# Infineon Technologies - SAF-C165UTAH-LF V1.3 Datasheet

**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | C166 |
| Core Size | 16-Bit |
| Speed | 36MHz |
| Connectivity | EBI/EMI, IOM-2/PCM, SPI, UART/USART, USB |
| Peripherals | DMA, POR, PWM, WDT |
| Number of I/O | 72 |
| Program Memory Size | - |
| Program Memory Type | ROMless |
| EEPROM Size | - |
| RAM Size | 3K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 3.6V |
| Data Converters | - |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 144-LQFP |
| Supplier Device Package | PG-TQFP-144 |
| Purchase URL | https://www.e-xfl.com/product-detail/infineon-technologies/saf-c165utah-lf-v1-3 |

  - S/G access support
- Four On-Chip Independent Full-Duplex HDLC Formatters
  - 8 independent 8-byte FIFOs for each transmit and receive channel
- USART Interface with AutoBaud Support (1,200 bit/s - 230,400 bit/s)
  - AT-Command sensitive AutoBaud Detection

## 1.1 Key Features

The C165UTAH is a new low-cost member of the Infineon Communication Controller family. The device has the following features:

- C166 Static Core with Peripherals including:
  - Full-static core up to 18 MIPS (@36 MHz)
  - Peripheral Event Controller (PEC) for 8 independent DMA channels
  - 16 Dynamically Programmable Priority-Level Interrupt System
  - Eight External Interrupts
  - Up to 72 SW-configurative Input/Output (I/O) Ports, some with Interrupt Capabilities
  - 8-bit or 16-bit External Data Bus
  - Multiplexed or Demultiplexed Address/Data Bus
  - Up to 8-Mbyte Linear Address Space for Code and Data
  - Five Programmable Chip-Select Lines with Wait-State Generator Logic
  - On-Chip 3,072-Byte Dual-Port SRAM for user applications
  - On-Chip 1,024-Byte Special Function Register Area
  - On-Chip PLL with Output Clock Signal
  - Five Multimode General Purpose Timers
  - On-Chip Programmable Watchdog Timer
  - Glueless Interface to EPROM, Flash EPROM and SRAM
  - Low-Power Management Supporting Idle-, Power-Down- and Sleep-Mode and additional CPU clock slow-down mode with mode control for each peripheral
  - USART interface with Auto Baud Rate detection up to 230,400 kbit/s
  - USART Baud Rate generation in asynchronous mode up to 2.25 MBaud @ 36 MHz
  - USART Baud Rate generation in synchronous mode up to 4.5 MBaud @ 36 MHz
  - USART standard Baud Rates generation with very small deviation (230.4 kBaud < 0.01%, 460.8 kBaud < 0.15 %, 691.2 kBaud < 0.04 %, 921.6 kBaud < 0.15 % ) @ 36 MHz
  - High speed Serial Synchronous Channel Interface (SSC) with ALIS-3.0 and AC97 compatibility up to 18 MBaud in SSC Master Mode and up to 9 MBaud in SSC Slave Mode @ 36 MHz

**Table 8 Power/Ground Signals**

| Pin No. | Symbol | Input (I) Output (O) | Function |
|---|---|---|---|
| 130 | VDDU | - | **Digital Supply Voltage for USB Transceiver** VDDU supplies the USB transceiver only and is internally not connected to VDD in order to separate possible noise influence. External, on board level, the VDDU can be connected to the same power supply as VDD. |
| 127 | VSSU | - | **Digital Ground for USB Transceiver** VSSU is connected to the USB transceiver only and is internally not connected to the common ground in order to separate possible noise influence. External, on board level, the VSSU can be connected to the same ground as VSS. |

**Table 9 Unconnected Pins**

| Pin No. | Symbol | Input (I) Output (O) | Function |
|---|---|---|---|
| 1, 2, 36, 72, 73, 74, 106, 107, 108 | unconnected | - | These pins are unconnected - no function. Reserved for future use. |

a memory table. The C165UTAH has 8 PEC channels each of which offers such fast interrupt-driven data transfer capabilities.

### Memory Areas

The memory space of the C165UTAH is configured in a Von Neumann architecture which means that code memory, data memory, registers and I/O ports are organized within the same linear address space which covers up to 8 MBytes. The entire memory space can be accessed bytewise or wordwise. Particular portions of the on-chip memory have additionally been made directly bit addressable.

**A 16-bit wide internal RAM (IRAM)** provides fast access to General Purpose Registers (GPRs), user data (variables) and system stack. The internal RAM may also be used for code. A unique decoding scheme provides flexible user register banks in the internal memory while optimizing the remaining RAM for user data. The size of the internal RAM is 3 KByte.

The CPU disposes of an actual register context consisting of up to 16 wordwide and/or bytewide GPRs, which are physically located within the on-chip RAM area. A Context Pointer (CP) register determines the base address of the active register bank to be accessed by the CPU at a time. The number of register banks is only restricted by the available internal RAM space. For easy parameter passing, a register bank may overlap others.

A system stack of up to 1024 words is provided as a storage for temporary data. The system stack is also located within the on-chip RAM area, and it is accessed by the CPU via the stack pointer (SP) register. Two separate SFRs, STKOV and STKUN, are implicitly compared against the stack pointer value upon each stack access for the detection of a stack overflow or underflow.

Hardware detection of the selected memory space is placed at the internal memory decoders and allows the user to specify any address directly or indirectly and obtain the desired data without using temporary registers or special instructions.

**For Special Function Registers** 1024 Bytes of the address space are reserved. The standard Special Function Register area (SFR) uses 512 bytes, while the Extended Special Function Register area (ESFR) uses the other 512 bytes. (E)SFRs are wordwide registers which are used for controlling and monitoring functions of the different on-chip units. Unused (E)SFR addresses are reserved for future members of the C165UTAH family with enhanced functionality.

