Reset Controller

What you will learn

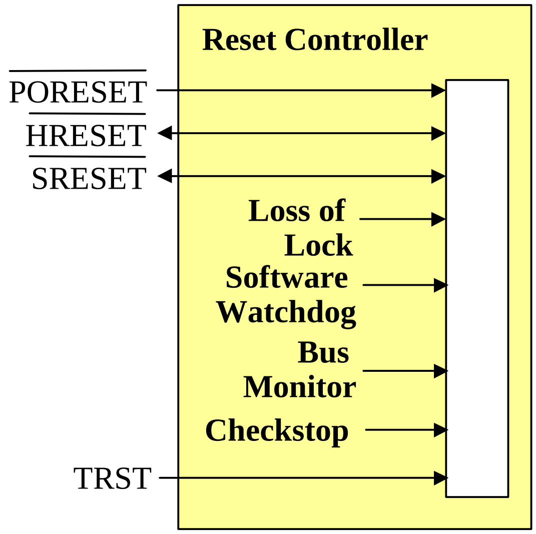
- What reset sources are available?
- How the reset inputs are handled
- How to initialize the clocks
- How to configure from reset
- How to initialize from reset

What is the Reset Controller?

Definition

The reset controller responds to the assertion of a reset source. The action it takes depends on the source, but typically it will execute the routine at location 0x100 in the exception vector table.

Block Diagram



Reset Controller

$\overline{\text{PORESET}}$

$\overline{\text{HRESET}}$

$\overline{\text{SRESET}}$

Loss of Lock

Software Watchdog

Bus Monitor

Checkstop

TRST

Reset Status Register

The Reset Status Register (RSR) indicates the last source to cause a reset.

Features

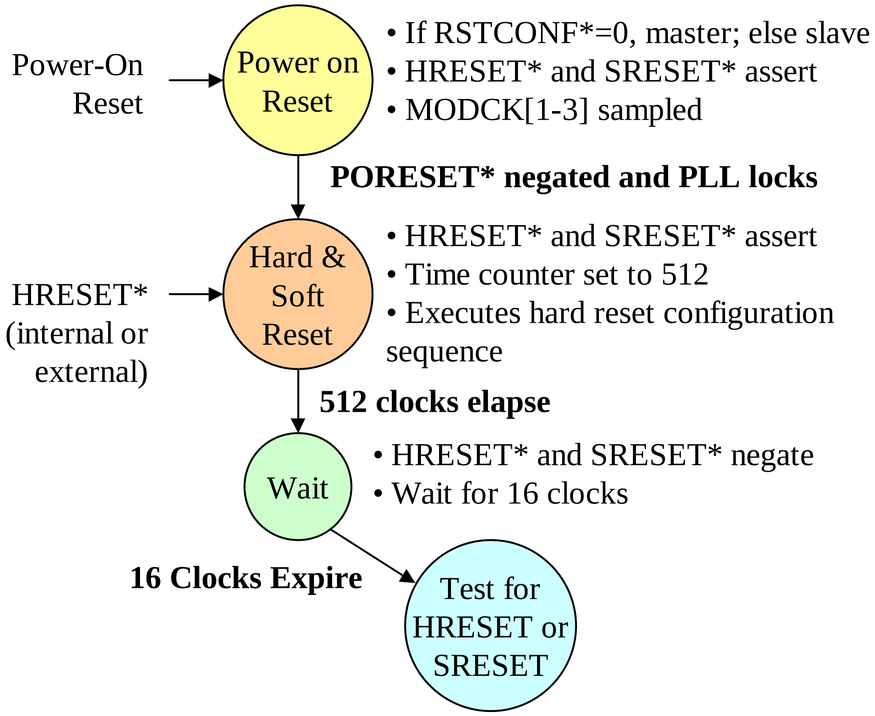
1. There are eight sources of interrupts. Four are external pins and four are internal sources.
2. HRESET* and SRESET* are bidirectional. They can be asserted by external devices, therefore, inputs. From PORESET, they will be asserted also, therefore they can be outputs.

How the Reset Pin Inputs are Handled

Introduction

This diagram shows the operation and interactions of Power-on-reset, Hard Reset, and Soft Reset.

