# Infineon Technologies - C161CSLFCAFXQMA1 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 | 25MHz |
| Connectivity | CANbus, EBI/EMI, I²C, SPI, UART/USART |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 93 |
| Program Memory Size | - |
| Program Memory Type | ROMless |
| EEPROM Size | - |
| RAM Size | 10K x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.5V ~ 5.5V |
| Data Converters | A/D 12x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 128-LQFP |
| Supplier Device Package | PG-TQFP-128-2 |
| Purchase URL | https://www.e-xfl.com/product-detail/infineon-technologies/c161cslfcafxqma1 |

Data Sheet, V3.0, Jan. 2001

# C161CS-32R/-L
# C161JC-32R/-L
# C161JI-32R/-L

## 16-Bit Single-Chip Microcontroller

Microcontrollers

Infineon technologies

Never stop thinking.

**Table 2 Pin Definitions and Functions** (cont'd)

| Symbol | Pin No. | Input Outp. | Function | | |
|---|---|---|---|---|---|
| **P5** | | I | Port 5 is a 12-bit input-only port with Schmitt-Trigger char. The pins of Port 5 also serve as analog input channels for the A/D converter, or they serve as timer inputs: | | |
| P5.0 | 27 | I | AN0 | | |
| P5.1 | 28 | I | AN1 | | |
| P5.2 | 29 | I | AN2 | | |
| P5.3 | 30 | I | AN3 | | |
| P5.4 | 31 | I | AN4 | | |
| P5.5 | 32 | I | AN5 | | |
| P5.6 | 33 | I | AN6 | | |
| P5.7 | 34 | I | AN7 | | |
| P5.12 | 37 | I | AN12, | T6IN | GPT2 Timer T6 Count Inp. |
| P5.13 | 38 | I | AN13, | T5IN | GPT2 Timer T5 Count Inp. |
| P5.14 | 39 | I | AN14, | T4EUD | GPT1 Timer T4 Ext. Up/Down Ctrl. Inp. |
| P5.15 | 40 | I | AN15, | T2EUD | GPT1 Timer T5 Ext. Up/Down Ctrl. Inp. |

**Table 2 Pin Definitions and Functions** (cont'd)

| Symbol | Pin No. | Input Outp. | Function |
|---|---|---|---|
| **PORT1** | | IO | PORT1 consists of the two 8-bit bidirectional I/O ports P1L and P1H. It is bit-wise programmable for input or output via direction bits. For a pin configured as input, the output driver is put into high-impedance state. PORT1 is used as the 16-bit address bus (A) in demultiplexed bus modes and also after switching from a demultiplexed bus mode to a multiplexed bus mode. |
| P1L.0-7 | 103-110 | | |
| P1H.0-7 | 113-120 | | |
| | | | The following PORT1 pins also serve for alternate functions: |
| P1H.4 | 117 | I/O | CC24IO CAPCOM2: CC24 Capture Inp./Compare Outp. |
| P1H.5 | 118 | I/O | CC25IO CAPCOM2: CC25 Capture Inp./Compare Outp. |
| P1H.6 | 119 | I/O | CC26IO CAPCOM2: CC26 Capture Inp./Compare Outp. |
| P1H.7 | 120 | I/O | CC27IO CAPCOM2: CC27 Capture Inp./Compare Outp. |
| XTAL2 | 123 | O | XTAL2: Output of the oscillator amplifier circuit. |
| XTAL1 | 124 | I | XTAL1: Input to the oscillator amplifier and input to the internal clock generator<br>To clock the device from an external source, drive XTAL1, while leaving XTAL2 unconnected. Minimum and maximum high/low and rise/fall times specified in the AC Characteristics must be observed. |
| XTAL3 | 126 | I | XTAL3: Input to the 32-kHz oscillator amplifier and input to the internal clock generator |
| XTAL4 | 127 | O | XTAL4: Output of the oscillator amplifier circuit.<br>To clock the device from an external source, drive XTAL3, while leaving XTAL4 unconnected. Minimum and maximum high/low and rise/fall times specified in the AC Characteristics must be observed. |

### Interrupt System

With an interrupt response time within a range from just 5 to 12 CPU clocks (in case of internal program execution), the C161CS/JC/JI is capable of reacting very fast to the occurrence of non-deterministic events.

The architecture of the C161CS/JC/JI supports several mechanisms for fast and flexible response to service requests that can be generated from various sources internal or external to the microcontroller. Any of these interrupt requests can be programmed to being serviced by the Interrupt Controller or by the Peripheral Event Controller (PEC).

In contrast to a standard interrupt service where the current program execution is suspended and a branch to the interrupt vector table is performed, just one cycle is ‘stolen’ from the current CPU activity to perform a PEC service. A PEC service implies a single byte or word data transfer between any two memory locations with an additional increment of either the PEC source or the destination pointer. An individual PEC transfer counter is implicity decremented for each PEC service except when performing in the continuous transfer mode. When this counter reaches zero, a standard interrupt is performed to the corresponding source related vector location. PEC services are very well suited, for example, for supporting the transmission or reception of blocks of data. The C161CS/JC/JI has 8 PEC channels each of which offers such fast interrupt-driven data transfer capabilities.

A separate control register which contains an interrupt request flag, an interrupt enable flag and an interrupt priority bitfield exists for each of the possible interrupt sources. Via its related register, each source can be programmed to one of sixteen interrupt priority levels. Once having been accepted by the CPU, an interrupt service can only be interrupted by a higher prioritized service request. For the standard interrupt processing, each of the possible interrupt sources has a dedicated vector location.

Fast external interrupt inputs are provided to service external interrupts with high precision requirements. These fast interrupt inputs feature programmable edge detection (rising edge, falling edge or both edges).

Software interrupts are supported by means of the ‘TRAP’ instruction in combination with an individual trap (interrupt) number.

**Table 3** shows all of the possible C161CS/JC/JI interrupt sources and the corresponding hardware-related interrupt flags, vectors, vector locations and trap (interrupt) numbers.

*Note: Interrupt nodes which are not used by associated peripherals, may be used to generate software controlled interrupt requests by setting the respective interrupt request bit (xIR).*

