The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205200

FR Family MB91460 Series, Start91460.asm

This application note describes the startup file “start91460.asm” version 2.5 or later provided by Cypress for the MB91460 series-microcontrollers.

Contents

1 Introduction....1
2 Settings of the Start91460.asm....1
2.1 Controller Device (DEVICE)....2
2.2 Boot / Flash Security (BOOT_FLASH_SEC)....2
2.3 Stack Type and Stack Size (STACKUSE, …)....3
2.4 Copy code from Flash to I-RAM (I_RAM)....4
2.5 Low-Level Library Interface (CLIBINIT)....5
2.6 C++ startup (CPLUSPLUS)....5
2.7 Clock Configuration....6
2.8 External Bus Interface (EXTBUS)....9
3 User Clock Settings (CLOCKSPEED == CLOCK_USER)....10
3.1 Clock source(CLOCKSOURCE, ENABLE_SUBCLOCK)....11
3.2 PLL ratio (PLLSPEED)....12
3.3 PLL Auto Gear-Up and –Down (DIV_G, MUL_G)....14
3.4 Clock divider (CPUCLOCK, PERCLOCK, EXTBUSCLOCK)....16
3.5 CAN clock (PSCLOCKSOURCE, PSDVC, CANCLOCK)....17
3.6 Voltage Regulator (REGULATORCTRL, REGULATORSEL)....19
3.7 Memory Controller (FLASHCONTROL, FLASHREADT, FLASHMWT2)....21
4 Section and Data Declaration....24
4.1 Default Sections....24
4.2 Additional Sections....25
5 Document History....26

1 Introduction

All C/C++-language applications need special startup code, which initializes all data areas generated by the compiler and which performs a basic configuration of the microcontroller.

This application note describes the startup file “start91460.asm” version 2.5 or later provided by Cypress for the MB91460 series-microcontrollers.

Main Features of the Startup File “start91460.asm”:

- - Configuration part for easy step by step set-up of the startup code
- - Startup code for pre-setting of core and external bus registers
- - Startup code for memory initialization according to FCC911-compiler sections
- - Startup file is linkage order independent

2 Settings of the Start91460.asm

The startup file has to be configured for the target system (hardware) and the application software. The startup file provides several prepared settings, which have to be checked and which have possibly to be changed. The header of the startup file contains information how to find the lines containing the options to be checked. All lines with the comment extension “<<<” were pointing to available settings.

In the following paragraphs all available settings are described.

2.1 Controller Device (DEVICE)

Set the target device the application is running in. This setting is used by the startup file to check the availability of registers and to check the plausibility of settings.

Available settings for DEVICE:

- MB91464A
- MB91467B
- MB91467C
- MB91467D
- MB91469G
- MB91465K
- MB91463N
- MB91461R
- MB91467R
- MB91465X
- Others

Note: This list will be adapted, depending on new devices

Note: The setting “others” should only be used for the case that desired device of the MB91460 Series is not listed.

Example:

The target controller is the MB91F467DA. This requires the setting.

```
#set   DEVICE          MB91467D                ; <<< select device
```

2.2 Boot / Flash Security (BOOT_FLASH_SEC)

The most flash devices have two flash and two boot security vectors. It is important to set the four vectors correctly. Valid boot security vectors means that the build-in boot ROM cannot be entered anymore.

Available settings for BOOT_FLASH_SEC:

- OFF: The security feature is switched off. The section SECURITY_VECTORS is reserved and the vectors are set to 0xFFFFFFFF.
- ON: The security vectors are not set. But the section SECURITY_VECTORS is reserved. The security vectors have to be set in the user application.

Note: This feature is not available for all devices of the MB91460 Series. The datasheet of the target device lists the available features.

Note: If the device MB91469G is selected, the section SECURITY_VECTORS is located from 0x24:8000 – 0x24:800F.

Note: If the ROMless device MB91461R is selected no section SECURITY_VECTORS is reserved.

Note: For all other devices, including “others” the section SECURITY_VECTORS is located from 0x14:8000 – 0x14:800F.

Example:

The security feature of target controller should be disabled.

```
#set    BOOT_FLASH_SEC   OFF          ; <<< BOOT and Flash Security Vector
```

2.3 Stack Type and Stack Size (STACKUSE, …)

2.3.1 STACKUSE

The setting STACKUSE specifies the stack type that is configured, when the application main()-function is called. The devices of MB91460 Series provide two different types of stack: system stack and user stack. The application can use either system stack only or it can use both system stack and user stack.

System stack is always used for interrupt function. It is automatically selected, if hardware interrupts are executed or if software interrupts are called. All stack operation of interrupt handlers will work with the system stack.

Outside of interrupt handlers the pre-selected stack type is used. This is either user stack (option USRSTACK) or system stack (option SYSSTACK). If SYSSTACK is set, only the system stack area has to be prepared. All operation will work on the same stack. The necessary safety margin has to be reserved for one stack only. SYSSTACK should be used, if the necessary RAM consumption for stack has to be low. If USRSTACK is set, both system stack and user stack have to be prepared. The necessary safety margin has to be provided twice. USRSTACK should be used, if separated stack areas are necessary for management reasons. This might be the case with schedulers, operating systems or other applications.

Available settings for STACKUSE:

- USRSTACK
- SYSSTACK

Example:

The user stack should be used outside of interrupt handlers. This requires the setting.

```
#set    STACKUSE        USRSTACK                ; <<< set active stack
```

2.3.2 STACK_RESERVE, STACK_SYS_SIZE, STACK_USR_SIZE

These settings specify the amount of bytes to be reserved for stacks. This value has to cover all:

- parameters passed over stack
- return addresses for function calls
- local variables (except static)
- temporary data due to compiler optimization
- interrupt context on stacks
- safety margin

For estimating necessary stack size the compiler offers the “-INF stack” option. With it the compiler generates stack information files (extension “stk”), which list the number of bytes necessary to execute each function. These stack information files are already available for all library functions and can be found in the Lib\911\ subdirectory of the Softune Workbench installation.

If the support tool “C-Analyzer” is installed, the additional tool “Musc” is able to collect these data and to calculate the minimum stack size for the whole application.

Available settings for STACK_RESERVE:

- ON
- OFF

Note: If STACK_RESERVE is ON, the sections USTACK and SSTACK are reserved in this module. Otherwise, they have to be reserved in other modules. If STACK_RESERVE is OFF, the size definitions STACK_SYS_SIZE and STACK_USR_SIZE have no meaning.

Example:

A system stack size of 0x200-4 bytes and a user stack size of 0x400-4 bytes should be reserved.

```
#set   STACK_RESERVE   ON                ; <<< reserve stack area in
;                                        ;     this module
#set   STACK_SYS_SIZE  0x200-4           ; <<< byte size of System stack
#set   STACK_USR_SIZE  0x400-4           ; <<< byte size of User stack
```

2.3.3 STACK_FILL, STACK_PATTERN

With theses settings the stacks can be filled with a specific pattern. Filling the stack with pattern allows dynamically checking of the stack area, which had already been used.

Available settings for STACK_FILL:

- ON
- OFF

Note: If STACK_ STACK_FILL is OFF, the pattern definition STACK_PATTERN has no meaning.

