Res w: x x x x STPB ODD PAR LEN x x x x 0 0 0 0 Res	
UA1C		0x019		
UA0BR0	UART Baud Rate Register Low Byte	0x0A2	w: Bit 7 to 0 of Baud Rate 0 0 0 0 0 0 0 0 Res	
UA1BR0		0x01A		
UA0BR1	UART Baud Rate Register High Byte	0x0A3	w: x x x Bit 12 to 8 of Baud Rate - - - 0 0 0 0 0 Res	
UA1BR1		0x01B		
UA0IM	UART Interrupt Mask Register	0x0A4	w: x x x x x ADR BRK RCVD - - - - - 0 0 0 Res	
UA1IM		0x01C		
UA0CA	UART Compare Address Register	0x0A5	w: Bit 7 to 0 of address 0 0 0 0 0 0 0 0 Res	
UA1CA		0x01D		
UA0IF	UART Interrupt Flag Register	0x0A6	r: Test Test Test Test Test ADR BRK RCVD - - - - - x 0 0 Res	
UA1IF		0x01E		

Table 33–30: Universal ports (base addr. 0xF90400)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
U0D	Universal Port Data/ Segment 0 Register	0x000	r/w D7 D6 D5 D4 D3 D2 D1 D0 Port r/w SG7_0 SG6_0 SG5_0 SG4_0 SG3_0 SG2_0 SG1_0 SG0_0 LCD 0 0 0 0 0 0 0 0 Res	14.3.
U1D		0x010		
U2D		0x020		
U3D		0x030		
U4D		0x040		
U5D		0x050		
U6D		0x060		
U7D		0x070		
U8D		0x080		
U0TRI	Universal Port Tristate/Segment 1 Register	0x001	r/w T7 T6 T5 T4 T3 T2 T1 T0 Port r/w SG7_1 SG6_1 SG5_1 SG4_1 SG3_1 SG2_1 SG1_1 SG0_1 LCD 1 1 1 1 1 1 1 1 Res	
U1TRI		0x011		
U2TRI		0x021		
U3TRI		0x031		
U4TRI		0x041		
U5TRI		0x051		
U6TRI		0x061		
U7TRI		0x071		
U8TRI		0x081		
U0NS	Universal Port Normal-Special/Segment 2 Register	0x002	r/w S7 S6 S5 S4 S3 S2 S1 S0 Port r/w SG7_2 SG6_2 SG5_2 SG4_2 SG3_2 SG2_2 SG1_2 SG0_2 LCD 0 0 0 0 0 0 0 0 Res	
U1NS		0x012		
U2NS		0x022		
U3NS		0x032		
U4NS		0x042		
U5NS		0x052		
U6NS		0x062		
U7NS		0x072		
U8NS		0x082		

Table 33–30: Universal ports (base addr. 0xF90400)

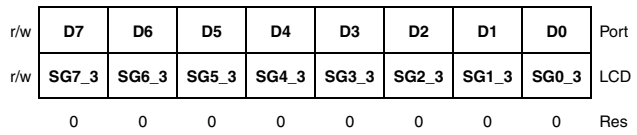
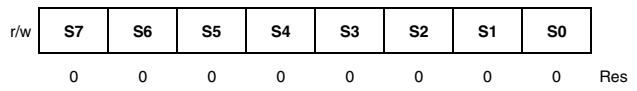
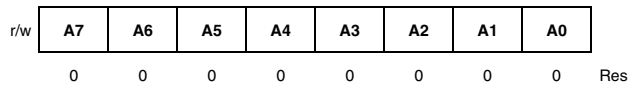
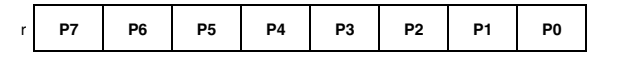
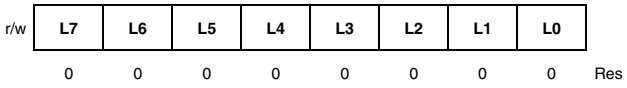
Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
U0DPM	Universal Port Double Pull-Down Mode/ Segment 3 Register	0x003		14.3.
U1DPM		0x013		
U2DPM		0x023		
U3DPM		0x033		
U4DPM		0x043		
U5DPM		0x053		
U6DPM		0x063		
U7DPM		0x073		
U8DPM		0x083		
U0SLOW	Universal Port Slow Mode Register	0x004		
U1SLOW		0x014		
U2SLOW		0x024		
U3SLOW		0x034		
U4SLOW		0x044		
U5SLOW		0x054		
U6SLOW		0x064		
U7SLOW		0x074		
U8SLOW		0x084		
U0LVL	Universal Port Level Register	0x005		
U1LVL		0x015		
U2LVL		0x025		
U3LVL		0x035		
U4LVL		0x045		
U5LVL		0x055		
U6LVL		0x065		
U7LVL		0x075		
U8LVL		0x085		

Table 33–30: Universal ports (base addr. 0xF90400)

Mnemonic	Register Name	Offs.	Register Configuration 7 6 5 4 3 2 1 0	Section
U0PIN	Universal Port Pin Register	0x006		14.3.
U1PIN		0x016		
U2PIN		0x026		
U3PIN		0x036		
U4PIN		0x046		
U5PIN		0x056		
U6PIN		0x066		
U7PIN		0x076		
U8PIN		0x086		
U0MODE	Universal Port Mode Register	0x007		
U1MODE		0x017		
U2MODE		0x027		
U3MODE		0x037		
U4MODE		0x047		
U5MODE		0x057		
U6MODE		0x067		
U7MODE		0x077		
U8MODE		0x087		

34. Control Register and Memory Interface

34.1. Control Register CR

When exiting Reset, the device will start up in a configuration defined by the CR setting. For details on how to set the CR see chapter “Core Logic”.

A full description of the functionality of all CR bits is given below. Among others, the CR allows to configure the memory interface for connection to a variety of external memories.

CR Control Register

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	x	x	x	3
r/w	STPCLK	RESLNG	x	x	x	TSTTOG	EWE	PSA	2
r/w	EB2	TFT	TETM	EB1	EBW	EASY	MFM		1
r/w	JTAG	ENDIAN	MAP		IBOOT	IROM	IRAM	ICPU	0
	Value of memory location 0x20 to 0x23								Res

The upper half word of register CR is loaded from location 0x22/0x23 only if flag EBW is at zero. If EBW is at one, the upper half word is initialized to 0xFFFB.