State Diagram



Power-On Reset

Power on Reset

- If RSTCONF*=0, master; else slave
- HRESET* and SRESET* assert
- MODCK[1-3] sampled

PORESET* negated and PLL locks

HRESET* (internal or external)

Hard & Soft Reset

- HRESET* and SRESET* assert
- Time counter set to 512
- Executes hard reset configuration sequence

512 clocks elapse

Wait

- HRESET* and SRESET* negate
- Wait for 16 clocks

16 Clocks Expire

Test for HRESET or SRESET

Description

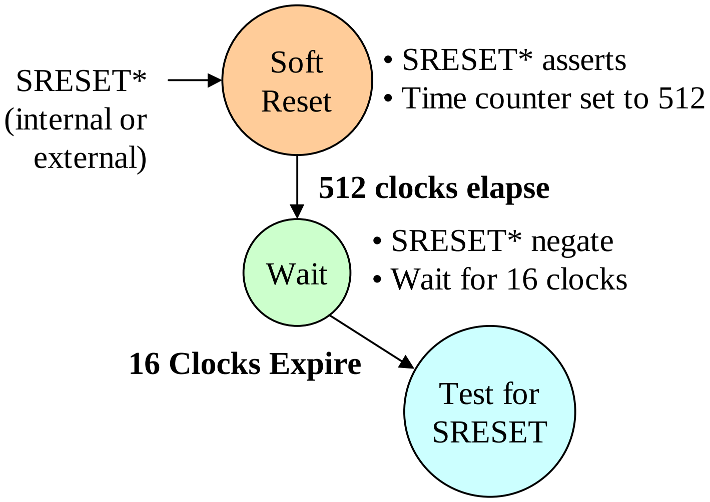
1. When the power-on-reset pin is asserted (for at least 16 clocks after external power is at least 2/3 VCC), the MPC8260 enters the Power on Reset State. If the RSTCONF pin is asserted, this device becomes a master, otherwise a slave.
2. When PORESET is negated and the PLL locks, the 8260 moves to the Hard and Soft Reset state where assertion of the hard and soft reset pins is assured for 512 clocks and the hard reset configuration occurs.
3. After 512 clocks elapses, the 8260 moves into the Wait state for 16 clocks. Here HRESET and SRESET are negated.
4. Finally, the 8260 goes into the Test for HRESET and SRESET state where, if either pin should be asserted, it will again respond.

How Soft Reset Inputs are Handled

Introduction

This diagram shows the operation of Soft Reset.

State Diagram



SRESET* (internal or external)

