# Interfacing the Intel Flash 28F001BX-T to Your 186 Based System

### Why Should You Use FLASH?

FLASH provides an easy and cost effective method of implementing system firmware and configuration. It allowsremote, non hardware based upgrades. Flash can be erased electrically, unlike EPROMs and are typically higher density and cheaper than EEPROMs. Unlike battery backed RAM, Flash is nonvolatile. They are ideal for a Solid State disk/card, possibly storing data, an operating system, or even user information.

### Flash Concerns

- Requires 12V to program (5 volt systems can use a charge pump).
- 10,000 minimum to 100,000 typical Erase/Program cycles.
- More expensive per device cost compared to Ram or Eprom.

### Why the 28F001BX-T Flash?

The 28F001BX-T provides 128K Bytes (1MBit) of CMOS Flash memory and contains many features, such as:

- TTL compatible control inputs.
- Versatile memory organization:

| *Memory Blocks* | Memory Address |
|---|---|
| One 8 Kb Boot Block w/lock out | 1E000 to 1FFFF |
| Two 4KB Parameter Blocks | 1D000 to 1DFFF 1C000 to 1CFFF |
| One 112KB Main Block | |

- Contains both a Command register and Status register.
- On chip state machine controls block erase and byte program.
- Quick Pulse programming algorithm and Quick-Erase algorithm integrated internally by the state machine.
- Data Protection provided by hardware and register.

- Inteligent Identifier outputs manufacturer code and device code.
- Deep Power-Down feature, dissipates a maximum of 0.25uW.
- 10,000 to 100,000 Block Erase/Program cycle limit.

## Hardware Design

To accommodate the 16 bit data bus of the C186, two Flash memories are used. Figure 1 illustrates the Flash interface to the 80C186. The chip enables are tied to UCS, which is configured for a 256K byte block size, starting at C0000H. This puts the Boot Block at FC000H. The Block Protect in Figure 1 will tie the PWD pin to $V_{CC}$ or +12VDC. If tied to $V_{CC}$, it will not be possible to write to the Boot block, otherwise strapping it to 12 volts will allow erasure of the block. The signal VPPEN is used to turn on and off 12 volts to $V_{PP}$. This line can be connected to one of the ports on the C186. When activating this signal, be sure to allow some switching time for the Mosfet transistor. The 12 volt supply should be stable and able to supply at least 60mA of current. The address lines (LA1-LA17) connected to the Flash need to be latched using ALE from the 80C186.

## Program/Erasure

There are two registers provided on the 28F001BX Flash, one to write commands and the other to read the status of the device. The command register does not occupy any memory locations but instead is accessed during a write operation to the device. When using commands that are block specific it is necessary to write the command to an address within the block. Be sure to access the correct even or odd block address when doing byte operations.

Programming the Flash requires $V_{PP}$ to be driven to 12VDC. This can be accomplished by using a jumper to supply 12 volts to $V_{PP}$ or a port can be used. A circuit is provided for turning the 12 volts on or off using the VPPEN signal in Figure 1. The VPPEN signal should be controlled by an output port from the C186.

## Commands

1. Read Array/Reset: This mode enables the Flash to be read from the microprocessor. The device enters this mode either by issuing the Read Array command (0FFH), after device power up or after exiting power down.

2. Inteligent Identifier: This mode is used to identify the type of Flash to the system that will program it. Issuing the identifier command (90H) followed by a read from address 00000H and 00001H will return the manufacturer and device code respectively (89H Intel, 94H 28F001BX-T). A valid command needs to be written afterward to exit the operation; for example Read Array (0FFH).

3. Read Status Register: Writing the read status command (70H) allows subsequent reads to the status register. Only after issuing another command will it stop reading from the status register.

   **Definition of the Status Register:**
   Bit 3 $V_{PP}$ status
   1= $V_{PP}$ is low and the operation was aborted. This bit must be cleared before programming/erasing can be reattempted.

0= $V_{PP}$ is valid.

Bit 4 Program status
1= Error during byte programming.
0= Successful byte program.

Bit 5 Erase status
1= Error in block erasure. When both Bit 5 and Bit 4 are set, then an improper erase command sequence was entered.
0= Successful block erase.

Bit 6 Erase Suspend Status
1= Erase Suspended.
0= Erase in Progress/Completed.