### External Bus Interface

In order to meet the needs of designs where more memory is required than is provided on chip, up to 8 MBytes of external RAM and/or ROM can be connected to the microcontroller via its external bus interface. The integrated External Bus Controller (EBC) allows to access external memory and/or peripheral resources in a very flexible

**Note:** The upper 256 Bytes of SFR area, ESFR area and internal RAM are bit-addressable (see shaded blocks in **Figure 10**).

Code accesses are always made on even byte addresses. The highest possible code storage location in the internal RAM is either 00'FDFE$_H$ for single word instructions or 00'FDFC$_H$ for double word instructions. The respective location must contain a branch instruction (unconditional), because sequential boundary crossing from internal RAM to the SFR area is not supported and causes erroneous results.

Any word and byte data in the internal RAM can be accessed via indirect or long 16-bit addressing modes, if the selected DPP register points to data page 3. Any word data access is made on an even byte address. The highest possible word data storage location in the internal RAM is 00'FDFE$_H$. For PEC data transfers, the internal RAM can be accessed independent of the contents of the DPP registers via the PEC source and destination pointers.

The upper 256 Byte of the internal RAM (00'FD00$_H$ through 00'FDFF$_H$) and the GPRs of the current bank are provided for single bit storage, and thus they are bit addressable.

### System Stack

The system stack may be defined within the internal RAM. The size of the system stack is controlled by bitfield STKSZ in register SYSCON (see table below).

| <STKSZ> | Stack Size (Words) | Internal RAM Addresses (Words) |
|---|---|---|
| 0 0 0 $_B$ | 256 | 00'FBFE$_H$...00'FA00$_H$ (Default after Reset) |
| 0 0 1 $_B$ | 128 | 00'FBFE$_H$...00'FB00$_H$ |
| 0 1 0 $_B$ | 64 | 00'FBFE$_H$...00'FB80$_H$ |
| 0 1 1 $_B$ | 32 | 00'FBFE$_H$...00'FBC0$_H$ |
| 1 0 0 $_B$ | 512 | 00'FBFE$_H$...00'F800$_H$ |
| 1 0 1 $_B$ | --- | Reserved. Do not use this combination. |
| 1 1 0 $_B$ | --- | Reserved. Do not use this combination. |
| 1 1 1 $_B$ | 1024 | 00'FDFE$_H$...00'F600$_H$ (Note: No circular stack) |

For all system stack operations the on-chip RAM is accessed via the Stack Pointer (SP) register. The stack grows downward from higher towards lower RAM address locations. Only word accesses are supported to the system stack. A stack overflow (STKOV) and a stack underflow (STKUN) register are provided to control the lower and upper limits of the selected stack area. These two stack boundary registers can be used not only for protection against data destruction, but also allow to implement a circular stack with hardware supported system stack flushing and filling (except for option '111').

The technique of implementing this circular stack is described in chapter "System Programming".

• Jumps to  non-aligned double word instructions in the internal ROM space
• Testing Branch Conditions immediately after PSW writes

## 5.4 CPU Special Function Registers

The core CPU requires a set of Special Function Registers (SFRs) to maintain the system state information, to supply the ALU with register-addressable constants and to control system and bus configuration, multiply and divide ALU operations, code memory segmentation, data memory paging, and accesses to the General Purpose Registers and the System Stack.

The access mechanism for these SFRs in the CPU core is identical to the access mechanism for any other SFR. Since all SFRs can simply be controlled by means of any instruction, which is capable of addressing the SFR memory space, a lot of flexibility has been gained, without the need to create a set of system-specific instructions.

Note, however, that there are user access restrictions for some of the CPU core SFRs to ensure proper processor operations. The instruction pointer IP and code segment pointer CSP cannot be accessed directly at all. They can only be changed indirectly via branch instructions.

The PSW, SP, and MDC registers can be modified not only explicitly by the programmer, but also implicitly by the CPU during normal instruction processing. Note that any explicit write request (via software) to an SFR supersedes a simultaneous modification by hardware of the same register.

**Note:** Any write operation to a single byte of an SFR clears the non-addressed complementary byte within the specified SFR.
Non-implemented (reserved) SFR bits cannot be modified, and will always supply a read value of '0'.

### The System Configuration Register SYSCON

This bit-addressable register provides general system configuration and control functions. The reset value for register SYSCON depends on the state of the PORT0 pins during reset (see hardware effectable bits).

most significant bits are ignored. The respective physical GPR address calculation is identical to that for the short 4-bit GPR addresses. For single bit accesses on a GPR, the GPR's word address is calculated as just described, but the position of the bit within the word is specified by a separate additional 4-bit value.

