

TriCore

AP32178

cstart

Application Note V1.0 2011-08

Microcontrollers

Edition 2011-08 Published by Infineon Technologies AG 81726 Munich, Germany © 2011 Infineon Technologies AG All Rights Reserved.

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (www.infineon.com).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.



TriCore	
Revision H	istory: V1.0, 2011-08
Previous V	ersion: none
Page	Subjects (major changes since last revision)

We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to: mcdocu.comments@infineon.com

Application Note



Table of Contents

Table of Contents

1	Preface	5
2	Introduction	5
3	Overview	5
4	Implementation and Usage	9



1 Preface

This application note describes a user's startup code implementation on the TriCore processor architecture [1,2] for the AUDO MAX-family. The document is aimed at developers who write or design applications for the TriCore.

This application note assumes that readers have access to the TriCore Architecture Manual [1] and TriCore User Manual [3-6], and have at least some knowledge about the following sections of the user's manual:

- Startup SoftWare (SSW) (see section BootROM content in [3-5])
- Clock system of the System Control Unit (SCU) (see section Clock System overview in [3-5])
- ENDINIT protection and watchdog timer (WDT) (see section Watchdog Timer in [3-5])
- The TriCore instruction set.

See References on Page 13 for more information on the TriCore manuals and other relevant documentation.

2 Introduction

Compilers for the TriCore processor are available by third party Infineon tool partners and offers user's startup code with their tool chain. It is provided as C source code or assembler source code. The source file for the user's startup code named cstart.c for Tasking, crt0.S for Hightec and crt0.s for Wind River. This application note is written explicitly for Tasking users. It improves and extends the default Tasking startup file cstart.c in four ways. First it improves the PLL initialization and implements a program flow exactly as described in the user' manual. Second it extends the number of registers that could be configured in the startup code. It especially offers configured only once at startup. Grouping them together makes it possible to clear and set the ENDINIT bit only once. This practice saves execution time which is often critical at startup. An endinit_clear()/endinit_set() programming sequence typically requires about 0.5 µs running at 180 MHz CPU frequency. Third a fast ENDINIT bit clear and set routines are offered as inline functions. Fourth the cstart.h header files comes with PLL initialization values for most popular configurations of the TriCore AUDO-MAX family. To limit the jump of the dynamic current consumption the PLL initialization uses a ramp-up sequence.

Together these modifications of the default Tasking startup code give the user a quick start programming the TriCore. With entering the C main() function the processor is already running at the configured CPU frequency and configured modules frequencies.

3 Overview

The PLL uses two different start-up mechanisms depending on the triggering reset. Upon a power-on reset the PLL starts to supply the system in Precscaler Mode. The starting frequency is 16.6 MHz. A system reset brings the PLL control register in the SCU to the defined reset values and the system clock operates in free-running mode at $f_{VCOBASE}/16$. In both cases the SSW in the BootROM restores the clock system to free-running mode before jumping to the user's startup code located at the User STartup ADDress STADD. Tasking named this address the RESET vector. Two addresses are valid: 0xA000000 for starting from internal flash memory module (internal start) or 0xA1000000 for starting from external EBU space (external start). The SSW therefore evaluates the HWCFG[7..0] pins. For external start the EBU reads its configuration parameter from internal memory 000004_H (see section External Bus Unit in [7], chapter 'Boot Process' respectively 'Configuration Word Fetch Process').

The major design goal of the user's startup code is to initialize the processor and to bring up the PLL quickly, to configure major CSFR and other ENDINIT protected SFR registers. The steps are illustrated in Figure 1. The changes made to the original code are mainly related to the PLL ramp-up sequence and the ability to configure more ENDINIT protected sfr registers. The execution time on a TC1798 running at 300 MHz CPU frequency of the startup code is about 250-350 μ s, where the largest single part (230 μ s) is the ramp-up sequence using six steps with a delay in between two steps of 20 μ s. Details of the PLL ramp-up sequence are illustrated in Figure 2. A block diagram of the Clock Generation Unit (CGU) is shown in Figure 3. The current consumption during a PLL ramp-up sequence with just four steps is shown in Figure 4. Formulas for the dynamic current consumption are given in the data sheet.

The internal Watchdog starts after reset in Time-Out Mode. With the startup code presented by this application node the watchdog would enter Prewarning Mode after $4 \times f_{FPI}/16384$ which is measured to 950 µs. The



Overview

execution time of the startup code as configured in this application note is less than 300 µs. To safe time the watchdog is serviced and the ENDINIT bit is set after all ENDINIT protected registers are configured.

