*Application Report*
*SPRA628 - March 2000*

# *TMS320VC5421 Bootloader Technical Reference*

*Tai Nguyen, Bill Winderweedle* *C5000 Applications Team*

**ABSTRACT**

A bootloader is provided on the TMS320VC5421 for loading user code to on-chip memory at reset. The bootloader code is contained within the 2K ROM in each of the two DSP subsystems in ’5421. Support is provided in ROM code for three different modes of boot load operation: parallel 8-bit, parallel 16-bit, and serial EEPROM. In addition, HPI boot mode is supported with the ROM disabled.

Unless otherwise noted, the information contained in this document should be considered ADVANCE INFORMATION on new products in the sampling or pre-production phase of development. Information and specifications in this document are subject to change without notice.

**Contents**

**List of Figures**

**List of Tables**

## 1 Introduction

An important aspect of microprocessor based systems is the bootload process, in which the system is initialized and normal operations begin. A typical ’5421 based system includes a primary host processor, and the ’5421 functioning as a secondary slave processor. At system reset, the host processor is usually used to initialize the ’5421. These systems require an initialization process for the ’5421 to accept information from the host processor whenever the system is reset.

Other systems consist of some nonvolatile memory (i.e., EPROM), as well as some volatile memory (i.e., RAM). The nonvolatile memory is usually too slow to execute real time code from. Therefore, it is only used to initialize the faster RAM. For this type of system, it is desirable to have an initialization, or bootload, process that automatically copies the system program from the nonvolatile memory to the fast RAM when the system is reset.

The ’5421 bootloader is a program contained in the on-chip ROM that can be used to initialize the system memory after reset. Several options or boot modes are available to support various methods of initialization. The following boot modes are included in the ’5421 bootloader:

- Parallel-16 Boot Mode: loads code via 16-bit wide asynchronous memory device using the external memory interface.
- Parallel-8 Boot Mode: loads code via 8-bit wide asynchronous memory device using the external memory interface.
- Serial EEPROM Boot Mode: loads code via an external master device such as 8-bit wide serial EEPROMs using McBSP 2.

Unlike many other ’54x devices, the ROM bootloader on the ’5421 does not provide the HPI, parallel I/O, and standard serial port boot modes. HPI boot can be accomplished by placing the device in reset, loading the code into internal RAM, and then releasing the device from reset with the ROM disabled (XIO=0). This is similar to the RAM loading process used on the ’5420, except that there is no ROM on the ’5420. For details on reset sequencing, refer to Bootloader Operating Modes, in the TMS320VC5421 datasheet (SPRS098).

The bootloader allows multiple memory sections to be loaded into different destination addresses within the system RAM. These sections can be either program or data. However, the bootloader cannot directly boot into data space. If sections must be placed in data memory space, these sections can initially be stored in program space, and then copied over by the main program at run time. Another way of initializing sections in data space is using the on-chip DARAM that can be mapped into both program space and data space. Otherwise, sections to be stored in data memory must follow the sections to be stored in program memory and these sections will be booted last.

After reset, if the on-chip ROM is enabled (XIO = 1 and GPIO0/ROMEN = 1), the ’5421 automatically begins execution of the bootloader. After the initialization is performed, the bootloader loads the system RAM according to the boot mode selected and then causes the ’5421 to begin execution of the loaded code. At this point, the bootload process is complete, and the ’5421 performs the intended system function. Whenever the system is reset again with the boot ROM enabled, the ’5421 starts execution of the bootloader again, and the entire bootload process is repeated.

## 2 TMS320VC5421 Considerations

On the ’5421, the following items should be considered by the user and bootloader designer:

- The memory maps of the CPU and the DMA show the shared memory at different addresses (CPU and DMA memory maps are shown in Appendix A). The user must use the DMA memory map address for the load address appropriately and ensure the download sections must not exceed 32K words (i.e., the load address) for both data and program. Each section must be within the DMA memory addresses.
- The shared memory with respect to the DMA is part of two extended program pages (see attached DMA memory map). Since the DMA address registers are only 16 bits wide, the extended address register of the DMA also has to be programmed correctly.
- For parallel-8 and parallel-16 boot modes, external PROM devices should be mapped into external data memory with respect to the DMA memory map, to achieve the PROM maximum address range (i.e. 256K). For specific addresses, refer to the DMA External Data Memory Map in Appendix A. The data memory with respect to the CPU memory map is limited to 64K.
- The entry point of the application program must not utilize a program memory address within the range of 00_E000h to 00_FFFFh. This is to avoid possible conflicts with an enabled local ROM in this address range.
- DSP subsystem B/A must not attempt to use the external memory interface (EMIF) while the other DSP subsystem (A/B) is boot loading the application program. Each DSP subsystem may meet this requirement by remaining in reset or by executing only from on-chip memory (e.g., its own ROM or local DARAM). HPIRS places DSP subsystem A in control of the EMIF pins. For access to the EMIF after reset, DSP subsystem B must set the general purpose I/O control register bits (CORE SEL, XIO GRANT, XIO REQ). See the ’5421 datasheet, for more details on the EMIF arbitration between CPU XIO and DMA.
- Either core must not fetch any instructions from memory location 00 E000h to 00 FFFFh while the system is modifying the XIO pin.

