Order this document
by EB386/D

MOTOROLA
digital dna™

Semiconductor Products Sector
Engineering Bulletin

EB386

STAR12 D-Family Compatibility Considerations

**By Joachim Krücken
BEOS MCU Design
Munich/Germany**

Introduction

During the development of the STAR12 Family a lot of attention was paid to make the devices compatible to each other as much as possible.

The purpose of this document is to describe differences amongst the family members. It is not intended to replace the specification of the devices. Also hints are given to gain full binary compatibility of Software developed on an umbrella (larger device) to the smaller production device. The following areas are covered:

1. Memory scheme and paging
2. Peripherals
    a. Which Devices contain which peripherals
    b. Address space of the register map
    c. interrupt vectors
3. Pin locations and functionality

Memory layout

Table 1 gives a first overview of the memory sizes of the various devices.

Table 1 Memory Sizes by Device

Device	Flash	RAM	EEPROM
Dx256	256K	12K	4K
Dx128	128K	8K	2K
Dx64	64K	4K	1K
Dx32	32K	2K	1K

A total of 5 registers determine the size (MEMSIZ0, MEMSIZ1) and location of the Register space, RAM, EEPROM and Flash/ROM (INITRG, INITRM, INITEE). If memory blocks are mapped to the same addresses, the priority is Register, RAM, EEPROM, Flash/ROM from top to bottom. Figure 1 shows the first one of two registers.

Device	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	reg_sw0	0	eep_sw1	eep_sw0	0	ram_sw2	ram_sw1	ram_sw0
Dx256	0	0	1	0	0	1	0	1
Dx128	0	0	0	1	0	0	1	1
Dx64	0	0	0	1	0	0	0	1
Dx32	0	0	0	1	0	0	0	0

Figure 1 MEMSIZ0 (Base+$1C) Read-Only Register

Register Space

The register memory mapping is determined by the INITRG register shown in **Figure 2**.

Base+$11	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read:	0	REG14	REG13	REG12	REG11	0	0	0
Write:	▨					▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

Figure 2 INITRG Register

The register space size is determined by the MEMSIZ0 register ($1C) bit regsw0 and hard coded for each device.

Table 2 Register Space Memory Mapping out of Reset

reg_sw0	Size	Begin Address	End Address	Map Boundary
1	2K	$0000	$07FF	2K (Lower 32K)
0	1K	$0000	$03FF	2K (Lower 32K)

RAM

The RAM memory mapping is determined by the INITRM register shown in **Figure 3**.

Base+$10	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read:	RAM15	RAM14	RAM13	RAM12	RAM11	0	0	RAMHAL
Write:						▨	▨	
Reset:	0	0	0	0	1	0	0	1

Figure 3 INITRM Register

The RAM size is determined by the MEMSIZ0 register ($1C) bits ram_sw2:ram_sw0 and hard coded for each device.

Table 3 RAM Memory Mapping out of Reset

ram_sw2	ram_sw1	ram_sw0	Size	Begin Address	End Address	Map Boundary
1	1	1	16K	$0000	$3FFF	16K
1	1	0	14K	$0800	$3FFF	16K H/L
1	0	1	12K	$1000	$3FFF	16K H/L
1	0	0	10K	$1800	$3FFF	16K H/L
0	1	1	8K	$0000	$1FFF	8K
0	1	0	6K	$0800	$1FFF	8K H/L
0	0	1	4K	$0000	$0FFF	4K
0	0	0	2K	$0800	$0FFF	2K[1]

[1] This setting will also be chosen for RAM sizes < 2K

The RAMHAL bit allows a non-power of 2 (14K, 12K, 10K and 6K) large RAM to be mapped to either the higher or lower end of the map. For 16K, 8K, 4K, 2K this bit is ignored. All bits in the INITRM register below the map boundary are ignored. E.g. for 10K, 12K, 14K, 16K bits RAM15:RAM14, for 8k and 6K RAM15:13, for 4K RAM15:12, and for 2K RAM15:11 are valid.

EEPROM

The EEPROM memory mapping is determined by the INITEE register shown in **Figure 4**.

Base+$12	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Read:	EE15	EE14	EE13	EE12	EE11	0	0	EEON
Write:						▨	▨	
Reset:	0	0	0	0	0	0	0	0

Figure 4 INITEE Register

The EEPROM size is determined by the MEMSIZ0 register ($1C) bits eep_sw1:eep_sw0 and hard coded for each device.

Table 4 EEPROM Memory Mapping out of Reset

eep_sw1	eep_sw0	Size	Begin Address	End Address	Map Boundary
1	1	8K	$0000	$1FFF	8K
1	0	4K	$0000	$0FFF	4K
0	1	2K	$0000	$07FF	2K(1)
0	0	0	-	-	No EEPROM

1. This setting will be chosen for all EEPROM sizes < 2K

All bits in the INITEE register below the map boundary are ignored. E.g. for 8k or 6K EE15:13, for 4K EE15:12, and for 2K EE15:11 are valid.

EEPROM blocks smaller than 2K are mapped more than once into the 2K address space. E.g. a 512Byte EEPROM mapped to $3800 is repeated from $3800 - $39FF, $3A00 - $3BFF, $3C00 - $3DFF and $3E00 - $3FFF.

