ADVANCE INFORMATION

CDC16xxF-E Automotive Controller Family User Manual

CDC1605F-E Automotive Controller Emulator Specification

Edition March 31, 2003
6251-606-2AI

MICRONAS

Contents

Contents, continued

Contents, continued

1. Introduction

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

The IC is a single-chip controller for use in automotive applications. The CPU on the chip is an upgrade of the 65C02 with 16-bit internal data and 24-bit address bus. The chip consists of timer/counters, an interrupt controller, a multichannel A/D converter, a stepper motor and LCD driver, CAN interfaces and PWM outputs.

1.1. Features

Table 1–1: CDC16xxF Family Feature List

	This Document:							
Item	**CDC1605F-E EMU**	**CDC1607F-E MCM Flash**	**CDC1631F-E MASK ROM**	**CDC1605F-C EMU**	**CDC1607F-C MCM Flash**	**CDC1641F-C Mask ROM**	**CDC1652F-C Mask ROM**	**CDC1672F-C Mask ROM**
Core								
CPU	16-bit 65C816, featuring software compatibility with its 8-bit NMOS and CMOS 6500-series predecessors							
CPU-Active Operation Modes	FAST, SLOW and DEEP SLOW			FAST and SLOW				
Power Saving Modes (CPU Inactive)	WAKE and IDLE			-				
EMI Reduction Mode	selectable in FAST mode							
Oscillators	4 MHz to 12 MHz Quartz, RC			4 MHz to 12 MHz Quartz				
RAM	6 KB		2 KB	6 KB		2.75 KB	4 KB	6 KB
ROM	ROMless, external program storage with up to 16 MB, internal 2 KB Boot ROM	256 KB Flash, bottom boot configuration, internal 2 KByte Boot ROM	64 KB	ROMless, external program storage with up to 16 MB, internal 2 KB Boot ROM	256 KB Flash, bottom boot configuration, internal 2 KB Boot ROM	90 KB	128 KB	216 KB

Table 1–1: CDC16xxF Family Feature List, continued

Item	This Document: CDC1605F-E EMU	CDC1607F-E MCM Flash	CDC1631F-E MASK ROM	CDC1605F-C EMU	CDC1607F-C MCM Flash	CDC1641F-C Mask ROM	CDC1652F-C Mask ROM	CDC1672F-C Mask ROM
Multiplier, 8 by 8 bit	✔			-				
Digital Watchdog	✔							
Central Clock Divider	✔							
Interrupt Controller expanding NMI	16 inputs,16 priority levels							
Port Interrupts including Slope Selection	4 inputs							
Port Wake-Up Inputs including Slope / Level Selection	✔			-				
Patch Module	10 ROM locations		5 ROM locations	10 ROM locations		5 ROM locations	6 ROM locations	
Boot System	allows in-system downloading of code and data into RAM via serial link		-	allows in-system downloading of code and data into RAM via serial link		-	-	-
Analog								
Reset/Alarm	Combined Input for Regulator Input Supervision							
Clock and Supply Supervision	✔							
10-bit ADC, charge balance type	9 channels (5 channels selectable as digital input)							
ADC Reference	VREF Pin							
Comparators	P06COMP with 1/2 AVDD reference							
LCD	Internal processing of all analog voltages for the LCD driver							

Table 1–1: CDC16xxF Family Feature List, continued

Item	This Document: CDC1605F-E EMU	CDC1607F-E MCM Flash	CDC1631F-E MASK ROM	CDC1605F-C EMU	CDC1607F-C MCM Flash	CDC1641F-C Mask ROM	CDC1652F-C Mask ROM	CDC1672F-C Mask ROM
Communication								
DMA	1 DMA Channel for serving the Graphics Bus interface		-	1 DMA Channel for serving the Graphics Bus interface		-	1 DMA Channel for serving the Graphics Bus interface	
UART	3: UART0, UART1 and UART2		1: UART0	3: UART0, UART1 and UART2		1: UART0	3: UART0, UART1 and UART2	
Synchronous Serial Peripheral Interfaces	2: SPI0 and SPI1		1: SPI0	2: SPI0 and SPI1		1: SPI0	2: SPI0 and SPI1	
Full CAN modules V2.0B	3: CAN0, CAN1 and CAN2 with 256-byte object RAM each (LCAN000F)		1: CAN0 with 256-byte object RAM (LCAN000F)	3: CAN0, CAN1 and CAN2 with 256-byte object RAM each (LCAN0009)		1: CAN0 with 256-byte object RAM (LCAN0009)	2: CAN0 and CAN1 with 256-byte object RAM each (LCAN0009)	
DIGITbus	1 master module		-	1 master module		-	1 master module	
Input & Output								
Universal Ports selectable as 4:1 mux LCD Segment/Backplane lines or Digital I/O Ports	up to 52 I/O or 48 LCD segment lines (=192 segments), in groups of two configurable as I/O or LCD							
Universal Port Slew Rate	HW preselectable							
Stepper Motor Control Modules with High-Current Ports	5 Modules, 24 dI/dt controlled ports							
8-bit PWM Modules	5 Modules: PWM0, PWM1, PWM2, PWM3 and PWM4		3 Modules: PWM0, PWM1, PWM2	5 Modules: PWM0, PWM1, PWM2, PWM3 and PWM4		2 Modules: PWM0, PWM1	5 Modules: PWM0, PWM1, PWM2, PWM3 and PWM4	
Audio Module with auto-decay	✔							
SW select. Clock outputs	2							
Polling / Flash Timer Output	1 High-Current Port output operable in Power Saving Mode			-				

Table 1–1: CDC16xxF Family Feature List, continued

Item	This Document: CDC1605F-E EMU	CDC1607F-E MCM Flash	CDC1631F-E MASK ROM	CDC1605F-C EMU	CDC1607F-C MCM Flash	CDC1641F-C Mask ROM	CDC1652F-C Mask ROM	CDC1672F-C Mask ROM
Timers & Counters								
16-bit free running counters with Capture/ Compare modules	CCC0 with 3CAPCOM							
16-bit timers	1: T0							
8-bit timers	2: T1 and T2							
Real Time Clock, Delivering Hours, Minutes and Seconds	✔			-				
Miscellaneous								
Scalable layout in CAN, RAM and ROM	-	✔		-	✔			
Various randomly selectable hardware options	Most options software-programmable, copy from user program storage during system start-up		Mask programmed according to user specification	Most options software-programmable, copy from user program storage during system start-up				
Core Bond-Out	✔	-		✔	-			
Supply Voltage	4.5 V to 5.5 V							
Temperature Range	T_{case}: –40 °C to +105 °C			T_{amb}: –40 °C to +85 °C				
Package								
Type	Ceramic 177PGA	Plastic 100QFP 0.65 mm pitch		Ceramic 177PGA	Plastic 100QFP 0.65 mm pitch			
Bonded Pins	176	100		176	100			

1.2. Abbreviations

AM	Audio Module
CAN	Controller Area Network Module
CAPCOM	Capture/Compare Module
CPU	Central Processing Unit
DMA	Direct Memory Access Module
ERM	EMI Reduction Module
IR	Interrupt Controller
LCD	Liquid Crystal Display Module
P06COMP	P0.6 Alarm Comparator
PINT	Port Interrupt Module
PSM	Power Saving Module
PWM	8-Bit Pulse Width Modulator Module
RTC	Real time Clock
SM	Stepper Motor Control Module
SPI	Serial Synchronous Peripheral Interface
T0	16-Bit Timer 0
T1, T2	8-Bit Timers 1 and 2
UART	Universal Asynchronous Receiver Transmitter

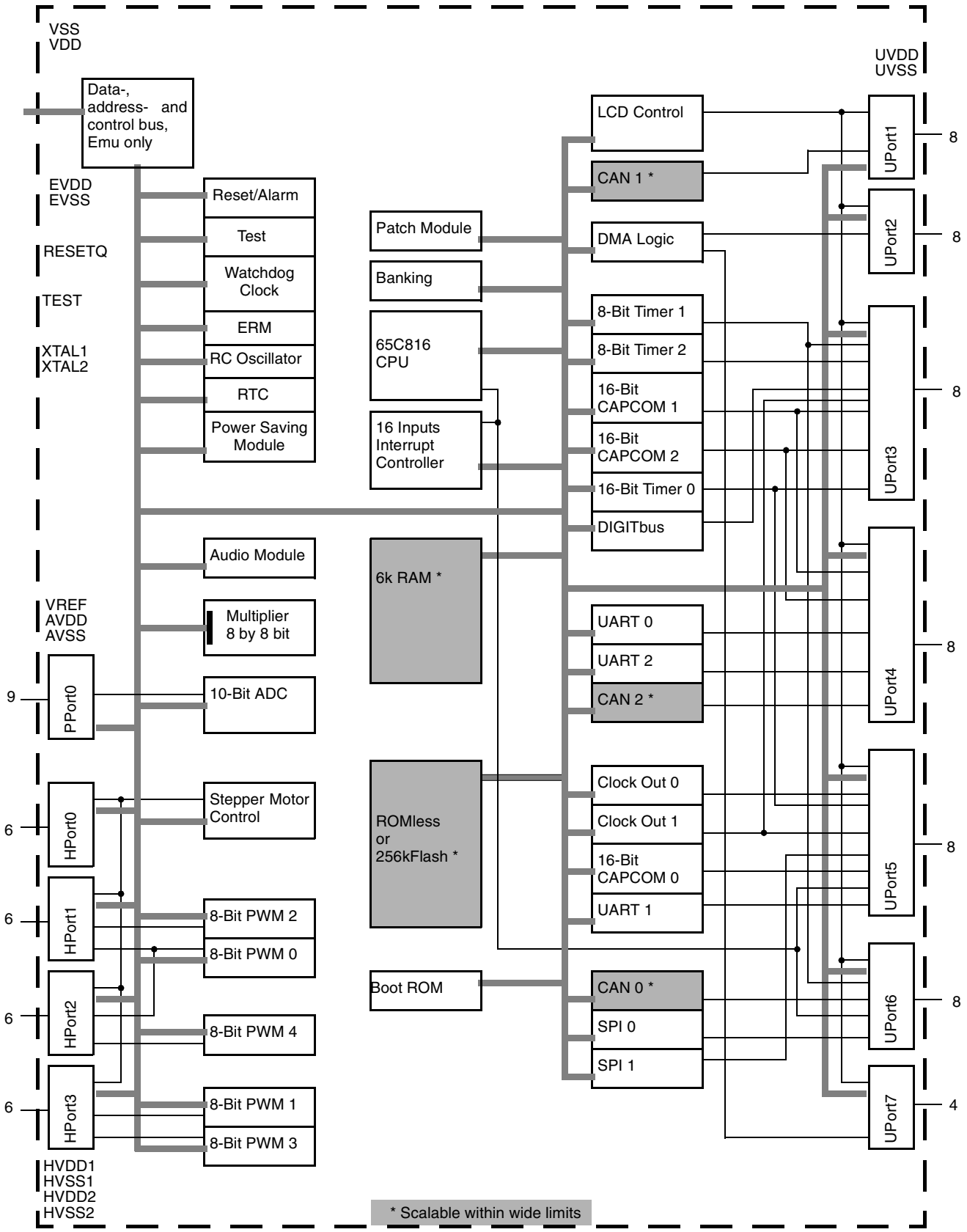


Fig. 1–1: Block diagram of CDC1605F-E/CDC1607F-E

2. Package and Pins

2.1. Pin Assignment

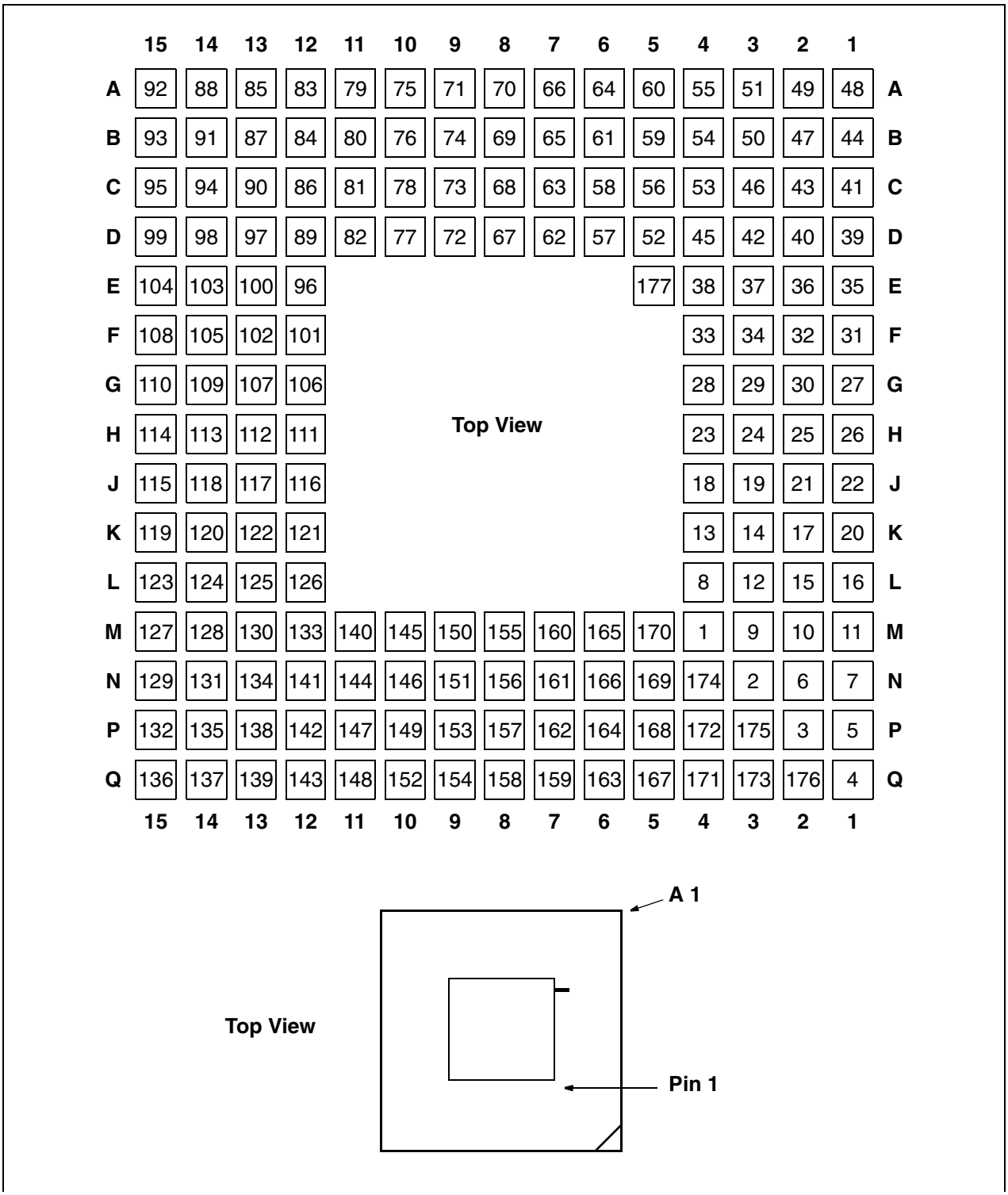
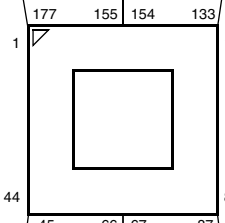


Fig. 2–1: Pin Map of CPGA177 Package

Pin Functions					Co-ord.	Pin No.
Bus Mode	LCD Mode	Port Special Out	Port Special In	Basic Function		
				EADB21	M8	155
				EADB20	N8	156
				NC	P8	157
				NC	Q8	158
				EPH2	Q7	159
				EBE	M7	160
				EVPA	N7	161
				EVDA	P7	162
				ESTOPCLK	Q6	163
				UVSS	P6	164
				UVDD	M6	165
ADB7	SEG3.7	T2-OUT		U3.7	N6	166
ADB6	SEG3.6	CC1-OUT		U3.6	Q5	167
ADB5	SEG3.5	SPI1-CLK-OUT	SPI1-CLK-IN	U3.5	P5	168
ADB4	SEG3.4	T0-OUT	**WP0**	U3.4	N5	169
ADB3	SEG3.3	CC2-OUT		U3.3	M5	170
ADB2	SEG3.2	DIGIT-OUT	DIGIT-IN	U3.2	Q4	171
ADB1	SEG3.1	CO1	SPI1-D-IN	U3.1	P4	172
ADB0	SEG3.0	SPI1-D-OUT		U3.0	Q3	173
				NC	N4	174
				NC	P3	175
				NC	Q2	176
Extra pin for avoiding wrong chip insertion. Connect to System Ground.					E5	177
				NC	M4	1
				NC	N3	2
				NC	P2	3
				NC	Q1	4
	SEG6.7	CAN0-TX	MULTI_TEST_IN	U6.7	P1	5
	SEG6.6	PINT1-OUT	CAN0-RX/**WP1**	U6.6	N2	6
	SEG6.5	T1-OUT	SPI0-D-IN	U6.5	N1	7
			Connect to System Ground		L4	8
	SEG6.4	SPI0-D-OUT		U6.4	M3	9
				TEST	M2	10
				RESETQ	M1	11
				XTAL2	L3	12
				XTAL1	K4	13
				VSS	K3	14
				VDD	L2	15
	SEG6.3	SPI0-CLK-OUT	SPI0-CLK-IN	U6.3	L1	16
	SEG6.2	T1-OUT	PINT2-IN/**WP5**	U6.2	K2	17
	SEG6.1	LCD-CLK-OUT	PINT1-IN/**WP4**	U6.1	J4	18
	SEG6.0	LCD-SYNC-OUT	PINT0-IN/**WP3**	U6.0	J3	19
WEQ	SEG1.7	CAN1-TX		U1.7	K1	20
CEQ	SEG1.6		CAN1-RX/**WP2**	U1.6	J2	21
ITSTOUT	SEG1.5	LCD-CLK-OUT		U1.5	J1	22
RWQ	SEG1.4	LCD-SYNC-OUT		U1.4	H4	23
PH2	BP3			U1.3	H3	24
OEQ	BP2			U1.2	H2	25
BE	BP1			U1.1	H1	26
RDY	BP0	ITSTOUT		U1.0	G1	27
				EADB17	G4	28
				EADB0	G3	29
				EDB7	G2	30
				EDB6	F1	31
				EDB5	F2	32
				EDB4	F4	33
				EDB3	F3	34
				EDB2	E1	35
				EDB1	E2	36
				EDB0	E3	37
			Connect to System Ground		E4	38
				EVSS1	D1	39
				NC	D2	40
				NC	C1	41
				NC	D3	42
				NC	C2	43
				NC	B1	44
				EVDD1	D4	45
				EOEQ	C3	46
				ECEQ	B2	47
				EADB1	A1	48
STOPCLK		SMB1+		H1.5	A2	49
VPQ		SMB1-		H1.4	B3	50
VPA		SMB2+		H1.3	A3	51
			Connect to System Ground		D5	52
VDA		SMB2-	SMB-COMP	H1.2	C4	53
DB7		SME1+/PWM2		H1.1	B4	54
DB6		SME1-/PWM0		H1.0	A4	55
				HVDD1	C5	56
				HVSS1	D6	57
DB5		SME2+		H0.5	C6	58
DB4		SME2-	SME-COMP	H0.4	B5	59
DB3		SMA1+		H0.3	A5	60
DB2		SMA1-		H0.2	B6	61
DB1		SMA2+		H0.1	D7	62
DB0		SMA2-	SMA-COMP	H0.0	C7	63
		SMD1+		H3.5	A6	64
		SMD1-		H3.4	B7	65
		SMD2+		H3.3	A7	66



NC = not connected, leave vacant

Pin No.	Co-ord.	Pin Functions				
		Basic Function	Port Special In	Port Special Out	LCD Mode	Bus Mode
154	Q9	EADB22				
153	P9	EADB23				
152	Q10	U7.0			SEG7.0	
151	N9	U7.1			SEG7.1	
150	M9	U7.2		GRDQ	SEG7.2	
149	P10	U7.3		GWRQ	SEG7.3	
148	Q11	U4.0	CAN2-RX/**WP7**		SEG4.0	ADB8
147	P11	U4.1		CAN2-TX	SEG4.1	ADB9
146	N10	U4.2	UART2-RX		SEG4.2	ADB10
145	M10	U4.3		UART2-TX	SEG4.3	ADB11
144	N11	U4.4	UART0-RX/**WP8**		SEG4.4	ADB12
143	Q12	U4.5		UART0-TX	SEG4.5	ADB13
142	P12	U4.6	CC2-IN	CC1-OUT	SEG4.6	ADB14
141	N12	U4.7	CC1-IN		SEG4.7	ADB15
140	M11	Connect to System Ground				
139	Q13	U5.0	CC0-IN	CO1	SEG5.0	
138	P13	U5.1	INT-TEST-IN	CC0-OUT	SEG5.1	
137	Q14	U5.2	LCD-CLK-IN	AM-PWM	SEG5.2	
136	Q15	U5.3	LCD-SYNC-IN	AM-OUT	SEG5.3	
135	P14	U5.4	IRQ	UART1-TX	SEG5.4	
134	N13	U5.5	ABORTQ	CO0	SEG5.5	
133	M12	NC				
132	P15	NC				
131	N14	NC				
130	M13	NC				
129	N15	NC				
128	M14	U5.6	PINT3/**WP6**	PWM2	SEG5.6	
127	M15	U5.7	PINT3/UART1-RX	PINT0-OUT	SEG5.7	
126	L12	U2.0/GD0			SEG2.0	ADB16
125	L13	U2.1/GD1			SEG2.1	ADB17
124	L14	U2.2/GD2			SEG2.2	ADB18
123	L15	U2.3/GD3			SEG2.3	ADB19
122	K13	U2.4/GD4			SEG2.4	ADB20
121	K12	U2.5/GD5			SEG2.5	ADB21
120	K14	U2.6/GD6			SEG2.6	ADB22
119	K15	U2.7/GD7			SEG2.7	ADB23
118	J14	AVSS				
117	J13	AVDD				
116	J12	VREF				
115	J15	NC				
114	H15	P0.1	P0.1 digital input			
113	H14	P0.2	P0.2 digital input			
112	H13	P0.3	P0.3 digital input			
111	H12	P0.4	P0.4 digital input			
110	G15	P0.5	P0.5 digital input			
109	G14	P0.6	P0.6 Comparator input			
108	F15	P0.7				
107	G13	P0.8				
106	G12	P0.9				
105	F14	EADB16				
104	E15	EADB15				
103	E14	EADB14				
102	F13	EADB13				
101	F12	EADB12				
100	E13	EADB11				
99	D15	EADB10				
98	D14	EADB9				
97	D13	EWEQ				
96	E12	Connect to System Ground				
95	C15	EADB18				
94	C14	EVSS2				
93	B15	NC				
92	A15	NC				
91	B14	NC				
90	C13	NC				
89	D12	NC				
88	A14	NC				
87	B13	NC				
86	C12	EVDD2				
85	A13	EADB19				
84	B12	EADB8				
83	A12	EADB7				
82	D11	EADB6				
81	C11	EADB5				
80	B11	EADB4				
79	A11	EADB3				
78	C10	EADB2				
77	D10	H2.0	SMC-COMP	SMC2-		
76	B10	H2.1		SMC2+		
75	A10	H2.2		SMC1-		
74	B9	H2.3		SMC1+		
73	C9	H2.4	**WP9**	PWM0		
72	D9	H2.5/POL		PWM4		
71	A9	HVSS2				
70	A8	HVDD2				
69	B8	H3.0		PWM1		
68	C8	H3.1		PWM3		
67	D8	H3.2	SMD-COMP	SMD2-		

Fig. 2–2: Pin Assignment for **CPGA177** Package

2.2. Package Outline Dimensions

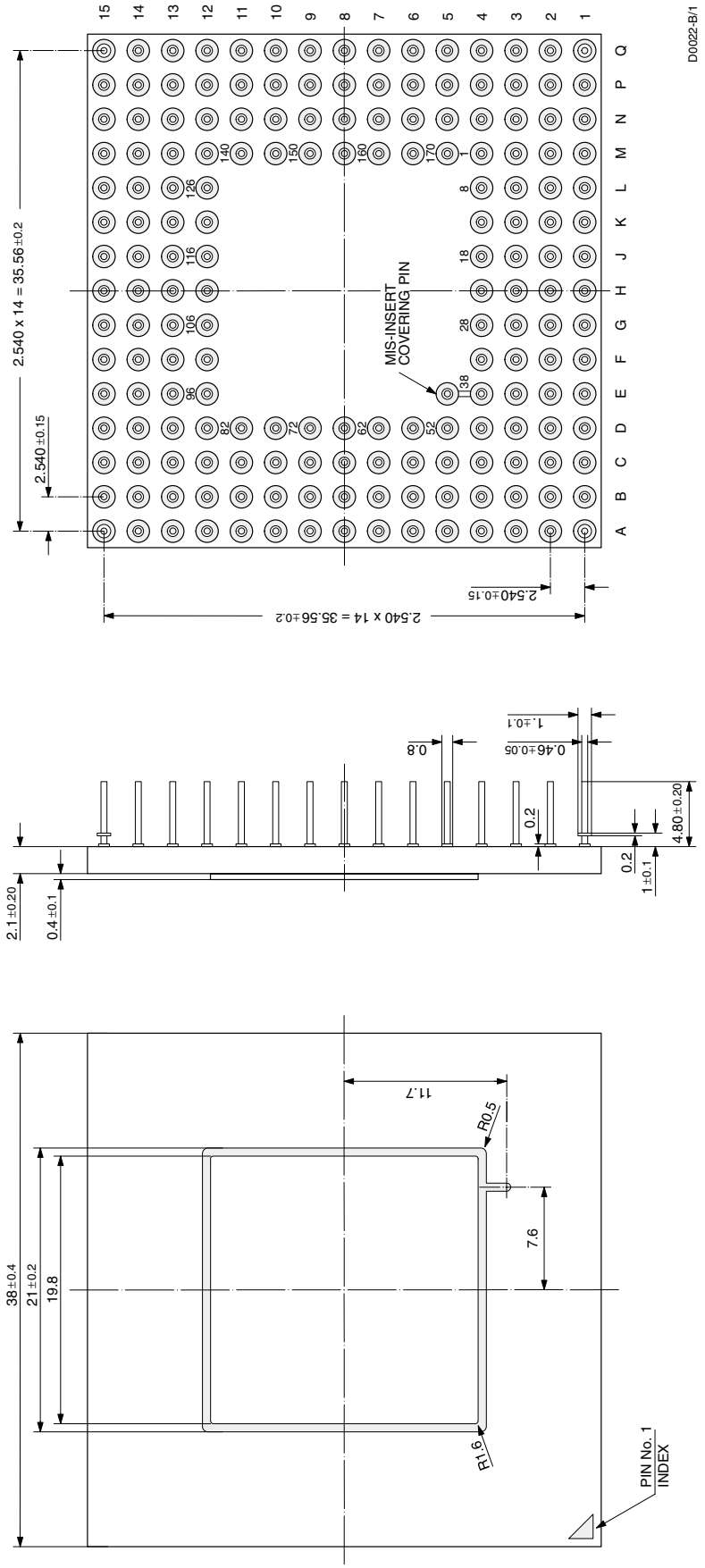


Fig. 2–3: CPGA177 Ceramic Pin Grid Array 177-Pin (Weight approx. 14g)

2.3. Multiple Function Pins

2.3.1. U-Ports

Apart from their basic function (digital I/O), Universal Ports (prefix “U”) have overlaid alternative functions (see Fig. 2–2 on page 12).

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.

Bus Mode is used for testing purposes only. It is controlled by the Control Register (CR) setting. Refer to section “Control Word” for more information.

2.3.2. H-Ports

Apart from their basic function (digital I/O), High Current Ports (prefix “H”) have overlaid alternative functions (see Fig. 2–2 on page 12).

How to enable Basic Function, Special In and Special Out mode is explained in the functional description of the H-Ports.

The Bus Mode is used for testing purposes only. It is controlled by the Control Register (CR) setting. Refer to section “Control Word” for more information.

2.3.3. Emulator Bus

In the CPGA177 package, the Emulator Bus (prefix “E”) serves as connection to an external emulation hardware.

In the PQFP100 MCM Package it is internally connected to the Flash program storage and is not available to the outside.

Its function is controlled by register CR. Refer to section “Control Word” for more information.

2.4. Pin Function Description

ABORTQ
The Abort input serves as the CPU’s ABORTQ input provided that flag EXTIR in register SR2 is set. Active LOW.

ADB0 to ADB23
These 24 lines form the address bus for memory and I/O exchange. The function is controlled by register CR.

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

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

AVDD
This is the positive power supply for ADC, P06COMP and ERM. AVDD should be kept at VDD ±0.5 V.

AVSS
This is the negative reference for the ADC and the negative power supply for ADC, P06COMP and ERM. Connect to system ground.

BE
The Bus Enable output signal shows the state of the internal CPU.
1: Internal CPU has bus access.
0: Internal CPU has no bus access.

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

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

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

CC0-IN, CC1-IN, CC2-IN
These signals are the capture inputs of the CAPCOM0, CAPCOM1 and CAPCOM2 modules.

CC0-OUT, CC1-OUT, CC2-OUT
These signals are the compare outputs of the CAPCOM0, CAPCOM1 and CAPCOM2 modules.

CEQ
This output signal connects to external memory’s CEQ pin and reduces its power consumption when CPU operates in SLOW mode. Active LOW.

CO0, CO1
The signals Clock Out 0 and 1 provide frequency outputs. Both are connected to internal prescaler and multiplexer. They can be hard-wired by HW Option. Refer to section “Hardware Options” for setting the CO0 bytes 00FFABh, 00FFB2h, 00FFB3h, 00FFB6h and the CO1 byte 00FFACh.

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.

DB0 to DB7
The eight bidirectional Data Bus lines provide the 8-bit data bus for use during data exchanges between the microprocessor and external memory or peripherals. The function is controlled by register CR.

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

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

EADB0 to EADB23
These 24 lines form the address bus for external memory access on the Emulator Bus. The function is controlled by register CR.

EBE
The Emulator Bus Enable output signal shows the state of

the internal CPU.
1: Internal CPU has bus access.
0: Internal CPU has bus no access.

ECEQ
Emulator Bus Chip Enable output signal connects to external memory's CEQ pin and reduces its power consumption when CPU operates in SLOW mode. Active LOW.

EDB0 to EDB7
These eight bidirectional Emulator Data Bus lines provide the 8-bit data bus for use during data exchanges between the microprocessor and external memory or peripherals.

EOEQ
The Emulator Bus Output Enable signal connects to the OEQ pin of external memory for read access. Active LOW.

EPH2
This output is the system clock of the Emulator Bus. It provides the timing for external memory access.

ESTOPCLK
If the Emulator Stop Clock input signal is HIGH, all the peripheral modules are halted (f_{OSC} is stopped), whereas the clock PH2 and the CPU remain active. Thus it is possible to read the registers and memory for debugging purposes.

For normal operation connect ESTOPCLK to System Ground or leave it floating (internal pull-down).

EVDA
The Emulator Bus Valid Data Address output signal shows the state of the internal CPU. It must be considered together with the signal EVPA (Emulator Bus Valid Program Address). Each signal indicates a valid memory address when high.

Table 2–1: Valid Memory Address

EVDA, VDA	EVPA, VPA	State
0	0	Internal Operation. Address and data bus available. Address bus may be invalid.
0	1	Valid program address. May be used for program cache control.
1	0	Valid data address. May be used for data cache control.
1	1	Opcode fetch. May be used for program cache control and single step control.

EVDD1, EVDD2
These two lines form the positive power supply of the Emulator Bus drivers.

EVPA
Refer to EVDA.

EVSS1, EVSS2
These two lines form the negative supply of the Emulator Bus drivers. Connect to system ground.

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

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

GRDQ
This Graphics IC Read line provides the control signal for read accesses via the GD7 to GD0 bus. Active LOW.

GWRQ
This Graphics IC Write line provides the control signal for write accesses via the GD7 to GD0 bus. Active LOW.

H0.0 to H3.5
The High Current Ports are intended for use as digital I/O which can drive higher currents than the Universal Ports. Each of the high current ports H0 to H3 is six bits wide.

HVDD1, HVDD2
The pins HVDD1 and HVDD2 are the positive power supply of the high current ports H0.0 to H3.5. HVDD1 feeds ports H0 and H1. HVDD2 feeds ports H2 and H3. HVDD1 and HVDD2 should be kept at VDD ±0.5 V. Be careful to design the PCB traces for carrying the considerable operating current on these pins.

HVSS1, HVSS2
The pins HVSS1 and HVSS2 are the negative power supply for the high-current ports H0.0 to H3.5. HVSS1 feeds ports H0 and H1. HVSS2 feeds ports H2 and H3. Both have to be hard-wired to system ground. Be careful to layout sufficient PCB traces for carrying the considerable operating current on these pins.

INT-TEST-IN
Test input signal for Interrupt Controller.

IRQ
IRQ serves as the CPU's IRQ input, provided that flag EXTIR in register SR2 is set. Active LOW.

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.

MULTI-TEST-IN
This is a test input line. It is intended for factory test only. The application should not use this signal.

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

PH2
The System Clock output signal provides timing for external

read or write operations. Addresses are valid (after the Address Setup Time) following the negative transition of PH2. The function is controlled by register CR.

PINT0-IN, PINT1-IN, PINT2-IN
The Port Interrupt 0, 1 and 2 inputs serve as inputs to the PINT.

PINT3
The Port Interrupt 3 input serves as input to the PINT. HW option FFC2h has to be set to determine whether U5.6 or U5.7 is the source of PINT3-IN.

PINT0-OUT, PINT1-OUT
The Port Interrupt 0 and 1 outputs carry the output signals of the PINT.

POL
Output of the Polling Module.

PWM0 to PWM4
These are the outputs of the PWM. Some of these PWM signals are directed to two pins.

P0.1 to P0.9
These 9 analog ports are the multiplexed input channels of the ADC. Analog port P0.6 is additionally input to the P06COMP.

The analog ports P0.1 to P0.5 can also be used as five digital input lines. The digital function of the analog ports should be disabled (P0DIN in register SR1 set to 0) when operating these ports as analog inputs to avoid leakage currents.

RDY
This output signal shows the state of the internal CPU.
1: CPU active
0: CPU halted by IR or DMA

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
This input/output signal controls data exchange in cooperation with DB and ADB. It can be driven from the external CPU if the internal CPU is switched off.
1: CPU reads data.
0: CPU writes data.

SEG1.4 to SEG7.3
These pin functions serve as Segment drivers for a 4:1 multiplexed LCD.

SMA to SME
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 SME can drive a two-phase bipolar stepper motor.

SMA-COMP to SME-COMP
These lines are comparator inputs that connect to one line each of the SMA to SME 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.

STOPCLK
If the input signal Stop Clock is HIGH, all the peripheral modules are halted (f_{OSC} is stopped). But the clock PH2 and the CPU remain active. Thus it is possible to read the registers and memory for debugging purposes. TEST must be held high to enable STOPCLK.

TEST
The main function of the Test pin is to define the source for the Control Word fetch during reset. If TEST is held low during reset, the Control Word is fetched via the Emulator Bus (from internal Flash program storage in MCM package). If TEST is held high during reset, the Control Word is fetched via the Test Bus.

For normal operation with internal code connect TEST to System Ground or leave it floating (internal pull-down).

TEST must be held high in active mode to enable STOPCLK.

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, T2-OUT
These signals are connected to the zero outputs of T1 and T2. T1-OUT is directed to several pins.

UART0-RX, UART1-RX, UART2-RX
These are the Receive input lines of UART0, UART1 and UART2. Polarity of the signals is settable by HW options FFB4h, respectively FFB5h.

UART0-TX, UART1-TX, UART2-TX
These are the data output lines of UART0, UART1 and UART2. Polarity of signals is settable by HW options FFB4h, respectively FFB5h.

UVDD
The pin UVDD is the positive supply voltage for the U-Port output stages (see Fig. 2–4 for external connection).

UVSS
The pin UVSS is the negative power supply for the U-Port output stages. It has to be connected to system ground (see Fig. 2–4).

U1.0 to U7.3
Universal ports are intended for use as digital I/O or as LCD driver outputs. The ports U1 to U6 consist of eight bit lines each. Port U7 is 4 bits wide.

VDA
The signal Valid Data Address must be considered together with the signal VPA (Valid Program Address). They show the state of the internal CPU. If the internal CPU is switched off, both signals may be driven from the outside. Together they indicate a valid memory address when HIGH (see Table 2–1 on page 15).

VDD
The pin VDD is the positive supply voltage for the internal digital modules (see Fig. 2–4 for external connection).

VPA
Refer to VDA.

VPQ
The Vector Pull output indicates that the CPU addresses a vector location during an interrupt sequence. VPQ is low during the last two interrupt sequence cycles, during which time the CPU reads the interrupt vector.

VREF
This pin is the positive reference input for the ADC. The voltage at this pin must be set to a level between 2.56 Volts and AVDD.

VSS
The pin VSS is the negative supply terminal of the internal digital modules (see Fig. 2–4 for external connection).

WEQ
The Write Enable output signals a write access to external memory. 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. 2–4 for external connection).

XTAL2
This is the quartz oscillator output pin for two pin oscillator circuits (see Fig. 2–4 for external connection).

2.5. External Components

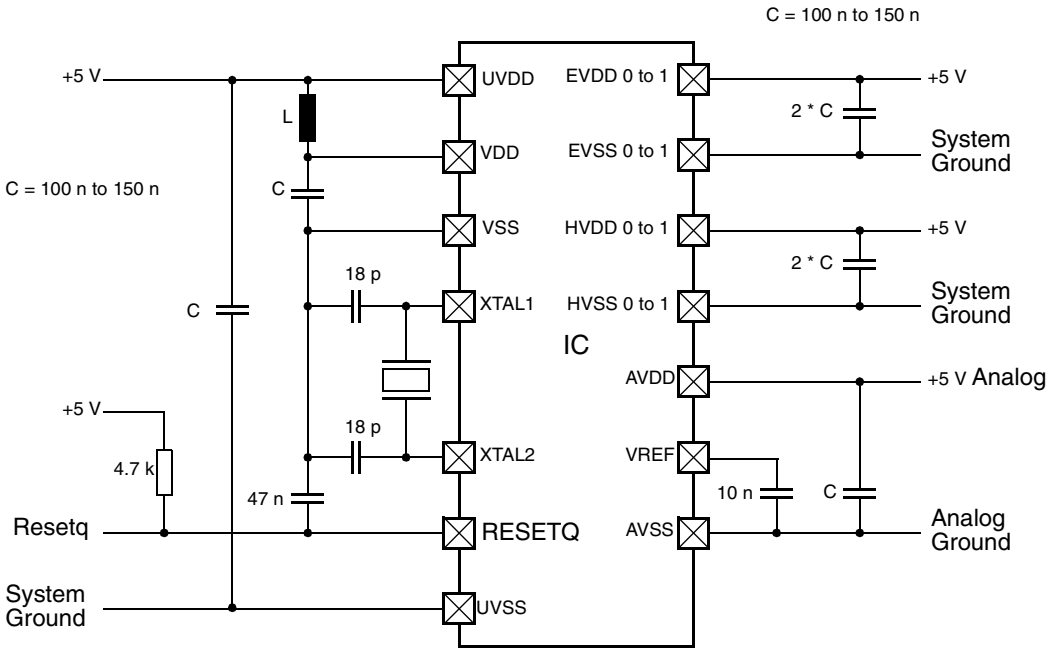


Fig. 2–4: Recommended external supply and quartz connection for low electromagnetic interference (EMI)

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. A frequency too low will

reduce decoupling effectiveness, increase RF emissions and may affect device operation adversely.

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 47nF 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

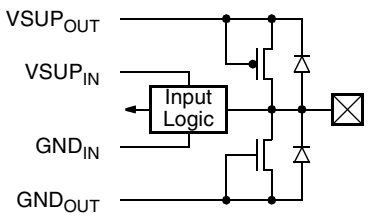


Fig. 2–5: Input Pins

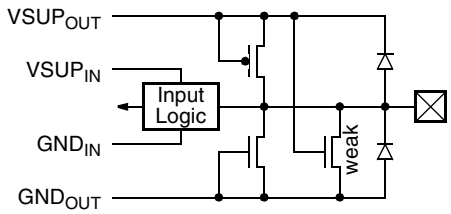


Fig. 2–6: Input Pins with Pull-Down

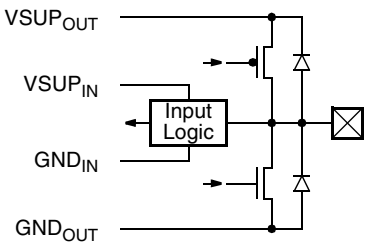


Fig. 2–7: Push Pull I/O Pins

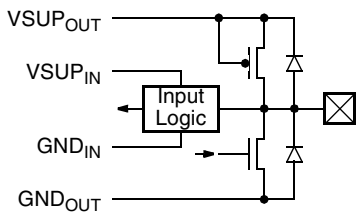


Fig. 2–8: Open Drain I/O

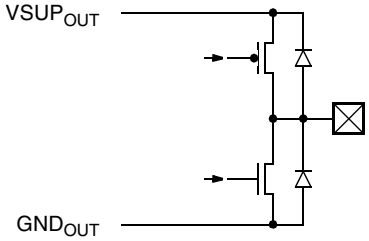


Fig. 2–9: Push Pull Output Pins

Table 2–2: I/O Supply Catalog

Pin Names	Figure	$VSUP_{OUT}$	GND_{OUT}	$VSUP_{IN}$	GND_{IN}
XTAL1, XTAL2	2–5	UVDD	UVSS	VDD	VSS
ESTOPCLK	2–6				
TEST					
U-Ports	2–7				
H-Ports		HVDD	HVSS		
P-Ports		AVDD	UVSS		
EDB0 to EDB7		EVDD	EVSS		
RESETQ	2–8	UVDD	UVSS		
EADB0 to EADB19, ECEQ, EOEQ, EWEQ	2–9	EVDD	EVSS		
EADB20 to EADB23, EPH2, EVDA, EVPA, EBE		UVDD	UVSS		

3. Electrical Data

3.1. Absolute Maximum Ratings

Table 3–1: $UV_{SS}=HV_{SS1}=HV_{SS2}=EV_{SS1}=EV_{SS2}=AV_{SS}=0V$

Symbol	Parameter	Pin Name	Min.	Max.	Unit
V_{SUP}	Core Supply Voltage Port Supply Voltage Analog Supply Voltage SM Supply Voltage 1 SM Supply Voltage 2 Flash Port Supply Voltage 1 Flash Port Supply Voltage 2	VDD UVDD AVDD HVDD1 HVDD2 EVDD1 EVDD2	–0.3	6.0	V
ΔV_{DD}	Voltage Difference between V_{DD} and AV_{DD} resp. UV_{DD}	VDD, AVDD UVDD	–0.5	0.5	V
I_{SUP}	Core Supply Current Port Supply Current Flash Port Supply Current	VDD, VSS UVDD, UVSS EVDD1, EVSS1 EVDD2, EVSS2	–40	40	mA
IA_{SUP}	Analog Supply Current	AVDD, AVSS	–20	20	mA
IH_{SUP}	SM Supply Current @T_{CASE}=105C, Duty Factor=0.71 [1])	HVDD1, HVSS1 HVDD2, HVSS2	–380	380	mA
V_{in}	Input Voltage	U-Ports, XTAL,RESETQ, TEST	UV_{SS}–0.5	UV_{DD}+0.7	V
		P0-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	–5	5	mA
		H-Ports	–60	60	mA
t_{oshsl}	Duration of Short Circuit in Port SLOW Mode to UVSS or UVDD	U-Ports except U3.2 in DP Mode		indefinite	s
T_j	Junction Temperature under Bias		–45	115	°C
T_s	Storage Temperature		–45	125	°C
P_{max}	Maximum Power Dissipation			0.8	W

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

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 or any other conditions beyond those indicated in the “Recommended Operating Conditions/Characteristics” of this specification is not implied. Exposure to absolute maximum ratings conditions for extended periods may affect device reliability.

3.2. Recommended Operating Conditions

Table 3–2: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0V$

Symbol	Parameter	Pin Name	Min.	Typ [1)]	Max.	Unit
V_{DD}	Supply Voltage Port Supply Voltage Analog Supply Voltage Flash Port Supply Voltage 1 Flash Port Supply Voltage 2	VDD UVDD AVDD EVDD1 EVDD2	4.5	5	5.5	V
HV_{DD}	SM Supply Voltage 1 SM Supply Voltage 2	HVDD1 HVDD2	4.75	5	5.25	V
ΔV_{DD}	Voltage Difference between VDD and AVDD resp. UVDD	VDD, AVDD UVDD	-0.2		0.2	V
dAV_{DD}	AVDD Ripple, Peak-to-Peak	AVDD			200	mV
f_{XTAL}	XTAL Clock Frequency	XTAL1	4		12	MHz
	XTAL Clock Frequency using ERM	XTAL1	4		10	MHz
T_j	Junction Temperature		-40		110	C
V_{il}	Low Input Voltage	U-Ports H-Ports P0-Ports TEST			$0.51*V_{DD}$	V
		E-Ports			0.8	V
V_{ih}	High Input Voltage	U-Ports H-Ports P0-Ports TEST	$0.86*V_{DD}$			V
		E-Ports	2.2			V
RV_{il}	Reset Active Input Voltage	RESETQ			0.9	V
WRV_{il}	Reset Active Input Voltage during Power Saving Modes and Wake Reset	RESETQ			0.6	V
RV_{im}	Reset Inactive and Alarm Active Input Voltage	RESETQ	1.6		2.1	V
RV_{ih}	Reset Inactive and Alarm Inactive Input Voltage	RESETQ	2.9			V
WRV_{ih}	Reset Inactive during Power Saving Modes	RESETQ	UV_{DD} - 0.4V			V
V_{REFi}	ADC Reference Input Voltage	VREF	2.56		AV_{DD}	V
$P0V_i$	P0 ADC Input Port Input Voltage	P0-Ports	0		V_{REFi}	V
Clock Input from External Generator						
XV_{il}	Clock Input Low Voltage	XTAL1			$0.2*V_{DD}$	V
XV_{ih}	Clock Input High Voltage	XTAL1	$0.8*V_{DD}$			V
D_{XTAL}	Clock Input High-to-Low Ratio	XTAL1	0.45		0.55	

3.3. Characteristics

Table 3–3: UV_{SS} = HV_{SS1}= HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0 V, 4.5 V< V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1}= EV_{DD2} < 5.5 V, 4.75 V < HV_{DD1} = HV_{DD2} < 5.25 V, T_{CASE} = −40 °C to +105 °C, f_{XTAL}= 10 MHz

Symbol	Parameter	Pin Na.	Min.	Typ [1]	Max.	Unit	Test Conditions
Package							
R_{thjc}	Thermal Resistance from Junction to Case			15		C/W	
Supply Currents							CMOS levels on all Inputs, no Loads on Outputs difference between any two VDDs within ±0.2 V
I_{DDF}	VDD FAST Mode Supply Current	VDD		15	25	mA	
I_{DDS}	VDD SLOW Mode Supply Current	VDD			1.7	mA	all Modules OFF [2], [6]
I_{DDD}	VDD DEEP SLOW Mode Supply Current	VDD			1.4		all Modules OFF [2], [6]
I_{DDI}	VDD IDLE Mode Supply Current	VDD		50	75	µA	f_{xtal} = 4 MHz [6]
				60	90	µA	f_{xtal} = 10 MHz [6]
				70	100	µA	internal RC oscill.
I_{DDW}	VDD WAKE Mode Supply Current	VDD		30	50	µA	
UI_{DDa}	UVDD Active Supply Current	UVDD			0.3	mA	no Output Activity, LCD Module ON
AI_{DDa}	AVDD Active Supply Current	AVDD		0.2	0.4	mA	ADC ON, ERM OFF
				1	2	mA	ERM ON, f_{XTAL}= 8.4 MHz
AI_{DDq}	Quiescent Supply Current	AVDD		1	10	µA	ADC and ERM OFF
UI_{DDq}		UVDD		1	10	µA	no Output Activity, LCD Module OFF
EI_{DDq}		EVDD1 EVDD2		1	10	µA	no Output Activity, LCD Module OFF
HI_{DDq}		Sum of all HVDD1 HVDD2		1	20	µA	no Output Activity, SM Module OFF
Inputs							

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0$ V, 4.5 V< $V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1} = EV_{DD2}$ < 5.5 V, 4.75 V < $HV_{DD1} = HV_{DD2}$ < 5.25 V, T_{CASE} = –40 °C to +105 °C, f_{XTAL} = 10 MHz

Symbol	Parameter	Pin Na.	Min.	Typ [1]	Max.	Unit	Test Conditions
V_{ilh}	Low to High Input Threshold Voltage	all Inp. except EPorts, XTAL	0.68* V_{DD}	0.76* V_{DD}	0.84* V_{DD}	V	
V_{ihl}	High to Low Input Threshold Voltage	all Inp. except EPorts, XTAL	0.53* V_{DD}	0.61* V_{DD}	0.69* V_{DD}	V	
V_{ilh}-V_{ihl}	Input Hysteresis	all Inp. except EPorts, XTAL	0.1* V_{DD}	0.15* V_{DD}	0.2* V_{DD}	V	[3]
I_i	Input Leakage Current	U-Ports H-Ports P0-Port P06 VREF E-Ports	–1 –10 –1 –0.2 –1 –1		1 10 1 0.2 1 1	µA	0 < Vi < UV_{DD} 0 < Vi < HV_{DD} 0 < Vi < AV_{DD} 0 < Vi < AV_{DD} 0 < Vi < AV_{DD} 0 < Vi < EV_{DD}
I_{pd}	Input Pull-Down Current	TEST E-Ports	25 25	80 80	170 170	µA	Vi = UV_{DD} Vi = EV_{DD}, when unused
I_{pu}	Input Pull-Up Current	E-DB	–170	–80	–25	µA	Vi = 0, when unused
Outputs							
V_{ol}	Port Low Output Voltage	U-Ports			0.4	V	Io = 2 mA
		E-Ports			0.4	V	Io = 0.5 mA
		H-Ports	0.125		0.5	V	Io = 27 mA Io = 40 mA@TCASE = -40 °C Io = 30mA@TCASE = 25 °C
ΔV_{ol}	Spread of Vol Values within one SM Driver Module	H-Ports	–50		50	mV	
V_{oh}	Port High Output Voltage	U-Ports	UV_{DD}– 0.4			V	Io= –2 mA
		E-Ports	EV_{DD}– 0.4			V	Io= –0.5 mA
		H-Ports	HV_{DD}– 0.5		HV_{DD}– 0.125	V	Io = –27 mA Io = –40 mA@TCASE = –40 °C Io = –30 mA @TCASE = 25 °C
ΔV_{oh}	Spread of Voh Values within one SM Driver Module	H-Ports	–50		50	mV	

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0$ V, 4.5 V< $V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1} = EV_{DD2}$ < 5.5 V, 4.75 V < $HV_{DD1} = HV_{DD2}$ < 5.25 V, T_{CASE} = –40 °C to +105 °C, f_{XTAL} = 10 MHz

Symbol	Parameter	Pin Na.	Min.	Typ [1]	Max.	Unit	Test Conditions
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*UVDD Pin Short to UVSS
ΣLI_{o2}	Internal LCD-High Supply Short Circuit Current	U-Ports	0.3		–0.3	mA	Pin short to UVDD Pin short to 1/3*UVDD
I_{shf}	Port Fast Short Circuit Current	U-Ports	TBD	9	TBD	mA	Pin Short to UVDD or UVSS, Port FAST Mode
I_{shs}	Port Slow Short Circuit Current	U-Ports	TBD	2.5	TBD	mA	Pin Short to UVDD or UVSS, Port SLOW Mode
I_{shsd}	Port Slow Short Circuit Current, DP Mode	U3.2	TBD	5	TBD	mA	Pin Short to UVDD, Port SLOW and Double Pull-Down Modes
Comparators							
V_{BG}	Internal Reference Voltage		1.125	1.25	1.375	V	
t_{BG}	Internal Voltage Reference Setup Time after Power-Up				10	µs	
V_{REFR}	RESET Comparator Reference Voltage	RESETQ	1*VBG		1*VBG	V	
RV_{lh}– RV_{hl}	RESET Comparator Hysteresis, symmetrical to VREFR	RESETQ	0.25		0.375	V	[3]
WRV_{ihl}	Reset Active high to low Voltage during Power Saving Modes and Wake Reset	RESETQ	0.6		1.1	V	
V_{REFA}	ALARM Comparator Reference Voltage	RESETQ	2*VBG		2*VBG	V	
AV_{lh}– AV_{hl}	ALARM Comparator Hysteresis, symmetrical to VREFA	RESETQ	0.1		0.15	V	[3]

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

Table 3–3: UV_{SS} = HV_{SS1}= HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0 V, 4.5 V< V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1}= EV_{DD2} < 5.5 V, 4.75 V < HV_{DD1} = HV_{DD2} < 5.25 V, T_{CASE} = –40 °C to +105 °C, f_{XTAL}= 10 MHz

Symbol	Parameter	Pin Na.	Min.	Typ [1]	Max.	Unit	Test Conditions
V_{REFPOR}	VDD Power On Reset Threshold	VDD	2.88* VBG		2.88* VBG	V	
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 VREFP06	P06	0.1		0.24	V	[3]
V_{REFSM}	SM Comparator Reference Voltage	H00, H04, H12, H20, H32	1/9* HVDD –0.07		1/9* HVDD +0.07	V	
t_{CDEL}	RESET, ALARM, P06, SM Comparator Delay Time	RESETQ P06 H00 H04 H12 H20 H32			100	ns	Overdrive = 50 mV
ADC							
LSB	LSB Value			VREF /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)		–3 –2.5		3 2.5	LSB	V_{REF} = 2.56 V V_{REF} = 5.12 V
ZE	Zero Error: difference between the output of an ideal and an actual ADC for zero input voltage		–1		1	LSB	V_{REF} = 2.56 V and 5.12 V
FSE	Full-Scale Error: difference between the output of an ideal and an actual ADC for full-scale input voltage		–1		1	LSB	V_{REF} = 2.56 V and 5.12 V
TUE	Total Unadjusted Error: maximum sum of integral non-linearity, zero error and full-scale error		–4 –3		4 3	LSB	V_{REF} = 2.56 V V_{REF} = 5.12 V
QE	Quantization Error: uncertainty because of ADC resolution		–0.5		0.5	LSB	V_{REF} = 2.56 V and 5.12 V

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0$ V, 4.5 V< $V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1} = EV_{DD2}$ < 5.5 V, 4.75 V < $HV_{DD1} = HV_{DD2}$ < 5.25 V, T_{CASE} = –40 °C to +105 °C, f_{XTAL} = 10 MHz

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		–4.5 –3.5		4.5 3.5	LSB	V_{REF} = 2.56 V V_{REF} = 5.12 V
R	Conversion Range	P0-Ports	AVSS		VREF	V	2.56 V < VREF < AVDD
A	Conversion Result			INT (Vin/ LSB)		hex	AVSS < Vin < VREF
			000			hex	Vin < = AVSS
					3FF	hex	Vin > = VREF
t_c	Conversion Time		4			µs	
t_s	Sample Time		2			µs	
Ci	Input Capacitance during Sampling Period			15		pF	
Ri	Serial Input Resistance during Sampling Period			5		kOhm	
SPI (Fig. 3–1, Fig. 3–2)							
t_{soci}	Data out Setup Time with internal clock	U3.0, U6.4			60 [4]	ns	@Cl = 30 pF, port fast mode
t_{hoci}	Data out Hold Time with internal clock	U3.0, U6.4	–60		60 [4]	ns	@Cl = 30 pF, port fast mode
t_{soce}	Data out Setup Time with external clock	U3.0, U6.4			$3/f_{XTAL}$ +60 [4]	ns	@Cl = 30 pF, port fast mode
t_{hoce}	Data out Hold Time with external clock	U3.0, U6.4	$2/f_{XTAL}$ –60			ns	@Cl = 30 pF, port fast mode
t_{si}	Data in Setup Time with external clock	U3.1, U6.5	$1/f_{XTAL}$+ 60 [4]			ns	
t_{hi}	Data in Hold Time with external clock	U3.1, U6.5	$1/f_{XTAL}$+ 60 [4]			ns	

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

Table 3–3: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0$ V, 4.5 V< $V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1} = EV_{DD2}$ < 5.5 V, 4.75 V < $HV_{DD1} = HV_{DD2}$ < 5.25 V, T_{CASE} = –40 °C to +105 °C, f_{XTAL} = 10 MHz

Symbol	Parameter	Pin Na.	Min.	Typ [1]	Max.	Unit	Test Conditions
CAN (Fig. 3–3)							
t_{srx}	rx-strobe Time	CAN rx	0		10 [4]	ns	reference is XTAL1 rising edge
t_{dtx}	tx-drive Time	CAN tx	15		60 [4]	ns	reference is XTAL1 falling edge @Cl = 30 pF, port fast mode
DIGITbus (Fig. 3–4)							
t_{btj}	Bit Time jitter	DIGIT-OUT			±10 [4]	ns	rising edges, internal clock master
t_{fed}	Falling edge delay	DIGIT-OUT	15 [5]		t_{BIT}/64 +60 [4]	ns	reference is nominal falling edge

[1] **Typical values describe typical behavior at room temperature (25 °C, unless otherwise noted), with typical Recommended Operating Conditions applied (derived from device characterization, 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 the ERM is active, this time value is increased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[5] When the ERM is active, this time value is decreased by 0.121/fXTAL, e.g. 15.125 ns @8 MHz.
[6] Measured with external clock. Add 170 µA @ 4 MHz, 200 µA @ 10 MHz for operation on typical quartz with SR3.XTAL = 0 (Oscillator RUN mode).

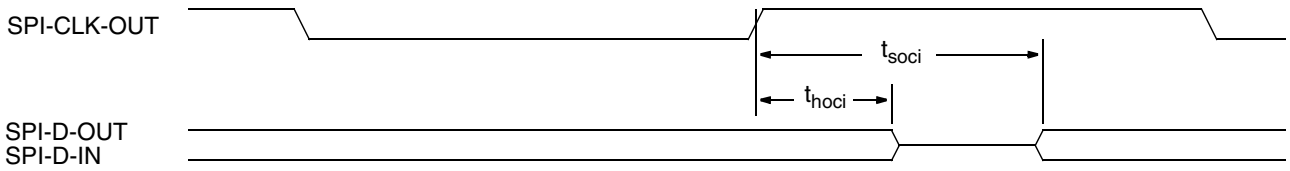


Fig. 3–1: SPI: Send and Receive Data with Internal Clock. Timing is valid for inverted clock too (Data valid at positive edge).

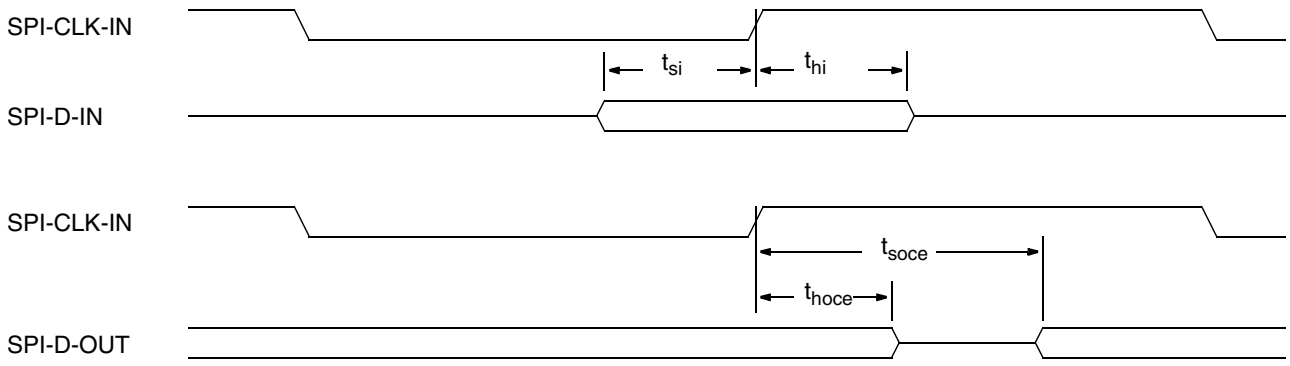


Fig. 3–2: SPI: Send and Receive Data with External Clock. Timing is valid for inverted clock too (Data valid at positive edge).

