
PGT302 – Embedded Software Technology

PART 2

Hardware Platform 1

Objectives for Part 2

- Need to DISCUSS and ANALYZE the following topics:
 - Board (Raspberry Pi) specifications
 - startup sequence, bootloader
 - Microcontroller (ARM) details
 - Features: I/O, Timer, Interrupt
- Need to ANALYZE some simple C codes
 - I/O access
- Need to DEVELOP some simple C codes
 - I/O access

Raspberry Pi

- SBC to promote computer science in schools
 - By Raspberry Pi Foundation (<http://raspberrypi.org>)
 - “... computer to inspire children ...”
- Low cost, yet huge features
 - \$25 - \$35 SBC boards with graphics engine
- Commercial Facts:
 - First batch (10,000 units) were sold before made!
 - (October 2014) 3.8 million boards sold
- Main models (as of 20160902)
 - Pi 1 (Model A,A+,B,B+), Pi 2, Pi 3
 - Pi Compute Module, Pi Zero!

Platform Specifications

- Raspberry Pi 1
 - Broadcom's BCM2835 SoC package on RAM
 - CPU: 700 MHz ARM1176JZF-S (ARM11 core, ARMv6 ISA)
 - GPU: 250 MHz Broadcom VideoCore IV (with OpenGL engine and MPEG codecs)
 - RAM: 256MB (model A/A+) or 512MB (model B/B+) SDRAM
 - USB Hub, HDMI AV output
 - Composite video (PAL/NTSC – not since B+)
 - Analog audio and I²S (Inter-IC / Integrated Interchip – not related to I²C)

Platform Specifications (cont.)

- Raspberry Pi 1 (cont.)
 - Ethernet (beginning model B) USB adapter using SMSC lan9514-jzx
 - GPIO with some specific functions (UART, SPI, I2C, I2S)
 - 5V powered (micro-USB/GPIO header)
 - 300mA@1.5W (Model A), 200mA@1W (Model A+), 700mA@3.5W (Model B), 600mA@3.0W (Model B+)
 - On-board storage: (micro-)SD card (boot media)

Startup sequence @bootstrap

- GPU boots stage 1 bootloader (on-chip ROM)
 - can access FAT on (micro)SD card
 - fetches stage 2 bootloader (bootcode.bin)
- GPU loads bootcode.bin (on-chip L2 cache)
 - (older) fetches stage 3 bootloader (loader.bin) for ELF utils, then fetches
 - (newer) ELF utils included
 - looks for configurable bootloader (start.elf)

Startup sequence @bootstrap (cont.)

- GPU executes start.elf (RAM)
 - a.k.a. VideoCore OS (stays on to manage GPU)
 - prepare/split memory for GPU/CPU use
 - parses config.txt and cmdline.txt
 - (default) fetches kernel.img (ARM binary)
 - load @0x8000 and release reset on ARM core

