SIEMENS

Microcontrollers

ApNote AP1653

☑ : Additional file
AP165301.EXE available

Flash Protection on 88C166 Flash Devices

Siemens 88C166 16-bit microcontrollers provide 32 KBytes Flash memory on-chip, which can be protected against data operand acceses and program branches into the Flash from any location outside the Flash. This Application Note gives hints and examples how the Flash protection on 88C166 Flash devices can be enabled and how the protection can be temporary disabled (e.g. for program updates) by a program running inside the Flash.

Author : Peter Kliegelhöfer / HL DC AT Microcontroller Application Support

Contents **Page**

AP1653 ApNote - Revision History		
Actual Revision : 07.98	Previous Revision : none (Original version)	
Page of actual Rev.	Page of prev.Rel.	

1 Introduction

Siemens 88C166 Flash devices provide 32 KBytes Flash memory on-chip, which can be protected against data operand acceses and program branches into the Flash from any location outside the Flash. This Application Note gives hints and examples how the Flash protection can be enabled and how the protection can be disabled by a program running inside the Flash.
In the appendix all the sources of the used test pattern are listed, which are part of the additional self-extracting zip-file “AP1652xx.EXE”. As evaluation hardware, an Ertec EVA166 evaluation board and some additional hardware was used.
The test environment makes use of the internal boostrap loader which is part of the 88C166 boot-ROM.
Devices with the following marking text have been tested: SAB88C166-5Q ES-CA, SAB88C166-5M ES-CA, SAB88C166-5M V59 and SAB88C166-5M ES-CC.
For testing basically two routines are used:
1. PROT166
This routine has to be loaded into and executed out of external (on the evaluation board: RAM) memory and *activates* the Flash protection by programming a special (unerasable) bit “UPROG”. This routine just has an effect when the controller is in a special mode (Test Mode).
2. UFP166
This routine controls the success of Flash protection *deactivation*. It has to be programmed into the internal Flash memory (e.g. via the Flash/ OTP memory programming tool “Memtool” which is part of this Application Note). The routine temporary unlocks the (previously protected) Flash and displays the Flash contents on a dedicated 16-bit port. On the test board the actual status of program execution or the contents of registers or memory can be controlled via an LED line which is connected to PORT2.

2 Flash Protection

If active, Flash protection prevents data operand accesses and program branches into the on-chip Flash area from any location outside the Flash memory itself. Data operand accesses and branches to Flash locations are exclusively allowed for instructions executed from the Flash memory itself. Erasing and programming is not possible while Flash protection is active.

Flash protection is controlled by two different bits:

- The user-accessible write-only Protection Activation bit (RPROT) in register FCR and
- The one-time-programmable Protection Enable bit (UPROG).

Bit **UPROG** is a ’hidden’ one-time-programmable bit only accessible in a special mode (Test Mode, see Section 3). Once programmed to ’1’, this bit is unerasable ie. it is not affected by the Flash erase mechanism.

To activate Flash Protection bit UPROG must have been programmed to ’1’, and bit RPROT in register FCR must be set to ’1’. Both bits must be ’1’ to activate Flash protection.

To deactivate Flash Protection bit RPROT in register FCR must be cleared to ’0’. If any of the two bits (UPROG or RPROT) is ’0’, Flash protection is deactivated.

Generally Flash protection will remain active all the time. If it has to be deactivated intermittently, eg. to call an external routine or to reprogram the Flash memory, bit RPROT must be cleared to '0'.
To access bit **RPROT** in register FCR, an instruction with a 'mem, reg' addressing mode must be used, where the first operand has to represent the FCR address (any even address within the active address space of the Flash memory) and the second operand must refer to a value which sets the RPROT bit to '0', e.g.:

```
MOV     FCR, ZEROS              ; Deactivate Flash Protection
```

RPROT is the only bit in the FCR which can be accessed in Flash standard mode without having to enter the Flash writing mode. Other bits in the FCR are not affected by such a write operation. However, this access requires an instruction executed out of the internal Flash memory itself.

After reset bit RPROT is set to '1'. For devices with protection disabled (UPROG = '0') this has no effect. For devices with protection enabled this ensures that program execution starts with Flash protection active from the beginning.

In order to maintain uninterrupted Flash protection, be sure not to clear bit RPROT unintentionally by FCR write operations. Otherwise the Flash protection is deactivated.

3 Test Mode Selection

The one-time-programmable Protection Enable bit UPROG is only accessible in a special mode, in the following called "Test Mode". Test Mode selection in entered when:

- a state transition occurs on pin EBC0 in a defined time window after the rising edge of the $\overline{\text{RSTIN}}$ signal

The transition on EBC0 must be within

- 2 XTAL1 clock periods after the rising edge of $\overline{\text{RSTIN}}$ (50ns @ 40 MHz) for SAB88C166
- 1 XTAL1 clock period after the rising edge of $\overline{\text{RSTIN}}$ (50ns @ 20 MHz) for SAB88C166W

However, there must be a short delay between $\overline{\text{RSTIN}}$ rising edge and EBC0 transition. On the evaluation board the additional logic controlling EBC0 uses $\overline{\text{RSTIN}}$ as control signal for the EBC0 transitions. Tests have shown that, on the evaluation board, the transition on the EBC0 signal may practically be coincident with the $\overline{\text{RSTIN}}$ rising edge: it follows this edge within only 2-3ns (the commutation time of a few logic gates). Therefore e.g. two inverters are applicable for delivering the desired delay.
The selection of the Test Mode is not dependent on the static value of EBC1, but only on the application of a pulsed value on the EBC0 pin across the $\overline{\text{RSTIN}}$ rising edge. The EBC1 static value and the EBC0 pulsed value are selectable according to the desired external bus configuration. The final level of EBC0 is relevant for the external bus configuration.
On the test board, in order to use a 16 bit non-multiplexed data bus, Test Mode is selected by applying EBC1 static at '1' and EBC0 with a rising edge towards "1" after the $\overline{\text{RSTIN}}$ rising edge. If there are different needs in another test environment about the external bus configuration, one must

change the EBC1 static value and the polarity of the EBC0 pulse according to the 88C166 external bus specification.

