SIEMENS

Microcontrollers

ApNote AP1652

☑ : Additional file
AP165201.EXE available

Bootstrap Loader Support

on C163-16F Flash Devices (Bx-step)

C163-16F Flash devices have no internal boot ROM and therefore no bootstrap loader which is needed for in-system programming of virgin Flash in single chip applications. From design step Bx on C163-16F Flash devices provide a mechanism for switching the reset vector depending on PORT0 configuration. This application note gives hints and examples how this feature can be used to achieve bootstrap loader support on C163-16F Flash.

Author : Peter Kliegelhöfer / HL DC AT Microcontroller Application Support

Contents **Page**

AP1652 ApNote - Revision History		
Actual Revision : 07.98	Previous Revision : none (Original version)	
Page of actual Rev.	Page of prev.Rel.	

1 Introduction

In general a bootstrap loader is used to load startup code and/ or Flash programming routines from a remote code source via a serial interface without requiring additional external boot memory.
On most C16x microcontrollers a bootstrap loader code is usually stored in a special internal boot-ROM (or equivalent non-volatile memory section). However, such a bootstrap loader code memory is not available on C163-16F.

From design step Bx on C163-16F devices provide a mechanism for switching the reset vector depending on PORT0 configuration. This feature can be used to achieve bootstrap loader support on C163-16F Flash. For this it is necessary to pre-program the user Flash with a bootstrap loader code. On one hand this can be done by the user himself. On the other hand Siemens reserves the option to deliver pre-programmed C163-16F Flash devices in the future with a special marking text.

This application note gives hints and examples how to prepare C163-16F Flash devices for bootstrap loader functionality.

2 Pre-programming the internal user Flash

From design step Bx on two different start addresses can be selected after reset when pin $\overline{EA}$ is pulled “high”, depending on PORT0 configuration.

The start address is

00’0000h	**when P0L.4 is pulled “high” or left open during reset** or
02’0000h	**when P0L.4 is pulled “low” during reset**.

Basically these address areas are covered by internal Flash memory. Single chip mode (= internal Flash enabled) is entered, when pin $\overline{EA}$ (external access) is “high” during reset.
Mapping of the physical Flash address 00’0000h to the logical address 00’0000h (which concerns the lowest 32K block of the internal Flash memory) depends on bit “ROMS1” in register SYSCON. With ROMS1 = 0 the lowest 32K of internal Flash are mapped to segment 0 (00’0000h...00’7FFFh). Because the interrupt vector table addresses (which are very important for every user program) point to the lowest addresses of segment 0, the BSL code must be stored in segment 2.

Since a virgin C163-16F Flash device has no on-chip bootstrap loader, it is necessary to use a hardware system which enables booting from external memory, for instance the Ertec EVA163 evaluation board or the Phytec C163 starter kit.

In the appendix there is a tested bootstrap loader (BSL163) which can be programmed into the internal user Flash. In-system programming of on-chip Flash/ OTP memory is supported by the memory programming tool “Memtool” which is also part of this application note.

3 Important hint

Assuming that there is a pre-programmed C163-16F Flash device, already soldered on PCB, well prepared for in-system programming end-of-line.

- It must be considered that the bootstrap loader code resides *unprotected* in the internal Flash memory at location $02'0000_H$, which is the bottom of Flash sector 2 (figure 1).
- **Care has to be taken when erasing or programming the Flash memory. Especially it is important to know, that there is no way to reboot the C163-16F Flash without external memory, if sector 2 of the internal Flash memory is erased and there is no valid boot code in sector 0.**

4 Bootstrap loader activation

Pin P0L.4 activates the bootstrap loader by code fetches to location $02'0000_H$, when “low” during reset and when pin $\overline{EA}$ is “high”. The bootstrap loader at this location may then, as in the example program BSL163, loads the start code into the internal RAM of the C163-16F Flash via the serial interface ASC0. The main effect of pin P0L.4 beeing “low” during reset is that it changes the reset value of the code segment pointer CSP from 00_H (default) to 02_H. In this case the internal program will be started at 0000_H in segment 2, with ROMS1 = ’0’ and $\overline{EA}$ = ’high” at reset time.
Design step BA is the first step which offers the possibility for switching the reset vector. Therefore the described bootstrap loader functionality is not implementable on earlier steps (AA, AB). By default the C163-16F Flash starts fetching code from location $00'0000_H$, the bootstrap loader is off.

5 Bootstrap loader handling

Because the bootstrap loader resides in the normal accessable Flash memory , there is no protection against unintentional erase. Furthermore the user has to take care not to erase sector 2 until a valid and bootable application code has been programmed successfully to sector 0.

