Texas Instruments

Application Report
SPRA607 - November 1999

Understanding the TMS320C54x Memory Map and Examining an Optimum C5000 Memory Interface

Mathew George, Jr. (Joe)

Texas Instruments Incorporated
Natural Microsystems

ABSTRACT

The Texas Instruments (TI™) TMS320C54x DSP family contains various derivatives with the same CPU, but different memory mixes and peripherals. Since the code is compatible between derivatives, you can port to different parts depending on needed system cost and functionality (The ’C548, ’C549, ’C5410, and ’C5402 are almost pin-for-pin compatible making migrating between them easy if some special considerations are taken). Thus, understanding how the memory maps differ between the different parts is important. Deriving a flexible and optimum common memory architecture in hardware that translates to a mapping in software is useful because it in turn easily allows porting between the different parts. This application report attempts to first break down and explain the ’C54x memory maps. It explores an optimum memory interface schematic and mapping that consequently should allow easy porting between the different ’C54x parts. Thus, making this effort ahead of time should ease reconfiguring software for different system costs and functionality

Contents

List of Figures

List of Tables

1 Introduction

The Texas Instruments (TI™) TMS320C54x DSP family contains various derivatives with the same CPU, but different memory mixes and peripherals. Since the code is compatible between derivatives, it is nice to be able to port to different parts depending on needed system cost and functionality (The ’C548, ’C549, ’C5410, and ’C5402 are almost pin-for-pin compatible making migrating between them easy if some special considerations are taken). Thus understanding how the memory maps differ between the different parts is important. And deriving a flexible and optimum common memory architecture in hardware that translates to a mapping in software is useful because it in turn easily allows porting between the different parts. This application report attempts to first break down and explain the ’C54x memory maps. Then it explores an optimum memory interface schematic and mapping that consequently should allow easy porting between the different C54x parts. Thus, making this effort ahead of time should ease reconfiguring software for different system costs and functionality

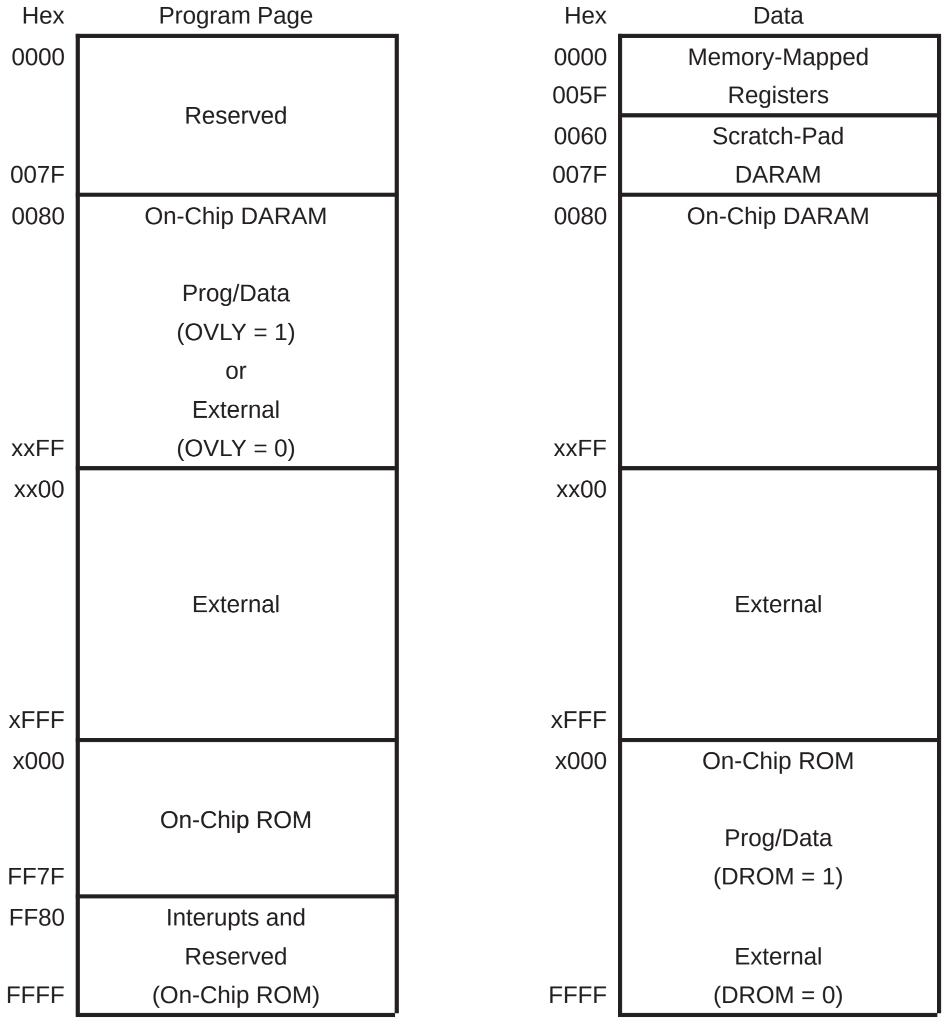
2 Memory Maps

The various ’C54x parts have a common modified Harvard architecture memory map style that varies with the amount of on-board memory and external address pins. Various bits located in the PMST register (see *TMS320C54x CPU and Peripheral Guide* – SPRU131) allow the programmer to vary the use of the memory map and the logical blocks enabled by them. Each memory map can be characterized by the particular ’C54x part, the logical memory block, and the particular bit that configures it. In the following sections we build up the ’C54x parts from lower amount of memory parts to higher amount of memory parts to observe their commonality. Remember that the CPUs are all the same, hence allowing portability.