## 3 Boot Modes

Bootloader mode selection is determined by the state of the GPIO1 pin and the first word read by the bootloader. If GPIO1 = 1, the bootloader will execute the SPI EEPROM boot mode. If GPIO1 = 0, the bootloader will determine which parallel boot mode to be executed. The table below describes the configuration of ’5421 pins for ROM enable/disable and boot modes.

**Table 1. TMS320VC5421 Pin Configuration for Various Boot Modes**

| XIO | GPIO0/ROMEN | GPIO1 | Description |
|---|---|---|---|
| 0 | x | x | ROM is disabled and 8K words of on-chip DARAM are mapped in. The CPU will fetch from internal RAM when released from reset. This configuration can be used for HPI boot. |
| 1 | 0 | x | ROM is disabled and 8K words of on-chip DARAM are mapped in. The CPU will fetch from external memory devices when released from reset. This configuration can be used when a custom bootloader resides in external memory devices. |
| 1 | 1 | 0 | ROM is enabled and 8K words of on-chip DARAM are mapped out. Parallel boot mode is selected. |
| 1 | 1 | 1 | ROM is enabled and the 8K words of on-chip DARAM are mapped out. Serial EEPROM boot mode is selected. |

The following flowchart shows the selection of modes for bootloader operation. In addition, the details of the boot sequence for each boot mode will be discussed in the following sections.

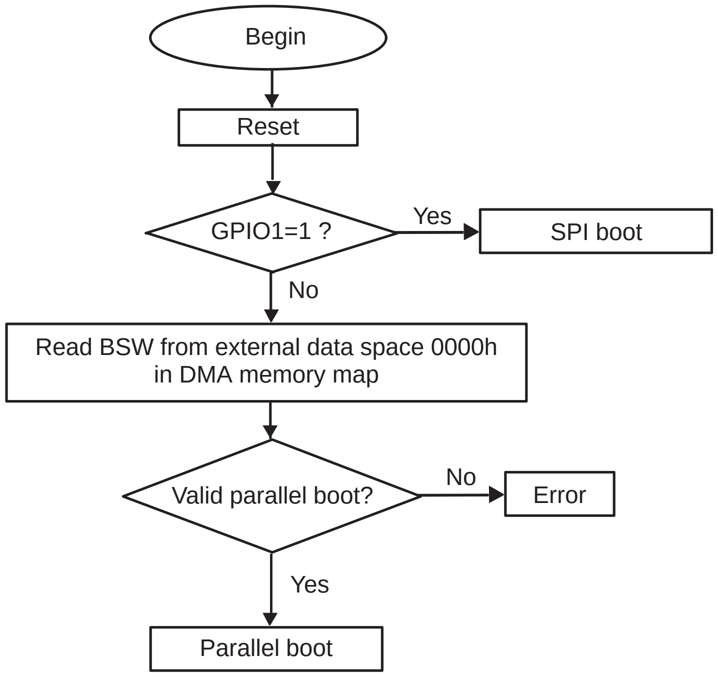

**Figure 1. Bootloader Mode Selection Flowchart**

## 3.1 Parallel Boot Mode

The parallel boot mode is intended for boot loading from a parallel EPROM or EEPROM device. This boot mode uses the external memory interface of the ’5421 to transfer code from 8-bit or 16-bit external data memory in DMA memory map to program space. The parallel boot mode is selected by providing a particular word at a certain external memory address.

## 3.2 Parallel Boot Selection

The parallel boot is selected via the GPIO1 pin. Proper selection of the boot mode requires a low state on the GPIO1 pin for a minimum of 50 CPU cycles after the ’5421 is reset. The bootloader will check the data memory location 0000h for the Boot Selection Word (BSW). The first word from this address will determine whether it is a valid code to boot and the data length. The first word at this address should be 10AAh for 16-bit parallel boot. For 8-bir parallel boot mode, the first two bytes will be 08h and Ah. The eight most significant bits indicate the memory width where the source program resides (8/16 bits wide). The eight least significant bits are the Bootloader Recognition Byte (BRB), which is 0AAh. If the first word is not 08AAh or 10AAh, then the bootloader will stay in an endless loop at the end of the bootloader program.

