Integrator - Frequently Asked Questions v2.04

Table of contents

(o denotes new section, or section updated since last release of FAQ.)

1. General questions about the Integrator Family

o *What is Integrator?*

Integrator is a generic name for a range of development platforms and modules from ARM. A typical Integrator system comprises a platform board (motherboard) and a core module.

In general, the platform board provides the development system's peripheral interfaces and non-volatile program memory (ROM). The core module acts as a processor header, providing the ARM core and the majority of the system RAM. Core modules can also be used as a standalone software development system, with no ROM or peripherals.

Other modules can be added to expand the functionality of a basic system, for example, logic modules allow custom logic to be added to the system, and interface modules add extra peripheral interfaces.

The following Integrator development boards have been produced so far:

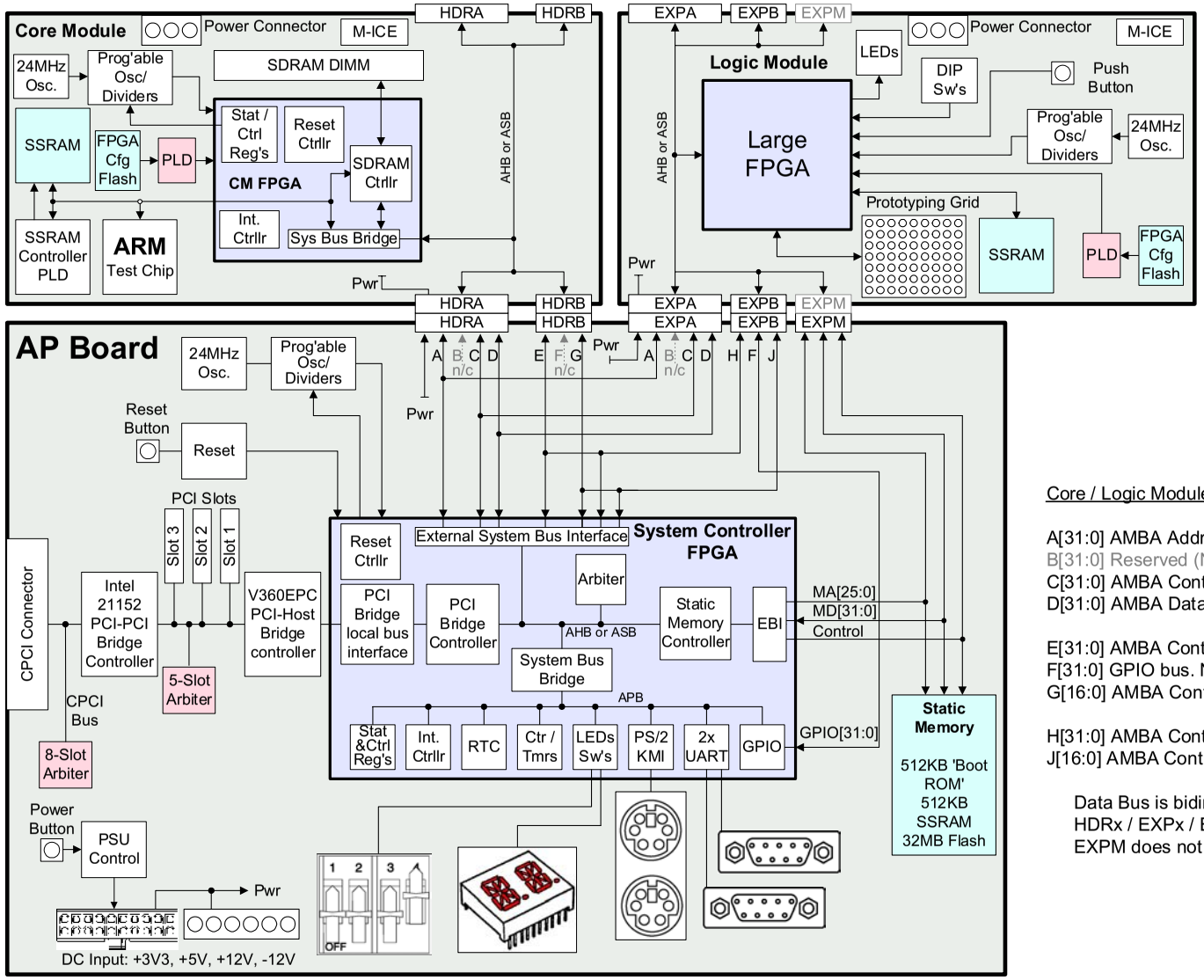
Board	Function	Status
Motherboards		
Integrator/CP	Compact Platform baseboard, compatible with the following core modules: CM920T, CM922T, CM922T-XA10, CM946E-S CM966E-S, CM1026EJ-S, CM1136JF-S	Current †
Integrator/AP	ASIC development platform	Discontinued
Integrator/AP (No CPCI)	ASIC development platform with CPCI connector omitted	Discontinued
Integrator/SP	Standard development platform	Discontinued
Core Modules		
Integrator/CM922T-ETM (70C)	Core module with ARM922T processor (with ETM9)	Current †
Integrator/CM922T-XA10	Core module with Altera Excalibur embedded processor PLD	Current †
Integrator/CM926EJ-S	Core module with ARM926EJ-S processor	Current †
Integrator/CM946E-S (66C,D)	Core module with ARM946E-S processor (with ETM9)	Current †
Integrator/CM1026EJ-S	Core module with ARM1026EJ-S processor	Current †
Integrator/CM1136JF-S	Core module with AM1136JF-S processor	Current †
Integrator/CM7TDMI	Core module with ARM7TDMI processor	Discontinued
Integrator/CM720T	Core module with ARM720T processor	Discontinued
Integrator/CM740T	Core module with ARM740T processor	Discontinued
Integrator/CM920T (47B)	Core module with ARM920T processor	Discontinued
Integrator/CM920T (70C)	Core module with ARM920T processor	Discontinued
Integrator/CM920T-ETM (70C)	Core module with ARM920T processor (with ETM9)	Discontinued
Integrator/CM940T (47B)	Core module with ARM940T processor	Discontinued
Integrator/CM966E-S (66C,D)	Core module with ARM966E-S processor (with ETM9)	Discontinued
Integrator/SCM926EJ-S	Core module with ARM926EJ-S processor in a Xilinx FPGA	Discontinued, LR*
Integrator/CM10200 (67C)	Core module with ARM10200 reference device	Discontinued, LR*
Integrator/CM10200E (98B)	Core module with ARM10200E reference device	Discontinued
Logic Modules		
Integrator/LM-XCV600E+	Logic Module for Xilinx XCV600E - XCV2000E FPGAs	Discontinued
Integrator/LM-EP20K600E+	Logic Module for Altera EP20K600E – EP20K1000E FPGAs	Discontinued
Integrator/LM-XCV400+	Logic Module for Xilinx XCV400 - XCV1000 FPGAs	Discontinued
Interface Modules		
Integrator/IM-LT1 (106C)	Logic Tile Interface Module – Allows logic tiles to be used on an AP or CP system, or standalone	Current
Integrator/IM-LT2 (106C)	As IM-LT1 board, but fully populated inc. FPGA. Allows 1 or more LTs to emulate a core module.	Current, LR*
Integrator/IM-LT3	Logic Tile Interface Module – Allows core tiles to be used standalone, or on a CP motherboard.	Current
Integrator/IM-AD1	Interface Module for automotive development	Discontinued
Integrator/IM-PD1	Interface Module for PDA development	Discontinued

Other modules		
Integrator/AM	Analyzer Module – Gives logic analyzer connectivity to the system bus when mounted on a core or logic module	Discontinued
Development Systems		
Integrator/PP1	Porting Platform built into ATX PC case. Comprises: Integrator/AP + CM920T + PCI Ethernet & VGA cards + ATX case	Discontinued
Integrator/PP2	Porting Platform built into custom chassis. Comprises: Integrator/AP + CM920T + LM-XCV2000E + IM-PD1 + LCD panel + Custom chassis	Discontinued
Integrator/SPP1	Symbian Porting Platform. As PP1, + Symbian OS in binary form.	Discontinued
Integrator/SPP2	Symbian Porting Platform. As PP2, + Symbian OS in binary form.	Discontinued

Notes:

- *LR = Limited Release – Was not offered for general sale
- † = Current, but will be phased out. Please refer to the 'RealView Hardware Platform Obsolescence' document
- Where different PCBs exist with the same core (and vice-versa), board part numbers are shown in parentheses. Please see FAQ entry 'How does the Integrator part numbering system work?' for more details
- Stock of core modules is subject to silicon availability. Please check with your supplier or local ARM sales office.

The illustration on the following page is a block diagram of a typical Integrator/AP based system, comprising a single core module, a logic module and an AP motherboard.



Core / Logic Module Connections

A[31:0] AMBA Address Bus
B[31:0] Reserved (N/C on CM and LM stacks)
C[31:0] AMBA Control Bus
D[31:0] AMBA Data Bus

E[31:0] AMBA Control Bus & system signals
F[31:0] GPIO bus. N/C on CM stack. Input by default.
G[16:0] AMBA Control Bus & system signals

H[31:0] AMBA Control Bus & system signals
J[16:0] AMBA Control Bus & system signals

Data Bus is bidirectional, even when AHB.
HDRx / EXPx / EXPM all have PSU connections
EXPM does not exist on later Logic Modules.

- ***What clock speed do the boards operate at?***

  This question is asked quite often, and due to the complexity of the Integrator system, the answer is not straightforward. The performance that you will be able to get from your Integrator system may depend on many factors:

  - Which ARM core module you are using
  - Which test chip is fitted to your core module
  - CPU and bus clock speeds
  - Whether cache memory is being used
  - Bus clocking mode (if applicable)
  - Which motherboard (if any) your system is based on, e.g. Integrator/AP, Integrator/CP
  - Which area of memory your application is running from
  - Which area of memory your data is stored in

  Due to the large scope of this subject, we have created a separate FAQ document covering these topics, entitled 'Integrator Performance FAQ', which can be downloaded from the ARM website. Please follow the 'Integrator' link on the following web page: http://www.arm.com/support/faq.

  The performance FAQ also contains a large amount of experimentally obtained data on memory access performance for the various memory regions in a variety of Integrator systems.

- ***How much on-board memory is there?***

  Core modules
  All core modules are fitted with some synchronous SRAM and a single SDRAM DIMM socket. The amount of SSRAM fitted varies between the different core module types. Older core module designs have 256KB of SSRAM; newer modules have 1MB or more of ZBT SSRAM fitted.

  Core modules also have some flash memory fitted, but with the exception of the Excalibur core module, this is used to store FPGA configurations, and is not available to store user code.

  The core module DIMM socket accommodates standard PC SDRAM modules in sizes of 16MB to 256MB. SDRAM DIMMs may or may not be supplied with core modules. There is an FAQ entry giving guidance on DIMM selection. Please see FAQ 'Which types of SDRAM DIMMs will work with my core module?'

  The Excalibur core module does not provide a controller for its SDRAM DIMM socket as standard. Instead, the embedded DDR SDRAM controller in the Excalibur PLD is used to drive the 128MB of on-board DDR SDRAM.

  Logic modules
  Logic modules are fitted with some SSRAM – 256KB on the LM-XCV400+ board, and 1MB of ZBT SSRAM on the LM-XCV600E+ and LM-EP20K600E+. There is also some flash memory on logic modules, which is used for storing configuration data for the FPGA.

  Motherboards
  AP and SP Motherboards are fitted with 512KB of SRAM, connected to the external bus interface, along with 32MB of flash memory, which is available to store user programs. This is known as the 'Application flash.' There is another flash device known as the 'boot ROM' on the Integrator AP motherboard. The boot ROM contains the system controller FPGA configuration data and the code that runs when the Integrator system is powered up, or reset. See related FAQ entry for more detail.

  The Integrator/CP baseboard provides only flash memory - 16MB as standard. This is used mainly for Application flash, but the top 256KB of the 16MB is reserved for the boot monitor. RAM on the Integrator/CP is provided by the core module, as described above.

  The analyzer module and the IM-PD1, IM-AD1 do not have any memory fitted, although the analyzer module has a 32 pin DIL socket, which can be used to connect an EPROM or EPROM emulator to the system. See the Integrator/AM User Guide for details.

The following table is a summary of the memory on each Integrator board:

Board	SSRAM	SDRAM	Flash device/function		
			FPGA Config	Boot ROM	Application
Integrator/AP	512KB	-	512KB (shared)		32MB
Integrator/SP	512KB	-	512KB (shared)		32MB
Integrator/CP	-	-	-	16MB (shared)	
Integrator/CM7TDMI	256KB	0-256MB	512KB	-	-
Integrator/CM720T	256KB	0-256MB	512KB	-	-
Integrator/CM740T	256KB	0-256MB	512KB	-	-
Integrator/CM9x0T (47B)	256KB	0-256MB	512KB	-	-
Integrator/CM920T (70C)	1MB (ZBT)	0-256MB	4MB	-	-
Integrator/CM920T-ETM (70C)	1MB (ZBT)	0-256MB	4MB	-	-
Integrator/CM922T-XA10	2.4MB	128MB	8MB	8MB	8MB
Integrator/CM926EJ-S	1MB (ZBT)	0-256MB	4MB	-	-
Integrator/CM946E-S	1MB (ZBT)	0-256MB	4MB	-	-
Integrator/CM966E-S	1MB (ZBT)	0-256MB	4MB	-	-
Integrator/CM10200	2MB (ZBT)	32-256MB	4MB	-	-
Integrator/CM10200E	2MB (ZBT)	64-256MB	8MB	-	-
Integrator/LM-XCV400+	256KB	-	2MB	-	-
Integrator/LM-XCV600E+	1MB (ZBT)	-	4MB	-	-
Integrator/LM-EP20K600E+	1MB (ZBT)	-	4MB	-	-
Integrator/IM-AD1	-	-	-	-	-
Integrator/IM-PD1	-	-	-	-	-
Integrator/AM	-	-	-	DIL socket	-

- ***What does the memory map look like?***
- ***Can I change the Integrator memory map?***

The exact memory map seen by the ARM core on an Integrator core module depends on several factors:

- Which core module is being used
- Is the core module being used stand-alone or on a motherboard?
- Are there any other core modules in the system?
- Are there any logic modules present in the system?

The Integrator system uses a distributed address decoding scheme, where each plug-in module has the responsibility of decoding its own address space. In addition, the address decoder on the motherboard (if present) produces default responses (bus error) for any modules which are not present in the system.

The memory map seen by the ARM core on a stand-alone core module is fairly limited. Depending on how the core module has been configured by the user, the core may see: On-chip memory, memory devices on the core module itself and registers implemented in the core module's FPGA. Typically, all these reside in the bottom 272MB of the ARM's memory map (addresses below 0x11000000).

