M44C260

MARC4 – 4-bit Microcontroller

The M44C260 is a member of the TEMIC family of 4-bit single chip microcontrollers. It contains ROM, RAM, EEPROM, parallel I/O ports, 1 timer with watchdog function, 2 × 8/16-bit multifunction timer/counter and the on-chip clock generation.

Features

- 4-bit HARVARD architecture
- 1 µs instruction cycle
- 4K × 8-bit application ROM
- 256 × 4-bit RAM
- 16 × 8-bit EEPROM
- 16 bidirectional I/O's
- 8 hard and software interrupt levels
- 2 × 8-bit multifunction timer/counter
- Interval timer with watchdog
- 32 kHz on-chip oscillator

Benefits

- Low power consumption
- Power down mode < 1 µA
- 2.4 to 6.2 V supply voltage
- Self test functions
- High level programming language in qFORTH

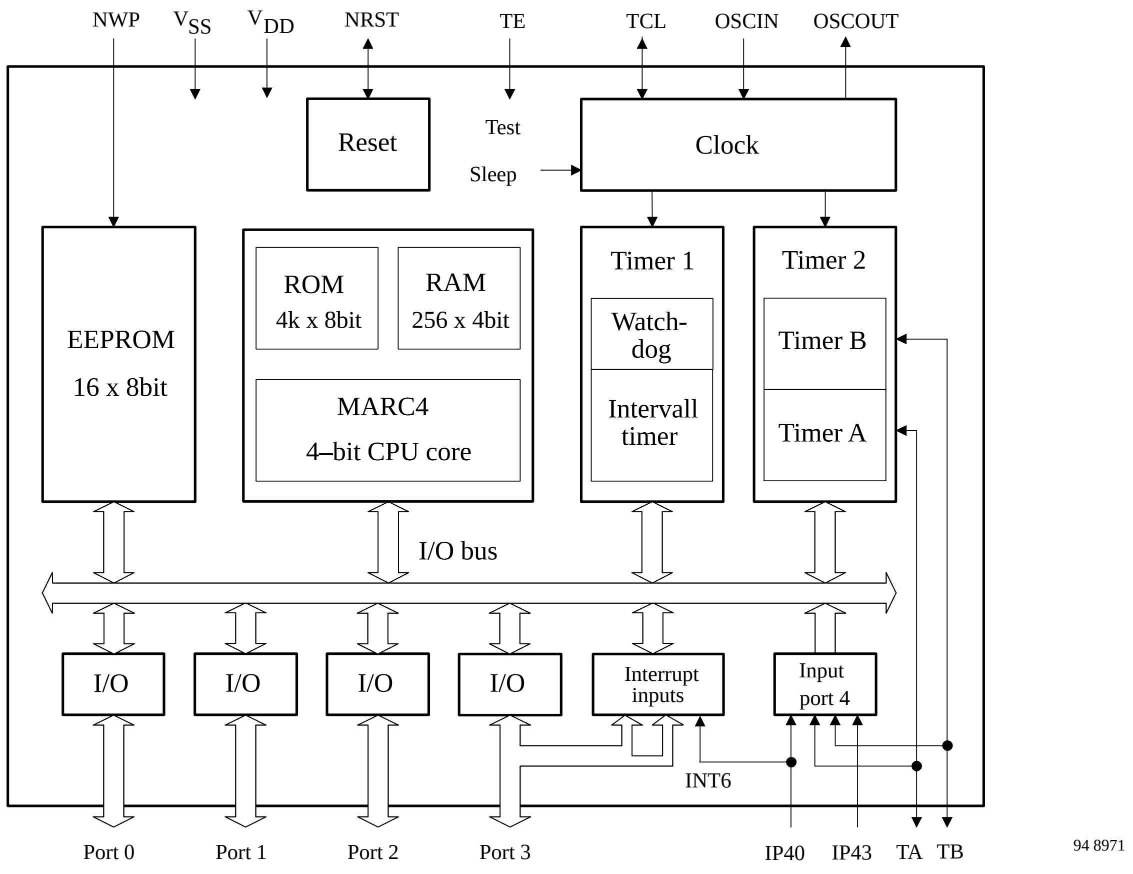


Figure 1. Block diagram

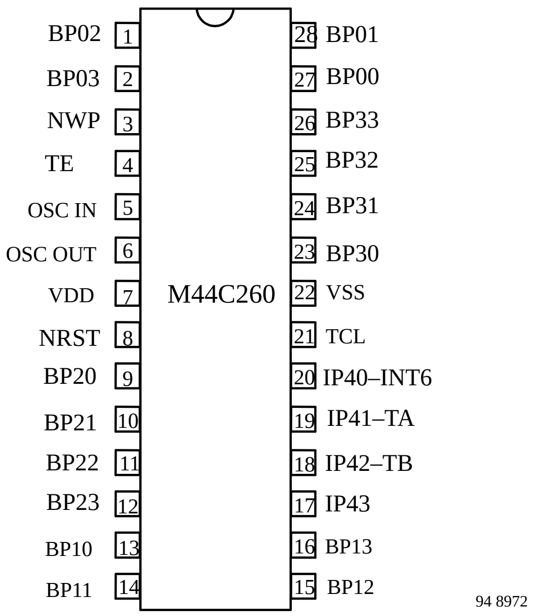


Figure 2. Pin connections SSO28–FN (other SSO packages on request)

Table 1. Pin description

Name	Function
V_{DD}	Power supply voltage +2.4 to +6.2 V
V_{SS}	Circuit ground
BP00 – BP03	4 bidirectional I/O lines of port 0 *
BP10 – BP13	4 bidirectional I/O lines of port 1 *
BP20 – BP23	4 bidirectional I/O lines of port 2 *
BP30 – BP33	4 bidirectional I/O lines of port 3 with alternate interrupt function. A negative transition on BP30/BP31 requests an INT2-, and on BP32/BP33 an INT3-interrupt if the corresponding interrupt-mask is set.
IP40-INT6	Input port 40 line/interrupt 6 input * A negative transition on this input requests an INT6 interrupt if the IM6 mask bit is set.
IP41-TA	Timer/counter I/O/Input port 41 line * This line can be used as programmable I/O of counter A or as port 41 input.
IP42-TB	Timer/counter I/O/input port 42 line * This line can be used as programmable I/O of counter B or as port 42 input.
IP43	Input port 43 line *)
NWP	EEPROM write protect input, a logic low on this input protects EEPROM rows 12 to 15.
OSCIN	Oscillator input (32-kHz crystal).
OSCOUT	Oscillator output (32-kHz crystal).
NRST	Reset input/output, a logic low on this pin resets the device. An internal watchdog reset is indicated by a low level on this pin.
TCL	External system clock I/O. This pin can be used as input to provide the C with an external clock or as output of the internal system clock.
TE	Testmode input. This input is used to control the test modes and the function of the TCL pin.

*) The I/O ports have CMOS output buffers. As input they are available with pull-up or pull-down resistors. Please see the order information.

Contents

Contents (continued)

1 MARC4 Architecture

1.1 General Description

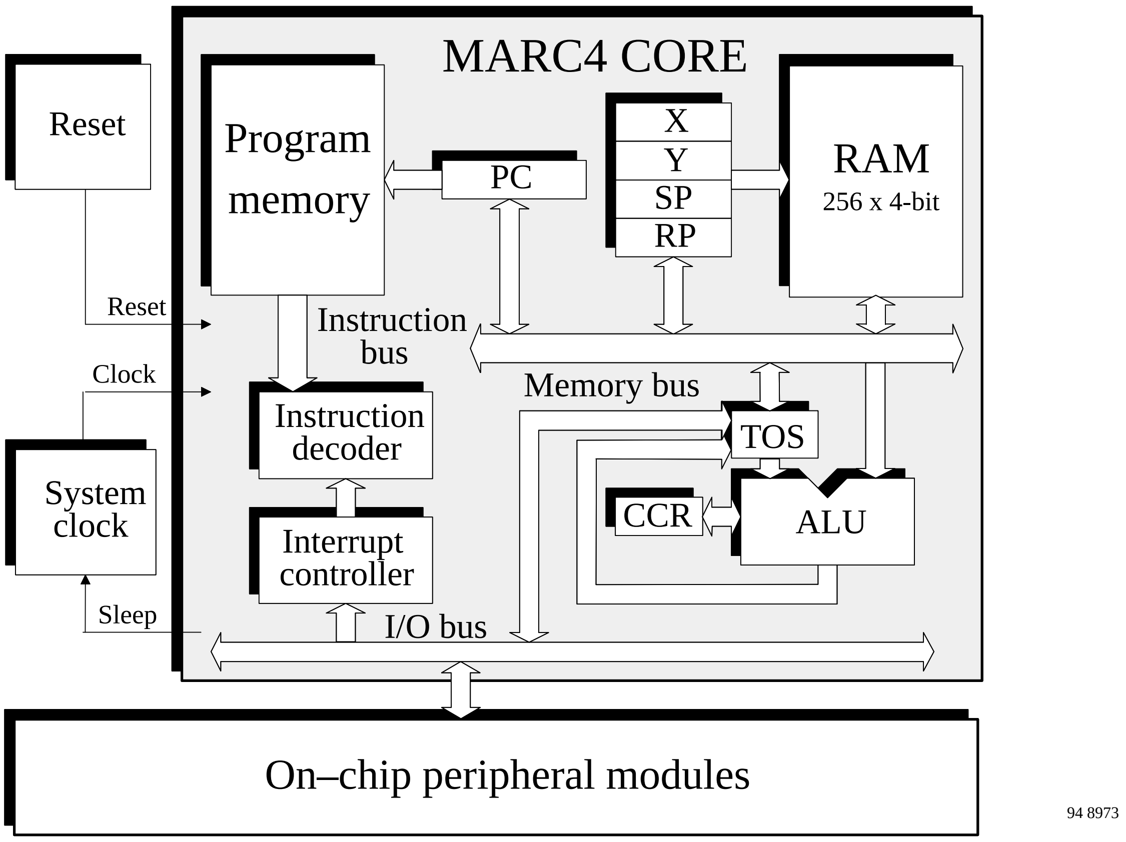


Figure 3. MARC4 core

The MARC4 microcontroller consists of an advanced stack based 4-bit CPU core and on-chip peripherals. The CPU is based on the HARVARD architecture with physically separate program memory (ROM) and data memory (RAM). Three independent buses, the instruction bus, the memory bus and the I/O bus are used for parallel communication between ROM, RAM and peripherals. This enhances program execution speed by allowing both instruction prefetching, and a simultaneous communication to the on-chip peripheral circuitry. The integrated powerful interrupt controller with eight prioritized interrupt levels, supports fast processing of hardware events.

The MARC4 is designed for the high level programming language qFORTH. The core contains both FORTH stacks, expression stack and return stack. This architecture allows high level language programming without any loss in efficiency or code density.

1.2 Components of MARC4 Core

The core contains ROM, RAM, ALU, program counter, RAM address register, instruction decoder and interrupt controller. The following sections describe each of this parts.

1.2.1 Program Memory (ROM)

The program memory (ROM) is mask programmed with the customer application program during the fabrication of the microcontroller. The ROM is addressed by a 12-bit wide program counter, thus limiting the program size to a maximum of 4 Kbytes. An additional 1 Kbyte ROM is available for test software only.

The user ROM starts with a 512 byte segment (zero page) which contains predefined start addresses for interrupt service routines and special subroutines accessible with single byte instructions (SCALL). The corresponding memory map is shown in figure 4.

Look-up tables of constants can also be held in ROM and are accessed via the MARC4's built-in TABLE instruction.

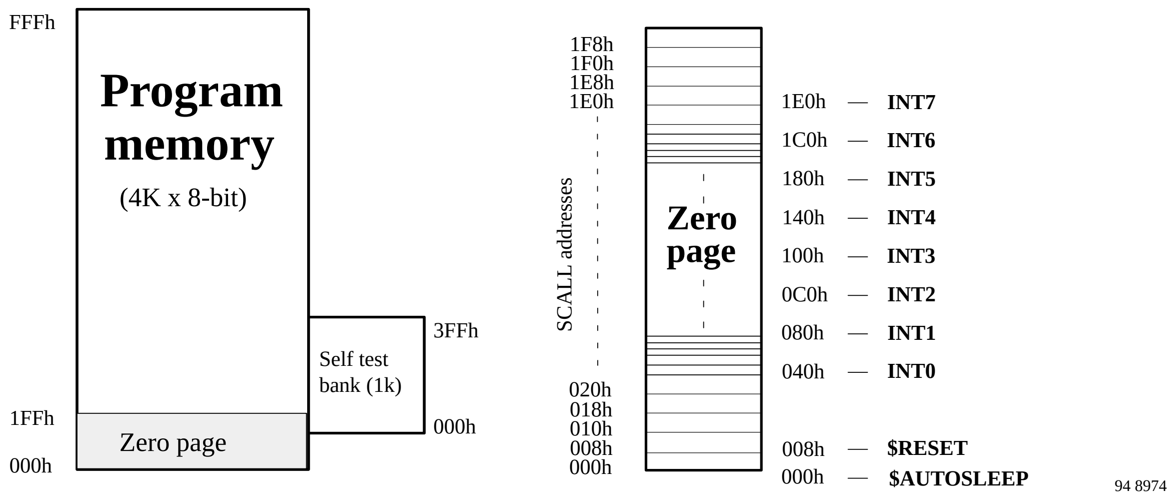


Figure 4. Progam memory map

1.2.2 Data Memory (RAM)

The MARC4 contains 256 x 4-bit wide static random access memory (RAM). It is used for the expression stack, the return stack and data memory for variables and arrays. The RAM is addressed by any of the four 8-bit wide RAM address registers SP, RP, X and Y.

- Expression Stack

The 4-bit wide expression stack is addressed with the expression stack pointer (SP). All arithmetic, I/O and memory reference operations take their operands from, and return their result to the expression stack. The MARC4 performs the operations with the top of stack items (TOS and TOS-1). The TOS register contains the top element of the expression stack and works like an accumulator. This stack is also used for passing parameters between subroutines, and as a scratchpad area for temporary storage of data.

- Return Stack

The 12-bit wide return stack is addressed by the return stack pointer (RP). It is used for storing return addresses of subroutines, interrupt routines and for keeping loop index counts. The return stack can also be used as a temporary storage area.

The MARC4 instruction set supports the exchange of data between the top elements of the expression stack and the return stack. The two stacks within the RAM have a user definable location and maximum depth.

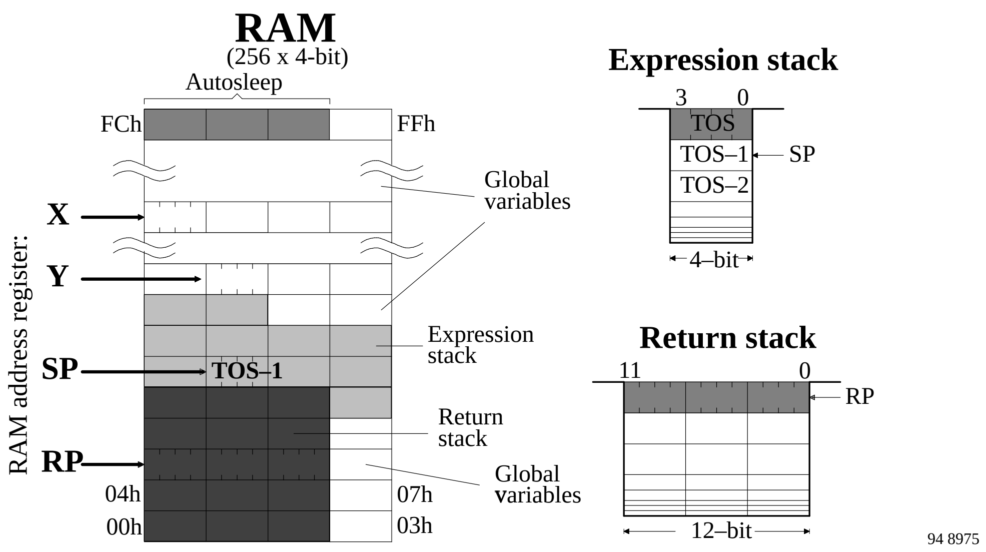


Figure 5. RAM map

1.2.3 Registers

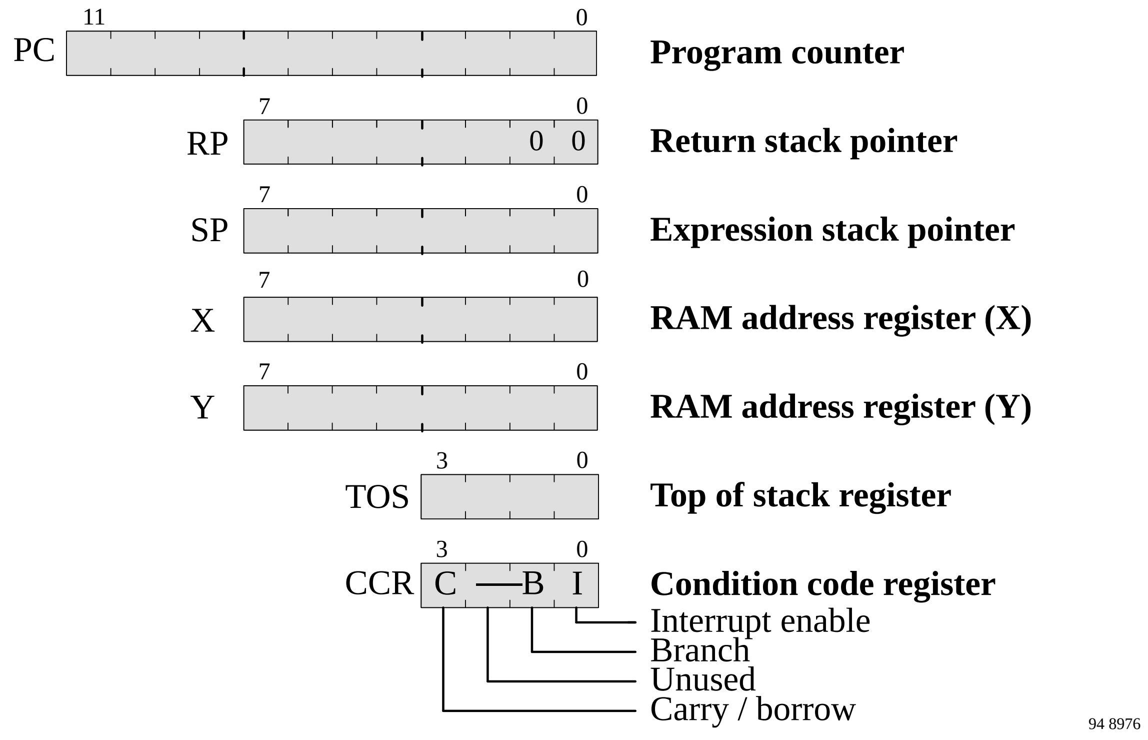


Figure 6. Programming model

The MARC4 controller has six programmable registers and one condition code register. They are shown in the following programming model.