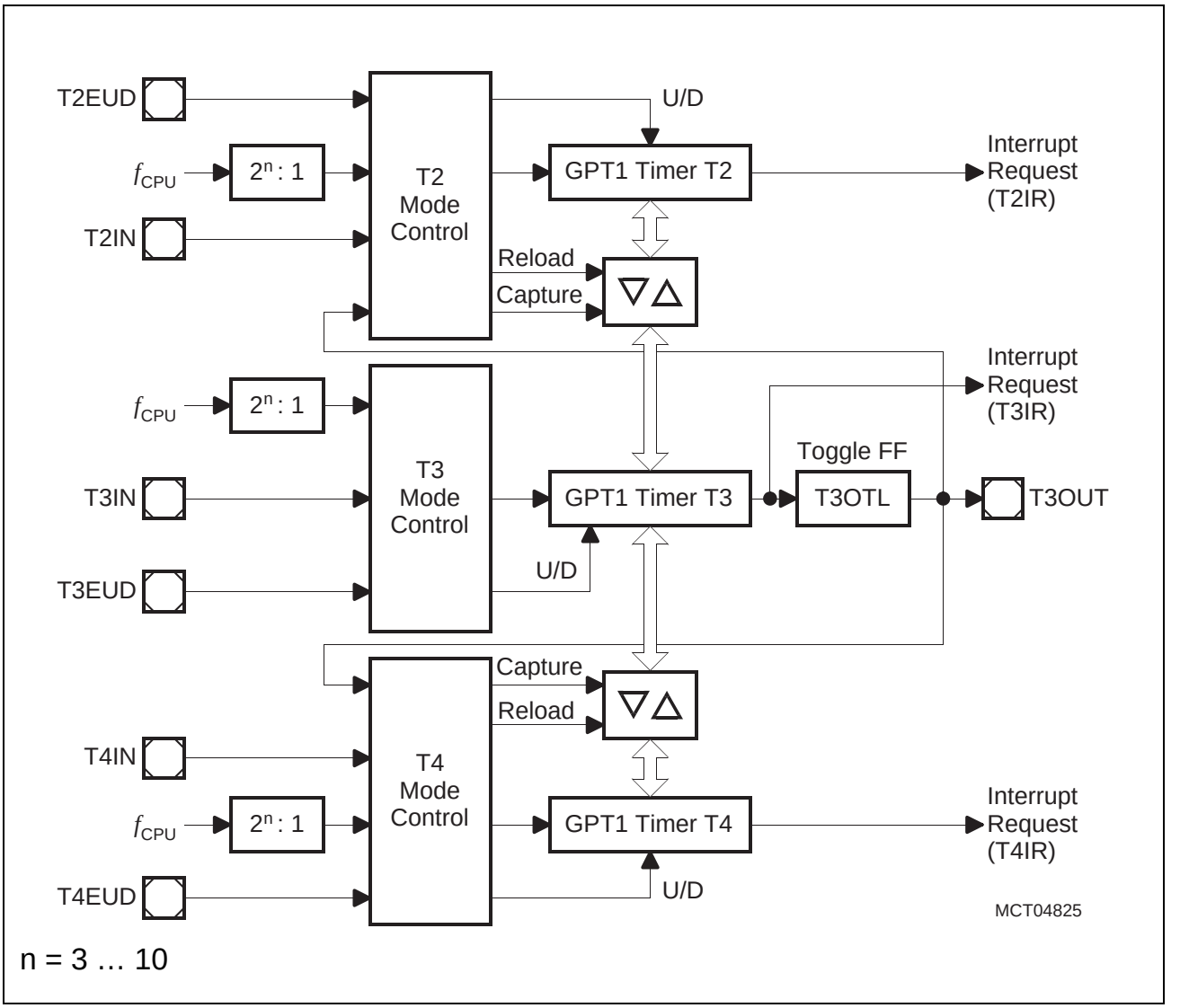


**Figure 6 Block Diagram of GPT1**

With its maximum resolution of 8 TCL, the **GPT2 module** provides precise event control and time measurement. It includes two timers (T5, T6) and a capture/reload register (CAPREL). Both timers can be clocked with an input clock which is derived from the CPU clock via a programmable prescaler or with external signals. The count direction (up/down) for each timer is programmable by software or may additionally be altered dynamically by an external signal on a port pin (TxEUD). Concatenation of the timers is supported via the output toggle latch (T6OTL) of timer T6, which changes its state on each timer overflow/underflow.

The state of this latch may be used to clock timer T5, and/or it may be output on pin T6OUT. The overflows/underflows of timer T6 can additionally be used to clock the CAPCOM timers T0 or T1, and to cause a reload from the CAPREL register. The CAPREL register may capture the contents of timer T5 based on an external signal transition on the corresponding port pin (CAPIN), and timer T5 may optionally be cleared

### CAN-Modules

The integrated CAN-Modules handle the completely autonomous transmission and reception of CAN frames in accordance with the CAN specification V2.0 part B (active), i.e. the on-chip CAN-Modules can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

The modules provide Full CAN functionality on up to 15 message objects each. Message object 15 may be configured for Basic CAN functionality. Both modes provide separate masks for acceptance filtering which allows to accept a number of identifiers in Full CAN mode and also allows to disregard a number of identifiers in Basic CAN mode. All message objects can be updated independent from the other objects and are equipped for the maximum message length of 8 bytes.

The bit timing is derived from the XCLK and is programmable up to a data rate of 1 MBaud. Each CAN-Module uses two pins of Port 4 or Port 8 to interface to an external bus transceiver. The interface pins are assigned via software.

**Module CAN2** (C161CS only) is identical with the first one, except that it uses a separate address area and a separate interrupt node.

The two CAN modules can be internally coupled by assigning their interface pins to the same two port pins, or they can interface to separate CAN buses.

*Note: When one or both of the on-chip CAN Modules are used with the interface lines assigned to Port 4, the interface lines override the segment address lines and the segment address output on Port 4 is therefore limited to 6/4 bits i.e. address lines A21/A19 … A16. $\overline{CS}$ lines can be used to increase the total amount of addressable external memory.*

### IIC Module

The integrated IIC Bus Module handles the transmission and reception of frames over the two-line IIC bus in accordance with the IIC Bus specification. The on-chip IIC Module can receive and transmit data using 7-bit or 10-bit addressing and it can operate in slave mode, in master mode or in multi-master mode.

Several physical interfaces (port pins) can be established under software control. Data can be transferred at speeds up to 400 kbit/sec.

Two interrupt nodes dedicated to the IIC module allow efficient interrupt service and also support operation via PEC transfers.

*Note: The port pins associated with the IIC interfaces feature open drain drivers only, as required by the IIC specification.*

### Watchdog Timer

The Watchdog Timer represents one of the fail-safe mechanisms which have been implemented to prevent the controller from malfunctioning for longer periods of time.

The Watchdog Timer is always enabled after a reset of the chip, and can only be disabled in the time interval until the EINIT (end of initialization) instruction has been executed. Thus, the chip's start-up procedure is always monitored. The software has to be designed to service the Watchdog Timer before it overflows. If, due to hardware or software related failures, the software fails to do so, the Watchdog Timer overflows and generates an internal hardware reset and pulls the $\overline{\text{RSTOUT}}$ pin low in order to allow external hardware components to be reset.

The Watchdog Timer is a 16-bit timer, clocked with the system clock divided by 2/4/128/256. The high byte of the Watchdog Timer register can be set to a prespecified reload value (stored in WDTREL) in order to allow further variation of the monitored time interval. Each time it is serviced by the application software, the high byte of the Watchdog Timer is reloaded. Thus, time intervals between 20 µs and 671 ms can be monitored (@ 25 MHz).
The default Watchdog Timer interval after reset is 5.24 ms (@ 25 MHz).

### Oscillator Watchdog

The Oscillator Watchdog (OWD) monitors the clock signal generated by the on-chip oscillator (either with a crystal or via external clock drive). For this operation the PLL provides a clock signal which is used to supervise transitions on the oscillator clock. This PLL clock is independent from the XTAL1 clock. When the expected oscillator clock transitions are missing the OWD activates the PLL Unlock / OWD interrupt node and supplies the CPU with the PLL clock signal. Under these circumstances the PLL will oscillate with its basic frequency.