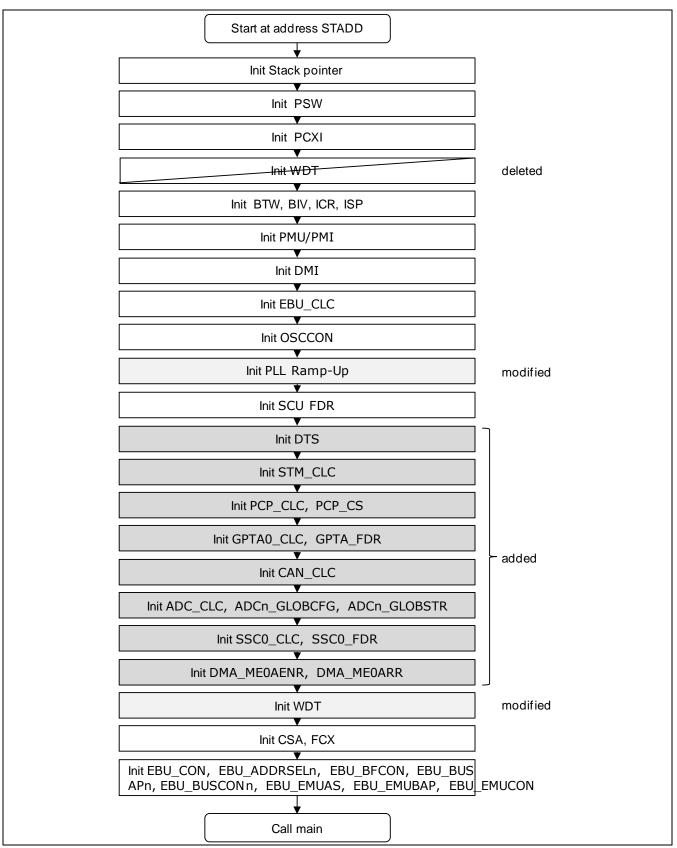


Figure 1 Startup code Flow Diagram



Overview

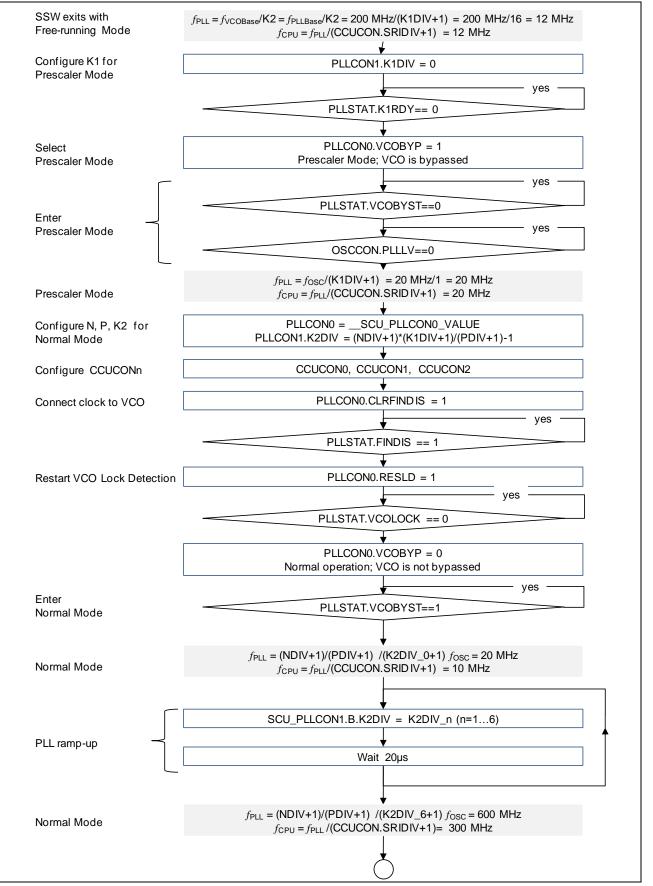
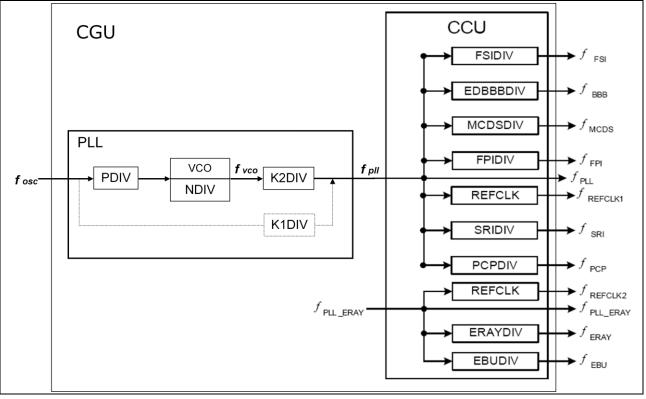