Example:

The stacks should be filled with the pattern 0x55AA6699.

```
#set   STACK_FILL      ON         ; <<< fills the stack area with pattern
#set   STACK_PATTERN   0x55AA6699 ; <<< the pattern to write to stack
```

2.4 Copy code from Flash to I-RAM (I_RAM)

If this option is activated code located in the section IRAM is copied during startup from the ROM to the instruction RAM. The code is linked for the instruction RAM. This is for example useful for flash routines, which must be executed in the instruction RAM or for routines, which should be run in the instruction RAM to benefit from the shorter wait states of the instruction RAM.

The segment CODE has to be renamed to IRAM for the code, which should be copied the the instruction RAM during the startup. To rename the segment CODE to IRAM, the following pragma instruction can be used.

```
#pragma segment CODE = IRAM
```

Available settings for I_RAM:

- ON
- OFF

Example:

The code in the segment IRAM should be copied to the instruction RAM during startup.

```
#set    I_RAM           ON        ; <<< select  if  code  in  section IRAM
;                                       should be copied
```

2.5 Low-Level Library Interface (CLIBINIT)

The setting CLIBINI specifies whether the startup code has to call the stream initialization function of the C-library.

The stream initialization is necessary only, if streamed IO-functions are used. These functions (e.g. printf()) also require the definition of application specific low-level functions. For more information refer to the compiler help.

Available settings for CLIBINIT:

- ON
- OFF

Example:

The stream initialization should not be done.

```
#set    CLIBINIT        OFF          ; <<< select ext. libray usage
```

2.6 C++ startup (CPLUSPLUS)

In the C++ specifications, when external or static objects are used, a constructor must be called followed by the main function. Because four-byte pointers to the main function are stored in the EXT_CTOR_DTOR section, call a constructor sequentially from the lower address of the four addresses in that section. If using C++ sources, activate this function to create the section EXT_CTOR_DTOR.

Available settings for CPLUSPLUS:

- ON
- OFF

Example:

C++ is not used.

```
#set    CPLUSPLUS       OFF          ; <<< activate if c++ files are used
```

2.7 Clock Configuration

2.7.1 Clock Selection (CLOCKSPEED)

There are different default configurations available, where all necessary settings for clocks and the related registers are made. These configurations should not be changed. The default settings include the settings for the flash access and the regulator configuration, too.

Beside these configurations, there is the possibility to define a user configuration.

Available settings for CLOCKSPEED:

- NO_CLOCK clock registers are not set by the startup file.
- SUB_32KHZ_CPU__32KHZ_PER_32KHZ_EXT_32KHZ_CAN__2MHZ means:
  Main oscillation = 32 kHz, PLL is not active
  CPU clock (CLKB) = 32 kHz
  Peripheral clock (CLKP) = 32 kHz
  Ext. bus clock (CLKT) = 32 kHz
  CAN clock (CLKCAN) = 2 MHz, using main oscillation
- MAIN_4MHZ_CPU___2MHZ_PER__1MHZ_EXT__1MHZ_CAN__2MHZ means:
  Main oscillation = 4 MHz, PLL is not activated
  CPU clock (CLKB) = 2 MHZ
  Peripheral clock (CLKP) = 1 MHZ
  Ext. bus clock (CLKT) = 1 MHZ
  CAN clock (CLKCAN) = 2 MHz, using main oscillation
- PLL_4MHZ__CPU__48MHZ_PER_16MHZ_EXT_24MHZ_CAN_16MHZ
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 48 MHZ
  Peripheral clock (CLKP) = 16 MHZ
  Ext. bus clock (CLKT) = 24 MHZ
  CAN clock (CLKCAN) = 16 MHz, using PLLx
- PLL_4MHZ__CPU__64MHZ_PER_16MHZ_EXT_32MHZ_CAN_16MHZ means:
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 64 MHZ
  Peripheral clock (CLKP) = 16 MHZ
  Ext. bus clock (CLKT) = 32 MHZ
  CAN clock (CLKCAN) = 16 MHz, using PLLx
- PLL_4MHZ__CPU__80MHZ_PER_20MHZ_EXT_27MHZ_CAN_20MHZ
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 80 MHZ
  Peripheral clock (CLKP) = 20 MHZ
  Ext. bus clock (CLKT) = 27 MHZ
  CAN clock (CLKCAN) = 20 MHz, using PLLx

- PLL_4MHZ__CPU__80MHZ_PER_20MHZ_EXT_40MHZ_CAN_20MHZ
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 80 MHZ
  Peripheral clock (CLKP) = 20 MHZ
  Ext. bus clock (CLKT) = 40 MHZ
  CAN clock (CLKCAN) = 20 MHz, using PLLx
- PLL_4MHZ__CPU__96MHZ_PER_16MHZ_EXT_48MHZ_CAN_16MHZ
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 96 MHZ
  Peripheral clock (CLKP) = 16 MHZ
  Ext. bus clock (CLKT) = 48 MHZ
  CAN clock (CLKCAN) = 16 MHz, using PLLx
- PLL_4MHZ__CPU_100MHZ_PER_20MHZ_EXT_50MHZ_CAN_20MHZ
  Main oscillation = 4 MHz, PLL is activated
  CPU clock (CLKB) = 100 MHZ
  Peripheral clock (CLKP) = 20 MHZ
  Ext. bus clock (CLKT) = 50 MHZ
  CAN clock (CLKCAN) = 20 MHz, using PLLx
- PLL_10MHZ_CPU__60MHZ_PER_20MHZ_EXT_30MHZ_CAN_20MHZ
  Main oscillation = 10 MHz, PLL is activated
  CPU clock (CLKB) = 60 MHZ
  Peripheral clock (CLKP) = 20 MHZ
  Ext. bus clock (CLKT) = 30 MHZ
  CAN clock (CLKCAN) = 20 MHz, using PLLx
- PLL_20MHZ_CPU__60MHZ_PER_20MHZ_EXT_30MHZ_CAN_20MHZ
  Main oscillation = 20 MHz, PLL is activated
  CPU clock (CLKB) = 60 MHZ
  Peripheral clock (CLKP) = 20 MHZ
  Ext. bus clock (CLKT) = 30 MHZ
  CAN clock (CLKCAN) = 20 MHz, using PLLx
- CLOCK_USER means that the user configuration defined in the chapter 3.

Note: Not all frequencies are supported by every device. Please see the hardware manual.

Example:

The default configuration with a core frequency of 64 MHz should be selected.

```
#set    CLOCKSPEED      PLL_4MHZ__CPU__64MHZ_PER_16MHZ_EXT_32MHZ_CAN_16MHZ
```

2.7.2 Select Clock Modulator (CLOMO, CMPR)

The clock modulator is intended for the reduction of electromagnetic interference - EMI, by spreading the spectrum of the clock signal over a wide range of frequencies.

Available settings for CLOMO:

- ON
- OFF

Available settings for CMPR (CMPR):

- The available settings depend on the specific device. They are listed in the corresponding data sheet. Below some values for the MB91F467DA are listed.

For the whole table and the latest information, see the data sheet.