STPCLK **Stop Clock** (Emu parts only)
r/w1: Peripheral clocks are stopped in debug mode.
r/w0: Peripheral clocks are always active.
Timers are stopped with a resolution of $1/f_0$.

RESLNG **Reset Pulse Length**
r/w1: Pulse length is $8/F_{XTAL}$
r/w0: Pulse length is $2048/F_{XTAL}$
This bit specifies the length of the reset pulse which is output at pin RESETQ following an internal reset. If pin TEST is 1 the first reset after power on is short. The following resets are as programmed by RESLNG. If pin TEST is 0 all resets are long.

TSTTOG **TEST2 Pin Toggle** (Table 34–9)
This bit is used for test purposes only. If TSTTOG is true in IC active mode, pin TEST2 can toggle the multifunction pins between bus mode and normal mode.

EWE **Emu Bus Write Enable**
r/w1: Enabled (No data security).
r/w0: Disabled.
This flag has to be set to allow Emu bus writes or Flash programming (MCM). For details please refer to section “Device Lock Module (DLM)”.

PSA **Program Storage Access**
r/w1: 16-bit access.
r/w0: 32-bit access.
This bit allows, in EMU parts, to set the data bus access width to ROM and Flash program storage (Table 34–2).

EB2 **External Bus Flag 2** (Table 34–1)
r/w1: CE0Q and CE1Q select two external chips.
r/w0: OEQ and WEQ select one external chip connected to CE0Q (do not use CE1Q). For illustration see Fig. 34–13.

TFT **Trace Bus Full Trace** (Emu parts only, Table 34–3)

TETM **Trace Bus ETM** (Emu parts only, Table 34–3)

EB1 **External Bus Flag 1** (Emu/MCM parts only, Table 34–1)
r/w1: Power-saving mode of memory interface. Emu Bus configured for external Flash memory. Pin signals FBUSQ, BWQ0 to 3 and CE1Q are disabled and pulled low weakly. In CPU SLOW mode pin signal CE0Q activates flash memory only for 1/128th of access cycle.
r/w0: Emu Bus configured for standard external Memory. CE0Q and CE1Q always enable memory for full access cycles.

EBW **Emu Bus Width** (Emu/MCM parts only, Tables 34–1, 34–2)
r/w1: Emu Bus configured for 16-bit-wide external memory. For illustration see Fig. 34–13. Bits CR.PSA, CR.STPCLK and CR.RESLNG are set to one and bit CR.TSTTOG is set to zero.
r/w0: Emu Bus configured for 32-bit-wide external memory.

EASY **Emu Bus in Asynchronous Mode** (Table 34–1) (Emu/MCM parts only)
r/w1: Emu Bus configured for asynchronous external memory.
r/w0: Emu Bus configured for synchronous external memory. In synchronous mode the address bus (A) and chip enable (CExQ) latches are transparent.

MFM **Multifunction pin Mode** (Tables 34–9)

JTAG **Application JTAG Interface**
r/w1: Enabled if TEST2 pin is high (Fig. 34–1)
r/w0 Disabled

ENDIAN **Endian setting ARM Core**
r/w1: Little endian.
r/w0: Big endian.
Don’t change this flag dynamically

MAP **Mapping** (Table 34–4)

IBOOT **Internal Special Function ROM** (Tables 34–5, 34–8)

IROM **Internal ROM** (Table 34–6)

IRAM **Internal RAM** (Tables 34–7, 34–8)

ICPU **Internal CPU**
r/w1: Enable internal CPU.
r/w0: Disable internal CPU

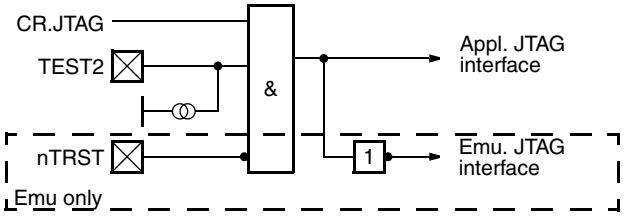


Fig. 34–1: Enabling JTAG interfaces

Table 34–1: Emu bus configuration for some commonly used external memories

EB2	EB1	EBW	EASY	External Memory Type	
				Program Memory (CE0Q)	Data or Special Function Memory (CE1Q)
1	0	0	0	32-Bit sync SRAM (e.g. MT55L256L32F)	
0	1	0	1	32-Bit async Flash (e.g. 2 x Am29F400BT or Am29LV400BT)	don't use
0	1	1	1	16-Bit async. Flash (e.g. Am29F400BT or Am29LV400BT)	

Table 34–2: Control of Emu bus timing by PSA and EBW

PSA	EBW	Bus Width	Number of cycles for 32-bit access
0	0	32 Bit	1 cycle
0	1	Don't use	
1	0	32 Bit	2 cycles
1	1	16 Bit	

Table 34–3: TETM and TFT usage

TFT	TETM	Trace Bus Mode	D0 to D31 active	ETM
1	1	Disabled (Gnd) (Except for DBGACK, nRESET, FSYS)	for external memory access only	Off
0	1	Analyzer	always	Off
1	0	ETM	for external memory access only	On
0	0	ETM	always	On

Table 34–4: MAP usage

MAP		Mapping Effect
1	0	
0	0	mirrors RAM base offset 0xC0.0000 to 0
0	1	maps ROM/Flash base offset 0x20.0000 to 0
1	x	mirrors special-function ROM base offset 0xF0.0000 to 0

Table 34–5: IBOOT usage

IBOOT	MFM		selected Special Function ROM source	
	1	0	QFP128	Emu
0	0	x	external via multifunction pins in bus mode	
	x	0		
	1	1	disable special-function ROM	ext. via Emu bus
1	x	x	internal special-function ROM	

Table 34–6: IROM usage

IROM	selected ROM/Flash source	
	QFP128	Emu
0	external via multifunction pins in bus mode	
1	internal ROM/Flash	external via Emu bus

Table 34–7: IRAM usage

IRAM	MFM		selected RAM source	
	1	0	QFP128	Emu
0	0	x	external via multifunction pins in Bus mode	
	x	0		
	1	1	disable RAM	ext. via Emu bus
1	x	x	internal RAM	