2.1 ’C541, ’C542, ’C543, ’C545, ’C546, and ’C5402 Memory Maps

Figure 1 shows the most basic ’C54x memory configuration. All the parts build upon this map. Note that the separate program and data spaces are shown as consistent with the modified Harvard architecture.

Please note on all memory maps in this document MP/MC_ is equal to zero, known as Microcomputer mode. For all the parts except the ’C5420 (see section 2.5), all this mode does is turn the ROM on in the program space.



NOTE: MP/$\overline{MC}$ – 0 (Microcomputer mode)

Figure 1. ’C541, ’C542, ’C543, ’C545, ’C546, and ’C5402 Memory Map

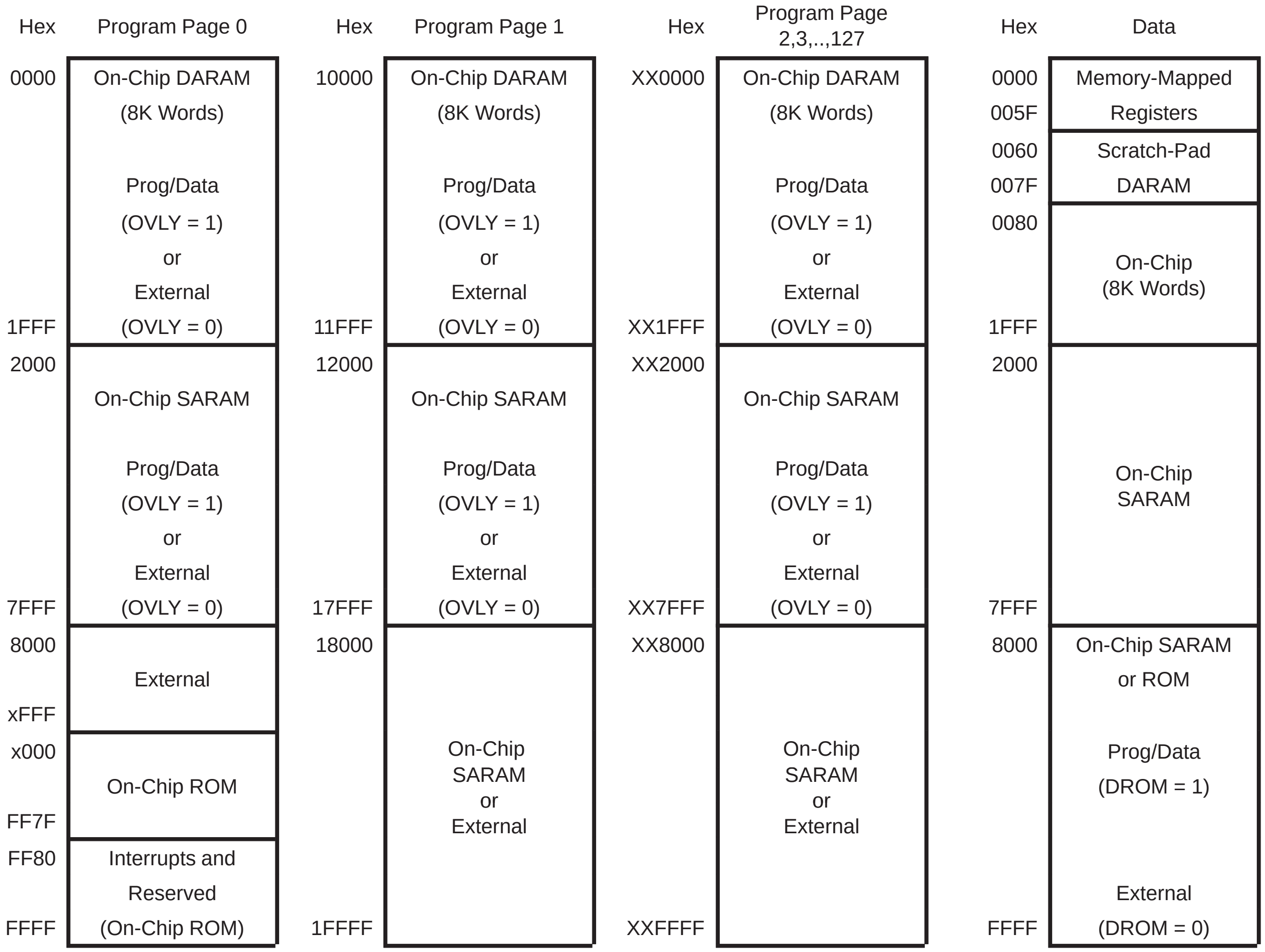
This map can be summarized as:

- In Program Space, the internal ROM block extends from 0FFFFh backwards if MP/$\overline{MC}$ bit=0.
- In Data Space, the internal ROM block extends from 0FFFFh backwards if DROM bit=1.
- In Data Space, memory mapped registers and 32 words of scratch-pad RAM extend from 0h to 7Fh by default (no bit).
- In Program Space, a Reserved area extends from 0h to 7Fh if OVLY=1.
- In Data Space, the internal DARAM block extends from 80h forwards by default (again no bit).
- In Program Space, the same internal DARAM block extends from 80h forwards if OVLY bit=1.