- Program Counter (PC)

The program counter (PC) is a 12-bit register that contains the address of the next instruction to be fetched from the program memory. Instructions currently being executed are decoded in the instruction decoder to determine the internal micro operations. For linear code (no calls or branches) the program counter is incremented with every instruction cycle. If a branch-, call-, return-instruction or an interrupt is executed the program counter is loaded with a new address. The program counter is also used with the TABLE instruction to fetch 8-bit wide ROM constants.

RAM address register

The RAM is addressed with the four 8-bit wide RAM address registers: SP, RP, X and Y. These registers allow access to any of the 256 RAM nibbles.

- Expression Stack Pointer (SP)

The stack pointer (SP) contains the address of the next-to-top 4-bit item (TOS-1) of the expression stack. The pointer is automatically pre-incremented if a nibble is moved onto the stack or post-decremented if a nibble is removed from the stack. Every post-decrement operation moves the item (TOS-1) to the TOS register before the SP is decremented. After a reset the stack pointer has to be initialized with ” >SP $xx ” to allocate the start address of the expression stack area.

- Return Stack Pointer (RP)

The return stack pointer points to the top element of the 12-bit wide return stack. The pointer automatically pre-increments if an element is moved onto the stack or it post-decrements if an element is removed from the stack. The return stack pointer increments and decrements in steps of 4. This means that every time a 12-bit element is stacked, a 4-bit RAM location are left unwritten. These location are used by the qFORTH compiler to allocate 4-bit variables. After a reset the return stack pointer has to be initialized with ” >RP FCh ”.

- RAM Address Register (X and Y)

The X and Y registers are used to address any 4-bit item in the RAM. A fetch operation moves the addressed nibble onto the TOS. A store operation moves the TOS to the addressed RAM location. Using either the pre-increment or post-decrement addressing mode arrays in the RAM can be compared, filled or moved.

- Top Of Stack (TOS)

The top of stack register is the accumulator of the MARC4. All arithmetic/logic, memory reference and I/O operations use this register. The TOS register gets the data from the ALU, the program memory, the RAM or via the I/O bus.

- Condition Code Register (CCR)

The 4-bit wide condition code register contains the branch, the carry and the interrupt enable flag. These bits indicates the current state of the CPU. The CCR flags are set or reset by ALU operations. The instructions SET_BCF, TOG_BF, CCR! and DI allow a direct manipulation of the condition code register.

Carry/Borrow (C)

The carry/borrow flag indicates that borrow or carry out of arithmetic logic unit (ALU) occurred during the last arithmetic operation. During shift and rotate operations this bit is used as a fifth bit. Boolean operations have no affect on the C flag.

Branch (B)

The branch flag controls the conditional program branching. When the branch flag was set by one of the previous instructions a conditional branch is taken. This flag is affected by arithmetic, logic, shift, and rotate operations.

Interrupt Enable (I)

The interrupt enable flag enables or disables the interrupt processing on a global basis. After reset or by executing the DI instruction the interrupt enable flag is reset and all interrupts are disabled. The µC does not process further interrupt requests until the interrupt enable flag is set again by either executing an EI, RTI or SLEEP instruction.

1.2.4 ALU

The 4-bit ALU performs all the arithmetic, logical, shift and rotate operations with the top two elements of the expression stack (TOS and TOS-1) and returns its result to the TOS. The ALU operations affect the carry/borrow and branch flag in the condition code register (CCR).

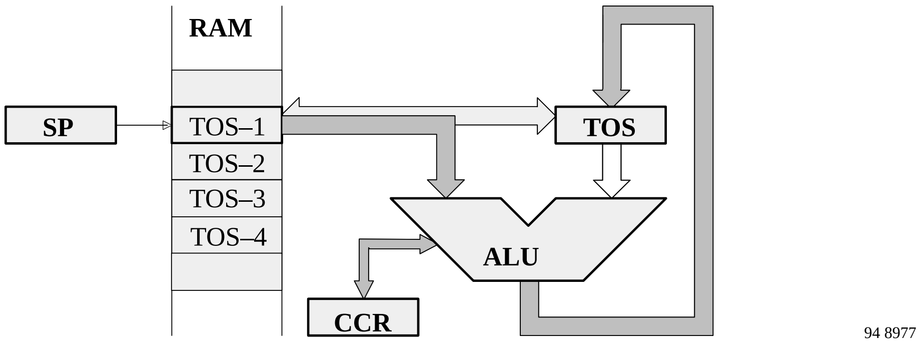


Figure 7. ALU zero address operations

1.2.5 Instruction Cycles

A MARC4 instruction word is one or two byte long and is executed within one or four machine-cycles. A machine–cycle consists of two system clocks (SYSCL). The MARC4 is a zero address machine. Most of the instructions are one byte long and are executed in only one machine–cycle. The CPU has an instruction pipeline, this allows the controller to fetch the next instruction from program memory at the same time as the present instruction is being executed. For more informations see section ”MARC4 Instruction Set Overview”.

1.2.6 I/O Bus

The I/O ports and the registers of the peripheral modules (timer 1, timer 2, EEPROM) are I/O mapped. The communication between the core and the on-chip peripherals takes place via the I/O bus and the associated I/O control bus. These buses are used for different functions: for read and write accesses, for the interrupt generation, to reset peripherals and for the SLEEP mode. With the MARC4 IN-instruction and OUT-instructions the I/O bus allows a direct read or write access to one of the 16 I/O addresses. More about the I/O access to the on-chip peripherals is described in the section ”Peripheral modules”.

The I/O buses are internal buses and are not accessible by the customer on the final microcontroller device, but they are used as the interface for the MARC4 emulation (see also the section “Emulation”).

1.2.7 Interrupt Structure

The MARC4 can handle interrupts with eight different priority levels. They can be generated from the internal and external interrupt sources or by a software interrupt from the CPU itself. Each interrupt level has a hard-wired priority and an associated vector for the service routine in the ROM (see table 2). The programmer can enable or disable interrupts all together by setting or resetting the interrupt enable flag (I) in the CCR.

Interrupt processing

For processing the eight interrupt levels the MARC4 contains an interrupt controller with the 8-bit wide interrupt pending and interrupt active register. The interrupt controller samples all interrupt requests during every non-I/O instruction cycle and latches them in the interrupt pending register. If no higher priority interrupt is present in the interrupt active register it signals the CPU to interrupt the current program execution. If the interrupt enable bit is set the processor enters an interrupt acknowledge cycle. During this cycle a SHORT CALL instruction to the service routine is executed and the current PC is saved on the return stack. An interrupt service routine is finished with the RTI instruction. This instruction sets the interrupt enable flag, resets the corresponding bits in the interrupt pending/active register and fetches the return address from the return stack to the program counter. When the interrupt enable flag is reset (interrupts are disabled), the execution of interrupts is inhibited but not the logging of the interrupt requests in the interrupt pending register. The execution of the interrupt will be delayed until the interrupt enable flag is set again. But note that interrupts are lost if an interrupt request occurs during the corresponding bit in the pending register is still set. After the reset (power-on, external or watchdog reset), the interrupt enable flag and the interrupt pending and interrupt active register are reset.

Interrupt latency

The interrupt latency is the time from the falling edge of the interrupt to the interrupt service routine being activated. In the MARC4 this takes between 3 to 5 machine cycles depending on the state of the core.

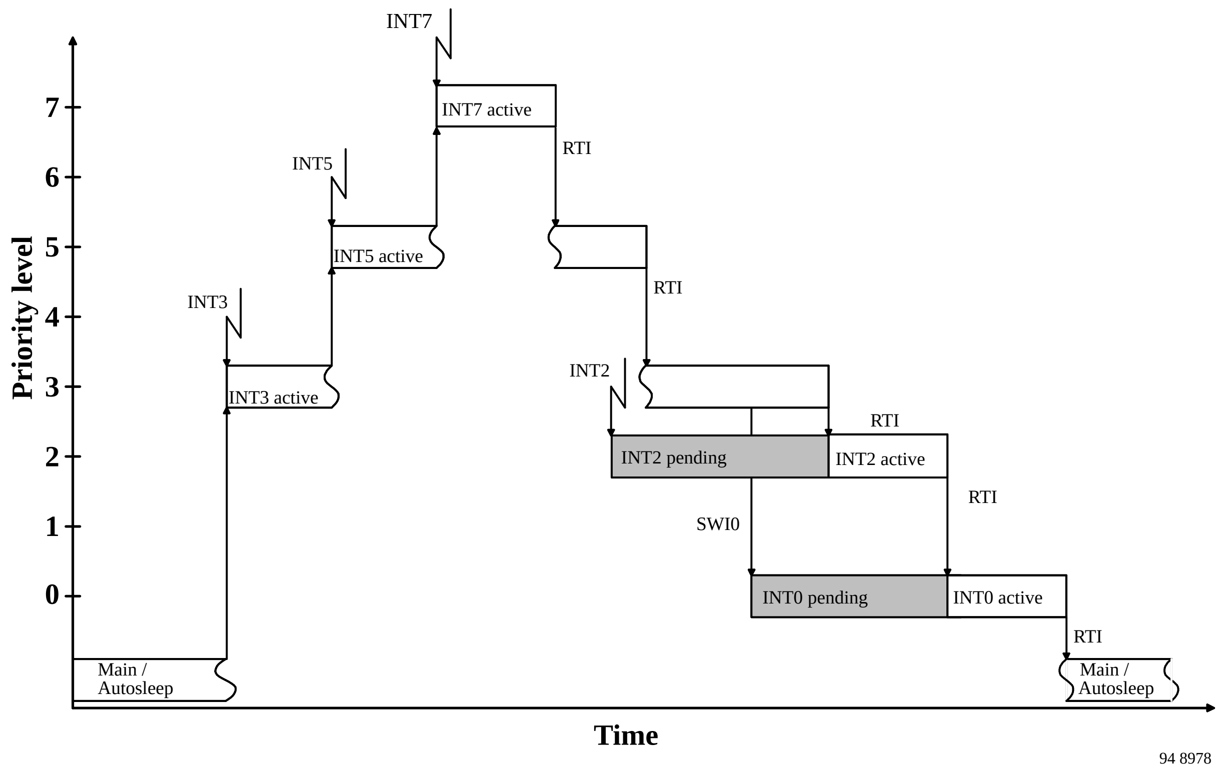


Figure 8. Interrupt handling

Table 2. Interrupt priority table

Interrupt	Priority	Vector Address	Interrupt Opcode (Acknowledge)	Function
INT0	lowest	040h	C8h (SCALL 040h)	Software interrupt (SWI0)
INT1		080h	D0h (SCALL 080h)	EEPROM write ready
INT2		0C0h	D8h (SCALL 0C0h)	External hardware interrupt, neg. edge at BP30 or BP31
INT3		100h	E0h (SCALL 100h)	External hardware interrupt, neg. edge at BP32 or BP33
INT4		140h	E8h (SCALL 140h)	Timer 1 interrupt
INT5		180h	F0h (SCALL 180h)	Timer 2 interrupt
INT6		1C0h	F8h (SCALL 1C0h)	External hardware interrupt, neg. edge at IP40 pin
INT7	highest	1E0h	FCh (SCALL 1E0h)	Software interrupt (SWI7)

Software Interrupts

The programmer can generate interrupts using the software interrupt instruction (SWI) which is supported in qFORTH by predefined macros named SWI0 to SWI7. The software triggered interrupt operates exactly like any hardware triggered interrupt. The SWI instruction takes the top two elements from the expression stack and writes the corresponding bits via the I/O bus to the interrupt pending register. Thus using the SWI instruction, interrupts can be re-prioritized or lower priority processes scheduled for later execution.

Hardware Interrupts

In the M44C260 are eleven hardware interrupt sources with six different levels. Each of these sources can be enabled or disabled separately with an interrupt mask bit in the IMR1 or IMR2 register.

Table 3. Hardware interrupts

Interrupt	Priority	Mask		Interrupt Source
		Register	Bit	
EEPROM write ready	INT1	EMS	IMEP	EEPROM end of write cycle
External interrupt port 3 (BP30 OR BP31)	INT2	IMR1	IM30 IM31	Negative edge at BP30 Negative edge at BP31
External interrupt port 3 (BP32 OR BP33)	INT3	IMR1	IM32 IM33	Negative edge at BP32 Negative edge at BP33
Timer 1 interrupt	INT4	IMR2	IMT1	Timer 1
Timer 2 interrupt	INT5	T2IC	IMAS IMAP IMBS IMBP	Timer A end of space/underflow Timer A end of pulse/capture Timer B end of space/underflow Timer B end of pulse/capture
Ext. interrupt IP40 input	INT6	IMR2	IM6	Negative edge at IP40 input

1.3 Reset

The reset puts the CPU into a well-defined condition. The reset can be triggered by switching on the supply voltage, by a break-down of the supply voltage, by the watchdog timer or by pulling the NRST pad to low.

After any reset the branch-, carry- and interrupt enable flag in the Condition Code Register (CCR) , the interrupt pending register and the interrupt active register are reset.

During the reset-cycle the I/O bus control signals are set to ’reset mode’ thereby initializing all on-chip peripherals.

A reset is finished with a short call instruction (opcode C1h) to the program memory address 008h. This activates the initialization routine $RESET. With that routine the stack pointers, variables in the RAM and the peripheral must be initialized.

Power-on Reset

The power-on reset ensures that the core is activated not before the operating supply voltage has been reached.

A reset is also generated when the supply voltage remains below the operating range for more than 5 ms.

External Reset (NRST)

An external reset can be triggered with the NRST pin. For the external reset the pin should be low for a minimum of two machine cycles.

Watchdog Timer Reset

If the watchdog timer function of Timer 1 is enabled a reset is triggered with every watchdog counter overflow. To suppress that the watchdog counter must be reset by an access to the CWD-register (see also Timer 1/watchdog counter).

The power-on reset and watchdog reset are indicated in the same way as an external reset on the NRST pad.

1.4 Clock Generation

The M44C260 has two oscillators, one RC oscillator for the system clock generation and an additional 32-kHz crystal oscillator. The system clock generator provides the core and Timer 2 with the clock. The system clock frequency of the M44C260 is programmable for 1 or 2 MHz. The crystal oscillator is used as an exact time base for Timer 1. If no exact timing is required, the controller does not need an external crystal. In this case Timer 1 is provided with the system clock.

The configuration for both oscillators is programmable with the clock status control register (CSC), which is a subport register located in port CSUB. The required configuration has to be initialized after reset in the $RESET routine. The default setting after a reset is 1 MHz system clock and an active 32-kHz crystal oscillator.

After power-on or a SLEEP instruction the clock generator needs a start-up time until it runs with an exact timing. The CRDY bit in the CSC register indicates the start-up phase.

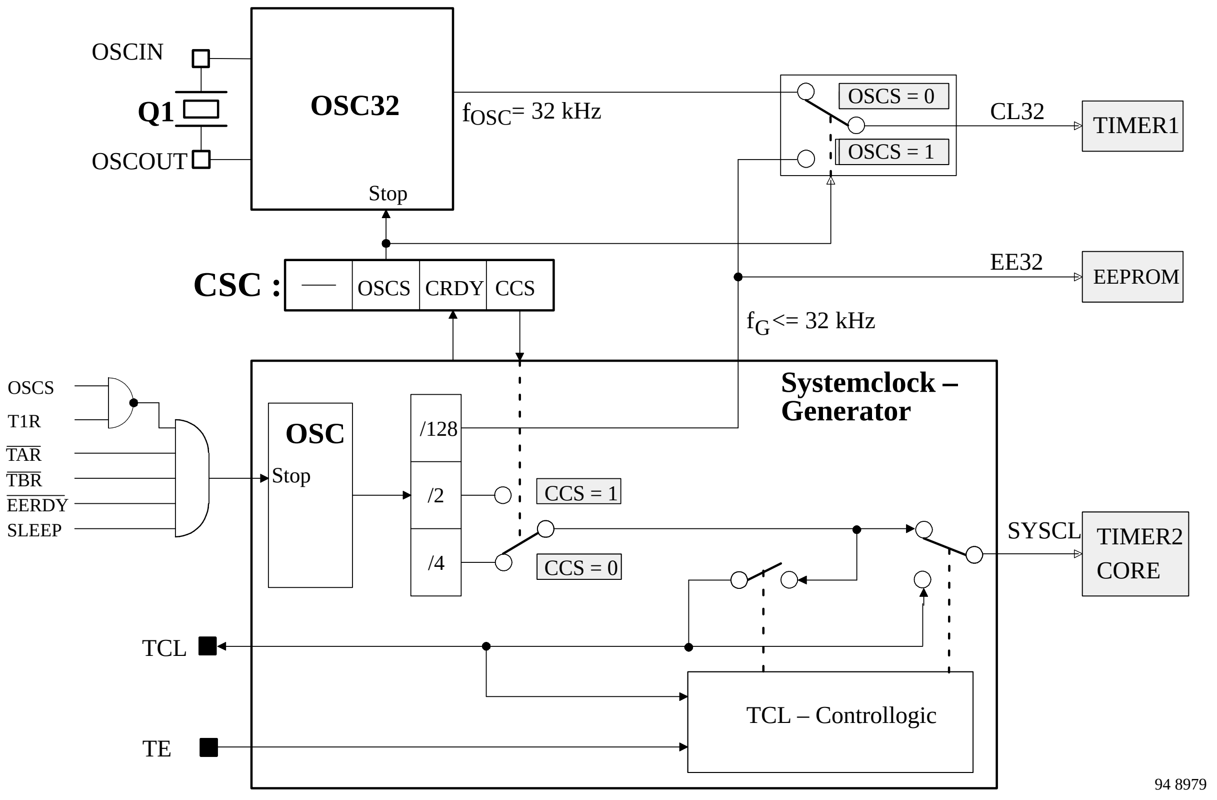


Figure 9. Clock module

1.4.1 Clock Status/Control Register (CSC)

Address: Ch Subaddress: 2h

	Bit 3	Bit 2	Bit 1	Bit 0
CSC	—	**OSCS**	**CRDY**	**CCS**

Reset value: X000h

OSCS	**Oscillator Stop** When OSCS = 0 the controller is configured to run with the 32-kHz crystal oscillator for Timer 1. When OSCS = 1 the 32-kHz oscillator stops. For µC operation without crystal, this bit must be set after reset. In that case Timer 1 is provided from the internal RC oscillator.
CRDY	**Clock Ready (status bit)** CRDY = 0 indicates the start-up time of the oscillators. CRDY = 1 indicates that the clock is ready. The µC runs with an exact timing.