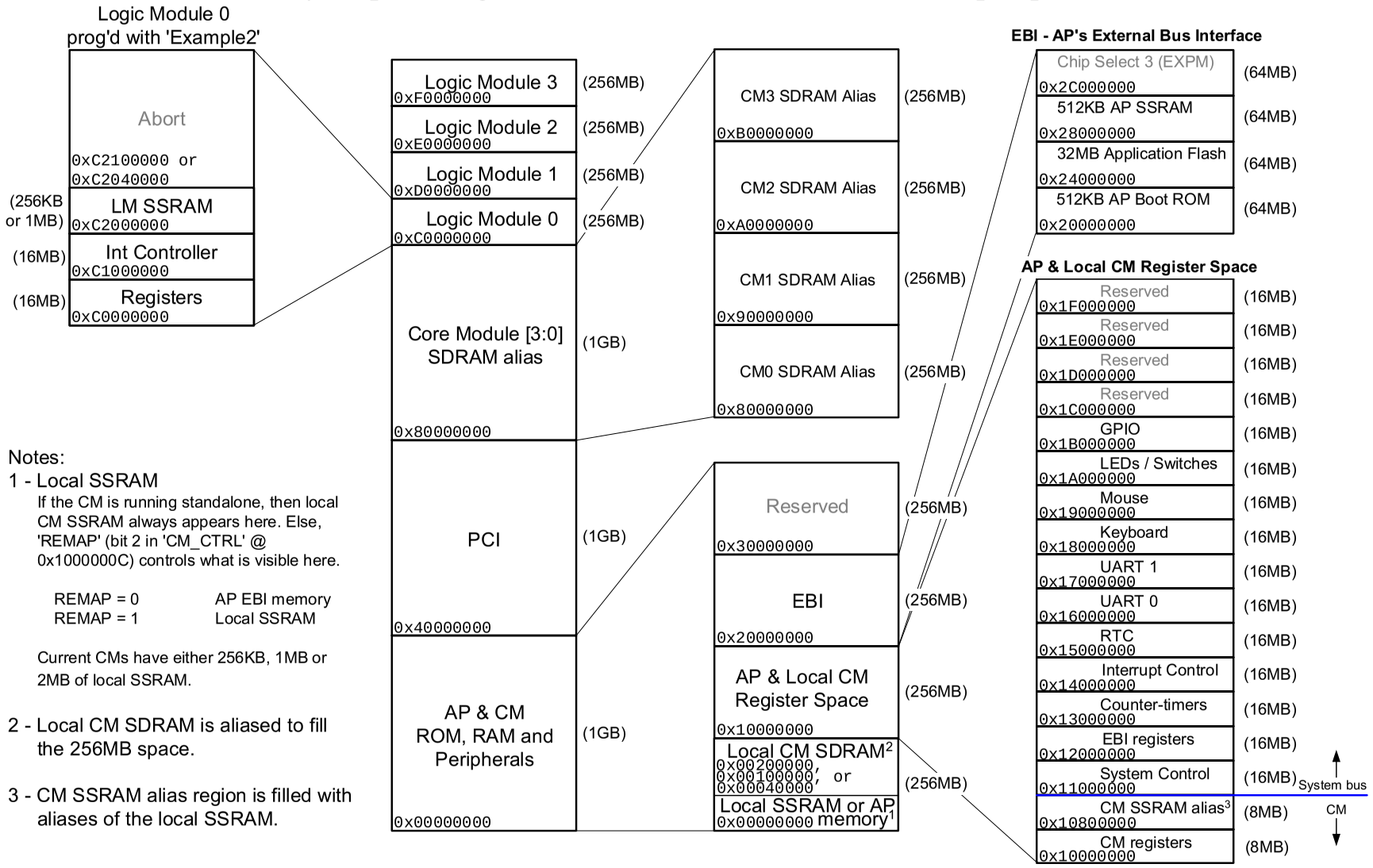
When a core module is fitted to an Integrator motherboard, the core module's nMBDET (motherboard detect) signal becomes grounded, and the core module's address decoder allows access to the upper 3.7GB of the 32bit address range, which will be decoded by other boards in the system.

The memory map of the Integrator system is fixed by the address decoder designs within the core modules and System Controller on the AP board. These cannot be easily changed, since ARM has not released the HDL source code for these devices. (See FAQ 'Can I get the HDL source code for the Integrator FPGAs / PLDs?').

The Integrator family of development boards have been designed to have largely compatible memory maps, although with each new board design, there are usually some subtle differences. For full details, the user guide for each board should be consulted, but the following three pages show top level memory map diagrams for several Integrator set-ups:

- A system comprising: Integrator/AP + core modules + logic modules
- An Integrator/CP system
- A system comprising: Integrator/AP + Integrator/CM922T-XA10 (Excalibur) core module.

Memory Map of Integrator/AP + CMs + LMs (From a CM's perspective)



Notes:

1 - Local SSRAM

If the CM is running standalone, then local CM SSRAM always appears here. Else, 'REMAP' (bit 2 in 'CM_CTRL' @ 0x1000000C) controls what is visible here.

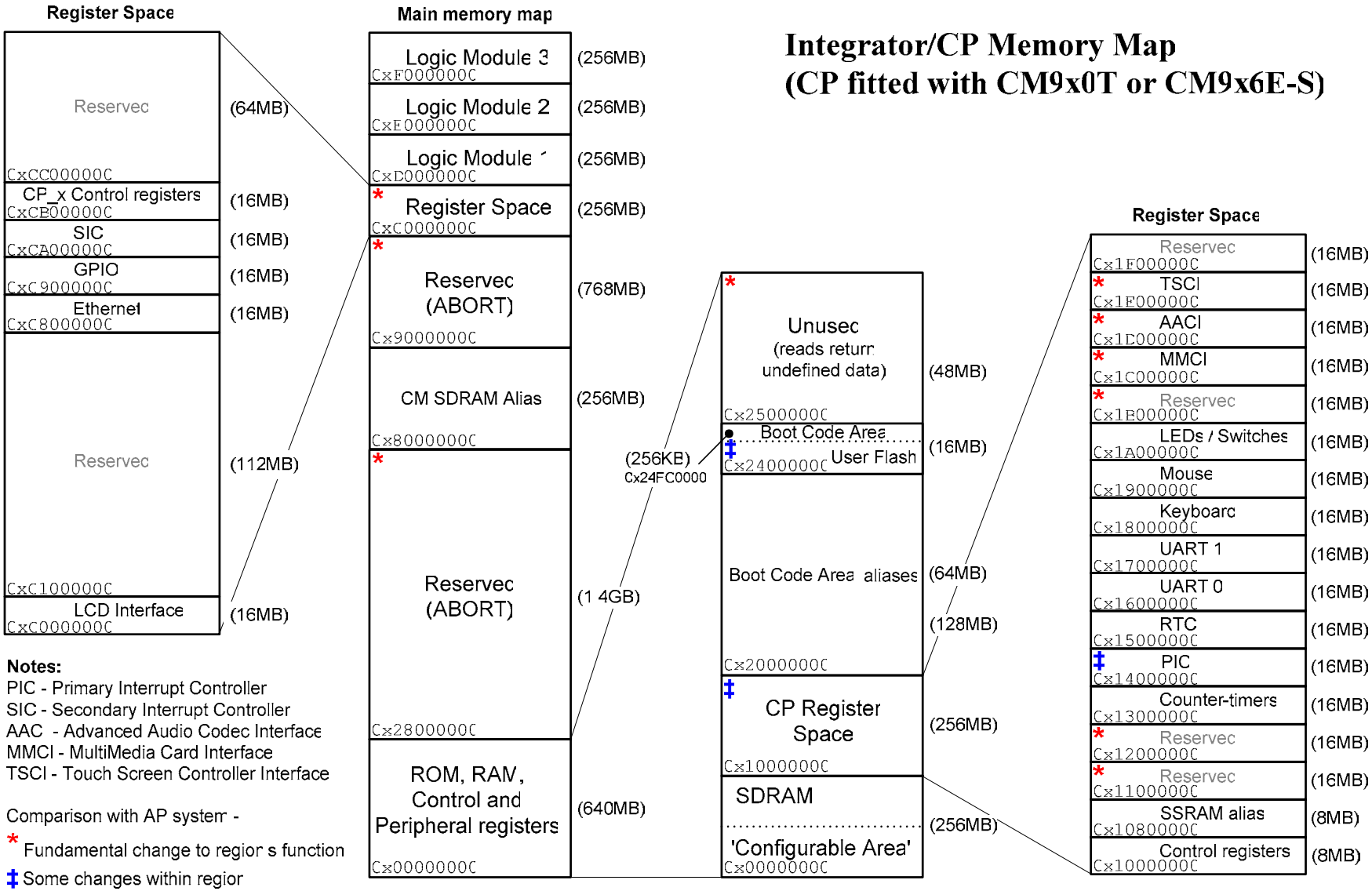
REMAP = 0	AP EBI memory
REMAP = 1	Local SSRAM

Current CMs have either 256KB, 1MB or 2MB of local SSRAM.

2 - Local CM SDRAM is aliased to fill the 256MB space.

3 - CM SSRAM alias region is filled with aliases of the local SSRAM.

Integrator/CP Memory Map
(CP fitted with CM9x0T or CM9x6E-S)



Notes:
PIC - Primary Interrupt Controller
SIC - Secondary Interrupt Controller
AAC - Advanced Audio Codec Interface
MMCI - MultiMedia Card Interface
TSCI - Touch Screen Controller Interface

Comparison with AP system -

* Fundamental change to region's function

‡ Some changes within region

Notes on boot sequence: At reset, the ARM 922T core inside the Excalibur device boots from CM EBI memory at 0x0. EBI0 flash is mapped here by default. The boot code is copied to and executed from RAM. The boot sequence is as follows:

1. Set up various embedded control registers. These control such things as memory map regions, bridge operation, clock speeds, etc.

2. Load one of four stored PLD designs into the Excalibur chip from the EBI0 flash device. The image is selected by either CFGSEL[1:0] or S1[2:1], depending on the state of S1[3]:

S1[n]					
[4]	[3]	[2]	[1]	Image #	Description
OFF	OFF	X	X	-	PLD image selected by CFGSEL[1:0]
OFF	ON	OFF	OFF	0b11	Image at 0x600000 (Integrator/CP image)
OFF	ON	OFF	ON	0b10	Image at 0x400000 (Integrator/AP image)
OFF	ON	ON	OFF	0b01	Image at 0x200000 (IM-PD1 image)
OFF	ON	ON	ON	0b00	Image at 0x000000 ('Basic Example' image)

If one of the factory programmed PLD images is loaded, the CM LEDs will indicate which image is in use:

PLD Image	LEDs [7:0]
CP	00110011
AP	11100111
IM-PD1	11110000
BE	01010101

3. Disable EBI0. The Excalibur 'stripe' region of the memory map now occupies the whole of the bottom 256MB of the memory map. The ARM core can now see on-board SDRAM at 0x0.

4. Read S2[8:7] and jump accordingly.

S2[n]		
[8]	[7]	Description
OFF	OFF	Remains in an infinite loop.
OFF	ON	Jumps direct to start of user flash (EBI2)
ON	X	Jumps direct to start of EBI1 flash

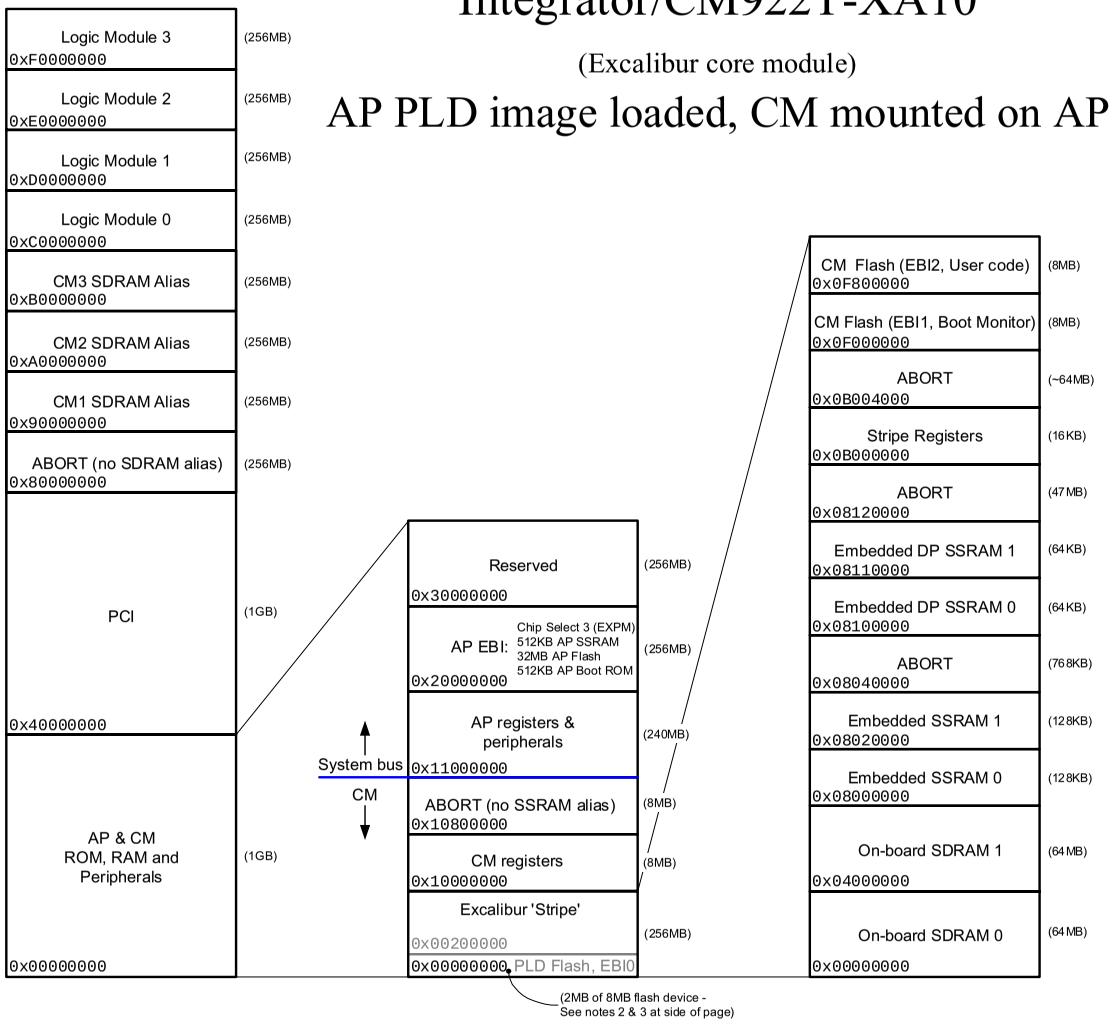
5. If control jumps to the core module boot monitor (EBI1 is factory programmed with this Boot monitor), then the 'boot switcher' element of the Boot monitor reads S2[2:1] and acts accordingly:

	S2[n]		
Mounted on AP	[2]	[1]	Description
YES	OFF	X	Jump to 0x20000000. On the AP board, this is the boot monitor.
YES	ON	OFF	Set clocks from SIB and Jump to selected user image in EBI2
YES	ON	ON	Run core module boot monitor in EBI1
NO	X	OFF	Set clocks from SIB and Jump to selected user image in EBI2
NO	X	ON	Run core module boot monitor in EBI1

Memory map for Integrator/CM922T-XA10

(Excalibur core module)

AP PLD image loaded, CM mounted on AP



- ***Can my Integrator system work with ASB and AHB bus standards?***

  Integrator/AP (or SP) system
  Older Integrator/AP and SP boards were programmed to support only the ASB bus standard. Support for ASB and AHB on the system bus is now available. See FAQ 'How do I re-program the FPGAs to the latest revision?' for more details.

  Older core modules (as supplied) only supported the ASB bus, but can be re-programmed to support both bus standards. Core modules where the core has a native AHB bus interface (CM922T-XA10, CM926EJ-S, CM946E-S, CM966E-S, CM10200, CM10200E) only support an AHB System bus.

  Integrator/CP System
  The Integrator/CP platform only supports the AHB-Lite standard as its system bus.

- ***How do I re-program the FPGAs to the latest revision?***

  The FPGAs, PLDs and associated configuration flash devices on Integrator boards can be upgraded in the field to the latest revisions.

  Tools required are a Multi-ICE unit, a utility called 'Progcards.exe', and a set of board files. Refer to http://www.arm.com/support/downloads/integrator.html to download the latest programming files, including the Progcards utility.

  It is also possible to revert to a previous configuration, as the old versions of the configuration files are provided. Full instructions for carrying out the reprogramming procedure are also included.

- ***Why do I get 'ERROR: Multi-ICE error (8)' when using Progcards?***

  This error was sometimes reported by versions of Progcards earlier than v2.11, when used with some PCs that have a 'slow' high speed frequency counter. This caused inaccurate programming delays, resulting in the above Multi-ICE timeout error.