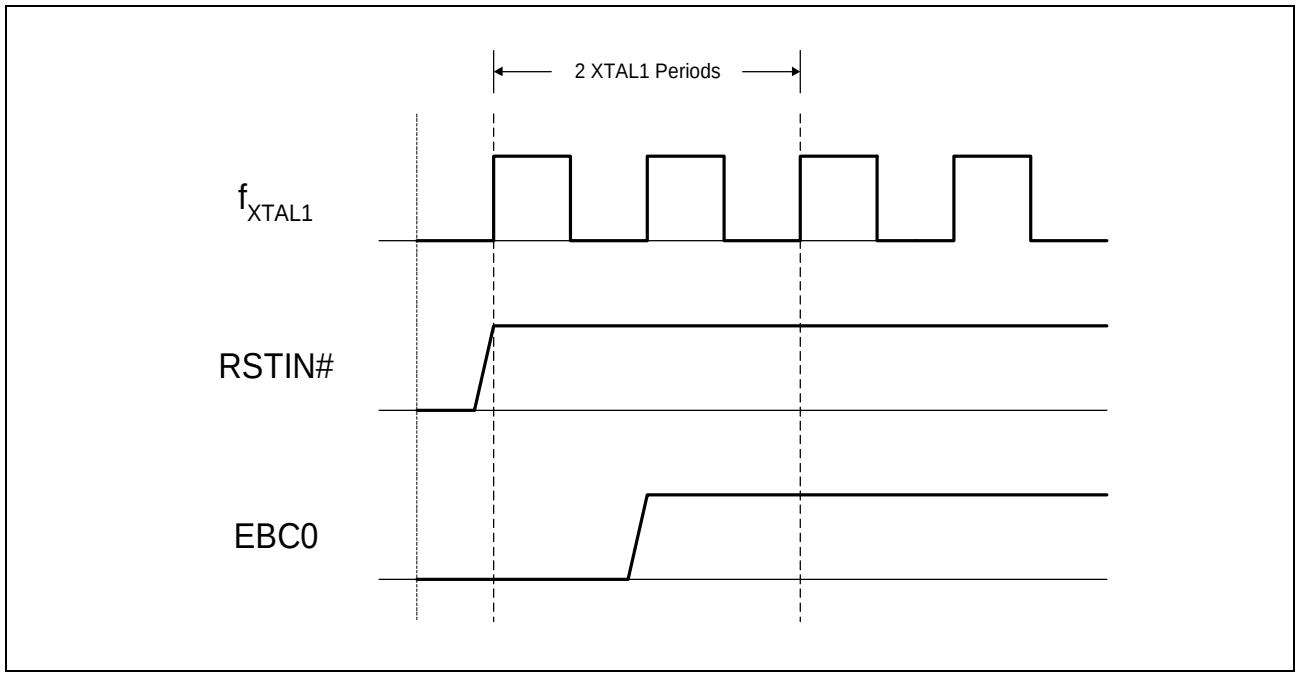

Figure 2
Test Mode selection via state transition at EBC0 input during RESET

Test Mode Selection:

EBC1	EBC0	ALE
S	T	0

S = static value; depends on the external bus configuration
T = state transition at EBC0 input during RESET (either 0->1 or 1->0, final value depending on target bus type)

Instruction/ data sources during Test Mode:

Instruction Fetch	Operand Fetch
external memory	internal Flash memory

When Test Mode is selected at the reset exit, then the instruction fetch is performed from external memory (on the evaluation board: RAM), while the internal Flash is seen as data space.

4 Test Environment

Hardware overview

Ertec EVA166 evaluation board with the following configuration:

- 16-bit non-multiplexed data bus
- external RAM
- no external EPROM/ ROM monitor (sockets D1/ D3 left open)
- 20 MHz oscillator
- P-MQFP-100 package adaptor for the 88C166 controller
- switch S0 in position “user”

Additional hardware:

- reset circuit
- (delay) logic
- NMI push button
- bootstrap loader activation circuit
- LED line (16 LED’s) at PORT2

Software overview

- Ertec monitor software “MON16X2” (Windows NT) with modified file “boot.166”
- test routine “UFP”
- test routine “PROT166”

Hardware/ Software description

The P-MQFP-100 adaptor has to be modified: controller pin $\overline{\text{BUSACT}}$ shall be connectable either to Vcc or to GND. This makes it possible
a) to activate the bootstrap loader also on CA-step devices (activation needs $\overline{\text{BUSACT}}$ “high” during reset)
b) to run a user program out of the internal Flash (all steps).

On the evaluation board an additional logic generating EBC0 uses $\overline{\text{RSTIN}}$ as control signal to generate the EBC0 transitions. The used logic consists of two inverters which simply delay the reset signal. To meet the necessary timing for Test Mode selection a reset signal with fast falling and - important - fast rising edges is needed. Because the R/C reset circuit on EVA166 does not deliver the needed signal form, an external reset circuit has to be used.