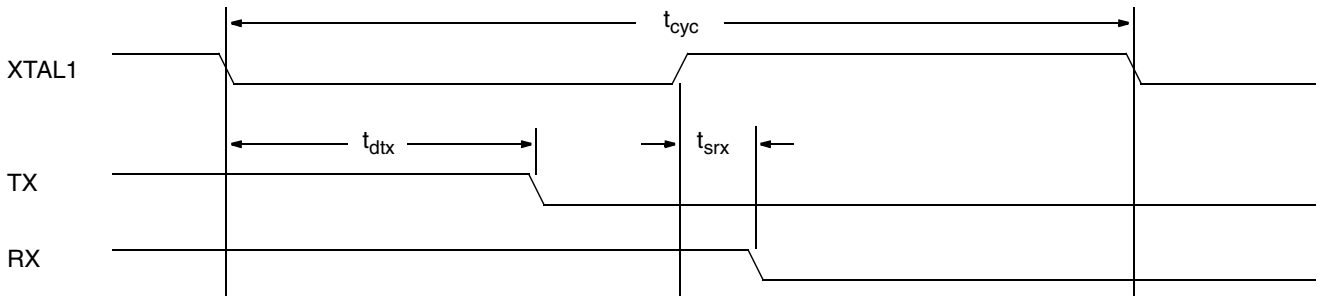
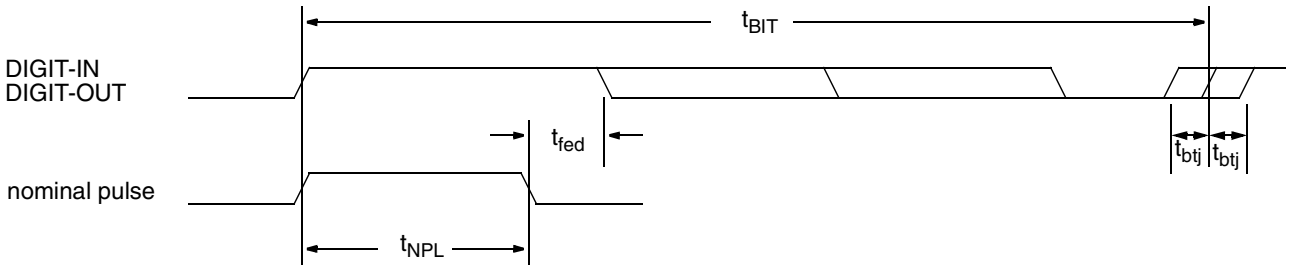


Fig. 3–3: CAN I/O Timing.



t_{NPL}: Nominal programmed Pulse Length. Depends on programmed phase, Baudrate 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. 3–4: DIGIT bus I/O Timing

3.4. Recommended Crystal Characteristics

Table 3–4: $UV_{SS} = HV_{SS1} = HV_{SS2} = EV_{SS1} = EV_{SS2} = AV_{SS} = 0$ V, 4.5 V < $V_{DD} = AV_{DD} = UV_{DD} = EV_{DD1} = EV_{DD2}$ < 5.5 V, 4.75 V< $HV_{DD1} = HV_{DD2}$ < 5.25 V, T_{CASE} = –40 °C to +105 °C

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
f_P	Parallel Resonance Frequency @ CL = 12 pF	4		12	MHz	
R_1	Series Resonance Res. for 50 ms Oscillation Start-Up time @C_L = 12 pF @ f_P = 4 MHz			380 320	Ohm	START-UP RUN
	@ f_P = 6 MHz			230 160	Ohm	START-UP RUN
	@ f_P = 8 MHz			150 95	Ohm	START-UP RUN
	@ f_P = 10 MHz			100 60	Ohm	START-UP RUN
C_{EXT}	External Oscillation Capacitances for C_L = 12 pF, connected to VSS		18		pF	

3.5. Flash/EMU Port Characteristics

Table 3–5: $EV_{DD} = V_{DD}$ = 4.5 V to 5.5 V, T_{CASE} = –40 °C to 85 °C

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t_{PHD}	internal PH2 delay		6	8	ns	
t_{ADS}	Address Setup Time		15 + 0.5 20 30	19 + 0.7 26 40	ns ns/pf ns ns	C_{EADB} = 0 pF C_{EADB} = 10 pF C_{EADB} = 30 pF
t_{ADH}	Address Hold Time		8	10	ns	C_{EADB} = 10 pF
t_{DSR}	Data Setup Read Time		6	12	ns	
t_{DSW}	Data Setup Write Time		9 +0.5 14 24	14 + 0.7 21 35	ns ns/pf ns ns	C_{EDB} = 0 pF C_{EDB} = 10 pF C_{EDB} = 30 pF
t_{DHR}	Data Hold Time Read		6	8	ns	C_{EDB} = 0 pF
t_{DHW}	Data Hold Time Write		8	10	ns	C_{EDB} = 0 pF
t_{WOS}	Write/Output Enable Setup		6	10	ns	$C_{EOQ,EWQ}$ = 0 pF
t_{WOH}	Write/Output Enable Hold		6	9	ns	$C_{EOQ,EWQ}$ = 0 pF
t_{BES}	Bus Enable Setup		14	24	ns	C_{EBE} = 0 pF
t_{BEH}	Bus Enable Hold		7	10	ns	C_{EBE} = 0 pF
t_{VS}	EV_{DA}, EV_{PA} Setup Time		21	30	ns	$C_{EVDA,EVPA}$ = 0 pF

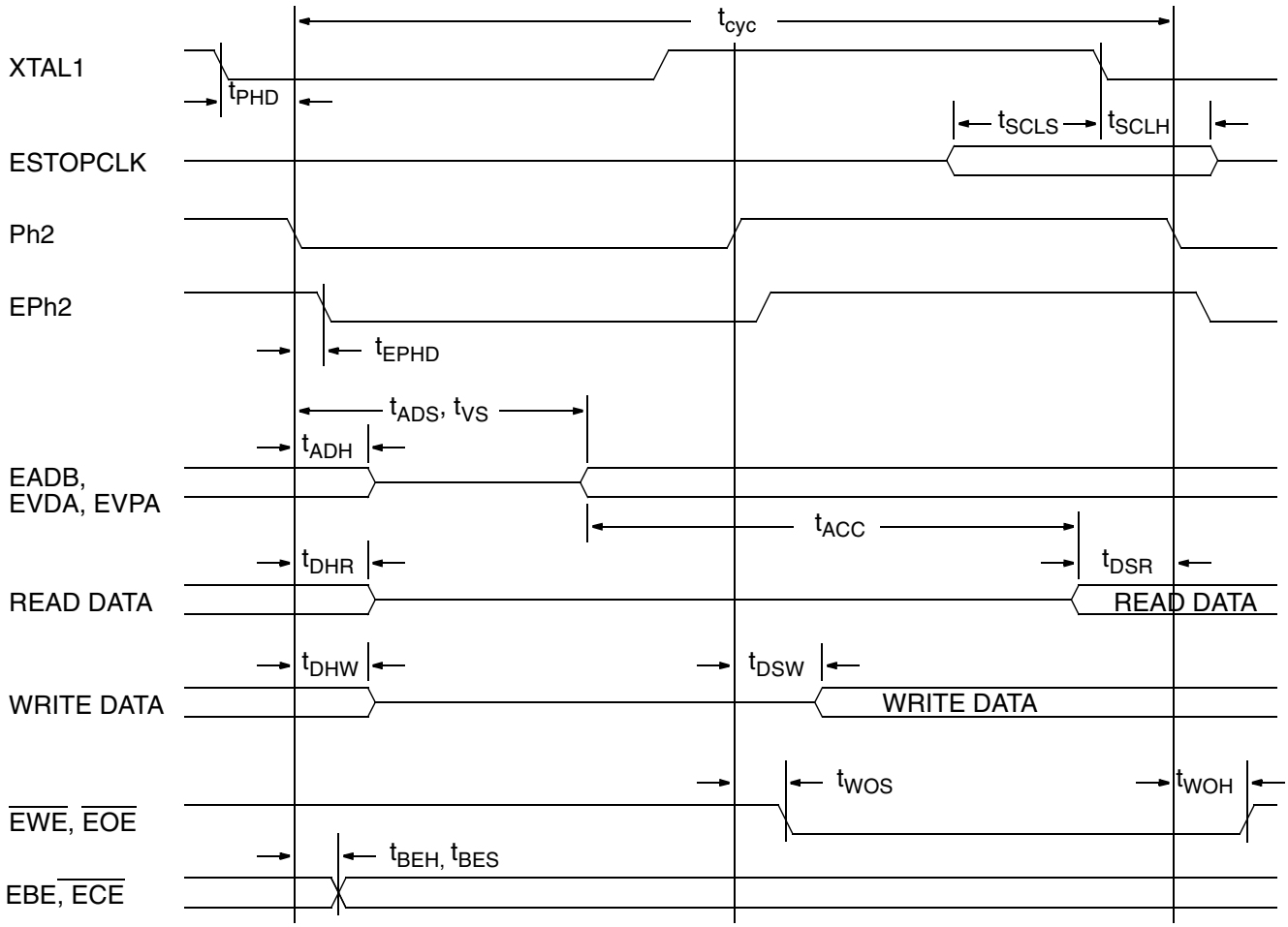


Fig. 3–5: Emu Bus Timing

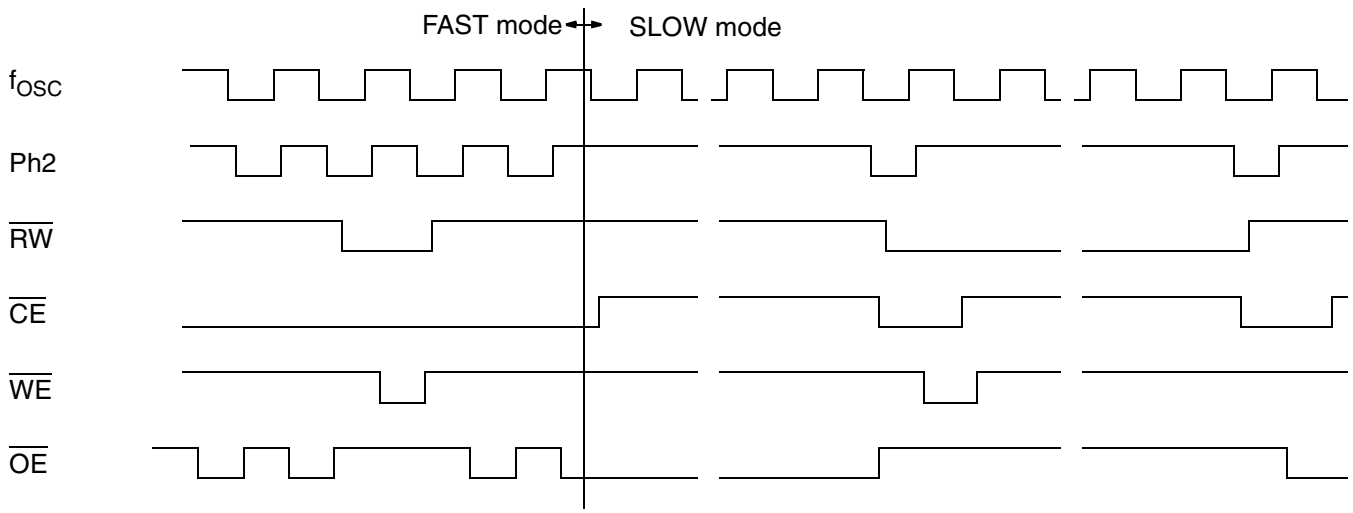


Fig. 3–6: Memory Access Signals

Emulator ports: $\overline{CE}$ (=ECEQ) may be used for low power mode. Input data are latched with the rising edge of $\overline{CE}$.
Test ports: Be careful with $\overline{CE}$ working in low power mode.

There are no input data latches at the test ports. Always pull $\overline{CE}$ down.

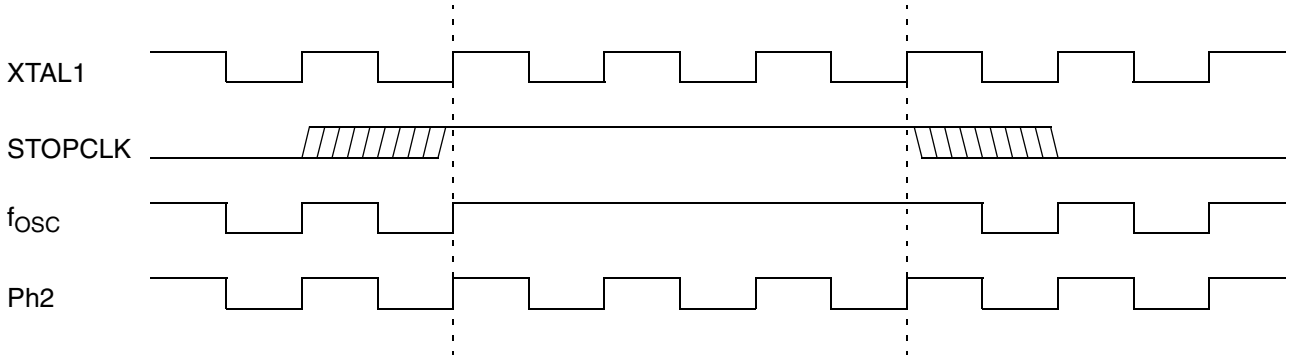


Fig. 3–7: Stop Clock (only if TEST Pin = 1)

STOPCLK is used for emulation and test mode. It is active with TEST input pulled to V_{DD} only. With the stopped f_{OSC} all peripheral modules, such as timers and UARTs, are frozen.

But with the runni ng Ph2 clock the CPU is able to read and write the internal registers and memory. The pin ESTOPCLK has an internal pulldown. The pin STOPCLK (H1.5, Test mode) has no internal pulldown, so in test mode an external pulldown is needed.

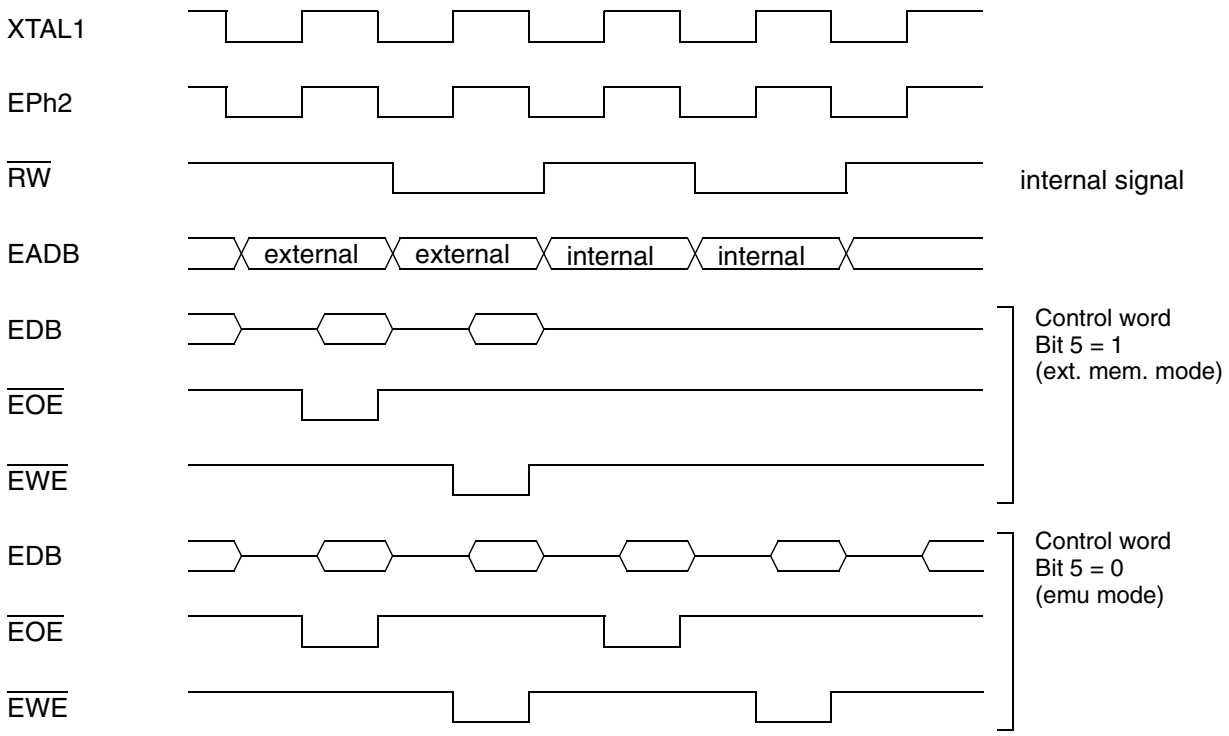


Fig. 3–8: Internal/External Memory Access

4. CPU and Clock System

The core basically consists of the CPU, RAM and ROM.

A Memory Banking module is included to allow access to more than 64 k memory with an address bus limited to 16 bits.

ROM is subdivided in Boot ROM and Flash EEPROM.

In normal operation, after RESET, the CPU starts executing boot loader SW code from the Boot ROM and then jumps into the Flash EEPROM. Please find detailed information in section Boot System.

In mask-ROM-derivative ICs, the code execution starts in factory-defined mask ROM.

4.1. W65C816

The CPU is fully compatible to WDC's W65C816 microprocessor. This is a processor with 16-bit registers/accumulator, an 8-bit data bus and a 24-bit address bus. A software switch determines whether the processor is in the 8-bit emulation mode, supporting SW compatibility to its 8-bit predecessors, or in the 16-bit native mode. For further information on the CPU core, please refer to the WDC W65C816 Data Sheet.

Table 4–1: Major Differences between Processors and Modes

Item	65C816 Emulation	65C816 Native	65C02
ABS, X ASL, LSR, ROL, ROR with no page crossing	7 cycles	7 cycles	6 cycles
jump indirect, operand = XXFF	5 cycles	5 cycles	6 cycles
branch across Page	4 cycles	3 cycles	4 cycles
decimal mode	no additional cycle	no additional cycle	add 1 cycle
RWQ signal during read-modify-write instructions	RWQ=0 during modify and write cycles	RWQ=0 only during write cycle	RWQ=0 only during write cycle
abort signal	yes	yes	no
accumulator	8/8 bits	16 or 8/8 bits	8 bits
addressing modes	25	25	16
address space	16 M	16 M	64 K
bank registers	yes	yes	none

Table 4–1: Major Differences between Processors and Modes

Item	65C816 Emulation	65C816 Native	65C02
block moves	of little use	yes	none
break flag	yes	no	yes
break vector	FFFE.FFFF	FFE6.FFE7	FFFE.FFFF
direct page indexed	wraps	crosses page	wraps
flags after interrupt	D=0	D=0	D=0
flags after reset	D=x	D=0	D=0
index registers	8 bits	8 or 16 bits	8 bits
instructions	256	256	178
interrupts	FFF4.FFFF	FFE4.FFEF	FFFA.FFFF
mnemonics	92	92	64
special page	direct page	direct page	zero page
stack	page 1	bank 0	page 1
unused opcodes	none	none	NOP

4.1.1. Processor Modes

The 65C816 CPU allows operation in two modes:

– Native mode: 16-bit mode

– Emulation mode: 8-bit mode for emulation of 65C02 properties

Table 4–1 gives some details on the differences between modes. Refer to the WDC W65C816 Data Sheet for more information and on how to switch between modes.

After reset, Emulation mode is active.

However, Native mode is recommended for normal use. To make maximum usage of the CPU's 16-bit properties, switch to Native mode after reset.

4.1.2. Emulation of 65C02

When using the Emulation mode to design software for a derivative containing only the 65C02 8-bit CPU, care has to be taken to use only the features available with that CPU. Table 4–1 gives a comparing overview. Refer to the WDC W65C816 Data Sheet for more information.

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 only certain portions of the circuit are powered.

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

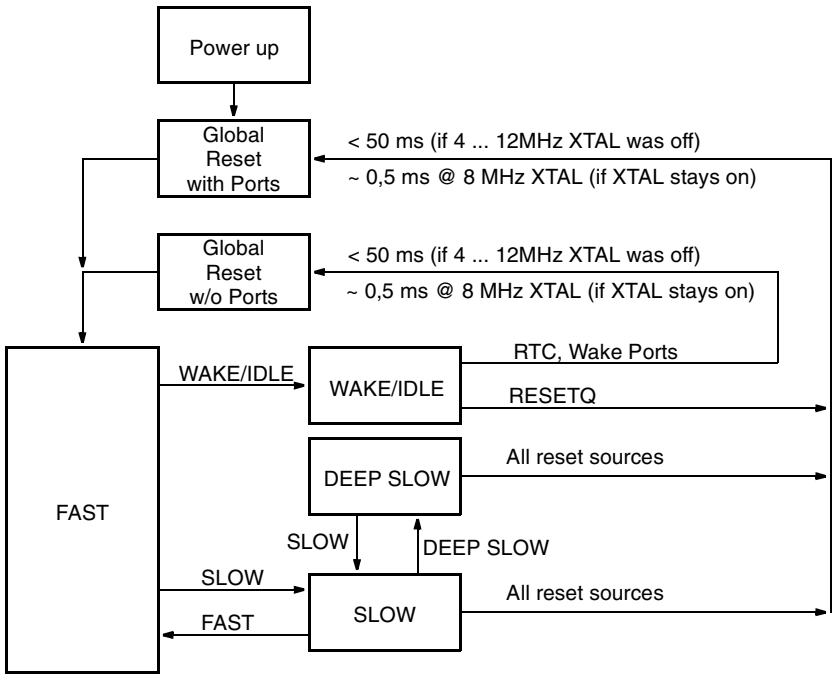


Fig. 4–1: Operating Modes State Diagram

4.2.1. CPU-Active Modes

The CPU can be operated in three different CPU-active modes (Table 4–2). 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 4–2 shows the operability of the peripheral modules in the various CPU-active modes.

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}.

4.2.1.2. SLOW Mode

To considerably reduce power consumption, the user can reduce the internal CPU clock frequency to 1/256 of the normal f_{XTAL} value. In this CPU SLOW mode, program execution is reduced to 1/256 of the normal speed, but clocking of most other modules remains unaffected. The modules that are affected by CPU SLOW mode are:

1. CPU and Interrupt Controller with all internal and external interrupts

2. RAM, ROM and DMA

3. Watchdog

Some modules must not be operated during CPU SLOW mode (e.g. CAN). Refer to module sections for details (see Table 4–2 on page 35).

CPU SLOW mode is enabled by clearing flag CPUFST in standby register SR1. The CPU clock frequency reduction to $f_{XTAL}/256$ will take effect after a maximum delay of 256 f_{XTAL} periods.

Returning CPU to FAST mode is done by setting flag CPUFST to HIGH. The CPU clock frequency will immediately change to its normal f_{XTAL} value.

4.2.1.3. DEEP SLOW Mode

To further reduce power consumption beyond SLOW mode, DEEP SLOW mode also disables most of the internal peripheral clocking system. Table 4–2 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.

To prevent undefined behavior don't switch between DEEP SLOW and FAST mode directly but select SLOW mode in between.

Changing to or from DEEP SLOW mode is done by writing SR3.FCLO (see Section 6.3. "Standby Registers" on page 55 for details).

4.2.2. Power-Saving Modes

Power-saving modes are activated by the CPU. The complete core logic will immediately terminate operation and power will be reduced. The result is a device current consumption that is greatly reduced, to the amount of leakage currents. The internal SRAM and CAN-RAM keep their programmed data and all U, P, and H-Port registers keep their programmed state.

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 necessitates 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 avoid this hazard.

4.2.2.1. WAKE Mode

The WAKE mode is the most current-saving operation mode. All device circuits are stopped or powered down except the Port Wake Module (Table 4–3).

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 sets SR3.WAID.

The device will immediately enter WAKE mode by resetting all circuitry, stopping all clocks, and powering down all analog circuitry. As long as all Port inputs - except analog inputs via P-ports P0.1 to P0.9 - 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 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, which 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. IDLE Mode

IDLE 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. These circuits and the RTC are kept alive (Table 4–3) as well as the Port Wake Module.

The RTC allows the CPU to select from one-second to one-day clocks as wake signal (see Section 8.4. "Operation of RTC Module" on page 69 for details). Apart from serving as wake source, the CPU may use the RTC as a 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 H2.5 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 IDLE mode, the CPU has to configure the desired RTC wake clock (see Section 8.4. "Operation of RTC Module" on page 69 for details), beside the desired Wake Port(s) (see Section 8.6. "Operation of Port Wake Module" on page 71 for details).

To enter IDLE mode, the CPU sets SR3.WAID.

The device will immediately enter IDLE mode by resetting all circuitry, stopping the unused clocks, and powering down all

Table 4–2: CPU-Active Modes and their effect on peripheral modules

Module	FAST	SLOW	DEEP SLOW
Core			
Digital Watchdog	✔	✔	✔
IR Interrupt Controller Unit	✔	✔	✔
Port Interrupts	✔	✔	✔
Port Wake Module	✔	✔	✔
Memory Patch Module	✔	✔	✔
Analog			
A/D Converter	✔	✔	
ALARM, P06 and Comparators	✔	✔	✔ [3])
LCD Module	✔	✔	✔ [2])
Communication			
DMA	✔	✔	✔
UART	✔	✔	
SPI	✔	✔	
CAN	✔	✔ [3])	
DIGITbus	✔	✔ [3])	✔ [2]) [3])
Input & Output			
Ports	✔	✔	✔
Stepper Motor Module	✔		
PWM	✔	✔	
Audio Module	✔		
Clock Outputs	✔	✔	✔ [2])
Timers & Counters			
Capture Compare Module	✔	✔ [1])	✔ [2]) [3])
Timers	✔	✔	✔ [2])
RTC/Polling Module	✔	✔	✔

[1]) Avoid write access to CCxI
[2]) Only clocks f5 and slower are available from Clock Divider
[3]) Don't access registers or CAN RAM

analog circuitry. As long as all Port inputs - analog inputs via P-ports P0.1 to P0.9 excepted - 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 leakage and the requirement of the enabled oscillator(s) and the slow-clocked RTC and Polling Modules.

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. See chapter "Core Logic" for details on internal reset sources. After reset, the CPU starts in FAST mode, as usual.

Table 4–3: Power Saving Modes and related functionality vs. CPU-Active Modes

Operating Mode		Modules which can be activated:	SRAM, CAN-RAM	Port Registers	Available Wake Sources
Power Saving Modes	WAKE	- Port Wake Module	data retained	state retained	Wake Ports
	IDLE	all WAKE Mode modules plus: - 4 ... 12 MHz XTAL and/or 20..50 k RC Oscillator - Real-Time Clock - Polling Module			Wake Ports and RTC
CPU-Active Modes		in principle all, for limitations see Table 4–2	active	active	Wake sources usable as interrupt

4.3. Clock System

This IC contains a quartz oscillator circuit that only requires external connection of a quartz and of 2 oscillation capacitors.

The XTAL Oscillator generates a 4 to 12 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 SR3.XTAL may be cleared by SW to set the XTAL Oscillator to RUN mode, where current consumption is at its standard level.

Switching between START-UP and RUN modes must not be done @ F_{QUARZ} >= 10 MHz or with the ERM active, as this might lead to unpredictable behavior of the clock system.

This module is permanently active except during power saving mode, where continued operation may be selected in register OSC.XM.

Table 4–4: Operating Mode Selection and Effect on Clocks

SR1.CPUFST	SR3.FCLO	SR3.WAID	Operating Mode	PH2	f_0
0	0	0	SLOW	$f_{XTAL}/256$	f_{XTAL}
1	0	0	FAST	f_{XTAL}	f_{XTAL}
0	1	0	DEEP SLOW	$f_{XTAL}/256$	f0 to f4 = V_{DD}
1	0	1	WAKE/ IDLE	V_{DD}	V_{DD}

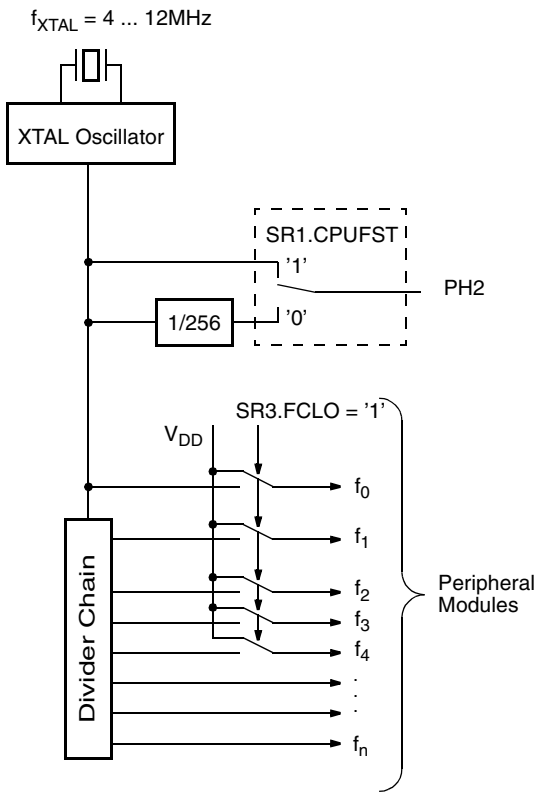


Fig. 4–2: Clock System

The oscillator clock drives a system clock divider that supplies the various modules with its specific clock. Module clock selection is software-defined in some cases, hardware or HW option-defined in other cases. The module descriptions give details.

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%, respectively 20% duty factor.

Two Clock Output signals CO0 and CO1 provide external visibility of internal clocks.

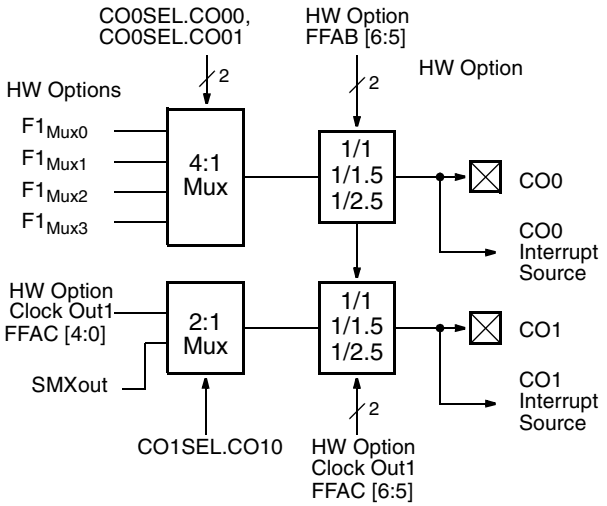


Fig. 4–3: Clock Outputs Diagram

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 4–5). 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 4–5) have to be configured 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 CPU SLOW mode.

Table 4–5: HW Options and Ports

Signal	HW Options		Initialization	
	Item	Address	Item	Setting
CO0	CO0 Prescaler	FFABh	CO0 output	U5.5 special out
	F_{Mux0}	FFABh		
	F_{Mux1}	FFB2h		
	F_{Mux2}	FFB3h		
	F_{Mux3}	FFB6h		
CO1	CO1 Prescaler	FFACh	CO1 output	U3.1 or U5.0 special out
	Clock Out	FFACh		
	SMX Out	-		

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 4–6: Clock Out 0 Selection

CO01	CO00	
0	0	$F1_{Mux0}$
0	1	$F1_{Mux1}$
1	0	$F1_{Mux2}$
1	1	$F1_{Mux3}$

CO1SEL Clock Out 1 Selection

	7	6	5	4	3	2	1	0
w	x	x	x	x	x	x	x	CO10
	x	x	x	x	x	x	x	0

Res

CO10 WAKE/IDLE
w 0 : Clock Out
w 1: SMX Out

4.4. EMI Reduction Module (ERM)

The EMI Reduction Module distributes the electromagnetic energy radiated from VLSI embedded controllers among many spectral lines in the frequency spectrum.

The module performs modulation of the quartz clock phase by selecting delay taps in a delay-locked loop in a pseudo random manner, to arrive at the system clock. The individual system clock edges thus appear delayed by pseudo randomly selected time values ranging from 0% to 10% of the quartz signal cycle length.

Differening from PLL-based EMI reduction devices, the maximum dislocation of a system clock edge from its quartz reference is only +10% of the quartz signal cycle length.

Due to its inherent frequency stability the module is fully applicable to ICs containing clocks, timers and synchronous communication links to other systems, e. g., CAN interfaces.

Features

– Reduction of Clock Related Electromagnetic Interference
– DLL-Based Modulation of Clock Phase
– Full Quartz Frequency Stability
– Maximum Clock Edge Dislocation +10% of Cycle

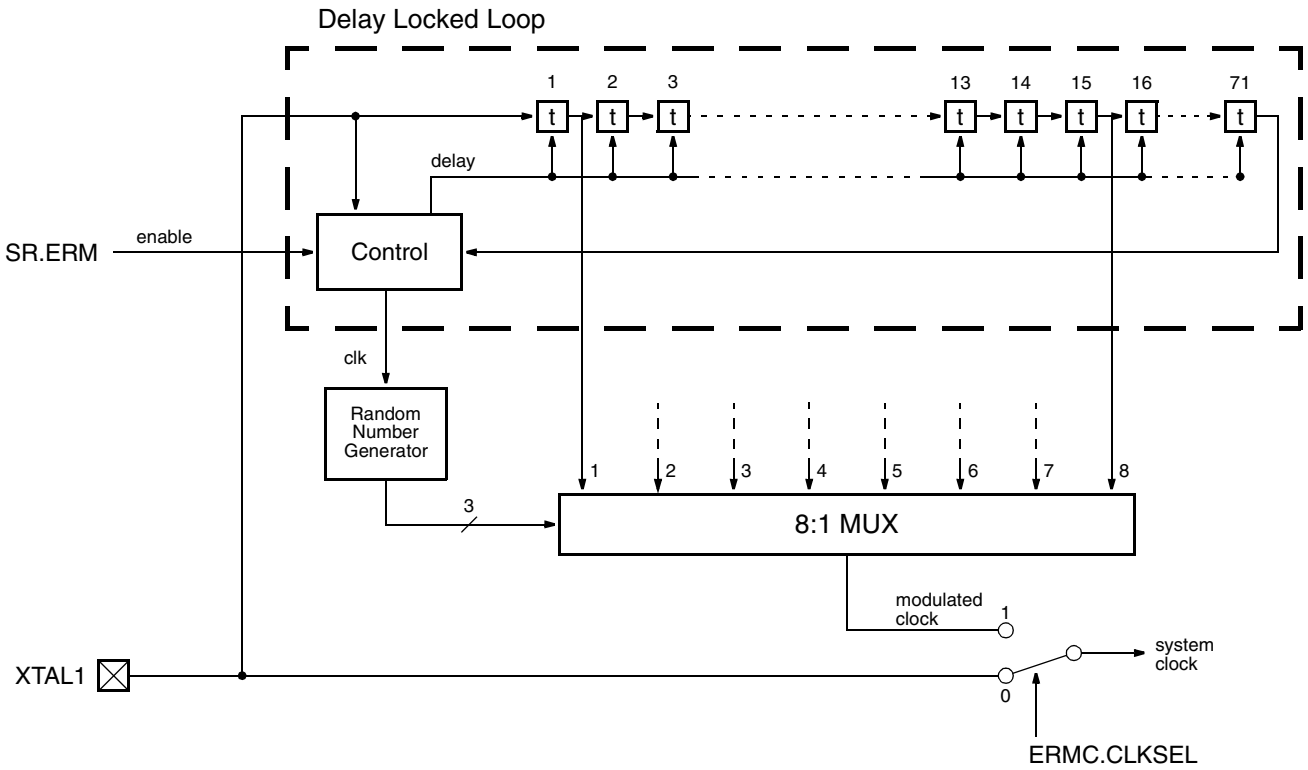


Fig. 4–4: Block Diagram

4.4.1. Principle of Operation

4.4.2. General Remarks

The edges of the input clock are discretely delayed by 1/71 to 15/71 (21.1%) of the low time of a clock cycle (see Fig. 4–5). The selection of the clock edge is done by means of a pseudo-random sequence.

The integral parts of the ERM are the Delay Locked Loop (DLL) and the Random Number Generator (RNG).

The DLL consists of a chain of 71 controllable delay elements. Within a locked loop, the total delay time of the chain is controlled to equal the low time of the input clock. To reduce the influence of supply noise the DLL is operated on AVDD.

The RNG generates a sequence of pseudo-random values. Three bits serve to select one output from the first eight delay elements. The selected output represents the modulated clock. The selection of the three bits achieves a “high randomness” even for short time intervals. In the worst case the same output is selected three times in succession.

4.4.3. Initialization

After Reset the EMI Reduction Module is in standby mode (inactive). In standby mode, all internal registers are reset.

Setting the standby bit SR1.ERM activates the ERM.

Before entering operation, a setup time of at least 100 µs has to elapse.

4.4.4. Operation

Setting register ERMC to 01h immediately selects the phase modulated clock as system clock. Flag ERMC.CLKSEL is readable and indicates the busy state.

4.4.5. Inactivation

To deactivate the ERM first deselect the modulated clock by resetting register ERMC to 00h. Then SR1.ERM may be reset to return the whole module to standby mode.

4.4.6. Precautions

For all modules using the modulated system clock or derived clocks, the following facts have to be kept in mind:

– The maximum operating frequency is reduced.

– The sampling time point of clocked inputs is modulated. For this reason, e.g., a CAN or UART module may appear less failure-tolerant.

– The output timepoint of clocked output signals is modulated as well.

– The sampling time of an ADC is modulated slightly.

4.4.7. Results of Application

According to product spectrum measurements of supply current and electromagnetic radiation, the EMI Reduction Module is capable of reducing the energy in spectral lines of the frequency spectrum which occur on multiples of the oscillator frequency.

The ERM distributes the spectral energy contained in oscillator harmonics over the spectrum between them. Thus the noise level is increased by a slight 1 to 2 dB.

The higher the share of system events that are in synchronism with the oscillator (current peaks due to clocking in digital circuits), the more distinct the reduction in the spectrum.

The reduction takes effect from the 6th oscillator harmonic up, and reaches values from 6 to12 dB over a wide range of frequencies.

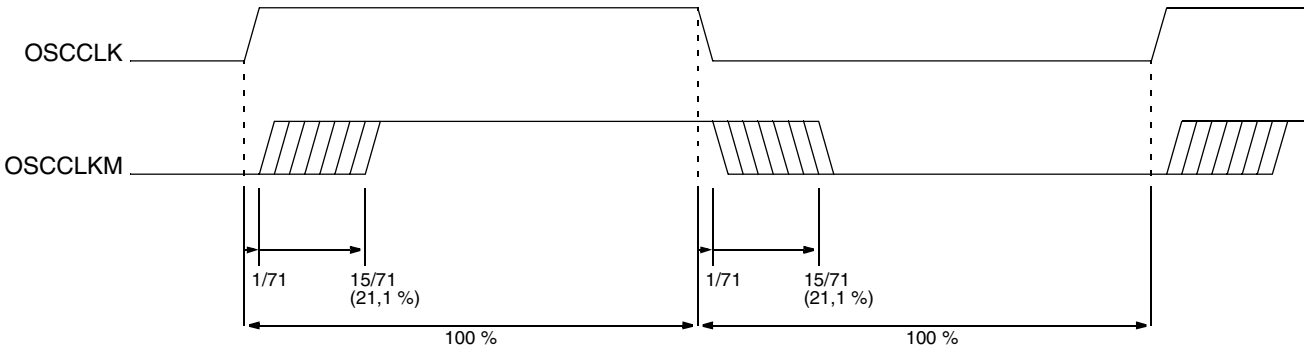


Fig. 4–5: Timing of modulated clock

4.4.8. Registers

ERMC **EMI Reduction Module Control Register**

	7	6	5	4	3	2	1	0	
r	x	x	x	x	x	x	x	CLKSEL	
w	x	x	0	0	0	0	0	CLKSEL	
	x	x	0	0	1	0	0	0	Res

CLKSELClock Select
r/w0:System clock is unmodulated
r/w1:System clock is modulated

Other bits in this register are used for test purposes. They must all be written to zero for proper operation.

5. Memory and Boot System

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 boot ROM contains up to 2 KB of firmware boot code. The functionality is described in section Boot System.

The 100PQFP Multi Chip Module also contains a 256 KB Flash EEPROM of the AMD Am29F200AB type (bottom boot configuration) or Am29F200AT type (top boot configuration). These devices exhibit electrical byte program and sector erase functions. Erase sectors are of various sizes (Fig. 5–1). Refer to the AMD data sheet for details.

Future mask ROM derivatives will contain no Boot ROM and may be specified to contain less internal RAM and ROM than this IC. RAM will always grow upwards from physical address 000000h. ROM will grow down from 00FFFFh to 002000h and then upwards from 010000h.

5.1.1. Address Map

Emulation mode: E=1
Native mode: E=0

Table 5–1: Reserved (physical) Addresses

Addresses	Mode	Usage
00FFA0 - C3	Emu & Native	HW Options
00FFC4 - E3	Emu & Native	Interrupt Controller Vectors
00FFE4 - E5	Native only	(reserved) COP
00FFE6 - E7	Native only	BRK
00FFE8 - E9	Native only	ABORT
00FFEA - EB	Native only	NMI (expanded by Interrupt Controller)
00FFEC - ED	Emu & Native	reserved
00FFEE - EF	Native only	IRQ
00FFF0 - F1	Emu & Native	Manufacturer ROM Identification
00FFF2	Emu & Native	reserved
00FFF3	Emu & Native	Control word
00FFF4 - F5	Emu only	(reserved) COP
00FFF6 - F7	Emu & Native	reserved
00FFF8 - F9	Emu only	ABORT
00FFFA - FB	Emu only	NMI (expanded by Interrupt Controller)
00FFFC - FD	Emu & Native	RESET
00FFFE - FF	Emu only	IRQ/BRK

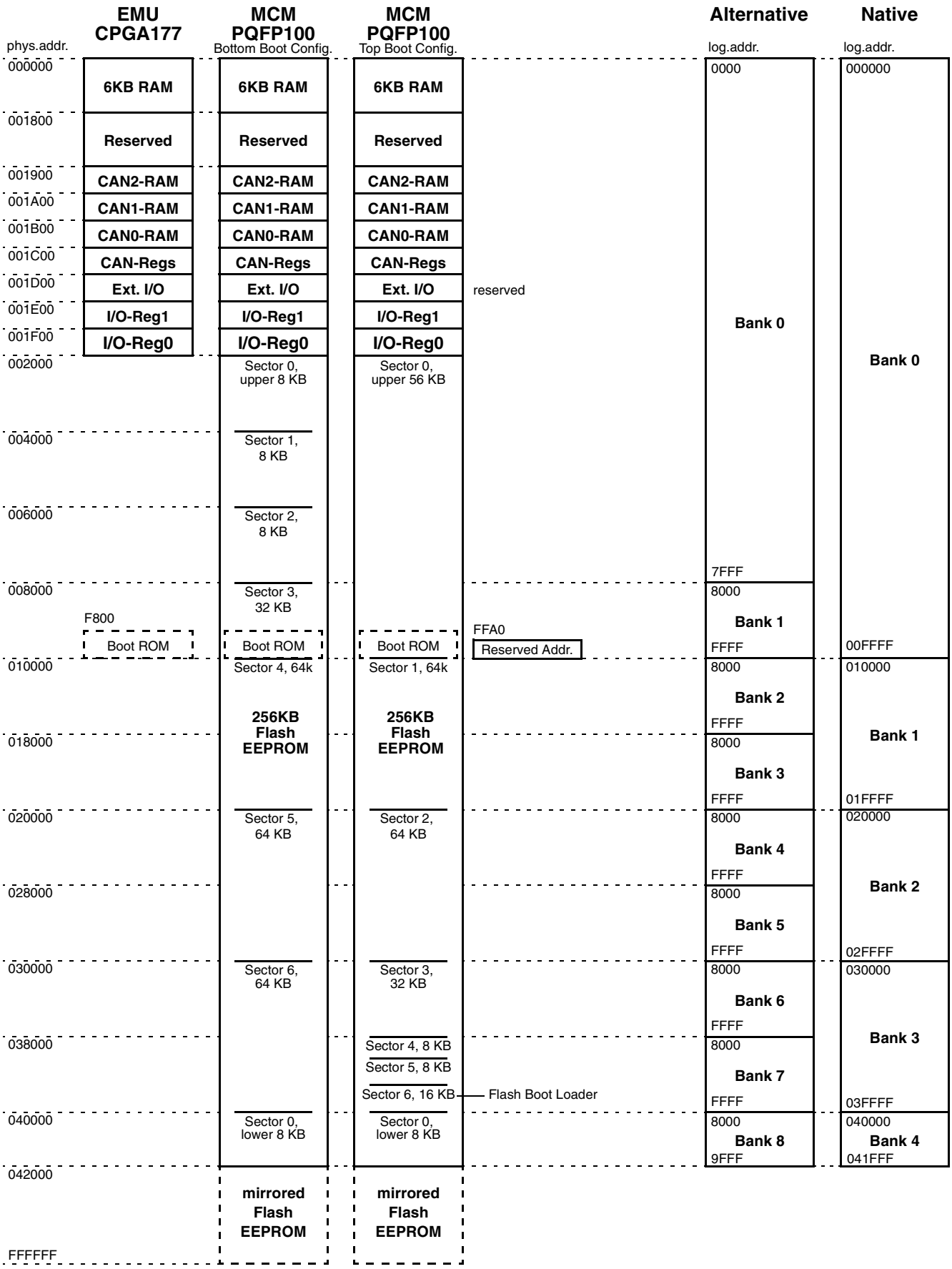


Fig. 5–1: Address Map

5.2. Memory Banking

The 24-bit address bus of the 65C816 CPU allows access to 16 MB of memory space. The upper 8 bits of the addresses are delivered as bank address, which is multiplexed with the data value on the internal data/address lines of the CPU. This kind of native banking is supported by the 65C816 internal bank registers (PBR, DBR) by using the processor's native 16-bit instruction set.

The CPU may be used to emulate the behavior of the 8-bit processor W65C02. This CPU only allows access to 64 KB of memory space. To allow access to the expanded memory range above 64 KB, an alternative banking logic is implemented.

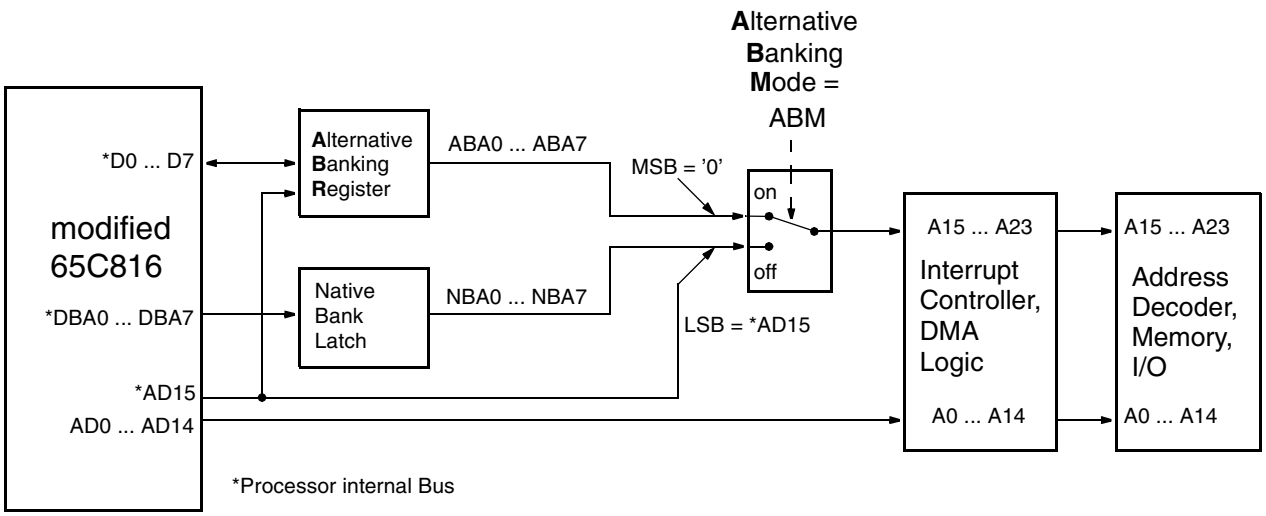


Fig. 5–2: Block diagram Memory Banking

The banking mode is toggled with flag **ABM** of the **SR2** register:

ABM = '**0**': **Native** Banking mode
(default after RESET),

ABM = '**1**': **Alternative** Banking mode.

The alternative bank no. is programmed in the **Alternative Banking Register** = **ABR**.

5.2.1. Native Banking Mode

The bank address is present on the 65C816 data/address lines during the first half of each processor bus cycle. To get the whole address bus available during the second half of the bus cycles, the bank address is demultiplexed with a transparent latch, triggered with the processor clock, PHI2(in). This mode is supported by the native instruction set since it supports the CPU's internal bank registers PBR and DBR, which values are multiplexed on the data/address lines, no matter if the processor is running in native or emulation mode. If the internal bank registers are not supplied, e.g., if only the 8-bit instruction set is applied, the address range is limited to 64 KB, the size of one Native Bank, because the banking registers keep their initial 0-values which they get during RESET.

After RESET the native banking mode is active and the whole 65C816 address range of 24 bits is available.

By setting ABM to "1" the native banking mode is switched off, and the alternative banking mode is activated instead.

5.2.2. Alternative Banking Mode

To use the CPU with the 8-bit instruction set of the 65(C)02 and reaching more than the addressable 64 KB, a specific banking hardware is implemented: The physical address range above 32 KB is separated into several banks of which only one at a time is enabled and selected by the 8-bit ABR (Alternative Banking Register), which is programmable like any other standard 8-bit peripheral register by writing the desired value into its specific address. The contents of the ABR are also readable, so the software may check the current bank at any time. The applied software is responsible for programming the ABR with the correct bank number at the right time. Since the upper 32 KB range is switched immediately after programming the ABR, correct function is not guaranteed if it´s changed by a program sequence running in a switched bank. ABR settings need to be done in the lower 32 KB, which is the non-switchable master bank, resp., alternative bank number 0.

With ABR = 0, the lower 32 KB appear as bank 0 in the upper address range, so bank 0 is identical to the non-switchable master bank. Be careful when operating bank 0 in the upper 32 KB area. RAM, I/O pages and reserved addresses may be manipulated unintentionally.

RESET initializes ABR = 1, so as to have control byte and reset vector in the same physical addresses as with active native banking mode. Also Interrupt vectors have to reside in

the alternative bank number 1, because the interrupt controller generates the appropriate address of bank 1, but does not change the contents of the ABR. Interrupt functions have to reside in the non-switchable master bank (alternative bank 0). Otherwise they need to be in each used bank, as after getting the vector, the unchanged contents of the ABR determine the current bank which is valid if A15 is “1”.

After RESET the alternative banking mode is inactive. By setting ABM to “1”, the alternative banking mode is switched on.

ABR **Alternative Banking Register**

	7	6	5	4	3	2	1	0	
r/w	Alternative Bank Address								
	0	0	0	0	0	0	0	1	Res

5.2.3. Memory Banking Mode Selection

In the Native Banking Mode the logical address range is identical to the physical one. In the Alternative Banking Mode A15 is lost, because a non-switched address range is necessary. This cuts the addressable space in half. To retain the physical address range without holes as is in Native Banking Mode, the addresses are multiplexed.

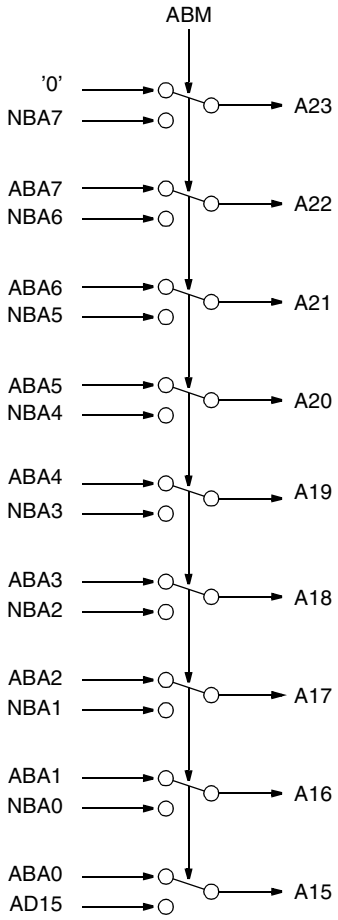
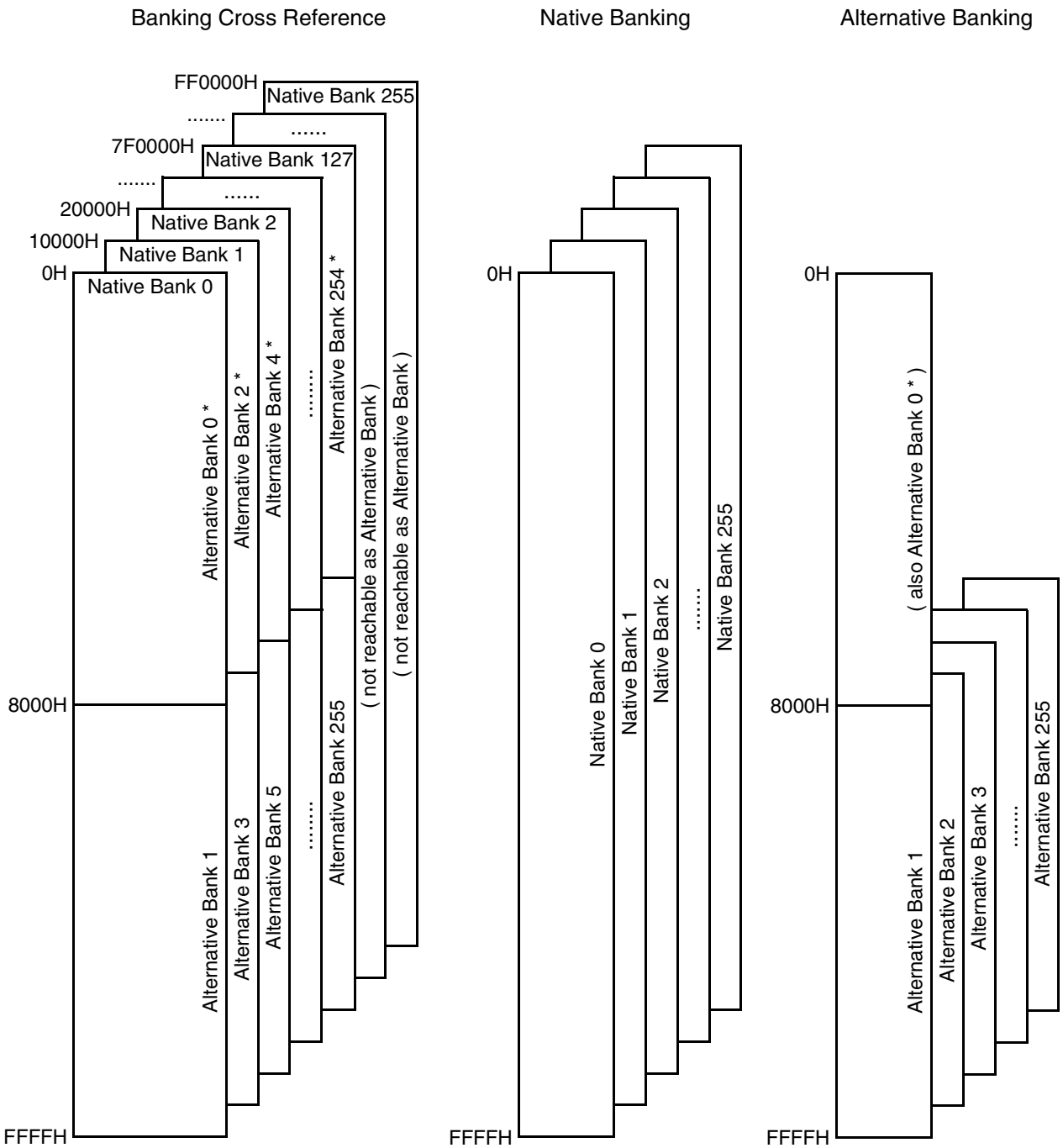


Fig. 5–3: Physical Address change depending on the Banking Mode



* The logical address range is 8000H ... FFFFH
Native Banks: logical addresses = physical addresses
Alternative Banks: logical address range = 8000H ... FFFFH

Fig. 5–4: Banking Map shown with the max. size of addressable memory (different from the implemented amount)

5.3. Boot System

The Boot System offers a very flexible method for the controller to receive and store data and programs, no matter if the Flash memory contains a working or faulty application, or no application at all. With the Boot System it is possible to fill the RAM with data and functions and to start execution from any address. Tasks like flash programming and diagnostic routines may be downloaded and started, either in lab, factory or in system.

After RESET, the CPU executes code from the Boot ROM. Its firmware, the Boot Loader, checks whether the application software in the flash memory must be started or if data must be downloaded via one of 6 receivers: 3 UARTs and 3 CANs.

5.3.1. Principle of operation

After RESET the Boot ROM is active, if the Test pin is left vacant or connected to ground. The Boot Loader is started and checks whether there is a wake-up from Power Saving Mode (see Table 6–7 on page 54), i. e., if CSW2.WKID is set. If so, it starts the application software in Flash Memory (or ROM). Otherwise, it tries to detect a download condition.

If no appropriate data can be received via either of its 6 interface receivers within a predefined time, the Boot Loader terminates itself by copying a program sequence into the RAM and executing it. This software part switches the Boot ROM off and starts the application software assumed within the Flash Memory. The detour via RAM is necessary because the Boot ROM hides the start-up address area of the flash memory that contains the processor Control Word and RESET Vector of the application.

When detecting a download condition, the time-out condition to terminate the boot sequence is extended with the detection of every new appropriate data until the download is completed. The Boot Loader jumps to a start address, which is assumed part of the code-header downloaded before.
Neither the Boot ROM is switched off, nor the Control Word or RESET Vector of the flash memory are treated automatically if a download occurs. This leaves a maximum of flexibility during and after the boot procedure.

5.3.2. The Boot ROM

The 2 kB Boot ROM covers the address range from 0F800H to 0FFFFH. With Test pin left vacant or pulled to low level at RESET, the Control Word is read out of the Boot ROM and copied into the Control Register. With the flag FLASH of the Control Register set to ’0’ the Boot ROM stays switched on and its control program, the Boot Loader, is started.
In a standard application (no download request) the control program in the flash memory is started after the Boot Loader has set the flag FLASH in the Control Register to ’1’ to switch the Boot ROM off at the appropriate time, which enables the Flash ROM in the same address range as the Boot ROM before.

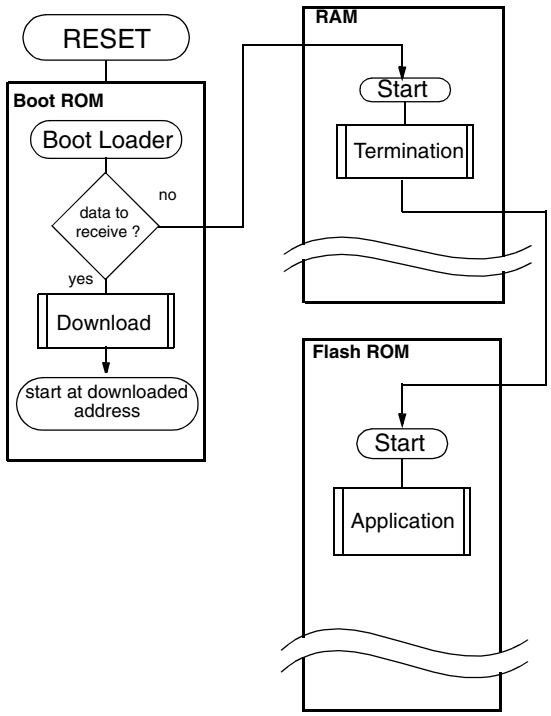


Fig. 5–5: Principle of the boot system

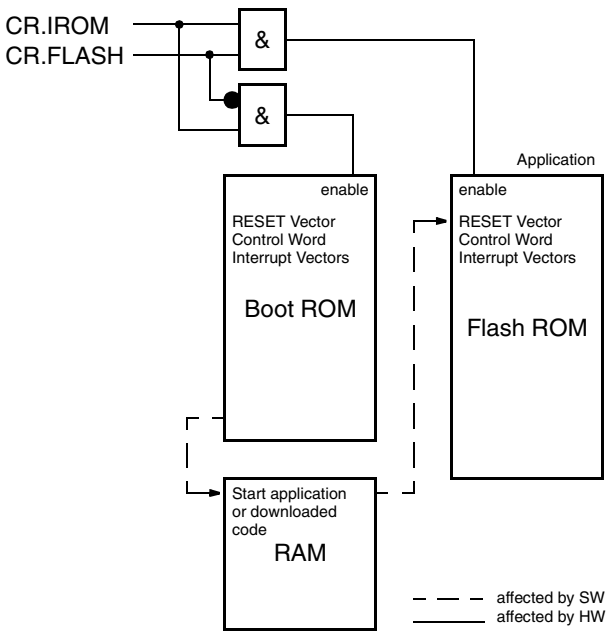


Fig. 5–6: Boot ROM Selection

5.3.3. The Boot Loader

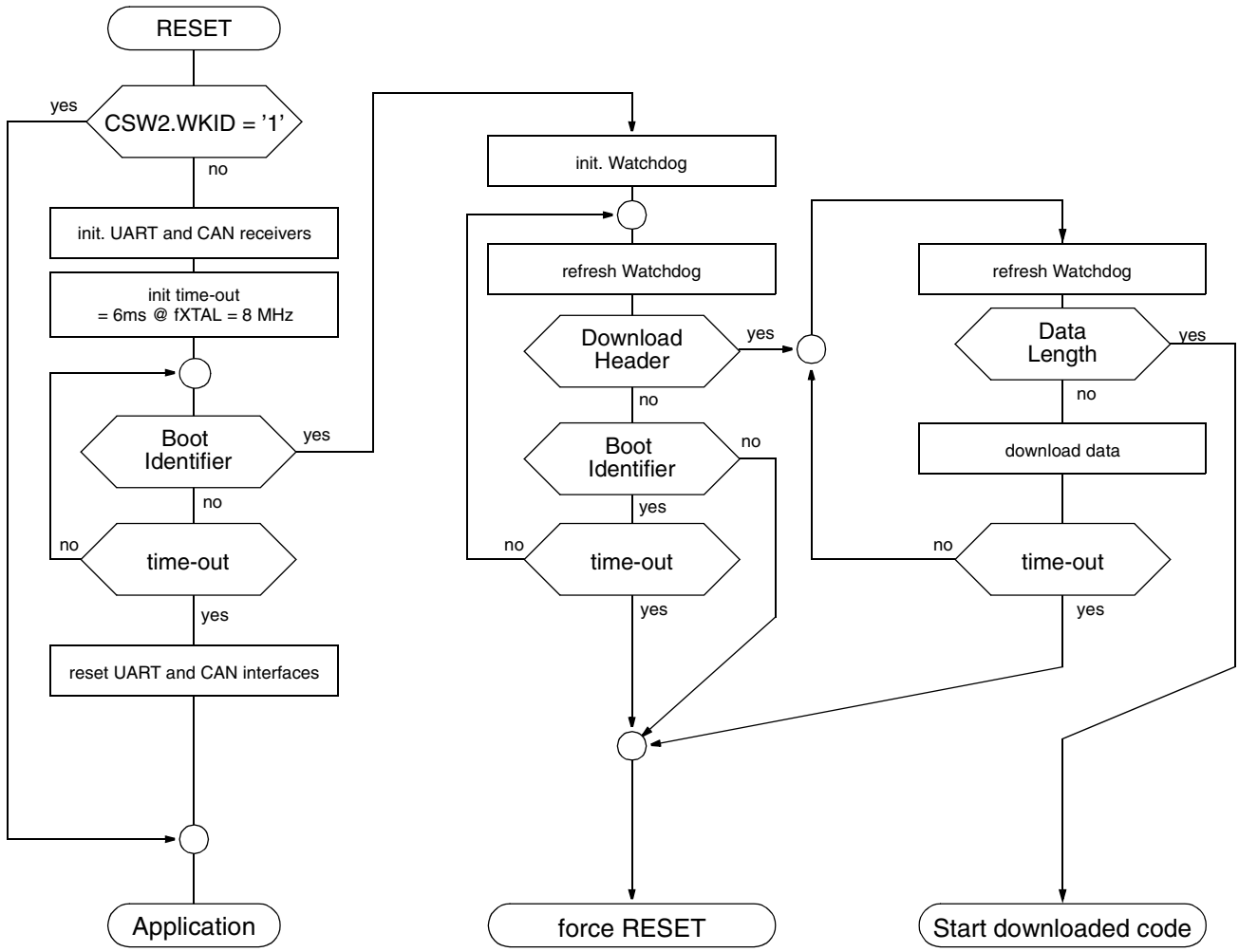


Fig. 5–7: Boot Loader flow-chart

During the first 480000 clock cycles (= 6 ms at 8 MHz processor clock) after reset, the Boot Loader tries to detect a start of the download condition, the so-called Boot Identifier, at one of the receivers of the 3 UARTs and 3 CANs.
If the complete Boot Identifier has been received within the appropriate time, the watchdog is initialized (with the value 6 for effective 6.144 ms at fXTAL = 8 MHz) to supervise further time-out conditions. This is not only to increase security, but also because the Boot Loader could have been started by a reset other than power-on, from a running application in which the watchdog was programmed before. Therefore the Boot Loader has to program it, too, otherwise it generates a reset after the max. WD time-off, which is default after reset.
Within further time-out conditions additional Boot Identifiers may occur, followed by the Download Header and the Download Data including separate Check Sums. At the end of the procedure the processor executes a jump to the start address which is part of the Download Header.
If a time-out condition is met, an unexpected byte has been received or if a Check Sum error occurs, a reset is forced by the watchdog to start the Boot Loader again. This subsequently will start the application because of absence of the Boot Identifier.

If the Boot Loader does not detect the Boot Identifier within the time limit, the application software assumed in the Flash ROM is started. For thís purpose a software part is copied into RAM and started. It toggles the flag FLASH of the Processor Control Register to switch the BOOT ROM off and reads the Control Word out of the Flash ROM, which is accessible now, and programs the processor's Control Register with that value. Last, it starts the application by a jump to the address defined in the RESET Vector of the Flash ROM.
The application can program its own watchdog value, as it was not treated by the Boot Loader before.

5.3.4. Used RAM

The Boot Loader uses and thus destroys the contents of addresses F5H to FFH in zero page, as well as 100H to 10AH and 1E8H to 1FFH in (stack) page 1, and 200H to 20DH in page 2.

5.3.5. Interfacing to a Download Source

Either one of the UARTs or CANs can be used to activate a download. The Boot Loader assumes a single-bus line as connection to the host. So either wired-or Rx/Tx interfaces or separate driver circuits may be used.

To allow download to several targets connected to the same bus, the Boot Loader does not generate a receipt. Since the UART and CAN busses work asynchronously, target response would result in bus collisions.