Table 34–8: CE1Q Selections

IRAM	IBOOT	CE1Q selects
0	x	external RAM
1	0	external special-function ROM
1	1	No external access

Table 34–9: TSTTOG and MFM usage in ROM/Flash parts

MFM		TSTTOG	TEST2 Pin	Multifunction Pins
1	0			
0	0	0	x	Bus mode 0
		1	0	Bus mode 0
			1	Port mode
0	1	0	x	Bus mode 1
		1	0	Bus mode 1
			1	Port mode
1	0	0	x	Bus mode 2
		1	0	Bus mode 2
			1	Port mode
1	1	x	x	Port mode

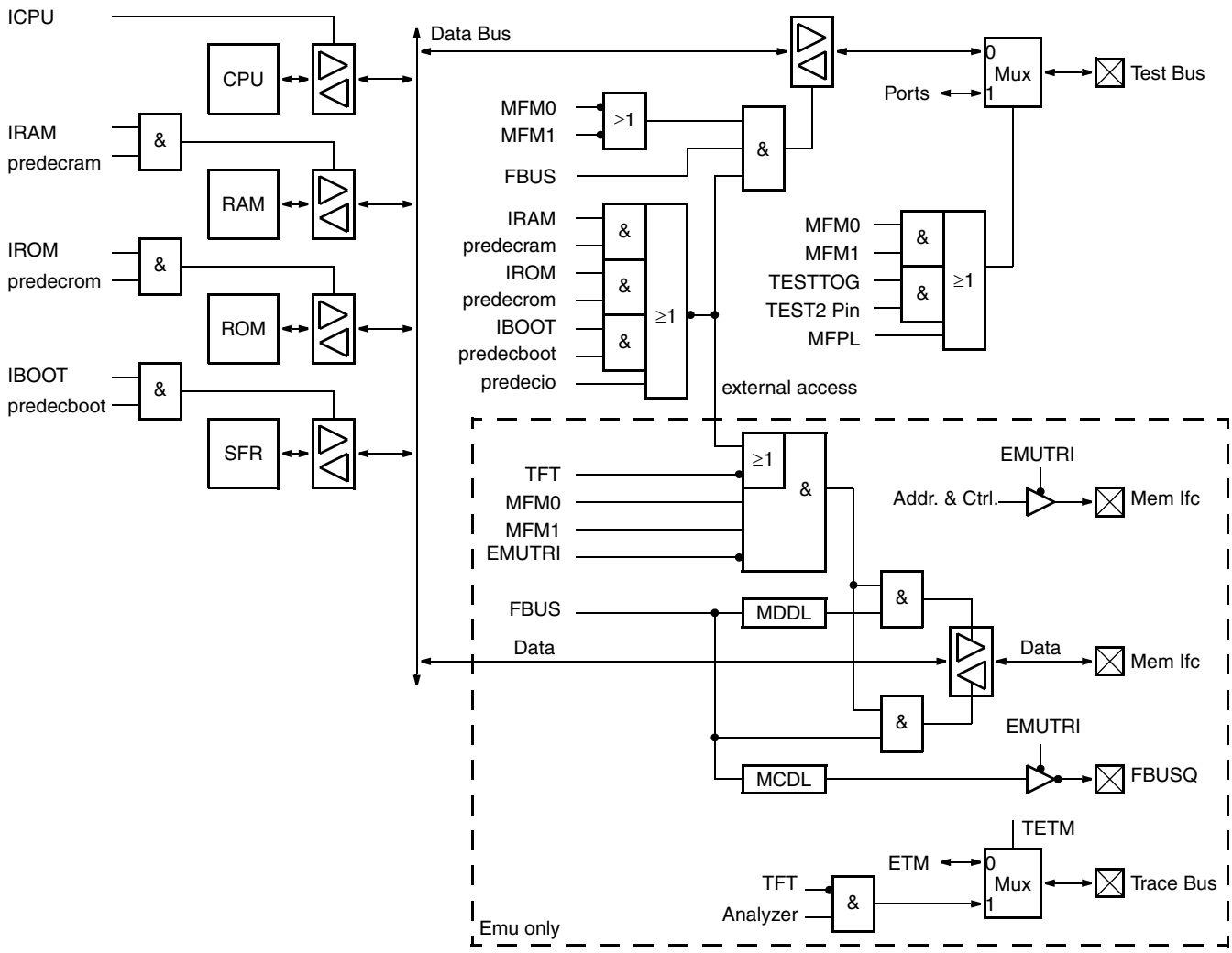


Fig. 34–2: Bus interfaces

34.2. Memory Clock Delay Lines

The memory clock delay lines allow access optimization to external synchronous memory. Only EMU parts need these registers to be configured.

MDDL and MCDL must only be modified in FAST mode, and only by increments of one by one.

MDL	Memory Delay								
	7	6	5	4	3	2	1	0	Offs
w	x	x	x	x	MDDL				1
w	x	x	x	x	MCDL				0
	0x0000								Res

MDDL **Memory Data Delay**
w0 to 15: Delay of data driven to memory port from 0 to 15 ns in 1 ns steps.

MCDL **Memory Clock Delay**
w0 to 15: Delay of clock FBUSQ for external memory from 0 to 15 ns in 1 ns steps.

34.3. External Memory Interface

34.3.1. Interfacing examples

EVDD = 5 V is required for the following interfacing examples.