Bit 7 Write State Machine Status
1= Ready.
0= Busy.

4. Clear Status Register: Errors reported by the status register may only be cleared by writing this command (50H).

5. Erase Setup/Erase Confirm: This command is used to erase one block at a time. Two command codes are consecutively written to begin the erasure (20H, D0H), eliminating accidental erasure. Any address in the block to be erased must be used when writing the commands.

6. Erase Suspend/Erase Resume: During the erasure of a block this command allows the process to be suspended by writing the command (B0H). After writing the command it is necessary to check the Erase Suspend Status bit to verify erase is suspended. To resume erasing write the command (D0H).

7. Program Setup/Program: Two commands are written to the command register. First, the command (40H) is written to the address of the location to be programmed (Byte address). Second, the data is written to the byte address.

## Programming Algorithm

### Block Erasure

To erase a block, first write the command 20H to an address in the block. Next confirm the erasure by writing D0H to the block. Because the command register does not occupy an addressable memory location, the only requirement is the written address be within the block. Once written, wait until the Write State Machine status bit is ready. If this wait is too long it may be necessary to suspend the erasure and check the status and memory. When the erasure is finished the Erase, Program, and $V_{PP}$ status bits need to be checked for errors. If no errors were detected then the block was successfully erased. The algorithm for erasing a block is summarized in Flowchart 1.

### Suspend/Resume Block Erasure

To suspend block erasure, write the command B0H to the block address. Next, check the Erase Suspend Status bit until it is Ready or set. The status bit needs to be checked because the Flash will not suspend its erasure until it reaches a checkpoint in its program. The Read Array command (0FFH) is written to return the device into the read mode. It is then possible to read the memory. When finished reading the memory, writing the command (D0H) to any address within the block will resume the erasure.

The algorithm for suspending and resuming an erasure is summarized by Flowchart 2. If an error is detected by the status register, the status register must be cleared before attempting the erasure.

### Byte Programming

Programming the Flash starts with writing the Program Setup command (040H) to the address of the byte to be programmed. Next, write the data to the same address. When done, check the Program and $V_{PP}$ status bits for errors. If no errors were detected then the byte was successfully programmed. The algorithm for programming a byte is summarized by Flowchart 3.

### Programming Tips

When programming Flash, the location to be programmed must be erased first. To erase the location the entire block will need to be erased. A byte is erased when all 8 bits are set (0FFH). But in reality it is possible to program a byte even if the bits are not all set. The program only needs to check that all of the source bits that are set are also set at the destination. This is useful if programming only a specific location and not a range of locations. This can be achieved by ANDing the Source with the compliment of the Destination:

$$E = \overline{D} \land S$$

Where: S)ource = Byte to be programmed
    D)estination = Current byte value at the memory location
    E)rase = If erase is equal to zero then the location can be programmed with the new byte otherwise the block will need to be erased.

It is also possible to program both the hi and low Flash devices at the same time. Commands need to be given in both the hi and low byte of the word. If there are interrupt service routines that may access the Flash memory then it is important to disable interrupts while programming or erasing the device. Thanks to Flash automation if the interrupt service routine does not access the Flash then there is no need to disable the interupt while programming/erasing.

Flash memory devices may program or erase at different rates. Therefore when programming or erasing both Flash devices simultaneously it is important to verify both.