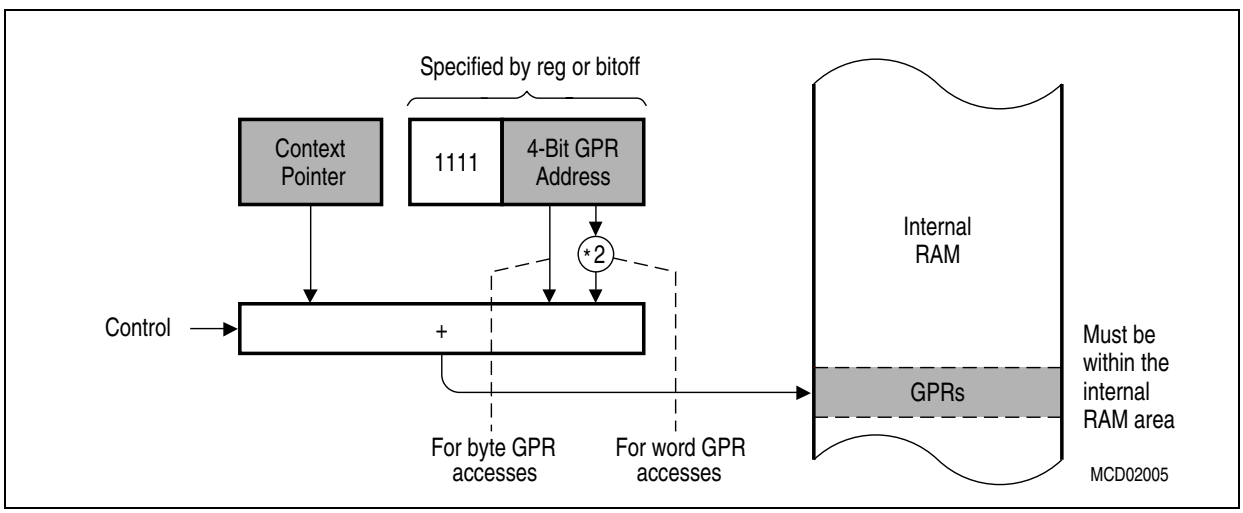


**Figure 19 Implicit CP Use by Short GPR Addressing Modes**

### Stack Pointer SP

This non-bit addressable register is used to point to the top of the internal system stack (TOS). The SP register is pre-decremented whenever data is to be pushed onto the stack, and it is post-incremented whenever data is to be popped from the stack. Thus, the system stack grows from higher toward lower memory locations.

Since the least significant bit of register SP is tied to '0' and bits 15 through 12 are tied to '1' by hardware, the SP register can only contain values from $F000_H$ to $FFFE_H$. This allows to access a physical stack within the internal RAM of the C165UTAH. A virtual stack (usually bigger) can be realized via software. This mechanism is supported by registers STKOV and STKUN (see respective descriptions below).

The SP register can be updated via any instruction, which is capable of modifying an SFR.

**Note:** Due to the internal instruction pipeline, a POP or RETURN instruction must not immediately follow an instruction updating the SP register.

# 6 DMA - External PEC (EPEC)

The EPEC provides fast and easy means to transfer single data between any memory location within the address space by using the XBUS. The advantages are reduced XBUS protocol handling and capability of addressing all system resources including internal RAM and SFR.

## 6.1 EPEC Functionality

The EPEC provides a DMA controller for the USB device core to provide fast and flexible data tranfer capability. The EPEC is implemented as a 16 channel controller with a 24-bit source pointer, a 24-bit destination pointer and a 10-bit Transmit Byte Length Counter with auto-increment support of two bytes (one word) per channel with Terminal Count (TC) indication (Interrupt pulse valid for one clock cycle). After TC is reached, the counter stops itself.

The EPEC is connected to the XBUS and to a proprietary 24-bit bus connected directly to the C166 CBC. The EPEC has the highest priority among other interrupts and PECs and does not participate in the interrupt priorization round. In case of an DPEC/EPEC collision, the DPEC will get priority and one instruction cycle later the EPEC is processed. The EPEC provides DMA like functionality by injecting a memory transfer instruction (mov [dest], [src]) into the decode stage of the pipeline and thus only needs one additional instruction cycle. Even in IDLE mode, the EPEC will be processed waking up the CPU for one instruction cycle and immediately going back to IDLE state.

## 6.2 EPEC Implementation

The EPEC control block is located in the CBC core with its main purpose to synchronize the external EPEC request to the internal T1-T4 states of the CPU and the priorization between DPEC and EPEC. It also drives the externally provided 24-bit source and destination pointer values on the internal memory address bus, thus controlling the whole timing with respect to the CPU.

The EPEC Block diagram is shown in **Figure 20** below.

The EPEC Transmit Byte Length registers (x) provide the 10-bit Transmit bytes length of the actual packet to be send to the USB endpoint#x and the related bit. Each EPEC channel can be cleared by SW, if necessary.

All USB source and destination pointers will be used in either receive or transmit direction, since the USB endpoint's direction of data is SW-configurable as IN, OUT or bidirectional (except endpoint 0) after USB device controller reset.

**EPEC_INT_REG Reset Value: $0000_H$**

**Table 21 EPEC_INT_REG Interrupt Register**

| Bit No. | Name | Function |
|---|---|---|
| 15:8 | RxTxSTART | Rx / Tx Start<br>Signal '1' indicates channel has started transfering data. |
| 7:0 | TXDONE_INTx (x=7..0) | TX packet transfer completed by EPEC<br>'1': transfer complete<br>'0': busy or idle |

The EPEC interrupt register indicates the end of an TX-packet transfer for an USB endpoint.

**EPEC_INTMSK_REG Reset Value: $0000_H$**

**Table 22 EPEC_INTMSK_REG Interrupt Register**

| Bit No. | Name | Function |
|---|---|---|
| 15:8 | RxTxSTARTMSK | Rx / Tx Start Mask<br>'1': masked<br>'0': not masked |
| 7:0 | TXDONE_INTMSKx (x=7..0) | Mask interrupt TX packet transfer completed by EPEC<br>'1': masked<br>'0': not masked |

The EPEC interrupt mask register masks out the end of an TX-packet transfer interrupt for an USB endpoint.