CCS	**Core Clock Select**
	CCS = 0 selects 1 MHz system clock (SYSCL/TCL)
	CCS = 1 selects 2 MHz system clock (SYSCL/TCL)

1.4.2 TCL Signal

The TCL pin can be used as input to supply the controller with an external clock. For this configuration the TCL pin must be held low for at least 0.5 ms during the reset cycle. The controller is working with clock frequencies up to 2.5 MHz. It is also possible to use the TCL pin as output to supply peripherals with the system clock. In this case the TE pin must be connected to V_{DD} level and the TCL pin must have a high impedance load.

1.5 Power Down Modes

The sleep mode is a shutdown condition which is used to reduce the average system power consumption in applications where the µC is not fully utilized. In this mode the system clock is stopped. The sleep mode is entered with the SLEEP instruction. This instruction sets the interrupt enable bit (I) in the condition code register to enable all interrupts and stops the core. During the sleep mode the peripheral modules remain active and are able to generate interrupts. The µC exits the sleep mode with any interrupt or a reset.

The sleep mode can only be kept when none of the interrupt pending or active register bits are set. The application of the $AUTOSLEEP routine ensures the correct function of the sleep mode.

The total power consumption is directly proportional to the active time of the µC. For a rough estimation of the expected average system current consumption, the following formula should be used:

$$\mathbf{I_{total} (V_{DD}, f_{Osc}) = I_{Sleep} + (I_{DD} * T_{active}/T_{total})}$$

I_{DD} depends on V_{DD} and f_{Osc}.

Systemclock Generator Stop

The M44C260 has different power down modes. When the MARC4 core enters the sleep mode and no on-chip peripheral needs a clock signal (SYSCL) the system clock oscillator is stopped. Therefor the programmer should stop timer 1 and timer 2 during the sleep mode if they are not required. If the 32-kHz oscillator is not used it should be stopped. Under this condition the power consumption is extremely low (see following table).

Table 4. Power consumption at different power down modes

Mode	CPU-Core	TIMER 1 [T1R] TIMER 2 [TAR, TBR] EEPROM [EERDY]	RC Osc.	32-kHz-Osc. [OSCS]	Power-Consumption [µA]
1	SLEEP	T1R=0 AND TAR=0 AND TBR=0 AND EERDY=1	STOP	STOP	< 1.0
2	SLEEP	T1R=X, TAR=0 AND TBR=0 AND EERDY=1	STOP	RUN	< 1.0
3	SLEEP	T1R=1 OR TAR=1 OR TBR=1 OR EERDY=0	RUN	STOP	< x
4	SLEEP	T1R=X, TAR=1 OR TBR=1 OR EERDY=0	RUN	RUN	< x
5	RUN	T1R=X, TAR=X, TBR=X, EERDY=X	RUN	STOP	< y
6	RUN	T1R=X, TAR=X, TBR=X, EERDY=X	RUN	RUN	< y

2 Peripheral Modules

2.1 Addressing Peripherals

The access to the peripheral modules (ports, registers) is executed via the I/O bus. The IN- or OUT-instruction allows the direct addressing of 16 I/O ports. For the peripherals with a large number of registers, extended addressing is used. With two I/O operations an extended I/O port allows the access to 16 subports. The first OUT-instruction writes the subport address to the subaddress register, the second IN- or OUT-instruction reads data from or writes data to the addressed subport.

Table 5. I/O-addressing

I/O Operation	qFORTH Instructions	Description
Port 0, 1, 2, 3, 4, T2SC, EMS		
I/O read	port IN	Read data from port
I/O write	data port OUT	Write data to port
T2SUB, CSUB		
Extended I/O read	subaddress port OUT	Write subaddress to port
	port IN	Read data from subaddress
Extended I/O write	subaddress port OUT	Write subaddress to port
	data port OUT	Write data to subaddress
Extended I/O short read	port IN	Read data from current subaddress
ESUB		
Extended I/O read (byte)	subaddress port OUT	Write subaddress to port
	port IN	Read data high nibble from subaddress
	port IN	Read data low nibble from subaddress
Extended I/O write (byte)	subaddress port OUT	Write subaddress to port
	data port OUT	Write data low nibble to subaddress
	data port OUT	Write data high nibble to subaddress

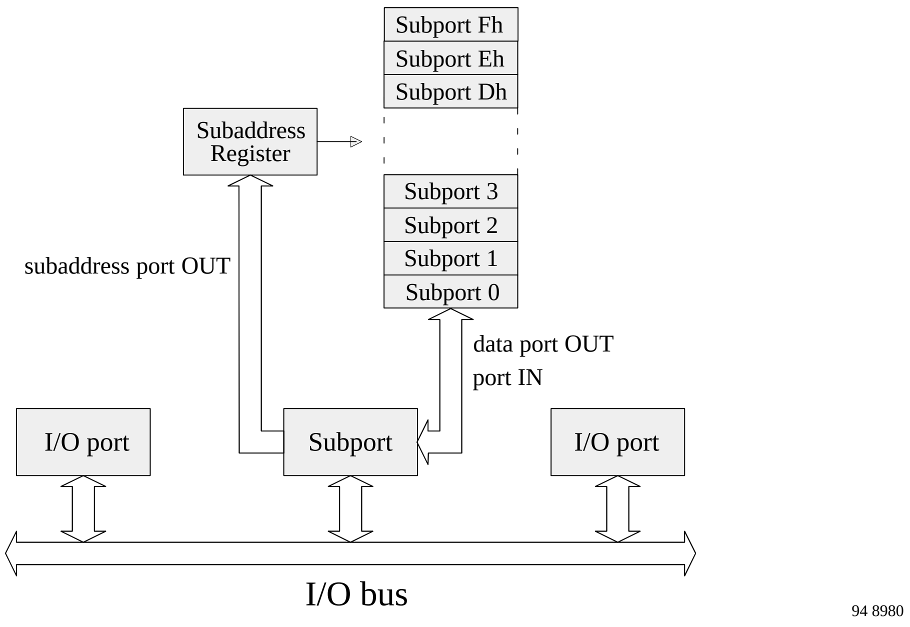


Figure 10. Extended I/O addressing

Table 6. Peripheral addresses

Addr.	Name	Function			
0	Port 0	Bidirectional port			
1	Port 1	Bidirectional port			
2	Port 2	Bidirectional port			
3	Port 3	Bidirectional port			
4	Port 4	Input port			
5	——				
6	——				
7	——				
8	T2SC	Timer 2 status and control register			
9	T2SUB	Subport for timer 2	Sub-address	Name	Register
			0	TARCH	Timer 2A space reload/capture register, high nibble
			1	TARCL	Timer 2A space reload/capture register, low nibble
			2	TARH	Timer 2A pulse reload register
			3	TARL	Timer 2A pulse reload register
			4	TBRCH	Timer 2B space reload/capture register, high nibble
			5	TBRCL	Timer 2B space reload/capture register, low nibble
			6	TBRH	Timer 2B pulse reload register
			7	TBRL	Timer 2B pulse reload register
			8	TAM1	Timer 2A mode register 1
			9	TAM2	Timer 2A mode register 2
			A	TBM1	Timer 2B mode register 1
			B	TBM2	Timer 2B mode register 2
			C	T2IC	Timer 2 interrupt control
			D	T2PC	Timer 2 prescaler control
			E	——	
			F	——	
A	EMS	EEPROM status register			
B	ESUB	Subport for EEPROM	Row 0 – Row F		
C	CSUB	Subport for watchdog, timer 1, interrupt masks, and clock generator	Sub-address	Name	Register
			0	WDC	Watchdog control register
			1	CWD	Clear watchdog counter
			2	CSC	Clock status/control register
			3	——	
			4	T1C	Timer 1 control register
			5	IMR1	Interrupt mask register 1
			6	IMR2	Interrupt mask register 2
			7-F	——	
D	——				
E	——				
F	——				

2.1.1 Input Port 4

Port 4 is the input port for the pins IP40, IP43, TA and TB. IP40 is also the interrupt input for INT6, and TA and TB are normally used for timer I/O functions.

	Bit 3	Bit 2	Bit 1	Bit 0
Input port 4	**IP43**	**TB/IP42**	**TA/IP41**	**IP40/INT6**

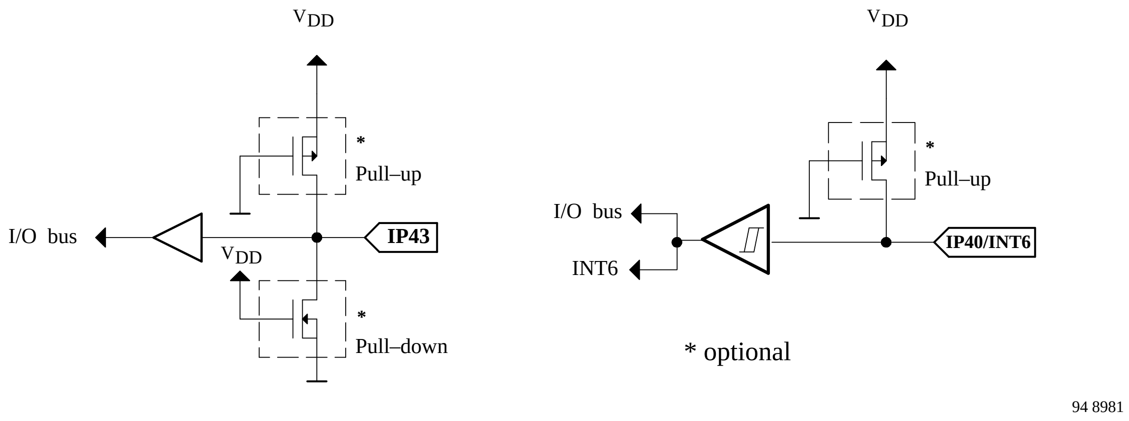


Figure 11. Input port IP40, IP43

2.1.2 Bidirectional Ports

Ports 0, 1, 2 and 3 are bidirectional 4-bit wide ports and may be used for data input or output. The data direction is programmable for a complete port only. The port is switched to output with an OUT-instruction and to input with an IN-instruction. The data written to a port will be stored into the output latches and appears immediately after the OUT-instruction at the port pin. After RESET all output latches are set to Fh and the ports are switched to input mode.

Note: Care must be taken when switching bidirectional ports from output to input. The capacitive load at this port may cause the data read to be the same as the last data written to this port. To avoid this, when switching the direction one of the following approaches should be used.

- Use two IN-instructions and DROP the first data nibble read. The first IN switches the port from output to input, DROP removes the first invalid nibble and the second IN reads the valid nibble.
- Use an OUT-instruction followed by an IN-instruction. With the OUT-instruction the capacitive load is charged or discharged depending on the optional pull-up /pull-down configuration. Write a “1” for pins with pull-up resistors and a “0” for pins with pull-down resistors.

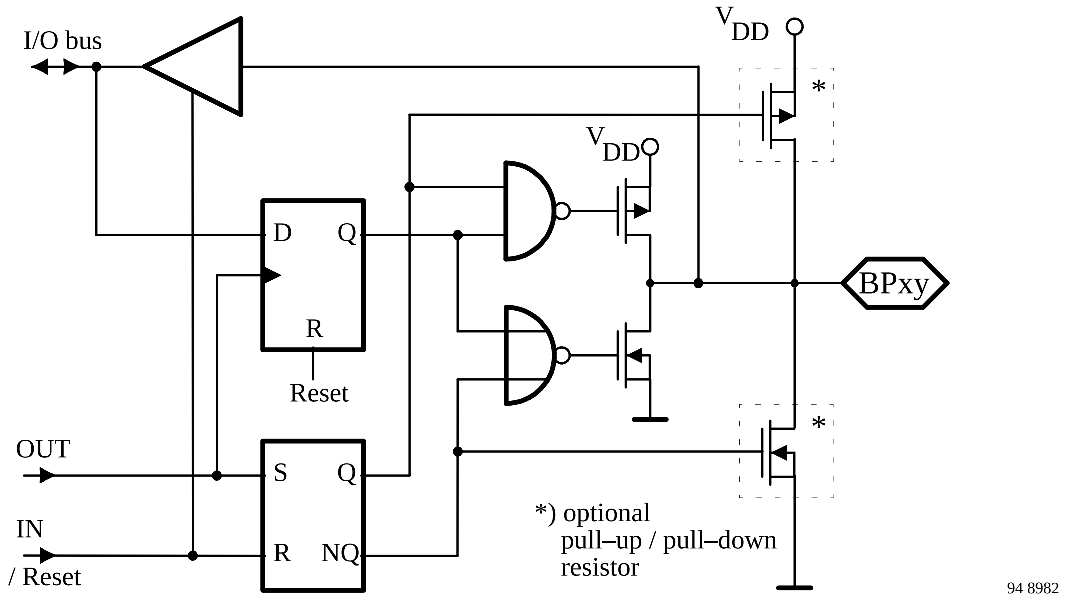


Figure 12. Bidirectional port

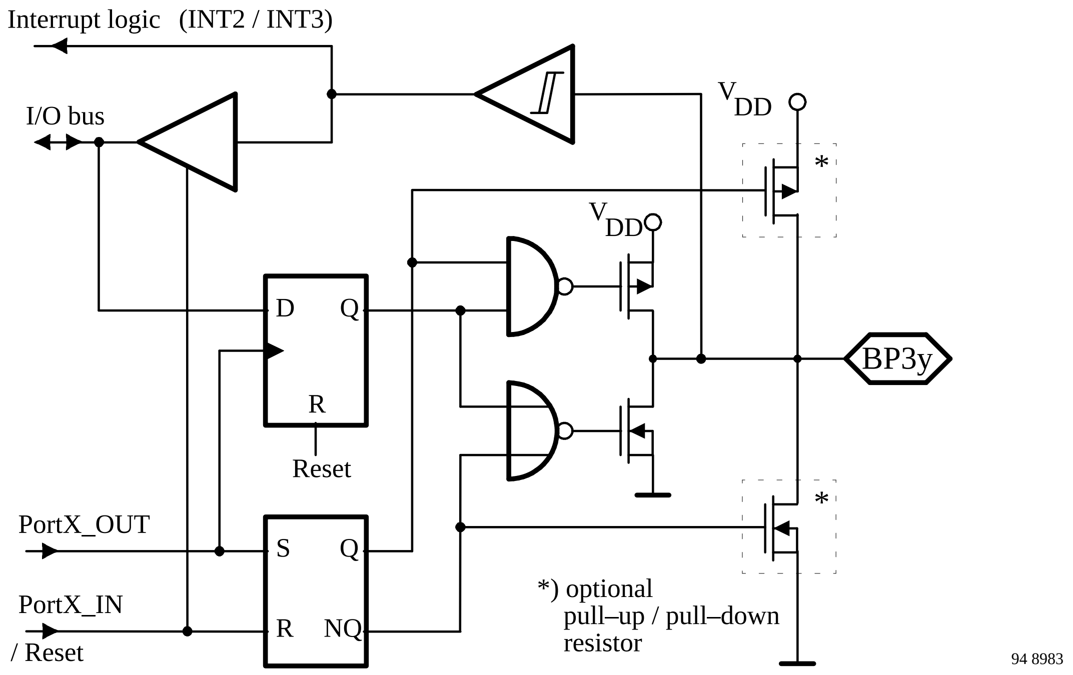


Figure 13. Bidirectional port 3 with interrupt input

2.1.3 External Interrupt Inputs

The pins IP40 and BP30 – BP33 can be used as external interrupt inputs. IP40 is used for INT6, BP32 and BP33 are used for INT3, and BP30 and BP31 are used for INT2. Pin IP40 is also used as an input port and BP30 – BP33 as a bidirectional port (see figure 11). Each of these external interrupt sources can be enabled or disabled with individually interrupt mask bits. A negative transition at one of these inputs requests an interrupt, when the corresponding mask bit is set. The interrupt masks are placed in the subport registers IMR1 and IMR2 of port CSUB.

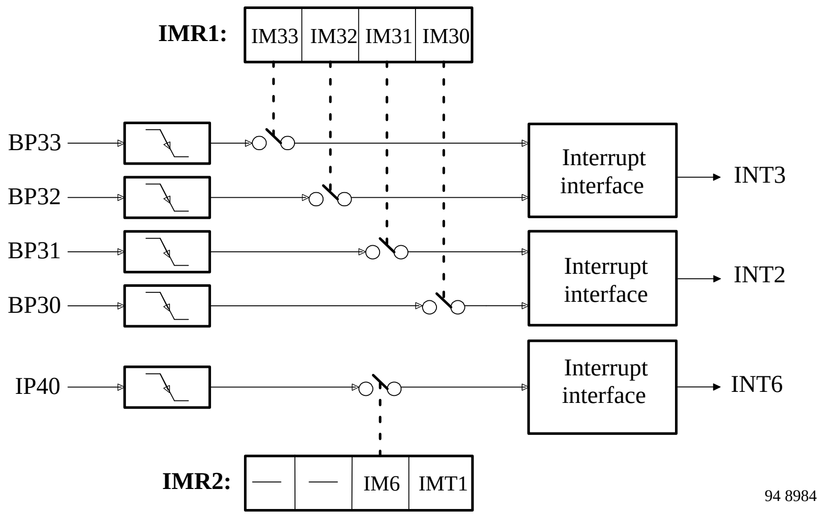


Figure 14. External interrupt inputs

2.2 Timer 1

Timer 1 is an interval timer for generating interrupts. Additional to the timer can be used as watchdog timer. The timer consists of a programmable 18 stage divider which is supplied with a 32-kHz clock and a 3-bit counter for the watchdog function (see figure 15). The time interval for a timer 1 interrupt (INT4) can be programmed with the timer control register from 1 ms up to 8.0 s. The timer 1 interrupt is maskable with the IMT1 bit.

The time interval for a watchdog reset can be programmed with the watchdog control register for 0.5, 2.0, 8.0 or 16.0 s. When the watchdog is active (WDR = 1, T1R = 1) the controller is reset with the overflow of the 3-bit watchdog counter. The application software has to ensure that the watchdog counter is reset by a write access to the CWD port before it overflows. Because the watchdog timer is supplied by the interval timer it is necessary that timer 1 is set active (T1R = 1).

