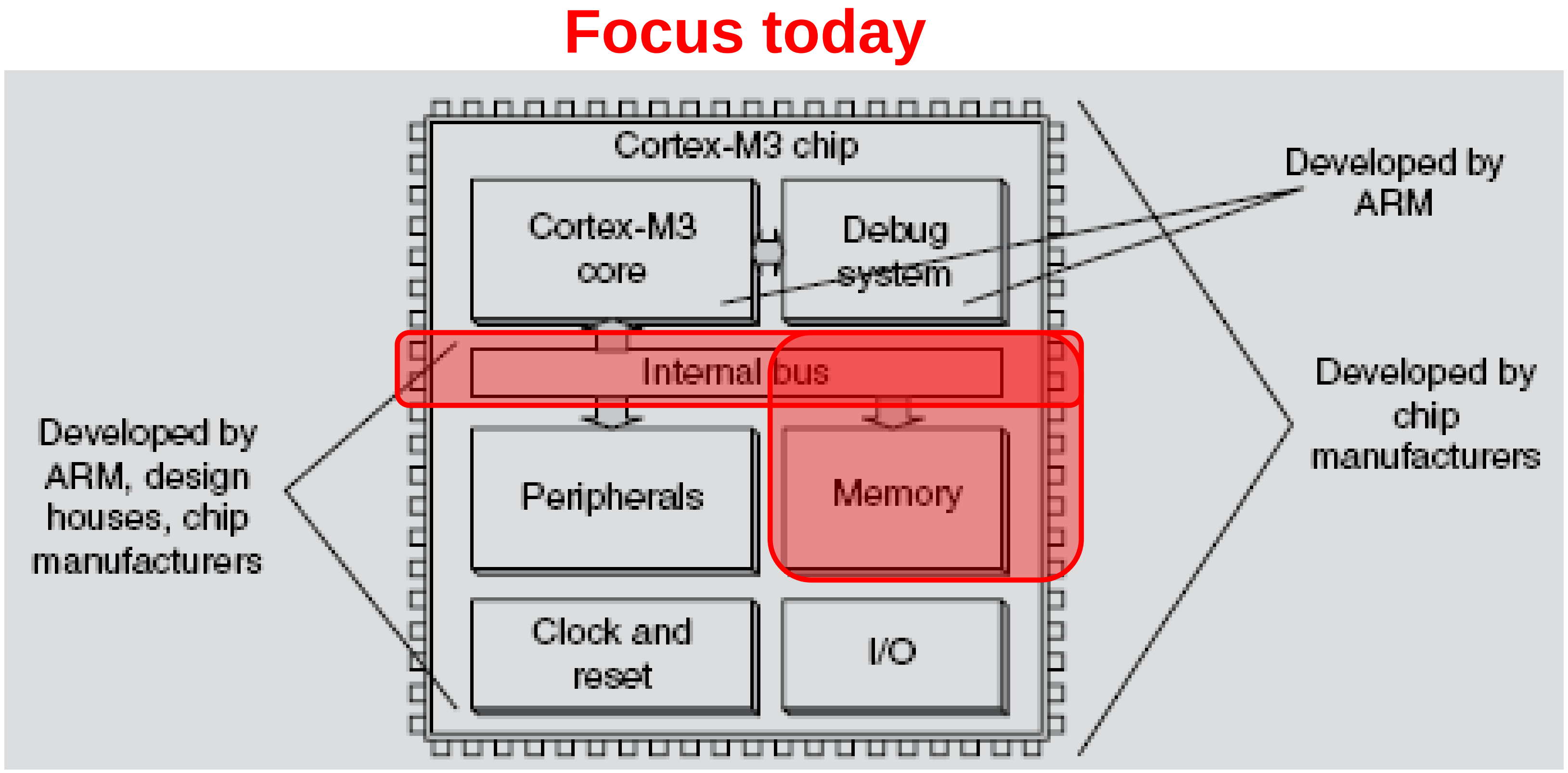
