87C196JT Specification Update

Automotive

December 1998

Notice: The 87C196JT may contain design defects or errors known as errata. Characterized errata that may cause the 87C196JT's behavior to deviate from published specifications are documented in this specification update.

Order Number: 273220-001

Contents

Revision History

Date	Version	Description
12/01/98	001	This is the new Specification Update document. It contains all identified errata published prior to this date.

Preface

As of July, 1996, Intel's Computing Enhancement Group has consolidated available historical device and documentation errata into this new document type called the Specification Update. We have endeavored to include all documented errata in the consolidation process, however, we make no representations or warranties concerning the completeness of the Specification Update.

This document is an update to the specifications contained in the Affected Documents/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Information types defined in Nomenclature are consolidated into the specification update and are no longer published in other documents.

This document may also contain information that was not previously published.

Affected Documents/Related Documents

Title	Order #
87C196KR, 87C196JV, 87C196JT, 87C196JR, and 87C196CA Advanced 16-bit CHMOS Microcontroller datasheet - Automotive	270827
8XC196Kx, 8XC196Jx, 87C196CA Microcontroller Family User's Manual	272258

Nomenclature

Errata are design defects or errors. These may cause the Product Name's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Product Name product. Intel may fix some of the errata in a future stepping of the component, and account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Errata exists in the stepping indicated. Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Page

(Page):	Page location of item in this document.

Status

Doc:	Document change or update will be implemented.
Fix:	This erratum is intended to be fixed in a future step of the component.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.
Eval:	Plans to fix this erratum are under evaluation.

Row

▌	Change bar to left of table row indicates this erratum is either new or modified from the previous version of the document.

Errata

No.	Steppings			Page	Status	ERRATA
	A	B	#			
1	X	X		10	NoFix	Executing Routines in the User's ROM While the Device is Operating in Serial Programming Mode
2	X	X		10	NoFix	A/D Conversion Error on First Conversion

Specification Changes

No.	Steppings			Page	Status	SPECIFICATION CHANGES
	A	B	#			
1	X	X		11		DC Characteristics, Idle Mode Current
2		X		11		A/D Conversion Times

Specification Clarifications

No.	Steppings			Page	Status	SPECIFICATION CLARIFICATIONS
	A	B	#			
1	X	X		12		EPA Timer Reset/Write Conflict
2	X	X		12		Valid Time Matches
3	X	X		12		P6__REG.4-.7 Not Updated Immediately
4	X	X		12		Write Cycle During Reset
5	X	X		12		Indirect Shift Count Value
6	X	X		12		P2.7 (CLKOUT)
7	X	X		12		EPA Overruns
8	X	X		14		Indirect Addressing With AutoIncrement

Documentation Changes

No.	Document Revision	Page	Status	DOCUMENTATION CHANGES
1	272258-002	15	Doc	High address inputs to the low EPROM should be A14:8 in Figure 15-18
2	272258-002	16	Doc	Addresses 201DH and 201FH should contain FFH in Table 4-2
3	272258-002	16	Doc	In Table 7-2, P2_DIR description change
4	231792-009	16	Doc	Section 7, Page 5-100, Port 5 Register Behavior
5	272529-002	17	Doc	Section 7, Page 18, Port 5 Register Behavior

Identification Information

Markings

A-step devices are identified by the letter “A” following the eight-digit FPO number on the top of the package.

B-step devices are identified by the letter “B” following the eight-digit FPO number on the top of the package.

Errata

1. Executing Routines in the User's ROM While the Device is Operating in Serial Programming Mode

Problem: All code fetches above the first 8K bytes of user ROM while the device is operating in serial port programming mode will be directed to external memory. Therefore, if the user wants to call any routines in the user ROM, the entire routine must be within the first 8K bytes of memory (0A000 – 0BFFFH in serial port programming mode). For example, if the RISM "GO" command is used with a target address of 0C000H, the device will attempt to fetch code from external memory rather than the on-board ROM.