In significant program parts the test software drives the controller into IDLE mode and the actual status of program execution or the contents of registers or memory are displayed on an LED line which is connected to PORT2. To allow step-by-step program execution, a NMI push button is provided to wake-up the controller from IDLE mode for continuing program execution. The NMI

circuit has to be disconnectable from pin $\overline{\text{NMI}}$ (e.g. by jumper). This is necessary for bootstrap loader activation (see next chapter).

The bootstrap loader is used to load the Ertec monitor program MON16X2, which in turn loads test pattern “PROT” into external RAM via serial interface ASC0. The boostrap loader can be activated/ deactivated by jumper. On the evaluation board external EPROM (and with this the Ertec ROM monitor) is not used.

File “boot.166”, which belongs to the Ertec monitor, has been modified: the “EINIT” instruction is substituted by two “NOP” instructions. Another modification in file “boot.166” makes it possible to set register SYSCON in the configuration file “MON16X.CFG”.

Examples:

```
SYSCON=004Eh        ; BUSACT=0 BTYP=01 (ROM enabled & mapped to segment 1), 1WS
SYSCON=04CDh          ; BUSACT=1 BTYP=11 (16 Bit NonMUX), 2 WS
```

In case of design step CC these modifications make it possible to read out the content of the internal 88C166 Flash memory with the Ertec monitor on an EVA166 evaluation board. The modified file “boot.166” is part of the application software.

Test program “UFP166” has to be programmed into the internal Flash memory (e.g. by using the Flash/ OTP in-system memory programming tool “Memtool”). This routine unlocks the (previously locked) Flash memory and copies another routine into the internal RAM. This second routine starts automatically after loading, reads out the contents of the (now temporarily unlocked) Flash memory and displays the result on PORT2 for control.

Test program “PROT166” has to be loaded via ASC0 into external RAM by using the Ertec monitor. The controller has to be in Test Mode when starting PROT166. During program execution the user is requested to apply Vpp at pin EBC1 by a dedicated pattern at PORT2. When the controller is not in Test Mode or Vpp is not correctly applied, the protection routine has no effect.

5 Miscellaneous

5.1 Bootstrap Loader Activation

According to Application Note “Bootstrap Loader 8xC166” (advance information 9.94) a little circuit for activation of the 88C166 on-chip bootstrap loader is required (BSL active, when jumper JP1 and JP2 are closed). Some device steps need in addition pin $\overline{\text{BUSACT}}$ pulled up “High” (jumper JP3 closed) during reset for BSL activation. Please refer to the dedicated errata sheet of the used device step. The connection of pin $\overline{\text{NMI}}$ to the NMI circuit must be open (see 5.2). Note that there is already a 10K pull-up resistor at the $\overline{\text{NMI}}$ pin on the Ertec EVA166 board.

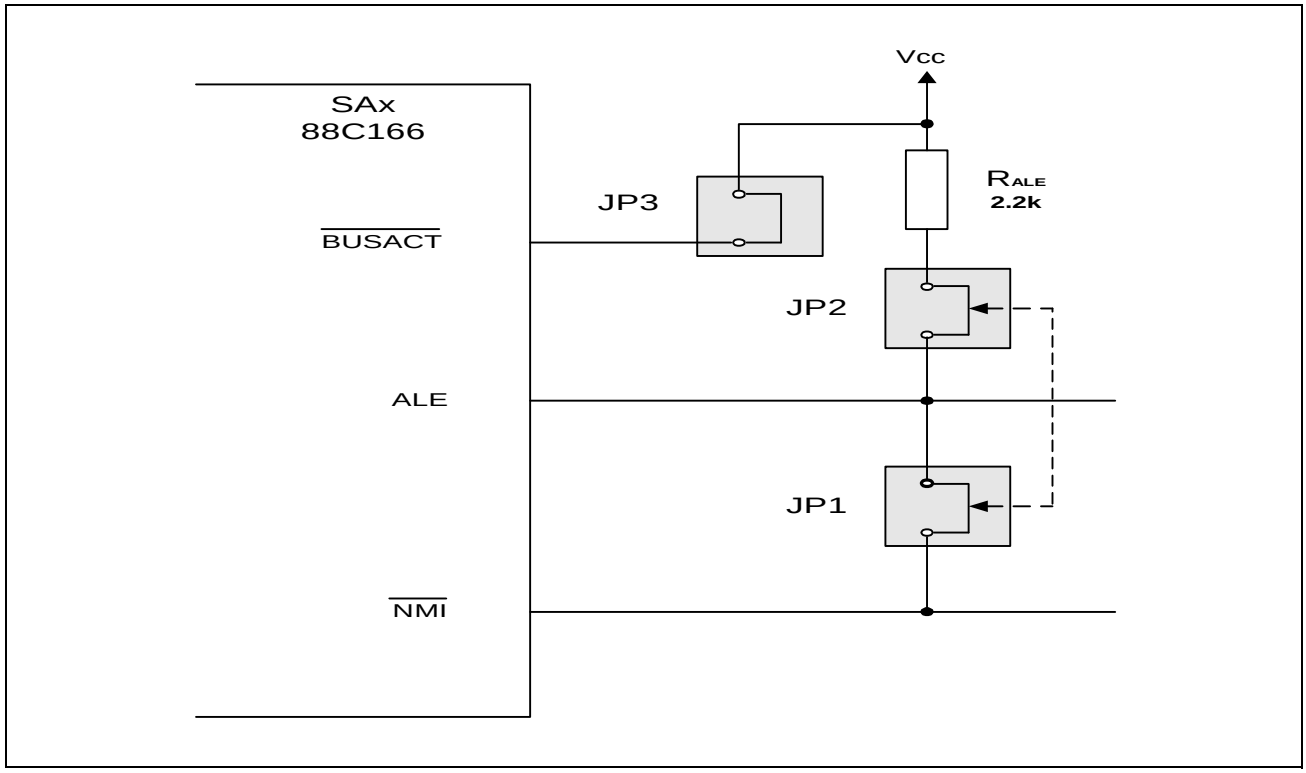

Figure 3
Exemplary circuit diagram for BSL entry

The bootstrap loader is used to load the monitor program via ASC0. The monitor program itself is used to load test pattern “PROT”. When the test pattern is properly loaded (location: external RAM) jumper JP1, JP2 and JP3 have to be disconnected in order to allow user program execution out of external (on the test board: RAM) memory after reset.

