参 考 資 料

TEXAS INSTRUMENTS
www.tij.co.jp

JAJA100– 11月 2007年

C672x DSP ROM ブートの使い方

アプリケーション技術部

アブストラクト

本資料は、C672x DSP (C6720 / C6722 / C6726 / C6727) の AIS (Application Image Script) フォーマットを使用せずにしないパラレル・フラッシュ・ブートモードを使用する方法を紹介します。
C672x はI2Cマスタ/スレーブ、SPIマスタ/スレーブ、UHPI、およびパラレル・フラッシュ・ブートを行うことができます。I2CおよびSPIブートは必ずAISを使用します。UHPIブートの場合は使用しません。パラレル・フラッシュ・ブートのみ、AISを使用する方法としない方法をユーザーが選択することができます。本資料ではAISを使わない場合のみ説明します。AISを使用する場合は参考文献を参照してください。

目次

1 概要 2
2 セカンド・ブートローダ 2
2.1 セカンド・ブートローダの概要 2
2.2 ビット・モードの選択 3
2.3 上位アドレス・バスの補足 4
3 ROMイメージの作成 5
4 参考文献 5
Appendix A. セカンド・ブートローダの例 6

図

図 1 セカンド・ブートローダのフローチャート 2
図 2 ブート・テーブルの形式 3
図 3 C6727 EMIF (BGAパッケージ) と 16ビット・フラッシュメモリの接続例 4

1　概要

パラレル・フラッシュ・ブートモードでは、C672x の内蔵 ROMに格納されているブートローダが以下の手順で DSP をブートします。

1. フラッシュメモリの先頭バイトをリードして 8ビットモードか16ビットモードかを決定します。
2. フラッシュメモリの先頭から１Kバイトのデータを内部 RAMの 0x10000000-0x100003FF 番地にコピーします。ここで、フラッシュメモリの先頭から１Kバイトの領域にはセカンド・ブートローダを格納しておきます。セカンド・ブートローダはユーザーが作成する必要があります。
3. 0x10000004 番地からプログラムを実行してユーザー・アプリケーション・コードをダウンロードします。ここで、ユーザー・アプリケーション・コードはブート・テーブル形式にしてフラッシュメモリに格納する必要があります。ユーザー・アプリケーション・コードのダウンロードが完了すると、エントリー・アドレスに分岐してプログラムを実行します。
4. 本資料では、セカンド・ブートローダの作成方法、ROM イメージの作成方法について説明します。AISを使用する場合は「Using the TMS320C672x Bootloader (文書番号 SPRAA69)」を参照してください。