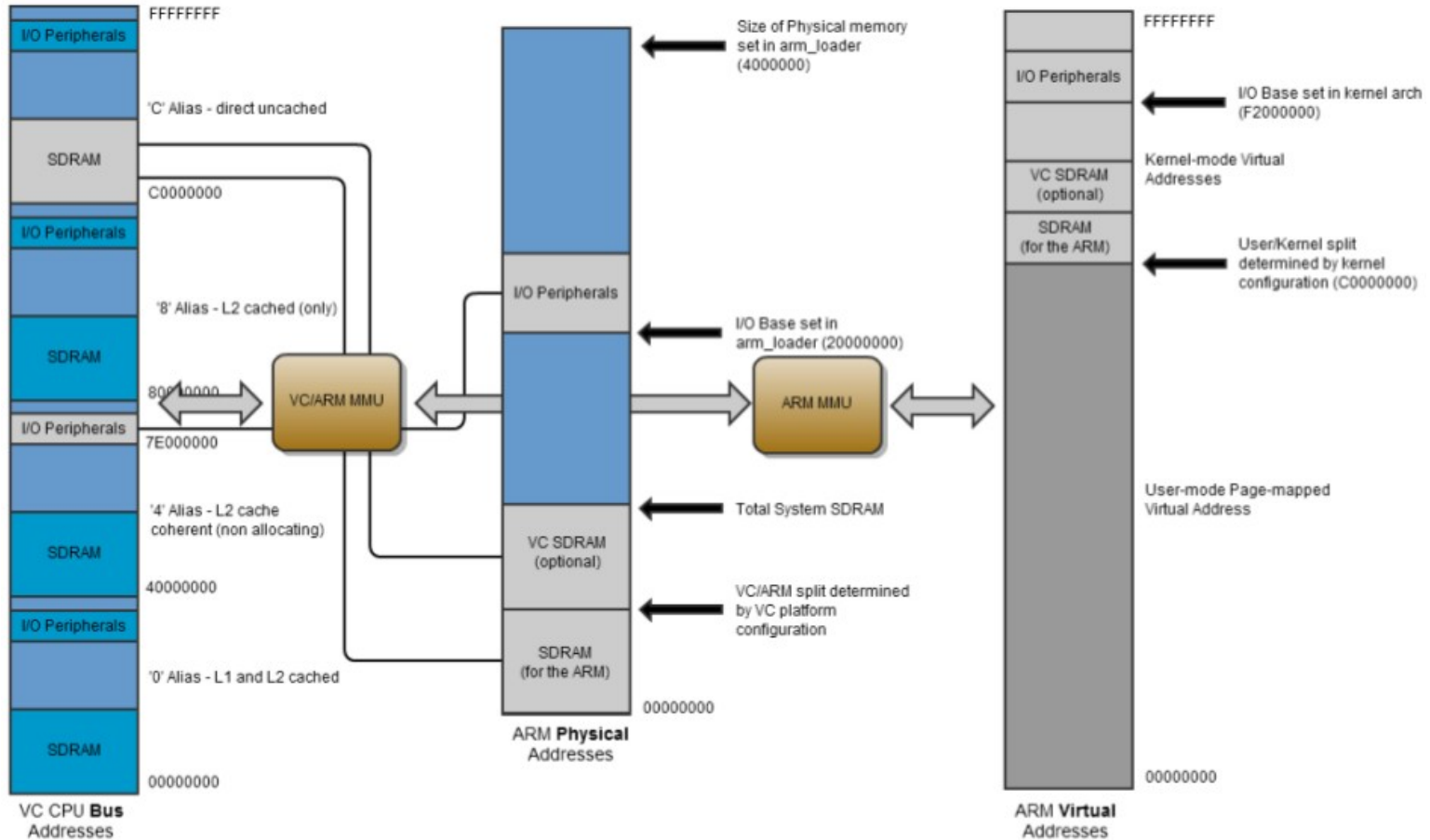
Microcontroller Details

- ARM1176JZF-S (ARM11 core, ARMv6 ISA)
 - 32-bit ARM11 integer core (no floating-point h/w)
 - Instruction/data MMU, supports AMBA AXI
 - 8-stage pipeline, branch prediction
 - TrustZone security (adds Secure Mode)
 - ARMv6 ISA with 16-bit Thumb instructions
 - supports Jazelle technology (Java bytecode)
- BCM2835 specifications
 - shared RAM with GPU (configurable size)

BCM2835 Address Map



BCM2835 ARM Peripherals



BCM2835 ARM Addresses

- Virtual Address
 - standard Linux kernel (RAM at top memory space)
 - kernel mode range [0xC0000000-0xEFFFFFFF]
 - user mode range [0x00000000-0xBFFFFFFF]
 - peripherals mapped to 0xF2000000 (base)
- Physical Address
 - RAM at 0x00000000 (size depends on start.elf)
 - peripherals mapped to 0x20000000 (base)

Thumb Instructions

- 16-bit instructions subset of the full ISA
 - has equivalent 32-bit version
 - thus, still operates on 32-bit core
 - 32-bit address space, 32-bit registers, 32-bit shifters & ALU, 32-bit memory transfer
 - higher performance than 16-bit architecture
 - higher code density than 32-bit architecture
- ARM1176JZF-S can easily change state
 - ARM state ↔ Thumb state
 - can also execute Java bytecodes (variable length)

Operating Modes

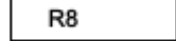

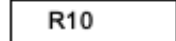
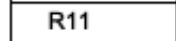
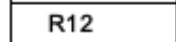
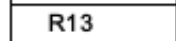
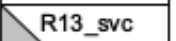
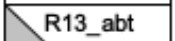
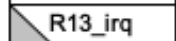
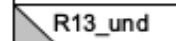
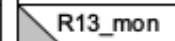
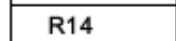
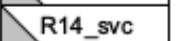
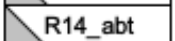
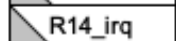
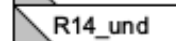
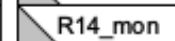
- ARM1176JZF-S has 8 operating modes
 - User (ARM): unprivileged common tasks
 - FIQ (ARM): privileged high priority (fast) interrupt
 - IRQ (ARM): privileged low priority (normal) interrupt
 - Supervisor (ARM): privileged on reset @ software interrupt [protected mode?]
 - Abort (ARM): privileged handler for memory access violations
 - Undef (ARM): privileged handler for undefined instructions
 - System (ARMv4): privileged user mode
 - Secure Monitor (TrustZone): privileged secure mode