Flash/ROM

All devices of the Star12 D-Family support paging for compatibility reasons, even the ones having <= 64K Flash memory. The page usage and memory size are determined by the MEMSIZ1 register ($1D) bits and hard coded for each device.

Base+$1D	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	rom_sw0	rom_sw1	0	0	0	0	pag_sw1	pag_sw0
Dx256	1	0	0	0	0	0	0	1
Dx128	1	0	0	0	0	0	0	0
Dx64	1	1	0	0	0	0	0	0
Dx32	1	0	0	0	0	0	0	0

Figure 5 MEMSIZ1 Read-Only Register

The bits shown in **Table 5** define which of the four 16K areas within the 16-bit address space of the HC12 are allocated to Flash/ROM

Table 5 Allocated Flash/ROM Physical Memory Space

rom_sw1	rom_sw0	Allocated Space	Comments
1	1	64K Bytes	Flash is allocated to all four 16K pages i.e. the whole 64K space[1]
1	0	48K Bytes	Flash is allocated to $4000 - $FFFF, i.e. the top three 16K pages[1]
0	1	16K Bytes	Flash is allocated only to the top 16K page $C000 - $FFFF
0	0	0K Bytes	Used only for Flash/ROM less devices.

[1] If ROMHM = 1 in the MISC register, Flash is allocated only the top two pages $8000 - $FFFF.

The bits in **Table 6** define how much of the 1MByte extended address space is allocated to On-Chip Flash/ROM.

Table 6 Allocated Off-Chip Memory options

pag_sw1	pag_sw0	Off-Chip Space	On-Chip Space
1	1	0K Bytes	1M Bytes
1	0	512K Bytes	512K Bytes
0	1	768K Bytes	256K Bytes
0	0	876K Bytes	128K Bytes

Table 7 summarizes the Flash allocation for all devices sorted by address range and PPAGE for each paged memory block. Figure 6 illustrates the contents of the table in a graphical form.

Table 7 Flash Address, Page Mapping and Protected Areas

MCU Address Range	PPAGE	Features	Dx256	Dx128	Dx64	Dx32
		Total Flash Blocks	**4 (0…3) 32Kx16 each**	**2 (0…1) 32Kx16 each**	**1 32Kx16**	**1 16Kx16**
$C000 - $FFFF	unpaged $3F	Flash Block	0	0	0	0
		Flash Block Relative Address	$C000 - $FFFF	$C000 - $FFFF	$C000 - $FFFF	$4000 - $7FFF
		Protection	$F800 - $FFFF $F000 - $FFFF $E000 - $FFFF $C000 - $FFFF	$F800 - $FFFF $F000 - $FFFF $E000 - $FFFF $C000 - $FFFF	$F800 - $FFFF $F000 - $FFFF $E000 - $FFFF $C000 - $FFFF	$F800 - $FFFF $F000 - $FFFF $E000 - $FFFF $C000 - $FFFF
$8000 - $BFFF	paged $3F	Flash Block	0	0	0	0
		Flash Block Relative Address	$C000 - $FFFF	$C000 - $FFFF	$C000 - $FFFF	$4000 - $7FFF
		Protection	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF

Table 7 Flash Address, Page Mapping and Protected Areas (Continued)

MCU Address Range	**PPAGE**	**Features**	**Dx256**	**Dx128**	**Dx64**	**Dx32**
$8000 - $BFFF	paged $3E	Flash Block	0	0	0	0
		Flash Block Relative Address	$8000 - $BFFF	$8000 - $BFFF	$8000 - $BFFF	$0000 - $3FFF
		Protection	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF
$8000 - $BFFF	paged $3D	Flash Block	0	0	0	*0*
		Flash Block Relative Address	$4000 - $7FFF	$4000 - $7FFF	$4000 - $7FFF	*$4000 - $7FFF*
		Protection	-	-	-	*$B800 - $BFFF* *$B000 - $BFFF* *$A000 - $BFFF* *$8000 - $BFFF*
$8000 - $BFFF	paged $3C	Flash Block	0	0	0	*0*
		Flash Block Relative Address	$0000 - $3FFF	$0000 - $3FFF	$0000 - $3FFF	*$0000 - $3FFF*
		Protection	-	-	-	*$8000 - $81FF* *$8000 - $83FF* *$8000 - $87FF* *$8000 - $8FFF*
$8000 - $BFFF	paged $3B	Flash Block	1	1	*0*	*0*
		Flash Block Relative Address	$C000 - $FFFF	$C000 - $FFFF	*$C000 - $FFFF*	*$4000 - $7FFF*
		Protection	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	*$B800 - $BFFF* *$B000 - $BFFF* *$A000 - $BFFF* *$8000 - $BFFF*	*$B800 - $BFFF* *$B000 - $BFFF* *$A000 - $BFFF* *$8000 - $BFFF*

Table 7 Flash Address, Page Mapping and Protected Areas (Continued)