5.2 NMI Circuit

In the following a suggestion for the NMI circuit is given. The capacitor and resistor R2 are provided for debouncing of the push button. Note that a pull-up resistor at pin $\overline{\text{NMI}}$ (R1) is already provided on the Ertec EVA166 evaluation board.

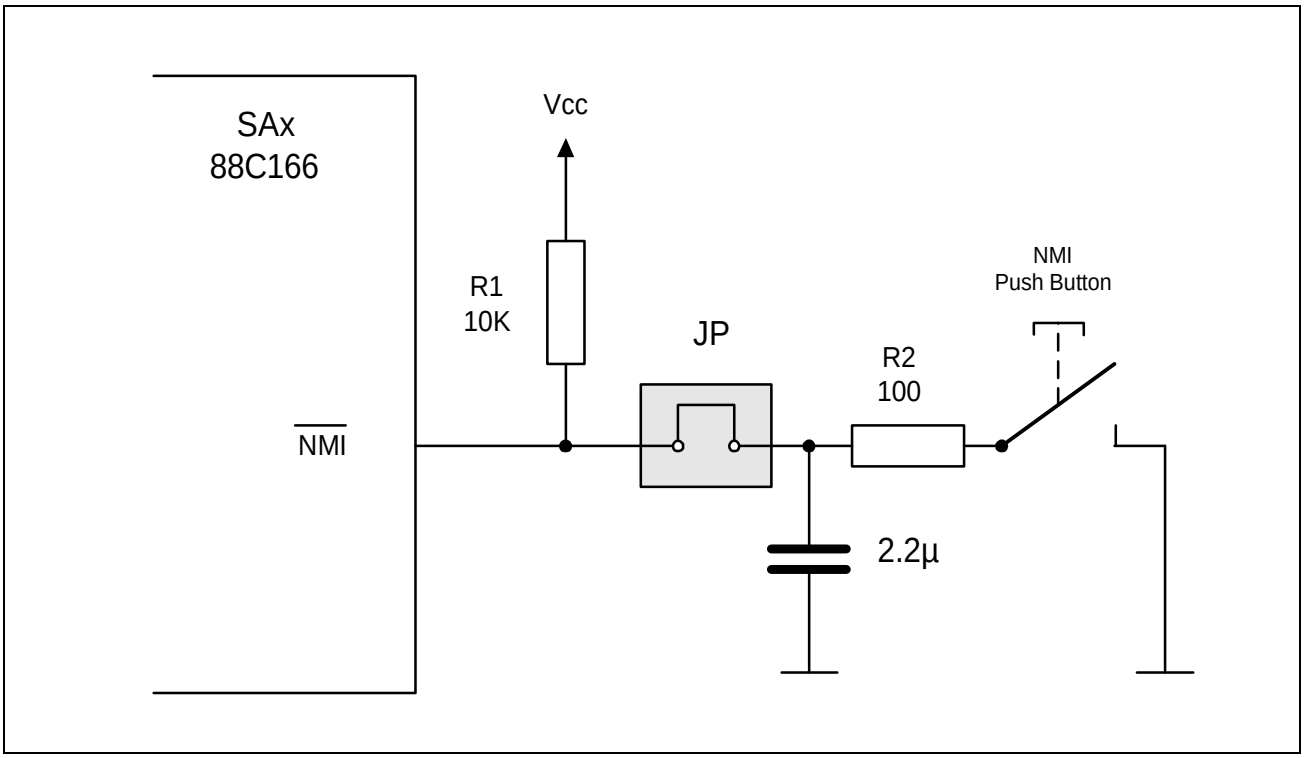

Figure 4
NMI circuit

5.3 Delay Logic

On the test board two inverters deliver the needed EBC0 transitions for Test Mode selection. The logic uses $\overline{\text{RSTIN}}$ as control signal. With jumper JP the circuit can be disconnected from EBC0 for configuring the bus via EBC0 and EBC1 pins.