## 6.4 EPEC Transfer Example

The EPEC (external peripheral event controller, external in the sense that it is external to the CPU block) controls the transfer of data between the USB block and the external or internal RAM.

| Nr. | Source of Interrupt or PEC Service Request | Interrupt Name | Enable Flag | Vector Location | Trap Number | SFR hex Addr |
|---|---|---|---|---|---|---|
| irq(26) | UDC TX Done5 | UTD5INT | UTD5IE | $00'00C8_H$ | $32_H / 50_D$ | F164 |
| irq(27) | UDC TX Done4 | UTD4INT | UTD4IE | $00'00C4_H$ | $31_H / 49_D$ | F162 |
| irq(28) | UDC TX Done3 | UTD3INT | UTD3IE | $00'00C0_H$ | $30_H / 48_D$ | F160 |
| irq(29) | UDC TX Done2 | UTD2INT | UTD2IE | $00'005C_H$ | $17_H / 23_D$ | FF86 |
| irq(30) | UDC TX Done1 | UTD1INT | UTD1IE | $00'0058_H$ | $16_H / 22_D$ | FF84 |
| irq(31) | UDC TX Done0 | UTD0INT | UTD0IE | $00'0054_H$ | $15_H / 21_D$ | FF82 |
| irq(32) | UDC RX Done7 | URD7INT | URD7IE | $00'0050_H$ | $14_H / 20_D$ | FF80 |
| irq(33) | UDC RX Done6 | URD6INT | URD6IE | $00'004C_H$ | $13_H / 19_D$ | FF7E |
| irq(34) | UDC RX Done5 | URD5INT | URD5IE | $00'0048_H$ | $12_H / 18_D$ | FF7C |
| irq(35) | UDC RX Done4 | URD4INT | URD4IE | $00'0044_H$ | $11_H / 17_D$ | FF7A |
| irq(36) | UDC RX Done3 | URD3INT | URD3IE | $00'0040_H$ | $10_H / 16_D$ | FF78 |
| irq(37) | UDC RX Done2 | URD2INT | URD2IE | $00'0080_H$ | $20_H / 32_D$ | FF9C |
| irq(38) | UDC RX Done1 | URD1INT | URD1IE | $00'0084_H$ | $21_H / 33_D$ | FF9E |
| irq(39) | UDC RX Done0 | URD0INT | URD0IE | $00'00F4_H$ | $3D_H / 61_D$ | F17A |
| irq(40) | EPEC | EPECINT | EPECIE | $00'00F8_H$ | $3E_H / 62_D$ | F17C |
| irq(41) | reserved | | | $00'00A0_H$ | $28_H / 40_D$ | FF98 |
| irq(42) | IOM-2 I/O | IOMIOINT | IOMIOIE | $00'00A4_H$ | $29_H / 41_D$ | FF9A |
| irq(43) | IOM-2 Channel0 TX | IOMC0TINT | IOMC0TIE | $00'00FC_H$ | $3F_H / 63_D$ | F17E |
| irq(44) | IOM-2 Channel0 RX | IOMC0RINT | IOMC0RIE | $00'0120_H$ | $48_H / 72_D$ | F182 |
| irq(45) | IOM-2 Channel1 TX | IOMC1TINT | IOMC1TIE | $00'0124_H$ | $49_H / 73_D$ | F18A |
| irq(46) | IOM-2 Channel1 RX | IOMC1RINT | IOMC1RIE | $00'0128_H$ | $4A_H / 74_D$ | F192 |
| irq(47) | reserved | | | $00'012C_H$ | $4B_H / 75_D$ | F19A |
| firq(0) | Fast ext. Interrupt | EX0INT | EX0IE | $00'0060_H$ | $18_H / 24_D$ | FF88 |
| firq(1) | Fast ext. Interrupt | EX1INT | EX1IE | $00'0064_H$ | $19_H / 25_D$ | FF8A |
| firq(2) | Fast ext. Interrupt | EX2INT | EX2IE | $00'0068_H$ | $1A_H / 26_D$ | FF8C |
| firq(3) | Fast ext. Interrupt | EX3INT | EX3IE | $00'006C_H$ | $1B_H / 27_D$ | FF8E |
| firq(4) | Fast ext. Interrupt | EX4INT | EX4IE | $00'0070_H$ | $1C_H / 28_D$ | FF90 |
| firq(5) | Fast ext. Interrupt | EX5INT | EX5IE | $00'0074_H$ | $1D_H / 29_D$ | FF92 |

**Table 26 Software controlled Interrupt Classes (Example)**

| ILVL (Priority) | GLVL | | | | Interpretation |
|---|---|---|---|---|---|
| | 3 | 2 | 1 | 0 | |
| 15 | | | | | PEC service on up to 8 channels |
| 14 | | | | | |
| 13 | | | | | |
| 12 | X | X | X | X | Interrupt Class 1<br>5 sources on 2 levels |
| 11 | X | | | | |
| 10 | | | | | |
| 9 | | | | | |
| 8 | X | X | X | X | Interrupt Class 2<br>9 sources on 3 levels |
| 7 | X | X | X | X | |
| 6 | X | | | | |
| 5 | X | X | X | X | Interrupt Class 3<br>5 sources on 2 levels |
| 4 | X | | | | |
| 3 | | | | | |
| 2 | | | | | |
| 1 | | | | | |
| 0 | | | | | No service! |

**Table 30 Alternate Functions of Port 3**

| Port 3 Pin | Alternate Function | |
|---|---|---|
| P3.0 | --- | No pin assigned |
| P3.1 | --- | No pin assigned |
| P3.2 | --- | No pin assigned |
| P3.3 | T3OUT | Timer 3 Toggle Output |
| P3.4 | --- | No pin assigned! |
| P3.5 | T4IN | Timer 4 Count Input (T3EUD Input, T2EUD Input) |
| P3.6 | T3IN | Timer 3 Count Input |
| P3.7 | T2IN | Timer 2 Count Input |
| P3.8 | MRST | SSC Master Receive / Slave Transmit |
| P3.9 | MTSR | SSC Master Transmit / Slave Receive |
| P3.10 | TxD0 | ASC Transmit Data Output |
| P3.11 | RxD0 | ASC Receive Data Input |
| P3.12 | $\overline{BHE}$/$\overline{WRH}$ | Byte High Enable / Write High Output |
| P3.13 | SCLK | SSC Shift Clock Input/Output |
| P3.14 | --- | No pin assigned |
| P3.15 | CLKOUT | System Clock Output |