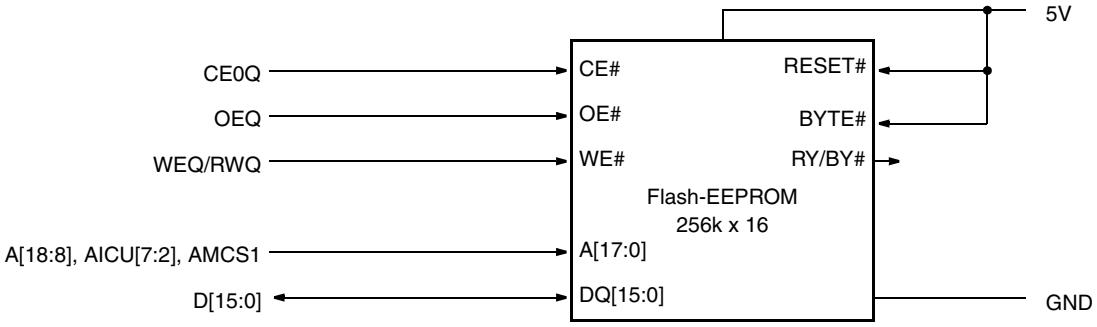


Fig. 34–3: Asynchronous Flash EEPROM (e.g. Am29F400B) as program memory

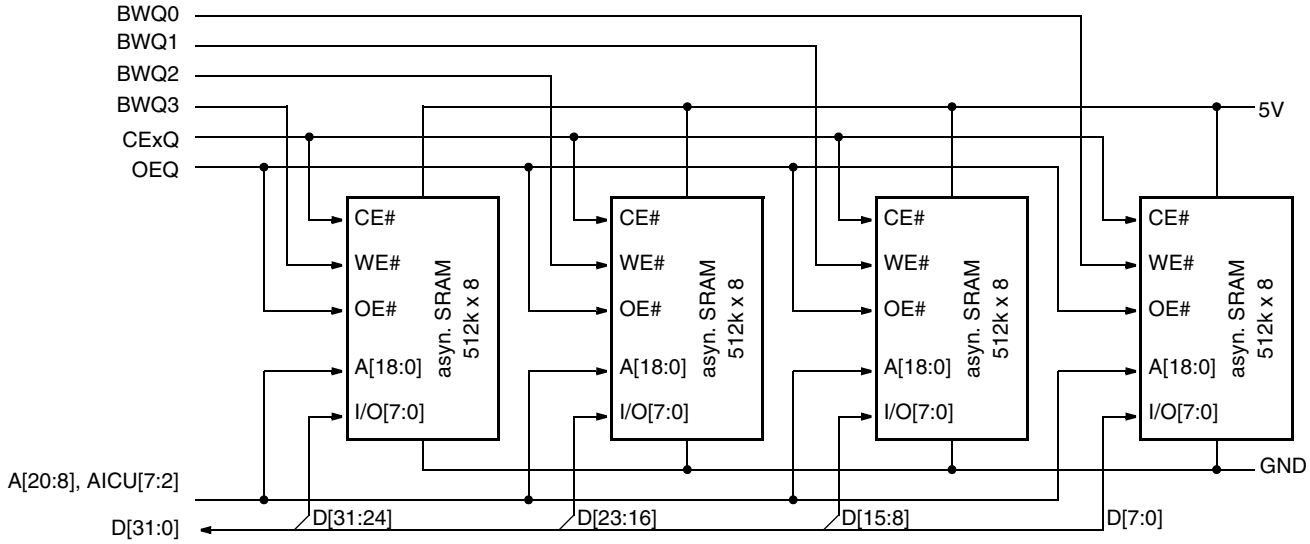


Fig. 34–4: Asynchronous SRAM (e.g. KM684002B) as emulation program memory

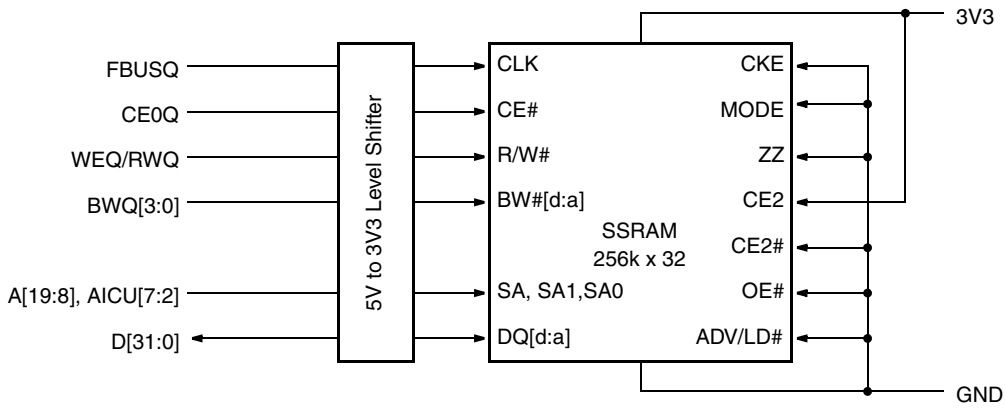


Fig. 34–5: Synchronous SRAM (e.g. MT55L256L32F) as emulation program memory

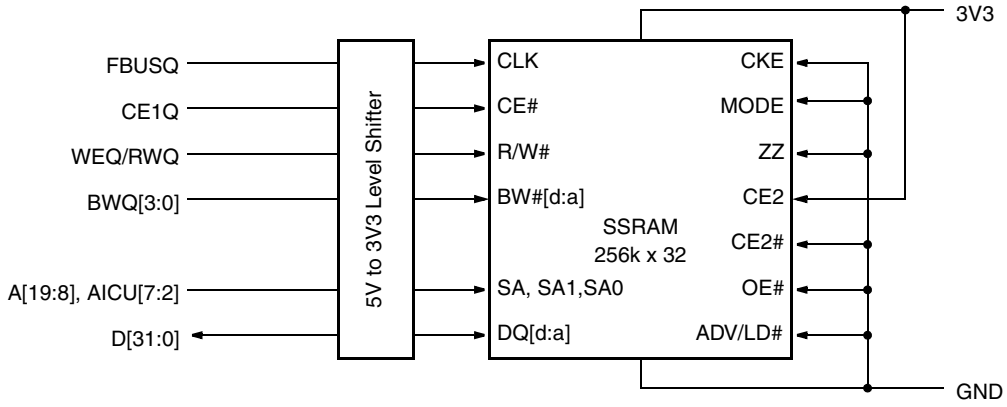


Fig. 34–6: Synchronous SRAM (e.g. MT55L256L32F) as emulation RAM or special-function memory

34.3.2. External Trace Interfacing

For a mapping of the IC pins to external trace tools see the "Specification of the Evaluation Board Kit (EVB)".