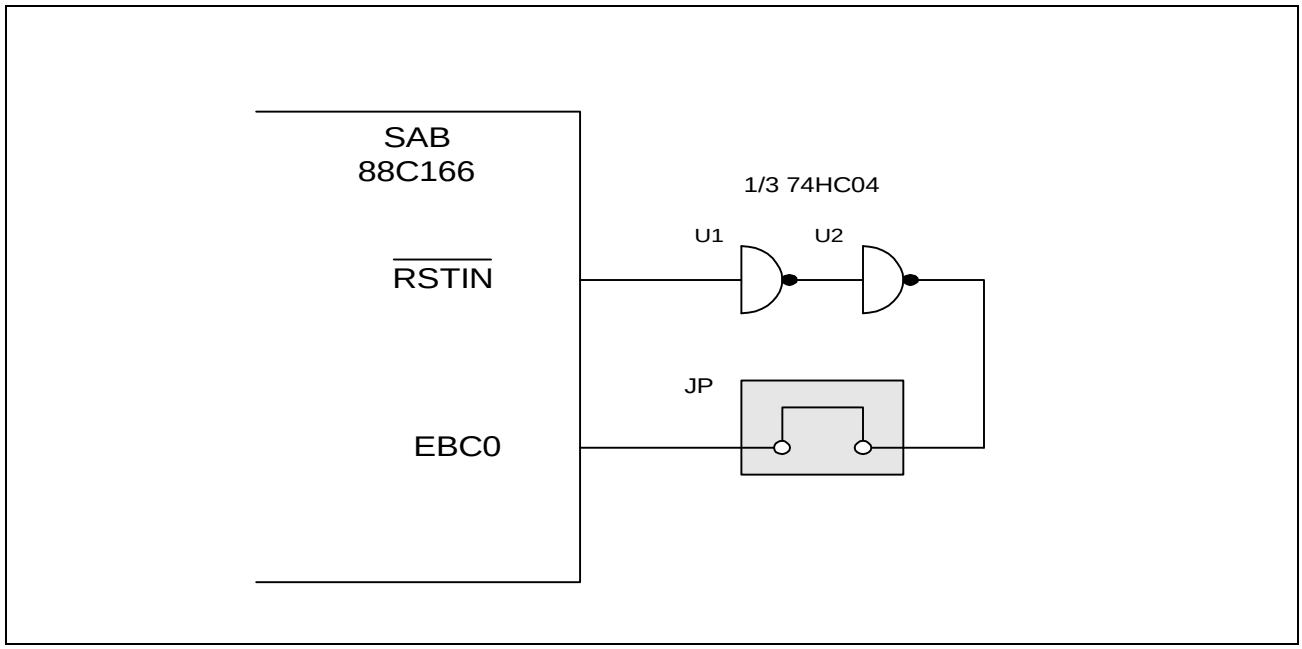

Figure 5
Delay Logic

5.4 Testing

Testing has to be performed in three steps:

- In the first step test pattern “UFP” has to be programmed into the internal Flash memory. This can be done e.g. by using the OTP/ Flash memory programming tool “Memtool” which is part of this ApNote. With a monitor program (e.g. Ertec EVA166) or a memory upload tool (e.g. “ROM_UP”) the programming success can be checked.

- In the second step test pattern “PROT” has to be executed for Flash protection *activation*. After reset (!) Flash protection is installed. For checking the success of Flash protection activation “PROT” may be started a second time. All read accesses to protected Flash memory or to the FCR deliver the same dummy value (e.g. “B88Bh” or “7887h”). All write accesesses to protected Flash memory or to the FCR are without any effect. Another possibility for checking is to perform a read access to the Flash by using a monitor program or a memory upload tool (see above).

- In the third step UFP has to be started as a user program out of the internal Flash memory for checking the success of temporary Flash protection *deactivation*. When UFP is started, the contents of the Flash memory which means the code of test pattern UFP itself should be seen wordwise at PORT2, beginning with address 00’0000h. The address will be incremented with each NMI (-> NMI Push Button).

Note:
Step two requires a special handling: at first test pattern “PROT” has to be loaded into external RAM by using the Ertec monitor. Afterwards the delay logic has to be connected to EBC0 pin and the BSL circuit has to be disconnected in order to allow user program execution out of external RAM memory. With the next reset “PROT” will be started. The program will ask for Vpp at EBC1 pin. It is also possible to apply Vpp *before* reset and to start “PROT” because for selection of the dedicated bus type (16-Bit Non-Mux on the test board) EBC0 and EBC1 must be “1” during reset. Tests have shown that a too long transition delay at EBC0 could lead to problems because then the bus mode will not be correctly evaluated.

Note:
According to the Flash specification in all cases Vcc has to be applied *before* and disconnected *after* Vpp.

Note:
All sources of the used test pattern are well documented and can be found in the following appendix. Studying the sources will ease the understanding of the test and of the Flash protection mechanism in general.

Note:
Regarding Memtool in some cases programming of the 88C166 Flash memory could fail with the first try, especially when the Flash has been erased before. It is then recommended to perform a reset and to re-try programming (have faith: it works !).

Appendix A

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;    NAME:            prot166.a66 (fp.hex)
;    Target:          88C166 Flash
;    Function:        activation of Flash protection
;    Author:          Peter Kliegelhöfer, HL DC AT
;    Date:            21.07.1998
;
;    (c) 1998 Siemens AG
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++
; Flash protection forever installed with the next reset

$INCLUDE (C166_F.EQU); include equate definitions file
$INCLUDE (PROT166_.EQU); include equate definitions file

     NAME             PROT166       ; SW module name
     ASSUME           DPP3:SYSTEM   ; apply DPP3 to SFR names
     REGDEF           R0-R15 private
     EXTERN           Unlock:NEAR

;*****************************************************************************
; Stack Size selection of between 32 and 256 words. SYSCON[14..13]

     __STKSZ          LIT      '0'  ; System stack sizes
                                    ; 0 = 256 words (Reset value)
                                    ; 1 = 128 words
                                    ; 2 =  64 words
                                    ; 3 =  32 words
     SSKDEF           __STKSZ       ; System stack size
     RBANK            REGDEF R0-R15 ; Register usage

;*****************************************************************************

ProtectCode0SECTION CODE PUBLIC 'CLASS_PROT166'; Open code section
ProtectProc0PROC TASK INTNO = 00h