  The solution is to use a later version of Progcards. We would recommend that the latest version of Progcards is always used. This can be downloaded from the ARM website. Please refer to the FAQ entry 'How do I re-program the FPGAs to the latest revision?' for further details.

- ***How does the Integrator system select which system bus type to use?***
- ***What do the CFGSEL[1:0] pins do?***
- ***How do I set the state of the CFGSEL[1:0] pins?***

  The majority of the logic that provides the AMBA system bus interfaces and peripherals (if any) on each Integrator board is implemented in FPGAs. Each FPGA's configuration is loaded at power-up from flash memory. Integrator core and logic modules have the capacity to store multiple FPGA configurations in flash.

  A pair of signals called CFGSEL[1:0] are used to indicate which FPGA configuration to load at power-on time. The motherboard sets the state of these signals, which are routed to each plugged-in module. Each module reads these signals to determine which FPGA configuration to load, according to the following table:

CFGSEL[1:0]	Configuration	Description
0 0	ASB	AP or SP platform with ASB system bus
0 1	Reserved	
1 0	AHB	AP or SP platform with AHB system bus
1 1	CP-AHB-Lite	Integrator/CP only. AHB-Lite system bus

  When the platform motherboard is programmed, part of the programming procedure changes the state of the CFGSEL signals, hence you do not have to manually configure the state of these lines.

When an Integrator motherboard is present in the system, you should not attempt to drive the CFGSEL lines from elsewhere in the system, as permanent damage to motherboard components may result.

Each plug-in module has its CFGSEL[1:0] lines biased to sensible states with resistors, so that a valid FPGA configuration is loaded when the board is operated 'stand-alone.' In this situation, for core modules which have been programmed for use on an AP or SP motherboard, the programmed system bus type is largely irrelevant, as all bus transactions will occur only on the core module's local bus.

Core modules always use the CFGSEL signals to select an appropriate FPGA image, but logic modules have the option of either using CFGSEL, or using a pair of DIP switches to select the FPGA image. This option is selected by another DIP switch on the logic module. Please see your logic module user guide for full details.

For the Integrator AP and SP platform boards, FPGA configurations have been developed to implement AHB and ASB system buses. The following core module FPGA configurations have been developed:

Core Module		Configurations available		
		ASB on AP or SP	AHB on AP or SP	CP
CM7TDMI	HBI-0056	●	●	
CM720T	HPI-0050	●	●	
CM740T	HBI-0058	●	●	
CM920T	HBI-0047	●	●	
CM920T	HBI-0070	●	●	●
CM920T-ETM	HBI-0070	●	●	●
CM922T-XA10	HBI-0100		●	●
CM926EJ-S	HBI-0087		●	●
CM940T	HBI-0047	●	●	
CM946E-S	HBI-0066		●	●
CM966E-S	HBI-0066		●	●
CM10200	HBI-0067		●	
CM10200E	HBI-0098		●	

- ***What do the 'FPGA OK' or 'DONE' LEDs indicate?***

Most Integrator boards have a green LED labelled 'FPGA OK' or 'DONE'. The purpose of these LEDs is to indicate that the system's FPGAs have successfully loaded valid configurations from flash memory. The system is held in reset until all programmable logic is configured. (See FAQ 'How does the reset circuitry work in an Integrator system' for more information).

The FPGA LED on most boards monitors the board's 'LOCAL DONE' signal, which indicates when the board's own FPGA has been configured. On some boards however, this LED monitors the 'GLOBAL DONE' signal, an open collector signal which is routed through each board in the system. Each board's 'LOCAL DONE' signal drives the 'GLOBAL DONE' signal, which in turn drives the system's reset circuitry.

The following table shows the function of the 'FPGA OK' or 'DONE' LED for each Integrator board:

Board		LED label	LED indicates
Integrator/AP	HBI-0048	FPGA OK	LOCAL DONE
Integrator/SP	HBI-0049	D6	LOCAL DONE
Integrator/CP	HBI-0086	N/A	N/A
CM7TDMI	HBI-0056	FPGA OK	LOCAL DONE
CM720T	HPI-0050	FPGA OK	GLOBAL DONE
CM740T	HBI-0058	FPGA OK	GLOBAL DONE
CM920T	HBI-0047	FPGA OK	GLOBAL DONE
CM920T	HBI-0070	DONE	LOCAL DONE
CM920T-ETM	HBI-0070	DONE	LOCAL DONE
CM922T-XA10	HBI-0100	DONE	LOCAL DONE
CM926EJ-S	HBI-0087	DONE	LOCAL DONE
CM940T	HBI-0047	FPGA OK	GLOBAL DONE
CM946E-S	HBI-0066	DONE	LOCAL DONE
CM966E-S	HBI-0066	DONE	LOCAL DONE
CM10200	HBI-0067	DONE	LOCAL DONE
CM10200E	HBI-0098	DONE	LOCAL DONE
Integrator/LM-XCV400+	HBI-0059	FPGA OK	LOCAL DONE
Integrator/LM-XCV600E+	HBI-0072	FPGAOK	LOCAL DONE
Integrator/LM-EP20K600E+	HBI-0073	FPGAOK	LOCAL DONE
Integrator/IM-LT1	HBI-0106	N/A	N/A
Integrator/IM-LT2	HBI-0106	DONE	LOCAL DONE
Integrator/LTXC2V4000+	HBI-0102	FPGA OK	LOCAL DONE

- ***Can I stack LMs and CMs together without a motherboard?***

  It is possible to stack logic modules and core modules together without a motherboard, but there are several issues which must be considered. These are covered in detail in Application Note 101, which can be downloaded from the documentation area on the ARM website. In summary, the issues are:

  - Core modules cannot be stacked on top of logic modules due to mechanical interference between adjacent board components. Possible board combinations are described in the FAQ: 'Some Integrator modules will not fit on top of others'.
  - One of the modules must provide a minimum sub-set of the functionality that would be provided by a motherboard, for example: JTAG signal loopback, some of the AMBA system bus infrastructure
  - Some modules have a 'stacking mode' PCB link to facilitate simpler stacking, whereas other modules (typically older designs) do not
  - Some module combinations result in excessive JTAG signal degradation, which prevents proper operation.

- ***Some Integrator modules will not fit on top of others***

  As described in the FAQ 'Can I stack LMs and CMs together without a motherboard?', it is possible to construct a development system by stacking together only core modules and logic modules, provided that certain conditions are met. One of these issues is that some modules can not be plugged together due to mechanical interference between some of the components on each board.

  When stacking boards that are in this way mechanically incompatible, it is sometimes possible to make all the connectors mate successfully by using increased force to push the boards together. This results in the PCBs being distorted, and stress being placed on components, solder joints and PCB traces. This is likely to cause failure, and should not be done.

  The following matrix shows:

  - Which Integrator modules fit together properly
  - Which modules appear to fit together, but do not without causing damage

- Which modules do not fit together at all
- If there are any other problems associated with a particular module combination.

Module on BOTTOM		Module on TOP												
Modules / Part numbers		CM7TDMI HBI-0056	CM720T HBI-0056	CM740T HBI-0058	CM920T HBI-0070	CM922 -XA10	CM9x0T HBI-0047	CM9x6E-S HBI-0066	CM10200 HBI-0067	CM10200E HBI-0098	LM-XCV 400+	LM-XCV 600E+	LM-EP20K 600E+	Integrator /AM
CM7TDMI	HBI-0056	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM720T	HBI-0050	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM740T	HBI-0058	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM920T[-ETM]	HBI-0070	●	●	●	●	!$_1$	●	●$_2$	●	●	●$_3$	●	●	●
CM922T-XA10	HBI-0086	!$_1$	!	!	!$_1$	●$_2$	!	!	!$_1$	!$_1$	!	●	●	!
CM9x0T	HBI-0047	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM9x6E-S	HBI-0066	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM10200	HBI-0067	●	●	●	●	●	●	●$_2$	●	●	●$_3$	●	●	●
CM10200E	HBI-0098	●	●	●	●	●	●	●	●	●	!	●	●	!
LM-XCV400+	HBI-0059	!$_1$	●$_3$	●$_3$	!$_1$	!$_1$	!	!	!$_1$	!	●	●$_{3,4}$	●$_{3,4}$	●
LM-XCV600E+	HBI-0072	!$_1$	!	!	!	●	!$_1$	!	!$_1$	!$_1$	●$_{3,4}$	●$_5$	●$_5$	●
LM-EP20K600E+	HBI-0073	!$_1$	!	!	!	●	!$_1$	!	!$_1$	!$_1$	●$_{3,4}$	●$_5$	●	●
Integrator/AM	HBI-0052	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$	n/a$_6$

Notes:

● - Boards will fit together with no mechanical interference

! - Boards will not fit together due to mechanical interference

1. Boards may appear to fit together if forced. This should not be done, since the PCBs will be distorted, eventually resulting in failure.
2. Potential problem: If ground loop near power connector on bottom surface of upper board was placed too close to the edge of the board during manufacture, the loop will foul the power connector on the board beneath.
3. Potential problem: If screw terminals on lower board are in a raised position, this could short 3V3 to 0V or 5V to 3V3.
4. LM-XCV400+ has 120-pin 'EXPM' connectors top & bottom, whereas standard LM-*600E+ have 200-pin 'EXPIM' connector (on top only). The two types do not mate.
5. LM-*600E+ have no EXPIM connector fitted in the J6 position (lower side) as standard. This means that if two such logic modules are stacked together, a number of FPGA I/Os routed to the EXPIM connector positions will not connect between boards.
6. The Analyzer Module is designed to fit on top of a stack of modules, so that the system bus signals can be monitored on a logic analyzer. For this reason, there are no module connectors on top of the Analyzer Module.

- ***Integrator AP does not appear to support FIQ interrupts***
- ***How are interrupts routed in an Integrator system?***

There is some confusion about the way in which the interrupt signals are routed in the Integrator system. The naming conventions used for interrupt signals on the Integrator circuit schematics are one source of confusion. Another is that there are several interrupt controllers in a typical Integrator system, and it is necessary to refer to the documentation for each board to understand the interaction between them. This FAQ attempts to clear up any such confusion.

Integrator/AP System

Core module interrupt controller

The ARM processor test chip on each core module has two interrupt inputs, nFIQ and nIRQ. Each core module has two mini interrupt controllers, one of which drives the ARM's FIQ input, and the other one the IRQ input.

These CM 'mini' interrupt controllers accept two inputs from the DCC (Debug Comms Channel) and one software generated interrupt, which is triggered by writing a specific value to a core module register. Please see your core module documentation for full details of operation.

Main interrupt controller

The system's main interrupt controller is on the AP motherboard. This controller is made up of eight identical interrupt controllers; two each for the four possible core modules in the system.

Two interrupt controllers are required per core module, since one controller generates IRQ interrupts, and the other generates FIQs. The outputs from the AP interrupt controller(s) are ANDed with the outputs from the core module interrupt controllers. This is either a wire-AND, or a true logical AND performed in the CM FPGA, depending on the core module type.

The AP controllers all accept the same set of 22 interrupt sources from the peripherals on the AP board, and from any attached logic modules. This means that any interrupt source can be routed to the nFIQ or nIRQ inputs of any attached core module, simply by enabling the required source in the correct AP controller. For interrupt controller register details, please refer to the Integrator/AP User Guide.

Traditionally, each logic module generates only one interrupt signal on its 'nIRQ0' output. The logic module interrupt signals are rotated as they pass down the stack, so each logic module's interrupt output appears on a different interrupt input pin on the AP board.

On the AP board, the four logic module interrupt inputs are labelled nIRQSRC[3:0] (IRQ Source), but the corresponding output pins on a logic module's connectors are simply labelled nIRQ[3:0] – the same as the corresponding pins on a core module, and a potential source of confusion. This naming convention is used because logic modules can also be mounted on the core module stack, if they are being used to emulate core modules.

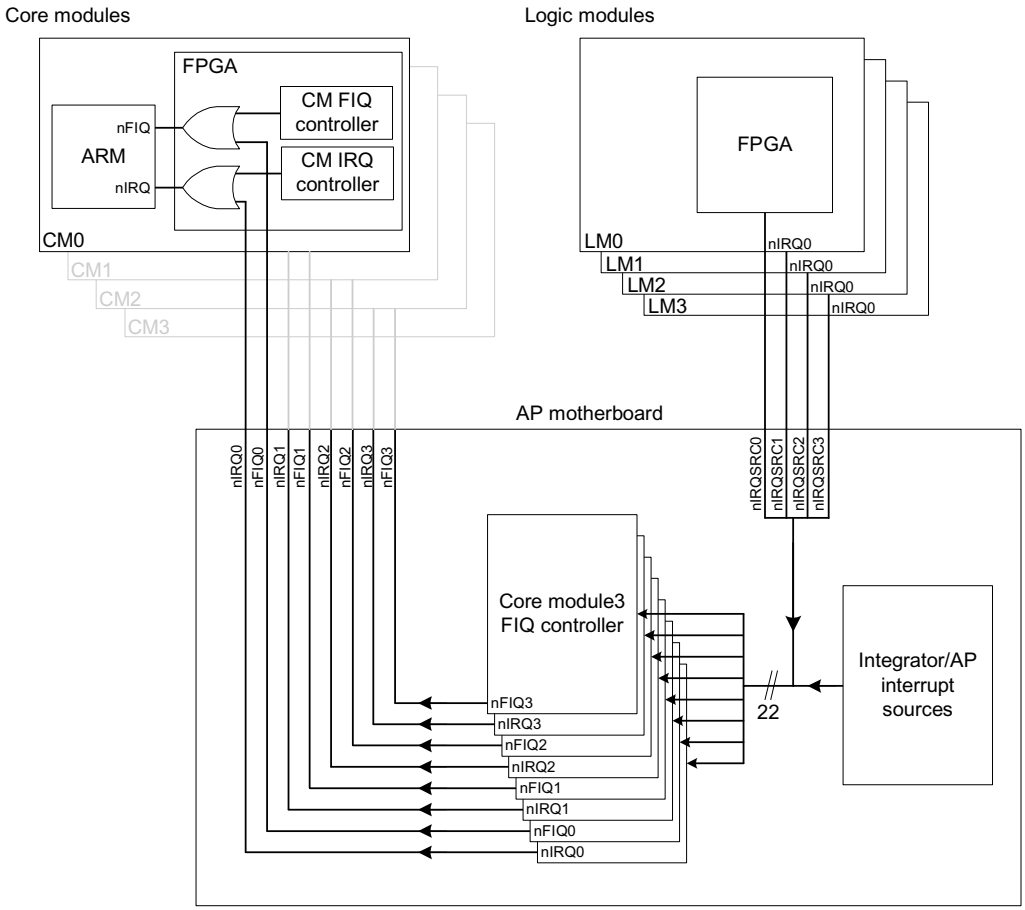
For the same reason, each logic module has a set of pins labelled nFIQ[3:0] which are not used when the logic module is placed on the logic module stack. The nFIQ pins on the AP board's logic module site are NC (No Connect).

Logic module interrupt controller
If a logic module design generates one or more interrupts, then some form of interrupt controller must be implemented in the logic module FPGA. Firstly, an interrupted ARM core must be able to write to the logic module interrupt controller to be able to clear the interrupt source. Second, if more than one interrupt source is required inside a single logic module, then depending on which type of logic module is used, the ARM core may need to interrogate the logic module to determine the source of the interrupt:

LM-XCV400+ module
This module only connects the nIRQ0 signal on its EXPB connectors to its FPGA. Thus, all interrupt sources within the FPGA must assert the nIRQ0 interrupt request line from the logic module. The interrupted ARM processor must then interrogate the logic module to determine which of its internal blocks caused the interrupt.