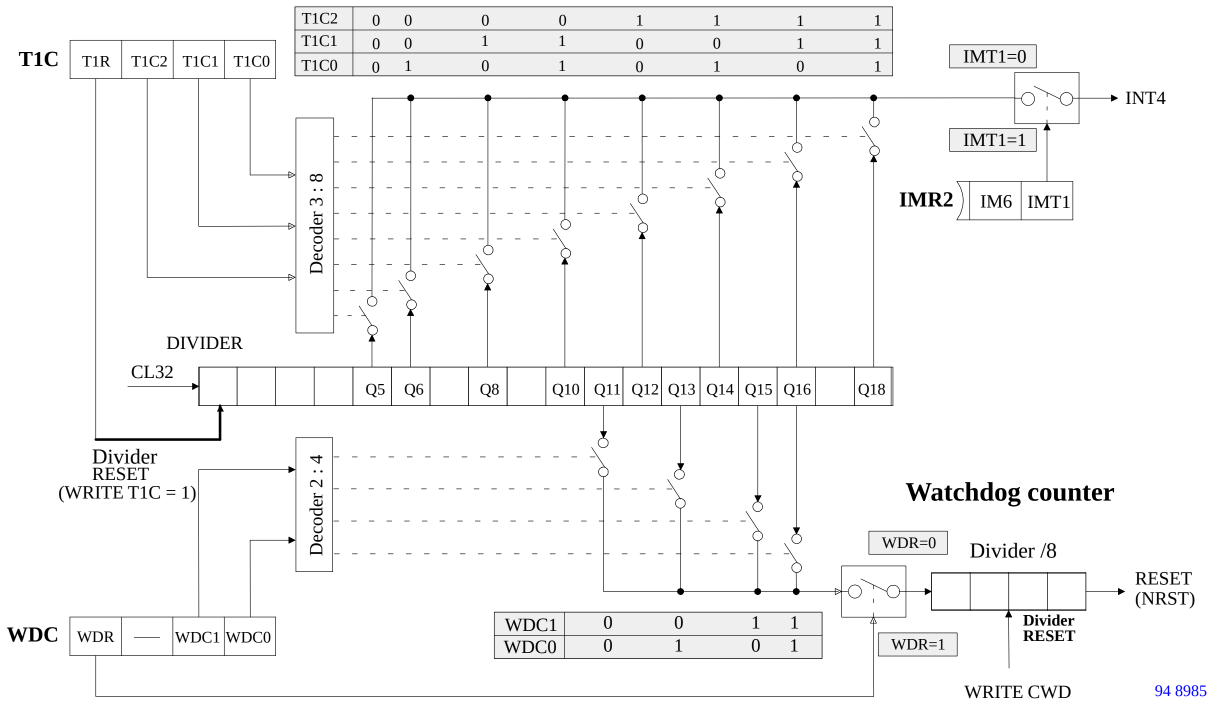


Figure 15. Timer 1

Timer 1 register

The registers of Timer 1 are I/O-mapped. They are subport register of port CSUB the access is made by extended I/O operations. The interval timer is controlled by the prescaler control register T1C. The interrupt mask IMT1 is placed in the interrupt mask register IMR2. The watchdog timer is controlled by the watchdog control register WDC and port CWD. A write access to CWD resets the watchdog counter.

2.2.1 T1C – Timer 1 Control Register

Address: ‘C’h Subaddress 4

	Bit 3	2	1	0
TIC	**T1R**	**T1C2**	**T1C1**	**T1C0**

Reset value: 0000b

T1R	Timer 1 reset T1R = 1 resets the interval timer
T1C2	Timer 1 control bit 2
T1C1	Timer 1 control bit 1
T1C0	Timer 1 control bit 0

This both bits select the time interval for a Timer 1 interrupt.

T1C2	T1C1	T1C0	Divider	Time Interval
0	0	0	32	0.9765625 ms
0	0	1	64	1.953125 ms
0	1	0	256	7.8125 ms
0	1	1	1024	31.25 ms
1	0	0	4096	125 ms
1	0	1	16384	500 ms
1	1	0	65536	2 s
1	1	1	262144	8 s

2.2.2 WDC – Watchdog Control Register

Address: ‘C’h Subaddress 0

	Bit 3	2	1	0
WDC	**WDR**	**—**	**WDM1**	**WDM0**

Reset value: 0x00b

WDR	Watchdog run WDR = 0 the watchdog counter is inactive and reset WDR = 1 the watchdog counter is active and able to generate a reset when Timer 1 is running
WDC1	Watchdog mode 1
WDC0	Watchdog mode 0

This both bits control the time interval for the watchdog reset.

WDM 1	WDM 0	Divider	Delay time to Reset (s)
0	0	2048	0.5
0	1	8192	2
1	0	32768	8
1	1	524288	16

2.3 Timer 2

Timer 2 consists of the two timer/counter blocks Timer A and Timer B. Each block has one 8-bit downcounter and a programmable prescaler. The clock inputs can be programmed to count the system clocks, Timer A clocks or external clocks. The maximum clock rate for external clocks is the half system clock frequency (SYSCL/2). Each counter has a reload register for the pulse time and a reload register for the space time. Every counter underflow toggles the output and reloads the downcounter alternately from the pulse reload register or from the space reload register. This allows the generation of any duty cycles.

Addition both counters have a capture mode. In this mode an external signal or the Counter B output causes the current counter value to be captured into the corresponding capture register.

The timer has two I/O pins, TA for Timer A and TB for Timer B. Used as output the pins have a high level during the pulse time and a low level during the space time of the timer. As input the pins are used for the external counter clock or the capture signal. The inputs have a programmable edge detection to select the active edge of an external clock or capture signal.

Interrupts can be generated when a counter underflow or a capture event occurs. The interrupt function for timer 2 can be programmed with the interrupt control register. Both counter blocks share one interrupt vector (INT5).

Timer 2 Modes

There are various timer/counter modes for both blocks of Timer 2. They can be used separately or combined. The timer modes can be programmed with the timer control and mode registers.

Single Timer Modes

- **8-bit timer**
  Counter A/B is supplied by the system clock and is used to generate timer interrupts.
- **Pulse width modulation**
  Counter A/B is supplied by the system clock. The TA/TB pin is used as counter output. The duty cycle can be programmed with the pulse and space reload register.
- **Capture mode**
  Counter A/B is supplied by the system clock. The TA/TB pin is used as input. An external signal at the input causes the current counter value to be captured into the capture register.
- **Event counter**
  Counter A/B counts external clocks at the TA/TB pin. The capture register contains the current counter value and can be read.

Combined Timer Modes

- **16-bit timer**
  Counter A is supplied system clock and its output is coupled with the input of Counter B. In this mode the counter is used to generate timer interrupts.
- **16 bit capture mode**
  Counter A is supplied with the system clock and Counter B with the output of Counter A. An external signal at the TA pin causes the current counter value will be captured into the capture registers.
- **16-bit event counter**
  The output of Counter A is coupled with the input of Counter B to count external clocks at TA. The capture register of both counters contain the current counter values.
- **Burst generator**
  Counter A is supplied the system clock and its output is coupled with the input of Counter B. The output of Counter B controls the output signal of Counter A at the TA pin. The TA output is enabled during the pulse and disabled during the space of Counter B.
- **Event counter with time gate**
  Counter A counts the clocks at the TA pin and Counter B is supplied with the system clock. Each underflow of Counter B causes the counter value of Counter A to be captured into its capture register.

Timer 2 Register

All timer register are I/O mapped. The access to the Timer 2 status control register (T2SC) can be done with a direct I/O operation to T2SC. The status is read with an IN operation and a command to control the timer is written with an OUT operation. The remaining registers of Timer 2 are subport registers of port T2SUB. The access to those registers needs an extended I/O operation. The timer function can be configured with the mode registers TAM1, TAM2, TBM1, TBM2 and the interrupt control register T2IC. The timing depends on the contents of the prescaler control register T2PC and the reload registers. The capture registers are used to read the counter value.

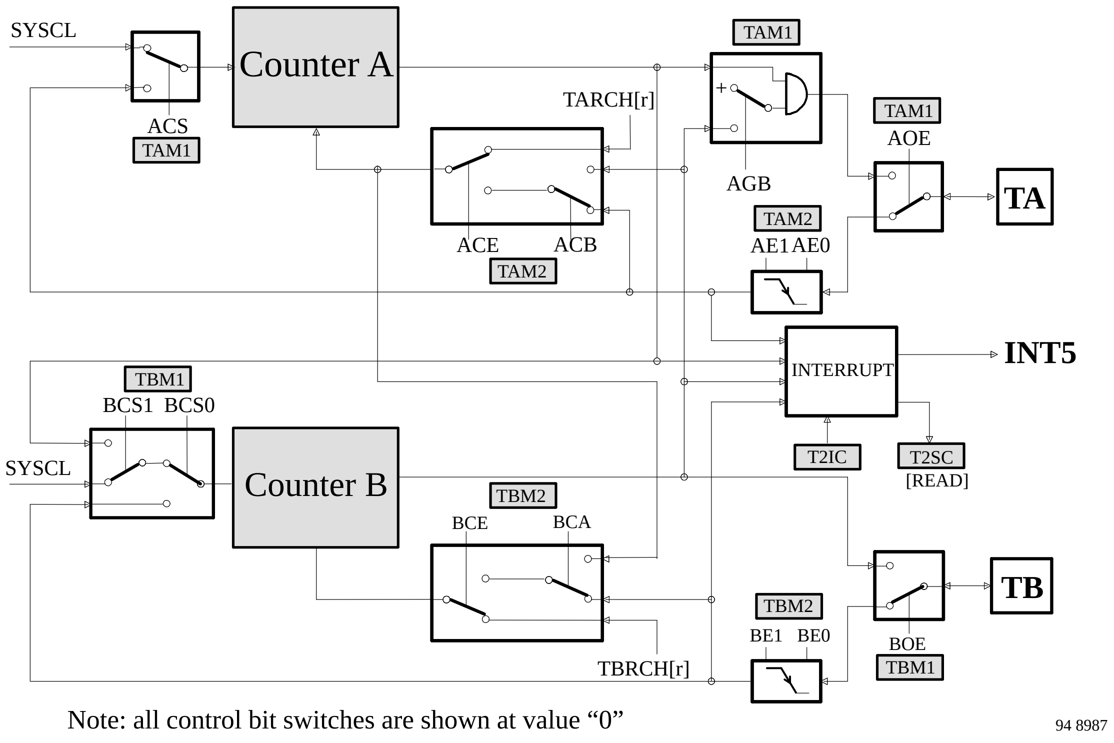


Figure 16. Timer 2

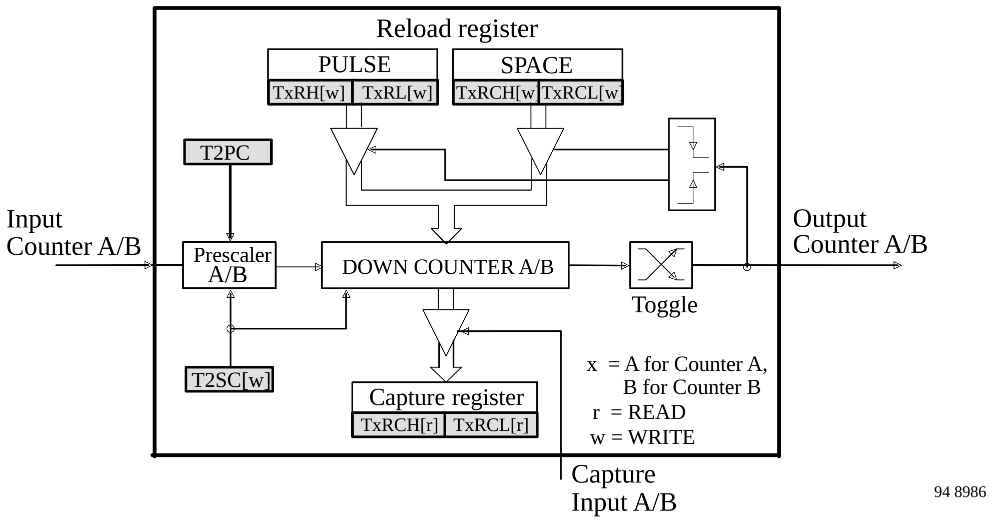


Figure 17. Counter A/B

2.3.1 Timer 2 Status/Control Register (T2SC)

Address: 8

	Bit 3	2	1	0	
Control register **Write (T2SC)**	**TBM**	**TAM**	**TBR**	**TAR**	**Reset value: 0000b**
Status register **Read (T2SC)**	**TBSU**	**TBPC**	**TASU**	**TAPC**	**Reset value: 0000b**

TAM, TAR: Timer A control bits to start or stop Timer A.

TBM, TBR: Timer B control bits to start or stop Timer B.

TBM	TAM	TBR	TAR	Timer 2 Commands
1	0	x	0	STOP_A
1	0	x	1	RUN_A
0	1	0	x	STOP_B
0	1	1	x	RUN_B
0	0	0	0	STOP_AB
0	0	1	1	RUN_AB
0	0	0	1	RUN_A-STOP_B
0	0	1	0	STOP_A-RUN_B
1	1	x	x	NOP

A STOP command resets the prescaler and counter.

A RUN command starts the counter with the next clock taking the value from the pulse reload register.

TBSU: Timer B end of space/underflow status bit.
When BCE* = 0 this bit will be set at the end of space time of Counter B.
When BCE = 1 this bit will be set with every Counter B underflow.

TBPC: Timer B end of pulse/capture status bit.
When BCE = 0 this bit will be set at the end of pulse time of Counter B.
When BCE = 1 this bit will be set when a capture event for Counter B occurs.

TASU: Timer A end of space/underflow status bit.
When ACE* = 0 this bit will be set at the end of space time of Counter A.
When ACE = 1 this bit will be set with each Counter A underflow.

TAPC: Timer A end of pulse/capture status bit.
When ACE = 0 this bit will be set at the end of pulse time of Counter A.
When ACE = 1 this bit will be set when a capture event for Counter A occurs.

*) ACE and BCE are the capture enable control bits in the timer mode registers TAM2 and TBM2.

The status bits TASU, TAPC, TBSU, TBPC will be reset after a READ access to T2SC!

2.3.2 Timer 2 Subport (T2SUB)

Address: 9

Table 7. Timer 2 subports

Subaddr.	Name	Meaning		Bit 3	Bit 2	Bit 1	Bit 0
0	TARCH [w]*	Timer A reload high		High-nibble			
	TARCH [r]*	Timer A capture high					
1	TARCL [w]*	Timer A reload low		Low-nibble			
	TARCL [r]*	Timer A capture low					
2	TARH	Timer A reload high		High-nibble			
3	TARL	Timer A reload low		Low-nibble			
4	TBRCH [w]*	Timer B reload high		High-nibble			
	TBRCH [r]*	Timer B capture high					
5	TBRCL [w]*	Timer B reload low		Low-nibble			
	TBRCL [r]*	Timer B capture low					
6	TBRH	Timer B reload high		High-nibble			
7	TBRL	Timer B reload low		Low-nibble			
8	TAM1	Timer A mode register 1		—	AGB	ACS	AOE
9	TAM2	Timer A mode register 2		ACB	ACE	AE1	AE0
A	TBM1	Timer B mode register 1		—	BCS1	BCS0	BOE
B	TBM2	Timer B mode register 2		BCA	BCE	BE1	BE0
C	T2IC	Timer 2 interrupt control		IMBS	IMBP	IMAS	IMAP
D	T2PC	Timer 2 prescaler control		BPC1	BPC0	APC1	APC0
E	—			—	—	—	—
F	—			—	—	—	—

* [w] write only, [r] read only

2.3.3 Timer 2 Reload Register

The 8–bit wide reload registers of Timer A and B are used to program the pulse and space width of the counter output signal.

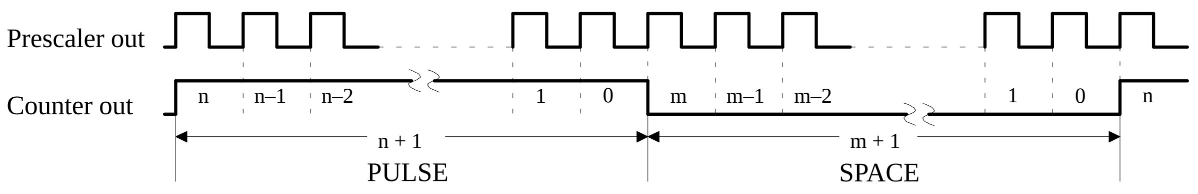
The first clock after a start command loads the downcounter with the value (n) from the pulse reload register and sets the counter output to 1. The downcounter decrements with each following clock and each underflow reloads alternately the value (m) from the space reload register or the value (n) from the pulse reload register and toggles the counter output.

The pulse and space width can be calculated as following:

Pulse time: Pulse = (n+1) x prescaler clocks

Spacetime: Space = (m+1) x prescaler clocks

$$0 \leq m, n \leq 255$$



Timer 2 Space Reload Register

The space reload register of Timer 2 is programmed by two write accesses to the subport addresses TARCH and TARCL or TBRCH and TBRCL of the Timer 2 subport T2SUB. The value (m) in the space reload register determines the space width. At the end of pulse the downcounter reloads the 8–bit value from the space reload register with the next clock of the prescaler output.

Space width: Space = (m+1) prescaler clocks
$0 \le m \le 255$

Timer 2 Pulse Reload Register

The pulse reload register of Timer 2 is programmed by two write accesses to the subport addresses TERH and TARL or TBRH and TBRL of the Timer 2 subport T2SUB. The value (n) in the pulse reload register determines the space width. At the end of space the downcounter reloads the 8–bit value from the pulse reload register with the next clock of the prescaler output.

Pulse width: Pulse = (n+1) prescaler clocks
$0 \le n \le 255$

2.3.4 Timer 2 Capture Register

The capture register is used to capture the current downcounter value when a capture event occurs. The value is kept in the capture register until the next capture event and can be read independent of the state of the downcounter. The capture events are programmable with the timer mode registers TAM2 and TBM2.

The capture registers are also used to read the counter value when the external capture mode is disabled. In this case the 8–bit counter value is transferred into the capture register by reading the high nibble TARCH or TBRCH. If the 16–bit event counter mode is enabled the complete 16–bit value is captured by reading first the high nibble TARCH of Timer A. This mechanism ensures the coherence of the counter high and low nibble during the read access.

2.3.5 Timer A Mode Register 1 (TAM1)

Address: 9 – Subaddress: 8

	Bit 3	2	1	0
TAM1	—	**AGB**	**ACS**	**AOE**

Reset value: 0000b

AGB	Counter **A** output gated by Counter **B** output AGB = 1 enables the burst generation mode. The output of Timer A is enabled during the pulse time of the Counter B and disabled (TA= 0) during the space time of the Counter B.
ACS	Counter **A** clock select This bit selects the source of the Counter A clock. When ACS = 0 the timer is supplied with internal SYSCL. When ACS = 1 the timer is supplied with an external clock on TA pin.
AOE	Timer **A** output enable AOE = 0 disables the counter output TA. AOE = 1 enables the counter output TA.

2.3.6 Timer A Mode Register 2 (TAM2)

Address: 9 – Subaddress: 9

	Bit 3	2	1	0
TAM2	**ACB**	**ACE**	**AE1**	**AE0**

Reset Value: 0000b

ACB Timer **A** captured by Timer **B**
Selects the capture source for Timer A. When ACB = 0 the signal at the TA pin is used to generate a capture event. When ACB = 1 each transition at the Counter B output is used to generate a capture event for Timer A.

ACE Timer **A** capture enable
ACE = 1 enables the capture mode for Counter A. The occurrence of a capture event causes that the current downcounter value is loaded into the capture register.

AE1 Timer **A** edge select bit **1**

AE0 Timer **A** edge select bit **0**
Whit these bits the active edge for the counter clocks and capture signal is selected.