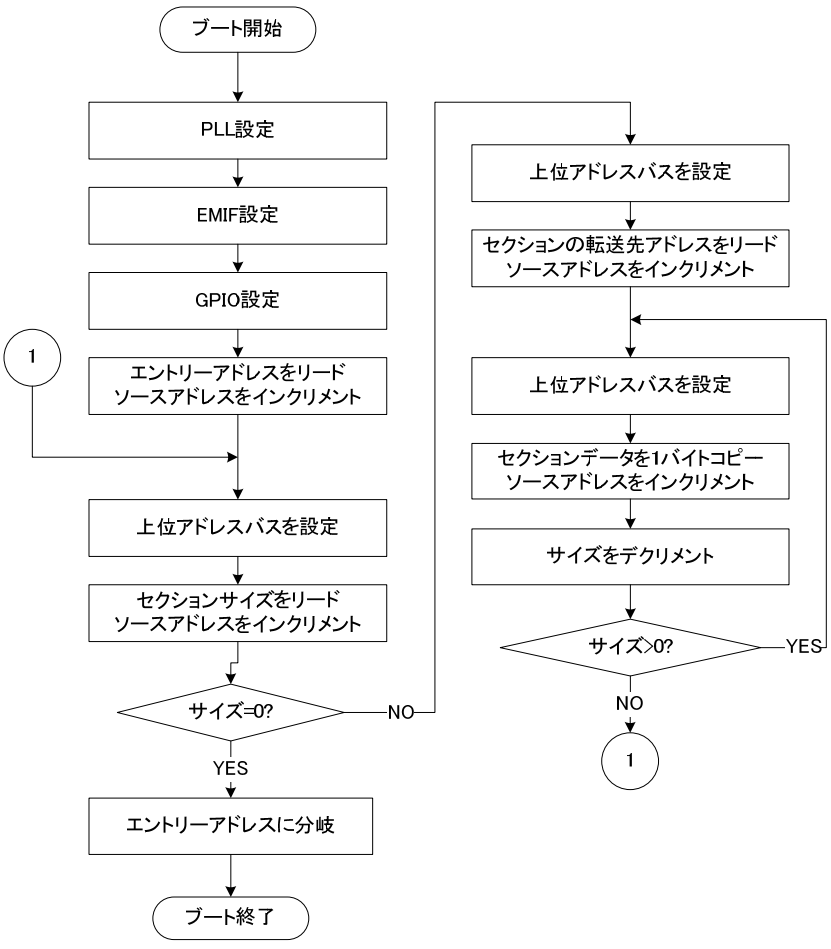
2　セカンド・ブートローダ

2.1　セカンド・ブートローダの概要

図１はセカンド・ブートローダの典型的なフローチャートを示しています。

本書では巻末に参考資料としてサンプルのセカンド･ブート･ローダを収録しています。このコードに関しては Appendix Aを参照してください。

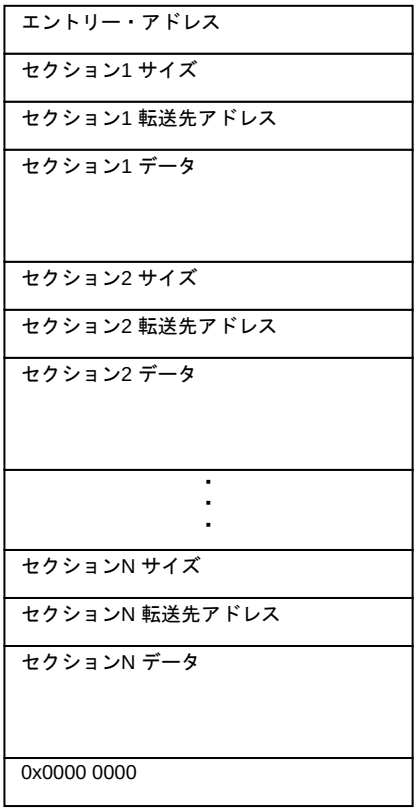
図１ セカンド・ブートローダのフローチャート



サンプルのセカンド・ブートローダでは、ユーザーコードをダウンロードする前に PLL、EMIF、GPIO の設定を行っています。PLL と EMIF の設定内容は、システム構成に応じて変更してください。GPIO は EMIF の上位アドレスを補足するために使用しています。上位アドレス・バスの補足については、**2.3 上位アドレス・バスの補足** で説明します。

ペリフェラルの初期化が終了すると、ユーザーアプリケーションコードのダウンロードを実行します。ユーザー・アプリケーション・コードはブート・テーブルの形式にする必要があります。図 2 にブート・テーブルの形式を示します。ブート・テーブルの作成方法は、**3 ROMイメージの作成** で説明します。ユーザー・アプリケーション・コードのダウンロードの流れを以下に示します。まず、エントリー・アドレスをリードします。次に、セクションのサイズをリードします。セクション・サイズが 0 でない場合には、1セクション分のセクションデータをダウンロードします。セクションデータのダウンロードが完了したら、次のセクションのセクション・サイズをリードします。セクション・サイズが 0 だった場合、ブートを終了してエントリー・アドレスに分岐します。

図 2 ブート・テーブルの形式



2.2 ビット・モードの選択

フラッシュメモリの先頭バイトは、8ビット・モードか16ビット・モードかを決定するフィールドして使用されます。サンプルのセカンド・ブートローダは16ビットモードを使用しています。

```
        .sect     ".boot_load"
; Parallel Flash Boot Mode Field
        .word     01010101h
```

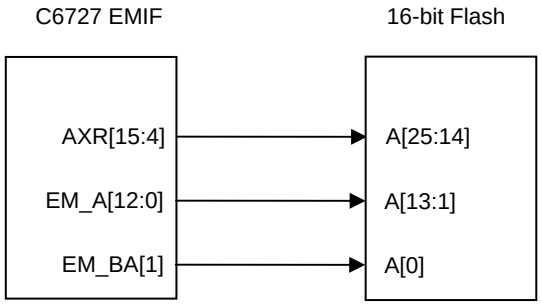
8ビットモードを使用する場合には、以下のように変更してください。

```
        .sect     ".boot_load"
; Parallel Flash Boot Mode Field
        .word     00000000h
```

2.3　上位アドレス・バスの補足

C672x EMIF のアドレス・バスは、BGAパッケージの場合 13本、QFPパッケージの場合 12本で構成されています。アドレスバスが足りない場合、GPIO等で補足する必要があります。サンプルのセカンド・ブートローダでは、AXR[15:4] ピンを EM_A[24:14] の代わりに使用しています。図 3 はAXR[15:4] を上位アドレスとして使用する場合のC6727 (BGA) と 16ビット・フラッシュとの接続図を示しています。

図 3 C6727 EMIF (BGAパッケージ) と 16ビット・フラッシュメモリの接続例



C672x EMIF のアドレス・バスは、常に32ビット・ワードアドレスを出力します。一方、ROMイメージ内ではバイト・アドレスが使用されるため、AXR[15:4] にはバイト・アドレスの A[26:14] を設定する必要があります。上位アドレスとして使用するピンを変更する場合、またQFPパッケージのデバイスを使用する場合には、上位アドレスの設定部分を変更してください。サンプルのセカンド・ブートローダ内には、上位アドレスの設定部分が3箇所あります。

以下に、AXR[15:4]に上位アドレスを設定するアセンブリ・コードの例を示します。

リスト1 AXR[15:4]に上位アドレスを設定するアセンブリ・コード例

```
                extu    a3, 5, 20, a6           ; extract A[26:15]
                shl     a6, 4, a0               ; AXR[15:4] <- a6 << 4      stw     a0, *a8
RL21:           ldw     *a8, a9                 ; wait till PDOUT is set correctly
        nop     4
                extu    a9, 16, 20, a9          ; extract AXR[15:4]
                cmpeq   a9, a6, a2              ; compare AXR[15:4] with A[26:15]
        [!a2]   b       RL21
```