### Programming Routine

### Source Code For Programming Routine

```
$TITLE ('Flash Memory Programming Routines for a 80C186 Based System')
$MOD186

NAME            FLASH_ROUTINES
EXTRN           VPP_CONTROL:FAR
```

```
;***************************************************************************
;Operating Software for Intel Flash 28F001BX-T
;(c) Copyright 1988  Intel Corporation.  All Rights Reserved.
;
;Refer to the article INTERFACING THE INTEL FLASH 28F001BX-T TO YOUR
;186 BASED SYSTEM for operation details.
;
; Version               Date                    Comments
;  V1.0      03/09/1992       Initial Design
;
;Design Information
;
; 1) This software is tailored for a 186 based system but byte routines
;    could be used with the 80C188 family.  There are routines for
;    programming by byte or by word.  This allows users to program one
;    flash memory at a time or both.
;
; 2) When programming or erasing Flash a 12 volt supply is needed on the VPP
;    signal.  A circuit for switching this on and off is provided in the
;    article INTERFACING THE INTEL FLASH 28F001BX-T TO YOUR 186 BASED
;    SYSTEM.  An I/O port of some sort will need to be used to address
;    the switching of Vpp on and off.  Since the software for addressing
;    Vpp enable is directly related to the board design, it is left to
;    the user to incorporate the external VPP_CONTROL software.  Keep in
;    mind that a delay is needed for the 12 volts to settle once turned on.
;
;Program Information
;
; 1) Most commands once finished, automatically output status information
;    when read.   This is valid for Block Erase, Program, & Erase Suspend.
;    To terminate the status operation, it is necessary to write another
;    valid command or if unknown Read Array/Reset may be used.
;
; 2) Procedure Check_Byte and Check_Word are using the following Algorithm:
;
;                 E = D̄ /\ S
;
;    Where:  S)ource      = Byte to be programmed
;            D)estination = Current byte value at the memory location
;            E)rase       = If erase is equal to zero then the location
;                           can be programmed with the new byte otherwise
;                           the block will need to be erased.
;
;***************************************************************************
READY_BIT               EQU     080H     ;Ready bit in the status register
SUSPEND_BIT             EQU      040H    ;Suspend bit in the status register
ADR0                    EQU     0000H    ;Manufacturers ID offset within the Flash
ADR1                    EQU     0001H    ;Device code ID offset within the Flash
RD_ID_BCMD              EQU      090H    ;Read Inteligent Identifier Byte Command
RD_ID_WCMD              EQU      09090H  ;Read Inteligent Identifier Word Command
RD_AR_BCMD              EQU      0FFH    ;Read Array/Reset Byte Command
RD_AR_WCMD              EQU      0FFFFH  ;Read Array/Reset Word Command
ER_SP_BCMD              EQU      020H    ;Erase Setup Byte Command
ER_SP_WCMD              EQU      02020H  ;Erase Setup Word Command
ER_CM_BCMD              EQU      0D0H    ;Erase Confirm Byte Command
ER_CM_WCMD              EQU      0D0D0H  ;Erase Confirm Word Command
```

```
ER_SU_BCMD              EQU     0B0H    ;Erase Suspend Byte Command
ER_SU_WCMD              EQU     0B0B0H  ;Erase Suspend Word Command
ER_RE_BCMD              EQU     0D0H    ;Erase Resume Byte Command
ER_RE_WCMD              EQU     0D0D0H  ;Erase Resume Word Command
RD_ST_BCMD              EQU     070H    ;Read Status Register Byte Command
RD_ST_WCMD              EQU     07070H  ;Read Status Register Word Command
PM_SP_BCMD              EQU     040H    ;Program Setup Byte Command
PM_SP_WCMD                      EQU      04040H         ;Program Setup Word Command
CL_ST_BCMD                      EQU      050H    ;Clear Status Byte Command
PM_ERROR_BMSK                   EQU     018H    ;Program Error Byte Mask
PM_ERROR_WMSK                   EQU     01818H          ;Program Error Word Mask
ER_ERROR_BMSK                   EQU     038H    ;Erase Error Byte Mask
ER_ERROR_WMSK                   EQU     03838H          ;Erase Error Word Mask
BL_ERROR_BMSK                   EQU     020H    ;Erase Error Bit
PG_ERROR_BMSK                   EQU     010H    ;Program Error Bit
VP_ERROR_BMSK                   EQU     008H    ;Vpp Error Bit
SE_ERROR_BMSK                   EQU     030H    ;Sequence Error Bits

FLASHC0                 SEGMENT AT 0C000H
                ORG     0000H
MAINC0                  DB      ?
FLASHD0                 SEGMENT         AT 0D000H
                ORG     0000H
MAIND0                  DB      ?
FLASHE0                 SEGMENT AT 0E000H
                ORG     0000H
MAINE0                  DB      ?
FLASHF0                 SEGMENT AT 0F000
                ORG     08000
PARAM1                  DB      ?
                ORG     0A000
PARAM2                  DB      ?
                ORG     0C000
BOOT                    DB      ?