Implication: This errata only affects code fetches from the user ROM. Data fetches to the entire ROM work correctly. It is not possible to execute code from above the first 8K byte of user ROM while the device is operating in Serial Port Programming mode.

Workaround: None.

Status: NoFix. Refer to the Summary Table of Changes to determine the affected.

2. A/D Conversion Error on First Conversion

Problem: The first A/D conversion performed after a reset of the device may produce a result that is less accurate than the accuracy specification in the datasheet. The amount of error is dependent on several environmental conditions including process variation, light exposure, temperature, and V_{CC}/V_{REF} differential.

Implication: If the application is sensitive to A/D accuracy, an error in the application may occur as a result of the first conversion error.

Workaround: Do not compare for the A/D result on the first conversion. All subsequent conversions should produce accurate results.

Status: NoFix. Refer to the Summary Table of Changes to determine the affected.

Specification Changes

1. DC Characteristics, Idle Mode Current

Problem: The DC Characteristics Idle Mode Current specification has been revised from a maximum of 30 mA to a maximum of 40 mA.

2. A/D Conversion Times

Issue: The second paragraph of the datasheet cover page incorrectly states the A/D conversion time as "< 5 µs". The correction reads: ...analog-to-digital converter with programmable S/H times with conversion times < 15 µs at 20 MHz,...

Specification Clarifications

1. EPA Timer Reset/Write Conflict

Problem: If software writes to the EPA timer at the same time that an EPA channel resets that timer, it is indeterminate which action will take precedence. Software should not write to a timer that is being reset by EPA signals.

2. Valid Time Matches

Problem: A timer must increment or decrement to the compare value in order for a match to occur. Loading a timer with a value that is equal to an EPA compare value does cause a match. Likewise, with an EPA compare value of 0, a timer reset does not cause a match.

3. P6__REG.4-.7 Not Updated Immediately

Problem: A value written to any of the upper four bits of P6_REG is temporarily held in a buffer until the corresponding P6_MODE bit is cleared, at which time the value is loaded into the P6_REG bit. A value read from a P6_REG bit is the value currently in the register, not the value in the buffer. Therefore, any change to a P6_REG bit can be read only after the corresponding P6_MODE bit is cleared.

4. Write Cycle During Reset

Problem: If a reset occurs while the microcontroller is writing to an external memory device, the contents of the external memory device may be corrupted.

5. Indirect Shift Count Value

Problem: The SHRL and SHLL instructions function correctly with count values 0–31, inclusive. However, a shift count value of XX100000B causes 32 shifts, which results in no shift taking place. With all other count values, the upper 3 bits are masked off and the remaining bits specify the number of shifts. Also, a shift count value of XX1XXXXXB causes the overflow flag and the overflow-trap flag to be set.

Implication: Customers using SHRL and SHLL instructions with a count value greater than 31 will be affected.

Workaround: Ensure that the count value never exceeds 31.

Status: Refer to Summary Table of Changes to determine the affected stepping(s).

6. P2.7 (CLKOUT)

Problem: P2.7 (CLKOUT) does not operate in open drain mode.

7. EPA Overruns

Problem: The EPA can lock up if overruns are handled incorrectly. Overruns occur when an EPA input transitions at a rate that cannot be handled by the EPA interrupt service routine. If no overrun handling strategy is in place, and if the following three conditions exist, a situation may occur where both the capture buffer and the EPAx_TIME register contain data, and no EPA interrupt pending bit is set:

- an input signal with a frequency high enough to cause overruns is present on an enabled EPA pin, and
- the overwrite bit is set (EPAx_CON.0 = 1; old data is overwritten on overrun), and

- the EPAx_TIME register is read at the exact instant that the EPA recognizes the captured edge as valid.