3　ROM イメージの作成

ROMイメージは以下の手順で作成します。

1. セカンド・ブートローダをプロジェクトに追加します。
2. リンカ・コマンドファイルを修正して、セカンド・ブートローダを 0x10000000-0x100003FF に配置します。サンプルのセカンド・ブートローダは、.boot_load というセクションとして定義されています。この .boot_load セクションを0x10000000-0x100003FF に配置します。
3. hex6x を使用して hex ファイルを生成します。hex ファイルはブート・テーブル形式にする必要があります。以下に hex 変換用のコマンドファイルの例を示します。

リスト2 A hex 変換用のコマンドファイル例

```
abc.out                         /* input COFF file */
-map abc.map                    /* generate hex map file */
-m3                             /* Motorola S3 format */
-memwidth 8                     /* 8-bit memory system */
-boot                           /* create boot table */
-bootorg 0x90000400             /* copy table address */
-bootsection .boot_load 0x90000000          /* give boot section & addr */

ROMS
{
FLASH: org = 0x90000000, len = 0x800000, romwidth = 16, files = {abc.hex}      /* output hex file */
}
```

hex 形式への変換方法に関する詳細は、「TMS320C6000 Assembly Language Tools User's Guide [SPRU186]」をご参照ください。

4　参考文献

1. Using the TMS320C672x Bootloader（文書番号 SPRAA69）
2. TMS320C6000 Assembly Language Tools User's Guide（文書番号SPRU186）

Appendix A.　　セカンド・ブートローダの例

リスト3 セカンド･ブートローダ例

```
;  Copyright 2006 by Texas Instruments Japan Limited.
;  All rights reserved. Property of Texas Instruments Japan Limited.
;  You may modify the programs on the condition of using the programs
;  solely and exclusively with semiconductor devices manufactured  by
;  or for TI.
;
;  THE PROGRAMS ARE PROVIDED "AS IS". TIJ MAKES NO WARRANTIES OR
;  REPRESENTATIONS, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING ANY
;  IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE,
;  LACK OF VIRUSES, ACCURACY OR COMPLETENESS OF RESPONSES, RESULTS AND LACK OF
;  NEGLIGENCE. TIJ DISCLAIMS ANY WARRANTY OF TITLE, QUIET ENJOYMENT, QUIET
;  POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY
;  RIGHTS WITH REGARD TO THE PROGRAMS OR YOUR USE OF THOSE PROGRAMS.
;
;
;  Second bootloader example for C672x
;
;  FIRST RELEASE DATE          : Dec/12/2006
;  LAST UPDATE                 : Dec/21/2006
;

;***************************** CAUTION **********************************
;
; THIS FUNCTION ONLY APPEARS AT SECOND BOOT LOAD
;
;***************************** CAUTION **********************************

                                .sect      ".boot_load"

; Parallel Flash Boot Mode Field

                                .word     01010101h

;***************************************************************************
; PLL init
;***************************************************************************

_pllInit:

; Required register addresses
PLLBASEADDR          .set       41000000h
```

```
PLLPID          .set    000h + PLLBASEADDR      ; PLL controller peripheral identification register
PLLCSR          .set    100h + PLLBASEADDR      ; PLL control/status register
PLLM            .set    110h + PLLBASEADDR      ; PLL multiplier control register
PLLDIV0         .set    114h + PLLBASEADDR      ; PLL controller divider register 0
PLLDIV1         .set    118h + PLLBASEADDR      ;  PLL controller divider register 1
PLLDIV2         .set    11Ch + PLLBASEADDR      ; PLL controller divider register 2
PLLDIV3         .set    120h + PLLBASEADDR      ; PLL controller divider register 3
PLLCMD          .set    138h + PLLBASEADDR      ; PLL controller command register
PLLSTAT         .set    13Ch + PLLBASEADDR      ; PLL controller status register
ALNCTL          .set    140h + PLLBASEADDR      ; PLL controller clock align control register
CKEN            .set    148h + PLLBASEADDR      ; Clock enable control register
CKSTAT          .set    14Ch + PLLBASEADDR      ; Clock status register
SYSTAT          .set    150h + PLLBASEADDR      ; SYSCLK status register

CFGBRIDGE       .set    40000024h

; Definition values which are used in PLL settings
TIMES12         .set    000Ch
TIMES13         .set    000Dh

DIVENABLED      .set    8000h
DIV1            .set    0000h
DIV2            .set    0001h
DIV3            .set    0002h
DIV4            .set    0003h
DIV5            .set    0004h
DIV6            .set    0005h

PLLGOSET        .set    0001h

PLLALN1         .set    0001h
PLLALN2         .set    0002h
PLLALN3         .set    0004h

; In PLLCSR, write PLLEN = 0 (bypass mode).
; Wait 4 cycles of the slowest of PLLOUT or reference clock source (CLKIN or OSCIN).
                mvkl    PLLCSR, a0
                mvkh    PLLCSR, a0
                ldw     *a0, a1
                nop     4
                clr     a1, 0, 1, a1                    ;PLLEN = 0
```

```
                stw     a1, *a0
                nop     9

;. In PLLCSR, write PLLRST = 1 (PLL is reset).
                set     a1, 3, 4, a1        ;PLLRST = 1
                stw     a1, *a0

; Program PLLDIV0 and PLLM.

                mvkl    PLLDIV0, a2
                mvkh    PLLDIV0, a2
                mvkl    DIVENABLED|DIV1, a3
                mvkh    DIVENABLED|DIV1, a3
                stw     a3, *a2                     ; PLLDIV0 = DIVENABLED | DIV1

                mvkl    PLLM, a4
                mvkh    PLLM, a4
                mvkl    TIMES12, a5
                mvkh    TIMES12, a5                 ; PLLM = TIMES12
                stw     a5, *a4

; Program PLLDIV1-n. Note that you must apply the GO operation to change these dividers to new ratios.
; Check that the GOSTAT bit in PLLSTAT is cleared to 0 to show that no GO operation is currently in progress.
                mvkl    PLLSTAT, a6
                mvkh    PLLSTAT, a6