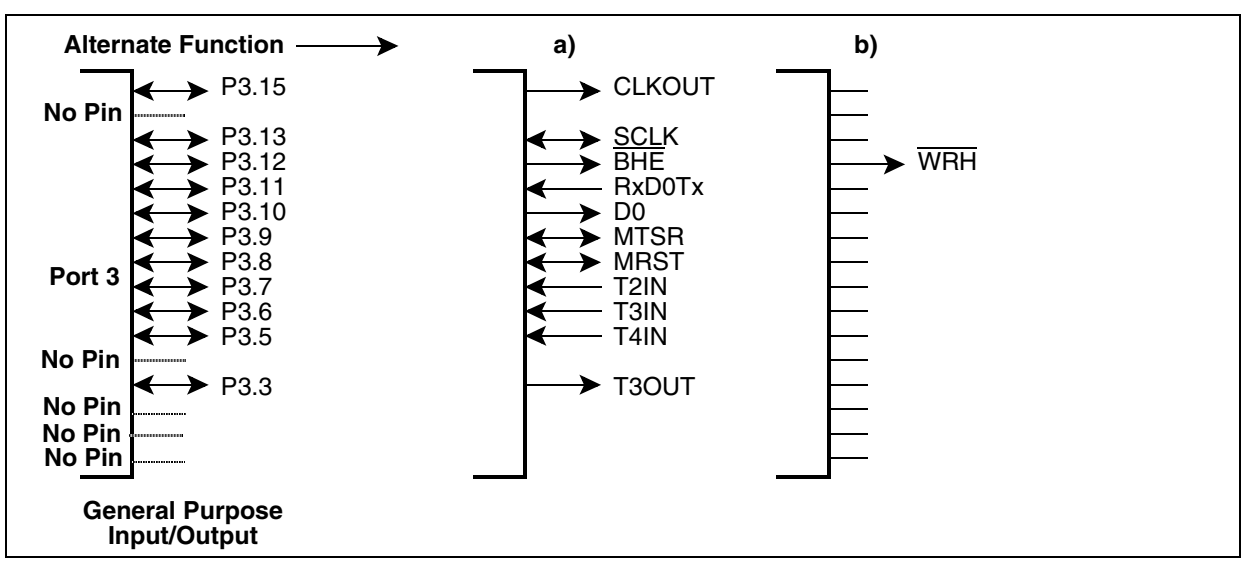


**Figure 35 Port 3 I/O and Alternate Functions**

The port structure of the Port 3 pins depends on their alternate function (see figures below).

When the on-chip peripheral associated with a Port 3 pin is configured to use the alternate input function, it reads the input latch, which represents the state of the pin, via

the two fixed dividers a fractional divider prescaler unit is available which allows to select prescaler divider ratios of n/512 with n=0-511. Therefore, the baudrate of ASC is determined by the module clock, the content of S0FDV, the reload value of S0BG and the operating mode (asynchronous or synchronous).

Register S0BG is the dual-function Baudrate Generator/Reload register. Reading BG returns the content of the timer BR_VALUE (bits 15...13 return zero), while writing to S0BG always updates the reload register (bits 15...13 are insiginificant).

An auto-reload of the timer with the content of the reload register is performed each time CON_BG is written to. However, if CON_R='0' at the time the write operation to BG is performed, the timer will not be reloaded until the first instruction cycle after CON_R='1'. For a clean baudrate initialization S0BG should only be written if CON_R='0'. If S0BG is written with CON_R='1', an unpredicted behaviour of the ASC may occur during running transmit or receive operations.

#### 12.1.7.1 Baudrates in Asynchronous Mode

For asynchronous operation, the baudrate generator provides a clock $f_{BRT}$ with 16 times the rate of the established baudrate. Every received bit is sampled at the 7th, 8th and 9th cycle of this clock. The clock divider circuitry, which generates the input clock for the 13-bit baudrate timer, is extended by a fracxtional divider circuitry, which allows the adjustment of more accurate baudrates and the extension of the baudrate range.

The baudrate of the baudrate generator depends on the following input clock, bits and register values :

- Input clock $f_{MOD}$
- Selection of the baudrate timer input clock $f_{DIV}$ by bits CON_FDE and CON_BRS
- If bit CON_FDE=1 (fractional divider) : value of register CON_FDV
- value of the 13-bit reload register S0BG

The output clock of the baudrate timer with the reload register is the sample clock in the asynchronous modes of the ASC. For baudrate calculations, this baudrate clock $f_{BR}$ is derived from the sample clock $f_{DIV}$ by a division by 16.

request flags have to be cleared for enabling the RTC module. An activated request flag sets directly the disabling status flag RTCDISS inside the RTCCLC register and disables the clock within the Clock Gating Module. Since the RTCCLC register is clocked with the bus clock, the disabling status of the RTC can be read and the RTC can be enabled again.