Figure 2 PLL initialization Flow Diagram (TC1798 300MHz)



Overview





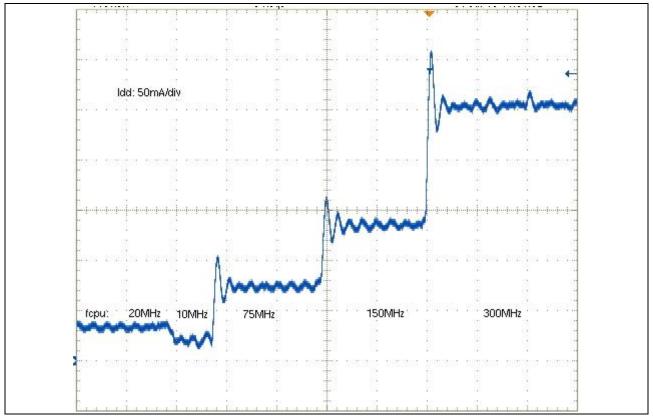


Figure 4 Current consumption during frequency Ramp-up sequence.



The implementation follows the default Tasking startup file cstart.c but modifies or adds certain parts as explained in section 3. The cstart.c and cstart.h files that come with this application notes replaces the Tasking C startup files

The new cstart.h header file offers popular configurations for the AUDO-MAX TriBoards.

- TC172x 80/132 MHz
- TC178x 132/180 MHz
- TC179x 240/270/300 MHz

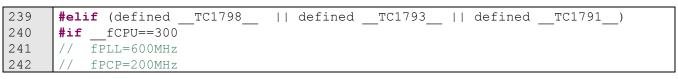
To enable one of these configuration the control program cctc should be called with option __<TriCore Derivative>__ and __fCPU=<frequency[MHz]>, for example $-D_TC1798_ -D_fCPU=300$. Select **Project > Properties** and navigate to **C/++ Build > Settings > C/C++ Compiler > Preprocessing** and add these symbols to the list of defined symbols (Figure 5).

These macros will select the appropriate settings in cstart.h. Listing 1 shows this configuration for the TC1798 running at 300MHz.

More changes to the cstart.h are also reflected by more options in the cstart editor within the Tasking EDE. Figure 6 for example shows the register page with ADC, CAN, GPTA, SCU, SSC and STM registers.

Details of the configurations are listed in Table 1 to Table 3.

type filter text	Settings 🗘 🗢 🖛
Resource Builders C/C++ Build Build Variables Discovery Options Environment Flash Processor Settings C/C++ General Project References Refactoring History Run/Debug Settings	Configuration: Debug [Active] Manage Configurations Tool Settings Build Steps Build Artifact Binary Parsers Error Parsers Image Global Options Image Configuration of '.sfr' file Image Configurations Automatic inclusion of '.sfr' file Image Configurations Image Configuration of '.sfr' file Image Configuration Store preprocessor output in <file>.pre Image Configuration Image Configuration Image Configuratin Image Configuration</file>
?	OK Cancel





244	// fFSI	=150MHz	
245	// fSRI	=300MHz	
246	// fFPI	=100MHz	
247	// fEDB	BB=150MHz	
248	// fref	CLK=25MHz	
249	// fMCD	S=150MHz	
250	// fEBU	=75MHz	
251	// fera	Y=300MHz	
252	// fout	=25MHz	
253	#define	SCU_PLLCON0_INIT	1
254	#define	SCU_PLLCON0_VALUE	0x1017600
255	#define	SCU_PLLCON1_INIT	1
256	#define	SCU_PLLCON1_VALUE	0x0
257	#define	SCU_PLLK2RAMPUP_INIT	1
258	#define	SCU_PLLK2RAMPUP_VALUE	0x08040201
259	#define	SCU_PLLK2RAMPUP_WAIT	6000
260	#define	SCU_CCUCON0_INIT	1
261	#define	SCU_CCUCON0_VALUE	0x2030105
262	#define		1
263	#define	SCU_CCUCON1_VALUE	0x30B03
264	#define	SCU_CCUCON2_INIT	1
265	#define	SCU_CCUCON2_VALUE	0x701
266		SCU_FDR_INIT	1
267	#define	SCU_FDR_VALUE	0x43FE
268	#define	FLASH0_FCON_INIT	1
269		FLASH0_FCON_VALUE	0x00074804
270		FLASH1_FCON_INIT	1
271	#define	FLASH1 FCON VALUE	0x00074804