_poll_pllstat0:
                ldw     *a6, b1
                nop     4
                extu    b1, 31, 31, b0
 [b0]   b       _poll_pllstat0                      ; while (*(int *)PLLSTAT == PLLGOWAIT);
                nop     5
; Program the RATIO field in PLLDIVn to the desired new divide-down rate.

                mvkl    PLLDIV1, a8
                mvkh    PLLDIV1, a8
                mvkl    DIVENABLED|DIV1, a9
                mvkh    DIVENABLED|DIV1, a9
                stw     a9, *a8                     ; PLLDIV1 = DIVENABLED | DIV1
                nop     9

                mvkl    PLLDIV2, a8
                mvkh    PLLDIV2, a8
                mvkl    DIVENABLED|DIV2, a9
```

```
                mvkh    DIVENABLED|DIV2, a9
                stw     a9, *a8                         ; PLLDIV2 = DIVENABLED | DIV2
                nop     9

                mvkl    PLLDIV3, a8
                mvkh    PLLDIV3, a8
                mvkl    DIVENABLED|DIV3, a9
                mvkh    DIVENABLED|DIV3, a9
                stw     a9, *a8                         ; PLLDIV3 = DIVENABLED | DIV3
                nop     9

; Set the ALN1-3 bits in ALNCTL to 1 so that SYSCLK1-3 are aligned after the GO operation.
                mvkl    ALNCTL, a8
                mvkh    ALNCTL, a8
                mvkl    PLLALN1|PLLALN2|PLLALN3, a9
                mvkh    PLLALN1|PLLALN2|PLLALN3, a9
                stw     a9, *a8                         ; PLLALNCTL = PLLALN1 | PLLALN2 | PLLALN3

; Set the GOSET bit in PLLCMD to 1 to initiate the GO operation to change the divide values and align SYSCLK1-3.
                mvkl    PLLCMD, a8
                mvkh    PLLCMD, a8
                mvkl    PLLGOSET, a9
                mvkh    PLLGOSET, a9
                stw     a9, *a8                         ; PLLCMD = PLLGOSET

; Read the GOSTAT bit in PLLSTAT
_poll_pllstat1:
                ldw     *a6, b1
                nop     4
                extu    b1, 31, 31, b0
        [b0]    b       _poll_pllstat1                  ; while (*(int *)PLLSTAT == PLLGOWAIT);


; Wait for PLL to properly reset. See device-specific data manual for PLL reset time.
; 125 ns < 4 cycles @ 25MHz
                nop     4

; In PLLCSR, write PLLRST = 0 to bring PLL out of reset.
                clr     a1, 3, 4, a1                    ;PLLRST = 0
                stw     a1, *a0

; Wait for PLL to lock. See device-specific data manual for PLL lock time.
```

```
; 187500 ns < 4688 cycles @ 25MHz
                mvkl    782, b0  ; 782 *6 = 4692
_wait_plllock
        [b0]    b       _wait_plllock
                sub     b0, 1, b0
                nop     4

; In PLLCSR, write PLLEN = 1 to enable PLL mode.
                ldw     *a0, a1
                nop     4
                set     a1, 0, 1, a1                    ;PLLEN = 1
                stw     a1, *a0

; CSP Bridge reset */
                mvkl    CFGBRIDGE, b4
                mvkh    CFGBRIDGE, b4
                mvkl    1, b5
                stw     b5, *b4                         ; CFGBRIDGE = 1
                mvkl    0, b5
                nop     9
                stw     b5, *b4                         ; CFGBRIDGE = 0

;*****************************************************************************
; EMIF init
;*****************************************************************************

_emifInit:

; Required register addresses
EMIFBASEADDR    .set    0F0000000h

AWCCR           .set    04h + EMIFBASEADDR      ; Asynchronous Wait Cycle Configuration Register
SDCR            .set    08h + EMIFBASEADDR      ; SDRAM Configuration Register
SDRCR           .set    0Ch + EMIFBASEADDR      ; SDRAM Refresh Control Register
A1CR            .set    10h + EMIFBASEADDR      ; Asynchronous 1 Configuration Register
SDTIMR          .set    20h + EMIFBASEADDR      ; SDRAM Timing Register
SDSRETR         .set    3Ch + EMIFBASEADDR      ; SDRAM Self Refresh Exit Timing Register
EIRR            .set    40h + EMIFBASEADDR      ; EMIF Interrupt Raw Register
EIMR            .set    44h + EMIFBASEADDR      ; EMIF Interrupt Mask Register
EIMSR           .set    48h + EMIFBASEADDR      ; EMIF Interrupt Mask Set Register
EIMCR           .set    4Ch + EMIFBASEADDR      ; EMIF Interrupt Mask Clear Register

; Definition values which are used in PLL settings
```

```
AWCCR_WP0           .set    0 << 28          ; WP0     : Insert wait cycle if AWAIT is low
AWCCR_MEMC          .set    16 <<  0         ; MEMC    : Max extended wait cycles (max = 16*(MEWC+1))

A1CR_SS             .set    0 << 31          ; SS      : Select WE mode
A1CR_EW             .set    1 << 30          ; EW      : Enable extended wait mode
A1CR_W_SETUP        .set    0 << 26          ; W_SETUP : 10 ns @ 100 MHz
A1CR_W_STROBE       .set    9 << 20          ; W_STROBE : 90 ns @ 100 MHz
A1CR_W_HOLD         .set    3 << 17          ; W_HOLD  : 40 ns @ 100 MHz
A1CR_R_SETUP        .set    0 << 13          ; R_SETUP : 10 ns @ 100 MHz
A1CR_R_STROBE       .set    9 <<  7          ; R_STROBE : 100 ns @ 100 MHz
A1CR_R_HOLD         .set    3 <<  4          ; R_HOLD  : 40 ns @ 100 MHz
A1CR_TA             .set    0 <<  2          ; TA      : 10 ns @ 100 MHz
A1CR_ASIZE          .set    1 <<  0          ; ASIZE   : 16-bit data bus

SDTIMR_T_RFC        .set    6 << 27          ; T_RFC : Auto refresh period (66ns)
SDTIMR_T_RP         .set    1 << 24          ; T_RP  : Precharge command period (20 ns)
SDTIMR_T_RCD        .set    1 << 20          ; T_RCD : Active to read or write delay (20 ns)