In direct drive mode the PLL base frequency is used directly ($f_{CPU}$ = 2 … 5 MHz).
In prescaler mode the PLL base frequency is divided by 2 ($f_{CPU}$ = 1 … 2.5 MHz).

*Note: The CPU clock source is only switched back to the oscillator clock after a hardware reset.*

**The oscillator watchdog can be disabled** by setting bit OWDDIS in register SYSCON. In this case (OWDDIS = '1') the PLL remains idle and provides no clock signal, while the CPU clock signal is derived directly from the oscillator clock or via prescaler or SDD. Also no interrupt request will be generated in case of a missing oscillator clock.

*Note: At the end of an external reset ($\overline{EA}$ = '0') bit OWDDIS reflects the inverted level of pin $\overline{RD}$ at that time. Thus the oscillator watchdog may also be disabled via hardware by (externally) pulling the $\overline{RD}$ line low upon a reset, similar to the standard reset configuration via PORT0. At the end of an internal reset ($\overline{EA}$ = '1') bit OWDDIS is cleared.*

**Table 6 Instruction Set Summary** (cont'd)

| Mnemonic | Description | Bytes |
|---|---|---|
| MOV(B) | Move word (byte) data | 2 / 4 |
| MOVBS | Move byte operand to word operand with sign extension | 2 / 4 |
| MOVBZ | Move byte operand to word operand. with zero extension | 2 / 4 |
| JMPA, JMPI, JMPR | Jump absolute/indirect/relative if condition is met | 4 |
| JMPS | Jump absolute to a code segment | 4 |
| J(N)B | Jump relative if direct bit is (not) set | 4 |
| JBC | Jump relative and clear bit if direct bit is set | 4 |
| JNBS | Jump relative and set bit if direct bit is not set | 4 |
| CALLA, CALLI, CALLR | Call absolute/indirect/relative subroutine if condition is met | 4 |
| CALLS | Call absolute subroutine in any code segment | 4 |
| PCALL | Push direct word register onto system stack and call absolute subroutine | 4 |
| TRAP | Call interrupt service routine via immediate trap number | 2 |
| PUSH, POP | Push/pop direct word register onto/from system stack | 2 |
| SCXT | Push direct word register onto system stack and update register with word operand | 4 |
| RET | Return from intra-segment subroutine | 2 |
| RETS | Return from inter-segment subroutine | 2 |
| RETP | Return from intra-segment subroutine and pop direct word register from system stack | 2 |
| RETI | Return from interrupt service subroutine | 2 |
| SRST | Software Reset | 4 |
| IDLE | Enter Idle Mode | 4 |
| PWRDN | Enter Power Down Mode (supposes $\overline{\text{NMI}}$-pin being low) | 4 |
| SRVWDT | Service Watchdog Timer | 4 |
| DISWDT | Disable Watchdog Timer | 4 |
| EINIT | Signify End-of-Initialization on RSTOUT-pin | 4 |
| ATOMIC | Begin ATOMIC sequence | 2 |
| EXTR | Begin EXTended Register sequence | 2 |
| EXTP(R) | Begin EXTended Page (and Register) sequence | 2 / 4 |
| EXTS(R) | Begin EXTended Segment (and Register) sequence | 2 / 4 |
| NOP | Null operation | 2 |

## Special Function Registers Overview

Table 7 lists all SFRs which are implemented in the C161CS/JC/JI in alphabetical order. **Bit-addressable** SFRs are marked with the letter "**b**" in column "Name". SFRs within the **Extended SFR-Space** (ESFRs) are marked with the letter "**E**" in column "Physical Address". Registers within on-chip X-peripherals are marked with the letter "**X**" in column "Physical Address".

An SFR can be specified via its individual mnemonic name. Depending on the selected addressing mode, an SFR can be accessed via its physical address (using the Data Page Pointers), or via its short 8-bit address (without using the Data Page Pointers).

*Note: Registers within device specific interface modules (CAN, SDLM) are only present in the corresponding device, of course.*

**Table 7 C161CS/JC/JI Registers, Ordered by Name**

| Name | Physical Address | 8-Bit Addr. | Description | Reset Value |
|---|---|---|---|---|
| **ADCIC b** | $FF98_H$ | $CC_H$ | A/D Converter End of Conversion Interrupt Control Register | $0000_H$ |
| **ADCON b** | $FFA0_H$ | $D0_H$ | A/D Converter Control Register | $0000_H$ |
| **ADDAT** | $FEA0_H$ | $50_H$ | A/D Converter Result Register | $0000_H$ |
| **ADDAT2** | $F0A0_H$ **E** | $50_H$ | A/D Converter 2 Result Register | $0000_H$ |
| **ADDRSEL1** | $FE18_H$ | $0C_H$ | Address Select Register 1 | $0000_H$ |
| **ADDRSEL2** | $FE1A_H$ | $0D_H$ | Address Select Register 2 | $0000_H$ |
| **ADDRSEL3** | $FE1C_H$ | $0E_H$ | Address Select Register 3 | $0000_H$ |
| **ADDRSEL4** | $FE1E_H$ | $0F_H$ | Address Select Register 4 | $0000_H$ |
| **ADEIC b** | $FF9A_H$ | $CD_H$ | A/D Converter Overrun Error Interrupt Control Register | $0000_H$ |
| **BUFFCON** | $EB24_H$ **X** | --- | SDLM Buffer Control Register | $0000_H$ |
| **BUFFSTAT** | $EB1C_H$ **X** | --- | SDLM Buffer Status Register | $0000_H$ |
| **BUSCON0 b** | $FF0C_H$ | $86_H$ | Bus Configuration Register 0 | $0000_H$ |
| **BUSCON1 b** | $FF14_H$ | $8A_H$ | Bus Configuration Register 1 | $0000_H$ |
| **BUSCON2 b** | $FF16_H$ | $8B_H$ | Bus Configuration Register 2 | $0000_H$ |
| **BUSCON3 b** | $FF18_H$ | $8C_H$ | Bus Configuration Register 3 | $0000_H$ |
| **BUSCON4 b** | $FF1A_H$ | $8D_H$ | Bus Configuration Register 4 | $0000_H$ |
| **BUSSTAT** | $EB20_H$ **X** | --- | SDLM Bus Status Register | $0000_H$ |
| **C1BTR** | $EF04_H$ **X** | --- | CAN1 Bit Timing Register | $UUUU_H$ |
| **C1CSR** | $EF00_H$ **X** | --- | CAN1 Control / Status Register | $XX01_H$ |
| **C1GMS** | $EF06_H$ **X** | --- | CAN1 Global Mask Short | $UFUU_H$ |