Registers



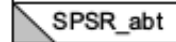
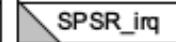

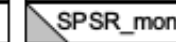
- ARM1176JZF-S has 40 registers (ARM state)
 - 33 x 32-bit general purpose registers
 - 7 x 32-bit status registers
- Register organization (diagram on next slide)
 - 16 general purpose (R0-R15)
 - 1@2 status registers (CPSR & SPSR)
 - R13 @ Stack Pointer
 - R14 @ Link Register
 - R15 @ Program Counter
 - SPSR available in privileged, non-user mode

Registers (cont.)

ARM state general registers and program counter

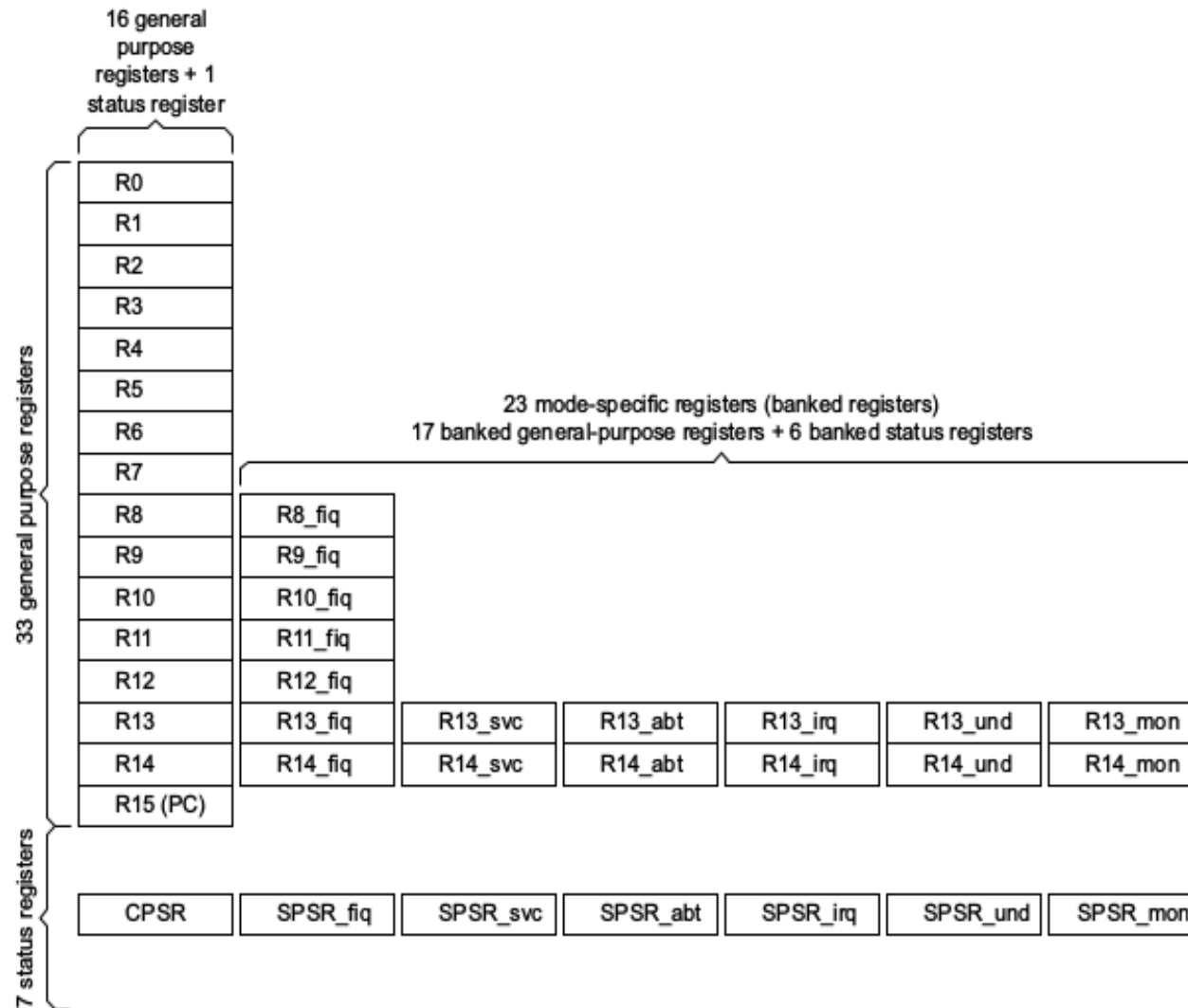
System and User	FIQ	Supervisor	Abort	IRQ	Undefined	Secure monitor
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	 R8_fiq	R8	R8	R8	R8	R8
R9	 R9_fiq	R9	R9	R9	R9	R9
R10	 R10_fiq	R10	R10	R10	R10	R10
R11	 R11_fiq	R11	R11	R11	R11	R11
R12	 R12_fiq	R12	R12	R12	R12	R12
R13	 R13_fiq	 R13_svc	 R13_abt	 R13_irq	 R13_und	 R13_mon
R14	 R14_fiq	 R14_svc	 R14_abt	 R14_irq	 R14_und	 R14_mon
R15	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