SDTIMR_T_WR         .set    1 << 16          ; T_WR  : Write recovery time (1 clk + 7.5 ns)
SDTIMR_T_RAS        .set    4 << 12          ; T_RAS : Active to precharge command (44 ns)
SDTIMR_T_RC         .set    6 <<  8          ; T_RC  : Active to active command period (66 ns)
SDTIMR_T_RRD        .set    1 <<  4          ; T_RRD : Active bank a to active bank b command (15 ns)

SDSRETR_T_RC        .set    6                ; T_RFC : Auto refresh period (66ns)

SDRCR_RR0           .set    4E2h             ; RR > (100 us/8) * fem_CLK = 1250 (0x4E2)
SDRCR_RR1           .set    5F8h             ; 64 ms, 4096 cycle refresh rate

SDCR_SR             .set    0 << 31          ; SR          : To avoid self refresh state
SDCR_NM             .set    1 << 14          ; NM          : narrow (16-bit)
SDCR_CL             .set    2 <<  9          ; CL          : 2 clock
SDCR_BIT11_9LOCK    .set    1 <<  8          ; BIT11_9LOCK : To allow the CL field to be written (legacy)
SDCR_IBANK          .set    2 <<  4          ; IBANK       : 4 internal SDRAM banks
SDCR_PAGESIZE       .set    2 <<  0;         ; PAGESIZE    : 1024 word pages


; Configure the Async memory (FPGA), EM_CS[2] space
; EMIF Asynchronous Wait Cycle Configuration Register
                mvkl    AWCCR, a0
                mvkh    AWCCR, a0
                mvkl    AWCCR_WP0|AWCCR_MEMC, a1
                mvkh    AWCCR_WP0|AWCCR_MEMC, a1
                stw     a1, *a0
```

```
; EMIF global control register
                mvkl    A1CR, a0
                mvkh    A1CR, a0
                mvkl    A1CR_SS|A1CR_EW|A1CR_W_SETUP|A1CR_W_STROBE|A1CR_W_HOLD|
                        A1CR_R_SETUP|A1CR_R_STROBE|A1CR_R_HOLD|A1CR_TA|A1CR_ASIZE, a1
                mvkh    A1CR_SS|A1CR_EW|A1CR_W_SETUP|A1CR_W_STROBE|A1CR_W_HOLD|
                        A1CR_R_SETUP|A1CR_R_STROBE|A1CR_R_HOLD|A1CR_TA|A1CR_ASIZE, a1
                stw     a1, *a0

; Configure the SDRAM, EM_CS[0] space:
;
; Following is the procedure to be followed if the SDRAM Power-up constraint was violated (Procedure B):
; Program SDTIMR and SDSRETR to satisfy the timing requirements for the attached SDRAM device.
; The timing parameters should be taken from the SDRAM datasheet.
;
; Program AC timing registers to meet SDRAM spec
                mvkl    SDTIMR, a0
                mvkh    SDTIMR, a0
                mvkl    SDTIMR_T_RFC|SDTIMR_T_RP|SDTIMR_T_RCD|SDTIMR_T_WR|SDTIMR_T_RAS|
                        SDTIMR_T_RC|SDTIMR_T_RRD, a1
                mvkh    SDTIMR_T_RFC|SDTIMR_T_RP|SDTIMR_T_RCD|SDTIMR_T_WR|SDTIMR_T_RAS|
                        SDTIMR_T_RC|SDTIMR_T_RRD, a1
                stw     a1, *a0

                mvkl    SDSRETR, a0
                mvkh    SDSRETR, a0
                mvkl    SDSRETR_T_RC, a1
                mvkh    SDSRETR_T_RC, a1
                stw     a1, *a0
; Program the RR field of SDRCR such that the following equation is satisfied: (RR * 8) / (fEM_CLK) > 100
; µs (or 200µs). For example, an EM_CLK frequency of 100MHz would require setting RR to 1251
; (0x4E3) or higher to meet a 100µs constraint.
;
; Program the Refresh Rate to satisfy the SDRAM power-up constraint
                mvkl    SDRCR, a2
                mvkh    SDRCR, a2
                mvkl    SDRCR_RR0, a3
                mvkh    SDRCR_RR0, a3
                stw     a3, *a2


; Program SDCR to match the characteristics of the attached SDRAM device. This will cause the
```

```
; auto-initialization sequence in Section 2.4.4 to be re-run with the new value of RR.
;
;.Program the Control Register to meet SDRAM spec
                mvkl    SDCR, a0
                mvkh    SDCR, a0
                mvkl    SDCR_SR|SDCR_NM|SDCR_CL|SDCR_BIT11_9LOCK|SDCR_IBANK|
                        SDCR_PAGESIZE, a1
                mvkh    SDCR_SR|SDCR_NM|SDCR_CL|SDCR_BIT11_9LOCK|SDCR_IBANK|
                        SDCR_PAGESIZE, a1
                stw     a1, *a0

; Perform a read from the SDRAM to guarantee that step 5 of this procedure will occur after the
; initialization process has completed. Alternatively, wait for 200 µs instead of performing a read.
; 200 µs < 60607 @ 300MHz < 10102 Branch instruction
                mvkl    2776h, b0
_poll_sdram_auto_init:
 [b0]   b       _poll_sdram_auto_init
                sub     b0, 1, b0
                nop     4

; Finally, program the RR field to match that of the attached device's refresh interval. See
; Section 2.4.6.1 details on determining the appropriate value.
                mvkl    SDRCR_RR1, a3
                mvkh    SDRCR_RR1, a3
                stw     a3, *a2

;*****************************************************************************
; section load
;*****************************************************************************
MCASP0BASEADDR          .set    44000000h
FLASHADDR_PAGEMASK      .set    90007fffh

PFUNC                   .set    010h + MCASP0BASEADDR       ; MCASP pin function register
PDIR                    .set    014h + MCASP0BASEADDR       ; MCASP pin direction register
PDOUT                   .set    018h + MCASP0BASEADDR       ; MCASP pin data output register

PFUNC_AXR15             .set    1 << 15         ; pin function as GPIO
PFUNC_AXR14             .set    1 << 14         ; pin function as GPIO
PFUNC_AXR13             .set    1 << 13         ; pin function as GPIO
PFUNC_AXR12             .set    1 << 12         ; pin function as GPIO
PFUNC_AXR11             .set    1 << 11         ; pin function as GPIO