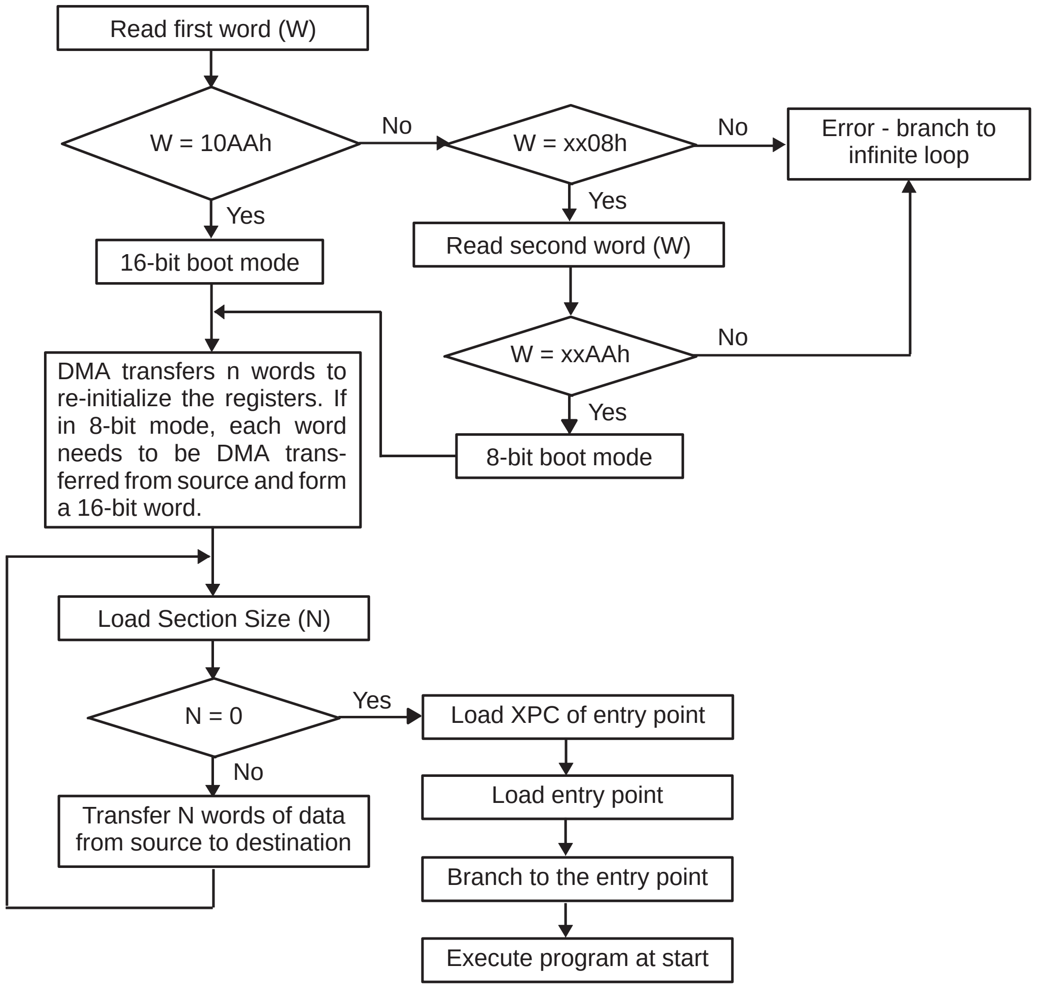
Read first word (W)
W = 10AAh
No
W = xx08h
No
Error - branch to infinite loop
Yes
Yes
16-bit boot mode
Read second word (W)
W = xxAAh
No
Yes
DMA transfers n words to re-initialize the registers. If in 8-bit mode, each word needs to be DMA transferred from source and form a 16-bit word.
8-bit boot mode
Load Section Size (N)
N = 0
Yes
Load XPC of entry point
No
Load entry point
Transfer N words of data from source to destination
Branch to the entry point
Execute program at start

**Figure 2. Parallel Boot Mode Selection Flowchart**

The flowchart above shows that if the first word read from the address 0000h of the external data memory matches the BSW properly, the bootloader will execute an external parallel boot. Otherwise, the bootloader will branch to the end of the bootloader and go into an endless loop.

Then, the DMA will be used to load code into the ’5421’s shared memory. The DMA will load one section at a time. Individual sections cannot be larger than 32K words and cannot span entire 32K-word blocks of physical memory. At the conclusion of each block, the DMA will be reconfigured to load code into the next block. This process is repeated until there are no sections left to transfer.

In 16-bit boot mode, the DMA directly loads code residing within external DMA data memory space into on-chip CPU memory space (within the internal DMA memory map). In 8-bit mode, the DMA temporarily loads code residing within external DMA data memory space into on-chip CPU data memory space (SARAM at 00_8000h to 00_FFFFh). The code is then packed and transferred to on-chip CPU memory space (within the internal DMA memory map).

## 3.3 Boot Table for Parallel Boot Mode

The boot table used for programming the parallel EEPROM is generated using the 8/16-bit parallel option of the Hex Conversion Utility (HEX500.EXE). An example command file for the Hex Conversion Utility for 8-bit parallel is shown below.

```
myfile.out /* Input COFF file name.
–e 0x0000 /* Entry point symbol.
–a /* ASCII hex output format.
–boot /* Bootload all sections in the input file.
–bootorg PARALLEL /* Create a parallel port boot table.
–memwidth 8 /* EPROM width is 8bits.
–o myfile.hex /* Output file name.
```

**Figure 3. Hex Conversion Example Command File for Parallel EPROM Boot Mode**

When the Hex Conversion Utility is invoked with the example command file above, it creates an ASCII hex file called myfile.hex, which can be used for programming a parallel EPROM. All of the sections from the input file are placed into the boot table and the entry point is set to the physical address 0x0000.