CODE                    SEGMENT

                ASSUME          CS:CODE, ES:FLASHC0
```

```
;*****************************************************************************
;READ_ID - Read the Inteligent Identifier
;
;Inputs
;        ES:DI - Flash Device Base Address
;Outputs
;        AX - Manufacturer Code
;        BX - Device Code
;*****************************************************************************

                 PUBLIC          READ_ID
READ_ID                 PROC             FAR
                 MOV     DI, ADR0                          ;Write the Read ID
                 MOV     WORD PTR ES:[DI],RD_ID_WCMD       ;command to 00000H
                 MOV     AX, WORD PTR ES:[DI]              ;Read the Manu ID
                 MOV     DI, ADR1                          ;Read the Device ID
                 MOV     BX, WORD PTR ES:[DI]              ;from address 00001H
                 MOV     WORD PTR ES:[DI],RD_AR_WCMD       ;Terminate the
                 RET                                       ;command with the
READ_ID                 ENDP                               ;reset command

;*****************************************************************************
;ERASE_BLK_BYTE - Erase a block of bytes
;
;Inputs
;        ES:DI - Block Base Address
;Outputs
;        AL - Error Codes         00H - Successful Erasure
;                                 08H - Vpp Error
;                                 20H - Block Erase Error
;                                 30H - Command Sequence Error
;*****************************************************************************
```

```
                PUBLIC          ERASE_BLK_BYTE
ERASE_BLK_BYTE                  PROC            FAR
                MOV     BYTE PTR ES:[DI],ER_SP_BCMD             ;Write the Erase setup
                MOV     BYTE PTR ES:[DI],ER_CM_BCMD             ;& confirm commands
ERASING_BYTE:           MOV     AL, BYTE PTR ES:[DI]            ;Check status Ready bit
                TEST    AL, READY_BIT                           ;Complete when MSB
                JZ      ERASING_BYTE                            ;is set
                AND     AX, ER_ERROR_BMSK                       ;Error codes in AX
                MOV     BYTE PTR ES:[DI], RD_AR_BCMD            ;
                RET
ERASE_BLK_BYTE                  ENDP

;*****************************************************************************
;ERASE_BLK_WORD - Erase a block of memory on two Flash memories
;
;Input
;       ES:DI - Block Base Address
;Output
;       AX - Error Codes        AH/AL
;                                00 - Successful Erasure
;                          08 - Vpp Error
;                                20 - Block Erase Error
;                                30 - Command Sequence Error
;*****************************************************************************

                PUBLIC          ERASE_BLK_WORD
ERASE_BLK_WORD                  PROC            FAR
                MOV     WORD PTR ES:[DI], ER_SP_WCMD            ;Write the Erase setup
               MOV     WORD PTR ES:[DI], ER_CM_WCMD             ;& confirm commands
ERASING_WORD:           MOV    AX, WORD PTR ES:[DI]             ;Read status bytes
                TEST    AL,AH                                   ;Check the Ready bit
                JNS     ERASING_WORD                            ;to see when done
                AND     AX, ER_ERROR_WMSK                       ;Error codes in AX
                MOV     WORD PTR ES:[DI], RD_AR_WCMD
                RET
ERASE_BLK_WORD                  ENDP
```

```
;******************************************************************************
;ERASE_SUS_BYTE - Suspend the byte erase sequence
;
;Inputs
;        ES:DI - Block Base Address
;Outputs
;        None
;******************************************************************************

                 PUBLIC          ERASE_SUS_BYTE
ERASE_SUS_BYTE                   PROC            FAR
                 MOV     BYTE PTR ES:[DI], ER_SU_BCMD             ;Write Suspend Command
SUSPEND_BYTE:            MOV     AL, BYTE PTR ES:[DI]             ;Check status if
                 TEST    AL, SUSPEND_BIT                          ;Erase has been
                 JNE     SUSPEND_BYTE                             ;suspended
                 MOV     BYTE PTR ES:[DI], RD_AR_BCMD             ;Write Read Command
                 ;Insert Flash Read code                          ;Read Flash Memory
                 MOV     BYTE PTR ES:[DI], ER_RE_BCMD             ;Resume Erase
                 RET
ERASE_SUS_BYTE                   ENDP

;******************************************************************************
;ERASE_SUS_WORD - Suspend the word erase sequence
;
;Inputs
;        ES:DI - Block Base Address
;Outputs
;        None
;******************************************************************************

                 PUBLIC          ERASE_SUS_WORD
ERASE_SUS_WORD                   PROC            FAR