LM-XCV600E+ or LM-EP20K600E+ modules
The above method can be used, or since these modules connect all four of their nIRQ[3:0] lines to the LM FPGA, it is possible for a module to assert more than one input to the AP interrupt controller. Care must be taken when designing such a development system to ensure that multiple modules do not assert the same interrupt source line.

The following diagram shows interrupt routing in an Integrator development system comprising an AP motherboard with one core module and four logic modules. Connections to other core module positions are shown in gray.



Interrupt routing in a typical Integrator development system

Integrator/CP system

The Integrator/CP system interrupt routing philosophy is different in several ways. Since there can only be one core module in the system, only two main interrupt controllers are needed instead of the eight needed on the AP. However, the interrupt routing scheme is complicated slightly, since it has an extra layer of interrupt controller, to incorporate the interrupt sources from the extra peripheral devices offered on this board. This is fully documented in the Integrator/CP User Guide.

- ***Spurious interrupts generated in an Integrator system***

In some circumstances, it is possible to see an interrupt occur in an Integrator system when all interrupt sources have apparently been disabled.

The asynchronous bridge designs used in some Integrator core modules contain a FIFO between the core module local bus and the Integrator system bus. (See FAQ entry 'Lack of system bus FIFO in some core modules'). Since the main interrupt controller is on the AP motherboard, a problem may arise if there is a situation where a program clears or disables an active interrupt source, then immediately re-enables interrupts on the ARM core. The problem may be seen whether IRQ or FIQ interrupts are used.

Imagine the following sequence of events:

1. The system is running with IRQ enabled and one or more Integrator Interrupt sources are enabled which can generate an IRQ.
2. An IRQ interrupt source is triggered.
3. ARM IRQ mode is entered and IRQ is automatically disabled by the ARM core.

4. The IRQ handler clears or disables the interrupt source by writing to the AP interrupt controller. **Note** - This write transaction will not reach its destination immediately, since it will be delayed, passing through the core module FIFO.
5. IRQ is re-enabled either by returning from the IRQ handler, or within the handler itself if re-entrant interrupts are being used.
6. An IRQ Interrupt occurs immediately, since the write transaction to disable/clear the interrupt source has not yet reached the AP interrupt controller.
7. Inspection of the AP interrupt controller registers from a debugger, or by code running on the ARM shows that there are no pending interrupts, since by now, the write transaction has passed through FIFO and the interrupt source has been cleared.

The solution is to add code to perform a read from any system bus address in between steps 4 and 5 above. This will cause the FIFO to be flushed, and the interrupt source cleared before IRQ interrupts are re-enabled.

- ***How do I prototype peripherals?***

The system bus is used to connect processor and expansion header cards to the system controller FPGA. Peripherals can be prototyped on an expansion card (logic module or logic tile) and connected to the system bus using the header connectors.

There is no external APB bus. To prototype APB peripherals you will need to synthesize an ASB-APB or AHB-APB bridge in a Logic Module FPGA (see list below).

You will need an address decoder to generate select lines (DSEL for ASB, HSEL for AHB) for your peripherals and/or APB bridge. You may also need an interrupt controller if you have many interrupt sources.

Example HDL source code (for slaves only) is provided on the Logic Module/Tile CD. The example comprises:

- AHB SSRAM controller
- AHB-APB bridge
- APB register peripheral (LEDs and switches)
- APB interrupt controller
- Address decoder

Example VHDL code for connecting a bus master to the Integrator system bus is given in Appendix C of the Integrator/AP user guide, ARM DUI 0098B.

- ***Can I build a multi-processor system?***
- ***Can I add a DSP?***

Integrator/AP

Yes, Integrator/AP platforms are designed to support up to 5 system bus masters, which can be core modules, logic modules from ARM or from third parties. A multi-processor system is constructed by stacking core modules on top of each other. There is special routing between connectors on the top and bottom of each module to ensure that clocks, arbiter and JTAG signals work properly in a multi-processor system.

Note: To ensure reliable debugging, it may be necessary to reduce the JTAG TCK to 1MHz or less when debugging multiple core modules. This is achieved by configuring the Multi-ICE server.The bus interface specification and interconnect details are open information.

If a DSP which exists as an ASIC is to be added, then some logic may be required to interface it to the system bus. It may also be appropriate to add some closely coupled memory to the DSP to increase performance. If the DSP design is soft IP, then it may be appropriate to synthesize the logic into an FPGA. An ARM logic module could be used for this purpose.

Integrator/CP
This platform incorporates an AHB-Lite system bus, on which there can only be one bus master, which is the ARM test chip on the core module. It is not possible to stack additional core modules on this platform. It is possible to add up to three logic modules for peripheral IP development, but it is not possible to add extra system bus masters in logic modules.

One way to connect another processor to the CP system bus would be to implement a shared memory interface in a logic module. Both processors could then exchange information via this shared memory.

- ***Can I connect to the co-processor interface on the core module's ARM test chip?***

Generally speaking, this is not possible, as core modules were not designed to provide this functionality. In practise, access to these signals depends on which core module you have. The co-processor interfaces on the test chips are not brought out directly to connectors on any of the core modules. On some core modules, the co-processor interface signals are connected to the core module FPGA, but ARM has not released the HDL source for these FPGAs, so modifying the FPGA design would be time-consuming, and is not recommended, or supported.

- ***What memory space is reserved for logic modules/logic tiles?***
- ***Use of nPPRES[3:0] and nEPRES[3:0] signals***

Integrator/AP
There is one gigabyte of address space from `0xC0000000-0xFFFFFFFF` allocated for expansion logic. It is possible to locate peripherals anywhere within this address space.

The gigabyte of address space is sub-divided into four 256MB spaces as it is possible to add up to four expansion cards (logic modules). There is no reason why a single card could not claim all the address space. If an expansion card is fitted, it must assert the required expansion present lines (nEPRES) and this signals to the system decoder that expansion logic will decode the address space. If no expansion cards are fitted the system decoder will respond to accesses in the expansion space with a bus error (abort). See text below for an explanation of these signals.

Integrator/CP
The same is true for the Integrator/CP, except that there is only space for three logic modules. The allocated range is `0xD0000000 – 0xFFFFFFFF`. The 256MB space at `0xC0000000`, which was allocated to LM0 in an Integrator/AP system has been used for peripheral space on the CP. See the Integrator/CP User Guide for full details.

On the Integrator/AP motherboard, the signals nPPRES[3:0] are used on the HDRA/HDRB stack whereas nEPRES[3:0] are used on the EXPA/EXPB stack. These signals occupy the same connector pin numbers on both stacks. The Integrator/CP only has one location for stacking plug-in modules, HDRA/HDRB.

Each logic module must tie its own nEPRES0 signal low and leave nEPRES[3:1] open circuit. These signals rotate as they move up/down the stack such that each Logic Module's nEPRES0 signal routes to a different nEPRES[3:0] signal on the motherboard. The address decoder within the system controller FPGA looks to see which of its nEPRES[3:0] lines are low and can thus determine which logic modules are fitted and turn off the default response for the memory ranges allocated to logic modules that are not present.

Most logic modules can claim more address space by grounding more than one nEPRES line. The original logic module (the LM-XCV400+) did not have this ability, since the nEPRES0 line on this module is hard-wired to 0V, and the other nEPRES lines are no connect.

The rotation is implemented as follows:

Signal	Bottom connector	Top connector
nEPRES0	pin 33	pin 37
nEPRES1	pin 35	pin 33
nEPRES2	pin 36	pin 35
nEPRES3	pin 37	pin 36

- ***How do I use Multi-ICE with Integrator?***

  Each module has a Multi-ICE connector for stand-alone operation. When multiple modules are stacked on a platform, Multi-ICE connects to the top module in the stack. The JTAG signals are routed down through each connector in the stack so that all devices appear in the scan chain. The Multi-ICE server allows the debugger to be connected to any processor in the scan chain.

  An Integrator/AP motherboard can accommodate two stacks of modules, and the JTAG scan chains for each stack are completely separate. Hence, Multi-ICE must be connected to the top of the logic module stack to program logic modules, then removed and connected to the top of the core module stack to download and debug ARM code.

- ***Multi-ICE appears to load a program into RAM, but it will not run. Why?***

  When using Multi-ICE and a debugger, it is possible that the boot code may not have finished running before Multi-ICE stops the processor. Also, it is possible to configure the board so that the boot monitor does not run at power-up. (See entry 'What is the power-up sequence?') The boot code is responsible for setting up the SDRAM registers and changing the memory map so that RAM is at address `0x00000000`; a process called remapping.

  If your board is configured so that flash memory is at `0x00000000`, and you try to use Multi-ICE to download code directly to flash, it will not work. In this situation, it may appear from the debugger window contents that your code has been downloaded, but as soon as execution is attempted, errors will occur. In this case, it is possible to manually remap memory before downloading programs. To do this type the following at the command prompt, or add it to your startup script:

  ```
  0x1000000C = 0x04          (ADW debugger)

  smem 0x1000000C 0x04 32    (AXD debugger)
  ```

  This will ensure that there is RAM at address `0x00000000`, but the SDRAM (if fitted) may still not be configured. The core module and CP User Guides explain how to configure the SDRAM, or alternatively you can run the boot monitor. To run the boot monitor manually, set the PC (R15) to the start of the boot ROM and then run. To do this type the following at the command prompt,

  ```
  pc = 0x20000000          (ADW debugger)
  go

  let pc 0x20000000        (AXD debugger)
  go
  ```

- ***Why does my C code fail to run on Integrator when using Multi-ICE?***

  If you are not using SDRAM, then you should modify the debugger internal variable $top_of_memory to the upper limit of the SSRAM, usually 256KB on older core module designs. (See 'How much memory is there?'). Any programs that are linked with the ARM C library will use this variable to set where the stack is placed. If this variable is not set then your stack may be located in the SDRAM region, which may be missing altogether, or unreliable if not correctly initialised. To set the top of stack position to be 256KB, type the following at the debugger command prompt:

  ```
  $top_of_memory = 0x40000              (ADW debugger)
  ```

```
let $top_of_memory 0x40000          (AXD debugger)
```

This instruction can be added to a script file if desired, or the change can be made from the 'Debugger Internals' window within the GUI.

- ***Does Integrator work big-endian?***

Endianess is a system issue; it affects not only the hardware, but also the application software and boot monitor (firmware).

Integrator/AP (or SP)
The Integrator/AP system as a whole is little-endian only. Some core modules will work in big-endian mode when operating 'stand-alone'. Refer to the core module User Guides for more information.

As Integrator is fully programmable it would be possible to construct a big-endian Integrator system, but ARM has not released an FPGA configuration to support this.

Integrator/CP
The supplied system logic configuration on the Integrator/CP will allow operation in big or little-endian modes. However, several aspects of the system do not support big-endian operation. These are described here:

The Ethernet controller is a little-endian device. If byte-swapping is performed in software, big-endian operation can be achieved.

The SDRAM controller in the first release of the CP system logic is set to little-endian configuration. In subsequent releases, the SDRAM controller endian configuration automatically changes when the endian setting is changed on the ARM core.

The supplied boot monitor software was built for a little-endian system.

- ***What is the power-up sequence***
- ***What do the DIL switches do?***

Integrator/AP (or SP)
The following sequence is true when 1 or more core modules are fitted to an Integrator/AP (or SP) motherboard.

While the boards power up, the system is held in reset. When the 3V3 supply is stable, the system controller FPGA starts to configure in master-parallel-down mode (refer to the Xilinx databook for further information). When configured, the FPGA OK LED turns on, and the FPGA holds the system in reset for a further 340us to allow the PLL clock generators to settle. Reset is de-asserted and the ARM processor(s) on the core module(s) will start to fetch instructions from address `0x00000000`. The memory device that is visible at `0x00000000` depends on the state of SW1-1. This selection is made by hardware:

SW1-1 = ON Boot ROM at `0x00000000`
SW1-1 = OFF Application flash at `0x00000000`. This is where user code can be stored.

Supposing a core starts executing code from the boot ROM, the following operations will be performed:

- Execution jumps to the 2nd word location at the boot ROM's permanent address (`0x20000004`)
- Sets up SDRAM (if any is fitted to the core module)
- Remaps memory so that there is RAM at address `0x00000000`
- Turns on the green LED (labelled MISC) on the core module
- Selects 'Asynchronous Mode' on cached (ARMxx0T) cores, but does not enable the caches.

- Checks the DIL switches and continues execution depending on the state of SW1-4. This is known as the 'boot switcher':

SW1-4 = ON Continues to run the boot monitor:

- Outputs a text banner on serial port A
- Loops waiting for communication from a terminal on Serial port A

SW1-4 = OFF Execution jumps to selected image in Application flash. Flash images can be programmed using the boot monitor itself, or the ARM Flash Utility (AFU). Selection of boot image number can be made from the boot monitor.

After this, the DIL switches can be accessed by user code, for whatever purpose is required.

If a core module is run stand-alone, the sequence is different, as there is no motherboard present. In this case, core module SSRAM is always present at `0x00000000`. On power-up, this memory will be filled with random instructions, which will be executed until Multi-ICE takes control of the core. The presence of a motherboard is detected by the nMBDET signal. Any accesses to resources assigned to or connected via the motherboard will result in a bus error, which will cause the ARM core to take an abort exception.

Integrator/CP
The startup sequence is exactly the same, but there are some hardware implementation differences:

There is no separate boot ROM device on the Integrator/CP; the top 256KB of the 16MB Application flash is reserved for the boot monitor. This 'boot code area' is aliased to appear at `0x20000000` in the system memory map; the address at which the boot ROM resides on an AP system. See 'Integrator/CP boot monitor requires a SDRAM DIMM' for related information.

- ***Where can I get circuit schematics and connector pinouts for Integrator boards?***
- ***Can I get the OrCad schematics for my Integrator board?***

  We do not supply OrCad schematics for any of our development boards products in source form. Circuit schematics in .pdf form, edif netlists and component lists (Bill of Materials) were included on the AFS (ARM Firmware Suite) CD-ROM, up to and including AFS version 1.3. AFS ships with each Integrator product purchased. Now, this information can be downloaded from the ARM website: http://www.arm.com/support/downloads/integrator.html

  Connector pinouts for the HDRA, HDRB, EXPA, EXPB, EXPM and EXPIM inter-board connectors are given in the appendices of each Integrator product's User Guide. A paper copy of the User Guide for each board is included with the board. Electronic versions of the manuals are available from the documentation area of ARM's website: http://www.arm.com/documentation/Boards_and_Firmware/index.html

- ***Can I get the PCB gerber files for my Integrator board?***

  We have made the top and bottom layer PCB gerber files for the logic modules and logic tiles available, to assist developers who are designing their own Integrator-compatible boards. Connector type and placement for the HDRx connectors on the range of ARM core modules is the same as for the EXPx connectors on the logic modules.

  Gerber files for other layers of the PCBs are not available. We cannot provide assistance in interpreting or using gerber files. If you do not have a gerber file viewer, it should be possible to use an Internet search engine to locate and obtain a free viewer from a 3rd party, from the Internet.

  To obtain the gerber files for your board, please contact your Integrator board supplier.

- ***How do I design my own Integrator-compatible boards?***

  The level of knowledge required will depend on the function of the board you are designing. For example, your board may contain AMBA slaves, masters, or both. Before commencing your design, you must understand how the Integrator family of boards operates. This information can be gained from the following sources:

  - **The AMBA 2.0 Specification** (available on ARM website.) This will enable you to understand how the Integrator's system bus operates. The system bus connects the various plug-in modules and motherboard resources together. Remember that the Integrator system bus uses a tri-state implementation of AHB. Integrator/CP uses an AHB-Lite system bus.
  - The latest version of the **Integrator/AP User Guide** contains information that will be useful in understanding how the boards communicate with each other. Particular attention should be paid to Appendix C, which describes the operation of the System Bus, and includes example VHDL source code for a bus master interface.
  - For slave designs, **example RTL source code** (in both Verilog and VHDL) is supplied with logic modules/tiles, on the accompanying CD-ROM. This code is known as 'Example2'.
  - **Integrator circuit schematics**. Originally these were provided on the AFS CD-ROMs, but now this information is available for download from the ARM website. See FAQ 'Where can I get circuit schematics and connector pinouts for Integrator boards?' for details.
  - For **mechanical details** showing board dimensions and connector placement, the appendices of the Integrator user guides should be consulted. We have also made the top and bottom layer PCB gerber files for the logic modules and logic tiles available for download. Please see FAQ 'Can I get the PCB gerber files for my Integrator board?'.
  - For **part number information** on the **connectors** used in the Integrator system, please refer to the FAQ 'Obtaining connectors used to stack modules in the Integrator family', and the Bill of Materials for each board.