The “’C548+” parts build upon the above map as mentioned in section 2.2.

2.2 ’C548, ’C549, ’C5410, and ’C5420 Memory Maps

In addition to all the features mentioned in section 2.1, the ’C548, ’C549, ’C5410, and ’C5420 memory maps have added features. The first one is more internal memory, while the second feature is more external memory. For the latter, memory is paged in program space only. Address pins A16–A22 are added to the pin list and supported in software with far branches and calls. Figure 2 shows this extended map:



NOTE: MP/$\overline{\text{MC}}$ – 0 (Microcomputer mode)

Figure 2. ’C548, ’C549, ’C5410, and ’C5420 Memory Map

The additions to this map can be summarized as:

- In Data Space, the internal SARAM(1) block extends from the end of the DARAM block forwards by default (again no bit).
- In Program Space, the same internal SARAM(1) block extends from the end of the DARAM block forwards if OVLY bit=1.
- In Program Space, an extra 7 bits of address available for paging as controlled by the XPC register.

There are some characteristics that are exclusive to the ’C5410 and the ’C5420 parts as mentioned in the next section.

2.3 ’C5410 and ’C5420 Memory Maps

In addition to all the features mentioned in section 2.2, the ’C5410 and ’C5420 memory maps have some something else in common. This feature is that they contain additional SARAM internally, though the use of the internal ROM is slightly limited. This one feature is summarized below:

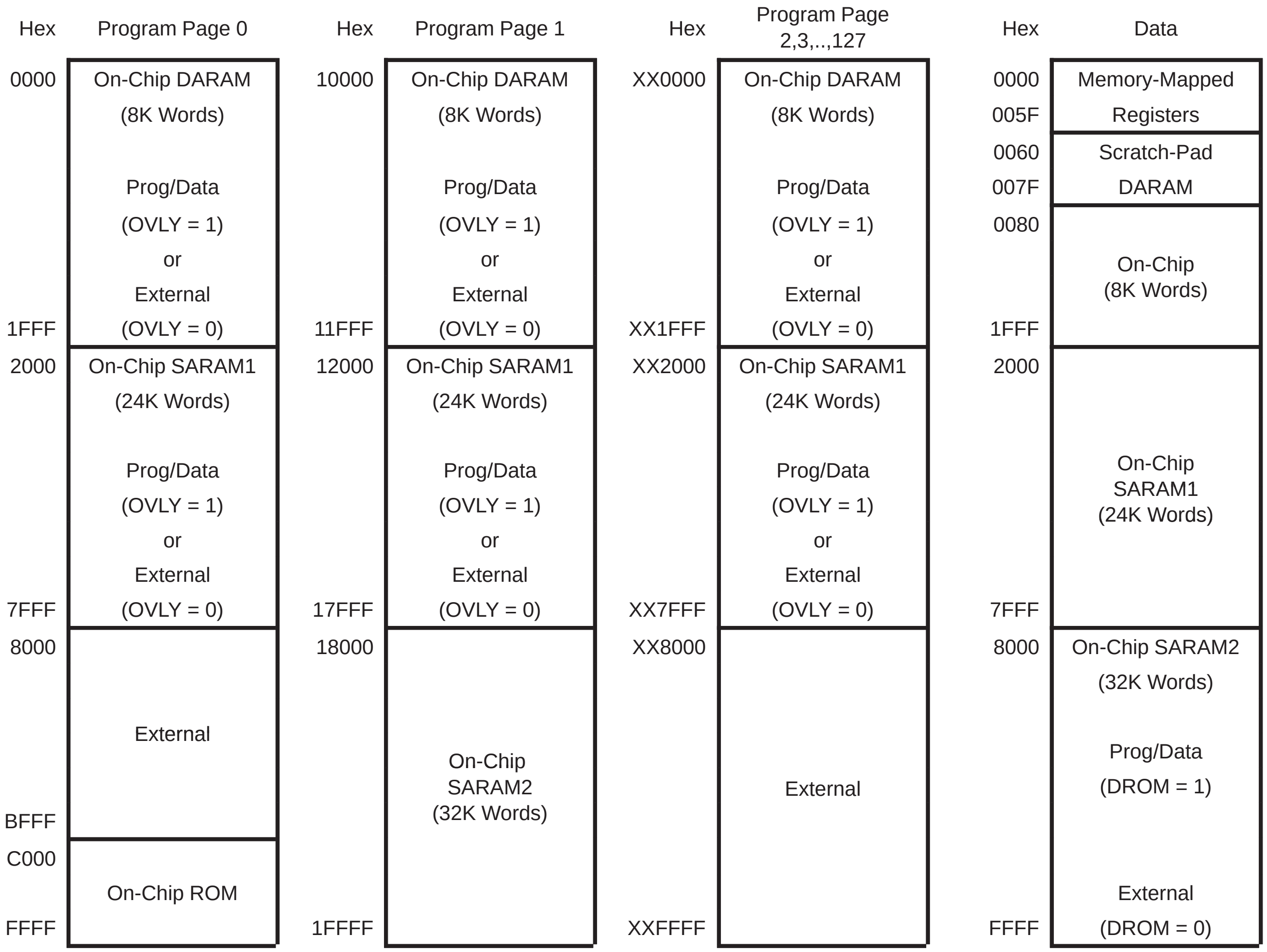
In Data Space any internal ROM block, if existing, is *not* mappable.