AE1	AE0	Active Edge for Counter Clock/Capture Events
0	0	positive edge at TA pin
0	1	negative edge at TA pin
1	0	first positive edge after timer start and then each transition at TA pin
1	1	first negative edge after timer start and then each transition at TA pin

2.3.7 Timer B Mode Register 1 (TBM1)

Address: 9 – Subaddress: Ah

	Bit 3	2	1	0
TBM1	**—**	**BCS1**	**BCS0**	**BOE**

Reset value: 0000b

BCS1 Timer **B** clock select bit **1**

BCS0 Timer **B** clock select bit **0**

These bits select the source of Counter B clock.

BCS1	BCS0	Counter B Input Signal
0	0	System clock (SYSCL)
1	0	Output signal of Counter A
x	1	External input signal at TB

BOE Timer **B** output enable
BOE = 0 disables the counter output TB.
BOE = 1 enables the counter output TB.

2.3.8 Timer B Mode Register 2 (TBM2)

Address: 9 – Subaddress: Bh

	Bit 3	2	1	0
TBM2	**BCA**	**BCE**	**BE1**	**BE0**

Reset value: 0000b

BCA Timer B is captured with Timer A capture signal. With BCA = 1 the external capture signal for Timer A is used to capture Timer B
simultaneously with Timer A.
Note: It is possible to capture Counter B by a read access to TARCH

BCE Timer **B c**apture enable
BCE = 1 enables the capture mode for Counter B. A capture event loads the current downcounter value into the capture register.

BE1 Timer **B** edge select bit **1**

BE0 Timer **A** edge select bit **0**
With these bits the active edge for the counter clocks and capture signal is selected.

BE1	BE0	Active Edge for Clock/Capture Events
0	0	positive edge on TB pin
0	1	negative edge on TB pin
1	0	first positive edge after start timer and then each transition on TB pin
1	1	first negative edge after start timer and then each transition on TB pin

2.3.9 Timer 2 Prescaler Control Register (T2PC)

Address: 9 – Subaddress: Dh

	Bit 3	2	1	0
T2PC	**BPC1**	**BPC0**	**APC1**	**APC0**

Reset Value: 0000b

BPC1	Timer **B** prescaler control bit **1**
BPC0	Timer **B** prescaler control bit **0** These bits determine the divider for the prescaler of Timer B.
APC1	Timer **A** prescaler control bit **1**
APC0	Timer **A** prescaler control bit **0** These bits determine the divider for the prescaler of Timer A.

BPC1/APC1	BPC0/APC0	Divider
0	0	1
0	1	4
1	0	16
1	1	64

2.3.10 Timer 2 Interrupt Control Register (T2IC)

Address: 9 – Subaddress: Ch

	Bit 3	2	1	0
T2IC	**IMBS**	**IMBP**	**IMAS**	**IMAP**

Reset value: 0000b

IMBS	Interrupt mask Timer B end of space/underflow IMBS = 1 enables an INT5 interrupt, if BCE* = 0 at the end of space of Counter B, or if BCE = 1 at each Counter B underflow.
IMBP	Interrupt mask Timer B end of pulse/capture IMBP = 1 enables an INT5 interrupt, if BCE = 0 at the end of pulse of Counter B, or if BCE = 1 with a capture event for Counter B.
IMAS	Interrupt mask Timer A end of space/underflow IMAS = 1 enables an INT5 interrupt, if ACE* = 0 at the end of space of Counter A, or if ACE = 1 at each Counter A underflow.
IMAP	Interrupt mask Timer A end of pulse/capture IMAP = 1 enables an INT5 interrupt, if ACE = 0 at the end of pulse of Counter A, or if ACE = 1 with a capture event for Counter A.

Each interrupt source can be enabled or disabled individually by setting the corresponding maskbit.

*) ACE and BCE are the capture enable control bits in the timer mode registers TAM2 and TBM2.

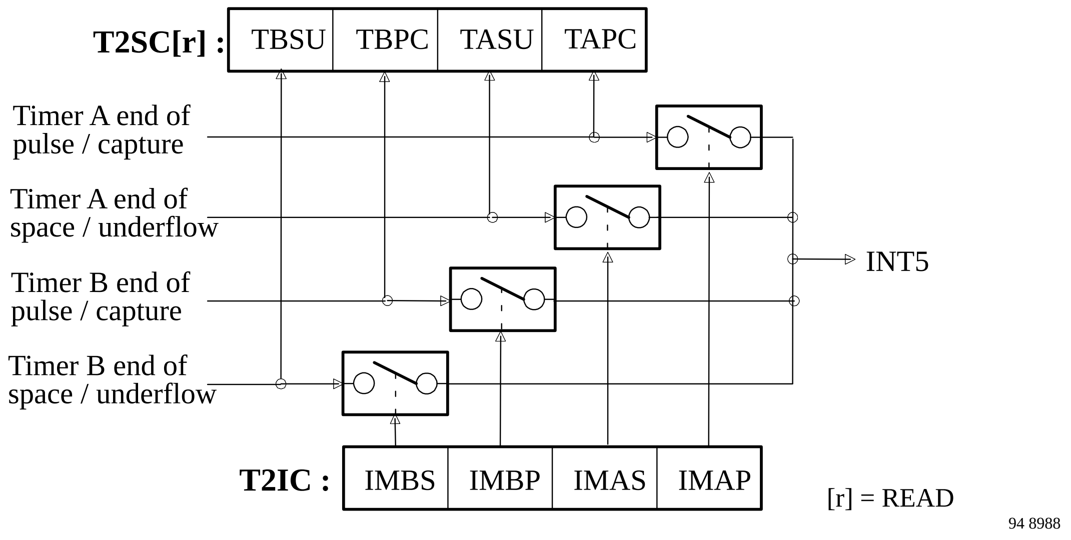

T2SC[r] :
TBSU
TBPC
TASU
TAPC
Timer A end of pulse / capture
Timer A end of space / underflow
Timer B end of pulse / capture
Timer B end of space / underflow
INT5
T2IC :
IMBS
IMBP
IMAS
IMAP
[r] = READ
94 8988

Figure 18. Timer 2 interrupt mask register

2.3.11 Timer I/O (TA/TB)

The timer I/O pins TA and TB are used as input for the external clock or capture signal and as output for the counter. The mode is controlled with AOE and BOE control bit. When AOE/BOE = 0 the pin is switched to input mode, when AOE/BOE = 1 the pin is switched to output mode. The pins also can be read with an IN-instruction via port 4 (TA with IP41 and TB with IP42).

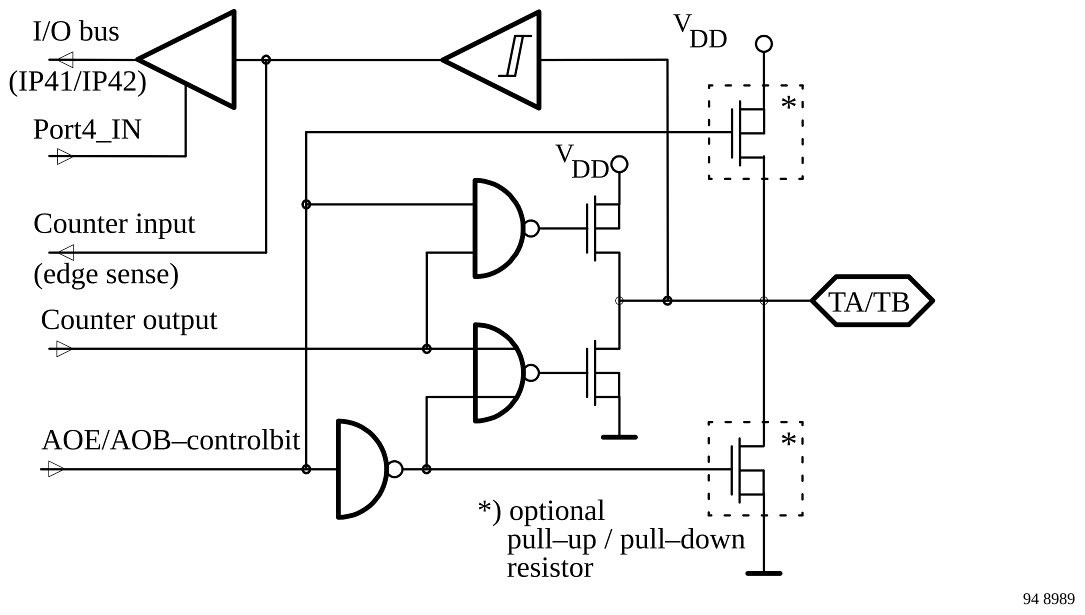


Figure 19. Timer I/O (TA/TB)

2.4 EEPROM

The EEPROM of the M44C260 is 128 bit wide and organized as an array of 16*8-bit. The EEPROM rows are I/O mapped and are subports of port ESUB. The access to any 8-bit row of the EEPROM is done by an extended 8-bit I/O operation or by special postincrement access. The EEPROM rows 12 to 15 can be write protected by hardware and software.

EEPROM SubPort (ESUB)
Address: Bh – Subaddress: 0-Fh

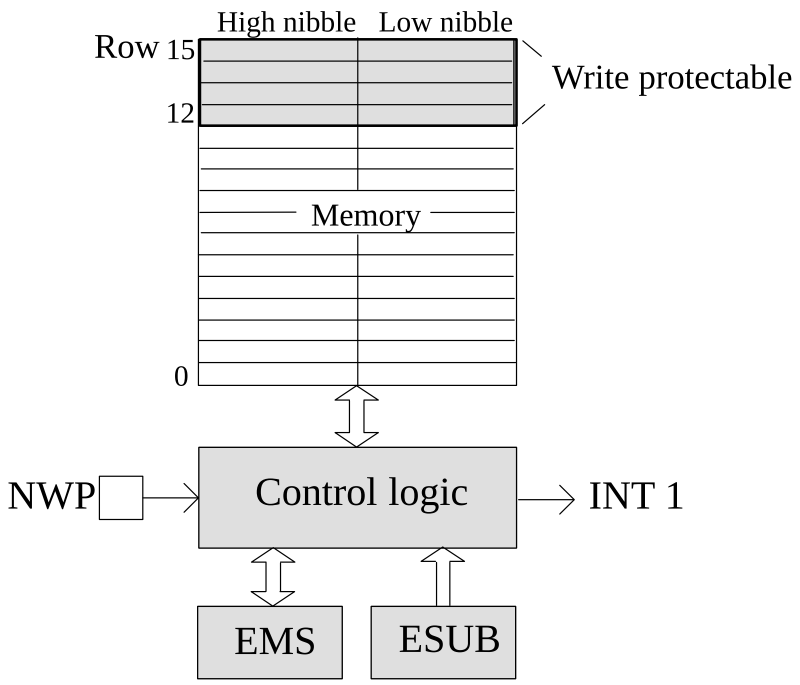


Figure 20. EEPROM

Read operation

A read operation needs an OUT- and two IN-instructions to port ESUB. First the the OUT operation writes the row address. The following two IN-instructions read the high nibble and then the low nibble of the addressed row.

qFORTH example:

```
Row address        ESUB OUT         (                    —)
                   EPSUB IN         (                    —Data_High)
                   EPSUB IN         (Data_High           — Data_High Data_Low)
```

Write operation

A write operation needs three OUT-instructions to port ESUB. The first operation writes the row address. The following two OUT-instructions write the low nibble and then the high nibble to the addressed row. After reset, rows 12 to 15 are write protected. To enable write operations to these rows the write enable bit (EWE) must be set. In all cases write accesses to these rows are disabled when pin NWP is low.

qFORTH example:

```
Row address        ESUB OUT         (Data_High Data_Low  — Data_High Data_Low)
                   EPSUB OUT        (Data_High Data_Low  —Data_High)
                   EPSUB OUT        (Data_High           —)
```

The internal EEPROM write cycle needs about 16 ms. During this cycle the EEPROM ready bit is reset (EPR = 0). After the data high nibble is written to the port ESUB the internal write cycle is started. During the internal write cycle (while EPR = 0), only read and write accesses to the EMS register are possible. All other EEPROM accesses have no effect.

Postincrement operations

The postincrement mode supports a fast access to consecutive EEPROM rows. A postincrement access is started by setting the EPI bit in the EEPROM mode register (EMS) followed by writing the row start address to port ESUB. After that the read or write operations to the consecutive EEPROM area, beginning at the start address, need only two IN- or OUT-instructions to read or write the data. The row address is incremented automatically after each complete row access (2 nibbles). A write access to the EEPROM mode register (EMS) terminates the postincrement mode.

Note: In the postincrement mode, it is not possible to change from read to write operations or vice versa before the current postincrement operation is finished.

Write ready interrupt (INT1)

At the end of the internal write cycle an interrupt is generated when the interrupt mask bit IMEP in the EEPROM mode register EMS is set. With this interrupt, successive write operations can be executed interrupt controlled within the INT1 interrupt service routine.

2.4.1 EEPROM Mode/Status Register (EMS)

Address: Ah

	Bit 3	2	1	0	
Mode register					
Write (EMS)	—	**EWE**	**EPI**	**IMEP**	Reset value: 0000b
Status register					
Read (EMS)	—	—	—	**EPR**	Reset value: xxx1b

EWE	EEPROM write enable bit EWE = 0 disables write accesses to rows 12-15 EWE = 1 enables write accesses to rows 12-15 when the NWP pin is high
EPI	EEPROM postincrement mode enable EPI = 1 activates a postincrement access after the next row address is written to port ESUB
IMEP	Interrupt mask for EEPROM write ready interrupt When IMEP is set an INT1 is generated with the end of the internal EEPROM write cycle
EPR	EEPROM ready status flag EPR = 0 indicates that the EEPROM is not ready for read or write operations (an internal write cycle is executed) EPR = 1 indicates that the EEPROM is ready for read and write operations

After a write access to the EMS-Register postincrement operations are terminated and any incomplete EEPROM read and write sequence must be started again!

3 Appendix

3.1 Emulation

For emulation all MARC4 controllers have a special emulation mode. It is activated by setting the TE pin to logic HIGH level during reset. In this mode the internal CPU core is inactive and the I/O buses are available via port 0 and port 1 to allow the emulator the access to the on-chip peripherals. The emulator contains a special emulation CPU with a MARC4 core and additional breakpoint logic and takes over the core function. The basic function of the emulator is to evaluate the customer's program and hardware in real time. Thus permits the analysation of any timing, hardware or software problems the simulation of the application. For more informations about emulation see "Emulator Manual".

3.2 MARC4 Instruction Set

The MARC4 instruction set is optimized for the high level programming language qFORTH. A lot of MARC4 instructions are qFORTH words. This enables the compiler to generate fast and compact program code. The MARC4 is a zero address machine with a compact and efficient instruction code. Most of the instructions are single byte instructions. This operations are performed and no source or destination address information. Only BRANCH, CALL and RAM access instructions need address informations and a length of two bytes for long address operations. In all there are five types of instruction formats with a length of one and two bytes.

Zero address operations like arithmetical, logical, shift and rotate operations are performed with data placed on the top of expression stack (TOS and TOS–1). Also I/O– and stack operations are single byte zero address operations and are performed with the top expression stack location.

A literal is a 4-bit constant value which is placed on the data stack. In the MARC4 native code they are represented as LIT_<value>, where <value> is the hexadecimal representation from 0 to 15 (0...F). This range is a result of the MARC4's 4-bit data width. The 6-bit short address and the 12-bit long address formats are both used to address the byte-wide ROM via CALL and conditional branch instructions. This results in a ROM address space of up to 4K*8-bit words.

The MARC4 instruction includes both short and long call instructions as well as conditional branch instructions. On execution the address part of the instructions word are directly loaded into the program counter. Long call and branch instructions can jump anywhere within the program memory area..The lower six bits from the short call (SCALL) and short branch (SBRA) instruction are handled in different way. The six bit SCALL address is multiplied by three and then loaded into the PC. This allows calls within the zero page (000 to 1FFh). The six bit SBRA address is loaded immediately into the lower six bits of the PC. This allows jumps within the 64 byte segment addressed by the upper six bits of the PC.

The CALL and SCALL instructions write the incremented program counter contents to the return stack. This address is loaded back to the PC when the associated EXIT or RTI instruction is encountered. The long RAM address format is used by the four 8-bit RAM address registers which can be pre-increment, post-decrement or loaded directly from the MARC4's internal bus. This results in a direct accessible RAM address space of up to 256 × 4-bit.