Soft Reset

- SRESET* asserts
- Time counter set to 512

512 clocks elapse

Wait

- SRESET* negate
- Wait for 16 clocks

16 Clocks Expire

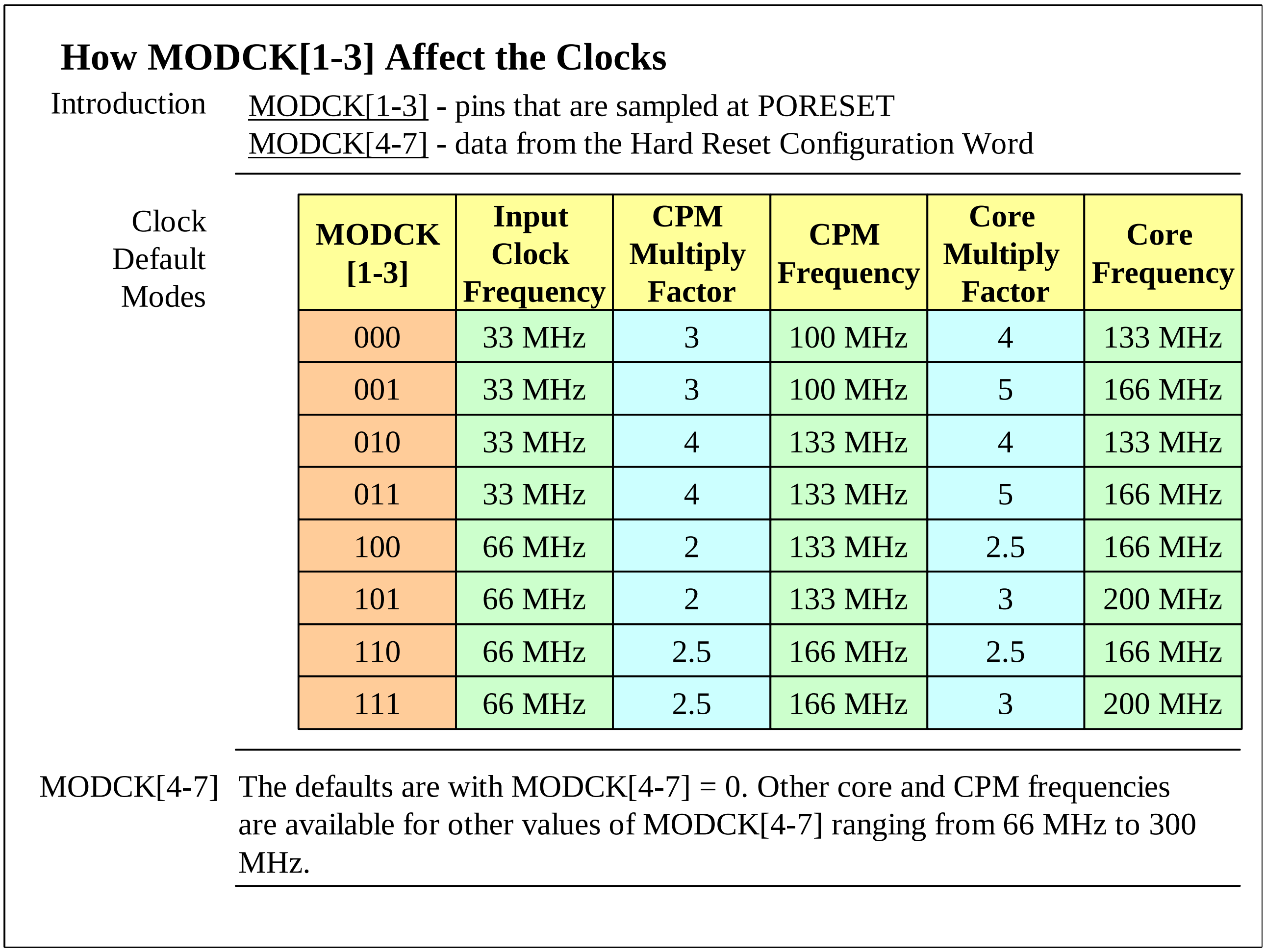
Test for SRESET

SRESET Action

When SRESET is asserted, internal hardware is reset but the hard reset configuration does not change.

Description

1. When When SRESET is asserted, the 8260 moves into the Soft Reset state.
3. After 512 clocks elapses, the 8260 moves into the Wait state for 16 clocks. Here SRESET are negated.
4. Finally, the 8260 goes into the Test for SRESET state where, if the SRESET pin should be asserted, it will again respond.



How MODCK[1-3] Affect the Clocks

Introduction

MODCK[1-3] - pins that are sampled at PORESET
MODCK[4-7] - data from the Hard Reset Configuration Word

Clock Default Modes

MODCK [1-3]	Input Clock Frequency	CPM Multiply Factor	CPM Frequency	Core Multiply Factor	Core Frequency
000	33 MHz	3	100 MHz	4	133 MHz
001	33 MHz	3	100 MHz	5	166 MHz
010	33 MHz	4	133 MHz	4	133 MHz
011	33 MHz	4	133 MHz	5	166 MHz
100	66 MHz	2	133 MHz	2.5	166 MHz
101	66 MHz	2	133 MHz	3	200 MHz
110	66 MHz	2.5	166 MHz	2.5	166 MHz
111	66 MHz	2.5	166 MHz	3	200 MHz

MODCK[4-7] The defaults are with MODCK[4-7] = 0. Other core and CPM frequencies are available for other values of MODCK[4-7] ranging from 66 MHz to 300 MHz.

Description

1. Here’s a summary of the operating frequency options for all values of MODCK[1-3].

2. Getting some clock information from the pins and some from the configuration word is a compromise between the number of pins used and the flexibility offered. One must roughly select the clock frequencies using pins for the system to start and fetch the reset configuration word. Then a fimer tune is possible.