5.3.6. Interface Protocol

The interface protocol falls into three sections:

1. **Boot Identifier**. It is continuously transmitted by the host. After reset, the Boot Loader waits for the Boot Identifier.

2. **Download Header**. It contains the start address, target address and data length.

3. **Download Data**. After the Download Header, the announced number of data bytes are transmitted by the host. The first byte is stored in the address defined in *target address*, the next one in *target address* + 1, and so on. After reception of the last byte the processor immediately executes a jump to the address in *start address*. Because the watchdog is treated during download, the downloaded program must also refresh it with the same value (= 6 for effective 6.144 ms at fXTAL = 8 MHz), starting with 06H, followed by its complement 0F9H, 06H, 0F9H, ... and so on.

No more than 6 ms (at fXTAL = 8 MHz) delay is allowed as delay between the single telegrams during boot identification and download, otherwise the Boot Loader stops reception and starts the application in Flash.
Time-out value (6 ms) and baud rate of UART and CAN relate to the processor clock (fXTAL).

As start of a download, the host has to keep generating Boot Identifiers while the targets get a reset. After a minimum of time, when the target running is stable, the host sends the Download Header, followed by the Download Data.

5.3.6.1. UART Protocol and Timing

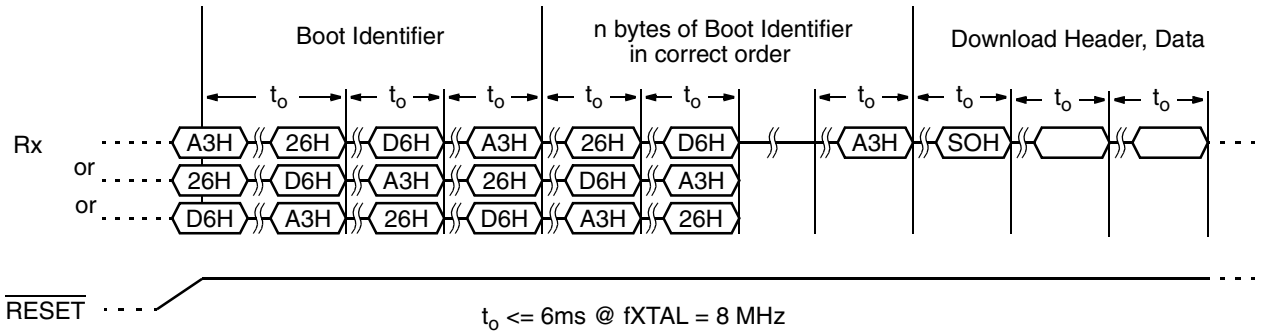


Fig. 5–8: UART Timing

The expected telegram format is: 1 Start bit, 8 Data bits, 1 even parity bit and 1 Stop bit at a rate of 9600Bd at 8 MHz processor clock. To use standard K-bus drivers the UART transmitter inverters are switched on.

Boot Identifier

After reset the Boot Loader tries to detect the Boot ID which consists out of 3 bytes with the values 26H, D6H and A3H respectively, of which each is expected within 6 ms (at fXTAL = 8 MHz) after the preceding one. The telegram may start with any value out of this queue, but the bytes have to appear in the right order. Any other value or the event of a bigger delay being detected, stops the Boot ID detection and starts the application.

Download Header

The Download Header has to follow the last byte of the Boot ID, the A3H, and starts with an ASCII *SOH* = 01H. It is followed by the start and target addresses of the download program and its length as 16-bit values of which the low bytes are transmitted first. The Download Header terminates with a 16-bit Check Sum, low byte transmitted first, which is the 2's complement of the sum, calculated from *SOH* up to and including *Data Length*. The sum of all bytes in the Download Header (including the *Check Sum*) should be zero.

Table 5–2: Download Header

Field	Size	Meaning
SOH	8 bit	start of Download Header
Start Address	16 bit	Boot Loader jumps to this address after download.
Target Address	16 bit	Boot Loader writes data to this address.
Data Length	16 bit	Number of data bytes transmitted after Download Header (*Check Sum* excluded).
Check Sum	16 bit	16-bit Check Sum from SOH to data length

Download Data

As many data bytes as defined in *Data Length* are expected, plus a 16-bit *Check Sum*, low byte transmitted first, which is the 2's complement of the sum, calculated over all data. The sum of all bytes (including the *Check Sum*) should be zero.

5.3.6.2. CAN Protocol and Timing

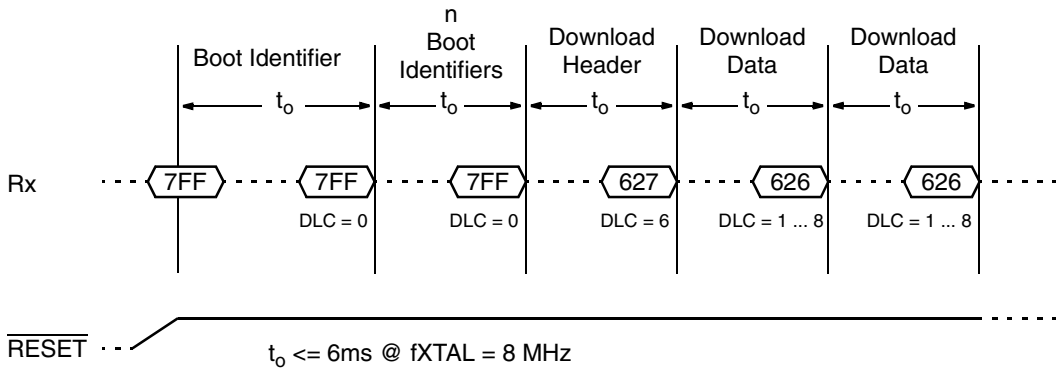


Fig. 5–9: CAN Timing

The Boot Loader expects CAN telegrams in standard format (11 bit identifier) at a bit rate of 125kBd at 8 MHz processor clock with control parameters set as follows: BPR = 7, TSEG1 = 3, TSEG2 = 2, SJW = 3, rx not inverted.

Boot Identifier (ID = 7FFH, DLC = 0)

After reset the Boot Loader expects the Boot Identifier within 6ms (@ fXTAL = 8 MHz), what consists out of a telegram with ID = 7FFH and DLC = 0. **As the transmitter is not active, at least one additional working CAN node has to work correctly on the bus, to generate the CAN bus acknowledge bits!** If the Boot ID can not be detected in time, the Boot Loader starts the application.

Download Header (ID = 627H, DLC = 6)

The Download Header telegram has to follow within 6 ms (@ fXTAL = 8 MHz) after the Boot ID. It contains the start and target addresses of the download program and its length as 16-bit values with low bytes first.

Table 5–3: Download Header

Field	Size	Meaning
Start Address	16 bit	Boot Loader jumps to this address after download.
Target Address	16 bit	Boot Loader writes data to this address.
Data Length	16 bit	Number of data bytes transmitted after Download Header (*Check Sum* excluded).

Download Data (ID = 626H, DLC = 1 ... 8)

As many data bytes as defined in *Data Length* are expected, plus a 16-bit *Check Sum*, low byte transmitted first, which is the 2's complement of the sum, calculated over all data. The sum of all bytes (including the *Check Sum*) should be zero.

6. Core Logic

6.1. Control Register CR

The Control Register CR serves to configure the ways, by which certain system resources are accessed during operation. The main purpose is to obtain a variable system configuration during IC test.

Upon each HIGH transition on the RESETQ pin internal hardware reads data from address location 00FFF3h and stores it to the CR. The state of the TEST and ESTOPCLK pins at this point in time, specifies which program storage source is accessed for this read.

Table 6–1: Control byte source

TEST	Control byte source
0 or NC	internal BOOT ROM (standard for stand-alone operation)
1	external via Multifunction pins in Bus mode (for test purposes only)

The system will thus start up according to the configuration defined in address location 00FFF3h, automatically copied to register CR.

CR **Control Register**

	7	6	5	4	3	2	1	0	
r/w	RESLNG	TSTTOG	x	MFM	TSTROM	IROM	IRAM	ICPU	ROM
r/w	RESLNG	TSTTOG	EBTRI	MFM	FLASH	IROM	IRAM	ICPU	Emu
	Value of 00FFF3h								Res

RESLNG **Reset Pulse Length**
r/w1: Pulse length is $4095/F_{XTAL}$
r/w0: Pulse length is $16/F_{XTAL}$
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 **TEST Pin Toggle** (Tables 6–2 and 6–3)
This bit is used for test purposes only. If TSTTOG is true in IC active mode, pin TEST can toggle the multifunction pins between Bus mode and normal mode.

EBTRI **Emulator Data Bus Tristate** (Table 6–3)

MFM **Multifunction pin Mode**
(Tables 6–2 and 6–3)

Table 6–2: TSTTOG and MFM usage in mask ROM parts

TSTTOG	MFM	TEST pin	Multifunction Pins
0	0	x	Bus mode
1	0	0	Bus mode
		1	normal mode
x	1	x	normal mode

Table 6–3: TSTTOG, EBTRI and MFM usage in Flash and EMU parts

TST-TOG	EBTRI	MFM	TEST pin	Multi-function Pins	Emula-tor Bus Pins
0	x	0	x	Bus mode	Flash mode
1	x	0	0	Bus mode	Flash mode
			1	normal mode	
x	0	1	x	normal mode	Eulator mode
	1				Flash mode

TSTROM **TestROM** (Table 6–4)

FLASH **FLASH EEPROM** (Table 6–5)

IROM **Internal ROM** (Tables 6–4 and 6–5)

Table 6–4: TSTROM and IROM usage in mask ROM parts

TSTROM	IROM	selected program storage
1	1	internal ROM
0		internal TestROM
x	0	external via Multifunction pins in Bus mode

Table 6–5: FLASH and IROM usage in FLASH and EMU parts

FLASH	IROM	selected program storage
1	1	internal FLASH EEPROM resp. Emulator Bus
0		internal BOOT ROM
x	0	external via Multifunction pins in Bus mode

IRAM **Internal RAM**
r/w1: Enable internal RAM.
r/w0: Disable internal RAM.

ICPU **Internal CPU**
r/w1: Enable internal CPU.
r/w0: Disable internal CPU.

Table 6–6: Some commonly used settings for address location 00FFF3h. A copy is automatically transferred to the CR during IC start-up.

Code	TEST Pin	Operation Mode
FFh	0	Stand-alone with internal ROM or Flash
ABh	1	External program storage connected to Multifunction pins in Bus mode
DFh	0	Emulator mode (CPGA177 package)

6.2. Reset Logic

6.2.1. Alarm Function

An 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 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 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. 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 alarm and reset (see Fig. 6–1 on page 51).

6.2.2. Internal Reset Sources

This IC contains three internal circuits that are able to generate a system reset: watchdog, supply supervision and clock supervision.

All three 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 prolongation counter starts, as soon as no internal reset source is active any more. It counts 4096 F_{XTAL} periods (for alternative settings refer to register CR) and then resets the reset request flag, thus releasing the RESETQ pin.

As long as the reset input comparator on the pin RESETQ detects the low level, all IC registers are held in reset state.

6.2.2.1. Supply Supervision

An internal bandgap reference voltage is compared to V_{DD}. A V_{DD} level below the Supply Supervision threshold VREF-POR will permanently pull the pin RESETQ low and thus hold the IC in reset (see Fig. 6–2 on page 52). With HW Option FFB8h, bit6 = 0, this reset source can be enabled/disabled by flag CMA in register CSW0 (see Section 6.2.2.2. on page 50).

6.2.2.2. Clock Supervision

The Clock Supervision monitors the frequency at the oscillator input XTAL1. A frequency level 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. 6–2 on page 52). With HW Option FFB8h, bit6 = 0, this reset source can be enabled/disabled by flag CMA in register CSW0.

A frequency exceeding the specified IC frequency is not detected.

There are two general operation options which can be selected in the HW Options field (address FFB8h):

1. Bit6 = 1:
Clock and Supply Supervision are permanently active. They can not be deactivated. The Watchdog must be serviced by SW. This mode is recommended for all stand-alone applications requiring high operational reliability.

2. Bit6 = 0:
Clock and Supply Supervision are active after reset, but can be enabled/disabled by the clock monitor active flag CMA of register CSW0. The Watchdog must be serviced only after a first write access to register CSW1. This mode is recommended for test and evaluation purposes only.

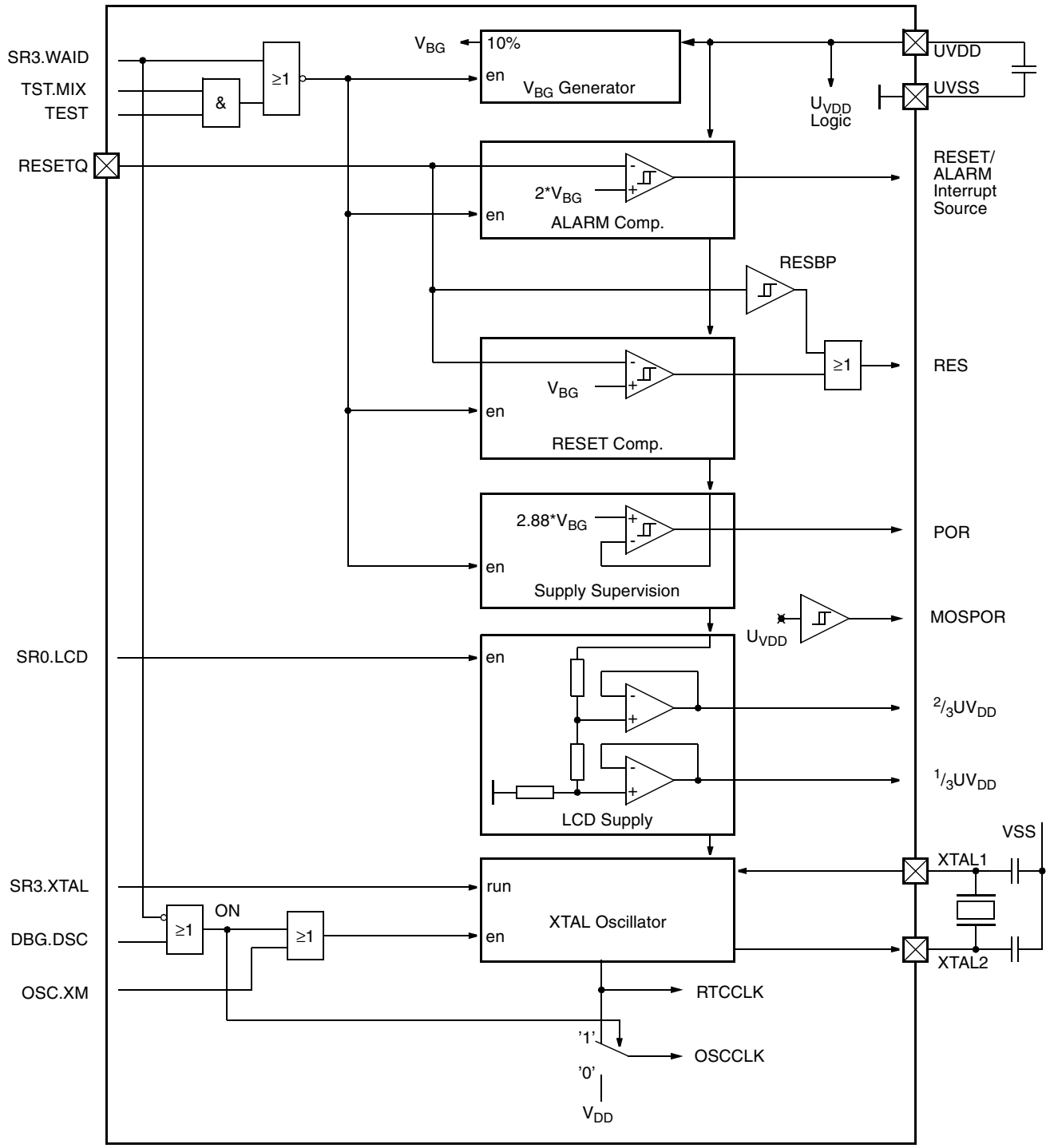


Fig. 6–1: UVDD Section

6.2.2.3. RESET Comparator

During power saving mode, the comparator function is not available and is bypassed by a simple CMOS Schmitt input. Full CMOS levels are thus required on this input in this mode.

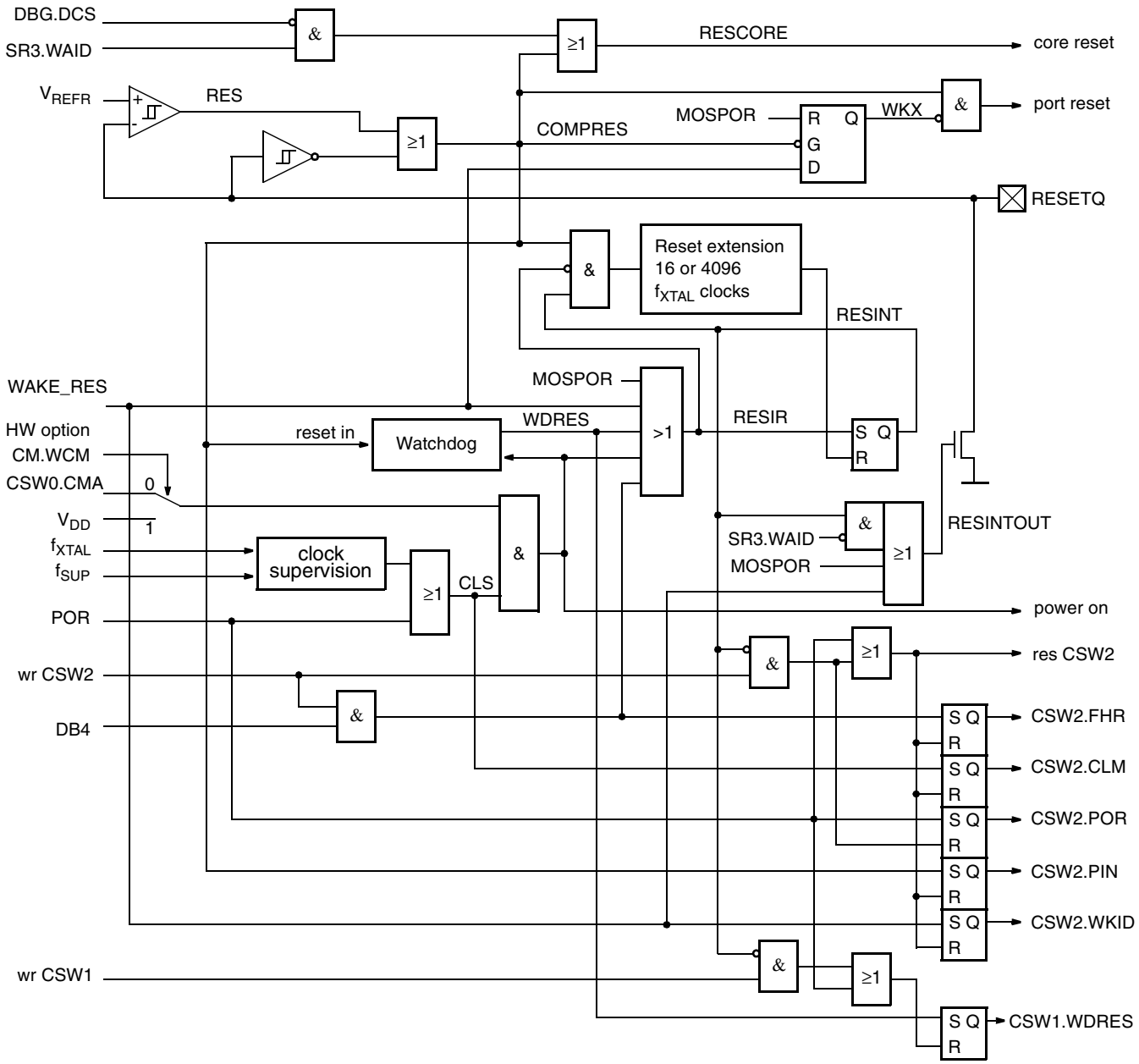


Fig. 6–2: Reset Logic Block Diagram

6.2.2.4. Watchdog

The Watchdog module serves to monitor undisturbed program execution. A failure of the program to retrigger the Watchdog within a preselectable time will pull the pin RESETQ low and thus reset the IC (Fig. 6–2 and 6–3). With HW Option FFB8h, bit6 = 0, this reset source is only enabled after a write access to register CSW1 (see Section 6.2.2.2. on page 50).

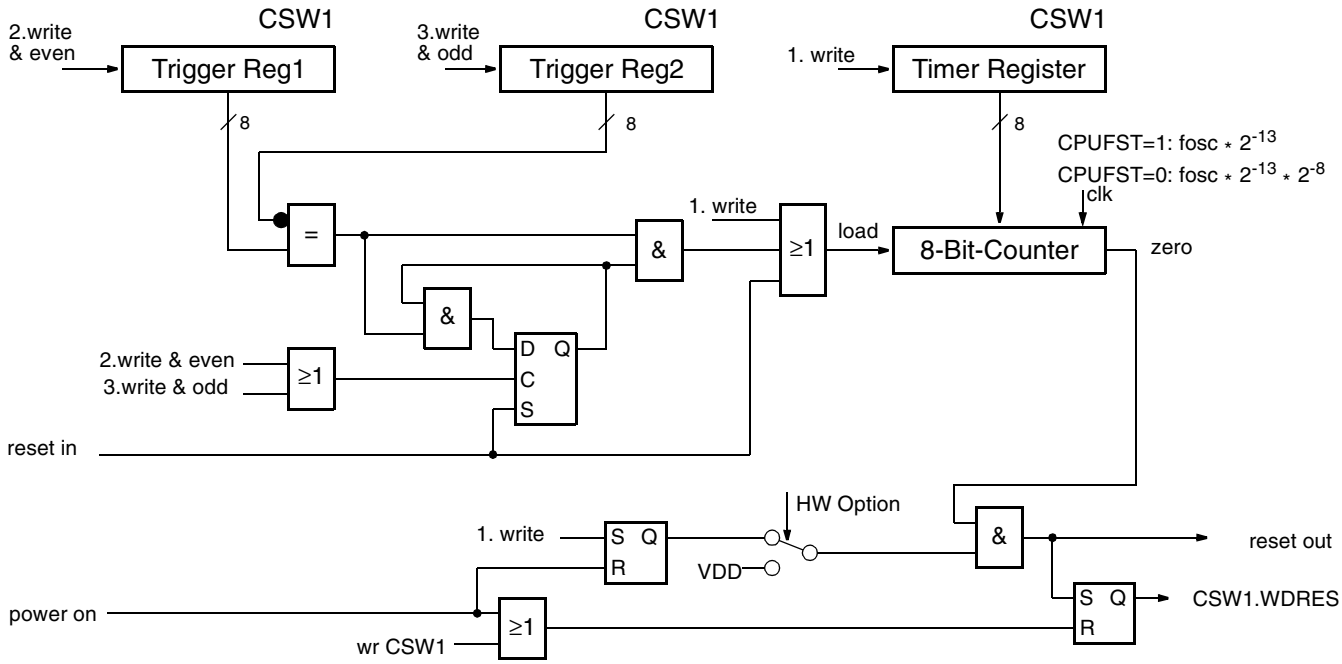


Fig. 6–3: Watchdog Block Diagram

The Watchdog contains a down-counter that generates a reset when it wraps from zero to FFh. It is reloaded with the content of the watchdog timer register, when, on a write access to register CSW1, watchdog trigger registers 1 and 2 contain bit complemented values. An IC reset resets the watchdog timer register to FFh, thus forcing the Watchdog to create a maximum reset interval.

The Watchdog is controlled by register CSW1. The first write access to it loads the timer register value setting the Watchdog's unretriggered reset interval. The desired interval can be programmed by setting the CSW1 value to:

$$\text{Value} = \frac{\text{Interval} \times f_{CPU}}{8192} - 1$$

The resolution of the Watchdog is $8192/f_{CPU}$.

The second and all following even-numbered write accesses load watchdog trigger register 1, the third and all following odd numbered write accesses load watchdog trigger register 2.

In all future, the SW has to write alternatingly to register CSW1 value and bit complement value, thus retriggering the up-counter. Failure to retrigger will result in an overflow of the up-counter generating a Watchdog reset.

It is not allowed to change a chosen value. Writing a wrong value to CSW1 immediately sets the flag WDRES in register CSW1 and prohibits further retriggering of the watchdog counter.

WDRES is true after a Watchdog reset. Only a Supply Supervision reset or a write access to register CSW1 clears it.

6.2.3. External Reset Sources

As long as the reset input comparator on the pin RESETQ detects the low level, the overall IC is reset. 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.

6.2.4. Summary of Module Reset States

After reset the IC modules are set to the reset state (Fig. 6–7)

Table 6–7: 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	Depends on mask option. EMU option: Switched off. SW activation is possible. Stand-alone option: Permanently active.
Clock monitor	Depends on mask option. EMU option: Active. SW may toggle. Stand-alone option: Permanently active.

6.2.5. Reset Registers

CSW0 Clock, Supply & Watchdog Register 0

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

This register controls the Supply and Clock Supervision modules.

CMA Clock and Supply Monitor Active

w1: Both Enabled.
w0: Both Disabled.

CSW1 Clock, Supply & Watchdog Register 1

	7	6	5	4	3	2	1	0	
r	x	x	x	x	x	x	x	WDRES	
w	Watchdog Time and Trigger Value								
	1	1	1	1	1	1	1	1	Res

This register controls the Watchdog module. Only values between 1 and 255 are allowed.

WDRES Watchdog Reset Source
r1: Watchdog was reset source.
Any write access to CSW1 resets this flag.

First write the desired watchdog time value to this register. On further writes, to retrigger the Watchdog, alternatingly write a value (not necessarily the former time value) and its bit complemented value. Using other values or changing the order of both values will cause the watchdog to generate a RESET.

Table 6–8: Source of last Hardware Reset

*)CSW2.WKID	CSW2.FHR	*)CSW2.CLM	*)CSW2.PIN	*)CSW2.POR	CSW1.WDRES	Reset Source
0	0	0	1	0	0	external from RESETQ pin
0	0	0	1	0	1	internal Watchdog
0	0	1	1	0	0	internal Clock Supervision
0	0	1	1	1	0	internal Supply Supervision
0	1	0	1	0	0	internal Forced Hardware
1	x	x	1	x	x	wake-up from WAKE/IDLE

The registers sum up the source of all HW resets that occurred since the last write to register CSW2.
*)Any write access to CSW2 resets these flags to 0.

The RTC can not be reset.

CSW2 Clock, Supply & Watchdog Register 2

	7	6	5	4	3	2	1	0	
r	TST	x	WKID	FHR	CLM	PIN	POR	x	0
w	x	x	0	FHR	0	0	0	x	0
	-	x	-	-	-	-	-	-	Res

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

WKID **Wake Reset from WAKE/IDLE Mode** (Table 6–8)

FHR **Forced Hardware Reset**
r0: (Table 6–8)
r1: (Table 6–8)
w1: force Reset
w0: no action

CLM **Clock Supervision Reset** (Table 6–8)

PIN **RESETQ Pin Reset** (Table 6–8)

POR **Supply Supervision Reset** (Table 6–8)

6.3. Standby Registers

The Standby Registers SR0, SR1, SR2 and SR3 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 about enabling and disabling procedures and the standby state refer to the specific module descriptions.

The minimum IC current consumption is obtained with SR0, SR1, SR2 and SR3 set to 00h.

SR0 Standby Register 0

	7	6	5	4	3	2	1	0	
r/w	SM	PWM1	PWM0	UART2	SPI1	CAN0	CCC	SPI0	
	0	0	0	0	0	0	0	0	Res

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

PWM1 **Pulse Width Modulator 1**
r/w1: Module active.
r/w0: Module off.

PWM0 **Pulse Width Modulator 0**
r/w1: Module active.
r/w0: Module off.

UART2 **UART 2**
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.

CCC **Capture Compare Counter**
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	
r/w	LCD	CPUFST	PSLW	UART0	ADC	P0DIN	TIM1	ERM	
	0	1	0	0	0	0	0	0	Res

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

CPUFST **CPU FAST Mode**
r/w1: FAST mode: $F_{CPU} = F_{XTAL}$
r/w0: SLOW mode: $F_{CPU} = F_{XTAL} / 256$

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.

P0DIN **Port 0 Digital Input**
r/w1: Enable digital inputs on analog ports P0.1 to P0.5
r/w0: Disable digital inputs on analog ports P0.1 to P0.5.

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

ERM **EMI Reduction Module**
r/w1: Module active.
r/w0: Module off.

SR2 Standby Register 2

	7	6	5	4	3	2	1	0	
r/w	TIM2	PWM3	PWM2	UART1	PWM4	DGB	EXTIR	ABM	
	0	0	0	0	0	0	0	0	Res

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

PWM3 **Pulse Width Modulator 3**
r/w1: Module active.
r/w0: Module off.

PWM2 **Pulse Width Modulator 2**
r/w1: Module active.
r/w0: Module off.

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

PWM4 **Pulse Width Modulator 4**
r/w1: Module active.
r/w0: Module off.

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

EXTIR **External Interrupt**
r/w1: Enable functions IRQ on pin U5.4 and ABORTQ on pin U5.5
r/w0: Disable functions IRQ on pin U5.4 and ABORTQ on pin U5.5

ABM **Alternative Banking Mode**
r/w1: Alternative Banking mode enabled
r/w0: Native Banking mode enabled.

SR3 Standby Register 3

	7	6	5	4	3	2	1	0	
r/w	x	x	x	XTAL	WAID	FCLO	CAN2	CAN1	
	x	x	x	1	0*)	0	0	0	Res

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

WAID **WAKE/IDLE**
(Reset with COMPRES || MOSPOR)
r/w 0 : WAKE/IDLE mode off
r/w 1: WAKE/IDLE mode on

FCLO **Fast Clock off**
r/w 0 : Fast clock on
r/w 1: Fast clock off

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

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

6.4. Test Registers

Test registers are for factory 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

7. Multiplier

The Multiplier provides the following function:

- Calculation of 8 bits x 8 bits = 16 bits

Features

– needs no wait states

– no status signals necessary to be checked separately by the application program

– for the application program the result is immediately available after writing into the multiplier register

– the registers for multiplicand and multiplier are not only writable but also readable

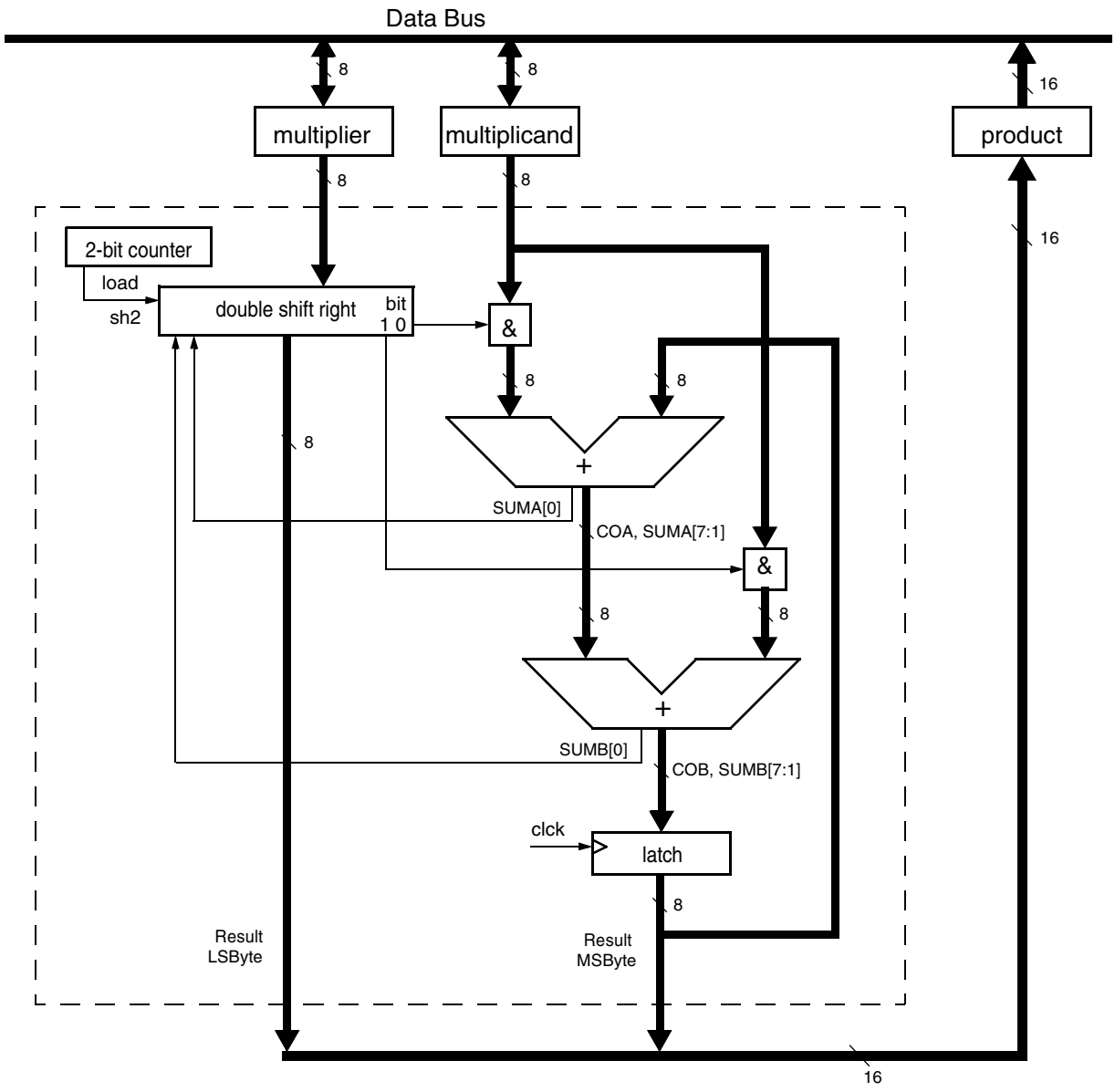


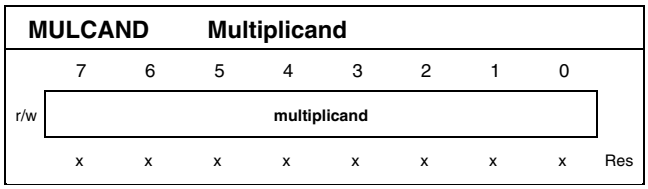
Fig. 7–1: Block diagram of the multiplier

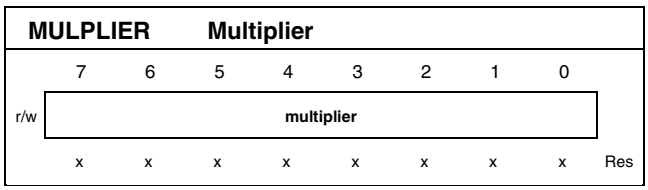
7.1. Functional Description

To obtain a 16-bit product, the application program first has to write the first factor into the 8-bit multiplicand register and then the second factor into the multiplier register. Writing the multiplier starts the multiplication.
The execution of the multiplication takes 4 CPU clock (PH2) cycles. CPU instructions with absolute addressing modes take at least 4 cycles to access the product register. According to this, immediately after writing into the multiplier register, the result is available for the application program.

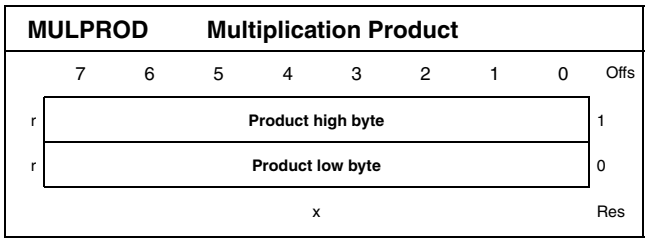
7.2. Registers

Two 8-bit registers serve as input and a 16-bit register delivers the result:





Writing into the Multiplier register starts the execution of the multiplication.



The Product register MULPROD holds the result of the multiplication after 4 CPU clock (PH2) cycles have elapsed.

7.3. Operation of the Multiplier

Since the execution of a multiplication takes less CPU cycles than an application program needs to access the result register, multiplications are done by simply writing the multiplicand followed by the multiplier and reading the product.

Precautions have to be taken in application programs using the multiplier, and in application programs which may be interrupted by a service routine also using the multiplier. If the procedure of writing the multiplicand and multiplier until reading the result register is interrupted, and the interrupt service routine overwrites the multiplier input registers, the result read by the background program is wrong.
To solve this problem, the multiplier input registers have to be saved before and restored after a multiplication takes place in the interrupt service routine.

8. Power-Saving Module (PSM)

To reduce the power consumption one of two Power-Saving Modes (WAKE and IDLE) can be selected. Non-power-saving modes are DEEP SLOW, SLOW and FAST, which differ from Power-Saving Modes by having the CPU running instead of a stopped CPU clock during WAKE/IDLE active. Most of the core logic is switched off in a Power-Saving Mode. Only hardware necessary for WAKE/IDLE is supplied.

Features

- Power reduction down to leakage current of wake mode possible
- Real-Time Clock (RTC) Module
- 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

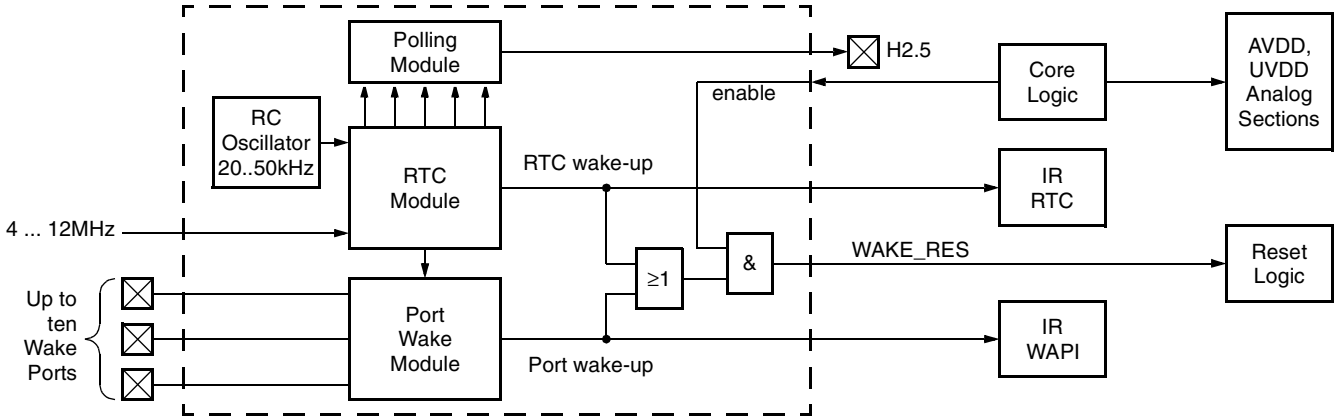


Fig. 8–1: Power-Saving Module

The major task of the Power-Saving Module is to supply a wake-up signal (WAKE_RES) for the main system or to generate an interrupt, as shown in figure 8–1. WAKE_RES is necessary to get the IC out of a Power-Saving Mode. Apart from WAKE_RES, a Power-Saving Mode can only be left by activating RESETQ at pin or by a power on reset. WAKE_RES is generated by a Port Wake Module for an event-driven wake-up, combined with an RTC Module for a cyclic wake-up.

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

The RTC Module has to provide the time of the day accurately down to a second, and generate an output signal, which can be used to trigger an interrupt or a wake-up signal. The RTC Module can be clocked by the on-chip quartz oscillator or by the Power-Saving Module built-in RC oscillator.

The Polling Module cyclically outputs a high pulse of programmable duration at port H2.5. 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.

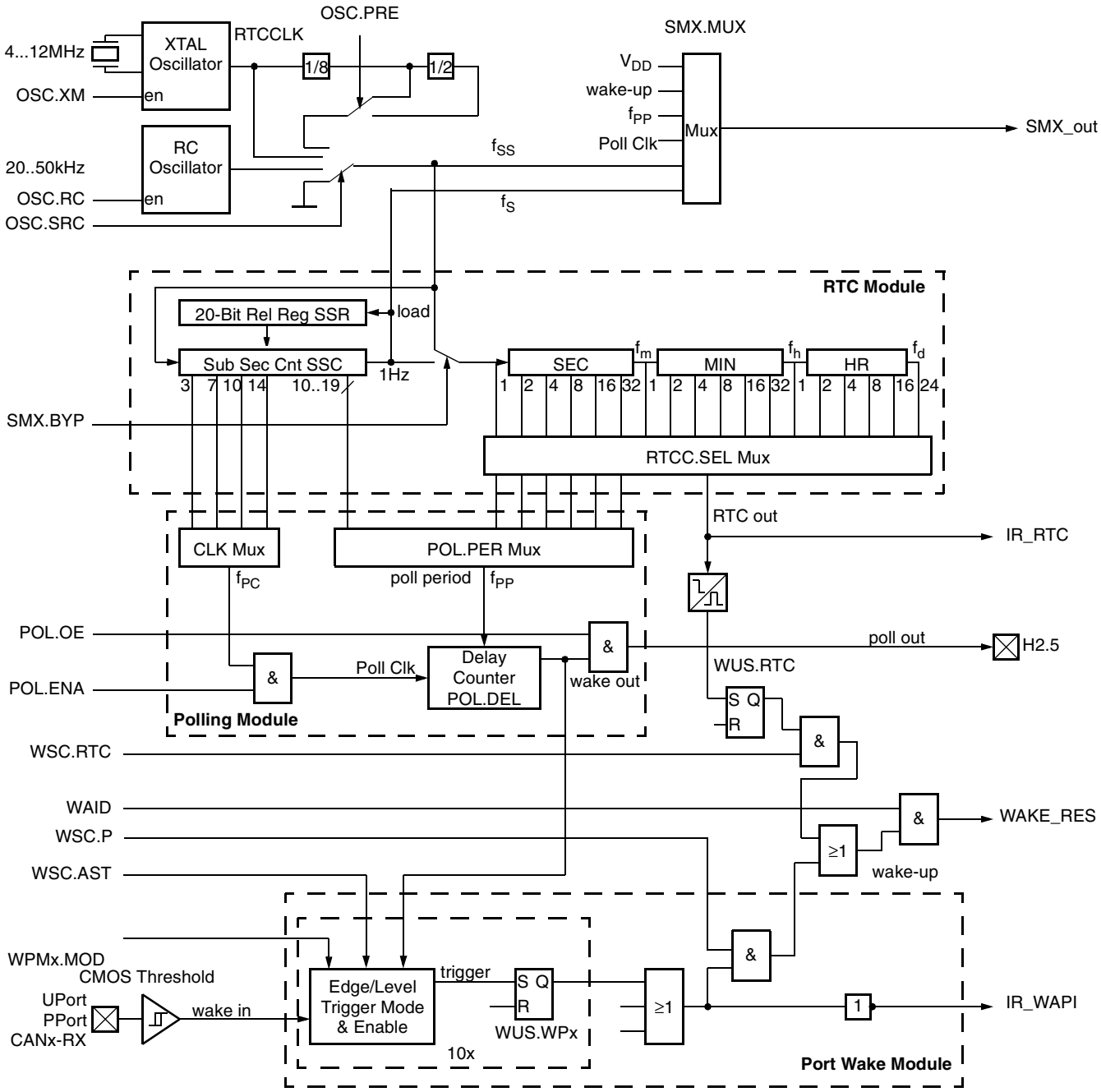


Fig. 8–2: Power-Saving Module Block Diagram

8.1. Functional Description

The power-saving logic combines all wake-up sources. 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 reset. The Polling Module drives an H-Port and generates a strobe signal to enable a Wake Port to trigger on a dedicated input level. Several internal clock signals can be output via CO1 (SMX_out).

8.1.1. RTC Module

To work as real-time counter, a sub-second counter (SSC) with its reload register (SSR) and a seconds, minutes and hours counter are part of the Power-Saving Module. As input

for the sub second counter (f_{SS}), either a clock of the quartz oscillator divided by 8 or 16 or that of the module built-in RC oscillator can be selected. The SSR has to be programmed with a value that yields a one Hertz beat f_S as output signal of the SSC to clock the seconds. f_S is the reload signal for SSC as well as the input clock to the seconds counter (RTC.SEC). An underrun of the down-counting SSC triggers the seconds counter to count up. 60 seconds trigger a minutes up-count (RTC.MIN), followed by the hours (RTC.HR).

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

As opposed to other internal clocks, the RTC can not be stopped (during emulation) by ESTOPCLK.

8.1.2. Polling Module

The polling logic periodically activates the output signal Wake Out which can be enabled by SW to drive port H2.5 (Poll Out). The rising edge of the Polling Period input (f_{PP}) defines the start and the Polling Clock (f_{PC}). Together with the delay counter, it defines the duration of the high time. This can be used to cyclically flash a LED or drive external circuitry.

The falling edge of the Wake Out signal is used to scan the input levels of that 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 8–6 for timing details about 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 control unit is designed for a polling period to be set equal to or greater than four times the polling delay.

8.1.3. 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 sets the corresponding wake source flag if necessary. Instead of the strobe signal, WSC.AST may be used, e.g., if no RTC subsystem (with Polling Logic) is 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 8–2 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 8–9 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.

8.2. Registers

OSC **Oscillator Source Register**

	7	6	5	4	3	2	1	0	Offs
r/w	RC	XK	XM	x	LD	PRE	SRC		0
	1	1	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 **4 ... 12MHz XTAL**
r/w1: always enabled
r/w0: disabled during power-saving mode

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

PRE **4 ... 12MHz XTAL / 8 or / 16**
r/w1: 4 ... 12MHz XTAL / 16
r/w0: 4 ... 12MHz XTAL / 8

SRC **Oscillator Source Select**
r/w0: 4 ... 12MHZ XTAL (divided by 8 or 16)
r/w1: Don't use, factory test only
r/w2: RC oscillator
r/w3: Ground
With OSC.LD set, writing to SRC selects a new oscillator source immediately. When OSC.LD is not set, a new oscillator source does not get valid before the next reload of the SSC. Consider that a read access returns the current source select, not a possibly requested one by a write with OSC.LD not set!

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, please refer to tables 8–1 to 8–3. The values 0 and 1 are not allowed. To avoid programming values not expected, never write single bytes of the SSR on their own, but always all 3 bytes without an interrupt by SSC read. This is necessary, as for reading SSC the register

hardware (master/slave) uses the same buffer registers as for writing SSR. Single bytes could still be filled with intermediate SSC values of a previous read. Writing SSR does not load the SSC immediately. This will be done automatically together with the next reload of the SSC. It can be forced immediately by setting OSC.LD. A new value does not get valid before the bus cycle has been written into its register. Therefore please wait for one f_{IO} cycle between write and read access, e. g., to verify a value just programmed.

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 saves MIN and SEC in the corresponding counters. This mechanism allows consistent read and write access. Since read and write use the same latches, don't mix these access types. A new value gets valid not before the following bus cycle has been written into its register. Therefore, please wait for one f_{IO} cycle between write and read access, e. g., to verify a value just programmed.

HR **Hours Counter**
r/w0 to 23:

MIN **Minutes Counter**
r/w0 to 59:

SEC **Seconds 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 8–4)

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, which will become valid when writing byte 1, 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 8–5)
Trigger mode for Wake Port WPx+y. For assignment of Wake Port and mode field please refer to table 8–9.

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 8–7)

PER **Select Polling Period** (Table 8–6)

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. 8–6).

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 on								Res

AST **Alternative to Strobe**
r/w1: Alternative 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 8–8)
Defines a signal which is output as SMX_out.

Table 8–1: SSR values for f_s = 1Hz with XTAL and XTAL/8 (Maximum adjusted resolution)

XTAL [MHz]	XTAL/8 [KHz]	Period [us]	Reload Value	Resolution [±ppm]	Per Day [±ms]
4	500	2.00	500000	1.00	86.40
5	652	1.60	625000	0.80	69.12
6	750	1.33	750000	0.67	57.60
8	1000	1.00	1000000	0.50	43.20

Table 8–2: SSR values for f_s = 1Hz with XTAL and XTAL/16 (Maximum adjusted resolution)

XTAL [MHz]	XTAL/16 [KHz]	Period [us]	Reload Value	Resolution [±ppm]	Per Day [±ms]
4	250	4.00	250000	2.00	172.80
5	313	3.20	312500	1.60	138.24
6	375	2.67	375000	1.33	115.20
8	500	2.00	500000	1.00	86.40
10	625	1.60	625000	0.80	69.12
12	750	1.33	750000	0.67	57.60

Table 8–3: SSR values for f_s = 1Hz with RC (Maximum adjusted resolution)

RC [KHz]	Period [us]	Reload Value	Resolution [±ppm]	Per Day [±ms]
20	50.00	20000	25.00	2160
32,768	30.52	32768	15.26	1318
50	20.00	50000	10.00	864

Table 8–4: SEL Usage

RTCC. 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 8–5: MOD Usage

WPMx. MODy			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) not in WAKE mode			

Table 8–6: PER Usage

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

Table 8–7: CLK Usage

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

Table 8–8: MUX Usage

SMX. MUX	Name	
0	V_{DD}	
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}	
7	Poll Clk	

Table 8–9: Wake Ports

Name	Basic Funct.	WPMx	WPMx.MODy	Special Functions
WP0	U3.4	0	0	T0-OUT / SEG3.4
WP1	U6.6		1	CAN0-RX / PINT1-OUT / SEG6.6
WP2	U1.6	2	0	CAN1-RX / SEG1.6
WP3	U6.0		1	PINT0-IN/LCD-SYNC_OUT/SEG6.0
WP4	U6.1	4	0	PINT1-IN/LCD-CLK_OUT/SEG6.1
WP5	U6.2		1	PINT2-IN/T1-OUT/ SEG6.2
WP6	U5.6	6	0	PINT3-IN/PWM2/ SEG5.6
WP7	U4.0		1	CAN2-RX/SEG4.0
WP8	U4.4	8	0	UART0-RX/SEG4.4
WP9	H2.4		1	PWM0

8.3. 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 CSW2 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.

8.3.1. Configuration of Wake Sources

8.3.1.1. Port Wake Module

If an external event-driven wake-up is necessary, the Port Wake Module has to be configured according to section 8.6. 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 8.4. and 8.5.

8.3.1.2. RTC Module

If a cyclic wake-up is necessary, the RTC Module has to be configured according to section 8.4. 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.

8.3.2. Configuration of Interrupts

During CPU active modes, the RTC Module and the Port Wake Module can be operated as interrupt sources. The interrupts have to be configured according to section “Interrupt Controller (IR)”.

8.3.3. WAKE/IDLE

With setting SR3.WAID (mode = WAKE/IDLE) a core reset signal is generated immediately.

A wake-up signal sets WAKE_RES to one, immediately pulling pin RESETQ to low. This sets SR1.CPUFST and disables the WAKE_RES output of the Power-Saving Module. When V_{DD} and PH2 are detected as stable, the signal CLS is cleared and the reset extension is started. After the reset extension has finished, the pin RESETQ is released. The Core reset signal gets inactive and CSW2.WKID is set.

The CPU starts execution at the reset vector address and can read the reset/wake-up reason in registers CSW1 and CSW2. The wake-up source can be read in register WUS and should be cleared thereafter, otherwise Wake Port interrupts are not possible.

8.3.4. Precautions

The SW has to guarantee the ability of wake-up. In the best case, a stable initialization and configuration of the wake logic is executed immediately before a power-saving mode is activated.

The necessary Wake Ports have to be configured as inputs.

Enabling wake-up by pin only is dangerous. Inadvertently or accidentally entering a power-saving mode with no wake source enabled, e. g. by a software bug, or if a flag is modified by electrical overstress (EOS) in a power-saving mode, a status may be reached which can only be terminated by a manually generated reset with low level at pin RESETQ or with a power on reset. Neither the watchdog nor the clock supervision or any other internal reset source terminates a power-saving mode.

Because neither the VBG generator nor the RESET comparator are enabled during power-saving mode, proper CMOS input levels (V_{il}=UV$_{SS}$±0.3 V and V_{ih}=UV$_{DD}$±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 with CMOS input levels (V_{il}=xV$_{SS}$±0.3 V and V_{ih}=xV$_{DD}$±0.3 V) applied to all ports - analog inputs via P-ports P0.1 to P0.9 excepted - not only the 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 (see Section 8.4.3. on page 70).

8.3.5. Debug Register

To allow debugging during an active power-saving mode, e.g., read or change the contents of the RAM, or read or change the contents of I/O or processor registers, the system clock must not be switched off as done normally when a Power-Saving Mode is switched on. By setting DBG.DCS the CPU keeps on running in a power-saving mode.

DBG **Debug Register**

	7	6	5	4	3	2	1	0	Offs
r/w	x	x	x	x	x	x	x	DCS	0
	x	x	x	x	x	x	x	0	POR

DCS **DISABLE CPU STOP**

r/w1: Disabled CPU stop (clock off) during power-saving mode (debugging)

r/w0: Stop CPU during power-saving modes (standard, no debugging)

8.3.6. Timing

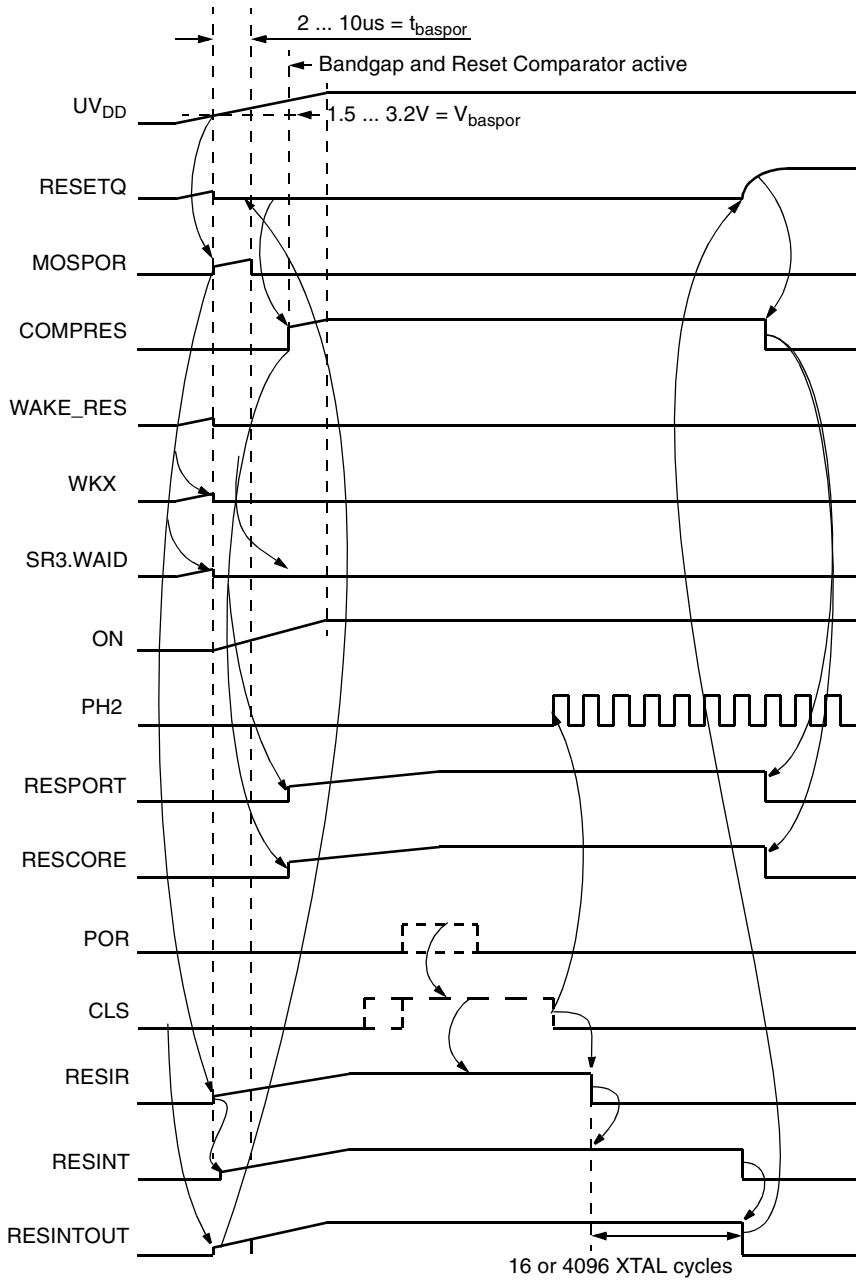


Fig. 8–3: Power-on Reset

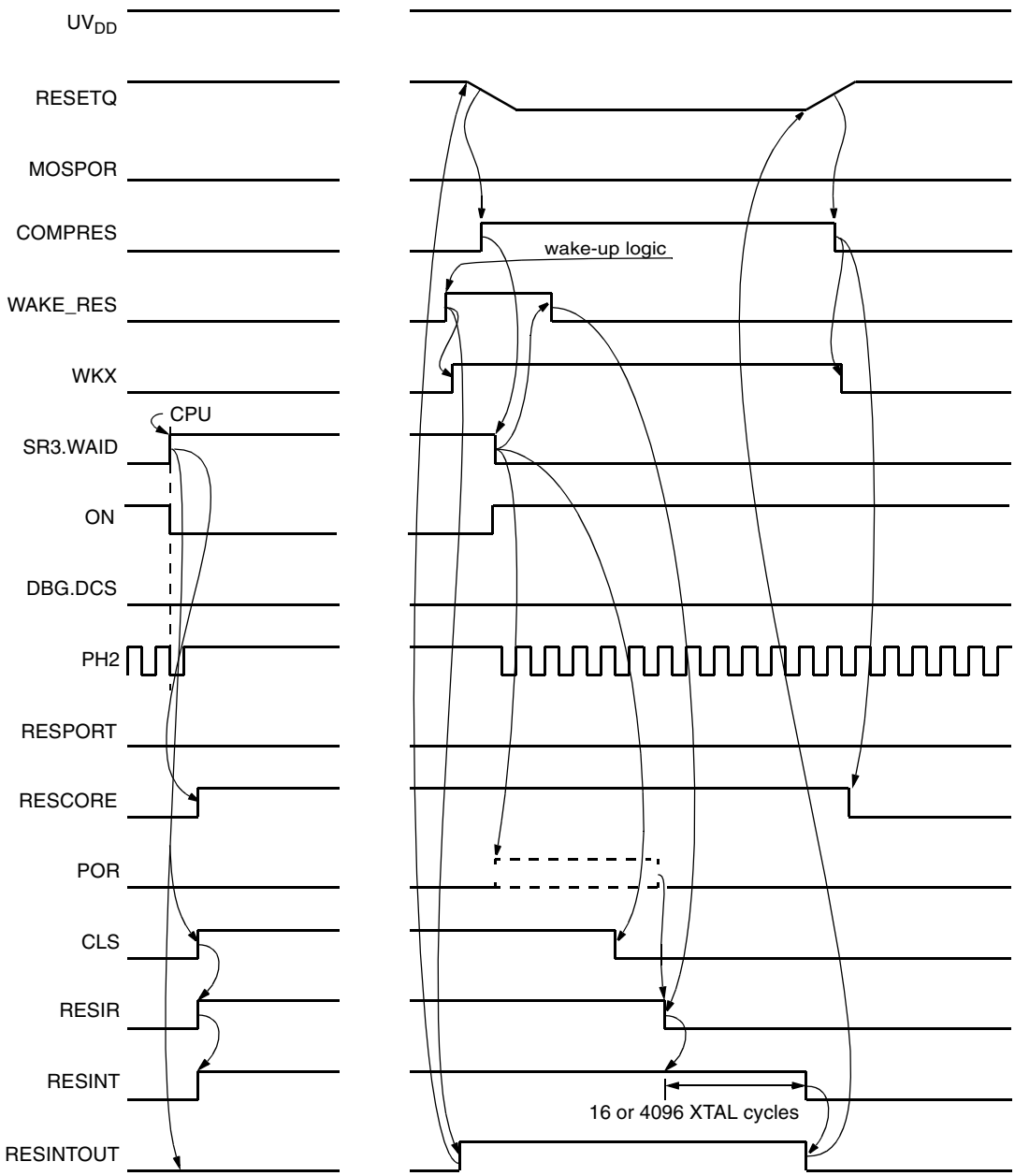


Fig. 8–4: Switching to WAKEIDLE with DBG.DCS inactive and Wake-up

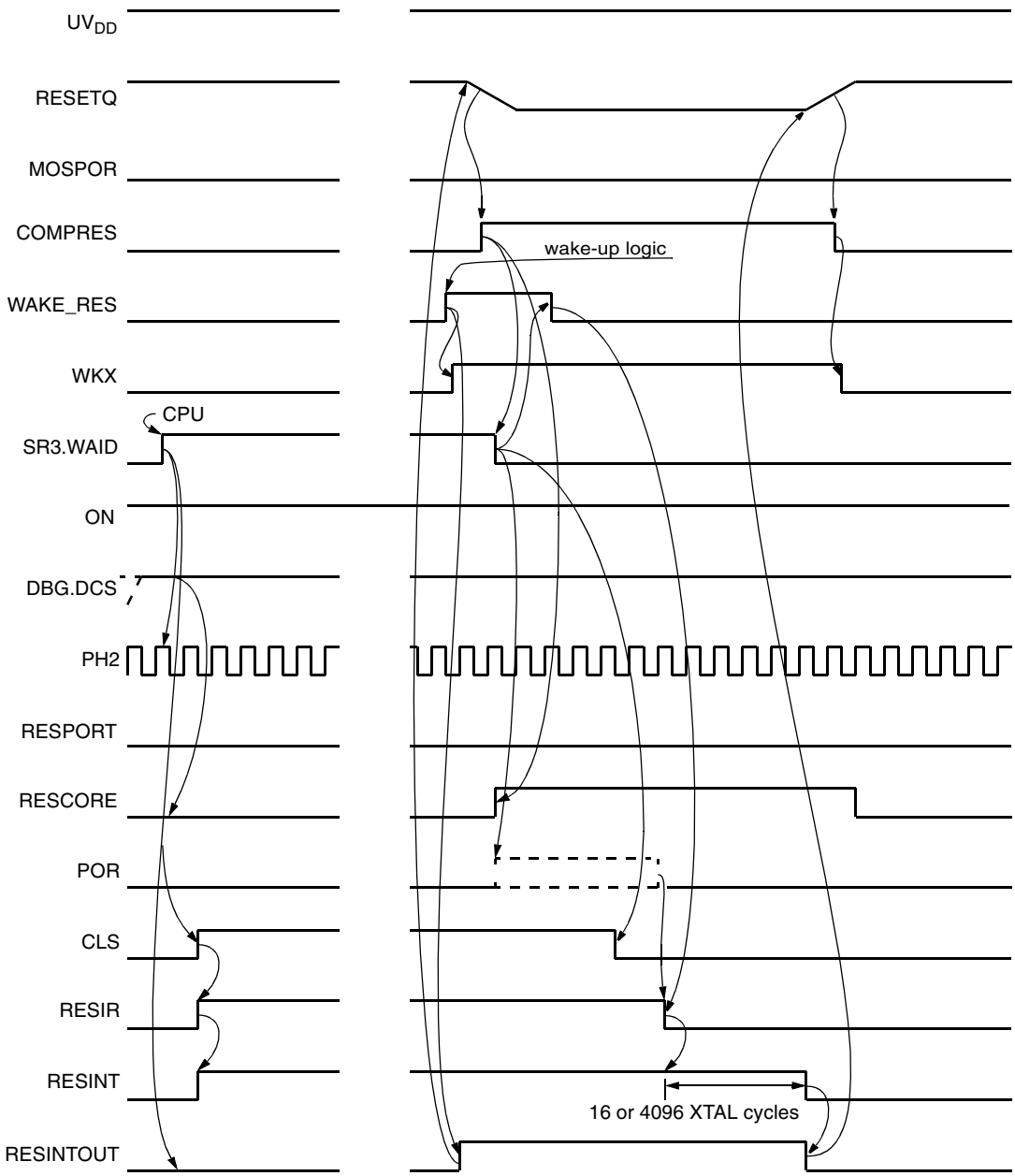


Fig. 8–5: Switching to WAKEIDLE with DBG.DCS active and Wake-up

8.4. Operation of RTC Module

8.4.1. 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. Therefore, the oscillator not required has to be switched off after every reset. Furthermore, the whole logic has to be initialized after a power-on reset.