Modulation Degree	Random No	CMPR	Base Clock	Fmin	Fmax
(k)	(N)	[hex]	[MHz]	[MHz]	[MHz]
1	3	026F	84	76.1	93.8
1	3	026F	80	72.6	89.1
1	5	02AE	80	68.7	95.8
2	3	046E	80	68.7	95.8
1	3	026F	76	69.1	84.5
1	5	02AE	76	65.3	90.8
2	3	046E	76	65.3	90.8
1	3	026F	72	65.5	79.9
1	5	02AE	72	62	85.8
1	7	02ED	72	58.8	92.7
2	3	046E	72	62	85.8
3	3	066D	72	58.8	92.7
1	3	026F	68	62	75.3
1	5	02AE	68	58.7	80.9
1	7	02ED	68	55.7	87.3
1	9	032C	68	53	95
2	3	046E	68	58.7	80.9
2	5	04AC	68	53	95
3	3	066D	68	55.7	87.3
4	3	086C	68	53	95
1	3	026F	64	58.5	70.7
1	5	02AE	64	55.3	75.9
1	7	02ED	64	52.5	82
1	9	032C	64	49.9	89.1
2	3	046E	64	55.3	75.9
2	5	04AC	64	49.9	89.1
3	3	066D	64	52.5	82
4	3	086C	64	49.9	89.1
…	…	…	…	…	…

Note: If the CLKCAN source is set either to main oscillator or to PLLx output then the clock for the CAN is not influenced by the clock modulation. If the CLKCAN source is set CPU clock (CLKB) then the clock for the CAN is also modulated if the clock modulator is enabled.

Note: If the clock modulator is enabled, the wait states of the internal flash wait states must be adapted to maximum frequency. Check the wait states settings in this case.

Note: This feature is not supported by every device. Check the data sheet if this feature is available..

Example:

The clock modulator should be enabled and the value of the CMPR should be set to 0x026F.

```
#set    CLOMO           ON              ; <<< Enable /disable clock modulator
#set    CMPR            0x026F          ; <<< Ref. to the data sheet, CMPR
```

2.8 External Bus Interface (EXTBUS)

If the external bus interface is used, it has to be configured properly. Please refer to the hardware manual and the data sheet for detailed information. The startup file supports the configuration of the external bus interface.

Available settings for EXTBUS:

- ON The ext. bus interface is enabled and will be configured.
- OFF The ext. bus interface is disabled. The port function registers are set to general I/O. The registers of external bus interface will not be touched by the startup file.
- DEFAULT Neither the register nor the respective port function registers are touched by the startup file.

Note: Not all devices support an external bus interface. The following devices do not offer an external bus interface: MB91464A, MB91467C, MB91465K, MB91463N, MB91465X.

Note: Be aware, that the device might startup in external bus mode by default after reset.

Example:

The external bus interface should be configured.

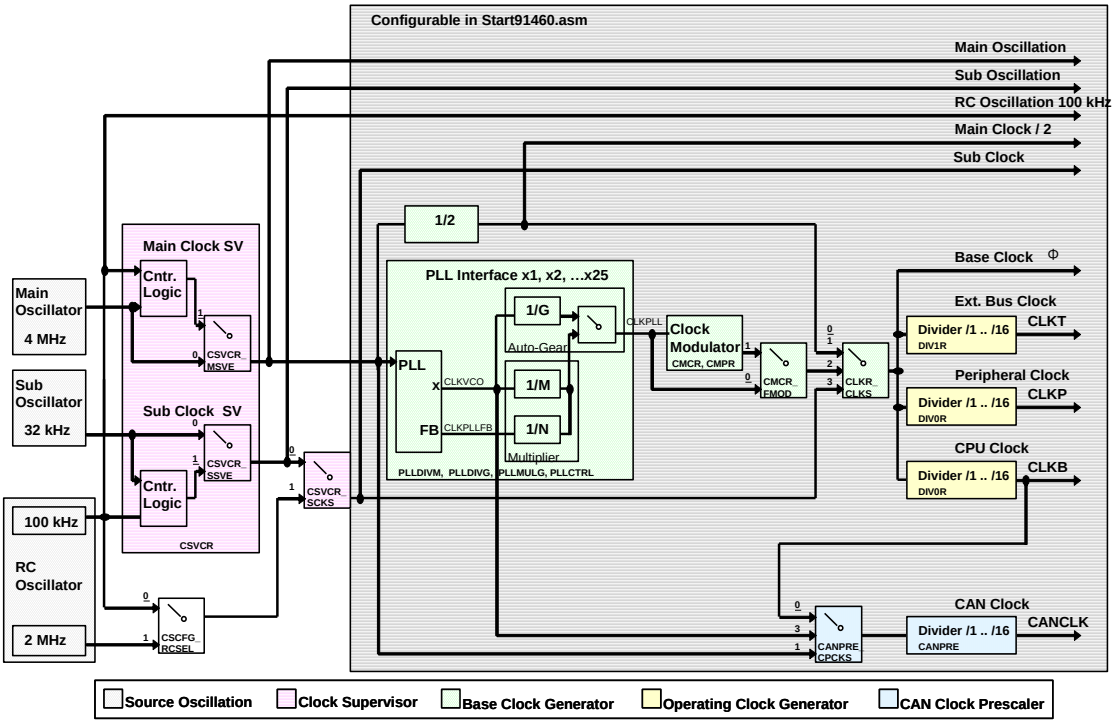
```
#set     EXTBUS          ON                  ; <<< Ext. Bus on/off
```

The startup file supports the configuration of the following registers, too.

- CSER Chip Select Enable Register
- ASR0-7 Area Select Registers 0-7
- ACR0-7 Area Configuration Registers 0-7
- AWR0-7 Area Wait Registers 0-7
- MCRA MEMORY SETTING REGISTER for extend type - A for SDRAM/FCRAM auto
- RCR Refresh control register
- TCR Pin/Timing Control Register
- CHER Cache Enable Register
- PFR0 – 10 Port Function Register 0-7

3 User Clock Settings (CLOCKSPEED == CLOCK_USER)

Beside the default configurations with the settings for clocks and the related registers, there is the possibility to define a user configuration for the settings. In the following the different settings are explained. The following figure shows which part of the clock settings can be configured in the start91460.asm.

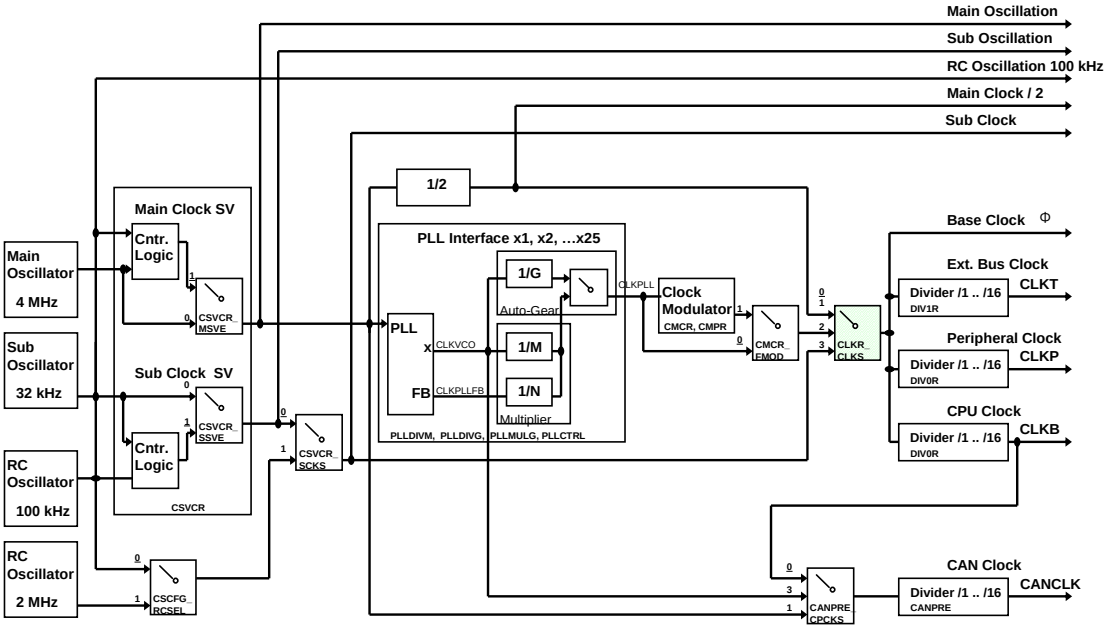


Beside the clock settings the startup supports the settings for the voltage regulators and the wait state settings of the internal flash, too. The clock modulator is not part of the user clock settings, but part of the general settings.

