Table of Contents

1.	easyDSP ?	
2.	Products type	5
3.	starting easyDSP	8
4.	Revision History	9
5.	Limitation	16
6.	Pod configuration	18
7.	How to use MCU	21
	7.1 C28x	21
	7.1.1 C28x programming	21
	7.1.2 C28x board setting	38
	7.1.3 How to use other SCI port than designated	49
	7.1.4 C28x cautions	51
	7.2 STM32	51
	7.2.1 STM32 programming	51
	7.2.2 STM32 hardware	59
	7.2.3 STM32 dual core	62
	7.2.4 STM32 RAM booting	68
	7.2.5 STM32 cautions	71
	7.3 S32	72
	7.3.1 S32K1 + SDK	72
	7.3.2 S32K/S32M + RTD	79
	7.4 AM263x	95
	7.4.1 AM263x software	95
	7.4.2 AM263x hardware	105
	7.5 TM4C	108
	7.6 MSPM0	112
	7.7 PSoC4	119
	7.7.1 PSoC4 software	119
	7.7.2 PSoC4 hardware	130
	7.8 XMC1	130

7.9 XMC4
7.10 RA135
7.10.1 RA hardware135
7.10.2 RA sofrware136
7.10.3 RA0143
7.11 RX146
7.11.1 RX hardware146
7.11.2 RX sofrware147
7.12 TX
7.13 TXZ3160
7.14 LPC
7.15 Cautions164
8. Menus
8.1 Project165
8.2 Edit
8.3 MCU
8.3.1 Common169
8.3.2 C28x
8.3.3 STM32177
8.3.4 S32179
8.3.5 AM263x182
8.3.6 TM4C184
8.3.7 MSPM0185
8.3.8 PSoC4187
8.3.9 XMC1
8.3.10 XMC4189
8.3.11 RA190
8.3.12 RX
8.3.13 TX, TXZ3194
8.3.14 LPC
8.4 Tools
8.5 Window

	8.6 Help	198
9. \	Windows	199
	9.1 Command	
	9.2 Watch	201
	9.3 Plot	203
	9.4 Chart	207
	9.5 Record	208
	9.6 Memory	210
	9.7 Array	211
	9.8 Tree	212
10.	. Trouble Shooting	212
	10.1 Common	212
	10.2 C28x	214
	10.3 STM32	219
11.	. Tips	221
	11.1 DA converter	221
	11.2 Others	223
	11.3 FAQ	223
12.	. Driver	224
	12.1 Driver Installation	224
	12.2 Driver Uninstallation	

1. easyDSP ?

Welcome to

easyDSP

for DSP developer, by DSP developer

Welcome to easyDSP for real-time MCU debugging

'easyDSP' is a powerful graphical user interface (GUI) for the maintaining, configuring and troubleshooting of embedded software with strict real-time requirements. The tool automatically extracts the symbol information from the files generated by the cross-compiler and presents the user with windows for the viewing, editing, logging and graphing of those symbols, in real-time, while the target software is executing. easyDSP communicates with the target MCU over a serial communication link, typically SCI (or USART). On the target, only a small "remote agent" needs to be called periodically in the background task. Since the remote agent runs on spare processor cycles, it does not interfere with the interrupt driven part of the software. This makes the tool ideal for interfacing with power electronics control software, where the control tasks need to be executed uninterruptedly and with minimal latency. The fact that easyDSP does not depend on JTAG/SWD for communicating with the target makes the tool operate reliably in environments with strong EMI and/or high-voltage isolation requirements. easyDSP can supports multiple operation so that you can control several MCU boards by using several easyDSPs in single PC.

easyDSP is designed for the real-time communication between MCU and an IBM PC or compatible running 64bits Windows . Supporting MCU :

- TI : C28x, TM4C, MSPM0 and AM263x series

- ST : STM32 series
- Infineon : PSoC4, XCM1 and XMC4 seriels
- Renesas : RA, RX series
- Toshiba : TX and TXZ3 series
- NXP : LPC1x00, S32K, S32M series

The detailed information is available \underline{here} . For the support of other MCU, please contact easydsp@gmail.com.

Customers are

universities in Korea (Seoul National, HanYang, SungKyunKwan, Kangwon, Busan, KAIST, ...), companies in Korea (Samsung, LG, Hyundai, LS, Onsemi, Infineon, ...),

company outside Korea (Yaskawa, Raytheon Technologies, Collins Aerospace, Carrier, General Motors, Delphi, Grid-bridge, R&D Dynamics, ADI American Distributors Ltd ...)

university outside Korea (FEEC@ECE, Virginia Tech)

easyDSP is not freeware. But it is provided "AS IS" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event shall easyDSP be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if easyDSP has been advised of the possibility of such damages. It is the user's responsibility to check for future updates to the easyDSP and to use the latest version.

For more information, visit <u>www.easydsp.com</u> or

mail to <u>easydsp@gmail.com</u> for program bug, improvement idea, and other technical inquiry, hr.oh@egreenpower.com for purchsing, AS, easyDSP Pod hardware inquiry.

Thank you.

Acknowledgment :

This software is based in part on the work of the Independent JPEG Group. This software is based in part on the work of the FreeType Team. This software is based on pugixml library (<u>http://pugixml.org</u>). pugixml is Copyright (C) 2006-2018 Arseny Kapoulkine.

2. Products type



In below, you need to purchase easyDSP type by type.

Туре	Supporting MCU	MCU Communication Channel	Comments
type 1	TI C28x	SCI	Standard isolation type. Digital isolator is used inside easyDSP pod for isolation purpose.
type 2	TI C28x	SCI	Optic cable isolation type. Stable long-distance communication with optic cable (HFBF1414Z/HFBF2412Z). Cable distance : variable(upto 200m) upon request.
type 3	ST STM32 TI AM263x TI TM4C TI MSPM0 Infineon PSoC4 Infienon XMC1 Infineon XMC4	USART or UART or SCI	Standard isolation type. Digital isolator is used inside easyDSP pod for isolation purpose.

Renesas RA	
Renesas RX	
Toshiba TX	
Toshiba TXZ3	
NXP LPC1x00	
NXP S32	

Please check the part number of MCU below. notation : [a,b] = a or b, x = any

MCU	part number
TI C28x	TMS320F280[1,2,6,8,9], TMS320F2801[5,6], TMS320F28044, TMS320F281[0,1,2], TMS320F2802[20,30,60,70], TMS320F2802[0,1,2,3,6,7,00], TMS320F2803[0,1,2,3,4,5],TMS320F2805[0,1,2,3,4,5],TMS320F2806[2,3,4,5,6,7,8,9],TMS32 0F2807[4,5,6], TMS320F28001[32,33,35,37], TMS320F28001[52,53,54,55,56,57], TMS320F28002[1,2,3,4,5],TMS320F28003[3,4,6,7,8,9],TMS320F28004[0,1,5,8,9],TMS320F2 823[2,4,5],TMS320F2833[2,3,4,5],TMS320C2834[1,2,3,4,5,6],TMS320F2837[4,5,6,7,8,9]S,T MS320F2837[4,5,6,7,8,9]D,TMS320F2838[4,6,8]S, TMS320F2838[4,6,8]D, TMS320F28P55xS[D,G,J], TMS320F28P65x[S,D][H,K]
TI AM263x	AM263[1,2,4]
TI TM4C	TM4C123[0,1,2,3][C3,D5,E6,H6], TM4C123[6,7][D5,E6,H6], TM4C123[A,B,F,G][E6,H6], TM4C129[0,2]NC, TM4C1294[K,N]C, TM4C1297NC, TM4C1299[K,N]C, TM4C129[C,D]NC, TM4C129E[K,N]C, TM4C129LNC, TM4C129X[K,N]C
TI MSPM0	MSPM0Lxxx[3,4,5,6,7], MSPM0Gxxx[5,6,7]
ST STM32	 STM32C011x[4,6], STM32C031x[4,6], STM32C051x[6,8] STM32C091x[B,C] STM32C091x[B,C] STM32F030x[4,6,8,C], STM32F031x[4,6], STM32F031x6, STM32F038x6, STM32F042x[4,6], STM32F048x6, STM32F051x[4,6,8], STM32F058x8, STM32F070x[6,B], STM32F042x[4,6], STM32F072x[8,B], STM32F078xB, STM32F091x[B,C], STM32F098xC, STM32F100x[4,6,8,B], STM32F103x[4,6,8,B,C,D,E,F,G], STM32F105x[8,B,C], STM32F102x[4,6,8,B], STM32F103x[4,6,8,B,C,D,E,F,G], STM32F105x[8,B,C], STM32F107x[B,C], STM32F205x[B,C,E,F,G], STM32F207x[C,E,F,G], STM32F215x[E,G], STM32F301x[6,8], STM32F302x[6,8,B,C,D,E], STM32F303x[6,8,B,C,D,E], STM32F318x8, STM32F301x[6,8], STM32F302x[6,8,B,C,D,E], STM32F373x[8,B,C], STM32F318x8, STM32F401x[B,C,D,E], STM32F405x[E,G], STM32F407x[E,G], STM32F401x[B,C,D,E], STM32F405x[E,G], STM32F407x[E,G], STM32F410x[8,B], STM32F411x[C,E], STM32F407x[E,G], STM32F413x[G,H], STM32F415xG, STM32F410x[8,B], STM32F411x[C,E], STM32F427x[G,I], STM32F413x[G,H], STM32F437x[G,I], STM32F410x[8,B], STM32F423xH, STM32F427x[G,I], STM32F479x[G,I], STM32F437x[G,I], STM32F722x[C,E], STM32F46x[C,E], STM32F730x8, STM32F732xE, STM32F733xE, STM32F767x[G,I], STM32F769x[G,I], STM32F750x8, STM32F756xG, STM32F765x[G,I], STM32G030x[6,8], STM32G031x[4,6,8], STM32G071x[8,B], STM32G081xB, STM32G080xE, STM32G081x[6,C,E], STM32G071x[6,8], STM32G431x[6,8,B], STM32G080xE, STM32G081x[6,C,E], STM32G071x[6,E], STM32G431x[6,8,B], STM32G441xB, STM32G081x[6,C,E], STM32G071x[8,B], STM32G484xE, STM32G491x[C,E], STM32G474x[B,C,E], STM32G431x[6,8,B], STM32G491x[C,E], STM32G474xE, STM32G491x[C,E], STM32G474xE, STM32G491x[C,E], STM32G474xE, STM32G491x[C,E], STM32G474xE, STM32H503xB, STM32G62xI, STM32H563xE, STM32H563xI, STM32H573xI,

	STM32H723x[E,G], STM32H725x[E,G], STM32H730xB, STM32H733xG, STM32H735xG, STM32H742x[G,I], STM32H743x[G,I], STM32H745x[G,I], STM32H747x[G,I], STM32H750xB, STM32H753xI, STM32H755xI, STM32H757xI, STM32H7A3x[G,I], STM32H7B0xB, STM32H7B3xI, STM32H010x[4 6 8 B], STM32L011x[3 4], STM32L021x4, STM32L031x[4 6]
	STM32L010x[4,6,8,B], STM32L011x[3,4], STM32L021x4, STM32L031x[4,6], STM32L041x6, STM32L051x[6,8], STM32L052x[6,8], STM32L053x[6,8], STM32L06[2,3]x8, STM32L07[1,2,3]x[8,B,Z], STM32L08[1,2]x[B,Z], STM32L083x[8,B,Z], STM32L100x[6,8,B,C], STM32L151x[6,8,B,C,E], STM32L152x[6,8,B,C,D,E], STM32L162x[C,D,E], STM32L100x[8,B]-A, STM32L151x[6,8,B,C]-A, STM32L152x[6,8,B,C]-A, STM32L162xC-A, STM32L15[1,2]xD-X, STM32L162xD-X, STM32L412x[8,B], STM32L422xB, STM32L431x[B,C], STM32L432x[B,C], STM32L433x[B,C], STM32L442xC, STM32L443xC, STM32L451x[C,E], STM32L452x[C,E], STM32L462xE, STM32L471x[E,G], STM32L475x[C,E,G], STM32L476x[C,E,G], STM32L486xG, STM32L496AE, STM32L496x[E,G], STM32L4A6xG, STM32L4P5x[E,G], STM32L4Q5xG, STM32L4R5x[G,I], STM32L4R7xI, STM32L4A6xG, STM32L45[5,7,9]xI, STM32L552x[C,E], STM32L4562xE, STM32U031x[4,6,8], STM32U073x[8,C], STM32U083xC, STM32U375x[E,G] MEM, STM32U385xG MEM, STM32U535x[B,C,E], STM32U545xE, STM32U575x[G,I], STM32U585xI, STM32U575x[G,I], STM32U585xI, STM32U595x[I,J], STM32U599x[I,J], STM32U5A5x[I,J],
	STM32U5A9xJ, STM32U5F[7,9]xJ, STM32U5G[7,9]xJ, STM32WB[10,15]xC, STM32WB30xE, STM32WB35x[C,E], STM32WB50xG, STM32WB55x[C,E,G,Y], STM32WB05xZ, STM32WB06xC, STM32WB07xC, STM32WB09xE, STM32WBA50xG NEW, STM32WBA52x[E,G], STM32WBA5[4,5]x[E,G], STM32WL33x[8,B,C], STM32WL5[4,5]xC, STM32WLE[4,5]x[8,B,C]
Infineon PSoC4	CY8C402[4,5], CY8C404[5,6], CY8C412[4,5],CY8C4126xxx-S42x, CY8C4126xxx-S43x, CY8C4126xxx-S44x, CY8C4126xxx-S45x, CY8C4126xxx-Mxxx, CY8C4127xxx-Sxxx, CY8C4127xxx-Mxxx, CY8C4127xxx-BLxxx, CY8C4128xxx-Sxxx, CY8C4128xxx-BLxxx, CY8C414[5,6,7,8], CY8C424[4,5], CY8C4246xxx-DSxxx, CY8C4246xxx-Mxxx, CY8C4246xxx- Lxxx, CY8C4247xxx-Mxxx, CY8C4247xxx-Lxxx, CY8C4247xxx-BLxxx, CY8C4248xxx-Lxxx, CY8C4248xxx-BLxxx, CY8C454[6,7,8], CY8C472[4,5], CY8C474[4,5]
Infienon XMC1	XMC1100-xxxxx0[008,016,032,064], XMC120x-xxxxx0[016,032,064,128,200], XMC130x-xxxxx0[016,032,064,128,200], XMC140x-xxxxx0[032,064,128,200],
Infineon XMC4	XMC410x-xxxx[64,128], XMC4200-xxxx256, XMC4300-xxxxx256, XMC440x-xxxxx[256,512], XMC450x-xxxxx[512,768,1024], XMC4700-xxxxx[1536,2048], XMC4800- xxxxx[1024,1536,2048]
Renesas RX	R5F5110[1,3,4,5,H,J], R5F5111[1,3,4,5,6,7,8,J], R5F5113[5,6,7,8], R5F5130[3,5,6,7,8], R5F513T[3,5], R5F5140[3,5,6], R5F5230[5,6], R5F5231[5,6,7,8], R5F523E[5,6], R5F523T[3,5], R5F523W[7,8], R5F524T[8,A,B,C,E], R5F524U[B,C,E], R5F526T[8,9,A,B,F], R5F5651[4,7,9,C,E], R5F565N[4,7,9,C,E,D,N], R5F5660[4,9], R5F566N[D,N], R5F566T[A,E,F,K], R5F5671[9,C,E], R5F571M[F,G,J,L], R5F572M[D,N], R5F572N[D,N], R5F572T[F,K]
Renesas RA	R7FA0E1x[5,7] ^{NEW} , R7FA2A1xB, R7FA2A2xD, R7FA2E1x[5,7,9], R7FA2E2x[3,5,7], R7FA2E3x[5,7], R7FA2L1x[9,B], R7FA4E1x[B,D], R7FA4E2x9, R7FA4L1x[B,D] ^{NEW} , R7FA4M1AB, R7FA4M2x[B,C,D], R7FA4M3x[D,E,F], R7FA4T1x[9,B], R7FA4W1xD, R7FA6E1x[D,F], R7FA6E2x[9,B],R7FA6M1xD, R7FA6M2x[D,F], R7FA6M3x[F,H], R7FA6M4x[D,E,F], R7FA6M5x[F,G,H], R7FA6T1x[B,D], R7FA 6T2x[B,D], R7FA6T3xB, R7FA8D1x[F,H], R7FA8M1x[F,H], R7FA8T1x[F,H], R7FA8E1xF,

	R7FA8E2xF
Toshiba TX and TXZ3	TMPM03[6,7]FW, TMPM06[1,6,7,8]FW, TMPM330F[D,W,Y], TMPM332FW, TMPM333F[D,W,Y], TMPM341F[D,Y], TMPM365FY, TMPM366FD, TMPM367FD, TMPM368FD, TMPM369FD, TMPM370FY, TMPM372FW, TMPM373FW, TMPM374FW, TMPM375FS, TMPM376FD, TMPM37AFS,TMPM380F[W,Y], TMPM381FW, TMPM383F[S,W], TMPM384FD,TMPM3U0FS, TMPM3U6F[W,Y], TMPM3V4F[S,W], TMPM3V6FW, TMPM440F[10,E], TMPM461F[10,15], TMPM462F[10,15], TMPM46BF10, TMPM3H0F[M,S], TMPM3H1F[P,S,W,U], TMPM3H2F[S,U,W], TMPM3H3F[S,U,W], TMPM3H4F[S,U,W], TMPM3H5F[S,U,W], TMPM3H6F[S,U,W], TMPM3HLF[D,Y,Z], TMPM3HMF[D,Y,Z], TMPM3HNF[D,Y,Z], TMPM3HPF[D,Y,Z], TMPM3HQF[D,Y,Z]
NXP S32 🏁	S32K11[6,8], S32K14[2,4,6,8], S32K31[0,1,2,4], S32K34[1,2,4,8], S32M24[1,2,3,4], S32M27[4,6]
NXP LPC1xxx	LPC13x[1,2,3], LPC131[5,6,7], LPC134[5,6,7], LPC15x[7,8,9], LPC175[1,2,4,6,8,9], LPC176[3,4,5,6,7,8,9], LPC177[3,4,6,7,8], LPC178[5,6,7,8], LPC181[2,3,5,7], LPC182[2,3,5,7], LPC183[3,7], LPC185[3,7], LPC18S[3,5]7

Please contact <u>easyDSP@gmail.com</u> for new MCU support.

3. starting easyDSP

For those who use easyDSP first time, please refer to below steps. The details could be different by target MCU.

Step	Process	Remark
1	hardware connection between easyDSP and MCU	Hardware connection between MCU and easyDSP. Please refer the help file 'How to user MCU'.
2	correction of user program	First, inlude the source file and header file for easyDSP communication into your project. You can find these files in the 'source' folder in the folder easyDSP is installed. Second, modify the #define variable in the header file according to your system. For some MCU, you don't need this process. Third, include this header file in the main.c and call the function for easyDSP communication. Please refer to the help file 'How to user MCU'.
3	creation of easyDSP project	Creates easyDSP project. Refer to the help file 'Menus>Project'.
4	MCU booting	Booting of MCU via either 'RAM booting' or 'Flash ROM' menus. Refer to the help file 'Menus>MCU'.
5	easyDSP monitoring	Monitoring of variables of MCU program by using versatile easyDSP windows.

6	modification of user program	For debugging of your program, change your program under IDE environment.
7	MCU booting	Like step 4, boot MCU with new user program.
8	easyDSP monitoring	Like step 5.

4. Revision History

Version	MCU	Revision items
	Common	 Sector selection in the flash programming dialog can be blocked by 'Freeze' check box for some MCU series Bug Fix : when using auto scale in Y-axis of plot window, same Y-axis range is applied to all plot windows
	TI C28x	 for Gen3 MCU, flash programming is supported even when the address alignment of its section is wrong new style of flash dialog box for F2837xS, F2807x, F28002x, F28003xand F28004x Bug Fix : for Gen3 MCU, flash programming could fail if the section size exceeds 0xFFFF wrong identification of used sector in flash dialog of F28Px and F28001x
ver 11.4 Apr/2025	ST STM32	 new function to erase all the flash (Erase chip button in the flash dialog) new support for STM32U375x[E,G] and STM32U385xG (source file easyStm32LL_v11.4.c is required) new support for STM32WBA50xG, STM32C051x[6,8], STM32C091x[B,C] and STM32C092x[B,C] The error that periodic writing 32 bytes 0xFF data to flash for the specific bootloader version of STM32H72x and STM32H73x is corrected support for both single and dual bank for STM32L471xE, STM32L475x[C,E], STM32L476x[C,E] and STM32L496xE Bug Fix : flash programming error for swapped dual bank of STM32U5, STM32L5, STM32H7 and STM32G0 compile error in the file "easyStm32LL v11.3.c" when using STM32H7 dual core and STM32WL3x incorrect page address ofdual bank mode of STM32F76[5,7,9]xI and STM32F77[7,8,9]xI flash programming for STM32WL33x is not working
	NXP S32K	- support new MCU S32K series : S32K11[6,8], S32K14[2,4,6,8], S32K31[0,1,2,4], S32K34[1,2,4,8], S32M24[1,2,3,4] and S32M27[4,6]
	RA	required)
ver 11.3	Common	- DWARF5 support improvement
Jan/2025	ST STM32	 support for STM32WB05xZ, STM32WB06xC, STM32WB07xC and STM32WB09xE (source file easyStm32LL_v11.3.c is required)

		 support for STM32H523x[C,E], STM32H533xE and STM32C071x[8,B] Bug Fix : STM32WB09xE is not supported
	Renesas RA	- support for R7FA8E1xF and R7FA8E2xF
	Renesas RX	Bug Fix :monitoring failure of 8 bytes variable and pointer variable (bug from ver 11.1)
	TI C28x	 TMS320F28P55xS series support (source file easy28x_bitfield_v11.2.c is required) Bug Fix : flash programming error for bank 4 of TMS320F28P65xDH
ver 11.2 May/2024	ST STM32	 STM32U0 series support (source file easyStm32LL_v11.2.c is required) support for STM32U5A5xI,STM32U5F7xJ, STM32U5F9xJ, STM32U5G7xJ and STM32U5G9xJ support for STM32WB09xE, STM32WBA54x[E,G], STM32WBA55x[E,G] and STM32WL33x[8,B,C]
	Renesas RA	 - RA2A2, RA8T1 MCU series support Bug Fix : no new project created for RA8D1 MCU
	Renesas RX	- RX23E-B series support
	Common	 Writing to variable is not allowed if the variable is located in the flash Bug Fix : malfunction of plot window when 'Total plot period' is more than 71582 minutes
	ST STM32	- STM32U535, STM32U545, STM32U595, STM32U599, STM32U5A5 and STM32U5A9 series are supported
	Renesas RA	- RA2E3, RA4E2, RA4T1, RA6E2, RA6T3, RA8D1 , RA8M1 MCU series support (together with easyRA_v11.1.c and easyRA_v11.1.h)
ver 11.1		Bug Fix : flash programming not available for RA6M5 with flash area 1.5MB or higher
Jan/2024	Renesas RX	- RX26T support
	NXP LPC1xxx	 support flash programming of LPC1500 series supportLPC1300, LPC1700 and LPC1800 (with onchip flash) series Bug Fix : wrong address recognition of 'array of union' variable
	TI AM2x	 - changes related to RAM booting and flash programming (app image file changeable, SBL baudrate changeable, no SBL image file provided by easyDSP) Bug Fix : no "MulticoreImageGen.exe" file exits in the easyDSP/Util folder. Bug from v10.8 to v11.
ver 11 Sep/2023	TI C28x	 Support for TMS320F28P65x (source file v11 is required) F2837xD, F2838xD : change of sharable memory management for CPU2 ram booting (source file v11 is required) Bug Fix : In case TMS320F2838xD CPU1 uses DriverLib libray : CM fails to flash boot if CPU2 is used (source file 'easy28x_driverlib_v11.c' is required)

	TI MSPM0	- Support for MSPM0 series				
ver 10.9 Jun/2023	ST STM32	 S upport for STM32H5 and STM32WBA series with new source file ¹easyStm32LL v10.9.c' Bug Fix : verifying flash failed for STM32H7, STM32L0, STM32L1, STM32L5 and STM32U5 in some case due to wrong flash programming supports NXP 1500 series (no support flash programming) 				
	LPC1500					
	Common	 Multi dimensional array is supported upto 10 dimension. In the previous version, only up to 4 dimension. Array window : when copying the selected cells to clipboard, easyDSP f fills the empty cells if any. 				
ver 10.8 Apr/2023	TI C28x	 Support for TMS320F280015x with new source files (easy28x_bitfield_v10.8.c or easy28x_driverlib_v10.8.c) Bug Fix : for TMS320F280013x, flash programming doesn't work out in some case 				
	ST STM32	- supports STM32C0 MCU series with new source file 'easyStm32LL v10.8.c'				
	Renesas RX	- supports RX MCU series				
ver 10.7	Common	 Watch window : variable row can be moved up and down Watch/Memory/Tree/Array windows : optionally highlight the changed cell with yellow background color 				
Jan/2023	TI AM2x	- Support for AM263x				
	TI TM4C	- Support for TM4C123x and TM4C129x				
ver 10.6 Nov/2022	TI C28x	 Support for TMS320F280013x with new source files (easy28x_bitfield_v10.6.c or easy28x_driverlib_v10.6.c) No need to run easyDSP as administrator Bug Fix : Symbol information is not extracted in multi core MCU from CPU2 (bug for version 10.5.1 only) 				
ver 10.5.1 Nov/2022	TI C28x	Bug Fix : Flash operation is not working with error message "The variables in flash API wrapper are not fully recognized!" (bug for version 10.3 and higher)				
ver 10.5 Nov/2022	Common	 No more support for old style memory window Memory Window : In case &var format is used as address input, if it is changed with code modification, the address of the window is automatically changed after MCU booting. Bug Fix : Command Window : incomplete auto variable seeking for struc/union/bitfield variables 				
	ST STM32	 Source file is updated to easyStm32LL_v10.5.c. With this, 1. STM32G0x : In the RAM booting and Flash Programmer dialog, entering bootloader is improved 2. STM32H7 dual core (STM32H745x, STM32H747x, STM32H755x and STM32H757x) : Data cache usable 				

		3. If FIFO is available to USART, you can use it to speed up easyDSP communication
	Toshiba TXZ3	- supports Toshiba TXZ3 MCU series
	Common	Bug Fix : - When using DWARF4 or DWARF5 debugging information format with ARM MCU, the address and bit location of bitfield variable is not correct in certain cases.
ver 10.4 May/2022	TI C28x	Bug Fix : - For 2838x, the flash operation is not working unless all the CPU1, CPU2 and CM are used in the project. This is the bug of v10.3 and v10.3.1 only.
	Toshiba TX	- supports Toshiba TX MCU series
ver 10.3.1 Apr/2022	TI C28x	Bug Fix : - For 2838x, the easyDSP project is not created/open unless all the CPU1, CPU2 and CM are used in the project. This is the bug of v10.3 only.
	Common	 Watch window : address column includes bit information in case ofbitfield variable (for ex, 0x1234@bit1-2) Chart window : 1 dimmentional array variable only. count input by user is blocked. It's fixed to array count. Tree window : mouse right click toggles the display mode (decimal => hex-decimal => binary => decimal) Memory window : versatile address input format and comment are enabled Faster symbol information processing Driver file updated to CDM212364_Setup.exe Display mode (hex or dec or bin) for bitfield variable is changeable Bug Fix : Member of anonymous structure/union variable is not properly displayed Anonymous bitfield member is not properly displayed Bitfield member with its size more than 4 bytes is not properly displayed
ver 10.3 Apr/2022	TI C28x	 When using multiple easyDSP projects for multi core MCU such as 2837xD and 2838xS/D, If the output file of CPU2 or CM is reloaded as requested by CPU1, below message box is displayed. easyDSP(2) The output file is now reloaded as requested by other easyDSP project! 32bit Windows is not supported for COFF debugging model Register window : no more support More stable operation of 'Flash API speed [bps]' function in flash dialog (introduced from v10.1) Bug Fix : For 2837xD and 2838xS/D, the error message "Cap't open * bin file!" could

		show up when the output file of CPU2 or CM is updated after entering to flash dialog. - For 2837xD and 2838xS/D, the old out file of CPU2 and CM could be used for RAM booting or flash writing if there is no easyDSP project is open for CPU2 and CM. - For 2837xD with coff debugging model, CPU2 program is not updated in the flash dialog
	ST STM32	- easyDSP uses the hex file IDE created when ram booing and flash programming. please make IDE create hex file in every compiling time. Note that the other option available in the previous easyDSP which easyDSP itself makes hex file is not available now !
		Bug Fix :Flash is programmed with the latest user program regardless of your choice if you use the hex file IDE created
	Infineon PSoC4	Bug Fix : Flash is programmed with the latest user program regardless of your choice
	Infineon XMC4	Bug Fix : Flash is programmed with the latest user program regardless of your choice
	Renesas RA	First release for Renesas RA MCU series
	Infineon XMC1	First release for Infineon XMC1 MCU series (only for monitoring. flash programming not supported)
	TI C28x	Bug Fix : F2837xS : flash dialog box not open (bug of v10.1)
ver 10.2 Jan/2022	Infineon PSoC4	First release for Infineon PSoC4 MCU series (RAM booting not supported)
	Infineon XMC4	First release for Infineon XMC4 MCU series (RAM booting not supported)
	Common	 New style memory window (<u>check futher</u>) Bug Fix : Character value (ex, 'A') can be assigned to non character type variable In array window, character value (ex, 'A') can't be assigned to character type variable floating value can be assgined to pointer variable to float or double or long double
ver 10.1 Nov/2021	TI C28x	 F28003x : newly supported (must use the latest easyDSP source file version 10.1, CCSv11 and compiler version is 21.6.0.LTS) F2802x, F2802x0, F2803x, F2805x, F2806x, F2807x, F2837xS, F2837xD, F28004x, F28002x, F2838xD and F2838xS : flash operation speed up (max. twice) Flash API speed [bps] 115200 Please choose bps to speed up. Note some bps could be not working. F2807x/F2837xS/F2837xD : supporting internal clock source

		 F2838xS/D CM : 'Enables fast verifying' checkbox in RAM booting dialog is now disabled. Multi core F2837xD and F2838xS/D MCU : When RAM booting or flash programing in easyDSP project for CPU1, the communication is paused in the easyDSP project for CPU2 and CM if the projects are open in the same PC. new easyDSP DriverLib source file (easy28x_driverlib_v10.1.c) : supports F28003x, new pin mux naming of C2000Ware_4_00_00_00 and 32bit address support for Gen3 MCU new easyDSP DriverLib source file (easy28x_cm_driverlib_v10.1.c) : enabled access to EtherCAT RAM area and ECC, address alignement and range check to prevent Hard Fault new easyDSP BitField source file (easy28x_bitfield_v10.1.c) : supports F28003x and 32bit address support for Gen3 MCU Bug Fix : struct/union variable recognition error (bug in v10) system error happens when accessing TI OTP memory area in Memory window F2838xS/D CM : failed in verifying RAM booting in some cases
	ST STM32	 No more support for HAL based easyDSP source file (due to more resource burden than LL based one) LL based easyDSP source file improvement (address alignment check and others) : please use easyStm32LL_v10.1.c STM32WB10xC and WB15xC : new support STM32U5 series : new support Bug Fix : used page of flash is not identified for some MCUs which has 128bytes page size
	Common	Bugs Fixed : Invalid struct or union variable is registered in tree window
ver 10 May/2021	TI C28x	 Improved auto bauding process for F2837xS, F2837xD and F2807x supports class type for C++ Improvements in flash dialog (except C2834x) check if all used flash sectors are selected to be erased before "Erase>Program" button is clicked button for all flash operation (erase to reset) update output file when operations to flash is requested (such as program, verify, select used or select not used), not when flash dialog is open. Bugs Fixed : F2837xD and F2838xS/D: Even though updated out file is declined by user in the flashROM dialog, updated out file is programmed for CPU2 and CM F2837xD and F2838xS/D: If *.out file is updated after entering to flashROM dialog or RAM booting dialog, updated out file is not programmed if easyDSP project for CPU2 or CM is not activated.
	ST STM32	 first release for ST STM32 series (dedicated easyDSP pod required) supporting F0, F1, F2, F3, F4, F7, G0, G4, H7, L0, L1, L4, L5, WB and WL series
ver 9.5 Dec/2020	TI C28x	 No more legacy bitfield source file from easyDSP installation package Timing of /BOOT pin of easyDSP pod is changed For more stable CPU2 RAM booting of F2837xD/F2838xD, easyDSP source file

		<pre>(easy28x_BitField_v9.5.c/ easy28x_DriverLib_v9.5.c) is upgraded. Please check the help file. Note - For RAM booting of F2837xD/F2838xD CPU2 : please use "easy28x_driverlib_v9.5.c" and "easy28x_bitfield_v9.5.c" source files Bug Fix : - wrong symbol display at 0x0 address in Memory window (v9.3 and v9.4 only) - 2807x, 2837xS, 2837xD CPU1 : incorrect reserved RAM region check for boot-rom (v9.4 only)</pre>
ver 9.4 Oct/2020	TI C28x	 Chart window improvement : Speed up for chart window update (helpful for big size array) by enabling 'Enable fast reading' option. More window update frequency. Paused when communication is failed a lot. Speed up for verifying RAM booting. Pls enable 'Enable fast verifying' option. Better autobauding of flashAPI wrapper in the flashROM dialog of 28002x, 2838x. speed up by skipping verifying of flashAPI wrapper booting in flashROM dialog (note : for 28002x, 2837x and 2838x, this function was applied from the previous version. It is applied now to all MCUs) one time reading of 4 and 8 bytes variable in the bitfield based source files (easy28_bitfield_v9.4.c and easy28_gen2_bitfield_v9.4.c) Change in the title of menu and its shortcut (Serial Booting, ALT+S -> Ram Booting, ALT+R) Note 2838x CM : please use "easy2838x_cm_driverlib_v9.4.c" Bug Fix : Chart window : in some cases, it is not updated properly after out file update Tree window : not valid variable with * operator in variable list (bug of v9.3 only) 2838x : When using 2838x CPU1 and CM, updated CM program is not reflected automatically to CM project after CM program is booted in CPU1 project 2807x, 2837xS, 2837xD CPU1, 2837xD CPU2, 2838x CPU2 : incorrect reserved RAM region check for boot-rom 2838x CM : failed address is not correct when verifying is failed 2838x CM : flash rom writing error when section start address is 64bit aligned
ver 9.3 Jun/2020	TI C28x	 checking before RAM booting if user code overlaps with memory region for boot rom and easyDSP New bitfield source now available for TMS320F280x, F281x and F28044 bitfield source now available for TMS320F2838xS/D for CPU1 and CPU2 value at address operator (*) is supported for pointer variable Bug Fix : Error in flashrom operation due to skipping booting with flashAPI wrapper time interval not working in watch window F2838x CM section alignment check error in flashrom dialog
ver 9.2	TI C28x	 TMS320F2838x is supported with DriverLib only TMS320F28002x is supported

Apr/2020		 set Rx input pin to pullup type to increase noise immunity New bitfield source now available for TMS320F2802x, F2802x0, F2803x, F2805x and F2806x Bug Fix : In some cases, bin file is not created In some cases, dwarf version 4 is not properly supported Windows are not updated after 'Reload *.out' menu execution treat pointer to struct/uniton variable as struct/unition variable in Tree window 28004x flash rom : 'select all' button not working in flash dialog window 28004x flash rom : not working if clock source is not external 20MHz
ver 9.1 Mar/2020	TI C28x	 DriverLib based easyDSP source files (28004x, 2807x, 2837xS and 2837xD) and example main.c new bitfield based easyDSP source files (28004x, 2807x, 2837xS, 2837xD, 2823x, 2833x and 2834x) and example main.c or main_gen2.c output file reloading menu supports ELF-based Embeded Application Binary Interface (EABI) improved flashAPI wrapper booting in flash rom dialog of 2837xD, 2837xS, 2807x and 28004x Bug Fix : pointer to struct variable is registered in Tree window v9.03 only : error in flashAPI wrapper booting in flash rom dialog of 2837xS, 2807x and 28004x
ver 9.03 Jan/2020	TI C28x	Bug Fix (Bugs only for ver 9.x) : - auto bauding failure in case of 2837xS, 2837xD, 2807x and 28004x - project is not open if project folder and folder of *.out file is different
ver 9.02 Dec/2019	TI C28x	- When creating new project, user need to set debugging model (either coff or dwarf) of compiler in its project setting. When opening existing project which was created before easyDSP verion 9.02, coff is selected by default.
ver 9.01 Dec/2019	TI C28x	Bug Fix : - For some cases, easyDSP can't tell compiler option correctly.
ver 9 Dec/2019	TI C28x	 supports the latest TI compiler version greater than ver.15 supports "symdebug:dwarf" compiler option No more support for "symdebug:coff" Note that coming update could be not available for "symdebug:coff" when using "symdebug:dwarf" compiler option, display variable type with its typedef name (ex, Uint32)
ver 1 to ver 9	TI C28x	- contact <u>easydsp@gmail.com</u>
ver. 1.0 Aug/1999	TI 3x	- First release

5. Limitation

Please kindly keep in mind some limitation when using easyDSP as belows.

Common

- 1. Only little endian is supproted.
- 2. easyDSP uses the interrupt service routine for its communication to MCU.

Therefore if the allocated resource time for the interrupt service routine for easyDSP communication is limited due to the lack of resource, easyDSP could be not proplerly working.

- 3. Value at address operator (*) is supported to pointer variable to basic type only, and for C28x only. That
- is, not supported to for example, pointer to pointer, pointer to array and so on.
- 4. Arrow operator (->) is not supported.
- 5. Writing to 'bit field' type variable is not allowed.
- 6. Multi dimensional array is supported upto 10 dimension.

Limitation

Pleae check the limitation of easyDSP by MCU. o = supported, x = not supported. Flash programming is not available in case the protection or security function is applied. For details, please check the relevant menu for each MCU.

Vendor	MCU	Monitoring	RAM booting	Flash program	Other limitations
	C28x	0	0	0	1. no support for OTP
тт	AM263x	0	o (1)	o (1)	1. limited by SBL
	TM4C	0	x	0	1. no support to EEPROM
	MSPM0	0	x	o (1)	1. only MAIN flash
ST	STM32	Ο	o (4)	o (1,2)	 No support to write to data memory, OTP memory and option bytes. No support to Trust Zone and Secure MPU Limitation from the bugs and limitations of MCU built-in bootloader. RAM booting is not supported for dual core MCU.
Infienon	PSoC4	0	x	o (2)	 No support for PSOC4000 MCU since UART is not available. Flash programming feasible with single- application bootloader configuration only.
	XMC1	0	x	x	
	XMC4	0	x	0	
Renesas	RA	O	x	o (1), x (2)	 For MCU with DLM, DLM state transition is not supported. Flash programming is not supported for RA0 series.
	RX	0	x	o (1,2)	1. Protected area by area protection or trusted

					memory is not programable. 2. For RX64M, RX660, RX66T, RX71M and RX72T series, programming of option setting memory is not supported.
Toshiba	ТΧ	0	х	0	
	TXZ3	0	х	0	
NXP	S32K S32M	0	х	0	no support to EEPROM
	LPC1x00	0	х	0	no support to EEPROM

6. Pod configuration

Pin Description

The signal pins of easyDSP pod are shown below . Its pin pitch is 2.54mm. For easyDSP connector in your board, please use either BHS-01-10P or XG4C-1031 connector. Note the arrow mark on the connector.



Pod type 1 and 2 : Pod for TI C28x MCU

#	name	Description
1	RX	Output pin connected to RX of MCU
2	GND	Ground pin. Connected to #10 pin internally to easyDSP pod.
3	ТХ	Input pin connected to TX of MCU
4	VDD	Voltage bias connected to VDDIO of MCU (ex : 3.3V)
5	/воот	Output pin with pseudo open collector. It becomes Low when entering bootrom by resetting MCU. Otherwise, no signal output from this pin.
6	reserved	Do not connect

7	reserved	Do not connect
8	reserved	Do not connect
9	/RESET	Output pin with pseudo open collector. It b ecomes Low when resetting MCU. Otherwise, no signal output from this pin.
10	GND	Ground pin. Connected to #2 pin internally to easyDSP pod.

Pod type 3 : Pod for Arm Cortex series and other cores (RX)

#	name	Description				
1	RX	Output pin connected to RX of MCU				
2	GND	Ground pin. Connected to #10 pin internally to easyDSP pod.				
3	ТХ	Input pin connected to TX of MCU				
4	VDD	Voltage bias connected to VDDIO of MCU (ex : 3.3V or 1.8V)				
5	/BOOT	Output pin with pseudo open collector. It becomes Low when entering bootrom by resetting MCU. Otherwise, no signal output from this pin.				
6	reserved	Do not connect				
7	воот	Output pin with pseudo open emitter. It becomes high when entering bootloader by resetting MCU. Otherwise, no signal output from this pin.				
8	reserved	Do not connect				
9	/RESET	Output pin with pseudo open collector. It b ecomes Low when resetting MCU. Otherwise, no signal output from this pin.				
10	GND	Ground pin. Connected to #2 pin internally to easyDSP pod.				

/BOOT, BOOT pin

These pins determines how MCU will boot after reset, either boot with flash to execute user program or boot with bootmode to conduct RAM booting or flash programming. These pins are not used at all (= no signal output) when MCU boot with flash. They are active only when MCU boot with boot mode as below :

MCU	Used boot pin	boot pin operation
-----	---------------------	--------------------

C28x XMC4 TX TXZ3 LPC1x00 S32	/BOOT	/BOOT pin becomes low when MCU reset. Around 1sec after MCU reset is released, /BOOT pin becomes open and no signal output.
STM32	BOOT	BOOT pin becomes high when MCU reset and keeps high during boot mode period. BOOT pin becomes open and no signal output when exiting boot mode (= exiting from Ram booting or flash dialog).
AM2x TM4C MSPM0	BOOT	BOOT pin becomes high when MCU reset. Around 1sec after MCU reset is released, BOOT pin becomes open and no signal output.
RA RX	/воот	/BOOT pin becomes low when MCU reset. RA : /BOOT pin becomes open and no signal output when entering "Command acceptance phase" during boot mode.
XMC1 PSOC4	not used	not used

You can use MCU pin that connects to /BOOT or BOOT pin in your application progam if you follow below guidline.

Please check the voltage level of the MCU pin at the beginning of your application program (input IO mode as reset default). Once the voltage level of the pin becomes high or low depending on used boot pin, you can set the MCU pin accordingly and start to use.

LEDs

There are two LEDs to indicate the status as below. Both LEDs should be ON during easyDSP operation (not blinking).

'DSP' or 'MCU' LED is on : MCU controller board is now power supplied (= #4 pin is live with 3.3V) 'USB' LED is on : easyDSP pod is well connected with easyDSP PC program. It's ON when easyDSP project is open, OFF when easyDSP project is closed.

Note) for optic cable easyDSP, DSP LED of PC side pod and USB LED for MCU side pod are not working. no special meaning to the color of LED.

Connection to and Disconnection from PC and MCU

Don't make any physical connection or disconnection of easyDSP pod to/from PC and MCU during MCU operation. It makes unintentional reset to MCU.

In case you can not avoid connection/disconnection during MCU operation, connect PC first then MCU, disconnect MCU first and then PC. This will minimize the chance of unintentional reset to MCU.

Connection to PC

If possible, please connect easyDSP pod directly to PC (not via USB extension port). And please use the new USB cable to secure its connection quality.

Specification

Items	Pod type 1 :	Pod Type 2 :	Pod Type 3 :	
	TI C28x MCU	TI C28x MCU	Arm Cortex-M	

	standard pod	optic cable pod	and RX standard pod
Supply voltage range to VDD	min 3, typ 3.3, max 5 [V]	same to left	min 1.65, max 5 [V]
Recommended supply voltage to VDD	3.3V	3.3V	MCU VDDIO (ex, 3.3V or 1.8V)
Input voltage range	-0.5 VDD+0.5 [V]	same to left	same to left
Supply current to VDD	max 3mA	max 50mA	max 10mA
min. isolation voltage	2.0kVrms@1min	-	2.0kVrms@1min
Operating free-air temperature	5 55 [°C]	same to left	same to left
Starage temperature range	-20 65 [°C]	same to left	same to left
Relative humidity (non- condensing)	max 90% rH	same to left	same to left
Size (without cables)	82 x 56 x 21 mm^3	same to left but two pods	81 x 42.5 x 21 mm^3
Weight (without cables)	140 g	330 g	62g
USB interface	USB 2.0 Hi-Speed	same to left	same to left

7. How to use MCU

7.1 C28x

7.1.1 C28x programming

7.1.1.1 common

BitField and DriverLib

There are two folders in the 'source/C28x' directory of installed easyDSP.



DriverLib

'BitField' folder : Bitfield based easyDSP source files.

'DriverLib' folder : DriverLib(C28x Peripheral Driver Library) based easyDSP source files

please refer to the <u>TI link</u> for further understanding of BitField/DriverLib. You can use only one out of two methods.

If you use the bitfield based functions in your project, then please use also bitfield based easyDSP source files, or vice versa.

Also check below which method is supported for which MCU.

	Bitfield	DriverLib
F28001x F28002x F28003x F28004x F2807x F2837x F2837x F2838x F28P55x F28P65x	Ο	Ο
C2834x F2823x F2833x F281x F280x F28044 F2802x0 F2802x F2802x F2803x F2805x F2805x F2806x	Ο	

Debugging model option

easyDSP supports below two debugging model options, --symdebug:dwarf and --symdebug:coff. Note that the latest TI C28x compiler (version 16 or above) doesn't support--symdebug:coff option. Accordingly further support for this option will be very limited. Recommend to use -- symdebug:dwarf option from now.

type filter text	Advanced Debug Options		← → ⇒ → →
General V Build V C2000 Compiler Processor Options Octimation	Configuration: Debug [Active]	~	Manage Configurations
Optimization Include Options Performance Advisor V Advanced Options Advanced Debug Options Language Options	Debugging model Optimize fully in the presence of debug (DEPRECATED) (optimize_with_debug, -mn) Keep unreferenced type info (default for elf w/ debug) (symdebug:keep.all_types)	Full symbolic debug (symdebug:dwaff, -g)	~ ~
Properties for Combo Control			- 0 X
c) is a constant of the consta	Advanced Debug Options		← + ⇒ + +
General V Euido C2000 Compiler Processor Options Optimization	Advanced Debug Options Configuration: Debug [Active]	~	← ★ ⇒ ▼ ▼ Manage Configurations

Endianness option

Properties for 2838xD_cm_DriverLib				_	
type filter text	General			¢	• => • •
 > Resource General > Build > ARM Compiler Processor Options 	Configuration: CM_RAM [Ac	ctive]	~	Manage Con	figurations
Optimization Include Options	🔞 Project 📑 Products				
ULP Advisor Predefined Symbols	Device Family: ARM				\sim
> Advanced Options > ARM Linker	Variant: <select or="" td="" type<=""><td>e filter text> ~</td><td>TMS320F28388D</td><td></td><td>~</td></select>	e filter text> ~	TMS320F28388D		~
ARM Hex Utility [Disabled] Debug	Connection: Texas Instrume	Connection: Texas Instruments XDS100v2 USB Debug Verify			
Debug Project Natures	Manage the	project's target-configuration	automatically		
	Tool-chain				
	Compiler version: TI	v18.12.3.LTS		~ N	lore
	Output type: Exe	ecutable		\sim	
	Output format: ea	bi (ELF)		\sim	
	Device endianness: litt	tle		\sim	
	Linker command file: 28	38x_RAM_Ink_cm_easyDSP.cm	d	~ Bro	owse
	Runtime support library: <a< td=""><td>automatic></td><td></td><td>∼ Bro</td><td>owse</td></a<>	automatic>		∼ Bro	owse
Show advanced settings			Apply an	d Close	Cancel

Only little endian is supported by easyDSP. Please set endianness like below.