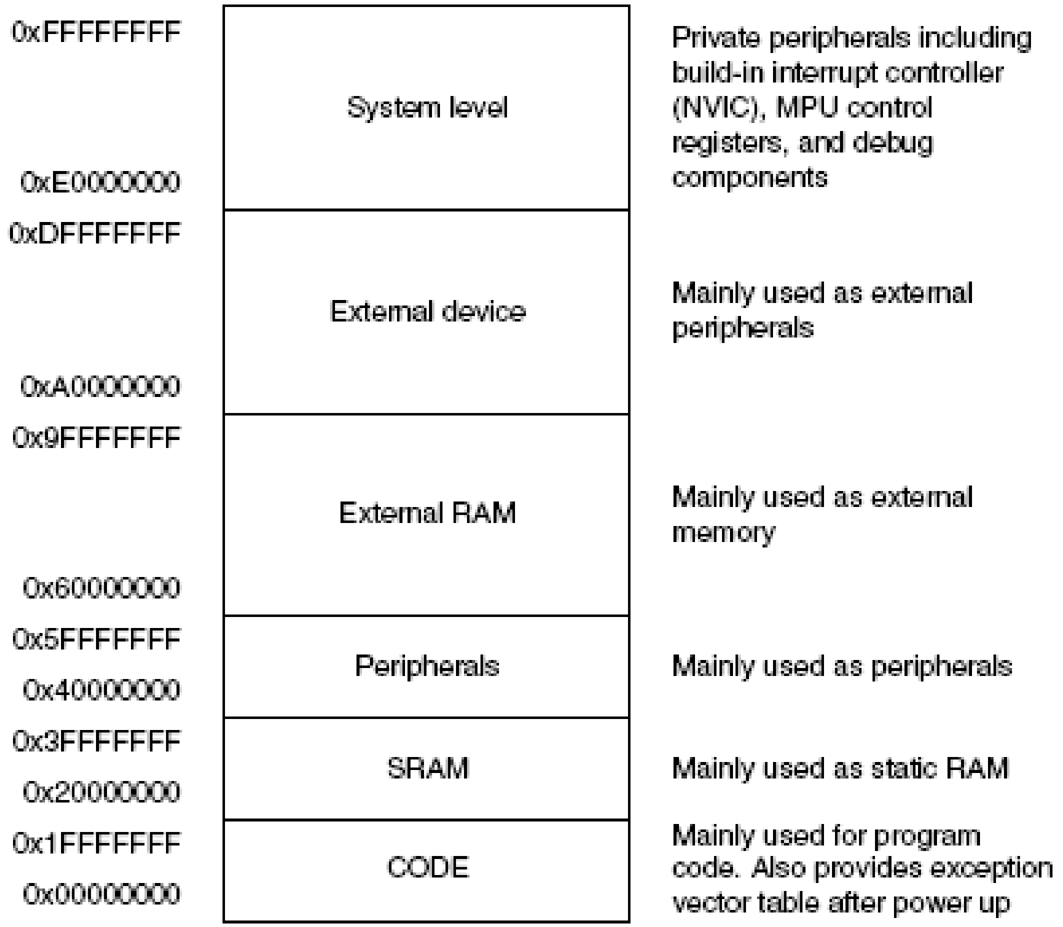
ARM Cortex-M3 Buses