Programming Model

RSR - Reset Status Register P. 5-4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-															

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
-										JTRS	CSRS	SWRS	BMRS	ESRS	EHRS

RMR - Reset Mode Register P. 5-5

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
-															CSRE

Hard Reset Configuration Word P. 5-7

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EARB	EXMC	CDIS	EBM	BPS		CIP	ISPS	L2CPC		DPPC		-	ISB		

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
BMS	BBD	MMR		LBPC		APPC		CS10PC		PHE	-	MODCK_HI			

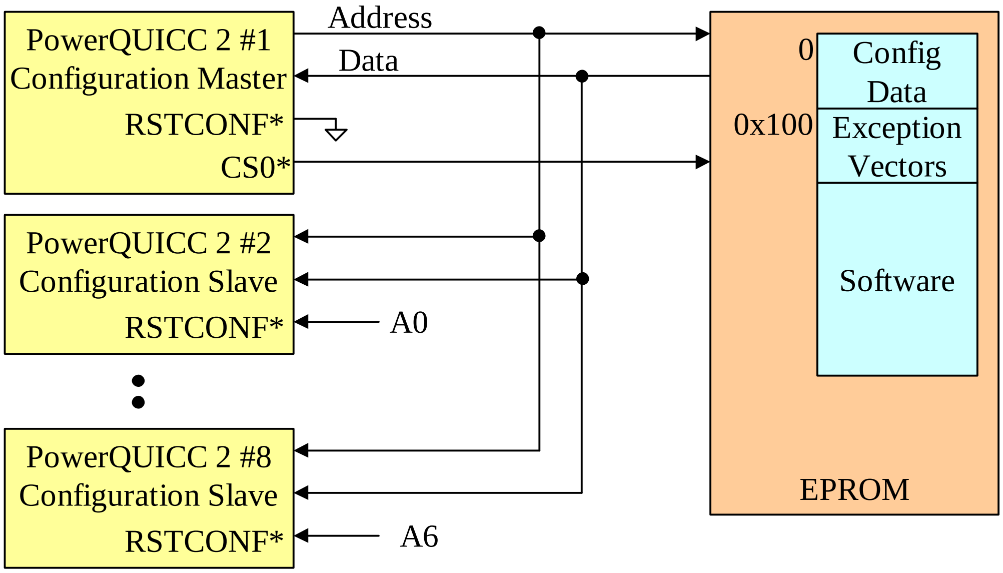
Description

1. This is the programming model for reset.
2. In the RSR, there are 6 bits, one for each possible reset source (except TRST). These bits can be read to determine the source of the interrupt. Then the bit is cleared by writing a 1 to it.
3. RMR has one bit which determines if a checkstop will cause a reset or if it will allow the processor to stay in the checkstop state.
4. The Hard Reset Configuration Word is initialized from reset providing initial values to a number of parameters such as ISB, the initial internal space base select.

How to Connect for Reset

Introduction The diagram below shows the connections for reset.

Block Diagram



Enables RSTCONF* sampled at PORESET negated edge. The configuration is performed while HRESET* is asserted.

Description 1. The PowerQUICC 2s in the system are setup such that there is one configuration master (RSTCONF always low), amd multiple slaves (RSTCONF connected to A0 to A6 as necesssary).
2. Configuration data is located at address 0 of the exception table.
3. To program on-board flash from reset, the user can drive RSTCNF high to select the default configuration, program the flash, then drive RSTCONF low during subsequent resets.

How the Master Configures the PowerQUICC 2s

Introduction

The diagram below shows the steps in configuration.

Configuration Master Reset Steps

Access	R/W	Address	Description
1	R	0xFE000000	Read 4 bytes and store to Hard Reset Config Word, Master
2	R	0xFE000008	
3	R	0xFE000010	
4	R	0xFE000018	
5	R	0xFE000020	Read 4 bytes and store to Hard Reset Config Word, 1st slave
6	R	0xFE000028	
7	R	0xFE000030	
8	R	0xFE000038	
9	W	0x7E000000	
		⋮	
35	R	0xFE0000E0	Read 4 bytes and store to Hard Reset Config Word, 7th slave
36	R	0xFE0000E8	
37	R	0xFE0000F0	
38	R	0xFE0000F8	
39	W	0xFC000000	

Description

1. All reads are 1 byte. All writes are 1 word.
2. Because the configuration master has no advance knowledge about the number of slaves, it always executes all accesses coming out of reset.