# 4 Serial EEPROM Boot Mode

The serial EEPROM boot mode is intended for a bootload of the ’5421 from an SPI based serial EEPROM. This mode configures McBSP2 in the clock-stop mode, with internal clocks and frames. After configuration, the McBSP sequentially accesses the serial EEPROM. The EEPROM must have a four-wire SPI slave type interface. The interface between the McBSP and EEPROM is shown in Figure 4.

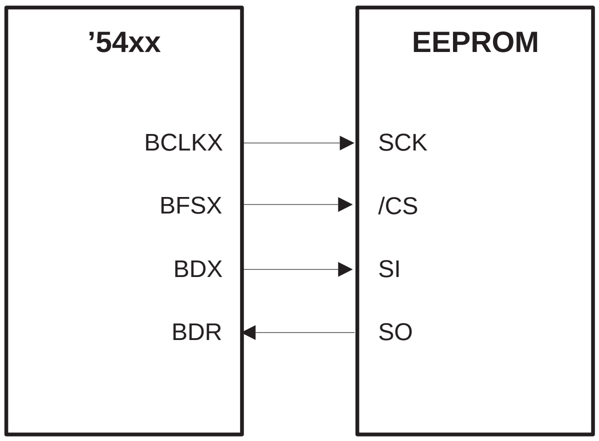

**Figure 4. McBSP to EEPROM Interface for Serial EEPROM Boot Mode**

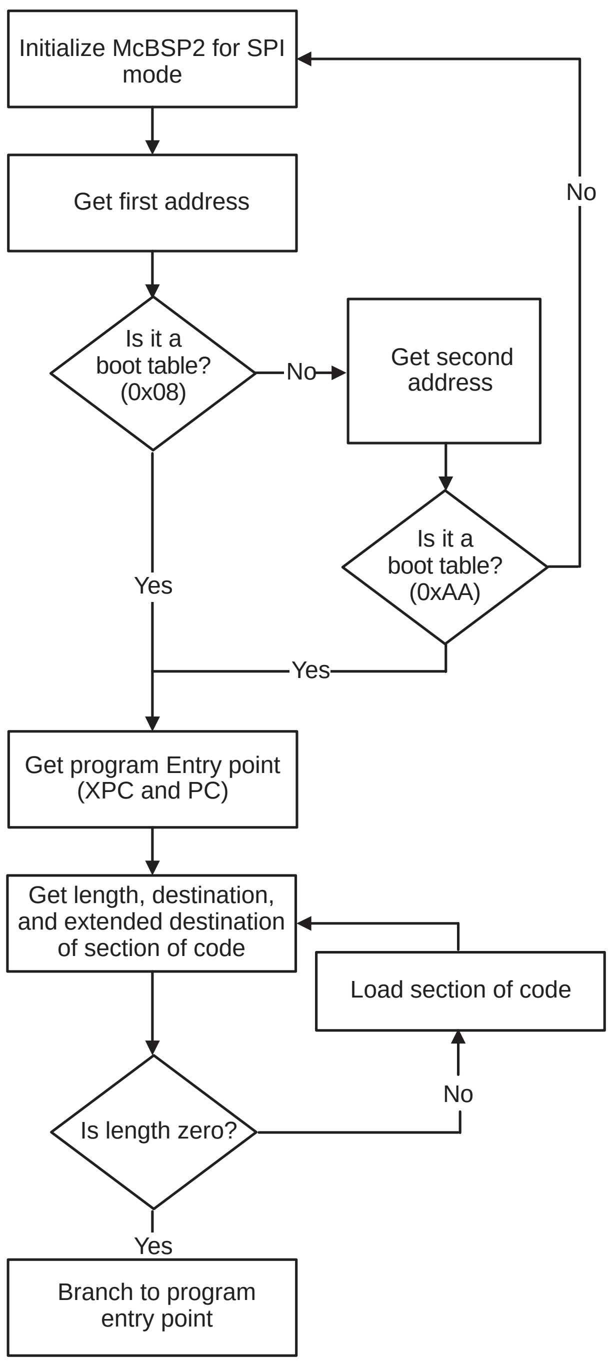
Initialize McBSP2 for SPI mode
Get first address
Is it a boot table? (0x08)
No
Get second address
Is it a boot table? (0xAA)
No
Yes
Yes
Get program Entry point (XPC and PC)
Get length, destination, and extended destination of section of code
Load section of code
No
Is length zero?
Yes
Branch to program entry point

**Figure 5. SPI Bootload Flowchart**

For each access, the McBSP transmits a 32-bit packet consisting of the 8-bit read instruction (03h), followed by the 16-bit read address, and a “place-holder” byte. The EEPROM ignores the last 8 bits in the packet, and uses this slot to shift out the addressed byte on the SO output pin. An example read access is shown below.