MCU Address Range	PPAGE	Features	Dx256	Dx128	Dx64	Dx32
$8000 - $BFFF	paged $3A	Flash Block	1	1	*0*	*0*
		Flash Block Relative Address	$8000 - $BFFF	$8000 - $BFFF	*$8000 - $BFFF*	*$0000 - $3FFF*
		Protection	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	*$8000 - $81FF* *$8000 - $83FF* *$8000 - $87FF* *$8000 - $8FFF*	*$8000 - $81FF* *$8000 - $83FF* *$8000 - $87FF* *$8000 - $8FFF*
$8000 - $BFFF	paged $39	Flash Block	1	1	*0*	*0*
		Flash Block Relative Address	$4000 - $7FFF	$4000 - $7FFF	*$4000 - $7FFF*	*$4000 - $7FFF*
		Protection	-	-	-	*$B800 - $BFFF* *$B000 - $BFFF* *$A000 - $BFFF* *$8000 - $BFFF*
$8000 - $BFFF	paged $38	Flash Block	1	1	*0*	*0*
		Flash Block Relative Address	$0000 - $3FFF	$0000 - $3FFF	*$0000 - $3FFF*	*$0000 - $3FFF*
		Protection	-	-	-	*$8000 - $81FF* *$8000 - $83FF* *$8000 - $87FF* *$8000 - $8FFF*
128K Internal Address Space Boundary						
$8000 - $BFFF	paged $37	Flash Block	2	-	-	-
		Flash Block Relative Address	$C000 - $FFFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	-	-	-

Table 7 Flash Address, Page Mapping and Protected Areas (Continued)

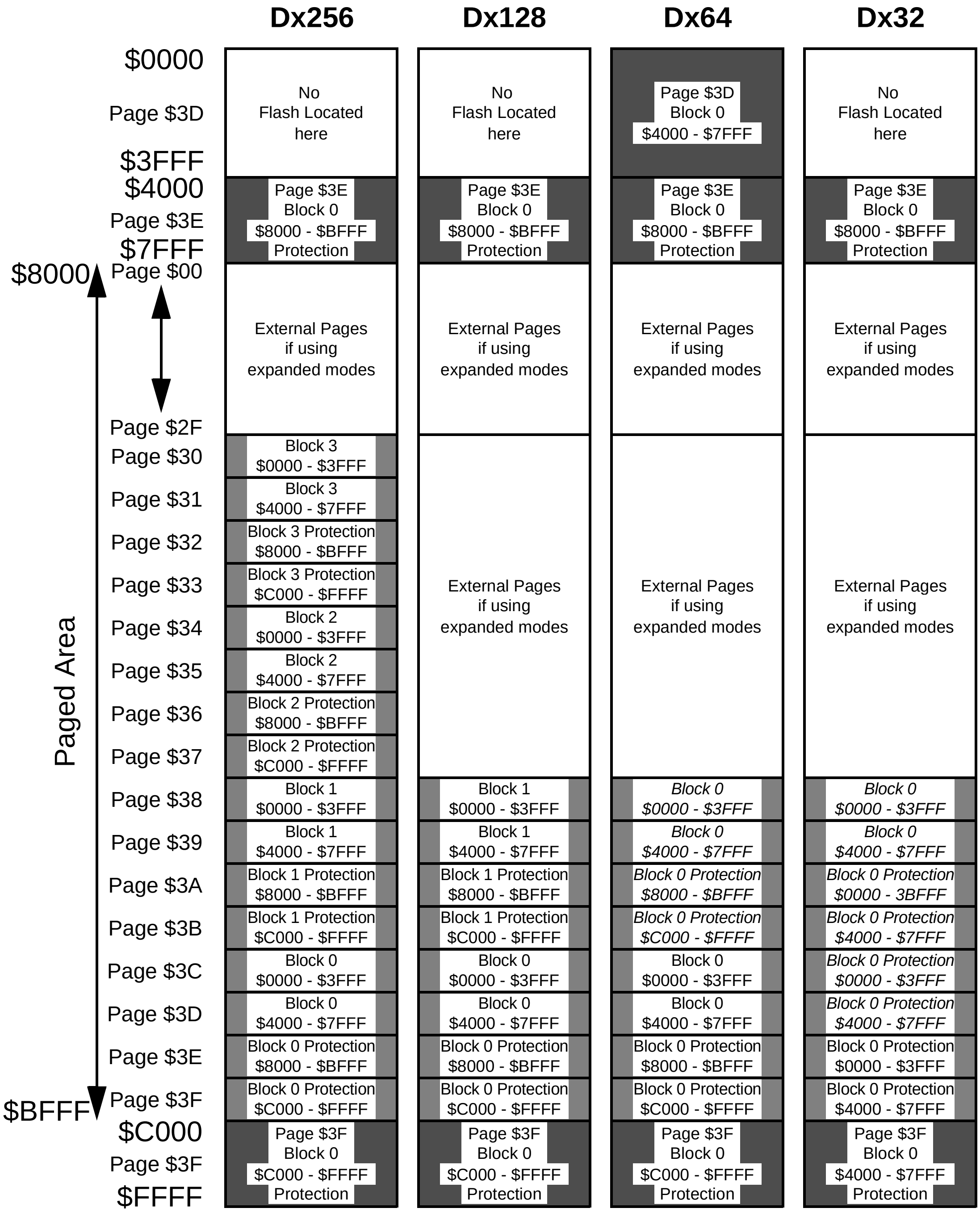
MCU Address Range	PPAGE	Features	Dx256	Dx128	Dx64	Dx32
$8000 - $BFFF	paged $36	Flash Block	2	-	-	-
		Flash Block Relative Address	$8000 - $BFFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	-	-	-
$8000 - $BFFF	paged $35	Flash Block	2	-	-	-
		Flash Block Relative Address	$4000 - $7FFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	-	-	-	-
$8000 - $BFFF	paged $34	Flash Block	2	-	-	-
		Flash Block Relative Address	$0000 - $3FFF	-	external in expanded modes	external in expanded modes
		Protection	-	-	-	-
$8000 - $BFFF	paged $33	Flash Block	3	-	-	-
		Flash Block Relative Address	$C000 - $FFFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	$B800 - $BFFF $B000 - $BFFF $A000 - $BFFF $8000 - $BFFF	-	-	-
$8000 - $BFFF	paged $32	Flash Block	3	-	-	-
		Flash Block Relative Address	$8000 - $BFFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	$8000 - $81FF $8000 - $83FF $8000 - $87FF $8000 - $8FFF	-	-	-