- ***Can I get the HDL source code for the Integrator FPGAs / PLDs?***

  Integrator/AP
  We do not release the source code for the Integrator/AP's system controller FPGA, the core module FPGAs and the PLDs on the core modules and motherboard. Although it is possible to re-program this logic with your own designs, we do not encourage this, and cannot provide support for boards which have been modified in this way.

  Integrator/CP
  The top-level VHDL source code for the CM FPGA, Alpha LED PLD and Flash/Ethernet PLD are supplied on the CP CD-ROM from CD v1.1 onwards. The ARM PrimeCells are supplied as 'black box' netlists. This will allow some modifications to be made, such as removing unwanted peripheral control logic.

  Most of the CM FPGA's capacity is used up by system functions, so if any quantity of logic is to be added to the system, addition of one or more logic modules may be necessary.

- ***Obtaining connectors used to stack modules in the Integrator family***

  The connectors used to stack modules in the Integrator family are manufactured by Samtec. You can download the mechanical dimensions, request samples, etc., from the Samtec website - http://www.samtec.com/

  The parts used are:

  Top:

  200 Way Quad Row Plug TOLC-150-32-F-Q
  120 Way Quad Row Plug TOLC-130-32-F-Q

Bottom:

200 Way Quad Row Socket SOLC-150-02-F-Q
120 Way Quad Row Socket SOLC-130-02-F-Q

- ***Is there an Angel port available?***

Angel is a remote debug agent, which allows the Integrator to be connected to a debugger running on the host computer via an RS232 serial port.

Integrator/AP
On very early Integrator/AP systems, Angel was not installed as standard, but was available as an image that could be programmed into the Application flash.

On current boards, Angel is pre-programmed into the Boot ROM, along with the boot monitor and FPGA configuration data. Angel can be installed as the boot image by entering 'bi 911' at the boot monitor prompt. Subsequently. Booting the system with DIP switches S1-1 ON and S1-4 OFF will run Angel. Refer to the ARM Firmware Suite (AFS) CD-ROM for full details.

If the FPGA configuration data is replaced with a recent image, using the Progcards utility, then the FPGA configuration data, Boot monitor and Angel will all be written to the Boot ROM. See FAQ entry 'How do I re-program the FPGAs to the latest revision?'.

Integrator/CP
Angel is not pre-installed on the Integrator/CP, as the FPGA configuration flash is not shared with the boot code area, as is the case in the AP system. The AFS 1.4 CD-ROM contains two Angel images for Integrator, 'angIntegrator.axf' and angIntegrator_SDRAM.axf.

'angIntegrator.axf' is built to run from the SSRAM on the AP motherboard at `0x28000000`. The Integrator/CP does not have SSRAM at this address, so the Angel port for this board (angIntegrator_SDRAM.axf) has been built to run from SDRAM.

This port of Angel should be loaded into the application flash area with a specified image number, so that it can be invoked from the boot monitor, as described above. If no SDRAM is fitted when the boot monitor runs, 'S.D.' is output on the alpha-numeric LED display, and the boot monitor will not proceed any further, remaining in an endless loop. See FAQ entry 'Integrator/CP displays 'S.D.' – What does this mean?' for more details.

- ***How can I implement semaphores in an Integrator system?***
- ***Does Integrator support HLOCK/SLOCK?***

An AMBA bus master may need to perform a sequence of memory accesses atomically, i.e. as an indivisible unit, without another master being granted the bus in between them. To do this it makes a request to the arbiter not to grant a different bus master until the set of transfers has completed. This is sometimes known as locking the bus. This is done using the HLOCK or SLOCK signal for AHB and ASB systems respectively.

The only ARM instruction that causes this signal to be asserted is the SWP instruction, which swaps the contents of a memory location and an ARM register by performing a memory read followed by a write to the same location. This instruction is normally used for semaphores; a mechanism for controlling access to shared resources in multiple process or multiple processor situations.

The integrator/AP implements an arbiter that supports locking of the system bus. ASB based Integrator systems should all fully support locking the system bus as the result of a core module executing a SWP instruction. However, AHB based systems have some limitations in this area due to the design of the bridge in core module FPGA.

Multiple processes running on a single ARM core module

The SWP instruction always results in an atomic read/write operation on the core module's local memory bus, for the following reasons:

- The SWP instruction (as per all instructions running on an ARM processor) cannot be interrupted part way through.
- The ARM core is the only master on that bus. This scenario will work on all Integrator systems, regardless of whether the core module asserts the system bus HLOCK signal.

Multiple AHB masters (e.g. core modules)

Integrator/AP systems
The bridge used in current core modules does not assert the system bus HLOCK signal when bus transactions that were generated by a SWP instruction are passed through it. This means that read/write transactions generated by a SWP instruction are no longer indivisible when they reach the system bus. Hence there is no shared area of memory between a core module and another master (e.g. a second core module) in which semaphores can be implemented using the SWP instruction.

Customers who need to implement semaphores between bus masters are advised to investigate alternative software algorithms, e.g. the bakery algorithm.

Integrator/CP systems
The Integrator/CP uses AHB-lite for its system bus. As this standard only supports one bus master, a SWP instruction will always be atomic on the system bus.

- ***How does the Integrator part numbering system work?***

There are several numbering systems that are used to describe various aspects of the Integrator development system. Traditionally, none of these part numbers have been referenced in the User Guides for each board type, although this is changing with more recent products.

Bare PCB part number
Etched into the top copper layer of each Integrator board is a part number of the form HPI-XXXXy, where 'XXXX' is a unique number for each board type, and 'y' is the revision letter for that board number. For example, the PCB used on the Integrator/LM-EP600E+ board has the part number HPI-0073A. This part number is usually obscured by a small white label, which shows the part number of the populated PCB.

Each Integrator PCB also has a small 'MOD' area with several check boxes etched in the copper on the surface of the PCB. If minor modifications to a board design are required, which do not warrant layout of a new PCB, modifications (such as a wire strap) may be made to the board, and one or more of the MOD boxes may be marked to indicate this.

Populated PCB part number
This is usually shown on a small, white label, which is placed on top of the bare PCB part number. The populated PCB part number has the form HBI-XXXXy, with the 'XXXX' and 'y' parts having exactly the same meaning (and usually the same values) as for the bare PCB part number. The prefix 'HHI' has also been used to mean the same thing on older boards.

Some boards are manufactured with slightly different build options, for example: Different resistor values are used to set the various supply voltages for different parts of the ARM test chip fitted to a core module. In this case, the populated PCB part number stays the same.

In some instances, the same PCB is used with different build options, to give two distinctly different products. For example, the Integrator/AP is available in two options; one with the CompactPCI connectors fitted, and one without. In cases such as this, a suffix is added to the populated board part number to describe the difference in the product. An example part number of this form is:

'HBI-0048D-nocpci' – which is the Integrator/AP with no CPCI connectors fitted.

Assembly number
Development systems offered as pre-built assemblies such as the Integrator/PP2 are given a part number of the form 'MSI-XXXXy'. The 'XXXX' and 'y' elements are a unique part and revision number for the assembly, and bear no relation to the board or kit part numbers.

ARM sales part number or 'kit' number
The following part number series are used by our sales offices to identify a kit of parts, which may contain one or more Integrator boards, cables, user manuals, CDs, etc. These kit numbers are used on the product pricing pages on our website, and in documentation such as sales invoices. For example:

CM920-BD-0113A is an Integrator/CM920T built with the HBI-0047 board.

The latter portion of a kit number (e.g. 0113A) is not related to the 'XXXXy' portion of the board and assembly part numbers, but is a unique kit number. In the past, these had the prefix 'KPI', e.g. 'KPI-0113A'. The letter at the end of a kit number is the revision of the kit. This revision letter is incremented when a different variant of the kit is offered. For example:

INAP1-BD-0109A – Integrator/AP with CPCI connectors fitted
INAP1-BD-0109B – Integrator/AP without CPCI connectors fitted.

- ***What causes Integrator boards to fail?***

  During the time that we have been producing and selling Integrator boards, we have found that relatively few boards fail in the field. Boards that do fail do not usually fail of their own accord, due to a manufacturing fault or component defect. Failures which prevent the boards from working are frequently caused by external factors. The most common failures are associated with the high-density header connectors, and the large FPGAs. Typical causes of failure and associated precautions are:

  - Careless insertion of a module into the connectors of the board beneath. This can cause bent pins resulting in open or short circuits in the connectors. Damage can also be caused by debris getting into the socket holes of the connectors on the bottom of the boards. Always check all header plugs and sockets before plugging modules together. If pins become bent, it may be possible to straighten them - see 'Integrator boot monitor gives corrupted output' for example symptoms and remedy.
  - Failure to check/simulate FPGA designs properly before testing on the Integrator. Often, FPGA I/Os are destroyed by unintentionally making the FPGAs drive large numbers of bus signals to incompatible states. Mass bus clashes can cause the FPGA to draw so much current that the whole device is destroyed. In one extreme case, enough heat was generated to melt the solder balls holding the BGA package to the PCB.
  - Failure to understand how Integrator signals are driven. Occasionally, a board is damaged by a user's design intentionally driving signals to inappropriate states. This is more likely when custom hardware interfaces with an Integrator board. Off-board connections which come from FPGAs are not buffered, since this would reduce flexibility and incur speed penalties. Refer to the Integrator user guides, circuit schematics and component manufacturers' data sheets for I/O capabilities and connections.
  - Excessive or reverse polarity PSU voltage input to the board. None of the current Integrator boards have integral over-voltage protection. Most Integrator boards are not reverse-polarity protected. We recommend that the supplied PSU is used where applicable. It is convenient to power the Integrator/AP board from a suitably loaded ATX power supply, although this method is not without problems. (See below, and FAQ entry: 'ATX PSU won't switch on'.) Take care when using a variable output voltage lab power supply; it is easy for the controls to be adjusted inadvertently. If your PSU has output current limiting facilities, it would be wise to set these to an appropriate limit, rather than leaving set to maximum.
  - Careless use of test equipment. It is not difficult to destroy components or PCB traces by shorting an oscilloscope or DVM probe between IC legs or connector pins. Shorts can also be caused by objects beneath the board. See next item:
  - Foreign conductive objects beneath a board, causing short circuits. Paper clips, staples on books, etc. can short adjacent PCB traces or vias, causing component or PCB damage. It is recommended that an Integrator system be placed on a non-conductive surface, or mounted

in an enclosure on suitable stand-offs. Apart from the Integrator/CP, Integrator boards are not over-current protected by fuses. This means that a power supply rail short on a board can easily damage the PCB. Note that most ATX power supplies have very large output current capability, compared to the current typically drawn by an Integrator system. For example, a typical 250W ATX supply can source 25A on the 5V rail, 15A on the 3V3 rail and 8A on the +12V rail. This is far more than is needed to burn out PCB traces, should a short occur.
- Poor attention to ESD procedures. This problem is worse in drier climates, where static charge build-up is more pronounced. Always observe anti-static precautions, as described in the user guide for your board.

- ***Is it possible to get an ARM7TDMI core module with an ETM?***

We do not produce a CM7TDMI with an Embedded Trace Macrocell, and there are no plans to so at present. At some point in the past, a few CM7TDMI core modules were released to customers with an ETM implemented in the core module FPGA, but this had some limitations:

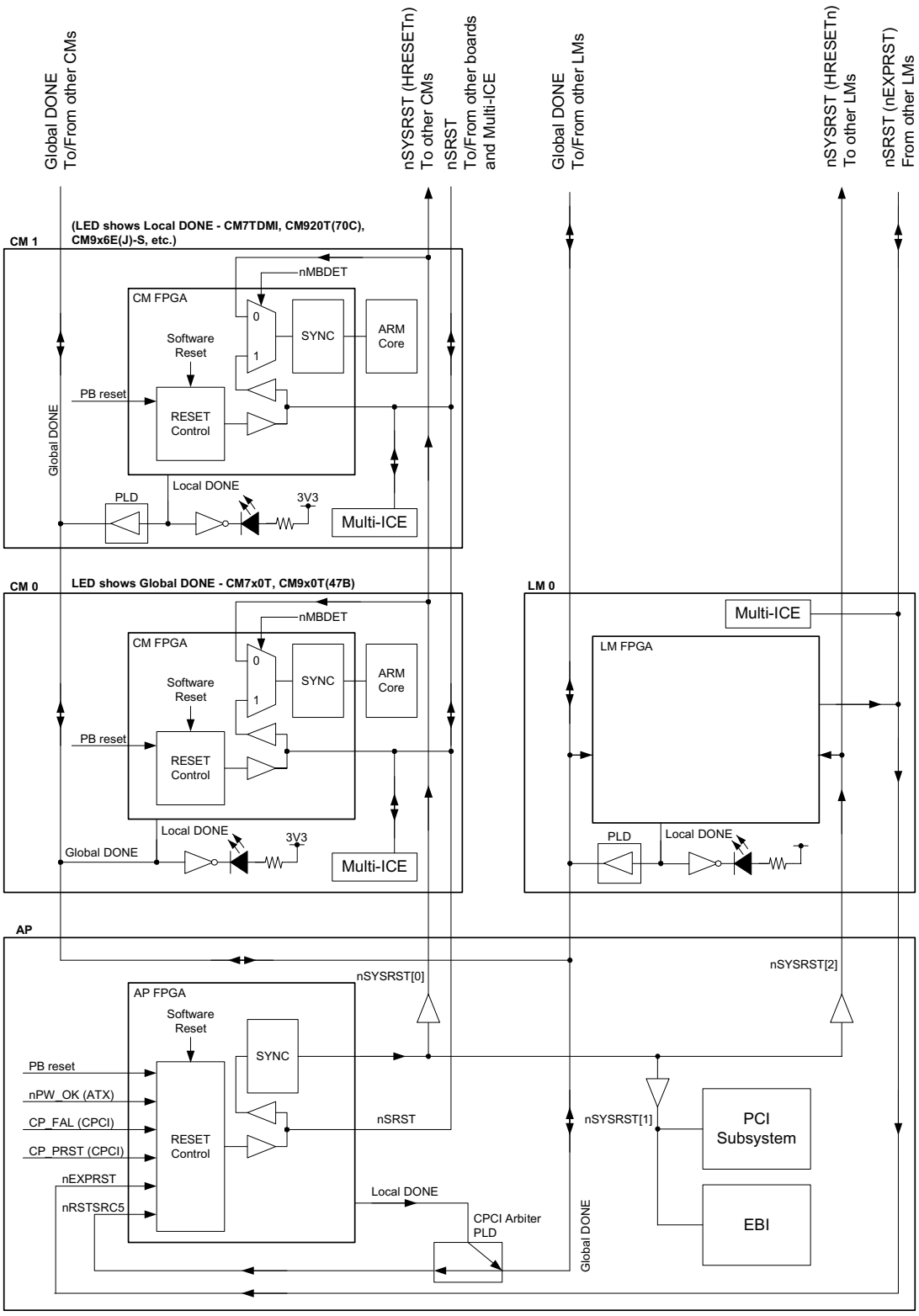
- The 'Trace' Mictor connector on the CM7TDMI has a different pinout to the one which is in common use on other development boards. This is not compatible with the standard LVDS header supplied with ARM's MultiTrace unit. We do not supply alternative LVDS headers with the alternative (older-style) pinout.
- The FPGA image was only suitable for use standalone, or on an AP motherboard programmed for an ASB system bus.