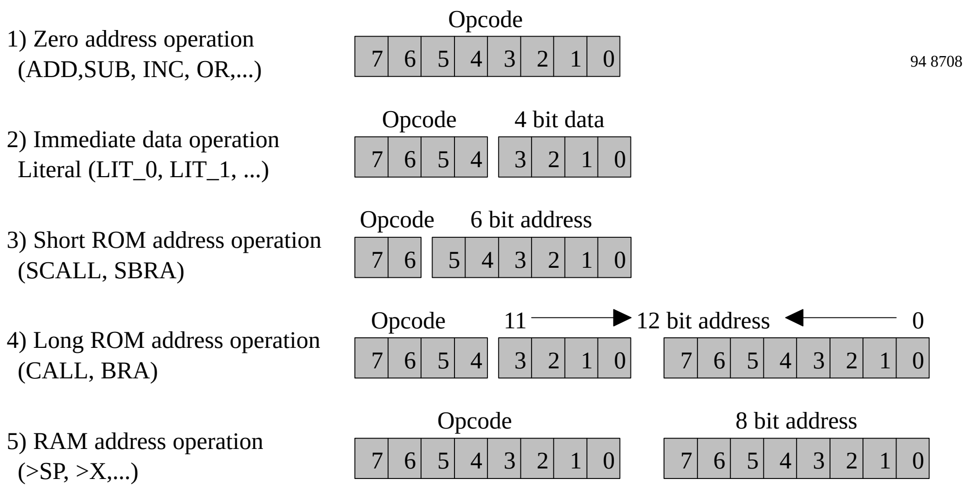


Figure 21. MARC4 Opcode Formats

3.2.1 MARC4 Instruction Set Overview

Mnemonic	Description	Cycles/ Bytes
	Arithmetic operations:	
ADD	Add	1/1
ADDC	Add with carry	1/1
SUB	Subtract	1/1
SUBB	Subtract with borrow	1/1
DAA	Decimal adjust	1/1
INC	Increment TOS	1/1
DEC	Decrement TOS	1/1
DECR	Decrement. 4-bit index on return stack	2/1
	Compare operations:	
CMP_EQ	Compare equal	1/1
CMP_NE	Compare not equal	1/1
CMP_LT	Compare less than	1/1
CMP_LE	Compare less equal	1/1
CMP_GT	Compare greater than	1/1
CMP_GE	Compare greater equal	1/1
	Logical operations:	
XOR	Exclusive OR	1/1
AND	AND	1/1
OR	OR	1/1
NOT	1’s complement	1/1
SHL	Shift left into carry	1/1
SHR	Shift right into carry	1/1
ROL	Rotate left through carry	1/1
ROR	Rotate right through carry	1/1
	Flag operations:	
TOG_BF	Toggle branch flag	1/1
SET_BFC	Set branch flag	1/1
DI	Disable all interrupts	1/1
CCR!	Store TOS into CCR	1/1
CCR@	Fetch CCR onto TOS	1/1
	Program branching:	
BRA $xxx	Conditional long branch	2/2
CALL $xxx	Long call (current page)	3/2
SBRA $xxx	Conditional short branch	2/1
SCALL$xxx	Short call (zero page)	2/1
EXIT	Return from subroutine	2/1
RTI	Return from interrupt	2/1
SWI	Software interrupt	1/1
SLEEP	Activate sleep mode	1/1
NOP	No operation	1/1

Mnemonic	Description	Cycles/ Bytes
	Register operations:	
SP@	Fetch the current SP	2/1
RP@	Fetch the current RP	2/1
X@	Fetch the contents of X	2/1
Y@	Fetch the contents of Y	2/1
SP!	Move the top 2 into SP	2/1
RP!	Move the top 2 into RP	2/1
X!	Move the top 2 into X	2/1
Y!	Move the top 2 into Y	2/1
>SP $xx	Store direct address to SP	2/2
>RP $xx	Store direct address to RP	2/2
>X $xx	Store direct address into X	2/2
>Y $xx	Store direct address into Y	2/2
	Stack operations:	
SWAP	Exchange the top 2 nibble	1/1
OVER	Copy TOS-1 to the top	1/1
DUP	Duplicate the top nibble	1/1
ROT	Move TOS-2 to the top	3/1
DROP	Remove the top nibble	1/1
>R	Move the top nibble onto the return stack	1/1
2>R	Move the top 2 nibble onto the return stack	3/1
3>R	Move the top 3 nibble onto the return stack	4/1
R@	Copy 1 nibble from the re-turn stack	1/1
2R@	Copy 2 nibbles from the return stack	2/1
3R@	Copy 3 nibbles from the return stack	4/1
DROPR	Remove the top of return stack (12-Bit)	1/1
LIT_n	Push immediate value (1 nibble) onto TOS	1/1
	ROM data operations:	
TABLE	Fetch 8-bit constant from ROM	3

Mnemonic	Description	Cycles/ Bytes
Memory operations:		
[X]@ [Y]@	Fetch 1 nibble from RAM indirect addressed by X- or Y-register	1/1
[+X]@ [+Y]@	Fetch 1 nibble from RAM indirect addr. by pre-increm. X- or Y-register	1/1
[X-]@ [Y-]@	Fetch 1 nibble from RAM indirect addr. by post-decrem. X- or Y-register	1/1
[>X]@ $xx [>Y]@ $xx	Fetch 1 nibble from RAM direct addressed by X- or Y-register	2/2
[X]! [Y]!	Store 1 nibble into RAM indirect addressed by [X]	1/1
[+X]! [+Y]!	Store 1 nibble into RAM indirect addressed by pre-incremented [X]	1/1
[X-]! [Y-]!	Store 1 nibble into RAM indirect addr. by post-decrem. X- or Y-register	1/1
[>X]! $xx [>Y]! $xx	Store 1 nibble into RAM direct addressed by X- or Y-register	2/2
I/O operations:		
IN	Read I/O-Port onto TOS	1/1
OUT	Write TOS to I/O port	1/1

3.2.2 qFORTH Language Overview

MARC4 controller are programmed in the high level language qFORTH which is based upon the FORTH-83 language standard, the **qFORTH** compiler generates native code for a 4-bit FORTH-architecture single chip microcomputer, the TEMIC **MARC4**.**MARC4** applications are all programmed in **qFORTH** which is designed specifically for efficient real time control. Since the qFORTH compiler generates highly optimized code, there is no advantage or point in programming the **MARC4** in assembly code. The high level of code efficiency generated by the qFORTH compiler is achieved by the use of modern optimization techniques such as branch-instruction size minimization, fast procedure calls, pointer tracking and many peephole optimizations.

Langage features:

Expandability
Many of the fundamental qFORTH operations are directly implemented in the MARC4 instruction set.

Stack oriented
All operations communicate with one another via the data stack and use the reverse polish form of notation (RPN)

Structured programming
qFORTH supports structured programming

Reentrant
Different tasks can share the same code.

Recursive
qFORTH routines can call themselves.

Native code inclusion
In qFORTH there is no separation of high level constructs from the native code mnemonics.

3.3 The qFORTH language -Quick Reference Guide

3.3.1 Arithmetic/Logical

-	**EXP (n1 n2 — n1-n2)**	Subtract the top two values
+	**EXP (n1 n2 — n1+n2)**	Add up the two top 4-bit values
-C	**EXP (n1 n2 — n1+/n+/C)**	1’s compl. subtract with borrow
+C	**EXP (n1 n2 — n1+n2+C)**	Add with carry top two values
1+	**EXP (n — n+1)**	Increment the top value by 1
1-	**EXP (n — n–1)**	Decrement the top value by 1
2*	**EXP (n — n*2)**	Multiply the top value by 2
2/	**EXP (n — n DIV 2)**	Divide the 4-bit top value by 2
D+	**EXP (d1 d2 — d1+d2)**	Add the top two 8-bit values
D-	**EXP (d1 d2 — d1-d2)**	Subtract the top two 8-bit values
D2/	**EXP (d — d/2)**	Divide the top 8-bit value by 2
D2*	**EXP (d — d*2)**	Multiply the top 8-bit value by 2
M+	**EXP (d1 n — d2)**	Add a 4-bit to an 8-bit value
M-	**EXP (d1 n — d2)**	Subtract 4-bit from an 8-bit value
AND	**EXP (n1 n2 — n1^n2)**	Bitwise AND of top two values
OR	**EXP (n1 n2 — n1 v n2)**	Bitwise OR the top two values
ROL	**EXP (—)**	Rotate TOS left through carry
ROR	**EXP (—)**	Rotate TOS right through carry
SHL	**EXP (n — n*2)**	Shift TOS value left into carry
SHR	**EXP (n — n/2)**	Shift TOS value right into carry
NEGATE	**EXP (n — –n)**	2’s complement the TOS value
DNEGATE	**EXP (d — –d)**	2’s complement top 8-bit value
NOT	**EXP (n — /n)**	1’s complement of the top value
XOR	**EXP (n1 n2 — n3)**	Bitwise Ex-OR the top 2 values

3.3.2 Comparisons

>	**EXP (n1 n2 —)**	If n1>n2, then branch flag set
<	**EXP (n1 n2 —)**	If n1<n2, then branch flag set
>=	**EXP (n1 n2 —)**	If n1>=n2, then branch flag set
<=	**EXP (n1 n2 —)**	If n1<=n2, then branch flag set
<>	**EXP (n1 n2 —)**	If n1<>n2, then branch flag set
=	**EXP (n1 n2 —)**	If n1=n2, then branch flag set
0<>	**EXP (n —)**	If n <>0, then branch flag set
0=	**EXP (n —)**	If n = 0, then branch flag set
D>	**EXP (d1 d2 —)**	If d1>d2, then branch flag set
D<	**EXP (d1 d2 —)**	If d1<d2, then branch flag set
D>=	**EXP (d1 d2 —)**	If d1>=d2, then branch flag set
D<=	**EXP (d1 d2 —)**	If d1<=d2, then branch flag set
D=	**EXP (d1 d2 —)**	If d1=d2, then branch flag set
D<>	**EXP (d1 d2 —)**	If d1<>d2, then branch flag set
D0<>	**EXP (d —)**	If d <>0, then branch flag set
D0=	**EXP (d —)**	If d =0, then branch flag set
DMAX	**EXP (d1 d2 — dMax)**	8-bit maximum value of d1, d2
DMIN	**EXP (d1 d2 — dMin)**	8-bit minimum value of d1, d2
MAX	**EXP (n1 n2 — nMax)**	4-bit maximum value of n1, n2
MIN	**EXP (n1 n2 — nMin)**	4-bit minimum value of n1, n2

3.3.3 Control Structures

AGAIN	**EXP (—)**	Ends an infinite loop BEGIN .. AGAIN
BEGIN	**EXP (—)**	BEGIN of most control structures
CASE	**EXP (n — n)**	Begin of CASE .. ENDCASE block
DO	**EXP (limit start —)**	Initializes an iterative DO..LOOP
	RET (— u|limit|start)	
ELSE	**EXP (—)**	Executed when IF condition is false
ENDCASE	**EXP (n —)**	End of CASE..ENDCASE block
ENDOF	**EXP (n — n)**	End of <n> OF .. ENDOF block
EXECUTE	**EXP (ROMAddr —)**	Execute word located at ROMAddr
EXIT	**RET (ROMAddr —)**	Unstructured EXIT from ‘:’-definition
IF	**EXP (—)**	Conditional IF .. ELSE .. THEN block
LOOP	**EXP (—)**	Repeat LOOP, if index+1<limit
<n> OF	**EXP (c n —)**	Execute CASE block, if n =c
REPEAT	**EXP (—)**	Unconditional branch to BEGIN of BEGIN .. WHILE REPEAT
THEN	**EXP (—)**	Closes an IF statement
UNTIL	**EXP (—)**	Branch to BEGIN, if condition is false
WHILE	**EXP (—)**	Execute WHILE .. REPEAT block, if condition is true
+LOOP	**EXP (n —)**	Repeat LOOP, if I+n < limit
	RET (u|limit|I — u|limit|I+n)	
#DO	**EXP (n —) RET (— u|u|n)**	Execute the #DO .. #LOOP block n-times
#LOOP	**EXP (—)**	Decrement loop index by 1 downto zero
	RET (u|u|I—u|u|I–1)	
?DO	**EXP (Limit Start —)**	if start=limit, skip LOOP block
?LEAVE	**EXP (—)**	Exit any loop, if condition is true
–?LEAVE	**EXP (—)**	Exit any loop, if condition is false

3.3.4 Stack Operations

0 .. Fh,	**EXP (— n)**	
0 .. 15	**EXP (— n)**	Push 4-bit literal on EXP stack
’ <name>	**EXP (— ROMAddr)**	Places ROM address of colon-definition <name> on EXP stack
<ROT	**EXP (n1 n2 n — n n1 n2)**	Move top value to 3rd stack pos.
>R	**EXP (n —) RET (— u|u|n)**	Move top value onto the return stack
?DUP	**EXP (n — n n)**	Duplicate top value, if n <>0
DEPTH	**EXP (— n)**	Get current expression stack depth
DROP	**EXP (n —)**	Remove the top 4-bit value
DUP	**EXP (n — n n)**	Duplicate the top 4-bit value
I	**EXP (— I) RET (u|u|I — u|u|I)**	Copy loop index I from return to expression stack
J	**EXP (— J)**	Fetch index value of outer loop [2nd return stack level
	RET (u|u|J u|u|I — u|u|J u|u|I)	entry]
NIP	**EXP (n1 n2 — n2)**	Drop second to top 4-bit value
OVER	**EXP (n1 n2 — n1 n2 n1)**	Copy 2nd over top 4-bit value
PICK	**EXP (× — n[x])**	Copy the x-th value from the expression stack onto TOS
RFREE	**EXP (— n)**	Get # of unused RET stack entries
R>	**EXP (— n) RET (u|u|n —)**	Move top 4-bits from return to expression stack
R@	**EXP (— n)**	Copy top 4-bits from return to expression stack
	RET (u|u|n — u|u|n)	
ROLL	**EXP (n —)**	Move n-th value within stack to top

ROT	**EXP (n1 n2 n — n2 n n1)**	Move 3rd stack value to top pos.
SWAP	**EXP (n1 n2 — n2 n1)**	Exchange top two values on stack
TUCK	**EXP (n1 n2 — n2 n1 n2)**	Duplicate top value, move under second item
2>R	**EXP (n1 n2 —)**	Move top two values from expression to return stack
	RET (— u|n2|n1)	
2DROP	**EXP (n1 n2 —)**	Drop top 2 values from the stack
2DUP	**EXP (d — d d)**	Duplicate top 8-bit value
2NIP	**EXP (d1 d2 — d2)**	Drop 2nd 8-bit value from stack
2OVER	**EXP (d1 d2 — d1 d2 d1)**	Copy 2nd 8-bit value over top value
2<ROT	**EXP (d1 d2 d — d d1 d2)**	Move top 8-bit value to 3rd pos’n
2R>	**EXP (— n1 n2)**	Move top 8-bits from return to expression stack
	RET (u|n2|n1 —)	
2R@	**EXP (— n1 n2)**	Copy top 8-bits from return to expression stack
	RET (u|n2|n1 — u|n2|n1)	
2ROT	**EXP (d1 d2 d — d2 d d1)**	Move 3rd 8-bit value to top value
2SWAP	**EXP (d1 d2 — d2 d1)**	Exchange top two 8-bit values
2TUCK	**EXP (d1 d2 — d2 d1 d2)**	Tuck top 8-bits under 2nd byte
3>R	**EXP (n1 n2 n3 —)**	Move top 3 nibbles from the expression onto
	RET (— n3|n2|n1)	the return stack
3DROP	**EXP (n1 n2 n3 —)**	Remove top 3 nibbles from stack
3DUP	**EXP (t — t t)**	Duplicate top 12-bit value
3R>	**EXP (— n1 n2 n3)**	Move top 3 nibbles from return to the expression stack
	RET (n3|n2|n1 —)	
3R@	**EXP (— n1 n2 n3)**	Copy 3 nibbles (1 entry) from the return
	RET (n3|n2|n1 — n3|n2|n1)	to the expression stack

3.3.5 Memory Operations

!	**EXP (n addr —)**	Store a 4-bit value in RAM
@	**EXP (addr — n)**	Fetch a 4-bit value from RAM
+!	**EXP (n addr —)**	Add 4-bit value to RAM contents
1+!	**EXP (addr —)**	Increment a 4-bit value in RAM
1–!	**EXP (addr —)**	Decrement a 4-bit value in RAM
2!	**EXP (d addr —)**	Store an 8-bit value in RAM
2@	**EXP (addr — d)**	Fetch an 8-bit value from RAM
D+!	**EXP (d addr —)**	Add 8-bit value to byte in RAM
D–!	**EXP (d addr —)**	Subtract 8-bit value from a byte in RAM
DTABLE@	**EXP (ROMAddr n — d)**	Indexed fetch of a ROM constant
DTOGGLE	**EXP (d addr —)**	Exclusive-OR 8-bit value with byte in RAM
ERASE	**EXP (addr n —)**	Sets n memory cells to 0
FILL	**EXP (addr n n1 —)**	Fill n memory cells with n1
MOVE	**EXP (n from to —)**	Move a n-digit array in memory
ROMByte@	**EXP (ROMAddr — d)**	Fetch an 8-bit ROM constant
TOGGLE	**EXP (n addr —)**	Ex-OR value at address with n
3!	**EXP (nh nm nl addr —)**	Store 12-bit value into a RAM array
3@	**EXP (addr — nh nm nl)**	Fetch 12-bit value from RAM
T+!	**EXP (nh nm nl addr —)**	Add 12-bits to 3 RAM cells
T–!	**EXP (nh nm nl addr —)**	Subtract 12-bits from 3 nibble RAM array
TD+!	**EXP (d addr —)**	Add byte to a 3 nibble RAM array
TD–!	**EXP (d addr —)**	Subtract byte from 3 nibble array

3.3.6 Predefined Structures

(ccccccc)		In-line comment definition
\ ccccccc		Comment until end of the line
: <name>	**RET (—)**	Begin of a colon definition
;	**RET (ROMAddr —)**	Exit; ends any colon definition
[FIRST]	**EXP (— 0)**	Index (=0) for first array element
[LAST]	**EXP (— n|d)**	Index for last array element
CODE	**EXP (—)**	Begins an in-line macro definition
END-CODE	**EXP (—)**	Ends an In-line macro definition
ARRAY	**EXP (n —)**	Allocates space for a 4-bit array
2ARRAY	**EXP (n —)**	Allocates space for an 8-bit array
CONSTANT	**EXP (n —)**	Defines a 4-bit constant
2CONSTANT	**EXP (d —)**	Defines an 8-bit constant
LARRAY	**EXP (d —)**	Allocates space for a long 4-bit array with up to 255 elements
2LARRAY	**EXP (d —)**	Allocates space for a long byte array
Index	**EXP (n|d addr—addr')**	Run-time array access using a variable array index
ROMCONST	**EXP (—)**	Define ROM look-up table with 8-bit values
VARIABLE	**EXP (—)**	Allocates memory for 4-bit value
2VARIABLE	**EXP (—)**	Creates an 8-bit variable
<n> ALLOT		Allocate space for <n+1> nibbles of un-initialized RAM
AT <address>		Fixed <address> placement
: INTx	**RET (— ROMAddr)**	Interrupt service routine entry
$AutoSleep		Entry point address on return stack underflow
: $RESET	**EXP (—)**	Entry point on power-on reset

3.3.7 Assembler Mnemonics

ADD	**EXP (n1 n2 — n1+n2)**	Add the top two 4-bit values
ADDC	**EXP (n1 n2 — n1+n2+C)**	Add with carry top two values
CCR!	**EXP (n —)**	Write top value into the CCR
CCR@	**EXP (— n)**	Fetch the CCR onto top of stack
CMP_EQ	**EXP (n1 n2 — n1)**	If n1=n2, then branch flag set
CMP_GE	**EXP (n1 n2 — n1)**	If n1>=n2, then branch flag set
CMP_GT	**EXP (n1 n2 — n1)**	If n1>n2, then branch flag set
CMP_LE	**EXP (n1 n2 — n1)**	If n1<=n2, then branch flag set
CMP_LT	**EXP (n1 n2 — n1)**	If n1<n2, then branch flag set
CMP_NE	**EXP (n1 n2 — n1)**	If n1<>n2, then branch flag set
CLR_BCF	**EXP (—)**	Clear branch and carry flag
SET_BCF	**EXP (—)**	Set branch and carry flag
TOG_BF	**EXP (—)**	Toggle the branch flag
DAA	**EXP (n>9 or C set — n+6)**	BCD arithmetic adjust [addition]
DAS	**EXP (n — 10+/n+C)**	9's complement for BCD subtract
DEC	**EXP (n — n–1)**	Decrement top value by 1
DECR	**RET (u|u|I — u|u|I–1)**	Decrement value on the return stack
DI	**EXP (—)**	Disable interrupts
DROPR	**RET (u|u|u —)**	Drop element from return stack
EXIT	**RET (ROMAddr —)**	Exit from current '**:**'-definition
EI	**EXP (—)**	Enable interrupts
IN	**EXP (port — data)**	Read data from an I/O port
INC	**EXP (n — n+1)**	Increment the top value by 1
NOP	**EXP (—)**	No operation