PFUNC_AXR10             .set    1 << 10         ; pin function as GPIO
```

```
PFUNC_AXR9              .set    1 << 9          ; pin function as GPIO
PFUNC_AXR8              .set    1 << 8          ; pin function as GPIO
PFUNC_AXR7              .set    1 << 7          ; pin function as GPIO
PFUNC_AXR6              .set    1 << 6          ; pin function as GPIO
PFUNC_AXR5              .set    1 << 5          ; pin function as GPIO
PFUNC_AXR4              .set    1 << 4          ; pin function as GPIO

PDIR_AXR15              .set    1 << 15         ; pin function as output
PDIR_AXR14              .set    1 << 14         ; pin function as output
PDIR_AXR13              .set    1 << 13         ; pin function as output
PDIR_AXR12              .set    1 << 12         ; pin function as output
PDIR_AXR11              .set    1 << 11         ; pin function as output
PDIR_AXR10              .set    1 << 10         ; pin function as output
PDIR_AXR9               .set    1 << 9          ; pin function as output
PDIR_AXR8               .set    1 << 8          ; pin function as output
PDIR_AXR7               .set    1 << 7          ; pin function as output
PDIR_AXR6               .set    1 << 6          ; pin function as output
PDIR_AXR5               .set    1 << 5          ; pin function as output
PDIR_AXR4               .set    1 << 4          ; pin function as output

; Configure AXR[15:4] pins as EMIF_A[26:15]
                mvkl    PFUNC, a0
                mvkh    PFUNC, a0
                mvkl    PFUNC_AXR15|PFUNC_AXR14|PFUNC_AXR13|PFUNC_AXR12|PFUNC_AXR11|
                        PFUNC_AXR10|PFUNC_AXR9|PFUNC_AXR8|PFUNC_AXR7|PFUNC_AXR6|
                        PFUNC_AXR5|PFUNC_AXR4, a1
                mvkh    PFUNC_AXR15|PFUNC_AXR14|PFUNC_AXR13|PFUNC_AXR12|PFUNC_AXR11|
                        PFUNC_AXR10|PFUNC_AXR9|PFUNC_AXR8|PFUNC_AXR7|PFUNC_AXR6|
                        PFUNC_AXR5|PFUNC_AXR4, a1
                stw     a1, *a0

; configure AXR[15:4] pins as output
                mvkl    PDIR, a0
                mvkh    PDIR, a0
                mvkl    PDIR_AXR15|PDIR_AXR14|PDIR_AXR13|PDIR_AXR12|PDIR_AXR11|PDIR_AXR10|
                        PDIR_AXR9|PDIR_AXR8|PDIR_AXR7|PDIR_AXR6|PDIR_AXR5|PDIR_AXR4, a1
                mvkh    PDIR_AXR15|PDIR_AXR14|PDIR_AXR13|PDIR_AXR12|PDIR_AXR11|PDIR_AXR10|
                        PDIR_AXR9|PDIR_AXR8|PDIR_AXR7|PDIR_AXR6|PDIR_AXR5|PDIR_AXR4, a1
                stw     a1, *a0

                ; clear PDOUT
                mvkl    PDOUT, a8
                mvkh    PDOUT, a8
```

```
                zero    a1
                stw     a1, *a8

; second boot loader
                mvkl    FLASHADDR_PAGEMASK, a10
                mvkh    FLASHADDR_PAGEMASK, a10

                mvkl    COPY_TABLE, a3          ; load table pointer
                mvkh    COPY_TABLE, a3
                zero    a6                      ; page address
                ldw     *a3++, b1               ; load entry point

copy_section_top:
                extu    a3, 5, 20, a6           ; extract A[26:15]
                shl     a6, 4, a0               ; AXR[15:4] <- a6 << 4
                stw     a0, *a8
RL21:   ldw     *a8, a9                         ; wait till PDOUT is set correctly
                nop     4
                extu    a9, 16, 20, a9          ; extract AXR[15:4]
                cmpeq   a9, a6, a2              ; compare AXR[15:4] with A[26:15]
        [!a2]   b       RL21
                nop     5
                ldw     *a3++, b0               ; read byte count

                extu    a3, 5, 20, a6           ; extract A[26:15]
                shl     a6, 4, a0               ; AXR[15:4] <- a6 << 4
                stw     a0, *a8
RL22:   ldw     *a8, a9                         ; wait till PDOUT is set correctly
                nop     4
                extu    a9, 16, 20, a9          ; extract AXR[15:4]
                cmpeq   a9, a6, a2              ; compare AXR[15:4] with A[26:15]
        [!a2]   b       RL22
                nop     5
                ldw     *a3++, a4               ; section start address

        [!b0]   b       copy_done               ; have we copied all sections?
                nop     5

copy_loop:
                extu    a3, 5, 20, a6           ; extract A[26:15]
                shl     a6, 4, a0               ; AXR[15:4] <- a6 << 4
                stw     a0, *a8
```

```
RL23:   ldw     *a8, a9                         ; wait till PDOUT is set correctly
                nop     4
                extu    a9, 16, 20, a9          ; extract AXR[15:4]
                cmpeq   a9, a6, a2              ; compare AXR[15:4] with A[26:15]
        [!a2]   b       RL23
                nop     5

                and     a10, a3, a11            ; clear A[26:15] to 0
                add     1, a3, a3

                ldb     *a11, b5
                sub     b0, 1, b0               ; decrement counter
        [ b0]   b       copy_loop               ; setup branch if not done
        [!b0]   b       copy_section_top
                zero    a1
        [!b0]   and     3, a3, a1
                stb     b5, *a4++
        [!b0]   and     -4, a3, a5              ; round address up to next multiple of 4
        [ a1]   add     4, a5, a3               ; round address up to next multiple of 4

copy_done:
                b       b1
                nop     5

;Address of the generated copy table
COPY_TABLE              .equ            0x90000400
```