8.4.2. Oscillator Source and Sub Second Counter

The Sub Second Counter SSC generates a 1 Hz output signal at underflow (0x000000 to 0xFFFFFF). 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.

On three occasions it is necessary to change SSR and OSC.SRC:

– Starting the SSC for the first time (after power-on).
  As 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 (RTC-ISR). Write the new reload value to the SSR register. Make sure that this will be completed before the next underflow of the SSC which happens to each second and simultaneously loads SSC with the new SSR value and switches the oscillator source.
– 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-ISR. Select the desired oscillator source in OSC.SRC and write the corresponding reload value to the SSR register. Make sure that this will be completed before the next underflow of the SSC which happens to each second and simultaneously loads SSC with the new SSR value and switches the oscillator source.

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 (POL.OE = WSC.RTC = WSC.P = 0) and disable RTC and WAPI interrupts.
2. Switch to the new oscillator source.
3. Initialize SSC, RTC and POL.
4. Clear WUS (WUS = 0xFFFF).
5. Enable the output signals again.

8.4.3. Disabling the RTC Module

This has to be done by selecting the RC oscillator (OSC.SRC=2) and disabling this source (OSC.RC=0). Selecting ground (OSC.SRC=3) disables the 32 kHz subsystem too, but this should be avoided to be compatible with future extensions.

8.4.4. 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 ... 12 MHz oscillator, a CPU access to SSC and RTC can be corrupted by a clock pulse. Because this situation can't be avoided, the SSC or the RTC register have to be read twice. If there is a difference between the two accesses 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.

8.4.5. RTC Output Multiplexer

All the taps of the second, minute and hour counters are connected to a multiplexer (Table 8–4) 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.

8.4.6. RTC Interrupt

The IR has to be initialized as described above.

The RTC has its own interrupt vector, thus further investigation of the interrupt source is not necessary. After an interrupt, the flag WUS.RTC is set. The register WUS.RTC is not necessary for the RTC ISR, and does not have to be handled by the RTC ISR.

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

A solution that would not affect the intended interrupts 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.

In this situation, the RTC interrupt may safely be temporarily disabled.

8.4.7. Signal Multiplexer

Various 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 8–8. 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 to measure the quartz frequency and calculate 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. Applications normally keep SMX.BYP cleared.

8.5. Operation of Polling Module

8.5.1. Reset

The whole logic is cleared by every kind of reset, even wake-up from power-saving mode resets the Polling Module. This means that the logic has to be initialized after every reset.

8.5.2. Initialization and Start

The Polling Module needs the RTC Module 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 8.4. for RTC Module initialization.

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

The port H2.5 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.

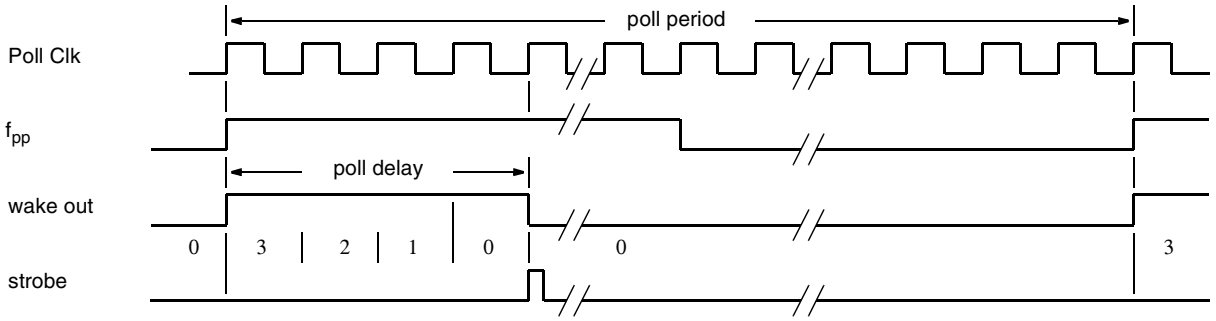


Fig. 8–6: Polling Timing

8.5.3. Stop

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

8.5.4. Restart

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

8.6. Operation of Port Wake Module

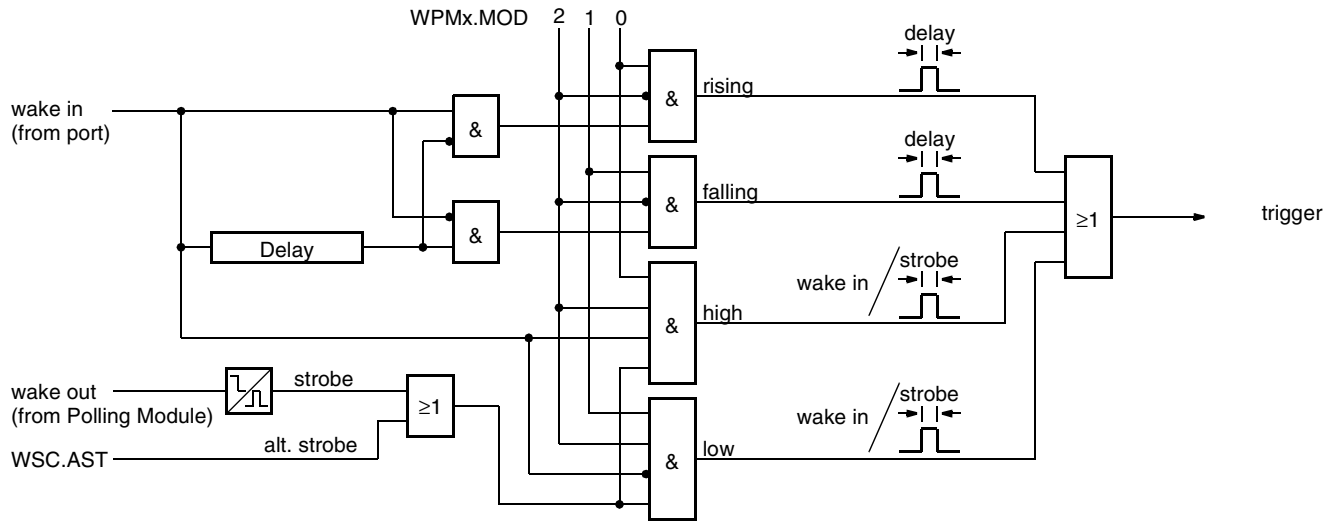


Fig. 8–7: Edge/Level Trigger Logic

8.6.1. Reset

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

8.6.2. Initialization

The corresponding ports must be configured as inputs.

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 which must not generate 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 with WSC.AST if a level triggered mode must be used. The whole register WUS has to be cleared.

8.6.3. Operation

The Port Wake Module can be operated by polling. It can generate an interrupt (IR_WAPI) or a wake-up signal to leave the 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 is not to 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 can't 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. Only clearing WSC.AST 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. The selected strobe signal source is valid for all Wake Ports. Mixing of the strobe signals sources (polling and alternative) is not possible.

If only the edge-triggered mode is used, the RTC Module and the Polling Module are not necessary for correct operation of the Port Wake Module.

8.6.4. Wake-up from Power-Saving Mode

Following 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, as otherwise neither wake-up nor Wake Port interrupt via IR_WAPI is possible.

8.6.5. Wake Port Interrupt

Following the initialization described above, the IR has to be initialized.

All Wake Ports are directed to an interrupt vector. After an interrupt the source can be read in register WUS. It should be cleared after reading, as otherwise no further Wake Port interrupts via 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.

9. Memory Patch Module

The Memory Patch Module allows the user to modify up to ten hardwired ROM locations by external means. This function is useful if faulty parts of software or data are detected after the ROM code has been cast into mask ROM.

Software loads addresses and the corrected code, e.g., from external non-volatile memory into the respective registers of the module. The module will then replace faulty code upon address match.

Single ROM locations are directly replaced. Longer faulty sequences may be repaired by introducing a jump to a new subroutine in RAM (e.g. opcode JSR requires 3 consecutive bytes to be patched). The RAM subroutine then may consist of any number of instructions, ending with a return to the next correct instruction in ROM. Thus it is possible to also include complex software modules.

ICs which are derived from the Emulator IC may have less patch cells. In this case the upper patch cells are not available.

Features

– patching of read data from up to 10 different ROM locations (24-bit physical address)

– automatic insertion of 1 CPU wait state for each patched access

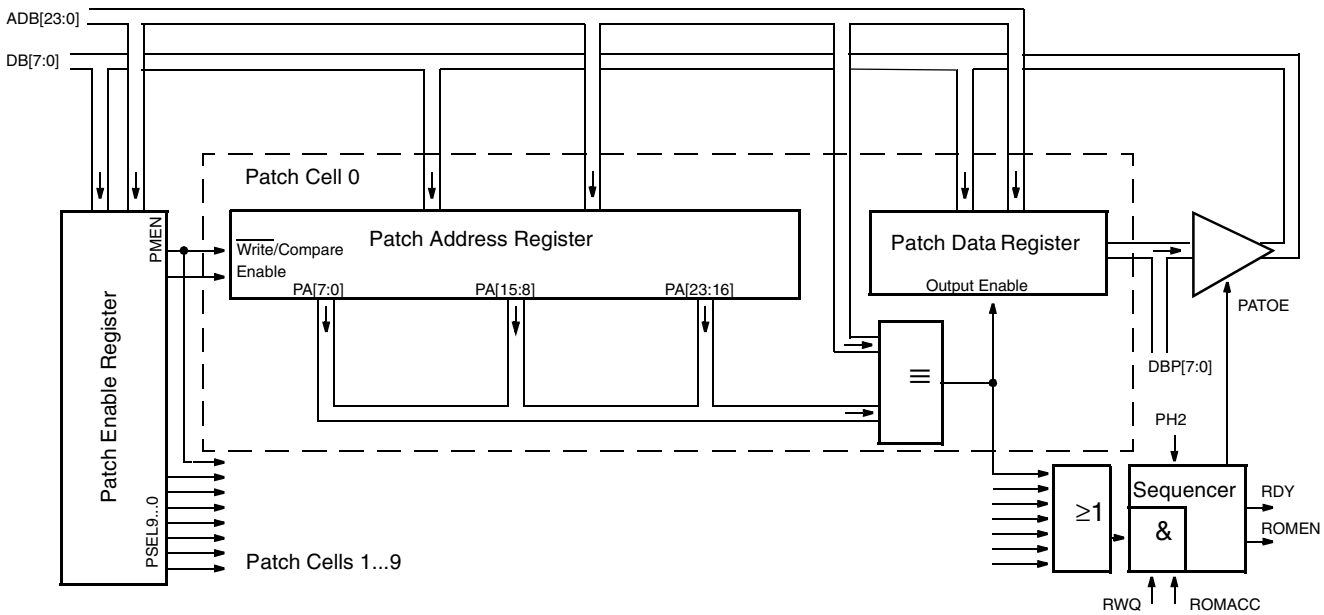


Fig. 9–1: Block Diagram

9.1. Principle of operation

9.1.1. General Remarks

The logic contains ten patch cells (see Fig. 9–1 on page 73), each consisting of a 24-bit compare register (Patch Address Register, PARn), a 24-bit address comparator, a Patch Enable Register (PERn) bit and an 8-bit Patch Data Register (PDR).

The current address information for a ROM access is fed to a bank of ten patch cells. In case of a match in one patch cell, and provided that the corresponding Patch Enable Register bit is set, a wait cycle for CPU is included by pulling down the RDY input of CPU for one cycle (see Fig. 9–2 on page 74). In the meantime, 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.

9.1.2. Initialization

After reset, as bit PER0.PMEN is reset to 0, all patch cell registers are in write mode and patch operation is disabled.

To initialize a patch cell, first set the corresponding PSEL bit in register PER0 or PER1 as a pointer. Then enter the 24-bit address to registers PAR2 (high byte), PAR1 (middle byte) and PAR0 (low byte) and the desired patch code to register PDR.

If desired, repeat the above sequence for other patch cells. Only set one PSEL pointer bit at a time in registers PER0 and PER1.

9.1.3. Patch Operation

To activate a number of properly initialized patch cells for ROM code patching, set all the corresponding PSEL bits in registers PER1, then PER0, setting bit PER0.PMEN to 1.

The Memory Patch Module will immediately start comparing the current address with 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.

9.1.4. Reconfiguration

To reconfigure the Memory Patch Module, first set PER0.PMEN to 0. The module will immediately terminate patch operation.

Then proceed as described in “Initialization” on page 73.

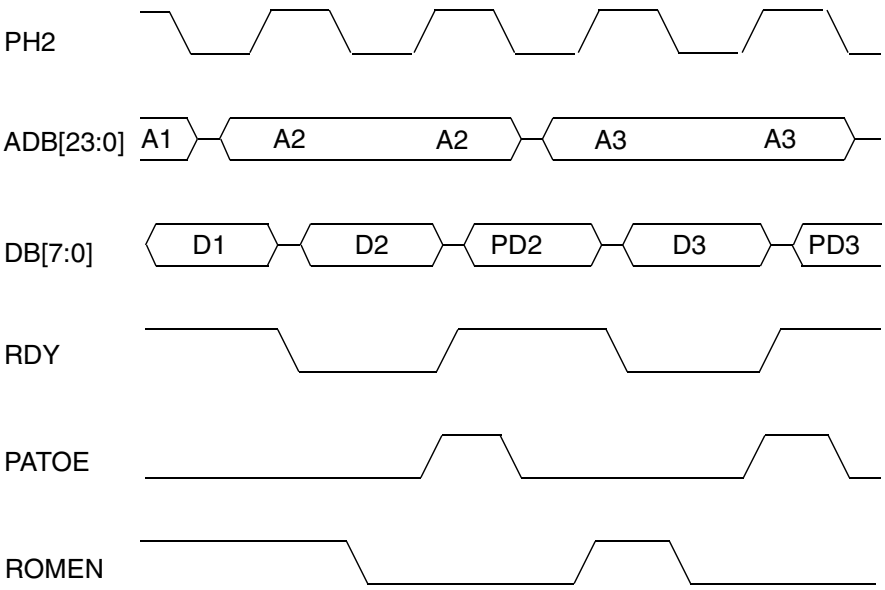


Fig. 9–2: Timing

9.2. Registers

PAR0 Patch Address Register 0

	7	6	5	4	3	2	1	0	Note
w	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	
	1	1	1	1	1	1	1	1	Res

PAR2 Patch Address Register 2

	7	6	5	4	3	2	1	0	Note
w	PA23	PA22	PA21	PA20	PA19	PA18	PA17	PA16	
	1	1	1	1	1	1	1	1	Res

PAR1 Patch Address Register 1

	7	6	5	4	3	2	1	0	Note
w	PA15	PA14	PA13	PA12	PA11	PA10	PA9	PA8	
	1	1	1	1	1	1	1	1	Res

PDR Patch Data Register

	7	6	5	4	3	2	1	0	Note
w	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	
	0	0	0	0	0	0	0	0	Res

PER0 Patch Enable Register 0

	7	6	5	4	3	2	1	0	Note
w	PSEL6	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	PSEL0	PMEN	
	0	0	0	0	0	0	0	0	Res

PER1 Patch Enable Register 1

	7	6	5	4	3	2	1	0	Note
w	x	x	x	x	x	PSEL9	PSEL8	PSEL7	
	x	x	x	x	x	0	0	0	Res

PA23 to 0 Patch Address
Upon occurrence of this address the patch cell replaces ROM data with data from PDR.

PD7 to 0 Patch Data
Data to replace false ROM data at certain address.

PSEL0 to 9 Select Patch Cell
w1: select cell for write or enable for patch
w0: disable patch cell
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

10. Interrupt Controller (IR)

The Interrupt Controller has 16 input channels. Each input has its own interrupt vector pointing to an interrupt service routine (ISR). One of 15 priority levels can be assigned to each input or the input can be disabled. The Interrupt Controller is connected to the NMI input of the CPU. However, despite the non-maskable interrupt input, it is possible to disable all interrupt sources together in the Interrupt Controller.

Features

- 16 interrupt inputs.
- 16 interrupt vectors.
- 15 individual priority levels.
- Global/individual disable of interrupts.
- Single interrupt service mode.

10.1. Principle of Operation

10.1.1. General Remarks

Interrupt requests are served in the order of their programmed priority level. Interrupt requests of the same priority level are served in descending order of interrupt input number.

Each of the 16 interrupt inputs clears a flag in the interrupt pending register (IRRET and IRP), which can be read by the user. A pending interrupt enables the output of the corresponding priority register (IRPRI10 to IRPRIFE) which is connected to a parallel priority decoder together with the other priority registers. The decoder outputs the highest priority and its input number to a latch. The latched priority is compared with the top entry of the priority stack. The top entry of the priority stack contains the priority of the currently served interrupt. Lower entries contain interrupts with lower priority whose interrupt service routines were started but interrupted by the higher priority interrupts above. If the latched priority is lower or equal than the top of stack priority, nothing happens. If the latched priority is higher than the top of stack priority, a NMI is sent to the CPU and the latched priority is pushed on the stack.

The Interrupt Controller signals an interrupt to the CPU via NMI input . After the current instruction is finished the CPU starts an interrupt sequence. First it puts the program bank register, the program counter high byte, the program counter low byte and the program status register into the stack. Then the CPU writes the vector address low byte (FFFA in 6502 mode, FFEA in 816 mode) to the bus. The Interrupt Controller recognizes this address and stops the CPU by the RDY signal. Now the Interrupt Controller writes the vector address low and high byte of the corresponding interrupt number to the bus and releases the CPU by releasing RDY. The CPU now operates with the new vector of the interrupt service routine.

When the Interrupt Controller writes the new vector to the address bus, the interrupt pending flag of this vector is set, indicating that no interrupt is pending.

The software must pull the top entry from the priority stack at the end of an interrupt service routine. This happens with the write access to the interrupt return register IRRET. Then the next entry (with lower priority) is visible at top of stack and is compared with the priority latch.

The Interrupt Controller and related circuitry is clocked by the CPU clock and participates in CPU FAST and SLOW mode.

10.1.2. Hardware settings

According to Section 10.3. some of the Interrupt Controller inputs allow selection of a source by HW option (cf. Table 10–3). This configuration has to be done prior to operation. Refer to “HW Options” for setting them.

10.1.3. Initialization

After reset, all internal registers are cleared but the Interrupt Controller is active. When an interrupt request arrives, it will be stored in the respective pending register IRP/IRRET. But it will not trigger an interrupt as long as its interrupt priority register IRPRIxy is set to zero.

The interrupt sources in peripheral modules have to be properly SW configurated prior to operation.

Before enabling individual inputs, make sure that no previously received signal on that input has cleared its pending flag which may trigger the Interrupt Controller. Clear all pending interrupts with the flag IRC.CLEAR to avoid such an effect.

10.1.4. Operation

Activation of an interrupt input is done by writing a priority value ranging from 1h to Fh to the respective IRPRIxy register. Upon an interrupt request, pending or fresh, the Interrupt Controller will immediately generate an interrupt.

During operation, changes in the priority register setting may be made to obtain varying interrupt servicing strategies. Flags IRC.DAINT, IRC.DINT and IRC.A1INT allow some variation in the Interrupt Controller response behavior.

10.1.5. Inactivation

There are two possibilities to disable an interrupt within the Interrupt Controller. Changing the priority of an interrupt input to zero disables this interrupt locally. Interrupts are globally disabled by writing a zero to flag IRC.DINT of register IRC.

During the evaluation period (see also Section 10.4.: Interrupt Timing) it is not possible to suppress an interrupt by changing priority.

A zero in the flag IRC.DINT of register IRC prevents the Interrupt Controller from pulling the signal NMI low. However, if this flag is set after the falling edge of NMI, the corresponding interrupt cannot be cancelled.

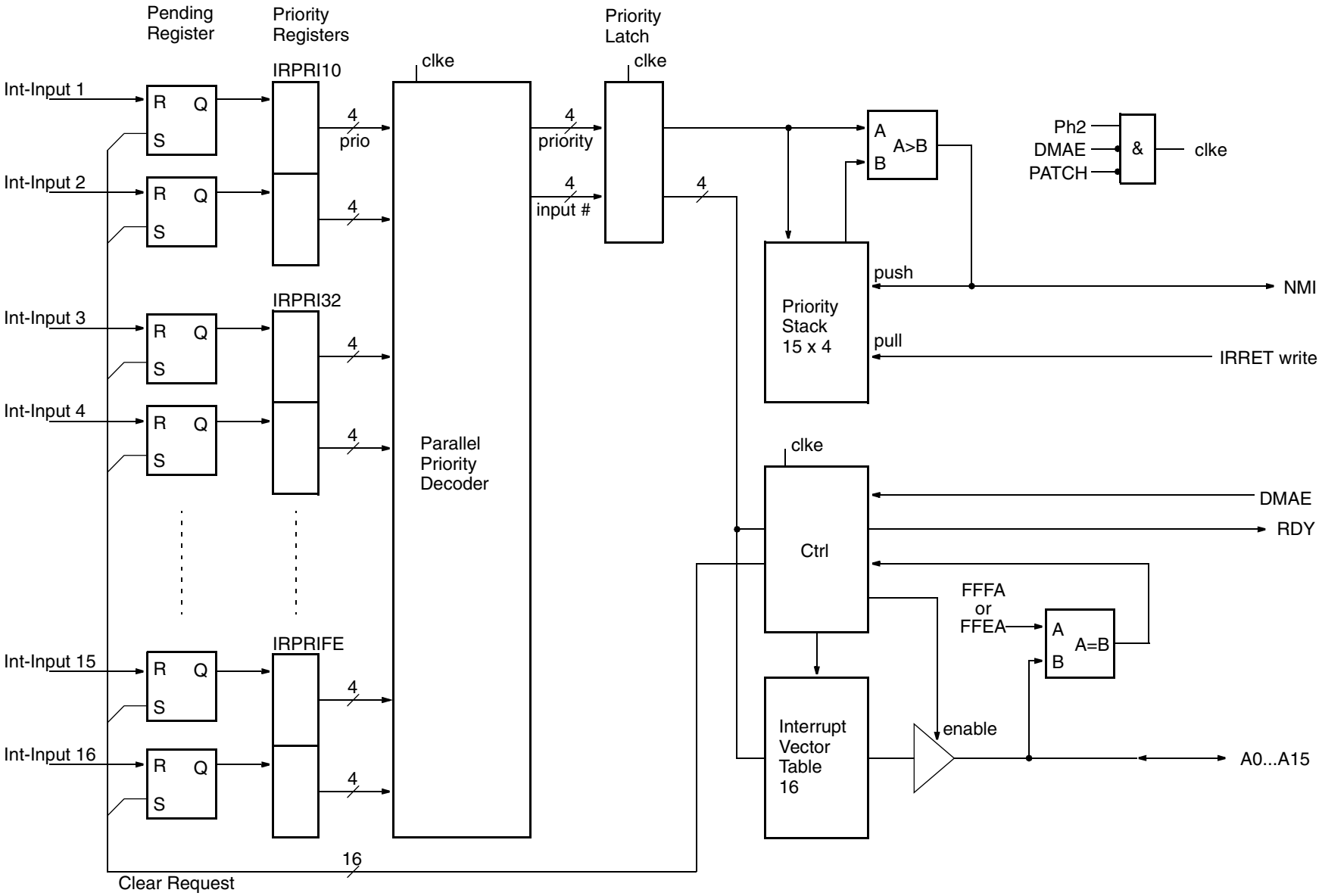


Fig. 10–1: Block Diagram

10.1.6. Precautions

10.1.6.1. Return from Interrupt

The write access to the IRRET must be performed just before the RTI command at the end of the interrupt service routine. After a write access to this location it is guaranteed that the next command (should be RTI) will be processed completely before a new interrupt request is signalled to the CPU. If the RTI command does not immediately follow the write to IRRET, an interrupt with the same priority may be detected before the corresponding RTI is processed. A stack underflow may occur because this may happen several times.

An interrupt with a higher priority than the one actually served, may interrupt between the write access to IRRET and the belonging RTI command. Now an interrupt request from the interrupted low priority interrupt may occur during service of the high priority interrupt. This one will be served after the RTI command of the high priority interrupt and before the RTI command of the first interrupted low priority interrupt. In this case, the return address and the PSW of the same interrupt are stored twice on the stack. This may happen several times and can cause a problem if the stack size is calculated without sufficient buffer or if the interrupt load is too high, which means that there is no time for the background loop.

10.1.6.2. Disable Interrupt

If an opcode fetch of a disable interrupt instruction (DI) happens one clock cycle after the falling edge of NMI (see Section 10.4.1. on page 83), it is possible, that an interrupt service routine (ISR) is active, though the corresponding interrupt is disabled. That is why after disabling an interrupt, and before accessing critical data, at least one uncritical instruction is necessary. This guarantees that the ISR is finished before critical data access and no further ISR can interrupt it.

As it is now possible that an ISR (Interrupt Service Routine) can lengthen the time between the disable interrupt instruction (DI) and the enable interrupt instruction (EI) indefinitely, it is necessary that an ISR first saves registers and enable interrupt flags and then enables interrupts. After interrupt execution enable flags and registers must be restored. This guarantees that other interrupts are not locked out during interrupt execution.

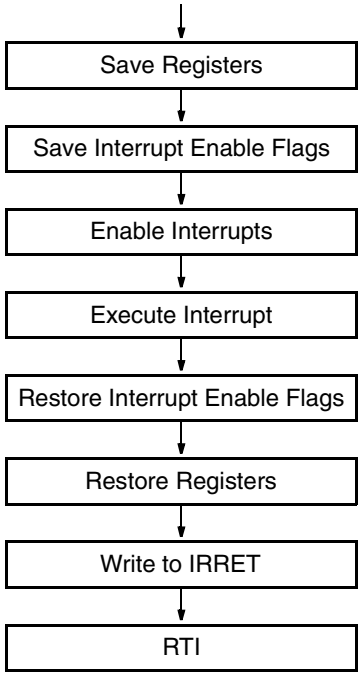


Fig. 10–2: Interrupt Service Routine

10.2. Registers

IRC **Interrupt Control Register**

	7	6	5	4	3	2	1	0
r	x	x	x	x	DAINT	DINT	x	x
w	x	x	x	RESET	DAINT	DINT	A1INT	CLEAR
				x	1	1	x	x

Res

RESET **Reset**
w1: No action.
w0: Momentary reset of the Interrupt Controller, all internal registers are cleared.
The reset of the Interrupt Controller happens with writing zero to this Flag. It is not necessary to write a one to finish the reset.

The standard Interrupt Controller function is performed by setting all flags to one. A hardware reset of the Interrupt Controller is performed by setting RESET low and the other flags to high.

DAINT **Disable after interrupt**
r1: Don't disable after interrupt.
r0: Disable Interrupt Controller after interrupt.
w1: Cancel this feature.
w0: Disable Interrupt Controller after interrupt.
This is the enable flag for the flag A1INT function.

DINT **Disable interrupt**
r1: Interrupts are enabled.
r0: All interrupts are disabled.
w1: Enable interrupts according to priority setting.
w0: Disable all interrupts.

A1INT **Allow one interrupt**
w1: No action.
w0: Serve one interrupt.
This is a momentary signal. With DAINT = 0, only one interrupt (with the highest priority) will be served.

The Flags DAINT and A1INT must be considered in common. They provide the possibility to serve interrupts one by one, only when the main program has enough time (see Table 10–1 on page 79).

CLEAR **Clear all requests**
w1: No action.
w0: Momentarily clears all interrupt requests.

Table 10–1: Single Interrupt Service

DAINT	A1INT	Resulting function
0	1	Disable after current interrupt.
0	0	Serve one interrupt request.
1	x	Normal interrupt mode.

IRE **Interrupts Enable Register**

	7	6	5	4	3	2	1	0	
w	A write access enables interrupts according to priority setting (same effect as setting IRC.DINT)								
	0	0	0	0	0	0	0	0	Res

IRE **Enable Interrupts**

A write access to this memory location enables interrupts according to priority setting (same effect as setting IRC.DINT).

Enabling interrupts using this register take effect not before the execution of the command, following the write access to IRE.

IRRET **Interrupt Return Register**

	7	6	5	4	3	2	1	0	
r	IPF7	IPF6	IPF5	IPF4	IPF3	IPF2	IPF1	IPF0	
w	A write access signals to the Interrupt Controller that the current request has been served								
	0	0	0	0	0	0	0	0	Res

IPF0 to 7 **Interrupt Pending Flag of Input 0 to 7**
r1: No interrupt is pending.
r0: Interrupt is pending.
w: Current request is finished.
For interrupt pending flags 8 to 15, please refer to the description of register IRP.

A write access to this memory location signals to the Interrupt Controller that the current request has been served.

IRPRI10 **Interrupt Priority, Inputs 0 and 1**

	7	6	5	4	3	2	1	0	
r/w	PRIO1				PRIO0				
	0	0	0	0	0	0	0	0	Res

IRPRI32 **Interrupt Priority, Inputs 2 and 3**

	7	6	5	4	3	2	1	0	
r/w	PRIO3				PRIO2				
	0	0	0	0	0	0	0	0	Res

IRPRI54 **Interrupt Priority, Inputs 4 and 5**

	7	6	5	4	3	2	1	0	
r/w	PRIO5				PRIO4				
	0	0	0	0	0	0	0	0	Res

IRPRI76 **Interrupt Priority, Inputs 6 and 7**

	7	6	5	4	3	2	1	0	
r/w	PRIO7				PRIO6				
	0	0	0	0	0	0	0	0	Res

IRPRI98 **Interrupt Priority, Inputs 8 and 9**

	7	6	5	4	3	2	1	0	
r/w	PRIO9				PRIO8				
	0	0	0	0	0	0	0	0	Res

IRPRIBA **Interrupt Priority, Inputs 10 and 11**

	7	6	5	4	3	2	1	0	
r/w	PRIO11				PRIO10				
	0	0	0	0	0	0	0	0	Res

IRPRIDC **Interrupt Priority, Inputs 12 and 13**

	7	6	5	4	3	2	1	0	
r/w	PRIO13				PRIO12				
	0	0	0	0	0	0	0	0	Res

IRPRIFE **Interrupt Priority, Inputs 14 and 15**

	7	6	5	4	3	2	1	0	
r/w	PRIO15				PRIO14				
	0	0	0	0	0	0	0	0	Res

PRIOn **Priority of input number n**
r: Priority of the corresponding interrupt input.
w: Priority of the corresponding interrupt input.
Priority zero prevents the Interrupt Controller from being triggered, but the pending register is not affected. All incoming requests are stored in the pending registers. Of two inputs

with the same PRIO setting, the input with the higher number has priority (see Table 10–2 on page 80).

Table 10–2: PRIOn usage

PRIOn	resulting function
0h	Interrupt input is disabled
1h	Interrupt input is enabled with lowest priority
:	:
Fh	Interrupt input is enabled with highest priority

IRP Interrupt Pending Register

	7	6	5	4	3	2	1	0	
r	IPF15	IPF14	IPF13	IPF12	IPF11	IPF10	IPF9	IPF8	
	0	0	0	0	0	0	0	0	Res

IPF8 to 15 Interrupt Pending Flag of Input 8 to 15
r1: No interrupt is pending.
r0: Interrupt is pending.
For interrupt pending flags 0 to 7, please refer to description of register IRRET.

10.3. Interrupt Assignment

While most interrupt assignments are hard-wired, some are configured via HW option (see Fig. 10–3 on page 82).

Table 10–3: Interrupt assignment

Inter-rupt Input	Interrupt Vector Address	Interrupt Source	HW Option
0	00FFE2-E3	INT-MUX 7	FFC1h
1	00FFE0-E1	INT-MUX 8	FFC1h
2	00FFDE-DF	Timer 0	
3	00FFDC-DD	PINT0	
4	00FFDA-DB	PINT1	
5	00FFD8-D9	INT-MUX 1	FFC0h
6	00FFD6-D7	INT-MUX 2	FFC0h
7	00FFD4-D5	INT-MUX 3	FFC0h
8	00FFD2-D3	CC2OR	
9	00FFD0-D1	CAN 0	
10	00FFCE-CF	INT-MUX 9	FFC2h
11	00FFCC-CD	UART 0 TX/RX	
12	00FFCA-CB	RESET/ALARM	
13	00FFC8-C9	INT-MUX 4	FFC0h
14	00FFC6-C7	INT-MUX 5	FFC1h
15	00FFC4-C5	INT-MUX 6	FFC1h

Table 10–4: INT-MUX 1 = HW Option addr. FFC0H

bit 1	bit 0	selects
0	0	CC0 COMP
0	1	Timer 2
1	0	CAN 2
1	1	Timer 1

Table 10–5: INT-MUX 2 = HW Option addr. FFC0H

bit 3	bit 2	selects
0	0	UART 2
0	1	P06 COMP
1	0	SPI 0
1	1	Timer 1

Table 10–6: INT-MUX 3 = HW Option addr. FFC0H

bit 5	bit 4	selects
0	0	PINT3-IN
0	1	SPI 1
1	0	UART 1
1	1	CC1 COMP

Table 10–7: INT-MUX 4 = HW Option addr. FFC0H

bit 7	bit 6	selects
0	0	CAN 2
0	1	SPI 0
1	0	DMA
1	1	PINT3-IN

Table 10–8: INT-MUX 5 = HW Option addr. FFC1H

bit 1	bit 0	selects
0	0	Timer 2
0	1	UART 1
1	0	SPI 1
1	1	DMA

Table 10–9: INT-MUX 6= HW Option addr. FFC1H

bit 3	bit 2	selects
0	0	Timer 2
0	1	DIGITbus
1	0	UART 2
1	1	PINT2-IN

Table 10–10: INT-MUX 7 = HW Option addr. FFC1H

bit 5	bit 4	selects
0	0	CC0OR
0	1	UART 1
1	0	IR-RTC
1	1	IR-WAPI

Table 10–11: INT-MUX 8 = HW Option addr. FFC1H

bit 7	bit 6	selects
0	0	CC1OR
0	1	PINT2-IN
1	0	IR-RTC
1	1	IR-WAPI

Table 10–12: INT-MUX 9= HW Option addr. FFC2H

bit 1	bit 0	selects
0	0	CAN 1
0	1	UART 1
1	0	IR-RTC
1	1	IR-WAPI

10.3.1. Interrupt Multiplexer

Ten interrupt inputs are directly connected to the respective module's interrupt output. Six interrupt inputs, 4 to 6 and 13 to 15, allow source selection via multiplexers. The multiplexers are configured by HW Option. Please refer to section HW Options for details.

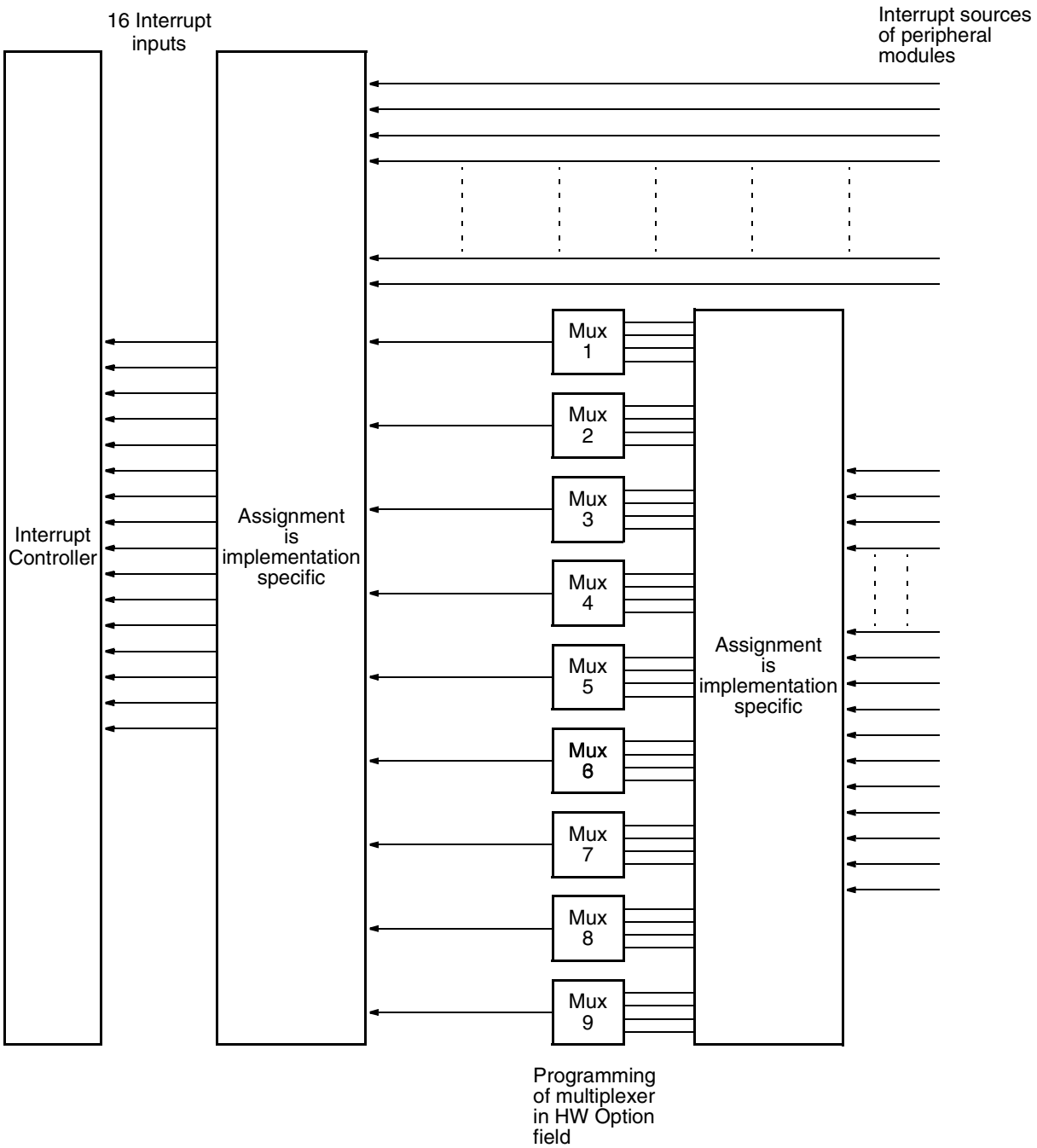


Fig. 10–3: Interrupt Assignment and Multiplexer

10.4. Interrupt Timing

10.4.1. Interrupt Response Time

The interrupt response time is calculated from the interrupt event up to the first interrupt vector on the address bus.

After an interrupt event, the Interrupt Controller starts evaluation with the first falling edge of Phi2. Evaluation requires one clock cycle until the Interrupt Controller pulls the signal $\overline{\text{NMI}}$ low.

After the falling edge of $\overline{\text{NMI}}$ the CPU finishes the actual command. If the falling edge of $\overline{\text{NMI}}$ occurs one clock cycle before an opcode fetch, the following command will be finished too. Then PC and status will be saved on stack before the low byte of the interrupt vector is written to the address bus.

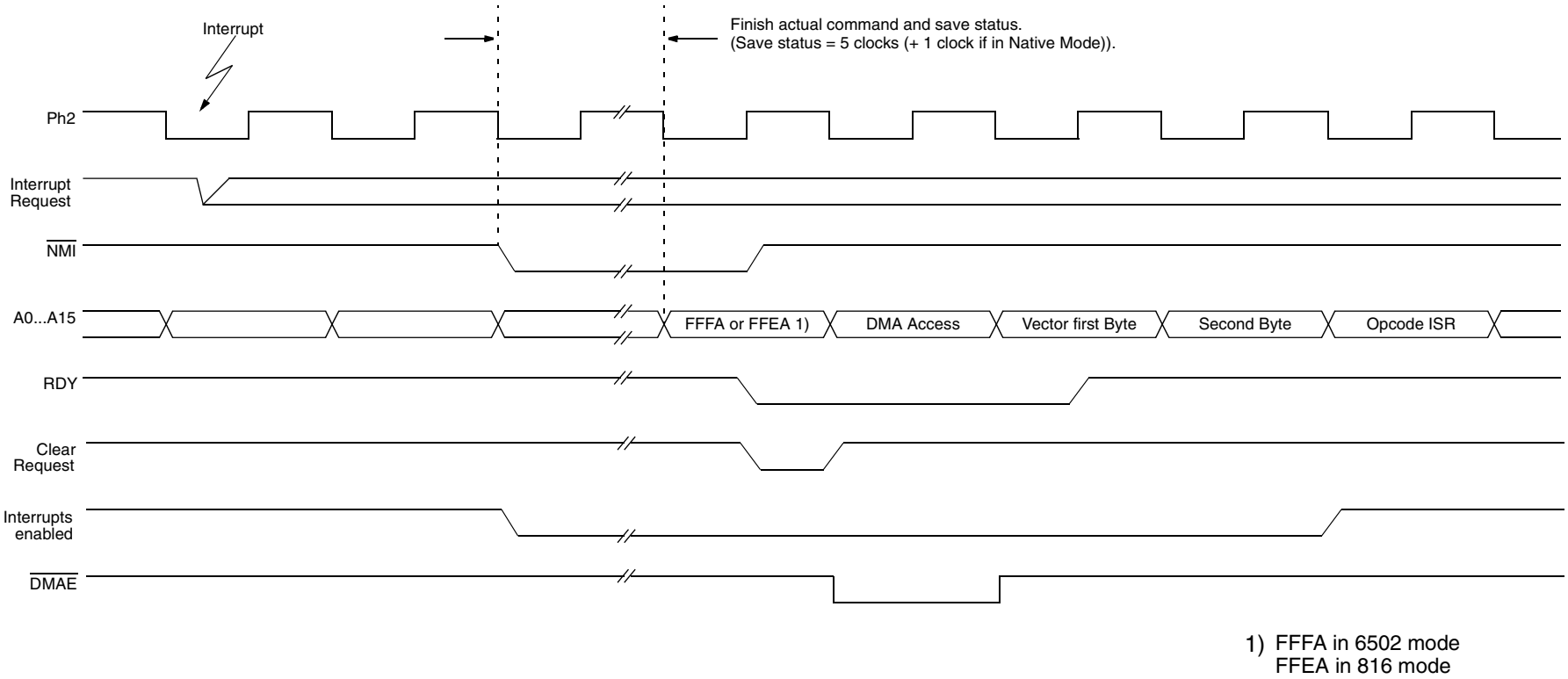


Fig. 10–4: Timing Diagram

10.5. Port Interrupt Module

Port interrupts are the interface of the Interrupt Controller to the external world. Four U-Port pins are connected to the module via their special input lines. Port interrupt 0 and 1 can scale down the interrupt load by prescalers. Port interrupt 2 and 3 share the interrupt input with signals from other sources. HW Option programmable multiplexers define which signal is actually connected to the Interrupt Controller.

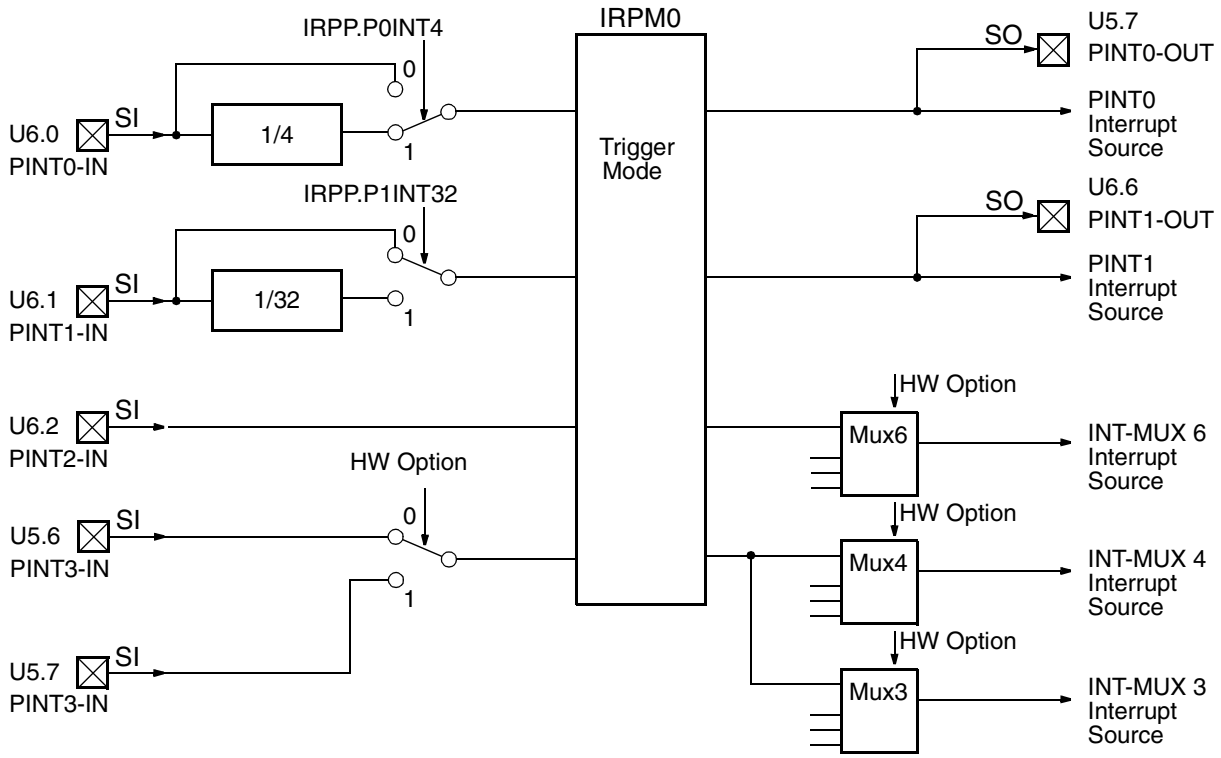


Fig. 10–5: Port Interrupts

The user can define the trigger mode for each port interrupt by the interrupt port mode register.

The Port interrupt prescaler can be switched by the interrupt port prescaler register. The pulse duty factor of the prescaler output is 50%.

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 10–6 and 10–7 for port prescaler active (P1INT32 or P0INT4 = 1).

Table 10–13: Module-specific settings

Module Name	HW Options		Initialization	
	Item	Address	Item	Setting
PINT0			PINT0-IN	U6.0 special in
			PINT0-OUT	U5.7 special out
PINT1			PINT1-IN	U6.1 special in
			PINT1-OUT	U6.6 special out
PINT2	Interrupt multiplexer 6	FFC1h	PINT2-IN	U6.2 special in
PINT3	Input pin selection	FFC2h	PINT3-IN	U5.6 or U5.7 special in
	Interrupt multiplexer 4	FFC0h		
	Interrupt multiplexer 3	FFC0h		

IRPM0	Interrupt Port Mode Register 0							
	7	6	5	4	3	2	1	0
w	Pit3		Pit2		PIT1		PIT0	
	0	0	0	0	0	0	0	0 Res

PITn Port interrupt trigger number n
This field defines the trigger behavior of the associated port interrupt (Table 10–14).

IRPP	Interrupt Port Prescaler Register							
	7	6	5	4	3	2	1	0
w	x	x	x	x	x	x	P1INT32	P0INT4
							0	0 Res

Table 10–14: PITn usage

PITn	Trigger Mode
0h	Interrupt source is disabled
1h	Rising edge
2h	Falling edge
3h	Rising and falling edges

P1INT32 Port 1 interrupt prescaler
w1: Indirect mode, 1:32 prescaler
w0: Direct mode, bypass prescaler

P0INT4 Port 0 interrupt prescaler
w1: Indirect mode, 1:4 prescaler
w0: Direct mode, bypass prescaler

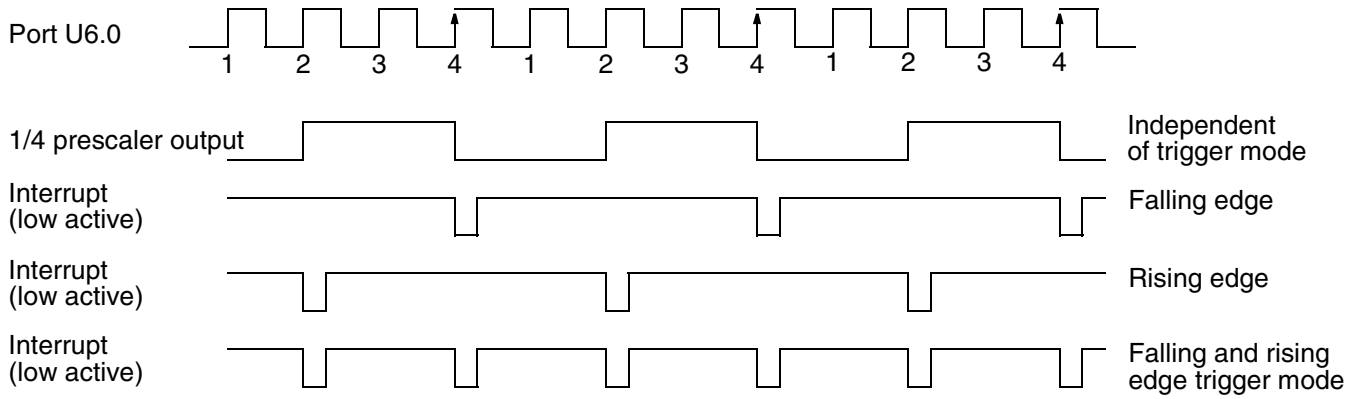


Fig. 10–6: Interrupt Timing (1/4 Prescaler On)

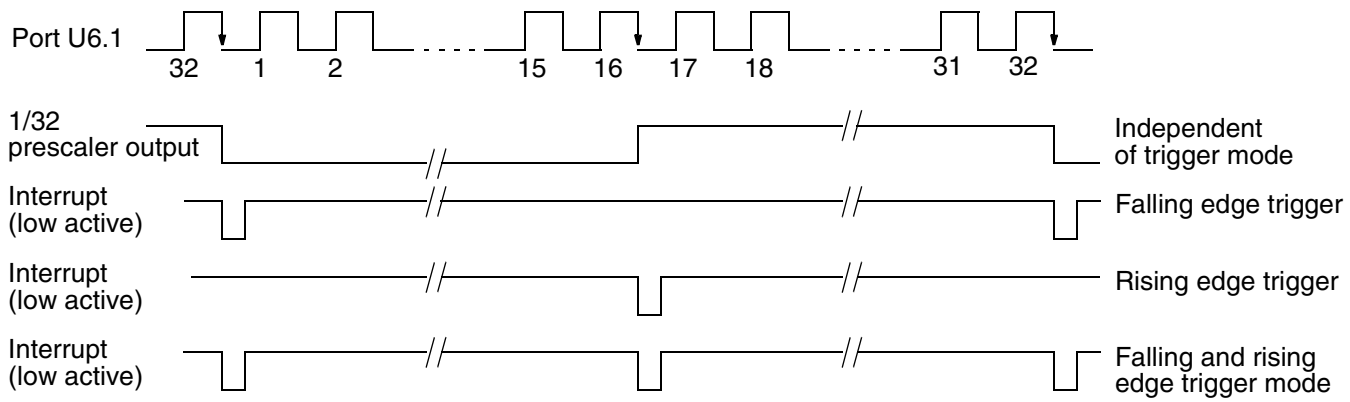


Fig. 10–7: Interrupt Timing (1/32 Prescaler On)

11. Ports

Three kinds of ports exist. The analog input port, Port 0, serves as input for the analog-to-digital converter. The universal ports, U1 to U7, serve as digital I/O and can be configured as LCD drivers. The high current ports, H0 to H3, serve as digital I/O and can be configured as stepper motor drivers.

11.1. Analog Input Port 0

The 9-bit-wide analog input port is called Port 0. Five of these analog input pins can additionally be configured as digital inputs. One pin is connected to a comparator, which can be selected as interrupt source.

Features

– 9-bit analog input multiplexer.

– 5 bits additionally usable as digital input ports.

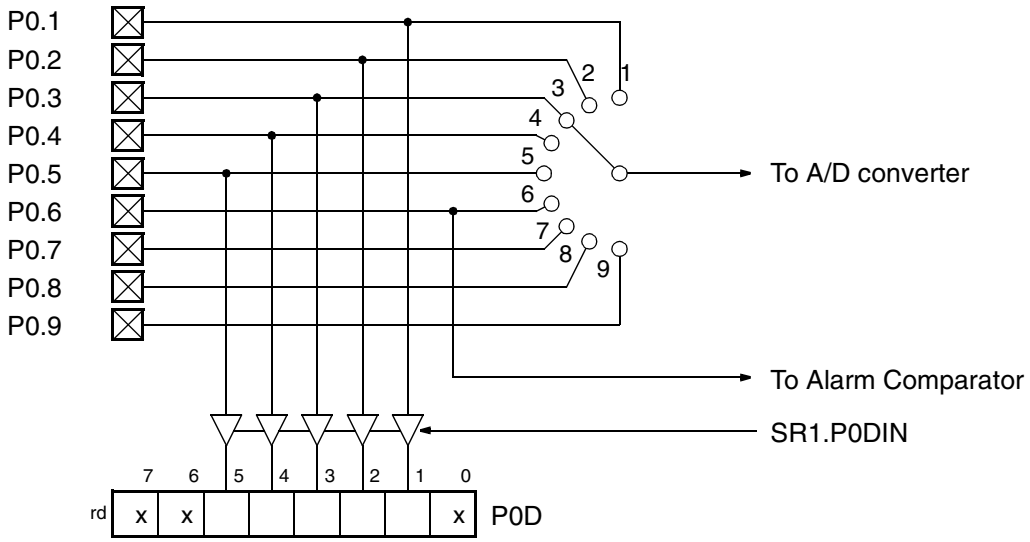


Fig. 11–1: Port 0 with Input Multiplexer and Undervoltage Alarm Comparator

The nine analog input lines are connected to a multiplexer. The output of this multiplexer is connected to a 10-bit A/D converter.

Port P0.6 is input of an undervoltage alarm comparator which is described in the A/D converter section.

Five of the analog input pins (P0.1 to P0.5) may be used as digital inputs if enabled by setting the flag P0DIN in the Standby Register SR1. The digital value of the input pins can be read in the Port 0 Data register P0D. These ports should either be used as analog or digital inputs.

D1 to 5	**Port 0 Digital Input 1 to 5**
r1:	High level at input pin.
r0:	Low level at input pin.

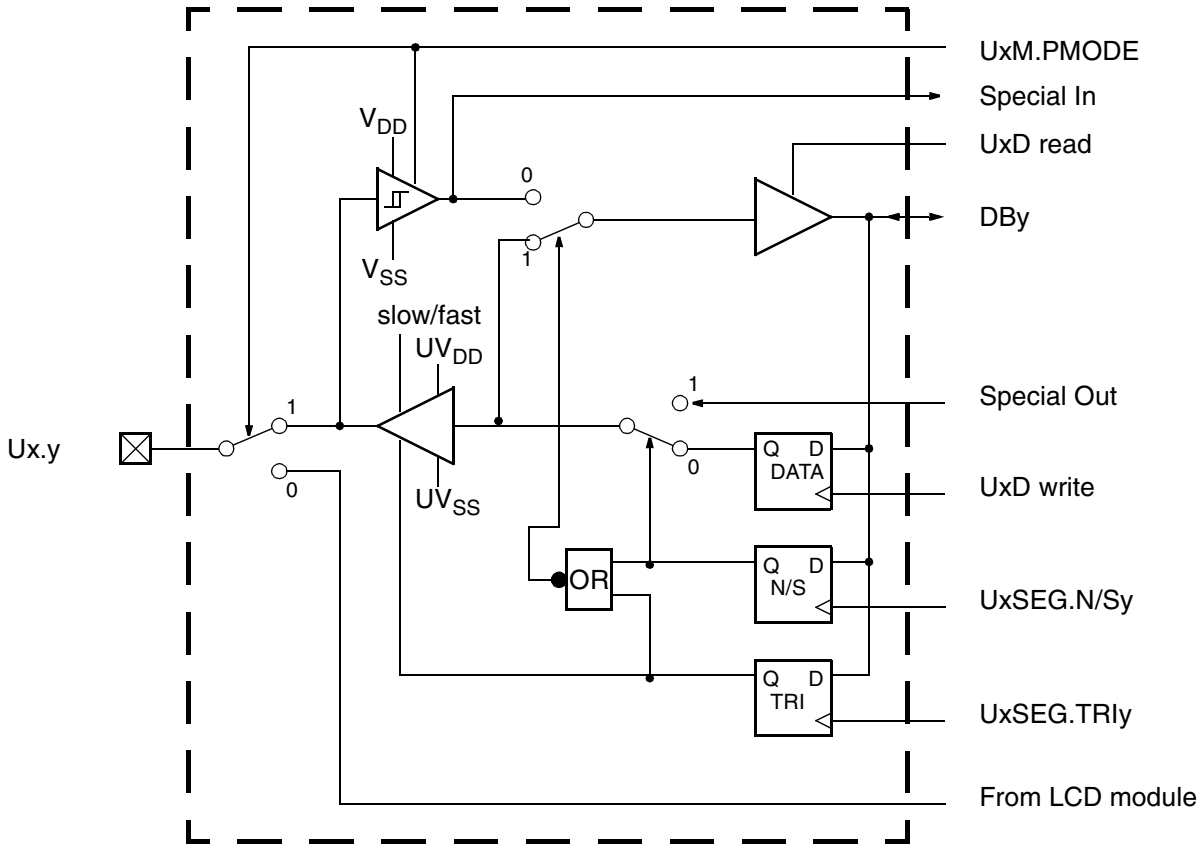
P0D	**Port 0 Data Register**							
	7	6	5	4	3	2	1	0
r	x	x	D5	D4	D3	D2	D1	x
	x	x	x	x	x	x	x	x Res

11.2. Universal Ports U1 to U7

There are 52 Universal Port pins. The universal ports U1 to U6 are 8 bits wide. The universal port U7 is 4 bits wide.

Features

- SW selectable as digital I/O or LCD driver.
- LCD mode: 1:4 multiplex, 5 V supply.
- I/O mode: Tristate output, current limited.
- Individually programmable to deliver constant current (slew rate).
- Schmitt hysteresis input buffer.



x: Port number 1 to 7
y: Port pin number 0 to 7

Fig. 11–2: Universal Port Circuit Diagram

Universal ports can be operated in different modes:

Table 11–1: Universal Ports Operating Modes

Modes		Function
Port Mode	Normal Input and Special Input	The SW uses the ports as digital input.
		The port input is additionally connected to specific hardware modules.
	Normal Output	The SW uses the ports as latched digital tristate output.
	Special Output	The output signals of specific hardware modules are directly port output source.
LCD Mode		The port bit serves as backplane/segment driver for an LC Display.

The Universal Port bits can be switched to LCD or Port mode in groups of two. The flag PMODE must be cleared to use port pins in LCD mode. If PMODE is set, they serve as I/O ports.

In both LCD and Port mode, the Port Slow mode may be defined for each individual Universal port bit. It reduces the current drive capability of the output stage.

After reset, all Universal Ports are in port, tristate condition.

11.2.1. Port Mode

Each port bit can be individually configured to several port modes. The output driver of each pin can be disabled (tristated) by setting the flag TRI. Set the flag N/S to select the source of the output value.

For Port mode, the UxM registers have to be set for mode selection, the UxSEG registers have to be properly set for individual port bit configuration and the UxD registers serve as I/O registers.

Table 11–2 shows configurations of flags if the corresponding flag PMODE is true (Port mode)

Table 11–2: Port Mode Register Settings

Mode	N/S	TRI	D	Function
Normal Input	x	1	x	READ of register UxD returns port pin input levels to data bus.
Normal Output	0	0	Data	WRITE to register UxD changes level of port pin output drivers. READ of register UxD returns the UxD register setting to the data bus.
Special Input	x	x	x	Port pin input level is presented to special hardware.
Special Output	1	0	x	Special hardware drives port pin. READ of register UxD returns port pin input levels to data bus.

In Port Mode, Special Input mode is always active. This allows manipulating the input signal to the special hardware through Normal Output operations by software.

As the Special Output mode allows reading the pin levels, the output state of the special hardware may be read by the CPU.

11.2.2. LCD Mode

For LCD Mode, the UxM registers have to be cleared for mode selection and the UxSEG registers serve as segment output data registers.

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 setting is necessary for these ports.

Port bits in LCD mode will always read as logical 0 as the port input buffer is turned off.

11.2.3. Port Slow Mode

The output drivers of all port pins together can be configured to operate in Fast or Slow mode by the Port Slow mode flag PSLW in register SR1.

All U-Ports exhibit two operating regions in the DC output characteristic (see Fig. 11–3). Near zero output voltage the internal driver transistors operate non-limited, to offer a low on-resistance. With larger output voltages, however, a limit is imposed on the output current. This measure helps to fight supply current transients and related EMI noise during port switching.

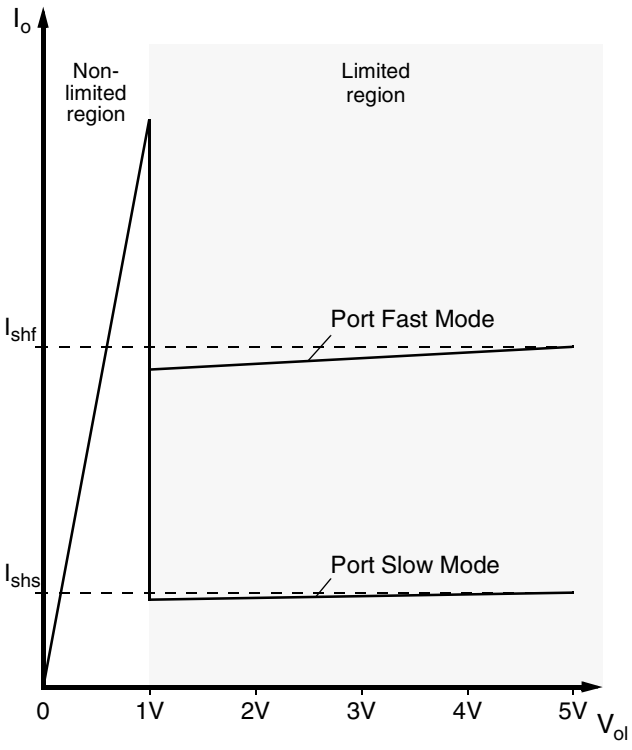


Fig. 11–3: Typical U-Port pull-down DC output characteristic (pull-up characteristic is complementary).

In this limited operating region, Port Fast mode and Port Slow mode select two different current limits I_{shf} and I_{shs}. Port Slow mode reduces the output current to a value where the output may even be shorted continuously to either supply rail. Thus, wired or-configurations can be put into practice. The external load resistance should be greater than 5 kOhms in Port Slow mode. Please note that in Port Normal Output mode, a READ of register UxD returns the UxD register setting, not the pin levels.

With PSLW = 0 all ports are in Fast Mode. Only port bits that are enabled via HW option will switch to Port Slow Mode with PSLW = 1. A port pin is in Fast Mode all the time if a zero is programmed to the appropriate HW option bit. It is not possible to switch this pin to Slow Mode. If a one is programmed, the pin can be toggled between Fast and Slow Mode by the flag PSLW. Please refer to section HW Options for information on port/option assignment.

It is recommended to place all LCD ports in the Port Slow mode.