The user clock settings can be found in the startup91460.asm in the chapter ”5.1 CLOCKSPEED == CLOCK_USER”.

3.1 Clock source (CLOCKSOURCE, ENABLE_SUBCLOCK)

The clock source of the base clock can be configured in the following.



Available settings for CLOCKSOURCE:

- NOCLOCK — The clock register are not set by the startup code
- MAINCLOCK — MB91461R: Base clock frequency = ¼ of main clock
  Others: Base clock frequency = ½ of main clock
- MAINPLLCLOCK — The PLL (programmable) is used as base clock
- SUBCLOCK — Te sub clock is selected as base clock

Available settings for ENABLE_SUBCLOCK:

- ON — Enable sub clock
- OFF — Sub clock is not enabled

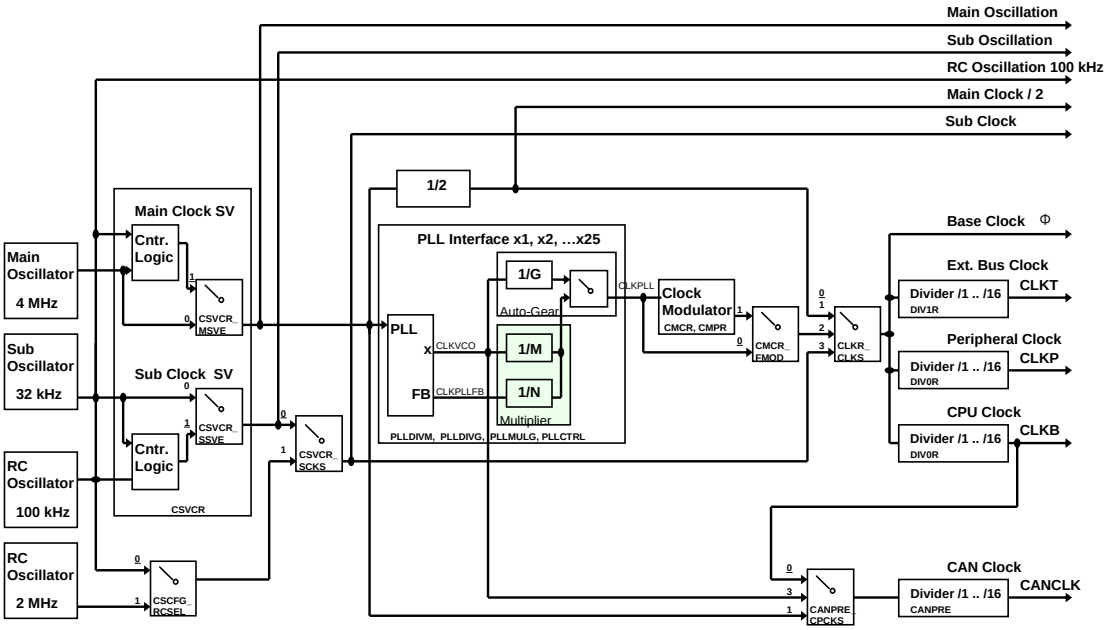
Example:

The PLL should be used and the sub clock should not be enabled.

```
#set CLOCKSOURCE     MAINPLLCLOCK ;<<< Clocksource
#set ENABLE_SUBCLOCK OFF          ;<<< Subclock: ON/OFF
```

3.2 PLL ratio (PLLSPEED)

The PLL multiplier can be configured individually, but there are recommended settings which should be used. The table below gives some recommendations. Please check the data sheet and the hardware manual for updated values.



Available settings for PLLSPEED:

- Main clock 4 MHz:

Name	Setting PLLSPEED	Resulting Base Clock	Remark
PLLx3	0x0B02	12 MHz	
PLLx4	0x0903	16 MHz	
PLLx5	0x0704	20 MHz	
PLLx6	0x0505	24 MHz	
PLLx7	0x0306	28 MHz	
PLLx8	0x0307	32 MHz	
PLLx9	0x0308	36 MHz	
PLLx10	0x0309	40 MHz	
PLLx11	0x010A	44 MHz	
PLLx12	0x010B	48 MHz	
PLLx13	0x010C	52 MHz	
PLLx14	0x010D	56 MHz	
PLLx15	0x010E	60 MHz	
PLLx16	0x010F	64 MHz	
PLLx17	0x0110	68 MHz	
PLLx18	0x0111	72 MHz	
PLLx19	0x0112	76 MHz	

Name	Setting PLLSPEED	Resulting Base Clock	Remark
PLLx20	0x0113	80 MHz	
PLLx21	0x0114	84 MHz	**Note**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx22	0x0115	88 MHz	**Note**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx23	0x0116	92 MHz	**Note**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx24	0x0117	96 MHz	**Note**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx25	0x0118	100 MHz	**Note**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R, MB91467D

- Main clock 10 MHz (MB91461R only):

Name	Setting PLLSPEED	Resulting Base Clock	Remark
PLL_10_MHzx2	0x0301	20 MHz	
PLL_10_MHzx3	0x0302	30 MHz	
PLL_10_MHzx4	0x0303	40 MHz	
PLL_10_MHzx5	0x0104	50 MHz	
PLL_10_MHzx6	0x0105	60 MHz	
PLL_10_MHzx7	0x0106	70 MHz	

- Main clock 10 MHz (MB91461R only):

Name	Setting PLLSPEED	Resulting Base Clock	Remark
PLL_20_MHzx2	0x0301	40 MHz	
PLL_20_MHzx3	0x0302	60 MHz	

Note: PLLSPEED corresponds to the registers PLLDIVM and PLLDIVN at the addresses 0x48Ch and 0x48Dh.

Note: Never exceed the maximum operation frequency. Check the corresponding data sheet.