ARM state program status registers

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	 SPSR_fiq	 SPSR_svc	 SPSR_abt	 SPSR_irq	 SPSR_und	 SPSR_mon

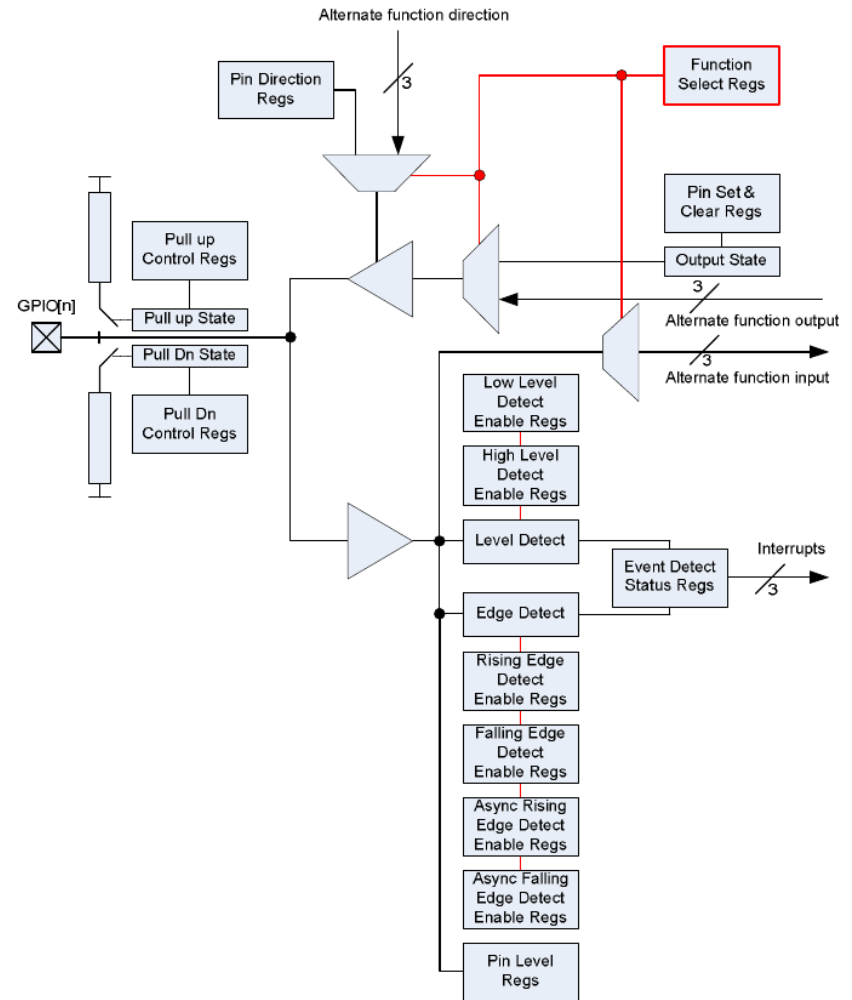
 = banked register

Registers (cont.)



BCM2835 GPIO

- 54 general purpose I/O GPIO
 - configured using 41 registers, split into 2 banks
 - each GPIO has at least 2 alternate functions
 - alternate functions may appear on both banks
 - pull-`{up,down}` states



BCM2835 GPIO (cont.)

- GPIO peripherals address
 - documented as 0x7E200000 (VC address)
 - accessed at 0x20200000 (ARM Physical Address)
- Commonly used registers
 - GPFSEL (x 6) → Function Select (R/W)
 - GPSET (x 2) → Output SET (W)
 - GPCLR (x 2) → Output CLR (W)
 - GPLEV (x 2) → Level (R)
 - others are for event detections (edges, levels, etc.)