ご注意

日本テキサス・インスツルメンツ株式会社（以下TIJといいます）及びTexas Instruments Incorporated（TIJの親会社、以下TIJないしTexas Instruments Incorporatedを総称してTIといいます）は、その製品及びサービスを任意に修正し、改善、改良、その他の変更をし、もしくは製品の製造中止またはサービスの提供を中止する権利を留保します。従いまして、お客様は、発注される前に、関連する最新の情報を取得して頂き、その情報が現在有効かつ完全なものであるかどうかご確認下さい。全ての製品は、お客様とTIJとの間に取引契約が締結されている場合は、当該契約条件に基づき、また当該取引契約が締結されていない場合は、ご注文の受諾の際に提示されるTIJの標準販売契約約款に従って販売されます。

TIは、そのハードウェア製品が、TIの標準保証条件に従い販売時の仕様に対応した性能を有していること、またはお客様とTIJとの間で合意された保証条件に従い合意された仕様に対応した性能を有していることを保証します。検査およびその他の品質管理技法は、TIが当該保証を支援するのに必要とみなす範囲で行なわれております。各デバイスの全てのパラメーターに関する固有の検査は、政府がそれ等の実行を義務づけている場合を除き、必ずしも行なわれておりません。

TIは、製品のアプリケーションに関する支援もしくはお客様の製品の設計について責任を負うことはありません。TI製部品を使用しているお客様の製品及びそのアプリケーションについての責任はお客様にあります。TI製部品を使用したお客様の製品及びアプリケーションについて想定されうる危険を最小のものとするため、適切な設計上および操作上の安全対策は、必ずお客様にてお取り下さい。

TIは、TIの製品もしくはサービスが使用されている組み合せ、機械装置、もしくは方法に関連しているTIの特許権、著作権、回路配置利用権、その他のTIの知的財産権に基づいて何らかのライセンスを許諾するということは明示的にも黙示的にも保証も表明もしておりません。TIが第三者の製品もしくはサービスについて情報を提供することは、TIが当該製品もしくはサービスを使用することについてライセンスを与えるとか、保証もしくは是認するということを意味しません。そのような情報を使用するには第三者の特許その他の知的財産権に基づき当該第三者からライセンスを得なければならない場合もあり、またTIの特許その他の知的財産権に基づきTI からライセンスを得て頂かなければならない場合もあります。