The following figure shows the disabling mechanism of the RTC module:

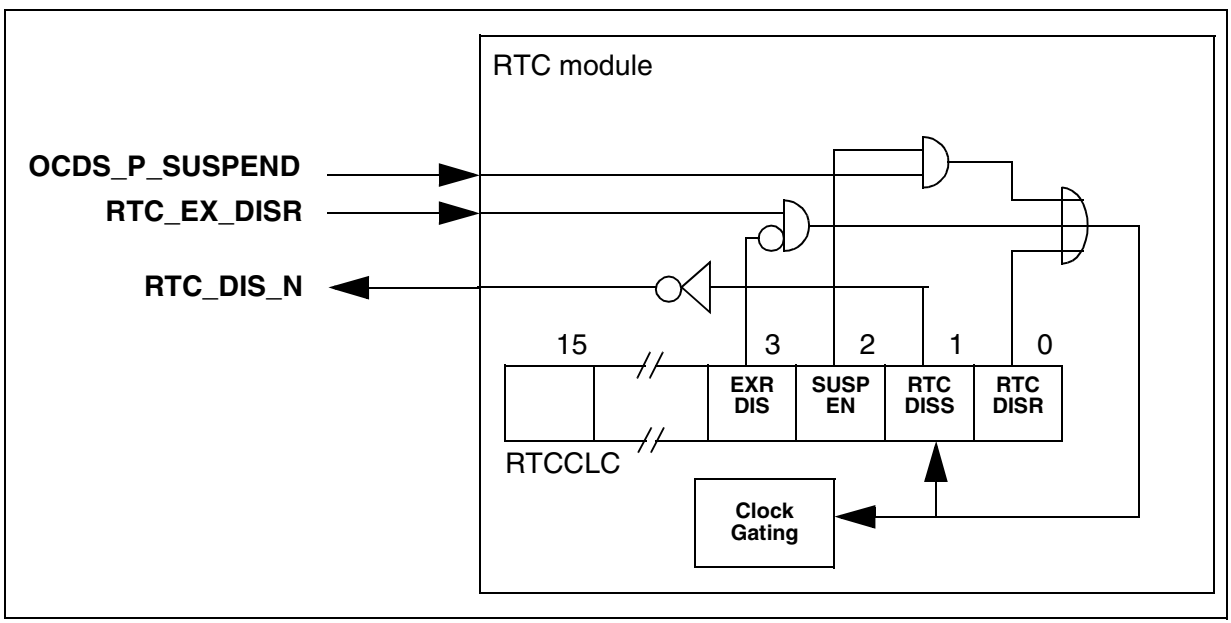


**Figure 96 Disabling Mechanism of the RTC**

In disable state no write access to RTC registers is possible. The only exeption is the RTCCLC register.

## 13.2.12 Register Definition of RTC module

The following table shows the register addresses map:

**Table 72 Address Map Overview**

| SFR Address | b/p | Register Name |
|---|---|---|
| $F0C8_H$ | | RTCCLC |
| $F1CC_H$ | b | RTCCON |
| $F0D0_H$ | | T14REL |
| $F0D2_H$ | | T14 |
| $F0D4_H$ | | RTCL |
| $F0D6_H$ | | RTCH |

**Table 75 USBD Register Set**

| $00EE00_H$ + ... | Register | Function |
|---|---|---|
| $48_H$ | USBD_TXEOD3 | EPEC/SW End-of-packet indication for USBD. |
| $4A_H$ | USBD_RXRR3 | USB Receive FIFO data register |
| $4C_H$ | USBD_RX_BYTECNT3 | USB receive packet length in bytes |
| $4E_H$ | USBD_TXWR4 | USB Transmit FIFO data register |
| $50_H$ | USBD_TXEOD4 | EPEC/SW End-of-packet indication for USBD. |
| $52_H$ | USBD_RXRR4 | USB Receive FIFO data register |
| $54_H$ | USBD_RX_BYTECNT4 | USB receive packet length in bytes |
| $56_H$ | USBD_TXWR5 | USB Transmit FIFO data register |
| $58_H$ | USBD_TXEOD5 | EPEC/SW End-of-packet indication for USBD. |
| $5A_H$ | USBD_RXRR5 | USB Receive FIFO data register |
| $5C_H$ | USBD_RX_BYTECNT5 | USB receive packet length in bytes |
| $5E_H$ | USBD_TXWR6 | USB Transmit FIFO data register |
| $60_H$ | USBD_TXEOD6 | EPEC/SW End-of-packet indication for USBD. |
| $62_H$ | USBD_RXRR6 | USB Receive FIFO data register |
| $64_H$ | USBD_RX_BYTECNT6 | USB receive packet length in bytes |
| $66_H$ | USBD_TXWR7 | USB Transmit FIFO data register |
| $68_H$ | USBD_TXEOD7 | EPEC/SW End-of-packet indication for USBD. |
| $6A_H$ | USBD_RXRR7 | USB Receive FIFO data register |
| $6C_H$ | USBD_RX_BYTECNT7 | USB receive packet length in bytes |
| $6E_H$ | USBD_CFGVAL | Current Configuration & Alternate Setting selected by Host |
| $70_H$ | USBC_CMD_RESET | USB Block Reset |

The detailed register description is shown below.