11.3. Universal Port Registers

Universal Port Data Registers UxD contain input/output data of the corresponding port.

The “x” in UxD means the number of the port. Thus UxD stands for U1D to U7D. Remember that port U7 is only 4 bits wide. For this reason not all of the described registers and flags are available for U7.

UxD Universal Port x Data Register

	7	6	5	4	3	2	1	0	
r/w	D7	D6	D5	D4	D3	D2	D1	D0	
	0	0	0	0	0	0	0	0	Res

D0 to 7 Universal Port Data Input/Output
r: Read pin level resp. data latch.
w: Write data to data latch.
To use a port pin as software output, the appropriate driver must be activated and the N/S flag must be programmed to Normal Mode.

Universal Port Segment registers UxSEG together with the Universal Port Mode registers UxM are used to configure the appropriate ports in Port Mode. In LCD Mode the registers UxSEG contain the data for two segments each. For instance, register U2SEG76 and U2M76 control port U2, bits 7 and 6.

Registers U1SEG10, U1SEG32, U1M10, U1M32 and U3SEG32 differ from the corresponding control registers of other ports in LCD mode. Please refer to the special description of these registers at the end of this section.

UxSEG10 Universal Port x Segment Register of Ux.1 and Ux.0

	7	6	5	4	3	2	1	0	
w	x	N/S1	TRI1	x	x	N/S0	TRI0	x	Port
w	SEG1.3	SEG1.2	SEG1.1	SEG1.0	SEG0.3	SEG0.2	SEG0.1	SEG0.0	LCD
	0	0	1	0	0	0	1	0	Res

UxM10 Universal Port x Mode Register of Ux.1 and Ux.0

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

UxSEG32 Universal Port x Segment Register of Ux.3 and Ux.2

	7	6	5	4	3	2	1	0	
w	x	N/S3	TRI3	x	x	N/S2	TRI2	x	Port
w	SEG3.3	SEG3.2	SEG3.1	SEG3.0	SEG2.3	SEG2.2	SEG2.1	SEG2.0	LCD
	0	0	1	0	0	0	1	0	Res

UxM32 Universal Port x Mode Register of Ux.3 and Ux.2

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

UxSEG54 Universal Port x Segment Register of Ux.5 and Ux.4

	7	6	5	4	3	2	1	0	
w	x	N/S5	TRI5	x	x	N/S4	TRI4	x	Port
w	SEG5.3	SEG5.2	SEG5.1	SEG5.0	SEG4.3	SEG4.2	SEG4.1	SEG4.0	LCD
	0	0	1	0	0	0	1	0	Res

UxM54 Universal Port x Mode Register of Ux.5 and Ux.4

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

UxSEG76 Universal Port x Segment Register of Ux.7 and Ux.6

	7	6	5	4	3	2	1	0	
w	x	N/S7	TRI7	x	x	N/S6	TRI6	x	Port
w	SEG7.3	SEG7.2	SEG7.1	SEG7.0	SEG6.3	SEG6.2	SEG6.1	SEG6.0	LCD
	0	0	1	0	0	0	1	0	Res

UxM76	Universal Port x Mode Register of Ux.7 and Ux.6								
	7	6	5	4	3	2	1	0	
w	x	x	x	x	x	x	x	PMODE	
	0	0	0	0	0	0	0	1	Res

N/S0 to 7 Normal/Special Mode Flag 0 to 7
w1: Special Mode. Special hardware drives pin.
w0: Normal Mode. Data latch drives pin.
The corresponding Port Mode flag PMODE must be true to make this flag valid. The N/S flag defines from which source the pin is driven if TRI is false.

TRI0 to 7 Tristate Flag 0 to 7
w1: Output driver is tristate
w0: Output driver is active
The corresponding Port Mode flag PMODE must be true to make this flag valid.

SEGh.0 to .3 LCD Segment Driver High, Bits 0 to 3
SEGl.0 to .3 LCD Segment Driver Low, Bits 0 to 3
In LCD Mode each port pin is controlled by a field of four LCD Segment bits. Each segment register UxSEG contains two fields of segment data, each four bit wide. “h” stands for the high, “l” for the low pin number. For instance, U2SEG76.SEG7.3 to U2SEG76.SEG7.0 control LCD segments driven by port U2.7.

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.

PMODE Port Mode Flag
Select the mode of the corresponding port pins.
w1: The two port pins are in Port mode.
w0: The two port pins are in LCD mode.

11.3.1. Special Register Layout of Universal Port 1

Universal Port 1 pins U1.0 to U1.3 provide backplane signals in LCD Mode. To operate any ports as LCD segment driver it is necessary to switch these ports to LCD mode. All four pins will be switched together (not in groups by two) to LCD Mode by clearing the flag PMODE in register U1M30.

Thus, U1M30 replaces U1M10 and U1M32.

U1M30	Universal Port 1 Mode Register of U1.3 to U1.0								
	7	6	5	4	3	2	1	0	
w	x	x	x	x	x	x	x	PMODE	
	0	0	0	0	0	0	0	1	Res

As backplane ports U1.0 to U1.3 require no segment data setting, SEG bits are not available in register U1SEG10 and U1SEG32.

U1SEG10	Universal Port 1Segment Register of U1.1 and U1.0								
	7	6	5	4	3	2	1	0	
w	x	N/S1	TRI1	x	x	N/S0	TRI0	x	Port
w	LCDSLV								LCD
	0	0	1	0	0	0	1	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 universal ports in LCD mode into the display. The flag LCDSLV is available only in LCD mode. This flag is not available in other universal port registers.

U1SEG32	Universal Port 1Segment Register of U1.3 and U1.2								
	7	6	5	4	3	2	1	0	
w	x	N/S3	TRI3	x	x	N/S2	TRI2	x	Port
	0	0	1	0	0	0	1	0	Res

11.3.2. Special Register Layout of Universal Port 3.2

U3.2, in Port Special Output mode, provides the DIGITbus connection. For this purpose it can be switched into a Double Pull-down Mode (DPM) by setting U3SEG32.DPM2, where

– the short circuit current I_{shs} is doubled (with Port Slow Mode enabled for U3.2 by HW Option, and SR1.PSLW set to 1)
– the output configuration is pull-down, not the standard push-pull.

By these means, this port may be configured to operate as connection to the wired-or, single-wire DIGITbus with external pull-up resistor.

U3SEG32	Universal Port 3 Segment Register of U3.3 and U3.2								
	7	6	5	4	3	2	1	0	
w	x	N/S3	TRI3	x	DPM2	N/S2	TRI2	x	Port
w	SEG3.3	SEG3.2	SEG3.1	SEG3.0	SEG2.3	SEG2.2	SEG2.1	SEG2.0	LCD
	0	0	1	0	0	0	1	0	Res

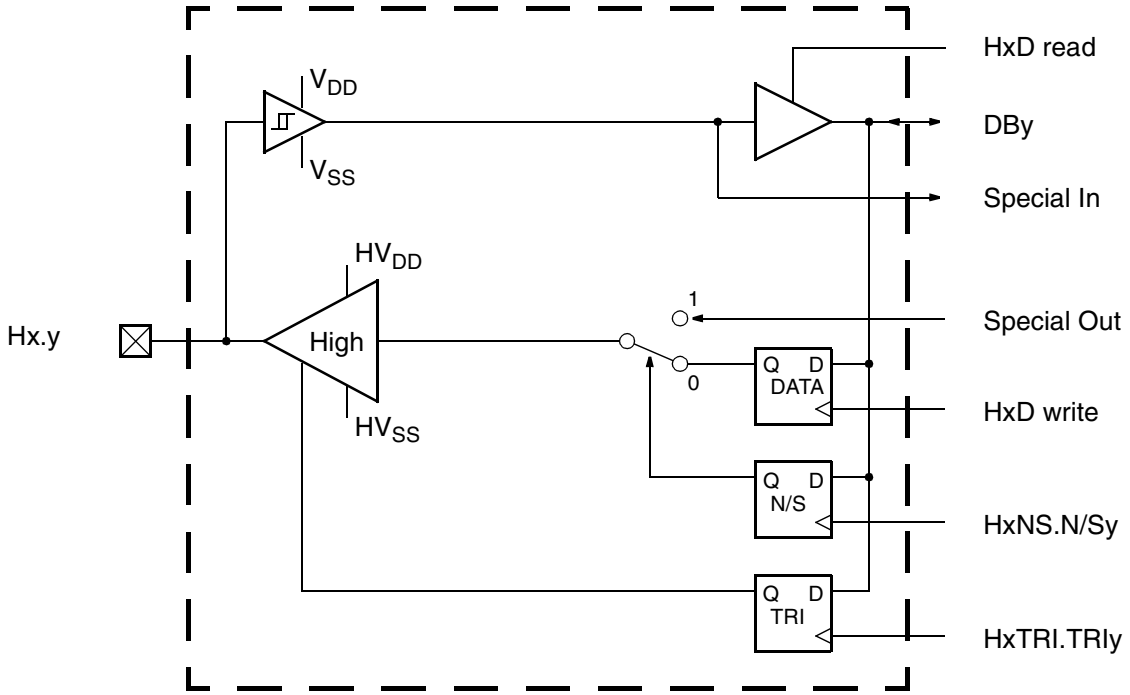
DPM Double Pull-Down Mode
w1: Output driver is pull-down, I_{shs} (Port Slow mode) doubled.
w0: Standard.

11.4. High Current Ports H0.0 to H3.5

The High Current Ports 0, 1, 2 and 3 are used to drive coils of stepper motors. Each port is 6 bits wide. They are similar to universal ports, but as the name says, they can drive higher currents. H-Ports can be operated via software like Universal Ports (Port Mode). Their Special Out connections are connected with the stepper motor module, or with the PWM output.

Features

- Tristate output.
- ±30 mA output current.
- Schmitt hysteresis input buffers.
- Reduced slew rate of current and voltage for driving resistive, capacitive or inductive loads.



x: Port number 0 to 3
y: Port pin number 0 to 5

Fig. 11–4: High Current Port Circuit Diagram

The H-Ports H0 and H1 are supplied by the power supply pins HVDD1 and HVSS1. The H-Ports H2 and H3 are supplied by the power supply pins HVDD2 and HVSS2.

Flag TRI is used to switch the output driver on and off, and the flag N/S defines the mode of each port pin. Table 11–3 shows the various selectable modes.

Table 11–3: Register Settings

Mode	N/S	TRI	D	Function
Normal Input	x	1	x	READ of register HxD returns port pin input levels to data bus.
Normal Output	0	0	Data	WRITE to HxD changes level of output, READ of HxD reads pin level

Table 11–3: Register Settings

Mode	N/S	TRI	D	Function
Special Input	x	x	x	Port pin input level is presented to special hardware.
Special Output	1	0	x	Special hardware drives port pin. READ of register HxD returns port pin input levels to data bus.

The Special Outputs of high current ports are connected to the Stepper Motor module or some PWMs. Please refer to Pin Assignment and Description for information on assignment of PWMs to H-Port pins.

Twenty of the twenty-four high-current ports are connected to the stepper motor control module. Two high-current ports, together with a coil, form an H-Bridge. Two H-Bridges are necessary to operate a stepper motor. The twenty stepper motor outputs can thus drive five stepper motors.

The N-channel and the P-channel transistor of the output driver are controlled separately. Thus crossover currents are eliminated.

The output levels of the ports during and after reset are low, to avoid floating coils.

11.5. High Current Port Registers

High Current Port Data registers are used to input/output digital values. The “x” means the number of the port. Thus HxD stands for H0D, H1D, H2D or H3D.

HxD — High Current Port x Data Register

	7	6	5	4	3	2	1	0	
r/w	x	x	D5	D4	D3	D2	D1	D0	
	0	0	0	0	0	0	0	0	Res

D0 to 5 H-Port Data Input/Output
r: Read pin level .
w: Write data to data latch.
To use a port pin as output, the appropriate driver must be activated and N/S mode flag must be set to normal mode.

The High Current Tristate and Normal/Special registers (HxTRI and HxN/S) are used to configure the corresponding port.

HxTRI — High Current Port x Tristate Register

	7	6	5	4	3	2	1	0	
w	x	x	TRI5	TRI4	TRI3	TRI2	TRI1	TRI0	
	0	0	0	0	0	0	0	0	Res

HxNS — High Current Port x Normal/Special Register

	7	6	5	4	3	2	1	0	
w	x	x	N/S5	N/S4	N/S3	N/S2	N/S1	N/S0	
	0	0	0	0	0	0	0	0	Res

N/S0 to 5 Normal/Special Mode Flag 0 to 5
w1: Special Mode. Special hardware drives pin.
w0: Normal Mode. Data latch drives pin.
The N/S flag defines from which source the pin is driven if TRI is false.

TRI0 to 5 Tristate Flag 0 to 5
w1: Output driver is tristate
w0: Output driver is active

12. A/D Converter (ADC)

This 10-bit analog-to-digital converter allows the conversion of an analog voltage in the range of 0 to U_{Ref}, into a digital value. A multiplexer connects the ADC to one of 9 analog input ports. A sample and hold circuit holds the analog voltage during conversion. The duration of the sampling time is programmable. The A/D conversion is done by a charge balance A/D converter using successive approximation.

Features

- A/D converter with 10-bit resolution.
- Successive approximation, charge balance type.
- Input multiplexer with 9 analog channels.
- Sample and hold circuit.
- 4/8/16/32 µs conversion selectable for optimum throughput/accuracy balance.
- 2.5 V to 5 V external reference input.
- Zero standby current, 300 µA active current.

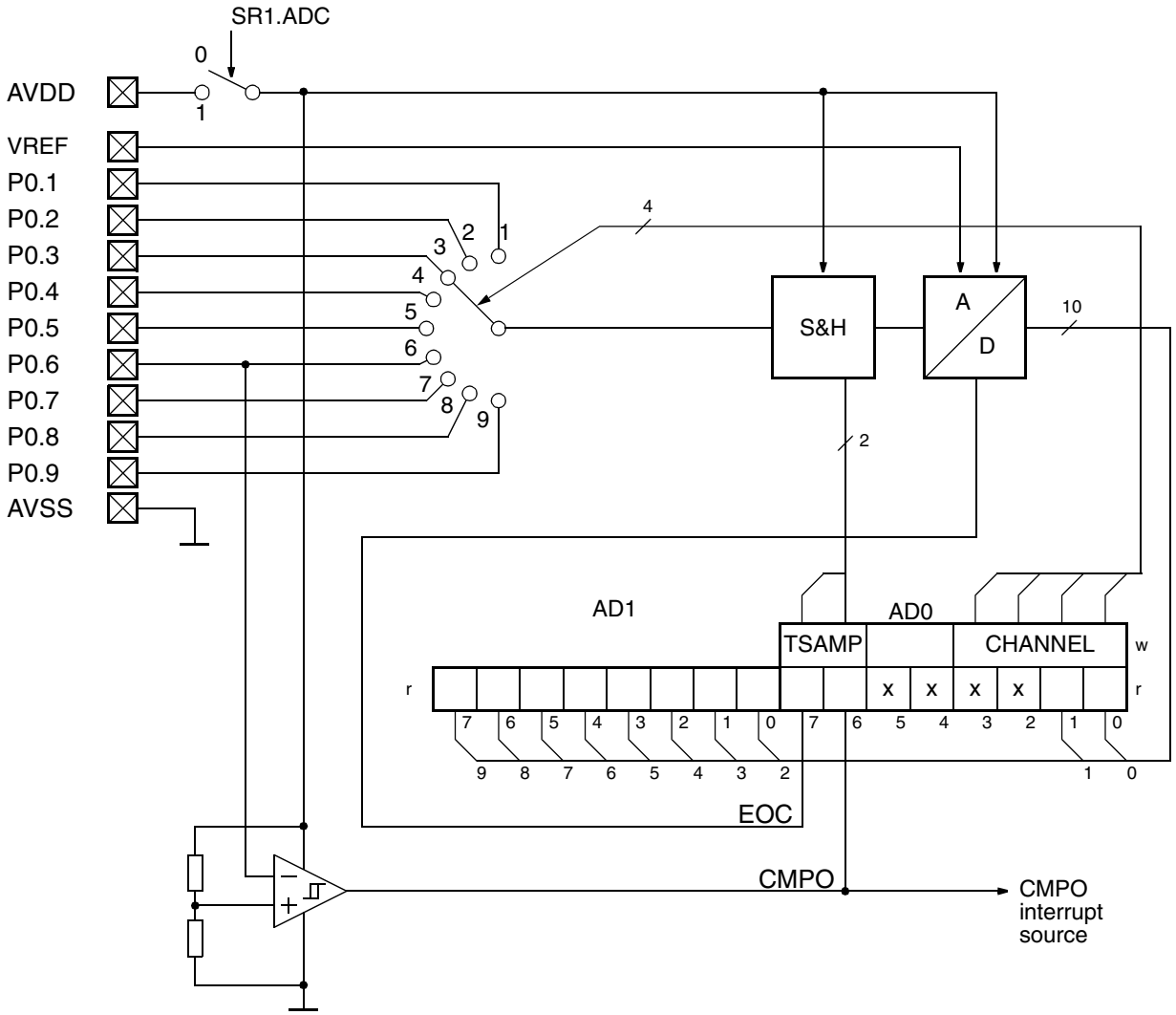


Fig. 12–1: Block Diagram

12.1. Operation

12.1.1. ADC

After reset, the module is off (zero standby current). The module is enabled by the flag SR1.ADC. The user must be sure that the flag End of Conversion (EOC) in register AD0 is true, before he starts to operate the module.

A write access to register AD0 indicating sample time and channel number starts the conversion. The flag EOC signals the end of conversion. The 10-bit result is stored in the registers AD1 (8 MSB) and AD0. The conversion rate depends on the software, the oscillator frequency, and the programmed sample time.

The module may be operated in CPU FAST or SLOW mode.

12.1.1.1. 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

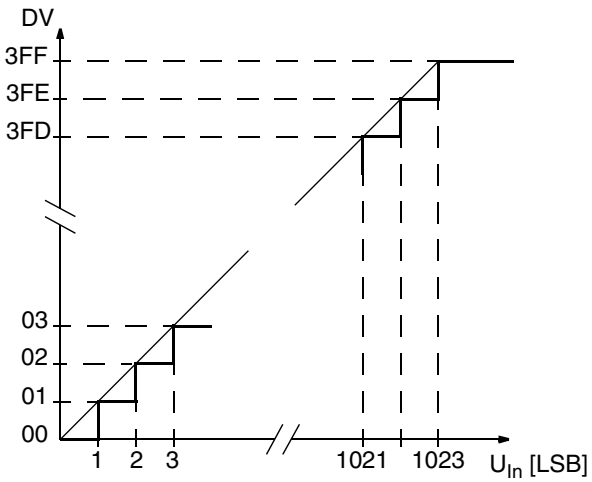


Fig. 12–2: Characteristic Curve

The voltage on the reference input pin VREF can be set to any level in the range from 2.56 V to AVDD.

12.1.1.2. Measurement Errors

The result of the conversion mirrors the voltage potential of the sampling capacitance (typically 15 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 the sampling period, the potential error on the sampling capacitance is less than ±0.1 LSB.

Measurement errors can 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 TSAMP in register AD1.
- 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 potential on the sampling capacitance reaches the full source potential, 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. It results in a DC current that appears as flowing from the high potential source through the IC into the low potential source. This current is explained by the fact that during the sampling periods the high potential source always up-charges the sampling capacitance while the low potential source always discharges it.

12.1.2. P0.6 Comparator

In addition to the A/D converter the module contains a comparator. The level on port P0.6 is compared to AVDD/2. The state of the comparator output can be read at flag CMPO in register AD0.

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 CMPO interrupt source is gated with an internal clock. This is the reason why interrupts are generated as long as the level at P0.6 is lower than the internal reference.

12.2. Registers

A write access to register AD0 starts the A/D conversion of the written channel number and sampling duration. The flag EOC signals the end of conversion. The result is stored in register AD1 (bit 9 to 2) and in register AD0 (bit 1 and 0).

AD0 ADC Register 0

	7	6	5	4	3	2	1	0	
r	EOC	CMPO	x	x	x	x	AN1	AN0	
w	TSAMP				CHANNEL				
	0	0	x	x	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

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 first conversion after enabling the module by setting SR1.ADC.

CMPO Comparator Output
r1: P0.6 is lower than reference.
r0: P0.6 is higher than reference.
Zero means that the voltage at P0.6 is higher than the comparator reference voltage.

TSAMP Sampling Time
TSAMP adjusts the sample time and the conversion time. The total conversion time is 20 clock cycles longer than the sample time.

Table 12–1: Sampling Time Adjustment

TSAMP	t_{Sample}	$t_{Conversion}$
0H	20 T_{OSC}	40 T_{OSC}
1H	60 T_{OSC}	80 T_{OSC}
2H	140 T_{OSC}	160 T_{OSC}
3H	300 T_{OSC}	320 T_{OSC}

Sampling starts one clock cycle after completion of the write access to AD0.

CHANNEL Channel of Input Multiplexer
CHANNEL selects from which pin of port P0 the conversion is done. The MSB of CHANNEL is bit 3. No port pin is connected to the ADC, if values are selected which are not recommended by table 12–2. In this case the result is not defined because the input of the A/D converter is open. After reset, CHANNEL is set to zero. No channel is selected in this case.

Table 12–2: ADC Input Multiplexer

CHANNEL	Port Pin
1H	P0.1
2H	P0.2
3H	P0.3
4H	P0.4
5H	P0.5
6H	P0.6
7H	P0.7
8H	P0.8
9H	P0.9

AN 9 to 0 Analog Value Bit 9 to 0
The 10-bit analog value is in the range of 0 to 1023. 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.

13. Timers (TIMER)

Three general-purpose timers are implemented. T0 is a 16-bit timer, T1 and T2 are 8-bit timers.

13.1. Timer T0

Timer T0 is a 16-bit auto-reload down counter. It serves to deliver a timing reference signal to the IRC, to output a frequency signal or to produce time stamps.

Features

- 16-bit auto-reload counter
- Time value readable
- Interrupt source output
- Frequency output

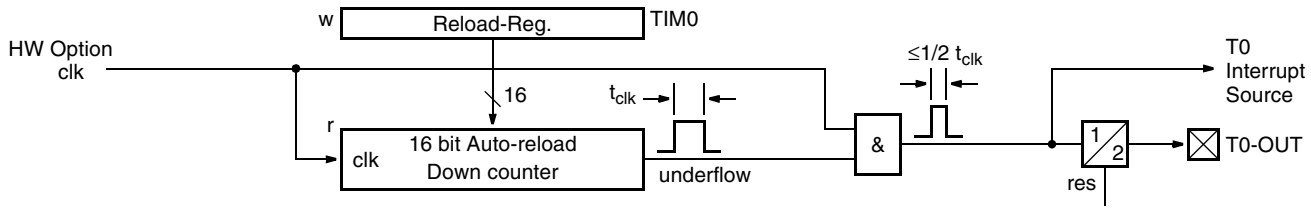


Fig. 13–1: Timer T0 Block Diagram

13.1.1. Principle of Operation

13.1.1.1. General Remarks

The timer's 16-bit down-counter is clocked by the input clock and counts down. Falling below zero, it generates an output pulse (underflow) to get reloaded with the value in its reload register which is counted down subsequently.

T0 is not affected by CPU SLOW mode.

13.1.1.2. Operation

The clock input frequency can be set via HW option (see Table 13–1 on page 98).

Prior to entering active mode, the U-Ports assigned to function as T0-OUT outputs have to be properly SW initialized (Table 13–1). 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 FFFFh 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.

Falling below zero, 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.

13.1.1.3. Precautions

16-bit CPU commands do not generally keep to a certain order in addressing high and low bytes of a register. Make sure that the command used performs reading a 16-bit value low byte first and writing high byte first.

In case of uncertainty use 8-bit commands.

Table 13–1: Module specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
T0	Input clock	FFA0h	T0-OUT output	U3.4 special out	

13.1.2. Registers

TIM0L T0 low byte

	7	6	5	4	3	2	1	0	
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

TIM0H T0 high byte

	7	6	5	4	3	2	1	0	
r	Latched high byte of down-counter								
w	High byte of reload value								
	1	1	1	1	1	1	1	1	Res

TIM0 has to be read low byte first and written high byte first.

Table 13–2: Reload Register Programming

Reload value	Output interrupt source frequency is divided by	Output T0-OUT is divided by
0000h	1	2
0001h	2	4
0002h	3	6
:	:	:
FFFFh	65536	131072

13.2. Timer T1 and T2

Timer T1 and T2 are 8-bit auto-reload down counters. They serve to deliver timing reference signals to the IR or to output frequency signals.

Table 13–3 describes implementation-specific HW Option addresses and enable flags of T1 and T2.

Features

- 8-bit auto-reload counter
- Interrupt source output
- Frequency output

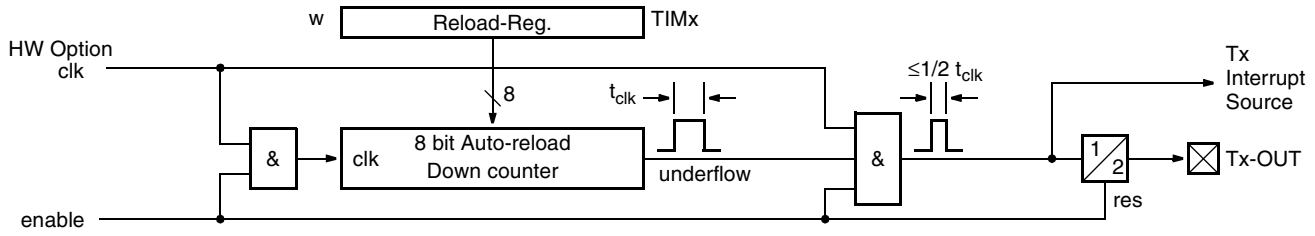


Fig. 13–2: Timer T1 and T2 Block Diagram

13.2.1. Principle of Operation

13.2.1.1. General Remarks

The timer's 8-bit down-counter is clocked by the input clock and counts down. Falling below zero, it generates an output pulse (underflow) to get reloaded with the value in its reload register which is counted down subsequently.

Tx is not affected by CPU SLOW mode.

13.2.1.2. Operation

The clock input frequencies can be set via HW options (see Table 13–3 on page 99). 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 13–3). The ports have to be configured Special Out. Refer to "Ports" for details.

To initialize a timer, reload register TIMx has to be set to the desired time value, still in standby mode.

To enter active mode, set the corresponding enable bit in the standby registers (see Table 13–3 on page 99). 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.

Falling below zero, 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 can be, but need not be, connected to the interrupt controller directly or via multiplexer. This is a HW option which is done by the factory only. Please refer to section Interrupt Controller.

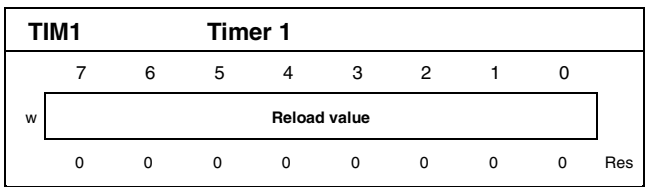
Returning Tx to standby mode by resetting its respective enable bit will halt its counter and will set its outputs LOW. The register TIMx remains unchanged.

The state of the down-counter is not readable.

Table 13–3: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
T1	Input clock	FFA7h	T1-OUT output	U6.2 or U6.5 special out	SR1.TIM1
T2	Input clock	FFA8h	T2-OUT output	U3.7 special out	SR2.TIM2

13.2.2. Registers



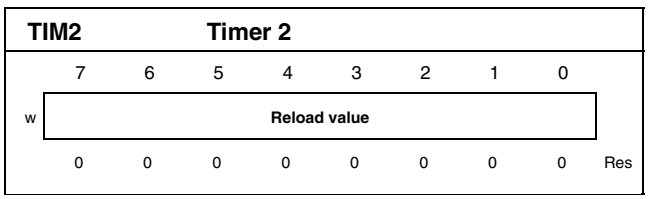


Table 13–4: Reload Register Programming

Reload value	Output interrupt source frequency is divided by	Output Tn-OUT is divided by
00h	1	2
01h	2	4
02h	3	6
:	:	:
FFh	256	512

14. Pulse Width Modulator (PWM)

A PWM is an 8-bit reload down-counter with fixed reload interval. It serves to generate a frequency signal with variable pulse width or, with an external low pass filter, as a digital-to-analog converter.

The number of PWMs implemented is given in table 14–1. The “x” in register names distinguishes the module number.

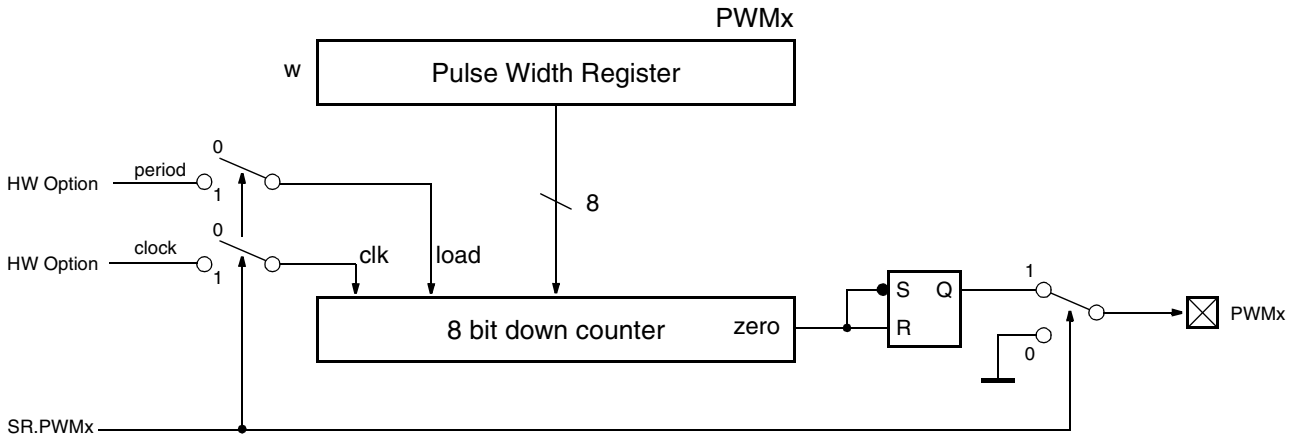


Fig. 14–1: PWM Block Diagram

14.1. Principle of Operation

14.1.1. General Remarks

A PWM’s 8-bit 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.

A PWM is not affected by CPU SLOW mode.

14.1.2. Hardware settings

The clock and period input frequencies can be set via HW option (Table 14–1). For full resolution a clock-to-period frequency ratio of 256 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.

Table 14–1: Module specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
PWM0	Clock and period	FFA1h FFA2h	PWM0 output	H1.0 or H2.4 special out	SR0.PWM0
	H1.0 SME/PWM0 output multiplexer	FFC2h			
PWM1	Clock and period	FFA3h FFA4h	PWM1 output	H3.0 special out	SR0.PWM1
PWM2	Clock and period	FFA5h FFA6h	PWM2 output	H1.1 or U5.6 special out	SR2.PWM2
	H1.1 SME/PWM2 output multiplexer	FFC2h			

Table 14–1: Module specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
PWM3	Clock and period	FFA1h FFA2h	PWM3 output	H3.1 special out	SR2.PWM3
PWM4	Clock and period	FFA3h FFA4h	PWM4 output	H2.5 special out	SR2.PWM4

Some of the PWM outputs share pins with outputs of other modules. The output multiplexer is controlled by HW option (Table 14–1).

14.1.3. 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 14–1). The ports have to be configured Special Out. Refer to “Ports” for details.

14.1.4. Operation

After reset, a PWM is in standby mode (inactive) and the output signal PWMx is LOW.

For entering active mode, set the respective enable bit (Table 14–1). Then write the desired pulse width value to register PWMx. 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.

Returning a PWM to standby mode by resetting its respective enable flag will immediately set its output LOW.

The state of the down-counters is not readable.

14.2. Registers

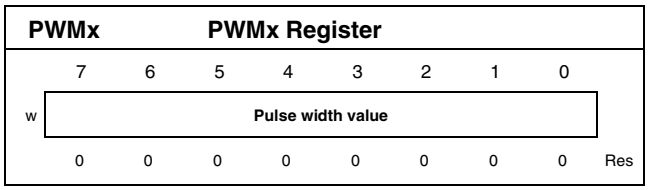


Table 14–2: Pulse Width Programming

Pulse width value	Pulse duty factor
00h	0% (Output is permanently low)
01h	1/256
02h	2/256
:	:
FEh	254/256
FFh	100% (Output is permanently high) [1])
[1]) Pulse duty factor 255/256 is not selectable.	

15. Capture Compare Module (CAPCOM)

The Capture Compare Module (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 exceeds 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

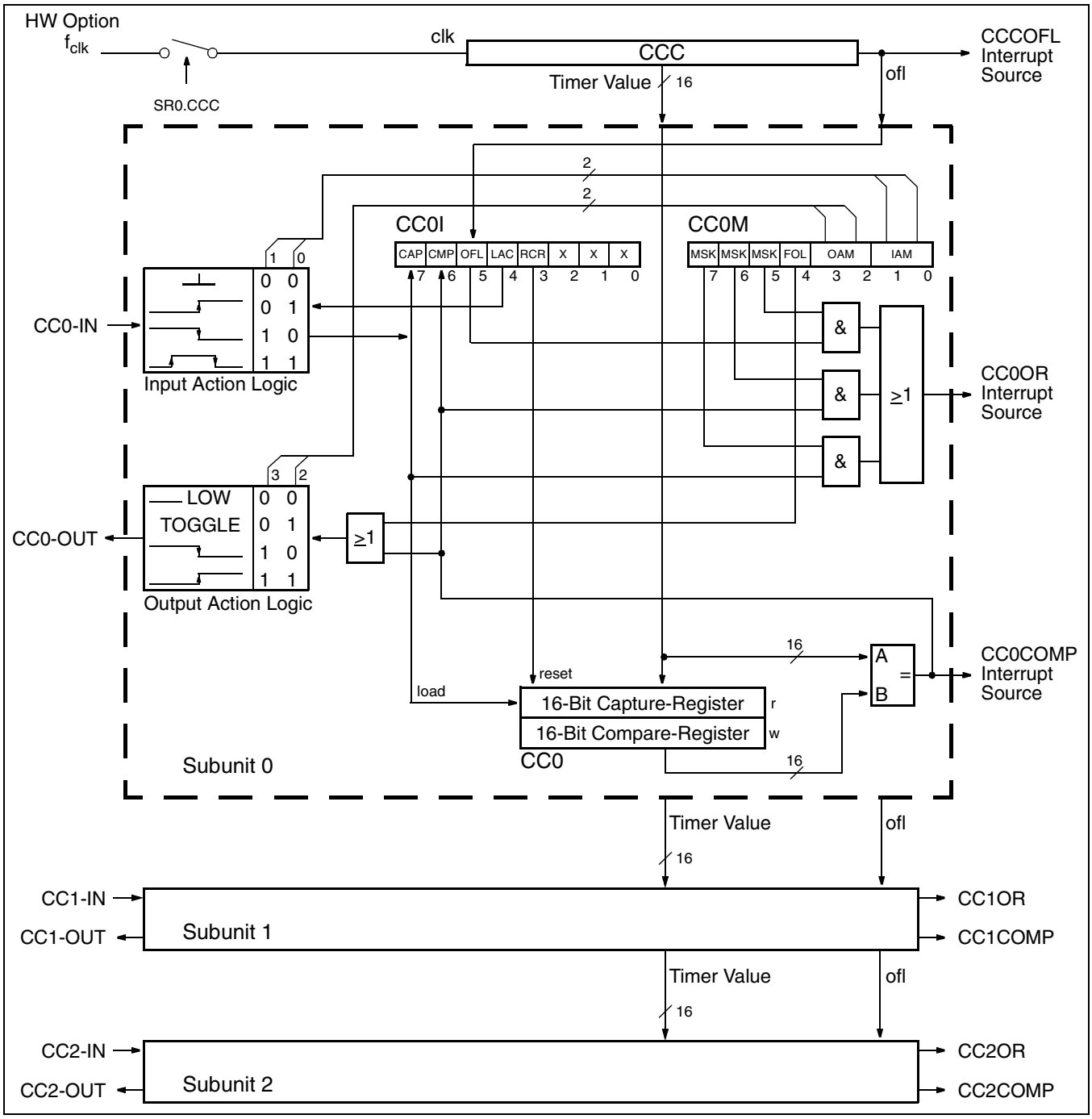


Fig. 15–1: CAPCOM Module Block Diagram

15.1. Principle of Operation

15.1.1. General Remarks

The Capture Compare Module (CAPCOM, Fig. 15–1) contains one common free-running 16-bit counter (CCC) and a number of capture and compare subunits (SU). More details are given in Table 15–1. 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.

For limitations on operating the CAPCOM module in CPU SLOW mode, see section 15.1.5.3.

15.1.2. Hardware Settings

The CCC clock frequency must be set via HW option (Table 15–1). Refer to “HW Options” for setting them.

15.1.3. Initialization

After system reset the CCC and all SUs are in standby mode (inactive).

In standby mode, the CCC is reset to value 0000h. Capture and compare registers CCx are reset. 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 startup out of standby mode is guaranteed.

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 15–1). The Output Action ports have to be configured as special out and the Input Capture ports as special in. Refer to “Ports” for details.

15.1.3.1. 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 15.2. for details.

Please note that the compare register CCx is reset in standby mode. It can only be programmed in active mode.

15.1.4. Operation of CCC

For entering active mode of the entire CAPCOM module set the enable bit (Table 15–1).

The CCC will immediately start up-counting with the selected clock frequency and will deliver this 16-bit value to the SUs.

Table 15–1: Unit specific settings

Subunit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SU0	Input clock	FFA9h	CC0-OUT	U5.1 special out	SR0. CCC
			CC0-IN	U5.0 special in	
SU1			CC1-OUT	U4.6 & U3.6 special out	
			CC1-IN	U4.7 special in	
SU2			CC2-OUT	U3.3 special out	
			CC2-IN	U4.6 special in	

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 generation of an overflow interrupt event. The interrupt (CCCOFL) is fed directly to the Interrupt Controller and also to all SUs where further processing takes place.

15.1.5. Operation of Subunit

15.1.5.1. 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 when activated.

– The Output Action logic is triggered.
Four different reactions are selectable for the Output Action signal: according to field CCxM.OAM (Table 15–2) the equal state will lead to a high or low level, or toggling or inactivity on this output.
Another means to control the Output Action is bit CCxM.FOL. E.g. 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.

15.1.5.2. Capture and Input Action

The Input Action logic operates independently from the Output Action logic and is triggered by an external input in a way defined by field CCxM.IAM. As shown in Table 15–3 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 when 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 compare 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. Please 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. Please ensure that SW is prepared to handle such a situation.

For testing purposes, a permanent reset (FFFFh) may be forced on capture register CCx by setting bit CCxI.RCR. Please make sure that the reset is only temporary.

15.1.5.3. Interrupts

Each SU supplies two internal interrupt events:

1. Input Capture event and

2. Comparator equal state.

As previously explained, interrupt events will set the corresponding flags in register CCxI. In addition to the above mentioned two, the CCC Overflow interrupt event sets flag CCxI.OFL in each SU. Thus, three interrupt events are available in each SU. The corresponding flags 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 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. However, as explained in 15.1.5.2. 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 oscillator frequency, regardless of CPU SLOW 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 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 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.

15.1.6. Inactivation

The CAPCOM module is inactivated and returned to standby mode (power down mode) by setting the enable bit to 0. Section 15.1.3. applies.

CCxI and CCxM are only reset by system reset, not by standby mode.

15.1.7. Precautions

16-bit CPU commands do not generally keep to a certain order in addressing high and low bytes of a register. Make sure that the command used performs reading and writing a 16-bit value low byte first.

In case of uncertainty use 8-bit commands.

15.2. Registers

The CAPCOM module counter has to be read low byte first to avoid inconsistencies.

CCC **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

CCC **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	MSK	MSK	MSK	FOL	OAM		IAM		
	0	0	0	0	0	0	0	0	Res

MSK **Mask Flag**
r/w1: Enable.
r/w0: Disable.
These mask flags refer to the corresponding event flags in CAPCOM interrupt register.

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, by writing another “one”, the Output Action logic. 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 15–2: 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 15–3: 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.

OVL **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 7.1.5.2

RCR **Reset Capture Register**
r/w1: Reset capture register permanently to FFFFh.
r/w0: Release capture register.

CCx **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

CCx **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

16. Stepper Motor Module (SMM)

The SMM serves to control 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 16–4.

Features

- Multichannel pulse width modulated output
- Outputs offset for improved EMC properties
- Four quadrant operation
- 8-bit resolution
- HW Option selectable output cycle frequency

16.1. Functional Description

An 8-bit, free-running counter FRC (see Fig. 16–2) operates on the f_{SM} input clock (generally 4MHz) and creates an 8-bit counter word that is fed to a number of control units SMx.

A control unit (Fig. 16–2) 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 SMM (cf. Fig. 16–1). 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, signalling completion of the downloading.

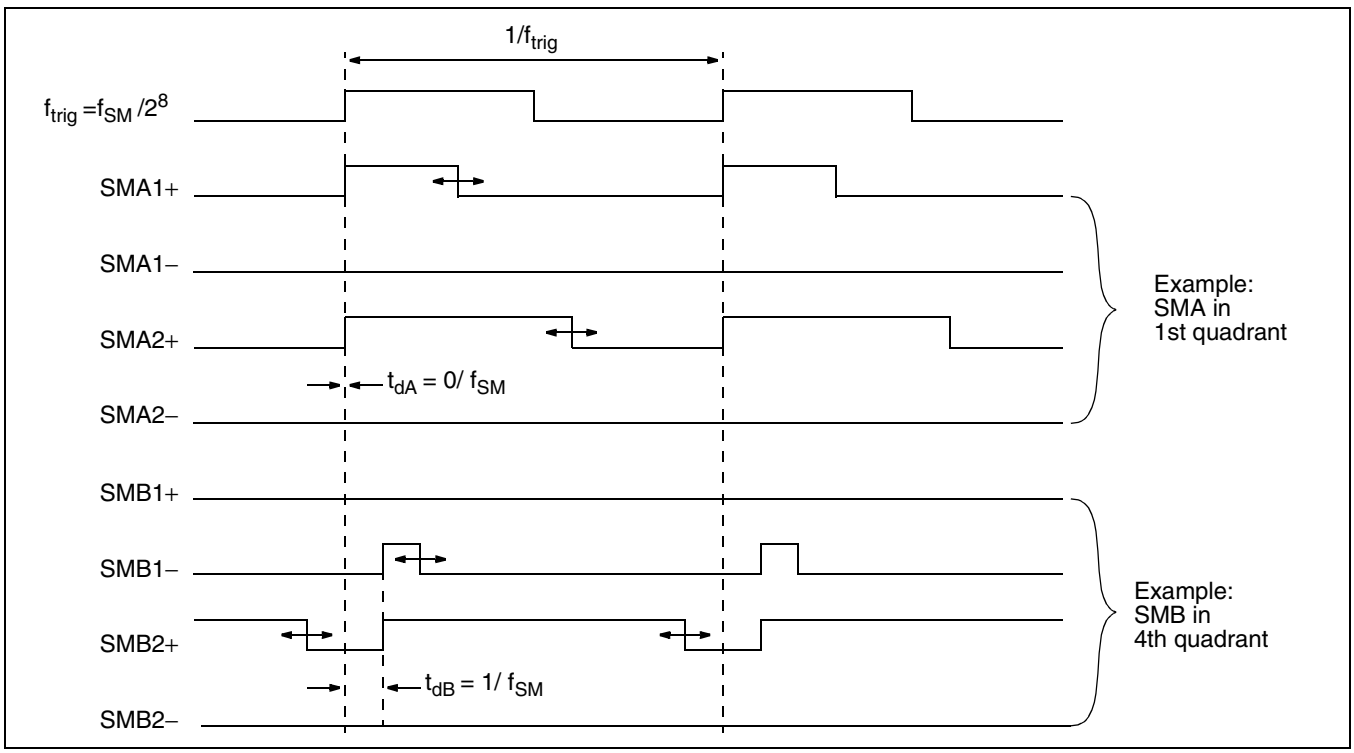


Fig. 16–1: Timing Diagram of Output Signals

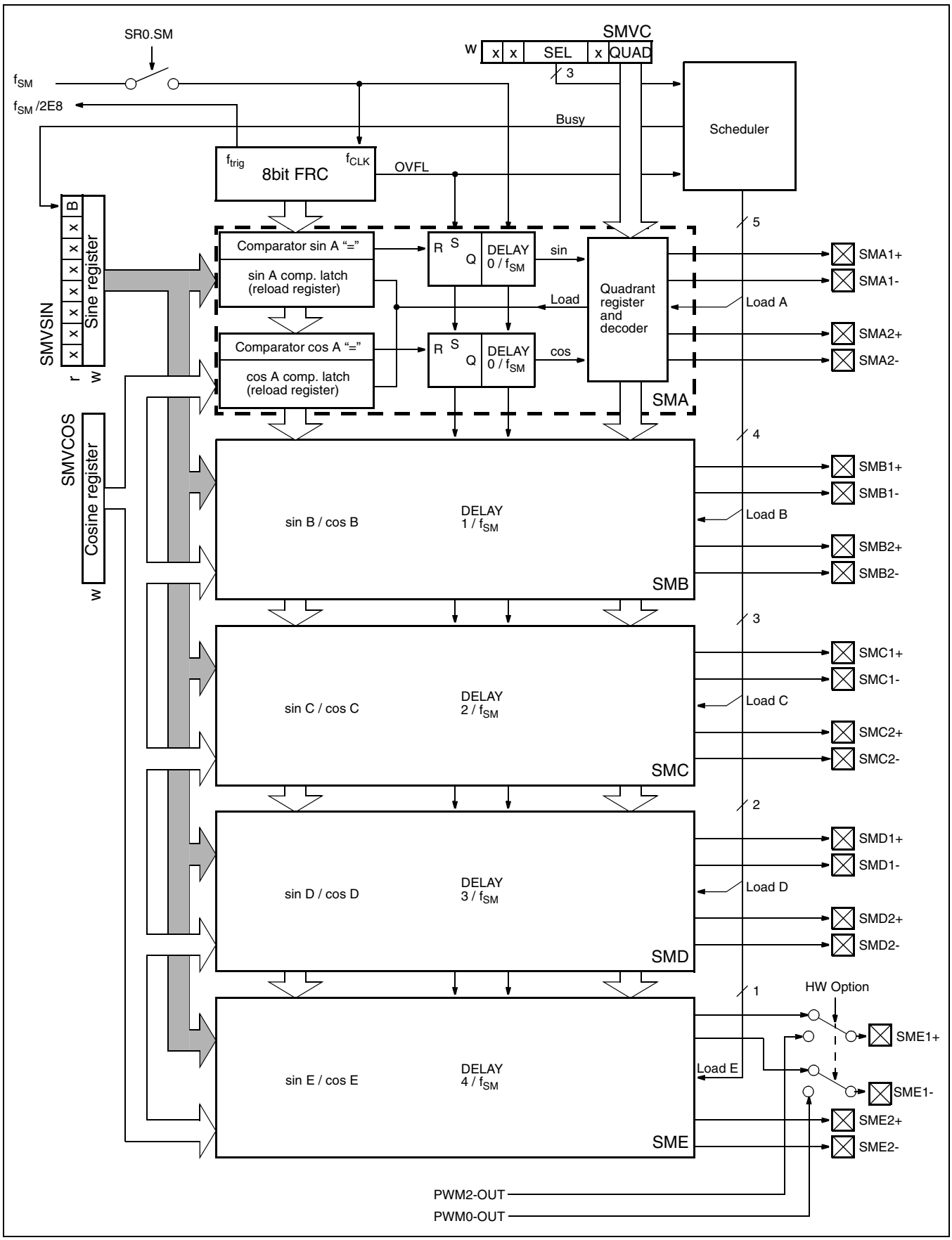


Fig. 16–2: Block Diagram of Output Generation Circuit

16.2. 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 16–1)**

QUAD **Quadrant selection field (Table 16–2)**

Table 16–1: SEL usage

SEL	selected control unit
000	SMA
001	SMB
010	SMC
011	SMD
100	SME
No other values are permitted.	

Table 16–2: QUAD setting and resulting control unit output signal function

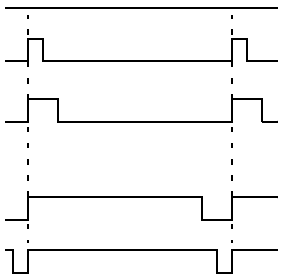
QUAD	Control unit output signal function			
	SMx1+	SMx1-	SMx2+	SMx2-
00	sine	VSS	cosine	VSS
01	sine	VSS	VSS	cosine
10	VSS	sine	VSS	cosine
11	VSS	sine	cosine	VSS

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 16–3: Usage of SMVSIN and SMVCOS registers

Value	Duty factor	Pulse Diagram
00h	0/256 (continuously low)	
01h	1/256	
02h	2/256	
:	:	
FEh	254/256	
FFh	255/256 [1])	
[1]) 256/256 (continuously high) is not available.		

16.3. Principle of Operation

The SMM may only be operated in CPU FAST mode.

16.3.1. Hardware settings

Prior to entering active mode, the f_{SM} input clock has to be set by HW Option (see Table 16–4 on page 111). 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 or of PWM modules as an alternative. Refer to Table 16–4 for the necessary settings.

Refer to section “HW Options” for details.

16.3.2. 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 16–4). The H-Ports have to be configured Special Out. Refer to “Ports” for details.

16.3.3. Operation

After reset, the SMM is in standby mode (inactive). The output lines to the H-Ports are low.

Table 16–4: Unit specific settings

Contr. Unit	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
SMA			SMAn+/- outputs	H0.0 to H0.3 special out	SR0.SM
SMB			SMBn+/- outputs	H1.2 to H1.5 special out	
SMC			SMCn+/- outputs	H2.0 to H2.3 special out	
SMD			SMDn+/- outputs	H3.2 to H3.5 special out	
SME	SME/PWM selection	FFC2h	SMEn+/- outputs	H0.4 to H1.1 special out	
All	Input clock selection	FFAEh			

For entering active mode, set bit SR0.SM. The FRC will immediately start counting but the control units' output lines will still be low.

16.3.3.1. 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 16–3) on the proper pins (see Table 16–2).

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.

16.3.4. Inactivation

Returning the SMM to standby mode by resetting bit SR0.SM will immediately halt the FRC, return all output signals to 0, reset all internal registers.

16.4. Rotor Zero Position Detection (RZPD)

In addition to 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 from Siemens VDO Automotive (SV) and may only be used with SV's prior approval.

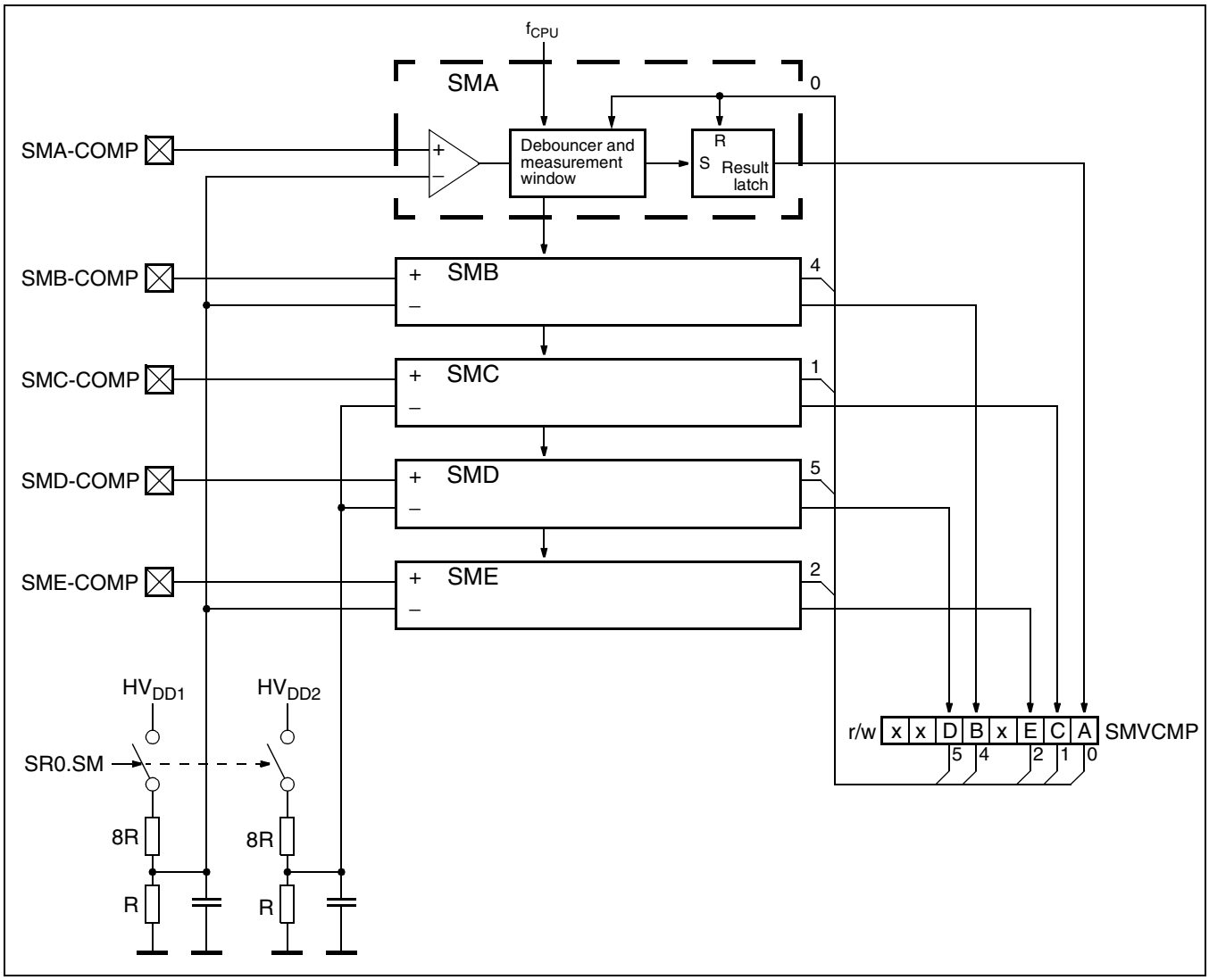


Fig. 16–3: Block Diagram of Rotor Zero Position Detection Circuit

16.4.1. Functional Description

Each control unit contains circuitry to detect an induced voltage resulting from the rotation of the connected motor's rotor (Fig. 16–3). 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.

16.4.2. Registers

SMVCMP SMM Comparator Register

	7	6	5	4	3	2	1	0	
r/w	x	x	ACRD	ACRB	x	ACRE	ACRC	ACRA	
	x	x	0	0	x	0	0	0	Res

ACRA to E	Analog Comparator Control and Result for SMA to SME
r0:	Capture result: no induced voltage detected
r1:	Capture result: induced voltage detected
w0:	Stop capture and clear result flag
w1:	Start capture

16.4.3. 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 16–5) temporarily has to be operated as an input to an internal analog comparator. Reconfigure this port as Special Input. Refer to "Ports" for details.

Table 16–5: RZPD input ports

Contr. Unit	Initialization	
	Item	**Setting**
SMA	SMA-COMP input	H0.0 special in
SMB	SMB-COMP input	H1.2 special in
SMC	SMC-COMP input	H2.0 special in
SMD	SMD-COMP input	H3.2 special in
SME	SME-COMP input	H0.4 special in

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 $1/f_{CPU}$) - 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, a '1' was still not read from the questioned unit's result flag SMVCMP.ACRx, this indicates that the 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.

17. 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

17.1. Principle of Operation

17.1.1. General Remarks

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. 17–3 on page 117).

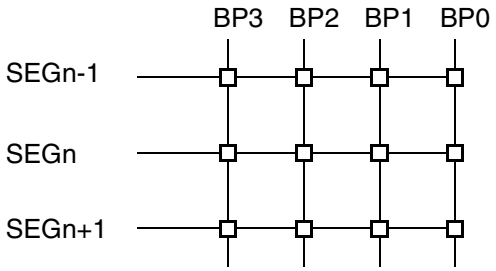


Fig. 17–1: Segments and Backplanes

A segment pin can drive 4 different voltage levels (UVDD, 1/3 VDD, 2/3 VDD, VDD) in LCD mode. The output of each segment pin is controlled by the segment field of the corresponding UxSEGy register. Each such register contains the segment fields of two neighboring segment pins. A segment field is 4 bits wide. Each flag (0 to 3) of a segment field corresponds to a backplane line (BP0 to BP3). If the segment flag, corresponding with the backplane line BPx is true, 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 UxSEGy register. It is validated (loaded to all corresponding slave registers) for all segment U-Ports simultaneously by a write access to register U1SEG10.

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.

17.1.2. Hardware settings

The LCD frame frequency is settable by HW option FFADh. The resulting frame frequency is the selected input frequency, divided by 120. It should be in the range from 50 Hz 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 PFST in register SR1 to HIGH to enable Port Slow mode.

17.1.3. 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 registers UxSEGy and by validating it by a write access to register U1SEG10 (write 00h for master mode, FFh for slave mode), before the LCD module is enabled.

17.1.4. Operation

For entering active mode, set flag LCD in register SR1. 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 registers UxSEGy and by validating it by a write access to register U1SEG10 (write 00h while in master mode, FFh 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 SR1 will immediately return all segment and backplane U-Ports to the output LOW state.

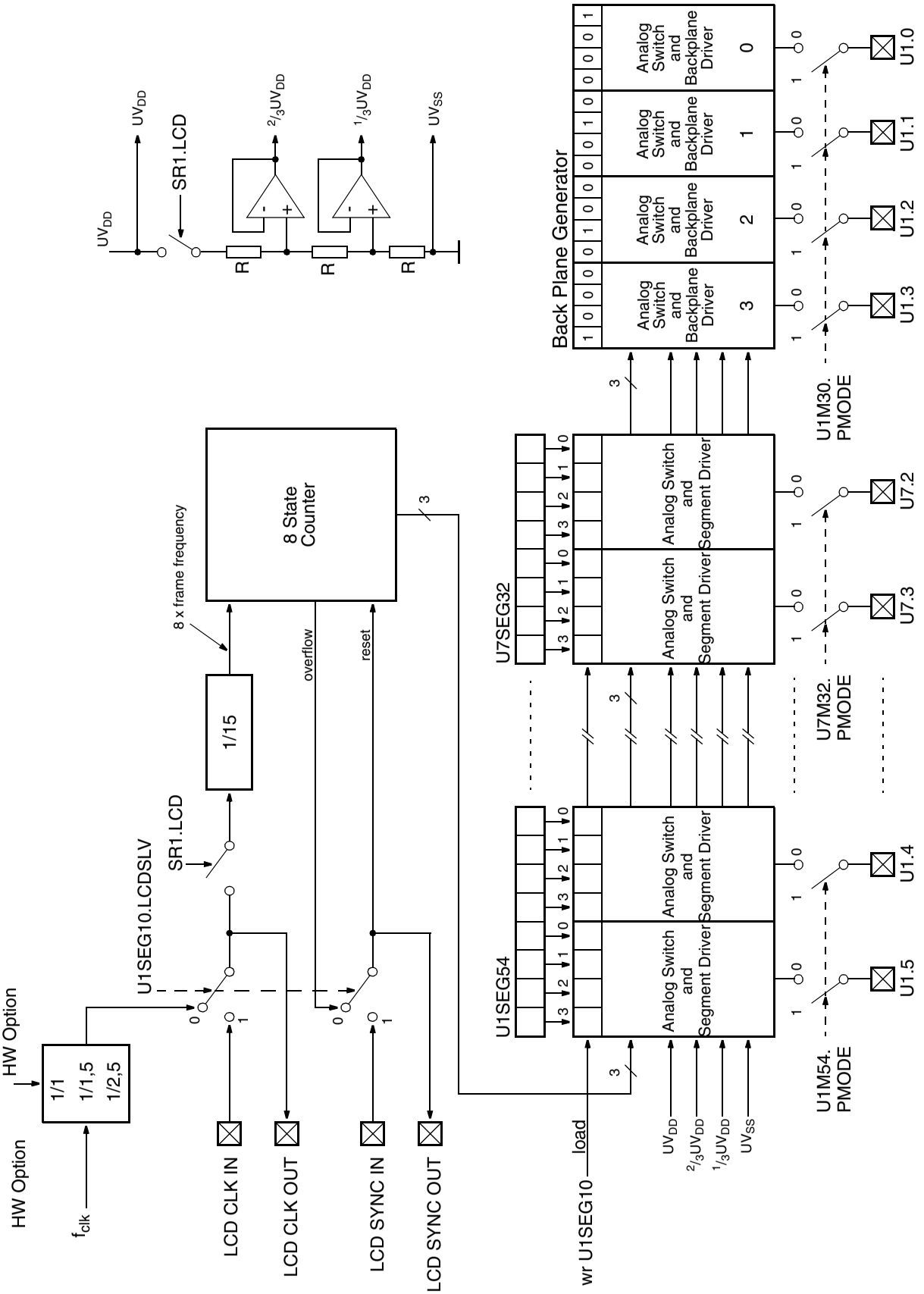


Fig. 17–2: Block Diagramm

The state of the segment registers is not readable.

All LCD operations are not affected by CPU SLOW mode.

17.1.5. 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 mode. Special signals provide phase and frequency synchronism for the LCD frame among the cascaded ICs.

For master mode, set flag LCDSLV in register U1SEG10 LOW. The module always directs signal LCD-SYNC-OUT to pins U1.4 and U6.0 and LCD-CLK-OUT to pins U1.5 and U6.1. They connect to external slave ICs' SYNC-IN and CLK-IN inputs for synchronization.

For slave mode, set flag LCDSLV in register U1SEG10 HIGH. Configure pins U5.2 and U5.3 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 17.3.