Rather in Data Space, the internal SARAM2 block extends from the 8000h–0FFFF if DROM bit=1.

Next we will look at the ’C5410 and ’C5420 memory maps individually, starting with the ’C5410.

2.4 ’C5410 Memory Map

The ’C5410 adds more internal program memory than ’C548/’C549 (64Kx16 total internal RAM). Figure 3 shows the ’C5410 map.



NOTE: MP/$\overline{\text{MC}}$ – 0 (Microcomputer mode)

Figure 3. ’C5410 Memory Map

In addition to all the features mentioned in the section 2.3, the ’C5410 memory map adds:

In Program Space, the internal SARAM2 block extends from 18000h to 1FFFFh (i.e. XPC=1) by default (no bit).

2.5 ’C5420 Memory Map

The ’C5420 puts 2 DSP cores with much more memory than the ’C5410 on a single chip (100Kx16 total internal RAM) . A special host port interface (HPI16) is multiplexed with the external memory interface (XIO) for both cores (to keep pin count down). The memory map for each core is shown on Figure 4.

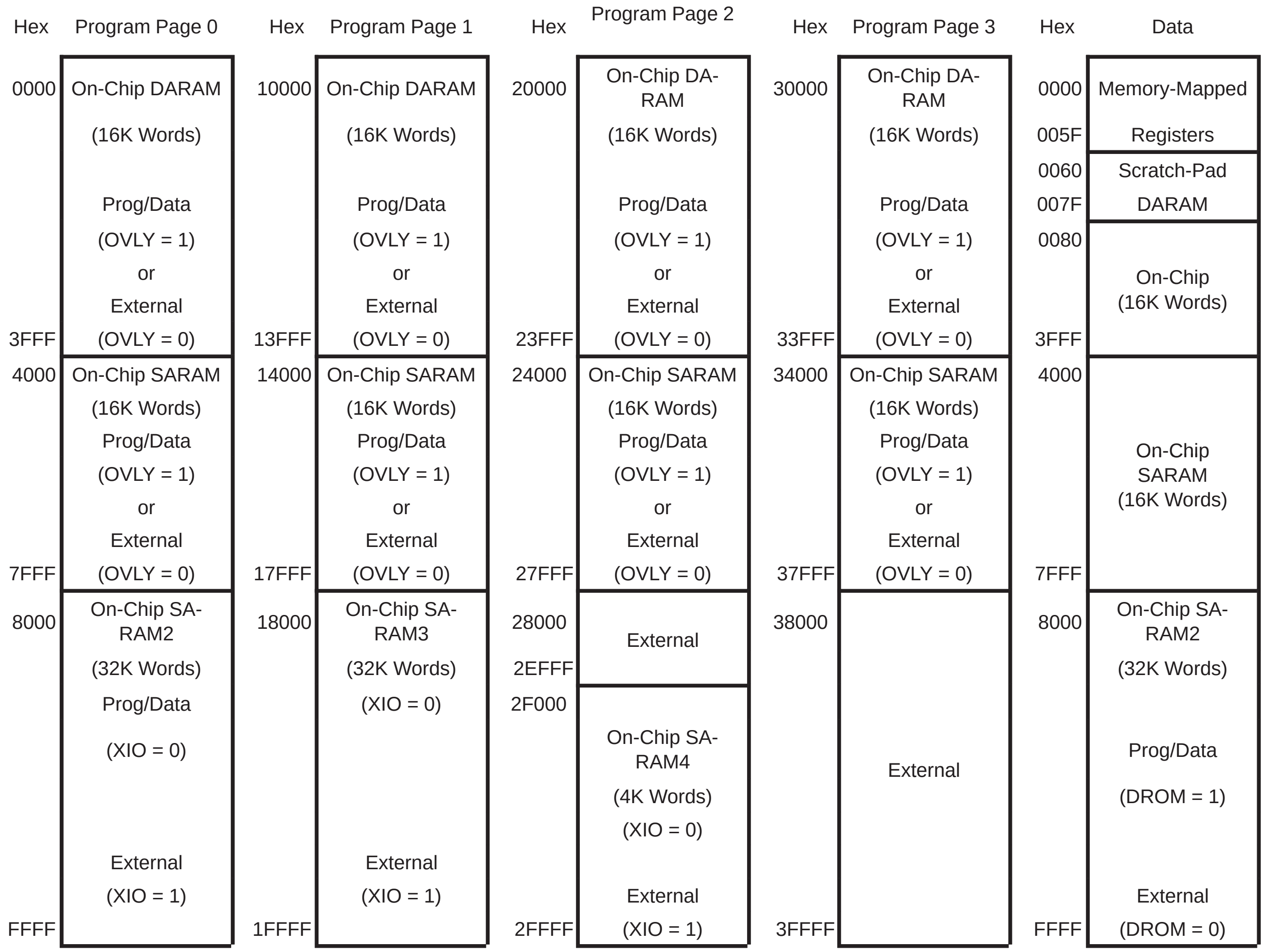


Figure 4. ’C5420 Memory Map

In addition to all the features mentioned in section 2.3, the ’C5420 memory map has:

- In Program Space, no ROM.
- In Program Space, the internal SARAM2 block extends from 8000h to 0FFFFh by default (again no bit).
- In Program Space, the internal SARAM3 block extends from 18000h to 1FFFFh (i.e. XPC=1) by default (no bit).
- In Program Space, the internal SARAM4 block extends from 2F000h to 2FFFFh (i.e. XPC=2) by default (no bit).

Note that XIO is a pin that is reflected in the PMST register in the MP/$\overline{\text{MC}}$ bit.

The previous sections showed the logical blocks in the ’C54x family. But some further examination of the physical blocks for performance purposes is worth discussing.

2.6 Performance Considerations

Note that the logical memory blocks ROM, DARAM, and SARAM are split into physical blocks that are sized anywhere from 2K x 16 to 8K x 16. Each of these physical blocks have separate bussing which can be advantageous for CPU and peripheral operations within a single cycle. Please refer to the ’C54x Datasheets (SPRS039, SPRS075, SPRS079, and/or SPRS080,) for the physical block granularity for each part. And the *TMS320C54x Mnemonic Instruction Set User’s Guide* (SPRU172) explains how to configure code and data in these blocks per instruction for the best instruction cycle benchmarks.

3 Adding External Memory to ’C548, ’C549, ’C5410, and ’C5420 Memory

External memory can be added to all of the ’C54x parts on the external bus. With the “pre-C548” parts, this is usually a simple task of attaching one or two banks of 64K x 16 external SRAM (see *TMS320C54x Evaluation Module Technical Reference* – SPRU135). The ’C548+ parts have an inherent extended program memory paging scheme that allows addressing more than 64K of program. An XPC register in the CPU along with Far instructions allows paging of 7 additional address lines in program space. Data space does not affect A16–A22 (It must be paged with external logic – see *TMS320C5X Memory Paging .Expanding its Address Reach Interface Application Brief* – SPRA242). Thus when you look at the memory maps in Figure 1–Figure 4, attaching external memory might not be so straightforward. Let us examine why.

3.1 OVLY Bit and Separate Program and Data

The OVLY bit located in the PMST register allows code to run out of external program memory when OVLY=0. The following external memory interface in Figure 5 is borrowed from the PCMCIA DSP MediaCard Application Note (SPRA052) as a separate program and data memory model (64K each) using a 128k x 16 asynchronous SRAM.

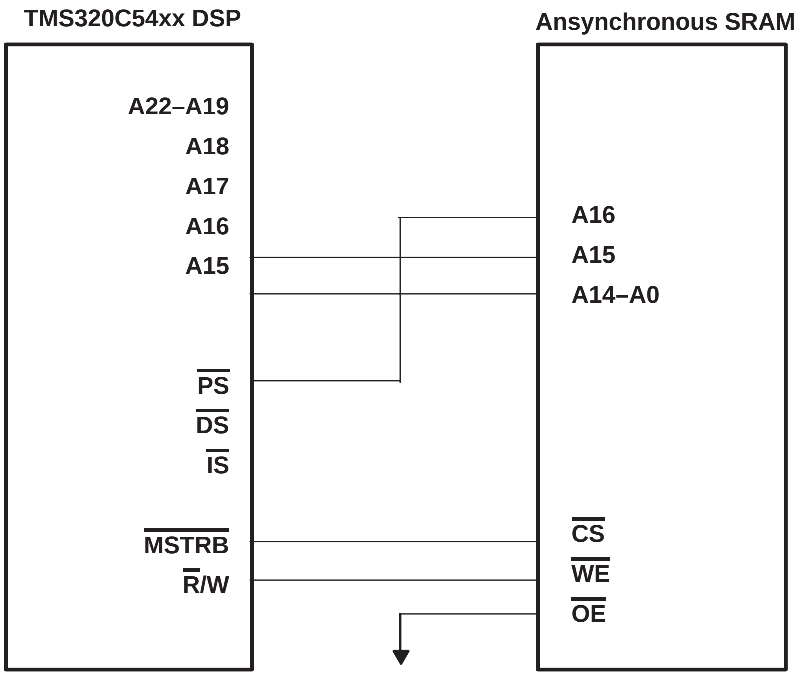


Figure 5. Separate Program and Data Memory Configuration

This configuration is convenient for 128K x 16 RAM but if used with anything larger (i.e connect PS –> A17 for a 256K x 16 RAM), data *cannot* be paged and thus those 64 K pages of data will not be addressable and thus lost (Reworking Table 1 in the section 4 can prove this). Another disadvantage of this method is the classic problem of SRAM (and other external memory) speeds not keeping up with processor speeds. Thus external program memory with OVLY=0 will probably need to run at 1 wait state on the faster (>66 MIP parts) and not at 0 wait state as seen on the slower ’C54x parts (<66 MIP. Refer to the TMS320C54x Data Sheet – SPRS039 – for the latest specifications). But internal program memory is 0 wait state for any speed ’C54x part, so running out of internal program memory is an option mentioned below.