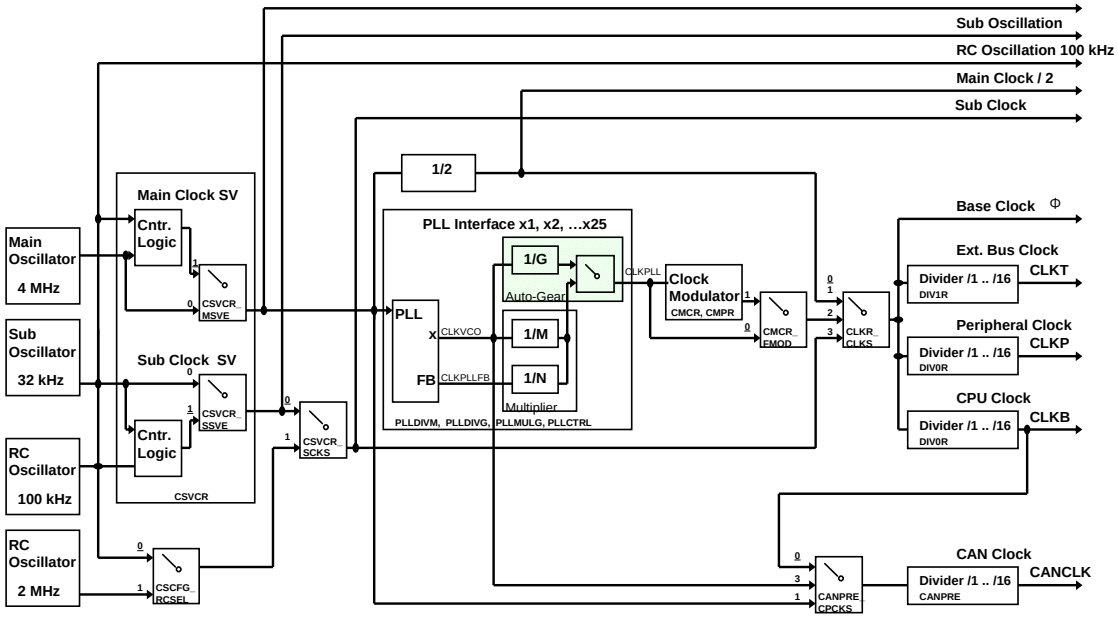
Example:

The PLL should be set to 4 MHz x 16 = 64 MHz.

```
#set PLLSPEED        0x010F      ;<<< 0x48Ch, 0x48Dh: PLLDIVM/N ;  64 MHz
```

3.3 PLL Auto Gear-Up and –Down (DIV_G, MUL_G)

To avoid voltage drops and surges when switching the clock source from oscillator to high frequency PLL/DLL output (or vice versa), a clock smooth gear-up and gear-down circuitry is implemented with the PLL interface. The table below gives some recommendations. Please check the data sheet and the hardware manual for updated values.



Available settings for DIV_G (PLLDIVG), MUL_G (PLLMULG):

- Main clock 4 MHz:

Name	Setting DIV_G	Setting MUL_G	Remark
PLLx3	0x0F	0x1F	
PLLx4	0x0F	0x1F	
PLLx5	0x0F	0x1B	
PLLx6	0x0F	0x17	
PLLx7	0x0F	0x17	
PLLx8	0x0F	0x17	
PLLx9	0x0F	0x17	
PLLx10	0x0F	0x17	
PLLx11	0x0F	0x0B	
PLLx12	0x0F	0x0B	
PLLx13	0x0F	0x0B	
PLLx14	0x0F	0x0F	
PLLx15	0x0F	0x0F	
PLLx16	0x0F	0x0F	
PLLx17	0x0F	0x0F	
PLLx18	0x0F	0x13	
PLLx19	0x0F	0x13	
PLLx20	0x0F	0x13	

Name	Setting DIV_G	Setting MUL_G	Remark
PLLx21	0x0F	0x13	**Not**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx22	0x0F	0x17	**Not**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx23	0x0F	0x17	**Not**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx24	0x0F	0x17	**Not**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R
PLLx25	0x0F	0x17	**Not**: MB91V460A, MB91464A, MB91465K, MB91463N, MB91467R, MB91467D

- Main clock 10 MHz (MB91461R only):

Name	Setting DIV_G	Setting MUL_G	Remark
PLL_10_MHzx2	0x0F	0x1F	
PLL_10_MHzx3	0x0B	0x1F	
PLL_10_MHzx4	0x0B	0x1F	
PLL_10_MHzx5	0x0B	0x1F	
PLL_10_MHzx6	0x0B	0x1F	
PLL_10_MHzx7	0x0B	0x1F	

- Main clock 20 MHz (MB91461R only):

Name	Setting DIV_G	Setting MUL_G	Remark
PLL_20_MHzx2	0x0B	0x1F	
PLL_20_MHzx3	0x0B	0x1F	

Note: DIV_G corresponds to the register PLLDIVG at the addresses 0x48Eh.

Note: MUL_G corresponds to the register PLLMULG at the addresses 0x48Fh.

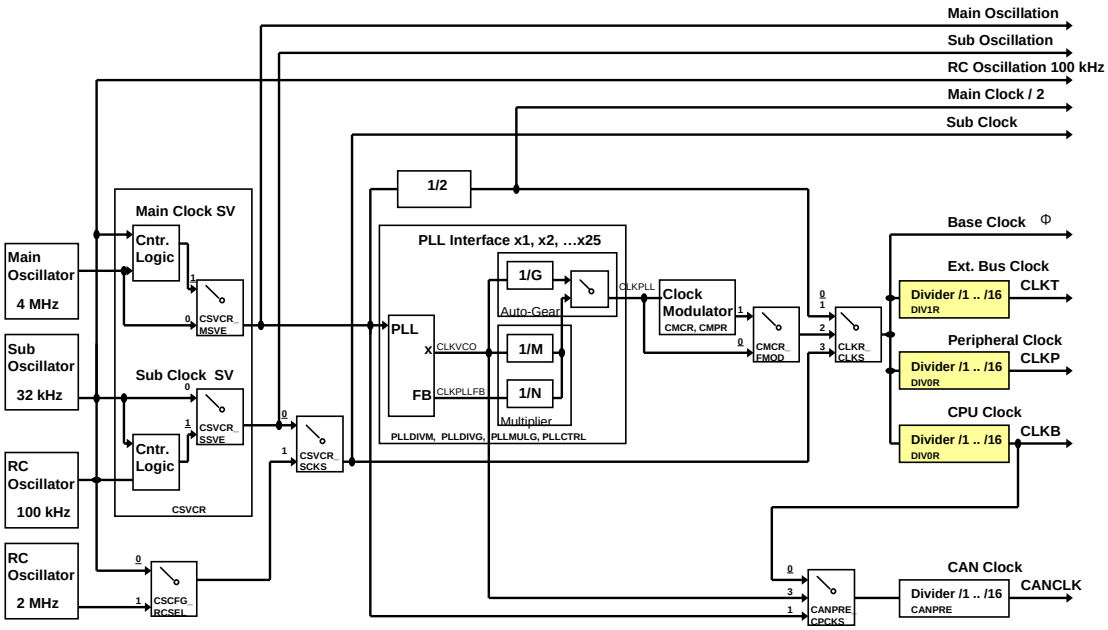
Example:

The PLL should be set to 4 MHz x 16 = 64 MHz.

```
#set DIV_G          0x0F        ;<<< 0x48Eh: PLLDIVG;
#set MUL_G          0x0F        ;<<< 0x48Fh: PLLMULG;
```

3.4 Clock divider (CPUCLOCK, PERCLOCK, EXTBUSCLOCK)

The base clock can be divided for the different clock trees. There are clock divider for the CPU clock (CLKB), the peripheral clock (CLKP) and the external bus interface clock (CLKT). The divider can be configured independently.