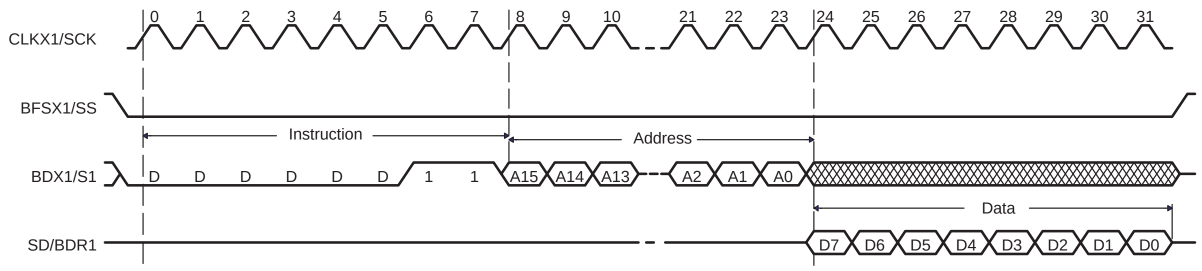

**Figure 6. Example Read Access for Serial EEPROM Boot Mode**

The McBSP clock rate is set to $f_{CLKOUT}$/250, or 400Khz for a 100Mhz device. This low bit rate ensures compatibility with most EEPROM devices. The McBSP is configured with CLKSTP = 2, CLKXP = 0, and CLKXM = 1, for use as an SPI master. The relevant interface timings for this mode are given in the McBSP section of the ’5421 datasheet. In addition, more specific McBSP details are provided in the *TMS320C54X DSP Enhanced Peripherals Reference Set Volume 5* (SPRU302).

## 4.1 Serial EEPROM Boot Mode Selection

The serial EEPROM mode is selected via the GPIO1 pin. Proper selection of the boot mode requires a high state on the GPIO1 pin for at least 50 CPU cycles after the ’5421 is reset.

Similar to the parallel boot modes, the DMA is used to boot load code into the ’5421’s shared memory. The DMA will boot load one section at a time. Individual sections cannot be larger than 32K words and cannot span entire 32K-word sections of physical memory located within the DMA memory map. At the conclusion of each block, the DMA will be reconfigured to load code into the next block. This process will be repeated for all sections.

After the serial EEPROM boot process completes, the XF signal, which is high immediately after reset, is toggled low. If the EEPROM has an active low HOLD input, this event can be used to automatically disable the EEPROM after the bootload is completed.

## 4.2 Boot Table for Serial EEPROM Boot Mode

The boot table used for programming the serial EEPROM is generated using the 8-bit serial option of the Hex Conversion Utility (HEX500.EXE). An example command file for the Hex Conversion Utility is shown below.

```
myfile.out /* Input COFF file name.
–e 0x0000 /* Entry point address.
–a /* ASCII hex output format.
–boot /* Bootload all sections in the input file.
–bootorg SERIAL /* Create a serial port boot table.
–memwidth 8 /* EEPROM width is 8 bits.
–o myfile.hex /* Output file name.
```

**Figure 7. Hex Conversion Command File Example for Serial EEPROM Boot Mode**

When the Hex Conversion Utility is invoked with the example command file above, it creates an ASCII hex file called myfile.hex, which can be used for programming a serial EEPROM. All of the sections from the input file are placed into the boot table, and the entry point is set to the physical address 0x0000.

## 5 Boot Table

The code to be loaded by the bootloader must be arranged in a particular format. This arrangement is known as the boot table, and in addition to the sections being loaded, it contains information about the destinations and lengths of the sections. The TMS320C54x COFF to Hex Conversion Utility (Version 1.2 or higher) is used to create the boot table in the correct format. The –v548 assembler command line option should be used when assembling the application code. Note that version 1.2 or higher of the ’C54x code generation tools must be used to generate the proper boot table for the ’5421. Earlier versions of these tools do not support the enhanced bootloader options of the ’548/549/5410/5402/5409/5421. Tool versions prior to 1.2 will produce a boot table format intended for earlier ’C54x devices without generating warnings or errors. Refer to the *TMS320C54x Assembly Language Tools Users Guide* (SPRU102) for more information on the COFF to hex conversion utility. The table below shows the source program data stream for parallel boot in 16-bit word mode.

| |
|---|
| O8AAh or 10AAh |
| Initialize value of SWWSR$_{16}$ |
| Initialize value of BSCR$_{16}$ |
| Entry point(XPC)$_{7}$ |
| Entry point$_{16}$ |
| Size of 1st section$_{16}$ |
| Destination of 1st section(XPC)$_{7}$ |
| Destination of 1st section$_{16}$ |
| Code word(1)$_{16}$ |
| . |
| . |
| Code word(N)$_{16}$ |
| Size of Mth section$_{16}$ |
| Destination of Mth section(XPC)$_{7}$ |
| Destination of Mth section$_{16}$ |
| Code word(1)$_{16}$ |
| . |
| Code word(N)$_{16}$ |
| 0000h |