Listing 1 PLL specific configuration in cstart.h. TC1798 with fCPU=300Mhz shown

c1798 Registers	SCU_PL	LCON0: PLL Configuration 0 Register	(default: 0x0001c600) —	
▷ ADC ▷ BUS	Value:	0x1017600 Default	🔽 Initia	lize in startup code
⊳ CAN ⊳ CSFR	Bit#	Description	Value	Access
> EBU	0	VCOBYP: VCO Bypass	Normal operation,	rw
> GPTA	1	VCOPWD: VCO Power Saving Mo	Normal behavior	rw
> OCDS	2	MODEN: Modulation Enable	Frequency modul	rw
PCP2	4	SETFINDIS: Set Status Bit PLLSTA	Bit PLLSTAT.FINDI	w
⊿ SCU	5	CLRFINDIS: Clear Status Bit PLLST	Bit PLLSTAT.FINDI	w
SCU_CCUCON0: CCU Clock Control Register 0	6	OSCDISCDIS: Oscillator Disconne	In case of a PLL Io	rw
SCU_CCUCON1: CCU Clock Control Register 1	9-15	NDIV: N-Divider Value	0x3b	rw
SCU_CCUCON2: CCU Clock Control Register 2	18	RESLD: Restart VCO Lock Detection	0x0	w
SCU_FDR: Fractional Divider Register SCU OSCCON: OSC Control Register	24-27	PDIV: P-Divider Value	0x1	rw
SCU_DISCCON: DISC Control Register				
SCU_PLLCON0: PLL Configuration 0 Register				
SCU PLLK2RAMPUP: PLL K2 Ramp up Register				
SCU TRAPDIS: Trap Disable Register				
▷ SSC				
⊳ STM	6	3		
CPU Reset Values				
Set CPU defaults				

Figure 6 start editor: Register page



Table 1	TC179x PLL configuration examples
---------	-----------------------------------

Parameter	TC179x 300MHz	TC179x 270MHz	TC179x 240MH
Clock Diver Option	3	3	3
fosc	20 MHz	20 MHz	20 MHz
<i>f</i> vcobase	200 MHz	200 MHz	200 MHz
PLLCON0.PDIV	1	1	1
PLLCON0.NDIV	0x3B	0x35	0x2F
PLLCON1.K1DIV	0	0	0
PLLCON1.K2DIV	0	0	0
CCUCON0.PCPDIV	2	2	2
CCUCON0.FSIDIV	3	3	3
CCUCON0.SRIDIV	1	1	1
CCUCON0.FPIDIV	5	5	5
CCUCON1.EDBBBDIV	3	3	3
CCUCON1.REFCLKDIV	0xB	0xB	0xB
CCUCON1.MCDSDIV	3	3	3
CCUCON2.EBUDIV	7	7	7
FDR.STEP	0x3FE	0x3FE	0x3FF
$f_{VCO} = f_{OSC} \times (NDIV+1)/(PDIV+1)$	600 MHz	540 MHz	480 MHz
$f_{PLL} = f_{OSC} \times (NDIV+1)/((PDIV+1)^*(K2DIV+1))$	600 MHz	540 MHz	480 MHz
$f_{PCP} = f_{PLL} / (PCPDIV+1)$	200 MHz	180 MHz	160 MHz
$f_{FSI} = f_{PLL} / (FSIDIV+1)$	150 MHz	135 MHz	120 MHz
$f_{SRI} = f_{PLL} / (SRIDIV+1)$	300 MHz	270 MHz	240 MHz
$f_{\text{FPI}} = f_{\text{PLL}} / (\text{FPIDIV+1})$	100 MHz	90 MHz	80 MHz
$f_{\text{EDBBB}} = f_{\text{PLL}} / (\text{EDBBBDIV+1})$	150 MHz	135 MHz	120 MHz
$f_{REFCLK} = f_{PLL} / 2 / (REFCLKDIV+1)$	25 MHz	22.5 MHz	20 MHz
$f_{MCDS} = f_{PLL} / (MCDSDIV+1)$	150 MHz	135 MHz	120 MHz
$f_{EBU} = f_{PLL} / (EBUDIV+1)$	75 MHz	67.5 MHz	60 MHz
$f_{\text{ERAY}} = f_{\text{PLL}} / (\text{ERAYDIV+1}) =$	300 MHz	270 MHz	240 MHz
$f_{OUT} = f_{FPI} \times (1/(0x400\text{-}STEP)) =$	25 MHz	22.5 MHz	40 MHz
PLL ramp up sequence	6 steps: 20, 66.7, 120, 200, 300, 600 MHz	6 steps: 20, 67.5,135,180, 270, 540 MHz	5 steps: 20, 68.6, 120, 24 480 MHz