Available settings for CPUCLOCK (DIV0R_B), PERCLOCK (DIV0R_P), EXTBUSCLOCK (DIV1R_T):

Name	Setting	Remark
BASECLOCK_DIV1	0x00	clock = 1/1 base clock
BASECLOCK_DIV2	0x01	clock = 1/2 base clock
BASECLOCK_DIV3	0x02	clock = 1/3 base clock
BASECLOCK_DIV4	0x03	clock = 1/4 base clock
BASECLOCK_DIV5	0x04	clock = 1/5 base clock
BASECLOCK_DIV6	0x05	clock = 1/6 base clock
BASECLOCK_DIV7	0x06	clock = 1/7 base clock
BASECLOCK_DIV8	0x07	clock = 1/8 base clock
BASECLOCK_DIV9	0x08	clock = 1/9 base clock
BASECLOCK_DIV10	0x09	clock = 1/10 base clock
BASECLOCK_DIV11	0x0A	clock = 1/11 base clock
BASECLOCK_DIV12	0x0B	clock = 1/12 base clock
BASECLOCK_DIV13	0x0C	clock = 1/13 base clock
BASECLOCK_DIV14	0x0D	clock = 1/14 base clock
BASECLOCK_DIV15	0x0E	clock = 1/15 base clock
BASECLOCK_DIV16	0x0F	clock = 1/16 base clock

Note: CPUCLOCK corresponds to the register DIV0R_B at the addresses 0x486h.

Note: PERCLOCK corresponds to the register DIV0R_P at the addresses 0x486h.

Note: EXTBUSCLOCK corresponds to the register DIV1R_T at the addresses 0x487h.

Note: Never exceed the maximum operation frequency. Check the corresponding data sheet.

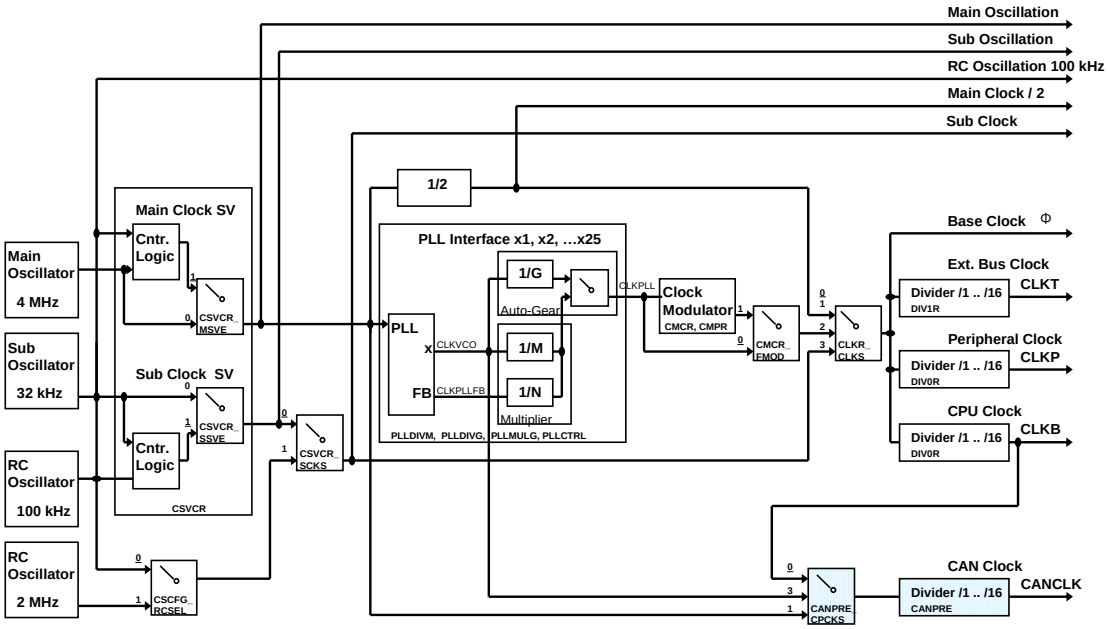
Example:

The base clock frequency is 64 MHz. The CPU clock should run with 64 MHz, the peripheral with 16 MHz and the external bus clock with 32 MHz.

```
#set CPUCLOCK        0x00        ;<<< 0x486h: DIV0R_B;     => /1 ;  64 MHz
#set PERCLOCK        0x03        ;<<< 0x486h: DIV0R_P;     => /4 ;  16 MHz
#set EXTBUSCLOCK     0x01        ;<<< 0x487h: DIV1R_T;     => /2 ;  32 MHz
```

3.5 CAN clock (PSCLOCKSOURCE, PSDVC, CANCLOCK)

The CAN prescaler clock source and source clock divider can be configured in the following.



Available settings for PSCLOCKSOURCE (CANPRE):

- PSCLOCK_CLKB The CLKB is the clock source for the CAN clock prescaler.
- PSCLOCK_PLL The CLKVCO is the clock source for the CAN clock prescaler.
- PSCLOCK_MAIN The main oscillation is the clock source for the CAN clock prescaler.

The following table shows the selectable prescaler source clock frequency depending on the selectable PLL multiplier. The frequencies are only valid; if the settings for the PLL multiplication are the ones, defined above.

- Main clock 4 MHz:

PLL Multiplier	PSCLOCK_CLKB (CLKB)	PSCLOCK_PLL (CLKVCO)	PSCLOCK_MAIN (Main Oszillation)
PLLx3	12 MHz	144 MHz	4 MHz
PLLx4	16 MHz	160 MHz	4 MHz
PLLx5	20 MHz	160 MHz	4 MHz
PLLx6	24 MHz	144 MHz	4 MHz
PLLx7	28 MHz	112 MHz	4 MHz
PLLx8	32 MHz	128 MHz	4 MHz
PLLx9	36 MHz	144 MHz	4 MHz
PLLx10	40 MHz	160 MHz	4 MHz
PLLx11	44 MHz	88 MHz	4 MHz
PLLx12	48 MHz	96 MHz	4 MHz
PLLx13	52 MHz	104 MHz	4 MHz
PLLx14	56 MHz	112 MHz	4 MHz
PLLx15	60 MHz	120 MHz	4 MHz
PLLx16	64 MHz	128 MHz	4 MHz
PLLx17	68 MHz	136 MHz	4 MHz
PLLx18	72 MHz	144 MHz	4 MHz
PLLx19	76 MHz	152 MHz	4 MHz
PLLx20	80 MHz	160 MHz	4 MHz
PLLx21	84 MHz	168 MHz	4 MHz
PLLx22	88 MHz	176 MHz	4 MHz
PLLx23	92 MHz	184 MHz	4 MHz
PLLx24	96 MHz	192 MHz	4 MHz
PLLx25	100 MHz	200 MHz	4 MHz

- Main clock 10 MHz (MB91461R only):

PLL Multiplier	PSCLOCK_CLKB (CLKB)	PSCLOCK_PLL (CLKVCO)	PSCLOCK_MAIN (Main Oszillation)
PLL_10_MHzx2	20 MHz	160 MHz	10 MHz
PLL_10_MHzx3	30 MHz	120 MHz	10 MHz
PLL_10_MHzx4	40 MHz	160 MHz	10 MHz
PLL_10_MHzx5	50 MHz	100 MHz	10 MHz
PLL_10_MHzx6	60 MHz	120 MHz	10 MHz
PLL_10_MHzx7	70 MHz	140 MHz	10 MHz

- Main clock 20 MHz (MB91461R only):

PLL Multiplier	PSCLOCK_CLKB (CLKB)	PSCLOCK_PLL (CLKVCO)	PSCLOCK_MAIN (Main Oszillation)
PLL_20_MHzx2	40 MHz	160 MHz	20 MHz
PLL_20_MHzx3	60 MHz	120 MHz	20 MHz

Available settings for PSDVC (CANPRE_DVC):

- The devisor can be configured in the range of 1 to 16, with 0x0 means a devisor of 1 and 0xf a devisor of 16.