3.2 OVLY Bit and Combined Program and Data

An option exists to run program internally on the ’C54x. When OVLY=1 the mapping allows internal data memory (DARAM and/or SARAM) to be overlaid into program memory. This allows code to run 0 wait state at processor speed and take advantage of internal bussing for single-cycle operation of many instructions. An inherent disadvantage of overlapping this area of the memory is the reduction of total memory mapping area (see *PCMCIA DSP MediaCard Application Note* – SPRA052). Another is needing to remember that 1000h in program is the same as 1000h in data and such must be kept separate in the linker command file.

So we can follow this same model of combining program and data memory in internal memory to the external memory. We borrow from Fast TMS320C5x External Memory Interface Application Brief (SPRA241) and using a similar 128K x 16 memory as seen in Figure 6, Figure 7 shows a combined program and data configuration.

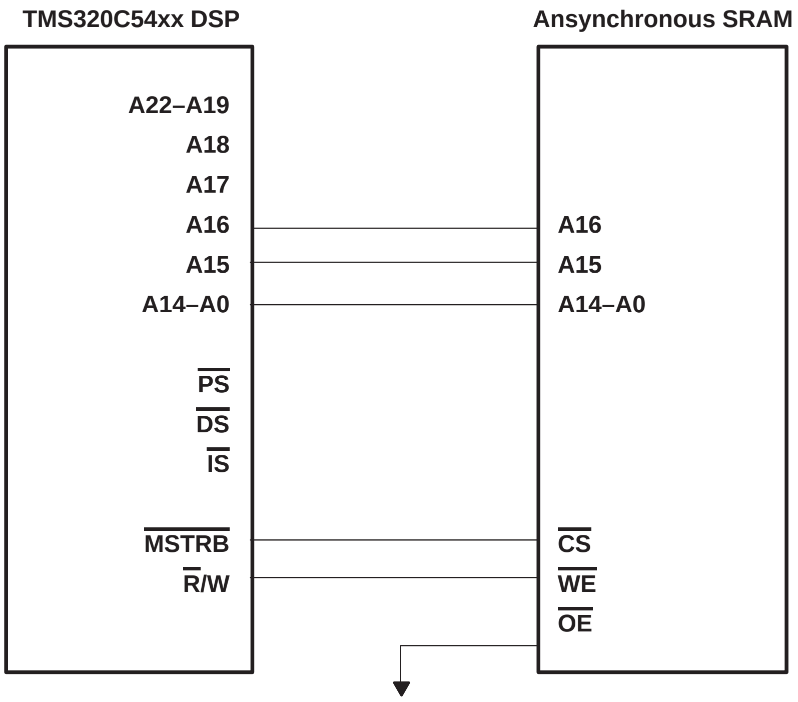


Figure 6. Combined Program and Data Memory Configuration

Since PS/DS is not being decoded here and if any other devices are on the parallel bus, another signal like IS or BR should be monitored to turn off the SRAM (see *PCMCIA DSP MediaCard Application Note* – SPRA052). Note again that data space CANNOT be paged and thus those 64 K pages of data will not be addressable in data. But at least those pages are available in program space .

3.3 Memory “Floors” and “Ceilings”

As mentioned in the previous section, one subtle point to note is that since the ’C54x CPU only allows paging of program memory, in the above figure location 18000h is not available in data memory (in other words the DS pin will never go low above 0FFFFh). Note that if OVLY = 1 in Fig 6, then the external locations that overlap the internal memory blocks (e.g. 1000h on any ’C54x DSP) can only be accessed if OVLY = 0 And though OVLY can theoretically be changed on the fly, most systems choose to keep OVLY = 1 instead of paging in program. An effect of this configuration is the flexibility that a combined program and data memory can give a system.

But to keep all internal memories accessible, they must be bypassed in the external map. In other words, place the external memories at an address higher than the internal memories for the particular device. This is done by shifting the DSP address pins “left” when interfacing to the SRAM address (an example is given in Figure 7). Thus a floor is required in the memory map internal memory it has.

Similarly on the upper end of the address range a ceiling needs to be comprehended so that the ’C54x can address all needed external memory up to the available 22 bits of address minus the “floor” (an issue with the ’C5420 will be discussed in Section 3.5) .

Now that we have introduced the ideas, let us examine an example in the next section.

3.4 Optimum Combined Program and Data Memory Map

In Figure 7 an optimum combined program and data memory configuration is proposed based on Figure 6 with a memory block resolution of 32 K. This resolution is enabled by omitting and address pin, A15 of the DSP. This omission causes odd/even pages of mirror images in the memory map (see Table 1), but makes sense from a memory map standpoint (looking at Figure 1 – Figure 4) since it conveniently bifurcates the native 16-bit address block of the ’C54x.