How to Determine the State of the Pins during Reset

Introduction

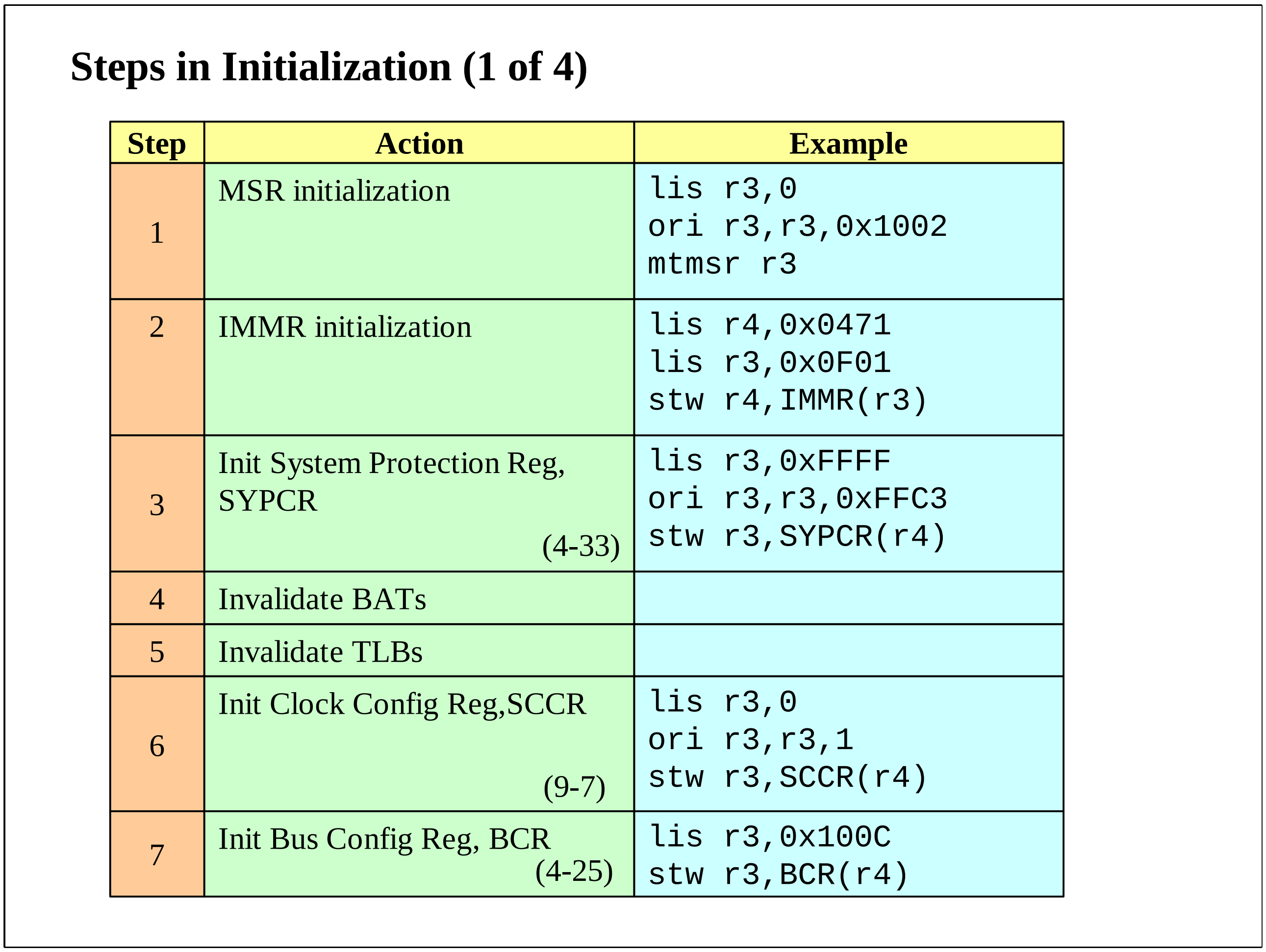
The following table describes the state of the pins for PORESET*, HRESET* and SRESET*.