**Figure 8. Program Data Stream for Parallel Boot in 16-Bit Word Mode**

# Appendix A Memory Maps

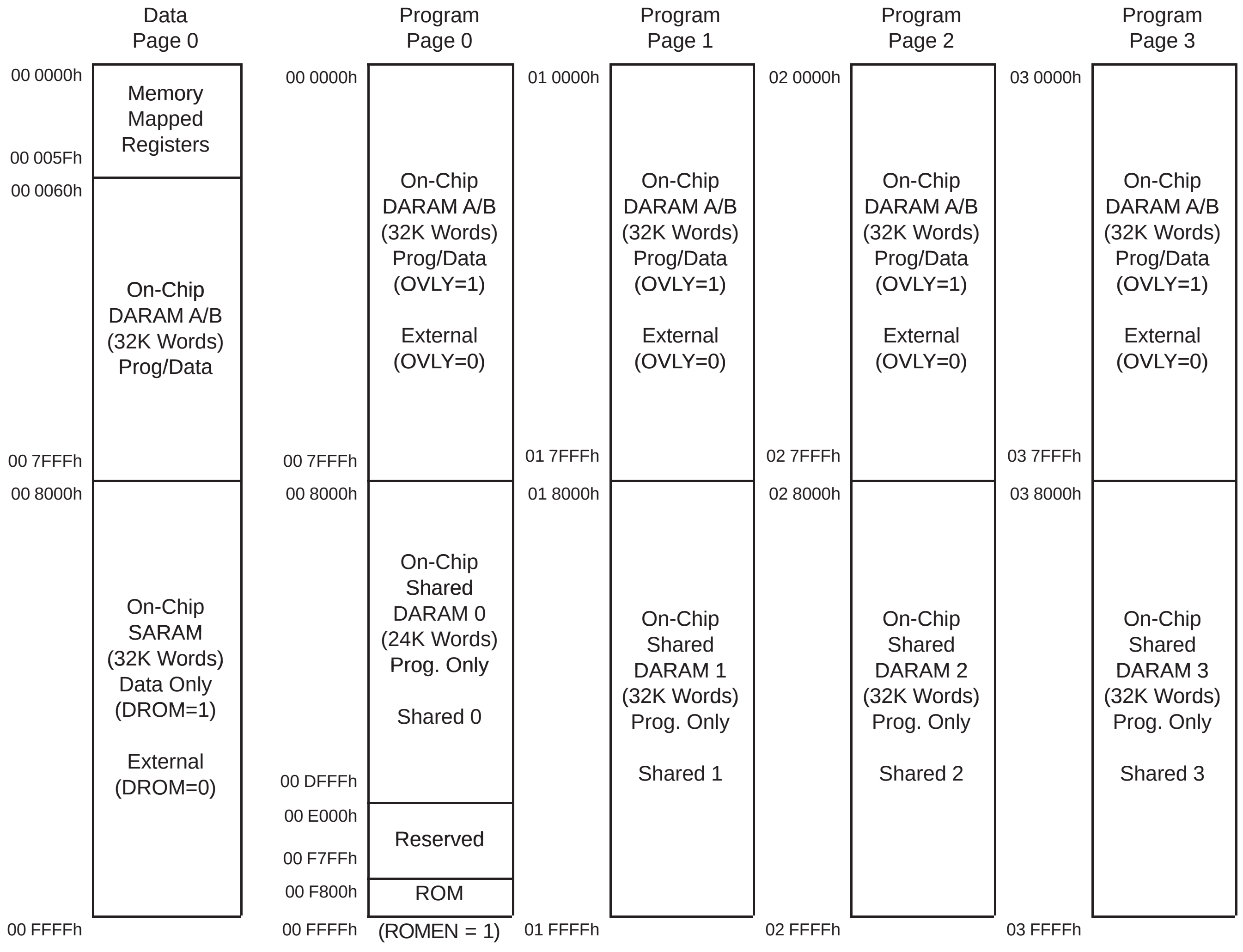
Data Page 0
Program Page 0
Program Page 1
Program Page 2
Program Page 3
00 0000h
Memory Mapped Registers
00 005Fh
00 0060h
On-Chip DARAM A/B (32K Words) Prog/Data
00 7FFFh
00 8000h
On-Chip SARAM (32K Words) Data Only (DROM=1)
External (DROM=0)
00 FFFFh
00 0000h
On-Chip DARAM A/B (32K Words) Prog/Data (OVLY=1)
External (OVLY=0)
00 7FFFh
00 8000h
On-Chip Shared DARAM 0 (24K Words) Prog. Only
Shared 0
00 DFFFh
00 E000h
Reserved
00 F7FFh
00 F800h
ROM
00 FFFFh
(ROMEN = 1)
01 0000h
On-Chip DARAM A/B (32K Words) Prog/Data (OVLY=1)
External (OVLY=0)
01 7FFFh
01 8000h
On-Chip Shared DARAM 1 (32K Words) Prog. Only
Shared 1
01 FFFFh
02 0000h
On-Chip DARAM A/B (32K Words) Prog/Data (OVLY=1)
External (OVLY=0)
02 7FFFh
02 8000h
On-Chip Shared DARAM 2 (32K Words) Prog. Only
Shared 2
02 FFFFh
03 0000h
On-Chip DARAM A/B (32K Words) Prog/Data (OVLY=1)
External (OVLY=0)
03 7FFFh
03 8000h
On-Chip Shared DARAM 3 (32K Words) Prog. Only
Shared 3
03 FFFFh