**Table 94 Reception of MONITOR Data**

| Control/ Status Bit | Register | Bit | Function |
|---|---|---|---|
| Control | MOCR | MRC | MR Bit Control |
| | | MIE | MER Interrupt Enable |
| | | MRE | MDR Interrupt Enable |
| Status | MOSR | MDR | Data Received Interrupt |
| | | MER | End of Reception Interrupt |

## Transmit FIFO

**Address:** **s_adr_x + $02_H$**
**Name:** **TFIFO_x**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA | | | | | | | | | | | | | | | |

| Field | Bits | Type | Value | Description |
|---|---|---|---|---|
| Data | 15:0 | W | $0000_H$ | A write access provides data to the TFIFO. Up to ten bytes of transmit data can be written to the TFIFO following an ISTAH.XPR interrupt.<br>Data can be written either word wide or byte wide (with the valid byte at Low position) |

## Interrupt Status Register

**Address:** **s_adr_x + $04_H$**
**Name:** **ISTAH_x**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RME | RPF | RFO | FFO | RESERVED | | | | XPR | XMR | XDU | XDOV | RESERVED | | | |

| Field | Bits | Type | Value | Description |
|---|---|---|---|---|
| RME | 15 | R | 0 | **Receive Message End**<br>The end of a frame has been received. The message length and additional information may be obtained from RBCH, RBCL and the STAR register. |
| RPF | 14 | R | 0 | **Receive Pool Full**<br>A data word/byte has been received and is available in the RFIFO. |

during Slow Down mode, automatically the standard Idle mode is selected as configured with SYSCON3 register.

The Sleep mode is controlled by bitfield SLEEPCON within register SYSCON1.

**SYSCON1 ($F1DC_H$ / $EE_H$)** **ESFR-b** **Reset Value: $0000_H$**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | SLEEPCON | |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | rw | rw |

**Note:** SYSCON1 is write protected after the execution of EINIT unless it is released via the unlock sequence.

General description of SYSCON1 bits:

| Bit | Function |
|---|---|
| **SLEEPCON** | **SLEEP Mode Configuration**<br>'0 0': normal IDLE mode<br>'0 1': SLEEP mode with running RTC<br>'1 0': reserved<br>'1 1': SLEEP mode with stopped RTC and stopped OSC |

Before entering Sleep mode with the IDLE instruction, the continuation of instruction processing **after** termination of Sleep mode must be prepared as known from standard Idle mode. For wakeup with interrupt, four general possibilities of continuation can be selected, which are controlled (prepared) as follows:

- **Continuation with first instruction after the IDLE instruction** will be enabled if
  - interrupts are globally disabled with the Interrupt Enable bit in PSW, **or**
  - the interrupt is enabled by global (PSW) and by individual (interrupt control register) enable bit, but the current CPU priority level (in PSW) of IDLE instruction is higher than the interrupt level.
- **Continuation with first instruction of dedicated interrupt service routine** will be selected if
  the interrupt is enabled by global (in PSW) and by individual (interrupt control register) enable bit, and the CPU priority level of IDLE instruction is lower than the interrupt level, thus the enabled interrupt has highest priority. Additionally, PEC Transfer for this interrupt is not enabled. The continuation with the dedicated service routine is **always** performed in case of NMI hardware traps, independently of any enable bit or CPU priority level.
- **Execution of one PEC Transfer and resumption of Sleep mode** will be selected if the interrupt is enabled by global (in PSW) and by individual (interrupt control register) enable bit, and the CPU priority level of IDLE instruction is lower than the interrupt

## 21.5 Peripheral Management Module

This module especially serves for power management support, controlling dynamically the operation and thus the power consumption of the different peripherals on PD Bus and XBUS. In each situation (eg. several system operating modes, standby, etc.) only those peripherals may be kept running which are required for the respective functionality. All others can be switched off. It also allows the operation control of whole groups of peripherals.

Peripheral's operation is disabled or enabled by controlling the specific clock input. This function also is supported in idle and/or slow down mode.

The Real Time Clock (RTC) may be fed by a separate clock driver, so it can be kept running even in power down mode.

While a peripheral is disabled its output pins remain in the state they had at the time of disabling.

**Note:** In contrast to the peripheral management of Infineon's 16x family the registers of a **disabled** module are not accessable. Only the clock control register of the platform peripheral is accessable. Note, the register access is not compatible to the C167CS.

The user gets access to the flexible operation control of peripherals via the SYSCON3 register. This register is defined as follows:

**SYSCON3 (F1D4$_H$ / EA$_H$)** **ESFR-b** **Reset Value:0000$_H$**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GRP DIS | re-served | PLL DIS | USBTDIS | | reserved | | PER DIS8 | PER DIS7 | PER DIS6 | reserved | | PER DIS3 | PER DIS2 | PER DIS1 | PER DIS0 |
| rw | - | rw | rw | | - | - | rw | rw | rw | - | - | rw | rw | rw | rw |

.

| Bit | Function |
|---|---|
| **PERDISx** | **Peripheral Disable Flag 0 - 14**<br>'0': Module is enabled; the peripheral is supplied with the clock signal<br>'1': Module is disabled; the clock input of peripheral is disabled |
| **GRPDIS** | **Peripheral Group Disable Flag (PD-Bus and X-Bus Peripherals)**<br>'0': Peripheral clock driver for peripheral group is enabled<br>'1': Peripheral clock driver for peripheral group is disabled |

# 23 Register Set

This section summarizes all registers, which are implemented in the C165UTAH and explains the description format which is used in the chapters describing the function and layout of the SFRs.
For easy reference the registers are ordered according to two different keys (except for GPRs):

- Ordered by address, to check which register a given address references,
- Ordered by register name, to find the location of a specific register.

## 23.1 Register Description Format

In the respective chapters the function and the layout of the SFRs is described in a specific format which provides a number of details about the described special function register. The example below shows how to interpret these details.