34.3.3. Memory Interface Characteristics

Table 34–10: UVSS = UVSS1 = FVSS = HVSSn = EVSSn = AVSS = 0 V, 3.5 V < AVDD = UVDD = UVDD1 < 5.5 V, 4.5 V < EVDDn < 5.5 V, TCASE = 0°C to 35°C, C_L = 70 pF

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
DFBUSQ	FBUSQ High to Low Ratio	47.5		52.5	%	PLL mode
Synchronous SRAM						
t_{sAS}	sync Address Setup Time	0		12.5	ns	
t_{sAH}	sync Address Hold Time		0		ns	
t_{sCES}	sync Chip Enable Setup	0		12.5	ns	
t_{sDSR}	sync Data Setup Read Time	18.5			ns	
t_{sDHR}	sync Data Hold Read Time		0		ns	
t_{sDSW}	sync Data Setup Write Time	0		10	ns	
t_{sDDT}	sync Data Drive Tristate		0		ns	
Asynchronous SRAM						
t_{aAS}	async Address Setup Time	0		2.5	ns	F_{SYS}<40MHz
t_{aAH}	async Address Hold Time		0		ns	
t_{aCES}	async Chip Enable Setup	0		2.5	ns	F_{SYS}<40MHz
t_{aOES}	async Output Enable Setup		0		ns	
t_{aBWS}	async Byte Write Setup		0		ns	
t_{aDSR}	async Data Setup Read	18.5			ns	
t_{aDHR}	async Data Hold Read Time		0		ns	

Table 34–10: UVSS = UVSS1 = FVSS = HVSSn = EVSSn = AVSS = 0 V, 3.5 V < AVDD = UVDD = UVDD1 < 5.5 V, 4.5 V < EVDDn < 5.5 V, TCASE = 0°C to 35°C, C_L = 70 pF

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{aDSW}	async Data Setup Write	0		10	ns	
t_{aDDT}	async Data Drive Tristate		0		ns	

Table 34–11: UVSS = UVSS1 = FVSS = HVSSn = EVSSn = AVSS = 0 V, 3.5 V < AVDD = UVDD = UVDD1 < 5.5 V, 4.5 V < EVDDn < 5.5 V, TCASE = –40°C to 85°C, C_L = 70 pF

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
DFBUSQ	FBUSQ High to Low Ratio	47.5		52.5	%	PLL mode
Synchronous SRAM						
t_{sAS}	sync Address Setup Time	0		12.5	ns	
t_{sAH}	sync Address Hold Time		0		ns	
t_{sCES}	sync Chip Enable Setup	0		12.5	ns	
t_{sDSR}	sync Data Setup Read Time	20.5			ns	
t_{sDHR}	sync Data Hold Read Time		0		ns	
t_{sDSW}	sync Data Setup Write Time	0		10	ns	
t_{sDDT}	sync Data Drive Tristate		0		ns	
Asynchronous SRAM						
t_{aAS}	async Address Setup Time	0		2.5	ns	F_{SYS}<40 MHz
t_{aAH}	async Address Hold Time		0		ns	
t_{aCES}	async Chip Enable Setup	0		2.5	ns	F_{SYS}<40 MHz
t_{aOES}	async Output Enable Setup		0		ns	
t_{aBWS}	async Byte Write Setup		0		ns	
t_{aDSR}	async Data Setup Read	20.5			ns	
t_{aDHR}	async Data Hold Read Time		0		ns	
t_{aDSW}	async Data Setup Write	0		10	ns	
t_{aDDT}	async Data Drive Tristate		0		ns	

Table 34–12: UVSS = UVSS1 = FVSS = HVSSn = EVSSn = AVSS = 0 V, 3.5 V < AVDD = UVDD = UVDD1 < 5.5 V, 3 V < EVDDn = FVDD < 3.6 V, TCASE = –40°C to 85°C, C_L = 10 pF

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
Asynchronous Flash						
t_{aAS}	async Address Setup Time	0		2.5	ns	F_{SYS}<40MHz
t_{aAH}	async Address Hold Time		0		ns	
t_{aCES}	async Chip Enable Setup	0		2.5	ns	F_{SYS}<40MHz
t_{aOES}	async Output Enable Setup		0		ns	
t_{aBWS}	async Byte Write Setup		0		ns	

Table 34–12: UVSS = UVSS1 = FVSS = HVSSn = EVSSn = AVSS = 0 V, 3.5 V < AVDD = UVDD = UVDD1 < 5.5 V, 3 V < EVDDn = FVDD < 3.6 V, TCASE = –40°C to 85°C, C_L = 10 pF

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{aDSR}	async Data Setup Read	18.5			ns	
t_{aDHR}	async Data Hold Read Time		0		ns	
t_{aDSW}	async Data Setup Write	0		10	ns	
t_{aDDT}	async Data Drive Tristate		0		ns	