Table 7 Flash Address, Page Mapping and Protected Areas (Continued)

MCU Address Range	PPAGE	Features	Dx256	Dx128	Dx64	Dx32
$8000 - $BFFF	paged $31	Flash Block	3	-	-	-
		Flash Block Relative Address	$4000 - $7FFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	-	-	-	-
$8000 - $BFFF	paged $30	Flash Block	3	-	-	-
		Flash Block Relative Address	$0000 - $3FFF	external in expanded modes	external in expanded modes	external in expanded modes
		Protection	-	-	-	-
256K Internal Address Space Boundary						
$8000 - $BFFF	paged $00 … $2F	768K Byte external space	external in expanded modes	external in expanded modes	external in expanded modes	external in expanded modes
$4000 - $7FFF	unpaged $3E	Flash Block	0	0	0	0
		Flash Block Relative Address	$8000 - $BFFF	$8000 - $BFFF	$8000 - $BFFF	$0000 - $3FFF
		Protection	$4000 - $41FF $4000 - $43FF $4000 - $47FF $4000 - $4FFF	$4000 - $41FF $4000 - $43FF $4000 - $47FF $4000 - $4FFF	$4000 - $41FF $4000 - $43FF $4000 - $47FF $4000 - $4FFF	$4000 - $41FF $4000 - $43FF $4000 - $47FF $4000 - $4FFF
$0000 - $3FFF	unpaged $3D	Flash Block	-	-	0	-
		Flash Block Relative Address	-	-	$4000 - $7FFF	-
		Protection	-	-	-	-

Remarks

1. If ROMHM = 1 all Flash is removed from address range $0000 - $7FFF.
2. When developing for a device with 32K Byte Flash using non-banked memory model, the PPAGE register should be written to $3E to map the $3E page into the $8000 - $BFFF address range.
3. When developing for a device with 64K Byte Flash using a non-banked memory model, the PPAGE register should be written to $3C to map the $3C page into the $8000 - $BFFF address range.
4. Flash can be addressed in the bottom 16k page ($0000 - $3FFF) only for the Dx64. This addressing scheme can not be emulated by any of the devices with larger Flash. It offers however the advantage of a larger non-banked flash space.



		Dx256	Dx128	Dx64	Dx32
$0000 $3FFF	Page $3D	No Flash Located here	No Flash Located here	Page $3D Block 0 $4000 - $7FFF	No Flash Located here
$4000 $7FFF	Page $3E	Page $3E Block 0 $8000 - $BFFF Protection	Page $3E Block 0 $8000 - $BFFF Protection	Page $3E Block 0 $8000 - $BFFF Protection	Page $3E Block 0 $8000 - $BFFF Protection
$8000	Page $00 Page $2F	External Pages if using expanded modes	External Pages if using expanded modes	External Pages if using expanded modes	External Pages if using expanded modes
	Page $30	Block 3 $0000 - $3FFF	External Pages if using expanded modes	External Pages if using expanded modes	External Pages if using expanded modes
	Page $31	Block 3 $4000 - $7FFF			
	Page $32	Block 3 Protection $8000 - $BFFF			
	Page $33	Block 3 Protection $C000 - $FFFF			
	Page $34	Block 2 $0000 - $3FFF			
	Page $35	Block 2 $4000 - $7FFF			
	Page $36	Block 2 Protection $8000 - $BFFF			
	Page $37	Block 2 Protection $C000 - $FFFF			
	Page $38	Block 1 $0000 - $3FFF	Block 1 $0000 - $3FFF	*Block 0* *$0000 - $3FFF*	*Block 0* *$0000 - $3FFF*
	Page $39	Block 1 $4000 - $7FFF	Block 1 $4000 - $7FFF	*Block 0* *$4000 - $7FFF*	*Block 0* *$4000 - $7FFF*
	Page $3A	Block 1 Protection $8000 - $BFFF	Block 1 Protection $8000 - $BFFF	*Block 0 Protection* *$8000 - $BFFF*	*Block 0 Protection* *$0000 - 3BFFF*
	Page $3B	Block 1 Protection $C000 - $FFFF	Block 1 Protection $C000 - $FFFF	*Block 0 Protection* *$C000 - $FFFF*	*Block 0 Protection* *$4000 - $7FFF*
	Page $3C	Block 0 $0000 - $3FFF	Block 0 $0000 - $3FFF	Block 0 $0000 - $3FFF	*Block 0 Protection* *$0000 - $3FFF*
	Page $3D	Block 0 $4000 - $7FFF	Block 0 $4000 - $7FFF	Block 0 $4000 - $7FFF	*Block 0 Protection* *$4000 - $7FFF*
	Page $3E	Block 0 Protection $8000 - $BFFF	Block 0 Protection $8000 - $BFFF	Block 0 Protection $8000 - $BFFF	Block 0 Protection $0000 - $3FFF
$BFFF	Page $3F	Block 0 Protection $C000 - $FFFF	Block 0 Protection $C000 - $FFFF	Block 0 Protection $C000 - $FFFF	Block 0 Protection $4000 - $7FFF
$C000 $FFFF	Page $3F	Page $3F Block 0 $C000 - $FFFF Protection	Page $3F Block 0 $C000 - $FFFF Protection	Page $3F Block 0 $C000 - $FFFF Protection	Page $3F Block 0 $4000 - $7FFF Protection