_32_PROG_MODEEQU 8081h         ; EQU for set 32 bit mode and programming mode
                               ; FWMSET = 1, WDWW = 1, FWE = 1
SET_UPROG_FRMEQU 4400h         ; EQU for set bit 14 (UPROG) and 10 (FRM)
                               ; UPROG = 1, FRM = 1

     ;=========================================================================
     ; Init micro
     ;=========================================================================

     calls            0, FLASHPROC  ; Enable Flash on Segment 1
     DISWDT                         ; Disable watchdog timer
     mov              DPP0,#4h      ; adjust DPP
     mov              DPP1,#5h
     NOP                            ; make sure that new DPP values are used
     MOV              STKOV,   #?SYSSTACK_BOTTOM + 6*2   ; Set stack underflow
                                                         ; pointer.
     MOV              STKUN,   #?SYSSTACK_TOP            ; Set stack overflow
                                                         ; pointer.
     MOV              SP,      #?SYSSTACK_TOP            ; Set stack pointer.
     MOV              CP,      #RBANK                    ; Set context pointer.
     NOP                                                 ; make sure that new CP
                                                         ; value is used
     mov              DP2, #0FFFFh  ; define PORT2 as output
     nop

     ;=========================================================================
     ; Test program start
     ;=========================================================================
```

```
    mov             P2, #05555h   ; 0.: output "0101010101010101"
    idle
    nop

    ;=============================================================================
    ; Output Flash content
    ;=============================================================================

    mov             r3,ZEROS      ; 2.: output first data word
    mov             P2, [r3]      ; of mapped Flash (01'0000h)
                                  ;
    idle                          ; in case of already protected
    nop                           ; Flash: dummy read 7887h or B88Bh

    ;=============================================================================
    ; Now apply Vpp to EBC1
    ;=============================================================================

    mov             P2,#0C003h    ; 3.: output "11------------11"
    idle
    nop

    ;=============================================================================
    ; Enter writing mode
    ;=============================================================================

    call            Unlock        ; perform unlock sequence

    ;=============================================================================
    ; Output FCR
    ;=============================================================================

    mov             P2, dpp1:FCR  ; 4.: output FCR; value should
                                  ; be 0010h if Vpp is applied
                                  ; (VPPRIV=set)
                                  ; dummy read 7887h or B88Bh in
                                  ; case of already protected Flash
    idle
    nop

    ;=============================================================================
    ; Set FLASH protection mode
    ;=============================================================================

    mov             R0, #_32_PROG_MODE    ; set R0 register for 32 bit mode
                                          ; and programming mode
    mov             R1, #SET_UPROG_FRM        ; set R1 register for bit 14 (UPROG)
                                              ; and bit 10 (FRM)
    mov             DPP1:FCR, R0              ; set 32 bit mode and programming
                                              ; mode
    mov             DPP1:FCR, R0              ; dummy (needed !!!)
; write line TWO times !!!
    mov             DPP1:SFCR, R1             ; set bit 14 (UPROG) and bit 10 (FRM)

    ;=============================================================================
    ; Wait 1 s to assure programming
    ;=============================================================================

    bclr            T3R                       ; Stop timer T3
    mov             T3,    #ONE_SEC           ; Initialize timer value
    mov             T3IC,  #0                 ; Disable T3 interrupts, clear
                                              ; possible T3 interrupt
                                              ; requests
    mov             T3CON, #PROTECTTCON       ; Initialize timer control,
                                              ; start timer
wait:
```

```
     jnb             T3IR,  wait             ; Wait until timer underflows

     ;===========================================================================
     ; Exit writing mode
     ;===========================================================================

     mov             R0, ZEROS
     mov             DPP1:FCR, R0
     mov             DPP1:SFCR, R0

     ;===========================================================================
     ; Return from subroutine
     ;===========================================================================

     mov             P2, #0aaaah             ; 5.: output "1010101010101010"
     idle
     nop
     retv                                    ; Exit from routine

ProtectProc0ENDP

     ;===========================================================================
     ; NMI interrupt service routine
     ;===========================================================================

NMIProc0PROC TASK INTNO = 02h

     bfldh           TFR, #80h, #00h         ; clear NMI flag
     reti

NMIProc0ENDP

FLASHPROC PROC FAR

     ;===========================================================================
     ; Configure internal Flash
     ;===========================================================================

; ATTENTION: make sure that 'SYSCON' references to the right address !
; (88C166: FF0Ch/ 86h; C167: FF12h/ 89h)

     MOV             DPP3,#3h      ; SYSCON: enable internal Flash and map
     MOV             r1,#0FF0Ch    ; it to segment 1
     MOV             r5,#0040h     ; stack = 256 words, BTYP = 01, 15 WS
                                   ; (for slow external memory)
     MOV             [r1],r5       ; CPU DPPx refresh (CPU HW bug)
     MOV             DPP0,#0h
     MOV             DPP1,#1h
     MOV             DPP2,#2h
     MOV             DPP3,#3h
     rets