A word register looks like this:

***REG_NAME* (*A16*$_H$ / *A8*$_H$)** **E**/SFR **Reset Value: * * * *$_H$**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **res.** | **res.** | **res.** | **res.** | **res.** | **write only** | **hw bit** | **read only** | **std bit** | **hw bit** | **bitfield** | | | | **bitfield** | |
| - | - | - | - | - | w | rw | r | rw | rw | rw | | | | rw | |

| Bit | Function |
|---|---|
| ***bit(field)name*** | ***Explanation of bit(field)name***<br>*Description of the functions controlled by this bit(field).* |

A byte register looks like this:

***REG_NAME* (*A16*$_H$ / *A8*$_H$)** **E**/SFR **Reset Value: - - * *$_H$**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | **std bit** | **hw bit** | **bitfield** | | | | **bitfield** | |
| - | - | - | - | - | - | - | - | rw | rw | rw | | | | rw | |

**Elements:**

| | |
|---|---|
| REG_NAME | Name of this register |
| A16 / A8 | Long 16-bit address / Short 8-bit address |
| SFR/**ESFR**/**XReg** | Register space (SFR, ESFR or External/XBUS Register) |
| (* *) * * | Register contents after reset<br>**0/1**: defined value, '**X**': undefined, '**U**': unchanged (undefined ('**X**') after power up) |
| hwbit | Bits that are set/cleared by hardware are marked with a shaded access box |

| Physical Addr | Register Name | Type | 8-bit Addr | Description | Reset Value |
|---|---|---|---|---|---|
| $FE76_H$ | P1HPHEN | SFR | $3B_H$ | Port 1 High Pin Hold Enable Register | $0000_H$ |
| $FE78_H$ | P2PUDSEL | SFR | $3C_H$ | Port 2 Pull-Up/Down Select Register | $0000_H$ |
| $FE7A_H$ | P2PUDEN | SFR | $3D_H$ | Port 2 Pull Switch On/Off Register | $0000_H$ |
| $FE7C_H$ | P2PHEN | SFR | $3E_H$ | Port 2 Pin Hold Enable Register | $0000_H$ |
| $FE7E_H$ | P3PUDSEL | SFR | $3F_H$ | Port 3 Pull-Up/Down Select Register | $0000_H$ |
| $FE80_H$ | P3PUDEN | SFR | $40_H$ | Port 3 Pull Switch On/Off Register | $0000_H$ |
| $FE82_H$ | P3PHEN | SFR | $41_H$ | Port 3 Pin Hold Enable Register | $0000_H$ |
| $FE84_H$ | P4PUDSEL | SFR | $42_H$ | Port 4 Pull-Up/Down Select Register | $0000_H$ |
| $FE86_H$ | P4PUDEN | SFR | $43_H$ | Port 4 Pull Switch On/Off Register | $0000_H$ |
| $FE88_H$ | P4PHEN | SFR | $44_H$ | Port 4 Pin Hold Enable Register | $0000_H$ |
| $FE90_H$ | P6PUDSEL | SFR | $48_H$ | Port 6 Pull-Up/Down Select Register | $0000_H$ |
| $FE92_H$ | P6PUDEN | SFR | $49_H$ | Port 6 Pull Switch On/Off Register | $0000_H$ |
| $FE94_H$ | P6PHEN | SFR | $4A_H$ | Port 6 Pin Hold Enable Register | $0000_H$ |
| $FE96_H$ | P7PUDSEL | SFR | $4B_H$ | Port 7 Pull-Up/Down Select Register | $0000_H$ |
| $FE98_H$ | P7PUDEN | SFR | $4C_H$ | Port 7 Pull Switch On/Off Register | $0000_H$ |
| $FE9A_H$ | P7PHEN | SFR | $4D_H$ | Port 7 Pin Hold Enable Register | $0000_H$ |
| $FEAA_H$ | S0PMW | SFR | $55_H$ | ASC IrDA PMW Control Register | $0000_H$ |
| $FEAE_H$ | WDT | SFR | $57_H$ | Watchdog Timer Register (RO) | $0000_H$ |
| $FEB0_H$ | S0TBUF | SFR | $58_H$ | Serial Channel 0 Transmit Buffer Register (WO) | $0000_H$ |
| $FEB2_H$ | S0RBUF | SFR | $59_H$ | Serial Channel 0 Receive Buffer Register (RO) | $xxxx_H$ |
| $FEB4_H$ | S0BG | SFR | $5A_H$ | Serial Channel 0 Baud Rate Generator Reload Register | $0000_H$ |
| $FEB6_H$ | S0FDV | SFR | $5B_H$ | ASC Fractional Divide Register | $0000_H$ |
| $FEC0_H$ | PECC0 | SFR | $60_H$ | PEC Channel 0 Control Register | $0000_H$ |
| $FEC2_H$ | PECC1 | SFR | $61_H$ | PEC Channel 1 Control Register | $0000_H$ |
| $FEC4_H$ | PECC2 | SFR | $62_H$ | PEC Channel 2 Control Register | $0000_H$ |
| $FEC6_H$ | PECC3 | SFR | $63_H$ | PEC Channel 3 Control Register | $0000_H$ |
| $FEC8_H$ | PECC4 | SFR | $64_H$ | PEC Channel 4 Control Register | $0000_H$ |
| $FECA_H$ | PECC5 | SFR | $65_H$ | PEC Channel 5 Control Register | $0000_H$ |
| $FECC_H$ | PECC6 | SFR | $66_H$ | PEC Channel 6 Control Register | $0000_H$ |
| $FECE_H$ | PECC7 | SFR | $67_H$ | PEC Channel 7 Control Register | $0000_H$ |
| $FED0_H$ | PECSN0 | SFR | $68_H$ | PEC Segment No Register | |
| $FED2_H$ | PECSN1 | SFR | $69_H$ | PEC Segment No Register | |
| $FED4_H$ | PECSN2 | SFR | $6A_H$ | PEC Segment No Register | |
| $FED6_H$ | PECSN3 | SFR | $6B_H$ | PEC Segment No Register | |
| $FED8_H$ | PECSN4 | SFR | $6C_H$ | PEC Segment No Register | |