The FPGA image for a CM7TDMI core module, with Xilinx XC4062 FPGA is available for download from the ARM website, see FAQ 'How do I re-program the FPGAs to the latest revision?' for the URL.

- ***How does the reset circuitry work in an Integrator system?***

The reset controller design and signal routing in the Integrator system is somewhat complex, and operation is slightly different depending on whether a core module is used standalone or mounted on a motherboard.

The following diagram shows the reset signal routing for an Integrator/AP system with core and logic modules attached. A functional description is given below the picture.



Both the motherboard and core module contain reset control logic. The motherboard reset controller is the main reset controller in the system, and this generates the system master reset signal, nSYSRST. This is actually the AMBA system reset signal, which is called ‘HRESETn’ in an AHB system or

'BnRES' in an ASB system. This signal is routed to each logic block on the AP, and every core and logic module in the system. There are a number of sources which can cause a system reset to occur:

- FPGADONE signal (aka Global DONE). This is input to the AP board reset controller as 'nRSTSRC5'
- Core module nSRST signal (can be driven by Multi-ICE JTAG debug hardware or CM reset controller)
- Logic module nEXPRST signal (same function and header pin number as nSRST, but on the logic module stack)
- ATX Power OK signal (nPW_OK)
- Push button (on motherboard)
- CompactPCI power fail signal (CP_FAL)
- Software generated reset (writing a specific value to a register in the AP causes a reset).

When the system is powered up, the Xilinx FPGA on each board starts to load its configuration image from its respective flash memory device. While the FPGAs are configuring, they hold their 'DONE' outputs LOW. All the local DONE signals from the FPGAs are combined in a wire-AND configuration, to produce the system-wide 'Global_DONE' signal. There is an LED on each board, which indicates the state of either the local or global done signal, depending on the board type. (See FAQ 'What do the 'FPGA OK' or 'DONE' LEDs indicate?').

Global_DONE is an input to the system reset controller on the AP motherboard. It is also input to the FPGA on a logic module, so that the user's design can make use of it if required. The system is held in reset until all FPGAs have configured. This ensures that no system activity can take place until all system components are ready.

The nSYSRST signal can subsequently be asserted from any of the other sources on the AP board, listed above. Subsequent resets do not cause the FPGAs to be reloaded, with the exception of the Excalibur core module.

Core modules also contain a reset controller, which have several inputs:

- Multi-ICE nSRST signal
- Core module reset button
- Motherboard reset controller output, nSYSRST
- Software generated reset (writing a specific value to a register in the CM causes a reset)

When the CM is used standalone, these sources reset the ARM core directly. When the CM is mounted on an AP motherboard however, the system reset signal, nSYSRST is the only direct reset source to the ARM processor test chip. The core module reset sources are routed down to the AP reset controller on the nSRST line, and arrive back as nSYSRST.

In summary, whenever any of the AP, CM or LM reset sources is asserted, a system reset occurs. Since nSRST is also driven LOW during any system reset event, Multi-ICE (and consequently the debugger) are alerted that a reset has occurred.

- ***What is the difference between Integrator/PP1, PP2, SPP1 and SPP2 systems?***

PP1 / SPP1

The Integrator/PP1 and Integrator/SPP1 are ready-built, tested development systems. They consist of exactly the same hardware. This comprises:

- Standard ATX desktop PC case with ATX PSU
- Integrator/AP motherboard
- Integrator/CM-920 core module, fitted with 128MB SDRAM DIMM
- Intel Pro/100+ Server adapter PCI card
- Video Logic Neon 250 PCI Graphics card
- Standard 'PS/2' style PC keyboard and mouse

The difference between the PP1 and SPP1 systems is that the SPP1 includes a binary image of SymbianOS 7, pre-programmed into flash. A utility called PPFU (Porting Platform Flash Utility) is also included. This enables the developer to load and run ARM executable files either in RAM or flash on the Integrator system. File transfer is done using TFTP (Trivial File Transfer Protocol) across Ethernet.

PP2 / SPP2
The Integrator/PP2 and Integrator/SPP2 are also ready-built, tested development systems, but instead of being housed in an ATX case, the PP2 and SPP2 have a custom-designed chassis, into which the Integrator boards and other hardware are mounted. Instead of a PCI graphics card, an ARM interface module provides the VGA interface, which drives a built-in color LCD panel. An external monitor can also be driven if required.

These systems comprise:

- Custom designed chassis with PSU
- Integral back-lit VGA color LCD panel with optional touch screen
- Integrator/AP motherboard
- Integrator/CM-920T core module, fitted with 128MB SDRAM DIMM
- Integrator/LM-XCV2000E logic module
- Integrator/IM-PD1 Interface module
- Intel Pro/100+ Server adapter PCI card
- Standard PC keyboard and mouse

The PP2 and SPP2 systems are identical in terms of hardware. The difference between the two products is that the SPP2 has SymbianOS 7 binaries pre-programmed into flash, the same as the Integrator/SPP1. The PPFU utility is also included.

The SymbianOS binaries included with both these systems were generated from OS version 7.0.3. If you require the full BSP (Board Support Package), build scripts, etc. you should contact Symbian.

2. Questions Specific to the Integrator/AP system

- ***What is contained in the boot ROM?***
- ***Where is the Integrator/AP's FPGA configuration data stored?***

The boot ROM contains the system controller FPGA configuration data and the code that runs when the Integrator system is powered up, or reset. The boot ROM is a 512KB Atmel flash device. The FPGA configuration data is stored from the top of that device downwards. The boot code is stored from the bottom of that device upwards.

- ***Can I change the boot ROM software?***

Yes, but it is preferable to use the Application Flash area for storing user code. The core can be made to boot directly from the Application Flash area by setting DIP switch SW1-1 to OFF.

It is possible to re-program the boot ROM by using a program called the boot flash utility (bootFU), which is supplied as part of the ARM Firmware Suite (AFS). This is described in a readme file on the CD. This should not be confused with the ARM flash utility (AFU) which is used to program the Application flash. For full details of the boot monitor firmware, refer to the AFS Reference Guide and CD-ROM.

Reprogramming the boot ROM device on an Integrator/AP is not recommended, as the flash device used on this board cannot be partially erased. Any change requires the FPGA configuration data to be backed up to Application flash, followed by a full erase. The FPGA data is replaced when the boot ROM is re-programmed. If this procedure is interrupted, it is not uncommon for the FPGA configuration data to be lost, rendering the board inoperative. If this happens, it will be necessary to replace the FPGA configuration data using Multi-ICE and the Progcards utility. See 'How do I re-program the FPGAs to the latest revision?' for more details.

- ***ATX PSU won't switch on***
- ***ATX PSU cuts out***

ATX PSUs are Switch Mode Power Supplies (SMPSs). Any SMPS has a minimum current, which must be drawn from the supply to allow it to operate correctly. With typical ATX supplies, if this lower limit is not satisfied, the PSU will either not start up, or will start up and then shut down shortly thereafter.

The current drawn by a typical Integrator development system is often lower than the minimum allowable current for an ATX PSU; hence this problem may be seen when powering Integrator boards from such a PSU.

The solution is to add a load resistor to the +5V supply output from the ATX PSU. This can be done by connecting a suitably rated power resistor between any pair of red and black wires on the PSU's output wiring harness. A convenient way of doing this is to use one of the spare disk drive power connectors.

We have found that most ATX PSUs are content with at least a 1A load on the 5V rail. Ohm's law shows that in order for 1A to be drawn from a 5V supply, a 5Ω resistor would be required. The power dissipated in such a resistor would be 5W. Therefore, a 5Ω resistor with a maximum power rating of at least 5W is needed. It is sensible to use a resistor with a power rating significantly greater than this, in order that the resistor does not run at its maximum temperature, and get hot enough to injure someone. The resistor can be kept cooler by clamping it to a suitable heatsink, e.g. the ATX PSU case.

It is not usually necessary to put dummy loads on the other supply rails (i.e. 3V3, ±12V, -5V) as the whole PSU is often regulated from the +5V supply.

- ***PL010 UARTs – Lack of hardware flow control***
- ***Can I upgrade the PL010 UARTs in the Integrator/AP to PL011?***

  The UARTs used on the Integartor/AP motherboard are PrimeCell PL010. The serial port sockets on the Integrator/AP are equipped with the modem status and control signals: DCD, DSR, CTS, DTR and RTS, but the PL010 UART does not properly support hardware handshaking. Instead, these signals are accessed and controlled through memory-mapped registers in the System Controller FPGA.

  The state of the modem status inputs DCD, DSR and CTS can be examined by reading the UART flag registers, UARTx_FR. These inputs can also be made to generate an interrupt to the main interrupt controller, by setting up the UART_ICR register.

  The state of the modem control outputs, DTR and RTS can be set by writing to the relevant bits in the System Control register. Please see the User Guide for the Integrator/AP for full details.

  It is not feasible for the end user to replace the PL010 blocks in the AP's System Controller FPGA with the PL011 UART, which does support hardware handshaking, as ARM has not released the HDL source code for this FPGA.

- ***Can I use the AP's 'External Interrupts Connector', J20?***
- ***How can I add an external interrupt source to the Integrator system?***

  No connector is fitted in the J20 position on a standard Integrator/AP. If the AP board is using System Controller FPGA Build 26 or higher, then this connector can be used to add external interrupt sources to the system.

  The pads on the PCB are placed to accept a 10-way DIL header (2 rows of 5 pins on a 0.1" pitch). The pinout for this connector can be seen in the circuit schematics for the AP board, but in summary, pins 1 & 10 are GND, and pins 2 through 9 are external interrupt inputs 0 through 7 respectively. The System Controller FPGA design NANDs all eight inputs together, so a '0' on any J20 input will cause the EXTINT bit (bit 21) in the AP's interrupt controller registers to be set, indicating that an interrupt has occurred. See the Integrator/AP User Guide for a full description of the interrupt controller.

  It should be noted that these inputs are not latched, so any external circuitry that generates an interrupt should assert the interrupt input until code running on the ARM clears the interrupt source. Failure to do this may result in an interrupt not being acknowledged by the ARM core, or multiple interrupts being generated. Simply connecting a pushbutton switch to the J20 pads is not sufficient.

  The external interrupt facility on J20 is not supported by any of the System Controller FPGA builds prior to Build 26. In this case, the end user should not connect anything to the J20 pads to avoid damage to the Integrator/AP or any connected circuitry. It is possible to update the FPGA configuration on such an Integrator board, so that J20 can be used. See FAQ 'How do I re-program the FPGAs to the latest revision?'

  An alternative method of connecting external interrupts to the system would be to use the logic module interrupt source pins on the EXPB connector. If you do not posses a logic module, you will need to obtain the correct mating connector to avoid having to make solder connections direct to the AP board. Connector details are described in FAQ 'Obtaining connectors used to stack modules in the Integrator family'.

  The logic module interrupt inputs on the AP motherboard are labelled nIRQSRC[3:0], on the circuit schematics. These signals feed the EXPINT[3:0] bits in the AP's interrupt controller. This is documented in the Integrator/AP User Guide.

- ***What is the difference between the core and logic module sites on an AP?***

  One difference between HDRB and EXPB is the way in which interrupts are handled. Core modules receive interrupts, whereas logic modules (which implement peripherals) usually generate interrupts.

The signals on HDRB and EXPB concerned with interrupts are different. All signals on these pins are driven open-collector (open-drain) so that there is no conflict if logic and core modules are accidentally connected together.

Note: A logic module could be used to implement a processor, in which case it will act like a core module and expect to receive interrupts. In this case it should be installed in the HDRA/B stack.

On the Integrator/AP, the 32 GPIO lines are routed to the EXPB connector, but not the HDRB connector. The GPIO signals are therefore not available on the Integrator/SP.

- ***What is the difference between the Integrator/AP and the Integrator/SP?***

The Integrator/AP was designed to allow development of ARM-based ASICs, both in terms of hardware and software. The Integrator/SP has less functionality, and was intended to be primarily a software development platform.

The main differences are:

1. The Integrator/SP is no longer available.
2. The Integrator/AP has three 5V PCI slots, whereas the SP has no PCI slots.
3. The AP has an extra set of connectors for installing expansion logic modules. The SP only has connectors for processor header cards (core modules.)
4. The AP board is approximately ATX size, whereas the SP is designed to fit in a Compact PCI rack.

- ***What peripherals are included on an Integrator/AP?***

The standard peripherals, implemented in the system controller FPGA, are 2 UARTs, 3 timers, a real-time clock (RTC), 32-bits of GPIO, keyboard and mouse interfaces (KMI) and an interrupt controller. There is also a simple peripheral which is used to read 4 switches and write to 4 LEDs and a 2-digit alpha-numeric display. The UART, KMI and RTC are ARM PrimeCell peripherals which are separately licensable if required for inclusion into your logic design.

Connected to the AMBA system bus is a static memory controller, also known as the external bus interface. This provides the interface to the boot ROM, flash and motherboard SRAM.

- ***What types of PCI cards are compatible with the Integrator/AP?***
- ***Does ARM provide PCI card driver software for use with Integrator?***

There are many different types of PCI cards commercially available, most of which have been produced for the PC market. When choosing a PCI card to use with the Integrator/AP, one should ensure that the card is designed for 5V operation. (This is the same requirement for 32bit PCI slots in most personal computers.) It would also be beneficial if some driver source code is available for the card, or detailed technical documentation which could be used to help write a driver, as ARM does not provide this software. There are some PCI initialisation routines on the AFS CD. See AFS documentation for details.

It is possible to modify the Integrator/AP board to accept 3.3V PCI cards, although this is not a straightforward task. The PCI Specification defines 2 types of expansion connectors: one for 3.3V and one for 5V. These connectors have different keyways, so you cannot plug a 3.3V PCI board in a 5V slot and vice versa. The Integrator/AP board is sold with 5V PCI connectors and the 'PCI V(I/O)' link, WL2 is set for 5V PCI I/O voltage by default.

If you want to use 3.3V PCI cards you will need to:

1. Remove one or more of the PCI connectors and replace with a 3.3V connector. PCB rework equipment will be required to carry out this task. Note that it may be possible to make any PCI

card fit in a 5V PCI connector by carefully removing the plastic keyway from the connector. This is not a recommended solution, as damage to adjacent connector pins would be likely.

2. Change the WL2 link to select 3.3V. This link is located near the PCI slots.

Caution: WL2 selects the I/O voltage for all the PCI slots, so you cannot use a 3.3V card and a 5V card at the same time, since you could damage the cards and the Integrator/AP board. Schematic diagrams for the AP (and other Integrator boards) can be downloaded from http://www.arm.com/support/downloads/integrator.html.

- ***How do I fit my Integrator/AP into an ATX case?***
- ***What are the differences between the standard Integrator/AP and the 'no CPCI' version?***

The Integrator/AP PCB is ATX size, but the board's Compact PCI connectors, J1 and J2 extend past the end of the board, making installation in most ATX computer cases impossible. There are ATX cases, which accept the AP board with only minor or no modification, but these are hard to find.

It is not feasible to remove the CPCI connectors without damaging the PCB, but it if the CPCI interface is not required, it is possible to make the board fit in some cases by removing the plastic lugs from the centre of the CPCI connector, J1.

In spring 2002, a slightly modified version of the Integrator/AP board was made available, with the CPCI connectors missing. This makes installation possible in a greater variety of ATX cases. The modified AP has board part number HBI-0048D-nocpci, and is sold as kit number INAP1-BD-0109B. The board electronics (PCI to CPCI bridge, CPCI arbiter, etc.) are still present, but are effectively redundant. It is only the CPCI connectors that are missing.