FLASHPROC ENDP
ProtectCode0ENDS                   ; Close code section
     END
```

Appendix B

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;    NAME: unlock.a66
;
;    88C166 Flash-EPROM Unlock-Routine
;
;    1)  Entering Flash writing mode
;    2)  Executing a waiting loop during which the device
;        sets up its internal high voltage.
;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

$INCLUDE (C166_F.EQU)              ; Constant definitions
$INCLUDE (UNLOCK_.EQU)             ; Constant definitions

     NAME             UNLOCK
     ASSUME           DPP3:SYSTEM
     REGDEF  R0-R15 private
     PUBLIC           Unlock

UnlockCode0SECTION CODE PUBLIC 'CLASS_UNLOCK'; Open section code
Unlock               PROC     NEAR

     ;=====================================================================
     ; U N L O C K   S E Q U E N C E
     ;=====================================================================

     mov              R2, #4000h    ; Initialize Flash pointer
     mov              DPP1:FCR, R2  ; First pass of unlock sequence
     mov              [R2], R2      ; Second pass of unlock sequence

     ;=====================================================================
     ; N O W   F L A S H   I S   I N   W R I T I N G   M O D E
     ;=====================================================================

     ;=====================================================================
     ; Wait UNLOCKTIME for charge pump
     ;=====================================================================

     mov              T3IC,  #0     ; Initialize timer interrupt control
     mov              T3,    #UNLOCKTVAL   ; Initialize timer value
     mov              T3CON, #UNLOCKTCON   ; Initialize timer control,
                                           ; start timer
WaitUnlock:
     jnb              T3IR,  WaitUnlock    ; Wait until timer underflows
     bclr             T3IR                 ; Clear timer interrupt request
     bclr             T3R                  ; Stop timer
     ret                                   ; Return to calling routine
Unlock               ENDP
UnlockCode0ENDS                            ; Close section code
     END
```

Appendix C

```
;------------------------------------------------------------------------------
; Name of file: C166_f.equ
; Oscillator frequency in MHz*100
; e.g.: 40MHz => @SET(FCPU,4000) , 39.32MHz => @SET(FOSC,3932)
;------------------------------------------------------------------------------

@SET (FOSC,2000)
FOSC EQU              @FOSC
FCR                   EQU 07FFEh
SFCR                  EQU 07FFEh
```

Appendix D

```
$SAVE
$NOLIST
;------------------------------------------------------------------------------
; Name of file: Prot166_.equ
; Timer reload value to gain the one second when the timer counts downwards
;    ONE_SEC*(1024/f(OSC)) = 1s
;    ONE_SEC = f(OSC)*1s/1024
;    ONE_SEC = (FOSC*1e4*)/1024
;    ONE_SEC = FOSC*10000/1024
;                             , e.g. set FOSC = 2000 for 20 MHz
;                             , =>ONE_SEC = 19531,25
;    ONE_SEC = (FOSC*10000/1024) + 1
;                             , + 1 for avoiding errors by division
;------------------------------------------------------------------------------
ONE_SEC EQU           ((FOSC*10000/1024) + 1)
;------------------------------------------------------------------------------
; Timer 3 Configuration Value
;------------------------------------------------------------------------------
PROTECTTCONEQU        0000000011000111b
;                             xxxxxxxx||x|||||
;                                     || ||f(OSC)/1024
;                                     || ||
;                             || Timer mode: internal clock
;                                     ||
;                                     |Timer Run
;                                     |
;                                     Count Down
$RESTORE
```

Appendix E

```
$SAVE
$NOLIST
;------------------------------------------------------------------------------
; Name of file: unlock_.equ
; Wait time in microseconds needed to enter the writing mode of the 88C166
;------------------------------------------------------------------------------
UNLOCKTIMEEQU         10
;------------------------------------------------------------------------------
; Timer reload value to gain the unlock time specified by UNLOCKTIME when
; the timer counts downwards
;    UNLOCKTVAL * (8/f(OSC)) = time
;    UNLOCKTVAL = f(OSC)*time/8
;    UNLOCKTVAL = (FOSC*1e4*UNLOCKTIME*1e-6)/8
;    UNLOCKTVAL = FOSC*UNLOCKTIME/800
;                             , e.g. set FOSC = 2000 for 20 MHz
;                             , e.g. set UNLOCKTIME = 10 for 10 us
;    UNLOCKTVAL = (FOSC*UNLOCKTIME/800) + 1
;                             , + 1 for avoiding errors by division
```

```
;------------------------------------------------------------------------------
UNLOCKTVAL EQU       (( 2 * UNLOCKTIME * FOSC / 1600) + 1) / 2
;------------------------------------------------------------------------------
; Timer 3 Configuration Value
;------------------------------------------------------------------------------
UNLOCKTCONEQU        0000000011000000b
;                            xxxxxxxx||x|||||
;                                    || ||f(OSC)/8
;                                    || ||
;                            || Timer mode: internal clock
;                                    ||
;                                    |Timer Run
;                                    |
;                                    Count Down
$RESTORE
```

Appendix F

```
; Name of file: prot166.ILO
LOCATE
PUBTOGLB
prot166.obj
unlock.obj
NOVECINIT
CLASSES
(
     CLASS_UNLOCK
     CLASS_PROT166    (02000h to 7fffh)
)
```

Appendix G

```
ECHO OFF
REM          Name of file: prot166.bat
REM         batchprogram for compiling, assembling and linking
REM          of the Demo-SW for activating the 88C166 Flash protection
REM
REM          Tasking Compiler V6.0
ECHO ON

del *.hex
del *.map
del *.lst
m166 prot166.a66 CASE
IF ERRORLEVEL 1 GOTO ENDE
a166 prot166.src DEBUG SYMB CASE EXTEND NOMOD166 STDNAMES(reg166.def) SEGMENTED
IF ERRORLEVEL 1 GOTO ENDE
m166 unlock.a66 CASE
IF ERRORLEVEL 1 GOTO ENDE
a166 unlock.src DEBUG SYMB CASE EXTEND NOMOD166 STDNAMES(reg166.def) SEGMENTED
IF ERRORLEVEL 1 GOTO ENDE
l166 @prot166.ilo TO prot166.out
ihex166 -l16 -i16 -oprot166.hex prot166.out
rem ren prot166.hex fp.hex
rem copy fp.hex c:\eva-boards\eva16a
rem copy fp.hex c:\eva-boards\eva16a_p
del *.obj
del *.src
rem del *.lst
: ENDE
```