Section alignment when using Gen3 MCU

easyDSP uses TI's flash API to access onchip flashrom. TI flash API of Gen.3 MCU (for example. F2807x, F28001x, F28002x, F28003x, F28004x, F2837x, F2838x and F28Px) requires section alignment on the address (min. 4 words boundary or recommended 8 words boundary) depending on MCU. That is, the start address of the section should be either 0x*0, 0x*4, 0x*8 or 0x*C for C28x core and either 0x*0 or 0x*8 for Arm Cortex-M4 (ex, F2838x CM). As shown below linker command file example from TI, it is already applied as recommended value for default sections like .text **but you need to do it yourself for your own section**.

```
ker command file excerpt of TMS320F28388 CPU1/CPU2>
SECTIONS
{
   codestart
                   : > BEGIN, ALIGN(8)
                    : >> FLASH1 | FLASH2 | FLASH3 | FLASH4, ALIGN(8)
   .text
                    : > FLASH4, ALIGN(8)
   .cinit
                    : > FLASH1, ALIGN(8)
  .switch
                    : > RESET, TYPE = DSECT /* not used, */
   .reset
   .stack
                    : > RAMM1
#if defined(__TI_EABI__)
   .init_array : > FLASH1, ALIGN(8)
   .bss
                 : > RAMLS5
   .bss:output
                 : > RAMLS3
   .bss:cio
                 : > RAMLS5
   .data
                  : > RAMLS5
  .sysmem
                 : > RAMLS5
  /* Initalized sections go in Flash */
                 : > FLASH5, ALIGN(8)
   .const
#else
  .pinit
                 : > FLASH1, ALIGN(8)
                 : > RAMLS5
   .ebss
                 : > RAMLS5
  .esysmem
  .cio
                  : > RAMLS5
  /* Initalized sections go in Flash */
   .econst : >> FLASH4 | FLASH5, ALIGN(8)
#endif
```

Linker option

It is recommended the entry point is set to the 'code_start' label (in TI's DSP28x_CodeStartBranch.asm with watch-dog disabled). This is done by linker option -e in the project build options, that is,-ecode_start. It prevents unintentional watch dog reset during c_int00 operation which could happen in long size program where it takes long time to initialize many variables.

7.1.1.2 multi cores

Multi core MCU

Target MCUs are F28P65xD, F2827xD, F2838xS and F2838xD.

Predefined symbols

Predefined symbols such as CPU1, CPU2, CM and _FLASH are referred in easyDSP source files when multi-core MUC is used.

If target core is CPU2, CPU2 should be predefined. If target core is CM, CM should be predefined. These symbols are usually predefined by CCS. But please check.

easyDSP help

Properties for 2838xD_cpu1_DriverLib		$ \Box$ >	<
type filter text	Predefined Symbols	← → ⇒ →	000
 Resource General Build C2000 Compiler Processor Options 	Configuration: CPU1_FLASH [Active]	Manage Configurations	
Optimization Include Options Performance Advisor Predefined Symbols	Pre-define NAME (define, -D) _FLASH DEBUG CPUT	🛃 🜒 🗑 🥳 💡	₽T
> Advanced Options > C2000 Linker C2000 Hex Utility [Disabled] Debug Project Natures			

Using debugger

Don't use multi-core booting related functions (easyDSP_Boot_Sync) easyDSP is providing in case you use debugger. Debugger will load the memory of each core. please refer to #define USE_DEBUGGER of main.c in easyDSP source file folder.

easyDSP uses MCU resource for multi core Ram booting

Some MCU resource is used by easyDSP to implement CPU2/CM ram booting. Please check below table. You should not use these resource before CPU2/CM booting (calling of easyDSP_Boot_Sync() function) in your code. But you can use them after the booting.

МСИ	F2837xD	F2838xS F2838xD	F28P65xD
Resource used by easyDSP during ram booting CPU2 and CM	IPC_FLAG0 IPC_FLAG5 IPC_FLAG31	IPC_FLAG0 IPC_FLAG5 IPC_FLAG6 IPC_FLAG30 IPC_FLAG31 CPU1 to CPU2 MSGRAM1 CPU1 to CM MSGRAM1	IPC_FLAG0 IPC_FLAG5 CPU1 to CPU2 MSGRAM0

Flash booting location of F2838x and F28P65xD for CPU2 and CM

In the source file of easyDSP, the flash booting location is fixed : For F2838xD CPU2 and CM, it is set to sector 0. For F28P65xD CPU2, it is set to bank 3. In case you like to change its location, please modify below part in easyDSP_Boot_Sync() function in the easyDSP source file.

F2838x BitField : ezDSP_Device_bootCPU2(BOOTMODE_BOOT_TO_FLASH_SECTOR0); ezDSP_Device_bootCM(BOOTMODE_BOOT_TO_FLASH_SECTOR0);

F2838x DriberLib : Device_bootCPU2(BOOTMODE_BOOT_TO_FLASH_SECTOR0); Device_bootCM(BOOTMODE_BOOT_TO_FLASH_SECTOR0);

F28P65xD BitField : ezDSP_Device_bootCPU2(BOOTMODE_BOOT_TO_FLASH_BANK3_SECTOR0); F28P65xD DriverLib : Device_bootCPU2(BOOTMODE_BOOT_TO_FLASH_BANK3_SECTOR0);

Restriction of memory use for RAM booting of F2838x and F28P65xD

RAM booting via SCI port for CPU2 and CM of F2838x and F28P65xD is not supported by TI. easyDSP uses workaround to boot CPU2 and CM via SCI. First, boot CPU1 via SCI with user program then boot CPU2/CM with small agent program (not user program) via 'IPC message copy to RAM' boot mode. Then this agent program downloads user program to CPU2 and CM via SCI. With this, there is some restriction of memory usage to CPU2 and CM for this agent operation. Please check below table and reflect this to command file accordingly.

	Restriction of memory usage in user program when ram booting of F2838x	Restriction of memory usage in user program when ram booting of F28P65xD
CPU1 user program	no restriction	no restriction
CPU2 user program	part of M1 RAM (0x400 - 0x7F7) can't be used as initialized section	part of M1 RAM (0x400 - 0x5FF) can't be used as initialized section
CM user program	part of S0 RAM (0x2000.0800 - 0x2000.0FFF) can't be used as initialized section	

Change in CPU2 RAM booting of F2837xD and F2838xD from easyDSP source file version 11 MM

Before easyDSP source file version 11, for CPU2 ram booting of F2837xD and 2838xD, all the GSRAM (Global Shared RAM) are allocated to CPU2 during CPU2 ram booting and then allocated to CPU1 after ram booting in the easyDSP_SCIBootCPU2() function of easyDSP source file.

So, ram booting related code of CPU1 (.text section of easyDSP_SCIBootCPU2() function) should be located to LSRAM (Local Shared RAM). And if required from CPU2 user program, CPU1 should allocate GSRAM to CPU2 after CPU2 ram booting.

This way requires lots of restriction and caution and not any longer recommended. In the source file version 11, GSRAM is allocated to neither CPU1 nor CPU2 in the easyDSP_SCIBootCPU2() function.

Instead, in the CPU1 program main.c, the required GSRAM is allocated to CPU2 before calling easyDSP_SCIBootCPU2().

With this, no more restriction and caution needed.

Booting sequence and syncronization of F2837xD and F28P65xD

The flash booting is executed in a sequence of CPU1 and then CPU2 without any synchronization between.

The RAM booting is executed in same sequence with synchronization (i.e. the end of easyDSP_Boot_Sync() is synchronized).

Note that necessary memory should be allocated to CPU2 before CPU1 is calling easyDSP_Boot_Sync().

						Syn	c between	cores
Case fo	or RAM I	booting						
CPU1	RAM Booting	entrance to main()	Allocation of necessary RAM to CPU2		call easyDSP_Boot_Sync() (waiting for CPU2 booting)			
CPU2				CPU2 Booting	entrance to main()	call easyDSP_Boot_Sync()	
								time
Case fo	Case for flash booting							
CPU1	flash Booting	entrance to main()	Allocation of necessary RAM and flash to CPU2	call easyDSP (commanding	_Boot_Sync() CPU2 booting	CPU1 running		

	booting	manity	Total and hash to croz	(commanding	croz booding)			
CPU2				CPU2 Booting	entrance to main()	call	easyDSP_Boot_Sync() (no action)	CPU2 running

Booting sequence and syncronization of F2838x

The flash booting is executed in a sequence of CPU1, CPU2 and CM without any synchronization between.

The RAM booting is executed in same sequence with synchronization between (i.e. the end of easyDSP_Boot_Sync() is synchronized).

Note that necessary memory should be allocated to CPU2 and CM before CPU1 is calling easyDSP_Boot_Sync().

Case fe	or RAM I	booting								-,	1
CPU1	RAM Booting	entrance to main()	Allocation of necessary RAM to CPU2		easyDSP_Boot_Sync() (waiting for CPU2 and CM booting)						
CPU2				c	PU2 Booting	entrance to main()		(v	easyDSP_Boot vaiting for CM	:_Sync() booting)	
СМ						CM Booting	er	ntrance to main()	easyD	OSP_Boot_Sync()	time
Case fo	or flash	booting									
CPU1	Flash Booting	entrance to main()	Allocation of necessary RAM and flash to CPU2	y easyDSP_Boot_Sync() 2 (commanding CPU2 and CM booting)							
CPU2				c	PU2 Booting	entrance to main()	eas	yDSP_Boot (no actio	_Sync() n)	_	
СМ					CM Booting	entrance main()	to	easyDSP_B (no a	oot_Sync() ction)		time

F2838x CPU2 and CM clock

When CPU1 boots CPU2 and CM, CPU1 set their clock frequency to 200MHz and 125MHz respectively. If you like to change them, you should modify the related source file by yourself.

When out file has been changed

The output file (*.out) is changed whenever the user program is compiled. When you download the new output file by either RAM booting or flash programming in the easyDSP project connected to CPU1, the easyDSP project connected to another cores should be updated by new output file too. In case easyDSP for multi cores are all connected to the same PC, this process is done automatically, meaning easyDSP project for CPU1 asks easyDSP project for CPU2 to load new output file. In case they are open in different PC, you have to load new output file for another cores manually, by clicking 'MCU > Reload *.out' menu.

7.1.1.3 using BitField

SCI ISR (Interrupt Service Routine)

easyDSP uses an SCI interrupt to communicate with TMS320F28x. Therefore, the user program should include SCI ISR (Interrupt Service Routine) code which easyDSP provides. It depends on TMS320F28x type.

You can find these source files at the folder of easyDSP installation 'source\C28x\BitField'. note) For F2838x CM, DriverLib based source file should be used.

C28x series	SCI ISR files		
F28001x F28002x F28003x F28004x F2807x F2837x F2838xS CPU1 F2838xD CPU1 F2838xD CPU2 F28P55x F28P65x	easy28x_bitfield_v11.2.c easy28x_bitfield_v11.2.h		
C2834x F2823x/2833x F2802x/F2802x0 F2803x F2805x F2806x F280x F280x F281x F281x F28044	easy28x_gen2_bitfield_v <mark>9.4</mark> .c easy28x_gen2_bitfield_v <mark>9.4</mark> .h		

Name and its role of key functions in ISR code is easyDSP_SCI_Init() : Initializes SCI easy_RXINT_ISR() : ISR for RX_INT easy_TXINT_ISR() : ISR for TX_INT easyDSP_SPI_Flashrom_Init() : for external SPI flashrom booting of C2834x easyDSP_Boot_Sync() : multi-core MCU (F2837xD, F2838xS, F2838xD) boot and synchronization

You SHOULD change some #define variables in the header file (not source file) accordingly to your target system.

For example, below selection is targeting for F2807x + CPUCLK 150MHz + LSPCLK = CPUCLK/4 + easyDSP communication @ 115200 bps.

#define F28P65xS	0
#define F28P65xD_CPU1	0
#define F28P65xD_CPU1_CPU2	0
#define F28002x	0
#define F28003x	0
#define F28004x	0
#define F2807x	0
#define F2837xS	0
#define F2837xD_CPU1	0
#define F2837xD_CPU1_CPU2	0
#define F2838xS_CPU1	0
#define F2838xS_CPU1_CM	0
#define F2838xD_CPU1	0

#define F2838xD_CPU1_CPU2	0
#define F2838xD_CPU1_CM	0
#define F2838xD_CPU1_CPU2_CM	1

#define CPU_CLK	15000000L
#define LSP_CLK	(CPU_CLK/4)
#define BAUDRATE	115200L

Please note that in case of MotorWare[™], LSP_CLK should be same to CLK_CLK.

All variables in the ISR have prefix 'ezDSP_'. Please don't change these variables during your easyDSP operation.

Interrupt Nesting

Interrupts are automatically disabled when an interrupt service routine begins. In other words, once easyDSP ISR has been executed, your higher priority ISR can't be executed until easyDSP ISR has been completed.

easyDSP source file provides buit-in interrupt nesting function assuming easyDSP SCI ISR has the lowest priority.

For further information about interrupt nesting, please check http://processors.wiki.ti.com/index.php/Interrupt_Nesting_on_C28x

Run easyDP ISR fast on the flash

To run easyDSP ISR fast and stable when system is running on the flash, please use #pragma in the top-most part of easyDSP source file. Please refer to TI application note for 'ramfuncs' or '.TI.ramfunc' section operation.

in the part of header file easy28x bitfield.h

#if (F2823x || F2833x || C2834x)
#pragma CODE_SECTION(easy_RXINT_ISR, "ramfuncs");
#else
#pragma CODE_SECTION(easy_RXINT_ISR, ".TI.ramfunc");
#endif

NOTE) ".TI.ramfunc" is used instead of "ramfuncs" in case the latest MCU (ex, 2837x, 2807x, 28004x) is used with the latest TI Support Library version (and compiler). Please check the file "F28x_SysCtrl.c" to understand which one is proper.

NOTE) Especially when your program runs on the flash and program/erase the flash at the same time with TI flash API, ISR of easyDSP should run on the ram, not on the flash. Any ISR routines that are executed during flash API function call must completely reside outside of the flash and must not expect to read data from the flash.

Single core programming

easyDSP requires appropriate interrupt settings to communicate with MCU. Below box shows its example. At first, please set up the other interrupts except SCI. Then, call easyDSP_SCI_Init(). In the call to the functions, related registers are set up for SCI communication and interrupts. Also please check main_gen2.c or main_gen3.c example file in the source/C28x/bitfield folder.

```
#include " easy28x_bitfield_v11.2.h" or ....
#include " easy28x_gen2_bitfield_v9.4.h" or ....
main(void) {
```

// below function should be called after other interrupts settings and before while(1)
easyDSP_SCI_Init();

while(1) { }

}

C2834x programming for external SPI flash

Since 2834x doesn't have internal flash, easyDSP supports external flashs with SPI interface. They are AT25DF021(2M bit), AT25DF041(4M bit), AT26DF081(8M bit), AT25DF321(32M bit), M25P20(2M bit), M25P40(4M bit), M25P80(8M bit), M25P16(16M bit), M25P32(32M bit) manufactured by ATMEL or Numonyx. SPI-A port setting is necessary for this. Also please check main gen2.c example file in the source/C28x/bitfield folder.

```
#include "easy28x_gen2_bitfield_v9.4.h "
main(void) {
    // SCI port setting for easyDSP
    easyDSP_SCI_Init();
    //SPI-A port setting for external flash
easyDSP_SPI_Flashrom_Init();
    while(1) {
    }
}
```

F2837xD, F28P65xD, F2838xD multi core programming

The use of header file and easyDSP_SCI_Init() function is same to that of single core MCU. In addtion, easyDSP_Boot_Sync() function is required to boot and synchronize CPU2 and CM. This function should be called in all cores (CPU1, CPU2 and CM) program. Pease check main_gen3.c example file in the source/C28x/bitfield folder.

```
#include "easy28x_bitfield_v11.2.h"
main(void) {
```

```
InitSysCtrl();
```

// if CPU1 program, allocate the necessary sharable memory to CPU2 and CM // before easyDSP_Boot_Sync() is called		
<pre>// call this after sharable memory allocation and before easyDSP_SCI_Init() easyDSP_Boot_Sync();</pre>		
easyDSP_SCI_Init();		
while(1) { }		

7.1.1.4 using DriverLib

ISR (Interrupt Service Routine) for SCI

easyDSP uses an SCI interrupt to communicate with TMS320F28x. Therefore, the user program should include SCI ISR (Interrupt Service Routine) code which easyDSP provides. You can find these source files at the folder of easyDSP installation 'source\C28x\DriverLib'.

C28x series	SCI ISR files
F28001x F28002x F28003x F28004x F2807x F2837x F2838x CPU1 and CPU2 F28P55x F28P65x	easy28x_DriverLib_v11.2.c easy28x_DriverLib_v11.2.h
F2838x CM	easy28x_cm_DriverLib_v10.1.c easy28x_cm_DriverLib_v10.1.h

Name and its role of key functions in ISR code is easyDSP_SCI_Init() : Initializes SCI easyDSP_UART_Init() : Initializes UART of TMS320F2838x CM easy_RXINT_ISR() : ISR for RX_INT easyDSP_Boot_Sync(void) : Multi core MCU (F2837xD, F2838xS and 2838xD) booting and synchronization

You SHOULD change some #define variables in the early part of the source accordingly to your target system. For example, below selection is targeting for F2807x + easyDSP communication @ 115200 bps.

#define F28002x	0
#define F28003x	0
#define F28004x	0
#define F2807x	1
#define F28P65xS	0

 #define F28P65xD_CPU1
 0

 #define F28P65xD_CPU1_CPU2
 0

 #define F2837xS
 0

 #define F2837xD_CPU1
 0

 #define F2837xD_CPU1_CPU2
 0

 #define F2838xD_CPU1
 0

 #define F2838xD_CPU1_CPU2
 0

 #define F2838xD_CPU1_CPU2
 0

 #define F2838xD_CPU1_CPU2
 0

 #define F2838xD_CPU1_CPU2
 0

Please note that DEVICE_LSPCLK_FREQ constant in device.h file should be matching to your system since SCI baudrate setting of easyDSP is based on that. All variables in the ISR have prefix 'ezDSP_ ' . Please don ' t change these variables during your easyDSP operation.

Interrupt Nesting

Interrupts are automatically disabled when an interrupt service routine begins. In other words, once easyDSP ISR has been executed, your higher priority ISR can't be executed until easyDSP ISR has been completed.

easyDSP source file provides buit-in interrupt nesting function assuming easyDSP SCI ISR has the lowest priority.

For further information about interrupt nesting, please check http://processors.wiki.ti.com/index.php/Interrupt Nesting on C28x

Run easyDP ISR fast and stable on the flash

To run easyDSP ISR fast and stable when system is running on the flash, please use #pragma in the easyDSP header file. Please refer to TI application note for '.TI.ramfunc' section operation.

in the header file, easy28x_driverlib.h

#pragma CODE_SECTION(easy_RXINT_ISR, ".TI.ramfunc");

NOTE) Especially when your program runs on the flash and program/erase the flash at the same time with TI flash API, ISR of easyDSP should run on the ram, not on the flash. Any ISR routines that are executed during flash API function call must completely reside outside of the flash and must not expect to read data from the flash.

Single core MCU programming

easyDSP requires appropriate interrupt settings to communicate with MCU. Below box shows its example. At first, please set up the other interrupts except SCI. Then, call easyDSP_SCI_Init(). In the call to the functions, related registers are set up for SCI communication and interrupts. Also please check main.c example file in the source/C28x/driverlib folder.

```
# include " easy28x_DriverLib_v11.2.h"
main(void) {
    Device_init();
    // below function should be called after other interrupts settings
    easyDSP_SCI_Init();
    while(1) {
    }
```

}

Multi core programming for CPU1 and CPU2 : F28P65xD, F2837xD, F2838xS and F2838xD

The use of header file and easyDSP_SCI_Init() function is same to that of single core MCU. In addtion, easyDSP_Boot_Sync() function is required to boot and synchronize CPU2. This function should be called in both CPU1 and CPU2 program. Pease check main.c example file in the source/C28x/DriverLib folder.

```
#include " easy28x_DriverLib_v11.2.h"
main(void) {
    Device_init();
    // called after Device_init() and before easyDSP_SCI_Init()
    easyDSP_Boot_Sync();
    easyDSP_SCI_Init();
    while(1) {
    }
}
```

Multi core programming for F2838x CM

The use of header file and easyDSP_UART_Init () function is similar to that of single core MCUs. In addtion, easyDSP_Boot_Sync() function is required to boot and synchronize CM. Pease check main_cm.c example file in the source/C28x/DriverLib folder.

```
#include " easy28x_cm_DriverLib_v10.1.h"
main(void) {
    CM_init();
    // called after CM_init() and before easyDSP_UART_Init()
    easyDSP_Boot_Sync();
    easyDSP_UART_Init();
    while(1) {
      }
}
```

7.1.1.5 F2837xD and F28P65xD usage

How to connect easyDSP

We need two easyDSP pods and two easyDSP programs and connect them properly to each CPU1 and CPU2 for proper communication. easyDSP program can be executed with multiple instances with its program title like easyDSP, easyDSP(2).

Careful procedure should be taken to connect first easyDSP program (titled easyDSP) to CPU1 and then second easyDSP program (titled easyDSP(2)) to CPU2.

First, you connect single easyDSP pod to PC and then to SCI-A port of CPU1. Run easyDSP program and open the project for CPU1. Then the easyDSP program and its project is connected to CPU1. Then connect another easyDSP pod to PC and then to SCI-B port of CPU2. Run another easyDSP program and open the project for CPU2.

NOTE) RAM booting and flash rom operation is possible for both CPU1 and CPU2 even with single easyDSP pod and single easyDSP program. But in this case, the communication after booting with CPU2 is not supported.

NOTE) Please use the single PC to connect easyDSP for both CPU1 and CPU2. This enables the communication between two easyDSP programs and some mutual activities.



Project creation

easyDSP project for CPU1 requires two output files, one for CPU1 and another for CPU2. If you don't specify the output file for CPU2, then you can not boot CPU2. And the communication with easyDSP is fixed to CPU1.

easyDSP project for CPU2 requires the output file for CPU2 only. It should be same to the out file for CPU2 used in the easyDSP project for CPU1.

<easyDSP program 1>

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	TMS320F2837xD CPU1 Debugging model (only for TI 28x) dwarf	
Part number	TMS320F28377D CPU1	
Output File(s) —		Communication with easyDSP
CPU1	C:\temp\temp\temp1.out	
CPU2	C:\temp\cu2.out	
		NK Canad
		Cancer

<easyDSP program 2>

Project Settings			×
Basic Hardware	Miscellaneous		
MCU			
Vendor	TI		
Series	TMS320F2837xD CPU2 Debugging model (only for TI 28x) dwarf		
Part number	TMS320F28377D CPU2		
Output File(s) —			
CPU2	C:₩temp₩cpu2.out		
		ОК	Cancel

RAM booting and flash programming

RAM booting, flash programming and MCU reset for CPU1 and CPU2 are done by CPU1, accordingly done by easyDSP program connected to CPU1. The only thing that CPU2 does is verifying RAM booting of CPU2. Please check below table for the details.

If easyDSP for CPU1 and CPU2 are connected to the single PC, easyDSP for CPU2 pauses its communication when CPU1 is either RAM booting or flash programming.

operation	easyDSP program 1	easyDSP program 2
CPU1, CPU2 RAM booting	supported	Not supported
Verifying CPU1, CPU2 RAM booting	supported only for CPU1	supported only for CPU2
CPU1, CPU2 flashrom operation	supported	Not supported
CPU reset	supported	Not supported

7.1.1.6 F2838x usage

How to connect easyDSP

We need three easyDSP pods and three easyDSP programs and connect them properly to each CPU1, CPU2 and CM. easyDSP program can be executed with multiple instances with its program title like easyDSP, easyDSP(2) and easyDSP(3).

Careful procedure should be taken to connect first easyDSP program (titled easyDSP) to CPU1 and then second easyDSP program (titled easyDSP(2)) to CPU2 and so on.

First, you connect single easyDSP pod to PC and then to SCI-A port of CPU1. Run easyDSP program and open the project for CPU1. Then the easyDSP program and its project is connected to CPU1. Then connect another easyDSP pod to PC and then to SCI-B port of CPU2. Run another easyDSP program and open the project for CPU2.

Likewise, also for CM.

NOTE) RAM booting and flash rom operation is possible for CPU1, CPU2 and CM even with single easyDSP pod and single easyDSP program connected to CPU1.

But in this case, the communication after booting with CPU2 and CM is not supported.
NOTE) Please use the single PC to connect easyDSP for all CPU1, CPU2 and CM. This enables the communication between easyDSP programs and some mutual activities.



Project creation

easyDSP project for CPU1 requires max. three out files, one for CPU1, the other for CPU2 and finally last one for CM. If you don't use CPU2 or CM, please don't specify the out file of them. The communication with easyDSP is fixed to CPU1.

easyDSP project for CPU2 or CM requires the out file for CPU2 or CM only. It should be same out file to ones used in the easyDSP project for CPU1.

<easyDSP program 1>

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	TMS320F2838xD CPU1	
Part number	TMS320F28388D CPU1	
Output File(s)		Communication with easyDSP
CPU1	C:\temp\cu1.out	
CPU2	C:\temp\cpu2.out	
CPU3 (CM)	C:\temp\cpu3.out	
		OK Cancel

<easyDSP program 2>

Project Settings			×
Basic Hardware	Miscellaneous		
MCU			
Vendor	TI		
Series	TMS320F2838xD CPU2		
Part number	TMS320F28388D CPU2		
Output File(s) -			
CPU2	C:\temp\cpu2.out		
		OK	Cancel

<easyDSP program 3>

Project Settings		\times
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI 🔹	
Series	TMS320F2838xD CM	
Part number	TMS320F28388D CM	
Output File(s) —		
CPU3 (CM)	C:\temp\text{wcpu3.out}	
	OK Cance	4

RAM Booting and flash rom programming

RAM booting and flash programming for CPU1, CPU2 and CM are all done by CPU1, accordingly done by easyDSP program connected to CPU1. The verification of RAM booting can be done by each CPU. Please check below table for the details.

If easyDSP for CPU1, CPU2 and CM are connected to the single PC, easyDSP for CPU2 and CM pause their communication when CPU1 is either RAM booting or flash programming.

operation	easyDSP program 1	easyDSP program 2	easyDSP program 3
CPU1, CPU2, CM RAM booting	Supported	Not supported	Not supported
Verifying CPU1, CPU2, CM RAM booting	Supported only for CPU1	Supported only for CPU2	Supported only for CPU2
CPU1, CPU2, CM flashrom	Supported	Not supported	Not supported

operation			
CPU reset	Supported	Not supported	Not supported

7.1.2 C28x board setting

7.1.2.1 F28P65x

In this page, factory default is assummed. If you change User OTP (BOOTPIN_CONFIG, BOOTDEF), you should modify the configuration accordingly.

MCU check below two pins at the reset to decide the booting mode.

Doot Mode	GPIO72	GPIO84		
boot mode	(Default boot mode select pin 1)	(Default boot mode select pin 0)		
Parallel IO	0	0		
SCI / Wait Boot	0	1		
CAN	1	0		
Flash / USB	1	1		

Since easyDSP uses two kinds boot modes, SCI boot mode (RAM boot) and flash boot mode. Below connection is recommended between easyDSP and MCU.



The easyDSP connected to CPU1 should use SCI-A (GPIO13 and 12 fixed).

In case of dual cores MCU (for example, F28P65xD), 2nd easyDSP is required to connect CPU2 via SCI-B. In the easyDSP source file (easy28x_DriverLib.c or easy28x_bitfield.c), GPIO 86 and 87 is used for SCI-B. If another GPIO port is required for SCI-B, please change the hardware connection and modify the easyDSP source file (in the function of easyDSP_SCI_Init) accordingly by yourself.

For other considerations,

- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec.

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO72 via $2k\Omega$ series resistor

- /RESET pin is connected to reset generation circuit of MCU board (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

- Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.2 F2838x

Defino series TMS320F2838xD check below two pins at the reset to decide the booting mode.

Devi Mede	GPIO72	GPIO84		
Boot Mode	(Default boot mode select pin 1)	(Default boot mode select pin 0)		
Parallel IO	0	0		
SCI / Wait Boot	0	1		
CAN	1	0		
Flash / USB	1	1		

Since easyDSP uses two kinds boot modes, SCI boot mode (RAM boot) and flash boot mode. Below connection is recommended between easyDSP and MCU.

Note 1) GPIO28/29 should be used for SCIA

Note 2) factory default is assummed. Otherwise, the user should modify the configuration accordingly.



You need to use three easyDSP pods to communicate with CPU1, CPU2 and CM all. The easyDSP connected to CPU1 should use SCI-A (GPIO28 and 29 fixed).

The easyDSP connected to CPU2 can use either SCI-B, SCI-C or SCI-D but easyDSP recommends to use SCI-B as default in its source file.

The easyDSP connected to CM should use UART. easyDSP uses GPIO84/85 in its source file. In case you uses another GPIO pins for CPU2 and CM, the hardware connection and easyDSP source file (easyDSP_SCI_Init function in the file of easy28x_DriverLib.c or easy28x_bitfield.c) should be modified accordingly by yourself.

- Factory default setting is assumed (Don't change it)

- Power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec.

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO72 via $2k\Omega$ series resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.3 F2837xS/2807x

Both piccolo series TMS320F2807x and defino series TMS320F2837xS check below three pins at the reset to decide the booting mode.

MODE	GPIO72	GPIO84	/TRST	Boot mode
Mode EMU	X	X	1	Emulation Boot
Mode 0	0	0	0	Parallel I/O
Mode 1	0	1	0	SCI (RAM boot)
Mode 2	1	0	0	Wait Boot Mode
Mode 3	1	1	0	Get Mode (factory default = boot to flash)

easyDSP uses two kinds boot mode. SCI boot mode for RAM booting, GetMode boot mode for flash rom booting.

Below connection is recommended between easyDSP and MCU.



- Factory default setting is assumed

- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- connect SCIRXDA = GPIO85, SCITXDA = GPIO84

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- TX/RX pins are directly connected to MCU pins

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO72 via $2k\Omega$ series resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.4 F2837xD

Defino series TMS320F2837xD check below three pins at the reset to decide the booting mode.

MODE	GPIO72	GPIO84	/TRST	Boot mode
Mode EMU	X	X	1	Emulation Boot
Mode 0	0	0	0	Parallel I/O
Mode 1	0	1	0	SCI (RAM boot)
Mode 2	1	0	0	Wait Boot Mode
Mode 3	1	1	0	Get Mode (factory default = boot to flash)

easyDSP uses two kinds boot mode. SCI boot mode for RAM booting, GetMode boot mode for flash rom booting.

Below connection is recommended between easyDSP and MCU.

Note that GPIO84/85 should be used for SCIA. Please check 'How to use different port ?' session in case external memory interface is necessary.



You need to use two easyDSP pods to communicate with both CPU1 and CPU2.

one easyDSP connected to CPU1 should use SCI-A (GPIO84/85 fixed).

The other easyDSP connected to CPU2 can use either SCI-B, SCI-C or SCI-D but easyDSP recommends to use SCI-B GPIO 87/86 as default in its source file (easy28x_DriverLib.c or easy28x_bitfield.c) . If another GPIO port is required in your system, please change the hardware connection and modify the easyDSP source file (in the function of easyDSP_SCI_Init) accordingly by yourself.

- Factory default setting is assumed (Don't change it)
- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V
- TX/RX pins are directly connected to MCU pins
- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO72 via $2k\Omega$ series resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.5 F28P55x/F28001x/28002x/28003x/28004x

Under factory default (OTP_BOOTPIN_CONFIG_KEY != 0x5A) and no emulator connected, MCU checks below two pins at reset to determine the booting mode.

MODE	GPIO24	GPIO32	Boot mode
Mode 0	0	0	Parallel I/O
Mode 1	0	1	SCI / Wait (RAM boot)

Mode 2	1	0	CAN
Mode 3	1	1	Flash (USB)

easyDSP uses two kinds boot mode, SCI boot mode for RAM booting, Flash boot mode for flash rom booting.

Therefore, below connection is recommended between easyDSP and MCU.



- Factory default setting is assumed

- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- SCIA_RX = GPIO28, SCIA_TX = GPIO29

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- TX/RX pins are directly connected to MCU pins

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO24 via $2k\Omega$ series resistor

- /RESET pin is connected to reset generation circuit of MCU board (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.6 F2823x/2833x

Boot mode of TMS320F2823x/2833x at reset is decided based on the pin status of four pins.

MODE	GPIO87 XA15	GPIO86 XA14	GPIO85 XA13	GPIO84 XA12	Boot mode
F	1	1	1	1	Jump to Flash
E	1	1	1	0	SCI-A boot (RAM boot)
D	1	1	0	1	SPI-A boot

С	1	1	0	0	I2C-A boot
В	1	0	1	1	eCAN-A boot
Α	1	0	1	0	McBSP-A boot
9	1	0	0	1	Jump to XINTF x16
8	1	0	0	0	Jump to XINTF x32
7	0	1	1	1	Jump to OTP
6	0	1	1	0	Parallel GPIO I/O boot
5	0	1	0	1	Parallel XINTF boot
4	0	1	0	0	Jump to SARAM
3	0	0	1	1	Branch to check boot mode
2	0	0	1	0	Branch to Flash, skip ADC calibration
1	0	0	0	1	Branch to SARAM, skip ADC calibration
0	0	0	0	0	Branch to SCI, skip ADC calibration

easyDSP activates both /BOOT and /RESET pins low for RAM booting. It activates only /RESET pin low for the menu 'DSP>Reset DSP'.

An easyDSP uses either 'Jump to Flash' mode or 'SCI-A boot' by setting GPIO84 pin as 1 or 0 while other three pins are fixed to 1. Therefore below circuit configuration is recommended.



- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to either GPIO84 or GPIO85 via $2k\Omega$ resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /Reset pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.7 C2834x

TMS320C2834x checks below four pins at the reset to decide the booting mode.

MODE	GPIO87 XA15	GPIO86 XA14	GPIO85 XA13	GPIO84 XA12	Booting mode
E	1	1	1	0	SCI-A boot (for RAM booting)
D	1	1	0	1	SPI-A boot (for flashrom booting)

easyDSP activates both /BOOT and /RESET pins low for RAM booting. And it activates only /RESET pin low for the menu 'DSP>Reset DSP'. So please connect easyDSP as below so that easyDSP can select appropriate RAM booting mode (SCI-A).

Blue box of above table is the recommendation for flashrom booting. Hardware preparation is your task.



(SCI-A boot @ RAM booting. SPI-A boot @ flashrom booting)

And please note belows.

- SPI-A is used for easyDSP. You can't use SPI-A for your purpose.
- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V
- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected via $2k\Omega$ series resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

Caution !!

When you select menu 'MCU'>'Reset MCU', only /RESET pin is activated low. /BOOT is still high at that time.

Therefore don't use this menu if you are not ready to use SPI-A boot mode.

7.1.2.8 F2802x/2802x0/2803x/2805x/2806x

Piccolo series TMS320F2802x/2802x0/2803x/2805x/2806x checks below three pins at the reset to decide the booting mode.

MODE	GPIO37 TDO	GPIO34 CMP2OUT	/TRST	Boot mode
Mode EMU	Х	Х	1	Emulation Boot
Mode 0	0	0	0	Parallel I/O
Mode 1	0	1	0	SCI (RAM boot)
Mode 2	1	0	0	Wait
Mode 3	1	1	0	GetMode

easyDSP uses two kinds boot mode. SCI boot mode for RAM booting, GetMode boot mode for flashrom booting.

In case there is no emulator connected (that is /TRST=0), fix GIOP34 to '1' and connect /BOOT pin to GPIO37 as shown below connection.

cf) In case there is emulator connected, boot mode is decided based on the memory value at the specific address. Please refer to the TI manual for the details.



- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V
- connect SCIRXDA = GPIO28, SCITXDA = GPIO29
- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO37 via $2k\Omega$ series resistor
- /RESET pin is connected to reset generation circuit of MCU board
- (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.9 F281x

GPIOF4(SCITXDA)	GPIOF12(MDXA)	GPIOF3(SPISTEA)	GPIOF2(SPICLK)	Boot mode
1	х	Х	х	FLASH(0x3F7FF6)
0	1	X	X	SPI boot
0	0	1	1	SCI boot (SCI-A) (RAM boot)
0	0	1	0	H0 SARAM(0x3F8000)
0	0	0	1	OTP (0x3D7800)

TMS320F281x checks below four pins at the reset to decide the booting mode.

easyDSP uses two kinds boot mode. 'SCI' for RAM booting, 'Flash' for flashrom booting (yellow part in above table). Therefore, fix GPIOF2, GPIOF3 and GPIOF12 to '1', '1' and '0' respectively. And connect GPIOF4(SCITXDA) to /BOOT pin of easyDSP, as shown in below connection.



- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to SCITXDA via $2k\Omega$ series resistor

- /RESET pin is connected to reset generation circuit of MCU board

(Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.2.10 F280x

TMS320F280x checks below four pins at the reset to decide the booting mode.

Boot mode	GPIO18 SPICLKA SCITXB	GPIO29 SCITXDA	GPIO34
Jump to Flash 0x3F 7FF6	1	1	1
Call SCI-A boot loader (RAM boot)	1	1	0
Call SPI-A boot loader	1	0	1
Call I2C-A boot loader	1	0	0
Call eCAN-A boot loader	0	1	1
Jump to M0 SARAM 0x00 0000	0	1	0
Jump to OPT	0	0	1
Parallel GPIO Loader	0	0	0

easyDSP uses two kinds boot mode. 'SCI-A' for RAM booting, 'Jump to Flash' for flashrom booting (yellow part in above table). Therefore, fix GPIO18, GPIO29 to '1'. And connect GPIO34 to /BOOT pin of easyDSP, as shown in below connection.



- power pin (#4) of easyDSP 5x2 header should be connected to 3.3V

- TX/RX pins are directly connected to MCU pins

- In case there is a reset IC between easyDSP /RESET and MCU /XRS, it should transfer easyDSP /RESET signal to MCU /XRS within 0.5sec

- In case you insert buffer IC between easyDSP header and MCU, place buffer IC directly to easyDSP header so that all resistors can be connected to directly MCU

- /BOOT pin is connected to GPIO34 via $2k\Omega$ series resistor

- /RESET pin is connected to reset generation circuit of MCU board (Time duration of /RESET pin is around 500msec)

- In case you use pull-up resistor to each pin, the value of pull-up resistor should be higher than a few kilo ohm since there is 100Ω series resistor inside easyDSP pod

Please be careful when you use your own pull-up or pull-down resistor on the easyDSP signal pins. Please use appropriate filter circuit to your reset generation circuit to prevent unintentional reset generation.

7.1.3 How to use other SCI port than designated

If you use different ports for easyDSP than recommended in previous section, you can do monitoring operation but can't do RAM booting and flash programming since MCU has dedicated port for its SCI booting. In case you really need to use different port, you can try below method. Here TMS320F28377D is taken as an example but the other MCU can be used in similar way.

How to use the other ports than GPIO85 and GPIO84 with TMS320F28377D for EMIF :

First step, SCI booting done by GPIO85/GPIO84 and later monitoring done by the other GPIOs. To do so, additional hardware is necessary to switch easyDSP connection from GPIO85/GPIO84 to the other GPIOs right after booting completion. Please refer to below circuit whereDual SPDT (NLAS4684 from Onsemi,TS3A24159 from TI) is used. FPGA can be used too.

http://www.onsemi.com/pub_link/Collateral/NLAS4684-D.PDF

http://www.ti.com/lit/ds/symlink/ts3a24159.pdf



To switch easyDSP connection, one more GPIO (here, GPIOx) is used. You can use any GPIO which you don't use in your application. The operation mechanism as below.

- After reset, GPIOx is input pin as reset default. The pull-up resistor on GPIOx decides SPDT connection, which makes easyDSP connected to GPIO85/84.- Once SCI booting is completed, it's user's task to switch easyDSP connection to the other ports. You can do as below.
- Makes GPIOx as output port and set its value to low, which makes easyDSP connection to GPIO28/29.

- The above operation can be done ineasyDSP_SCI_Init() in CPU1.
- Please change original coding as recommended below.

/BOOT pin of easyDSP pod has pseudo open collector type, which means it becomes low during booting for flash programming or RAM booting but open after booting. So, no additional measures are required when using GPIO72 as EMIF. But please note that easyDSP pod connection or disconnection during MCU operation is not recommended since it could make a unintended noise signal to GPIO72.

Using Get mode helps ? :

You might think to try Get mode since you can use SCI BOOT 1 in Get Mode after changing Zx-BOOTCTRL register. Since Zx-BOOTCTRL register is located in OTP area, you can not change its contens twice. Also you can not use flash booting.

7.1.4 C28x cautions

* F ail to boot with big coding size ?

It could happen likely with TMS320C2834x series since its code size is normally much bigger than that of other MCU series. Why? It's because it takes long long time to initialize variables in c_int00 routine and therefore after some time watch-dog makes unintentional reset to MCU. To prevent watch-dog reset during c_int00 routine operation, it is strongly recommended the entry point is set to the 'code_start' label (in TI's DSP28x_CodeStartBranch.asm) with watch-dog disabled. This is done by linker option -e in the project build options, that is, -ecode_start.

* Operating XDS100 together with easyDSP ?

XDS100v1 (TI or 3rd parties emulator) supports multiple FTDI devices only for CCS v4. Therefore when you use XDS100v1 with CCS v3.3, easyDSP can't be used together.

* If you use other SCI ports than easyDSP recommends to use ?

For example, easyDSP recommends to use GPIO28, 29 for SCIRXDA and SCITXDA respectively when SCI-communicating with F28335. If you use GPIO36 and GPIO35 instead, you will face the booting failure. It's because TI does not support serial booting via these pins (GPIO36 and 35).

* What if MCU is at the reset during easyDSP communication ?

It depends. If the boot mode after the reset is flashrom booting, then the MCU will boot again with the flashrom. If the boot mode after the reset is RAM booting, then MCU will boot with the serial data which easyDSP send for communication. It finally causes fatal error and can damage your system.

7.2 STM32

7.2.1 STM32 programming

STEP 1 : Selection of USART channel and its configuration

It will be explained based on STM32CubeMX.

Steps

STM32CubeMX



	Configuration	
	Reset Configuration	
Set the	Solution Settings Solution Settings Solution Settings Solution Settings Solution Setting Solution Settin	📀 GPIO Settings
communication	Parameter Settings	Sources User Constants
with 8 bits, no	Configure the below parameters :	
bit .	Q Search (CrtI+F) ③ ③	0
Baud rate is	✓ Basic Parameters	
selectable.	Baud Rate 115200 Bi	its/s
	Word Length 8 Bits (inc	luding Parity)
If MCU supports	Parity None	
FIFO with 8	Stop Bits 1	
levels or more in		
USARI, please		
'Ryfifo Threshold'	 Advanced Parameters 	Descine and Transmit
to '1 eight full	Data Direction	Receive and Transmit
configuration'.	Over Sampling	Disable
Note that	Single Sample	Disable
easyStmLL.c	CiockPrescalel Eifa Mada	l Enable
version 10.5 or	Typic Threshold	1 eighth full configuration
	Pyffo Threshold	1 eighth full configuration
	TXIIIO THRESHOLD	r eight i fuir comiguration
	Reset Configuration	
Enable interrupt	NVIC Settings ● DMA Settin ● Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt through the setting the se	gs
Enable interrupt	 NVIC Settings ○ DMA Settin ○ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ○ NVIC ○ Code generation 	gs
Enable interrupt	 NVIC Settings OMA Settin Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throu NVIC Code generation ✓ Sort by Parameter 	gs CPIO Settings User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority
Enable interrupt Go to 'system Core > NVIC' tab and	 NVIC Settings Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throu NVIC Code generation ✓ Sort by Present Search (CrtI+F) ③ ③ ④ Show only 	gs GPIO Settings User Constants Enabled Preemption ugh EXTI line 25 I 3
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of	 NVIC Settings DMA Settin Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throu NVIC Code generation ✓ Sort by Presence Search (CrtI+F) ③ ④ ✓ Show only NVIC Interrupt Table 	gs GPIO Settings User Constants Enabled Preemption ugh EXTI line 25 I 3
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the	 NVIC Settings DMA Settin Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throu NVIC Code generation ☑ Sort by Presearch Search (CrtI+F) ③ ④ ☑ Show only NVIC Interrupt Table Non maskable interrupt 	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 ✓ remption Priority and Sub Priority 3 remption Priority and Sub Priority Force DMA channels Interrupts Enabled Preemption Priority Image: Construct of the second
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of	 NVIC Settings DMA Settin Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut NVIC Code generation ✓ Sort by Present Search (Crt/+F) ✓ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt Switting SW/Linstruction Switting SW/Linstruction ✓ DMA Setting ✓ DMA Setting ✓ DMA Setting ✓ DMA Setting ✓ NVIC Interrupt Table Non maskable interrupt Hard fault interrupt Switting SW/Linstruction ✓ Switter service call via SW/Linstruction ✓ NVIC Interrupt Switter service call via SW/Linstruction ✓ NVIC Interrupt ✓ Switter service call via SW/Linstruction ✓ NVIC Interrupt ✓ Other Interrupt ✓ NVIC Interrupt ✓ NVIC Interrupt ✓ NVIC Interrupt ✓ NVIC In	gs Image: GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 Image: GPIO Settings remption Priority and Sub Priority 3 remption Priority and Sub Priority 3 y enabled interrupts Image: GPIO Settings Image: GPIO Settings Image: GPIO Settings
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority.	 NVIC Settings DMA Settin Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut NVIC Code generation ✓ Sort by Present Search (CrtI+F) ③ ④ ✓ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service 	gs GPIO Settings User Constants User Constants ugh EXTI line 25 Image: Constant s remption Priority and Sub Priority 3 remption Priority and Sub Priority 3 remption Priority and Sub Priority 9 Image: Constant s Image: Constant s <
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority.	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ✓ Sort by Prise Search (Crt/+F) ◇ ② ✓ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer	gs CPIO Settings User Constants User Constants User Constants User Constants User Constants Ugh EXTI line 25
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority.	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ✓ Sort by Pr Search Search (CrtI+F) ③ ③ ✓ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority 0 C 0 O O O O O O O O O O O O O O O O O O O
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority.	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ✓ Sort by Pr Search Search (CrtI+F) ③ ③ ④ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25 ④ GPI0 RCC	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority O O O O O O O O O O O O O O O O O O
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority. Go to ' System	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ✓ Sort by Pr Search Search (CrtI+F) ◇ ③ ④ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25 ④ GPI0 RCC ● SYS ● USART	gs GPIO Settings User Constants User Constants User Constants ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority O O O O O O O O O O O O O O O O O O
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority. Go to ' System Core > GPIO >	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ◇ NVIC ◇ Code generation ✓ Sort by Prise ◇ Sort by Prise ◇ ◇ ◇ ✓ Show only NVIC Interrupt Table NVIC Interrupt Table Non maskable interrupt NVIC Interrupt Table Non maskable interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25 ◇ GPIO ◇ RCC ◇ SYS ◇ USART Search Signals Search (CrtI+F)	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority. Go to ' System Core > GPIO > USART' tab, set the	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ✓ Sort by Pr Search Search (CrtI+F) ◇ ③ ④ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25 ● GPIO RCC ● SYS USART Search Signals Search CrtI+F) ● DPIO mode	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority. Go to ' System Core > GPIO > USART' tab, set the GPIO pin status	◇ NVIC Settings ◇ DMA Settin ◇ Parameter Settings NVIC Interrupt Table USART1 global interrupt / USART1 wake-up interrupt throut ◇ NVIC ◇ Code generation ◇ NVIC ◇ Code generation ◇ Sort by Pr Search Search (CrtI+F) ◇ ③ ◇ Show only NVIC Interrupt Table Non maskable interrupt Hard fault interrupt System service call via SWI instruction Pendable request for system service Time base: System tick timer USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25 ◇ GPIO RCC SYS ◇ BIGNION PIN GPIO output GPIO mode GPIO Pul-up PA2 USART2_TX Low Alternate Function Push Pull Pull-up PA3 USART2_TX Low Alternate Function Push Pull Pull-up	gs GPIO Settings User Constants Enabled Preemption ugh EXTI line 25
Enable interrupt Go to 'system Core > NVIC' tab and set the priority of USART interrupt lowest. That is, the highest number of priority. Go to ' System Core > GPIO > USART' tab, set the GPIO pin status with Pull-up.	Image: Second Section Second Section S	gs GPIO Settings User Constants User Constants Enabled Preemption ugh EXTI line 25 3 remption Priority and Sub Priority y enabled interrupts Force DMA channels Interrupts Enabled Preemption Priority O O O O O O O O O O O O O O O O O O

Go to 'Project	er Selector		
Manager > Advanced Settings	Search (Crt1+F) 🔇 📀	1	
> Driver Selector'	GPIO		HAL
tab and choose	RCC		HAL
LL . easyDSP 🛛 🗸 🗸	JSART		
supports only 11	USART2		HAL
	USART1		LL
based source file.			