**Table 7 C161CS/JC/JI Registers, Ordered by Name** (cont'd)

| Name | Physical Address | 8-Bit Addr. | Description | Reset Value |
|---|---|---|---|---|
| **C1PCIR** | $EF02_H$ **X** | --- | CAN1 Port Control / Interrupt Register | $XXXX_H$ |
| **C1LARn** | $EFn4_H$ **X** | --- | CAN1 Lower Arbitration Reg. (msg. **n**) | $UUUU_H$ |
| **C1LGML** | $EF0A_H$ **X** | --- | CAN1 Lower Global Mask Long | $UUUU_H$ |
| **C1LMLM** | $EF0E_H$ **X** | --- | CAN1 Lower Mask of Last Message | $UUUU_H$ |
| **C1MCFGn** | $EFn6_H$ **X** | --- | CAN1 Message Config. Reg. (msg. **n**) | $UU_H$ |
| **C1MCRn** | $EFn0_H$ **X** | --- | CAN1 Message Control Reg. (msg. **n**) | $UUUU_H$ |
| **C1UARn** | $EFn2_H$ **X** | --- | CAN1 Upper Arbitration Reg. (msg. **n**) | $UUUU_H$ |
| **C1UGML** | $EF08_H$ **X** | --- | CAN1 Upper Global Mask Long | $UUUU_H$ |
| **C1UMLM** | $EF0C_H$ **X** | --- | CAN1 Upper Mask of Last Message | $UUUU_H$ |
| **C2BTR** | $EE04_H$ **X** | --- | CAN2 Bit Timing Register | $UUUU_H$ |
| **C2CSR** | $EE00_H$ **X** | --- | CAN2 Control / Status Register | $XX01_H$ |
| **C2GMS** | $EE06_H$ **X** | --- | CAN2 Global Mask Short | $UFUU_H$ |
| **C2PCIR** | $EE02_H$ **X** | --- | CAN2 Port Control / Interrupt Register | $XXXX_H$ |
| **C2LARn** | $EEn4_H$ **X** | --- | CAN2 Lower Arbitration Reg. (msg. **n**) | $UUUU_H$ |
| **C2LGML** | $EE0A_H$ **X** | --- | CAN2 Lower Global Mask Long | $UUUU_H$ |
| **C2LMLM** | $EE0E_H$ **X** | --- | CAN2 Lower Mask of Last Message | $UUUU_H$ |
| **C2MCFGn** | $EEn6_H$ **X** | --- | CAN2 Message Config. Reg. (msg. **n**) | $UU_H$ |
| **C2MCRn** | $EEn0_H$ **X** | --- | CAN2 Message Control Reg. (msg. **n**) | $UUUU_H$ |
| **C2UARn** | $EEn2_H$ **X** | --- | CAN2 Upper Arbitration Reg. (msg. **n**) | $UUUU_H$ |
| **C2UGML** | $EE08_H$ **X** | --- | CAN2 Upper Global Mask Long | $UUUU_H$ |
| **C2UMLM** | $EE0C_H$ **X** | --- | CAN2 Upper Mask of Last Message | $UUUU_H$ |
| **CAPREL** | $FE4A_H$ | $25_H$ | GPT2 Capture/Reload Register | $0000_H$ |
| **CC0** | $FE80_H$ | $40_H$ | CAPCOM Register 0 | $0000_H$ |
| **CC0IC** **b** | $FF78_H$ | $BC_H$ | CAPCOM Register 0 Interrupt Ctrl. Reg. | $0000_H$ |
| **CC1** | $FE82_H$ | $41_H$ | CAPCOM Register 1 | $0000_H$ |
| **CC10** | $FE94_H$ | $4A_H$ | CAPCOM Register 10 | $0000_H$ |
| **CC10IC** **b** | $FF8C_H$ | $C6_H$ | CAPCOM Reg. 10 Interrupt Ctrl. Reg. | $0000_H$ |
| **CC11** | $FE96_H$ | $4B_H$ | CAPCOM Register 11 | $0000_H$ |
| **CC11IC** **b** | $FF8E_H$ | $C7_H$ | CAPCOM Reg. 11 Interrupt Ctrl. Reg. | $0000_H$ |
| **CC12** | $FE98_H$ | $4C_H$ | CAPCOM Register 12 | $0000_H$ |
| **CC12IC** **b** | $FF90_H$ | $C8_H$ | CAPCOM Reg. 12 Interrupt Ctrl. Reg. | $0000_H$ |
| **CC13** | $FE9A_H$ | $4D_H$ | CAPCOM Register 13 | $0000_H$ |

6) These parameters describe the $\overline{\text{RSTIN}}$ pullup, which equals a resistance of ca. 50 to 250 kΩ.

7) The maximum current may be drawn while the respective signal line remains inactive.

8) The minimum current must be drawn in order to drive the respective signal line active.

9) This specification is valid during Reset and during Hold-mode or Adapt-mode. During Hold-mode Port 6 pins are only affected, if they are used (configured) for $\overline{\text{CS}}$ output and the open drain function is not enabled. The $\overline{\text{READY}}$-pullup is always active, except for Powerdown mode.

10) This specification is valid during Reset and during Adapt-mode.

11) Not 100% tested, guaranteed by design and characterization.

## Power Consumption C161CS/JC/JI

(Operating Conditions apply)

| Parameter | Symbol | Limit Values | | Unit | Test Condition |
|---|---|---|---|---|---|
| | | min. | max. | | |
| Power supply current (active) with all peripherals active | $I_{DD}$ | – | 15 + 2.5 × $f_{CPU}$ | mA | $\overline{\text{RSTIN}}$ = $V_{IL}$ $f_{CPU}$ in [MHz][1] |
| Idle mode supply current with all peripherals active | $I_{IDX}$ | – | 5 + 1.5 × $f_{CPU}$ | mA | $\overline{\text{RSTIN}}$ = $V_{IH1}$ $f_{CPU}$ in [MHz][1] |
| Idle mode supply curr., Main osc, with all peripherals deactivated, PLL off, SDD factor = 32 | $I_{IDOM}$[2] | – | 500 + 50 × $f_{OSC}$ | µA | $\overline{\text{RSTIN}}$ = $V_{IH1}$ $f_{OSC}$ in [MHz][1] |
| Idle mode supply curr., Aux. osc, with all peripherals deactivated, PLL off, SDD factor = 32 | $I_{IDOA}$[2] | – | 100 | µA | $V_{DD}$ = $V_{DDmax}$ $f_{OSC}$ = 32 kHz[3] |
| Sleep and Power-down mode supply current with RTC running on main oscillator | $I_{PDRM}$[2] | – | 200 + 25 × $f_{OSC}$ | µA | $V_{DD}$ = $V_{DDmax}$ $f_{OSC}$ in [MHz][3] |
| Sleep and Power-down mode supply current with RTC disabled | $I_{PDO}$ | – | 50 | µA | $V_{DD}$ = $V_{DDmax}$[3] |

1) The supply current is a function of the operating frequency. This dependency is illustrated in **Figure 10**. These parameters are tested at $V_{DDmax}$ and maximum CPU clock with all outputs disconnected and all inputs at $V_{IL}$ or $V_{IH}$.

2) This parameter is determined mainly by the current consumed by the oscillator (see **Figure 9**). This current, however, is influenced by the external oscillator circuitry (crystal, capacitors). The values given refer to a typical circuitry and may change in case of a not optimized external oscillator circuitry.

3) This parameter is tested including leakage currents. All inputs (including pins configured as inputs) at 0 V to 0.1 V or at $V_{DD}$ - 0.1 V to $V_{DD}$, all outputs (including pins configured as outputs) disconnected.

P0.15-13 (P0H.7-5). Register RP0H can be loaded from the upper half of register RSTCON under software control.