Figure 6 Flash Layout

External address space

All derivatives offer in expanded modes an external address space in the 16k page starting at $8000 with a corresponded setting of the PPAGE register. Another unpaged memory area from $4000 - $7FFF can be freed up by setting the ROMHM=1 bit. A third option for a smaller window is the free space in the bottom 16k page. Depending on the device the register space might need remapping or setting the ROMHM bit.

Example 1

In this example the register space is always located at $0000, and the RAM space grows downward from $3FFF. Allocating the registers in the direct page is very common for I/O intensive programs making use of the faster access. The recommended initialization is:

```
initrg = 0x00;
initrm = 0x39;
initee = 0x09;
```

This will map any RAM independent of its size such that its top address is located at $3FFF (usually the STACK location). The variables can than start at, for example $3800, if the final target is a device using 2K bytes of RAM. The EEPROM would be always reach to $0FFF, having the protection and reserved locations from $0FF0 - $0FFF independent of the EEPROM size. In case of a 4K block only the top 3K are accessible. If full 4K EEPROM is required, the registers can be mapped on top of RAM or into the 16k page starting at $4000. This layout is shown also in **Figure 7**.

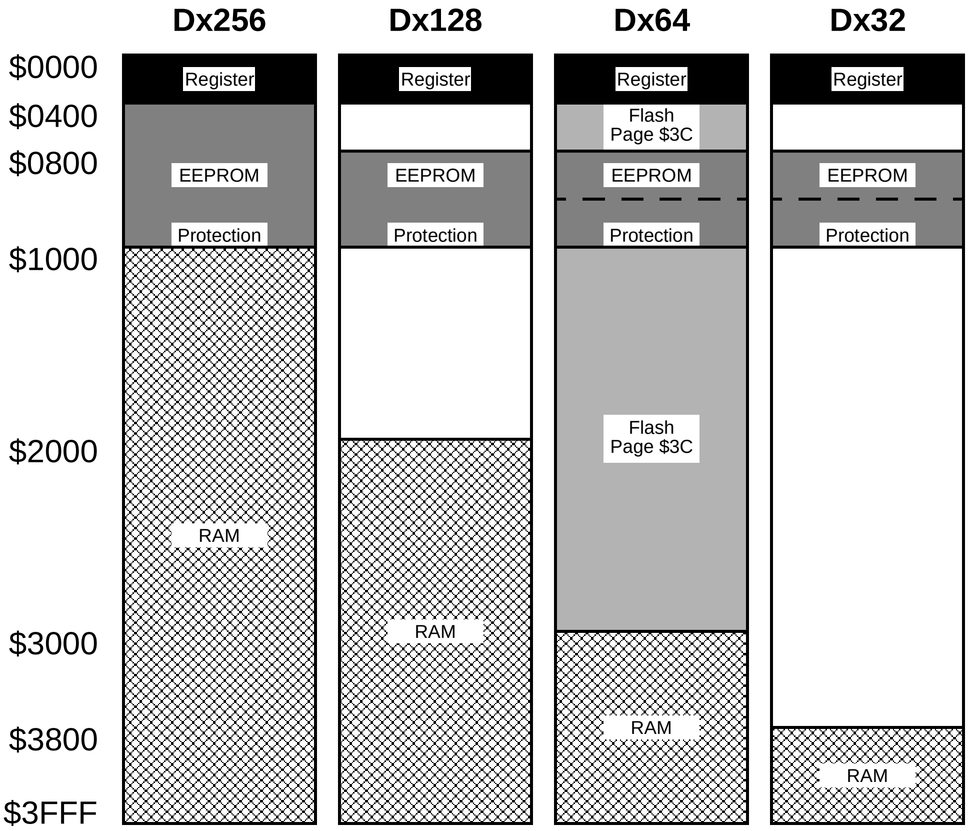


Figure 7 Example 1 Layout

Example 2

In this example the RAM is located at $0000 and grows towards higher addresses. This configuration is often used in highly computing intensive programs allocating the variables in the direct page.

```
initrg = 0x30;
initrm = 0x00;
initee = 0x39;
```