The input frequency at which this occurs depends on the length of the interrupt service routine as well as other factors. Unless the interrupt service routine includes a check for overruns, this situation will remain the same until the device is reset or the EPAx_TIME register is read. The act of reading EPAx_TIME allows the buffered time value to be moved into EPAx_TIME. This clears the buffer and allows another event to be captured. Remember that the act of transferring the buffer contents to the EPAx_TIME register is what actually sets the EPAx interrupt pending bit and generates the interrupt.

Workaround: Any one of the following methods can be used to prevent or recover from an EPA overrun situation.

- Clear EPAx_CON.0

  When the overwrite bit (EPAx_CON.0) is zero and both the EPAx_TIME register than the buffer are full, the EPA does not consider a captured edge until the EPAx_TIME register is read and the data in the capture buffer is transferred to EPAx_TIME. This prevents overruns by ignoring new input capture events when both the capture buffer and EPAx_TIME contain valid capture times. The OVRx pending bit in EPA_PEND is set to indicate that an overrun occurred.

- Enable the OVRx interrupt and read the EPAx_TIME register within the ISR

  If an overrun occurs, the overrun (OVRx) interrupt will be generated. The OVRx interrupt will then be acknowledged and its interrupt service routine will read the EPAx_TIME register. After the CPU reads the EPAx_TIME register, the buffered data moves from the buffer to the EPAx_TIME register. This sets the EPA interrupt pending bit.

- Check for pending EPAx interrupts before exiting an EPAx ISR

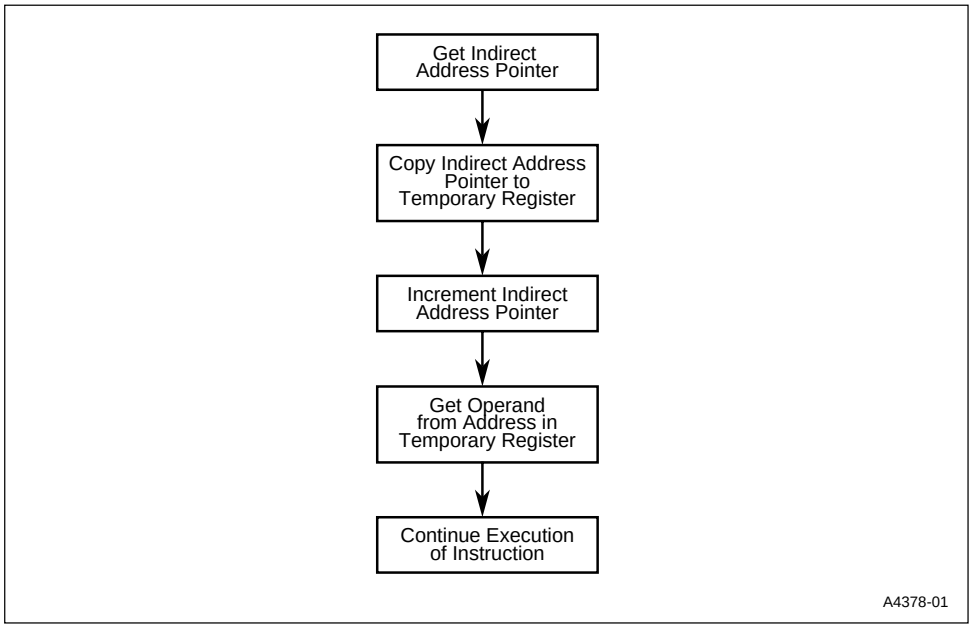
  Another method for avoiding this situation is to check for pending EPA interrupts before exiting the EPA interrupt service routine. This is an easy way to detect overruns and additional interrupts. It can also save loop time by eliminating the latency necessary to service the pending interrupt. However, this method cannot be used with the peripheral transaction server (PTS).

Status: Refer to Summary Table of Changes to determine the affected stepping(s).

8. Indirect Addressing With AutoIncrement

Problem: For indirect addressing with autoincrement, a pointer that points to itself results in an access to the incremented pointer address rather than the original pointer address.

The CPU stores the pointer's value in a temporary register, increments the pointer, then accesses the operand at the address contained in the temporary register, as shown in this flowchart.