**Table 10** associates the combinations of these three bits with the respective clock generation mode.

**Table 10 C161CS/JC/JI Clock Generation Modes**

| CLKCFG (P0H.7-5) | CPU Frequency $f_{CPU} = f_{OSC} \times F$ | External Clock Input Range[1)] | Notes |
|---|---|---|---|
| 1 1 1 | $f_{OSC} \times 4$ | 2.5 to 6.25 MHz | Default configuration |
| 1 1 0 | $f_{OSC} \times 3$ | 3.33 to 8.33 MHz | – |
| 1 0 1 | $f_{OSC} \times 2$ | 5 to 12.5 MHz | – |
| 1 0 0 | $f_{OSC} \times 5$ | 2 to 5 MHz | – |
| 0 1 1 | $f_{OSC} \times 1$ | 1 to 25 MHz | Direct drive[2)] |
| 0 1 0 | $f_{OSC} \times 1.5$ | 6.66 to 16.6 MHz | – |
| 0 0 1 | $f_{OSC} / 2$ | 2 to 50 MHz | CPU clock via prescaler |
| 0 0 0 | $f_{OSC} \times 2.5$ | 4 to 10 MHz | – |

1) The external clock input range refers to a CPU clock range of 10 … 25 MHz.

2) The maximum frequency depends on the duty cycle of the external clock signal.

### Prescaler Operation

When prescaler operation is configured (CLKCFG = $001_B$) the CPU clock is derived from the internal oscillator (input clock signal) by a 2:1 prescaler.
The frequency of $f_{CPU}$ is half the frequency of $f_{OSC}$ and the high and low time of $f_{CPU}$ (i.e. the duration of an individual TCL) is defined by the period of the input clock $f_{OSC}$.

The timings listed in the AC Characteristics that refer to TCLs therefore can be calculated using the period of $f_{OSC}$ for any TCL.

### Phase Locked Loop

When PLL operation is configured (via CLKCFG) the on-chip phase locked loop is enabled and provides the CPU clock (see table above). The PLL multiplies the input frequency by the factor **F** which is selected via the combination of pins P0.15-13 (i.e. $f_{CPU} = f_{OSC} \times$ **F**). With every **F**'th transition of $f_{OSC}$ the PLL circuit synchronizes the CPU clock to the input clock. This synchronization is done smoothly, i.e. the CPU clock frequency does not change abruptly.

Due to this adaptation to the input clock the frequency of $f_{CPU}$ is constantly adjusted so it is locked to $f_{OSC}$. The slight variation causes a jitter of $f_{CPU}$ which also effects the duration of individual TCLs.

The timings listed in the AC Characteristics that refer to TCLs therefore must be calculated using the minimum TCL that is possible under the respective circumstances.

The actual minimum value for TCL depends on the jitter of the PLL. As the PLL is constantly adjusting its output frequency so it corresponds to the applied input frequency (crystal or oscillator) the relative deviation for periods of more than one TCL is lower than for one single TCL (see formula and **Figure 12**).
For a period of $N \times$ TCL the minimum value is computed using the corresponding deviation $D_N$:

$$(N \times TCL)_{min} = N \times TCL_{NOM} - D_N \quad D_N\,[ns] = \pm(13.3 + N \times 6.3) / f_{CPU}\,[MHz],$$

where $N$ = number of consecutive TCLs and $1 \le N \le 40$.

So for a period of 3 TCLs @ 25 MHz (i.e. $N = 3$): $D_3 = (13.3 + 3 \times 6.3) / 25 = 1.288$ ns, and $(3TCL)_{min} = 3TCL_{NOM} - 1.288$ ns = 58.7 ns (@ $f_{CPU}$ = 25 MHz).

This is especially important for bus cycles using waitstates and e.g. for the operation of timers, serial interfaces, etc. For all slower operations and longer periods (e.g. pulse train generation or measurement, lower baudrates, etc.) the deviation caused by the PLL jitter is neglectible.

*Note: For all periods longer than 40 TCL the N = 40 value can be used (see **Figure 12**).*

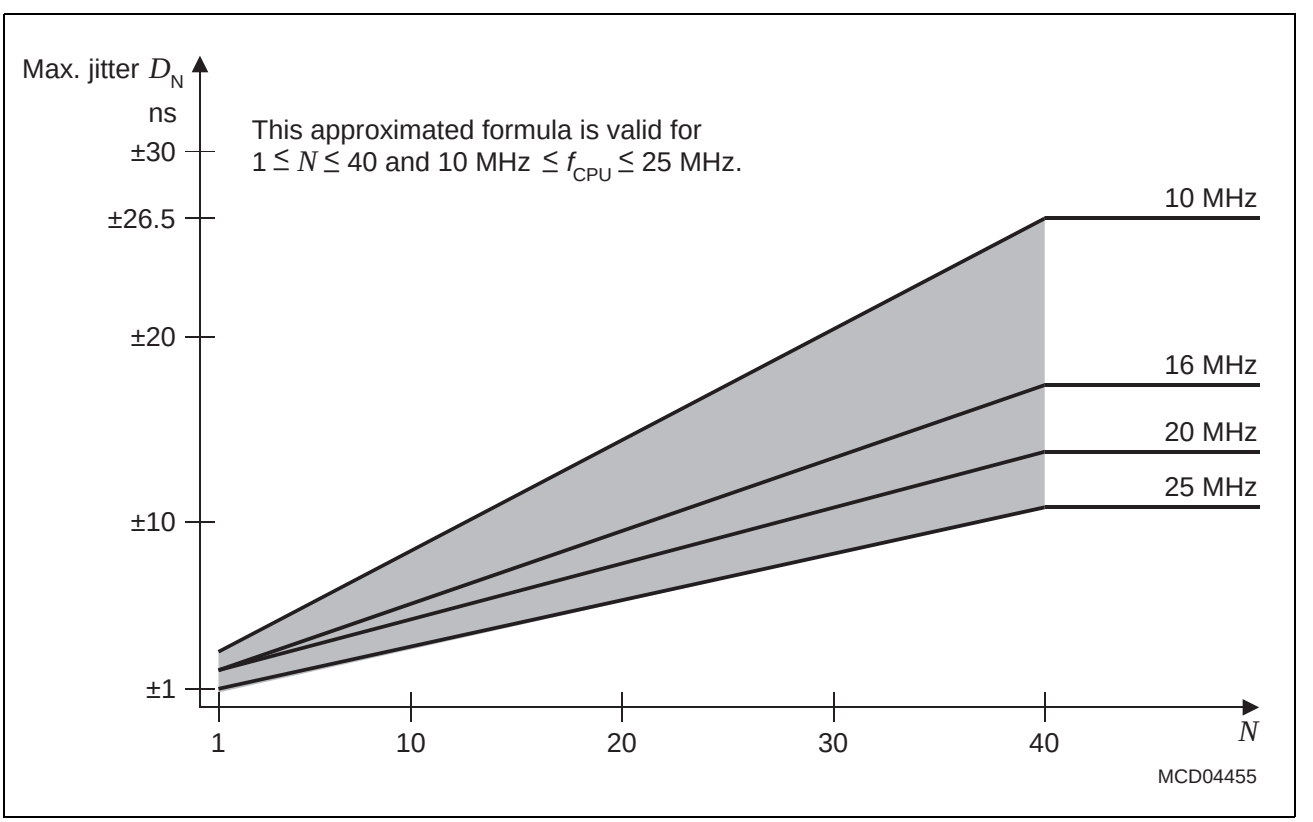


**Figure 12 Approximated Maximum Accumulated PLL Jitter**

8) During the sample time the input capacitance $C_{AIN}$ can be charged/discharged by the external source. The internal resistance of the analog source must allow the capacitance to reach its final voltage level within $t_S$. After the end of the sample time $t_S$, changes of the analog input voltage have no effect on the conversion result.
Values for the sample time $t_S$ depend on programming and can be taken from **Table 14**.