- ***Debugger reports 'Could not stop target processor' with Integrator Logic Module in system***

The debugger may report 'Could not stop target processor' when using a LM-XCV400+ logic module with an ARM7TDMI.

To ensure that the Integrator system operates correctly, a pull-up on the FPGA input pad for the signal GLOBAL_DONE (pin C13) is required on ALL logic module designs. Specify this during the synthesis stage.

Alternatively, a 10K resistor can be fitted on the logic module prototyping grid between holes E9 and H0.

- ***What do the red and green pushbuttons on the Integrator/AP do?***
- ***Multi-ICE can no longer get control of the processor. How do I fix this?***

The red pushbutton is the power button. It allows the Integrator system to be switched on and off only when powered by an ATX power supply. The green pushbutton is the system reset button.

If you get the system into a state where the debugger has lost communication with the board then you may need to power cycle the board by pressing the RED power button twice. The GREEN reset button will apply a system reset to the processor and peripherals, but will not clear any breakpoints or watchpoints set in the EmbeddedICE macrocell, which is inside the ARM core. This is because at power up the system asserts nTRST to the ARM core, but only asserts the system reset (either nRESET, BnRES, HRESETn or nSYSRST) when the GREEN button is pressed.

- ***What is the EXPM connector used for?***
- ***Can these signals be reused for a different purpose?***

The EXPM connector carries signals for the EBI (External Bus Interface). The EBI is normally used to allow the Integrator/AP motherboard to access the Intel flash, SSRAM and the boot ROM.

The EBI signals are routed to the EXPM connector and can be used to allow a 4th memory mapped device, with similar timing and control signals, to be added. It is also possible to add additional banks of flash memory.

The Integrator/LM-XCV400+ attaches to this connector and therefore can be used to develop an EBI memory mapped peripheral. In general you should develop peripherals using one of the AMBA buses rather than the EBI (e.g. ASB, AHB or APB).

If developing AMBA based peripherals you may wish to use the EXPM connector on the Integrator/LM for a different purpose, for example:

1. To allow the Integrator/LM to directly control the memory devices on the motherboard. Note that the SSRAM cannot be controlled in this way.
2. To reuse these signals as general purpose IO

There is a signal, EBIEN, that allows you to disable the EBI's drivers. This would then allow the Logic Module to control the memory on the motherboard for its own purposes. If the chip select signals (nMCS[3:0]) are never asserted no memory device will ever be selected and the other signals (i.e. MA[25:0], MD[31:0], nMCSN[3:0], nMWR[3:0] and nMOE) could then be used as IO.

With EBIEN low, accesses to the EBI at `0x20000000` - `0x2FFFFFFF` and `0x00000000` - `0x0FFFFFFF` on the system bus will no longer return valid data. Exact behavior varies with different System Controller FPGA builds. Prior to build 26, the EBI responds with garbage data. From build 26 onwards, the EBI makes no bus response whatsoever, meaning that user logic must respond to these address ranges on the system bus.

Consequently if you wish to boot from the motherboard's flash or boot ROM you will need to change the level of this signal once your code has copied itself to RAM on the Core Module. This arrangement has some disadvantages and may not be suitable for all applications. One way to drive EBIEN low is by using the corresponding jumper on the Integrator/AM.

Please note that while EBIEN and the 4 chip select signals (nMCS[3:0]) are on the EXPM connector they do not all connect to pins on the logic module's FPGA. You may need to connect these signals to spare pins on the FPGA using the prototyping grid.

If you just want to use the EXPM connector of the Integrator/LM to provide additional IO signals, a more realistic way of doing this is to disconnect the EXPM connection to the motherboard, hence you can do what you like with these pins on the logic module. Some ways to achieve this are:

1. Carefully remove the EXPM connector from the motherboard or the underside of the Logic Module. If you are experienced with performing this kind of work on PCBs, and have the right tools, this is a good option.
2. Mount the Logic Module on the core module stack (this has no EXPM connector). This is very easy to do, but you should consult the Integrator/LM-XCV400+ User Guide (ARM DUI 0130A). In particular, pay attention to sections 3.2 'Bus interfaces' and 2.5 'Differences between core and logic modules'.

- ***Integrator Angel requires a SDRAM DIMM***

The pre-built Angel image supplied on the ARM Firmware Suite 1.0 CD only runs on an Integrator system consisting of an Integrator/AP or /SP, and an Integrator/CM that has a SDRAM DIMM fitted. This is because this version of Angel places the stack in the core module's SDRAM region.

From AFS 1.1 onwards, the pre-built Angel image for an Integrator/AP system will run even if there is no SDRAM DIMM fitted. This is because the stack is placed in SDRAM if a DIMM is fitted, or in local SSRAM if not.

Use of Angel on other system set-ups requires a rebuild of Angel, mapped to that target.

More information on the Integrator version of Angel can be found in Appendix A of the ARM Firmware Suite Manual.

- ***Integrator boot monitor gives corrupted output***

Occasionally, a user reports that the boot monitor on the Integrator/AP is producing corrupted output to an attached serial terminal. Below are example screen shots of correct and corrupted boot monitor startup banners:

a) Example of correct boot monitor output	b) Example of corrupted boot monitor output
ARM bootPROM [Version 1.1] Rebuilt on Mar 8 2000 at 14:46:44 Running on a Integrator Evaluation Board Board Revision V1.0, ARM7TDMI Processor Memory Size is 256KBytes, Flash Size is 32MBytes Copyright (c) ARM Limited 1999 - 2000. All rights reserved. Board designed by ARM Limited Hardware support provided at http://www.arm.com/ For help on the available commands type ? or h boot Monitor >	AAMMbbooPPOO VVrriinn1111 eeuull nnMMrr 0000aa 4444::44RRnniiggoo > ssEE aauuttoo oorr BBaaddRRvvssoo ss ssPPooeessrrMMmmrr iieeii ssyyee,,FFaahhSSzz ss%BBttssCCp prrggtt(())AAMMLLmmttdd1199 0000 ll iihhssrrssrree..BBaaddddssggee yy%

If your Integrator system produces corrupted boot monitor output, similar to '**b**' above, then it is likely that pin 1 of J5 (the Integrator/AP HDRA connector) has been bent or broken off.

This is bit A0 of the system address bus, and the fault may go un-noticed until the ARM core tries to do a byte access on the system bus, as is necessary to execute a printf() style function. This type of operation is used by the boot monitor to display the startup banner. As can be seen, alternate bytes are either doubled up or missing altogether, as the ARM unintentionally accesses alternate byte locations in the motherboard memory.

If a pin is bent, it may be possible to rectify the problem by carefully straightening the pin with a suitable implement, e.g. a pair of thin, long-nosed pliers, or by sandwiching the pin between two very small flat-head screwdrivers.

If it is not possible to straighten a pin, if the pin has already been broken, or breaks in an attempt to straighten it, then the board should be returned to its place of purchase for repair. It is most likely that a repair fee will be charged, since this type of failure is not covered by the product's warranty. Customers should not attempt repairs on Integrator boards, since further damage to the boards may result.

To prevent this type of problem occurring in the first place, great care should be taken when connecting Integrator modules together. Frequent connection and disconnection of modules should be avoided. It is advisable to check that all plug pins are straight and that the mating socket holes are free from obstruction immediately before connecting boards together.

- ***Integrator/AP RTC (Real Time Clock) loses its value at reset***

The RTC on the Integrator/AP is implemented in a RAM-based FPGA, which means that the RTC logic ceases to exist when power is removed from the board. Unfortunately, the RTC value is also cleared whenever the Integrator system is reset, by any of the available reset sources.

Whilst it would be possible to modify the RTC logic in the FPGA so that the RTC registers retained their values through a reset, there are no plans to make this modification at present.

3. Questions Specific to the Integrator/CP system

- ***My Integrator/CP has the MAC address transposed***
- ***How do I program the Ethernet MAC address on Integrator/CP?***

The LAN91C111 is the Ethernet Controller (MAC/PHY) on the Integrator/CP. The Ethernet MAC address for this chip is stored in an EEPROM, which is programmed with a default value when the board is manufactured.

Integrator/CP baseboards should be pre-programmed at the factory with a MAC address of the form `00-02-F7-XX-XX-XX`. The first Integrator/CP baseboards manufactured had the MAC addresses programmed with the bytes in the wrong order. For example, if the board was supposed to have the address `00-02-F7-01-02-03`, the actual value used would have been `03-02-01-F7-02-00`.

This can be corrected by changing the value stored in the Ethernet controller EEPROM. The EEPROM can be written by the ARM core via some registers implemented in the baseboard PLD. For information about how to program these registers, please see the Integrator/CP user guide, section 9.2, and the LAN91C111 datasheet, downloadable from http://www.smsc.com.

A utility to perform this operation automatically, 'CPMACfix.exe' is included on the Integrator/CP CD-ROM v1.1, and in the Downloads area of the ARM website. A Multi-ICE unit is required to use this program.

You can also program this value manually by using the 'Fix_EEPROM' utility. Please see FAQ 'The Ethernet interface on my Integrator/CP has stopped working'.

- ***The Ethernet interface on my Integrator/CP has stopped working***

The Integrator/CP incorporates an RJ-45 Ethernet socket, which is driven by an SMSC LAN91C111 Ethernet controller IC. Some users have reported problems with these boards where the supplied selftest.axf program cannot find the board's Ethernet interface, or an OS with Ethernet support can no longer initialize this interface.

This is likely to be caused by a corrupted Ethernet configuration EEPROM. A utility called Fix_EEPROM has been written to allow the user to reprogram this device. This is covered in a separate FAQ and download on the ARM website.

- ***Integrator/CP boot monitor requires a SDRAM DIMM***
- ***Integrator/CP displays 'S.D.' – What does this mean?***

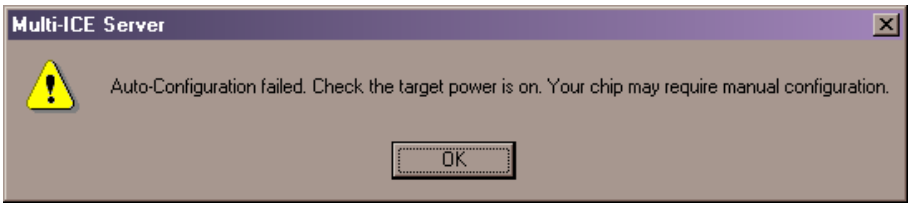
The boot monitor in an Integrator system traditionally runs direct from flash memory. Integrator/AP has physically separate flash areas for system firmware and user firmware, but on Integrator/CP the same flash device is used to store the boot monitor and user code.

This presents an obvious problem that code which changes the contents of flash memory cannot run from that same flash memory. On the Integrator/CP platform, the boot monitor copies itself to SDRAM, which allows it to erase or program the flash memory device.

If an Integrator/CP board is started up with DIP switch S1[1] set to ON (run the boot monitor), and there is no SDRAM DIMM present, the alpha-numeric display will briefly show 'CP', then the letters 'S.D.' will be displayed, indicating that the boot monitor cannot continue, since there is no DIMM fitted. Execution will remain in an endless loop.

- ***Multi-ICE cannot Auto-Configure when CP system in CONFIG mode***

  If a Multi-ICE unit is connected to an Integrator/CP system that has been placed in CONFIG mode, and an 'Auto-Configure' is attempted, Multi-ICE Server will display the following dialogue:



  This is because the EPM3032A PLD on the CP baseboard prevents the Auto-Configure feature from working correctly. It is therefore necessary to load a configuration file (.cfg file extension) into Multi-ICE Server. Configurations for all current CP systems are included on the Integrator/CP CD-ROM.

  Before the Multi-ICE configuration can be loaded succesfully, it may also be necessary to edit the 'irlength.arm' file in the Multi-ICE installation directory, and add the following line of text. This describes the length of the PLD's Instruction Register to Multi-ICE:

```
EPM3032A=10
```

4. Questions Specific to Core Modules

- ***Size of TCM / cache memory present in ARM test chips***
- ***Variations in ETM size on Integrator core modules***

Core modules that incorporate a cached ARM core may have cache memory integral to the test chip. The amount of cache memory present will depend on the core and/or test chip type.

Some cores have fixed size cache memories, for example: ARM9x0T, ARM7x0T, ARM1020E.

For synthesizable cores, cache sizes are specified at synthesis time, so test chips from different silicon manufacturers may have different sized caches present. Examples are the ARM926EJ-S and ARM946E-S.

Some cores, namely the ARM926EJ-S, ARM966E-S and ARM946E-S may also have some SRAM integral to the test chip. This memory is known as TCM or Tightly Coupled Memory. The silicon designer specifies TCM size at synthesis time, so test chips from different manufacturers may have varying amounts of TCM.

Integrator core modules are manufactured with batches of test chips from different silicon vendors. Core modules with the same part number may have different test chips, and hence different TCM and cache sizes.

Some core modules have test chips that are equipped with an ETM (Embedded Trace Macrocell). As is true for TCM and cache memories, the ETM size may vary (or there may be no ETM), as boards are built with different batches of test chips.

It is not currently possible for a customer to specify which test chip type is required when an order is placed, although it may be advisable to ask the sales representative which test chip type is currently shipping, in order to ensure that the board will satisfy your requirements. It is especially important to bear this in mind when placing repeat orders for core modules.

- ***Which types of SDRAM DIMMs will work with my core module?***
- ***Problems accessing core module SDRAM***

Any DIMM that complies with the following should work with an Integrator core module:

- Must be SDRAM
- JEDEC compliant
- 168-pin device
- 3.3 volt operation
- Non-parity
- Unbuffered
- Must have SPD EEPROM
- Supported sizes: 16MB, 32MB, 64MB, 128MB, or 256MB
- CAS latency = 2 cycles, or CAS = 3 cycles if user code programs CM_SDRAM register
- PC66, PC100, or PC133
- Geometry as specified in the table below.

Some Core Modules are supplied with a compliant SDRAM DIMM fitted. Whether you are using the supplied DIMM or an alternative, it is likely that the hardware default SDRAM parameters programmed into to the core module SDRAM controller will not match those of the fitted DIMM. Therefore, you must program the CM_SDRAM register in the core module with the relevant parameters to ensure the SDRAM controller works properly with the specific memory module that you have installed. Failure to do this may result in slow or unreliable memory accesses.

The Integrator Boot monitor program contains a routine which interrogates the SPD (Serial Presence Detect) EEPROM on a fitted DIMM and sets up the SDRAM controller parameters accordingly. If you

are not running the boot monitor, you will need to include your own code to perform this task. There is some example code in the user documentation for each core module, and a full description of the registers involved.

The hardware default settings for the SDRAM controller may vary between core module type.

The table below shows all supported configurations:

		SIZEOFSDRAM (MB)																														
		16				32							64								128								256			
# of Row addr bits	(SPD Byte 3)	11	11	12	11	11	12	12	13	11	11	12	12	12	13	13	11	12	12	13	12	12	13	13	12	13	13	12	13	13	13
# of Column addr bits	(SPD Byte 4)	8	8	8	9	8	8	8	8	9	9	9	10	8	8	8	9	9	9	9	10	10	10	8	9	9	9	10	10	10	9
# of CS Banks	(SPD Byte 5)	2	1	1	1	2	2	1	1	2	1	1	1	2	2	1	2	2	1	1	2	1	1	2	2	2	1	2	2	1	2
# of BA Banks	(SPD Byte 17)	2	4	2	2	4	2	4	2	2	4	2	2	4	2	4	4	2	4	2	2	4	2	4	4	2	4	4	2	4	4
# of BA bits		1	2	1	1	2	1	2	1	1	2	1	1	2	1	2	2	1	2	1	1	2	1	2	2	1	2	2	1	2	2
Total # of Rows,Cols,CS,BA bits		22	22	22	22	23	23	23	23	23	23	23	24	24	24	24	24	24	24	24	25	25	25	25	25	25	25	26	26	26	26

On some older CM designs, neither A12 nor A13 on the DIMM socket are connected to the CM FPGA, so for these modules, only DIMMs with a maximum of 12 row/column address lines can be used from the above compatibility table.