Therefore, if the pointer points to itself, the CPU accesses the operand at the incremented address contained in the pointer.

For example, assume ax = 1CH and bx = 40H. The following code causes the CPU to access the operand at the incremented address:

```
ld      ax,#ax
ldb     bx,[ax]+

ld      1CH,#1CH        ;1CH←1CH   ;load location 1CH with value 1CH
ldb     40H,[1CH]+      ;temp←1CH  ;save 1CH into temp register
                        ;1CH←1DH   ;increment the contents of 1CH
                        ;40H←1DH   ;load the contents of location 1CH
                                   ;(location 1CH
                                   ;now contains the value 1DH) into 40H
```

Workaround: Avoid using an indirect address pointer that points to itself.

For example, assume ax = 1CH, bx = 1DH, and cx = 40H. The following code causes the CPU to access the operand at the intended, unincremented address:

```
ld      ax,#bx      ;where bx≠ax
ldb     cx,[ax]+

ld      1CH,#1DH        ;1CH←1DH   ;load location 1CH with value 1DH
ldb     40H,[1CH]+      ;temp←1DH  ;save 1DH (contents of 1CH) into temp
                                   ;register
                        ;1CH←1EH   ;increment the contents of 1CH
                        ;40H←1DH   ;load the contents of temp into 40H
```

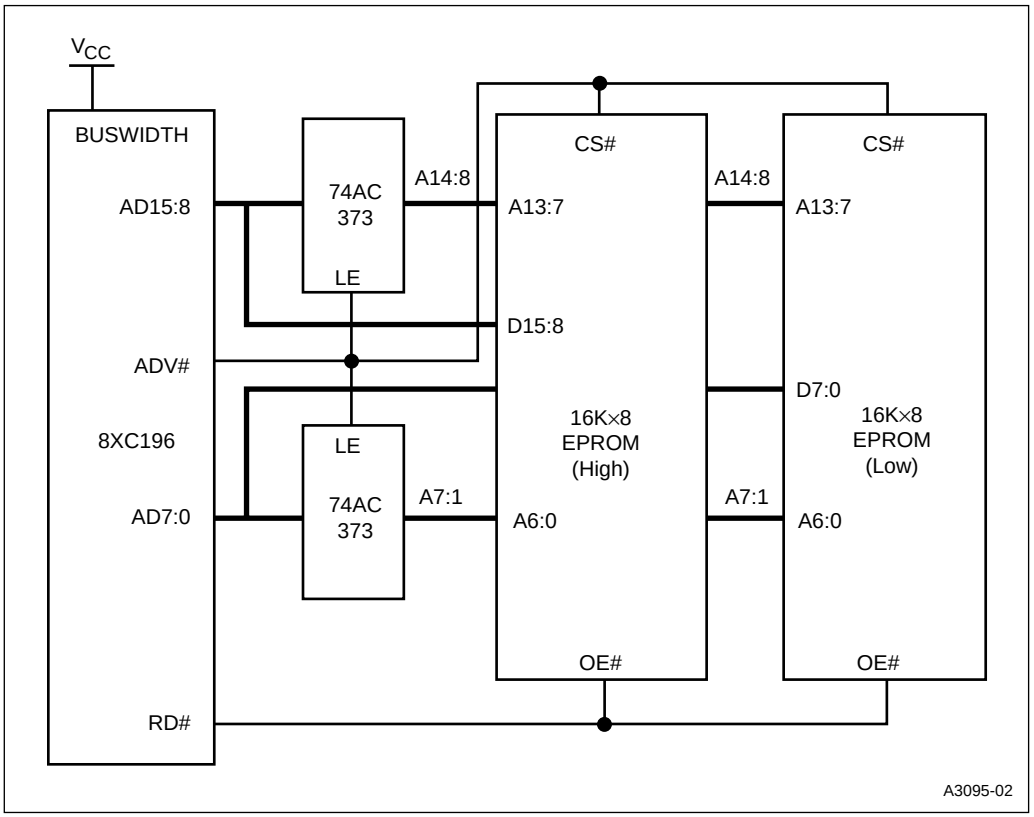
Status: Refer to Summary Table of Changes to determine the affected stepping(s).