BCM2835 Timer

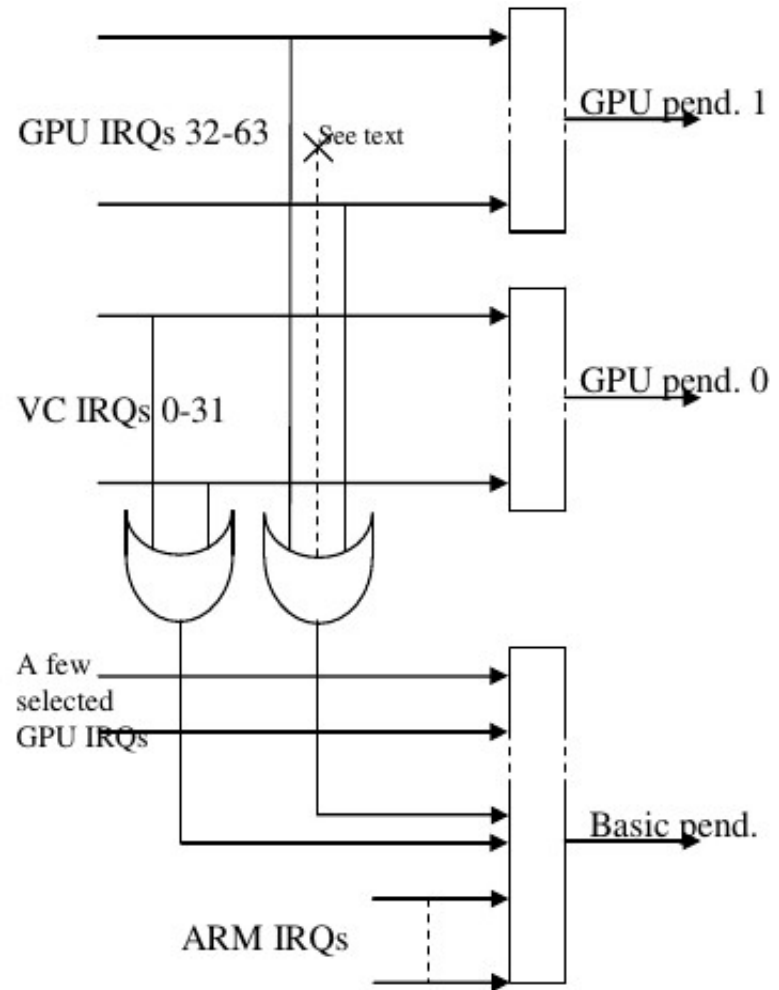
- System timer
 - 4 x 32-bit timer channels, 1 x 64-bit free-running counter (stopped in debug mode)
 - each timer channel has 32-bit comparator (against lower 32-bit free-running) → flagged when match
 - documented address @0x7E003000 (7 x 32-bit registers)
- ARM timer
 - 1 x 32-bit timer, 1 x 32-bit free-running
 - based on ARM AP804 (but different)
 - clock derived from system clock (will change in low power mode → for accuracy, use system timer!)

BCM2835 Interrupt

- Two interrupt sources:
 - GPU peripherals
 - local ARM peripherals
- ARM specific interrupts:
 - 1 x timer
 - 1 x mailbox (NOT FOR GENERAL USE)
 - 2 x doorbells (NOT FOR GENERAL USE)
 - 2 x GPU halted interrupts
 - 2 x Address/access error interrupts
- Each interrupt source has an enable bit (RW) and an interrupt pending bit (RO)

BCM2835 Interrupt (cont.)

- Interrupt pending bit
 - no vector module
 - pending bits organized as shown
 - 1 x 32-bit basic pending
 - 2 x 32-bit GPU pending
- NO interrupt priority
 - use FIQ
 - resolved by software handler



BCM2835 Interrupt (cont.)

- Interrupt registers @ 0x7E00B000 (base)
 - 10 x 32-bit registers
- Effective base address for ARM @0x2020B200

Address offset ⁷	Name
0x200	IRQ basic pending
0x204	IRQ pending 1
0x208	IRQ pending 2
0x20C	FIQ control
0x210	Enable IRQs 1
0x214	Enable IRQs 2
0x218	Enable Basic IRQs
0x21C	Disable IRQs 1
0x220	Disable IRQs 2
0x224	Disable Basic IRQs

BCM2835 Bare metal Programming

- No standard libraries, core programming
 - no stdio! no stdlib!
 - a lot of pointers usage!
 - build our own custom 'library'!
- For more details, refer to
 - <http://github.com/azman/my1barepi>

The End of Part 2