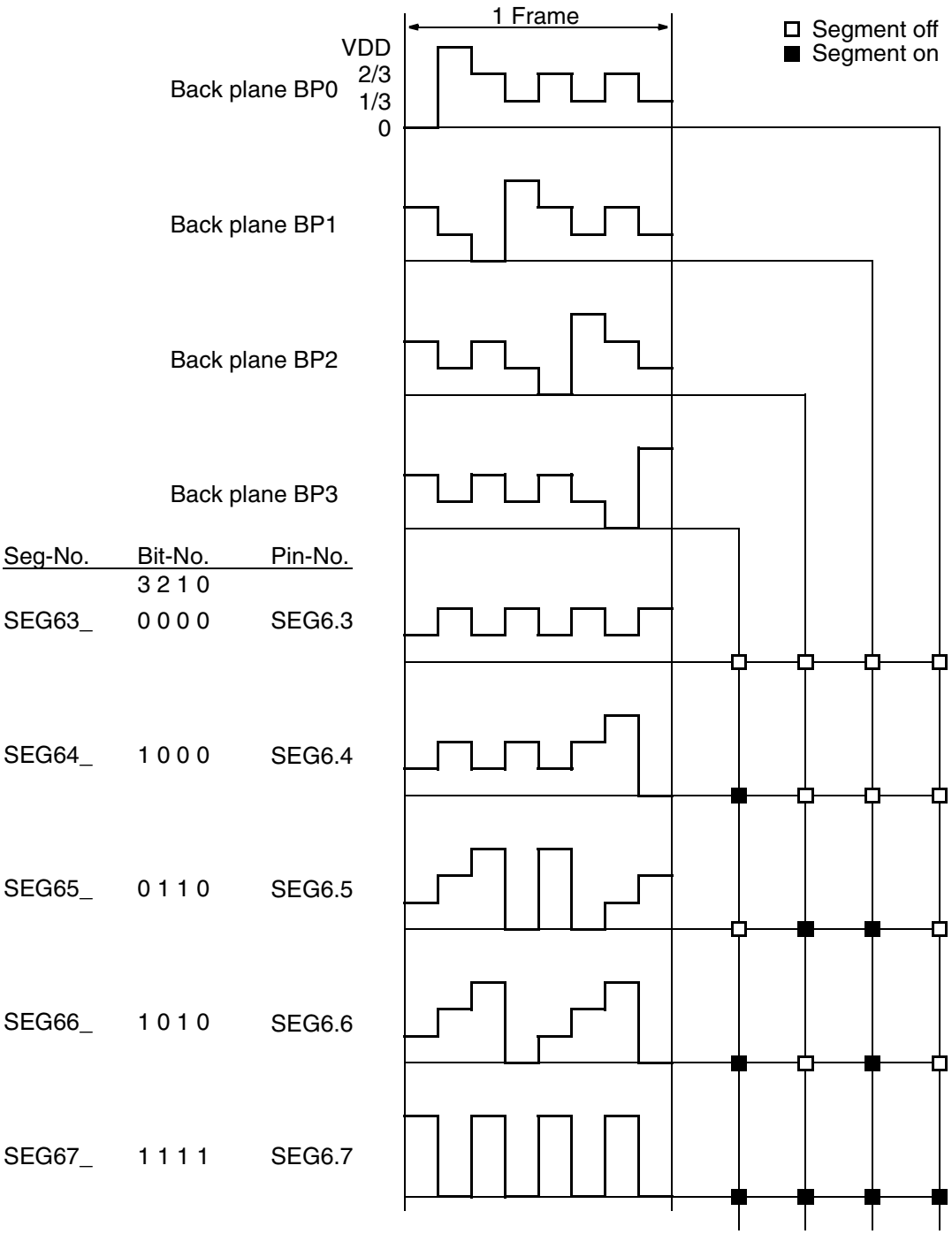


Fig. 17–3: Frame Timing Diagram

A segment at a crossing of backplane and segment lines is turned black when at the same time the backplane driver outputs a full swing and the segment driver outputs a full swing of opposite polarity.

17.2. Registers

U-Port registers U1SEG10, U1SEG32 and U1M30 play special roles during operation of the LCD module.

U1SEG10 Universal Port x Segment Register of Ux.1 and Ux.0

	7	6	5	4	3	2	1	0	
w	LCDSLV	x	x	x	x	x	x	x	LCD
	0	0	1	0	0	0	1	0	Res

LCDSLV LCD Module is Slave
w1: validate all segment information and select slave mode
w0: validate all segment information and select master mode

U1SEG54 Universal Port 1 Segment Register of Pin U1.5 and U1.4

	7	6	5	4	3	2	1	0	
w	SEG15.3	SEG15.2	SEG15.1	SEG15.0	SEG14.3	SEG14.2	SEG14.1	SEG14.0	LCD
	0	0	1	0	0	0	1	0	Res

Register U1SEG54 is an example for any Universal Port Segment Register with exception of U1SEG10 and U1SEG32. Refer to section Ports for detailed description.

SEG15.3 Segment # 3 of Pin U1.5
This bit defines the segment at the crossing of the lines SEG1.5 (Pin U1.5) and BP3 (U1.3).
w1: Segment is on (black)
w0: Segment is off (white)

SEG15.2 Segment # 2 of Pin U1.5
This bit defines the segment at the crossing of the lines SEG1.5 (Pin U1.5) and BP2 (U1.2).

SEG15.1 Segment # 1 of Pin U1.5
This bit defines the segment at the crossing of the lines SEG1.5 (Pin U1.5) and BP1 (U1.1).

SEG15.0 Segment # 0 of Pin U1.5
This bit defines the segment at the crossing of the lines SEG1.5 (Pin U1.5) and BP0 (U1.0).

17.3. Software Hints for Cascading LCD Modules

17.3.1. Power-On and Start-Up Procedure

1. The SW in master and slave configures the corresponding IC.

Table 17–1: 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.

17.3.2. Operation

In order to obtain optimum synchronization of LCD switchover, a change of display must be coordinated between master and slave (preferably via IPI) so that the time lag between write accesses to U1SEG10 of the master and of the slave is kept as small as possible. Suggestion: Lower ms range or customer specification.

17.3.3. 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 then output a low signal. The LCD display is now inactive.

18. DMA

The DMA module allows read and write access to an external IC with minimum CPU interaction.

The module 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 the 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.), are written and read by CPU operations supported and timed by the DMA module.

The CPU programs the DMA module to copy an array of data from the internal to the external IC's memory. The CPU writes to or reads from external registers while the DMA module generates the necessary timing and control signals.

Features

- 3 operating modes:
  direct memory write access (DMA) without CPU interaction,
  support and timing of CPU write access to ext. registers,
  support and timing of CPU read access to ext. registers
- 16 MB maximum DMA block size
- one byte DMA block alignment, no confinement to banks
- DMA sequence interruptible by CPU or interrupt controller
- CPU cycles are stolen only during CPU bus access
- flag for CPU-DMA conflict
- Interrupt on DMA transfer finished
- External bus cycle time selectable from Fxtal/2 to Fxtal/256
- Automatic generation of CPU wait states to support read access to external registers

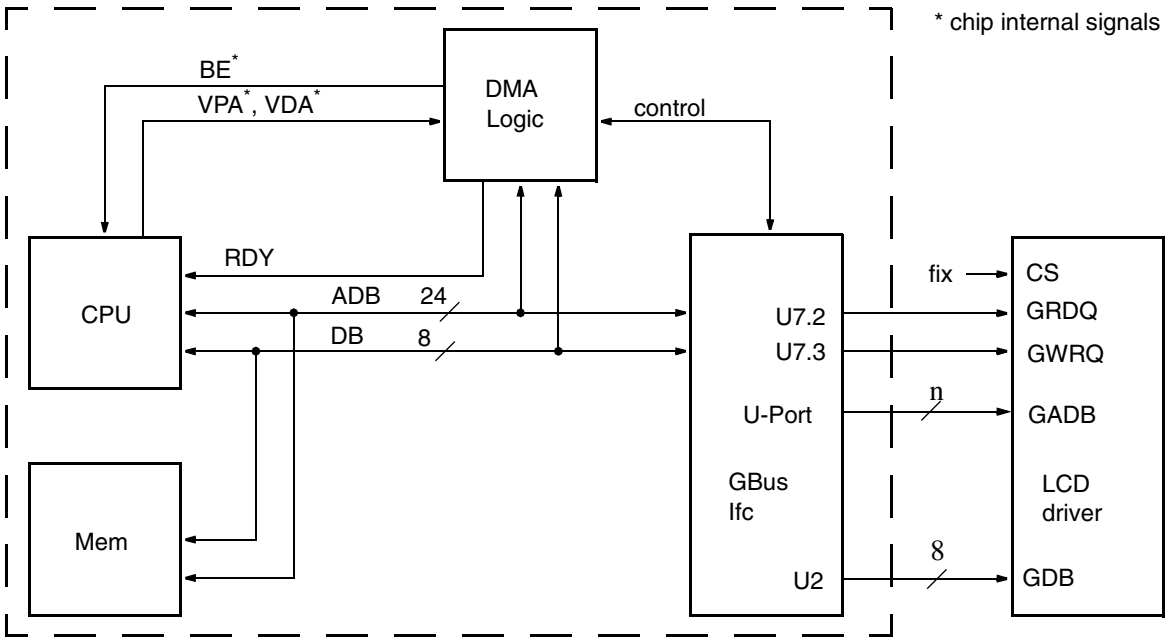


Fig. 18–1: Block Diagram

18.1. Principle of Operation

18.1.1. DMA Write Mode

To select the desired write timing, the corresponding bits have to be set in the DMA Initial Condition Register (DIC).

To obtain the 8-bit pixel data GDB7 .. 0 on the U2.7 .. U2.0 pins, all U2 bits have to be configured as port, normal, outputs. To obtain the GWRQ control output on U7.3, this port has to be configured as port, special, output. Other signals necessary to control the external IC have to be realized by software using other ports.

The range of internal addresses to be transmitted is finally set by first writing the 3 (lower) physical start address bytes to registers DSA2, DSA1, DSA0, then the 3 (upper) physical end address bytes to registers DEA2, DEA1 and DEA0.

The module generates the physical 24-bit address, as it is presented to the memory, even with alternative banking mode. Refer to the Memory Banking section for translation of logical addresses as generated by the CPU to physical addresses.

Upon writing DEA0 the DMA module will immediately begin presenting data on U2 and corresponding control signals on U7.3.

During the necessary DMA access cycles to the internal addresses a wait cycle is generated for the CPU whenever it tries to perform a bus cycle itself.

During operation, a DMA transfer active status bit (DCS.DTA) is readable from the DMA Control and Status register DCS to indicate that a requested transfer is not finished. A busy status bit (DCS.BSY) is readable to indicate that the DMA is currently active and not halted by the Interrupt Controller or by software. Other bits in the DCS are writable to immediately stop a running DMA sequence, and to restart it, or to allow the Interrupt Controller, when active, to stop and when inactive, to restart it.

Upon reaching the end address setting the DMA module finishes operation. At this time the DMA interrupt source output is triggered.

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.

18.1.2. CPU Write Mode

To select the desired write timing and to use U2 data writes as cycle trigger, the corresponding bits have to be set in the DMA Initial Condition Register (DIC).

To obtain the GWRQ control output on U7.3, this port has to be configured as port, special, output. Other signals necessary to control the external IC have to be realized by software using other ports.

The CPU configures and operates U2 as port, normal, output. Upon the CPU writing the U2 Data Register (U2D), the DMA module will present corresponding control signals for one write cycle to the external IC on U7.3.

If the CPU tries to rewrite U2D before the previous cycle is finished, the DMA module halts the CPU by generating the appropriate number of wait cycles.

During operation a busy status bit (BSY) in the DMA Control and Status Register (DCS) is readable to indicate the busy condition.

18.1.3. CPU Read Mode

To select the desired read timing and to use U2 data reads as cycle trigger the corresponding bits have to be set in the DMA Initial Condition Register (DIC).

To obtain the GRDQ control output on U7.2, this port has to be configured as port, special, output. Other signals necessary to control the external IC have to be realized by software using other ports.

The CPU configures and operates U2 as port, normal, input. Upon the CPU reading the U2 Data Input (U2D), the DMA module will present corresponding control signals for one read cycle from the external IC on U7.2.

The DMA module halts the CPU by generating the appropriate number of wait cycles, until the data read cycle from the external IC is complete. Thus one CPU read of U2D is sufficient to read data from external IC.

18.2. Registers

The DMA control registers are located in the I/O area.

Table 18–1: DMA Control Registers

Mnemonic	Name
DCS	DMA Control/Status
DIC	DMA Initial Configuration
DSA0	Start Address 0 (low byte)
DSA1	Start Address 1
DSA2	Start Address 2
DEA0	End Address. 0 (low byte)
DEA1	End Address. 1
DEA2	End Address. 2

The registers DSAx (start address) and DEAx (end address) are writable by the CPU. Write the address of the first byte to be transferred into the start address register. After the DMA has finished it points to the first byte after the transferred block. This makes it easy to transfer consecutive blocks without rewriting the start address register every single time. Only the end address has to be updated in this case. The end address has to be initialized with the address pointing to the first byte after the block to be transferred. Writing the low byte of the end address (DEA0) starts the DMA sequence. The first transfer starts after the next opcode fetch. The compare signal stops the DMA sequence if both pointers, DSAx and DEAx, point at the same memory location. A start or restart is not possible in this case.

The DMA Control and Status Register (DCS) shows the CPU whether a requested DMA transfer is not yet finished (DTA flag), whether a DMA transfer is active and not halted, or a CPU access is not yet finished (BSY flag), or if a DMA CPU conflict (DCC flag) has occurred. DCC is set true, if the data, mode or tristate register of U-Port 2 is addressed by the CPU though the DMA logic is active. DCC will not be set, if the respective registers of U-Port 7 are addressed. It allows the CPU to stop (interrupt) and start (continue) a DMA sequence. Furthermore, it lets the CPU define whether an interrupt may stall a DMA sequence.

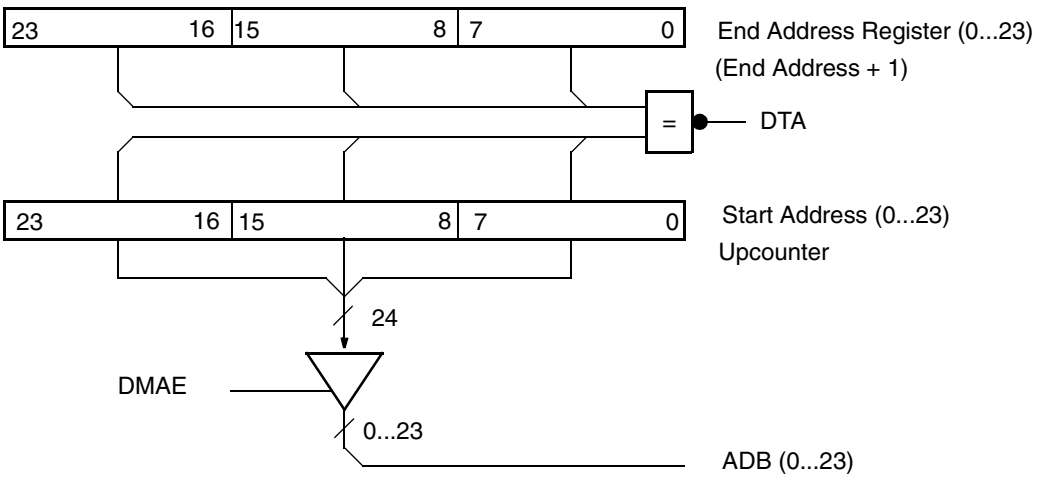


Fig. 18–2: DMA Address Generator

DCS	DMA Control and Status Register								
	7	6	5	4	3	2	1	0	Note
r/w	x	x	x	DTA	DSI	DCC	STP	BSY	
	0	0	0	0	0	0	0	0	Res

DTA **DMA Transfer Active**
r1: DMA transfer active.
r0: No DMA transfer active.

DSI **DMA Stopped by Interrupt**
r/w1: DMA stops during interrupt.
r/w0: DMA continues during interrupt.
DSI should only be modified when DTA is zero.

DCC **DMA CPU Conflict**
r1: DMA CPU conflict.
r0: No DMA CPU conflict.
w0: Clears DCC flag.

STP **Stop DMA sequence**
w1: Stops the DMA sequence.
w0: Nothing happens.

BSY **Busy**
r1: DMA or CPU sequence is active.
r0: DMA or CPU sequence is not active.
w1: Starts the DMA sequence.
w0: Nothing happens.

The two flags BSY and STP should be used in common. Table 18–2 shows the possible bit combinations.

Table 18–2: BSY and STP Usage

STP	BSY	
0	0	No action
0	1	Start DMA sequence
1	0	Stop DMA sequence
1	1	Not allowed

With BSY true, the CPU must neither access the ports or the DMA address registers, nor change the DMA Initial Configuration register (DIC). Even after interrupting a DMA sequence by setting the STP flag, it is necessary to guarantee that the BSY flag was cleared by the DMA logic, before changing those registers.

With DSI active a DMA sequence is stopped while an interrupt is served. In this case the DMA sequence doesn't lengthen an interrupt service routine. The start of an interrupt sends a stop signal to the DMA logic. Only the first of the nested interrupts sends this stop signal. The end of an interrupt (the last interrupt if they are nested) sends a start signal to the DMA logic. From this reason, each interrupt can start a DMA sequence if DSI is true. The only way to avoid a wrong start is to guarantee that the two address registers DSAx and DEAx point to the same memory location.

DSI active implies other restrictions to the user:
You can stop a DMA sequence within an interrupt, but it is not continued until the end of the last of nested interrupts. You can start a DMA sequence within an interrupt and it is not stopped by nested interrupts, until the main program is interrupted again.

Normally, a DMA sequence will be started by writing the end address to the appropriate registers. Writing to DEA0 starts the DMA sequence. So this byte has to be written last. If a DMA sequence is interrupted by SW, it has to be continued by rewriting DEA0 or by writing a one to BSY. A new DMA sequence may be started too by setting BSY, if the end address hasn't changed. In this case the start address has to be rewritten because it points at a position after the last transferred byte.

The DMA Initial Configuration Register (DIC) contains wait state stuff like the duration of external access cycles or whether wait states should be generated. If the flag WSA (Wait States Active) is cleared, the U-Port 2 may be used as normal I/O or LCD port. In this case no wait states are generated with CPU access but with DMA access. The bits WSA, WS0, WS1 and WS2 must not be modified if DTA or BSY are true.

DIC	DMA Initial Configuration Register							
	7	6	5	4	3	2	1	0
w	x	WS2	WS1	WS0	x	x	x	WSA
	0	0	0	0	0	0	0	0

Note: Res

WS2 to 0 Wait States Bit 2 to 0
Write only field for programming the number of wait states.

Table 18–3: Wait States

WS2 to 0	#WS	Ext. Bus Frequency	@ 10 MHz
0	2	Phi2/2	5 M (Reset)
1	4	Phi2/4	2.5 M
2	8	Phi2/8	1.25 M
3	16	Phi2/16	625 k
4	32	Phi2/32	312.5 k
5	64	Phi2/64	156.25 k
6	128	Phi2/128	78 k
7	256	Phi2/256	39 k

WSA Wait States Active
w1: Wait states at CPU access to U2 and generation of GWRQ and GRDQ.
w0: No wait states (RDY) and no control signals (GWRQ, GRDQ) at CPU access.
At DMA wait states are always generated.

The wait state logic controls the duration of external bus cycles (see Table 18–3 on page 122). It defines the maximum number of cycles the CPU is stopped. Not each kind of access stops the CPU the maximum number of cycles. Only a read from the external memory causes the wait state logic to generate the (programmable) maximum number of wait states. A write stops the CPU only if the previous access is not finished. A DMA sequence causes the CPU to stop for one Phi2 clock at each byte transfer. The next DMA happens after n-1 Phi2 clocks if n is the programmed number of wait states.

The timing in table 18–4 relate on a system clock of 10MHz. The external address has to be written by SW to the data latch. So the SW must guarantee the required address setup time. The GWRQ high to GWRQ low time ratio is symmetrical at DMA accesses. The SW has to guarantee the required GWRQ high time at CPU accesses. In this case the programmer can not rely on self timing with the RDY signal. At consecutive write accesses the GWRQ high time is exactly 2 Phi2 clocks.

Table 18–4: Bus timing at 10 MHz

#WS	2	4	8	16	32	64	128	256	Units
t_{WDS} min.	125	325	725	1525	3125	6325	12725	25525	ns
t_{ACC} max.	150	350	750	1550	3150	6350	12750	25550	ns
t_{DWH} max.	100	200	400	800	1600	3200	6400	12800	ns
t_{DDH} max.	50 (always constant 1/2 Phi2)								ns
t_{CWH} max.	200 (always constant 2 Phi2)								ns

t_{WDS}: Write data setup time
t_{ACC}: Read access time
t_{DWH}: DMA write high time
t_{DDH}: DMA write data hold time
t_{CWH}: CPU write high time

18.3. Ports

The access to external memory is done by universal ports. The assignment to external signals is shown in table 18–5.

Table 18–6 shows the settings of the port configuration registers in different modes.

Table 18–5: Port Assignment

Port	Name	
U2.0	GDB0	External data bus
:	:	
U2.7	GDB7	
1)	GADB	External address bus
U7.2	GRDQ	External read signal
U7.3	GWRQ	External write signal
1) Any universal port may be used as address output port.		

Table 18–6: Port Configurations

Mode	Register	Setting	
DMA Write, CPU Write	U2SEG10, 32, 54, 76	00H	Normal, out
	U2M10, 32, 54, 76	01H	Port mode
	U7SEG32	44H	Special, out
	U7M32	01H	Port mode
CPU Read	U2SEG10, 32, 54, 76	22H	Normal, in
	U2M10, 32, 54, 76	01H	Port mode
	U7SEG32	44H	Special, out
	U7M32	01H	Port mode

18.4. SW Application Hints

CPU must run in CPU FAST mode when accessing external memory. Don't try to access (CPU or DMA) in CPU SLOW mode. Port fast mode is recommended, to guarantee the timing between control signals and data.

Don't try to access the external memory or the involved ports while a DMA sequence is running (DCS.DTA=1).

The ports must be initialized depending on the kind of access (DMA, write, read). Don't reconfigure ports or addresses (DSAx, DEAx) as long as the flags DCS.DTA or DCS.BSY are true. This may cause problems, particularly if more than two wait states are programmed. The flag BSY will be set if the CPU accesses external addresses too. BSY will be cleared after the transfer cycle has finished.

All involved ports and external addresses may be read or written as long as there is no DMA working (DTA=0, BSY =0). If a DMA is working (DTA=1) or a CPU access to an external address is not finished (BSY=1), it is neither allowed to access the involved ports and the DMA start and end address pointers, nor to change the configuration of the wait state logic. If there is an access, this is a programming error. Both accesses (DMA and CPU) are disturbed. A flag signals the occurrence of this conflict for debugging purposes.

There are two different modes to operate the DMA logic. In the DMA high priority mode a DMA sequence is not affected by an interrupt. DMAs take place during an interrupt service routine. Each DMA steals one cycle of the ISR. In the DMA low priority mode, a DMA sequence is stalled by an interrupt and continues after the end of the ISR. The ISR is not lengthened by an active DMA sequence.

18.4.1. DMA High Priority Mode

The flag DSI is, and remains, set to zero. A DMA may interrupt an interrupt service routine. A DMA sequence may be started, stopped, the registers and ports may be modified everywhere if you make sure that the flags BSY/DTA are false.

The overall initialization has to be done once after each reset.

Overall initialization:

– Configure WSA, WS0, WS1 and WS2. Set DSI to zero.
– Switch data port U2.0 to U2.7 to normal mode.
– Switch address port to normal mode and activate output driver.
– Switch control port U7.2 and U7.3 to special mode and activate output driver.

DMA write:

– Activate output driver of data port U2.0 to U2.7.
– Write destination address to address port.
– Write start address to registers DSA2 to DSA0.
– Write end address + 1 to registers DEA2 to DEA0. Writing to register DEA0 starts the DMA sequence.

CPU write:

– Activate output driver of data port U2.0 to U2.7.
– Write external address to address port.
– Write to data port U2.

CPU read:

– Deactivate output driver of data port U2.0 to U2.7.
– Write external address to address port.
– Read from data port U2.

18.4.2. DMA Low Priority Mode

With the flag DSI set to one, any interrupt will stop a DMA sequence. The user may modify the DMA logic setting within an ISR as long as flag DTA is zero. If you want to stop a running DMA sequence by setting flag STP, it is advisable to clear flag DSI with the same write access to DSC. Otherwise the next interrupt would restart this interrupted DMA sequence.

The overall initialization has to be done once after each reset.

Overall initialization:

– Configure WSA, WS0, WS1 and WS2. Set DSI to one.
– Switch data port U2.0 to U2.7 to normal mode.
– Switch address port to normal mode and activate output driver.
– Switch control port U7.2 and U7.3 to special mode and activate output driver.

DMA write:

– Activate output driver of data port U2.0 to U2.7.
– Write destination address to address port.
– Clear DSI.
– Write start address to registers DSA2 to DSA0.
– Write end address + 1 to registers DEA2 to DEA0. Writing to register DEA0 starts the DMA sequence.
– Set DSI.

CPU write:

– Activate output driver of data port U2.0 to U2.7.
– Write external address to address port.
– Write to data port U2.

CPU read:

– Deactivate output driver of data port U2.0 to U2.7.
– Write external address to address port.
– Read from data port U2.

18.4.2.1. Unwanted DMA Interrupts in Low Priority Mode

On leaving interrupt service in Low Priority Mode (DSI=1), the DMA-HW generates a compare of start and end address register. If they are not equal, a halted DMA sequence is continued. But if they are equal, a DMA interrupt is generated instantly. DMA interrupts are generated at every compare, if the result is equal. Thus, an unwanted DMA interrupt is generated every time the DSI function tries to restart a DMA that has no transfers pending.

To work around these unwanted interrupts, the following measures should be taken:

– No special measures have to be taken if a new DMA sequence is initiated within the DMA ISR (Interrupt Service Routine).
– If no new DMA sequence has to be initiated within a DMA ISR, it is necessary to either clear flag DSI (High Priority Mode), or to disable the DMA interrupt at the Interrupt Controller, within the DMA ISR.
– After starting a DMA sequence from within the main routine, DSI has to be set and/or DMA interrupt has to be enabled again.

18.5. Timings

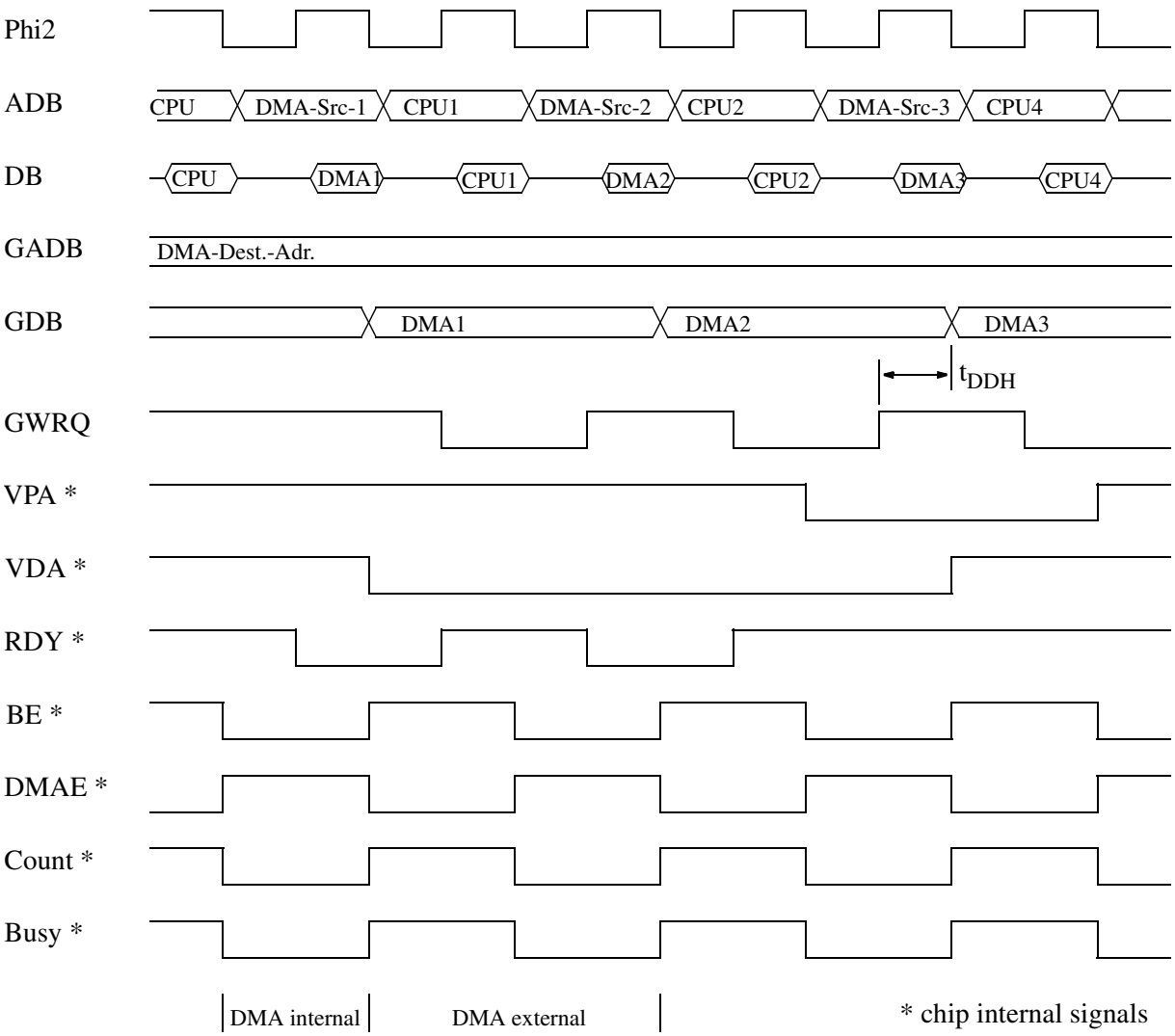


Fig. 18–3: DMA Write, WS = 2

Every second cycle is stolen from the CPU. The external DMA transfer lasts 2 cycles. The CPU computes an internal operation at the same time with the third DMA. In this case the CPU is not stopped. CPU3 is not visible on the buses ADB or DB.

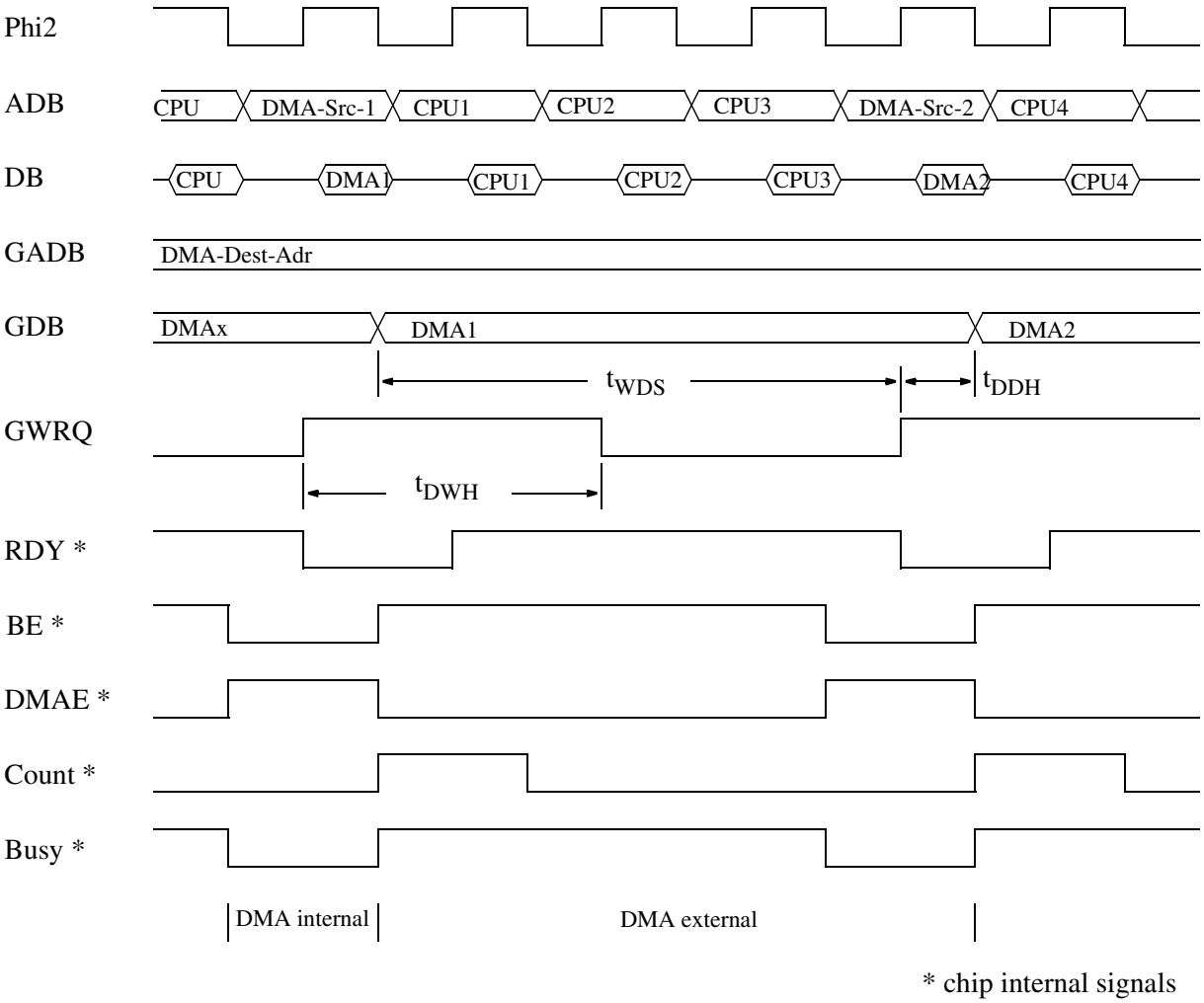


Fig. 18–4: DMA Write, WS = 4

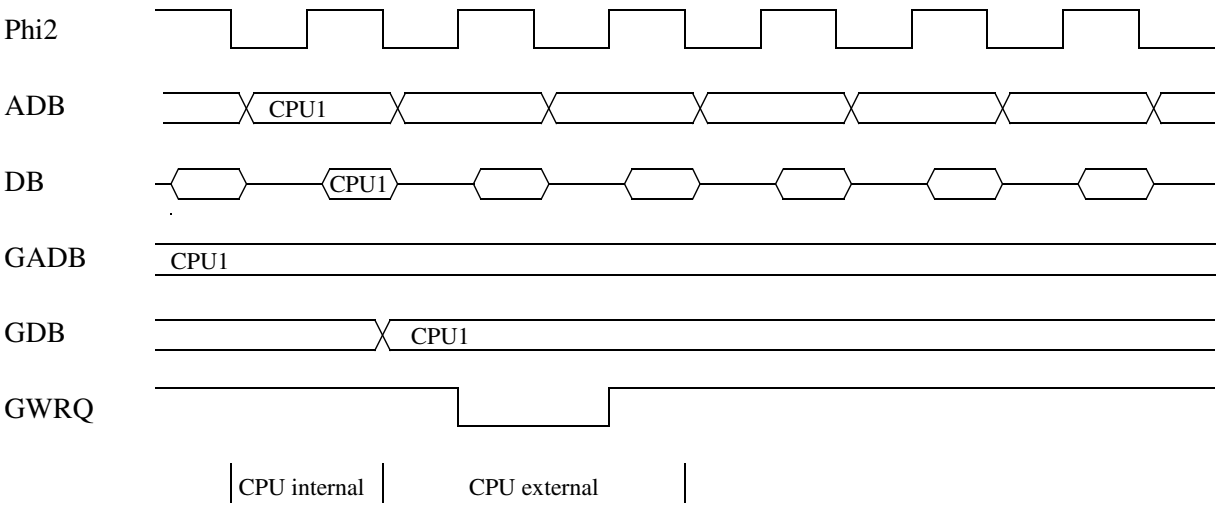


Fig. 18–5: CPU Write, WS = 2

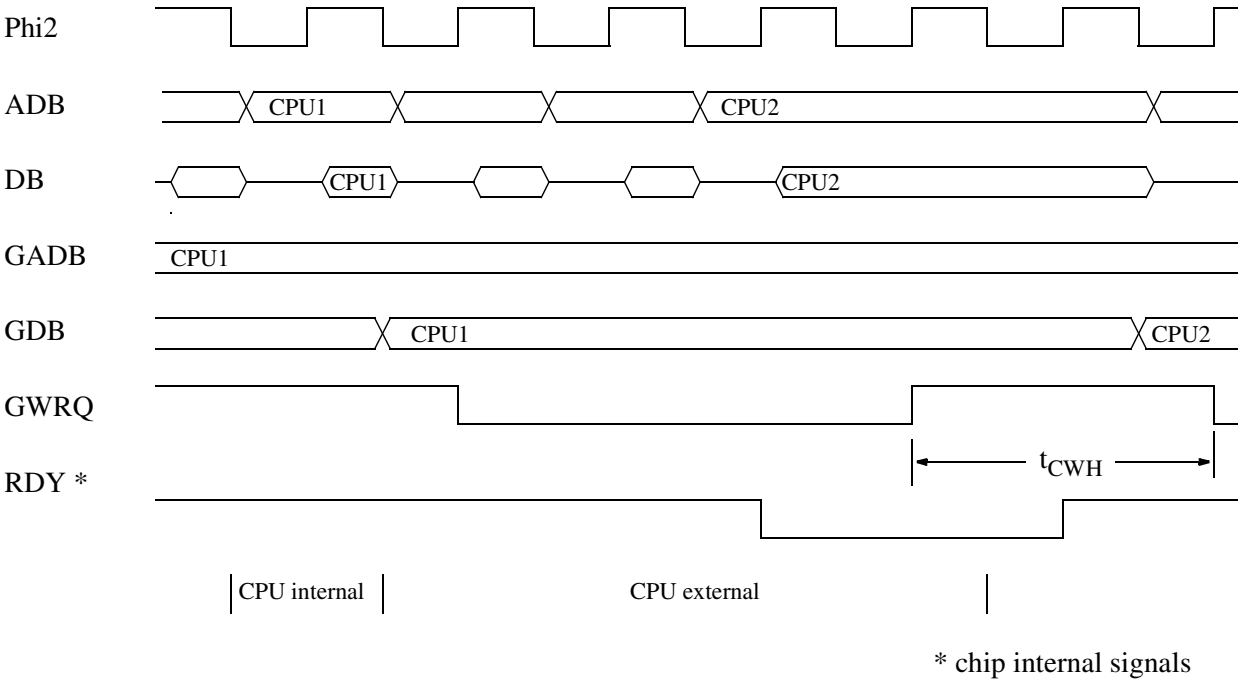


Fig. 18–6: CPU Write, WS = 4, with RDY because CPU rewrites to fast.

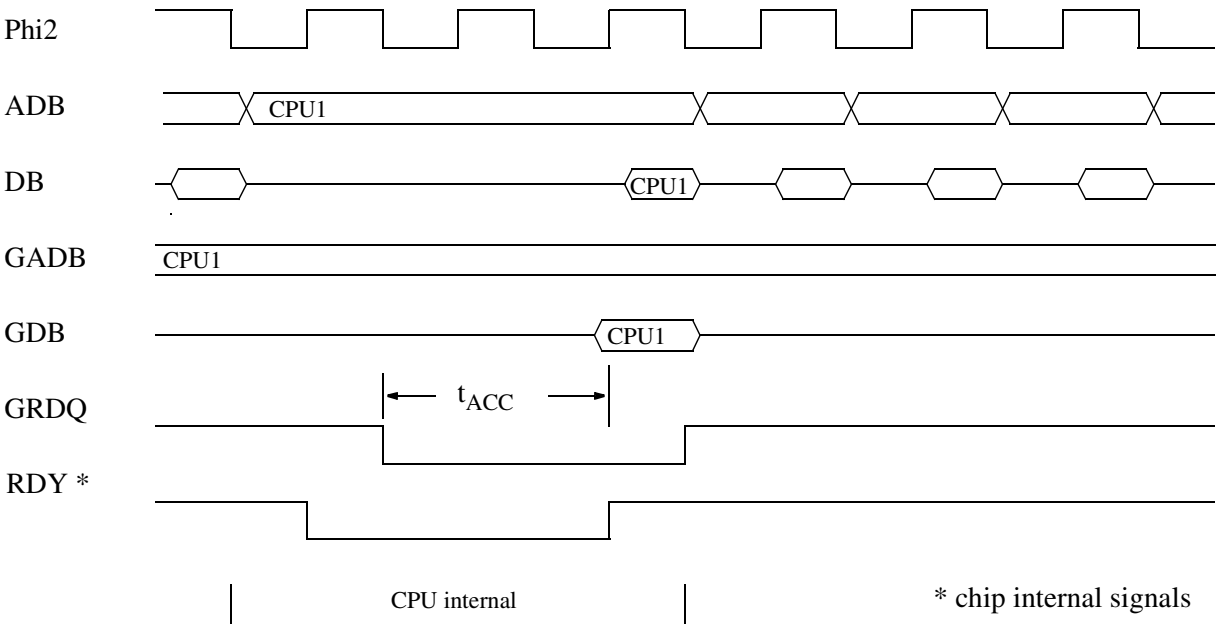


Fig. 18–7: CPU Read, WS = 2

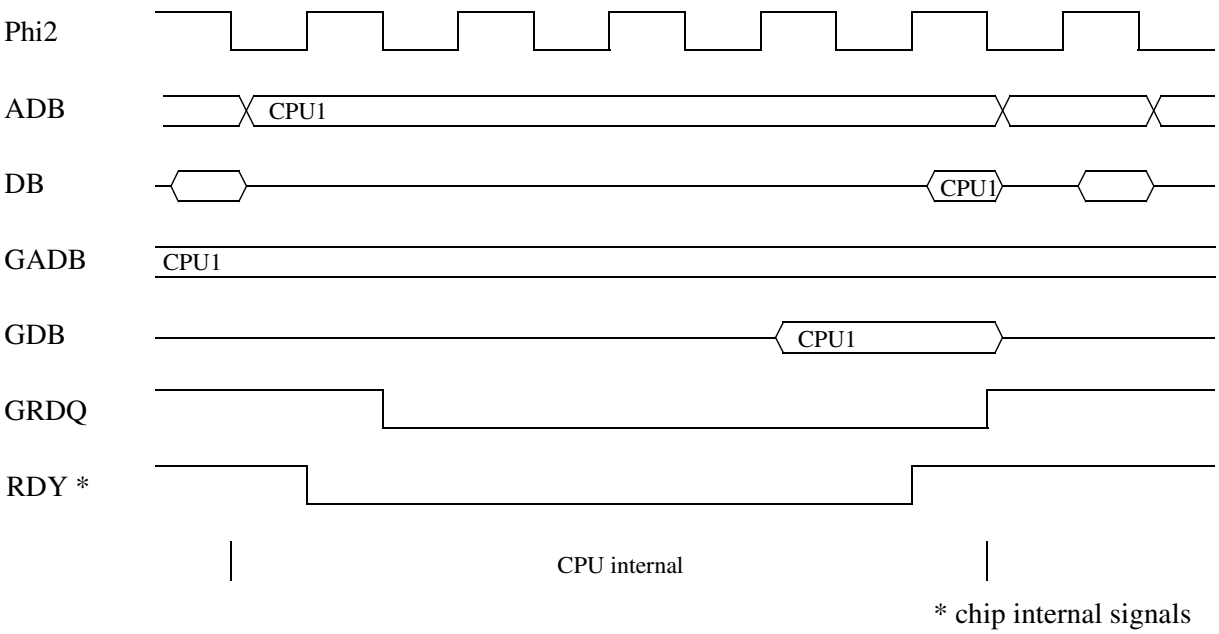


Fig. 18–8: CPU Read, WS = 4

19. Serial Synchronous Peripheral Interface (SPI)

An SPI module provides a serial input and output link to external hardware. An 8 or 9-bit data frame can be transmitted synchronized to an internally or externally generated clock.

The number of SPIs implemented is given in Table 19–1. The “x” in register names distinguishes the module number.

Features

- 8 or 9-bit frames
- Internal or external clock
- Programmable data valid edge
- Three internal clock sources programmable
- Input deglitcher for clock and data

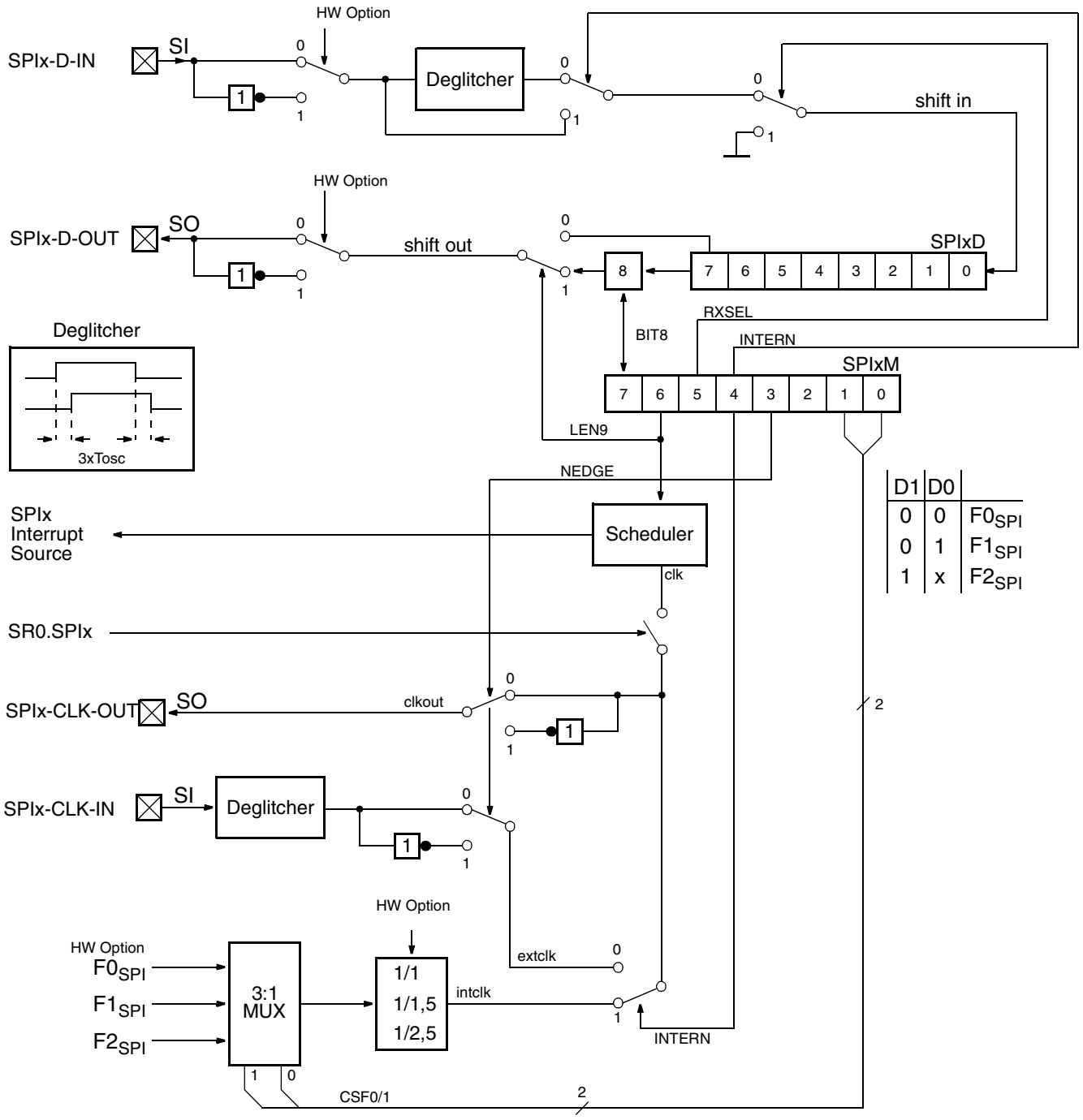


Fig. 19–1: Block Diagram

19.1. Principle of Operation

19.1.1. General Remarks

A 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 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.

SPIs are not affected by CPU SLOW mode.

19.1.2. Hardware settings

Clock frequency settings and the polarity of the data connections of the SPIs can be set via HW Options (Table 19–1). Refer to “HW Options” for setting them.

19.1.3. 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 19–1). Refer to “Ports” for details.

For entering active mode of a SPI, set the respective enable bit (Table 19–1).

Prior to operation, the desired clock frequency and telegram length have to be selected.

19.1.3.1. 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 or divided by 1.5 or 2.5. This selection is done by HW Option too. The shift clock is output by signal SPIx-CLK-OUT which can be inverted by the flag NEDGE of register SPIxM.

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 and can be inverted by the flag NEDGE.

The data valid edge of the clock is defined by flag NEDGE in register SPIxM.

Table 19–1: Module specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
All SPIs	F0SPI clock	FFAFh			
	F1SPI clock	FFB0h			
	F2SPI clock	FFB1h			
SPI0	D in inversion	FFAFh	SPI0-D-IN input	U6.5 special in	SR0. SPI0
	D out inversion	FFAFh	SPI0-D-OUT output	U6.4 special out	
	Prescaler	FFAEh	SPI0-CLK-IN input	U6.3 special in	
			SPI0-CLK-OUT output	U6.3 special out	
SPI1	D in inversion	FFB0h	SPI1-D-IN input	U3.1 special in	SR0. SPI1
	D out inversion	FFB0h	SPI1-D-OUT output	U3.0 special out	
	Prescaler	FFAEh	SPI1-CLK-IN input	U3.5 special in	
			SPI1-CLK-OUT output	U3.5 special out	

19.1.3.2. 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.

19.1.4. Operation

19.1.4.1. 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. 19–2, 19–3). At the end of the frame, an interrupt source signal is generated which may be selected to trigger an interrupt.

19.1.4.2. 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.

19.1.5. Inactivation

Returning an SPI module to standby mode by resetting its respective enable bit (Table 19–1) will immediately terminate any running receive or transmit operation and will reset all internal registers.

19.1.6. 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, please make sure that the external clock does not start until after SPIxD has been written, otherwise correct data transfer is not be guaranteed.

Attention must be paid to the neutral level of the external clock. Neutral level is defined high when data are valid on the rising clock edge. Neutral level is low otherwise.

19.2. 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	NEDGE	x	CSF		
	0	0	0	0	0	0	0	0	Res

BIT8 **Bit 8 of Rx/Tx Data**
r/w: Rx/Tx data bit.

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.

NEDGE **Negative Edge Selection**
r/w0: Data valid at rising edge.
r/w1: Data valid at falling edge.

CSF **Clock Selection Field**
wr: Source of internal clock (Table 19–2)

Table 19–2: CSF usage

Bits 1 0	Source of internal clock
0 0	F0SPI
0 1	F1SPI
1 x	F2SPI

19.3. Timing

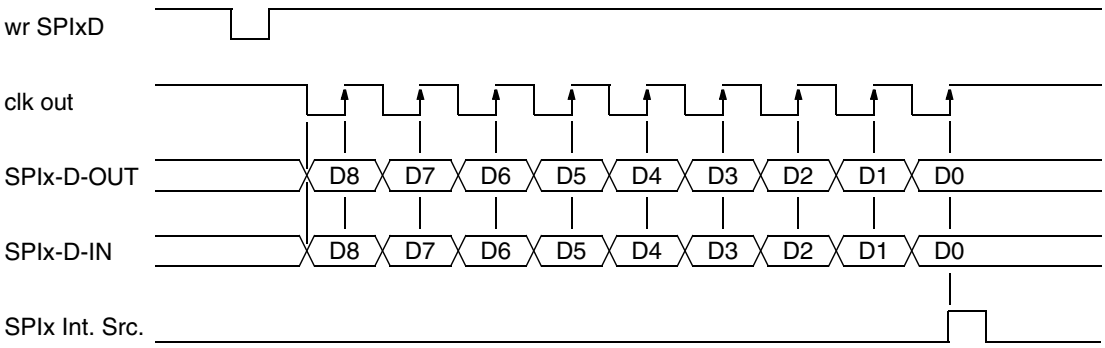


Fig. 19–2: Nine bit frame. Data valid at rising edge.

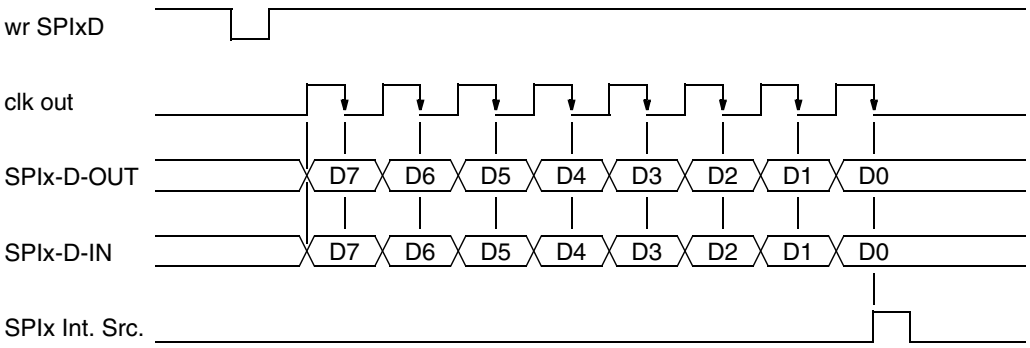


Fig. 19–3: Eight bit frame. Data valid at falling edge.

20. 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 20–1. The “x” in register names distinguishes the module number.

Features

- 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.

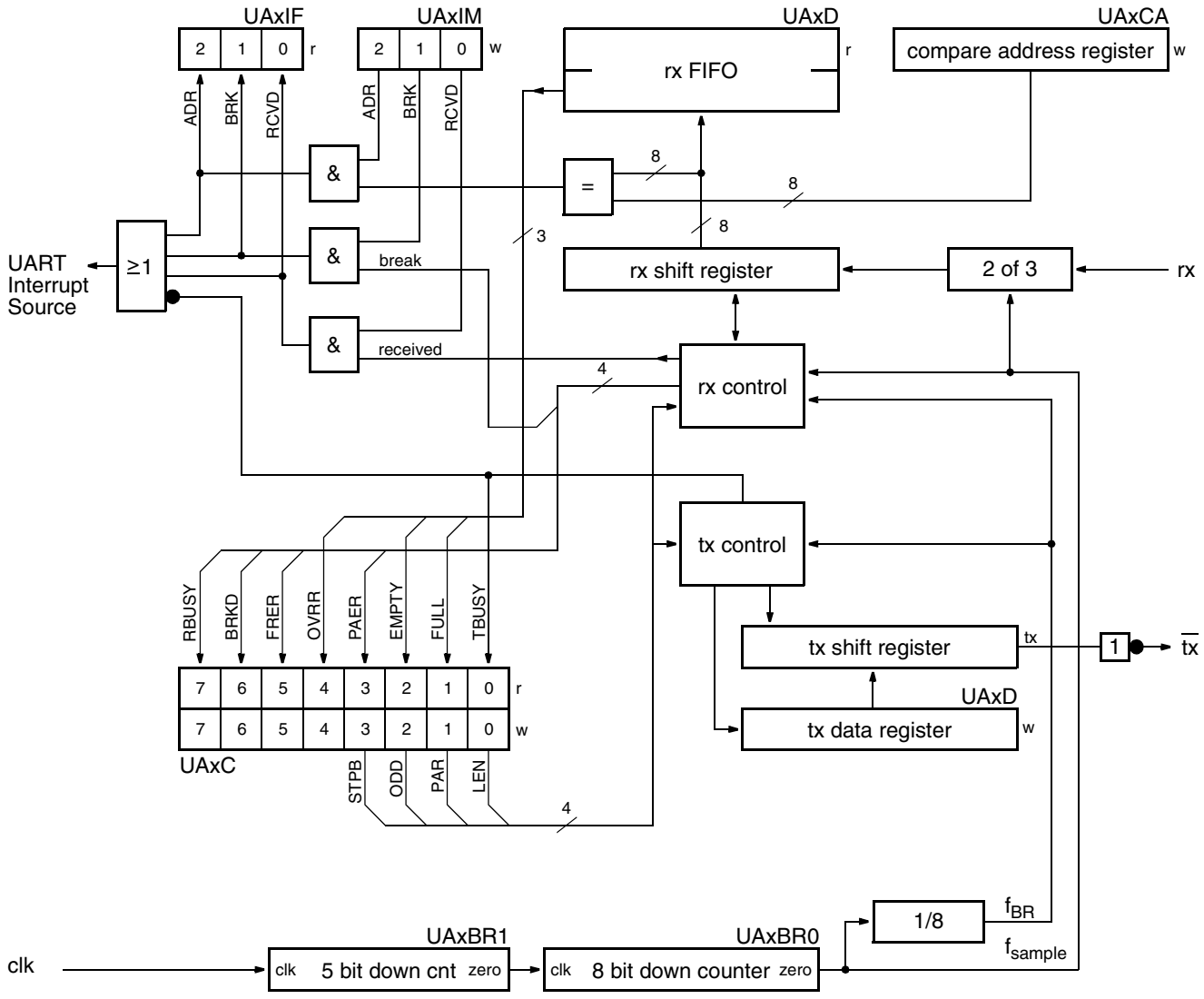


Fig. 20–1: Block Diagram

20.1. Principle of Operation

20.1.1. General Remarks

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.

A UART module is not affected by CPU SLOW mode.

A UART module is only capable of receiving telegrams that differ by no more than ± 2.5% from its own baud rate setting.

20.1.2. Hardware settings

The polarity of most RX and TX connections of the UART can be set via HW Options (See table 20–1 and figure 20–2). Refer to “HW Options” for setting them.

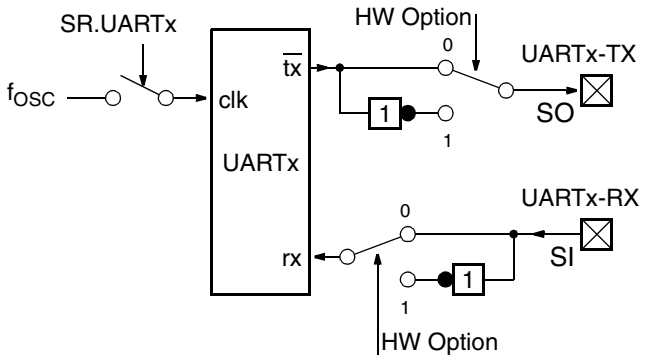


Fig. 20–2: Context Diagram

20.1.3.2. 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 20–3 on page 138).

20.1.3. Initialization

After reset, a UART is in standby mode (inactive).

Prior to entering active mode, the U-Ports assigned to function as RX input and TX output have to be properly SW configurated (Table 20–1). The RX port has to be configured Special In and the TX port has to be configured Special Out. Refer to “Ports” for details.

For entering active mode of a UART, set the respective enable bit (Table 20–1).

Prior to operation, the desired baud rate, telegram format, compare address and interrupt source configuration have to be done.

20.1.3.1. 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 13bit setting. Write UAxBR0 first, UAxBR1 last.

The baud rate generator is a 13-bit down-counter which is clocked by f_{OSC}. It generates the sample frequency:

$$f_{sample} = \frac{f_{OSC}}{\text{Value of Baud Rate Registers} + 1}$$

Its output frequency f_{sample} is divided by eight to generate the baud rate (bit/second).

$$BR = \frac{f_{OSC}}{(\text{Value of Baud Rate Registers} + 1) \times 8} = \frac{f_{sample}}{8}$$

$$\text{Value of Baud Rate Registers} = \frac{f_{OSC}}{BR \times 8} - 1$$

Table 20–1: Module-specific settings

Module Name	HW Options		Initialization		Enable Bit
	Item	Address	Item	Setting	
UART0	RX inversion	FFB4h	UART0-RX input	U4.4 special in	SR1.UART0
	TX inversion	FFB4h	UART0-TX output	U4.5 special out	
UART1	RX inversion	FFB5h	UART1-RX input	U5.7 special in	SR2.UART1
	TX inversion	FFB5h	UART1-TX output	U5.4 special out	
UART2	RX inversion	FFB4h	UART2-RX input	U4.2 special in	SR0.UART2
	TX inversion	FFB4h	UART2-TX output	U4.3 special out	

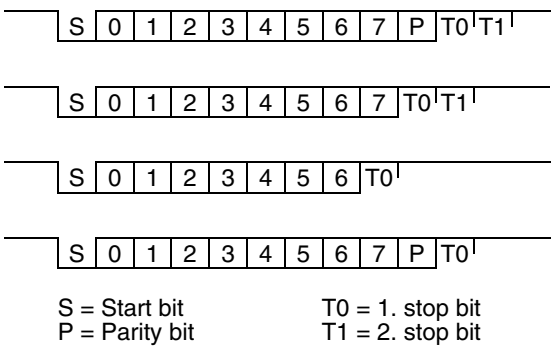


Fig. 20–3: Examples of Telegram Formats

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 20–2: 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.

20.1.3.3. Compare Address

The content of the Compare Address register UAxCA is compared with each received telegram. If they 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.

20.1.3.4. 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.

20.1.4. Operation

With proper HW configuration and SW initialization, a UART module is ready to transmit and receive telegrams in the selected format.

20.1.4.1. Transmit

A write access to UART Data register UAxD immediately loads the transmit shift register and starts transmission by sending the start bit. The flag TBUSY in register UAxD 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.

20.1.4.2. 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 reset. 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).

20.1.4.3. 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 have been received but the SW has 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.

20.1.5. Inactivation

Returning a UART module to standby mode by resetting its respective enable bit (Table 20–1) will immediately terminate any running receive or transmit operation and will reset all internal registers.

20.2. Timing

The duration of a telegram results from the total telegram length in bits (L_{TG}) (see Table 20–3 on page 138) 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. 20–4 and 20–5). 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.

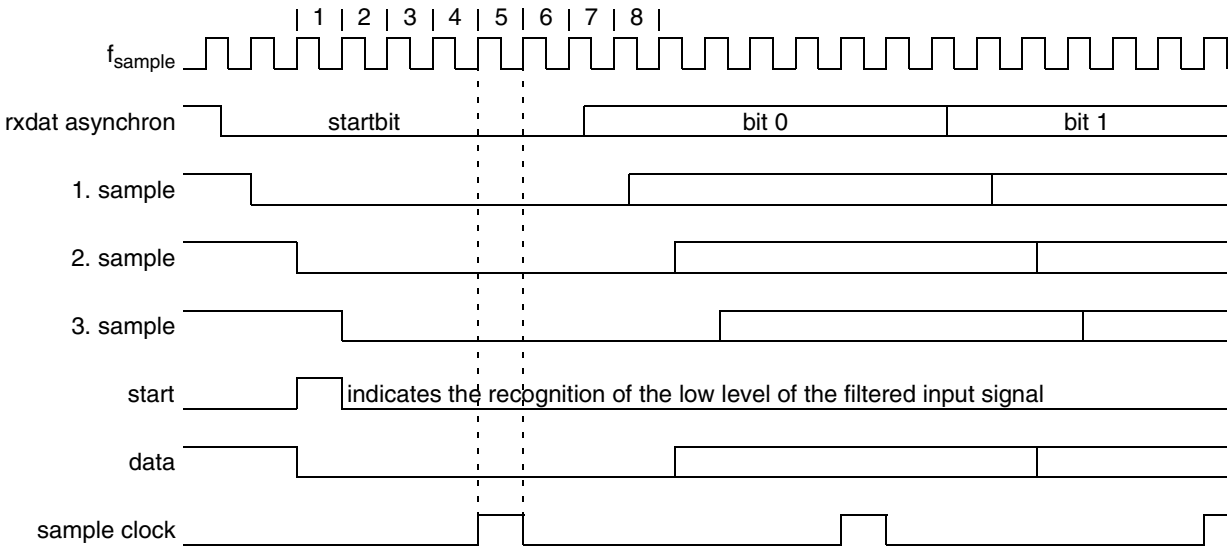


Fig. 20–4: Start of Telegram

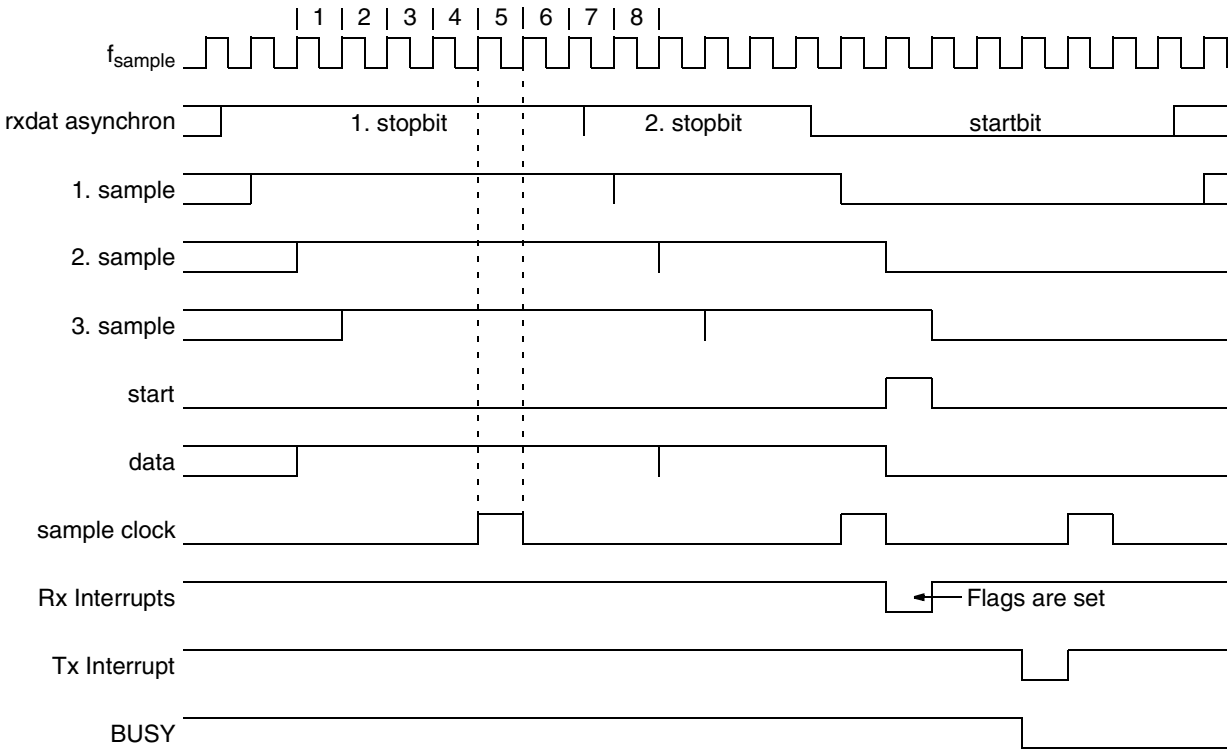


Fig. 20–5: End of Telegram

20.3. 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: 7-bit frame.
w1: 8-bit frame.