**Figure A–1. CPU Memory Map**

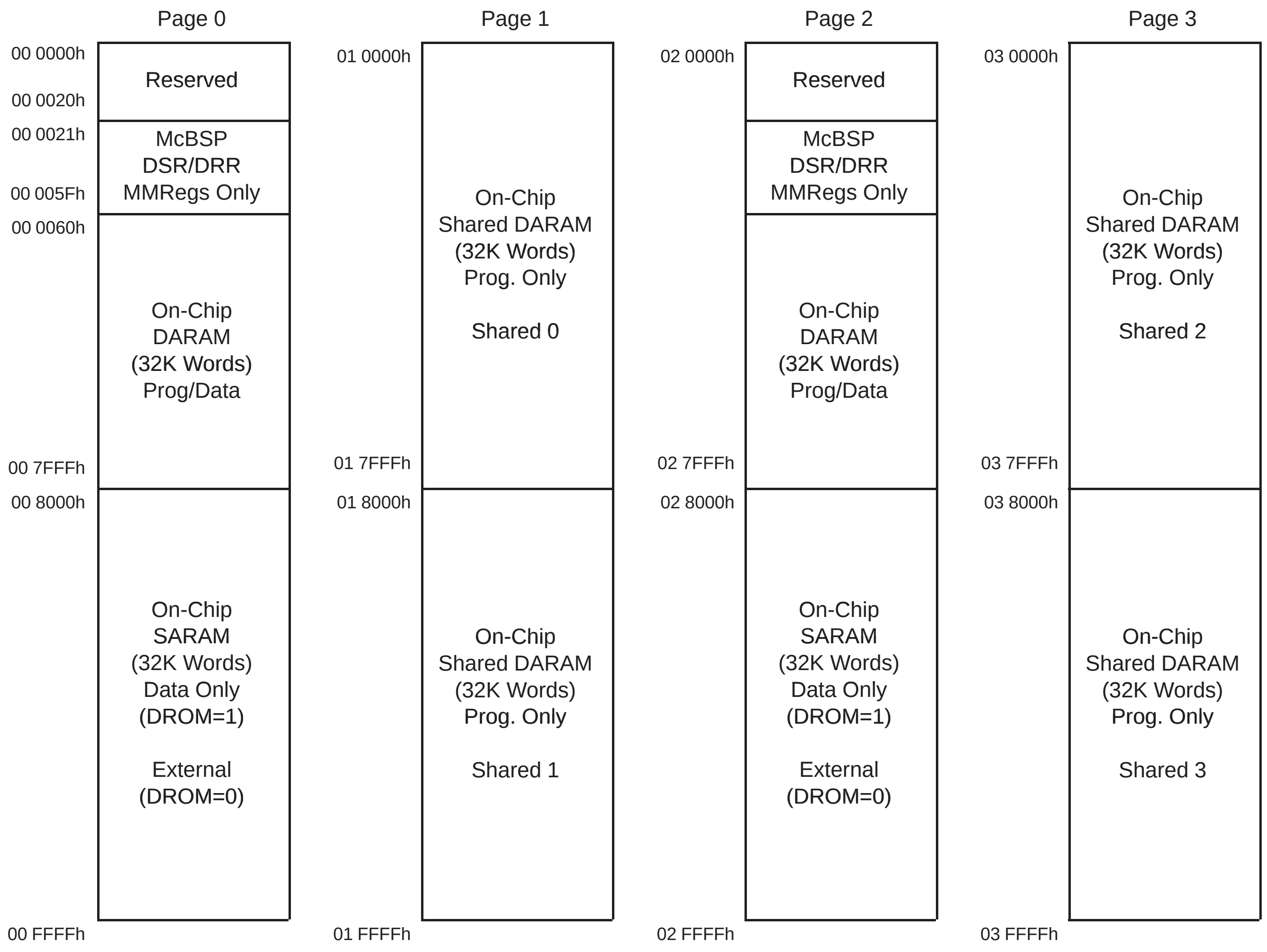
Page 0
00 0000h
Reserved
00 0020h
00 0021h
McBSP DSR/DRR MMRegs Only
00 005Fh
00 0060h
On-Chip DARAM (32K Words) Prog/Data
00 7FFFh
00 8000h
On-Chip SARAM (32K Words) Data Only (DROM=1)
External (DROM=0)
00 FFFFh
Page 1
01 0000h
On-Chip Shared DARAM (32K Words) Prog. Only
Shared 0
01 7FFFh
01 8000h
On-Chip Shared DARAM (32K Words) Prog. Only
Shared 1
01 FFFFh
Page 2
02 0000h
Reserved
McBSP DSR/DRR MMRegs Only
On-Chip DARAM (32K Words) Prog/Data
02 7FFFh
02 8000h
On-Chip SARAM (32K Words) Data Only (DROM=1)
External (DROM=0)
02 FFFFh
Page 3
03 0000h
On-Chip Shared DARAM (32K Words) Prog. Only
Shared 2
03 7FFFh
03 8000h
On-Chip Shared DARAM (32K Words) Prog. Only
Shared 3
03 FFFFh