NOT	**EXP (n — /n)**	1’s complement of the top value
RP!	**EXP (d —)**	Store as return stack pointer
RP@	**EXP (— d)**	Fetch current RET stack pointer
RTI	**RET (RETAddr —)**	Return from interrupt routine
SLEEP	**EXP (—)**	Enter ’sleep-mode’, enable all interrupts
SWI0 SWI7	**EXP (—)**	Software triggered interrupt
SP!	**EXP (d —)**	Store as stack pointer
SP@	**EXP (— d)**	Fetch current stack pointer
SUB	**EXP (n1 n2 — n1-n2)**	2’s complement subtraction
SUBB	**EXP (n1 n2 — n1+/n2+C)**	1’s compl. subtract with borrow
TABLE	**EXP (— d)**	
	RET (RetAddr RomAddr —)	Fetches an 8-bit constant from an address in ROM
OUT	**EXP (data port —)**	Write data to I/O port
X@	**EXP (— d)**	Fetch current × register contents
[X]@	**EXP (— n)**	Indirect × fetch of RAM contents
[+X]@	**EXP (— n)**	Pre-incr. × indirect RAM fetch
[X–]@	**EXP (— n)**	Postdecr. × indirect RAM fetch
[>X]@ $xx	**EXP (— n)**	Direct RAM fetch, × addressed
X!	**EXP (d —)**	Move 8-bit address to × register
[X]!	**EXP (n —)**	Indirect × store of RAM contents
[+X]!	**EXP (n —)**	Pre-incr. × indirect RAM store
[X–]!	**EXP (n —)**	Postdecr. × indirect RAM store
[>X]! $xx	**EXP (n —)**	Direct RAM store, × addressed
Y@	**EXP (— d)**	Fetch current Y register contents
[Y]@	**EXP (— n)**	Indirect Y fetch of RAM contents
[+Y]@	**EXP (— n)**	Pre-incr. Y indirect RAM fetch
[Y–]@	**EXP (— n)**	Postdecr. Y indirect RAM fetch
[>Y]@ $xx	**EXP (— n)**	Direct RAM fetch, Y addressed
Y!	**EXP (d —)**	Move address to Y register
[Y]!	**EXP (n —)**	Indirect Y store of RAM contents
[+Y]!	**EXP (n —)**	Pre-incr. Y indirect RAM store
[Y–]!	**EXP (n —)**	Postdecr. Y indirect RAM store
[>Y]! $xx	**EXP (n —)**	Direct RAM store, Y addressed
>RP $xx	**EXP (—)**	Set return stack pointer
>SP $xx	**EXP (—)**	Set expression stack pointer
>X $xx	**EXP (—)**	Set × register immediate
>Y $xx	**EXP (—)**	Set Y register immediate

Notes:

RET (–)	Return address stack effects
EXP (–)	Expression (or data) stack effects
True condition	Means branch flag set in CCR
False condition	Means branch flag reset in CCR
n	4-bit data value
d	8-bit data value
addr	8-bit RAM address
ROMaddr	12-bit ROM address

4 Electrical Characteristics

4.1 Absolute Maximum Ratings

Voltages are given relative to V_{SS}.

Parameters	Symbol	Value	Unit
Supply voltage	V_{DD}	– 0.3 to + 7.0	V
Input voltage (on any pin)	V_{IN}	V_{SS} –0.3 ≤ V_{IN} ≤ V_{DD} +0.3	V
Output short circuit duration	t_{short}	indefinite	sec
Operating temperature range	T_{amb}	–40 to +85	°C
Storage temperature range	T_{stg}	–40 to +130	°C
Thermal resistance (PLCC)	R_{thJA}	110	K/W
Soldering temperature (t ≤ 10 s)	T_{sd}	260	°C

Stresses greater than those listed under absolute maximum ratings may cause permanent damage to the device. This is a stress rating only and functional operational the device at any condition above those indicated in the operational section of these specification is not implied. Exposure to absolute maximum rating condition for an extended period may affect device reliability. All inputs and outputs are protected against high electrostatic voltages or electric fields. However, precautions to minimize built-up of electrostatic charges during handling are recommended. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (e.g., V_{DD}).

4.2 DC Operating Characteristics

Supply voltage V_{DD} = 2.4 to 6.2 V, V_{SS} = 0 V, T_{amb} = –40 to 85°C, unless otherwise specified

Parameters	Test Conditions / Pins	Symbol	Min.	Typ.	Max.	Unit
Power supply						
Active current (CPU active)	V_{DD} = 2.4 V f_{SYSCL}=1MHz f_{SYSCL}=2MHz V_{DD} = 6.2 V f_{SYSCL}=1MHz f_{SYSCL}=2MHz	I_{DD}		0.9 1.3 2.1 3.6	1.6 2.1 3.1 5.2	mA mA mA mA
Power down current (CPU sleep, RC oscillator active)	V_{DD} = 2.4 V f_{SYSCL}=1MHz f_{SYSCL}=2MHz V_{DD} = 6.2 V f_{SYSCL}=1MHz f_{SYSCL}=2MHz	I_{PD}		0.3 0.4 0.6 0.8	0.5 0.8 1.0 1.3	mA mA mA mA
Sleep current (CPU sleep, RC oscillator inactive)	V_{DD} = 2.4 V V_{DD} = 6.2 V	I_{Sleep}		0.4 0.5	1.0 1.0	µA µA
Sleep current (CPU sleep, RC oscillator inactive)	V_{DD} = 2.4 V V_{DD} = 6.2 V T_{amb} = 25°C	I_{Sleep}			0.8 0.0	µA µA

Supply voltage V_{DD} = 2.4 to 6.2 V, V_{SS} = 0 V, T_{amb} = 25°C, unless otherwise specified

Parameters	Test Conditions / Pins	Symbol	Min.	Typ.	Max.	Unit
Power-on reset threshold voltage: Note x figure xx						
POR threshold voltage		V_{POR}	1.5		2.1	V
POR hysteresis		ΔV_{POR}		100		mV
Schmitt-trigger input voltage: Pin INT6, TA, TB, port 40 and port 3						
Negative-going threshold voltage	V_{DD} = 2.4 to 6.2 V	V_{T-}	V_{SS}		0.3*V_{DD}	V
Positive-going threshold voltage	V_{DD} = 2.4 to 6.2 V	V_{T+}	0.7*V_{DD}		V_{DD}	V
Hysteresis (VT ± VT–)	V_{DD} = 2.4 to 6.2 V	V_H		0.1*V_{DD}		
Input voltage: Pin NRST, TE, NWP, TCL, and port 0, 1, 2, port 43:						
Input voltage LOW	V_{DD} = 2.4 to 6.2 V	V_{IL}	V_{SS}		0.2*V_{DD}	V
Input voltage HIGH	V_{DD} = 2.4 to 6.2 V	V_{IH}	0.8*V_{DD}		V_{DD}	V
Input current: Bidirectional ports 0, 1, 2, 3, input port 4 with pull-up resistor Pin NRST, TCL, INT6						
Input LOW current	V_{DD}= 2.4 V V_{IL}= V_{SS} V_{DD}= 6.2 V	I_{IL}	–2.7 –28	–6.7 –60	–13 –103	µA µA
Input current: Bidirectional ports 0, 1, 2, 3, input port 4 with pull-down resistor Pin TE, NWP, TA, TB						
Input HIGH current	V_{DD} = 2.4 V V_{IH} = V_{DD} V_{DD} = 6.2 V	I_{IH}	2.7 30	6.3 60	12 100	µA µA
Output current: Bidirectional ports 0, 1, 2, 3 and TA, TB						
Output LOW current	V_{DD} = 2.4 V V_{OL} = 0.2*V_{DD} V_{DD} = 6.2 V	I_{OL}	0.8 6	1.6 11	2.8 17	mA mA
Output HIGH current	V_{DD} = 2.4 V V_{OH} = 0.8*V_{DD} V_{DD} = 6.2 V	I_{OH}	–0.6 –4	–1.3 –7.5	–2.2 –12	mA mA
Output current: Pin TCL						
Output LOW current	V_{DD} = 2.4 V V_{OL} = 0.2*V_{DD} V_{DD} = 6.2 V	I_{OL}	1.6 12	3.2 22	5.6 34	mA mA
Output HIGH current	V_{DD} = 2.4 V V_{OH} = 0.8*V_{DD} V_{DD} = 6.2 V	I_{OH}	–1.2 –8	–2.6 –15	–4.4 –24	mA mA

4.3 AC Characteristics

Supply voltage V_{DD} = 2.4 to 6.2 V, V_{SS} = 0 V, T_{amb} = 25°C, unless otherwise specified

Parameters	Test Conditions / Pins	Symbol	Min.	Typ.	Max.	Unit
Timer 2A and 2B input timing						
Timer input clock		f_{TI}			SYSCL	–
Timer input LOW time	Rise/fall time < 10 ns	t_{TIL}	50			ns
Timer input HIFG time	Rise/fall time < 10 ns	t_{TIH}	50			ns
Interrupt request input timing						
Int. request LOW time	Rise/fall time < 10 ns	t_{IRL}	50			ns
Int. request HIGH time	Rise/fall time < 10 ns	t_{IRH}	50			ns
TCL clock						
TCL input clock		f_{TCL}			2	MHz
TCL input LOW time		t_{TCLL}	0,250			µs
TCL input HIGH time		t_{TCLH}	0.250		10	µs
TCL rise time TCL fall time		t_{TCLR} t_{TCLF}			10	ns
Reset timing						
Power-on reset time		T_{POR}		100	500	µs
NRES input LOW time		T_{POR}	4*SYSCL			µs
EEPROM write cycle						
EEPROM write time	Note 1	t_{EEW}		16		ms
EEPROM write cycles		n_W	$5*10^5$	10^6		–
Operation cycle time						
System clock cycle	CCS = 1 Note 1 CCS = 0	t_{SYSCL}		477 954		ns ns
RC oscillator						
Frequency	Note 1	f_{RC1}		1048		kHz
Stability	Note 1	$\Delta f/f$		2000		ppm
Stabilization time	Note 1	t_S		1		mss
32 kHz oscillator						
Frequency		f_X		32.768		kHz
Start up time		t_{SQ}				s
Stability	Note 2	$\Delta f/f$	–10		10	ppm
Integrated input/output capacitances		C_{IN} C_{OUT}		10		pF
External 32 kHz crystal parameters						
Crystal frequency		f_X		32.768		kHz
Series resistance		RS		30	50	kΩ
Static capacitance		C0		1.5		pF
Dynamic capacitance		C1		3		fF

Note 1: With connected crystal (pin 5, 6) and after start up time of crystal oscillator.

Note 2: Depend on the connected quartz crystal.

Crystal

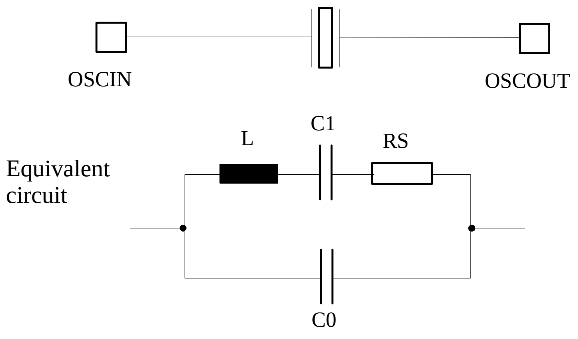


Figure 22. Equivalent crystal circuit

Power-on reset

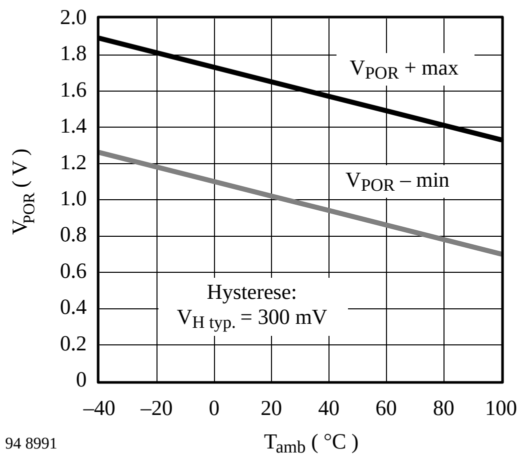


Figure 23. Thresholds for POR vs. ambient temperature

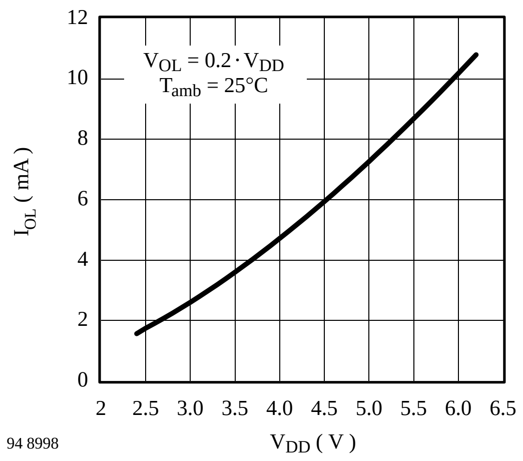


Figure 24. Output LOW current vs. supply voltage

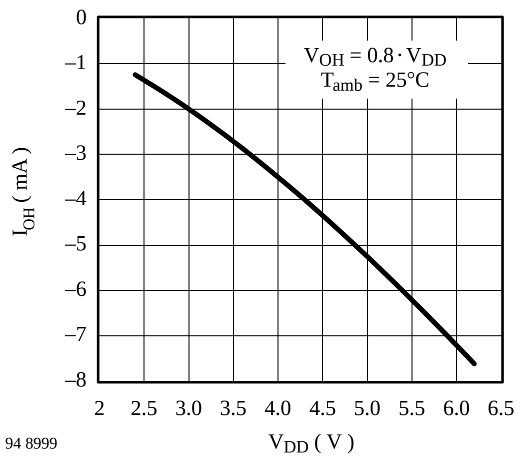


Figure 26. Output HIGH current vs. supply voltage

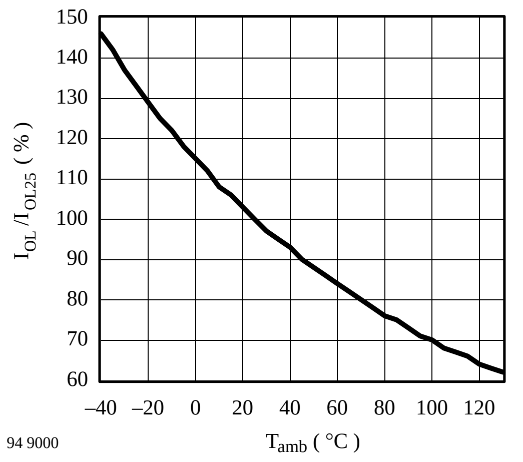


Figure 25. Output LOW current standardized to 25°C vs. temp.

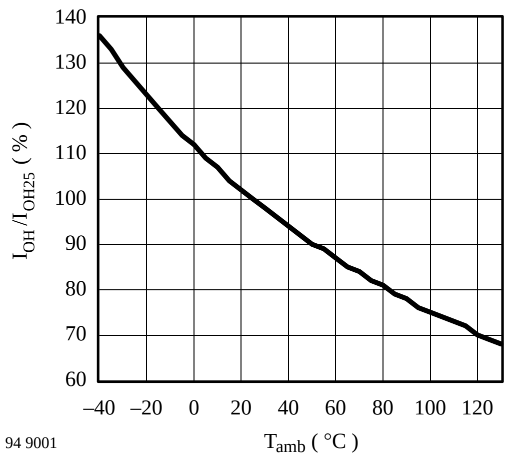


Figure 27. Output HIGH current standardized to 25°C vs temp.