STEP 2 : USART interrupt service routine based on LL

Thanks to smaller resource consumption than HAL, Only LL based easyDSP communication is supported.

For easyDSP to communicate with MCU via USART, source file for USART ISR (Interrupt Service Routine) should be included in your project.

Below is the source code based on LL and it's located in the folder 'Source > STM32' in the installed easyDSP.

easyStm32LL_v11.4.c easyStm32LL_v11.4.h

Please check below step by step procedure to modify your application code. For the additional settings for dual core MCU, please refer to <u>this page</u>.

steps	source code example
Define target MCU as 1 in the easyStm32LL.h file. No change to easyStm32LL.c file.	<pre>////////////////////////////////////</pre>

	<pre>/* USER CODE BEGIN Includes */ #include "easyStm32LL vx.y.h" // x.y = version x.y /* USER CODE END Includes */ int main(void) {</pre>
In the beginning of main.c, include easyStm32LL_vx.y.h where x.y is version.	
After calling MX_USARTx_UART_Init(), call easyDSP_init(USARTz)	MX_USART1_UART_Init();
z = selected USART channel. In this example USART1 is used.	<pre>/* USER CODE BEGIN 2 */ easyDSP init(USART1); /* USER CODE END 2 */</pre>
	<pre>while (1) {</pre>
In the beginning of stm32xxx_it.c file where ISR is defined, include easyStm32LL.h.	<pre>/* USER CODE BEGIN Includes */ #include "easyStm32LL vx.y.h" // x.y = version x.y /* USER CODE END Includes */ void USART1_IRQHandler(void) { /* USER CODE BEGIN USART1_IRQn 0 */ ez USARTx IRQHandler();</pre>
call ez_USARTx_IRQHandler() in the selected USART IRQ handler function.	<pre>/* USER CODE END USART1_IRQn 0 */ /* USER CODE BEGIN USART1_IRQn 1 */ /* USER CODE END USART1_IRQn 1 */ } In this example, the ISR file is stm32g0xx, it c</pre>
	In this example, the ISK me is stillszybxx_it.c.

STEP 3 : Dual core

The code of each CPU should be located in the different page of flash.

STEP 4 : IDE setting

1. Hex file (intel format) is used for ram booting and flash programming. So it should exist and be created in every compiling time in the same folder to output file (ex *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming and ram booting. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Please set your IDE to create hex file in every compilation accordingly.

Example of STM32CubeIDE :

Settings						⟨¬ ▼ ¬> ▼ 8
Configuration:	Debug [Active]			~	Manag	ge Configurations
🛞 Tool Setting	s 🎤 Build Steps 🖳	Build Artifact	Binary Parsers	🔕 Error	Parsers	
MCU Toolchain MCU Settings MCU Post build outputs		Convert to	binary file (-O binary Intel Hex file (-O ihe Motorola S-record fi	/) x) le (-O srec))	

2. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly (for example, -g option). The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. As an example with Stm32CubeIde, uncheck the 'Discard unused sections' box.

Example of STM32CubeIDE :



Settings	⟨¬ ▼ ¬⟩ ▼ 8°
Configuration: Debug [Active]	✓ Manage Configurations
🛞 Tool Settings 🎤 Build Steps 🕊	Build Artifact 🗟 Binary Parsers 🔇 Error Parsers
 MCU Settings MCU Post build outputs MCU GCC Assembler General Debugging Preprocessor Include paths Miscellaneous MCU GCC Compiler General Debugging Preprocessor General Debugging Preprocessor Include paths MCU GCC Compiler General Debugging Preprocessor Include paths Optimization Warnings Miscellaneous MCU GCC Linker Contraction Libraries Miscellaneous 	Linker Script (-T) \${workspace_loc:/\${ProjName}/STM32G071RBTX_FL/ System calls Minimal implementation (specs=nosys.specs) Generate map file (-WI,-Map=) Add symbol cross reference table to map file (-WI,cref) Discard unused sections (-WI,gc-sections) Verbose (-WI,verbose) Do not use standard start files (-nostartfiles) Do not use default libraries (-nodefaultlibs) No startup or default libs (-nostdlib)

Example of KEIL uVision :

Options for Target 'G071 HAL KEIL'	Х
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Select Folder for Objects Name of Executable: G071 HAL KEIL	
 Create Executable: G071 HAL KEIL₩G071 HAL KEIL ✓ Debug Information Create HEX File ☐ Browse Information Create Library: G071 HAL KEIL₩G071 HAL KEIL.lib 	
Options for Target 'G071 HAL KEIL'	×
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
✓ Use Memory Layout from Target Dialog X/O Base: Make RW Sections Position Independent R/O Base: Make RO Sections Position Independent R/W Base Don't Search Standard Libraries disable Warnings:	
Scatter File	
Misc controls -no_remove Linker control -cpu Cortex-M0+ *.o -library type=microlibstrictscatter "G071 HAL KEIL \$G071 HAL KEIL \$\scilled{G071 HAL KEIL \$\scil	
string	

Example of IAR Embedded Workbench :

Category: General Options Static Analysis Runtime Checking	Multi-file Cor	npilation Unused Publics		Fac	tory Settings
C/C++ Compiler	List	Preprocessor	Diagnos	stics MIS	RA-C:2004
Assembler	MISRA-C	:1998	Encodings	Extra	Options
Output Converter	Language 1	Language 2	Code	Optimizations	Output
Custom Build Build Actions Linker Debugger Simulator	Generate Code section text	debug information name:	•		

Category:					Factory Se	ettings
General Options Static Analysis Runtime Checking						
C/C++ Compiler	#define	Diagnostics	Checksum	Encodings	Extra O	ptions
Assembler	Config	Library Input	Optimizations	Advanced	Output	List
Output Converter Custom Build Build Actions	Output file G071 H/	ename: ALIAR.out		·		
Linker Debugger Simulator		le debug informati	on in output			

7.2.2 STM32 hardware

STEP 1 : Selection of USART channel and its pins for boot mode operation

easyDSP uses USART communication to interface with MCU and also for flash programming under bootloader. So, first step should be choosing proper USART channel and its pins.

Please check ST's application note (<u>AN2606 : STM32 microcontroller system memory boot mode</u>) and choose USART channel and its pins on your needs. UART channel is not usable.

(Note : as of Apr 2025, no information about STM32WL3x in the AN2606. Please use USART1 Rx (PA15) and USART1 Tx (PA1) for QFN48 package or USART1 Rx (PB14) and USART1 Tx (PA1) for QFN32 package)

Note that USART channel should be supported by bootloader. For example, in case of STM32F413x, check the table below.

AN2606

STM32F413xx/423xx devices bootloader

Bootloader	Feature/Peripheral	State	Comment
	USART1	Enabled	Once initialized the USART1 configuration is: 8-bit, even parity and 1 Stop bit
USART1 bootloader	USART1_RX pin	Input	PA10 pin: USART1 in reception mode
	USART1_TX pin	Output	PA9 pin: USART1 in transmission mode
	USART2	Enabled	Once initialized the USART2 configuration is: 8-bit, even parity and 1 Stop bit
USART2 bootloader	USART2_RX pin	Input	PD6 pin: USART2 in reception mode
	USART2_TX pin	Output	PD5 pin: USART2 in transmission mode
	USART3	Enabled	Once initialized the USART3 configuration is: 8-bit, even parity and 1 Stop bit
USART3 bootloader	USART3_RX pin	Input	PB11 pin: USART3 in reception mode
	USART3_TX pin	Output	PB10 pin: USART3 in transmission mode

Table 67. STM32F413xx/423xx configuration in system memory boot mode (continued)

If you choose USART2, then you should usePD5 and PD6 pin.

Accrodingly set the PD5 and PD6 as USART2 in the STM32CubeMX.



Caution-1 : Below MCU-USART-Pin combination is not recommended due to its restriction.

MCU	USART	Pin	Limitation
STM32F03xx4/6	USART1	PA14 PA15	SWD not available during bootloader operation because PA14(SW_CLK) is used by bootloader.
STM32F030xC STM32F05xxx STM32F030x8 STM32F04xxx STM32F070x6 STM32F070xB STM32F071xx STM32F071xx STM32F072xx STM32F09xxx	USART2	PA14 PA15	SWD not available during bootloader operation because PA14(SW_CLK) is used by bootloader.

Caution 2 : Due to bugs in the bootloader (esp. with old version), please don't use below MUC-BL ID-USART combinations.

Please check AN2606 for its details.

Please note that the other combination than this could be not working due to undocumented

bugs.

MCU	BL ID	USART
STM32F105xx/107xx	V2.0 (0x20)	USART1,USART2
STM32F412xx	V9.0 (0x90)	USART3
STM32G05xxx/061xx	V5.0 (0x50)	USART2
STM32H74xxx	V13.2 (0xD2)	USART2
STM32H75xxx		
STM32L552xx	V13.0 (0xD0)	USART3
STM32L562xx		
STM32L47xxx/48xxx	V9.2 (0x92)	USART2
STM32L496xx/4A6xx	V9.3 (0x93)	USART2, USART3
STM32L4P5xx/Q5xx	V9.0 (0x90)	USART2, USART3
STM32L4Rxx/4Sxx	V9.2 (0x92)	USART2, USART3
STM32L4RxG/4SxG	V9.2 (0x92)	all USARTx

STEP 2 : easyDSP pod connection

Connect easyDSP pod to the USARTx selected in step 1. In case of STM32F1, STM32F4 and STM32L1, pulldown to BOOT1 pin. easyDSP pod VDD pin is connected to MCU VDD pin. easyDSP pod TX and RX pin is pulled up with 100k Ohm resistor inside of easyDSP pod.

In case there is a reset IC between easyDSP /RESET and MCU NRST, it should transfer easyDSP /RESET signal to MCU within 0.5sec.



Note)

STM32WB0 and STM32WL33xx : PA10 pin is BOOT0 pin.

STM32H74xxx/75xxx : don't pulldown PB15 pin.

STM32G03xx/04xxx : don't pulldown PA3 pin if the version of bootloader is either v5.1 or v5.2. STM32C011xx : On WLCSP12, SO8N, TSSOP20 and UFQFN20 packages, USART1 PA9/PA10 IOs are remapped on PA11/PA12.

STM32C031xx : On TSSOP20 and UFQFN28 packages, USART1 PA9/PA10 IOs are remapped on PA11/PA12.

STEP 3 : MCU option byte

The option byte of MCU should be set properly before using easyDSP. Since easyDSP can't change it, It's your task to change option byte by using Stm32CubeProgrammer.

For easyDSP to access the memory, no protection or security should be active such as

- RDP (Readout Protect)
- WRP (Write Protect)
- PCROP (Proprietary code read-out protection)
- Securable memory

easyDSP contols BOOT0 pin to determine MCU boot mode after reset, either boot from flash (BOOT0 pin low) or boot from system memory (BOOT0 pin high). Option bytes in the MCU should be set accordingly.

Below captures from Stm32CubeProgrammer could be different slightly depending MCU type.

- BOOT_LOCK should be not used so that easyDSP can use bootloader : BOOT_LOCK = unchecked

BOOT_LOCK	Unchecked : Boot based on the pad/option bit configuration Checked : Boot forced from Main Flash memory
BOOT_LOCK	Unchecked : CPU1 CM4 Boot lock disabled Checked : CPU1 CM4 Boot lock enabled
C2BOOT_LOCK	Unchecked : CPU2 CM0+ Boot lock disabled Checked : CPU2 CM0+ Boot lock enabled

- If MCU has product state, it should be OPEN for flash programming.

		Life state code.
		ED : Open
		17 : Provisioning
PRODUCT_STATE	ED 🔻	2E : Provisioned
		72 : Closed

- NRST pin should be reset input pin : NSRT_MODE = 1 or 3

NRST_MODE	3	-

0 : Reserved
 Reserved
 Reserved
 Reset Input only: a low level on the NRST pin generates system reset, internal RESET not propagated to the NSRT pin
 2 : GPID: standard GPID pad functionality, only internal RESET possible
 Bidirectional reset: NRST pin configured in reset input/output mode (legacy mode)

- Boot mode should be determined by BOOT0 pin which is controlled by easyDSP : nBOOL_SEL = unchecked, BOOT_SEL = checked, nBOOT1 = checked, nSWBOOT0 = checked

nSWBOOT0	Unchecked : BOOT0 taken from the option bit nBOOT0 Checked : BOOT0 taken from PB8/BOOT0 pin
nBOOT1	Unchecked : Boot from Flash if BOOT0 = 0, otherwise Embedded SRAM1 Checked : Boot from Flash if BOOT0 = 0, otherwise system memory
BOOT_SEL	Unchecked : BOOT0 signal is defined by nBOOT0 option bit Checked : BOOT0 signal is defined by BOOT0 pin value
nBOOT_SEL	Unchecked : BOOT0 signal is defined by BOOT0 pin value (legacy mode) Checked : BOOT0 signal is defined by nBOOT0 option bit

- If different boot areas can be selected through the BOOT pin and the boot base address programmed in the BOOT_ADD0 and

BOOT_ADD1 option bytes, the BOOT_ADD0 and BOOT_ADD! should be the address of flash and system memory respectively.

for ex, in case	of STM32H7A3,		
Name	Value		
BOOT_CM7_ADD0	Value 0x800 Address 0x80	000000 Define	the boot address for Cortex-M7 when BOOT0=
BOOT_CM7_ADD1	Value 0x1ff0 Address 0x1f	f00000 Define	the boot address for Cortex-M7 when BOOT0=
n case of STM	32F767		
Name	Value		
BOOT_ADD0	Value 0x80 Address	0x00200000	Define the boot address when BOOT0=0
BOOT_ADD1	Value 0x40 Address	0x00100000	Define the boot address when BOOT0=:
case of STM3 BCM4	2H7 dual core MCU, bo Vr Ch	th cores are b nchecked : CM4 b necked : CM4 b	oot-enabled. oot disabled oot enabled
BCM7	Vr Ch	nchecked : CM7 b hecked : CM7 b	oot disabled oot enabled
n case of STM3	2WL dual core :		
C2OPT		Unchecked : SBRV will addr Checked : SBRV will addr	ess SRAM1 or SRAM2, from start address 0x2000 0000 + SE ess Flash memory. from start address 0x0800 0000 + SBRV.

7.2.3 STM32 dual core

Target MCU

STM32H745x, STM32H747x, STM32H755x, STM32H757x (CPU1 = Arm Cortex-M7, CPU2 = Arm Cortex-M4) STM32WL55xx, STM32WL54xx (CPU1 = Arm Cortex-M4, CPU2 = Arm Cortex-M0+)

Common

MCU cores are classified with 4 kinds in terms of easyDSP.

Yellow core : core that easyDSP pod is connected to and easyDSP communicates with Orange core : core that easyDSP pod is not connected to but easyDSP communicates with Blue core : core that easyDSP doesn't communicate with Gray core : core that doesn't run

	Connected to easyDSP pod	Communicated with easyDSP	Running core
core	Yes	Yes	Yes
core	No	Yes	Yes
core	No	No	Yes
core	No	No	No

STM32 dual core MCU has 2 cores. Please choose core type either yellow or orange core based on your application.

Since blue and gray core has no operation with easyDSP, no easyDSP related setting is required for them.

In the project settings, you can designate the output file of the running cores. If two cores are running in the user program, two output files can be specified. These output files are used when flash programming.

Also check the core which easyDSP is communicating with (monitoring).

Below example shows the case that two cores are running (and therefore easyDSP supports flash programming of two cores) and easyDSP is monitoring only CPU1.

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	ST 💌	
Series	STM32 H7	
Part number	STM32H755xI	
Output File(s) -	C:₩temp₩cpu1.out	Communication with easyDSP
CPU2 (M4)	C:\temp\temp\tempu2.out	
		OK Cancel

When easyDSP monitors two cores CPU1 and CPU2, to devide the variable name of each core, easyDSP adds prefix to the original name, "1:" to CPU1 variables, "2:" to CPU2 variables. For example, if the name of variable is "var1" in your CPU1 program, easyDSP displays it as "1:var1".

STM32WL dual core

easyDSP offers two options as below. The arrow in the picture means the data flow between easyDSP and CPU.

easyDSP project should be created for all the yellow cores.



Case 1 :

Each CPU has a own connection to easyDSP pod. For each CPU, please set accordingly to <u>what is</u> <u>described in the previous pages</u>.

If you register optional output file, each easyDSP project can flash for both CPU1 and CPU2. If not, each easyDSP project can flash only one CPU.

Settings	CPU1	CPU2
easyDSP project	register CPU1 output file regieter CPU2 output file (optional) check CPU1 check box	register CPU2 output file regieter CPU1 output file (optional) check CPU2 check box
main.c	call easyDSP_init(USARTn)	call easyDSP_init(USARTm)
stm32h7xx_it.c	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()

Case 2 :

easyDSP is connected to CPU1 and makes an access to all the memory via CPU1. CPU2 can't be used for this purpose.

Therefore like single core MCU, easyDSP related settings are same to <u>what is described in the previous</u> <u>pages</u>. There is no easyDSP related setting to CPU2.

Settings	CPU1
easyDSP	register CPU1 and CPU2 output files
project	check CPU1 and CPU2 check boxes

main.c	call easyDSP_init(USARTn)
stm32h7xx_it.c	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()

STM32H7 dual core

Depending on data cache usage (Stm32CubeMx > System Core > CORTEX_M7 > Parameter Settings > Cortex Interface Settings > CPU DCache), easyDSP offers three different connections. The arrow in the picture means the data flow between easyDSP and CPU.



Case 1 :

Each CPU has a connection to easyDSP. This configuration can be used independent of data chache usage.

For each CPU, please set accordingly to <u>what is described in the previous pages</u>. If you register optional output file, each easyDSP project can flash for both CPU1 and CPU2. If not, each easyDSP project can flash only one CPU.

Settings	CPU1	CPU2		
easyDSP project	register CPU1 output file register CPU2 output file (optional) check CPU1 check box	register CPU2 output file register CPU1 output file (optional) check CPU2 check box		
easyStm32LL.h	EZ_DUAL_CORE = 1 EASYDSP_IS_CONNECTED_TO_THIS_CORE = 1 EZ_USE_SEV_INT = 0	EZ_DUAL_CORE = 1 EASYDSP_IS_CONNECTED_TO_THIS_CORE = 1 EZ_USE_SEV_INT = 0		
main.c	call easyDSP_init(USARTn)	call easyDSP_init(USARTm)		
stm32h7xx_it.c	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()		

Case 2 :

If data cache is not used, easyDSP can access all the memory via CPU1. CPU2 can't be used for this purpose. Therefore like single core MCU, easyDSP related settings are same to <u>what is described in the previous pages</u>.

There is no easyDSP related setting to CPU2.

Settings	CPU1
easyDSP project	register CPU1 and CPU2 output files check CPU1 and CPU2 check boxes
easyStm32LL.h	EZ_DUAL_CORE = 1 EASYDSP_IS_CONNECTED_TO_THIS_CORE = 1 EZ_USE_SEV_INT = 0
main.c	call easyDSP_init(USARTn)
stm32h7xx_it.c	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler()

Case 3 :

If data cache is enabled, easyDSP uses SEV interrupt and dedicated shared memory to avoid cache coherence issue.

easyDSP pod can be connected to either CPU1 or CPU2. Please select the proper CPU for easyDSP pod connection based on your application.

SEV interrupt should be enabled with the lowest priority in the STM32CubeMx > System Core > NVIC1 and NVIC2.

NVIC1 Interrupt Table	Enabled	Preemption Priority	
Non maskable interrupt	\checkmark	0	
Hard fault interrupt	~	0	
Memory management fault	~	0	
Pre-fetch fault, memory access fault	~	0	
Undefined instruction or illegal state	~	0	
System service call via SWI instruction	~	0	
Debug monitor	\checkmark	0	
Pendable request for system service	\checkmark	0	
Time base: System tick timer	\checkmark	0	
USART3 global interrupt	~	15	
CM4 send event interrupt for CM7	~	15	

NVIC2 Interrupt Table	Enabled	Preemption Priority
Non maskable interrupt	\checkmark	0
Hard fault interrupt	~	0
Memory management fault	\checkmark	0
Pre-fetch fault, memory access fault	\checkmark	0
Undefined instruction or illegal state	\checkmark	0
System service call via SWI instruction	\checkmark	0
Debug monitor	\checkmark	0
Pendable request for system service	\checkmark	0
Time base: System tick timer	\checkmark	0
USART1 global interrupt	Image: A start and a start	15
CM7 send event interrupt for CM4	✓	15

The shared memory could be located anywhere but the location of SRAM4 is recommended. Note that 1. This memory area (32 bytes from start address) should not be used by both CPU1 and CPU2. Please take care of linker script file.

The start address should be aligned to 32 bytes. For example, 0x38000000 or 0x38000020
 This memory are should be non cacheable. MPU settings are necessary in the Stm32CubeMx > System Core > CORTEX_M7.

Cortex Memory Protection Unit Region 0 Settings

MPU Region	Enabled
MPU Region Base Address	0x38000000
MPU Region Size	32B
MPU TEX field level	level 1
MPU Access Permission	ALL ACCESS PERMITTED
MPU Instruction Access	DISABLE
MPU Shareability Permission	ENABLE
MPU Cacheable Permission	DISABLE
MPU Bufferable Permission	DISABLE

Finally include easyDSP source file to both CPU1 and CPU2 projects and set properly as below table.

Settings	CPUx (easyDSP pod is connected to)	CPUy (easyDSP pod is not connected to)		
easyDSP project	register CPU1 and CPU2 output files check CPU1 and CPU2 check boxes	no easyDSP project		
easyStm32LL.h	EZ_DUAL_CORE = 1 EASYDSP_IS_CONNECTED_TO_THIS_COR E = 1 EZ_USE_SEV_INT = 1 EZ_SHARED_MEM_ADDRESS = user defined	EZ_DUAL_CORE = 1 EASYDSP_IS_CONNECTED_TO_THIS_COR E =0 EZ_USE_SEV_INT = 1 EZ_SHARED_MEM_ADDRESS = user defined		
main.c	call easyDSP_init(USARTn)	call easyDSP_init(0)		
stm32h7xx_it. c	call USARTx_IRQHandler() in the ez_USARTx_IRQHandler() call CMx_SEV_IRQHandler() in the ez_SEV_IRQHandler()	call CMx_SEV_IRQHandler() in the ez_SEV_IRQHandler()		

7.2.4 STM32 RAM booting

You can skip this page if you don't use RAM booting.

easyDSP is supporting RAM booting using boot loader of MCU.

Therefore, all the differences from RAM booting with debugger comes from bootloader.

Please note that ram booting using boot loader has some limitation such as limited RAM area and some bugs in boot loader.

Please refer to below guideline for its implementation.

Steps	Example or further explanation				
	1. Below MCU can't support RAM booting. STM32F04xxx STM32F070x6 STM32L01xxx/02xxx STM32L031xx/041xx				
	2. If bootloader of MCU is not the latest one, RAM booting is blocked. Please check the MCU and bootloader version in the table. If the latest bootloader is in the MCU, no limitation. For its details, please check the latest version of AN2606 (STM32 microcontroller system memory boot mode).				
	MCU Bootloader version				
	STM32H74xxx STM32H75xxx	V13.2 (0xD2)			
1. Limitations	STM32L552xx STM32L562xx	V13.0 (0xD0)			
	STM32L47xxx	V10.1 (0xA1)			
	STM32L48XXX	V9.0 (0X90)			
	STM32F100xx STM32F101xx STM32F102xx STM32F103xx (except STM32F101xF, STM32F101xG, STM32F103xF, STM32F101xG)	V2.0 (0x20)			
	3. no RAM booting supported for	dual core MCU (H745, H	747, H755, H757, WL5x)		

2. Modification			Table 145. Bootloader device	-dependent	parameters	s (continued)	
of RAM		STM32 Series	Device	PID	BL ID	RAM	System memory
memory map in the linker script file			STM22540-mm/44-mm	0-442	0x31	0x20002000 - 0x2001FFFF	
			51M32P40XXX/41XXX	0,413	0×90	0x20003000 - 0x2001FFFF	
			STM32F42xxx/43xxx	0x419	0x70 0x91	0x20003000 - 0x2002FFFF	
User code			STM32F401×B(C)	0x423	0xD1	0x20003000 - 0x2000FFFF	
can't reside in the RAM area			STM32F401xD(E)	0x433	0xD1	0x20003000 - 0x20017FFF	
which MCU		F4	STM32F410xx	0x458	0xB1	0x20003000 - 0x20007FFF	0x1FFF0000 - 0x1FFF77FF
using.			STM32F411xx	0x431	0xD0	0x20003000 - 0x2001FFFF	
Also there is a memory area			STM32F412xx	0x441	0x90	0x20003000 - 0x2003FFFF	
which is not accessible in			STM32F448xx	0x421	0×90	0x20003000 - 0x2001FFFF	
the bootload mode.			STM32F469xx/479xx	0x434	0×90	0x20003000 - 0x2005FFFF	
So linker			STM32F413xx/423xx	0x463	0x90	0x20003000 - 0x2004FFFF	
So, linker script file should be modified so that user code reside in the RAM properly. Please check the RAM area usable for RAM booting in the latest AN2606(STM3 2 microcontroller system memory boot mode) .		This exa In STM: /* Memo MEMORY { RAM FLASH } But first Therefo /* Memo MEMORY { RAM FLASH }	ample is based on STM32F413 32F413ZHTX_RAM.Id file, RAM pries definition */ (xrw) : ORIGIN = 0x200000 (rx) : ORIGIN = 0x800000 t (rx) : ORIGIN = 0x800000 t 12k byte is used by bootload re please modify RAM area to pries definition for RAM boot (xrw) : ORIGIN = 0x2000 (rx) : ORIGIN = 0x8000	<pre>der and u start fro ing*/ 3000, LENC 3000, LE </pre>	defined a TH = 3201 TH = 1530 ser code m 0x2000 .ENGTH = 12 NGTH = 12	s below. K 6K can't use th 03000. 308K 536K	is area.
3. Locate ISR vector table in the first address of RAM memory		For RAN RAM me So, the Stm32C SECTIO You dor In case	I booting, easyDSP assumes a emory. vector table should be locate CubeIde, this condition is met NS. Since this is default feature of t need to do any additional j you use another Ide, please	ISR vecto d in the f by placin ire of link ob if you make sur	or table is irst addre ig .isr_ve er script use Stm3 e this cor	located in t ess of RAM r ctor in the f file Stm32C 32CubeIde. ndition is im	he first address on nemory. In case of irst part of ubeIde generates plemented.

	<pre>/* Sections */ SECTIONS { /* The startup code into "RAM" Ram type memory */ .isr_vector : { . = ALIGN(4); KEEP(*(.isr_vector)) /* Startup code */ . = ALIGN(4); } >RAM</pre>		
4. register the modified linker script file in the linker option.	 Toolchain Version Tool Settings Build Steps Puild Artifact Binary Parsers Error Parsers MCU Settings MCU Post build outputs MCU GCC Assembler Generat map file (-WI, -Map=) Add symbol cross reference table to map file (-WI,cref) Preprocessor Include paths Werbose (-WI,verbose) Verbose (-WI,verbose) Do not use default libraries (-nodefaultlibs) Do not use default libraries Molu GCC Linker General Minewallaneous MCU GCC Linker General Miscellaneous MCU GCC Linker General Miscellaneous 		
5. Change of vector table address	Again, this example is based on STM32F413. system_stm32f4xx.c BEFORE change : VECT_TAB_OFFSET is defined as 0x00 for flashrom booting. /* #define VECT_TAB_OFFSET 0x00 /*!< Vector Table base offset field. This value must be a multiple of 0x200. */ system_stm32f4xx.c AFTER change : Since the user code starts from 0x20003000, VECT_TAB_OFFSET should be changed to 0x3000. Please define VECT_TAB_SRAM and set the VECT_TAB_OFFSET to 0x3000. Note) you need to define USER_VECT_TAB_ADDRESS in some MCU cases (ex, STM32L5, STM32U3) Below is the recommendation. You can easily switch between RAM booting and flash booting by defining VECT_TAB_SRAM or not respectively. #ifdef VECT_TAB_OFFSET 0x3000 #else #define VECT_TAB_OFFSET 0x3000 #else #define VECT_TAB_OFFSET 0x3000 /*!< Vector Table base offset field. This value must be a multiple of 0x200. */ #endif		

6. Others	Depending on the MCU and its bootloader version, further consideration is necessary :
	case1 : Bootlader version 9.0 with STM32H74x/H75x stack pointer in the STM32H743ZITX_RAM.Id file as shown below
	<pre>_estack = ORIGIN(RAM_D1) + LENGTH(RAM_D1); /* end of "RAM_D1" Ram type memory */ should be changed to below by adding -16.</pre>
	_estack = ORIGIN(RAM_D1) + LENGTH(RAM_D1) - 16; /* Application stack pointer must be lower than (RAM end @ - 16 bytes) */
	<pre>case 2 : STM32WB55 Below three lines should be inserted at the end of STM32WB55RGVX_RAM.ld file. MAPPING_TABLE (NOLOAD) : { *(MAPPING_TABLE) } >RAM_SHARED MB_MEM1 (NOLOAD) : { *(MB_MEM1) } >RAM_SHARED MB_MEM2 (NOLOAD) : { _sMB_MEM2 = . ; *(MB_MEM2) ; _eMB_MEM2 = . ; } < /FONT ></pre>
	case 3 :STM32WBA NEW stack pointer in the STM32WBA52CGUX_RAM.ld file as shown below
	_estack = ORIGIN(RAM) + LENGTH(RAM); should be changed to below by adding -16. _estack = ORIGIN(RAM) + LENGTH(RAM) - 16;

7.2.5 STM32 cautions

Some communication IO pins are set to output pin during bootloader operation

Sometimes MCU enters into bootloader operation. For example, RAM booting and flash operation of easyDSP are executed in the bootloader operation of MCU. Some MCU enters bootloader after reset if the flash of MCU is empty.

Special care should be taken for your board design considering that some communication IO pins are set as output pin during bootloader operation. You can identify these pins with ST's application note (<u>AN2606 : STM32 microcontroller system memory boot mode</u>). In your board design, there should be no damage even under bootloader operation which sets some IO pins the output. For example, if these IO pins are connected to directly VDD or GND, the damage could be caused.

Full rebuild of STM32CubeIDE

STM32CubeIDE requests full rebuild if the project setting has a major change. In this case, all files in the compiler's output folder will be deleted. If your easyDSP project is located in the compiler's output folder, all easyDSP files also will be deleted.


USART baud rate

If the allowable resource for USART interrupt is limited, high baud rate could make overrun error.

Bank mode in the MCU name

In case of some STM32 MCU, single or dual bank is specified in the MCU name only when bank mode should be specified. That is, there is no bank mode in the STM32 MCU name either when bank mode is fixed (single or dual) in the MCU or when there is no need for understanding bank mode for easyDSP operation.

7.3 S32

7.3.1 S32K1 + SDK

This page assumes that the user uses S32 Configuration Tools and S32K1 SDK API.

STEP 1 : Hardware

Please select the UART channel and pins according to your board. No constraints to selectable channel and pin.

Then connect them to easyDSP like below.

In case flash programming is not used, no need to connect /BOOT and /RESET pins.



Other considerations :

- In case there is a reset IC between easyDSP /RESET and MCU /RESET, it should transfer the signal within 0.5sec.

- TX and RX pin of easyDSP header is pulled up with 100k Ohm resistor inside of easyDSP pod.

STEP 2 : S32 Configuration Tools

As explained in STEP1, please select the UART channel and pins. And set the configuration tool (Pins tab) accordingly.

Also kindly set the identifier as 'EZ_TX' and 'EZ_RX' respectively for TX and RX pins. In below example, PTA2 and PTA3 are chosen as RX and TX respectively with LPUARTO. Also set the pin properties as shown in Rounting Details tab. Note that pull-up should be set.



And add the lpuart module in the Drivers.

Components 🛛 🦞 Peripherals	- 8	Select configuration cor	nponent — 🗆 X
type filter text		Select which components sl	hould be offered All
Drivers	0	type filter text	
edma_config_EDMA	osif	Configuration component	Component description
-		💧 Ipspi	LPSPI configuration
PAL	0	Iptmr	Low Power Timer
		Ipuart	LPUART Configuration
OS	0	mpu_config	S32 SDK Peripheral Driver for Memory Protection Unit (MI
		osif	OSIF configuration
Libraries	0	bdb_config	Programmable Delay Block
		bower_manager	Power Configuration
Middleware	0	💩 rtc	Real-Time Clock
		💧 trgmux	S32 SDK Peripheral Driver for Trigger MUX Control (TRGN
		d wdog_config	S32 SDK Peripheral Driver for WatchDog Timer (WDOG)
			OK Cancel

And set the module properties. Its name should be set same to below. The UART channel is set as STEP2. In this example, it is set as LPUARTO as same as STEP2.

Also set communication properties as shown. The baudrate should be same one to one in the easyDSP project setting.

Components 23 🖞 Peripherals	- 0	Start 🕄 ez_lpuart	3			- 0
type filter text	0 11	LPUART Confi	guration [Drivers	5]		۵ 🖌 🌑
Drivers	0	Name ez_lpuart				Custom name 🗹
edma_config_EDMA ez_lpuart osi	f	Mode General Mode			Peripheral LPUART_0	
PAL	0	V LPUART configura	ition set		Preset Custom	
05	0	Driver State Structure	Name ez_IpUar	tState		
librarian		V LPUART Config	uration + ×			
Libraries		ez Ipuart InitCor	Namo	er levert leitCeefie		
Middleware	0		Read-only			
			Transfer Type	Interrupts		
			Baudrate	115200		
			Actual Baudrate	115942		
			Parity Mode	Disabled		
			Stop Bits	1		
			Bits per_char	8		
			RX DMA Channel	0		
			TX DMA Channel	0		

Also make sure the clock to the UART channel is set properly and enabled. Please refer to below example.

un Mode RUN	Clock Name	Enable	Control	Source	Divider	DivType	Frequency	Monito
	ADC0_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
NOTE:	CMP0_CLK	\checkmark		Bus clock			48 MHz	
 To provide clock for peripheral, us 	CRC0_CLK	\checkmark		Bus clock			48 MHz	
For example, FTMn, when enabling	DMA0_CLK	\checkmark		System clock			48 MHz	
	DMAMUX0_CLK	\checkmark		Bus clock			48 MHz	
	EIM0_CLK	\checkmark		System clock			48 MHz	
SOSC DIV2 CIN	ERM0_CLK	\checkmark		System clock			48 MHz	
SIRC DIV2 Clock LPSPI0_CLK	FLEXCAN0_CLK	\checkmark		System clock			48 MHz	
FIRC DIV2 Clock 8 MHz	FTFC0_CLK	\checkmark		Flash clock			24 MHz	
	FTM0_CLK	\checkmark		SCG SIRC DIV1 clock			8 MHz	
	FTM1_CLK	\checkmark		SCG SIRC DIV1 clock			8 MHz	
SOSC DIV2 Clock	FlexIO0_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
	LPI2C0_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
THIC Dive clock	LPITO_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
	LPSPI0_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
	LPSPI1_CLK	\checkmark		SCG SIRC DIV2 clock			8 MHz	
SOSC DIV2 Cloph	LPUART0_CLK			SCG SIRC DIV2 clock			8 MHz	
	LPUART1_CLK	\checkmark		SCG SOSC DIV2 clock			8 MHz	
	MPU0_CLK	\checkmark		System clock			48 MHz	
· · · · · · · · · · · · · · · · · · ·	MSCM0_CLK	\checkmark		System clock			48 MHz	
SOSC DIVA CIA	PDB0_CLK	\checkmark		System clock			48 MHz	
SIRC DIV2 Clock LPUART1_CLK	PORTA_CLK	\checkmark		Bus clock			48 MHz	
FIRC DIV2 Clock 8 MHz	PORTB_CLK	\checkmark		Bus clock			48 MHz	
	PORTC_CLK	\checkmark		Bus clock			48 MHz	
	PORTD_CLK	\checkmark		Bus clock			48 MHz	
SOSC DIV2 CION	PORTE_CLK	\checkmark		Bus clock			48 MHz	
SIRC DIV2 Clock	RTC0_CLK	\checkmark		Bus clock			48 MHz	

So far, the setting is for the communication with easyDSP for monitoring variables. If you like to use the easyDSP bootloader for flash programming, the following process is also required

because easyDSP bootloader uses flash driver. Please add flash component in the Drivers and change the names as shown below.

🗬 Components 🖾 🦞 Peripherals		- 8	Select configuration co	mponent		\times
type filter text		•	Select which components sl	hould be offered All		
Dri	vers	0	type filter text		 	
edma_config_EDMA	ez_lpuart	osif	Configuration component	Component description		
			📤 flash	FLASH		
P	AL	0	flexcan_config	FlexCAN Configuration		
			flexio_i2s_config	Flexio I2S		
c	DS	0	flexio_i2c_config	Flexio I2C		
			flexio_spi_config	Flexio SPI		
Libr	aries	0	flexio_uart_config	Flexio UART		
			IlexTimer_ic	FTM configuration		
Middl	leware	0	IlexTimer_mc	FTM configuration		
			IlexTimer_oc	FTM configuration		
			IlexTimer_pwm	FTM configuration		
			flexTimer_qd	FTM configuration		
				ОК	Cance	

Components 🕴 🖞 Peripherals	- 0	😨 Start 🔒 e	z_lpuart 😫 ez_flash 💈	2						- D
type filter text	0 11	FLASH [[Drivers]							🖹 🚡 💶
Drivers	0	Name ez_fla	ish							Custom name 🗹
edma_config_EDMA ez_flash ez_lpuart	osif	Mode Gene	al					Peripheral FTFC		
PAL	0	V FLASH C	onfiguration							Preset Custom
05	0	🗸 User (Configuration List	+ ×						
Libraries	0		Configuration	Read-only	PFlash base address	PFlash size	DFlash base address	EERAMBase address	Callback	
Middleware	0	0	ez_Flash_InitConfig	M	0x0000000	0x40000	0x1000000	0x14000000	NULL_CALLBACK	

STEP 3 : Source code correction for easyDSP bootloader

Please skip this step if you don't program flash with easyDSP. You can find the source file flash_driver.c which is generated by Configuration Tool in the below location. easyDSP uses two functions. To make them run in the ram, first declare them as in the red box in the beginning of the file,



then add the macro like below at the location of function definition in the middle of the file.

```
START FUNCTION DEFINITION RAMSECTION
status t FLASH DRV EraseSector (const flash ssd config t * pSSDConfig,
                                uint32 t dest,
                                uint32 t size)
ł
    .... // contents of this function
3
END FUNCTION DEFINITION RAMSECTION
START FUNCTION DEFINITION RAMSECTION
status t FLASH DRV Program(const flash ssd config t * pSSDConfig,
                            uint32_t dest,
                            uint32_t size,
                            const uint8 t * pData)
ł
    .... // contents of this function
}
END FUNCTION DEFINITION RAMSECTION
```

In case the Configuration Tool detects the correction of this file and ask like below, please choose 'Keep existing'.



STEP 4 : Calling easyDSP functions

Three files are provided for easyDSP communication and flash programming (easyS32K1_SDK.h, easyS32K1_SDK_comm.c and easyS32K1_SDK_boot.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\S32).

In case you use the easyDSP bootloader to program flash, define EZ_BOOTLOADER_USE as 1 in the easyS32K1_SDK.h file. In case you don't use the easyDSP bootloader for flash programming, define BOOTLOADER_USE as 0.



Please include easyS32K1_SDK.h in the main.c. And in the main(), call easyDSP_init() after the initialization of MCU.

In the easyDSP_init() function, all necessary setting for easyDSP monitoring are done.

In case you use easyDSP for flash programming, call easyDSP_boot() after setting of clock and pins.
 #include "easyS32K1_SDK.h"

```
int main (void)
{
    /* Initialize and configure clocks - see clock manager component for details */
    CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
                 g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
    CLOCK SYS UpdateConfiguration (OU, CLOCK MANAGER POLICY AGREEMENT);
    /* Initialize pins - See PinSettings component for more info */
    PINS_DRV_Init(NUM_OF_CONFIGURED_PINS0, g_pin_mux_InitConfigArr0);
#if EZ_BOOTLOADER_USE
    // Right after clock and pin setting, call easyDSP_boot() to enable flash programming
   easyDSP boot();
#endif
   // reset of initial setting
    .
   // call easyDSP init() to enable easyDSP monitoring
    easyDSP_init();
    // loop forever
    while(1)
    {
        .
        .
    }
}
```

STEP 5 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder of output file (for example, *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compiling time. Please refer to the setting of S32DS below.

easyDSP help



2. For easyDSP monitoring, the debug information should be included in the output file (for example, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in S32DS, no check in the check box in the linker option. Remove unused sections (-Xlinker --qc-sections)

STEP 6 : Limitation of easyDSP bootloader

1. To program flash, the bootloader should be provided since there is no ROM bootloader in this MCU. The bootloader easyDSP provides is the function (name : easyDSP_boot) and it resides in the user program. Therefore it can program flash only when it is already programmed in the flash. In case flash is empty or flash doesn't have easyDSP bootloader, you can't enter into the bootloader and will see the message below. In this case, you have to use the debugger to program flash. And in same principle, you have to use debugger to program easyDSP bootloader into flash at the beginning.

easyDSP		×
	Bootloader was not entered !	
	ОК]

2. easyDSP bootloader runs on RAM and it uses about 1.25kB RAM memory space (for -O1 optimization option).

7.3.2 S32K/S32M + RTD

It is assumed that the user uses S32 Configuration Tools and RTD (Real-Time Drivers).

STEP 1 : Hardware

Please select the UART channel and pins according to your board. No constraints to selectable channel and pin except LPUART1 is not usable for S32M. Then connect them to easyDSP like below. In case flash programming is not used, no need to connect /BOOT and /RESET pins.



Other considerations :

- In case there is a reset IC between easyDSP /RESET and MCU /RESET, it should transfer the signal within 0.5sec.

- TX and RX pin of easyDSP header is pulled up with 100k Ohm resistor inside of easyDSP pod.

STEP 2 : S32 Configuration Tools for easyDSP monitoring

As explained in STEP1, please select the UART channel and pins. And set the configuration tool (Pins tab) accordingly.

Also kindly set the identifier as 'EZ_TX' and 'EZ_RX' respectively for TX and RX pins.

In below example, PTA2 and PTA3 are chosen as RX and TX respectively with LPUARTO.

Also set the pin properties as shown in Rounting Details tab. Note that pull-up should be set.