Documentation Changes

1. High address inputs to the low EPROM should be A14:8 in Figure 15-18

Issue: The next revision of the user's manual will include the following corrected graphic.

Figure 1. 16-bit System with EPROM



2. Addresses 201DH and 201FH should contain FFH in Table 4-2

Issue: The next revision of the user's manual will include the following corrected table.

Table 1. Special-purpose Memory Addresses

Hex Address	Description
207F 205E	Reserved (each byte must contain FFH)
205D 2040	PTS vectors
203F 2030	Upper interrupt vectors
202F 2020	Security key
201F	Reserved (must contain FFH)
201E	Reserved (must contain FFH)
201D	Reserved (must contain FFH)
201C	Reserved (must contain FFH)
201B	Reserved (must contain 20H)
201A	CCB1
2019	Reserved (must contain 20H)
2018	CCB0
2017 2016	OFD flag (see page 13-12 and page 16-8)
2015 2014	Reserved (each byte must contain FFH)
2013 2000	Lower interrupt vectors

3. In Table 7-2, P2_DIR description change

Issue: The next revision of the user's manual will include the following corrected register description.

Mnemonic	Address	Description
P2_DIR	1FCBH	Port Direction Register Each bit controls the configuration of the corresponding pin. Clearing a bit configures the corresponding pin as a complementary output; setting a bit configures the corresponding pin as an open-drain output or a high-impedance input.

4. Section 7, Page 5-100, Port 5 Register Behavior

Problem: Description of behavior of bits associated with unbonded-out pins P5.1, P5.4, P5.5, P5.6 and P5.7 in registers P5_PIN, P5_MODE, P5_REG, and P5_DIR has been changed.

Entire Section 7 (including Table) under "87C196JR/JQ C-STEP TO JT A-STEP DESIGN CONSIDERATIONS":

- **Old section 7**

  On the JR-C, P5.1, P5.4, P5.5, P5.6 and P5.7 are not bonded out but are present internally on the device.

This allows the programmer to write to the port registers and clear, set or read the pin even though it is not available to the outside world. However, to maintain compatibility with JT A-step and future devices, it is recommended that the corresponding bits associated with the removed pins not be used to conditionally branch in software. These bits should be treated as reserved.

On the JT A-step, unused port logic for these five port pins has been removed from the device and is not available to the programmer. Corresponding bits in the port registers have been "hardwired" to provide the following results when read:

Register Bits		When Read
P5_PIN.x	(x=1,4,5,6,7)	1
P5_REG.x	(x=1,4,5,6,7)	1
P5_DIR.x	(x=1,4,5,6,7)	1
P5_MODE.x	(x=1,4,6)	0
P5_MODE.x	(x = 5) (EA# = 0)	1
P5_MODE.x	(x = 5) (EA# = 1)	0
P5_MODE.x	(x = 7)	1

Writing to these bits will have no effect.

- **New Section 7**

  On the JR-C, P5.1, P5.4, P5.5, P5.6 and P5.7 are not bonded out but are present internally on the device. This allows the programmer to write to the port registers and clear, set or read the pin even though it is not available to the outside world.

  On the JT A-step, unused port logic for these five port pins has been removed. The data read from the P5_PIN, P5_REG, P5_MODE, and P5_DIR register bits associated with these removed pins can be unpredictable on these devices due to the removed logic. Therefore, these bits should not be used to conditionally branch in software. These bits should be treated as reserved.

Affected Docs: *87C196JT 20 MHz Advanced 16-bit CHMOS Microcontroller* datasheet - 272529

5. Section 7, Page 18, Port 5 Register Behavior

(Same as Document Change #004 above).