Sample time and conversion time of the C161CS/JC/JI's A/D Converter are programmable. **Table 14** should be used to calculate the above timings.
The limit values for $f_{BC}$ must not be exceeded when selecting ADCTC.

**Table 14 A/D Converter Computation Table**

| ADCON.15\|14 (ADCTC) | A/D Converter Basic Clock $f_{BC}$ | ADCON.13\|12 (ADSTC) | Sample time $t_S$ |
|---|---|---|---|
| 00 | $f_{CPU}$ / 4 | 00 | $t_{BC} \times 8$ |
| 01 | $f_{CPU}$ / 2 | 01 | $t_{BC} \times 16$ |
| 10 | $f_{CPU}$ / 16 | 10 | $t_{BC} \times 32$ |
| 11 | $f_{CPU}$ / 8 | 11 | $t_{BC} \times 64$ |

Converter Timing Example:

| | | |
|---|---|---|
| Assumptions: | $f_{CPU}$ | = 25 MHz (i.e. $t_{CPU}$ = 40 ns), ADCTC = '00', ADSTC = '00'. |
| Basic clock | $f_{BC}$ | = $f_{CPU}$ / 4 = 6.25 MHz, i.e. $t_{BC}$ = 160 ns. |
| Sample time | $t_S$ | = $t_{BC} \times 8$ = 1280 ns. |
| Conversion time | $t_C$ | = $t_S$ + 40 $t_{BC}$ + 2 $t_{CPU}$ = (1280 + 6400 + 80) ns = 7.8 µs. |

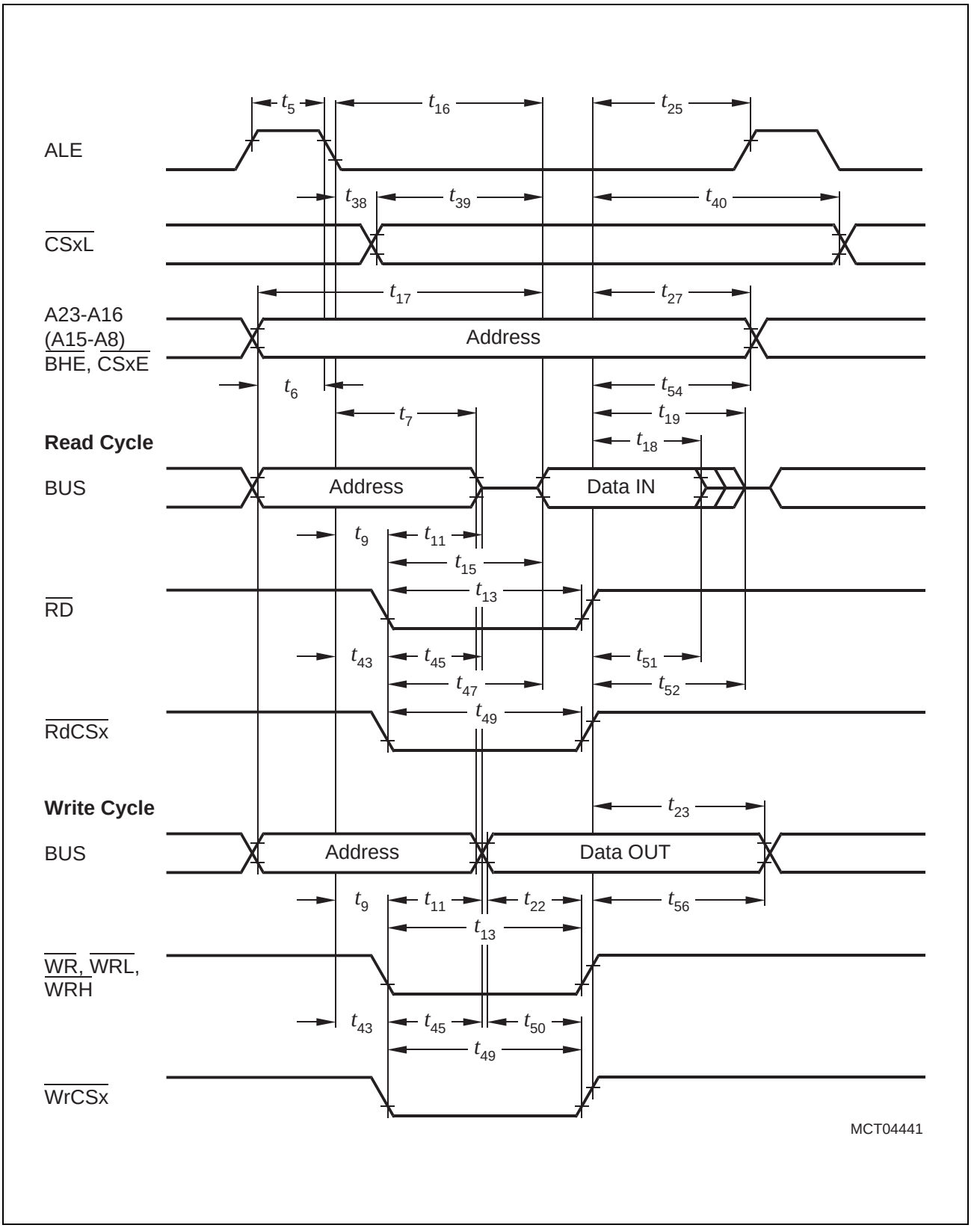


**Figure 18 External Memory Cycle:
Multiplexed Bus, No Read/Write Delay, Normal ALE**

## AC Characteristics

### Demultiplexed Bus

(Operating Conditions apply)
ALE cycle time = 4 TCL + 2$t_A$ + $t_C$ + $t_F$ (80 ns at 25 MHz CPU clock without waitstates)