Processor vs. MCU



Memory map

- Statically defined memory map (faster addr decoding) 4GB of address psace


0xFFFFFFFF
System level
Private peripherals including build-in interrupt controller (NVIC), MPU control registers, and debug components
0xE0000000
0xDFFFFFFF
External device
Mainly used as external peripherals
0xA0000000
0x9FFFFFFF
External RAM
Mainly used as external memory
0x60000000
0x5FFFFFFF
Peripherals
Mainly used as peripherals
0x40000000
0x3FFFFFFF
SRAM
Mainly used as static RAM
0x20000000
0x1FFFFFFF
CODE
Mainly used for program code. Also provides exception vector table after power up
0x00000000

Pipeline

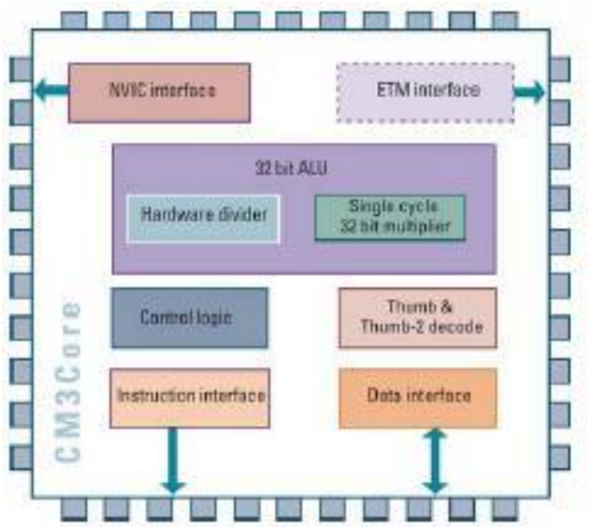
Harvard architecture

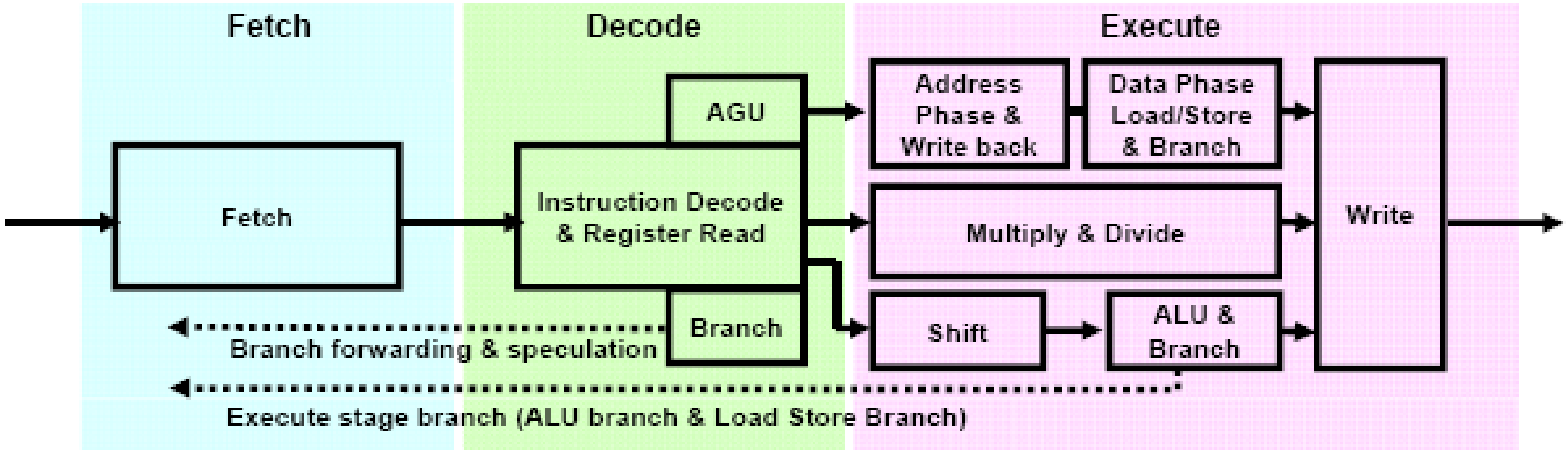
- Separate Instruction & Data buses enable parallel fetch & store