Image: Section of the Content of th	E Pins 🖄	Periph	eral Signals	성 Power	Groups					🗖 🗖 📴 Package 🖂						ତ୍ ପ୍ ପ	- 🚍 🤹 🕤
Pn mm Lubit dentifie PORt FM ADC UPUAT TROMUX, IN2 1 P183 FM1, CHL ADC0, 255 IFINAUX, IN2 TROMUX, IN2 TROMUX, IN2 1 P183 FM1, CHL ADC0, 255 IFINAUX, IN2 TROMUX, IN2 TROMUX, IN2 1 P183 FM1, CHL ADC0, 255 IFINAUX, IN2	880	w 10	-0 O+ O	• \$ 2	ີ່ 🔎 type filt	ertext											
23 PR3 PR3 PR4 P	Pin Pin r	name L	abel lo	dentifier	PORT	FTM	ADC	LPUART	TRGMUX		8		014 114 014	20 20	8 7		
2 P1R2 P1R2 P1R1 P1R1 <t< td=""><td>23 PTB3</td><td>3</td><td></td><td></td><td>PTB3</td><td>FTM1 CH1[]</td><td>ADC0 SE7</td><td></td><td>TRGMUX IN</td><td>12</td><td>11</td><td>L L L</td><td>TA TA</td><td>12 H H</td><td>14</td><td></td><td></td></t<>	23 PTB3	3			PTB3	FTM1 CH1[]	ADC0 SE7		TRGMUX IN	12	11	L L L	TA TA	12 H H	14		
3 P181 P181 <t< td=""><td>24 PTB2</td><td>2</td><td></td><td></td><td>PTB2</td><td>FTM1 CH0[]</td><td>ADC0 SE6</td><td></td><td>TRGMUX IN</td><td>43</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></t<>	24 PTB2	2			PTB2	FTM1 CH0[]	ADC0 SE6		TRGMUX IN	43							
2 PTB0 PTB0 PTC0 PTC0 <t< td=""><td>25 PTB1</td><td>1</td><td></td><td></td><td>PTB1</td><td>TCLK0</td><td>ADC0 SE5</td><td>LPUARTO TX</td><td>-</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></t<>	25 PTB1	1			PTB1	TCLK0	ADC0 SE5	LPUARTO TX	-								
PTC9 PTC9 FTM [kT1 UPUARTI [kL] 29 PTC3 PTC3 FTM [kT1 UPUARTI [kL] 30 PTC3 PTR3 PTR3 IPUARTI [kL] IPUARTI [kL] 31 VD2 PTR3 PTR4 PTR4 PTR4<	26 PTBC	0			PTB0		ADC0 SE4	LPUARTO RX			PTD1				-	PTA2	
23 PTC3 PTC3 PTM LPUARTI_RT3 PTM	27 PTCS	9			PTC9	FTM1 FLT1	_	LPUART1 TX			PTDD	AD C0		CAND		PTA3	
29 P1A7 <	28 PTC8	8			PTC8	FTM1_FLT0		LPUART1_RX				CMP0		FLECIO			
9 95,40 Image: Status interrupt Configuration Image: Status interrupt	29 PTA7	7			PTA7	FTM0 FLT2	ADC0 SE3	LPUARTI RTS			PTES	FTM		FTMD		PTD2	
31 MOD_41 PTB13 PTB14 PTB14 PTB15 P	√ 30 VSS_	_40				_	_				PTE4	FTM1		JTAG		PTD3	
22 PB133 PB133 PB134 PT02 PT02 PT03 PT04 PT03 PT03 PT04 PT04 PT04 PT04 PT04 PT04 PT05 PT05 PT05 PT05 PT05 PT03 PT03 PT04 PT04 PT04 PT04 PT04 PT04 PT04 PT04 PT04 PT05 PT05 PT05 PT05 PT05 PT04	√ 31 VDD	2_41									ADD 7	LPI2C0		LPSPI0	_	PTR13	
33 PTD3 PTD3 VD0.4 VD0.	32 PTB1	13			PTB13	FTM0_CH1						LPSPII		LP IMPU	_	11010	
34 PTD2 <	33 PTD:	13			PTD3				TRGMUX_IN	14	VDDA	OSC		PORTA	_	VDD_41	
B13 E12, X E2, XX E2, XX FTA3 IEUARD TX FTA3 IEUARD TX FTA3 IEUARD TX	34 PTD2	12			PTD2				TRGMUX_IN	15 V	SS_10	PORTB		PORTC		√SS_40	
1/2 1	✓ 35 PTA	3 E	Z_TX E	Z_TX	PTA3			LPUARTO_TX			8787	PORTD		PORTE		BT 07	
37 Pital Pi	✓ 36 PTA2	2 E	Z_RX E	Z_RX	PTA2			LPUARTO_RX			107	Platform		PowerAnd Ground		110	
38 P100 P100 ACQ_2SC (PUARD_CTS TRGMUX_QUT3) 39 P107 P107 P100 P100 40 P106 P106 FTMI_QD_PHB IPUART_RX 41 P1013 P1014 P101 P101 P101 42 P1013 P1014 P101 P101 P101 43 P1011 P1011 P1010 P1010 P1010 44 P1013 P1011 P1011 P1010 P1010 45 P105 P105 P105 P105 P105 46 P105 P105 P105 P105 P105 47 P103 P105 P105 P105 P105 5 Pooling Details P105 P105 P105 P105 5 pooling Details P type filter text P105 P105 P105 7 P105 P105 Simal Arrow Routed pin/signal Label Idem Power group Direction Interrupt Status Filtig(SF) is disabled Unlocked Finabled PullUp Disable Arrow Routed pin/signal Label Idem Power group Direction Interrupt Status Filig(SF) is disabled Unlocked Finabled PullUp Disable Arrow Routed pin/signal Label Idem Power group Direction Interrupt Status Filig(SF) is disabled Unlocked Finabled PullUp Disabled rid n'rd n'rd n'rd n'rd <td>37 PTA1</td> <td>1</td> <td></td> <td></td> <td>PTA1</td> <td>FTM1_CH1[]</td> <td>ADC0_SE1</td> <td>LPUARTO_RTS</td> <td>5 TRGMUX_O</td> <td>UTO I</td> <td>PTB6</td> <td>RTC</td> <td></td> <td>SWD</td> <td></td> <td>PTC8</td> <td></td>	37 PTA1	1			PTA1	FTM1_CH1[]	ADC0_SE1	LPUARTO_RTS	5 TRGMUX_O	UTO I	PTB6	RTC		SWD		PTC8	
39 PIC7 PIC7 PIC7 FIM. (20, PHA LPUART, TX 44 PIC6 PIC6 PIC6 FIM. (20, PHA LPUART, TX 44 PIC6 PIC6 PIC6 FIM. (20, PHA LPUART, TX 44 PIC6 PIC6 PIC6 FIM. (20, PHA LPUART, TX 44 PIC13 PIC13 PIC14 PIC14 PIC14 45 PIC14 PIC10 PIL11 FIM. (204 PIC14 45 PIC3 PIC3 PIC4 FIM. (204 PIC3 46 PIC4 PIC4 PIC4 PIC4 PIC4 47 PID3 PIC3 PIC4 FIM. (204 PIC3 47 PID3 PIC3 PIC4 FIM. (204 PIC4 47 PID3 PIC3 PIC4 FIM. (204 PIC4 47 PID3 PIC3 PIC4 FIM. (204 PIC4 50 pickts Prove Filter test Prove Filter test Prove Filter test Sin. Arrow Routed pin/signal Label Iden Power group Direction Interrupt Status Filog (SF) is disabled Unlocked Full Enable Pull Select Digital Filter Drive Strength Pasive Filter Initial Value 5 Dicutarity and Distal	38 PTAC	0			PTA0		ADC0_SE0	LPUART0_CT	S TRGMUX_O	IUT3	PTD16	TRGMUX				PTC9	
40 PTC6 PTC6 FTM (CD, PHB LPUARTI_RX 41 PTA13 PTA12 FTM (CH) PTB \$32K118_LQFP48 - LQFP 48 package PTB1 42 PTA12 PTA12 FTM (CH) PTB \$32K118_LQFP48 - LQFP 48 package PTB1 44 PTA10 PTA11 PTM (CH) PTB \$32K118_LQFP48 - LQFP 48 package PTB1 45 PTC5 PTC5 PTC5 PTC5 PTC5 PTC5 PTC5 47 PTA5 PTC4 FTM (CH) PTA1 PTA14 PTA14<	39 PTC7	7			PTC7	FTM1_QD_PHA		LPUART1_TX									
41 PTA13 PTA13 PTA13 FTM1_CH7 42 PTA12 PTA13 FTM1_CH7 PTA14 P	40 PTC6	6			PTC6	FTM1_QD_PHB		LPUART1_RX			FIDIS					FIBU	
12 P1A12 P1A12 F1M1_CH6 43 P1A11 P1A11 P1A11 P1A11 44 P1A10 P1A10 F1M1_CH4 Image: Control Charles and	41 PTA1	13			PTA13	FTM1_CH7					PTEP	S32K118_I	_QFP48 - L	.QFP 48 pa	ckage	PTB1	
44 P1A10 P1A11 P1A11 F1M1_CH5 44 P1A10 P1A10 F1M1_CH4 P1A10 45 P1C5 P1C5 P1C5 46 P1C4 P1A10 P1A10 P1A10 47 P1A3 P1A3 TCLK1 P1A10 47 P1A3 P1A3 TCLK1 P1A10 58 posting Details P1A10 P1A10 P1A10 70 Poptalition Paint 2 0 0 0 8 P1A10 Arrow Routed pin/signal Label Idem Power group Direction Interrupt Status Fing (SF) is disabled Unlocked Finabled Pull Up Disabled Nr/a N/a n/a 8 DPLANT nd C. Signatz Painter Painter Status Fing (SF) is disabled Unlocked Finabled Pull Up Disabled Nr/a N/a N/a	42 PTA1	12			PTA12	FTM1_CH6											
44 P120 P1410 P1410 F1M1_CH4 Image: State of the state o	43 PTA1	11			PTA11	FTM1_CH5											
45 PTC3 PTC3 PTC3 PTC4	44 PTA1	10			PTA10	FTM1_CH4					8	S 14 13	102	015	182		
46 PTC4 PTC4 FTML (PLK) PTC4 FTML (PLK) PTC4 FTML (PLK) PTC4 FTML (PLK) PTC4 PTC4 <td>45 PTC5</td> <td>5</td> <td></td> <td></td> <td>PTC5</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>a.</td> <td></td> <td>م م م</td> <td>14 14 14</td> <td>a. a.</td> <td></td> <td></td>	45 PTC5	5			PTC5						a.		م م م	14 14 14	a. a.		
If p PIAS PIAS TCLK1 In provid movid	46 PTC4	4			PTC4	FTM1_CH0[]											
	47 PTA5	5			PTA5	TCLK1											
Routing Details of BOARD Intel® 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	40 DTA /	4			DTA 4	_											
Genesis Routing Details For Signals P type filter text For BOARD_InterPrint & 2 © Routing Details for BOARD_InterPrint & 2 OF Signals Pull Enable Pull Enable Pull Select Digital Filter Drive Strength Passive Filter Initial Value # Peripheral Si Arrow Routed pin/Signal Lock Register Pull Enable Pull Select Digital Filter Drive Strength Passive Filter Initial Value 36 LPUARD nd <																	
Protecting Details for 80ARD_IntError 2 C A Power group Direction Interrupt Status Interrupt Status Interrupt Status Excelling (SF) is disabled Pull Enable Pull Select Digital Filter Drive Strength Passive Filter Initial Value 3 IPUARTI rd [36] PTA2 EZ_RX Input Dan't modify Interrupt Status Flag (ISF) is disabled Unlocked Enabled Pull Up Disabled n/a n/a n/a	Routing D	Details															= -
Peripheral Sin. Arrow Routed pin/signal Label Iden Power group Direction Interrupt Status Interrupt	Pins Sign	nals 🔎 ty	pe filter text														
# Peripheral St Arrow Routed pin/signal Label Iden Power group Direction Interrupt Configuration Lock Register Pull Enable Pull Select Digital Filter Drive Strength Passive Filter Initial Value 36 LPUARTID rxd - [36] PTA2 EZ_RX Input Don't modify Interrupt Status Flag (ISF) is disabled Unlocked Enabled Pull Up Disabled n/a n/a n/a	Routing De	etails for B	DARD_InitPin	s 2	$\mathbf{ightarrow}$												
36 LPUARTO r.d - [36] PTA2 EZ_RX EZ_RX Input Don't modify Interrupt Status Flag (ISF) is disabled Unlocked Enabled Pull Up Disabled n/a n/a n/a	# Pe	eripheral	Si Arro	w Route	d pin/signal	Label Iden	Power group	Direction	nterrupt Status	Interrupt Configuration	Lock Register	Pull Enable	Pull Select	Digital Filter	Drive Strength	Passive Filter	Initial Value
	36 LP	PUARTO	nxd <-	[36] P	TA2	FZ RX FZ RX	5 1	Innut	Don't modify	Interrupt Status Flag (ISF) is disabled	Unlocked	Enabled	Pull Un	Disabled	n/a	n/a	n/a
35 LPUARTO txd -> [35] PTA3 FZ TX FZ TX Output Don't modify Internant Status Flag (ISE) is disabled Unlocked Enabled Pull Un Disabled n/a n/a n/a	35 LP	PUARTO	txd ->	[35] P	TAB	FZ TX FZ TX		Output	Don't modify	Interrupt Status Flag (ISF) is disabled	Unlocked	Enabled	Pull Un	Disabled	n/a	n/a	n/a

And add the Lpuart_Uart and IntCtrl_Ip module in the Drivers. If they exist, no need to add again.

🖥 Components 🔀 🦞	Peripherals	- 8	Select configuration co	omponent		
type filter text		•	Select which components	should be offered All		
Ν	MCAL	0	type filter text			
-			Configuration componen	t Component description	Category	Required S 4
D	rivers	U	💧 Gpio_Dio	Gpio_Dio Configuration	Drivers	platform.d
	De et les		IntCtrl_lp	IP configuration	Drivers	platform.d
OSIT_1	Port_ip		💧 Lpi2c	Lpi2c configuration	Drivers	platform.d
			LPit_Gpt	LPIT_GPT IPL Configuration	Drivers	platform.d
			💧 Lpit_lcu	LPIT Driver	Drivers	platform.d
			💧 Lpspi	Lpspi Configuration	Drivers	platform.d
			Lptmr_Gpt	LPTMR_GPT IPL Configuration	Drivers	platform.d
			Lptmr_lcu	LPTMR Driver	Drivers	platform.d
			Lpuart_Lin	Lpuart Lin Configuration	Drivers	platform.d
			Lpuart_Uart	Lpuart Uart Configuration	Drivers	platform.d
			A MPU	Memory Protection In Driver	Drivers	platform.d

And set the Lpuart_Uart module properties.

Please enable 'Uart Callback Capability' in the tab 'GeneralConfiguration' and set the name of callback as 'ez_RxCallBack'.

Also set the various properties in the tab 'UartGlobalConfig'. In this example, LPUARTO is selected as STEP1 and 2. The baudrate should be same one to one in the easyDSP project setting.

🔁 Lpuart_Uart 🔀									
Lpuart Uart Configuration [Drivers]									
Name Lpuart_Uart									
Mode LPUART UART Mode									
Name ConfigTimeSupport GeneralCon	nfiguration	UartGlobalConfig							
Name	GeneralConfiguration								
Uart Development Error Detection	\leq								
Uart Timeout Method	OSIF_COU	NTER_DUMMY							
Uart Timeout Duration	1000000								
Uart DMA Enable	\Box								
Uart Callback Capability									
✓ UartCallback									
0 ez_RxCallBack									
+									
✓ Parameter for Uart Callback									
Add item by clicking on plus button									

Ĝ Lpuart_Uart_1 ⊠									
Lpuart Uart Con	figuration [Drivers]								
Name Louart Uart 1									
	ada.								
Mode LPOART OART MC									
Name ConfigTimeSupp	ort GeneralConfiguration UartGlob	alConfig							
Name UartGlobal	Config								
VartChannel	$+ \times \land \vee$								
ez_UartChannel	Name	ez_UartChannel							
	UartHwUsing	LPUART_IP							
	UartClockFunctionalGroupRef	BOARD_BootClockRUN							
	 DetailModuleConfiguration 								
	Name	DetailModuleConfiguration							
	Uart hardware channel	LPUART_0							
	Desire Baudrate	LPUART_UART_BAUDRATE_115200							
	Uart Asynchronous Method	LPUART_UART_IP_USING_INTERRUPTS							
	✓ Tx DMA channel								
	+								
	✓ Rx DMA channel								
	÷								
	Uart Parity Type LPUART_UART_IP_PARITY_DISABLED								
	Uart Stop Bit Number	LPUART_UART_IP_ONE_STOP_BIT							
	Uart Word Length	LPUART_UART_IP_8_BITS_PER_CHAR							
	Uart Internal Loopback Mode Ena	ble							

And set the IntCtrl_Ip module properties. In the tab 'Interrupt Controller', please enable the interrupt of target LPUART channel and set its priority lowest (highest value). In the tab 'Generic Interrupt Setting', set its interrupt handler as 'EZ_LPUART_UART_IP_IRQHandler'. For some MCU, the setting of these tabs are combined to single tab.

ក្និ IntCtrl_lp_1 🛛					
Name ConfigTimeSu	upport Genera	I Configuration Interrupt C	ontroller Generic Interrupt Set	ttings	
+ × ^	×				
0	Name	IntCtrlConfig_0			
	V Platfe	ormIsrConfig			
	#	Name	Interrupt Name	Interrupt Enabled	Priority
	0	PlatformlsrConfig 0	DMA0 IRQn		0
	1	PlatformlsrConfig 1	DMA1 IROn		0
	2	PlatformlsrConfig 2	DMA2 IROn		0
	3	PlatformlsrConfig 3	DMA3 IRQn		0
	4	PlatformlsrConfig 4	DMA Error IROn		0
	5	PlatformlsrConfig 5	ERM IROn		0
	6	PlatformlsrConfig 6	RTC IROn		0
	7	PlatformlsrConfig 7	RTC Seconds IROn		0
	8	PlatformlsrConfig 8	LPTMR0 IROn		0
	9	PlatformlsrConfig 9	PORT IROn		0
	10	PlatformlsrConfig 10	CANO ORed IROn		0
	11	PlatformlsrConfig 11	CANO ORed 0 31 MB IROn		0
	12	PlatformlsrConfig 12	FTM0 Ch0 Ch7 IROn		0
	13	PlatformlsrConfig 13	FTM0 Fault IROn		0
	14	PlatformlsrConfig 14	FTM0 Ovf Reload IROn		0
	15	PlatformlsrConfig 15	FTM1 Ch0 Ch7 IROn		0
	16	PlatformlsrConfig 16	FTM1 Fault IROn		0
	17	PlatformlsrConfig 17	FTM1 Ovf Reload IRQn		0
	18	PlatformlsrConfig 18	FTFC IROn		0
	19	PlatformlsrConfig 19	PDB0 IROn		0
	20	PlatformlsrConfig 20	LPIT IRQn		0
	21	PlatformlsrConfig 21	PMC SCG CMU IROn		0
	22	PlatformlsrConfig 22	WDOG IRQn		0
	23	PlatformlsrConfig 23	RCM IROn		0
	24	PlatformlsrConfig 24	LPI2C0 Master Slave IRQn		0
	25	PlatformlsrConfig 25	FLEXIO IROn		0
	26	PlatformlsrConfig 26	LPSPI0_IRQn		0
	27	PlatformlsrConfig 27	LPSPI1 IRQn		0
	28	PlatformlsrConfig 28	ADC0 IRQn		0
	29	PlatformlsrConfig 29	CMP0_IRQn		0
	30	PlatformlsrConfig 30	LPUART1 RxTx IRQn		0
	31	PlatformlsrConfig 31	LPUARTO RxTx IROn		3

Int	Ctrl_lp_1	122		
lode	IP Mo	ode		
Vam	e Conf	igTimeSupport General Cor	nfiguration Interrupt Controlle	er Generic Interrupt Settings
Nar	me	intRouteConfig		
	Platf	ormisrConfig		
	Fiatio	onnisiconing		
	#	Name	Interrupt Name	Handler
	0	PlatformlsrConfig_0	DMA0_IRQn	undefined_handler
	1	PlatformlsrConfig_1	DMA1_IRQn	undefined_handler
	2	PlatformlsrConfig_2	DMA2_IRQn	undefined_handler
	3	PlatformlsrConfig_3	DMA3_IRQn	undefined_handler
	4	PlatformlsrConfig_4	DMA_Error_IRQn	undefined_handler
	5	PlatformlsrConfig_5	ERM_IRQn	undefined_handler
	6	PlatformlsrConfig_6	RTC_IRQn	undefined_handler
	7	PlatformlsrConfig_7	RTC_Seconds_IRQn	undefined_handler
	8	PlatformlsrConfig_8	LPTMR0_IRQn	undefined_handler
	9	PlatformlsrConfig_9	PORT_IRQn	undefined_handler
	10	PlatformlsrConfig_10	CAN0_ORed_IRQn	undefined_handler
	11	PlatformlsrConfig_11	CAN0_ORed_0_31_MB_IRQn	undefined_handler
	12	PlatformlsrConfig_12	FTM0_Ch0_Ch7_IRQn	undefined_handler
	13	PlatformlsrConfig_13	FTM0_Fault_IRQn	undefined_handler
	14	PlatformlsrConfig_14	FTM0_Ovf_Reload_IRQn	undefined_handler
	15	PlatformlsrConfig_15	FTM1_Ch0_Ch7_IRQn	undefined_handler
	16	PlatformlsrConfig_16	FTM1_Fault_IRQn	undefined_handler
	17	PlatformlsrConfig_17	FTM1_Ovf_Reload_IRQn	undefined_handler
	18	PlatformlsrConfig_18	FTFC_IRQn	undefined_handler
	19	PlatformlsrConfig_19	PDB0_IRQn	undefined_handler
	20	PlatformlsrConfig_20	LPIT_IRQn	undefined_handler
	21	PlatformlsrConfig_21	PMC_SCG_CMU_IRQn	undefined_handler
	22	PlatformlsrConfig_22	WDOG_IRQn	undefined_handler
	23	PlatformlsrConfig_23	RCM_IRQn	undefined_handler
	24	PlatformlsrConfig_24	LPI2C0_Master_Slave_IRQn	undefined_handler
	25	PlatformlsrConfig_25	FLEXIO_IRQn	undefined_handler
	26	PlatformlsrConfig_26	LPSPI0_IRQn	undefined_handler
	27	PlatformlsrConfig_27	LPSPI1_IRQn	undefined_handler
	28	PlatformlsrConfig_28	ADC0_IRQn	undefined_handler
	29	PlatformlsrConfig_29	CMP0_IRQn	undefined_handler
	30	PlatformlsrConfig_30	LPUART1_RxTx_IRQn	undefined_handler
	31	PlatformlsrConfig_31	LPUART0_RxTx_IRQn	EZ_LPUART_UART_IP_IRQHand

Also make sure the clock to the UART channel is set properly and enabled. Please refer to below example.

🕽 • Clocks Diagram 🐹 📑 Clocks Table 📃 🗖	🔺 Overview 💿 Periphe	ral Clock View 🔀 🛛	Code Pr	eview 🔢 Registers 🔚 Details \land Clock Consumer:	;		- 1
	Clock Name	Enable	Control	Source	Divider	DivType	Frequency
Run Mode RUN Clock Development Error Detect Disabled	EIM0_CLK	\checkmark		CORE_CLK	/1		48 MHz
Clock User Mode Support Dirabled	ERM0_CLK	\checkmark		CORE_CLK	/1		48 MHz
	FLEXCAN0_CLK	\checkmark		CORE_CLK	/1		48 MHz
Clock Timeout Method OSIF_COUNTER_DUMMY Get Clock Frequency API	FTFC0_CLK	\checkmark		SLOW_CLK	/1		24 MHz
Disabled Enable Cmu Notification Disabled CmuNotification	FTM0_CLK	\checkmark		SIRCDIV1_CLK	/1		8 MHz
NULL PTR	FTM1_CLK	\checkmark		SIRCDIV1_CLK	/1		8 MHz
	FlexIO0_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
SIRCDIV2_CLK	GPIO0_CLK	\checkmark		CORE_CLK	/1		48 MHz
SOSCDIV2_CLK	LPI2C0_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
\sim	LPIT0_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
	LPO_1K_CLK	\checkmark		LPO_32K_CLK	/ 32		1 kHz
	LPO_32K_CLK	\checkmark		LPO_128K_CLK	/4		32 kHz
12 FIRCDIV2_CLK	LPO_CLK	\checkmark		The Low Power Oscillator (LPO) 128 KHz RC oscillator	/1		128 kHz
SIRCDIV2_CLK 71 * 8 MHz	LPSPI0_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
SOSCDIV2_CLK	LPSPI1_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
	LPUART0_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
	LPUART1_CLK	\checkmark		SIRCDIV2_CLK	/1		8 MHz
FIRCDIV2_CL	MPU0_CLK	\checkmark		CORE_CLK	/1		48 MHz
Iz SIRCDIV2_CLK	MSCM0_CLK	\checkmark		CORE_CLK	/1		48 MHz
SOSCDIV2_CLK	PDB0_CLK	\checkmark		CORE_CLK	/1		48 MHz
V	DODTA CLV			DIIC CLV	71		40 MALI-

STEP 3-1 : S32 Configuration Tools for easyDSP boot loader of S32K1x

So far until STEP 2, the setting is for the communication with easyDSP for monitoring variables. If you like to use the easyDSP bootloader for flash programming of S32K1x, the following process is also required because easyDSP bootloader uses flash driver. Please add Ftfc_Ip and Gpio_Dio component in the Drivers.

🖢 Components 🛛		- 8	Select configuration con	nponent		
type filter text			Select which components sh	nould be offered All		
	MCAL	0	type filter text			
			Configuration component	Component description	Category	Required S
	Drivers	Ð	Flexio_Uart	Flexio Uart Configuration	Drivers	platform.d
latCarl in 1	Louised Hand 1	- 26 - 1	A Ftfc_Eep_lp	FTFC EEP Configuration	Drivers	platform.d
interr_ip_1	Lpuan_Dan_1	OSII_1	A Ftfc_lp	Ftfc_lp Configuration	Drivers	platform.d
	Port In 1		Æ Ftm_Gpt	FTM_GPT IPL Configuration	Drivers	platform.d
	Port_ip_1		A Ftm_lcu	FTM Driver	Drivers	platform.d
			Æ Ftm_Mcl_lp	FTM	Drivers	platform.d
			A Ftm_Ocu	FTM OCU Configuration	Drivers	platform.d
			A Ftm_Pwm	Ftm Pwm driver	Drivers	platform.d
			A Ftm_Qdec_lp	FTM Qdec Configuration	Drivers	platform.d
			A Gpio_Dio	Gpio_Dio Configuration	Drivers	platform.d
			i IntCtrl In	IP configuration	Drivers	platform.d

For Gpio_Dio component, the default setting is ok.

For Ftfc_Ip component, please disable 'Fls Timeout Supervision Enabled' button.

ලී Ftfc_lp 🔀								
Ftfc_Ip Configuration [Drivers]								
Name Ftfc_lp								
Mode Non-Autosar Mode								
~	v							
Name FIsConfigSet FIsGeneral AutosarExt FI	sPublishedInformation							
Name	FtfcGeneral							
Enable development error check at IP level								
Fls ECC Handling HardfaultHandler								
Fls ECC Handling ProtectionHook	\Box							
Fls Erase Verification Enabled	\Box							
Fls Write Verification Enabled	\Box							
Fls Timeout Supervision Enabled	<u> </u>							
FIs Timeout Method	OSIF_COUNTER_DUMMY							
Fls Async Write Timeout	2147483647							
Fls Async Erase Timeout	2147483647							
FIs Sync Write Timeout	2147483647							
FIs Sync Erase Timeout	2147483647							
Fls Async Abort Timeout	32767							

ਿਊ Ftfc_lp 🔀

Ftfc_Ip Configuration [Drivers]								
Name	Ftfc_lp							
Mode	Non-Autosar	Mode						
~								
Name	FlsConfigSet	FlsGeneral	AutosarExt	FlsPublishedInformation				
Name	e			AutosarExt				
Fls En	able User Mod	le Support	(
Fls Sy	nchronize Cao	:he	(
Fls In	valid Prefetch	Buffer From	RAM (2				

STEP 3-2 : S32 Configuration Tools for easyDSP boot loader of S32K3x

If you like to use the easyDSP bootloader for flash programming of S32K3x, please add C40_Ip and Siu2_Dio components in the Drivers.

accomponents 💼	×	
type filter text		€↑
	MCAL	0
	Drivers	0
C40_lp	IntCtrl_lp	Lpuart_Uart
osif_1	Siul2_Dio_1	Siul2_Port_1

For Siu2_Dio component, the default setting is ok.

For C40_Ip c	omponent,	please d	lisable 'F	ls Timeout	Supervision	Enabled'	button.
C40 In ∞					·		

_g C40_lp 🐹								
C40_Ip Configuration [Drivers]								
Name C40_Ip								
Mode Non-Autosar Mode								
~								
Name FlsConfigSet FlsGeneral AutosarExt	FlsPublishedInformation							
Name	C40General							
Enable development error check at IP level								
Fls ECC Handling HardfaultHandler								
Fls ECC Handling ProtectionHook								
Fls Erase Verification Enabled	\Box							
Fls Write Verification Enabled								
Fls Timeout Supervision Enabled								
Fls Timeout Method	OSIF_COUNTER_DUMMY							
FIs Async Write Timeout	2147483647							
Fls Async Erase Timeout	2147483647							
Fls Sync Write Timeout	2147483647							
FIs Sync Erase Timeout	2147483647							
Fls Async Abort Timeout	32767							

<u>ද</u> C40_	lp 🛛						
C40_Ip Configuration [Drivers]							
Name	C40_lp						
Mode	Mode Non-Autosar Mode						
~	v						
Name	FlsConfigSet	FlsGeneral	AutosarExt	FlsPublishedInformation			
Nam	e		Autosar	Ext			
Fls Er	nable User Mod	le Support					
Fls Sy	nchronize Cac	he					
FIs Da	ata Error Suppr	ession					
FIs BI	ock 4 Pipe Sele	ct	FLS_CO	MMAND_PIPE_0			

STEP 3-3 : S32 Configuration Tools for easyDSP boot loader of S32M24x

If you like to use the easyDSP bootloader for flash programming of S32M24x, please add Ftfc_Mem_InFls_Ip and Gpio_Dio component in the Drivers.

🍓 Components 🗙	🖞 Periphera	ls 🗖 🗖	🔀 Add Configuration Com	ponent Instance	
type filter text			Select, which components s	hould be offered All	
	MCAL	0	type filter text		
	Drivers	0	Configuration component & Flexio_Spi	Component description Spi Configuration	Category Drivers
BaseNXP	Gpio_Dio	IntCtrl_lp	 Flexio_Uart Ftfc_Mem_Eep_Ip 	Flexio Uart Configuration FTFC EEP Configuration	Drivers Drivers
Lpuart_U	Uart	Port_lp_1	Etfc_Mem_InFls_Ip Etfc_Mem_InFls_Ip Etfc_Mem_InFls_Ip	Ftfc_Mem_InFls_Ip Driver FTM_GPT IPL Configuration	Drivers Drivers
			 Ftm_lcu Ftm_Mcl_lp 	FTM Driver FTM	Drivers Drivers
			 Etm_Ocu Etm_Pwm 	FTM OCU Configuration Ftm Pwm driver	Drivers Drivers
			A Ftm_Qdec_lp i Gpio Dio	FTM Qdec Configuration Gpio Dio Configuration	Drivers Drivers
			A Hvm	HVM Configuration	Drivers
			1 IntCtrl_Ip IPV_Mpu_Ip	IP configuration IP configuration	Drivers
			▲ Lpi2c	Lpi2c configuration	Drivers
				ОК	Cancel

For Gpio_Dio component, the default setting is ok.

For Ftfc_Mem_InFls_Ip component, the default setting is ok. Please note that you have to disable 'Mem Timeout Supervision Enabled' button in 'MemGeneral' tab and 'Mem Synchronize Cache' button in 'MemAutosarExt' tab.

STEP 3-4 : S32 Configuration Tools for easyDSP boot loader of S32M27x

If you like to use the easyDSP bootloader for flash programming of S32M27x, please add C40_Ip and Siu2_Dio components in the Drivers.

🍋 Components 🗙	:	
type filter text		
	MCAL	0
	Drivers	0
BaseNXP	C40_lp	Cache_lp
IntCtrl_lp	Lpuart_Uart	Siul2_Dio
	Siul2_Port	

For Siu2_Dio component, the default setting is ok.

For C40_Ip component, the default setting is ok. Please note that you have to disable 'Mem Timeout Supervision Enabled' button in 'MemGeneral' tab and 'Mem Synchronize Cache' button in 'MemAutosarExt' tab.

STEP 4-1 : Source code correction for easyDSP bootloader of S32K1x

From STEP 3-1, the relavant codes are generated and you can find Ftfc_Fls_Ip.h and Ftfc_Fls_Ip.c files in the folder RTD>include and RTD>src respectively.

easyDSP bootloader uses these flash API functions and they should run on the ram, not on the flash. To make these functions run on the ram :

First, in the file Ftfc_Fls_Ip.h, find the location of function declaration, and change like below red boxes.



Second, in the file Ftfc_Fls_Ip.c, find the location of static function declaration, and change like below red boxes.



Third, again in the file Ftfc_Fls_Ip.c, find the location of Ftfc_Fls_Ip_SectorErase function definition, and disable Ftfc_Fls_Ip_SectorErasePreCheck function.

```
Ftfc_Fls_Ip_StatusType Ftfc_Fls_Ip_SectorErase(uint32 u32SectorStartAddress)
{
    Ftfc_Fls_Ip_StatusType eRetVal;
    boolean bAddressValid = FTFC_ADDRESS_VALID(u32SectorStartAddress);
    boolean bSectorAligned = FTFC_SECTOR_ALIGNED(u32SectorStartAddress);
    DEV_ASSERT_FTFC (bAddressValid) ;
    DEV_ASSERT_FTFC (bSectorAligned) ;
    /* Unused variables */
    (void) bAddressValid;
    (void) bSectorAligned;
    /* Check(if erase suspended is possible) if any ongoing erase suspended and abort it ^{\prime}
    eRetVal = Ftfc_Fls_Flash_AbortSuspended();
    if (STATUS_FTFC_FLS_IP_SUCCESS == eRetVal)
    -{
        /* Pre-check before starting erase operation */
        //eRetVal = Ftfc Fls Ip SectorErasePreCheck(u32SectorStartAddress); // commented by easyDSP
    }
```

In case the Configuration Tool detects the correction of this file and ask to revert it, don't revert it.

STEP 4-2 : Source code correction for easyDSP bootloader of S32K3x

From STEP 3-2, the relavant codes are generated and you can find C40_Ip.h and C40_Ip.c files in the folder RTD>include and RTD>src respectively.

easyDSP bootloader uses these flash API functions and they should run on the ram, not on the flash. To make these functions run on the ram :

First, in the file C40_Ip.h, find the location of function declaration, and change like below red boxes.



Second, in the file C40_Ip.c, find the location of static function declaration, and change like below red boxes.

In case the Configuration Tool detects the correction of this file and ask to revert it, don't revert it.

STEP 4-3 : Source code correction for easyDSP bootloader of S32M24x

From STEP 3-1, the relavant codes are generated and you can find Ftfc_Fls_Ip.h and Ftfc_Fls_Ip.c files in the folder RTD>include and RTD>src respectively.

easyDSP bootloader uses these flash API functions and they should run on the ram, not on the flash. To make these functions run on the ram :

First, in the file Ftfc_Fls_Ip.h, find the location of function declaration, and change like below red boxes.

STEP 4-4 : Source code correction for easyDSP bootloader of S32M27x

From STEP 3-2, the relavant codes are generated and you can find C40_Ip.h and C40_Ip.c files in the folder RTD>include and RTD>src respectively.

easyDSP bootloader uses these flash API functions and they should run on the ram, not on the flash. To make these functions run on the ram :

First, in the file C40_Ip.h, find the location of function declaration, and change like below red boxes.

STEP 5 : Calling easyDSP functions

Three files are provided for easyDSP communication and flash programming (easyS32_RTD.h, easyS32_RTD_comm.c, easyS32_RTD_boot.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\S32).

In the file of easyS32_RTD.h, you should set some macros. First, the target LPUART channel for easyDSP. In this example below, it is set as LPUART0. Second, in case you use the easyDSP bootloader to program flash, define EZ_BOOTLOADER_USE as 1.

Please include easyS32_RTD.h in the main.c. And in the main(), call easyDSP_init() after the initialization of MCU. In the easyDSP_init() function, all necessary setting for easyDSP monitoring are done.

Note that the clock, pins and interrupt should be set properly for easyDSP monitoring. In case you use easyDSP for flash programming, call easyDSP_boot() right after setting of clock and pins.

```
#include "easyS32_RTD.h"
int main(void)
{
    // Init clock
   Clock_Ip_Init(&Clock_Ip_aClockConfig[0]);
#if defined (FEATURE_CLOCK_IP_HAS_SPLL_CLK)
    // Busy wait until the System PLL is locked
   while (CLOCK IP PLL LOCKED != Clock Ip GetPllStatus());
   Clock Ip DistributePll();
#endif
    // Initialize all pins in case of S32K1
   Port_Ci_Port_Ip_Init(NUM_OF_CONFIGURED_PINS0, g_pin_mux_InitConfigArr0);
    // Initialize all pins in case of S32K3
   Siul2_Port_Ip_Init(NUM_OF_CONFIGURED_PINS0, g_pin_mux_InitConfigArr0);
#if EZ_BOOTLOADER_USE
    // Right after clock and pin setting, call easyDSP boot() to enable flash programming
   easyDSP_boot();
#endif
    // Initialize IRQs
    IntCtrl_Ip_Init(&IntCtrlConfig_0);
   IntCtrl Ip ConfigIrqRouting (&intRouteConfig);
   // reset of initial setting
    // call easyDSP_init() to enable easyDSP monitoring
   easyDSP init();
    // loop forever
   while(1)
    {
        .
        .
    }
}
```

STEP 6 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder of output file (for example, *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compiling time. Please refer to the setting of S32DS below.

easyDSP help



2. For easyDSP monitoring, the debug information should be included in the output file (for example, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in S32DS, no check in the check box in the linker option. Remove unused sections (-Xlinker --qc-sections)

STEP 7 : Limitation of easyDSP bootloader

1. To program flash, the bootloader should be provided since there is no ROM bootloader in this MCU. The bootloader easyDSP provides is the function (name : easyDSP_boot) and it resides in the user program. Therefore it can program flash only when it is already programmed in the flash. In case flash is empty or flash doesn't have easyDSP bootloader, you can't enter into the bootloader and will see the message below. In this case, you have to use the debugger to program flash. And in same principle, you have to use debugger to program easyDSP bootloader into flash at the beginning.

easyDSP	X
Bootloader was not entered !	
ОК	

2. easyDSP bootloader runs on RAM and it uses some RAM memory space. It is about 2.4K bytes for S32K1, 4.8K bytes for S32K3 at the optimization option -O1.

7.4 AM263x

7.4.1 AM263x software

STEP 1 : Core selection

MCU cores are classified with 4 types in terms of easyDSP.

Yellow core : core that easyDSP pod is connected to and easyDSP communicates with Orange core : core that easyDSP pod is not connected to but easyDSP communicates with Blue core : core that easyDSP doesn't communicate with

Gray core : core that doesn't run

	Connected to easyDSP pod	Communicated with easyDSP	Running core
core	Yes	Yes	Yes
core	No	Yes	Yes
core	No	No	Yes
core	No	No	No

AM263x has max 4 cores. Please choose core type either yellow or orange core based on your application. Any core of AM263x could be yellow or orange core.

Since blue and gray core has no operation with easyDSP, no easyDSP related setting is required for them.

Together with data cache usage, several cases are available as below.

Case 1 :

It is the case that easyDSP monitors multi cores (core a and b) and at least one of them uses data cache and IPC RPMessage is usable for core to core communication.

easyDSP pod is connected to core a via UARTO, the variable of core a is accessed by core a. To avoid cache coherence issue, the variable (actually its memory location) of core b is accessed by core b via core to core communication by IPC RPMessage. Please refer to the arrow for data flow

between easyDSP and cores.



Case 2 :

In the case 1 but IPC RPMessage is not usable, easyDSP pod should be connected to each core.



Case 3 :

It is the case that easyDSP monitors multi cores and data cache is disabled in these cores. All the variables (and their memory location) are accessed by the core easyDSP pod is connected to.



Case 4 :

It is the case that easyDSP monitors single core. In this case, we don't care whether the data cache is enabled or not.



STEP 2 : SysConfig setting

easyDSP uses the code generated by SysConfig. Below figures are made based on SysConfig 1.13.0.

Since easyDSP communicates with MCU via UART0, please disable 'Debug Log > Enable UART Log' or use another UART than UART0 for it.

▼ TI DRIVER PORTIN	G LAYER (Debug Log @	
Clock	1/1 🔮 🕀	Debug Log ()	
Debug Log	1/1 🔮 🕀	Enable Error Log Zone	\checkmark
MPU ARMv7	7/16 🕑 🕀	Enable Warning Log Zone	\checkmark
RAT	\oplus	Enable Info Log Zone	
TIMER	\oplus	Enable CCS Log	
 TI DRIVERS (23) 		Enable 000 Ebg	
ADC	\oplus	Enable Memory Log	
BOOTLOADER	\oplus	Enable UART Log	
CMPSS	\oplus	Enable Shared Memory Log Writer	\checkmark
DAC	\oplus	Enable Shared Memory Log Reader	\checkmark
ECAD		Little control of Log reader	

UART related setting is required for all the cores easyDSP pod is connected to, that is, yellow cores. The name of UART module should be 'EZDSP_UART'. The baudrate is selectable but it should be same to that of easyDSP project setting. The data format should be 8bit data, one stop bit and no parity bit. The priority of UART interrupt should be as low as possible such as 15. TX and RX pins are that of UART0 MUXMODE 0. Exceptionally, UART of core f in STEP1 could be other UART than UART0. Please check below for details.

▼ TI DRIVER PORTING LAYE	R (DPL) (5)						
Clock	1/1 🥑 🕀	Global Parameters Settings th	hat affect all instances				^
Debug Log	1/1 🥑 🕀				_		_
MPU ARMv7	4/16 🕑 🕀	UART (1 Added)		⊕ ADD	T F F	REMOV	E ALL
RAT	\oplus	Lawrence			_		-
TIMER	\oplus	EZDSP_UART					
 TI DRIVERS (23) 		Name	EZDSP UART				
ADC	\oplus	Operational Mode	16v				*
BOOTLOADER	\oplus		100				· ·
CMPSS	\oplus	Baudrate	115200				
DAC	(\pm)	Clock Freq	4800000				
ECAP	()	Data Length	8-bit				Ŧ
EDMA	1/1 🔮 🕀	Stop Bit	1-bit				-
EPWM	Ð	Parity Type	None				-
EQEP	(†)	Enable Hardware Flow Control					
	Ð	Transfer Made					_
GPIO	1	Transfer Mode	Interrupt Mode				*
120		Interrupt Priority	15				
IPC	(U) (H)	RX Trigger Level	1				Ŧ
LIN	÷.	TX Trigger Level	1				-
MCAN	÷	Read Transfer Mode	Blocking				-
MCSPI	÷	Read Transfer Callback	NULL				
MPU FIREWALL	\oplus	Write Transfer Mode	Blocking				-
PRU (ICSS)	\oplus	Write Transfer Callback	NULL				
QSPI	1/1 🕑 🕀	David Datum Made	Full				_
RTI	\oplus	Read Return Mode	Full				0
SDFM	\oplus	UART Instance	UARTO			-	_ 🖸
UART	1 🔮 🕀	Signals 1	Pins	Pull Up/I	Down	Slew F	Rate
WDT	(\pm)			Pull Up	.	High	· ·
 TI BOARD DRIVERS (4) 	<u>_</u>	UART RX Pin(UART0_RXD)	A7 🔻	Pull Up	*	High	•
EEPROM	\oplus	UART TX Pin(UART0_TXD)	A6 💌	🗗 Pull Up	-	High	-
ETHPHY	(+)						

IPC setting is required for all the cores using IPC RPMessage (core a and b in STEP1). 'IPC Notify + IPC RP Message' should be used. And 'RP Message Number of Buffers' should be min.1 and 'RP Message Buffer Size' should be min 64. They are increased in case IPC RPMessage is also used for other purpose than easyDSP. Also no cache should be used for the shared buffer location (memory

16KB from 0x7200000).

TI DRIVER PORTING LAYER (IRC @	() 400	
▼ TI DRIVERS (23)		GADD	- REMOVE ALL
ADC 🕀	R5FSS0 Core 0 (self)	IPC Notify + IPC RP Message	Ψ
BOOTLOADER 🕀	R5FSS0 Core 1	IPC Notify + IPC RP Message	-
CMPSS 🕀	R5FSS1 Core 0	IPC Notify + IPC RP Message	*
DAC 🕀	P5ESS1 Core 1	IPC Notify + IPC RP Message	
ECAP 🕀		a a a a a a a a a a a a a a a a a a a	
EDMA 🕀	RP Message Number of Buffers	1	*
EPWM 🕀	RP Message Buffer Size (Bytes)	64	*
EQEP 🕀	RP Message Shared Memory (Bytes)	1536	
FSI_RX			
FSI_TX	Other Dependencies		^
GPIO			
12C 🕀			
IPC 1/1 🔮 🕀			
✓ TI DRIVER PORTING LAYER (
Clock 1/1 🔮 🕀	Global Parameters Settings that affect all ins	tances	^
Debug Log 1/1 🤡 🕀			
MPU ARMv7 7/16 🛇 🕀	MPU ARMv7 (7 of 16 Added) ③	(+) ADD	F REMOVE ALL
RAT (+)			÷.
			U
	CONFIG_MPU_REGION1		Ô
	CONFIG MPU REGION2		ń
CMPSS (+)			0
DAC (+)	CONFIG_MPU_REGION3		Ô
ECAP $\stackrel{\smile}{\oplus}$	CONFIG MPU REGION4		ń
EDMA			0
EPWM 🕀	CONFIG_MPU_REGION5		Ô
EQEP 🕀	-		
FSI_RX	Name	CONFIG_MPU_REGION5	
FSI_TX 🕀	Region Start Address (hex)	<u>0x72000000</u>	
GPIO (+)	Region Size (bytes)	16 KB	Ψ
	Access Permissions	Supervisor RD+WR, User RD+WR	•
	Region Attributes	Non Cached	-
	-		
MCAN (1)	Allow Code Execution		

'Supervion RD+WR' is required for the memory area that easyDSP can access so that easyDSP reads/writes the memory location.

▼ TI DRIVER PORTING LA	AYER (DPL) (5)			
Clock	1/1 🥑 🕀	Global Parameters Settings that	affect all instances	^
Debug Log	1/1 🥑 🕀			
MPU ARMv7	4/16 🕑 🕀	MPU ARMv7 (4 of 16 Added) ②	🕂 ADD 📑 FRI	MOVE ALL
RAT	\oplus			-
TIMER	\oplus	CONFIG_MPU_REGION0		
 TI DRIVERS (23) 		CONFIG MPU REGION1		Ê
ADC	\oplus	•		0
BOOTLOADER	\oplus	CONFIG_MPU_REGION2		Ō
CMPSS	\oplus	CONFIG MPU REGIONS		÷
DAC	\oplus			U
ECAP	\oplus	Name	CONFIG_MPU_REGION3	
EDMA	1/1 🕑 🕀	Region Start Address (hex)	0x7000000	
EPWM	\oplus	Pagion Siza (bytas)	2 MB	
EQEP	\oplus	Region Size (bytes)	2 100	
FSI_RX	\oplus	Access Permissions	Supervisor RD+WR, User BLOCK	*
FSI_TX	\oplus	Region Attributes	Cached	Ψ.
GPIO	1 🔮 🕀	Allow Code Execution		
I2C	\oplus	Sub-Region Disable Mark (hex)	0x0	
IPC	\oplus			

STEP 3 : easyDSP project and MCU project

According to STEP1, easyDSP project should be generated to all the yellow cores, and user MCU project should be modified for all the yellow and orange cores.

For the yellow and orage cores, please include easyDSP header and source file (easyAM_v*.*.h, easyAM_v*.*.c) into user MCU project. The suffix of file name will different by its version. You can find these file in the folder easyDSP is installed (\source\AM2x). And set the #define directives based on your application.

```
// Specify whether easyDSP pod is connected to this core
// Define 1 if easyDSP pod is connected to this core
#define EASYDSP POD IS CONNECTED TO THIS CORE
// Specify whether easyDSP communicates with single core or multi cores
// Define 1 if easyDSP communicates with multi cores
// Define 0 if easyDSP communicates with single core
#define EASYDSP IS COMMUNICATING WITH MULTI CORES
// If easyDSP communicates with multi cores, Specify data cache is enabled or not in that cores
// Define 1 if data cache is enabled in the at least one core easyDSP communicates with
// Define 0 if data cache is disabled in all the cores that easyDSP communicates with
#if EASYDSP_IS_COMMUNICATING_WITH_MULTI_CORES
#define D_CACHE_IS_ENABLED
                 1
#endif
// If easyDSP communicates with multi cores with data cache enabled, Specify IPC RPMessage end point
// It should range from 0 to 63
#if EASYDSP_IS_COMMUNICATING_WITH_MULTI_CORES
#if D_CACHE_IS_ENABLED
#define MAIN CORE SERVICE END PT
                    (12U)
#define REMOTE_CORE_SERVICE_END_PT (130)
#endif
#endif
```

And call easyDSP_init() function in the proper location after some initialization functions.

```
#include "easyAM_v*.*.h"
int main()
{
    System_init();
    Board_init();
    Drivers_open();
    Board_driversOpen();
    .
    .
    .
    easyDSP_init();
    .
    .
}
```

Below is the detailed explanation by cases.

Case 1 :

If core a, b, c and d are CPU1, 2, 3 and 4 respectively, the easyDSP project is set as below. The output files of all the running cores are registered. And CPU1 and CPU2 are checked as cores communicating with easyDSP.

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	AM263x Sitara	
Part number	AM2634	
Grade	Grade M	
Output File(s) —		Communication with easyDSP
CPU1 (R5_0_0	C:\temp\cpu1.out	
CPU2 (R5_0_1	C:\temp\cpu2.out	
CPU3 (R5_1_0	C:\temp\cuidebcoulder	
CPU4 (R5_1_1		
		OK Cancel

The setting in the header file as below. Also two end points (m and n) should be set for IPC RPMessage.

	Yellow core	Orange core
setting in easyAM. h	EASYDSP_POD_IS_CONNECTED_TO_THIS_CO RE =1 EASYDSP_IS_COMMUNICATING_WITH_MULTI _CORES =1 D_CACHE_IS_ENABLED =1 MAIN_CORE_SERVICE_END_PT = m REMOTE_CORE_SERVICE_END_PT = n	EASYDSP_POD_IS_CONNECTED_TO_THIS_CO RE = 0 EASYDSP_IS_COMMUNICATING_WITH_MULTI _CORES = 1 D_CACHE_IS_ENABLED = 1 MAIN_CORE_SERVICE_END_PT = m REMOTE_CORE_SERVICE_END_PT = n

Case 2 :

If core e, f, c and d are CPU1, 2, 3 and 4 respectively, the easyDSP project for core e is set as below.

Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	AM263x Sitara	
Part number	AM2634	
Grade	Grade M	
Output File(s) —		Communication with easyDSP
CPU1 (R5_0_0	C:\temp\cu1.out	V
CPU2 (R5_0_1)	C:\temp\cu2.out	
CPU3 (R5_1_0	C:\temp\cpu3.out	
CPU4 (R5_1_1)		
		OK Cancel

the easyDSP	project for core f is set as below.	
Project Settings		×
Basic Hardware MCU Vendor Series Part number	Miscellaneous TI AM263x Sitara AM2634	
Grade Output File(s) —	Grade M	Communication with easyDSP
CPU1 (R5_0_0 CPU2 (R5_0_1 CPU3 (R5_1_0 CPU4 (R5_1_1)) [1) [C:\themp\deltacpu2.out] 2) [1) [
	[OK Cancel

To do RAM booting and flash programming, easyDSP pod should be connected to the core via UARTO. Therefore register all the output files of running cores to easyDSP project of core e (connected to easyDSP pod via UARTO) so that easyDSP project of core e can perform RAM booting and flash programming.

On the other hand, don't perform RAM boooting and flash programming in the easyDSP project of core f.

In case that user program of core f is updated and downloaded to core f by easyDSP project of core e, the easyDSP project of core f needs to reload its output file to update its symbolic information. This is done automatically if both easyDSP projects (core e and core f) are running in the single PC. Then easyDSP project of core f shows the message box below.

easyDSP(2	2)	×
4	The output file is now reloaded as requested by other easyDSP project!	
	ОК	

If both easyDSP projects run in the separate PC, then user need to do manually by executing the menu 'MCU > Reload *.out' in the easyDSP project of core f.