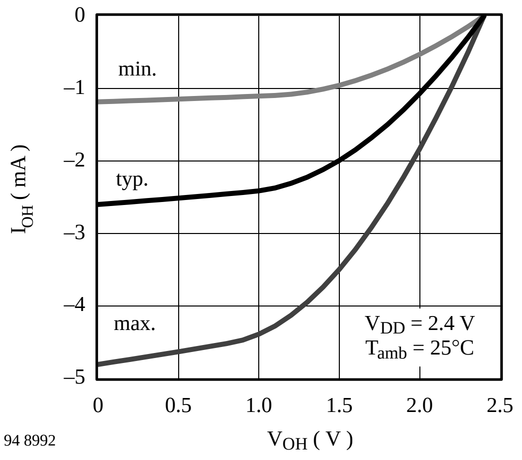


Figure 28. Output HIGH current vs. output HIGH voltage

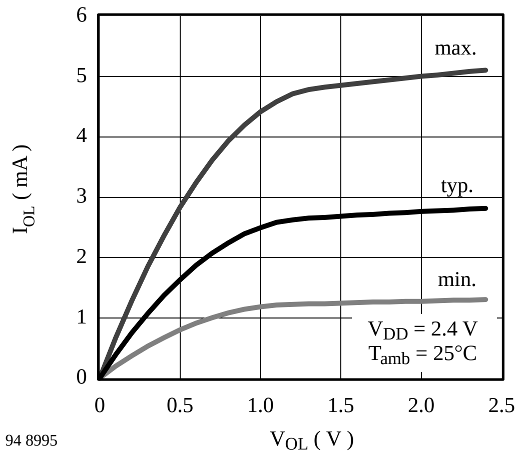


Figure 31. Output LOW current vs. output LOW voltage

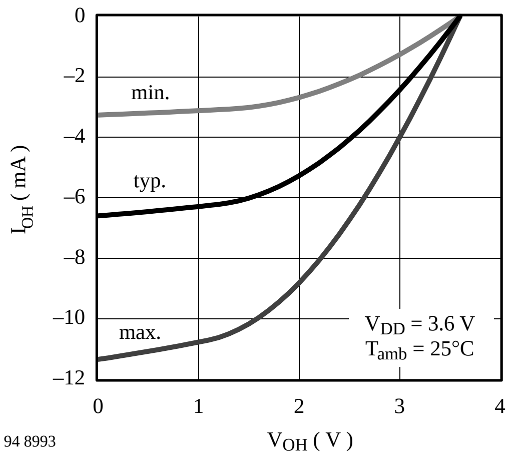


Figure 29. Output HIGH current vs. output HIGH voltage

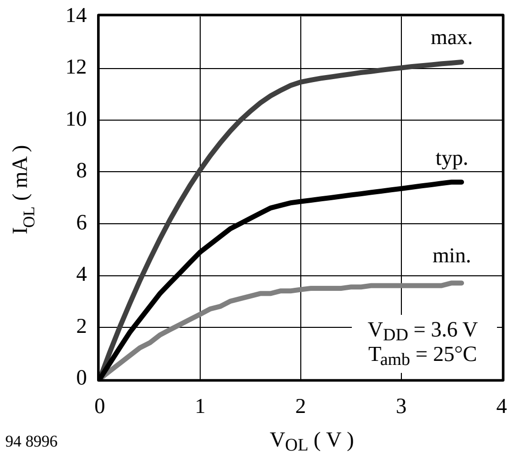


Figure 32. Output LOW current vs. output LOW voltage

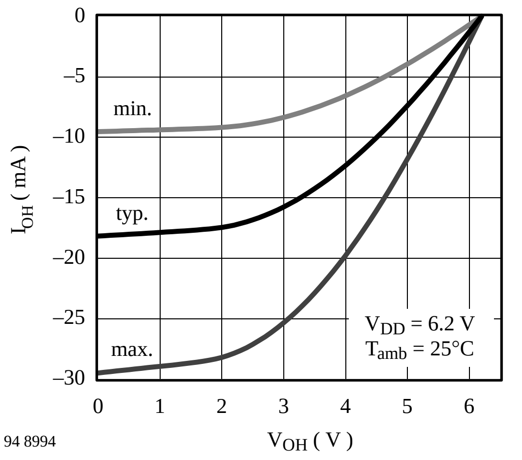


Figure 30. Output HIGH current vs. output HIGH voltage

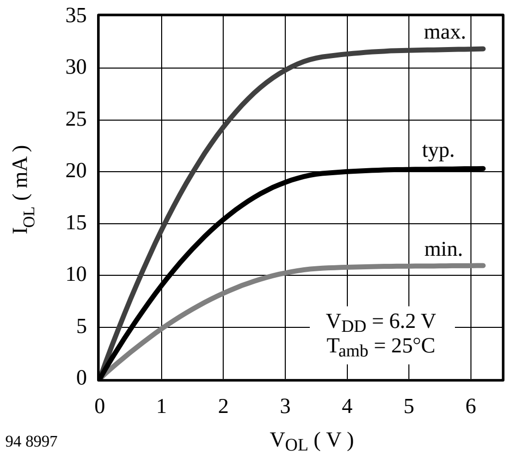


Figure 33. Output LOW current vs. output LOW voltage

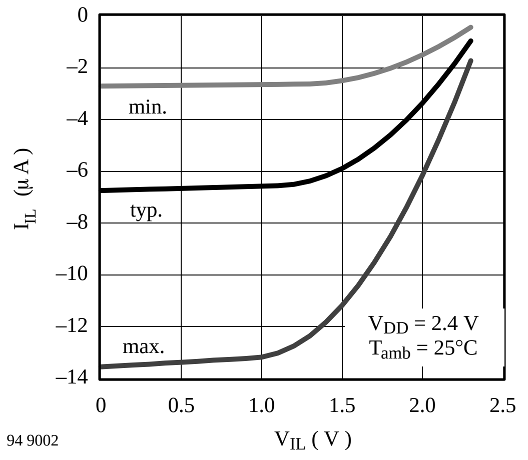


Figure 34. Input LOW current vs. input LOW voltage

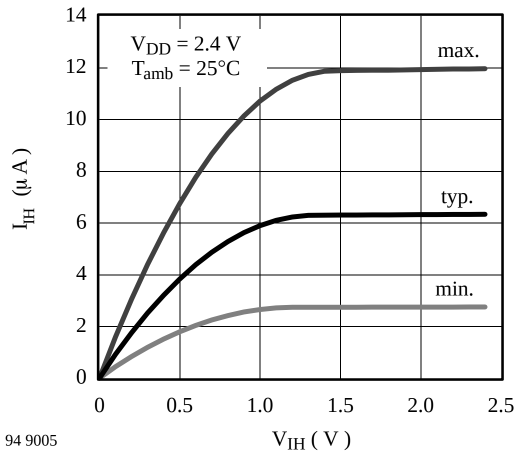


Figure 37. Input HIGH current vs. input HIGH voltage

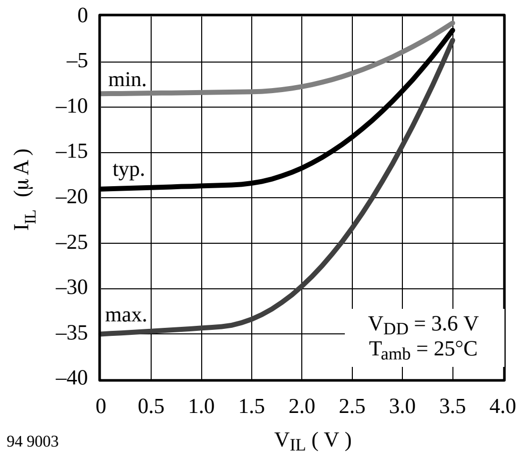


Figure 35. Input LOW current vs. input LOW voltage

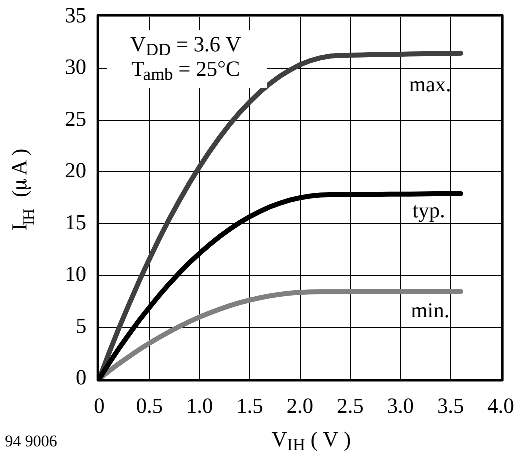


Figure 38. Input HIGH current vs. input HIGH voltage

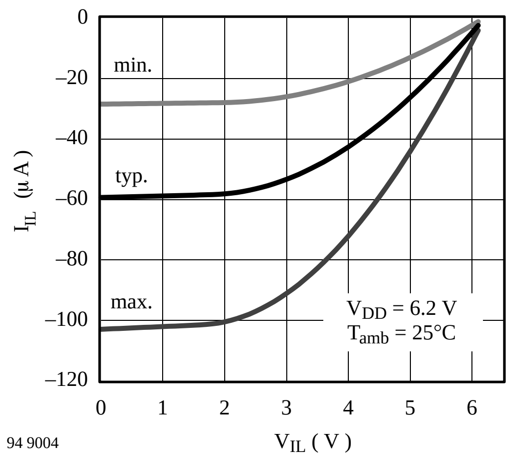


Figure 36. Input LOW current vs. input LOW voltage

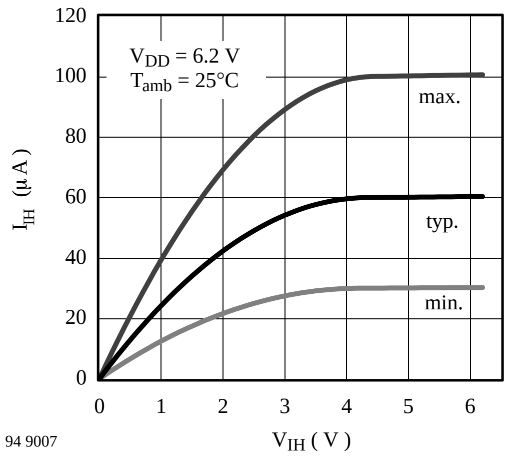


Figure 39. Input HIGH current vs. input HIGH voltage

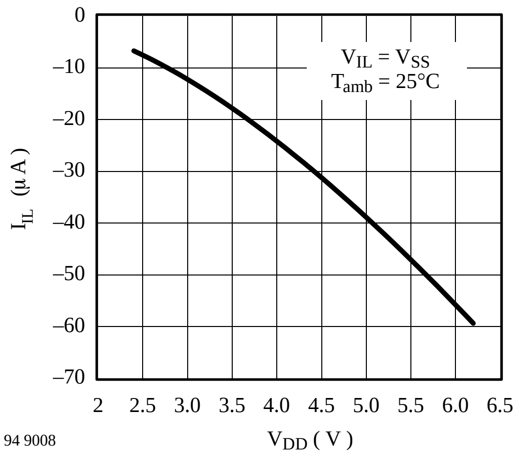


Figure 40. Input LOW current vs. supply voltage

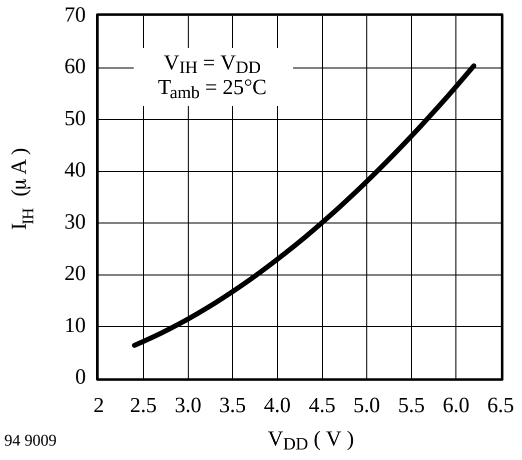


Figure 42. Input HIGH current vs. supply voltage

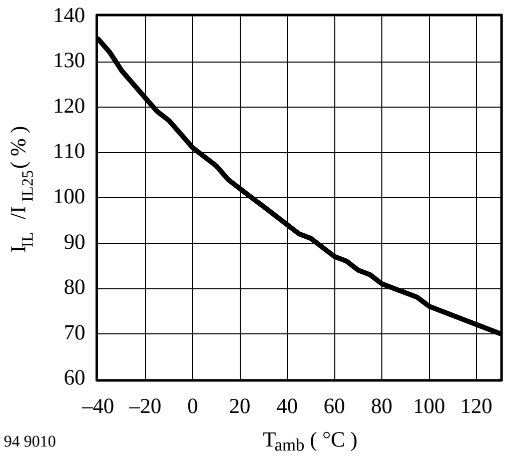


Figure 41. Input LOW current standardized to 25°C vs. temperature

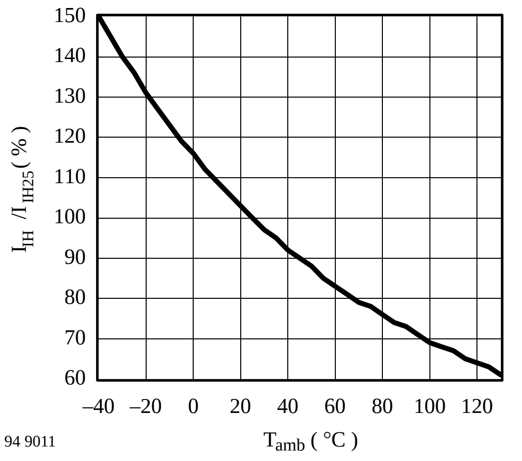


Figure 43. Input HIGH current standardized to 25°C vs. temperature

4.4 Schmitt-Trigger Inputs

The following figures show the Schmitt-trigger input specs used at timer inputs TA, TB and interrupt inputs.

Note: The values for switch levels are standardized to supply voltage.

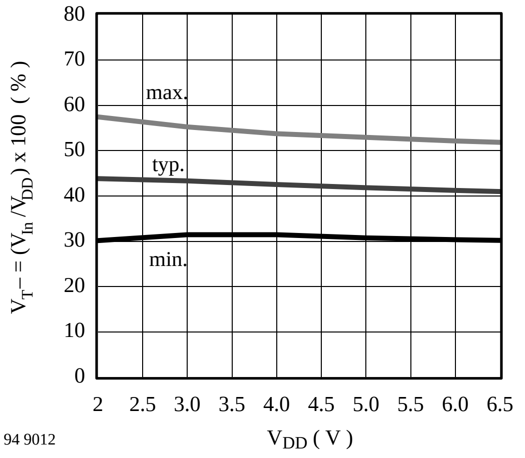


Figure 44. Schmitt-trigger positive going threshold voltage

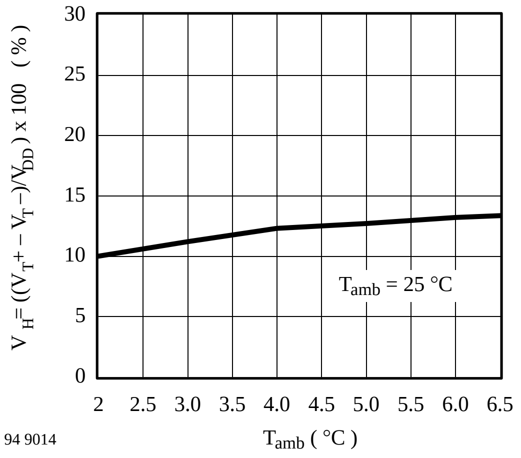


Figure 46. Schmitt-trigger hysteresis vs. supply voltage

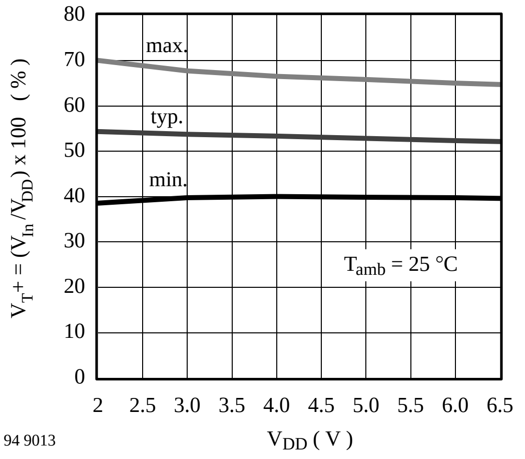


Figure 45. Schmitt-trigger negative going threshold voltage

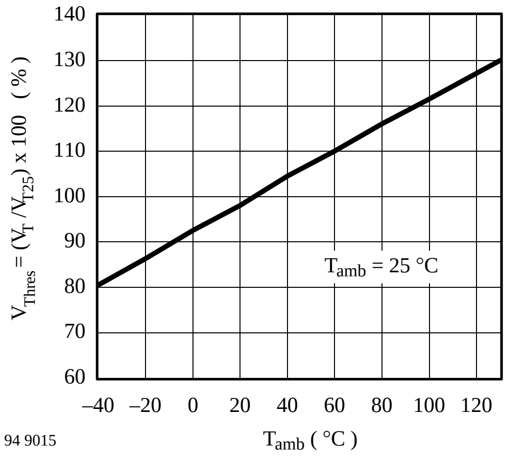


Figure 47. Threshold temperature drift

Note: For a pulse to be recognizable, it must be a minimum of 50 ns long with a rise time ≤ 10 ns.

5 Pad Layout

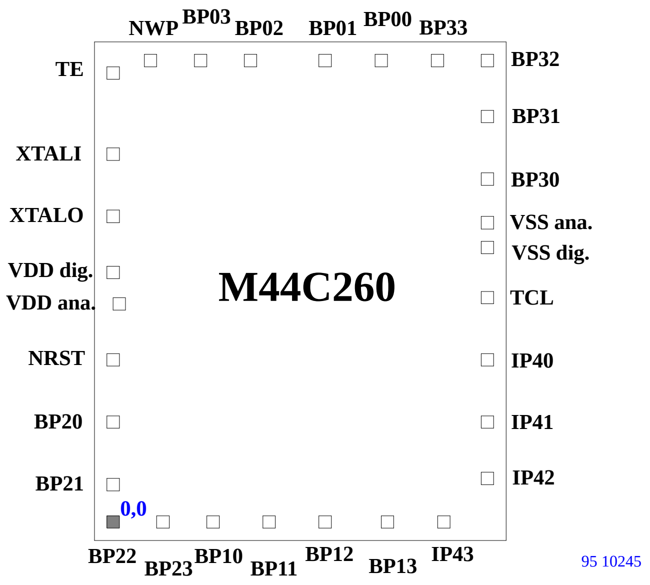


Figure 48. Pad Layout

Table 8. Pad coordinates

Number	Name	X point	Y point	Number	Name	X point	Y point
1	BP22	0.0	0.0	16	BP32	3056.0	3741.5
2	BP23	404.5	0.0	17	BP33	2651.5	3741.5
3	BP10	809.0	0.0	18	BP00	2247.0	3741.5
4	BP11	1398.5	0.0	19	BP01	1830.5	3741.5
5	BP12	1811.0	0.0	20	BP02	1136.5	3741.5
6	BP13	2223.5	0.0	21	BP03	720.0	3741.5
7	IP43	2686.5	0.0	22	NWP	303.5	3741.5
8	IP42	3056.0	509.0	23	TE	0.0	3660.0
9	IP41	3056.0	965.0	24	XTALI	0.0	3103.0
10	IP40	3056.0	1363.0	25	XTALO	0.0	2625.0
11	TCL	3056.0	1792.0	26	VDD dig.	0.0	2315.0
12	VSS dig.	3056.0	2247.5	27	VDD ana.	24.0	2044.0
13	VSS ana.	3056.0	2457.5	28	NRST	0.0	1707.0
14	BP30	3056.0	2720.5	29	BP20	0.0	1164.5
15	BP31	3056.0	3301.0	30	BP21	0.0	424.5

The M44C260 is also available in the form for COB mounting. Therefore the substrate, i.e., the backside of the die, sould be connected to V_{SS}.

Die size: 3.51 mm x 4.19 mm

Pad size: 90 µm * 90 µm

Thickness: 380 ± 25 µm

6 Ordering Information

Pin options

Please select the option setting from the list below.

Pin	Output		Input	
	CMOS	Open Drain	Pull Up	Pull Down
BP00				
BP01				
BP02				
BP03				
BP10				
BP11				
BP12				
BP13				
BP20				
BP21				
BP22				
BP23				
BP30				
BP31				
BP32				
BP33				
IP40–INT6	■	■		
IP41–TA				
IP42–TB				
IP43	■	■		
NWP	■	■		
TE	■	■		

ROM code

Please insert ROM CRC.

Size: ________ KByte **CRC:** ________ hex

Approval

Date: ____________ Signature: ____________________

Ozone Depleting Substances Policy Statement

It is the policy of **TEMIC TELEFUNKEN microelectronic GmbH** to

1. Meet all present and future national and international statutory requirements.

2. Regularly and continuously improve the performance of our products, processes, distribution and operating systems with respect to their impact on the health and safety of our employees and the public, as well as their impact on the environment.

It is particular concern to control or eliminate releases of those substances into the atmosphere which are known as ozone depleting substances (ODSs).

The Montreal Protocol (1987) and its London Amendments (1990) intend to severely restrict the use of ODSs and forbid their use within the next ten years. Various national and international initiatives are pressing for an earlier ban on these substances.

TEMIC TELEFUNKEN microelectronic GmbH semiconductor division has been able to use its policy of continuous improvements to eliminate the use of ODSs listed in the following documents.

1. Annex A, B and list of transitional substances of the Montreal Protocol and the London Amendments respectively

2. Class I and II ozone depleting substances in the Clean Air Act Amendments of 1990 by the Environmental Protection Agency (EPA) in the USA

3. Council Decision 88/540/EEC and 91/690/EEC Annex A, B and C (transitional substances) respectively.

TEMIC can certify that our semiconductors are not manufactured with ozone depleting substances and do not contain such substances.

TEMIC TELEFUNKEN microelectronic GmbH, P.O.B. 3535, D-74025 Heilbronn, Germany
Telephone: 49 (0)7131 67 2831, Fax Number: 49 (0)7131 67 2423