PORESET* and HRESET* Table

If...	Then for PORESET* and HRESET*...
Pin can be input or output	Pin is an input
Pin is three-statable	Pin is in high-impedance
Pin is always an output	Pin is driven to negated logical value

SRESET*

Pin assignments and direction are not affected by SRESET*.



Steps in Initialization (1 of 4)

Step	Action	Example

1	MSR initialization	`lis r3,0`
		`ori r3,r3,0x1002`
		`mtmsr r3`

2	IMMR initialization	`lis r4,0x0471`
		`lis r3,0x0F01`
		`stw r4,IMMR(r3)`

3	Init System Protection Reg, SYPCR (4-33)	`lis r3,0xFFFF`
		`ori r3,r3,0xFFC3`
		`stw r3,SYPCR(r4)`

4	Invalidate BATs	
5	Invalidate TLBs	

6	Init Clock Config Reg,SCCR (9-7)	`lis r3,0`
		`ori r3,r3,1`
		`stw r3,SCCR(r4)`

7	Init Bus Config Reg, BCR (4-25)	`lis r3,0x100C`
		`stw r3,BCR(r4)`

Comments

Step 1:Enables the machine check exception and recoverable interrupt mode.
Step 2: Initializes IMMR to a new value. The original value input to the hardware configuration word at reset must be known. If the original value is satisfactory, this step can be skipped.
Step 3: Sets up the software watchdog and bus monitor (hardware watchdog). Can be written to only one time.
Step 4: The valid bits for the Block Address Translation in the MMU can come up valid from reset. They must be invalidated.
Step 5: The Translation Lookaside Buffer entries in the MMU can come up valid from reset. They must be invalidated.
Step 6: The System Clock Control register affects low power and BRGCLK.
Step 7: Affects various configuration parameters of the 60x bus including External Bus Mode, Secondary Cache, and parity type.

Steps in Initialization (2 of 4)

Step	Action	Comment

8	Init 60x Bus Arbiter Config Reg, PPC_ACR (4-27)	`li r3,2`
		`stb r3,PPC_ACR(r4)`

9	Init 60x Bus Arbitration-Level Regs, PPC_ALRH & L (4-28)	`lis r3,0x0126`
		`ori r3,r3,0x7893`
		`stw r3,PPC_ALRH(r4)`

10	Init SIU Module Config Reg, SIUMCR (4-30)	`lis r3,0x0820`
		`stw r3,SIUMCR(r4)`

11	Init 60x Bus Transfer Error Status & Control Reg, TESCR1 (4-34)	`li r3,0x4000`
		`stw r3,TESCR1(r4)`

12	Init Local Bus Transfer Error Status & Control Reg, LTESCR1 (4-36)	`li r3,0x4000`
		`stw r3,LTESCR1(r4)`

13	Init Memory Refresh Timer Prescalar Reg, MPTPR (10-32)	`li r5,0x4000`
		`sth r5,MPTPR(r4)`

Comments

Step 8: Defines the arbiter modes and the parking master on 60x bus.
Step 9: Defines the arbitration priority of the various bus masters.
Step 10: Selects the functionality of many of the “system” pins such as the IRQs, parity pins and L2 cache pins.
Step 11: Enables (or disables) parity on 60x bus.
Step 12: Enables (or disables) parity on local bus.
Step 13: Initializes the refresh prescalar register.

Steps in Initialization (3 of 4)

Step	Action	Comment
14	Init the Memory Controller (10-1)	