Appendix H

```
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;    NAME:            UFP166.a66
;    Target:          88C166 Flash
;    Function:        - temporary unlock of Flash protection
;                     - load Flash read routine into IRAM
;                     - output of Flash content (PORT2)
;    Author:          Peter Kliegelhöfer, HL DC AT
;    Date:            21.07.1998
;
;    (c) 1998 Siemens AG
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
     NAME             UFP166        ; SW module name
     ASSUME           DPP3:SYSTEM   ; apply DPP3 to SFR names
     REGDEF           R0-R15 private
;*****************************************************************************
     FCR              equ      07FFEh
     RAMSTART         equ      0FA40h

; Stack Size selection of between 32 and 256 words. SYSCON[14..13]
     __STKSZ                   LIT          '0'; System stack sizes
                                            ; 0 = 256 words (Reset value)
                                            ; 1 = 128 words
                                            ; 2 =  64 words
                                            ; 3 =  32 words
     SSKDEF           __STKSZ               ; System stack size
RBANKREGDEF R0-R15                          ; Register usage

;*****************************************************************************

UFP166Code0      SECTION CODE PUBLIC 'CLASS_UFP166'; Open code section
UFPProc0         PROC TASK INTNO = 00h

     ;=========================================================================
     ; Init micro
     ;=========================================================================

     DISWDT                                          ; Disable watchdog timer
     MOV         STKOV, #?SYSSTACK_BOTTOM + 6*2      ; Set stack underflow pointer.
     MOV         STKUN,       #?SYSSTACK_TOP         ; Set stack overflow pointer.
     MOV         SP,          #?SYSSTACK_TOP         ; Set stack pointer.
     MOV         CP,          #RBANK                 ; Set context pointer.
     NOP                                             ; make sure that new CP value
                                                     ; is used
     mov         DP2, #0FFFFh                        ; define PORT2 as output
     nop
     mov         P2, #05555h                         ; output "-1-1-1-1-1-1-1-1"
     idle
     nop

     ;=========================================================================
     ; Temporary disable of 88C166 Flash protection by clearing bit RPROT
     ; in the FCR of the non-mapped Flash
     ;=========================================================================

     mov              dpp1,#1h      ; RPROT bit can be accessed in Flash
     mov              r0,ZEROS      ; standard mode without having to enter
     mov              dpp1:FCR,r0   ; Flash writing mode. Other bits are not
                                    ; affected by such a write operation

     ;=========================================================================
     ; Copy Flash read routine into IRAM and start execution when finished
     ;=========================================================================
```

```
     mov             r1, #0FA40h
     mov             r2, #02600h
LOOP2:mov            [r1+], [r2]
     add             r2,#2h
     cmp             r1, #0FA54h
     jmpr            cc_Z, STEX
     jmp             cc_UC, LOOP2
STEX:jmps            0, 0FA40h
     retv                                ; dummy exit

UFPProc0ENDP

     ;=========================================================================
     ; NMI interrupt service routine
     ;=========================================================================

NMIProc0PROC TASK INTNO = 02h

     bfldh           TFR, #80h, #00h     ; clear NMI flag
     reti

NMIProc0ENDP

     ;=========================================================================
     ; Data section
     ;=========================================================================

UFP166Data0SECTION DATA PUBLIC 'CLASS_DUFP166'; Open data section

     ; data location in Flash: 02600h - 02700h
     ; code location in IRAM:  0FA40h - 0FA54h (20 bytes)

                                     ; LOW byte first...
     dw              0f1e6h          ; mov r1, #2000h      ;(Flash not mapped...)
     dw              02000h          ;
     dw              05198h          ; mov r5, [r1+]
     dw              0f5f6h          ; mov P2, r5
     dw              0ffc0h          ;
     dw              07887h          ; idle
     dw              08787h          ;
     dw              000cch          ; nop
     dw              000eah          ; jmpa 0FA44h
     dw              0fa44h          ;

UFP166Data0ENDS
UFP166Code0ENDS                      ; Close code section
     END                             ; Stop assemblation
```

Appendix I

```
; Name of file: Ufp166.ILO
LOCATE
PUBTOGLB
ufp166.obj
NOVECINIT
CLASSES
(
     CLASS_UFP166     (02000h to 02500h)
     CLASS_DUFP166    (02600h to 02700h)
)
```

Appendix J

```
ECHO OFF
REM          Name of file: UFP166.bat
REM          batchprogram for compiling, assembling and linking
REM          of the Demo-SW for temporary disable of the 88C166
REM          Flash protection
REM
REM          For Tasking Compiler V6.0
ECHO ON
del *.hex
del *.map
del *.lst
m166 UFP166.a66 CASE
IF ERRORLEVEL 1 GOTO ENDE
a166 UFP166.src DEBUG SYMB CASE EXTEND NOMOD166 STDNAMES(reg166.def) SEGMENTED
IF ERRORLEVEL 1 GOTO ENDE
l166 @UFP166.ilo TO UFP166.out
ihex166 -l16 -i16 -oUFP166.hex UFP166.out
rem copy UFP166.hex c:\eva-boards\eva16a
rem copy UFP166.hex c:\eva-boards\eva16a_p
rem copy UFP166.hex h:\c\progtool\pls3\memtool
del *.obj
del *.src
rem del *.lst
: ENDE
```