The register space is allocated at $3000 right after the 12K maximum RAM space. Variables can be configured to start at $0000 with the stack growing downward from $07FF for a part with 2K RAM. The EEPROM is aligned to the top at $3FFF having the protection and reserved locations from $3FF0 - $3FFF independent of the EEPROM size. In case of a 4K block only the top 3K are accessible. If full 4K EEPROM is required, the registers can be mapped on top of RAM or into the 16k page starting at $4000. This layout is shown also in **Figure 8**.

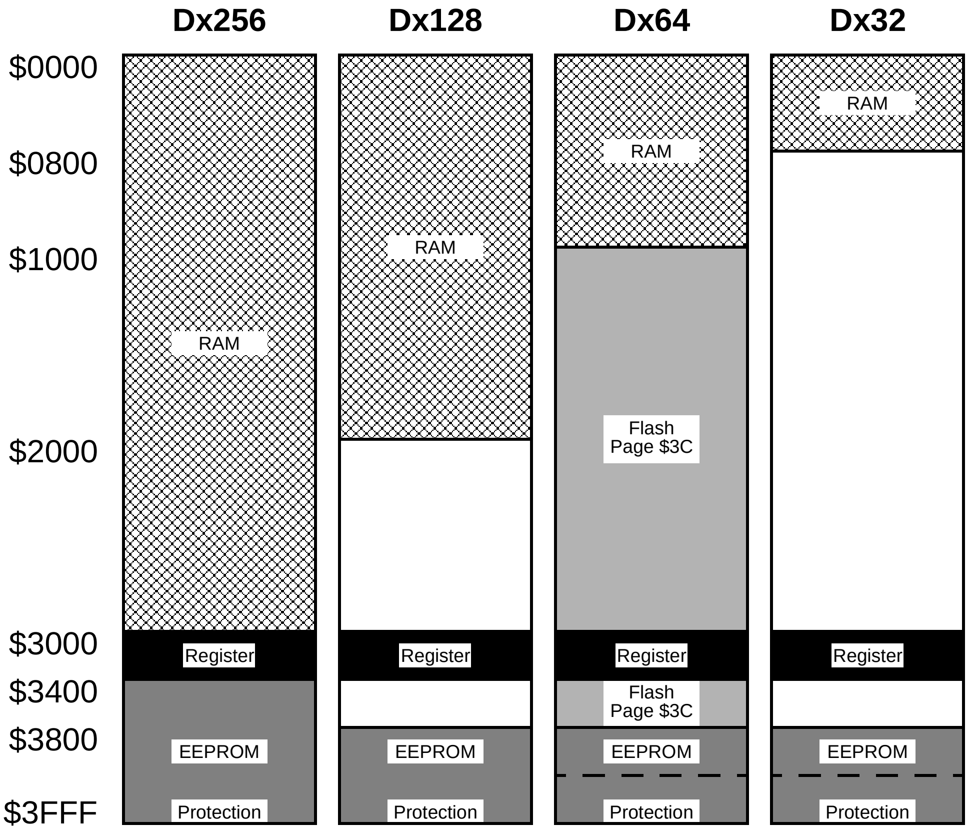


Figure 8 Example 2 Layout

Peripherals

Table 8 shows the various configurations available on the various devices of the D-Family. All address locations of the peripheral modules are consistent across the D-Family. The areas occupied by non-implemented peripheral modules read back as zero.

Table 8 Peripherals by Device sorted by Device Types

	Pins	CAN0	CAN1	CAN2	CAN3	CAN4	J1850	IIC	SCI0	SCI1	SPI0	SPI1	SPI2	Byteflight	ATD0	ATD1	PWM	Timer [1]
9S12DP256	112	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	8	E
9S12DT256	112	✔	✔			✔		✔	✔	✔	✔	✔	✔		✔	✔	8	E
9S12DG256	112	✔				✔		✔	✔	✔	✔	✔	✔		✔	✔	8	E
9S12DJ256	112	✔				✔	✔	✔	✔	✔	✔	✔	✔		✔	✔	8	E
9S12DJ256	80	✔				✔	✔	✔	✔	✔	✔	✔	✔		✔		7	E
9S12DT128	112	✔	✔			✔		✔	✔	✔	✔	✔			✔	✔	8	E
9S12DG128	112	✔				✔		✔	✔	✔	✔	✔			✔	✔	8	E
9S12DJ128	112	✔				✔	✔	✔	✔	✔	✔	✔			✔	✔	8	E
9S12DB128	112	✔				✔			✔	✔	✔	✔		✔	✔	✔	8	E
9S12DG128	80	✔				✔		✔	✔	✔	✔	✔			✔		7	E
9S12DJ128	80	✔				✔	✔	✔	✔	✔	✔	✔			✔		7	E
9S12D64	112	✔						✔	✔	✔	✔				✔	✔	8	E
9S12DJ64	112	✔					✔	✔	✔	✔	✔				✔	✔	8	E
9S12D64	80	✔						✔	✔	✔	✔				✔		7	E
9S12DJ64	80	✔					✔	✔	✔	✔	✔				✔		7	E
9S12D32	80	✔							✔	✔	✔				✔		7	E
9S12DB32	80								✔		✔			✔	✔		4	E

[1] “E” denotes “Enhanced Capture Timer”.