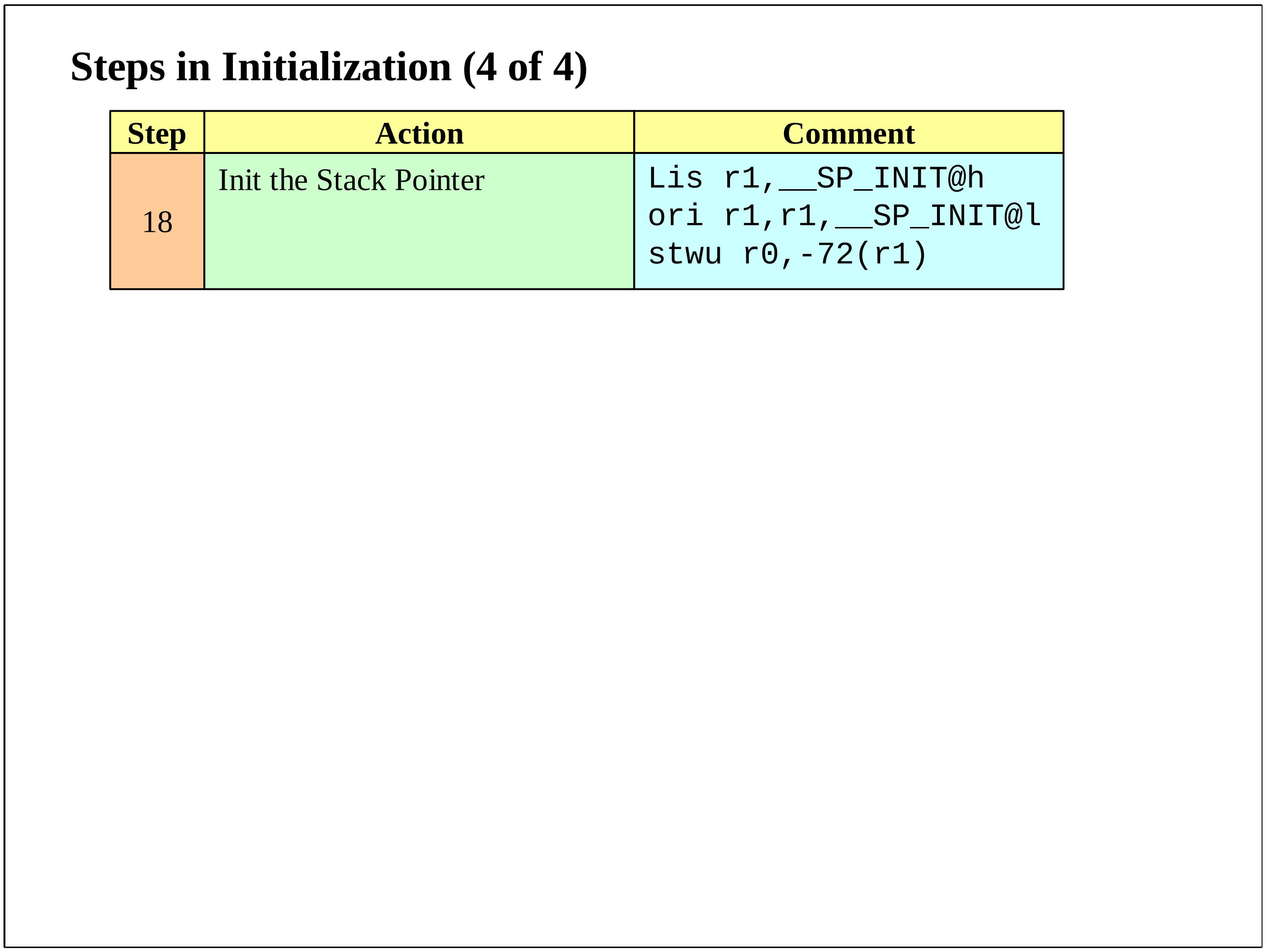
15	Relocate the EVT from ROM to RAM	`vloop:`
		`  addic. r3,r3,-1`
		`  lwz    r6,0(r5)`
		`  addi   r5,r5,4`
		`  stw    r6,0(r2)`
		`  addi   r2,r2,4`
		`  bne    vloop`

16	Init BAT Regs	

17	Init Caches	`mfspr r5,HID0`
		`ori   r5,r5,0x8800`
		`andi  r6,r5,0xF7FF`
		`mtspr HID0,r5`
		`isync`
		`sync`
		`mtspr HID0,r6`
		`isync`
		`sync`

Comments

Step 14:Initialize the memory contoller for the desired configuration.
Step 15: Relocate the exception vector table from ROM to RAM. If necessary, MSR[IP] should be changed.
Step 16: Initialize and enable the MMU. In this example, only Block Address Translation is used. If Page Translation is to be used, it should be initialized here also.
Step 17: Initialize and enable the caches.



Steps in Initialization (4 of 4)

Step	Action	Comment

18	Init the Stack Pointer	`Lis r1,__SP_INIT@h`
		`ori r1,r1,__SP_INIT@l`
		`stwu r0,-72(r1)`

Comments Step 18: Initialize the stack pointer, r1.