Table 20–3: 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.

21. 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: at least 16 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.

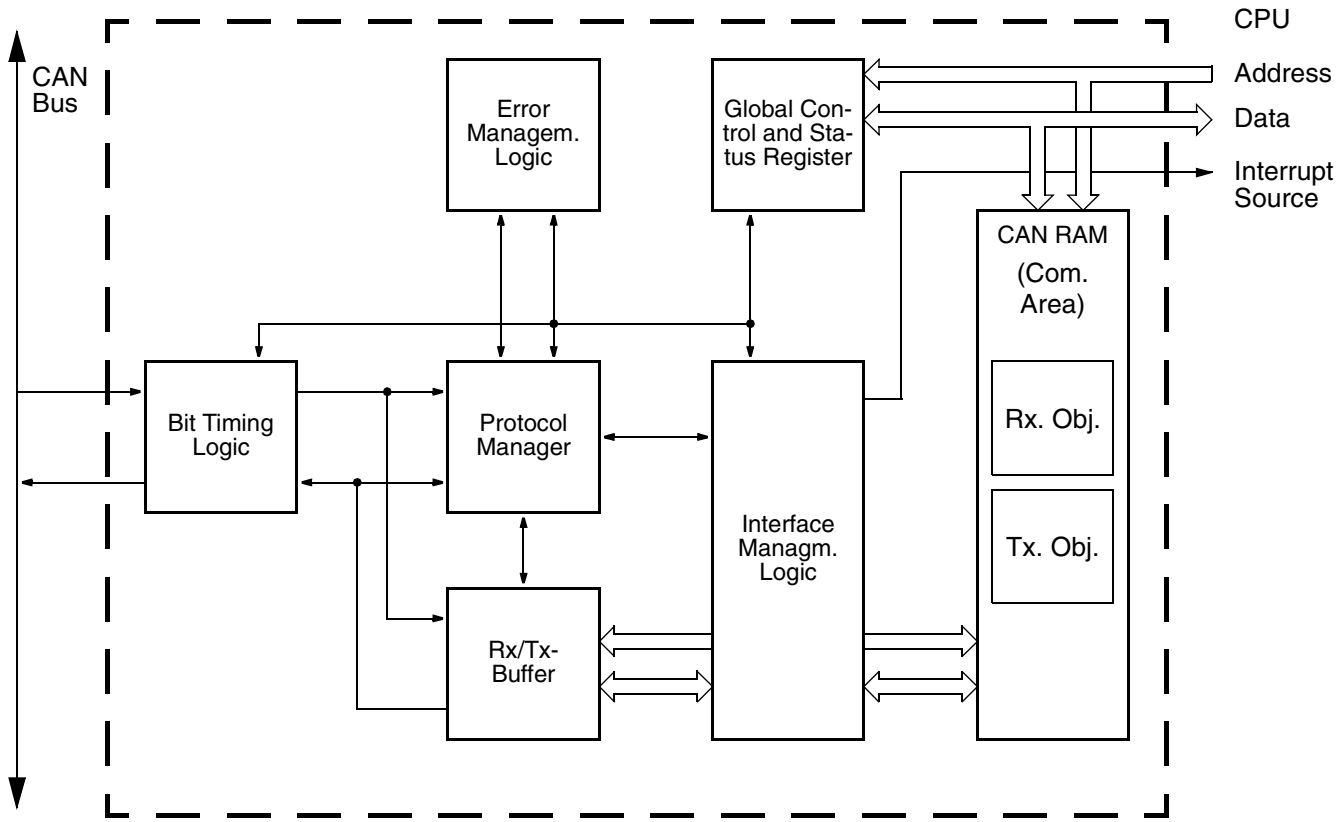


Fig. 21–1: Block diagram of the CAN bus interface

21.1. 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

21.2. Functional Description

21.2.1. 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 4–2 on page 35).

21.2.2. 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 Com. 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 byte. 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 Com. Area.

21.2.3. 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.

Global Control and Status

0	Control	CTR
	Status	STR
	Error Status	ESTR
	Interrupt Index	IDX
	ID Mask 28 ... 21	IDM
	ID Mask 20 ... 13	
	ID Mask 12 ... 5	
	ID Mask 4 ... 0	
	Bit Timing 1	BT1
	Bit Timing 2	BT2
	Bit Timing 3	BT3
	Input Control	ICR
	Output Control	OCR
	Transmit Error Counter	TEC
	Receive Error Counter	REC
15	Error Status Mask	ESM
16	Capture Timer low	CTIM
17	Capture Timer high	

Communication Area

0	Control	Com.-Obj. 1
1	ID 28 ... 21	
2	ID 20 ...13	Telegram Descriptor TD
	ID 12 ... 5	
	ID 4 ... 0 and Control	
	DLC and Control	
	Data 0	
	Data 1	
	Data 2	
	Data 3	
	Data 4	
	Data 5	
	Data 6	
	Data 7	
	Time Stamp low	
15	Time Stamp high	
16	TD	Com.-Obj. 2
	Data and Time Stamp	
31		
32	TD	Com.-Obj. 3
	Data and Time Stamp	
47		
n*16	TD	Com.-Obj. n
	Data and Time Stamp	
n*16+15		
(n+1)*16	Control: CM = 7	End of Com. Area

Fig. 21–2: 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.
r/w1: Rescan.
The microprocessor can set this flag in order to initiate a transmit telegram search at the beginning of the Com. 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 apropriate 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 stuff error.
r1: Stuff 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 has been the cause of an interrupt, the interrupt index points to the corresponding telegram descriptor (CANxIDX = 0..253). If an error has been 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 29 to 21								low
r/w	Identifier Mask Bits 20 to 13								
r/w	Identifier Mask Bits 12 to 5								
r/w	Identifier Mask Bits 4 to 0					x	x	x	high
	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	BPR	BPR	BPR	BPR	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: Pre-scaler value.
The baud rate pre-scaler sets the length of a time quantum for the bit timing logic.

$t_Q = t_{XTAL} \times (BPR + 1)$.

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: $t_Q = t_{XTAL}$
1: $t_Q = t_{XTAL} \times 2$
2: $t_Q = t_{XTAL} \times 3$
3: $t_Q = t_{XTAL} \times 4$ etc.

CANxBT2 Bit Timing Register 2

	7	6	5	4	3	2	1	0	
r/w	rsvd	TSEG2	TSEG2	TSEG2	TSEG1	TSEG1	TSEG1	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	SJW	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). Values greater than 4 must not be used.

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	0	0	0	Res

XREF **External Reference**
r/w0: The internal reference is used.
r/w1: The external reference is used where available.

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 7 to 0								low
r	Timer Bit 15 to 8								high
	0	0	0	0	0	0	0	0	Res

The Capture Timer is incremented with a clock pulse derived from the CAN bus. As 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).

21.2.4. Communication Area (CA)

The CA is located in the CAN-RAM. It consists of com. 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 into the CA in order of priority . This starts with the highest priority (the lowest identifier). The identifier defines the priority of a Tg. If the first eleven bits of an ext. Tg are the same as the identifier of a std. Tg, the Tg with standard identifier has higher priority.

21.2.4.1. Telegram Descriptor (TD)

The telegram descriptor is 6 bytes (TD0 to TD5) long and forms the beginning of a CO. Telegrams with std. and ext. identifiers have different TDs. They differ only in the length of the identifiers. 18 bits are therefore not allocated in the TD of a std. Tg. They cannot be used by the application because they are overwritten by the reception of a Tg.

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. 21–3: 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 has been 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 has been 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 Tg 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 com. 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.

21.2.4.2. 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.

21.2.4.3. 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 Tg 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. 21–4: Time stamp

21.3. Application Notes

21.3.1. 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 21–1). The RX port has to be configured Special In and the TX port has to be configured Special Out. Refer to “Ports” for details.

For entering active mode of a CAN, set the respective enable bit (Table 21–1).

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 ter-

Table 21–1: Module-specific settings

Module Name	Initialization		Enable Bit
	Item	Setting	
CAN0	CAN0-RX input	U6.6 special in	SR0.CAN0
	CAN0-TX output	U6.7 special out	
CAN1	CAN1-RX input	U1.6 special in	SR3.CAN1
	CAN1-TX output	U1.7 special out	
CAN2	CAN2-RX input	U4.0 special in	SR3.CAN2
	CAN2-TX output	U4.1 special out	

minate 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.

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.

21.3.2. Handling the COs

21.3.2.1. 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 wants to access a com. object, 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. 21–5: 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. 21–6: 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.

21.3.2.2. 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.

21.3.2.3. 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.

21.3.2.4. 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.

21.3.2.5. 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.

21.3.2.6. 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.

21.3.2.7. 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 then ensures 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.

21.3.2.8. 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.

21.3.2.9. 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.

21.3.3. 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.

21.3.4. 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.

21.3.5. 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.

21.3.6. 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.

21.3.7. 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.

21.3.7.1. 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.

21.3.7.2. 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, one uses the built-in acceptance filtering and sets up a CO specifically 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.

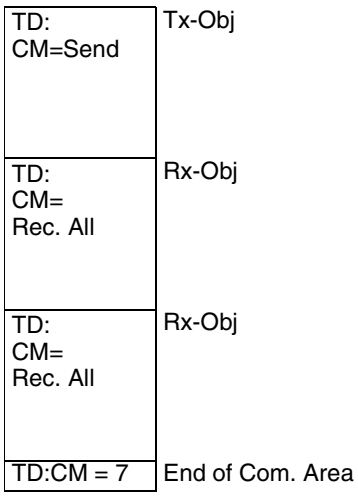


Fig. 21–7: Example: CA of a BasicCAN with 2 Rx-buffers

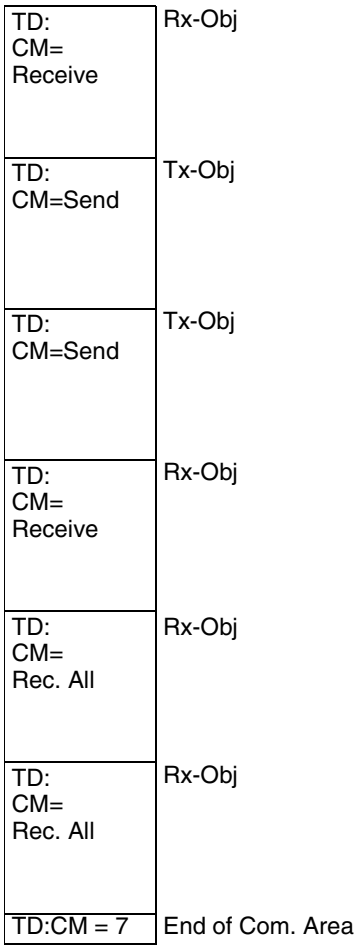


Fig. 21–8: Example: CA of a FullCAN with 2 Rx-objects, 2 Tx-objects, and 2 Rx-buffers

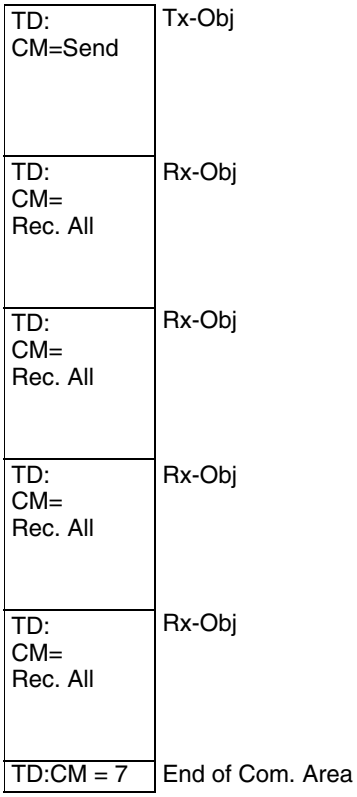


Fig. 21–9: Example: CA of a BasicCAN with 4 Rx-buffers

21.3.7.3. Bus Monitor

With some Rx-All-COs it is possible to construct a user-friendly bus monitor. The CPU has merely 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.

21.3.7.4. 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.

There is a maximum number of 16 COs possible in a CAN-RAM of 256 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.

$$\text{Max. Number CO} = \frac{t_{CA\ SCAN}}{t_{CO\ SCAN}}$$

$$t_{CO\ SCAN} = 9\ t_{XTAL}$$

$$t_{CA\ SCAN} = 28\ t_{Bit}$$

$$t_{Bit} = (3 + TSEG1 + TSEG2)\ t_Q$$

$$t_Q = (BPR + 1)\ t_{XTAL}$$

$t_{CA\ Scan}$ 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.

$$K_{BI} = \frac{Z_{BI}}{Z_G}$$

$$\text{Max. Number CO} = \frac{K_{BI}\ t_{CA\ SCAN}}{t_{CO\ SCAN}}$$

Z_{BI} is the number of BI cycles in the total cycles (Z_G), over a relatively long period (mean value). K_{BI} therefore represents a correction factor.

Example for an 8-bit CPU:

The Load and Store-Accu commands require 4 cycles.
OP code Adr.L Adr.H DB
Of the 4 cycles, only the last occupies the CAN-RAM. If a block move without loop is programmed,
LDA 600;
STA 680;
LDA 601;
STA 681;
etc.
then only 3 of 4 cycles remain for the BI, i.e. 75%. K_{BI} would then be 0.75. The maximum number of COs is then 18. This applies only when source and destination lie in the CAN-RAM. If one of the two lies outside, then K_{BI} is 0.875.

If, of course, a loop is programmed,
LDA 600,X Z_{BI} = 3 of 4
STA 680,X Z_{BI} = 3 of 4
DEX Z_{BI} = 2 of 2
BNE NXT Z_{BI} = 2 of 2

10 of 12 cycles are available to the BI. This gives a K_{BI} of 0.833. The BI can then handle 20 COs. If source or destination are not in the CAN-RAM, there are as many as 22.

Example for an 16-bit CPU:

The Load and Store-Accu commands require 5 cycles.
OP code Adr.L Adr.H DBL DBH

Of the 5 cycles, the last two occupy the CAN-RAM. In the worst case, 3 of 5 cycles remain for the BI, i.e. 60%. K_{BI} would then be 0.6. The maximum number of COs is then 14.

21.4. 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.

21.4.1. 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.

21.4.2. 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.

21.4.2.1. 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.

$$t_{Bit} = t_{SYNCSEG} + t_{TSEG1} + t_{TSEG2}$$

$$t_Q = t_{XTAL}(BPR + 1)$$

$$t_{SYNCSEG} = 1\ t_Q$$

$$t_{TSEG1} = (TSEG1 + 1)\ t_Q$$

$$t_{TSEG2} = (TSEG2 + 1)\ t_Q$$

$$t_{SJW} = SJW\ t_Q$$

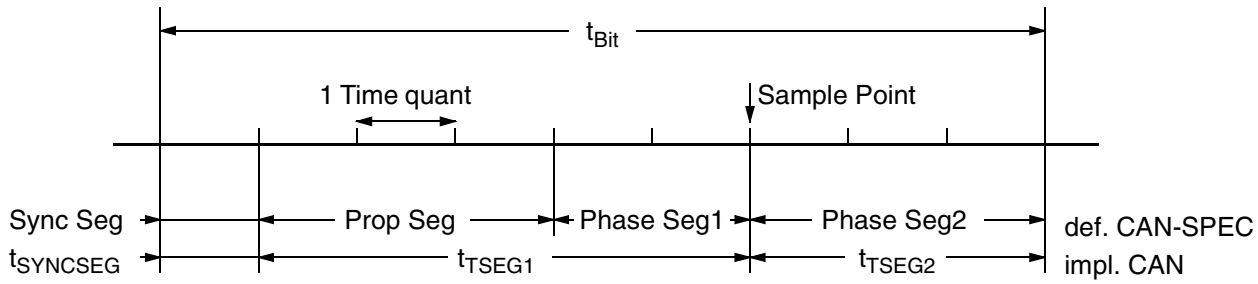


Fig. 21–10: Bit Timing Definition

The baud rate is then calculated as follows:

$$BR = \frac{1}{t_{Bit}}$$

$$t_{Bit} = t_{XTAL}(BPR + 1)\ (3 + TSEG1 + TSEG2)$$

$$BR = \frac{f_{XTAL}}{(BPR + 1)\ (3 + TSEG1 + TSEG2)}$$

21.4.2.2. 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 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.

21.4.2.3. Influence of ERM on CAN Timing

When 29 bits are transferred without intermediate resynchronization and with a synchronization jump width of 4 time

quants and prescaler value of 0, the CAN module can handle a maximum baudrate offset of

$$\pm\frac{4 \cdot f_{BAUD}}{2 \cdot 29 \cdot f_{XTAL}}$$

With f_{XTAL} = 8MHz and f_{BAUD} = 1MBd the result is ±0.86%.

The ERM introduces a limited uncertainty in the position of the actual sample time point which may vary from clock to clock by 0 to 0.121 f_{XTAL}.

Due to this phase modulation, the above maximum baud rate offset is reduced by a small amount:

$$\pm\frac{0.121 \cdot f_{BAUD}}{2 \cdot 29 \cdot f_{XTAL}}$$

With f_{XTAL} = 8MHz and f_{BAUD} = 1MBd the result is ±0.026%, and the maximum baud rate offset is reduced to ±0.83%.

Furthermore, due to this modulation, the propagation delay that the CAN node can produce increases by 0.121 f_{XTAL}. During system design, this increased delay has to be taken into consideration.

$$\text{Prop.Seg.} \geq 2 \cdot f_{XTAL} \cdot \left(\text{ext. delay} + t_{dtxmax} - t_{srxmin} + \frac{0{,}121}{f_{XTAL}}\right)$$

21.4.2.4. 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.

21.4.2.5. 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.

21.4.2.6. 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).

21.5. 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.

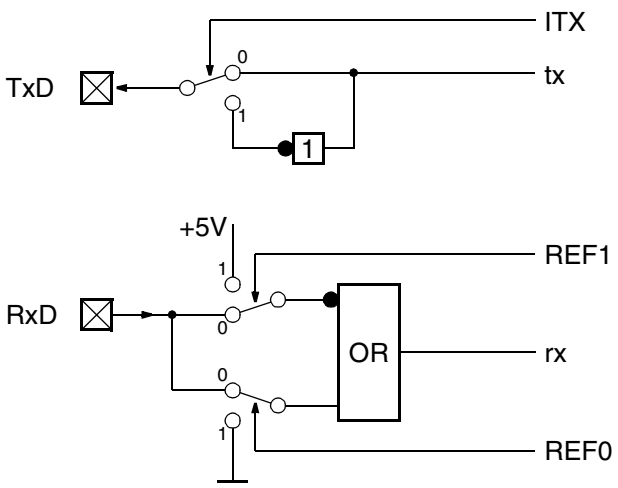


Fig. 21–11: Bus Coupling

Table 21–2: 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 21–3: Logical Level Receiving

REF1	REF0	RxD	rx	Bus Level	Remarks
0	0	x	1	Does not 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	Does not work	

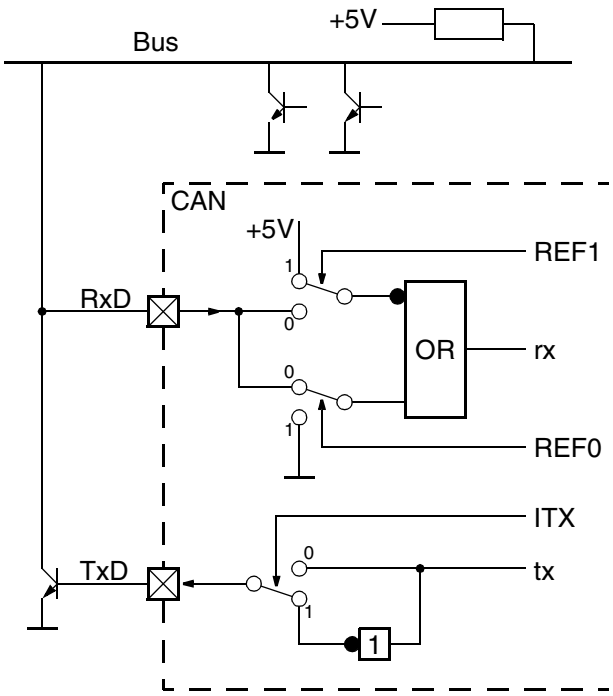


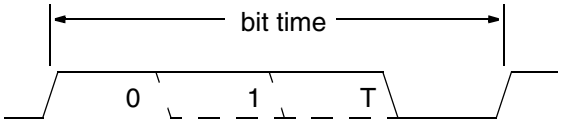
Fig. 21–12: Minimum Bus

22. DIGITbus System Description

22.1. 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 is 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 permanently high bus (100% High Time) means that the bus is passive high. The bus is active if there are consecutive T-Signs, ones or zeros.

A permanently 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 also permitted. For instance TTTTAAATDDDDTDDDDDTT is a valid telegram format on the DIGITbus.

A telegram consisting of one address only is possible, too. In this case, the length of the data field is zero.

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.

When 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. Every bus node may receive them.

22.2. 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.

22.2.1. Wake-up

If the DIGITbus is passive high, (permanent high level for more than one bit period), a slave can pull the bus line down 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/Baudrate).

22.2.2. 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.

22.2.3. 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 nodes 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 separately for each bus node.

22.2.4. Abort Transmission

The Abort Transmission feature is an option that allows the implementation of some kind of rip cord with the DIGITbus. In the event of an alarm, 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 cannot be stopped. The transmission of the alarm telegram is delayed until after the end of the reception in this case. Only the currently sending bus node can abort the transmission.

22.3. Standard Functions

The following standard functions have to be included in every DIGITbus implementation.

22.3.1. 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.

22.3.2. 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 own clock sources nevertheless have to synchronize on the bus clock if they want to transmit or receive data.

22.3.3. 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 the case of a multimaster system, collision detection and arbitration capability.

22.3.4. 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.

22.3.5. Send Data

Every master must be able to send a data field, and some slaves are also able to. A data field is preceded by an address or data field and one T-Sign.

22.3.6. Receive Data

Every master must be able to receive a data field, and some slaves are also able to. 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 data fields of variable lengths are possible too.

22.3.7. 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, immediately stops sending and 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.

22.4. Optional Functions

The following optional functions may be designed into a certain DIGITbus implementation.

22.4.1. Abort Transmission

A master 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.

22.4.2. 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 generating 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.

22.4.3. Correct Phase

Bus nodes which do not generate the bus clock may use the procedure described above to adjust their phase. They have to answer to a special address with sending back a zero. Afterwards they will receive a correction value with another special address. With this value they can adjust the point where they pull the bus line to modify a "T" to a one or a zero.

22.4.4. Generate Wake-up

If the DIGITbus is passive high (no bus clock, always high level), the clock master may become 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.

22.4.5. 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 which uses this feature.

22.4.6. 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.

22.4.7. Receive Reset

The clock master generates the rising edge at the end of a bit time. He will detect the reset condition described above and set a flag if the rising edge is delayed for at least 1/8 of the bit time.

23. DIGITbus Master Module

23.1. Introduction

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 are 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 singlemaster 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.
- Measurement of pulse width for phase correction.
- Phase correction.
- Reception of wake-up and bus reset signal.
- Register interface to the CPU.

23.2. 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 permanent low or high, or 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.

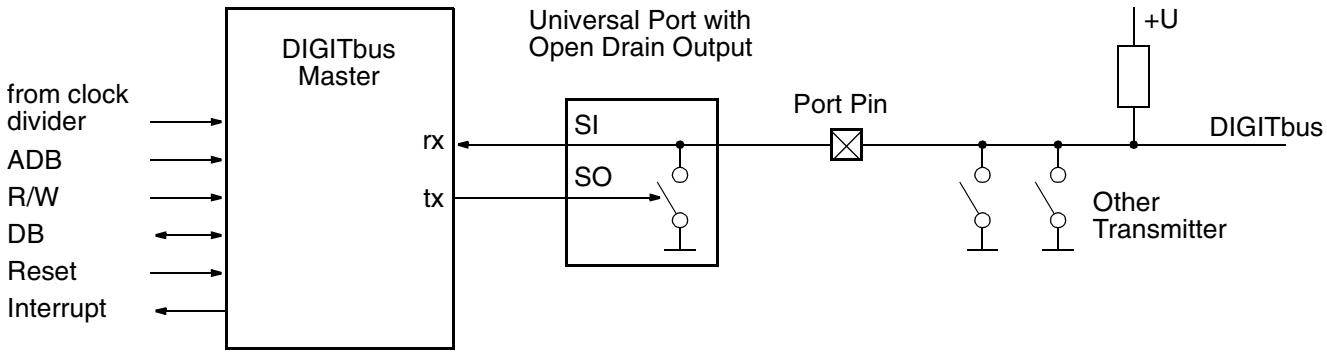


Fig. 23–1: Context Diagram

23.3. Functionality

23.3.1. 3-bit-Prescaler

The programmable 3-bit-Prescaler supplies the module with clock signals. It scales down the clock divider clock by factor 1, 2, 3 to 8 (see Table 23–2 on page 162). The output is 64 times the bus clock. The desired input frequency from the clock divider is hardware programmable.

23.3.2. 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, as it must filter and watch the bus line for a wake-up signal.

23.3.3. Transmit T

The transmit T logic sends a continuous stream of T-signs, if active. It outputs a permanent high if it is inactive.

23.3.4. 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 enforced by the SW using the universal port in normal mode directly. The bus line may be released or pulled low.

23.3.5. 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. In addition, it measures the pulse width of each non T-sign. It creates a bus reset signal if the active bus is held 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.

23.3.6. Send Telegram

The send telegram logic will be enabled by the transmit FIFO and the receive telegram logic when four consecutive T-signs have been 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).

23.3.7. 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 singlemaster system).

23.3.8. Collision Detection

The collision detection logic compares each incoming bit with the currently outgoing bit. A difference is signalled 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.

23.3.9. 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.

23.3.10. 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.

23.3.11. Interrupt

Several flags of the status registers are connected with the interrupt source signal by a logical-OR. The interrupt output can be masked by a flag in the control register.

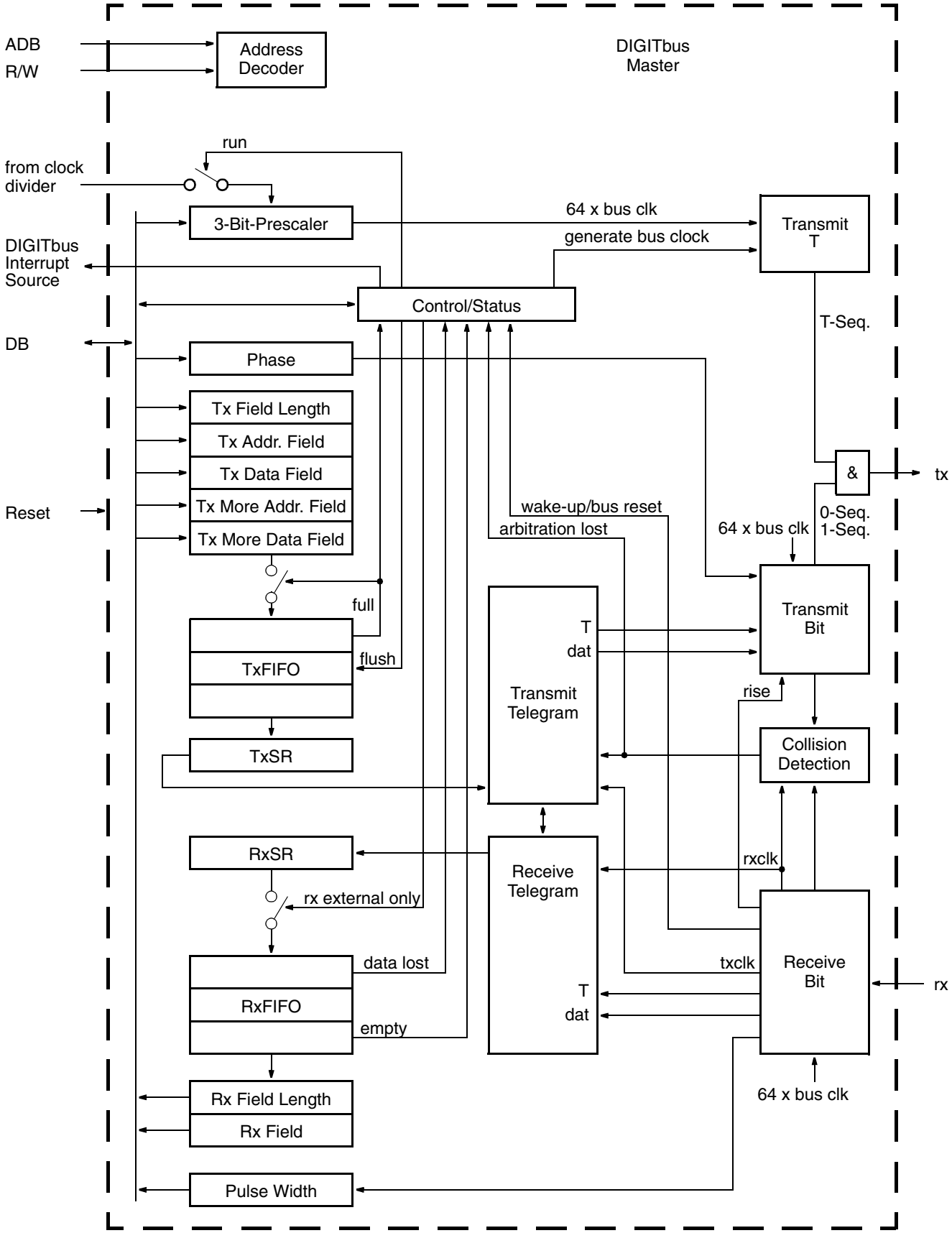


Fig. 23–2: Block Diagram

23.4. Registers

The register mnemonic prefix “DG” stands for DIGITbus.

Table 23–1: 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 absolutely 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 the 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 23–2: Clock Prescaler

PSC hex	Divide by	Bus Clock in kHz		
		@ 6 MHz	@ 8 MHz	@ 10 MHz
0	1	93.75	125.0	156.25
1	2	46.9	62.5	78.1
2	3	31.25	41.7	52.1
3	4	23.4	31.25	39.1
4	5	18.75	25.0	31.25
5	6	15.6	20.8	26.0
6	7	13.4	17.9	22.3
7	8	11.7	15.6	19.5

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.

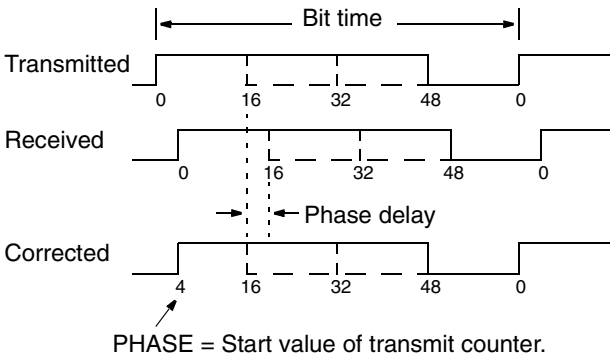


Fig. 23–3: Phase Correction

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. 23–4 on page 163)

NOF **Tx FIFO is Not Full**
r1: There is at least one entry free.
r0: Full.
It generates an interrupt only at the precise moment when the limit is passed. It doesn't generate interrupts when the FIFO is empty (see Fig. 23–4 on page 163).

TGV **Telegram Valid**
r1: Telegram valid
r0: Telegram not valid
w0: Clear flag
This flag will be set if two consecutive T-signs have been 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.

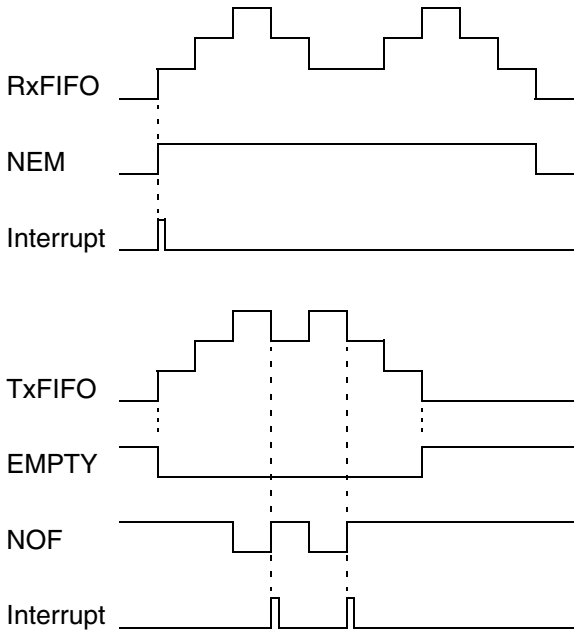


Fig. 23–4: Rx- and TxFIFO Timing

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 be set too 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 23.5.7. on page 167).

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 23–3: 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	EOF	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, EOF, LEN and register DGRTMA are not valid if NEM is false.

FTYP Field Type
r1: Address field
r0: Data field

EOF End of Field
r1: Last byte of a field
r0: Not last byte of a field
If EOF 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 field length defines how many bits of the front entry of the receive FIFO carry valid bits. They are right-aligned (Table 23–4). The real length of the field is unlimited. The user must count the bytes fetched from the FIFO to calculate the real field length.

Table 23–4: 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 23–5 illustrate the interpretation of register DGRTMD. They are valid for an address field (FTYP = 1) or a data field (FTYP = 0).

Table 23–5: DGRTMD Interpretation Examples

LEN	EOF	
6	1	Last byte of a field. The six rightmost 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	x	EMPTY	x	x	x	x	x	x
	x	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.

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 at least one bit time before rewriting TxFIFO.

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 23–4 on page 164). DGTL must be written before the first byte of the actual bit field is written to the FIFO. It only has 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.

23.5. Principle of Operation

23.5.1. Reset

The module reset signal resets all registers and internal HW. The same module reset signal does a standby bit in a standby register.

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 is 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).

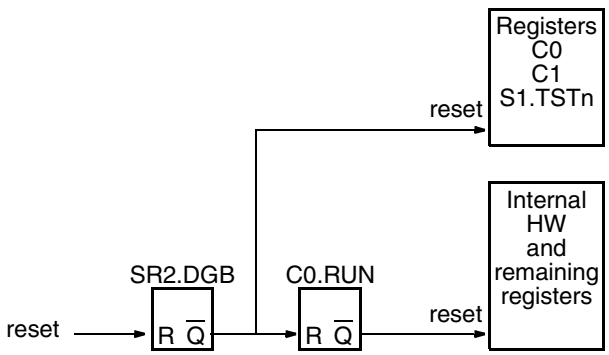


Fig. 23–5: Reset Structure

23.5.2. Initialization

The corresponding port must be configured as special out open drain.

After reset, and after setting flag DGB in standby register SR2, the DIGITbus master is inactive. The global enable flag RUN must be set together with the appropriate prescaler entry PSC, to activate the module.

23.5.2.1. Clock Master

The flag GBC (generate bus clock) must be set, if the DIGITbus master should generate the bus clock. The module now acts as clock master of the connected DIGITbus system. It outputs a stream of T-signs.

23.5.2.2. 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, as 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.

23.5.2.3. Singlemaster System

In a singlemaster 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.

23.5.2.4. 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 has not been lost, the receive FIFO must be read to empty it. The flag RXO has to be cleared in a multimaster system.

Table 23–6: 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 multimaster system)
1	x	1	1	Receive external only. (Recommended in singlemaster system)

23.5.3. 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 module 8 has to be written to register DGTL. This must be done once for each field, and before any entry in 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 into 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 bit, 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.

A new address field is transmitted if at least four consecutive T-signs were 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 have cancelled 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.

23.5.3.1. 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 be completely 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 immediate reaction to a 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 of sending 24 bits. Possibly, 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.

23.5.4. Reception

Every non-T-sign is shifted into the receive shift register. If it is full, or if a T-sign was 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 EOF 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 has 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.

23.5.4.1. 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.

23.5.5. 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 signalled 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 being transmitted can cause problems. Hence, please make sure that bus clock generation is switched off only if bus is idle (T-signs).

23.5.6. 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 have been received after the aborted telegram.

It is not possible to abort a telegram or a field which is transmitted by another bus node.

23.5.7. 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.

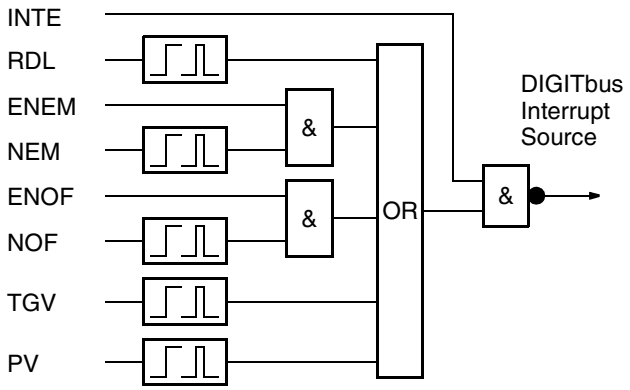


Fig. 23–6: Interrupt Sources

23.5.8. 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 don't affect PW. It must be read before the falling edge of the next non-T sign.

23.5.9. 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 1/15 of the whole bit length.

23.5.10. 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.

– DIGITbus Master HW damaged.

23.5.11. Precautions

Don't use indirect addressing when you write to a DIGITbus register. An unwanted read access to the same address can be the result and a read access to the RxFIFO output register modifies the content of this FIFO. Received data are lost in this case.

If a telegram is aborted by FLUSH, normally there is a TGV interrupt with the reception of the second T sign after the last bit of the aborted telegram. The next TGV interrupt signals the transmission of the alarm telegram. If a transmit telegram is aborted by FLUSH before transmission has actually started, then the first TGV interrupt of the aborted telegram doesn't occur. In this case the TGV interrupt signals the transmission of the alarm telegram.

Don't access DIGITbus registers in CPU SLOW mode. This can cause interrupts. Operation of the module in CPU SLOW mode is allowed.

23.6. Timings

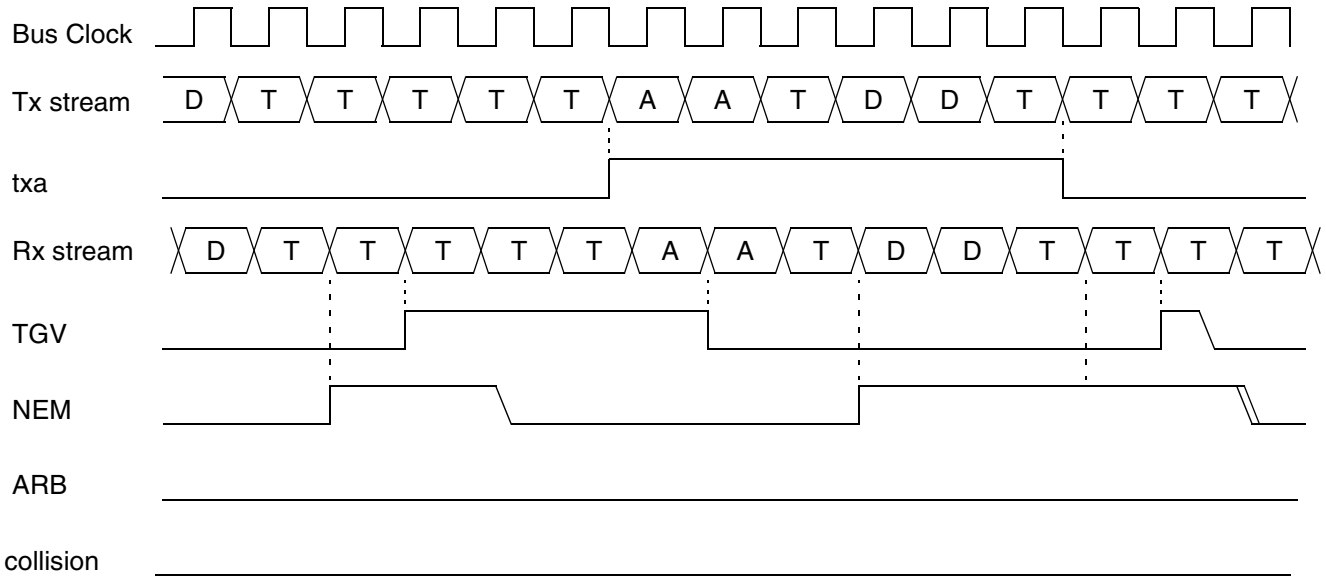


Fig. 23–7: Tx Timing

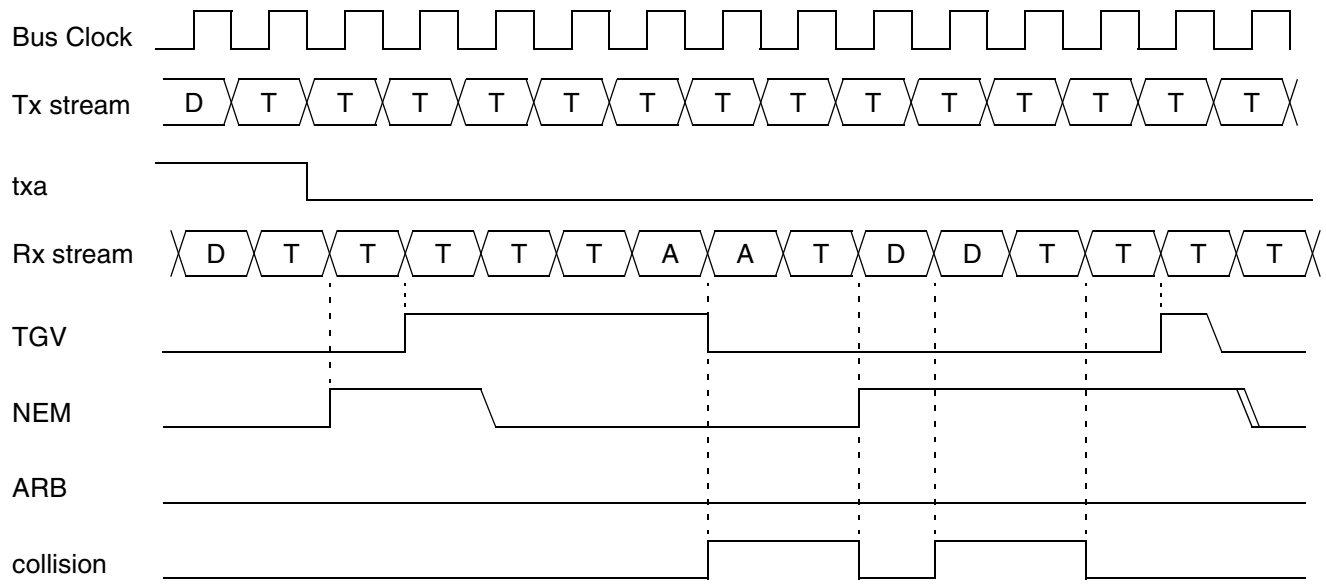


Fig. 23–8: Rx Timing

24. 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

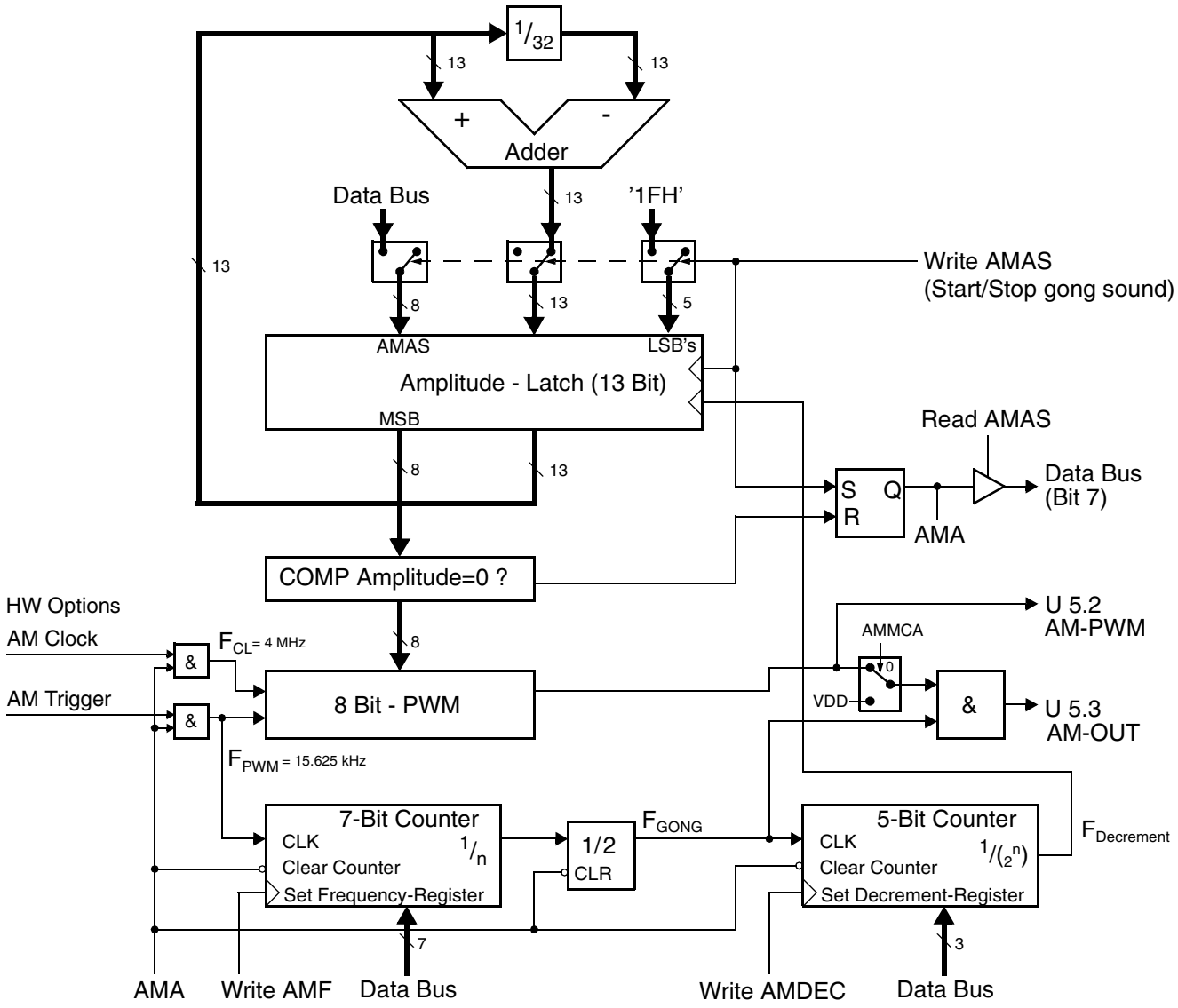


Fig. 24–1: Block diagram of the audio module

24.1. Functional Description

The gong sound frequency is adjusted with the Audio Module Frequency Register (AMF). F_{GONG} is F_{PWM} divided by twice the [AMF value + 1]. The length of AMF is 7 bit, so values from 0 to 127 can be written.

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 Audio Module Decrement Register (AMDEC)), a new amplitude value is calculated ($F_{Decrement}$). The falling edge of the amplitude decrement frequency $F_{Decrement}$ latches the output of the adder into the amplitude latch (13 - bit), and simultaneously the 8 MSB's into the PWM.

During the first low cycle of F_{GONG} following the active $F_{Decrement}$ edge, the PWM ialready runs 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 exponentially decreasing amplitude.

As soon as the 8 MSBs of the amplitude latch reach zero, the AMA will be reset, which deactivates the audio module.

The audio module is only operable in the CPU FAST mode.

24.1.1. Hardware Settings

The AM clock frequency F_{CL} is set by HW option FFB7h. The AM trigger frequency F_{PWM} (PWM reload frequency) is set by HW option FFC3h. Select 4MHz for F_{CL} and $F_{CL}/256$ for F_{PWM} to achieve the standard Audio Module functionality.

24.1.2. Initialization

To connect the audio module output with the corresponding output pin (U5.3), in the register U5M32 the flag PMODE has to be set to switch into the Port Mode. Additionally the port U5.3 has to be switched into the Special Out Mode by setting the flag TRI1 to '0' and the flag N/S1 to '1' in the register U5SEG32.

There is one register to set the gong sound frequency (AMF) and another one to set the gong sound duration (AMDEC) before the gong sound can be started. Both their reset values are zero.

24.1.3. Start Gong

The gong sound is started by writing the initial amplitude value into the Audio Module Amplitude & Status Register (AMAS). Simultaneously with the write to AMAS the Flag Audio Module Active (AMA) is set, which enables the F_{PWM}- and F_{CL}-inputs. The setting of AMA to '1' also enables the F_{GONG}- and $F_{Decrement}$ - counter.

24.1.4. Restart Gong

It's possible to restart the gong sound simply by writing a new initial amplitude value to the AMAS (independently from the former initial value or the current value of the register. Note: The current amplitude value can't be read out). The new gong sound will start immediately with a low cycle of F_{GONG}.

24.1.5. Stop Gong

The gong sound will stop automatically, as soon as the amplitude value in the AMAS reaches zero. This will reset the AMA, which indicates the inactive audio module.

To stop the gong sound, just write 00H into the AMAS. The gong sound then will stop immediately with the writing of 00H. (also indicated by AMA).

A continuous tone will never stop automatically. It can also be stopped by writing 00H into AMAS.

24.1.6. Decay of Sound

The decay characteristic used for this gong sound is described by the following exponential function (see Fig. 24–3 on page 172):

$$A_n = A_0 \, (1 - {}^1\!/_{32})^n$$

with A_0 = initial amplitude
A_n = amplitude after n $F_{Decrement}$ cycles
n = int (t * $F_{Decrement}$)

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 the value of the Audio Module Frequency Register (AMF):

$$F_{Decrement} = F_{Gong} / 2^{GDF}$$

for
GDF settings of 0 .. 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

$$n_\tau \cong 32$$

This means that τ is correlating with $F_{Decrement}$. The higher $F_{Decrement}$, the shorter is τ.

With an initial amplitude of FFH 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.

$$t_{255 \to 0} = 193 \frac{1}{F_{Decrement}} = 193 \frac{2^{GDF}}{F_{GONG}}$$

With an initial amplitude lower than FFH the gong sound duration is shorter.

To sum up, it can be said that the total duration of the gong sound depends on F_{Gong}, set with AMF, the setting of the Gong sound Duration Factor GDF and the setting of the ini-

tial amplitude (see Table 24–1 on page 171). Please note that senseless combinations of register values are also possible.

Table 24–1: Total gong sound running time from A_0 = FFH to A_{Final} = 00H for selected gong sound frequencies (approx. 6τ; F_{PWM}=15.625 kHz)

F_{Gong} (AMF)	GDF=0	GDF=1	GDF=2	GDF=3	GDF=4	GDF= 5
61.0 Hz (min.)	3.20 s	6.39 s	12.8 s	25.6 s	51.1 s	86.6 s
100 Hz	1.95 s	3.9 s	7.8 s	15.6 s	31.2 s	62.4 s
601 Hz	0.324 s	0.649 s	1.30 s	2.60 s	5.19 s	10.4 s
2.60 kHz	75.0 ms	150 ms	300 ms	600 ms	1.2 s	2.4 s
7.81kHz (max.)	25.0 ms	49.9 ms	99.9 ms	200 ms	399 ms	799 ms

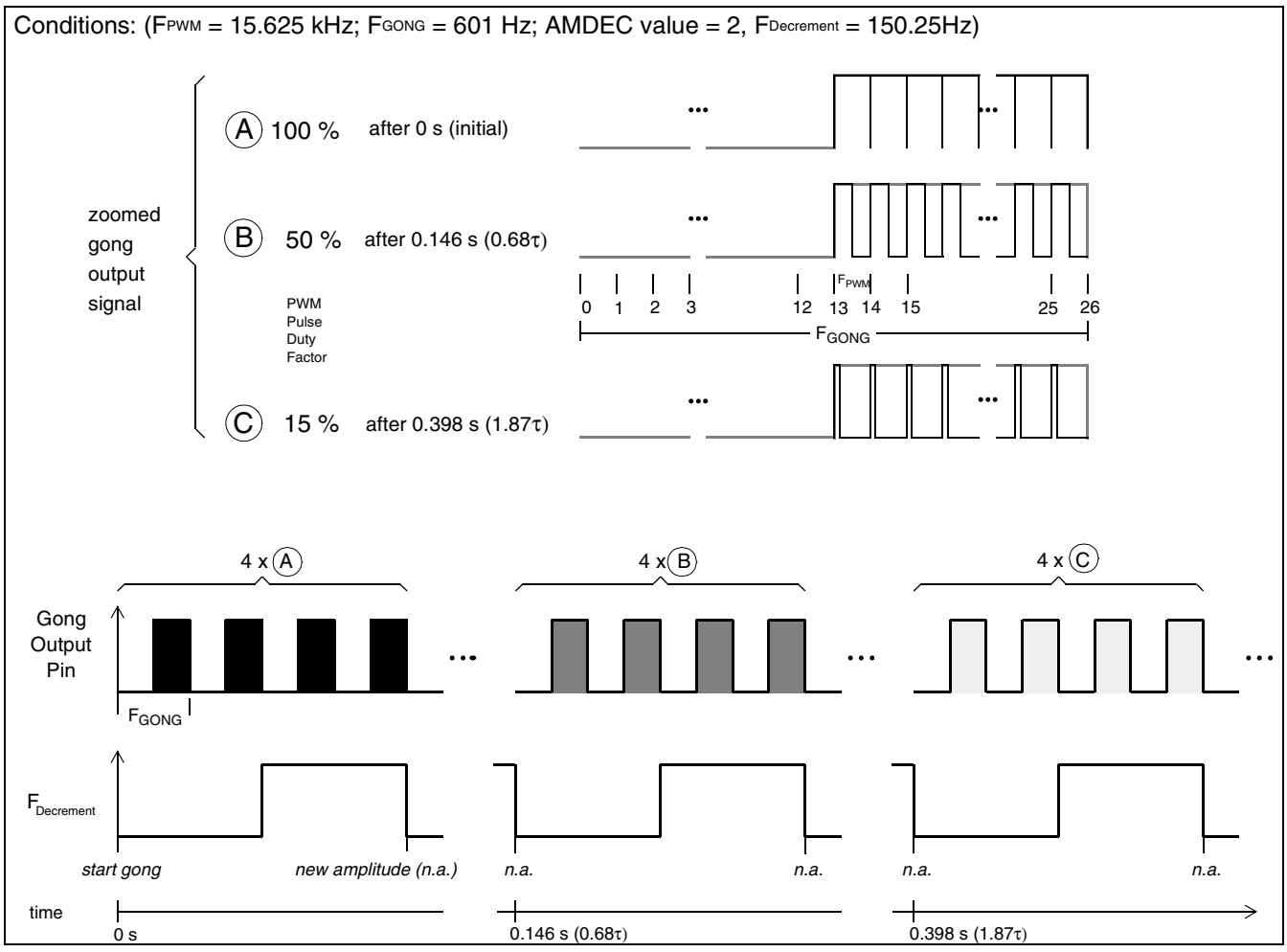


Fig. 24–2: Example sections of the audio module output signal

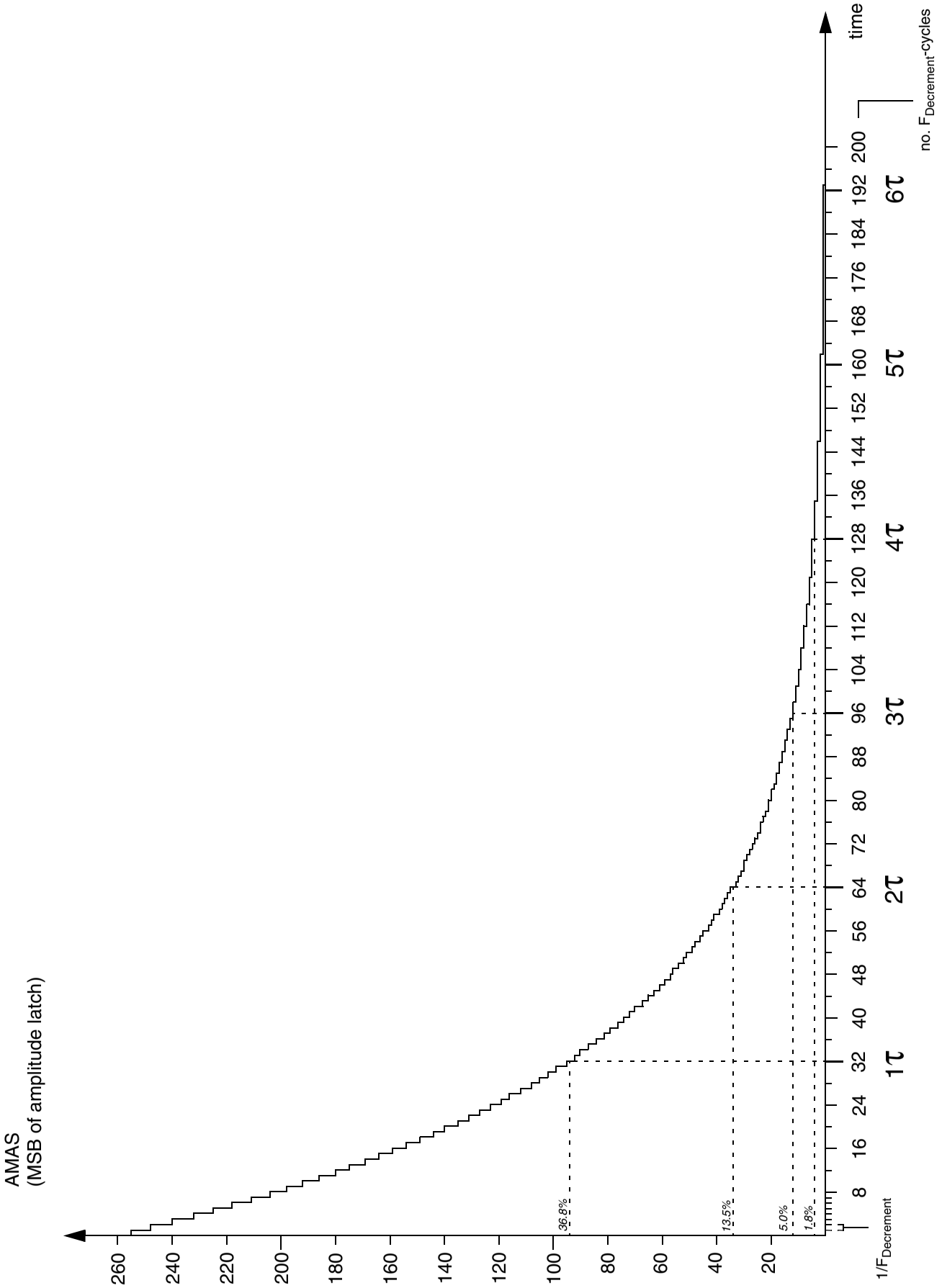


Fig. 24–3: Decay of Sound - Function

24.2. Registers

The Audio Module control registers are located in the I/O area.

Table 24–2: Audio Module Control Registers

Mnemonic	Name
AMAS	Audio Module Amplitude/Status
AMF	Audio Module Frequency
AMDEC	Audio Module Decrement

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 gong sound amplitude. Writing the value 00H into this register during an active gong sound deactivates the gong sound immediately, while writing a value > 00H is restarting the gong sound immediately with the new Initial Amplitude.

wxx: (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:

$$value = \frac{F_{PWM}}{2F_{GONG}} - 1$$

$$F_{GONG} = \frac{F_{PWM}}{2(value + 1)}$$

With the 7bit *value* ranging from 0 to 127, the programmable gong sound frequency range is 61 Hz .. 7.81 kHz (see Table 24–3 on page 173).

It's possible to write a new gong sound frequency during an active audio module (AMA = '1').

Table 24–3: Examples for AMF-values

AMF - value	resulting F_{Gong}
127 (max.)	61.0 Hz (min.)
:	:
77	100 Hz
:	:
12	601 Hz
:	:
2	2.60 kHz
:	:
0 (min.)	7.81kHz (max.)

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 Maximal Constant Amplitude Flag

w1: Activate the AMMCA mode.
w0: Deactivate the AMMCA mode.

With the flag AMMCA the Audio Module Maximal Constant Amplitude (AMMCA) 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 a amplitude update frequency of F_{GONG} / 2 to F_{GONG} / 32 according this equation:

$$F_{Decrement} = \frac{F_{GONG}}{2^{GDF}} \qquad GDF = 0 \ldots 5$$

A value of 6 or 7 means no 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 '00H' to AMAS or change the gong sound duration factor to let the

tone decay. It's possible to change GDF during an active gong sound (AMA = '1').

Table 24–4: Definition of GDF

GDF	gong sound duration factor
0H	1
1H	2
2H	4
3H	8
4H	16
5H	32
6H	continuous tone
7H	

25. Hardware Options

25.1. 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_{osc} to $f_{osc}/2^{17}$ plus some internal signals (see Table 25–2 on page 179)

– interrupt source selection for interrupt inputs 0, 1, 5, 6, 7, 10, 13, 14 and 15

– Special Out signal selection for some U- and H-ports

– Rx/Tx polarity selection for SPI and UART modules

– U-port Port Slow Mode selection

The setting of the Hardware Option takes place in two steps:

1. selection is done by programming dedicated address locations with the desired options' code

2. activation is done by a read access to these dedicated address locations at least once after each reset.

Address locations 00FFB8h through 00FFBFh do not allow random setting. Their respective Hardware Options are hard-wired and can only be altered by changing a production mask for this IC. By default all U-Ports have the Port Slow Option set with the exception of U1.0 to U1.3 (Port Fast Option set). The Watchdog and Clock Monitor are SW activated by default.

Future mask ROM derivatives of this IC will not require (but will tolerate) activation of option settings by read accesses because ROM as well as options will be hard-wired. Instead, the manufacturer will automatically process the setting of the dedicated address locations, as given in the ROM code file, to set the required mask changes.

To ensure compatible option settings in this IC and mask ROM derivatives when run with the same ROM code, it is recommended to always read locations 00FFA0h through 00FFC3h directly after reset. Be aware that the non-programmable locations 00FFB8h through 00FFBFh may not be compatible among this IC and the mask ROM derivative.