On newer CMs, A12 and A13 are connected, but currently, the A13 address line is always driven LOW by the FPGA. Hence the 256MB limit.

The following table shows which boards make connection to these address lines:

Board number	Name	A12, A13 routed to DIMM
HPI-0047B	CM9x0T	no
HPI-0058A	CM740T	no
HPI-0050B	CM720T	yes
HPI-0056B	CM7TDMI	yes
HPI-0066C,D	CM9x6E-S	yes
HPI-0070C	CM920T	yes
HPI-0070C	CM920T-ETM	yes
HPI-0087A,B	CM926EJ-S	yes
HPI-0067C	CM10200	yes

- ***What are the differences between the Integrator/CM920T core module types?***

ARM has produced a number of different variants of the CM920T core module. This FAQ attempts to clear up any confusion regarding the differences between each type.

Fundamentally, there are two PCB types on which CM920T core modules are based, namely HBI-0047B and HBI-0070C.

HBI-0047B
The original CM920T was the HBI-0047B. This core module type always has 256KB of SSRAM fitted, and has one Mictor connector which is connected to only a few local bus signals. This board makes use of the Xilinx XL4036XLA as its CM controller FPGA. This board is never fitted with an ETM-equipped test chip, since there is no facility to fit a trace connector to the board. Also, the '47B board cannot be used as part of an Integrator/CP system, since the FPGA is too small to incorporate all the peripherals.

HBI-0070C
Some time later, a new CM920T board was designed, HBI-0070C. This board is sometimes referred to as the 'CM920T-ETM', although boards have been produced both with and without ETM-equipped test chips.

This board has 1MB of ZBT SSRAM, and has either 4 or 5 Mictor connectors fitted. The 4 Mictors always present connect to the local ASB address, data, control buses, and to some other test chip and PLD pins. The optional, fifth Mictor connector is the trace connector, which may or may not be fitted, depending on whether the test chip is equipped with an ETM. This board design uses either the Xilinx XCV600, or XCV600E FPGA for its CM controller FPGA.

The '70C board can be used on an Integrator/CP platform if the FPGA configuration flash is programmed with the correct image. Images exist for boards with either XCV600 and XCV600E FPGAs. The latest board configuration files can be downloaded from the ARM website.

Versions of the FPGA image for the '70C board provide some extra memory mapped registers which are not present on the '47B board. (See table below, or CM920T-ETM User Guide for list of registers). This information is documented in the User Guides for these boards, but it is not highlighted as a specific difference between the two boards.

This has implications if software which has been developed on a '70C board, is subsequently run on a '47B board. If the software makes use of any of the extra registers provided on the '70C board, it will be necessary to modify this software to remove references to, or dependancies on these registers. In doing so, some of the extra functionality that is available with the '70C board will be lost.

The extra registers offered on the '70C board are:

Reg. Name	Address	Description
CM_AUXOSC	0x1000001C	Auxiliary oscillator control
CM_INIT	0x10000024	Core module initialisation register
CM_REFCNT	0x10000028	Reference clock cycle counter
CM_FLAGS	0x10000030	General purpose flags read register
CM_FLAGSET	0x10000030	Set register for the above
CM_FLAGSCLR	0x10000034	Clear register for the above
CM_NVFLAGS	0x10000038	General purpose non-volatile flags read register
CM_NVFLAGSET	0x10000038	Set register for the above
CM_NVFLAGSCLR	0x1000003C	Clear register for the above

- ***Core module generates 'wrong' AMBA cycle types on the system bus***

All Integrator core modules use an asynchronous bridge between the local memory bus (on which the core is located) and the system bus (to which the motherboard and/or your own IP may be attached).

The design of the core module bridges is such that the behavior you see on the system bus usually does not match that of the core itself. The burst types may be changed and the pattern of IDLE, BUSY, NONSEQ and SEQ cycles may also be lost.

Thus you should keep in mind when attempting to validate the operation of your IP by programming it into a logic module, that it will not be exposed to the same bus behavior as you will see in your final system.

All that is guaranteed is the sequence of transactions produced by the bridge will be legal AMBA transactions. Nothing else is guaranteed about what will be seen on the system bus, but we can summarize the behavior seen on current AHB based systems. However, if any of the bridge designs are updated, the following behavior may change, perhaps without notice.

CM7TDMI, CM720T, CM740T, CM920T, CM940T, CM920T-ETM
These core modules include the bridge described in their various user guides with two FIFOs that improve performance when writing through the bridge. All these CMs when operated in an AHB system will drive HBURST=0b001 when accessing memory. This is the INCR burst type and specifies a burst of unspecified length. Such a burst can be of any length (including one). Like all AHB bursts it must not cross 1KB boundaries, so this limits the maximum length. Unlike all other burst types INCR

conveys no additional meaning, in effect it tells the recipient device to look only at HTRANS for information on transfer type.

Customers working to validate their design often want to see other burst types being generated, for example INCR4 and INCR8. Unfortunately this is not possible with these modules.

When configured for use in an Integrator/AP-based system, the bridge in these core modules interconnects two buses that are completely asynchronous, hence part of the overhead of using this bridge is a non-deterministic synchronization penalty and as a result it is difficult to give a precise description of its behavior.

Performance will be significantly slower when accessing though the bridge, especially in the read direction. Thus, even if the core is performing a series of uninterrupted sequential accesses, once these emerge from the bridge you will find that significant additional cycles have been added.

This bridge will generate BUSY, IDLE, SEQ and NONSEQ type cycles.

We have provided some information on access timings for different memory areas within the Integrator system. This is available in a separate FAQ document. Please see the FAQ entry 'What clock speed do the boards operate at?' for details.

CM946E-S, CM966E-S, CM926EJ-S
These core modules were implemented with a different bridge design to the core modules described above. This bridge does not include a FIFO between the memory bus (ARM core bus) and the system bus. This was not documented properly in the original CM9x6E-S user guide. (See FAQ 'Lack of system bus FIFO in some core modules' for further details.)

These CMs will always drive HBURST = `0b000` when accessing memory. This is a single transfer. Since we can only perform single accesses, this forces us to never use a sequential (SEQ) access.

Again this bridge is asynchronous, but the lack of FIFO means that both reads and writes to the system bus will be slow.

A significant number of IDLE cycles will be seen on the system bus between actual transfers. BUSY cycles are not generated by the current version.

CM922T-XA10
This CM's system bus bridge can only generate transactions of type HBURST = `0b000` ('SINGLE').

CM10200, CM10200E
These core modules can show two burst types on the system bus – 'SINGLE' and 'INCR'. INCR will be generated when the ARM1020 core performs a 64-bit bus access, which is translated into two 32-bit accesses on the system bus by the core module bridge. Any 32-bit access by the ARM core will generate a SINGLE type transfer.

Integrator/CP
Core modules used on the Integrator/CP make use of the ADK (AMBA Development Kit) bridge, which does not contain a FIFO. This bridge does correctly propagate HBURST information, however.

- ***How can I maximize performance across the system bus bridge?***

  This FAQ entry has been amalgamated into the new 'Integrator Performance FAQ' document. See 'What clock speed do the boards operate at?' for details.

- ***Further guidelines on configuring core module clocks***

  The contents of this section have been moved to the new Integrator Performance FAQ document. Please see main FAQ entry 'What clock speed do the boards operate at?' for details.

- ***Lack of system bus FIFO in some core modules***

  This FAQ entry has been amalgamated into the new 'Integrator Performance FAQ' document. See FAQ 'What clock speed do the boards operate at?' for details.

- ***New features retrofitted to existing core modules***

  Newer Integrator core modules contain some additional functionality compared to the older modules. Where possible these new features have been retrofitted to older core modules in newer versions of the CM controller FPGA design.

  In general, these features were added after the documentation was released, so it is unlikely that this functionality is described in your user guide.

  SSRAM alias
  Originally the SSRAM could only appear once in the memory map, at address zero. Newer FPGA images also place this memory at an alias address of `0x10800000`. Unlike the image at address zero, the SSRAM is always visible here regardless of the state of the REMAP bit. Since 8MB is allocated for this alias (`0x10800000 to 0x10FFFFFF`), the SSRAM will be aliased multiple times across this address range.

  The introduction of this particular feature is documented in the release notes supplied with FPGA (or 'boardfile') updates.

  Core module local memory bus clock cycle counter
  The register CM_LMBUSCNT increments at the speed of the local memory bus. This register is set to zero at reset but otherwise can not have its value changed. This size of this counter is usually 16 bits and is located at address `0x10000018`.

  Core module reference clock cycle counter
  The register CM_REFCNT increments at the speed of the reference clock, 24MHz. This register is set to zero at reset but otherwise can not have its value changed. This size of this counter is usually 32 bits and is located at address `0x10000028`.

  Which core modules have these features?
  The table below summarises which core modules had these features from the outset, and which you will need to update to the 'current' version of the FPGA to obtain this functionality. Unfortunately, we cannot specify a minimum FPGA build number for each feature at this time.

  If a core module is not listed below then these features should be documented in the core module's user guide.

Core Module		SSRAM alias		CM_LMBUSCNT		CM_REFCNT	
		Older	Current	Older	Current	Older	Current
CM7TDMI	HBI-0056	No	Yes	No	16bit	No	32bit
CM720T	HPI-0050	No	Yes	16bit	16bit	No	No
CM740T	HBI-0058	No	Yes	16bit	16bit	No	No
CM920T	HBI-0047	No	Yes	16bit	16bit	No	No
CM920T	HBI-0070	Yes	Yes	16bit	16bit	32bit	32bit
CM920T-ETM	HBI-0070	Yes	Yes	16bit	16bit	32bit	32bit
CM922T-XA10	HBI-0100	n/a	No	n/a	No	n/a	32bit
CM926EJ-S	HBI-0087	n/a	Yes	n/a	32bit	n/a	32bit
CM940T	HBI-0047	No	Yes	16bit	16bit	No	No
CM946E-S	HBI-0066	Yes	Yes	32bit	32bit	32bit	32bit
CM966E-S	HBI-0066	Yes	Yes	32bit	32bit	32bit	32bit
CM10200	HBI-0067	Yes	Yes	16bit	16bit	32bit	32bit
CM10200E	HBI-0098	n/a	Yes	n/a	32bit	n/a	32bit

- ***Problems running Thumb code in Big Endian on CM946E-S***

  This issue only affects some CM946E-S boards (HBI-0066x) manufactured in 2002 (serial numbers starting 2xxxxxxx. The markings on the affected test chip are:

  NEC JAPAN
  ARM946E-S R1
  020412902 *(Or very similar).*
  UX4L

  These boards were manufactured with a test chip that contained a peculiarity in the ARM CPU which causes BL instructions executed in Thumb state and in Big Endian mode to load an incorrect value into the PC (R15). This will usually cause a program crash, which may be denoted by a number of possible symptoms, e.g: Spurious data aborts, Undefined Instructions or Unrecognised SWIs being reported by the debugger.

  If your software development is affected by this issue, the only solution is to obtain a core module with a different test chip type fitted. Please contact ARM Support for further details, quoting this FAQ.

  Note that if the board is exchanged for one with a different test chip, there is no guarantee that the replacement board will have the same size TCMs, caches or ETM. (See FAQ: ‘Size of TCM / cache memory present in ARM test chips’.)

5. Questions Specific to Logic Modules

- ***How do I program the FPGA on a logic module (or logic tile)?***
- ***Multi-ICE needed to program logic module (or logic tile) FPGA configuration flash***

  There are fundamentally two ways of programming the FPGA on an ARM logic module or logic tile – Directly, or Indirectly.

  The direct method results in a volatile configuration, which is lost when the system is powered down, since the FPGAs are RAM-based. The FPGA on the module can be directly programmed using ARM's Multi-ICE JTAG interface unit and a software utility called Progcards, or programming can be done using the FPGA manufacturer's proprietary JTAG programming tool and software. Please refer to your module user guide for instructions on jumper settings, etc.

  The indirect method involves programming FPGA configuration binaries into the module's flash memory. A selected configuration image is then loaded from flash into the FPGA every time the board is powered on, or in the case of logic tiles, also optionally at the press of the reload button. The FPGA configuration flash can only be programmed using ARM's Multi-ICE unit. ARM's RVI (RealView ICE) or any 3rd party JTAG interface units are not suitable, since the Progcards board programming utility only works with the Multi-ICE software.

- ***Can't program Altera logic module when stacked on top of a CP system***
- ***Can't program more than one Altera logic module in a stack of boards***

  If a LM-EP20K600E+ logic module is placed in a stack of boards, it may not be possible to program the logic module's FPGA or configuration flash memory, depending on which other boards are in the stack. This problem is due to the JTAG signal quality on the LM becoming degraded to the point where JTAG communication is not possible.

  There are several workarounds:

  1. Temporarily remove the logic module to be programmed from the stack and program it stand-alone. When programming is complete, remove the CONFIG jumper and replace the board in the stack. Obviously, it is only possible to program the module's FPGA configuration flash in this manner, since any configuration programmed directly into the FPGA will be lost when the board is powered down to refit it to the stack.

     If much re-programming of the FPGA configuration flash is required, then this solution will not only be inconvenient, but damage to the board connectors will likely soon result. One of the following, more permanent solutions is therefore recommended.

  2. In some situations, for example: If there are only two boards in the stack above the motherboard, it might be possible to improve signal quality sufficiently by increasing the value of the series termination resistors used in the TCK path on the logic module. This is described in greater detail in Application Note 101, 'Stacking Integrator Modules'. If this does not work in your particular situation, it will be necessary to buffer the TCK signal, as described below.

  3. Buffer the JTAG TCK signal where it enters the logic module. This can either be done by adding extra buffers to the board, or by making use of the unused gates in the discrete logic ICs on the module. This is described in detail in Application Note 101, 'Stacking Integrator Modules'.

- ***Can't configure FPGAs directly when more than one logic module in a stack***

  As described in the logic module User Guides, it is possible to either program logic module FPGAs directly, or to program the FPGA configuration flash. ARM's Multi-ICE unit can be used to perform both these tasks. The FPGA vendors' own programming tools can only be used to program the FPGA directly.

When a stack of modules is used, it is not possible to program designs directly into the FPGAs, since the configuration pins on all FPGAs in the stack are not driven independently. Consequently, when any one device is initialized at the start of its programming sequence, the other devices in the stack are also cleared, and any programming is lost.

The only solution is to program the FPGA configuration flash memory on all modules in the stack, and set the DIP switches to load the required configurations. It will be necessary to write a custom board file (.brd file extension) to make the Progcards utility program each flash device in turn. When the programming procedure is complete, the CONFIG jumper can be removed from the module at the top of the stack. When power is cycled, all modules will load configurations from their respective flash memories.

- ***What speed grade FPGAs are fitted to your logic modules?***
- ***What is the exact part number of the FPGA fitted to your logic module?***

The following table lists all the logic modules and interface modules with FPGAs that ARM has produced and sold, along with the exact part number and speed grade of the FPGA fitted to each board:

Board type	Part No.	FPGA part no.	Speed grade	
Integrator/LM-XCV400+	HBI-0059	XCV1000 BG560AFP0013 D2042531A	-5	Range: -4, -5, -6. (-6 is fastest)
Integrator/LM-XCV600E+	HBI-0072	XCV2000E FG680AMS0101 D2062267A	-6	Range: -6, -7, -8. (-8 is fastest)
Integrator/LM-EP20K600E+	HBI-0073	EP20K1000EFC672-2 N ABA720043D	-2	Range: -1, -2. (-1 is fastest)
Integrator/IM-LT2	HBI-0106	XCV600E-6 FG680C	-6	Range: -6, -7, -8. (-8 is fastest)