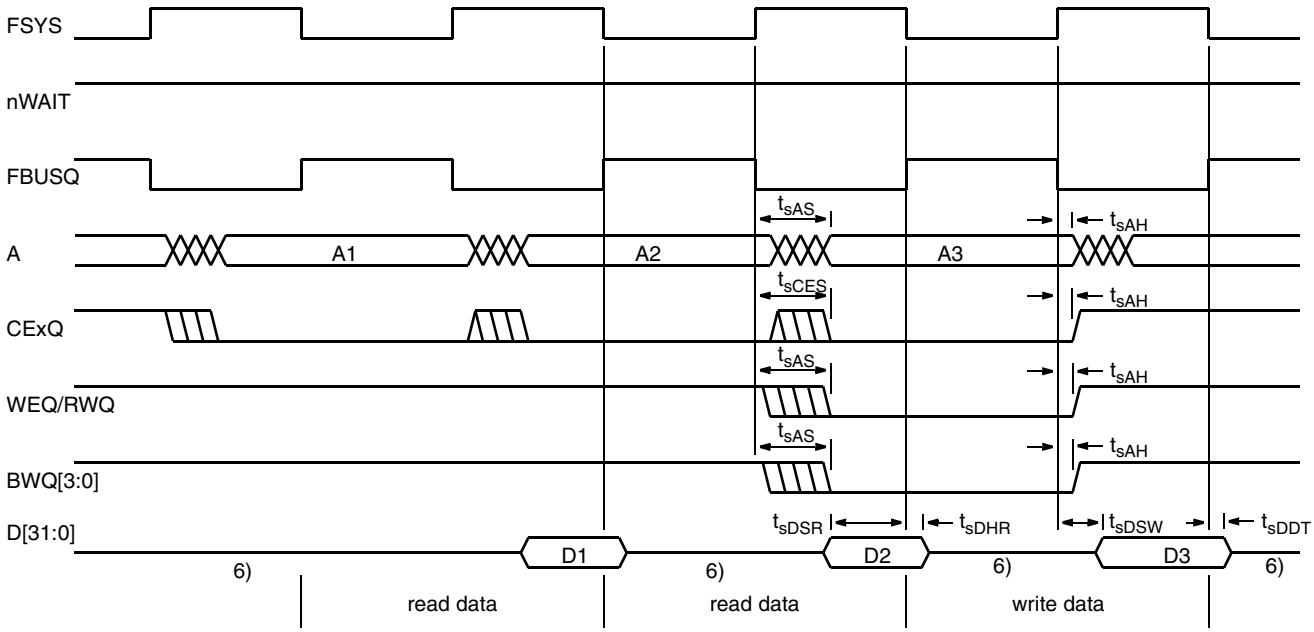


Fig. 34–7: Sync SRAM timing, 0 wait states

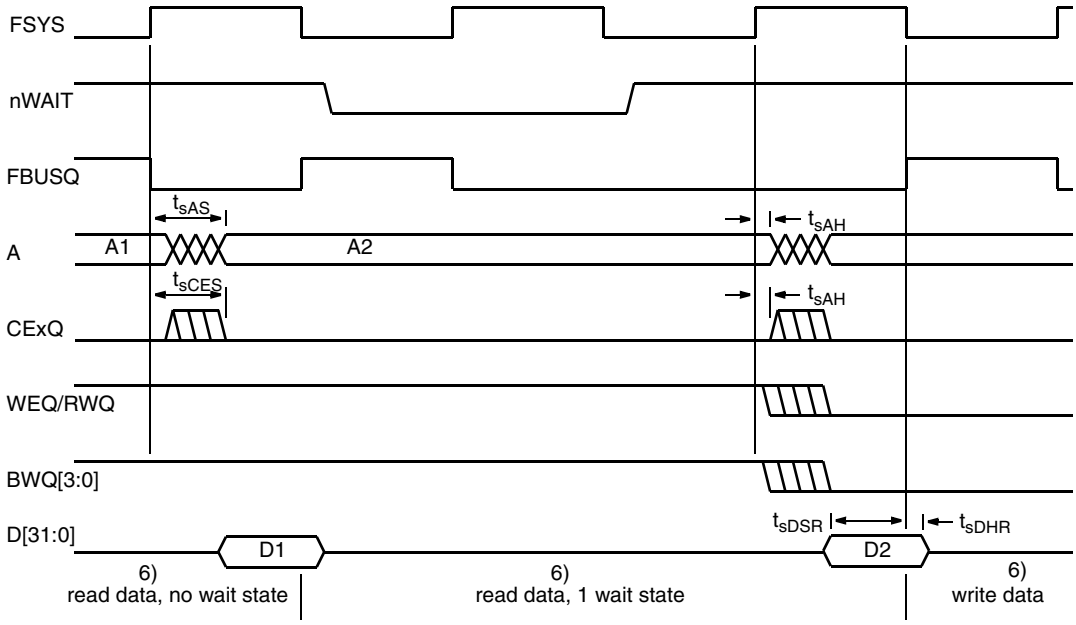


Fig. 34–8: Sync SRAM timing, read with wait state, followed by a write cycle

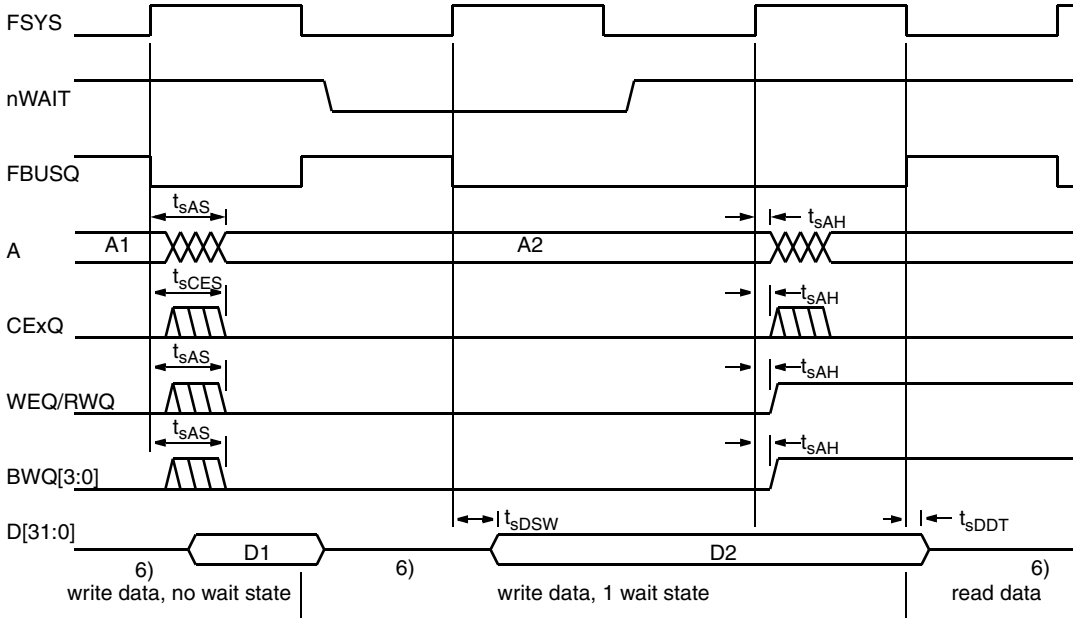


Fig. 34–9: Sync SRAM timing, write with wait state, followed by a read cycle

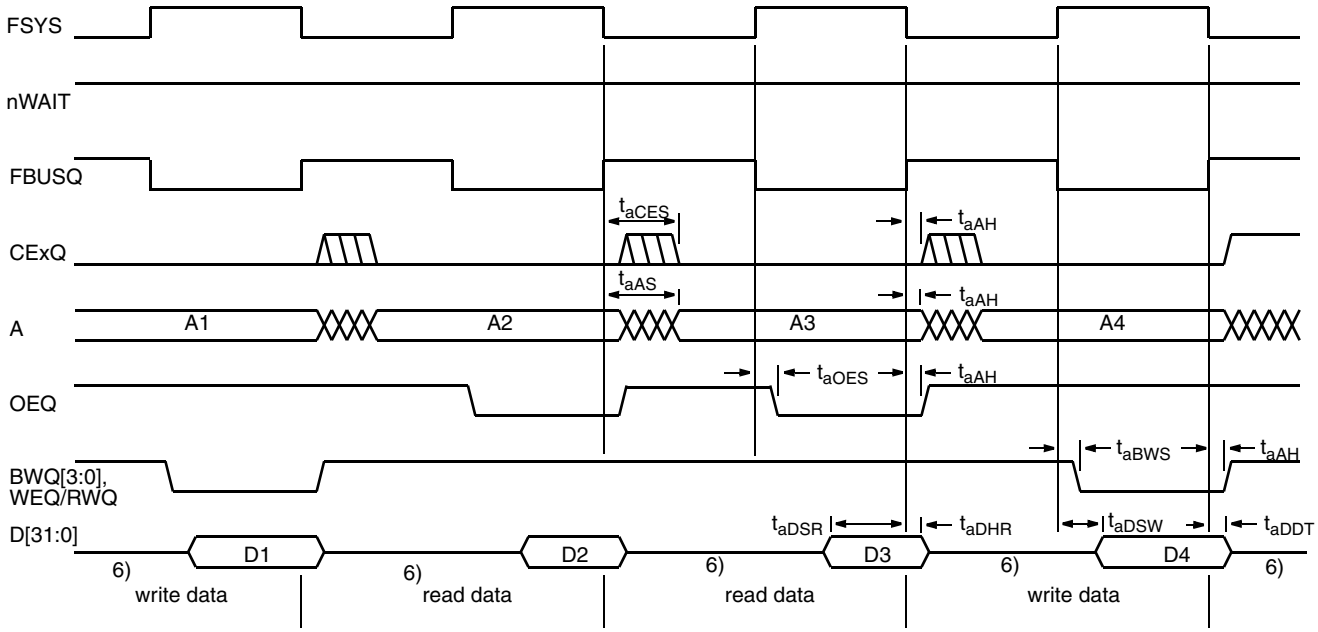


Fig. 34–10: Async SRAM/Flash timing, 0 wait states

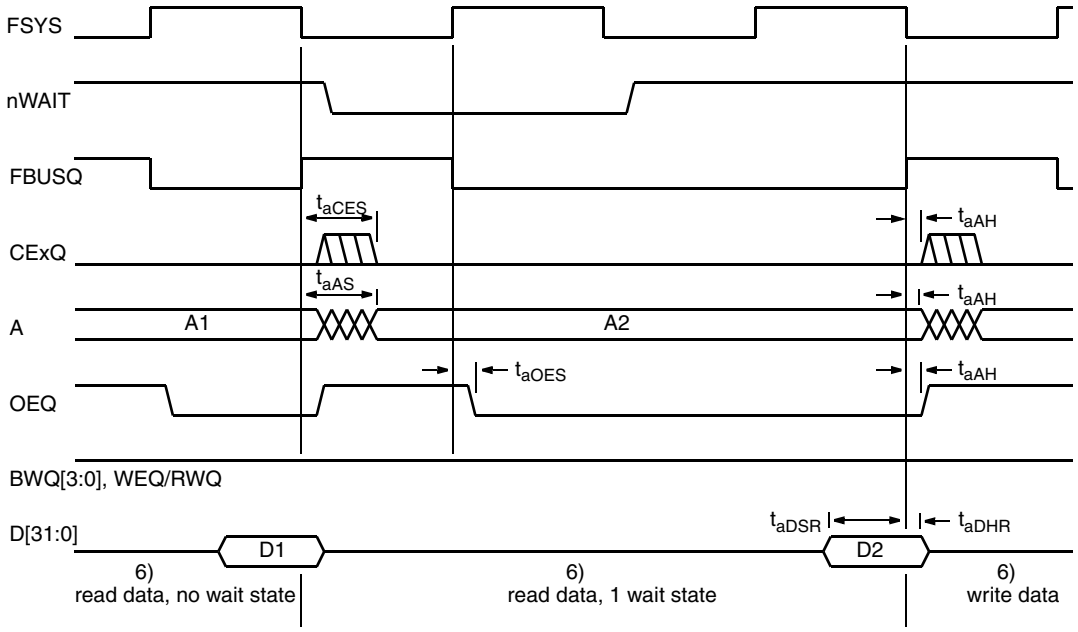