The setting in the header file as below.

	Yellow core
setting in	EASYDSP_POD_IS_CONNECTED_TO_THIS_CORE =1
easyAM.h	EASYDSP_IS_COMMUNICATING_WITH_MULTI_CORES = 0

Case 3 :

If core g, h, c and d are CPU1, 2, 3 and 4 respectively, the easyDSP project for core g is set as below. The output files of all the running cores are registered. And CPU1 and CPU2 are checked as cores communicating with easyDSP.

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI •	
Series	AM263x Sitara	
Part number	AM2634	
Grade	Grade M	
Output File(s) —		Communication with easyDSP
CPU1 (R5_0_0	C:\temp\cpu1.out	V
CPU2 (R5_0_1	C:\temp\cu2.out	V
CPU3 (R5_1_0	C:\temp\cu3.out	
CPU4 (R5_1_1		
		OK Cancel

The setting in the header file as below.

	Yellow core
setting in easyAM.h	EASYDSP_POD_IS_CONNECTED_TO_THIS_CORE =1 EASYDSP_IS_COMMUNICATING_WITH_MULTI_CORES =1 D_CACHE_IS_ENABLED = 0

Case 4 :

If core i, j, c and d are CPU 1, 2, 3 and 4 respectively, the easyDSP project for core i is set as below. The output files of all the running cores are registered. And CPU1 is checked as core communicating with easyDSP.

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	AM263x Sitara	
Part number	AM2634	
Grade	Grade M	
Output File(s) —		Communication with easyDSP
CPU1 (R5_0_0	C:\temp\temp\temp\temp\temp1.out	v
CPU2 (R5_0_1	C:\temp\cout	
CPU3 (R5_1_0	C:\temp\cu3.out	
CPU4 (R5_1_1		
		OK Cancel

The setting in the header file as below.

	Yellow core
setting in	EASYDSP_POD_IS_CONNECTED_TO_THIS_CORE = 1

easyAM.h EASYDSP_IS_COMMUNICATING_WITH_MULTI_CORES =0

STEP 4 : linker.cmd

In the linker.cmd file, the start address of RAM should be same to or larger than 0x7004.0000 for all cores, as it is in the TI example project.

MEMORY

```
{
    .
    .
    .
    OCRAM : ORIGIN = 0x70040000 , LENGTH = 0x40000
    .
    .
}
```

STEP 5 : Variable name

Note that the variable name in the easyDSP is changed when easyDSP is communicating with multi cores.

This is not to mix the variable name from different cores. The variable name 'var' of CPUx (x = 1,2,3 or 4) is changed to 'x:var'. < /FONT>

STEP 6 : IDE setting

1. Make sure that rprc file (*.rprc) is generated in every compilation with the same name and in the same folder to the output file. This is the default setting of TI CCS. rprc file is used for RAM booting and flash programming.

2. The debugging information should be included in the output file. This is the default setting of TI CCS. Otherwise, easyDSP can not recognize the variable.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker option. If necessary, you can set the linker option so that the unused

variables are not excluded.

oe filter text	Miscellaneous	$\langle \neg \bullet \neg \neg \rangle \bullet$
Build		
> SysConfig		
 Arm Compiler 	Configuration: Debug [Active]	Manage Configurations
Processor Options		
Optimization		
Predefined Symbols	Aggressively reduce size of the DWARF information (compress_dwarf)	
Advanced Options	Disable conditional linking and ignore .clink (disable clink, -i)	
Control Options		
Advanced Debug Option	Select trampoline minimization algorithm (minimize_trampolines)	
Language Options	Add <function> to preferred placement order list (preferred_order)</function>	🗐 🗊 🖉 🖓
Parser Preprocessing Opt		
Diagnostic Options		
Runtime Model Options		
Unusual Runtime Model		
Optimization Information		
Instrumentation Options		
Command Files		
Miscellaneous		
✓ Arm Linker		
Basic Options		
File Search Path		
Command File Preproces		
Diagnostics		
Linker Output		
Symbol Management		
Runtime Environment		
Miscellaneous	Strict compatibility checking (strict_compatibility)	
Linker optimization	Minimum space between non-adjacent trampolines (trampoline min spacing)	
Arm Hex Utility [Disabled]	minimum space between non adjacent dampointes (- dampointe_min_spacing)	
Debug	Eliminate sections not needed in the executable (unused_section_elimination)	off
~	Zero initialize ELF uninitialized sections (zero_init)	

7.4.2 AM263x hardware

Connection to easyDSP

easyDSP uses 'UART' boot mode for RAM booting and flash programming, and uses 'QSPI(4S) - Quad Read Mode' boot to run user program in the flash.

Boot Mode	SOP3	SOP2	SOP1	SOP0
QSPI (4S) - Quad Read Mode	0	0	0	0
UART	0	0	0	1
QSPI (1S) - Single Read Mode	0	0	1	0
QSPI (4S) - Quad Read UART Fallback Mode	0	1	0	0
QSPI (1S) - Single Read UART Fallback Mode	0	1	0	1
DevBoot	1	0	1	1

According to the table above, SOP1, SOP2 and SOP3 pins should be low while SOP0 pin is connected to BOOT pin of easyDSP header so that easyDSP can control MCU boot mode.

It is highly recommended to connect RX and TX pins of easyDSP header to MCU UARTO (MUXMODE 0). Otherwise RAM booting and flash programming is not supported.

In case RX and TX pins of easyDSP header are connected to UART other than UART0 (MUXMODE 0), don't connect BOOT and /RESET pin of easyDSP header.

The flash should be connected to MCU QSPI0 and its 'Sector Erase' command should work with 64kB block such as part number S25FL128SAGNFI000 which is used in TI evaluation board. #4 pin of easyDSP header is connected to MCU VDDS33.



Note :

- 25MHz XTAL clock source is required.

- MCU captures SOPx pin status \sim 1ms after PORz release and decides boot mode. So, kindly make sure there would be no signal output from any circuitry connected to SOPx pin \sim 2ms after PORz release.

- TX and RX pin of easyDSP header is pulled up with 100k Ohm resistor inside of easyDSP pod.

- In case there is a reset IC between easyDSP /RESET and MCU PORz, it should transfer easyDSP /RESET signal to MCU within 0.5sec.

easyDSP connection to AM263x Launchpad

The manual work to connect easyDSP to TI AM263x Launchpad is shown below. Note that all the switches of SW1 should be ON and #2 pin of U27 should be detached from PCB.



107
7.5 TM4C

TM4C setting

STEP 1 : Hardware

easyDSP uses MCU's ROM boot loader to access the flash memory. So the UARTO channel (PA0/PA1) that is used in the ROM boot loader should be used for easyDSP.

Otherwise, easyDSP can support only monitoring, not flash programming. Also the source file easyTM4C.c should be modified accordingly by you.

PXn pin acts as a boot pin and you can select it in the easyTM4C.h file. But caution should be taken when selecting boot pin :

1. PC0-3, PD7 and PE7 can't be used for TM4C129x MCU

2. PC0-3, PD7 and PF7 can't be used for TM4C123x MCU

3. In case other circuitry is connected to this pin than easyDSP BOOT pin, this circuit should not issue the output signal until \sim 1sec after MUC reset release.



Other considerations :

- In case there is a reset IC between easyDSP /RESET and MCU -RST, it should transfer easyDSP /RESET signal to MCU -RST within 0.5sec.

- TX and RX pin of easyDSP header is pulled up with 100k Ohm resistor inside of easyDSP pod.

STEP 2 : Modification of easyDSP header file

Two files are provided for easyDSP communication (easyTM4C.h and easyTM4C.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\TM4C).

In the file, please set a target MCU, MCU clock, baudrate of easyDSP communication and boot pin. The baud rate should be same to that of easyDSP project.

```
// step 1 : set target MCU
11
      if TM4C129x is used, set EZ_USE_TM4C129x as 1
11
      if TM4C123x is used, set EZ USE TM4C123x as 1
#define EZ_USE_TM4C129x
              0
#define EZ USE TM4C123x
               1
// step 2 : set the system clock frequency
11
      for example, 120000000L for TM4C129x, 80000000L for TM4C123x
#define EZ_SYS_CLK_FREQ
              80000000L
// step 3 : set the baud rate for UART communication with easyDSP
11
      it should be same to the baudrate of easyDSP project
#define EZ BUAD RATE
              230400
// step 4 : boot pin (PXn) selection
11
      don't use PCO-3, PD7, PE7 for TM4C129x
11
      don't use PCO-3, PD7, PF0 for TM4C123x
11
      below example sets PB5 as a boot pin
#define EZ SYSCTL PERIPH GPIOX SYSCTL PERIPH GPIOB
#define EZ_GPIO_PORTX_BASE GPIO_PORTB_BASE
#define EZ GPIO PIN n
                GPIO PIN 5
```

STEP 3 : Calling easyDSP functions

Please include easyTM4C.h in the main.c. And in the main(), call easyDSP_boot() very begining and call easyDSP_init() after the initialization of MCU.

In the easyDSP_boot() function, it is decided which code will be executed, either user program in the flash or ROM boot loader, depending on the status of boot pin. In case you don't use flash programming by easyDSP, no need for this function.

In the easyDSP_init() function, all necessary setting for easyDSP monitoring are done.

```
int main(void)
ł
    // the very beginning, call easyDSP_boot() to enable ROM boot loader if required
    easyDSP_boot();
    // initial setting
    .
    .
    // call easyDSP init() to enable easyDSP monitoring
    easyDSP_init();
    // loop forever
    while(1)
    Ł
        .
        .
        .
    }
}
```

STEP 4 : IDE setting

#include "easyTM4C.h"

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder of output file (ex *.out) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compiling time.

Please refer to the setting of CCS. Especially for CCS, memory width should be 8.

Properties for TM4C129_Blinky		— L X
type filter text	Arm Hex Utility	
Resource CCS General CCS Build Arm Compiler Arm Linker	Configuration: Debug [Active]	✓ Manage Configurations
Arm Hex Utility General Options	Enable 'Arm Hex Utility'	

Properties for TM4C129_Blinky		- 🗆 X
type filter text	Output Format Options	← → ⇒ 8
 > Resource CCS General CCS Build > Arm Compiler > Arm Linker 	Configuration: Debug [Active]	✓ Manage Configurations
 Arm Hex Utility General Options Diagnostics Options Boot Table Options Output Format Options 	Output format Intel hex (intel, -i)	~
Properties for TM4C129_Blinky		- D X
type filter text	General Options	
 > Resource CCS General CCS Build > Arm Compiler > Arm Compiler 	Configuration: Debug [Active]	V Manage Configurations
 Arm Hex Utility General Options Diagnostics Options Boot Table Options Output Format Options Load Image Options Additional Array Format Optior Builders C/C++ Build C/C++ General Debug Project Natures Project References Run/Debug Settings 	Output as bytes rather than target addressing (byte, -byte) Specify CMAC key file name and enable CMAC (cmac=file) Specify entrypoint address or symbol name (entrypoint, -e=addr) Exclude section from hex conversion (exclude, -exclude=section)	 € £ 2 5 £
	Specify fill value (fill, -fill=val) Select image mode (image, -image) Include linker fill sections in images (linkerfill, -linkerfill) Specify map file name (map, -map=file) Specify memory width (memwidth, -memwidth=width)	
	Specify output file names (outfile, -o=file) Quiet Operation (quiet, -quiet, -q) Specify rom width (romwidth, -romwidth=width) Zero based addressing (zero, -zero, -z)	\${BuildArtifactFileBaseName}.hex
Hide advanced settings	,,,	Apply and Close Cancel

2. For easyDSP monitoring, the debug information should be included in the output file (ex, *.out). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded.

4. To compile inline functions in the easyTM4C.c, plase enables c99 mode in the compiler options if necessay.

STEP 5 : Other setting

1. To allow easyDSP to access the flash, the protection feature of flash should be disabled so that the flash may be written, erased, executed or read.

2. EN bit of BOOTCFG register of MCU should be 1. With this, the booting mechanism is decided by easyDSP_boot() function.

3. easyDSP can perform flash programming only when either all the flash is empty or easyDSP source file is programmed in the flash.

For the other situation than above, you will face the error message below and you should use

debugger to program flash.



7.6 MSPM0

MSPM0 Setting 🚥

STEP 1 : SysConfig - NONMAIN

easyDSP uses the code generated by SysConfig. Below figures are made based on SysConfig 1.16.1. At first, you can set the NONMAIN area such as BCR and BSL configuration. If you use TI factory default, you can skip this step 1. If not, please check below.

First, set the BCR configuration.

Fast Boot Mode is disabled. And BSL is enabled.

 MSPM0 DRIVER LIBRARY (7) 			
V SYSTEM (9)		0,000	IF REMOVE ALL
Board 1/1 📀 🕀			
DMA 🕀	Quick Profiles		Ť
GPIO 1 🥑 🕀	Debug Security Profiles	Security Level 0 - No restrictions	*
MATHACL 🕀			
NONMAIN 1/1 📀 🕀	Reat Configuration Pouting (PCP) Configuration		~
RTC 🕀	Boot Configuration Routine (BCR) Configuration		
SYSCTL 1/1 📀 🕀			
SYSTICK 1/1 📀 🕀	Debug Security Policy Configuration		^
WWDT 🕀			
V ANALOG (6)	SWD Mass Erase and Factory Reset Configuration		^
ADC12 🕀			
COMP 🕀	Flash Memory Static Write Protection (SWP) Configuration	1	^
DAC12 🕀			
GPAMP 🕀	Enable Fast Boot Mode		
OPA 🕀	BCR Configuration ID	0x1	
VREF 🕀	Expected BCR Configuration CRC	0x1879DAC3	
✓ COMMUNICATIONS (6)	Enable RSI		
12C (+)	Enable boe		

Second, set the BSL configuration.

If necessary, set the 32 byte password for entering to bootstrap mode. It's all 0xFF by TI factory default.

BSL Invoke Pin Check should be enabled.

You can use default BSL invoke pin or you can change it to another pin but BSL invoke pin level should be high in any case.

If necessary, set the UART pin. Finally enable BSL read out.

Bootstrap Loader (BSL) Configuration		~
BSL Access[0]	0xFFFFFFFF	
BSL Access[1]	0xFFFFFFF	
BSL Access[2]	0xFFFFFFF	
BSL Access[3]	0xFFFFFFF	
BSL Access[4]	0xFFFFFFF	
BSL Access[5]	0xFFFFFFF	
BSL Access[6]	0xFFFFFFF	
BSL Access[7]	<u>0xFFFFFFF</u>	
BSL GPIO Invoke Configuration		~
Enable BSL Invoke Pin Check		
Use Default BSL Invoke Pin		
BSL Invoke Pin	PA18	~
BSL Invoke Pin PINCM	40	
BSL Invoke Pin Level	High	v
BSL UART Pin Configuration		~
UART Peripheral	UART0	
UART TX Pin	PA10	-
UART TX Pad Number	21	
UART TX Mux	2	
UART RX Pin	PA11	*
UART RX Pad Number	22	
UART RX Mux	2	
BSL I2C Pin Configuration		^
BSL Plugin Configuration		~
BSL Flash Plugin Enable		
Alternate BSL Configuration		~
Use Alternate BSL Configuration		
BSL Configuration ID	0x1	
BSL App Version	0xFFFFFFF	
BSL Read Out Enable		
BSL Security Alert Configuration	Ignore security alert	Ψ.
Expected BSL Configuration CRC	0x7AEDD188	

Note : Since easyDSP can't program NONMAIN flash memory region (such as BCR and BSL configuration area), please use the debugger or any other tool to program NONMAIN flash.

STEP 2 : Hardware

As confitured in STEP 1 or by TI factory default, connect BSL_invoke, BSLRX and BSLTX to easyDSP header.

If you use TI factory default (No change to NONMAIN flash in STEP 1), refer to the target MCU

datasheet to identify pin number of those pins. For instance, BSLRX, BSLTX and BSL_invoke has pin number 26, 27 and 22 respectively for MSPM0L1306xRHB. For instance, BSLRX, BSLTX and BSL_invoke has pin number 57, 56 and 11 respectively for MSPM0G3507SPM. For your information, BSL_RX and BSL_TX belong to UART0.



Other considerations :

- Direct connection between easyDSP /RESET and MCU NRST.
- RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.
- In case pull-up resistor is attached, resistor value should be higher than several k Ohm.

STEP 3 : SysConfig - UART

Since BSL_RX and BSL_TX use UART0, create UART0 peripheral with the name of 'UART_0'. The target baud rate is selectable but it should be same to that of easyDSP project setting. The data format should be 8bit data, one stop bit and no parity bit. FIFO should be enabled with its RX and TX FIFO threshold level as below.

╤ Type Filter Text 🗙 🗸	\leftarrow \rightarrow Software $ ightarrow$ UART	(j <> @ 49 🕴
V MSPM0 DRIVER LIBRARY (7)		
V SYSTEM (9)	OART (1 01 4 Added)	TADD REMOVE ALL
Board 1/1 🥑 🤆	OUART_0	n a
DMA (†		
GPIO 1 🥑 🤂) Name	UART_0
MATHACL (F) Selected Peripheral	UART0
NONMAIN 1/1 🥑 🕀) (
RTC () Quick Profiles	*
SYSCTL 1/1 🔮 🕀	UART Profiles	Custom 👻
SYSTICK 1/1 🕑 (+		
WWDI (4	Pasic Configuration	~
ADC12	Sasic configuration	
COMP G	UADT Initialization Configuration	U III
DAC12 (4) OART Initialization Configuration	
GPAMP (F	Clock Source	BUSCLK
OPA (†) Clock Divider	Divide by 1 👻
VREF (4) Calculated Clock Source	32.00 MHz
✓ COMMUNICATIONS (6)	Target Baud Rate	230400
I2C (+) Coloulated Boud Pate	220215 02
I2C - SMBUS) Calculated Baud Rate	230213.83
MCAN (†) Calculated Error (%)	0.0799
SPI (†) Word Length	8 bits 👻
UART 1/4 🥑 🤆) Parity	None 👻
UART - LIN 🧲) Stop Bits	One 👻
V TIMERS (6)	HW Flow Control	Disable HW flow control
TIMER - CAPTURE		
TIMER - COMPARE (4		
TIMER - PWM	Advanced Configuration	~
TIMER - QEI	UART Mode	Normal UART Mode
Timer Fault	Communication Direction	TX and RX
✓ SECURITY (2)	Oversempling	16%
AES (+) s in size	
TRNG	Enable FIFOs	
V DATA INTEGRITY (1)	RX FIFO Threshold Level	RX FIFO contains >= 1 entry
CRC (†) TX FIFO Threshold Level	TX FIFO is empty
V READ-ONLY (1)	Analog Glitch Filter	Disabled 👻
EVENT 1/1 🤡 🤆	Digital Glitch Filter	0
	Calculated Digital Glitch Filter	0.00 s
	RX Timeout Interrupt Counts	0
	Calculated RX Timeout Interrupt	0.00 c
		0.00 S
	Enable Internal Loopback	
	Enable Majority Voting	
	Enable MSB First	
	Retention Configuration	~
	Low-Power Register Retention	Registers retained
	Disable Retention APIs	
		-

Receive and Transmit interrupt should be enabled with the lowest priority level. It is recommended to have pull-up resistor for TX and RX pins. Finally as configured in step 1, RX and TX pins are set.

Extend Configuration		~
Enable Extend Features		
Interrupt Configuration		~
Enable Interrupts	Receive, Transmit	-
Interrupt Priority	Level 3 - Lowest	Ŧ
DMA Configuration Ø		~
Configure DMA RX Trigger	None	-
Configure DMA TX Trigger	None	*
Pin Configuration		~
TX Pin		~
Direction	Output	Ţ
IO Structure	High-Drive	· ·
Enable pin configuration		
Digital IOMUX Features		~
Internal Resistor	Pull-Up Resistor	-
Invert	Disabled	-
Drive Strength Control	High	.
High-Impedance	Disabled	Ψ
RX Pin		~
Direction	Input	~
IO Structure	High-Drive	
Enable pin configuration		
Digital IOMUX Features		~
Internal Resistor	Pull-Up Resistor	-
Invert	Disabled	*
Hysteresis Control	Disabled	*
Wakeup Logic	Disabled	•
PinMux Peripheral and Pin Configuration		~
UART Peripheral	UARTO	€
RX Pin	PA11/57	_ 🗄
TX Pin	PA10/56	₽
Other Dependencies		^

STEP 4 : easyDSP source file

Please include driverlib from TI in your project since easyDSP uses it for UART communication. Two files are provided for easyDSP communication (easyMSPM0.h, easyMSPM0.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\MSPM0). Please include easyMSPM0.h in the main.c. And in the main(), call easyDSP_init() after the initialization of MCU.

In the easyDSP_init() function, all the setting for easyDSP monitoring are done.

STEP 5 : IDE

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder of output file (ex *.out) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compiling time.

Please refer to the setting of CCS. Especially for CCS, memory width should be 8.

easyDSP help

Properties for G3507_Blinky		– 🗆 X
type filter text	Output Format Options	
> Resource		
General	C. C. J. Debug [Action]	
✓ Build	Configuration: Debug [Active]	 Manage Configurations
 Arm Compiler 		
Processor Options		
Optimization	Output format Intel hex (intel, -i)	~
Include Options Predefined Symbols		
> Advanced Options		
> Arm Linker		
✓ Arm Hex Utility		
General Options Diagnostics Options		
Boot Table Options		
Output Format Options		
Load Image Options		
Arm Objcopy Utility [Disabled]		
> Debug		
< >		
Show advanced settings		Apply and Close Cancel
🍄 Properties for G3507_Blinky		— 🗆 X
type filter text	General Options	
> Resource	•	
General		
✓ Build	Configuration: Debug [Active]	✓ Manage Configurations
> SysConfig Arm Compiler		
Processor Options	Output as bytes rather than target addressing (byte -byte)	
Optimization	Coordinated by the server and another CMAC (arrest)	
Include Options Predefined Symbols	specify CNIAC key file name and enable CNIAC (cmac)	
> Advanced Options	Specify entrypoint address or symbol name (entrypoint, -e)	
> Arm Linker	Exclude section from hex conversion (exclude, -exclude)	🛃 🔊 뭘 한 문티
✓ Arm Hex Utility General Options		
Diagnostics Options	Specify fill value (fill, -fill)	
Boot Table Options	Select image mode (image, -image)	
Output Format Options	Include linker fill sections in images (linkerfill, -linkerfill)	
Load Image Options Additional Array Format Option	Specify map file name (map, -map)	
Arm Objcopy Utility [Disabled]	Specify memory width (memwidth -memwidth)	8
> Debug		
	specify output file names (outfile, -o)	S{BUIIdArtifactFileBaseName}.hex
	Quiet Operation (quiet, -quiet, -q)	
	Specify rom width (romwidth, -romwidth)	
< >	Zero based addressing (zero, -zero, -z)	
Show advanced settings		Apply and Close Cancel

2. For easyDSP monitoring, the debug information should be included in the output file (ex, *.out). And the option of assembler, compiler and linker should be set accordingly.

3. Depending on compiler's optimization level and linker setting, the unused variables could be excluded from the debug information and not shown in the easyDSP.

If you like to avoid this, don't use compiler optimization and set the linker option properly. Like below in case of CCS.

easyDSP help

😚 Properties for G3507_Blinky	_	
type filter text	Miscellaneous 🤤	• => - 8
 > Resource General > Build > SysConfig > Arm Compiler > Arm Compiler > Arm Compiler > Arm Linker Basic Options File Search Path > Advanced Options Command File Preprocessin Diagnostics Linker Output Symbol Management Buitime Environment 	Configuration: Debug [Active] Manage Cor Aggressively reduce size of the DWARF information (compress_dwarf) Disable conditional linking and ignore .clink (disable_clink, -j) Select trampoline minimization algorithm (minimize_trampolines) Add <function> to preferred placement order list (preferred_order)</function>	figurations
Miscellaneous Linker optimization Arm Hex Utility [Disabled] Arm Objcopy Utility [Disabled] > Debug	Strict compatibility checking (strict_compatibility) Minimum space between non-adjacent trampolines (trampoline_min_spacing Eliminate sections not needed in the executable (unused_section_elimination	g)

4. In case of CCS, to program NONMAIN memory area of flash, below option should be set.

🁌 Main 🛛 👕 Program	📟 Target	bource	Common	
--------------------	----------	--------	--------	--

Device Texas Instruments XDS110 USB	Debug Probe/CORTEX_M0P
Program/Memory Load Options	Reset Configuration
Auto Run and Launch Options	Reset target before program load
Misc/Other Options MSPM0 Flash Settings	Reset target after program load
	Reset Type
	O Soft reset
	Hard reset
	Erase Configuration
	!!!Warning: Modifying NONMAIN incorrectly, or erasing it without programming can permanently lock the devic See MSPM0 documentation for more details
	Erase method
	O Erase MAIN memory only
	Erase MAIN and NONMAIN memory (see warning above)
	C Erase MAIN and NONMAIN necessary sectors only (see warning above)
	O Erase MAIN memory sectors by range (specify below)
	O Do not erase Flash memory

7.7 PSoC4

7.7.1 PSoC4 software

Single-application bootloader configuration is required for easyDSP to access onchip flash of MCU. In other configuration, easyDSP can monitor the variables but can not program flash.

Below software setting is explained based on PSoC Creator 4.4.

It is assumed that you are already familiar with bootloader and bootloadable. If not please check the manual from Infineon.



Bootloader

STEP 1 : Bootloader project

Please make a schematic as below by dragging the components from component catalog. And change the name of bootloader component to Bootloader_UART. You can add other components if necessary (ex, LED).



First set the 'Bootloader_UART' component as below capture. Note that 'Wait for command time' should be more than 2000ms.

If required, you can set the security key.

Configure 'Bootloader_UART'		?	×
Name: Bootloader_UART			
General Built-in			۹ ۵
Options Communication component: Dual-application bootloader Golden image support Auto application switching Copier Wait for command Wait for command time (ms): Bootloader application version: Packet checksum type: Fast bootloadable application validation Ø Bootloader application validation Security key: 0x 11 2	UART_1 ✓ 2000 ↓ (0: wait forever) ↓ 0x0000 ● Basic summation ✓ idation ↓ 2 33 44 55 66	Optional commands Get flash size Verify row Erase row Get row checksum Verify application checksum Send data Sync bootloader Get application status Get metadata	
Datasheet	ОК	Apply Cance	1

Second set the UART component as below capture. Use 'UART Basic' tab as its default. Note that 115200bps, 8bits, one stop and no parity is used. In 'UART Advanced' tab, buffer size should be changed.

easyDSP help

Configure 'UART_1'			?	×
Name: UART_1				
Configuration	UART Basic UART A	dvanced UART Pins Bu	ilt-in	٩ ۵
Mode:	Standard ~			
Direction:	TX + RX ~			
Baud rate (bps):	115200 V Actu	al baud rate (bps): 117647 (i)	1	
Data bits:	8 bits 🗸 🗸			
Parity:	None 🗸			
Stop bits:	1 bit \sim			
Oversampling:	12 ≑			
Clock from termin	al			
Median filter				
Retry on NACK				
Inverting RX Enable wakeup f	m Doop Sloop Mada			
	ing			
Datasheet	ОК	Apply	Car	ncel

easyDSP help

onfigure 'UART_1'		?	×
Name: UART_1			
Configuration UART Basic	UART Advanced UART Pins	Built-in	۹ ۵
Buffers size RX buffer size: 272	Interrupt None Internal External	MA RX output TX output	
Interrupt sources UART done TX FIFO not full TX FIFO empty TX FIFO overflow TX FIFO underflow TX lost arbitration TX NACK TX FIFO level	 RX FIFO not empty RX FIFO full RX FIFO overflow RX FIFO underflow RX frame error RX frame error RX parity error RX FIFO level Break detected Break 	width: 11	
FIFO levels TX FIFO: 0 ~ Multiprocessor mode Address (hex): 2 +	RX FIFO: 7 RX FIFO drop On parity error	Dr	
Mask (hex): FF 🜲 Accept matching address in F	X FIFO	or	
RTS Polarity: Active CTS Polarity: Active	Low ~ RTS FIFO lev	el: 4	
Datasheet	ОК Аррју	Cano	el

Please select UART pins according to your design. In this example, P0.4 and P0.5 are used.



Finally call Bootloader_UART_Start() function in the beginning of main(). With this, all set for bootloader project.

```
int main(void)
{
    Bootloader UART Start();
    //CyGlobalIntEnable; /* Enable global interrupts. */
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    for(;;)
    {
        /* Place your application code here. */
    }
}
```

STEP 2 : MCU flash programming with bootloader project

You have to program bootloader project to MCU after compiling bootloader project. If necessary, flash are for bootloader project can be protected.

easyDSP can't program the flash for bootloader project.

Once bootloader project is programmed to flash, easyDSP can program bootloadable project.

STEP 3 : Bootloadable project

Please make the schematic like below from component catalog. Please change the name of UART component to UART_ezDSP.

You can also add other components according to your program (not shown here).

			-												-																						-		
			-																														-				-		
																								1															
				÷	B١	D C	ъt	lc)ê	a	da	ał	Ы	е	- 1	1.								ч	Ъ	ΑI	R	т	۰.	e	z	D	s	F	۰.				
				÷	-	-	~		~		-			Ϋ.	_	÷														-	-	-	-						
				Ł	B	10	0	١t	lc	15	ac	1:	ał	ъl	A	I.							1				L	1/	Δ.	R	т								
				Ł	_	~~	~	~~	10	~				~	~	4							4				`	~	•		•				4				
			-	Ł												ь.							÷												а.		-		
				Ł												ь.							÷												а.				
				Ł												ь.							÷												а.				
				Ł												Ŀ.							÷												а.				
				т												L.							1												а.				
				L	_	_	_	_	_	_	_	_	_	_	_	ц.							L	_	_	_	_	_	_	_	_	_	_	_	а.				
															-						1								1	St	a	nd	la	rc	£.,				
																																			۰.				
	1	1		1	1	1	1	1	1	1						1	1	1	1		1	1	1				1	1	1	1	1	1			1				
			1						1										1			1					1						1				_		
	1	1		1						1							1	1	1		1	1					1	1	1		1						-		

Setting of each component as below : First for Bootloader component. Please use 'General' tab as it is. Also register Bootloader project hex or elf file to 'Dependencies' tab.

Configure 'Bootloadable_1'	? ×							
Name: Bootloadable_1								
General Dependencies Built-in	4 ۵							
Application version: 0x0000								
Application ID: 0x0000								
Application custom ID: 0x0000000								
Manual application image placement								
Placement address: 0x00000000								
Checksum exclude section size (bytes): 0	Checksum exclude section size (bytes): 0							
Datasheet OK Apply	Cancel							
Configure 'Bootloadable_1'	? ×							
Name: Bootloadable_1								
General Dependencies Built-in	4 Þ							
Bootloadable projects require a reference to the associated project's HEX and ELF files. The HEX files extension is *.her extension depends on IDE and can be *.elf, *.out, *.axf, or o Bootloader HEX file: \#BootLoader.cydsn\#CortexM0\#ARM_GCC_541\#Deb	Bootloader x. The ELF files other. ug₩BootLoader.he							
	Browse							
Bootloader ELF file:								
\BootLoader.cydsn\CortexM0\ARM_GCC_541\Deb	ug₩BootLoader.elf							
	Browse							
Datasheet OK Apply	Cancel							

Second for UART component.

Please set the communication speed (bps) in the 'UART Basic' tab. It should be same to bps setting of easyDSP project. But it could be different from bps of bootloader project above. Also note to use 8bits, no parity, 1 bit stop bit. Also set the parameters of ' UART Advanced' tab as below.

easyDSP help

Configure 'UART_ezDS	P'			?	×
Name: UART_ezD	SP				
Configuration	UART Basic	JART Advanced	UART Pins Built	-in	4 ک
Mode:	Standard 🗸 🗸				
Direction:	TX + RX \sim				
Baud rate (bps):	115200 ~	Actual baud r	ate (bps): 117647 🔒		
Data bits:	8 bits 🛛 🗸				
Parity:	None ~				
Stop bits:	1 bit \sim				
Oversampling:	12 🌲				
Clock from termina	al				
Median filter					
Retry on NACK					
Inverting RX					
Enable wakeup fr	om Deep Sleep M	ode			
Low power receiv	ring				
Datasheet		OK	Apply	Cance	el

easyDSP help

Configure 'UART_ezDSP' ?	×
Name: UART_ezDSP	
Configuration UART Basic UART Advanced UART Pins Built-in	٩ ۵
Buffers size Interrupt DMA RX buffer size: 8 Image: State and	
□ TX FIFO empty ☑ RX FIFO underflow □ TX FIFO overflow ☑ RX FIFO underflow □ TX FIFO underflow ☑ RX frame error □ TX lost arbitration □ RX parity error □ TX NACK □ RX FIFO level □ TX FIFO level □ TX FIFO level	
FIFO levels TX FIFO: 0 V RX FIFO: 7 V	
Multiprocessor mode RX FIFO drop Address (hex): 2 Mask (hex): FF Accept matching address in RX FIFO	
Flow control RTS Polarity: Active Low RTS FIFO level: 4 CTS Polarity: Active Low	
Datasheet OK Apply Car	icel

Priority of UART interrupt is recommended to be low not to interrupt higher priority interrupt routine.

Start Page TopDesign.cysch TopDesign.cysch LED_blinky.cydwr							
Instance Name	Interrupt Number	Priority (0 - 3)					
UART_ezDSP_SCB_IRQ	8	3					
🐗 Pins 🕅 Analog 🕑 Clocks 💉 Interrupts 🧖 System 🖺 Directives 🧃 Flash Security							

UART pins should be same to pins of bootloader project. In this example, P0.4 and P0.5 are used.



Source files (easyPSoC4.h and easyPSoC4.c) are provided for easyDSP communication. Please include them in your project. You can find them in the folder of easyDSP installation (\source\PSoC).

Finally call easyDSP_init() function in the main(). Withi this, you are ready to use easyDSP.

```
#include "project.h"
#include "easyPSoC4.h"
int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    easyDSP init();
    for(;;)
    {
        /* Place your application code here. */
     }
}
```

STEP 4 : IDE setting

For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.
 *.cyacd file is used for flash programming. So it should exist in the same folder to output file.
 The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in PSoC4 creator, set the 'Remove Unused Sections' false.

Build Settings			?	×
Configuration:	Debug (Active)	~		
Toolchain:	ARM GCC 5.4-2	016-q2-update ~		
È LED_blinky ⊕- Code Gene ⊕- Debug ⊕- Customizer ⊕- ARM GCC S ⊕ General ⊕ Assemb ⊕ Compile ⊕	ration 5.4-2016-q2-up l bler er teral timization nmand Line timization nmand Line timization	✓ Optimization Remove Unused Sections False Remove Unused Sections Remove Unused Sections Allow the linker to remove unused sections. Must be used in conjunction volume/sections option. mcpu=cortex-m0plus -mthumb -L Generated_Source\#PSoC4 -WI,-Map,\$ \${ProjectShortName}.map -T Generated_Source\#PSoC4\#Cm0plusgcc.ks specs=nano.specs -g ffunction-sections -O3 ffat-to-objects	vith the OutputD	ir}/
		OK Apply	Cancel	

7.7.2 PSoC4 hardware

Please connect easyDSP header RX and TX pin to the selected UART pins of MCU. Also connect easyDSP header #4 pin to VDDD.

RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.



- In case there is a reset IC between easyDSP /RESET and MCU XRES, it should transfer easyDSP /RESET signal to MCU XRES within 0.5sec.

- In case pullup resistor is attached, resistor value should be higher than several k Ohm.

7.8 XMC1

STEP 1 : Hardware

Please connect easyDSP header RX and TX pins to directly UART pins (either P1.3/P1.2 or P0.14/P0.15 pair).

RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.

Also connect easyDSP header #4 pin to VDDP.



STEP 2 : Modification of easyXMC1.h file

Two files are provided for easyDSP communication (easyXMC1.h and easyXMC1.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\XMC). Since XMC Peripheral Library is used in the files, this library should be included in your project.

And modify easyXMC1.h file according to your target USIC channel and baudrate. The baud rate should be same to that of easyDSP project.

Also allocate 8 receive FIFO buffer and 8 transmit FIFO buffer to the channel of USIC easyDSP uses while avoiding conflict to FIFO buffer of the other channel of USIC module.

```
// Select channel (0 or 1) of USIC0 :
// define 1 to target channel. 0 to another.
#define USE USIC0 CH0
                         // use USIC0 channel 0 with pins P0.14 + P0.15
                   0
#define USE USIC0 CH1
                         // use USIC0 channel 1 with pins P1.3 + P1.2
                    1
// Set UART configuration
#define EZ_BUAD_RATE
                            // baud rate for UART communication for easyDSP
                    230400U
#define EZ_TX_BUFFER_START
                   0
                            // FIFO Buffer Partitioning for channel 0 of USICO.
#define EZ_RX_BUFFER_START
                            // FIFO Buffer Partitioning for channel 0 of USICO.
                   8
```

STEP 3 : Calling easyDSP_init()

Pleae include easyXMC1.h in the top of main.c and call easyDSP_init() in the main().

```
#include "easyXMC1.h"
int main(void)
{
    easyDSP_init();
    while(1);
}
```

STEP 4 : IDE setting

1. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

2. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in the Dave, set the 'Remove Unused Sections' unclicked.

Configuration: [Debug [Active]	✓ Manage Configuration:
Tool Settings	🎤 Build Steps 🚇	Build Artifact 🗟 Binary Parsers 😣 Error Parsers
 Debugg Warning Warning Prep Dire Opti Opti Opti Miso ARM-Gi Prep Dire Miso ARM-Gi War Miso Comparison Compar	ing gs CC C Compiler processor ctories imization nings cellaneous CC Assembler processor ctories nings cellaneous CC C Linker eral aries	Script file (-T) /linker_script.ld Browse Do not use standard start files (-nostartfiles) Do not use default libraries (-nodefaultlibs) Do not use default libraries (-nostdlib) Remove unused sections (-Ninkergc-sections) Print removed sections (-Xlinkerprint-gc-sections) Omit all symbol information (-s) Runtime library Newlib-nano Provide default newlib system calls (-specs=nosys.specs) Add floating point support for printf

7.9 XMC4

STEP 1 : Hardware

When XMC4 encounters Power On Reset (PORST) as the reset type, it gets to choose from one of four boot modes based on what is read off the boot pins (JTAG TCK and TMS).

ТСК	TMS	Boot mode
0	1	Normal
0	0	ASC BSL
1	1	ВМІ
1	0	CAN BSL

Since the easyDSP supports only two boot modes (Normal and ASC BSL), TCK pin should be low (0) and TMS pin should be selectable (0 or 1) by easyDSP -BOOT pin during power on reset. Internally to MCU, TCK pin has weak pull-down and TMS pin has weak pull-up. So, external pull down/up resistor is optional. Please connect easyDSP header RX and TX pins to directly P1.4 and P1.5 respectively. Connection to other UART pins than P1.4 and P1.5 will bring no operation. RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.

Also connect easyDSP header #4 pin to VDDP.



Other considerations :

- In case there is a reset IC between easyDSP /RESET and MCU -PORST, it should transfer easyDSP /RESET signal to MCU -PORST within 0.5sec.

- In case pull-up resistor is attached, resistor value should be higher than several k Ohm.

STEP 2 : Modification of easyXMC4.h file

Two files are provided for easyDSP communication (easyXMC4.h and easyXMC4.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\XMC). Since XMC Peripheral Library is used in the files, this library should be included in your project.

And modify easyXMC4.h file according to your target MCU and easyDSP communication baudrate. The baud rate should be same to that of easyDSP project.

Also allocate 8 receive FIFO buffer and 8 transmit FIFO buffer to the channel of USIC easyDSP uses while avoiding conflict to FIFO buffer of the other channel of USIC module.

STEP 3 : Calling easyDSP_init()

Pleae include easyXMC4.h in the top of main.c and call easyDSP_init() in the main().

```
#include "easyXMC4.h"
int main(void) {
    easyDSP_init();
    /* Infinite loop */
    for(;;) {
    }
}
```

STEP 4 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file (ex *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compilation. For example, if DAVE IDE is used :

Settings	
Configuration: Debug [Active]	✓ Manage Configurations
🛞 Tool Settings 🎤 Build Steps 🚇 Bu	uild Artifact 📓 Binary Parsers 😣 Error Parsers
 Debugging Warnings ARM-GCC C Compiler Preprocessor Directories Optimization Warnings Miscellaneous ARM-GCC Assembler Preprocessor Directories Warnings Miscellaneous Section Miscellaneous Section Miscellaneous 	Output file format (-O)

2. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's

optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in the Dave, set the 'Remove Unused Sections' unclicked.

Settings		← - ⇒
Configuration: Debug [Active]	∽ Manage	Configurations
🛞 Tool Settings 🎤 Build Steps 🚇 Bu	uild Artifact 📓 Binary Parsers 🔇 Error Parsers	
 Debugging Warnings ARM-GCC C Compiler Preprocessor Directories Optimization Warnings Miscellaneous ARM-GCC Assembler Preprocessor Directories Warnings Miscellaneous ARM-GCC C Linker General Libraries Miscellaneous 	Script file (-T)/linker_script.ld Do not use standard start files (-nostartfiles) Do not use default libraries (-nodefaultlibs) No startup or default libs (-nostdlib) Remove unused sections (-Xlinkergc-sections) Print removed sections (-Xlinkerprint-gc-section Omit all symbol information (-s) Runtime library Newlib-nano Provide default newlib system calls (-specs=nosys Add floating point support for printf Add floating point support for scanf	Browse ons) s.specs)

7.10 RA

7.10.1 RA hardware

Connection to easyDSP

Direct connection of SCI9 RXD9 and TXD9 of MCU to easyDSP pod is recommended. Please note that SCI9 should be used to program flash by easyDSP.

RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.

Also connect easyDSP header #4 pin to VCC.

Other considerations :

- In case there is a reset IC between easyDSP /RESET and MCU RES, it should transfer easyDSP /RESET signal to MCU RES within 0.5sec.

- In case pull-up resistor is required, resistor value should be higher than several k Ohm.



In case you can't use SCI9, you can use the other SCI channel but only monitoring is available (flash programming not feasible). In this case let /BOOT and /RESET pins be open.

Compatibility to Debugger

easyDSP uses RXD9 and TXD9 pin of MCU which overlaps with some debugger pins such as JTAG TDO, JTAG TDI and SWD SWO in case of some of RA4, RA6 and RA8 MCU. Therefore, in this case, you have to use SWD without SWO.

7.10.2 RA sofrware

RA software (excluding RA0)

easyDSP provides the source file for its communication based on FSP(Flexible Software Package). Hereafter, FSP setting will be explained based on version 3.5.0.

STEP 1 : FSP setting

First, activate FSP by clicking 'configuration.xml' file.

Then go to the Stacks tab and generate UART stack. Depending on MCU type, either r_sci_uart or r_sci_b_uart module should be used.

	🔹 🔄 New Stack > 🛛 🐣 Extend Stack > 🥫
	Analog >
	Artificial Intelligence >
	Audio >
	Bootloader >
	CapTouch >
CAN (r_can)	Connectivity >
I2C Communication Device (rm_comms_i2c	c) DSP >
I2C Master (r_iic_master)	Graphics >
I2C Master (r_sci_i2c)	Input >
I2C Shared Bus (rm_comms_i2c)	Monitoring >
I2C Slave (r_iic_slave)	Motor >
I2S (r_ssi)	Networking >
SPI (r_sci_spi)	Power >
SPI (r_spi)	Security >
UART (r_sci_uart)	Sensor >
USB Composite (r_usb_composite)	Storage >
USB HCDC (r_usb_hcdc)	System >
USB HHID (r_usb_hhid)	Timers >
USB HMSC (r_usb_hmsc)	Transfer >
USB Host Vendor class (r_usb_hvnd)	Search
USB PCDC (r_usb_pcdc)	
USB PHID (r_usb_phid)	
USB PMSC (r_usb_pmsc)	
USB Peripheral Vendor class (r_usb_pvnd)	

No setting to DTC Driver since it is not used. Click UART stack to set the properties.

Stacks Configuration	
Threads 🔹 New Thread 🔊 Remove 🕞	HAL/Common Stacks
 ✓ <u>See HAL/Common</u> ④ g_ioport I/O Port (r_ioport) ④ g_uart0 UART (r_sci_uart) 	
	Transmission Reception [Not [Recommended but optional]

New Window) 🖬 🗄 🐱 🖓 👻 🏷 🖨	* 🔶 🕶 🔿 🕶 🔁
Editor	>			
Appearance	>			
Show View	>	۲	Build Targets	
Perspective	>	Ec	C/C++ Projects	
Navigation	>	₽	Console	Alt+Shift+Q, C
- Desferrer		202	Documents	
Preferences		1≣	Include Browser	
		₽;	Optimization Assistant	
			Outline	Alt+Shift+Q, O
		æ	Problem Details	
		<u></u>	Problems	Alt+Shift+Q, X
		6	Project Explorer	
			Properties	
		S	Search	Alt+Shift+Q, S
		`	Smart Browser	
			Smart Manual	
		ø	Tasks	
			Other	Alt+Shift+Q, Q

In case 'properties' window is not shown, use below menus. Window Help

All the necessary change is shown in red at below picture :

First, enable FIFO if target MCU supports FIFO for this SCI channel.

Also, change its module name to 'g_easyDSP'. And set the channel # to 9 in order to use SCI9 and select baud rate properly. Later in your easyDSP projec setting, the same baudrate should be used. Then change callback name to 'easyDSP_callback' and set its interupt priority to lowest one. TXD9 and RXD9 pins should be selected according to hardware setting (<u>check RA hardware setting</u>).

In the following explanation, P109 and P110 are used for TXD9 and RXD9 respectively . Properties \times Problems \bigcirc Console \bigcirc Smart Browser \bigcirc Smart Manual

g_easyDSP UART (r_sci_uart)

Settings	Property	Value
API Info	✓ Common	
	Parameter Checking	Default (BSP)
	FIFO Support	Enable
	DTC Support	Disable
	Flow Control Support	Disable
	RS-485 Support	Disable
	 Module g_easyDSP UART (r_sci_uart) 	
	✓ General	
	Name	g_easyDSP
	Channel	9
	Data Bits	8bits
	Parity	None
	Stop Bits	1bit
	✓ Baud	
	Baud Rate	115200
	Baud Rate Modulation	Disabled
	Max Error (%)	5
	✓ Flow Control	
	CTS/RTS Selection	Hardware RTS
	Software RTS Port	Disabled
	Software RTS Pin	Disabled
	✓ Extra	
	✓ RS-485	
	DE Pin	Disable
	DE Pin Polarity	Active High
	DE Port Number	Disabled
	DE Pin Number	Disabled
	Clock Source	Internal Clock
	Start bit detection	Falling Edge
	Noise Filter	Disable
	Receive FIFO Trigger Level	One
	✓ Interrupts	
	Callback	easyDSP_callback
	Receive Interrupt Priority	Priority 15
	Transmit Data Empty Interrupt Priority	Priority 15
	Transmit End Interrupt Priority	Priority 15
	Error Interrupt Priority	Priority 15
	✓ Pins	
	CTS9	None
	CTS_RTS9	None
	RXD9	P110
	TXD9	P109

Move to Pins tab and set the pin configuration so that the operation Mode is 'Asyncronous UART' and TXD9 is P109 and RXD9 is P110.

[RA6E1] FSP Configuration □						
Pin Configuration						
Select Pin Configuration			📑 Export to CSV file 🛛 🖺	Configur	e Pin Driver	Warnings
FPB_RA6E1.pincfg	✓ Ma	nage configurations	🗹 Generate data:	g_bsp_pi	n_cfg	
Pin Selection 🗄 🕀	⊟ ↓ <mark>a</mark>	Pin Configuration				
Type filter text		Name	Value	Lock	Link	
Connectivity/SCI		Pin Group Selection	Mixed			
Connectivity:Sci	^	Operation Mode	Asynchronous UART			
SCIU SCI1		✓ Input/Output				
scia		CTS9	None			
SCI2		CTS_RTS9	None		\Rightarrow	
SCIA		RXD9	🗸 P110		\Rightarrow	
	- 11	-	None		\Rightarrow	
Connectivity SDHI		TXD9	🛩 P109		\Rightarrow	
> Connectivity:SDFI						

In case of some RA4, RA6 and RA8 MCU series, TXD9 and RXD9 overlaps with some debugger pins. Please set the debugger operation mode to SWD without SWO use.