Available settings for CANCLOCK (CANCKD):

- Each can clock can be enabled or disabled independently, with '0' the CAN clock can be enabled and with '1' the CAN clock can be disabled.

```
- B'0 0 0 0 0 0 0 0
     | | | | | | | |
     | | | | | | | |_____ CAN 0
     | | | | | | |_______ CAN 1
     | | | | | |_________ CAN 2
     | | | | |___________ CAN 3
     | | | |_____________ CAN 4
     | | |_______________ CAN 5
     | |_________________ reserved (always '0')
     |___________________ reserved (always '0')
```

Not every CAN-Channel is supported by every device. Please check the data sheet.

Note: If the CLKCAN source is set either to main oscillator or to PLL output then the clock for the CAN is notinfluenced by the clock modulation.If the CLKCAN source is set core clock CLKB then the clock for the CAN is also modulated (if the clock modulator is enabled).

Note: For the maximum permitted frequency, please see the data sheet.

Note: If prescaler source is selected to PLL output: Even though it is possible to select no division ratio (:1) for the divide-by-C counter it is not recommended. The resulting output clock will have an odd clock duty ratio (direct PLL output can have up to 90:10 duty). Always select at least a division ratio > 1.

Example:

The PLL x output should be used as prescaler clock source, which has in this example 128 MHz. The prescaler should output 16 MHz. All CAN clocks should be enabled.

```
#set PSCLOCKSOURCE PSCLOCK_PLL ;<<< 0x4C0h: CANPRE;     => PLLx ; 128 MHz
#set PSDVC         0x07        ;<<< 0x4C0h: CANPRE_DVC; => /8   ;  16 MHz
#set CANCLOCK      0x00        ;<<< 0x4C1h: CANCKD; all CAN Clks enabled
```

3.6 Voltage Regulator (REGULATORCTRL, REGULATORSEL)

With the following settings the behavior of the main-regulator and sub-regulator in the device modes can be configured. The main-regulator can be enabled and disabled independently for Sub-run and STOP/RTC and the sub regulator output voltage for sub-run and STOP/RTC can be controlled.

The settings may depend on the configuration of the PLL frequency, the wait state settings of the internal flash interface, the device and the access type. Therefore it is necessary to check the data sheet.

The following table shows the settings for the main and the flash regulator of MB91F469G for flash read. Please check the latest data sheet for later information.

Core Clock (CLKB)	1.8 V Operation of main regulator and Flash?	1.9 V Operation of main regulator and Flash?
to 20 MHz	Yes	Yes
to 32 MHz	Yes	Yes
to 44 MHz	Yes	Yes
to 48 MHz	No	Yes
to 88MHz	Yes	Yes
to 100MHz	No	Yes

Available settings for REGULATORCTRL (REGCTR):

```
- B'0 0 0 0 0 0 0 0
     | | | | | | | |
     | | | | | | | |_____ MAINDSBL
     | | | | | | |________ MAINKPEN
     | | | | | |___________ Reserved
     | | | | |_____________ Reserved
     | | | |_______________ MSTBO (read only)
     | | |_________________ Reserved
     | |___________________ Reserved
     |_____________________ Reserved
```

- BIT[7:5]: Reserved
- BIT[4]: MSTBO - Main regulator Standby output flag. (read only)
- BIT[3:2]: Reserved
- BIT[1]: MAINKPEN - Main Regulator enable in STOP/RTC mode.
  0 - Main regulator disabled in STOP/RTC mode [Initial value]
  1 - Main regulator enabled in STOP/RTC mode
- BIT[0]: MAINDSBL - Main Regulator disable in Sub-run Mode.
  0 - Main regulator enabled in Sub-run mode [Initial value]
  1 - Main regulator disabled in Sub-run mode

Available settings for REGULATORSEL (REGSEL):

```
- B'0 0 0 0 0 0 0 0
     | | | | | | | |
     | | | | | | | |
     | | | | | | | |_____ SUBSEL0
     | | | | | | |________ SUBSEL1
     | | | | | |___________ SUBSEL2
     | | | | |_____________ SUBSEL3
     | | | |_______________ MAINSEL
     | | |_________________ FLASHSEL
     | |___________________ Reserved
     |_____________________ Reserved
```

- BIT[7:6]: Reserved
- BIT[5]: FLASHSEL - Flash memory supply mode.
  0 - Flash memory operation mode is 1.8V [Initial value]
  1 - Flash memory operation mode is 1.9V
- BIT[4]: MAINSEL - Main Regulator supply mode.
  0 - Main regulator operation mode is 1.8V [Initial value]
  1 - Main regulator operation mode is 1.9V
- BIT[3:0]: SUBSEL[3:0] - Sub-regulator voltage level
  0111 - 1.9V +/- 0.1V
  0110 - 1.8V +/- 0.1V (initial)
  0101 - 1.7V +/- 0.1V
  0100 - 1.6V +/- 0.1V
  0011 - 1.5V +/- 0.1V
  0010 - 1.4V +/- 0.1V
  0001 - 1.3V +/- 0.1V
  0000 - 1.2V +/- 0.1V

Note: The set level of the Sub-regulator voltage is only be effective in case of Main Regulator is switched off. Otherwise (with main regulator on) the default level is applied internally by hardware to the Sub-Regulator (the register setting is not changed in this case and will be applied next time the main regulator is switched off).

Example:

Main and flash operation mode should be 1.8 V and the sub regulator voltage should be set to 1.8 V, too.

```
#set REGULATORCTRL    0x00        ;<<< 0x4CFh: REGCTR;
#set REGULATORSEL     0x06        ;<<< 0x4CEh: REGSEL;
```

3.7 Memory Controller (FLASHCONTROL, FLASHREADT, FLASHMWT2)

The memory controller must be configured, depending on the core frequency and the target device. The startup file allows configuring the registers FCHCR, FMWT and FMWT2.

The following table shows the settings for the MB91F469G for flash read. Please check the latest data sheet for later information.

Core clock (CLKB)	ATD	ALEH	EQ	WEXH	WTC
to 20 MHz	0	0	0	-	1
to 32 MHz	0	0	1	-	2
to 44 MHz	0	0	3	-	3
to 48 MHz	0	0	1	-	2
to 88 MHz	1	1	3	-	4
to 100 MHz	1	1	3	-	4

The following table shows the settings for the MB91F467D for flash read. Please check the latest data sheet for later information.