                 MOV     WORD PTR ES:[DI], ER_SU_WCMD             ;Write Suspend Command
SUSPEND_WORD:                    MOV     AX, WORD PTR ES:[DI]     ;Check status if
                 AND     AL,AH                                    ;Erase has been
                 TEST    AL, SUSPEND_BIT                          ;suspended
                 JNE     SUSPEND_WORD
                 MOV     WORD PTR ES:[DI], RD_AR_WCMD             ;Write Read Command
                 ;Insert Flash Read code                          ;Read Flash Memory
                 MOV     WORD PTR ES:[DI], ER_RE_WCMD             ;Resume Erase
                 RET
ERASE_SUS_WORD                   ENDP

;******************************************************************************
;PROG_BLK_BYTE - Program an odd or even block of bytes
;
;Inputs
;        ES:DI - Destination Block Base Address
;        DS:SI - Source Block Base Address
;        CX    - Number of Bytes
;Outputs
;        AL    - Error Codes      00H - Successful Program
;                                 04H - Vpp Range Error
;                                 10H - Byte Program Error
;******************************************************************************

                 PUBLIC          PROG_BLK_BYTE
PROG_BLK_BYTE            PROC            FAR
NEXT_BYTE:               MOV     BYTE PTR ES:[DI], PM_SP_BCMD     ;Write Program
Command
                 MOVS            BYTE PTR ES:[DI], BYTE PTR DS:[SI];Program Byte
                 INC     DI                                       ;Odd or Even bytes only
WRITING_BYTE:            MOV     AL, BYTE PTR ES:[DI]             ;Check status if
                 TEST    AL, READY_BIT                            ;byte program is
                 JZ      WRITING_BYTE                             ;finished.
                 AND     AX, PM_ERROR_BMSK                        ;Save errors in AX
                 JNZ     PBB_EXIT                                 ;and get out if any.
                 DEC     CX                                       ;Count down bytes
                 JNE     NEXT_BYTE
PBB_EXIT:        MOV     BYTE PTR ES:[DI], RD_AR_BCMD             ;Write Read Command
                 RET
PROG_BLK_BYTE            ENDP
```

```
;****************************************************************************
;PROG_BLK_WORD - Program a block of Words
;
;Inputs
;        ES:DI - Destination Block Base Address
;        DS:SI - Source Block Base Address
;        CX    - Number of Words
;Outputs
;        AX    - Error Codes              AH/AL
;                                  00H - Successful Program
;                                  04H - Vpp Range Error
;                                  10H - Byte Program Error
;****************************************************************************

                 PUBLIC          PROG_BLK_WORD
PROG_BLK_WORD                    PROC          FAR
NEXT_WORD:      MOV     WORD PTR ES:[DI], PM_SP_WCMD              ;Write Program
Command
                 MOVS            WORD PTR ES:[DI], WORD PTR DS:[SI];Program Word
WRITING_WORD:   MOV     AX, WORD PTR ES:[DI]                      ;Check status if
                 TEST    AL,AH                                    ;the word programmed.
                 JNS     WRITING_WORD
                 AND     AX, PM_ERROR_WMSK                        ;Save error in AX
                 JNZ     PBW_EXIT                                 ;and get out if any
                 DEC     CX                                       ;Count down words
                 JNE     NEXT_WORD
PBW_EXIT:        MOV     WORD PTR ES:[DI], RD_AR_WCMD              ;Write Read Command
                 RET
PROG_BLK_WORD                    ENDP

;****************************************************************************
;CHECK_BYTE - Check if the source byte can be programmed at the destination
;              without erasing the block.
;
```

```
;Inputs
;        ES:DI - Destination Block Base Address
;        BL    - Source Byte
;Outputs -
;        AL    - Erase Flag       0        -  no need to erase block
;                                 Nonzero  -  erase the block
;*****************************************************************************

                 PUBLIC          CHECK_BYTE
CHECK_BYTE               PROC            FAR
                 MOV     BYTE PTR ES:[DI], RD_AR_BCMD              ;Write Read Command
                 MOV     AL, BYTE PTR ES:[DI]                      ;Read Destination Byte
                 NOT     AL                                        ;Complement Dest Byte
                 AND     AL,BL                                     ;Perform logical AND
                 RET                                               ;function with Dest
CHECK_BYTE               ENDP                                      ;and Source bytes.

;*****************************************************************************
;CHECK_WORD - Check if the source word can be programmed at the destination
;              without erasing the block.