Pin Selection 📄 🕀 🕞	↓ <mark>a</mark>	Pin Configuration	😲 Cycle Pin Group		
Type filter text		Name	Value	Lock	Link
Connectivity USBES		Operation Mode	SWD		
> Input:CTSU		✓ Input/Output			
> Input IPO		TCK	None		\Rightarrow
		TDI	None		\Rightarrow
GranhierBDC		TDO	None		\Rightarrow
Starser/OSDL		TMS	None		\Rightarrow
Storage:QSPI		SWCLK	✓ P300	Ê	\Rightarrow
> Storage:SDHi		SWDIO	✓ P108	E Contraction of the second se	
> System:BUS		SWO	None	A	
> V System:CGC					
✓ ✓ System:DEBUG					
DEBUG0					

The input pullup and higher drive capability is recommended to the pins TXD9 and RXD9.

		📑 Export to CSV file	Configure Pin Dri	iver Warnings
✓ N	Nanage configurations	🗹 Generate data:	g_bsp_pin_cfg	
E + -↓ªz	Pin Configuration			
	Name	Value	Link	
	Symbolic Name	ARDUINO D1 PMOD2 M		
^	Comment			
	Mode	Peripheral mode		
	Pull up/down	Input pull-up		
	Output Type	CMOS		
	Drive Capacity	Н		
	✓ Input/Output			
	P109	SCI9 TXD9		
		📑 Export to CSV file 🚿	Configure Pin Dri	iver Warnings
~ №	Aanage configurations	Generate data:	g_bsp_pin_cfg	
E ⊕ ⊨ ↓ªz	Pin Configuration			
	Name	Value	Link	
	Symbolic Name	ARDUINO D0 PMOD2 M		
^	Comment			
	Mode	Peripheral mode		
	Pull up/down	Input pull-up		
	IRQ	None		
	Output Type	CMOS		
	Drive Capacity	H		
	✓ Input/Output			
	P110	SCI9_RXD9	4	
		✓ Manage configurations Image: Image	Image configurations Image configurations Image configuration Image configuration Image configurations Image configuration Image configurations Image configuration Image configuration Image	Image configurations Image configurations Image configuration Image configuration Image configuration Image configurations Image configuration Image configurations Image configuration Image configurations Image configuration Image configuration Image configuration I

some MCUs (for example, RA8E1) can enable/disable the clock input to SCI. In this case, the clock should be enabled in the 'Clocks' tab.

Clocks Configuration						Generate Project Conter
						Restore Defau
XTAL 20MHz				Clock Src: PLL1P	CPUCLK Div /1	✓ → CPUCLK 360MHz
~	PLL Src: HOCO	~			S→ ICLK Div /3	✓ → ICLK 120MHz
HOCO 20MHz V	PLL Div /2	✓ PLL1P Div /2	✓ → PLL1P 360MHz		>> PCLKA Div /3	✓ → PCLKA 120MHz
LOCO 32768Hz	PLL Mul x72.00	> > PLL1Q Div /2	✓ → PLL1Q 360MHz		>> PCLKB Div /6	✓ → PCLKB 60MHz
MOCO 8MHz	↓ PLL 720MHz	PLL1R Div /2	✓ → PLL1R 360MHz		>> PCLKC Div /6	✓ → PCLKC 60MHz
SUBCLK 32768Hz	PLL2 Disabled	~			> PCLKD Div /3	✓ → PCLKD 120MHz
	PLL2 Div /2	✓ PLL2P Div /2	✓ → PLL2P 0Hz		> PCLKE Div /3	✓ → PCLKE 120MHz
	V PLL2 Mul x96.00	> > PLL2Q Div /2	✓ → PLL2Q 0Hz		BCLK Div /12	✓ → BCLK 30MHz
	↓ PLL2 0Hz	PLL2R Div /2	✓ → PLL2R 0Hz		→ FCLK Div /6	✓ → FCLK 60MHz
				CLKOUT Disabled	✓ → CLKOUT Div /1	✓ → CLKOUT 0Hz
L				SCICLK Src: HOCO	✓ → SCICLK Div /1	$^{\vee} \longrightarrow$ SCICLK 20MHz
				SCICLK Disabled SCICLK Src: HOCO SCICLK Src: MOCO	→ SPICLK Div /4	$\sim \longrightarrow$ SPICLK 0Hz
				SCICLK Src: LOCO SCICLK Src: XTAL SCICLK Src: SUBCLK	\rightarrow CANFDCLK Div /8	$\sim \longrightarrow$ CANFDCLK 0Hz
_				SCICLK Src: PLL1P SCICLK Src: PLL1Q SCICLK Src: PLL1Q	→ UCK Div /5	✓ → UCK 0Hz
				SCICLK Src: PLL2P SCICLK Src: PLL2P SCICLK Src: PLL2Q SCICLK Src: PLL2R	→ OCTASPICLK Div /4	✓ → OCTASPICLK 0Hz

Finally generate code.

STEP 2 : Calling easyDSP_init()

Two files are provided for easyDSP communication (easyRA_v11.4.h and easyRA_v11.4.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\RA). Pleae include easyRA_v11.4.h in the hal_entry.c and call easyDSP_init() function.

```
#include "easyRA_v11.4.h"
void hal_entry(void)
{
    .
    .
    .
    .
    easyDSP_init();
}
```

STEP 3 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file (ex *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compilation. For example, if you use e2 studio IDE :

easyDSP help

Settings		⇔ - ⇔ * 8
Configuration: Debug [Active]	∽ Manage	Configurations
🛞 Tool Settings 🛞 Toolchain 🎤 Build Sta	eps 🎐 Build Artifact 🗟 Binary Parsers	8 Error Parsers
 Target Processor Optimization Warnings Debugging S GNU Arm Cross Assembler Preprocessor Includes Warnings Warnings Miscellaneous S GNU Arm Cross C Compiler Preprocessor Includes Optimization Warnings Optimization Warnings Miscellaneous S GNU Arm Cross C Linker General Libraries Miscellaneous S GNU Arm Cross Create Flash Image GNU Arm Cross Print Size General S GNU Arm Cross Print Size 	Output file format (-O) Intel HEX Section: -j .text Section: -j .data Other sections (-j)	

2. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker option. If necessary, you can set the linker option so that the unused variables are not excluded. For example, in the e2Studio, set the 'Remove Unused Sections' unclicked.

Remove unused sections (-Xlinker --gc-sections)

7.10.3 RAO

Connection to easyDSP

You can use either SAU or UARTA for easyDSP communication.

RX and TX pins of easyDSP pod are connected to MCU directl.

No other connection is required since the flash programming is not supported for RA0 series.
RX and TX pins of easyDSP pod are pulled up with 100kOhm resistor in the pod.



FSP setting

First, activate FSP by clicking 'configuration.xml' file. Then go to the Stacks tab and generate UART stack with either r_sau_uart or r_uarta module.



If r_sau_uart module is used for easyDSP communication, please set its properties :

The name of the module is g_easyDSP. Set the channel acc. to your board. The baud rate should be same to the one in the easyDSP project setting. The name of callback is easyDSP_callback. The priority of interrupts are the lowest (higher number). Finally set the pin number.



Property	Value
✓ Common	
Parameter Checking	Default (BSP)
Critical Section Guarding	Disabled
DTC Support	Disable
Enable Single Channel	Disable
Enable Fixed Baudrate	Enable
 Module g_easyDSP UART (r_sau_uart))
✓ General	
Name	g_easyDSP
Channel	0
Data Bits	8 bits
Parity	None
Stop Bits	1 bit
Bit Order	LSB First
✓ Baud	
Baud Rate	115200
✓ Extra	
Operation Clock	CKm0
Tx Signal Level	Standard
✓ Interrupts	
Callback	easyDSP_callback
Transmit End Interrupt Priority	Priority 3
Receive End Interrupt Priority	Priority 3
Error Interrupt Priority	Priority 3
✓ Pins	
RXD0	P100
TXD0	P101

If r_uarta module is used for easyDSP communication, similarly to r_sau_uart module, please set its properties like below.

Generate Project Content	g_easyD9	SP UART (r_uarta)	
conclute respect content	Settings	Property	Value
藯 New Stack > 💒 Extend Stack > 🕡 Remove	ADLInfo	✓ Common	
	API Into	Parameter Checking	Default (BSP)
g_easyDSP UART (r_uarta)		DTC Support	Disable
22-7/		Receive Error Interrupt Mode	Disabled
		 Module g_easyDSP UART (r_uarta) 	
		✓ General	
▲		Name	g_easyDSP
		Channel	0
Add DIC Driver for		Data Bits	8bits
[Recommended but recommended]		Parity	None
optional]		Stop Bits	1bit
-		✓ Baud	
		Baud Rate	115200
		✓ Extra	
		Transfer Order	LSB first
		Transfer level	Positive logic
		Clock output	Not Available
		✓ Interrupts	
		Callback	easyDSP_callback
		Receive Interrupt Priority	Priority 3
		Transmit Interrupt Priority	Priority 3
		Error Interrupt Priority	Priority 3
		✓ Pins	
		RXDA	P207
		TXDA	P208

Then go to the Pins tab, and set the pull-up to both RXD and TXD pin.

Pin Selection 📔 🕀 🖻	Pin Configuration
Type filter text	Name Value Link Symbolic Name ARDUINO_RX
> Ports	Comment
> 🗸 P1	Mode Peripheral mode Pull up/down Input pull-up
✓ ✓ P2 ✓ P200	IRQ None
✓ P201	Output Type CMOS
✓ P206	 ✓ Input/Output
✓ P207 ✓ P208	P207 VARTA_RXDA

Also check if the clock to the used communication channel is enabled in the Clocks tab. Finally generate the code.

Generate Project Content

Calling easyDSP_init() and IDE setting

Same to the other RA series. So please check here.

7.11 RX

7.11.1 RX hardware

To do monitoring and flash programming together, SCI1 should be connected to easyDSP. So connect RXD1 and TXD1 pins of MCU to the easyDSP RX and TX pins. Also connect easyDSP header #4 pin to MCU VCC.



Please check the corresponding pins by MCU type in the table below. The number of pin should be checked from MCU datasheet.

N	MCU RXD1		TXD1	UB or UB#
	RX110	P15	P16	N.A.
	RX111	P15	P16	P14/UB#
DV400	RX113	P15	P16	P14/UB#
KX100	RX130	P30	P26	N.A.
	RX13T	PB7	PB6	N.A.
	RX140	P30	P26	N.A.
	RX230 RX231	P30	P26	PC7/UB
	RX23E-A	P30	P26	N.A.
	RX23T	PD5	PD3	N.A.
RX200	RX23W	P30	P26	PC7/UB
	RX24T	PD5	PD3	N.A.
	RX24U	PD5	PD3	N.A.
	RX26T	PD5	PD3	N.A.
		PF0 (177/176-pin products)	PF2 (177/176-pin products)	
	RX64M	P26 (145/144/100-pin products)	P30 (145/144/100-pin products)	PC7/UB
	RX651	PF0 (177- and 176-pin products)	PF2 (177- and 176-pin products)	
	RX65N	P26 (145-, 144-, 100-, and 64-pin products)	P30 (145-, 144-, 100-, and 64-pin products)	PC7/UB
RX600	RX660	P26	P30	PC7/UB
		PF0 (224- and 176-pin products)	PF2 (224- and 176-pin products)	
	RX66N	P26 (145-, 144-, and 100-pin products)	P30 (145-, 144-, and 100-pin products)	PC7/UB
	RX66T	PD3	PD5	P00/UB
	RX671	P26	P30	PC7/UB
		PF0 (177/176-pin products)	PF2 (177/176-pin products)	
	RX71M	P26 (145/144/100-pin products)	P30 (145/144/100-pin products)	PC7/UB
RX700	RX72M	PF0 (224- and 176-pin products)	PF2 (224- and 176-pin products)	
	RX72N	P26 (144-, and 100-pin products)	P30 (144-, and 100-pin products)	PC7/UB
	RX72T	PD3	PD5	P00/UB

In case MCU has UB or UB# pin, it should be pulled down or pulled up respectively.

note) N.A. = not available

Other considerations :

- When reset, easyDSP /RESET pin goes low for 500msec around.

- In case there is a reset IC between easyDSP /RESET and MCU RES#, it should transfer easyDSP /RESET signal to MCU RES# within 0.5sec.

- RX and TX pins of easyDSP header are pulled up with 100kOhm resistor in the pod.

- In case you can't use SCI1, you can use another SCI channel but only monitoring is doable (flash programming not doable). In this case no need to connect /BOOT and /RESET pins.

7.11.2 RX sofrware

easyDSP uses the generated code from <u>RX Smart Configurator</u>. You can find the detailed process below based on RX Smart Configurator v1.40.

STEP 1 : Smart Configurator setting

Please add 'SCI Driver' component by cliking 'Add component' button in the 'Components' tab.

mponents	ù⊿l₂ - + ‡ ▼	Configure					
	ं र	📴 New Co	mponent				×
<pre>/pe filter text /</pre>		Software C Select com	Component Selection	ist		ŧ	
		Category	Drivers				``
		Function Filter	Communications				~
		Compon	ents	Short Na	Туре	Versio	on ^
		🖶 SCI Dr	iver	r_sci_rx	Firmware Integration Technology	4.40	
		SCI/SC	CIF Asynchronous Mode		Code Generator	1.12.0) \ >
		Show o	nly latest version ems that have duplicated functio	nality			
		Description Depender Depender This mod mode, sir	n ncy : r_bsp version(s) 7.20 ncy : r_byteq version(s) 1.40, 1.50 ule allows any number of SCI ch ngle master Simple SPI mode, or), 1.60, 1.70, 1.71, 1 annels on the MC master Synchrono	.80, 1.81, 1.82, 1.90, 2.00 U to run in full duplex Asynchronou us mode simultaneously. The drive	ıs ris	,
		Download Configure	the latest FIT drivers and middle general settings	ware			
				-		-	

Then r_sci_rx and r_byteq components are created.

Software component configura	ation
Components	≧⊿ ё₂ ⊏ ⊕ ‡ ◄
	😺 🐨
✓	^
🗸 🗁 Generic	
💣 r_bsp	
✓	
🗸 🗁 Communications	
😫 r_sci_rx	
✓ → Middleware	
🗸 🗁 Generic	
😫 r_byteq	
	~
Overview Board Clocks System Compone	ents Pins Interrupts

Since easyDSP uses SCI channel 1, 'r_sci_rx' components should be set accordingly. Please refer to the red line below.

The circular buffer is not required for easyDSP. TX and RX queue buffer size should be 12 and 2 respectively at its minimum.

oftware component confi	iguration	Generate Code
omponents 뉦 🛃 🖧 🕒 🕀 🗄	+), → Configure	
10 T	Property	Value
type filter text	✓ ∅ Configurations	
✓ → Startup	# Parameter checking	System Default
✓ (⇒ Generic	# Use ASYNC mode	Include
📄 r bsp	# Use SYNC mode	Not
V > Drivers	# Use SSPI mode	Not
✓ → Communications	# Use IRDA mode	Not
💁 r sci rx	# Use circular buffer in ASYNC mode	Unused
✓ → Middleware	# Byte value to transmit while clocking in data in SSPI mode	0xFF
✓ 🗁 Generic	# Include software support for channel 0	Not
💁 r byteg	# Include software support for channel 1	Include
	# Include software support for channel 2	Not
	# Include software support for channel 3	Not
	# Include software support for channel 4	Not
	# Include software support for channel 5	Not
	# Include software support for channel 6	Not
	# Include software support for channel 7	Not
	# Include software support for channel 8	Not
	# Include software support for channel 9	Not
	# Include software support for channel 10	Not
	# Include software support for channel 11	Not
	# Include software support for channel 12	Not
	# ASYNC mode TX queue buffer size for channel 0	80
	# ASYNC mode TX queue buffer size for channel 1	12
	# ASYNC mode TX queue buffer size for channel 2	80
	# ASYNC mode TX queue buffer size for channel 3	80
	# ASYNC mode TX queue buffer size for channel 4	80
	# ASYNC mode TX queue buffer size for channel 5	80
	# ASYNC mode TX queue buffer size for channel 6	80
	# ASYNC mode TX queue buffer size for channel 7	80
	# ASYNC mode TX queue buffer size for channel 8	80
	# ASYNC mode TX gueue buffer size for channel 9	80
	# ASYNC mode TX queue buffer size for channel 10	80
	# ASVNC mode TX queue buffer size for channel 11	80
	# ASYNC mode TX queue buffer size for channel 12	80
	# ASYNC mode BX queue buffer size for channel 0	80
	# ASYNC mode BX queue buffer size for channel 1	2
	# ASYNC mode RX queue buffer size for channel ?	80
	# ASYNC mode RY queue buffer size for channel 2	90
	# ASYNC mode RV queue buffer size for channel 4	00
	# ASYNC mode RX queue buffer size for channel 4	00
	# ASYNC mode RX queue buffer size for channel 5	00
	ASTING mode KA queue buffer size for channel 0	00
	ASYNC mode KX queue buffer size for channel /	80
	ASYNC mode KX queue butter size for channel 8	80
	# ASYNC mode KX queue buffer size for channel 9	80
	# ASYNC mode KX queue buffer size for channel 10	80
	# ASYNC mode RX queue buffer size for channel 11	80

TEI interrupt is not used. The interrupt priority level of ERI and TEI should be the lowest, 1.

oftware component co	nfigu	ration		
Components 🚵 🛃 🎘 📄 🕀	₽	Configure		
٤	5	Property		Value
type filter text		#	Transmit end interrunt	Disable
		#	GROUPBLO (ERI TEI) interrunt priority	1
V 🗁 Startup	^	#	TX/RX FIEO for channel 7	Not
V 🗁 Generic		#	TX/RX FIEO for channel 8	Not
r_bsp		#	TX/RX FIEO for channel 9	Not
V 🔁 Drivers		#	TX/RX FIEO for channel 10	Not
Communications		#	TX/RX FIEO for channel 11	Not
r_sci_rx		#	TX FIFO threshold for channel 7	8
V 🔁 Middleware		#	TX FIFO threshold for channel 8	8
V 🔁 Generic		#	TX FIFO threshold for channel 9	8
🍟 r_byteq		#	TX FIFO threshold for channel 10	8
		#	TX FIFO threshold for channel 11	8
		#	RX FIFO threshold for channel 7	8
		#	RX FIFO threshold for channel 8	8
		#	RX FIFO threshold for channel 9	8
		#	RX FIFO threshold for channel 10	8
		#	RX FIFO threshold for channel 11	8
		#	Received data match function for channel 0	Not
		#	Received data match function for channel 1	Not
		#	Received data match function for channel 2	Not
		#	Received data match function for channel 3	Not
		#	Received data match function for channel 4	Not
		#	Received data match function for channel 5	Not
		#	Received data match function for channel 6	Not
		#	Received data match function for channel 7	Not
		#	Received data match function for channel 8	Not
		#	Received data match function for channel 9	Not
		#	Received data match function for channel 10	Not
		#	Received data match function for channel 11	Not

at up Image: Configure Property Value generic Image: Configure Image: Control Use DTC/DMAC for transmit (SCI) 0 Image: Control Image: Control 0 Image: Contr Image: Control 0<	ware component confi	iguration	Generate Co
Property Value artup 0 Generic 0 * Use DTC/DMAC for transmit (SCI2) 0 * Chype 0 * Use DTC/DMAC for transmit (SCI3) 0 * Use DTC/DMAC for transmit (SCI10) 0 * Use DTC/DMAC for transmit (SCI11) 0 * Use DTC/DMAC for receive (SCI3) 0 <	ponents 🚵 🛃 🖧 🗐 🕀 🗄	⊉ j Configure	
<pre># Use DTC/DMAC for transmit (SCI) 0 # Use DTC/DMAC for receive (SCI) 0 # Use DTC/DMAC for recei</pre>	ت ت	Property	Value
artup • Generic • ** Lobp • ** Communications • ** raci, no • ** Generic • ** sci, no • ** communications • ** raci, no • ** sci, no • ** dee DTC/DMAC for transmit (SCI6) • ** sci, no • ** dee DTC/DMAC for transmit (SCI6) • ** dee DTC/DMAC for transmit (SCI10) • ** dee DTC/DMAC for transmit (SCI11) • ** dee DTC/DMAC for transmit (SCI12) • ** dee DTC/DMAC for receive (SC13) • ** dee CT/DMAC for recei		# Use DTC/DMAC for transmit (SCI1)	0
arup Generic	Con Standard A	# Use DTC/DMAC for transmit (SCI2)	0
Use DTC/DMAC for transmit (SCI4) 0 r_sbsp # Use DTC/DMAC for transmit (SCI6) 0 Communications # Use DTC/DMAC for transmit (SCI6) 0 Serieric # Use DTC/DMAC for transmit (SCI9) 0 Generic # Use DTC/DMAC for transmit (SCI9) 0 # Use DTC/DMAC for transmit (SCI9) 0 # Use DTC/DMAC for transmit (SCI9) 0 # Use DTC/DMAC for transmit (SCI10) 0 # Use DTC/DMAC for transmit (SCI12) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI1) 0 # <	Startup	# Use DTC/DMAC for transmit (SCI3)	0
Close ives Communications # Use DTC/DMAC for transmit (SCI5) 0 Status ives DTC/DMAC for transmit (SCI6) 0 Status ives DTC/DMAC for transmit (SCI7) 0 Generic # Use DTC/DMAC for transmit (SCI9) 0 # Use DTC/DMAC for transmit (SCI10) 0 0 # Use DTC/DMAC for receive (SCI0) 0 0 # Use DTC/DMAC for receive (SCI3) 0 0 # Use DTC/DMAC for receive (SCI3) 0 0 # Use DTC/DMAC for receive (SCI9) 0 0 # Use DTC/DMAC for receive (SCI10) 0 0 # Use DTC/DMAC for receive (SCI9) 0 0 # Use DTC/DMAC for receive (SCI10) 0 0 # Use DTC/DMAC for receive (SCI10) 0 0	V 🔁 Generic	# Use DTC/DMAC for transmit (SCI4)	0
wers in the DTC/DMAC for transmit (SCI9) 0 © r.s.jt n if Use DTC/DMAC for transmit (SCI9) 0 Senarc if Use DTC/DMAC for transmit (SCI9) 0 Senarc if Use DTC/DMAC for transmit (SCI9) 0 Senarc if Use DTC/DMAC for transmit (SCI9) 0 idelware 0 0 if Use DTC/DMAC for transmit (SCI9) 0 0 if Use DTC/DMAC for transmit (SCI1) 0 0 if Use DTC/DMAC for transmit (SCI1) 0 0 if Use DTC/DMAC for transmit (SCI2) 0 0 if Use DTC/DMAC for receive (SCI3) 0 0 if Use DTC/DMAC for receive (SCI1) 0 0 if Use DTC/DMAC for receive (SCI3) 0 0 if Use DTC/DMAC for receive (SCI3) 0 0 if Use DTC/DMAC for receive (SCI1) <	er_bsp	# Use DTC/DMAC for transmit (SCI5)	0
Communications Use DTC/DMAC for transmit (SCI9) Generic ''''''''''''''''''''''''''''''''''''	Drivers	# Use DTC/DMAC for transmit (SCI6)	0
br cscip br br Dit/DMAC for transmit (SCIB) 0 Genetic # Use DTC/DMAC for transmit (SCIB) 0 Br r_byteq Use DTC/DMAC for transmit (SCID) 0 # Use DTC/DMAC for receive (SCID) 0	Communications	# Use DTC/DMAC for transmit (SCI7)	0
iddleware iddleware iscencic if Use DTC/DMAC for transmit (SCII0) 0 if Use DTC/DMAC for transmit (SCII1) 0 if Use DTC/DMAC for receive (SCI) 0 if Use DTC/DMAC for receive (SCI) <td>😜 r_sci_rx</td> <td># Use DTC/DMAC for transmit (SCI8)</td> <td>0</td>	😜 r_sci_rx	# Use DTC/DMAC for transmit (SCI8)	0
Generic Gen	Middleware	# Use DTC/DMAC for transmit (SCI0)	0
** Ode DD/C/DMAC for transmit (SCI10) 0 #* Use DTC/DMAC for transmit (SCI11) 0 #* Use DTC/DMAC for treasmit (SCI12) 0 #* Use DTC/DMAC for receive (SCI0) 0 #* Use DTC/DMAC for receive (SCI1) 0 #* Use DTC/DMAC for receive (SCI2) 0 #* Use DTC/DMAC for receive (SCI3) 0 #* Use DTC/DMAC for receive (SCI3) 0 #* Use DTC/DMAC for receive (SCI6) 0 #* Use DTC/DMAC for receive (SCI6) 0 #* Use DTC/DMAC for receive (SCI6) 0 #* Use DTC/DMAC for receive (SCI9) 0 #* Use DTC/DMAC for receive (SCI9) 0 #* Use DTC/DMAC for receive (SCI10) 0 #* Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 #* Select channel DMAC in case using DMAC for transferring data (TX S	→ Generic	# Use DTC/DMAC for transmit (SCI10)	0
# Use DTC/DMAC for transmit (SCI11) 0 # Use DTC/DMAC for receive (SCI0) 0 # Use DTC/DMAC for receive (SCI2) 0 # Use DTC/DMAC for receive (SCI2) 0 # Use DTC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI4) 0 # Use DTC/DMAC for receive (SCI5) 0 # Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0	檺 r_byteq	# Use DTC/DMAC for transmit (SCI10)	0
# Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI2) 0 # Use DTC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC for tran		# Use DTC/DMAC for transmit (SCITI)	0
# Use DTC/DMAC for receive (SCI2) 0 # Use DTC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0		# Use DIC/DMAC for transmit (SCI12)	0
# Use DIC/DMAC for receive (SCI2) 0 # Use DIC/DMAC for receive (SCI3) 0 # Use DIC/DMAC for receive (SCI3) 0 # Use DIC/DMAC for receive (SCI5) 0 # Use DIC/DMAC for receive (SCI6) 0 # Use DIC/DMAC for receive (SCI6) 0 # Use DIC/DMAC for receive (SCI7) 0 # Use DIC/DMAC for receive (SCI9) 0 # Use DIC/DMAC for receive (SCI9) 0 # Use DIC/DMAC for receive (SCI9) 0 # Use DIC/DMAC for receive (SCI10) 0 # Use DIC/DMAC for receive (SCI11) 0 # Use DIC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC		Use DIC/DMAC for receive (SCI0)	0
# Use DIC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI4) 0 # Use DTC/DMAC for receive (SCI5) 0 # Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI7) 0 # Use DTC/DMAC for receive (SCI7) 0 # Use DTC/DMAC for receive (SCI7) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 <td< td=""><td></td><td># Use DIC/DMAC for receive (SCI1)</td><td>0</td></td<>		# Use DIC/DMAC for receive (SCI1)	0
# Use DTC/DMAC for receive (SCI3) 0 # Use DTC/DMAC for receive (SCI5) 0 # Use DTC/DMAC for receive (SCI5) 0 # Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI7) 0 # Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC f		# Use DTC/DMAC for receive (SCI2)	0
# Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC		# Use DTC/DMAC for receive (SCI3)	0
# Use DTC/DMAC for receive (SCI5) 0 # Use DTC/DMAC for receive (SCI7) 0 # Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transferring data (TX SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0		# Use DTC/DMAC for receive (SCI4)	0
# Use DTC/DMAC for receive (SCI6) 0 # Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI12) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 <td></td> <td># Use DTC/DMAC for receive (SCI5)</td> <td>0</td>		# Use DTC/DMAC for receive (SCI5)	0
# Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select cha		# Use DTC/DMAC for receive (SCI6)	0
# Use DTC/DMAC for receive (SCI8) 0 # Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transmit (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0		# Use DTC/DMAC for receive (SCI7)	0
# Use DTC/DMAC for receive (SCI9) 0 # Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transferring data (TX SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring		# Use DTC/DMAC for receive (SCI8)	0
# Use DTC/DMAC for receive (SCI10) 0 # Use DTC/DMAC for receive (SCI11) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transmit (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in ca		# Use DTC/DMAC for receive (SCI9)	0
# Use DTC/DMAC for receive (SCI1) 0 # Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transmit (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI8) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI8) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1		# Use DTC/DMAC for receive (SCI10)	0
# Use DTC/DMAC for receive (SCI12) 0 # Select channel DMAC in case using DMAC to transmit (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI0) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI11)<		# Use DTC/DMAC for receive (SCI11)	0
#Select channel DMAC in case using DMAC to transmit (TX SCI0)0#Select channel DMAC in case using DMAC for transferring data (TX SCI2)0#Select channel DMAC in case using DMAC for transferring data (TX SCI3)0#Select channel DMAC in case using DMAC for transferring data (TX SCI3)0#Select channel DMAC in case using DMAC for transferring data (TX SCI3)0#Select channel DMAC in case using DMAC for transferring data (TX SCI5)0#Select channel DMAC in case using DMAC for transferring data (TX SCI6)0#Select channel DMAC in case using DMAC for transferring data (TX SCI7)0#Select channel DMAC in case using DMAC for transferring data (TX SCI8)0#Select channel DMAC in case using DMAC for transferring data (TX SCI9)0#Select channel DMAC in case using DMAC for transferring data (TX SCI10)0#Select channel DMAC in case using DMAC for transferring data (TX SCI10)0#Select channel DMAC in case using DMAC for transferring data (TX SCI11)0#Select channel DMAC in case using DMAC for transferring data (TX SCI12)0#Select channel DMAC in case using DMAC for transferring data (RX SCI10)1#Select channel DMAC in case using DMAC for transferring data (RX SCI2)1#Select channel DMAC in case using DMAC for transferring data (RX SCI2)1#Select channel DMAC in case using DMAC for transferring data (RX SCI2)1#Select channel DMAC in case using DMAC for transferring data (RX SCI3)1<		# Use DTC/DMAC for receive (SCI12)	0
 # Select channel DMAC in case using DMAC for transferring data (TX SCI1) # Select channel DMAC in case using DMAC for transferring data (TX SCI3) # Select channel DMAC in case using DMAC for transferring data (TX SCI4) # Select channel DMAC in case using DMAC for transferring data (TX SCI5) # Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) # Select channel DMAC in case using DMAC for transferring data (TX SCI9) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) # Select channel DMAC in case using DMAC for transferring data (TX SCI11) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (RX SCI1) # Select channel DMAC in case using DMAC for transferring data (RX SCI2) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (# Select channel DMAC in case using DMAC to transmit (TX SCI0)	0
 # Select channel DMAC in case using DMAC for transferring data (TX SCI2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI8) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) <l< td=""><td></td><td># Select channel DMAC in case using DMAC for transferring data (TX SCI1)</td><td>0</td></l<>		# Select channel DMAC in case using DMAC for transferring data (TX SCI1)	0
 # Select channel DMAC in case using DMAC for transferring data (TX SCI3) # Select channel DMAC in case using DMAC for transferring data (TX SCI4) # Select channel DMAC in case using DMAC for transferring data (TX SCI5) # Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) # Select channel DMAC in case using DMAC for transferring data (TX SCI8) # Select channel DMAC in case using DMAC for transferring data (TX SCI9) # Select channel DMAC in case using DMAC for transferring data (TX SCI9) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) # Select channel DMAC in case using DMAC for transferring data (TX SCI11) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (RX SCI0) # Select channel DMAC in case using DMAC for transferring data (RX SCI0) # Select channel DMAC in case using DMAC for transferring data (RX SCI2) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (R		# Select channel DMAC in case using DMAC for transferring data (TX SCI2)	0
 # Select channel DMAC in case using DMAC for transferring data (TX SCI4) # Select channel DMAC in case using DMAC for transferring data (TX SCI5) # Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) # Select channel DMAC in case using DMAC for transferring data (TX SCI8) # Select channel DMAC in case using DMAC for transferring data (TX SCI8) # Select channel DMAC in case using DMAC for transferring data (TX SCI9) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) # Select channel DMAC in case using DMAC for transferring data (TX SCI11) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (TX SCI12) # Select channel DMAC in case using DMAC for transferring data (RX SCI0) # Select channel DMAC in case using DMAC for transferring data (RX SCI1) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI4) # Select channel DMAC in case using DMAC for transferring data (RX SCI6) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI3) # Select channel DMAC in case using DMAC for transferring data (RX SCI6) # Select channel DMAC in case using DMAC for transferring data (# Select channel DMAC in case using DMAC for transferring data (TX SCI3)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI5) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI6) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI8) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6)		# Select channel DMAC in case using DMAC for transferring data (TX SCI4)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI6) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI8) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 #		# Select channel DMAC in case using DMAC for transferring data (TX SCI5)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI0) # Select channel DMAC in case using DMAC for transferring data (TX SCI7) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI9) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 <p< td=""><td></td><td># Select channel DMAC in case using DMAC for transferring data (TX SCI6)</td><td>0</td></p<>		# Select channel DMAC in case using DMAC for transferring data (TX SCI6)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCIP) # Select channel DMAC in case using DMAC for transferring data (TX SCIP) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCIP) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCII) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCII) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCII2) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCII2) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCII) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCII) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # S		# Select channel DMAC in case using DMAC for transferring data (TX SCI7)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI0) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 <		# Select channel DMAC in case using DMAC for transferring data (TX SCIP)	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI9) # Select channel DMAC in case using DMAC for transferring data (TX SCI10) # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select		 Select channel DMAC in case using DMAC for transferring data (TX SCI0) Select channel DMAC in case using DMAC for transferring data (TX SCI0) 	0
# Select channel DMAC in case using DMAC for transferring data (TX SCI10) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI11) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (Select channel DWAC in case using DWAC for transferring data (TX SCI9) Select channel DMAC in case using DMAC for transferring data (TX SCI10) 	0
Select channel DMAC in case using DMAC for transferring data (TX SCI1) 0 # Select channel DMAC in case using DMAC for transferring data (TX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX S		Select channel DWAC in case using DWAC for transferring data (TX SCIIU)	0
# Select channel DMAC in case using DMAC for transferring data (IX SCI12) 0 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not		Select channel DMAC in case using DMAC for transferring data (TX SCITI)	0
Felect channel DMAC in case using DMAC for transferring data (KX SCI0) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 Hoclude software support for IrDA channel 5 Not Set be non-active lavel of the TYD rain		Select channel DMAC in case using DMAC for transferring data (IX SCI12)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI1) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active lavel of the TYD pin		Select channel DMAC in case using DMAC for transferring data (RX SCI0)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI2) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in		# Select channel DMAC in case using DMAC for transferring data (RX SCI1)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI3) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active level of the TYD pin		# Select channel DMAC in case using DMAC for transferring data (RX SCI2)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI4) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active lavel of the TXD zin		# Select channel DMAC in case using DMAC for transferring data (RX SCI3)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI5) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI0) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active lavel of the TYD pin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI4)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI6) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active lavel of the TYD pin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI5)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI7) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active lavel of the TYD pin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI6)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI8) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the proparties layed of the TXD sin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI7)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI9) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the propactive level of the TXP pin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI8)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI10) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the properties level of the TVD pip Include software support for IrDA channel 5		# Select channel DMAC in case using DMAC for transferring data (RX SCI9)	1
# Select channel DMAC in case using DMAC for transferring data (RX SCI11) 1 # Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 # Include software support for IrDA channel 5 Not # Set the non-active level of the TVD nin Include		# Select channel DMAC in case using DMAC for transferring data (RX SCI10)	1
Select channel DMAC in case using DMAC for transferring data (RX SCI12) 1 Include software support for IrDA channel 5 Not Set the non-active level of the TVD ain		# Select channel DMAC in case using DMAC for transferring data (RX SCI11)	1
# Include software support for IrDA channel 5 Not		# Select channel DMAC in case using DMAC for transferring data (RX SCI12)	1
# Set the non-active level of the TVD nin level of		# Include software support for IrDA channel 5	Not
Include the tever of the two on		# Set the non-active level of the TXD nin	Include

RXD1 and TXD1 pins of SCI1 should be enabled. The other pins of SCI1 are not used.

tware component config	uration	Generate Co
nponents 迠 🛃 🖧 🗖 🕀 🛱	Configure	
ت ت	Property	Value
pe filter text	# Set the non-active level of the RXD pin	Include
~ ~ .	# Receive data sampling timing adjustment CH0	Disable
Startup	# Receive data sampling timing adjustment CH1	Disable
V 🗁 Generic	# Receive data sampling timing adjustment CH2	Disable
er_osp	# Receive data sampling timing adjustment CH3	Disable
Drivers	# Receive data sampling timing adjustment CH4	Disable
	# Receive data sampling timing adjustment CH5	Disable
r_sci_tx	# Receive data sampling timing adjustment CH6	Disable
	# Receive data sampling timing adjustment CH7	Disable
Generic	# Receive data sampling timing adjustment CH8	Disable
er Loyted	# Receive data sampling timing adjustment CH9	Disable
	# Receive data sampling timing adjustment CH10	Disable
	# Receive data sampling timing adjustment CH11	Disable
	# Transmit signal transition timing adjustment CH0	Disable
	# Transmit signal transition timing adjustment CH1	Disable
	# Transmit signal transition timing adjustment CH2	Disable
	# Transmit signal transition timing adjustment CH3	Disable
	# Transmit signal transition timing adjustment CH4	Disable
	# Transmit signal transition timing adjustment CH5	Disable
	# Transmit signal transition timing adjustment CH6	Disable
	# Transmit signal transition timing adjustment CH7	Disable
	# Transmit signal transition timing adjustment CH8	Disable
	# Transmit signal transition timing adjustment CH9	Disable
	# Transmit signal transition timing adjustment CH10	Disable
	# Transmit signal transition timing adjustment CH11	Disable
	✓ I Resources	
	V 🗐 SCI	
	✓	
	👟 SCK0 Pin	Used
	NRXD0/SMISO0/SSCL0 Pin	Used
	🛰 TXD0/SMOSI0/SSDA0 Pin	Used
	CTSO#/RTSO#/SSO# Pin	Used
	✓	
	🛰 SCK1 Pin	Used
	RXD1/SMISO1/SSCL1 Pin	Used
	TXD1/SMOSI1/SSDA1 Pin	Used
	CTS1#/RTS1#/SS1# Pin	Used

Now in the 'r_byteq' components. At least, two queue control blocks are required for easyDSP. In case you don't use circular buffer in the 'r_sci_rx' component, set the 'Use disable interrupt to protect queue' as 'Unused'.

In case you use circular buffer, then set as 'Used'.

Software component configu	ration	🐻 Generate Code
Components 🚵 🛃 🎝 🕒 🕂 🚔	Configure	
type filter text	Property Configurations Parameter check Memory allocation for queue control blocks Number of static queue control blocks Use disable interrupt to protect queue Use disable interrupt to protect critical section	Value Use system default Static memory allocation 2 Unused Unused

In the 'r_bsp' component, set 'Processor Mode' as 'Stay in Supervisor mode'. ١ Software component configuration Generate Code Components 🚵 🛃 😓 🖃 🛱 Configure to, Property Value type filter text ✓ ☺ Configurations # User stack setting 2 stacks 🗸 🗁 Startup ۸ # User stack size 0x400 🗸 🗁 Generic # Interrupt stack size 0x100 💣 r_bsp # Heap size 0x400 🗸 🗁 Drivers # Initializes C input and output library functions Enable Communications Use BSP charget() function # Enable user stdio charget function 💱 r_sci_rx # User stdio charget function name ✓ → Middleware # Enable user stdio charput function Use BSP charput() function 🗸 🗁 Generic # User stdio charput function name 🔮 r_byteq # Processor Mode Stay in Supervisor mode # ID code 1 0xFFFFFFFF # ID code 2 0xFFFFFFFF # ID code 3 0xFFFFFFFF # ID code 4 0xFFFFFFFF

RXD1 and TXD1 pins are allocated in the 'Pins' tab. Please set 'Assignment' column so that it matchs with <u>the hardware setting</u>. Please check the MCU datasheet to allocate 'Pin Number' column.

Pin configuration	Ge	nerate Code
Hardware Resource $\mathbb{H} = \downarrow_{\mathbb{Z}}^{a} \mathbb{H}$ Pin Function		ି ଓ 🔣
Type filter text (* = any string, ? = any character)		All
All Enabled Function Assignment	Pin Number	Direction
Clock generator	Not assigned	None
Voltage detection circuit RTS1# / Not assigned	Not assigned	None
Clock trequency accuracy measurem RXD1 / P30/MTIOC4B/POE8#/TMRI3/RXD1/SMISO1/SSCL1/TS2/IRQ0	/ 20	1
SCK1 / Not assigned	Not assigned	None
Multi-function timer pulse unit 2 SMISO1 / Not assigned	Not assigned	None
SMOSI1 / Not assigned	Not assigned	None
SS1# / Not assigned	Not assigned	None
Senal communications interface SSCL1 / Not assigned	Not assigned	None
SSDA1 / Not assigned	Not assigned	None
TXD1 / P26/MTIOC2A/TMO1/TXD1/SMOSI1/SSDA1/TS4	/ 22	0
Pin Function Pin Number		

Finally generate code.



STEP 2 : Calling easyDSP_init()

Two files are provided for easyDSP communication (easyRX.h and easyRX.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\RX). First choose the baudrate of SCI communication to easyDSP. Also note it should be same to what you set in the easyDSP project setting.

Then please call the easyDSP_init() function in the main.c.

The priority level of SCI interrupt easyDSP uses is the lowest one (IPL[3:0] = 1). The priority level of the other user interrupt should be set higher than this.

STEP 3 : IDE setting

1. The output file easyDSP uses should have DWARF debugging information. Therefore when using CC-RX compiler, the output file with DWARF debugging information should be created in every compiling time. This is actually done as a default in e2 studio.

2. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Please set your IDE accordingly to create hex file in every compilation.

For example, if you use e2 studio IDE with GCC :



Or if you use e2 studio with CC-RX :

Properties for RX66N	— — ×	(
type filter text	Settings $\diamondsuit \star \dashv \star$	000
 > Resource Builders > C/C++ Build Build Variables Environment Logging Settings Stack Analysis Tool Chain Editor > C/C++ General Project Natures 	Settings Image: Configuration: Configuration: HardwareDebug [Active] Image: Configuration: Image: Configurations Image: Configuration: <td>×</td>	×
Project References Renesas QE Run/Debug Settings Task Tags > Validation	> The second	

3. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

4. The declared but unused variables could be excluded from the debug information depending on compiler's optimization level and linker option. In this case, you can't monitor this variable with easyDSP. If necessary, you can set the linker option so that the unused variables are not excluded. 5. easyDSP supports the little endian mode only.

For example, if you use e2 studio with GCC :

easyDSP help

Properties for RX66N_GCC			—	×
type filter text	Settings		← •	⇔ - 8
✓ Resource				
Linked Resources				
Resource Filters	Configuration: Hardware	ebug [Active]	Manage Configure	ations
Builders				
✓ C/C++ Build				
Build Variables	🛞 Tool Settings 🛞 Tool	chain 🛞 Device 🎤 Build Steps 🤇	ጕ Build Artifact 🛛 🗟 Binary Parser	rs 🔹
Environment				
Logging	CPU	Make the double data type be 64	bits in size (-m64bit-doubles)	
Settings	Optimization	CPU Type	RX66N	\sim
Tool Chain Editor	Debug		PY 2	_
> C/C++ General	🖄 Warnings	Architecture	RXv3	~
Project Natures	V 🛞 Library Generator	MTFU	None	\sim
Project References	🖄 Settings			_
Renesas QE	Compiler	Data Endian	Little-endian data	~
Run/Debug Settings	Source 🖉	Registers reserved for fast interrupt	Default	\sim
Task Tags	Includes			
> Validation	🖄 List	Max size of constant operand value	s Default	\sim

Or, if you use e2 studio with CC-RX :



7.12 TX

TX setting

STEP 1 : Hardware

easyDSP uses MCU's single boot mode to access the flash memory. So the SIO/UART channel that is used in the single boot mode should be used for easyDSP.

Otherwise, easyDSP can support only monitoring, not flash writing.

Please kindly check the datasheet of target MCU to identify which SIO/UART channel and which port pins are used in the single boot mode and connect them to easyDSP pod.



For example, below datasheet capture for TMPM370FY indicates : /BOOT of easyDSP pod should be connected to PF0 of MCU. TX of easyDSP pod should be connected to PE0 of MCU. RX of easyDSP pod should be connected to PE1 of MCU.

19.2.6 Interface specification

In Single Boot mode, an SIO channel is used for communications with a programming controller. The same configuration is applied to a communication format on a programming controller to execute the on-board programming. Both UART (asynchronous) and I/O Interface (synchronous) modes are supported. The communication formats are shown below.

UART communication

Communication channel : SIO channel 0

Serial transfer mode : UART (asynchronous) mode, half -duplex, LSB first

Data length : 8 bits Parity bit : None STOP bit : 1 bit

Baud rate : Arbitrary baud rate

• I/O Interface mode

Communication channel : SIO channel 0

Serial transfer mode : I/O interface mode, full -duplex, LSB first

Synchronization clock (SCLK0) : Input mode

Handshaking signal : PE4 configured as an output mode

Baud rate : Arbitrary baud rate

Table 19-3 Required Pin Connections

Pin		Interface	
		UART	I/O Interface Mode
	DVDD5	0	0
	DVDD5E	0	0
	DVSS	0	0
	AVDD5A	0	0
Power supply	AVSSA	0	0
pins	AVDD5B	0	0
	AVSSB	0	0
	VOUT3	0	0
	VOUT15	0	0
	RVDD5	0	0
Mode-setting pin	BOOT (PF0)	0	0
Reset pin	RESET	0	0
	TXD0 (PE0)	0	0
Communication pin	RXD0 (PE1)	0	0
	SCLK0 (PE2)	x	o (Input mode)
	PE4	×	o (Output mode)

Other considerations :

- DVDD could be either DVDD3 or DVDD5 depending MCU type.

- In case there is a reset IC between easyDSP /RESET and MCU -RESET, it should transfer easyDSP /RESET signal to MCU -RESET within 0.5sec.

- In case pull-up resistor is attached, resistor value should be higher than several k Ohm.

STEP 2 : Modification of easyTX.h file

Two files are provided for easyDSP communication (easyTX.h and easyTX.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\TX_TXZ). Since Peripheral Driver library from the MCU supplier are used in the files, this library should be included in your project.

First, include *_gpio.h and *_uart.h according to MCU. Also based on the hardware connection above, set the channel number and its port. Below example is made based on TMPM370. You should modify it according to target MCU. Finally set the baudrate of easyDSP communication. The baud rate should be same to that of easyDSP project.

```
// step 1 : change header files(*_gpio.h and *_uart.h) according to MCU
11
       and set the SIO/UART channel number and its port pins.
// for TMPM370 : SIO/UART channel 0 /w PE0 and PE1 port
#include "tmpm370 gpio.h"
#include "tmpm370 uart.h"
#define EZ UARTx CH
               0
               GPIO_PE
#define EZ_SIO_PORT
#define EZ_TXD_BIT_NUM GPIO_BIT_0
#define EZ RXD BIT NUM
                GPIO BIT 1
// step 2 : set the baud rate for SIO/UART communication with easyDSP
11
       it should be same to the baudrate of easyDSP project
```

STEP 3 : Calling easyDSP_init()

Pleae include easyTX.h in the top of main.c and call easyDSP_init() in the main() after the initialization of others.

```
#include "easyTx.h"
int main(void)
{
    easyDSP_init();
    while (1) {
      }
}
```

STEP 4 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file (ex *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'hex'. Pleae set your IDE accordingly to create hex file in every compiling time.

2. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded.

4. To compile inline functions in the easyTx.c, please enables c99 mode in the compiler options if required.

7.13 TXZ3

STEP 1 : Hardware

easyDSP uses MCU's single boot mode to access the flash memory. So the UART0 channel (PA1/PA2) that is used in the single boot mode should be used for easyDSP.