**Figure A–2. DMA On-Chip Program Memory Map**

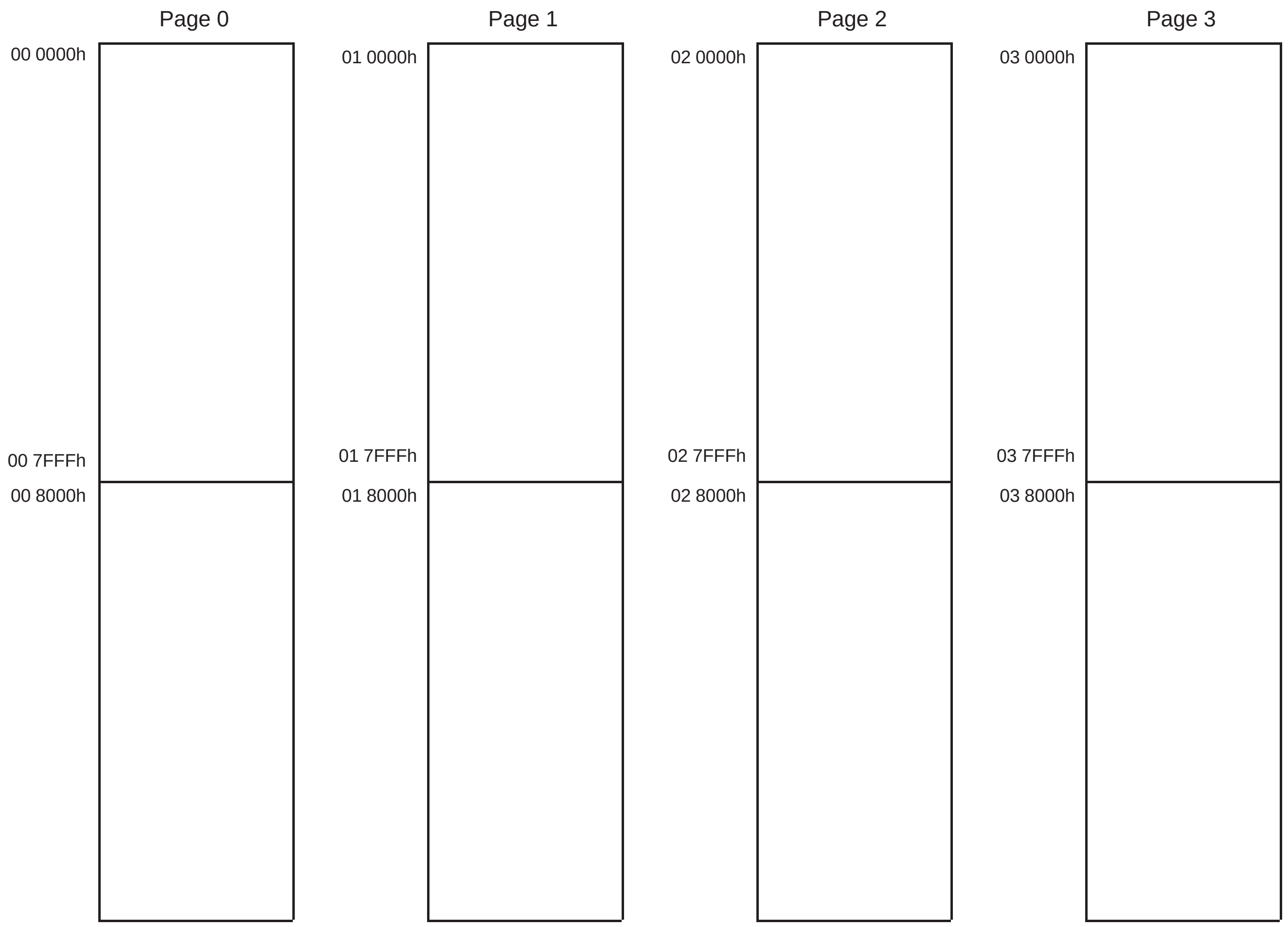

**Figure A–3. DMA External Data Memory Map (SLAXS=1, DLAXS=1)**