A possible „boot flow“ for the C163-16F Flash without external memory could be:

- Booting by pulling P0L.4 “low” and $\overline{EA}$ = ”high” (selects the bootstrap loader in sector 2 of the Flash memory),
- Provide 16 words of second level bootstrap code at the serial interface (see file “BSL163” in the appendix),
- Start of the 16 words out of the internal RAM (done automatically by the bootstrap loader),
- (Probably) loading a third level bootstrap routine via the serial interface into the internal RAM,
- Programming sector 0 of the Flash memory with the desired application program (probably by download via the serial interface),
- Reset of the C163-16F Flash by pulling P0L.4 “high” (selects the previously programmed code in sector 0 of the Flash memory), test for proper function of the code in sector 0, **this code should always support program download via the serial interface and execution out of the internal RAM**,
- Programming of the other sectors of the Flash memory; **it is recommended to leave the bootstrap loader in sector 2, i.e. to rewrite it in the case that sector 2 has been erased**.

Note:
Flash command sequences always must be executed from locations *outside* the Flash memory.

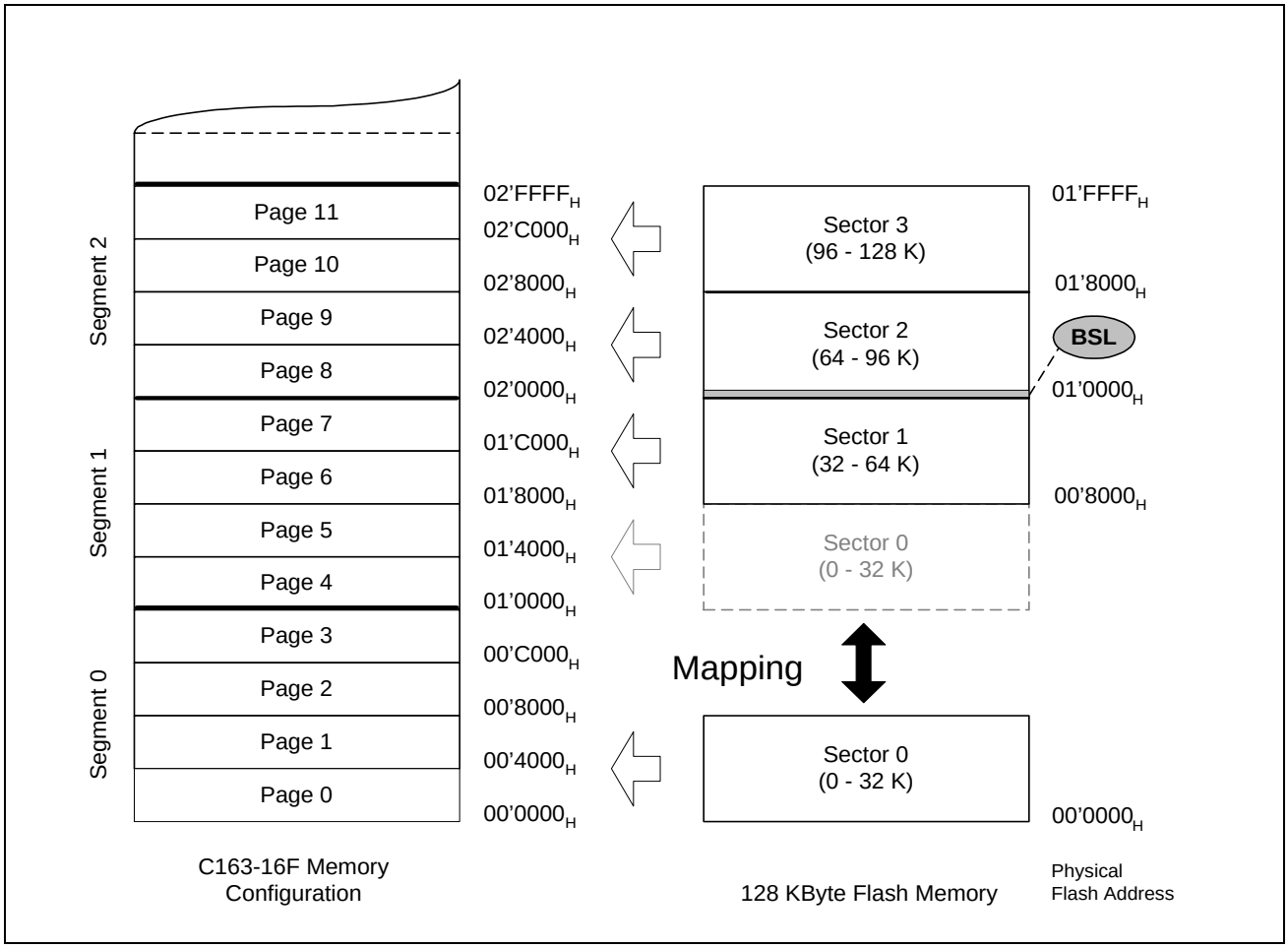

Figure 1
Location of the Bootstrap Loader on C163-16F

The bootstrap loader on C163-16F resides in the Flash memory at address 02'0000$_H$ (physical Flash address 01'0000$_H$) in the case of the example "BSL163" and occupies less than 100 Bytes.