Fig. 34–11: Async SRAM/Flash timing, read with wait state, followed by a write cycle

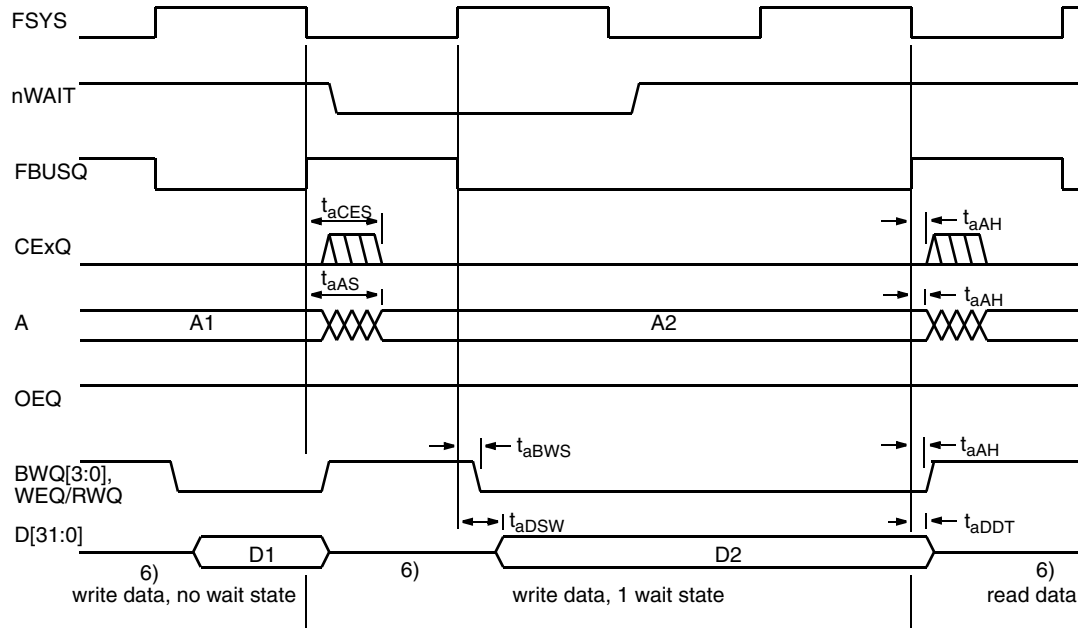


Fig. 34–12: Async SRAM/Flash timing, write with wait state, followed by a read cycle

[6]) During the high level of FBUSQ the previous data bus levels are weakly held. Thus the data bus is defined when the bus drivers are tristate and FBUSQ is high. See section ‘Electrical Characteristics’ for the weak hold currents for pull-down (Ipd) and for pull-up (Ipu).