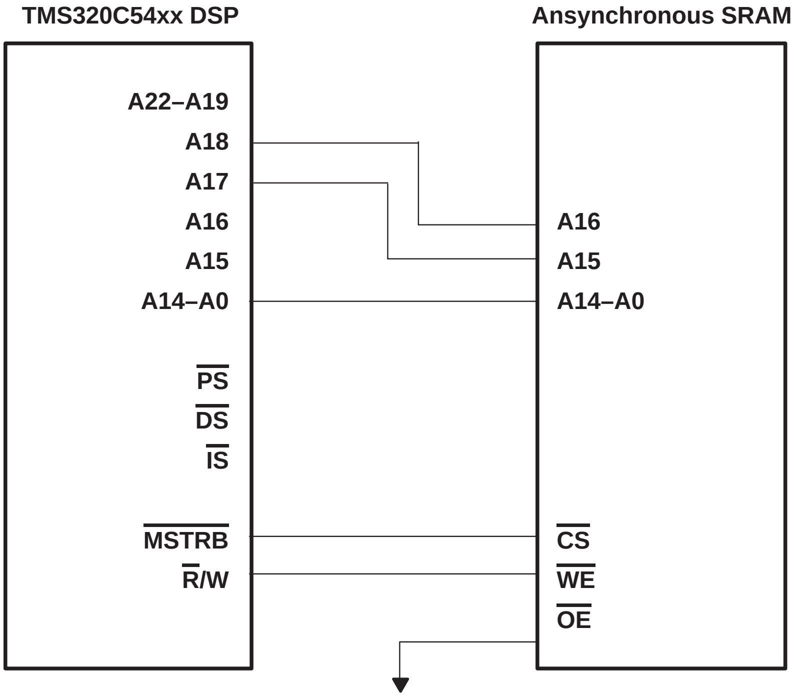


Figure 7. Optimum Combined Program and Data Memory Map

The ’C54x address pin A16 has been left-shifted to provide a floor as mentioned in section 3.3. Thus for all the DSP’s (’C5420 Considerations are mentioned in section 3.5) all the internal memory can be reached without flipping any PMST bits (though some paging in program memory XPC register is needed).

Regarding the ceiling, this example uses a 128K x16 SRAM.

Such a configuration remaps the DSP memory to SRAM mapping. The following chart derives that mapping and proves that no memory is wasted (Doing Figure 4 in a similar way would waste a few 32K blocks of SRAM by making them unaddressable). Thus, this chart could be a useful tool any modifications to Figure 7 need to be made. By working through such a chart, the designer can ensure that no memory is wasted. Also note due to the shift, a mirror image is caused for odd and even XPCs which should not be a problem if the programmer is aware of this fact.

Table 1. Example Optimum Map Derivation

’C548+ Range (OVLY=1)		SARAM Range	
$\overline{\text{PS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{DS}}$ = 0	XPC=n/a		
A22–A16 = 0000h			
A15–A0 = 8000h	ffffh	0000h	7ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=0		
A19–A16 = 0000b			
A15–A0 = 8000h	ffffh	0000h	7ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=1		
A19–A16 = 0001b			
A15–A0 = 8000h	ffffh	0000h	7ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=2		
A19–A16 = 0010b			
A15–A0 = 8000h	ffffh	8000h	ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=3		
A19–A16 = 0011b			
A15–A0 = 8000h	ffffh	8000h	ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=4		
A19–A16 = 0100b			
A15–A0 = 8000h	ffffh	10000h	17fffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=5		
A19–A16 = 0101b			
A15–A0 = 8000h	ffffh	10000h	17fffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=6		
A19–A16 = 0110b			
A15–A0 = 8000h	ffffh	18000h	1ffffh
$\overline{\text{DS}}$/$\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=7		