;
;Inputs
;        ES:DI - Destination Block Base Address
;        BX    - Source Word
;Outputs -
;        AX    - Erase Flag       0        -  no need to erase block
;                                 Nonzero  -  erase the block
;*****************************************************************************

                 PUBLIC          CHECK_WORD
CHECK_WORD               PROC            FAR
                 MOV     WORD PTR ES:[DI], RD_AR_WCMD              ;Write Read Command
                 MOV     AX, WORD PTR ES:[DI]                      ;Read Destination Word
                 NOT     AX                                        ;Complement Dest Word
                 AND     AX,BX                                     ;Perform logical AND
                 RET                                               ;function with Dest
CHECK_WORD               ENDP                                      ;and Source Words.
```

```
;*****************************************************************************
;PROG_BYTE - Program one byte
;
;Inputs
;       ES:DI - Destination Base Address
;       BL    - Source Byte
;Output
;       AL    - Error Code       00H - Successful Program
;                                08H - Vpp Range Error
;                                10H - Byte Program Error
;*****************************************************************************

                PUBLIC          PROG_BYTE
PROG_BYTE               PROC            FAR
                MOV     BYTE PTR ES:[DI], PM_SP_BCMD            ;Write Program command
                MOV     BYTE PTR ES:[DI], BL                    ;Program BL into Flash
WRITE_BYTE:             MOV     AL, BYTE PTR ES:[DI]            ;Check status Ready bit
                TEST    AL, READY_BIT                           ;if done programming.
                JZ      WRITE_BYTE
                AND     AX, PM_ERROR_BMSK                       ;Save any errors
                MOV     BYTE PTR ES:[DI], RD_AR_BCMD            ;Write Read command
                RET
PROG_BYTE               ENDP

;*****************************************************************************
;PROG_WORD - Program one word
;
;Inputs
;       ES:DI - Destination Base Address
;       BX    - Source Word
;Output
;       AX    - Error Code       AH/AL
;                                 00H - Successful Program
;                                 08H - Vpp Range Error
;                                 10H - Byte Program Error
;*****************************************************************************

                PUBLIC          PROG_WORD
PROG_WORD               PROC            FAR
                MOV     WORD PTR ES:[DI], PM_SP_WCMD            ;Write Program command
                MOV     WORD PTR ES:[DI], BX                    ;Program BX into Flash
WRITE_WORD:             MOV     AX, WORD PTR ES:[DI]            ;Check status Ready bit
                TEST    AL,AH                                   ;if done programming
                JNS     WRITE_WORD
                AND     AX, PM_ERROR_WMSK                       ;Save any errors
                MOV     WORD PTR ES:[DI], RD_AR_WCMD            ;Write Read command
                RET
PROG_WORD               ENDP

;*****************************************************************************
;ERROR_BYTE - Check for errors and Clear Status Register
;
;Input
;       AL - Error Codes         08H - Vpp Error
;                                10H - Byte Program Error
;                                20H - Block Erase Error
;                                30H - Command Sequence Error
;Output
;       None
;*****************************************************************************
```

```
                 PUBLIC          ERROR_BYTE
ERROR_BYTE               PROC            FAR
                 TEST    AL, BL_ERROR_BMSK                       ;Check for Block Erase
                 JE      BL_ERROR_HANDLER                        ;Error?
                 TEST    AL, PG_ERROR_BMSK                       ;Check for Program
                 JE      PG_ERROR_HANDLER                        ;Error?
                 TEST    AL, VP_ERROR_BMSK                       ;Check for Vpp
                 JE      VP_ERROR_BMSK                           ;Error?
                 AND     AL, SE_ERROR_BMSK                       ;Check for Command
                 CMP     AL, SE_ERROR_BMSK                       ;Sequence Error?
                 JE      SE_ERROR_HANDLER
                 JMP     EXIT_ERROR
BL_ER_HANDLER:                   ;Insert Block Erase Error Handler
PG_ER_HANDLER:                   ;Insert Program Error Handler
VP_ER_HANDLER:                   ;Insert Vpp Error Handler
SE_ER_HANDLER:                   ;Insert Command Sequence Error Handler
                 MOV     BYTE PTR ES:[DI], CL_ST_BCMD            ;Clear Status Register.
EXIT_ERROR:              RET
ERROR_BYTE               ENDP

CODE                     ENDS

                 END
```