TIのデータ・ブックもしくはデータ・シートの中にある情報を複製することは、その情報に一切の変更を加えること無く、かつその情報と結び付られた全ての保証、条件、制限及び通知と共に複製がなされる限りにおいて許されるものとします。当該情報に変更を加えて複製することは不公正で誤認を生じさせる行為です。TIは、そのような変更された情報や複製については何の義務も責任も負いません。

TIの製品もしくはサービスについてTIにより示された数値、特性、条件その他のパラメーターと異なる、あるいは、それを超えてなされた説明で当該TI製品もしくはサービスを再販売することは、当該TI製品もしくはサービスに対する全ての明示的保証、及び何らかの黙示的保証を無効にし、かつ不公正で誤認を生じさせる行為です。TIは、そのような説明については何の義務も責任もありません。

TIは、TIの製品が、安全でないことが致命的となる用途ないしアプリケーション（例えば、生命維持装置のように、TI製品に不良があった場合に、その不良により相当な確率で死傷等の重篤な事故が発生するようなもの）に使用されることを認めておりません。但し、お客様とTIの双方の権限有る役員が書面でそのような使用について明確に合意した場合は除きます。たとえTIがアプリケーションに関連した情報やサポートを提供したとしても、お客様は、そのようなアプリケーションの安全面及び規制面から見た諸問題を解決するために必要とされる専門的知識及び技術を持ち、かつ、お客様の製品について、またTI製品をそのような安全でないことが致命的となる用途に使用することについて、お客様が全ての法的責任、規制を遵守する責任、及び安全に関する要求事項を満足させる責任を負っていることを認め、かつそのことに同意します。さらに、もし万一、TIの製品がそのような安全でないことが致命的となる用途に使用されたことによって損害が発生し、TIないしその代表者がその損害を賠償した場合は、お客様がTIないしその代表者にその全額の補償をするものとします。

TI製品は、軍事的用途もしくは宇宙航空アプリケーションないし軍事的環境、航空宇宙環境にて使用されるようには設計もされていませんし、使用されることを意図されておりません。但し、当該TI製品が、軍需対応グレード品、若しくは「強化プラスティック」製品としてTIが特別に指定した製品である場合は除きます。TIが軍需対応グレード品として指定した製品のみが軍需品の仕様書に合致いたします。お客様は、TIが軍需対応グレード品として指定していない製品を、軍事的用途もしくは軍事的環境下で使用することは、もっぱらお客様の危険負担においてなされるということ、及び、お客様がもっぱら責任をもって、そのような使用に関して必要とされる全ての法的要求事項及び規制上の要求事項を満足させなければならないことを認め、かつ同意します。

TI製品は、自動車用アプリケーションないし自動車の環境において使用されるようには設計されていませんし、また使用されることを意図されておりません。但し、TIがISO/TS 16949の要求事項を満たしていると特別に指定したTI製品は除きます。お客様は、お客様が当該TI指定品以外のTI製品を自動車用アプリケーションに使用しても、TIは当該要求事項を満たしていなかったことについて、いかなる責任も負わないことを認め、かつ同意します。

弊社半導体製品の取り扱い・保管について

半導体製品は、取り扱い、保管・輸送環境、基板実装条件によっては、お客様での実装前後に破壊/劣化、または故障を起こすことがあります。

弊社半導体製品のお取り扱い、ご使用にあたっては下記の点を遵守して下さい。

1. 静電気
   - 素手で半導体製品単体を触らないこと。どうしても触る必要がある場合は、リストストラップ等で人体からアースをとり、導電性手袋等をして取り扱うこと。
   - 弊社出荷梱包単位（外装から取り出された内装及び個装）又は製品単品で取り扱いを行う場合は、接地された導電性のテーブル上で（導電性マットにアースをとったもの等）、アースをした作業者が行うこと。また、コンテナ等も、導電性のものを使うこと。
   - マウンタやはんだ付け設備等、半導体の実装に関わる全ての装置類は、静電気の帯電を防止する措置を施すこと。
   - 前記のリストストラップ・導電性手袋・テーブル表面及び実装装置類の接地等の静電気帯電防止措置は、常に管理されその機能が確認されていること。
2. 温・湿度環境
   - 温度：0～40℃、相対湿度：40～85%で保管・輸送及び取り扱いを行うこと。（但し、結露しないこと。）
   - 直射日光があたる状態で保管・輸送しないこと。
3. 防湿梱包
   - 防湿梱包品は、開封後は個別推奨保管環境及び期間に従い基板実装すること。
4. 機械的衝撃
   - 梱包品（外装、内装、個装）及び製品単品を落下させたり、衝撃を与えないこと。
5. 熱衝撃
   - はんだ付け時は、最低限260℃以上の高温状態に、10秒以上さらさないこと。（個別推奨条件がある時はそれに従うこと。）
6. 汚染
   - はんだ付け性を損なう、又はアルミ配線腐食の原因となるような汚染物質（硫黄、塩素等ハロゲン）のある環境で保管・輸送しないこと。
   - はんだ付け後は十分にフラックスの洗浄を行うこと。（不純物含有率が一定以下に保証された無洗浄タイプのフラックスは除く。）

以上