Otherwise, easyDSP can support only monitoring, not flash writing. Also the source file easyTXZ3.c should be modified accordingly by you.



Other considerations :

- DVDD could be either DVDD3 or DVDD5.

- In case there is a reset IC between easyDSP /RESET and MCU -RESET, it should transfer easyDSP /RESET signal to MCU -RESET within 0.5sec.

- In case pull-up resistor is attached, resistor value should be higher than several k Ohm.

STEP 2 : Modification of easyTXZ3.h file

Two files are provided for easyDSP communication (easyTXZ3.h and easyTXZ3.c). Please include them in your project. You can find them in the easyDSP installation folder (\source\TX_TXZ). Since Peripheral Driver library from the MCU supplier are used in the files, this library should be included in your project.

First, include the CMSIS header file according to target MCU.

Finally set the baudrate of easyDSP communication. The baud rate should be same to that of easyDSP project.

STEP 3 : Calling easyDSP_init()

Pleae include easyTXZ3.h in the main.c and call easyDSP_init() in the main() after the initialization of others.

```
#include "easyTXZ3.h"
int main(void)
{
    .
    .
    .
    easyDSP_init();
    while (1){
    }
}
```

STEP 4 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file (ex *.elf) with same file name. The hex file extension could be either 'hex' or 'ihex'. easyDSP first check if the hex file with extension 'hex' exists and use it for flash programming. If the hex file with extension 'hex' doesn't exist, easyDSP uses the hex file with extension 'ihex'. Pleae set your IDE accordingly to create hex file in every compiling time.

2. For easyDSP to access the variable, the debug information should be included in the output file (ex, *.elf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded.

4. To compile inline functions in the easyTXZ3.c, plase enables c99 mode in the compiler options if required.

7.14 LPC LPC1x00 setting

STEP 1 : Hardware

easyDSP uses MCU's USARTO channel to communicate with MCU and program flash. So check the below hardware connection by MCU type.



For LPC1500, pin by different MCU package is shown below.

ISP pin	LQFP48	LQFP64	LQFP100
ISP_0	PIO0_4	PIO1_9	PIO2_5
ISP_1	PIO0_16	PIO1_11	PIO2_4
U0_TXD	PIO0_15	PIO0_18	PIO2_6
U0_RXD	PIO0_14	PIO0_13	PIO2_7

For LPC1800, make sure that OTP memory is not programmed or the BOOT_SRC bits are all zero so that the boot mode is determined by the states of the boot pins P2_9, P2_8, P1_2, and P1_1.

Other considerations : TX and RX pin of easyDSP header is pulled up with 100k Ohm resistor inside of easyDSP pod.

STEP 2 : Use of LPCOpen library

easyDSP implements USART communication with MCU by using <u>NPCOpen</u> library. Therefore this library should be included in the user program.

STEP 3 : easyDSP source and header file

Two files are provided for easyDSP communication (easyLPC1x00_va.b.h and easyLPC1x00_va.b.c). Depending on its version, a and b are changeable. You can find them in the easyDSP installation folder (\source\LPC).

Please include them in your project according to target MCU.

In the header file, please set a target MCU or MCU package or baudrate of easyDSP communication. The baud rate should be same to that of easyDSP project.

It differs by target MCU series. Please refer to below for LPC1500.

```
// Package Selection : Please choose MCU package and define it as 1. set as 0 for others.
#define EZDSP_LQFP48
              0
#define EZDSP_LQFP64
              1
#define EZDSP LQFP100
              0
......
// step 2 : set the baud rate for USART communication with easyDSP
     it should be same to the baudrate of easyDSP project
11
115200L
#define EZDSP BAUDRATE
```

STEP 4 : Calling easyDSP_init() function

Please include easyLPC1x00_va.b.h in the main.c. And in the main(), call easyDSP_init() after the initialization of MCU.

In the easyDSP_init() function, all necessary setting for easyDSP monitoring are done.

STEP 5 : IDE setting

1. Hex file (Intel format) is used for flash programming. So it should be created in every compiling time in the same folder to output file (for example, *.axf) with same file name. Pleae set your IDE accordingly to create hex file in every compilation.

For example, if you use MCUXpresso IDE, register arm-none-eabi-objcopy -O ihex "\${BuildArtifactFileName}" "\${BuildArtifactFileBaseName}.hex" in the Post-build steps.

2. For easyDSP monitoring, the debug information should be included in the output file (for example, *.axf). And the option of assembler, compiler and linker should be set accordingly.

3. The unused variables could be excluded from the debug information depending on compiler's optimization level and linker setting. If necessary, you can set the linker option so that the unused variables are not excluded.

7.15 Cautions

* Reset pin of MCU

Don't connect or disconnect easyDSP pod during MCU operation. It could cause any unintentional reset to MCU. In case you have to connect, please connect easyDSP to PC first, then to MCU. In case you have to disconnect, please disconnect easyDSP from MCU first, then from PC. For your reference, reset signal is driven to low by easyDSP during 500msec. Therefore you can add enough filters to the reset pin of MCU.

* What is proper baud rate ?

Normally higher baud rate means faster communication. But MCU should be able to handle this much high baud rate data communication. For example, it takes around 86usec (1/115200bps*10bit) for easyDSP to send one byte to MCU at 115200bps baud rate. MCU should process this one byte data within next 86usec for proper communication. If higher prioritized routine takes most of time and very small time is left for ISR routine for SCI, then easyDSP fails its communication and display the value of variables as '?'.

* Various IDE

For Arm MCU, easyDSP is designed for a wide range of software integrated development environments (IDEs) but is not fully tested for all IDEs. If not working properly, report it to <u>easyDSP@gamil.com</u>.

* Variable is not displayed

Depending of options of compiler and linker, the variable could be not displayed in the easyDSP if this variable is not used meaningfully in the user program.

This variable is not displayed in the map file too. Or displayed but with its address 0. To display it in the easyDSP, please change the compiler/linker option accordingly.

8. Menus

8.1 Project

🎆 <u>N</u> ew	Ctrl+N	
🛃 Open	Ctrl+O	
🖉 Set & Save	Ctrl+S	
Save as		
<u>C</u> lose		
<u>D</u> elete		
1 2807x_BitField_FLAS	1 2807x_BitField_FLASH.ezd	
2 2838xD_cm_DriverL	2 2838xD_cm_DriverLib_FLASH.ezd	
<u>3</u> WB15 HAL flash del	ete later.ezd	
4 C:\Users\\G031 LL	4 C:\Users\\G031 LL flash.ezd	
<u>5</u> WB15 HAL flash.ezd		
<u>6</u> C:\Users\\WB15 L	6 C:\Users\\WB15 LL.ezd	
7 28002x_DriverLib_FLASH.ezd		
8 2837xS_BitField_FLA	SH.ezd	
E <u>x</u> it		

easyDSP deals with your working files with the project concept. The menus belongs to 'Project' menus are

'New' menu:

Clicking 'New' menu shows the dialog box where you can select the name of project file. The extension of project file should be "ezd".

🐯 New pro	ject file	\times
Look in:	test 🔽 🗢 🛍 📸 🛛	
	No items match your search.	
File name:	easyDSP.ezd Open	
Files of type:	Project file(*.ezd)	

And then you can set the properties of your project in the property sheet. The property sheet consists of three pages such as 'Basic', 'Hardware' and 'Miscellaneous'.

'Basic' page sets the target MCU and output file (*.out, *.elf, *.axf and *.x).

First set the target MCU. In case of some STM32 MCU, single or dual bank is specified in the MCU name only when bank mode should be specified. That is, there is no bank mode in the STM32 MCU name either when bank mode is fixed (single or dual) in the MCU or when there is no need for understanding bank mode for easyDSP operation.

For some TI C28x MCUs for which debugging model (either COFF or DWARF) should be specified, the combo box for this is shown. The debugging model should be same to that of compiler option. Please note that further improvement or bug fix for coff debugging model is stopped from easyDSP version 9.

Then the output file should be specified. The output file should exist before creating new easyDSP project.

Also except TI C28x with COFF debugging model, the output file should be DWARF debugging information.

Once the project is created, 'Basic' page is not edited any longer.

Project Settings	×
Basic Hardware Miscellaneous	
MCU	
Vendor TI 💌	
Series TMS320F2807x Debugging model (only for TI 28x)	varf 💌
Part number TMS320F28075	
Output File(s)	
CPU1 C:\temp\2807x_BitField.out	
	OK Cancel

In case of multi core MCU, please specify the output files for all the used cores of MCU in the user program. easyDSP uses these files for RAM booting and flash programming. Also specifiy the core

easyDSP is communicating with in the 'Communication with easyDSP' check boxes. In below figure, easyDSP is communicating with CPU1 and CPU2 while CPU1, CPU2, CPU3 and CPU4 is running in the MCU.

Project Settings		×
Basic Hardware	Miscellaneous	
MCU		
Vendor	TI	
Series	AM263x Sitara	
Part number	AM2634	
Grade	Grade O	
Output File(s) —		Communication with easyDSP
CPU1 (R5_0_0	C:₩temp₩cpu1.out	
CPU2 (R5_0_1	C:\temp\cpu2.out	
CPU3 (R5_1_0	C:\temp\Cpu3.out	
CPU4 (R5_1_1	C:\temp\temp\temp\temp\temp	
		OK Cancel

'Hardware' page sets the hardware configuration for easyDSP communication.

'Protocol' : This is disabled menu.

'Baud rate' : This value means baud rate at PC side which should be same to SCI/UARt baudrate of MCU.

'Wait-more time' : During communication with MCU, easyDSP wait for the response from MCU for certain period. This value extends the waiting time. Please set this value 1000 usec as a first step. If the communication fails due to slow response from MCU, please try to increase this value a little step by step (maximum value is 30000usec) until the communication becomes ok.

Project Settings		×
Basic Hardware	/liscellaneous	
Communication Con	figuration	
Protocol	Normal	
Baud rate	115200 💌 bps	
Wait-more time	1500 usec	
		Cancel

'Miscellaneous' page sets the remains.

'Seek ...' function is very useful when you type the variable name in the window (For ex, command window). It recommends candidates for variable name automatically.

'Stop...' function stops communication of easyDSP if the communication fails successively.

'Display printable ...' display not value but character in case either char or unsigned char variable has a value between 0x20 and 0x7F.

'Highlight ...' shows the changed value of variables in yellow background color. External editor : set the editor program to be called in the Tools>Editor menu.

Project Settings	×
Basic Hardware Miscellaneous	
Seek variables name automatically	
Stop communication if it fails successively	
Display printable character if the value of 'char' or 'unsigned char' variable is printable	
☐ Highlight changes in windows	
External <u>e</u> ditor	
	OK Cancel

'Open' menu:

opens the existing project.

'Set & Save' menu :

sets the properties of active project and then save.

'Close' menu:

closes current project.

'Delete' menu :

deletes all files easyDSP created.

easyDSP makes some files either in the project folder or in the folder the output file is located. They are

MCU	in the easyDSP project folder	in the folder where output file is located
Common	project name.ezd : saves properties of project project name.vars : saves information of variables project name.cfg : saves information of the others	
C28x	easyDSP_FlashApiWrapper.out easyDSP_FlashApiWrapper.ou~ easyDSP_FlashApiWrapper.ez.bin : files for flash operation	output file name.ez.bin : RAM booting and flash programming file (Gen2) output file name.ez.hex : flash programming file (Gen3)
PSOC		output file name.ez.cyacd : flash programming file

STM32 TM4C MSPM0 RA / RX PSOC XMC TX(Z) LPC S32	output file name.ez.hex : RAM booting (if doable) and flash programming file
AM2x	output file name.ez.appimage : RAM booting and flash programming file

8.2 Edit

Edit menu

<u>U</u> ndo	Ctrl+Z
Cu <u>t</u>	Ctrl+X
<u>С</u> ору	Ctrl+C
<u>P</u> aste	Ctrl+V

No need to explain ;-)

8.3 MCU

8.3.1 Common

MCU menu



'RAM Booting' menu 'Flash ROM' menu Please check the below. C28x STM32 S32 AM263x TM4C MSPM0 PSoC4 XMC1 XMC4 RA RX TX, TXZ3 LPC

'Reload *.out' menu

reloads output file (*.out, *.elf or *.axf). It comes in handy when you use debugger and easyDSP together or when you uses easyDSP only for communication (not using /RESET and BOOT pin). For the MCU easyDSP doesn't support flash programming such as XMC1, please use this menu to update symbol information whenever the user program is updated (programmed).

'Reset MCU' menu

The /RESET pin of easyDSP pod goes down to low for 500ms to make reset MCU. The /BOOT and BOOT pin of easyDSP pod are inactive : no signal output from them. For the MCU easyDSP doesn't support flash programming such as XMC1, this menu is not disabled.

/RESET



'Reset Communication' menu

It initializes the states of ISR for easyDSP.

'Pause(Resume) Communication' menu

It pauses the communication of easyDSP. This menu toggles into 'Resume Communication' menu.

'Communication Status' menu

It displays the target MCU of easyDSP pod and communication state such as read/write fail/success ratio. Over 90% of success ratio is mandatory to have fluent communication.



8.3.2 C28x

MCU menu (TI C28x)

್ಗ	RAM Booting	Alt+R
(m)	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	nication
?	Communicatio	on Status

'RAM Booting' menu

is for booting to RAM area only (NO flashrom area). During RAM booting, communications in all windows are temporarily paused.

RAM booting for TM	IS320F28x			×
Boot				
Verify				
Status : Verifyin 1sec el	g RAM booting apsed	Completed !		
Enable fast b	ooting	Enable fast verifying)	
Boot	Verify	Boot > Verify	Stop	Exit

'Boot' button starts booting operation. First it is checked if user program is appropriate for RAM booting. If it fails, booting operation stops.

In case the user program is re-compiled in the meantime, easyDSP detects it and asks you whether you will use new program.

Faster action will be tried if you check 'Enable fast' check box.

If 'Enable fast verifying' is not working properly due to limited resource availability, please disable this option.

Below error message during RAM booting indicates DSP didn't get into booting mode due to most likely wrong hardware connection.

RAM booting for TMS320F28x	×
Boot	
Verify	
Status : Reset DSP and trying auto bauding at 62956 bps	Failed !
Enable fast booting Enable fast verifying	
Boot Verify Boot > Verify	Stop Exit

'Verify' button check if the RAM booting was done correctly. If failed during verifying, below message comes out. It means that the data at address 0x240000 is now 0x159D which is supposed to be 0x0x28AD with proper booting.

Status : Verify failed!

Data mismatch @0x240000 : Boot=0x28AD, Read=0x159D

'Boot > Verify' button is doing 'Boot' and 'Verify' button consecutively. 'Stop' button stops any ongoing activity either booting or verifying.

'Flash ROM' menu for Gen2 MCU, F2837xD and F2838x

easyDSP help

On-chip flash programmer for TM	1S320F28x ×
file for flash writing C:\Users\chundaew\cdw\CIP0	DS\New product concept\uA\uA program\Debug\uA.bin
select target 28x device TMS320F2808 - clock 100Mł	nz (20Mhz x 5)
Code Security PasswordKey 00xFFFFKey 10xFFFFKey 20xFFFFKey 30xFFFFKey 40xFFFF	Erase Sector Selection Sector A: [3F4000-3F7FFF] Sector B: [3F0000-3F3FFF] Sector B: [3F0000-3F3FFF] Sector C: [3E0000-3EFFFF] Sector C: [3E0000-3EFFFF] Sector D: [3E8000-3EFFFF] Sector E: [3E4000-3E7FFF] Sector E: [3E4000-3E7FFF] Select None Select Used Select Not Used
Key 5 0xFFFF Key 6 0xFFFF	Operation Erase > Program > Reset > Exit
Key 7 0xFFFF Unlock	Erase > Program > Verify > Reset > Exit Erase Program Verify STOP !!! Reset > Exit
Checking bin file OK!	

On-chip flash programmer for TMS320F28377D		
Flash API speed [bps] 460800 -		EXIT
Code Security Password	-Erase or Blank Check Sector Sel	ection Freeze
	CPU1	CPU2
CPU1 CPU2	Sec A : [80000-81FFF]	Sec A : [80000-81FFF]
Z1 KEY0 0xFFFFFFF 0xFFFFFFF	Sec B : [82000-83FFF]	Sec B : [82000-83FFF]
	Sec C : [84000-85FFF]	Sec C : [84000-85FFF]
	Sec D : [86000-87FFF]	Sec D : [86000-87FFF]
Z1 KEY2 0xFFFFFFF 0xFFFFFFF	Sec E : [88000-8FFFF]	Sec E : [88000-8FFFF]
	Sec F : [90000-97FFF]	Sec F : [90000-97FFF]
	Sec G : [98000-9FFFF]	Sec G : [98000-9FFFF]
Z2 KEY0 0xFFFFFFF 0xFFFFFFF	Sec H : [A0000-A7FFF]	Sec H : [A0000-A7FFF]
72 KEY1 0xFFFFFFF 0xFFFFFFF	Sec I : [A8000-AFFFF]	Sec I : [A8000-AFFFF]
	□ Sec J : [B0000-B7FFF]	Sec J : [B0000-B7FFF]
Z2 KEY2 0xFFFFFFF 0xFFFFFFF	Sec K : [B8000-B9FFF]	Sec K : [B8000-B9FFF]
Z2 KEY3 0xFFFFFFF 0xFFFFFFF	Sec L : [BA000-BBFFF]	Sec L : [BA000-BBFFF]
	Sec M : [BC000-BDFFF]	Sec M : [BC000-BDFFF]
	Sec N : [BE000-BFFFF]	Sec N : [BE000-BFFFF]
Unlock	Select None Select Use	d Select Not Used Select All
Operation		
Erase > Program > Reset > Exit	Erase > Pro	ogram > Verify > Reset > Exit
Erase Blank Check Pro	gram Verify	STOP !!! Reset > Exit
Status		
CPU1 : Checking the validity of hex fileOK CPU2 : Checking the validity of hex fileOK		

On-chip flash programmer for TMS320F28	388D CPU1 CPU2 CM		
Flash API Speed [bps] 499200	-		FYIT
433200		Ereeze	
Code Security Password	Sector Selection for Erase or B	lank Check	
	CPU1	CPU2	CM
	Sec A : [80000-81FFF]	Sec A : [80000-81FFF]	Sec A : [200000-203FFF]
Z1 CSMPSWD0 0xFFFFFFF	Sec B : [82000-83FFF]	Sec B : [82000-83FFF]	Sec B : [204000-207FFF]
Z1 CSMPSWD1 0x4D7FFFFF	□ Sec C : [84000-85FFF]	Sec C : [84000-85FFF]	Sec C : [208000-20BFFF]
	Sec D : [86000-87FFF]	Sec D : [86000-87FF]	Sec D : [20C000-20FFFF]
Z1 CSMPSWD2 UXFFFFFFF	Sec E : [88000-8FFFF]	Sec E : [88000-8FFFF]	Sec E : [210000-21FFFF]
Z1 CSMPSWD3 0xFFFFFFFF	Sec F : [90000-9/FFF]	Sec F : [90000-97FFF]	Sec F : [220000-22FFFF]
	Sec G : [98000-9FFFF]		Sec G : [230000-23FFFF]
			Sec H : [240000-24FFFF]
Z2 CSMPSWD1 0x1F7FFFFF			Sec 1 : [250000-25FFFF]
72 CSMPSWD2 0xEEEEEEE	Sec K : [B8000-B9EEE]	Sec 5 [B8000-B7FF]	Sec J : [250000-26FFFF]
	Sec L [BA000-BBEEF]	Sec L : [BA000-BBEEE]	Sec K : [270000-275FF]
Z2 CSMPSWD3 0xFFFFFFF	Sec M (BC000-BDEEE)	Sec M [BC000-BDFFF]	Sec M : [278000-278EEE]
	Sec N : [BE000-BFFFF]	Sec N : [BE000-BFFFF]	Sec N : [270000-276FFF]
Unlock			
	Select Used	Select Not Used Select All	Select None
- Operation			
Operation		_	1
Erase > Progra	am > Reset > Exit	Erase > Program > 1	Verify > Reset > Exit
Erase Blank	Check Program	Verify	OP !!! Reset > Exit
Status			
CM : Checking the validity of bin file	OK		

It programs onchip flash of MCU. Note that the communication in other windows are temporarily paused.

Please follow below sequence.

step 1 : First select target device according to your MCU and clock configuration. This menu is available for some MCU only.

step 2 : Select the sectors for erasing or blank checking. Either use the buttons or click the checkboxes of sectors.

All sectors used in the user program are selected with 'Select Used' button. The other way around with 'Select Not Used' button.

For some MCUs, Freeze checkbox is provided to enable or disable sector selection. step 3 : When the buttons ('Erase', 'Blank Check', 'Program', 'Verify' or 'Unlock') are pressed first time, easyDSP boots MCU with the agency program (not user program) to handle flashrom manipulation.

If the output file (*.out) is updated meantime, easyDSP ask the user to use update output file or not.

One click for all operations possible (ex. 'Erase > Program > Reset > Exit' button) step 4 : Now MCU is booted and communicates with easyDSP for proper flashrom access. step 5 : when exiting this dialog box, easyDSP forces MCU to be reset. Then MCU boots with flashrom and user program starts.

note) above dialog box looks different depending on the MCU type

note) For 2837xD and 2838xS(D), this will program the supplied data portion in flash along with automatically generated ECC(Error Correction Code).

note) In case below menu is activated, bps of flashAPI wrapper can be selected to reduce flash operation time. Note that certain bps could not work.

This bps value has nothing to do with the bps value used in variable monitoring. So, don't need to match with the bps value in the easyDSP header file and in the project setting.

Flash API speed [bps] 115200 -

'Flash ROM' menu for C2834x series

SPI flash programmer	×
file to be programmed D:\easyDSP\Program\280x\test2808Flash\Debug\Test2808flash.bin	EXIT after RESET
SPI Flash AT25DF021	EXIT
Operation	
Erase chip Erase block Program	Verify
Program > Verify Erase chip > Program > Verify Erase block	ck > Program > Verify
STOP !!!	
0%	

Since 2834x doesn't have internal flash, easyDSP supports external flashs with SPI interface. They are AT25DF021(2M bit), AT25DF041(4M bit), AT26DF081(8M bit), AT25DF321(32M bit), M25P20(2M bit), M25P40(4M bit), M25P80(8M bit), M25P16(16M bit), M25P32(32M bit) manufactured by ATMEL or Numonyx. Other flashs which support same commands and features to above could be operated. There are two kinds 'Erase' function : 'Erase chip' erases all chip memory. 'Erase block' erases only the memory region which will be programmed with user program. Because 'Erase block' uses '4K byte block erasing' feature of ATMEL flash, the memory region to be erased will be normally larger than the actual code size, at the most, 4K bytes.

Please note that easyDSP does 'global unprotect' action to the flash during its operation. Also note that easyDSP sets LOSPCP = 2 and SPIBRR = 0 to control SPI-A boot mode speed.

'Flash ROM' menu for others

This is for Gen.3 single core MCU and Gen.3 multi core MCU like F28Px. It programs onchip flash of MCU with user program. Note that the monitoring of easyDSP is

It programs onchip flash of MCU with user program. Note that the monitoring of easyDSP temporarily paused during flash operation.



Please follow below sequence.

step 1 : If necessary, set the CSM key values and unlock CSM by using 'CSM password' and 'Unlock CSM' buttons

step 2 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

step 3 : When the buttons (Erase, Blank, Program+Verify, Verify) are clicked first time, MCU enters to single boot mode after reset.

step 4 : Execute necessary flash actions.

note) It programs the supplied data portion in flash along with automatically generated ECC(Error Correction Code).

step 5 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

8.3.3 STM32

MCU menu (ST STM32)

್ಗ	<u>R</u> AM Booting	Alt+R
Sand	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
° _x	Pause Commu	inication
?	Communicatio	on Status

'RAM Booting' menu

is for booting to RAM area only (NO flashrom area). During RAM booting, communications in all windows are temporarily paused.

STM32 RAM booting	Х
Operation	
Writing RAM > Boot > Exit Writing RAM > Verifying RAM > Boot > Exit STOP !!! Exit	
Checking memory region OK	

'Boot' button starts booting operation. First it is checked if user program is appropriate for RAM booting. If it fails, booting operation stops.

In case the user program is re-compiled in the meantime, easyDSP detects it and asks you whether you will use new program.

Before action, easyDSP check MCU's bootloader version and display it on the title bar of window.

STM32 RAM booting : BL ID = 0x90	Х
Operation	
Writing RAM > Boot > Exit Writing RAM > Verifying RAM > Boot > Exit STOP !!!	
Writing RAM @ 0x20004FF8 - 0x200050F7	

'Stop' button stops any ongoing activity.

Note that RAM booting is not supported for dual core MCU.

'Flash ROM' menu

It programs onchip flash of MCU with user program. Access to OTP memory, Data memory and option byte is not supported.

Its functionality could be limited with activated Trust Zone or Secure MPU.

Note that the communication in other windows are temporarily paused.

Erase > Program > Reset > Exit	Select	Index	Start Address	Size
		0		
		0	0x08000000	2K
		1	0x08000800	2K
e > Program > Verify > Reset > Exit		2	0x08001000	2K
		3	0x08001800	2K
		4	0x08002000	2K
Erase Erase chip		5	0x08002800	2K
		6	0x08003000	2K
Decement		7	0x08003800	2K
Program		8	0x08004000	2K
		9	0x08004800	2K
Varifi		10	0x08005000	2K
veniy		11	0x08005800	2K
		12	0x08006000	2K
STOP III		13	0x08006800	2K
0101 1		14	0x08007000	2K
		15	0x08007800	2K
Ontion		16	0x08008000	2K
Option		17	0x08008800	2K
Reset > Exit Exit	All	None	e Used	Not Used

Please follow below sequence.

step 1 : Select the flash pages to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkboxes of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 2 : If 'Option...' button is enabled, click it and select the proper option.

For example, you choose (not change) SWAP_BANK bit status for STM32G0B1.

Since the option is saved, you can choose option only when change.

step 3 : When the buttons ('Program', 'Verify', 'Erase' or 'Erase chip') are clicked first time, MCU enters to bootload mode after reset.

step 4 : Execute necessary flash actions.

'Erase chip' erases all the flash in the MCU regardless of selected check box.

step 5 : When exiting this dialog box, use 'Reset > Exit' button. It makes MCU reset and boot with flash. And user program starts.

If you exit this dialog box without MCU reset, MCU still stay in the bootload mode.

8.3.4 S32

MCU menu (NXP S32)
h	RAM Booting	Alt+R
	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
° _x	Pause Commu	inication
?	Communicatio	on Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

NOTE : this menu is working only when EZ_BOOTLOADER_USE is defined as 1 in the easyS32**.h file. For detailes, refer to <u>this page</u>.

It programs onchip flash of MCU with user program. During this operation, the monitoring of easyDSP is temporarily paused. Note that you have to disable any flash related protection feature in the MCU while using this menu.

	-Sector selection for era	sing	Freez
Erase > Program > Reset > Exit	Select	Start Address	Size
	✓ p-flash sector 0	0x00000000	2K
	✓ p-flash sector 1	0x0000800	2K
-	✓ p-flash sector 2	0x00001000	2K
Erase Blank	✓ p-flash sector 3	0x00001800	2K
	✓ p-flash sector 4	0x00002000	2K
1	✓ p-flash sector 5	0x00002800	2K
Program	✓ p-flash sector 6	0x00003000	2K
	✓ p-flash sector 7	0x00003800	2K
	✓ p-flash sector 8	0x00004000	2K
	✓ p-flash sector 9	0x00004800	2K
Verity	✓ p-flash sector 10	0x00005000	2K
	✓ p-flash sector 11	0x00005800	2K
1	✓ p-flash sector 12	0x00006000	2K
STOP	✓ p-flash sector 13	0x00006800	2K
	✓ p-flash sector 14	0x00007000	2K
	✓ p-flash sector 15	0x00007800	2K
	✓ p-flash sector 16	0x0008000	2K
	p-flash sector 17	0x00008800	2K
	p-flash sector 18	0x00009000	2K
	p-flash sector 19	0x00009800	2K
	p-flash sector 20	0x0000A000	2K
	D-flash sector 21	0x0000A800	2K
Reset > Exit Exit	Used Not	Used All	None

step 1 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 2 : When the buttons (Erase, Program, Verify) are clicked first time, MCU enters to bootloade (easyDSP_boot() function) after reset.

step 3 : Execute necessary flash actions. 'Blank' button is disabled.

step 4 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

Note :

To program flash, the bootloader should be provided since there is no ROM bootloader in this MCU. The bootloader easyDSP provides is the function (name : easyDSP_boot) and it resides in the user program. Therefore it can program flash only when it is already programmed in the flash. In case flash is empty or flash doesn't have easyDSP bootloader, you can't enter into the bootloader and will see the message below. In this case, you have to use the debugger to program flash. And in same principle, you have to

use debugger to program easyDSP bootloader into flash at the beginning.



8.3.5 AM263x

್ರ	RAM Booting	Alt+R
- And	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	inication
?	Communicatio	on Status

'RAM Booting' menu

It is for MCU booting to RAM area only (NO flashrom area) by using TI SBL UART mechanism. During RAM booting, easyDSP monitoring in all windows is temporarily paused.

AM263x RAM booting	\times
Application image C:\Users\chundaew\cdw\easyDSP\Program2\TI_AM2x\AM2634_LED_Blinky\Debug\AM2634_LED_Blinky.appimag	е
SBL UART	
SBL image C:\Users\chundaew\cdw\easyDSP\Program2\TI_AM2x\sbl_uart_am263\sbl_uart_am263x-lp_sdk_09_00_00_35.tiimage	
Baudrate 230400	
Operation	
RAM Boot > Exit STOP !!! Exit	
Ready	7

easyDSP doesn't provide SBL UART image file. It provides the method for RAM booting using the given SBL UART image file from either TI or your own.

Please follow below steps.

step 1 : Please select the application image file to be downloaded to RAM. By default, the app image file which easyDSP is using is selected. But you can change it by clicking 'Application image' button. step 2 : Please choose SBL UART image file via 'SBL image' button and then input the baudrate of SBL UART.

If you use the prebuilt SBL by TI (the files located in

C:\ti\mcu_plus_sdk_am263x_09_00_00_35\tools\boot\sbl_prebuilt folder for example), set the baudrate to 115200.

If you use your own SBL UART, set the baudrate according to your own SBL UART.

step 3 : 'RAM Boot > Exit' button starts booting operation. In case the user program is re-compiled in the meantime, easyDSP detects it and asks you whether you will use new program.

'Stop' button stops any ongoing activity.

'Flash ROM' menu

It programs user program to SPI flash. easyDSP monitoring in all windows is temporarily paused and below dialog box appears.

AM263x Flash Programming	×		
Application			
Image file C:\Users\chundaew\cdw\easyDSP\Program2\TI_AM2x\AM2634_LED_Blinky\Debug\AM2634_LED_Blinky.ez.appimage			
SBL UART Uniflash			
Image file C:\ti\mcu_plus_sdk_am263x_09_00_00_35\tools\boot\sbl_prebuilt\am263x-lp\sbl_uart_uniflash.release.tiimage			
Baudrate 115200			
SBL QSPI Image file C:\ti\mcu_plus_sdk_am263x_09_00_00_35\tools\boot\sbl_prebuilt\am263x-lp\sbl_qspi.release.tiimage Application image offset in flash : 0x 80000 Flashing SBL QSPI Verifying SBL QSPI			
Operation			
Flashing > Reset > Exit Flashing Verify Erase STOP !!! Reset > Exit Exit			
Ready			

easyDSP doesn't provide SBL image files themself. It provides the method for downloading SBL and flashing the application using the given SBL image files from either TI (prebuilt SBL) or your own. Please follow below steps.

step 1 : Please select the application image file to be downloaded to SPI flash. By default, the app image file which easyDSP is using is selected. But you can change it by clicking 'image file' button.

Please note that easyDSP generates app image file (file extension = ez.appimage) from *.rprc files created by IDE.

step 2 : Please choose SBL UART Uniflash image file via 'image file' button and then input the baudrate of the SBL.

If you use the prebuilt SBL by TI (the files located in

 $\label{eq:c:ti} C:\ti\mcu_plus_sdk_am263x_09_00_00_35\tools\boot\sbl_prebuilt\ folder\ for\ example),\ set\ the\ baudrate\ to\ 115200.$

If you use your own SBL, set the baudrate according to your own SBL.

step 3 : Please choose SBL QSPI image file via 'image file' button. And set the offset where the app image will be written to SPI flash.

For prebuilt SBL by TI, the offset is 0x80000. For your own SBL, set the offset accordingly. step 4 : Flashing SBL QSPI by clicking 'Flashing SBL QSPI' button. Once done, not required anymore until you change the SBL QSPI.

Once all set until step 4, you don't need to repeat the steps.

step 5 : Execute necessary flash actions by clicking buttons in the 'Operation' area. When the buttons ('Flashing', 'Verify' or 'Erase') are clicked first time, MCU enters to boot mode

after reset and SBL UART Uniflash is downloaded and runs.

Flashing is the successive action of Erase > Program > Verify. So, Erasing or Verifying before/after flashing is optional.

Note that flashing and verifying action is done in 192kB block unit.

step 6 : When exiting this dialog box, use 'Reset > Exit' button. It makes MCU reset and boot with QSPI (4S) - Quad Read Mode. And user program starts.

If you exit this dialog box without MCU reset, MCU still stay in SBL and the easyDSP monitoring will fail.

8.3.6 TM4C

MCU menu (TI TM4C)

FlashROM Alt+F
Ab
🟟 Reload *.out
🔗 Reset MCU
Reset Communication
⁰ _X Pause Communication
? Communication Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program. Note that the monitoring of easyDSP is paused with this menu.



Please follow below sequence :

step 1 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 2 : When the buttons (Erase, Program) are clicked first time, MCU enters to ROM boot loader after reset.

step 3 : Execute necessary flash actions.

step 4 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

Note : Since MCU ROM boot loader doesn't support verify function, easyDSP provides 'Verify (with easyDSP)' button instead. This is verification of flash contents by using easyDSP monitoring, not by ROM boot loader. This button is disabled once MCU enters ROM boot loader.

8.3.7 MSPM0

 RAM Booting Alt+R
 FlashROM Alt+F
 Reload *.out
 Reset MCU Reset Communication
 Pause Communication
 Communication Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs MAIN flash memory region of MCU with user program. Please disable any flash related protection feature in the MCU while using this menu. Since easyDSP can't support NONMAIN flash memory region (such as BCR and BSL configuration area), please use the debugger or any other tool to program NONMAIN flash.

When this menu is activated, the monitoring of easyDSP is temporarily paused.

MSPM0 flash programmer					
Operation Sector Selection to be Erased					
				⊢ Freeze	
Erase > Program > Reset > Exit		Туре	Start Address	Size	
		sector 0	0x00000000	1K	
Eroso > Brogrom > Vorify > Boost > Evit		sector 1	0x00000400	1K	
Liase > Program > Venily > Reset > LXit		sector 2	0x0000800	1K	
		sector 3	0x00000C00	1K	
Frase		sector 4	0x00001000	1K	
Liase		sector 5	0x00001400	1K	
		sector 6	0x00001800	1K	
Program		sector 7	0x00001C00	1K	
		sector 8	0x00002000	1K	
		sector 9	0x00002400	1K	
Verify		sector 10	0x00002800	1K	
		sector 11	0x00002C00	1K	
		sector 12	0x00003000	1K	
STOP !!!		sector 13	0x00003400	1K	
		sector 14	0x00003800	1K	
Reset > Exit Exit	A	II No	one Used	Not Used	
256-bit BSL access password FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF					

Please follow below sequence.

step 1 : Set the 32 bytes password to enter bootstrap mode. It is all 0xFF at TI production state. If you set them in SysConfig like below,

Bootstrap Loader (BSL) Configuration	~
BSL Access[0]	0x11223344
BSL Access[1]	0x55667788
BSL Access[2]	0xAABBCCDD
BSL Access[3]	0xDEADFACE
BSL Access[4]	0xFFFFFFF
BSL Access[5]	0xFFFFFFF
BSL Access[6]	0xFFFFFFF
BSL Access[7]	0xFFFFFFF

you can input like below.

256-bit BSL access password

step 2 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 3 : When the buttons (Erase, Program, Verify) are clicked first time, MCU enters to bootstrap mode after reset.

step 4 : Execute necessary flash actions.

'Verify' button acts differently depending on the 'BSL Read Out Enable' value in the SysConfig > BSL Configuration tab.

If read out is disabled (like TI factory default), it checks 1024 bytes CRC without reading the flash memory.

If read out is enabled, it reads the flash memory.

step 5 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

8.3.8 PSoC4

2	<u>R</u> AM Booting	Alt+R
Sim	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	inication
?	Communicatio	on Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program only for single-application bootloader configuration. Note that the monitoring of easyDSP is temporarily paused.

PSoC flash programmer : Silicon ID = 0x195C11A9, Silicon Rev = 0x00, Bootloader Ver = 0x01013C							
Operation Page Selection to be Erased							
Erase + Program > Start User Program			Array	Row	Start Address	Size	^
			0	0	0x00000000	128	
1			0	1	0x0000080	128	
E + P > Verify > Start User Program			0	2	0x00000100	128	
			0	3	0x00000180	128	
			0	4	0x00000200	128	
Erase			0	5	0x00000280	128	
			0	6	0x0000300	128	
			0	7	0x0000380	128	
Erase + Program			0	8	0x00000400	128	
			0	9	0x00000480	128	
			0	10	0x00000500	128	
Verify			0	11	0x00000580	128	
			0	12	0x0000600	128	
STOP !!!]	0	13	0x0000680	128	
]	0	14	0x00000700	128	
]	0	15	0x00000780	128	×
Start User Program All None Used Not Used							
Bootloader Security Key							
□ Use security key 0x □		_					
Verifying flash @ array=0, row=255 (0x7F80-0x7 2sec elapsed	7FFF)		OK				

Please follow below sequence.

step 1 : Select the flash array and row to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

step 2 : If bootloader security key is used, please input the key value after clicking 'Use security key' button.

step 3 : When the buttons (Erase, Erase+Program, Verify) are clicked first time, MCU enters to bootloader mode after reset.

Also silicon ID, selicon revision, bootloader version is displayed in the title bar.

step 4 : Execute necessary flash actions.

step 5 : Click 'Start User Program' button when exiting this dialog box. It makes MCU reset and user program starts.

note : erasing the flash where bootloader program is located is not enabled.

8.3.9 XMC1

MCU menu (Infineon XMC1)

7,	RAM Booting	Alt+R
Sand	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	inication
?	Communicatio	on Status

RAM Booting menu

This menu is disabled.

Flash ROM menu

This menu is disabled.

8.3.10 XMC4

MCU menu (Infineon XMC4)

h	<u>R</u> AM Booting	Alt+R
Sin	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
° _x	Pause Commu	nication
?	Communicatio	on Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program. Note that the monitoring of easyDSP is temporarily paused.

easyDSP help



Please follow below sequence.

step 1 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

step 2 : If necessary, use write-protection.

step 3 : When the buttons (Erase, Program+Verify, Verify) are clicked first time, MCU enters to bootloader mode after reset.

step 4 : Execute necessary flash actions.

step 5 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

NOTE) programming to not erased sector may causes malfunction.

8.3.11 RA

MCU menu (Renesas RA)

h	<u>R</u> AM Booting	Alt+R
Sand	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	inication
?	Communicatio	on Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program. Note that flash programming is not supported for RA0 series.

Note that the programming would be not available in case security or protection is set to the memory. When clicked, the monitoring of easyDSP is temporarily paused and dialog will be present as below.

peration	Sec	tor Selec	tion to t	e Erased	Freeze
Erase > Program > Reset > Exit		Туре	Block	Start Address	Size
		code	0	0x00000000	2K
France & Drammary & Varifie & Danat & F		code	1	0x0000800	2K
Erase > Program > Verity > Reset > E.	×II 🛛	code	2	0x00001000	2K
		code	3	0x00001800	2K
Fraça		code	4	0x00002000	2K
Elase		code	5	0x00002800	2K
		code	6	0x00003000	2K
Program		code	7	0x00003800	2K
		code	8	0x00004000	2K
		code	9	0x00004800	2K
Verify		code	10	0x00005000	2K
		code	11	0x00005800	2K
		code	12	0x00006000	2K
STOP !!!		code	13	0x00006800	2K
		code	14	0x00007000	2K
Reset > Exit E:	xit	All	None	Used	Not Used
rotection (used when required) D code (32 hex characters, MSB first) :					
hecking the validity of hex fileOK					

Please follow below sequence.

step 1 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the

checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

Note that erasing of option flash is not performed since it is not necessary.

step 2 : When the buttons (Erase, Program+Verify, Verify) are clicked first time, MCU enters to bootmode after reset.

For MCU without DLM(Device Lifecycle Management), ID code will be used to unlock MCU if required.

For MCU with DLM, DLM state transition is not supported.

step 3 : Execute necessary flash actions.

step 4 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

8.3.12 RX

MCU menu (Renesas RX)

7,	<u>R</u> AM Booting	Alt+R
Sand	<u>F</u> lashROM	Alt+F
ф	Reload *.out	
0	Reset MCU	
	Reset Commu	nication
°x	Pause Commu	inication
?	Communicatio	on Status

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program except the protected area by area protection or trusted memory.

Therefore please disable any flash related protection feature in the MCU while using this menu. When this menu is activated, the monitoring of easyDSP is temporarily paused.



Please follow below sequence.

step 1 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 2 : When the buttons (Erase, Program+Verify, Verify) are clicked first time, MCU enters to bootmode after reset.

step 3 : Execute necessary flash actions.

step 4 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

If boot mode ID code protection is enabled in the MCU, MCU enters to boot mode only when ID code you input matches.

If you set the ID code like below in the Smart Configurator, please set the ID code in the flash dialog as above.

Software component configuration				
Compon 🖄 🛃 🍃 🕂	Configure			
😺 ᢏ	Property ~ @ Configurations	Value		
V 🗁 Startup	# User stack setting # User stack size	2 stacks 0x1000		
✓ (⇒ Generic	# Interrupt stack size	0x400		
✓ → Drivers ✓ → Communications	 Initializes C input and output library functions 	Disable		
🔮 r_sci_rx	Enable user stdio charget function User stdio charget function name	Use BSP charget() function my_sw_charget_function		
	# Enable user stdio charput function # User stdio charput function name	Use BSP charput() function		
🍟 r_byteq	# Processor Mode	Stay in Supervisor mode		
	# ID code 1 # ID code 2	0x45010203 0x04050607		
	# ID code 3 # ID code 4	0x08090A0B 0x0C0D0E0F		

Note :

1. All the flash contents are erased before entering to boot mode if the control ID is neither 0x45 nor 0x52 for RX100 and RX200 MCU series.

2. For RX64M, RX660, RX66T, RX71M and RX72T series, programming of option setting memory is not supported.

8.3.13 TX, TXZ3

 RAM Booting
 Alt+R

 FlashROM
 Alt+F

 Reload *.out
 Alt+F

 Reset MCU
 Reset Communication

 Pause Communication
 Reset Communication

 Communication
 Communication

RAM Booting menu

This menu is not supported.

Flash ROM menu

It programs onchip flash of MCU with user program. Note that the monitoring of easyDSP is temporarily paused.

easyDSP help



Please follow below sequence.

step 1 : By clicking 'Password' button, set the password which is required to enter single boot mode.

For TX series, input 12 bytes value	(default = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF) in below	dialog box.
Password		×	
Password for boot mode (12 bytes, hex)	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF	ОК	

For TXZ3 series, input related values in below dialog box.

easyDSP help

Password for TXZ	Х
Password length in bytes (8 to 255) 12	
Address at the password length is located 0x 0001F000	_
Address at the password is located 0x 0001F001	_
Password (hex) 0102030405060708090A0B0C	-
Erase Chip OK	
	2

step 2 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

step 3 : When the buttons (Erase, Program+Verify, Verify) are clicked first time, MCU enters to single boot mode after reset.

step 4 : Execute necessary flash actions.

step 5 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

Note) programming to not erased sector may causes malfunction.

Note) Blank and Protect buttons are disabled.

8.3.14 LPC

MCU menu (NXP LPC1x00)



RAM Booting menu

This menu is disabled.

Flash ROM menu

It programs onchip flash of MCU with user program. During this operation, the monitoring of easyDSP is temporarily paused.

Note that you have to disable any flash related protection feature in the MCU while using this menu.

eration	Se	ctor Sel	ection to I	be Erased	□ Free
Erase > Program > Reset > Exit		Bank	Sector	Start Address	Size
		A	0	0x1A000000	8K
		A	1	0x1A002000	8K
Erase > Program > Verify > Reset > Exit		A	2	0x1A004000	8K
		A	3	0x1A006000	8K
Free		A	4	0x1A008000	8K
Erase		A	5	0x1A00A000	8K
		A	6	0x1A00C000	8K
Drogrom		A	7	0x1A00E000	8K
Program		A	8	0x1A010000	64K
		A	9	0x1A020000	64K
Verify		A	10	0x1A030000	64K
veniy		A	11	0x1A040000	64K
		A	12	0x1A050000	64K
STOP III		A	13	0x1A060000	64K
8101		A	14	0x1A070000	64K
		B	0	0x1B000000	8K
Report > Evit Evit		B	1	0x1B002000	8K
		B	2	0x1B004000	8K
		B	3	0x1B006000	8K
		I R	4	0x1B008000	8K
tive Flash Bank (if flash has two banks)——		All	None	Used	Not Use
Bank A C Bank B					
ecking the validity of hex fileOK					

Please follow below sequence.

step 1 : In case MCU has two flash banks (for example, part of LPC1800 series), select the active flash bank where your program will run after reset.

step 2 : Select the flash sector to be erased. Use 'All', 'None', 'Used', 'Not Used' buttons. Or click the checkbox of sectors.

All the sectors used in the user program are selected with 'Used' button. The other way around with 'Not used' button.

Freeze checkbox disables the sector selection.

step 3 : When the buttons (Erase, Program+Verify, Verify) are clicked first time, MCU enters to bootmode after reset.

step 4 : Execute necessary flash actions.

step 5 : Click 'Reset>Exit' button when exiting this dialog box. It makes MCU reset and user program starts.

8.4 Tools

🔄 Open Project Folder
Editor
Calculator

You can use various tools.

'Open Project Directory' : opens the folder of the active project. 'Editor': runs the editor which you set before in project settings. 'Calculator' : runs the calculator of Windows.

8.5 Window



Opening/closing/arranging windows.

8.6 Help

Help menu

🦹 <u>H</u> elp
Help(Korean)
<u>A</u> bout easyDSP

'Help...' : Opens this help file 'About easyDSP...' : basic information of easyDSP

9. Windows

9.1 Command



Command window is designed for writing or reading variables. The available commands are found by typing 'help' command. All commands are executed by enter-key input.



^{**} Update the block. If you select the block of commands by dragging mouse and then press this button, the commands which belong to the block are updated. If no block selected, the current line is updated. You can do it by clicking right button of mouse.

^{=!} Run the block. If you select the block of commands by dragging mouse and then press this button, the commands which belong to the block are executed. If no block selected, the current line is executed.

Insert new line without running command. When you press enter key, the corresponding line is running as command. If you use this tool button, new line is inserted without running command. Same as 'Ctrl-Enter' key press.

Read command file. Copy the file to the command window. No execution. You can also type 'r filename' in the command window.

Load command file. command file is the set of commands. Frequently used commands can be saved into command file and then use this function. You can also type 'I filename' in the command window.

Save the block of lines to file. Save the selected block of commands as a file. Afterwards, you load this file by 'l' commands.