25.2. Listing of Dedicated Addresses and Corresponding Hardware Options

Table 25–1: Hardware-Option-Dedicated Addresses

7	6	5	4	3	2	1	0

00FFA0 Timer 0 Clock Options

x	x	x	Clock options f1 to f31

00FFA1 PWM0, 3 Clock Options

x	x	x	Clock options f0 to f31 (all)

00FFA2 PWM0, 3 Trigger Options

x	x	x	Clock options f0 to f31 (all)
The high pulse width of the trigger period must be greater than the high pulse width of the clock the PWM is provided with.			

00FFA3 PWM1, 4 Clock Options

x	x	x	Clock options f0 to f31 (all)

00FFA4 PWM1, 4 Trigger Options

x	x	x	Clock options f0 to f31 (all)
The high pulse width of the trigger period must be greater than the high pulse width of the clock the PWM is provided with.			

Table 25–1: Hardware-Option-Dedicated Addresses

7	6	5	4	3	2	1	0

00FFA5 PWM2 Clock Options

x	x	x	Clock options f0 to f31 (all)

00FFA6 PWM2 Trigger Options

x	x	x	Clock options f0 to f31 (all)
The high pulse-width of the trigger period must be greater than the high pulse-width of the clock the PWM is provided with.			

00FFA7 Timer 1 Option

x	x	x	Clock options f0 to f31 (all)

00FFA8 Timer 2 Option

x	x	x	Clock options f0 to f31 (all)

00FFA9 CAPCOM Counter Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFAA DIGITbus Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFAB Clock Out 0: Mux0 Prescaler and Clock Option

x	x0: Mux out direct 01: Mux out / 1.5 11: Mux out / 2.5	Clock options f0 to f31 (all)

00FFAC Clock Out 1: Prescaler and Clock Option

x	x0: direct 01: 1/ 1.5 11: 1/ 2.5	Clock options f0 to f31 (all)

00FFAD LCD Module Prescaler and Clock Option

x	x0: direct 01: 1/ 1.5 11: 1/ 2.5	Clock options f0 to f31 (all)

00FFAE SMM, SPI0, SPI1 Clock Prescaler and SMM Clock Option

x	x0: direct 01: 1/ 1.5 11: 1/ 2.5	Clock options f0 to f31 (all)

Table 25–1: Hardware-Option-Dedicated Addresses

7	6	5	4	3	2	1	0

00FFAF SPI0 Input, Output and $F0_{SPI}$ Clock Option 1)

SPI0-D-OUT 0: direct 1: inverted	SPI0-D-IN 0: direct 1: inverted	x	Clock options f0 to f31 (all)
1) FFAE, Bits 6 & 5 define also the SPI0 and SPI1 prescaler setting.			

00FFB0 SPI1 Input, Output and $F1_{SPI}$ Clock Option 1)

SPI1-D-OUT 0: direct 1: inverted	SPI1-D-IN 0: direct 1: inverted	x	Clock options f0 to f31 (all)
1) FFAE, Bits 6 & 5 define also the SPI0 and SPI1 prescaler setting.			

00FFB1 $F2_{SPI}$ Clock Options

x	x	x	Clock options f0 to f31 (all)

00FFB2 Clock Out 0: Mux1 Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFB3 Clock Out 0: Mux2 Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFB4 UART0, 2 Input and Output

UART0 Tx 0: direct 1: inverted	UART0 Rx 0: direct 1: inverted	UART2 Tx 0: direct 1: inverted	UART2 Rx 0: direct 1: inverted	x	x	x	x

00FFB5 UART1 Options

UART1 Tx 0: direct 1: inverted	UART1 Rx 0: direct 1: inverted	x	x	x	x	x	x

00FFB6 Clock Out 0: Mux3 Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFB7 AM Clock Option

x	x	x	Clock options f0 to f31 (all)

00FFB8 Clock Monitor Options

x	Wdog & Clk Monitor: 0: deact. by Software 1: always active	x	x	x	x	x	x

Table 25–1: Hardware-Option-Dedicated Addresses

7	6	5	4	3	2	1	0

00FFB9 Universal Port 1 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBA Universal Port 2 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBB Universal Port 3 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBC Universal Port 4 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBD Universal Port 5 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBE Universal Port 6 Slow/Fast Options

0: Fast mode pin only 1: Slow or fast mode pin possible

00FFBF Universal Port 7 Slow/Fast Options

x	x	x	x	0: Fast mode pin only 1: Slow or fast mode pin possible

00FFC0 Interrupt Sources Multiplexer 1 to 4

Mux4:		Mux3:		Mux2:		Mux1:	
00	CAN 2	00	PINT3-IN	00	UART 2	00	CC0 COMP
01	SPI 0	01	SPI 1	01	P06 COMP	01	Timer 2
10	DMA	10	UART 1	10	SPI 0	10	CAN 2
11	PINT3-IN	11	CC1 COMP	11	Timer 1	11	Timer 1

00FFC1 Interrupt Sources Multiplexer 5 to 8

Mux8:		Mux7:		Mux6:		Mux5:	
00	CC1OR	00	CC0OR	00	Timer 2	00	Timer 2
01	PINT2-IN	01	UART 1	01	DIGITbus	01	UART 1
10	IR-RTC	10	IR-RTC	10	UART 2	10	SPI 1
11	IR-WAPI	11	IR-WAPI	11	PINT2-IN	11	DMA

Table 25–1: Hardware-Option-Dedicated Addresses

7	6	5	4	3	2	1	0

00FFC2 Interrupt Sources Multiplexer 9 and Port Multiplexer

H-Port 1.1 0: SME 1: PWM2	x	H-Port 1.0 0: SME 1: PWM0	PINT3-IN 0: at U5.6 1: at U5.7	x	x	Mux9: 00 CAN 1 01 UART 1 10 IR-RTC 11 IR-WAPI

00FFC3 AM Trigger Option

x	x	x	Clock options f0 to f31 (all)

Table 25–2: Clock Option Selection Code

Clock Option Number	Clock Signal	Selection Code
f0	$f_{OSC}/2^0$	xxx0.0000
f1	$f_{OSC}/2^1$	xxx0.0001
f2	$f_{OSC}/2^2$	xxx0.0010
f3	$f_{OSC}/2^3$	xxx0.0011
f4	$f_{OSC}/2^4$	xxx0.0100
f5	$f_{OSC}/2^5$	xxx0.0101
f6	$f_{OSC}/2^6$	xxx0.0110
f7	$f_{OSC}/2^7$	xxx0.0111
f8	$f_{OSC}/2^8$	xxx0.1000
f9	$f_{OSC}/2^9$	xxx0.1001
f10	$f_{OSC}/2^{10}$	xxx0.1010
f11	$f_{OSC}/2^{11}$	xxx0.1011
f12	$f_{OSC}/2^{12}$	xxx0.1100
f13	$f_{OSC}/2^{13}$	xxx0.1101
f14	$f_{OSC}/2^{14}$	xxx0.1110
f15	$f_{OSC}/2^{15}$	xxx0.1111
f16	$f_{OSC}/2^{16}$	xxx1.0000
f17	$f_{OSC}/2^{17}$	xxx1.0001
f18	V_{SS}	xxx1.0010
f19	Timer 0	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	$f_{OSC}/2^2$	xxx1.1100
f29, 30	V_{SS}	xxx1.1101 ...
f31	$f_{OSC}/2^{10}$	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.

Table 25–2: Clock Option Selection Code

Clock Option Number	Clock Signal	Selection Code
f18	V_{SS}	xxx1.0010
f19	Timer 0	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	$f_{OSC}/2^2$	xxx1.1100
f29, 30	V_{SS}	xxx1.1101 ...
f31	$f_{OSC}/2^{10}$	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.

26. Register Cross Reference Table V2.1

26.1. CAN Registers, memory page 1C

Address (hex)	Mnemonic	Block
1C00	CAN0CTR	CAN0
1C01	CAN0STR	
1C02	CAN0ESTR	
1C03	CAN0IDX	
1C04	CAN0IDM	
1C05		
1C06		
1C07		
1C08	CAN0BT1	
1C09	CAN0BT2	
1C0A	CAN0BT3	
1C0B	CAN0ICR	
1C0C	CAN0OCR	
1C0D	CAN0TEC	
1C0E	CAN0REC	
1C0F	CAN0ESM	
1C10	CAN0CTIM	
1C11		

Address (hex)	Mnemonic	Block
1C40	CAN1CTR	CAN1
1C41	CAN1STR	
1C42	CAN1ESTR	
1C43	CAN1IDX	
1C44	CAN1IDM	
1C45		
1C46		
1C47		
1C48	CAN1BT1	
1C49	CAN1BT2	
1C4A	CAN1BT3	
1C4B	CAN1ICR	
1C4C	CAN1OCR	
1C4D	CAN1TEC	
1C4E	CAN1REC	
1C4F	CAN1ESM	
1C50	CAN1CTIM	
1C51		

Address (hex)	Mnemonic	Block
1C80	CAN2CTR	CAN2
1C81	CAN2STR	
1C82	CAN2ESTR	
1C83	CAN2IDX	
1C84	CAN2IDM	
1C85		
1C86		
1C87		
1C88	CAN2BT1	
1C89	CAN2BT2	
1C8A	CAN2BT3	
1C8B	CAN2ICR	
1C8C	CAN2OCR	
1C8D	CAN2TEC	
1C8E	CAN2REC	
1C8F	CAN2ESM	
1C90	CAN2CTIM	
1C91		

26.2. I/O Register 1, memory page 1E

Address (hex)	Mnemonic	Block
1E64	PAR0	Patch Module
1E65	PAR1	
1E66	PAR2	
1E67	PDR	
1E68	PER0	
1E69	PER1	

Address (hex)	Mnemonic	Block
1E70	WUS	Power Saving Module
1E71		
1E74	SSR	
1E75		
1E76		
1E78	SSC	
1E79		
1E7A		
1E7C	RTC	
1E7D		
1E7E		
1E80	WPM0	
1E81	WPM2	
1E82	WPM4	
1E83	WPM6	
1E84	WPM8	
1E88	WSC	
1E90	OSC	
1E94	RTCC	
1E98	POL	
1E99		
1E9C	SMX	
1EA0	MULCAND	Multiplier
1EA1	MULPLIER	
1EA2	MULPROD	
1EA3		

26.3. I/O Register 0, memory page 1F

Address (hex)	Mnemonic	Block
1F00	CSW0	Core Logic
1F01	CR	
1F02	ERMC	ERM
1F08	SR0	Core Logic
1F09	SR1	
1F0A	SR2	
1F0B	SR3	
1F0C	DBG	Debug Register
1F0F	ABR	Memory Banking
1F10	SPI0D	SPI0
1F11	SPI0M	
1F12	SPI1D	SPI1
1F13	SPI1M	
1F14	CO0SEL	Core Logic
1F15	CO1SEL	
1F18	UA1D	UART1
1F19	UA1C	
1F1A	UA1BR0	
1F1B	UA1BR1	
1F1C	UA1IM	
1F1D	UA1CA	
1F1E	UA1IF	
1F1F	IRE	Interrupt Controller
1F20	IRC	
1F21	IRRET	
1F22	IRPRI10	
1F23	IRPRI32	
1F24	IRPRI54	
1F25	IRPRI76	
1F26	IRPRI98	
1F27	IRPRIBA	
1F28	IRPRIDC	
1F29	IRPRIFE	
1F2A	IRP	
1F2B	IRPM0	
1F2C	IRPP	
1F2D	AMAS	Audio Module
1F2E	AMF	
1F2F	AMDEC	
1F30	U2D	Universal Port 2
1F32	U2SEG10	
1F33	U2M10	
1F34	U2SEG32	
1F35	U2M32	
1F36	U2SEG54	
1F37	U2M54	
1F38	U2SEG76	
1F39	U2M76	
1F4E	TIM0	Timer 0
1F4F		
1F50	PWM0	PWM
1F51	PWM1	
1F52	PWM2	

Address (hex)	Mnemonic	Block
1F54	TIM1	Timer 1, 2
1F55	TIM2	
1F5A	SMVC	Stepper Motor Module
1F5B	SMVSIN	
1F5C	SMVCOS	
1F5D	SMVCMP	
1F5E	PWM3	PWM
1F5F	PWM4	
1F60	CSW1	Core Logic
1F61	CSW2	
1F64	UA2D	UART2
1F65	UA2C	
1F66	UA2BR0	
1F67	UA2BR1	
1F68	UA2IM	
1F69	UA2CA	
1F6A	UA2IF	
1F6C	CC0M	Capture Compare Module
1F6D	CC0I	
1F6E	CC0	
1F6F		
1F70	CC1M	
1F71	CC1I	
1F72	CC1	
1F73		
1F74	CC2M	
1F75	CC2I	
1F76	CC2	
1F77		
1F7C	CCC	
1F7D		
1F7E	P0D	Analog Input Port 0

Address (hex)	Mnemonic	Block
1F80	H0NS	High Current Port 0
1F81	H0TRI	
1F82	H0D	
1F84	H1NS	High Current Port 1
1F85	H1TRI	
1F86	H1D	
1F88	H2NS	High Current Port 2
1F89	H2TRI	
1F8A	H2D	
1F90	H3NS	High Current Port 3
1F91	H3TRI	
1F92	H3D	
1F98	U1D	Universal Port 1
1F99	U1SEG10	
1F9A	U1SEG32	
1F9B	U1M30	
1F9C	U1SEG54	
1F9D	U1M54	
1F9E	U1SEG76	
1F9F	U1M76	
1FA0	UA0D	UART0
1FA1	UA0C	
1FA2	UA0BR0	
1FA3	UA0BR1	
1FA4	UA0IM	
1FA5	UA0CA	
1FA6	UA0IF	
1FA8	AD0	AD Converter
1FA9	AD1	

Address (hex)	Mnemonic	Block
1FAC	U3D	Universal Port 3
1FAE	U3SEG10	
1FAF	U3M10	
1FB0	U3SEG32	
1FB1	U3M32	
1FB2	U3SEG54	
1FB3	U3M54	
1FB4	U3SEG76	
1FB5	U3M76	
1FB8	U4D	Universal Port 4
1FBA	U4SEG10	
1FBB	U4M10	
1FBC	U4SEG32	
1FBD	U4M32	
1FBE	U4SEG54	
1FBF	U4M54	
1FC0	U4SEG76	
1FC1	U4M76	
1FC4	U5D	Universal Port 5
1FC6	U5SEG10	
1FC7	U5M10	
1FC8	U5SEG32	
1FC9	U5M32	
1FCA	U5SEG54	
1FCB	U5M54	
1FCC	U5SEG76	
1FCD	U5M76	

Address (hex)	Mnemonic	Block
1FD0	U6D	Universal Port 6
1FD2	U6SEG10	
1FD3	U6M10	
1FD4	U6SEG32	
1FD5	U6M32	
1FD6	U6SEG54	
1FD7	U6M54	
1FD8	U6SEG76	
1FD9	U6M76	
1FDC	U7D	Universal Port 7
1FDE	U7SEG10	
1FDF	U7M10	
1FE0	U7SEG32	
1FE1	U7M32	
1FE8	DCS	DMA
1FE9	DIC	
1FEA	DSA	
1FEB		
1FEC		
1FED	DEA	
1FEE		
1FEF		
1FF0	DGC0	DIGITbus
1FF1	DGC1	
1FF2	DGS0	
1FF3	DGRTMD	
1FF4	DGTL	
1FF5	DGS1TA	
1FF6	DGTD	
1FF7	DGRTMA	
1FFD	TST3	TST
1FFE	TST1	
1FFF	TST2	

27. Register Quick Reference

Table 27–1: A/D Converter

Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
			7 6 5 4 3 2 1 0	
AD0	ADC Register 0	1FA8	r: EOC | CMPO | x | x | x | x | AN1 | AN0 w: TSAMP | | | CHANNEL 0 0 x x 0 0 0 0 Res	12.2.
AD1	ADC Register 1	1FA9	r: AN9 | AN8 | AN7 | AN6 | AN5 | AN4 | AN3 | AN2	

Table 27–2: Audio Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
			7 6 5 4 3 2 1 0	
AMAS	Audio Module Amplitude & Status Register	1F2D	w: Initial Amplitude r: AMA | x | x | x | x | x | x | x 0 x x x x x x x Res	24.2.
AMF	Audio Module Frequency Register	1F2E	w: x | Sound Frequency - 0 0 0 0 0 0 0 Res	
AMDEC	Audio Module Decrement Register	1F2F	w: AMMCA | x | x | x | x | GDF 0 - - - - 0 0 0 Res	

Table 27–3: Capture-Compare-Unit

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CC0M	CAPCOM 0 Mode Register	1F6C	r/w: MSK MSK MSK FOL OAM IAM 0 0 0 0 0 0 0 0 Res	15.2.
CC0I	CAPCOM 0 Interrupt Register	1F6D	r/w: CAP CMP OFL LAC RCR x x x 0 0 0 0 0 0 0 0 Res	
CC0	CAPCOM 0 Capture/ Compare Register low byte	1F6E	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	
	CAPCOM 0 Capture/ Compare Register high byte	1F6F	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	
CC1M	CAPCOM 1 Mode Register	1F70	r/w: MSK MSK MSK FOL OAM IAM 0 0 0 0 0 0 0 0 Res	
CC1I	CAPCOM 1 Interrupt Register	1F71	r/w: CAP CMP OFL LAC RCR x x x 0 0 0 0 0 0 0 0 Res	
CC1	CAPCOM 1 Capture/ Compare Register low byte	1F72	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	
	CAPCOM 1 Capture/ Compare Register high byte	1F73	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	
CC2M	CAPCOM 2 Mode Register	1F74	r/w: MSK MSK MSK FOL OAM IAM 0 0 0 0 0 0 0 0 Res	
CC2I	CAPCOM 2 Interrupt Register	1F75	r/w: CAP CMP OFL LAC RCR x x x 0 0 0 0 0 0 0 0 Res	

Table 27–3: Capture-Compare-Unit

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CC2	CAPCOM 2 Capture/ Compare Register low byte	1F76	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	15.2.
	CAPCOM 2 Capture/ Compare Register high byte	1F77	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	
CCC	CAPCOM Counter low byte	1F7C	r: Read low byte and lock CCC 0 0 0 0 0 0 0 0 Res	
	CAPCOM Counter high byte	1F7D	r: Read high byte and unlock CCC 0 0 0 0 0 0 0 0 Res	

Table 27–4: Controller Area Network 0

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN0CTR	Control Register	1C00	r/w: HLT | SLP | GRSC | EIE | GRIE | GTIE | BOST | rsvd 1 0 0 0 0 0 0 x Res	21.2.
CAN0STR	Status Register	1C01	r: HACK | BOFF | EPAS | ERS | rsvd | rsvd | rsvd | rsvd 1 0 0 0 x x x x Res	
CAN0ESTR	Error Status Register	1C02	r/w: GDM | CTOV | ECNT | BIT | STF | CRC | FRM | ACK 0 0 0 0 0 0 0 0 Res	
CAN0IDX	Interrupt Index Register	1C03	r/w: Interrupt Index 1 1 1 1 1 1 1 1 Res	
CAN0IDM	Identifier Mask Register	1C04	r/w: Identifier Mask Bits 29 to 21 low	
		1C05	r/w: Identifier Mask Bits 20 to 13	
		1C06	r/w: Identifier Mask Bits 12 to 5	
		1C07	r/w: Identifier Mask Bits 4 to 0 | x | x | x high 0 0 0 0 0 0 0 0 Res	

Table 27–4: Controller Area Network 0

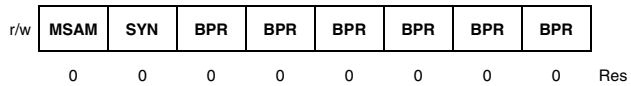
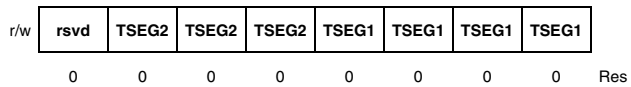
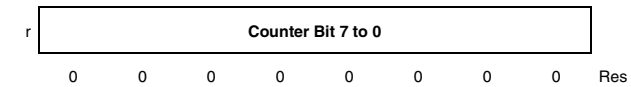
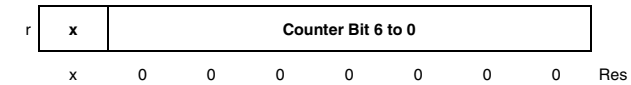
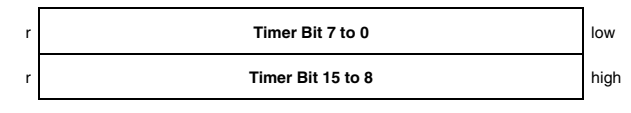
Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN0BT1	Bit Timing Register 1	1C08		21.2.
CAN0BT2	Bit Timing Register 2	1C09		
CAN0BT3	Bit Timing Register 3	1C0A		
CAN0ICR	Input Control Register	1C0B		
CAN0OCR	Output Control Register	1C0C		
CAN0TEC	Transmit Error Counter	1C0D		
CAN0REC	Receive Error Counter	1C0E		
CAN0ESM	Error Status Mask Register	1C0F		
CAN0CTIM	Capture Timer	1C10 1C11		

Table 27–5: Controller Area Network 1

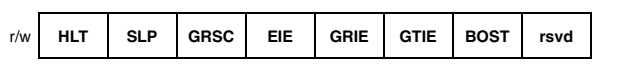
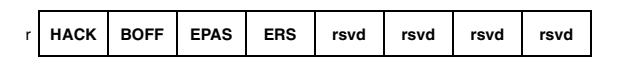
Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN1CTR	Control Register	1C40		21.2.
CAN1STR	Status Register	1C41		

Table 27–5: Controller Area Network 1

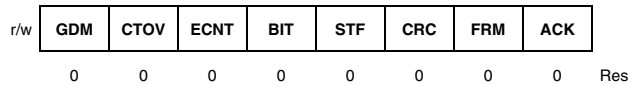
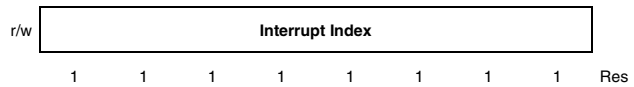
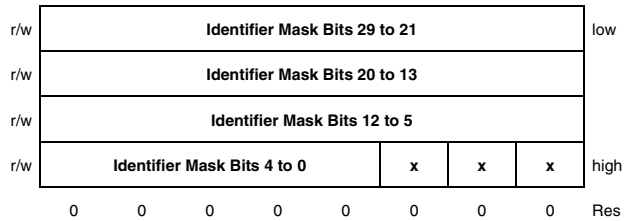
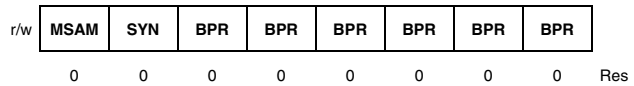
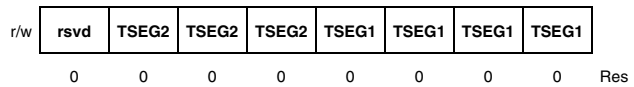
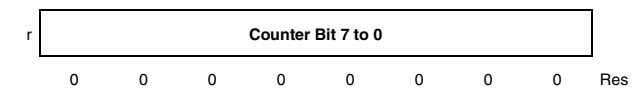
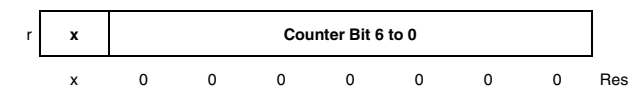
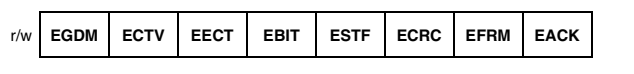
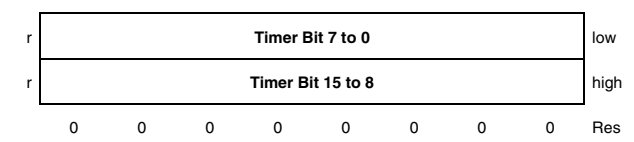
Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN1ESTR	Error Status Register	1C42		21.2.
CAN1IDX	Interrupt Index Register	1C43		
CAN1IDM	Identifier Mask Register	1C44 1C45 1C46 1C47		
CAN1BT1	Bit Timing Register 1	1C48		
CAN1BT2	Bit Timing Register 2	1C49		
CAN1BT3	Bit Timing Register 3	1C4A		
CAN1ICR	Input Control Register	1C4B		
CAN1OCR	Output Control Register	1C4C		
CAN1TEC	Transmit Error Counter	1C4D		
CAN1REC	Receive Error Counter	1C4E		
CAN1ESM	Error Status Mask Register	1C4F		
CAN1CTIM	Capture Timer	1C50 1C51		

Table 27–6: Controller Area Network 2

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN2CTR	Control Register	1C80	r/w: HLT | SLP | GRSC | EIE | GRIE | GTIE | BOST | rsvd 1 0 0 0 0 0 0 x Res	21.2.
CAN2STR	Status Register	1C81	r: HACK | BOFF | EPAS | ERS | rsvd | rsvd | rsvd | rsvd 1 0 0 0 x x x x Res	
CAN2ESTR	Error Status Register	1C82	r/w: GDM | CTOV | ECNT | BIT | STF | CRC | FRM | ACK 0 0 0 0 0 0 0 0 Res	
CAN2IDX	Interrupt Index Register	1C83	r/w: Interrupt Index 1 1 1 1 1 1 1 1 Res	
CAN2IDM	Identifier Mask Register	1C84	r/w: Identifier Mask Bits 29 to 21 low	
		1C85	r/w: Identifier Mask Bits 20 to 13	
		1C86	r/w: Identifier Mask Bits 12 to 5	
		1C87	r/w: Identifier Mask Bits 4 to 0 | x | x | x high 0 0 0 0 0 0 0 0 Res	
CAN2BT1	Bit Timing Register 1	1C88	r/w: MSAM | SYN | BPR | BPR | BPR | BPR | BPR | BPR 0 0 0 0 0 0 0 0 Res	
CAN2BT2	Bit Timing Register 2	1C89	r/w: rsvd | TSEG2 | TSEG2 | TSEG2 | TSEG1 | TSEG1 | TSEG1 | TSEG1 0 0 0 0 0 0 0 0 Res	
CAN2BT3	Bit Timing Register 3	1C8A	r/w: rsvd | rsvd | rsvd | rsvd | rsvd | SJW | SJW | SJW x x x x x 0 0 0 Res	
CAN2ICR	Input Control Register	1C8B	r/w: rsvd | rsvd | rsvd | rsvd | rsvd | XREF | REF1 | REF0 x x x x x 0 0 0 Res	
CAN2OCR	Output Control Register	1C8C	r/w: rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | rsvd | ITX x x x x x x x 0 Res	
CAN2TEC	Transmit Error Counter	1C8D	r: Counter Bit 7 to 0 0 0 0 0 0 0 0 0 Res	
CAN2REC	Receive Error Counter	1C8E	r: x | Counter Bit 6 to 0 x 0 0 0 0 0 0 0 Res	

Table 27–6: Controller Area Network 2

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
CAN2ESM	Error Status Mask Register	1C8F	r/w: EGDM ECTV EECT EBIT ESTF ECRC EFRM EACK 1 1 1 1 1 1 1 1 Res	
CAN2CTIM	Capture Timer	1C90	r: Timer Bit 7 to 0 low	21.2.
		1C91	r: Timer Bit 15 to 8 high 0 0 0 0 0 0 0 0 Res	

Table 27–7: Core Logic

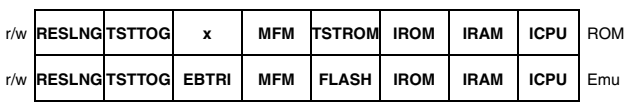
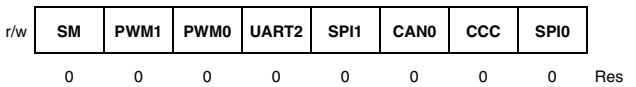
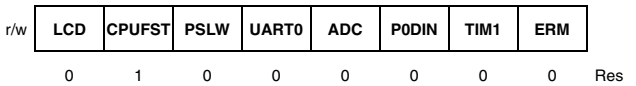
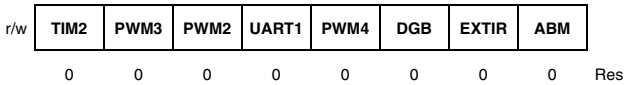
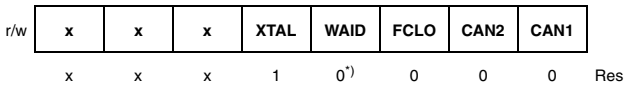
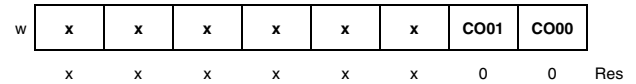
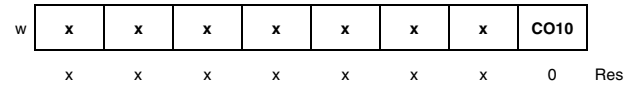
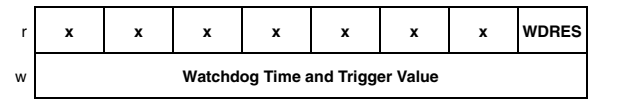
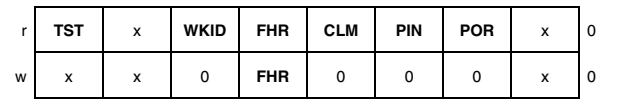
Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
CSW0	Clock, Supply and Watchdog Register 0	1F00		6.
CR	Control Register	1F01		
SR0	Standby Register 0	1F08		
SR1	Standby Register 1	1F09		
SR2	Standby Register 2	1F0A		
SR3	Standby Register 3	1F0B		
CO0SEL	Clock Out 0 Selection	1F14		
CO1SEL	Clock Out 1Selection	1F15		
CSW1	Clock, Supply and Watchdog Register 1	1F60		
CSW2	Clock, Supply and Watchdog Register 2	1F61		

Table 27–8: Debug Register

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
DBG	Debug Register	1F0C	r/w: x x x x x x x DCS 0 x x x x x x x 0 POR	8.3.5.

Table 27–9: DIGITbus

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
DGC0	Control Register 0	1FF0	r/w: RUN GBC ACT RXO X PSC 2 to 0 0 0 0 0 x 0 0 0 Res	23.4.
DGC1	Control Register 1	1FF1	r/w: INTE ENEM ENOF x PHASE 0 0 0 x 0 0 0 0 Res	
DGS0	Status Register 0	1FF2	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	1FF3	w: Transmit More Data r: RDL NEM FTYP EOF x LEN2 to 0 0 0 x x x x x x Res	
DGTL	Tx Length Register	1FF4	w: x FLUSH x x x LEN2 to 0 x 0 x x x 0 0 0 Res r: x EMPTY x x x x x x x 1 x x x x x x Res	
DGS1TA	Status 1 & Tx Address Register	1FF5	w: Transmit Address r: STATE PW5 to 0 0 1 0 0 0 0 0 0 Res	
DGTD	Tx Data Register	1FF6	w: Transmit Data x x x x x x x x Res	
DGRTMA	Rx Field & Tx More Address Register	1FF7	w: Transmit More Address r: Receive Field x x x x x x x x Res	

Table 27–10: DMA

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
DCS	DMA Control and Status Register	1FE8	r/w: x x x DTA DSI DCC STP BSY 0 0 0 0 0 0 0 0 Res	18.2.
DIC	DMA Initial Configuration Register	1FE9	w: x WS2 WS1 WS0 x x x WSA 0 0 0 0 0 0 0 0 Res	
DSA	DMA Start Address	1FEA	w: Bit 7 to 0	
		1FEB	w: Bit 15 to 8	
		1FEC	w: Bit 23 to 16 0 0 0 0 0 0 0 0 Res	
DEA	DMA End Address	1FED	w: Bit 7 to 0	
		1FEE	w: Bit 15 to 8	
		1FEF	w: Bit 23 to 16 0 0 0 0 0 0 0 0 Res	

Table 27–11: EMI Reduction Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
ERMC	EMI Reduction Module Control Register	1F02	r: x x x x x x x CLKSEL w: x x 0 0 0 0 0 CLKSEL x x 0 0 1 0 0 0 Res	4.4.8.

Table 27–12: High Current Port 0

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
H0NS	High Current Port 0 Normal/Special Register	1F80	w: x x N/S5 N/S4 N/S3 N/S2 N/S1 N/S0 0 0 0 0 0 0 0 0 Res	11.5.
H0TRI	High Current Port 0 Tristate Register	1F81	w: x x TRI5 TRI4 TRI3 TRI2 TRI1 TRI0 0 0 0 0 0 0 0 0 Res	
H0D	High Current Port 0 Data Register	1F82	r/w: x x D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	

Table 27–13: High Current Port 1

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
H1NS	High Current Port 1 Normal/Special Register	1F84	w: x x N/S5 N/S4 N/S3 N/S2 N/S1 N/S0 0 0 0 0 0 0 0 0 Res	11.5.
H1TRI	High Current Port 1 Tristate Register	1F85	w: x x TRI5 TRI4 TRI3 TRI2 TRI1 TRI0 0 0 0 0 0 0 0 0 Res	
H1D	High Current Port 1 Data Register	1F86	r/w: x x D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	

Table 27–14: High Current Port 2

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
H2NS	High Current Port 2 Normal/Special Register	1F88	w: x x N/S5 N/S4 N/S3 N/S2 N/S1 N/S0 0 0 0 0 0 0 0 0 Res	11.5.
H2TRI	High Current Port 2 Tristate Register	1F89	w: x x TRI5 TRI4 TRI3 TRI2 TRI1 TRI0 0 0 0 0 0 0 0 0 Res	
H2D	High Current Port 2 Data Register	1F8A	r/w: x x D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	

Table 27–15: High Current Port 3

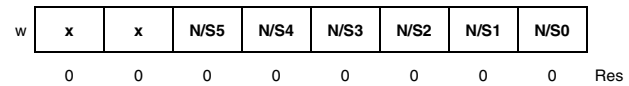
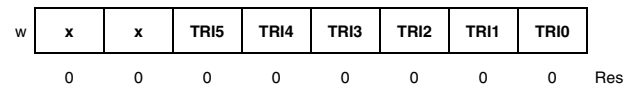
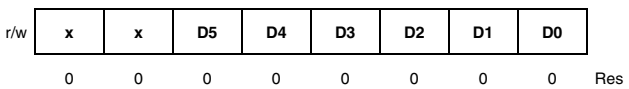
Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
H3NS	High Current Port 3 Normal/Special Register	1F90		11.5.
H3TRI	High Current Port 3 Tristate Register	1F91		
H3D	High Current Port 3 Data Register	1F92		

Table 27–16: Interrupt Controller

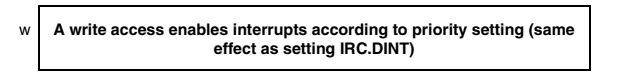
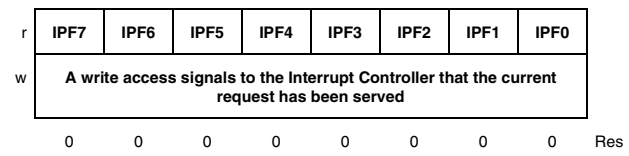
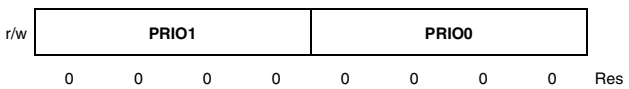
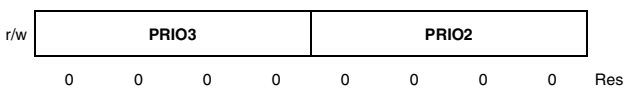
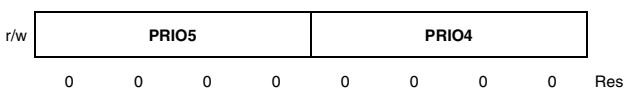
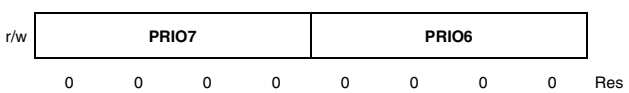
Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
IRE	Interrupt Enable Register			
IRC	Interrupt Control Register	1F20		10.2.
IRRET	Interrupt Pending and Return Register	1F21		
IRPRI10	Interrupt Priority, Inputs 0 and 1	1F22		
IRPRI32	Interrupt Priority, Inputs 2 and 3	1F23		
IRPRI54	Interrupt Priority, Inputs 4 and 5	1F24		
IRPRI76	Interrupt Priority, Inputs 6 and 7	1F25		

Table 27–16: Interrupt Controller

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
IRPRI98	Interrupt Priority, Inputs 8 and 9	1F26	r/w: PRIO9 | PRIO8 0 0 0 0 0 0 0 0 Res	10.2.
IRPRIBA	Interrupt Priority, Inputs 10 and 11	1F27	r/w: PRIO11 | PRIO10 0 0 0 0 0 0 0 0 Res	
IRPRIDC	Interrupt Priority, Inputs 12 and 13	1F28	r/w: PRIO13 | PRIO12 0 0 0 0 0 0 0 0 Res	
IRPRIFE	Interrupt Priority, Inputs 14 and 15	1F29	r/w: PRIO15 | PRIO14 0 0 0 0 0 0 0 0 Res	
IRP	Interrupt Pending Register	1F2A	r: IPF15 | IPF14 | IPF13 | IPF12 | IPF11 | IPF10 | IPF9 | IPF8 0 0 0 0 0 0 0 0 Res	
IRPM0	Interrupt Port Mode Register 0	1F2B	w: Pit3 | Pit2 | PIT1 | PIT0 0 0 0 0 0 0 0 0 Res	
IRPP	Interrupt Port Prescaler Register	1F2C	w: x | x | x | x | x | x | P1INT32 | P0INT4 0 0 Res	

Table 27–17: Memory Banking

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
ABR	Alternative Banking Register	1F0F	r/w: Alternative Bank Address 0 0 0 0 0 0 0 1 Res	5.2.2.

Table 27–18: Multiplier

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
MULCAND	Multiplicand	1EA0	r/w: multiplicand x x x x x x x x Res	
MULPLIER	Multiplier	1EA1	r/w: multiplier x x x x x x x x Res	
MULPROD	Multiplication Product	1EA3	r: Product high byte 1	
		1EA2	r: Product low byte 0 x Res	

Table 27–19: Patch Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
PAR0	Patch Address Register 0	1E64	w: PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0 1 1 1 1 1 1 1 1 Res	9.2.
PAR1	Patch Address Register 1	1E65	w: PA15 PA14 PA13 PA12 PA11 PA10 PA9 PA8 1 1 1 1 1 1 1 1 Res	
PAR2	Patch Address Register 2	1E66	w: PA23 PA22 PA21 PA20 PA19 PA18 PA17 PA16 1 1 1 1 1 1 1 1 Res	
PDR	Patch Data Register	1E67	w: PD7 PD6 PD5 PD4 PD3 PD2 PD1 PD0 0 0 0 0 0 0 0 0 Res	
PER0	Patch Enable Register 0	1E68	w: PSEL6 PSEL5 PSEL4 PSEL3 PSEL2 PSEL1 PSEL0 PMEN 0 0 0 0 0 0 0 0 Res	
PER1	Patch Enable Register 1	1E69	w: x x x x x PSEL9 PSEL8 PSEL7 x x x x x 0 0 0 Res	

Table 27–20: Port 0

Mnemonic	Register Name	Addr. (hex)	Register Configuration								Section
			7	6	5	4	3	2	1	0	
P0D	Port 0 Data Register	1F7E	r	x	x	D5	D4	D3	D2	D1	x		11.1.
			x	x	x	x	x	x	x	x	Res		

Table 27–21: Power Saving Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
WUS	Wake-Up Source Register	1E71	r/w RTC x x x x x WP9 WP8 1	8.2.
		1E70	r/w WP7 WP6 WP5 WP4 WP3 WP2 WP1 WP0 0	
			No HW reset Res	
SSR	Sub Second Reload Register	1E77	r/w x x x x x x x x 3	
		1E76	r/w x x x x Bit 19 to 16 2	
		1E75	r/w Bit 15 to 8 1	
		1E74	r/w Bit 7 to 0 0	
			No HW reset Res	
SSC	Sub Second Counter	1E7B	r x x x x x x x x 3	
		1E7A	r x x x x Bit 19 to 16 2	
		1E79	r Bit 15 to 8 1	
		1E78	r Bit 7 to 0 0	
			No HW reset Res	
RTC	Real Time Counter	1E7F	r/w x x x x x x x x 3	
		1E7E	r/w x x x HR 2	
		1E7D	r/w x x MIN 1	
		1E7C	r/w x x SEC 0	
			No HW reset Res	
WPM0	Wake Port Mode Register	1E80	r/w x MOD1 x MOD0 0	
WPM2		1E81	No HW reset Res	
WPM4		1E82		
WPM6		1E83		
WPM8		1E84		
WSC	Wake Source Control	1E88	r/w x x x x x AST RTC P 0	
			0x00 after UVDD power on Res	
OSC	Oscillator Source Register	1E90	r/w RC XK XM x LD PRE SRC 0	
			1 1 1 No HW reset Res	
RTCC	RTC Control Register	1E94	r/w x x x SEL 0	
			No HW reset Res	

Table 27–21: Power Saving Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
POL	Polling Register	1E99	r/w | x | CLK | x | PER | 1	
		1E98	r/w | ENA | OE | x | DEL | 0 0x00 Res	
SMX	Signal Multiplexer Register	1E9C	r/w | BYP | x | x | x | x | MUX | 0 0x00 Res	

Table 27–22: Pulse Width Modulator

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
PWM0	PWM 0 Register	1F50	w | Pulse width value 0 0 0 0 0 0 0 0 Res	14.2.
PWM1	PWM 1 Register	1F51		
PWM2	PWM 2 Register	1F52		
PWM3	PWM 3 Register	1F5E		
PWM4	PWM 4 Register	1F5F		

Table 27–23: Serial Synchronous Peripheral Interface 1

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
SPI1D	SPI 1 Data Register	1F12	r/w | Bit 7 to 0 of Rx/Tx Data 0 0 0 0 0 0 0 0 Res	19.2.
SPI1M	SPI 1 Mode Register	1F13	r/w | BIT8 | LEN9 | RXSEL | INTERN | NEDGE | x | CSF 0 0 0 0 0 0 0 0 Res	

Table 27–24: Stepper Motor Module

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
SMVC	SMM Control Register	1F5A	w: x, x, SEL, x, QUAD x x 0 0 0 x 0 0 Res	16.2.
SMVSIN	SMM Sine Register	1F5B	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	1F5C	w: 8bit Cosine Value 0 0 0 0 0 0 0 0 Res	
SMVCMP	SMM Comparator Register	1F5D	r/w: x, x, ACRD, ACRB, x, ACRE, ACRC, ACRA x x 0 0 x 0 0 0 Res	

Table 27–25: Test Registers

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
TST3	Test Register 3	1FFD	w: For testing purposes only 0 0 0 0 0 0 0 0 Res	6.4.
TST1	Test Register 1	1FFE	w: For testing purposes only 0 0 0 0 0 0 0 0 Res	
TST2	Test Register 2	1FFF	w: For testing purposes only 0 0 0 0 0 0 0 0 Res	

Table 27–26: Timer

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
TIM0L	Timer 0 low byte	1F4E	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	13.
TIM0H	Timer 0 high byte	1F4F	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	1F54	w: Reload value 0 0 0 0 0 0 0 0 Res	
TIM2	Timer 2 Register	1F55		

Table 27–27: Universal Asynchronous Receiver Transmitter 0

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
UA0D	UART 0 Data Register	1FA0	r: Receive register w: Transmit register x x x x x x x x Res	20.3.
UA0C	UART 0 Control and Status Register	1FA1	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	
UA0BR0	UART 0 Baudrate Register low byte	1FA2	w: Bit 7 to 0 of Baud Rate 0 0 0 0 0 0 0 0 Res	
UA0BR1	UART 0 Baudrate Register high byte	1FA3	w: x x x Bit 12 to 8 of Baud Rate - - - 0 0 0 0 0 Res	
UA0IM	UART 0 Interrupt Mask Register	1FA4	w: x x x x x ADR BRK RCVD - - - - - 0 0 0 Res	
UA0CA	UART 0 Compare Address Register	1FA5	w: Bit 7 to 0 of address 0 0 0 0 0 0 0 0 Res	
UA0IF	UART 0 Interrupt Flag Register	1FA6	r: Test Test Test Test Test ADR BRK RCVD - - - - - x 0 0 Res	

Table 27–28: Universal Asynchronous Receiver Transmitter 1

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
UA1D	UART 1 Data Register	1F18	r: Receive register w: Transmit register x x x x x x x x Res	20.3.
UA1C	UART 1 Control and Status Register	1F19	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	
UA1BR0	UART 1 Baudrate Register low byte	1F1A	w: Bit 7 to 0 of Baud Rate 0 0 0 0 0 0 0 0 Res	
UA1BR1	UART 1 Baudrate Register high byte	1F1B	w: x x x Bit 12 to 8 of Baud Rate - - - 0 0 0 0 0 Res	
UA1IM	UART 1 Interrupt Mask Register	1F1C	w: x x x x x ADR BRK RCVD - - - - - 0 0 0 Res	
UA1CA	UART 1 Compare Address Register	1F1D	w: Bit 7 to 0 of address 0 0 0 0 0 0 0 0 Res	
UA1IF	UART 1 Interrupt Flag Register	1F1E	r: Test Test Test Test Test ADR BRK RCVD - - - - - x 0 0 Res	

Table 27–29: Universal Asynchronous Receiver Transmitter 2

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
UA2D	UART 2 Data Register	1F64	r: Receive register w: Transmit register x x x x x x x x Res	20.3.
UA2C	UART 2 Control and Status Register	1F65	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	
UA2BR0	UART 2 Baud rate Register low byte	1F66	w: Bit 7 to 0 of Baud Rate 0 0 0 0 0 0 0 0 Res	
UA2BR1	UART 2 Baud rate Register high byte	1F67	w: x x x Bit 12 to 8 of Baud Rate - - - 0 0 0 0 0 Res	
UA2IM	UART 2 Interrupt Mask Register	1F68	w: x x x x x ADR BRK RCVD - - - - - 0 0 0 Res	
UA2CA	UART 2 Compare Address Register	1F69	w: Bit 7 to 0 of address 0 0 0 0 0 0 0 0 Res	
UA2IF	UART 2 Interrupt Flag Register	1F6A	r: Test Test Test Test Test ADR BRK RCVD - - - - - x 0 0 Res	

Table 27–30: Universal Port 1

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
U1D	Universal Port 1 Data Register	1F98	r/w: D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	11.3.
U1SEG10	Universal Port 1 Segments of U1.0, U1.1	1F99	w: x N/S1 TRI1 x x N/S0 TRI0 x Port w: LCDSLV LCD 0 0 1 0 0 0 1 0 Res	
U1SEG32	Universal Port 1 Segments of U1.2, U1.3	1F9A	w: x N/S3 TRI3 x x N/S2 TRI2 x Port 0 0 1 0 0 0 1 0 Res	
U1M30	Universal Port 1 Mode of U1.0 to U1.3	1F9B	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U1SEG54	Universal Port 1 Segments of U1.4, U1.5	1F9C	w: x N/S5 TRI5 x x N/S4 TRI4 x Port w: SEG5.3 SEG5.2 SEG5.1 SEG5.0 SEG4.3 SEG4.2 SEG4.1 SEG4.0 LCD 0 0 1 0 0 0 1 0 Res	
U1M54	Universal Port 1 Mode of U1.4, U1.5	1F9D	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U1SEG76	Universal Port 1 Segments of U1.6, U1.7	1F9E	w: x N/S7 TRI7 x x N/S6 TRI6 x Port w: SEG7.3 SEG7.2 SEG7.1 SEG7.0 SEG6.3 SEG6.2 SEG6.1 SEG6.0 LCD 0 0 1 0 0 0 1 0 Res	
U1M76	Universal Port 1 Mode of U1.6, U1.7	1F9F	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	

Table 27–31: Universal Port 2

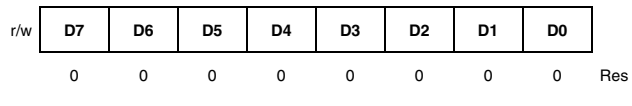
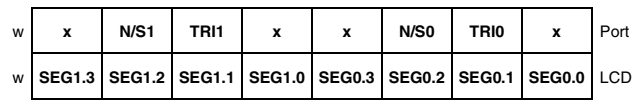
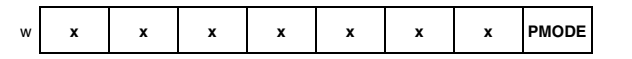
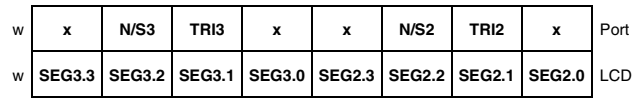
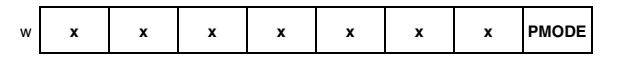
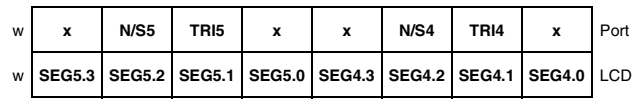
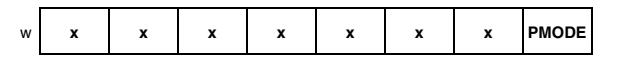
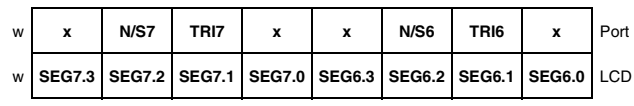
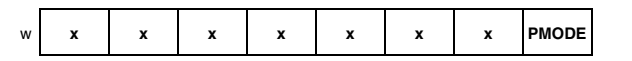
Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
U2D	Universal Port 2 Data Register	1F30		11.3.
U2SEG10	Universal Port 2 Segments of U2.0, U2.1	1F32		
U2M10	Universal Port 2 Mode of U2.0, U2.1	1F33		
U2SEG32	Universal Port 2 Segments of U2.2, U2.3	1F34		
U2M32	Universal Port 2 Mode of U2.2, U2.3	1F35		
U2SEG54	Universal Port 2 Segments of U2.4, U2.5	1F36		
U2M54	Universal Port 2 Mode of U2.4, U2.5	1F37		
U2SEG76	Universal Port 2 Segments of U2.6, U2.7	1F38		
U2M76	Universal Port 2 Mode of U2.6, U2.7	1F39		

Table 27–32: Universal Port 3

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
U3D	Universal Port 3 Data Register	1FAC	r/w D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	11.3.
U3SEG10	Universal Port 3 Segments of U3.0, U3.1	1FAE	w x N/S1 TRI1 x x N/S0 TRI0 x Port w SEG1.3 SEG1.2 SEG1.1 SEG1.0 SEG0.3 SEG0.2 SEG0.1 SEG0.0 LCD 0 0 1 0 0 0 1 0 Res	
U3M10	Universal Port 3 Mode of U3.0, U3.1	1FAF	w x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U3SEG32	Universal Port 3 Segments of U3.2, U3.3	1FB0	w x N/S3 TRI3 x DPM2 N/S2 TRI2 x Port w SEG3.3 SEG3.2 SEG3.1 SEG3.0 SEG2.3 SEG2.2 SEG2.1 SEG2.0 LCD 0 0 1 0 0 0 1 0 Res	
U3M32	Universal Port 3 Mode of U3.2, U3.3	1FB1	w x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U3SEG54	Universal Port 3 Segments of U3.4, U3.5	1FB2	w x N/S5 TRI5 x x N/S4 TRI4 x Port w SEG5.3 SEG5.2 SEG5.1 SEG5.0 SEG4.3 SEG4.2 SEG4.1 SEG4.0 LCD 0 0 1 0 0 0 1 0 Res	
U3M54	Universal Port 3 Mode of U3.4, U3.5	1FB3	w x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U3SEG76	Universal Port 3 Segments of U3.6, U3.7	1FB4	w x N/S7 TRI7 x x N/S6 TRI6 x Port w SEG7.3 SEG7.2 SEG7.1 SEG7.0 SEG6.3 SEG6.2 SEG6.1 SEG6.0 LCD 0 0 1 0 0 0 1 0 Res	
U3M76	Universal Port 3 Mode of U3.6, U3.7	1FB5	w x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	

Table 27–33: Universal Port 4

Mnemonic	Register Name	Addr. (hex)	Register Configuration									Section
				7	6	5	4	3	2	1	0	
U4D	Universal Port 4 Data Register	1FB8	r/w	D7	D6	D5	D4	D3	D2	D1	D0	11.3.
				0	0	0	0	0	0	0	0	Res
U4SEG10	Universal Port 4 Segments of U4.0, U4.1	1FBA	w	x	N/S1	TRI1	x	x	N/S0	TRI0	x	Port
			w	SEG1.3	SEG1.2	SEG1.1	SEG1.0	SEG0.3	SEG0.2	SEG0.1	SEG0.0	LCD
				0	0	1	0	0	0	1	0	Res
U4M10	Universal Port 4 Mode of U4.0, U4.1	1FBB	w	x	x	x	x	x	x	x	PMODE	
				0	0	0	0	0	0	0	1	Res
U4SEG32	Universal Port 4 Segments of U4.2, U4.3	1FBC	w	x	N/S3	TRI3	x	x	N/S2	TRI2	x	Port
			w	SEG3.3	SEG3.2	SEG3.1	SEG3.0	SEG2.3	SEG2.2	SEG2.1	SEG2.0	LCD
				0	0	1	0	0	0	1	0	Res
U4M32	Universal Port 4 Mode of U4.2, U4.3	1FBD	w	x	x	x	x	x	x	x	PMODE	
				0	0	0	0	0	0	0	1	Res
U4SEG54	Universal Port 4 Segments of U4.4, U4.5	1FBE	w	x	N/S5	TRI5	x	x	N/S4	TRI4	x	Port
			w	SEG5.3	SEG5.2	SEG5.1	SEG5.0	SEG4.3	SEG4.2	SEG4.1	SEG4.0	LCD
				0	0	1	0	0	0	1	0	Res
U4M54	Universal Port 4 Mode of U4.4, U4.5	1FBF	w	x	x	x	x	x	x	x	PMODE	
				0	0	0	0	0	0	0	1	Res
U4SEG76	Universal Port 4 Segments of U4.6, U4.7	1FC0	w	x	N/S7	TRI7	x	x	N/S6	TRI6	x	Port
			w	SEG7.3	SEG7.2	SEG7.1	SEG7.0	SEG6.3	SEG6.2	SEG6.1	SEG6.0	LCD
				0	0	1	0	0	0	1	0	Res
U4M76	Universal Port 4 Mode of U4.6, U4.7	1FC1	w	x	x	x	x	x	x	x	PMODE	
				0	0	0	0	0	0	0	1	Res

Table 27–34: Universal Port 5

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
U5D	Universal Port 5 Data Register	1FC4	r/w: D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	11.3.
U5SEG10	Universal Port 5 Segments of U5.0, U5.1	1FC6	w: x N/S1 TRI1 x x N/S0 TRI0 x Port w: SEG1.3 SEG1.2 SEG1.1 SEG1.0 SEG0.3 SEG0.2 SEG0.1 SEG0.0 LCD 0 0 1 0 0 0 1 0 Res	
U5M10	Universal Port 5 Mode of U5.0, U5.1	1FC7	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U5SEG32	Universal Port 5 Segments of U5.2, U5.3	1FC8	w: x N/S3 TRI3 x x N/S2 TRI2 x Port w: SEG3.3 SEG3.2 SEG3.1 SEG3.0 SEG2.3 SEG2.2 SEG2.1 SEG2.0 LCD 0 0 1 0 0 0 1 0 Res	
U5M32	Universal Port 5 Mode of U5.2, U5.3	1FC9	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U5SEG54	Universal Port 5 Segments of U5.4, U5.5	1FCA	w: x N/S5 TRI5 x x N/S4 TRI4 x Port w: SEG5.3 SEG5.2 SEG5.1 SEG5.0 SEG4.3 SEG4.2 SEG4.1 SEG4.0 LCD 0 0 1 0 0 0 1 0 Res	
U5M54	Universal Port 5 Mode of U5.4, U5.5	1FCB	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U5SEG76	Universal Port 5 Segments of U5.6, U5.7	1FCC	w: x N/S7 TRI7 x x N/S6 TRI6 x Port w: SEG7.3 SEG7.2 SEG7.1 SEG7.0 SEG6.3 SEG6.2 SEG6.1 SEG6.0 LCD 0 0 1 0 0 0 1 0 Res	
U5M76	Universal Port 5 Mode of U5.6, U5.7	1FCD	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	

Table 27–35: Universal Port 6

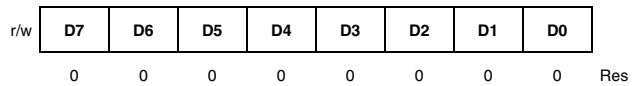
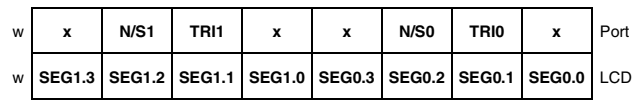
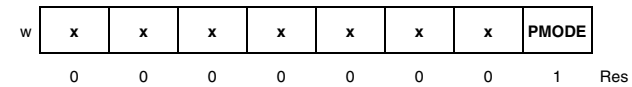
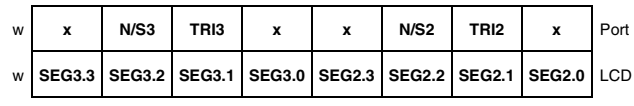
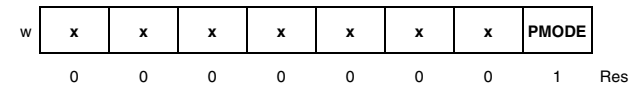
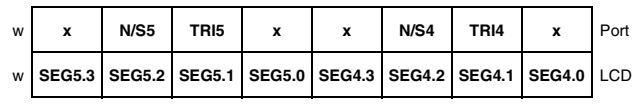
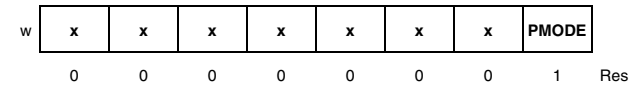
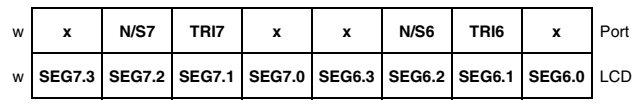
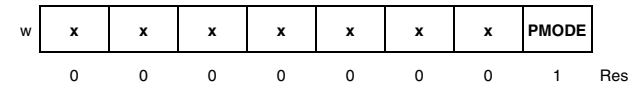
Mnemonic	Register Name	Addr. (hex)	Register Configuration	Section
U6D	Universal Port 6 Data Register	1FD0		11.3.
U6SEG10	Universal Port 6 Segments of U6.0, U6.1	1FD2		
U6M10	Universal Port 6 Mode of U6.0, U6.1	1FD3		
U6SEG32	Universal Port 6 Segments of U6.2, U6.3	1FD4		
U6M32	Universal Port 6 Mode of U6.2, U6.3	1FD5		
U6SEG54	Universal Port 6 Segments of U6.4, U6.5	1FD6		
U6M54	Universal Port 6 Mode of U6.4, U6.5	1FD7		
U6SEG76	Universal Port 6 Segments of U6.6, U6.7	1FD8		
U6M76	Universal Port 6 Mode of U6.6, U6.7	1FD9		

Table 27–36: Universal Port 7

Mnemonic	Register Name	Addr. (hex)	Register Configuration 7 6 5 4 3 2 1 0	Section
U7D	Universal Port 7 Data Register	1FDC	r/w: D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0 0 0 0 0 Res	11.3.
U7SEG10	Universal Port 7 Segments of U7.0, U7.1	1FDE	w: x N/S1 TRI1 x x N/S0 TRI0 x Port w: SEG1.3 SEG1.2 SEG1.1 SEG1.0 SEG0.3 SEG0.2 SEG0.1 SEG0.0 LCD 0 0 1 0 0 0 1 0 Res	
U7M10	Universal Port 7 Mode of U7.0, U7.1	1FDF	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	
U7SEG32	Universal Port 7 Segments of U7.2, U7.3	1FE0	w: x N/S3 TRI3 x x N/S2 TRI2 x Port w: SEG3.3 SEG3.2 SEG3.1 SEG3.0 SEG2.3 SEG2.2 SEG2.1 SEG2.0 LCD 0 0 1 0 0 0 1 0 Res	
U7M32	Universal Port 7 Mode of U7.2, U7.3	1FE1	w: x x x x x x x PMODE 0 0 0 0 0 0 0 1 Res	

28. Differences

This chapter describes differences of this document to predecessor document “CDC16xxF-E Automotive Controller Family User Manual”, Feb. 17, 2003, 6251-606-1AI.

#	Section	Description
1	Features	Table 1–1: "CDC16xxF Family Feature List" on page 5: Name and features of “Example E-Family” changed into “CDC1631F-E”, Multiplier, 8 by 8 bit added.
		Fig. 1–1: “Block diagram of CDC1605F-E/CDC1607F-E” on page 10: Multiplier, 8 by 8 bit added
2	External Components	New section with Fig. 2–4: “Recommended external supply and quartz connection for low electromagnetic interference (EMI)” out of section “Pin Function Description”. Value of C at RESETQ changed from 47 µ to 47 n, value of C at VREF changed from 10 µ to 10 n and text added.
3	Pin Circuits	New chapter
4	Electrical Data	Characteristics: mistake in writing corrected: Outputs, V_{ol}, Port Low Output Voltage, H-ports, ... “Io = 40 mA@TCASE = 40 °C” changed to “Io = 40 mA@TCASE = –40 °C”
5	CPU and Clock System	Table 4–1: "Major Differences between Processors and Modes" on page 32: Item “flags after reset” for 65C816 Emulation changed from “D not modified” to “D=0”.
6	Boot System	Principle of operation: Boot Loader check if started by wake-up from Power Saving Mode added.
		The Boot Loader: Fig. 5–7: “Boot Loader flow-chart” on page 46: Boot Loader check if started by wake-up from Power Saving Mode (CSW2.WKID = ’1’) added.
		Used RAM: Page 2 demand added
7	Multiplier	New Chapter
8	Core Logic	Table 6–1: "Control byte source" on page 49: Updated / minimized
		Fig. 6–1: “UVDD Section” on page 51: Updated
9	Power-Saving Module (PSM)	Timing: Fig. 8–3: “Power-on Reset” on page 67: updated
10	Interrupt Controller (IR)	Registers: New: Interrupts Enable Register (IRE)
		Interrupt Assignment: INT-MUX 1 assignment changed
11	CAN Manual	Updated from version LCAN0009 to version LCAN000F: Mainly bit “Bus-Off Stop Select” in control register and “Error Status Mask” register added
12	Hardware Options	Listing of Dedicated Addresses and Corresponding Hardware Options: INT-MUX 1 assignment changed
13	Register Cross Reference Table V2.1	New registers added

#	Section	Description
14	Register Quick Reference	New registers added
15	Differences	New Chapter

29. Data Sheet History

1. Advance Information: “CDC16xxF-E Automotive Controller Family User Manual”, Feb. 17, 2003, 6251-606-1AI. First release of the advance information. Originally created for the HW version CDC16xxF-E1.

2. Advance Information: “CDC16xxF-E Automotive Controller Family User Manual”, March 31, 2003, 6251-606-2AI. Second release of the advance information. Originally created for the HW version CDC16xxF-E2.

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. 6251-606-2AI