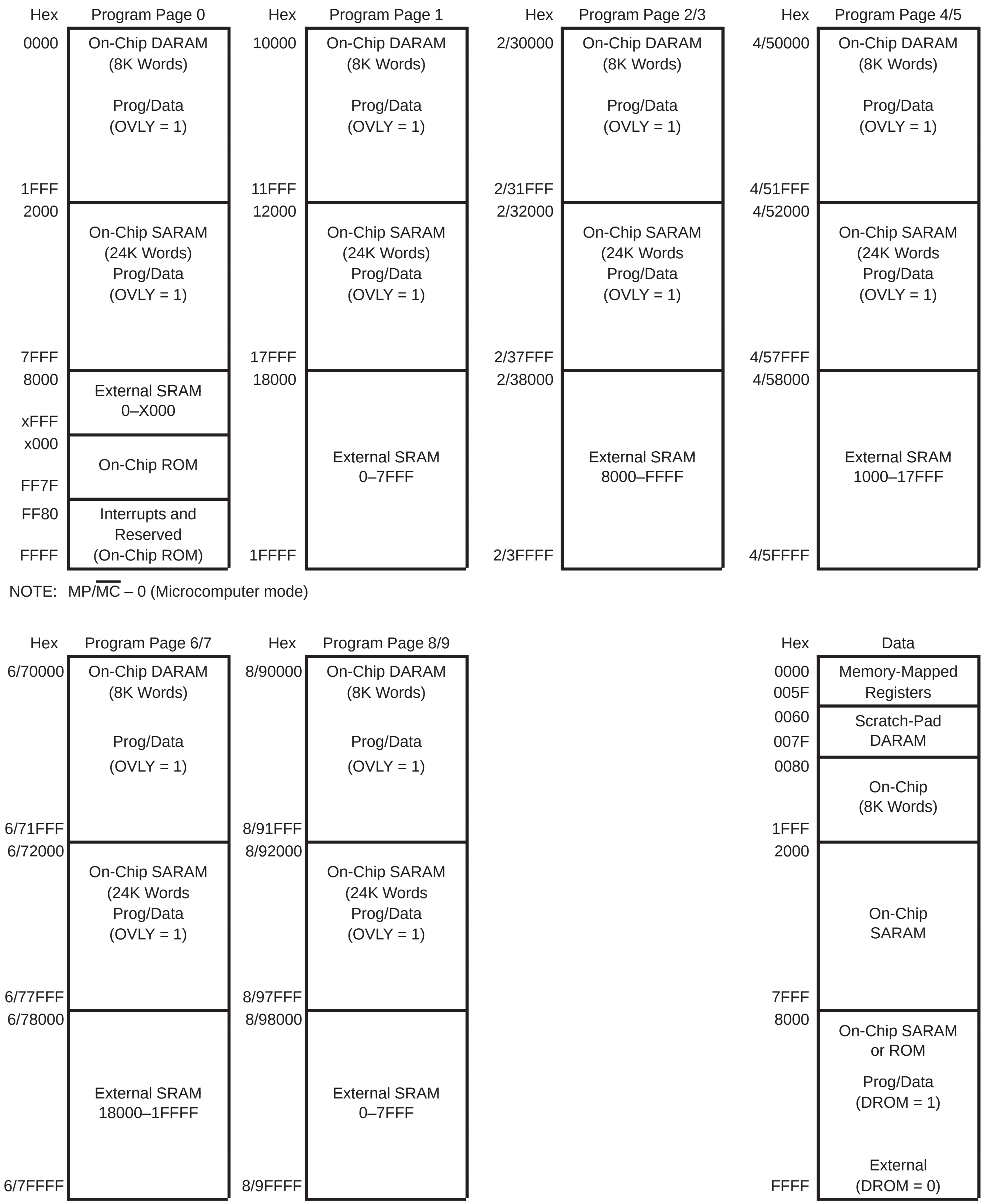
Table 1. Example Optimum Map Derivation (Continued)

’C548+ Range (OVLY=1)		SARAM Range	
A19–A16 = 0111b			
A15–A0 = 8000h	ffffh	18000h	1ffffh
$\overline{\text{DS}}/\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=8		
A19–A16 = 1000b			
A15–A0 = 8000h	ffffh	(0000h)	(7fffh)
$\overline{\text{DS}}/\overline{\text{IS}}$ = 1; $\overline{\text{PS}}$ = 0	XPC=9		
A19–A16 = 1001b			
A15–A0 = 8000h	ffffh	(0000h)	(7fffh)

In this chart the left two columns represent the address range mapping of ’C54x parts while the right two columns represent the corresponding address range of the SRAM based on the configuration in Figure 7. On the ’C54x side, one might draw a box around the upper 3 bits of A19–A16 (in binary) and concatenate it with a box around the lower 15 bits of A15–A0 (in hex) to get the SRAM mapping .

Note that the first two sets of rows show an overlapping of external program and data space. On the ’C54x side though A15–A0 is shown as starting at 8000h, A15 is ignored. This causes the XPC=odd/even mirror images described earlier in this section. Also A16–A19 could be extended to A22 if desired, but is not shown for lack of room and that reading this in binary seemed more clear.

With this configuration, the actual map coincides to something that looks similar to Figure 2. It is shown in Figure 8. Note that it assumes that OVLY=1. Thus xx0000–xx7FFF are always the same internal blocks. And so the far branch/call instructions will automatically go internal. The mirrored page addresses from page 2/3 on are denoted as “2/30000” on the memory map. For this 128Kx16 memory, page 0 mirrors back at page 8/9. A higher density memory will go to higher pages than 8/9.



NOTE: MP/$\overline{\text{MC}}$ – 0 (Microcomputer mode)

Figure 8. Optimum Combined Program and Data Map

Having shown this optimum interface, it should be mentioned that the ’C5420 does have some exceptions that are covered in the next section.

3.5 ’C5420 Considerations

Even though the ’C5420 CPU’s internal program address reaches to A22, it only pins out up to address A17. Thus the ceiling is less than on other parts. Some decode logic could be added to the ’C5420 upper address lines (A16 and A17) in Figure 7 to put memory at some of the mirror XPC’s. Of course, that logic changes some of the timing analysis that should be recalculated Also note that the ’C5420 SARAM program locations are dependent on XIO pin and MP/MC_ bit, not OVLY.

4 Conclusion

Hopefully the first part of this application note made the ’C54x memory maps clearer. With the optimum architecture presented in the later part of the paper, no memory is lost, all memory is available without having to flip bits, and the memory map architecture is common to *all* C54x family members. This architecture and design methodology should hopefully make your design with one or more ’C54x parts easier.

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.