* Commands :

Caution:

- 1: The number of character in one command line should not exceed 300
- 2: all commands should be small-character

Decimal system	
dec var	decimal display (default)
hex var	hex-decimal display
bin var	binary display

Assign and Display

1	
var =	display the value of var
&var =	display the address of var
*var =	In case var is pointer to basic type, display the value of variable pointed by pointer var note) not supported for Arm MCU
(*var).x =	In case var is pointer to struct/union type, display the value of variable x pointed by pointer var note) not supported for Arm MCU
var = number	assign value to var 0x***(hex-decimal) form is supported If var is float type, following dimension form is supported 3e-3, 23K, 23m, 0.34p For dimension usage, refer to 'watch window'
var1 = &var2	assign the address of var2 into var1 var1 should be int type or unsigned int type
var = 'character'	In case var is either char or unsigned char type, printable character can be assgined such as 'A'. To display its value also with printable character, please check the option 'Display printable character' optino in the 'miscellaneous' tab in the project setting.
var = <exp></exp>	assign the result of exp into var Example of expression: <e^pi-pi^e>, <1/ln(x)/x>, <exp(-1 100,2))="" pow(x=""></exp(-1></e^pi-pi^e>
*var = number or <exp></exp>	In case var is pointer to basic type variable, assign number or expression to the variable pointed by pointer var note) not supported for Arm MCU
(*var).x = number or <exp></exp>	In case var is pointer to struct or union variable, assign number or expression to the variable x pointed by pointer var note) not supported for Arm MCU
var = number Qn ex. aa = 3.3Q15	Q-format assignment. n at Qn is from 1 to 15 in case var is 16bit integer. n at Qn is from 1 to 31 in case var is 32bit integer. number could be float.
var = number Q ex. aa = 3.3Q	If you set default Q number to var, then you can omit to describe it. For example, if var has default Q number 12, then var=3.14Q has the same effect as var=3.14Q12.

var =	
<exp>Qn</exp>	expression in <> is automatically calculated and then processed as in number
var = <exp>Q</exp>	

Others

clear	clear all command window context
//	one line comment
help	list all commands
l file	load command file (default extension = cmd). That is, execute the contexts of command file by line-to-line.
r file	read command file. Copy the file to the command window. No execution.
skip	During executing command file, the commands after 'skip' command are ignored.

* Dimension / Q assignment functionality

Dimension format (ex, 2.3m, 400p) is for writing/reading float type variables.

Q format (ex, 3.14Q15) is for writing/reading integer type variables and is available only for 2x MCU series.

You can set their default configuration for each variables only in 'Watch' window. For more details, please refer to the 'Watch' window section in this help.

9.2 Watch

01 Watch-1				- • ×								
$\equiv \prec \prec \land \lor \uparrow \downarrow 2 \text{ sec } \checkmark$												
Name	Value	🔳 Туре	Address	Dimensi								
ezDSP_Version_SCI	1060	uns int	0x0000A800									
ezDSP_ulAddr	0x0000A828	uns long	0x0000A828									
u64Test	64	uint64_t	0x0000B46E									
u64aChartWindowTest	array	uint64_t [256]	0x0000B000									
u32Counter	41	uint32_t	0x0000A834									
IPC_Instance[0].IPC_Flag_Ctr_Reg	0x0005CE00	IPC_Flag_Ctr_Reg_t *	0x00090000									
fTest	3.00000238K	float	0x0000B46C	Kilo(K)								
	1	1	1	1								

You can read from or write to the variables in watch window. Note that only visible items are updated to reduce the communication burden of MCU. The function of buttons are ...

The function of buttons are ..

$\equiv \Rightarrow \Rightarrow \land \lor$	↑ ↓	2 sec	-
---	-----	-------	---

- (toggled) : displays all variables or only registered variables
- : registers variable (same to 'Insert' key)
- telete variable (same to 'Delete' key)
- : move up variable
- : move down variable

1 : loads the list of registered variables from the text file. The value of variables are not changed due to this action.

: saves the list of registered variables to text file. It saves the value of variables too. You can load this file in Command window so that you change variables to the value in current watch window. This action helps handle the variables related to board settings or event recording.

Some details for each column are.....

Column	Function
Name	It displays variable name. You can use 'value at address operator (*)' for TI C28x MCU. For example, *pointer variable when pointing to basic type (*pointer variable) when pointing to structure/union type
Value	It displays variable value. Mouse right click toggles the display mode (decimal => hex-decimal => binary => decimal). Hex-decimal number begins with "0x". Binary number begins with "0b". But display mode of pointer variable is fixed to hex-decimal. If you specify dimension, the value is displayed as like 100u, 1K, 1p and so on. If you specify Q-format, the value is displayed such as 3.14Q15. You can change the variable by clicking left mouse button or pressing enter-key. Either number or <expression> is possible as an input format. Various format is supported when you input the value to the variable. Please check the help file of 'Command' window help file</expression>
Туре	It displays the type of variable.
Address	It displays the address of variable.
Dimension	Depending on the variable type, this column can display either dimension or Q format. Dimension If the variable is floating-point type, you can set the dimension of variable. You can change the dimension by clicking left mouse button. You can also use dimension when writing to the variable. For example, writing "30u" is same as "0.0003". dimension $p = pico (10^{-12})$ dimension $n = nano (10^{-9})$ dimension $u = micro (10^{-6})$ dimension $m = mili (10^{-3})$ dimension $K = Kilo (10^{3})$ dimension $G = Giga (10^{9})$ Q format If the variable is integer type, you can set the Q format of variable. Q format is helpful especially to fixed point MCU. Q0 to Q15 can be applied to 16bit integer variable. Q0 to Q30 can be applied to 32bit integer variable. Once the variable is set by Q-format, it can be read/written as a float type variable. Plot and Chart window also displays Q-format integer variable as it is a floating-point type.



9.3 Plot

Plot window



This window plots the value of variables in real-time and saves its data for some time. If the dynamics of the variable is rather slower than the sampling interval, this window will act as an recorder. The integer variable with Q-format is displayed as it is float type. For example, 32bit integer variable with Q31 format is displayed within 1 and -1.

<u>Toolbar</u>

₽ | • | ↔ 🕑 | ↓ 🗜 -

i you can set the variable name, min/max/auto of Y-axis display and display mode. Maximum 8 variables can be displayed in one plot window.

The minimum sampling interval is 5msec. easyDSP reads the value of variable in every sampling interval, then displays it for 'total plot period' duration.

Please note that maximum count of data is limited to 4,294,967,295 per variable. In case PC memory is not enough, it will be less than that.

Please note that the sampling interval you set is not guaranteed. Most of cases, actual sampling interval is longer than your setting value especially when the data count is large. Also timer resolution of Windows systme is roughly 10msec.

The setting can be saved to and loaded from the file by clicking 'Save'/'Load' button. The colors and symbols are predefined as follows.

channel #1 : red - circle channel #2 : blue - square channel #3 : green - triangle channel #4 : violet - diamond channel #5 : black - right triangle channel #6 : weak green - left triangle channel #7 : grey - '+' shape channel #8 : orange - 'x' shape

-Cha	innel		Scale			⊢Display		
	Name		Min	Max	Auto	Symbol	Line	Visible
#1	m_uSeq	Remove	0	10		~	\checkmark	▼
#2	m_uSeqSeq	Remove	50	60			$\overline{\checkmark}$	
#3	m_fVdseRef	Remove	-600	600		◄	$\overline{\checkmark}$	◄
#4	m_fVqseRef	Remove	-600	600		▼	\checkmark	◄
#5	m_fldseRef	Remove	-100	100	\mathbf{V}			~
#6	m_flqseRef	Remove	-100	100	$\overline{\mathbf{v}}$		$\overline{\mathbf{v}}$	
#7	m_fldse	Remove	-100	100	\mathbf{V}		$\overline{}$	~
#8	m_flqse	Remove	-100	100			$\overline{\mathbf{v}}$	•
-Tim	e							
Sa	mpling interval 100	msec	Total plo	ot period	1	min		
- Sott	tings							
Sel	land Cau							

(toggled) : pauses graph / resumes graph.

eriod'.

: shows recent data. It shows the latest data fitting to current plot window size.

: saves the graph into graphic file (bmp, jpg, png formats) or save the graph data into text file (csv format as shown below).

m_uSeq				
date(year-month-day)	time(hour-min-sec)	time(mili_sec)	elapsed time(mili_sec)	value
2017-09-04	12:48:42	223	0	2
2017-09-04	12:48:42	359	136	2
2017-09-04	12:48:42	475	252	2
2017-09-04	12:48:42	597	374	2
2017-09-04	12:48:42	722	499	2
2017-09-04	12:48:42	848	625	2
2017-09-04	12:48:42	976	753	2
2017-09-04	12:48:43	98	875	2
2017-09-04	12:48:43	226	1003	2
2017-09-04	12:48:43	346	1123	2
2017-09-04	12:48:43	470	1247	2
2017-09-04	12:48:43	584	1361	2
2017-09-04	12:48:43	706	1483	2

 \mathbf{I} : saves the graph data to record file (file extension = rec). You can open the record file with record window.

Useful features

Tooltip function : The data value at the mouse cursor position will be displayed with small box
If the communication failed with MCU, the corresponding data point is not displayed at all. As shown below, the line looks broken.

Same when the user intentionally pauses the communication.



- Versatile line display mode by selecting symbol/line/visibility.





- X-axis zoom in/out possible with mouse wheeling.



- Screen dragging is possible in X-axis direction by dragging mouse. (mouse cursor has special shape in this mode)



9.4 Chart



It displays all data of 1-dim array type variable. So, you can use it as an software substitute for the oscilloscope, if your MCU program samples a certain variable into this array variable. Writing to the array is not allowed in Chart window.

It displays the Q-format integer variable as its fractional number. (Ex, 32bit integer with Q31 format is displayed in the range of +1/-1).

<u>Toolbar</u>

ዮ 🔅 🕨	$\leftarrow \leftrightarrow \rightarrow$	$\downarrow \downarrow$	20 sec 🔻	
-------	--	-------------------------	----------	--

T: When clicked, the below dialog box shows up and you can register upto 8 variables and its display properties.

'Channel' : You can select the one-dimensional array variable.

'Scale' : Select the Y-axis range. 'Auto' will adjust its scale automatically based on the variable values in every display.

'Display' : Determines its display mode. The data acquisition keeps going whatever its display mode is.

'Enable fast reading' : makes chart update faster when the MCU resource for communication with easyDSP is enough.

If this option is not working properly, the window becomes empty.

The colors and symbols are predefined as follows.

channel #1 : red - circle channel #2 : blue - square channel #3 : green - triangle channel #4 : violet - diamond channel #5 : black - right triangle channel #6 : weak green - left triangle channel #7 : grey - '+' shape channel #8 : orange - 'x' shape

Chart set	tings									×
Char	Name		D	Count	Range y.Min	y.Max		Display	EXT	ок
#1 #2	IPC_Instance[0].IPC_IntNum	•	Remove	8	-100	0	Auto	□ Symbol IV Line	Visible	
#3 #4	x_a x_b	•	Remove Remove	10 10	0	0	Auto	Symbol 🔽 Line	✓ Visible✓ Visible	Cancel
#5	u64aLongArray	•	Remove	3072	0	0	Auto	Symbol 🔽 Line	Visible	
#6 #7		• •	Remove	10	0	0	Auto	Symbol V Line	Visible	Enable fast reading
#8		•	Remove	10	0	0	Auto	▼ Symbol ▼ Line	Visible	

: updates graph only for one time. If your data are too large, updating them in every sampling interval takes so much time. Please use this toolbar in that case.

(toggled) :pauses graph update / resumes graph update.

💼 : shows left-most part of the graph

- ڬ : shows all graph data.
- : shows right-most part of the graph.

: saves the current graph into graphic file (bmp, jpg, png formats) or save the current graph data into text file (csv format).

: saves the graph data to record file (file extension = rec). You can open the record file with record window.< /FONT >

Useful features

- Please check the link how to use the graph

9.5 Record



It displays the data of record file (extension = rec) which was saved before in either Chart window or Plot window.

Thus, your first action is opening the record file by clicking ${}^{ ilde{ extbf{T}}}$ button.

When opening it, all settings you made before was automatically restored i.e. record file, zoom in/out area and various display mode.

<u>Toolbar</u>

₽ 🕂 🕶 🗩 🕇 🖡

¹: When clicked, the below dialog box shows up with the information of record file name and its saving time. The other part is same to that of either Chart or Plot window.

'Channel' : It just display the variable name and its data count as the record file has. No change is possible.

'Scale' : Select the Y-axis range. 'Auto' will adjust its scale automatically based on the variable values in every display.

'Display' : Determines its display mode.

The colors and symbols are predefined as follows.

channel #1 : red - circle channel #2 : blue - square channel #3 : green - triangle channel #4 : violet - diamond channel #5 : black - right triangle channel #6 : weak green - left triangle channel #7 : grey - '+' shape channel #8 : orange - 'x' shape

Name Count Min Max Image: Count Min Max Image: Count Min Max Image: Count			Display		scale ——	Y-axis		nnel	Cha
#1 m_uSeq 200 0 10 I Auto I Image: Symbol Image: Line Image: Vision visio	OK			<u> </u>	Max	Min	Count	Name	
#2 m_uSeqSeq 200 0 10 IT Auto IV Symbol IV Line IV Vis #3 m_fVdseRef 200 0 10 IT Auto IV Symbol IV Line IV Vis #4 m_fVdseRef 200 0 10 IT Auto IV Symbol IV Line IV Vis #5 Im fidseRef 200 0 10 IT Auto IV Symbol IV Line IV Vis	Visible OK	bol 🗹 Line 🔽 Visible	Auto Symbol R	Aut	10	0	200	m_uSeq	#1
3 m_fVdseRef 200 0 10 Г Auto IF Symbol IF Line IF Vis 44 m_fVqseRef 200 0 10 Г Auto IF Symbol IF Line IF Vis 45 m fldseRef 200 0 10 Г Auto IF Symbol IF Line IF Vis	Visible	bol 🔽 Line 🔽 Visible	T Auto	□ Aut	10	0	200	m_uSeqSeq	ŧ2
4 m_fVqseRef 200 0 10 □ Auto	Visible	bol 🔽 Line 🔽 Visible	T Auto	□ Aut	10	0	200	m_fVdseRef	3
15 m fldseRef 200 0 10 □ Auto 🔽 Symbol 🔽 Line 🗹 Vi	Visible Canc	bol 🔽 Line 🔽 Visible	T Auto	□ Aut	10	0	200	m_fVqseRef	4
	Visible	ibol 🔽 Line 🔽 Visible	T Auto	Aut	10	0	200	m_fldseRef	5
16 m_flqseRef 200 0 10 T Auto Vi	Visible	ibol 🔽 Line 🔽 Visible	T Auto	Aut	10	0	200	m_flqseRef	6
7 m_fldse 200 0 10 I Auto Vi	Visible	bol 🔽 Line 🔽 Visible	Auto Symbol R	Aut	10	0	200	m_fldse	7

• : shows the left-most part of graph.

😁 : shows all graph data.

: shows the right-most part of graph.

1 : load the record file. This is your first action to use this window.

: saves the current graph into graphic file (bmp, jpg, png formats) or save the current graph data into text file (csv format).

Useful features

- Please check the link how to use the graph

9.6 Memory

<u>Common</u>

You can monitor and change the memory under given address. But change of memory is available only for RAM memory.

Note that only visible items are updated to reduce the communication burden of MCU. So, please minimize the window size so that the communication burden of MCU could be also minimized. easyDSP limits the address range according to MCU. In case the adress is limited by easyDSP, the data of address is displayed as '-' without reading.

NOTE :

1. For Arm core MCU, HardFault is caused by accessing an invalid address or security setting. Please be careful when setting the address.

2. For a certain STM32 MCU with secure MPU activated, MCU can be stuck after memory access.

Memory-	1					8
Address 0	▼ 5 sec ▼					
Address	+0	+4	+8	+C	ASCII	^
0x70000000	C5068230	AD048230	010203A0	7C140202	00	
0x70000010	D39F9949	745A4C1A	92A77B1C	FF23BCCC	ILZt.{#.	
0x70000020	30379D4E	2A09060D	F7864886	0D01010D	N.70*.H	
0x70000030	81300005	300B3197	55030609	02130604	01.0U	
0x70000040	0B315355	03060930	0C080455	31435302	US1.0USC1	
0x70000050	060F3011	07045503	654E080C	6F592077	.0UNew Yo	
0x70000060	21316B72	03061F30	0C0A0455	78655418	rk1!0UTex	
0x70000070	49207361	7274736E	6E656D75	2C2E7374	as Instruments.,	
0x7000080	636E4920	3013312E	55030611	0A0C0B04	Inc.1.0U	
0x70000090	41544953	4D204152	0F315543	03060D30	SITARA MCU1.0	
0x700000A0	0C030455	626C4106	31747265	061D301F	UAlbert1.0	Υ.

Memory-	M Memory-2															×		
Address &ez_u32Addr // address ▼ bit width 8 ▼ 5 sec ▼																		
Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII	^
0x70072678	78	26	07	70	00	00	00	00	00	00	00	00	00	00	00	00	x&.p	
0x70072688	00	00	00	00	00	00	00	00	00	00	00	00	20	00	00	00		
0x70072698	20	00	00	00	01	00	00	00	01	00	00	00	01	00	00	00		
0x700726A8	00	00	F0	50	01	00	00	00	FF	FF	FF	FF	01	00	00	00	P	
0x700726B8	00	00	00	00	00	00	00	00	01	00	00	00	A0	22	07	70	".p	
0x700726C8	28	27	07	70	6D	27	07	70	78	56	34	12	10	26	07	70	('.pm'.pxV4&.p	
0x700726D8	EA	D6	FC	3D	D8	26	07	70	18	26	07	70	E0	26	07	70	=.&.p.&.p.&.p	
0x700726E8	E0	FF	FF	FF	4F	00	00	00	EC	26	07	70	EF	CD	AB	89	0&.p	
0x700726F8	78	56	34	12	B1	FF	FF	FF	FC	26	07	70	B1	FF	FF	FF	xV4&.p	
0x70072708	4F	00	00	00	38	26	07	70	40	26	07	70	91	23	08	70	O8&.p@&.p.#.p	
0x70072718	54	27	07	70	B1	FF	FF	FF	B1	FF	FF	FF	20	00	00	00	Т'.р	
0x70072728	15	03	00	00	73	27	07	70	4F	00	00	00	30	27	07	70	s'.pO0'.p	v
,																		

This window displays a memory with hex format and variable bit width (8/16/32 bits). To change its value, first select the row and click left button of mouse in the target location. Versatile address input is available such as 0x1234 (hex), 1234 (hex without 0x prefix) and &variable. Also comment (//) can be added to the address input such as '0x1234 // register'. In the address combo box, the recent addresses are registered so that you can easily swap between.

Total memory size to be displayed in a window is 1kB (0x400). But regular data update is limited to only visible area of window.

Note :

- The start address is 4B aligned for TI C28x MCU.
 example) if input address is 0x--0 or 0x--1, then start address is 0x--0.
 example) if input address is 0x--2 or 0x--3, then start address is 0x--2.
- The start address is 8B aligned for Arm core.
 example) if input address is 0x--0 to 0x--7, then start address is 0x--0.
 example) if input address is 0x--8 to 0x--F, then start address is 0x--8.

3. The first memory address shown in the window could be not the address you input in the adress combo box.

4. 1kB memory area is displayed from the start address.

5. In case &var format is used as an address input, if it is changed with code modification, the address of the window is automatically changed after MCU booting.

When easyDSP communicates with multi cores of ARM MCU

You can select which core accesses the memory.

This is useful in case each core has different memory contents.

If the start address is set by '&n:var' format, the core is fixed to CPUn.

Memory-1																		x
Core CPU1	•	Ad	Idres	s Ox	10001	1948				•	bit v	vidth	8	•	2 s	ec	•	
Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII	^
0x10001948	00	00	00	00	00	00	00	00	00	00	00	00	1A	04	00	00		
0x10001958	48	19	00	10	1C	00	00	00	00	00	00	00	00	00	00	00	Н	
0x10001968	24	0C	00	00	00	00	00	00	01	00	00	00	02	4C	04	00	\$L	
0x10001978	01	4C	04	00	01	4C	04	00	00	00	00	00	00	00	00	00	.LL	
0x10001988	00	00	00	00	00	00	00	00	0C	00	00	00	0D	00	00	00		
0x10001998	06	02	01	00	00	02	00	69	00	01	01	8B	E7	01	20	00	i	
0x100019A8	00	0C	C3	E7	01	20	0A	00	00	00	0A	00	02	00	00	00		
0x100019B8	69	00	02	00	00	01	02	06	0D	00	00	00	00	00	00	00	i	
0x100019C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0x100019D8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		
0x100019E8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00		Υ.

9.7 Array

E	🗄 Array-1														×
Γ	u16aArrayDim2 v uint16_t [100][100] 20 sec v														
	u16aArrayDim2	[*][0]	[*][1]	[*][2]	[*][3]	[*][4]	[*][5]	[*][6]	[*][7]	[*][8]	[*][9]	[*][10]	[*][11]	[*][12]	^
	[0][*]	0	1	2	3	4	5	6	7	8	9	10	11	12	
	[1][*]	10	11	12	13	14	15	16	17	18	19	20	21	22	Ī
	[2][*]	20	21	22	23	24	25	26	27	28	29	30	31	32	t I
	[3][*]	30	31	32	33	34	35	36	37	38	39	40	41	42	Ī
	[4][*]	40	41	42	43	44	45	46	47	48	49	50	51	52	t I
	[5][*]	50	51	52	53	54	55	56	57	58	59	60	61	62	Ī
	[6][*]	60	61	62	63	64	65	66	67	68	69	70	71	72	t I
	[7][*]	70	71	72	73	74	75	76	77	78	79	80	81	82	[
	[8][*]	80	81	82	83	84	85	86	87	88	89	90	91	92	_
	<													>	.::

In Array window, the values of array variable which is one dimensional or two dimensional are displayed with grid view.

Note that only visible cells are updated to reduce the communication burden of MCU.

The member of array should be fundamental type. Please use Tree window if the member of array is structural variable type. You can change its value by mouse left button or enter key. You can use 'copy-paste'. Especially with Microsoft Excel program. Please select block by clicking column or row of this array. You can select all by clicking the name of variable. Note that it could take addtional communication time since easyDSP first fills the empty cells (if any) before copying.

9.8 Tree

Tree window

E Tree-1	- • ×
Struct/Union 💌 GpioDataRegs 💌 5 sec 💌	
GpioDataRegs	<u> </u>
🚊 GpioDataRegs.GPADAT	
GpioDataRegs.GPADAT.all : 4294701056	
🖃 GpioDataRegs.GPADAT.bit	
GpioDataRegs.GPADAT.bit.GPIO0 : 0	
GpioDataRegs.GPADAT.bit.GPIO1:0	
GpioDataRegs.GPADAT.bit.GPIO2:0	
GpioDataRegs.GPADAT.bit.GPIO3:0	
GpioDataRegs.GPADAT.bit.GPIO4:0	
GpioDataRegs.GPADAT.bit.GPIO5:0	
GpioDataRegs.GPADAT.bit.GPIO6:0	
GpioDataReqs.GPADAT.bit.GPIO7:0	-

In tree window, the values of array, structure type variable are displayed with tree view. Note that only visible cells are updated to reduce the communication burden of MCU. By clicking left mouse button or enter key input, you can change the value of variable. By clicking right mouse button, you can change display mode (decimal => hex-decimal => binary = > decimal....).

10. Trouble Shooting

10.1 Common

Trouble : easyDSP communication fails at first try of easyDSP use

Shooting: there are several reasons for this. Please check below check points. check point 1 : If ram booting or flash programming is not successful, please check the hardware setting particularly for connector pin mapping, contact failure of connector and cable. You can check if the hardware and software setting is proper by running MCU with debugger and monitoring the variables by easyDSP.

check point 2 : The easyDSP source file and header file should be included in your project. check point 3 : #define variable should be set properly in the easyDSP header file. check point 4 : In the main.c, easyDSP related functions should be called. check point 5 : The baud rate of project setting should be same to that in the easyDSP header file. check point 6 : In the user program, don't allocate SCI or UART for easyDSP to another GPIO pins.

check point 7 : In the user program, don't allocate GPIO for easyDSP to another function. check point 8 : easyDSP ISR (Interrupt Service Routine) should have enough time resource to run properly. Please check below.

Trouble : communication fails due to the lack of time resource to easyDSP

Shooting: You have to secure the required time resource to easyDSP communication. Please try below methods.

- 1. Increases 'wait-more-time' in the project menu
- 2. Slows down the baud-rate
- 3. Minimize the number of variables of monitoring (For example, use Command Window only)
- 4. If possible, increase the priority of easyDSP ISR (SCI or UART)

Trouble : At first, easyDSP works well but soon it fails. Why?

Shooting 1 : easyDSP uses the lowest prioritized ISR (Interrupt Service Routine) of MCU by default. If higher prioritized interrupt routine starts to take most of time resource, then ISR for easyDSP doesn't work properly. Please refer to above trouble and shooting.

Shooting 1 : in a power electronics system with high voltage and high current switching operation, easyDSP communication could failed due to either conducted or radiated noise. Please take a measure to reduce the noise accordingly.

Trouble : easyDSP is not connected

Cause : mechanical connection is not stable

Shooting : please connect easyDSP directly to PC (not via USB extension port) or use different USB port or use new USB cable.



Trouble : Error message like below

Shooting : You will face below (or similar) error message with 32bit Windows. Please use 64bit Windows.



Trouble : can't access the website (<u>www.easydsp.com</u>)

Cause : due to limited traffic size allowable per day, its access is temporarily blocked. Shooting : please access the web site tomorrow.



· 사이트 관리자는 호스팅 홈페이지 '나의서비스관리' 메뉴에서 사양 변경 및 트래픽 리셋이 가능합

니다

10.2 C28x

Trouble shooting (TI C28x)

Trouble : when SCI-A GPIO port recommended by easyDSP could not be usable

To use RAM booting and flash programming with easyDSP, easyDSP should be connected to the designated SCI-A and GPIO port.

In case only monitoring is used with easyDSP, easyDSP can be connected to any SCI and any GPIO port, but you have to modify the easyDSP source file accordingly.

To use RAM booting and flash programming with easyDSP, but with other GPIO port than designated, please refer to the help file <u>'How to use other SCI port than designated'</u>.

It's about how to use designated SCI-A port during RAM booting or flash programming, and then use other SCI port than designated during monitoring.

Trouble : 'section not aligned' message in flashrom dialog

Status	
Checking bin file Failed !	
The output section starting 0x080002 should be aligned on a 4-word boundary !	
Please use 'ALIGN(4)' in linker command file !	

Shooting : easyDSP uses TI's flash API to access onchip flashrom. TI flash API of Gen.3 MCU (ex. F2807x, F28002x, F28004x, F2837x, F2738x) requires section alignment on the address (min. 4 words boundary or recommended 8 words boundary) depending on MCU. That is, the start address of the section should be either 0x*0, 0x*4, 0x*8 or 0x*C for C28x core and either 0x*0 or 0x*8 for Arm Cortex-M4 (ex, F2838x CM). In the picture above, the error is caused since the start address of the section is 0x*2.To avoid this problem, please align all sections linked to flash on a minimum 64-bit boundary in the linker command file for your code project. As shown below linker command file from TI, it is already applied as recommended value for default sections but you need to do it yourself for your own section.

If the program continues even after above measure, please check your map file (*.map) and identify which section makes error (the section starting from the address 0x080002 in the picture) and apply section specific measures.

```
<in case of TMS320F280049>
SECTIONS
{
              : > BEGIN, PAGE = 0, ALIGN(4)
: >> FLASH_BANK0_SEC2 | FLASH_BANK0_SEC3 | FLASH_BANK0_SEC5, PAGE = 0, ALIGN(4)
   codestart
   .text
                  : > FLASH_BANK0_SEC1, PAGE = 0, ALIGN(4)
: > FLASH_BANK0_SEC1, PAGE = 0, ALIGN(4)
   .cinit
   .switch
                : > FLASH_BANK0_SEC1,
   .reset
                  : > RESET,
                               PAGE = 0, TYPE = DSECT /* not used, */
                  : > RAMM1,
                                PAGE = 1
   .stack
#if defined(__TI_EABI__)
   .init_array :> FLASH_BANK0_SEC1,
.bss :> RAMLS5, PAGE
                                          PAGE = 0,
                                                           ALIGN(4)
                                 PAGE = 1
  .bss:output :> RAMLSS,
.bss:cio :> RAMLSO,
.data :> RAMLSS,
.sysmem :> RAMLSS,
                                   PAGE = 0
                                  PAGE = 0
                                  PAGE = 1
                                  PAGE = 1
   /* Initalized sections go in Flash */
   .const : > FLASH_BANK0_SEC4,
                                            PAGE = 0,
                                                           ALIGN(4)
                : > FLASH_BANK0_SEC1,
: > RAMLS5, PAGE
: > RAMLS5, PAGF
#else
   .pinit
                                            PAGE = 0,
                                                           ALIGN(4)
                                PAGE = 1
   .ebss
                                  PAGE = 1
   .esysmem
   .cio
                                   PAGE = 0
                  : > RAMLSØ,
   .econst
                 : > FLASH_BANK0_SEC4, PAGE = 0, ALIGN(4)
#endif
                  : > RAMGS0,
                               PAGE = 1
   ramgs0
                                PAGE = 1
                  : > RAMGS1,
  ramgs1
<in case of TMS320F28388 CPU1 and CPU2 >
SECTIONS
{
    codestart
                           : > BEGIN, ALIGN(8)
                           : >> FLASH1 | FLASH2 | FLASH3 | FLASH4, ALIGN(8)
    .text
                          : > FLASH4, ALIGN(8)
   .cinit
   .switch
                           : > FLASH1, ALIGN(8)
                           : > RESET, TYPE = DSECT /* not used, */
    .reset
                           : > RAMM1
    .stack
#if defined(__TI_EABI__)
    .init_array : > FLASH1, ALIGN(8)
                       : > RAMLS5
    .bss
                    : > RAMLS3
    .bss:output
                       : > RAMLS5
    .bss:cio
                       : > RAMLS5
    .data
    .sysmem
                      : > RAMLS5
   /* Initalized sections go in Flash */
                : > FLASH5, ALIGN(8)
    .const
#else
   .pinit
                   : > FLASH1, ALIGN(8)
                      : > RAMLS5
    .ebss
                      : > RAMLS5
   .esysmem
                      : > RAMLS5
    .cio
   /* Initalized sections go in Flash */
    .econst
              : >> FLASH4 | FLASH5, ALIGN(8)
#endif
```
<in case="" of="" ti<br="">SECTIONS</in>	4S320F28388 CM>
i .resetisr .vftable .table .text .cinit .pinit .switch .sysmem	<pre>: > CMBANK0_RESETISR, ALIGN(16) : > CMBANK0_SECTOR0, ALIGN(16) /* Application placed vector table in Flash*/ : > S0RAM /* Application placed vector table in RAM*/ : >> CMBANK0_SECTOR0 CMBANK0_SECTOR1, ALIGN(16) : > CMBANK0_SECTOR0, ALIGN(16) : >> CMBANK0_SECTOR0 CMBANK0_SECTOR1, ALIGN(16) : >> CMBANK0_SECTOR0 CMBANK0_SECTOR1, ALIGN(16) : >> S2RAM</pre>
.stack .ebss .econst .esysmem .data .bss .const	: > C1RAM : > C1RAM : > CMBANKØ_SECTORØ CMBANKØ_SECTOR1, ALIGN(16) : > C1RAM : > S3RAM : > S3RAM : >> CMBANKØ_SECTORØ CMBANKØ_SECTOR1, ALIGN(16)

Trouble : 'The address xxx is not flash area !' error message when entering to flash dialog

Status		
CPU1 : Checking the validity of bin file failed !		
The address 0x001500 is not flash area!		

Cause : Flash programming is not feasible since the initialized section is located at RAM. Particularly in the case that initial value of user variable in CLA program is set.

Shooting : Please refer to below manual capture. In case the initial value is needed, write the variable with the value in the main() or other C28x code.

10.2.3 C Language Restrictions

There are several additional restrictions to the C language for CLA.

Defining and initializing global/static data is not supported.

Since the CLA code is executed in an interrupt driven environment, there is no C system boot sequence. As a result, global/static data initialization must be done during program execution, either by the C28x driver code or within a CLA function.

Variables defined as const can be initialized globally. The compiler creates initialized data sections named .const_cla to hold these variables.

- CLA code cannot call C28x functions. The linker provides a diagnostic message if code compiled for C28
 colla code compiled for C14 on if code compiled for C14 colla code compiled for C28
- calls code compiled for CLA or if code compiled for CLA calls code compiled for C28.
- Recursive function calls are not supported.
 The use of function pointers is not supported.

Trouble : Booting is successful but verifying is not from the address 0 due to MCU reading failure

RAM booting for TMS320F28x				
Boot				
Status : Verifying RAM booting Failed ! Failed to read MCU at address 0x000000				
Enables fast booting Enables fast verifying				
Boot > Exit Boot	Verify	Stop	Exit	

Verifying is done by the easyDSP communication with MCU. So, any reason to block the communication could cause this problem.

cause-1 : The source files easyDSP provides for its communication is not included in the project Shooting-1 : please include them in the project and modify main.c file accordingly. Please refer to the help file.

cause-2 : user program sets the GPIO easyDSP is using improperly. Shooting-2 : please remove the GPIO seeting from your program.

cause-3 : enough time resource is not allocated to easyDSP communication Shooting-3 : For example, if ePWM interrupt has high frequency, please reduce it.

Trouble : compile error of easyDSP source file /w controlSUITE

cause : controlSUITE has the different register naming from C2000Ware Shooting : please use the latest TI source file (C2000Ware)

Trouble : MCU is working improperly when using a large number of variables or big size array

cause : bug of TI source file Shooting : please use the latest TI source file (C2000Ware)

Trouble : F2838x is not working with easyDSP

Shooting : In case your board has 20Mhz clock and your source file is based on C2000Ware_3_02_00_00 (or upward), please predefine USE_20MHZ_XTAL so that TI source files can be compiled based on 20MHz. Please check below excerpt from TI's C2000Ware_3_02_00_00 release note.

· F2838x driverlib and examples updated to use 25MHz XTAL clock as default input clock

Note: By default, Device_init function in driverlib and InitSysctr1 function in bitfield examples assumes that the XTAL frequency is 25MHz. If a 20MHz XTAL is used, please add a predefined symbol "USE_20MHZ_XTAL" in your CCS project. If a different XTAL is used, you need to update the PLL multipliers and dividers accordingly. Note that the latest F2838x controlCARDs (Rev.B and later) have been updated to use 25MHz XTAL by default. If you have an older 20MHz XTAL controlCARD (E1, E2, or Rev.A), refer to the controlCARD documentation on steps to reconfigure the controlCARD from 20MHz to 25MHz.

Trouble: Warning message as below before RAM booting is started



Shooting : change your cmd file so that your code is not overlapped with the reserved RAM memory for bootrom.

For example, 28377D has the reserved RAM memory for bootrom operation as shown in the table below (Excerpt from Technical Reference Manual (Literature Number: SPRUHM8I, Revised September 2019)).

Table 4-19. Reserved RAM and Flash Memory Map for CPU1

Memory	Description	Start Address	End Address	Length
RAM	Boot ROM	0x0000 0002	0x0000 0122	0x0121
	TI-RTOS ⁽¹⁾	0x0000 0780	0x0000 07FF	0x0080
Flash	TI-RTOS ⁽¹⁾⁽²⁾	0x0008 2000	0x0008 2823	0x0824

(1) If the user is not planning on using TI-RTOS in ROM, then these memory locations are free to be used by the application.
(2) For using the TLRTOS in flash sector A. TL recommende that this sector he made unserume or at minimum the sector should be applied on the sector sh

(2) For using the TI-RTOS in flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector should be verified that there is no secure zone claiming this sector.

In case your code is overlapped with this area, easyDSP detects it and shows warning message.

Trouble: RAM booting failed with message box below

Shooting : RAM booting is failed since program memory is allocated to flash area, not ram area. The address shown in the box (ex, 0x33D0FE) belongs to flash. Please change your link file to allocate all the memory to ram area and try again.



Trouble: Auto bauding failed

Shooting :

Mainly due to wrong hardware connection between easyDSP and your MCU board.

Step 1 : please check if your connection is correct. Hope you find misconnection in this step. Or, move to step 2.

Step 2 : please check the waveforms of easyDSP pins during booting. Also refer to the below sequence of easyDSP pin status.

In case /RESET of easyDSP is NOT directly connected to reset pin of DSP, please check reset pin status of DSP pin together.

Step 2-1 : please check if /BOOT is low when /RESET is changed from low to high.

In case power monitoring IC (TPSxxxx) is used to generate /XRS signal and /RESET is an input to the IC,

some cases it happens that /XRS becomes high after /BOOT is high, which will make booting failure.

Step 2-2 : please check RX and TX. After /BOOT pins are released high, 0x41 is sent from PC to MCU via RX.

Bauding bps could be different by MCU type and booting speed option.

Then MCU send 0x41 at the detected bps (ex. 38400bps here). Please check the waveforms and see what is missing in your board.



Trouble : compilation failed with below error message

undefined first referenced symbol in file

```
LL$$OR C:\\tidcs\\c28\\DSP2833x\\Project\\Debug\\easy2833x_sci_v7.3.obj
ULL$$CMP C:\\tidcs\\c28\\DSP2833x\\Project\\Debug\\easy2833x_sci_v7.3.obj
```

error: unresolved symbols remain error: errors encountered during linking; "./Debug/inverter.out" not built

>> Compilation failure

Shooting : The TMS320C28x does not directly support some C/C++ integer operations. Evaluating these operations is done with calls to run-time-support routines. These routines are hard-coded in assembly language. They are members of the object and source run-time-support libraries.

"ULL\$\$CMP" = unsigned long long comparison

"LL\$\$OR" = long long oring

Therefore, please include run-time library at compiling.

Trouble : Type of all variables are displayed as 'int'

Shooting: Please use the latest easyDSP version and set the proper debugging model (either coff or dwarf) in the project setting.

10.3 STM32

Trouble shooting (ST STM32)

Trouble: below error message from FlashROM or RAM booting dialog

Shooting : place your code to flash area for flash dialog operation. And place your code to RAM area for RAM booting dialog operation.

easyDSP help

nauon	1	age Sel	COLIDIT		
Erase > Program > Reset >	> Exit	Select	Index	Start Address	Size
			0	0x08000000	16K
	1		1	0x08004000	16K
Erase > Program > Verify > Res	set > Exit		2	0x08008000	16K
			3	0x0800C000	16K
			4	0x08010000	64K
Erase			5	0x08020000	128K
			6	0x08040000	128K
	1		7	0x08060000	128K
Program			8	0x08080000	128K
			9	0x080A0000	128K
	1		10	0x080C0000	128K
Verify			11	0x080E0000	128K
			12	0x08100000	128K
	1		13	0x08120000	128K
STOP !!!			14	0x08140000	128K
			15	0x08160000	128K
Reset > Exit	Exit	All	Non	e Used	Not Used
- Line - Failed					
lecking memory region Failed	0.000000055	descent to the			
e address range (0x20003000 -	0x200030FF)	doesn't belo	ng to fla	sn memory	

STM32 RAM booting	×
Operation	
Writing RAM > Boot > Exit Writing RAM > Verifying RA	M > Boot > Exit STOP !!! Exit
Checking memory region Failed ! The address 0x08000000-0x080000FF overlaps with the flas	n area !

Trouble: Failed to enter bootloader mode

Shooting :

Mainly due to wrong hardware connection between easyDSP and your MCU board.

Step 1 : please check if your connection is correct. Hope you find misconnection in this step. Or, move to step 2.

Step 2 : please check the waveforms of easyDSP pins during booting. Also refer to the below sequence of easyDSP pin status.

In case /RESET of easyDSP is NOT directly connected to reset pin of MCU, please check reset pin status of DSP pin directly.

Step 2-1 : please check if BOOT pin is high when /RESET pin is changed from low to high.

In case power monitoring IC (TPSxxxx) is used to generate NRST signal and /RESET is an input to the IC,

it could happen that NRST becomes high after BOOT is low, which will make booting failure.

Step 2-2 : please check RX and TX. After BOOT pin is low, 0x7F (even parity) is sent from PC

to MCU via RX.

Bauding bps could be either 115200bps or 57600bps or other value depending of MCU

type. Then MCU send 0x79 (even parity) to PC at the detected bps. With this handshake, bps of each side (easyDSP and MCU) are aligned.

In case you can't observe 0x79 at all, please modify the option byte accordingly.



11. Tips

11.1 DA converter

If your MCU board has DA converter, you can monitor the variables on the oscilloscope by outputing them via DA converter. It is very helpful in debugging your program. In this tip, it is explained how you can change the content of DA converter (that is, variable to display) easily in real time.

Step 1 : Modify da.h file

easyDSP supports c source file and its header file (da.c and da.h) for dac control. File da.h is like below.

```
// File name : da.c
// function : DA output control
// variable explanation(\#=1,2,3,4)
// da# : address of variable
// da# type = 0; the variable is float
// = 1; the variable is integer
// da#_mid : mid value
// da#_rng : da scale
// use this routine in EasyDSP as below
// da1=&var float
// da1_type=0
// da1 mid=0.
// da1 rng=20
// da2 = &var_int
// da2_type = 1
// da2_mid = 0.
// da2_rng = 20
```

```
#ifndef _DA_EasyDSP
#define _DA_EasyDSP
// you should specify the da address of your own
#define DA1_ADDR (*(int *)0X03C000e)
#define DA2_ADDR (*(int *)0X03C000d)
#define DA3_ADDR (*(int *)0X03C000b)
#define DA4 ADDR (*(int *)0X03C0007)
#define
extern unsigned int da1, da2, da3, da4, da1_type, da2_type, da3_type, da4_type;
extern float da1_rng, da1_val, da1_mid;
extern float da2_rng, da2_val, da2_mid;
extern float da3 rng, da3 val, da3 mid;
extern float da4_rng, da4_val, da4_mid;
// Notice : If you need faster DA output, please replace 'divide' part
// in the macro with 'multiply' accordingly.
// 12 bit DA
#define DA12(num) \
da##num##_val = (da##num##_type == 0 ? *(float *)da##num : (float)(*(int *)da##num)) ; \
DA##num##_ADDR = (int)((da##num##_val-da##num##_mid )* 0x7ff/da##num##_rng) +
0x800;
// 8 bit DA
#define DA8(num) \
da##num##_val = (da##num##_type == 0 ? *(float *)da##num : (float)(*(int *)da##num)) ; \
DA##num##_ADDR = (int)((da##num##_val-da##num##_mid )*0x7f/da##num##_rng) + 0x80 ;
#endif
```

At first, the address of da converter on your board should be defined correctly in the DA#_ADDR define lines(#=1,2,3,4). And then, you should also modify the macro function for dac output considering the feature of your dac's own. In above example code, 8 bit and 12bit dac with positive/negative output dac are shown.

Note : divide operation in the macro may need long time to be executed. For faster da output, replace it by the multiply operation.

Necessary variables are defined in the da.c file and their meanings are da# = The address of variable which is output to DA channel # da#_type = The type of variable. 1 = Integer, 0 = float da#_rng = range of display da#_mid = mid value of display

Step 2 : Modify your program

Make your MCU program contain the da.c and da.h you modified. And insert following macros where you want dac output is made .Normally, the insertion place is in the timer interrupt routine for repetitive output.

#include "da.h"

..... DA12(1); DA12(2); DA12(3); DA12(4);

Step 3 : Use easyDSP

Finally, you can control the da converter in the command window or other windows as follows.

da1=&var_float da1_type=0 da1_mid=0 da1_rng=20 da2 = &var_int da2_type = 1 da2_mid= 0 da2_rng = 20

11.2 Others

only for MCU flash programming with easyDSP 555

In case you like MCU flash programming only without using various easyDSP communication features, then please make a easyDSP project with the target output file (for example, *.out file), and go to flashROM menu and program flash.

Insert new line in command window

Basically, enter-key input in command window means the running of current line command. To insert new line without running command, two methods are supported. One is just clicking the tool bar of

new line [™]. The other is 'Ctrl + Enter' key input.

Confirm your assignment command in command window

You can change variable value by assignment commands(=). And then confirm change by clicking the

right button of mouse. This action is equal to the tool bar of 'update' 🧮.

Save some information on the flashrom

Because easyDSP supports sector erase of flashrom, you can use some sectors of flashrom for booting data and the other sectors for saving your system information.

11.3 FAQ

What's difference between easyDSP and Jtag/SWD debugger ?

easyDSP help

They have different purpose. Debugger is useful when you develop hardware and software in the beginning especially with breakpoint, step-in operation. But in some applications like motor drives, you can't use this features when the system is running. So during system operation, you need to monitor the variables in your code for system debugging. The variable monitoring with debugger has some limits such as limited number of variables, monitoring speed. Even worse is under very noisy environment (high current, high voltage switching) the debugger is sometimes disconnected. And for mass production, the debugger accessibility is limited to protect IP.

On the other hand, easyDSP has very stable connection all the time since it communicates with MCU with communication channel like SCI or UART.

When to use easyDSP, when to use debugger ?

Debugger is useful when you develop MCU board or its basic firmware. On the other hands, easyDSP is useful when you develop/debugg a high-level system algorithm. By combining debugger and easyDSP, the best debugging environment could be implemented.

How reliable is reading variable?

100% reliability is not guaranteed. easyDSP could read wrong value of variable.

How reliable is writing variable?

2 byte checksum is checked before writing to variable. So the probability of having incorrect writing is extremely remote. But not 100% guaranteed.

How reliable is writing flash rom?

Flashrom is written by clicking 'Program' button. But nothing is checked and verified during writing process. Therefore you should check it by yourself by clicking 'Verify' button afterwards.

Which value will be displayed if the reading operation fails ?

either '?' (ex, in watch window) or no display in plot and chart window.

Does easyDSP do compiling and linking C program?

No. They are done by compiler and linker provided by chip maker.

How many variables can I monitor using easyDSP?

As much as the resource of your PC, speed and memory are permitted.

12. Driver

12.1 Driver Installation

NOTE) 64bit Windows is mandatory !!

easyDSP uses FT2232 chip from FTDI as an USB controller IC. Therefore driver of easyDSP is same to D2XX Direct Driver of FT2232.

You can get all drivers in <u>http://www.ftdichip.com/Drivers/D2XX.htm</u>, all installation guidance in <u>http://www.ftdichip.com/Documents/InstallGuides.htm</u>.

Windows OS	How to install driver

easyDSP help

Windows 11 Windows 10 Windows 8.1 Windows 8 Windows 7	Just run the "CDM212364_Setup.exe" in "Driver" folder of easyDSP program BEFORE connecting easyDSP pod into your PC. To install the latest driver file always, please visit <u>http://www.ftdichip.com/Drivers/D2XX.htm</u> . Please refer to the below links for detailed installation process. <u>Windows 10/11 Installation Guide</u> <u>Windows 8 Installation Guide</u> <u>Windows 7 Installation Guide</u>
Windows Vista Windows XP	Not included in the installation files but you can download it Drivers : <u>http://www.ftdichip.com/Drivers/CDM/CDM20824_Setup.exe</u> Installation process : <u>Windows Vista Installation Guide</u> <u>Windows XP Installation Guide</u>

Since easyDSP uses FT2232 USB controller chip from FTDI, you can refer to the latest driver file (D2XX direct driver) and its installation guideline from FTDI.

http://www.ftdichip.com/Drivers/D2XX.htm

http://www.ftdichip.com/Documents/InstallGuides.htm

After the driver is well installed, you will find USB Serial Converter A/B in the device manager once the easyDSP pod is connected to PC.



12.2 Driver Uninstallation

In case general Windows way of driver removal is not successful, CDM Uninstaller can be used. CDM Uninstaller is a free application that can selectively remove Windows device drivers from the user's system as specified by the device Vendor ID and Product ID. This application comes as a command driven application or as a GUI executable.

The readme for the GUI version can be viewed <u>here</u>. Please refer to the readme for running the

application.

Both applications come as a zipped executable that needs to be extracted prior to running.

Download CDM Uninstaller (command line version + GUI version)

Major process is to add Vendor/Product ID and click 'Remove Devices'.

CDM Uninstaller				
Vendor ID 0403 Product ID 6001				
VID_0403 PID_6001	Add			
	<u>R</u> emove			
	<u>C</u> lear			
Generate uninstall log file				
Ready				
Remove Devices	Cancel			