Table 9 shows the differences in the memory map across the members of the D-Family. All unimplemented peripherals read $0000.

NOTE: *In the initial product launch phase all devices mentioned in* ***Table 8*** *will be supported by four different dies (9S12DP256, 9S12DTB128, 9S12DJ64, and 9S12DB32) with a super set of features. The 9SDB32 supports 80 pin packages only, while the other dies can be bonded in either 112 or 80 pin packages*

Table 9 Star12 Register Memory Map Differences

Address	Module[1]	9S12DP256	9S12DTB128	9S12DJ64	9S12DB32
$00E0 - $00E7	Inter IC Bus	✔	✔	✔	
$00E8 - $00EF	Byte Data Link Controller (BDLC)	✔	✔	✔	
$00F0 - $00F7	Serial Peripheral Interface (SPI1)	✔	✔		
$00F8 - $00FF	Serial Peripheral Interface (SPI2)	✔			
$0120 - $013F	Analog to Digital 10-bit 8 channels (ATD1)	✔	✔	✔	
$0180 - $01BF	CAN (CAN1)	✔	✔		
$01C0 - $01FF	CAN (CAN2)	✔			
$0200 - $023F	CAN (CAN3)	✔			
$0240 - $027F	Port Integration Module (PIM)[2]	✔	✔	✔	✔
$0280 - $02BF	CAN (CAN4)	✔	✔		
$0300 - $035F	Byteflight (BF)		✔		✔

[1] Not all features are available in some of the 80 Pin packages.

[2] The Port Integration modules differ from each other but are upwards compatible.

Table 10 shows the differences in the allocation of peripheral interrupts. The inputs for unimplemented peripheral modules are tied to ground so that no interrupts for those vectors can occur.

Table 10 Interrupt Vector Location Differences

Vector Address	Interrupt Source	9S12DP256	9S12DTB128	9S12DJ64	9S12DB32
$FFD0, $FFD1	ATD 1	✔	✔	✔	
$FFCE, $FFCF	Port J	✔	✔	✔	✔
$FFCC, $FFCD	Port H	✔	✔	✔	
$FFC2, $FFC3	BDLC	✔	✔	✔	
$FFC0, $FFC1	IIC Bus	✔	✔	✔	
$FFBE, $FFBF	SPI1	✔	✔		
$FFBC, $FFBD	SPI2	✔			
$FFAE, $FFAF	CAN 1 wake-up	✔	✔		
$FFAC, $FFAD	CAN 1 errors	✔	✔		
$FFAA, $FFAB	CAN 1 receive	✔	✔		
$FFA8, $FFA9	CAN 1 transmit	✔	✔		
$FFA6, $FFA7	MSCAN 2 wake-up	✔			
	Byteflight General		✔		✔
$FFA4, $FFA5	CAN 2 errors	✔			
	Byteflight Synchronize		✔		✔
$FFA2, $FFA3	CAN 2 receive	✔			
	Byteflight Receive		✔		✔
$FFA0, $FFA1	CAN 2 transmit	✔			
	Byteflight Receive FIFO		✔		✔
$FF9E, $FF9F	CAN 3 wake-up	✔			
$FF9C, $FF9D	CAN 3 errors	✔			
$FF9A, $FF9B	CAN 3 receive	✔			
$FF98, $FF99	CAN 3 transmit	✔			
$FF96, $FF97	CAN 4 wake-up	✔	✔	✔	
$FF94, $FF95	CAN 4 errors	✔	✔	✔	
$FF92, $FF93	CAN 4 receive	✔	✔	✔	
$FF90, $FF91	CAN 4 transmit	✔	✔	✔	

Register Differences

The following read-only registers differ on any device, since they identify the device itself. PARTIDH, PARTIDL, MEMSIZ1, MEMSIZ0. If 100% binary code should be developed, the programmer has to take care on this.

I/O Port Compatibility Considerations

This chapter discusses the **differences** in the Port Integration Module (PIM) Module Routing Register (MODRR) and the missing pins in 80 pin packages.

PIM Differences

Since the parts have a different set of peripherals, some minor differences in the PIM must be considered.

The most important register affected by changes is the module routing register shown in **Figure 9**.

Device	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		MODRR6	MODRR5	MODRR4	MODRR3	MODRR2	MODRR1	MODRR0
Dx256	0	✔	✔	✔	✔	✔	✔	✔
Dx128	0	0	✔	✔	✔	✔	✔	✔
Dx64	0	0	0	✔	0	0	✔	✔
Dx32	0	0	0	✔	0	0	✔	✔

Figure 9 Module Routing Register (Base+$257)

MODRR6

This bit is associated with SPI2 and therefor only available on devices supporting SPI2.