Parameter	TC178x 180MHz	TC178x 132MHz	
Clock Diver Option	2	1	
fosc	20 MHz	20 MHz	
fvcobase	200 MHz	200 MHz	
PLLCON0.PDIV	1	1	
PLLCON0.NDIV	0x47	0x41	
PLLCON1.K1DIV	0	1	
PLLCON1.K2DIV	3	4	
CCUCON0.PCPDIV	0	0	
CCUCON0.LMBDIV	0	0	
CCUCON0.FPIDIV	1	1	
CCUCON1.REFCLKDIV	0xB	0xB	
CCUCON1.MCDSDIV	1	1	
FDR.STEP	0x3FE	0x3FF	
$f_{VCO} = f_{OSC} \times (NDIV+1)/(PDIV+1)$	720 MHz	540 MHz	
$f_{PLL} = f_{OSC} \times (NDIV+1)/((PDIV+1)^*(K2DIV+1))$	180 MHz	540 MHz	
$f_{PCP} = f_{PLL} / (PCPDIV+1)$	180 MHz	180 MHz	
$f_{LMB} = f_{PLL}/(LMBDIV+1)$	180 MHz	270 MHz	
$f_{\text{FPI}} = f_{\text{PLL}} / (\text{FPIDIV+1})$	90 MHz	90 MHz	
$f_{\text{REFCLK}} = f_{\text{PLL}} / 2 / (\text{REFCLKDIV+1})$	7.5 MHz	22.5 MHz	
$f_{MCDS} = f_{PLL} / (MCDSDIV+1)$	90 MHz	135 MHz	
$f_{OUT} = f_{FPI} \times (1/(0x400-STEP))=$	22.5 MHz	22.5 MHz	
PLL ramp up sequence	3 steps: 20, 120, 180 MHz	3 steps: 20, 110, 132 MHz	

Table 2 TC178x PLL configuration examples



References

Parameter	TC172x 132MHz	TC172x 80MHz	
Clock Diver Option	2	1	
fosc	20 MHz	20 MHz	
fvcobase	200 MHz	200 MHz	
PLLCON0.PDIV	1	1	
PLLCON0.NDIV	0x41	0x3F	
PLLCON1.K1DIV	0	0	
PLLCON1.K2DIV	4	7	
CCUCON0.PCPDIV	0	0	
CCUCON0.LMBDIV	0	0	
CCUCON0.FPIDIV	1	0	
CCUCON1.REFCLKDIV	0xB	0xB	
CCUCON1.MCDSDIV	1	1	
CCUCON2.ERAYDIV	1	1	
FDR.STEP	0x3FF	0x3FF	
$f_{VCO} = f_{OSC} \times (NDIV+1)/(PDIV+1)$	660 MHz	640 MHz	
$f_{PLL} = f_{OSC} \times (NDIV+1)/((PDIV+1)^*(K2DIV+1))$	132 MHz	80 MHz	
$f_{PCP} = f_{PLL} / (PCPDIV+1)$	132 MHz	80 MHz	
$f_{\text{LMB}} = f_{\text{PLL}} / (\text{LMBDIV+1})$	132 MHz	80 MHz	
$f_{\text{FPI}} = f_{\text{PLL}} / (\text{FPIDIV+1})$	66 MHz	80 MHz	
$f_{\text{REFCLK}} = f_{\text{PLL}} / 2 / (\text{REFCLKDIV+1})$	5.5 MHz	3.33 MHz	
$f_{MCDS} = f_{PLL} / (MCDSDIV+1)$	66 MHz	40 MHz	
$f_{\text{ERAY}} = f_{\text{PLL}} / (\text{ERAYDIV+1}) =$	66 MHz	40 MHz	
$f_{OUT} = f_{FPI} \times (1/(0 \times 400\text{-STEP}))=$	33 MHz	40 MHz	
PLL ramp up sequence	3 steps: 20, 110, 132 MHz	2 steps: 20, 80 MHz	

Table 3 TC172x PLL configuration examples

5 References

[1] TriCore Architecture V1.3.8 2007-11, Infineon Technologies AG

- [2] http://www.infineon.com/tricore
- [3] TC1784 User's Manual V1.0 2009-07, Infineon Technologies AG
- [4] TC1798 User's Manual V1.1 2011-03, Infineon Technologies AG
- [5] TC1728 User's Manual V1.0D1 2011-03, Infineon Technologies AG

www.infineon.com

Published by Infineon Technologies AG