| Parameter | Symbol | | Max. CPU Clock = 25 MHz | | Variable CPU Clock 1 / 2TCL = 1 to 25 MHz | | Unit |
|---|---|---|---|---|---|---|---|
| | | | min. | max. | min. | max. | |
| ALE high time | $t_5$ | CC | 10 + $t_A$ | – | TCL - 10 + $t_A$ | – | ns |
| Address setup to ALE | $t_6$ | CC | 4 + $t_A$ | – | TCL - 16 + $t_A$ | – | ns |
| ALE falling edge to $\overline{RD}$, $\overline{WR}$ (with RW-delay) | $t_8$ | CC | 10 + $t_A$ | – | TCL - 10 + $t_A$ | – | ns |
| ALE falling edge to $\overline{RD}$, $\overline{WR}$ (no RW-delay) | $t_9$ | CC | -10 + $t_A$ | – | -10 + $t_A$ | – | ns |
| $\overline{RD}$, $\overline{WR}$ low time (with RW-delay) | $t_{12}$ | CC | 30 + $t_C$ | – | 2TCL - 10 + $t_C$ | – | ns |
| $\overline{RD}$, $\overline{WR}$ low time (no RW-delay) | $t_{13}$ | CC | 50 + $t_C$ | – | 3TCL - 10 + $t_C$ | – | ns |
| $\overline{RD}$ to valid data in (with RW-delay) | $t_{14}$ | SR | – | 20 + $t_C$ | – | 2TCL - 20 + $t_C$ | ns |
| $\overline{RD}$ to valid data in (no RW-delay) | $t_{15}$ | SR | – | 40 + $t_C$ | – | 3TCL - 20 + $t_C$ | ns |
| ALE low to valid data in | $t_{16}$ | SR | – | 40 + $t_A$ + $t_C$ | – | 3TCL - 20 + $t_A$ + $t_C$ | ns |
| Address to valid data in | $t_{17}$ | SR | – | 50 + 2$t_A$ + $t_C$ | – | 4TCL - 30 + 2$t_A$ + $t_C$ | ns |
| Data hold after $\overline{RD}$ rising edge | $t_{18}$ | SR | 0 | – | 0 | – | ns |
| Data float after $\overline{RD}$ rising edge (with RW-delay[1)]) | $t_{20}$ | SR | – | 26 + 2$t_A$ + $t_F$[1)] | – | 2TCL - 14 + 22$t_A$ + $t_F$[1)] | ns |
| Data float after $\overline{RD}$ rising edge (no RW-delay[1)]) | $t_{21}$ | SR | – | 10 + 2$t_A$ + $t_F$[1)] | – | TCL - 10 + 22$t_A$ + $t_F$[1)] | ns |

**Demultiplexed Bus** (cont'd)
(Operating Conditions apply)
ALE cycle time = 4 TCL + 2$t_A$ + $t_C$ + $t_F$ (80 ns at 25 MHz CPU clock without waitstates)

| Parameter | Symbol | | Max. CPU Clock = 25 MHz | | Variable CPU Clock 1 / 2TCL = 1 to 25 MHz | | Unit |
|---|---|---|---|---|---|---|---|
| | | | min. | max. | min. | max. | |
| Data float after $\overline{\text{RdCS}}$ (no RW-delay)[1] | $t_{68}$ | SR | – | 0 + $t_F$ | – | TCL - 20 + 2$t_A$ + $t_F$[1] | ns |
| Address hold after $\overline{\text{RdCS}}$, $\overline{\text{WrCS}}$ | $t_{55}$ | CC | -6 + $t_F$ | – | -6 + $t_F$ | – | ns |
| Data hold after $\overline{\text{WrCS}}$ | $t_{57}$ | CC | 6 + $t_F$ | – | TCL - 14 + $t_F$ | – | ns |

1) RW-delay and $t_A$ refer to the next following bus cycle (including an access to an on-chip X-Peripheral).

2) Read data are latched with the same clock edge that triggers the address change and the rising $\overline{\text{RD}}$ edge. Therefore address changes before the end of $\overline{\text{RD}}$ have no impact on read cycles.

3) These parameters refer to the latched chip select signals ($\overline{\text{CSxL}}$). The early chip select signals ($\overline{\text{CSxE}}$) are specified together with the address and signal $\overline{\text{BHE}}$ (see figures below).

## AC Characteristics

### CLKOUT and $\overline{\text{READY}}$

(Operating Conditions apply)

| Parameter | Symbol | | Max. CPU Clock = 25 MHz | | Variable CPU Clock 1 / 2TCL = 1 to 25 MHz | | Unit |
|---|---|---|---|---|---|---|---|
| | | | min. | max. | min. | max. | |
| CLKOUT cycle time | $t_{29}$ | CC | 40 | 40 | 2TCL | 2TCL | ns |
| CLKOUT high time | $t_{30}$ | CC | 14 | – | TCL - 6 | – | ns |
| CLKOUT low time | $t_{31}$ | CC | 10 | – | TCL - 10 | – | ns |
| CLKOUT rise time | $t_{32}$ | CC | – | 4 | – | 4 | ns |
| CLKOUT fall time | $t_{33}$ | CC | – | 4 | – | 4 | ns |
| CLKOUT rising edge to ALE falling edge | $t_{34}$ | CC | 0 + $t_A$ | 10 + $t_A$ | 0 + $t_A$ | 10 + $t_A$ | ns |
| Synchronous $\overline{\text{READY}}$ setup time to CLKOUT | $t_{35}$ | SR | 14 | – | 14 | – | ns |
| Synchronous $\overline{\text{READY}}$ hold time after CLKOUT | $t_{36}$ | SR | 4 | – | 4 | – | ns |
| Asynchronous $\overline{\text{READY}}$ low time | $t_{37}$ | SR | 54 | – | 2TCL + $t_{58}$ | – | ns |
| Asynchronous $\overline{\text{READY}}$ setup time[1)] | $t_{58}$ | SR | 14 | – | 14 | – | ns |
| Asynchronous $\overline{\text{READY}}$ hold time[1)] | $t_{59}$ | SR | 4 | – | 4 | – | ns |
| Async. $\overline{\text{READY}}$ hold time after $\overline{\text{RD}}$, $\overline{\text{WR}}$ high (Demultiplexed Bus)[2)] | $t_{60}$ | SR | 0 | 0 + $2t_A$ + $t_C$ + $t_F$[2)] | 0 | TCL - 20 + $2t_A$ + $t_C$ + $t_F$[2)] | ns |

1) These timings are given for test purposes only, in order to assure recognition at a specific clock edge.

2) Demultiplexed bus is the worst case. For multiplexed bus 2TCL are to be added to the maximum values. This adds even more time for deactivating $\overline{\text{READY}}$.
The $2t_A$ and $t_C$ refer to the next following bus cycle, $t_F$ refers to the current bus cycle.
The maximum limit for $t_{60}$ must be fulfilled if the next following bus cycle is $\overline{\text{READY}}$ controlled.