6 Miscellaneous

The acknowledge Byte (ID Byte) of the attached bootstrap loader is „D5$_H$". It is recommended to leave the BSL code in Flash sector 2. In his application software the user must reserve this Flash memory area (address range between 02'0000h and 02'007Fh).
Due to this fact the first address in Flash sector 2 which is freely available for user software is address 02'0080$_H$. Remember that *erase* can only be executed *sectorwise*.

Note:
For further information concerning bootstrap loading please refer to application note AP1607 „Bootstrap Loader C165/ C167".

Appendix A

```
;-----------------------------------------------
;# Name:         BSL163.A66
;# Version:      1.0
;# Function:     C163 Flash Bootstrap Loader
;# Memory:       Internal Flash
;# Author:       Peter Kliegelhoefer
;# Date:         21.10.1997
;# (C)1997 SIEMENS AG
;-----------------------------------------------

$segmented

name bsl163

aux section data at0FA40h
RamRoutineStartlabel word
StartAddressds 32
aux ends

SYSRAM DGROUP aux, SYSTEM
ASSUME DPP3:SYSRAM
RBank REGDEF
SSKDEF 1                        ;  1 = 0FB00h-0FBFFh

MAINCODE section code
MAINPROC proc TASK

;-------------------------------------------------------------------------------
;*  Bootstrap Loader: Receives 32 bytes via serial port 0 and stores them    *
;*  into the internal RAM. Afterwards it performs a jump to the beginning of *
;*  the loaded RAM area (0FA40h) and executes the loaded data as a program.  *
;-------------------------------------------------------------------------------
EndAddress  equ StartAddress + 31  ; End Address of RAM Area
Ack_byte    equ 0D5h               ; Acknowledge Byte of Bootstrap Loader
;-------------------------------------------------------------------------------
BSL_INIT:                       ; don't load syscon or buscon !
      DISWDT                    ; disable watchdog timer
      MOV   SP,#0FC00h          ; set stackpointer
      MOV   STKUN,#0FC00h       ; set stack underflow pointer
      MOV   STKOV,#0FA00h + 6   ; set stack overflow  pointer
      MOV   CP,#0FC00h          ; set registerbank
WaitStartBit:
      JB    P3.11,WaitStartBit  ; wait for start bit at RXD0
      BSET  T6R                 ; start timer T6
WaitStopBit:
      JNB   P3.11,WaitStopBit   ; wait for stop bit at RXD0
      BCLR  T6R                 ; stop timer T6
      MOV   MDL,T6
      SUB   MDL,#36             ; rounding & adjustment
      MOV   R1,#72              ; baudrate = (T6 / 72) - 1
      DIVU  R1
InitSerialPort:
      MOV   S0BG,MDL            ; load baudrate generator
      BSET  P3.10               ; initialize TXD0 output
      BSET  DP3.10
      MOV   S0CON,#8001h        ; initialize serial port 0:
                                ; 8-bit data, no parity, one stop bit
                                ; receiver not started: single wire option
SendAcknowledge:
      MOV   S0TBUF,#Ack_byte    ; send acknowledge byte for baudrate check
WaitTransmOver:
      JNB   S0TIR,WaitTransmOver    ; wait till end of transmission
      BCLR  S0TIR                   ; clear transmit interrupt request flag
      BSET  S0REN               ; receiver enabled
ReceiveData:
```

```
      MOV   R0,#StartAddress
ReceiveLoop:
      JNB   S0RIR,ReceiveLoop  ; wait for receive interrupt request
      MOVB  [R0],S0RBUF        ; store received byte
      BCLR  S0RIR              ; clear receive interrupt request flag
      CMPI1 R0,#EndAddress     ; all bytes received ?
      JMPR  CC_NE,ReceiveLoop  ; if not, continue loop
      JMPS  0,RamRoutineStart  ; yes: jump to RAM routine
      retv                     ; dummy

MAINPROC              endp
MAINCODE              ends

      end
```

Appendix B

```
;-------------------------------------------
;# Name:         BSL163.ILO
;# Function:     Input Locator File
;# (C)1997 SIEMENS AG
;-------------------------------------------

TASK
bsl163.lno
NOVECINIT
NOVECTAB
ad (se(MAINCODE(0020000h)))
```

Appendix C

```
@echo off
rem  ---------------------------------------
rem  Name:       BSL163.BAT
rem  Function:   Batchfile for generating the
rem              C163 Flash Bootstrap Loader
rem  Toolchain:  Tasking tools V5.1
rem  (C)1997 SIEMENS AG
rem  ---------------------------------------
@echo on



a166 bsl163.a66 nomod166 extend stdnames(reg163.def)
l166 bsl163.obj
l166 @bsl163.ilo to bsl163.out nowa(141)
ihex166 -l16 -i16 -obsl163.hex bsl163.out
```