Core clock (CLKB)	ATD	ALEH	EQ	WEXH	WTC
to 24 MHz	0	0	0	0	1
to 48 MHz	0	0	1	0	2
to 96 MHz	1	1	3	0	4

Available settings for FLASHCONTROL (FCHCR):

```
- B'0000000000000000
     ||||||||||||||||
     ||||||||||||||||__ SZ0   bit
     |||||||||||||||___ SZ1   bit
     ||||||||||||||____ ENAB  bit
     |||||||||||||_____ LOCK  bit
     ||||||||||||______ PFMC  bit
     |||||||||||_______ PFEN  bit
     ||||||||||________ DBEN  bit
     |||||||||_________ FLUSH bit
     ||||||||__________ TAGE  bit
     |||||||___________ REN   bit
     ||||||____________ Reserved
     |||||_____________ Reserved
     ||||______________ Reserved
     |||_______________ Reserved
     ||________________ Reserved
     |_________________ Reserved
```

- BIT[9]: REN - Non-cacheable area Range Enable
  0 - FCHA1 defines address mask (default)
  1 - FCHA1 defines second point for the non-cacheable address range from FCHA0 to FCHA1
- BIT[8]: TAGE - TAG RAM access Enable
  0 - Memory mapped TAG RAM access disabled (default)
  1 - Memory mapped TAG RAM access enabled
- BIT[7]: FLUSH - Flush instruction cache entries
  0 - Flushing the instruction cache entries has been completed
  1 - Actually flushing the instruction cache entries
- BIT[6]: DBEN - Data Buffer Enable
  0 - Buffering of read data is disabled (default)
  1 - Buffering of read data is enabled
- BIT[5]: PFEN - PreFetch Enable
  0 - Prefetch of instructions is disabled (default)
  1 - Prefetch of instructions is enabled

- BIT[4]: PFMC - Prefetch Miss Cache enable
  0 Standard cache algorithm (default)
  Prefetch misses are cached only
- BIT[3]: LOCK - Global lock of cache entries
  0 - Write of cache entries enabled (default)
  1 - Writing of cache entries is disabled, the cache contents is locked
- BIT[2]: ENAB - Instruction cache enable
  0 - The instruction cache is disabled (default)
  1 - Enable the instruction cache
- BIT[1:0]: SZ[1:0] - Cache size configuration (only valid for MB91V460)
  00 - 0kByte - Cache disabled
  01 - 4kByte (1024 entries)
  10 - 8kByte (2048 entries)
  11 - 16kByte (4096 entries) (default)

Available settings for FLASHREADT (FMWT):

```
- B'0000000000000000
     ||||||||||||||||
     ||||||||||||||||__ EQ0   bit
     |||||||||||||||___ EQ1   bit
     ||||||||||||||____ EQ2   bit
     |||||||||||||_____ EQ3   bit
     ||||||||||||______ ATD0  bit
     |||||||||||_______ ATD1  bit
     ||||||||||________ ATD2  bit
     |||||||||_________ FRAM  bit
     ||||||||__________ WTC0  bit
     |||||||___________ WTC1  bit
     ||||||____________ WTC2  bit
     |||||_____________ WTC3  bit
     ||||______________ WEXH0 bit
     |||_______________ WEXH1 bit
     ||________________ WTP0  bit
     |_________________ WTP1  bit
```

- WTP[1:0]: Wait cycles for FLASH in page access
- WEXH[1:0]: Minimum WEX High timing requirement
- WTC[3:0]: Wait cycles for FLASH memory access
- FRAM: Wait cycles for F-Bus general purpose RAM memory access
- ATD[2:0]: Duration of the ATDIN signal for FLASH memory access
- EQ[3:0]: Duration of the EQIN signal for FLASH memory access

Available settings for FLASHMWT2 (FMWT2):

```
- B'00000000
     ||||||||
     ||||||||__ Reserved
     |||||||___ ALEH2
     ||||||____ ALEH1
     |||||_____ ALEH0
     ||||______ Reserved
     |||_______ Reserved
     ||________ Reserved
     |_________ Reserved
```

- ALEH[2:0]: Duration of the ALEH time for FLASH memory access

Example:

The memory controller should be configured for the MB91F467D. The core frequency should be 64 MHz.

```
#set FLASHCONTROL    0x032       ;<<< 0x7002h: FCHCR;
#set FLASHREADT      0xC413      ;<<< 0x7004h: FMWT;
#set FLASHMWT2       0x10        ;<<< 0x7006h: FMWT2;
```

4 Section and Data Declaration

The ANSI specification requires that global data are initialized. This has to be done by the startup code. However, the compiler has to provide a mechanism to collect those data, which have to be cleared (set to zero) and those data, which have to be pre-loaded with a value. The FCC911S provides default sections, which contain dedicated groups of data. A section is a unit that can be handled by the linker. It either contains initialized data or reserved areas. All sections used in an application can be found in the linkage map after linking project.

4.1 Default Sections

The compiler FCC911S has the following default sections

Section Type	Section Name	Type	Boundary Alignment [Byte]	Write	Initial Value
Code section	CODE	CODE	2	Disabled	Provided
Initialized section	INIT	DATA	4	Enabled	Provided
Constant section	CONST	CONST	4	Disabled	Provided
Data section	DATA	DATA	4	Enabled	Not Provided
I/O section	IO	IO	4	Enabled	Not Provided
Vector section	INTVECT	CONST	4	Disabled	Provided
C++ Init section	EXT_CTOR_DTOR	CONST	4	Disabled	Provided

Code section:

The code section stores machine codes. This section corresponds to the procedure section for the C language.

Initialized section:

The initialized section stores the initial value attached variable area. For the C language, this section corresponds to the area for external variables without the const attribute, static external variables, and static internal variables. The ANSI specification requires that global data are initialized. This is done during the startup code.

Constant section:

The constant section stores the write-protected initial value attached variable area. For the C language, this section corresponds to the area for const attribute attached external variables, static external variables, and static internal variables.

Data section:

The data section stores the area for variables without the initial value. For the C language, this section corresponds to the area for external variables (including those which are with the const attribute), static external variables, and static internal variables. The ANSI specification requires that data section must be cleared. This is done during the startup code.

I/O section:

I/O section stores the area for the __io-qualified variables. For the C language, this section corresponds to the area for __io-qualified external variables (including those which are provided with the const attribute), static external variables, and static internal variables. The default section name is IO.

Vector section:

This section is where interrupt vector tables are stored. In C, a vector table is generated only when its generation is specified in #pragma intvect. The default section name is INTVECT.

C++ Init section:

This section is where tables for indicating the entry of functions constituting and destroying static objects are stored. It must be used at startup.

4.2 Additional Sections

Beside the default sections, the startup defines, depending on the settings several additional settings.

Section Type	Section Name	Type	Boundary Alignment [Byte]	Write	Initial Value
Start section	CODE_START	CODE	4	Disabled	Provided
System stack	SSTACK	DATA	4	Enabled	Not Provided
User stack	USTACK	DATA	4	Enabled	Not Provided
Security vectors	SECURITY_VECTORS	CODE	4	Disabled	Provided
Instruction RAM Section	IRAM	CODE	4	Disabled	Provided

Start section:

This section stores the startup.

System stack:

This section is where the system stack is located. The size of the system stack can be defined in the startup code.

User stack:

This section is where the user stack is located. The size of the user stack can be defined in the startup code.

Security vectors:

This section stores the security vectors.

Instruction RAM Section:

This section stores the code, which should be executed in the IRAM.

5 Document History

Document Title: AN205200 - FR Family MB91460 Series, Start91460.asm

Document Number: 002-05200

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	03/30/2007	Initial release
*A	5084078	NOFL	01/13/2016	Migrated Spansion Application Note from MCU-AN-300021-E-V10 to Cypress format
*B	5868944	AESATMP9	08/31/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2007-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.