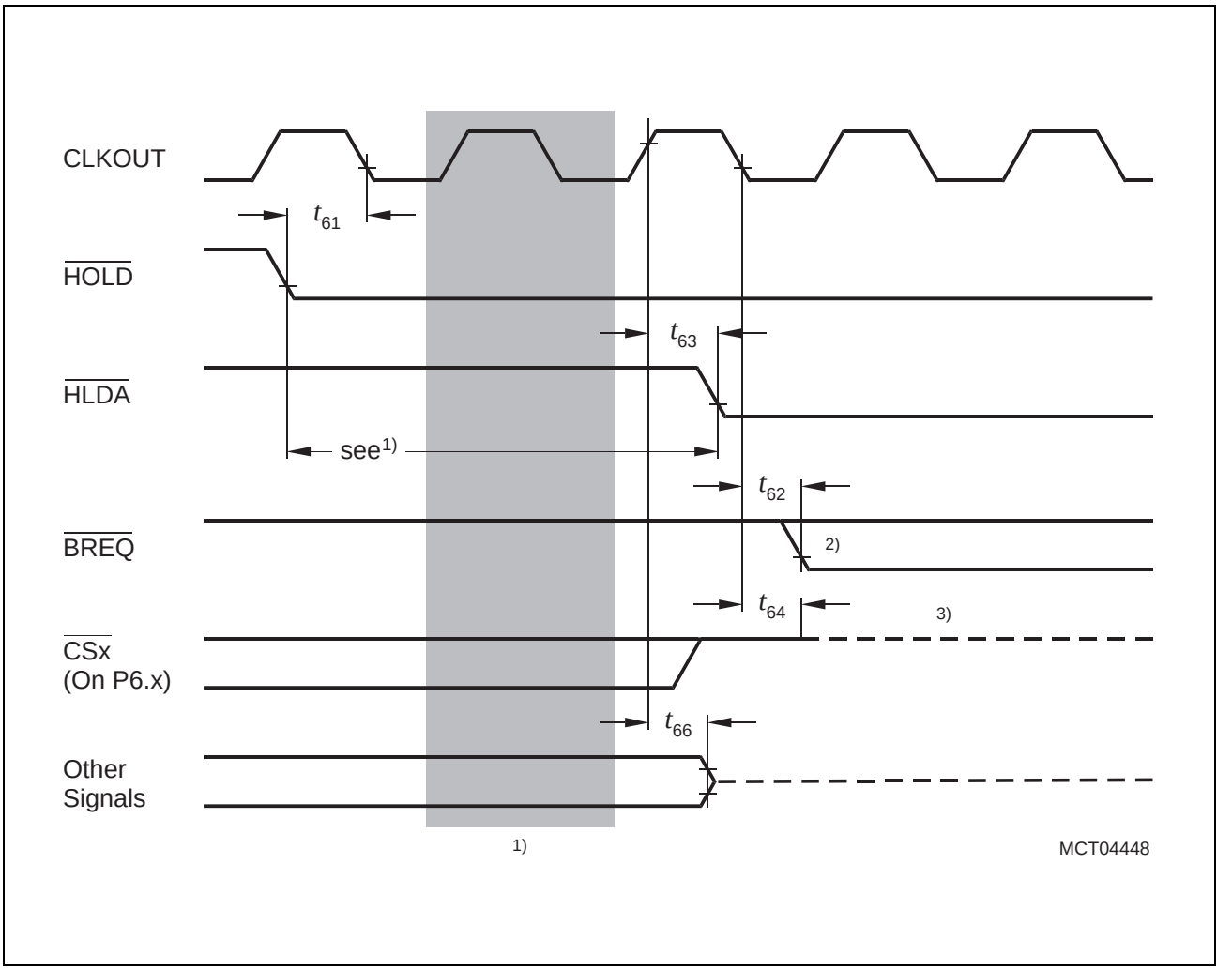


**Figure 25 External Bus Arbitration, Releasing the Bus**

**Notes**
1) The C161CS/JC/JI will complete the currently running bus cycle before granting bus access.
2) This is the first possibility for $\overline{\text{BREQ}}$ to get active.
3) The $\overline{\text{CS}}$ outputs will be resistive high (pullup) after $t_{64}$.

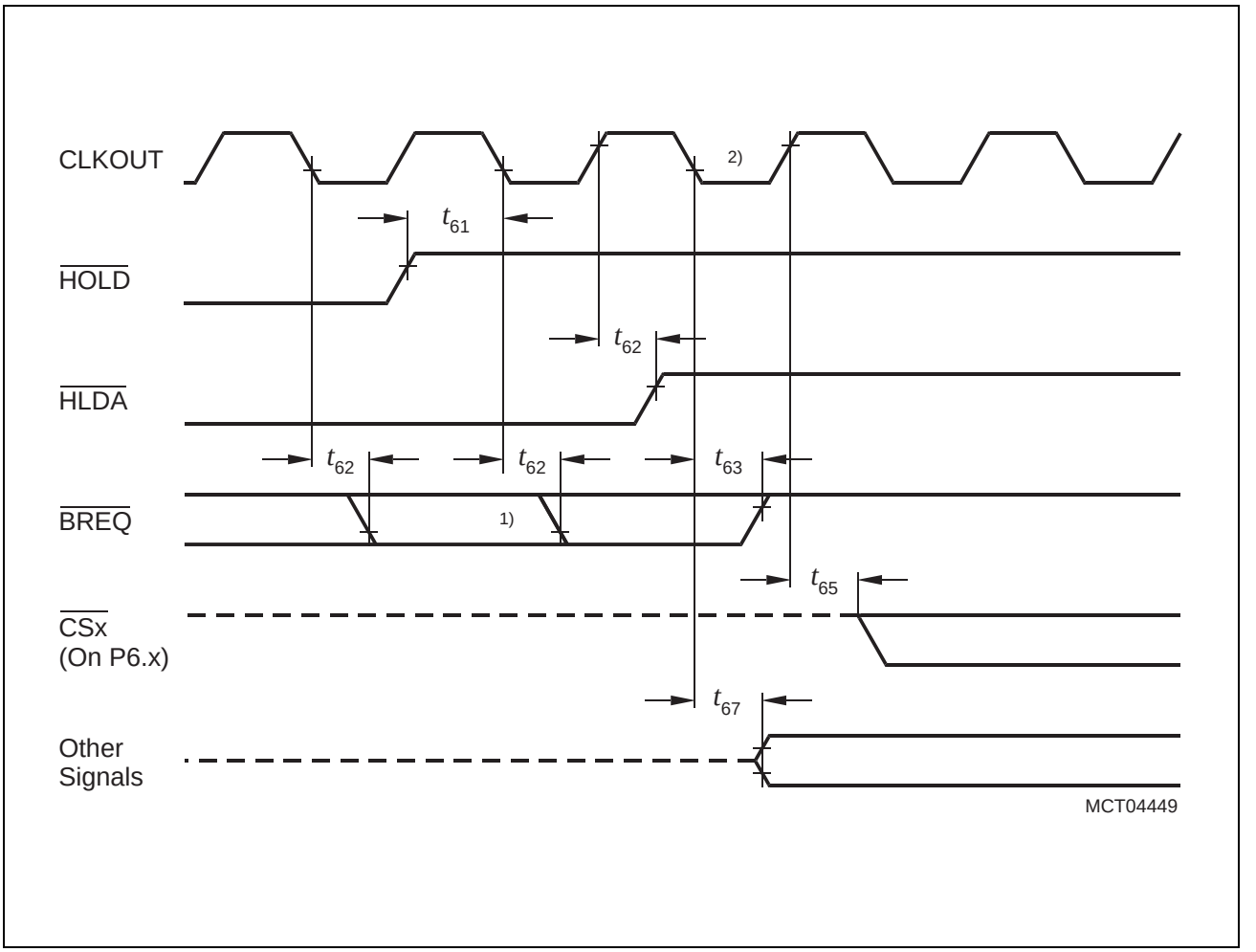


**Figure 26 External Bus Arbitration, (Regaining the Bus)**

**Notes**

1) This is the last chance for BREQ to trigger the indicated regain-sequence.
Even if BREQ is activated earlier, the regain-sequence is initiated by HOLD going high.
Please note that HOLD may also be deactivated without the C161CS/JC/JI requesting the bus.

2) The next C161CS/JC/JI driven bus cycle may start here.