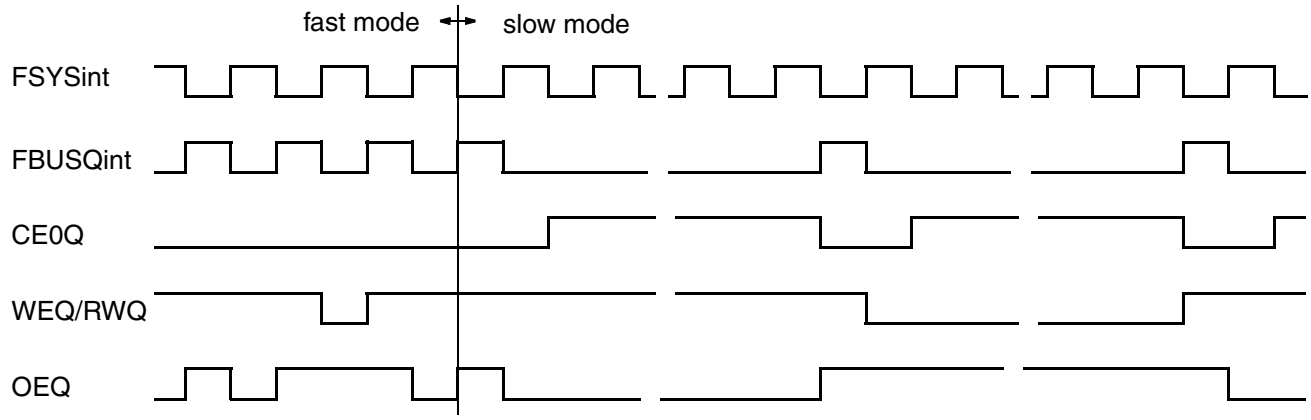


Fig. 34–13: CE0Q timing in PLL/FAST and SLOW/DEEP SLOW modes (CR.EB2 set to 0, CR.EB1 and CR.EASY set to 1)

CE0Q is used for low power mode. Input data are latched with the rising edge of CE0Q and are weakly held as long as CE0Q stays high.

35. Differences

This chapter describes differences between this document and the predecessor document “CDC32xxG-C Automotive Controller Family Hardware Manual, CDC3205G-C Automotive Controller Specification“ (6251-579-1PD).

Section	Description
Introduction	Table 1-1 changed.
Pins	Figure 2-3 changed.
Electrical Characteristics	Characteristics: Changed value: UI_{DDi}, UI_{DDp}, UI_{DDpe}, UI_{DDf}, t_{soci}, t_{hoci}, t_{soce}, t_{hoce}, t_{si}, t_{hi}, t_{btj}, t_{fed} Deleted parameters: t_{srx}, t_{dtx} New parameters: t_{lOd}, t_{so}, t_{ho} Figure 3-3: deleted.
CPU and Clock System	Figure 4-2 modified.
	Section 4.7.3.1 corrected.
Memory and SFR	Figure 5-1 corrected.
Core Logic	Figure 6-1 corrected.
	Section 6.5.5 clarified.
ICU	Features extended.
AVDD Analog Section	Section 15.7 clarified.
Stepper Motor Module	Section 20. re-ordered.
UART	Figure 25-1 changed.
I^2C	Figure 26-1 changed.
	Section 26.2 clarified.
CAN	Section 27.2.3 clarified.
Control Register and Memory Interface	Section 34.1. clarified.
	Table 34-1 simplified.
	Table 34-10, changed values: t_{sAS}, t_{sCES}, t_{sDSR}, t_{sDSW}, t_{sDDT}, t_{aAS}, t_{aCES}, t_{aDSR}, t_{aDSW}, t_{aDDT}
	Table 34-11, added parameters: DFBUSQ, t_{sAH}, t_{sCES}, t_{sDSR}, t_{sDHR}, t_{sDSW}, t_{sDDT}, t_{aAS}, t_{aAH}, t_{aCES}, t_{aOES}, t_{aBWS}, t_{aDSR}, t_{aDHR}, t_{aDSW}, t_{aDDT}
	Table 34-12 added.

36. Data Sheet History

1. Advance Information: “CDC32xxG-C V1.0 Automotive Controller - Family User Manual, CDC3205G-C Automotive Controller Specification”, Feb. 21, 2002, 6251-579-1AI.
First release of the advance information.
Originally created for HW versions CDC3205G-C1 and CDC3207G-C1.

2. Advance Information: “CDC32xxG-C V2.0 Automotive Controller - Family User Manual, CDC 3205G-C Automotive Controller Specification”, June 6, 2002, 6251-579-2AI.
Second release of the advance information.
Originally created for HW versions CDC3205G-C2 and CDC3207G-C2.

3. Preliminary Data Sheet: “CDC32xxG-C Automotive Controller - Family User Manual, CDC 3205G-C Automotive Controller Specification”, June 12, 2003, 6251-579-1PD.
First release of the preliminary data sheet.
Originally created for HW version CDC3205G-C2.

4. Data Sheet: “CDC 32XXG-C Automotive Controller - Family User Manual, CDC 3205G-C Automotive Controller Specification”, Feb. 10, 2005, 6251-579-1DS.
First release of the data sheet.
Originally created for HW version CDC3205G-C2.

Micronas GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@micronas.com
Internet: www.micronas.com

Printed in Germany
Order No. 6251-579-1DS