MODRR5

This bit is associated with SPI1 and therefor only available on devices supporting SPI1.

MODRR4

This bit is associated with SPI0 and therefor only available on devices supporting SPI0. On devices supporting only 80 pin packages (D32) the SPI0 is only available on port M5:2 if the MODRR4 bit is set. This ensures backwards compatibility with family members supporting 112 pin as well as 80 pin packages.

MODRR[3:2]

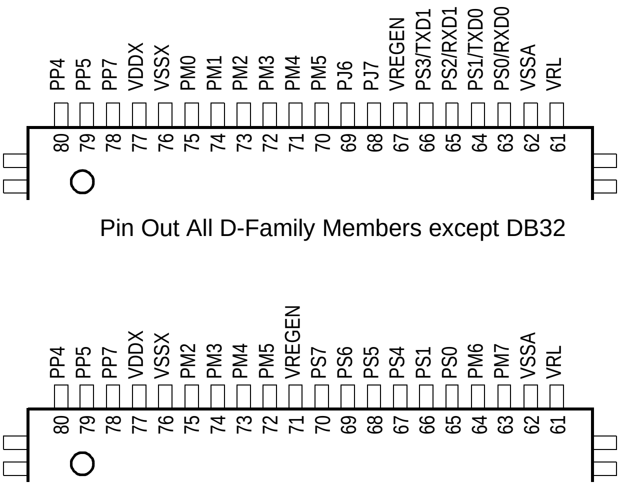
Those bits are only available on devices supporting CAN4 module.

NOTE: *Writing to unused bits will have no effect. Those bits will stick to “0”.*

80 Pin Package Pitfalls

In the 80 QFP package several things need to be considered to avoid problems with floating pins etc. With the exception of the Dx32, one die supports the 112 pin as well as the 80 pin package for each device. This means that the PIM module registers of non bonded pins are available on those chips. On the devices supporting 80 Pins only (Dx32) they are not available on silicon and the respective addresses in the PIM area read back “0”. When developing for 80 pin packages using a 112 pin capable umbrella part several control registers must be written to avoid internally floating pins.

The DB32 bond out differs from all other 80 Pin bond-outs as shown in **Figure 10**.



Pin Out All D-Family Members except DB32

Pin Out 9S12DB32

Figure 10 Bond-out 9S12Dx vs. 9S12DB32

Table 11 80 Pin Package Bond-out Summary

Ports	DP256/DTB128/D64	D32	DB32
Port H	Not bonded	Not on Silicon	Not on Silicon
Port J[1:0]	Not bonded	Not on Silicon	Not on Silicon
Port J[7:6]	✔	✔	Not bonded
Port K7, [5:0]	Not bonded	Not bonded	Not bonded
Port M[7:6]	Not bonded	Not bonded	✔
Port M[1:0]	Not bonded	✔	Not bonded
Port PP6	Not bonded	Not on Silicon	Not on Silicon
Port S[7:4]	Not bonded	Not bonded	✔
Port S[3:2]	Not bonded	✔	Not bonded
PAD[15:8]	Not bonded	Not on Silicon	Not on Silicon

For non-bonded ports the software has to assure that the internal inputs do not float. Address locations for ports shown as “Not on Silicon” read back as “0”.

Port H

In order to avoid floating nodes the ports should be either configured as outputs by setting the data direction register (DDRH at Base+$0261) to $FF, or enabling the pull resistors by writing a $FF to the pull enable register (PERH at Base+$0264).

Port J

Port J pull-up resistors are enabled out of reset on all four pins (7:6 and 1:0). Therefor care must be taken not to disable the pull enables by clearing the PERJ at Base+$026C bits J7:6 and J1:0.

Port K

Since port K is integral part of the core the relevant registers are also available on the devices supporting only 80 pin packages (Dx32). Port K pull-up resistors are enabled out of reset, i.e. Bit 7 = PUKE = 1 in the register PUCR at Base+$000C. Therefor care must be taken not to clear this bit.

Port M

Since all pins and register bits of Port M are available on all chips PM7:6 respectively PM1:0 must be configured as outputs or their pull resistors must be enabled to avoid floating inputs.

Port P

Port PP6 is not bonded out in the 80 pin packages. Port PP6 must be configured as output or its pull resistor must be enabled to avoid a floating input.

Port S

Since all pins and register bits of Port S are available on all chips PS7:4 respectively PS3:2 must be configured as outputs or their pull resistors must be enabled to avoid floating inputs.

ATD1

The pins PAD08-PAD15 associated with ATD1 are not available in the 80 Pin package. On the devices supporting only 80 pin packages (Dx32) ATD1 is not available on silicon and register addresses $120 - $13F read “0”. Out of reset the ATD is disabled preventing current flows in the pins. Do not modify the ATD1 registers!

How to reach us:

USA/EUROPE: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140
HOME PAGE: http://motorola.com/sps/
JAPAN: Motorola Japan Ltd.; SPS, Technial Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569
ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-266668334
CUSTOMER FOCUS CENTER: 1-800-521-6274

MOTOROLA