Advanced 3-Stage Pipeline

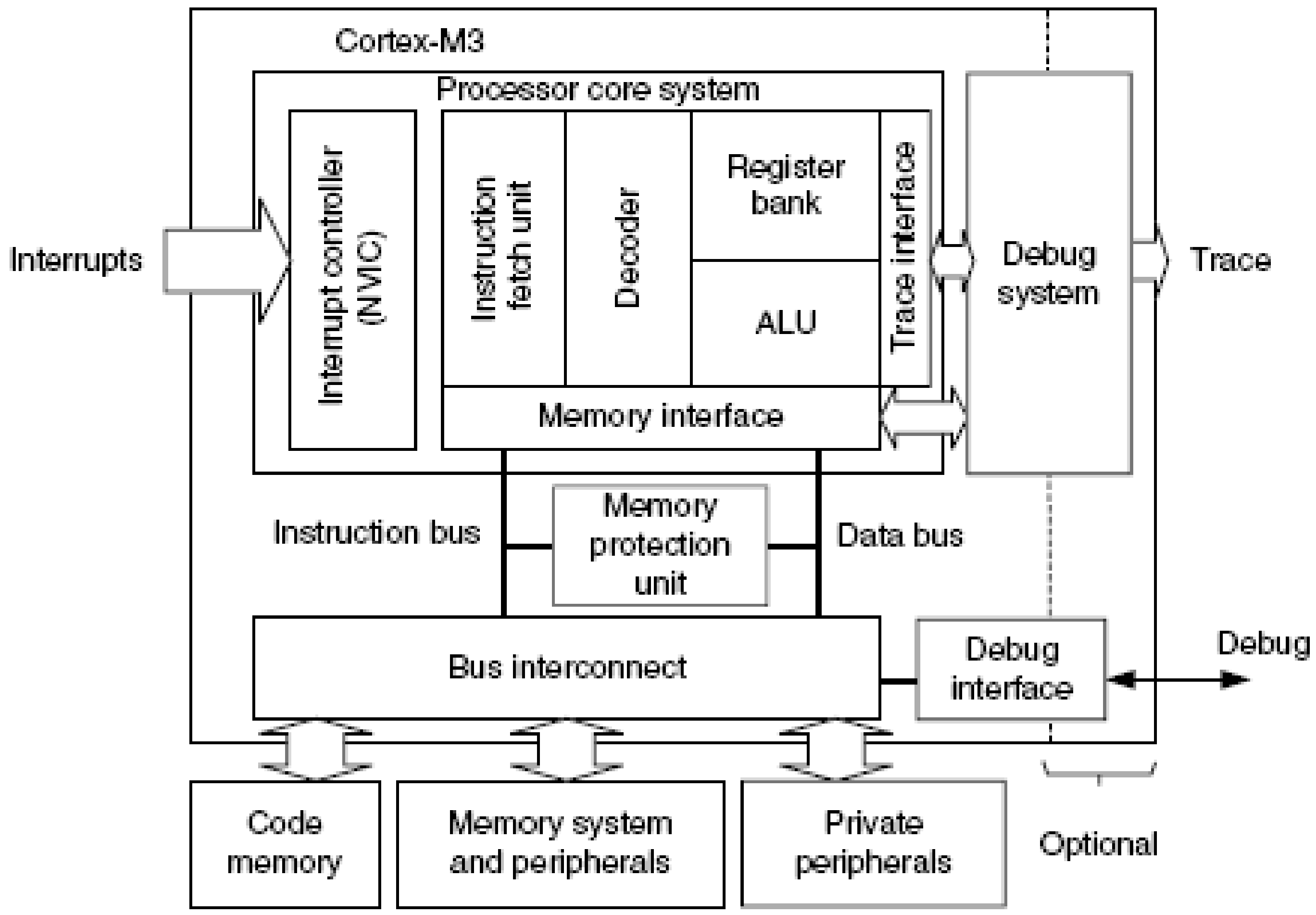
- Includes Branch Forwarding & Speculation

Additional Write-Back via Bus Matrix




Fetch
Decode
Execute
Fetch
AGU
Instruction Decode & Register Read
Branch
Address Phase & Write back
Data Phase Load/Store & Branch
Multiply & Divide
Shift
ALU & Branch
Write
Branch forwarding & speculation
Execute stage branch (ALU branch & Load Store Branch)

Core Architecture Diagram



Accessing memory locations from C

- Memory has an address and value
- Can equate a pointer to desired address
- Can set/get de-referenced value to change memory

```
#define  SYSREG_SOFT_RST_CR  0xE0042030

uint32_t *reg = (uint32_t *)(SYSREG_SOFT_RST_CR);

main () {
  *reg |= 0x00004000; // Reset GPIO hardware
  *reg &= ~(0x00004000);
}
```

Some useful C keywords

- const
  - Makes variable value or pointer parameter unmodifiable
  - const foo = 32;
- register
  - Tells compiler to locate variables in a CPU register if possible
  - register int x;
- static
  - Preserve variable value after its scope ends
  - Does not go on the stack
  - static int x;
- volatile
  - Opposite of const
  - Can be changed in the background
  - volatile int I;

What happens when this “instruction” executes?

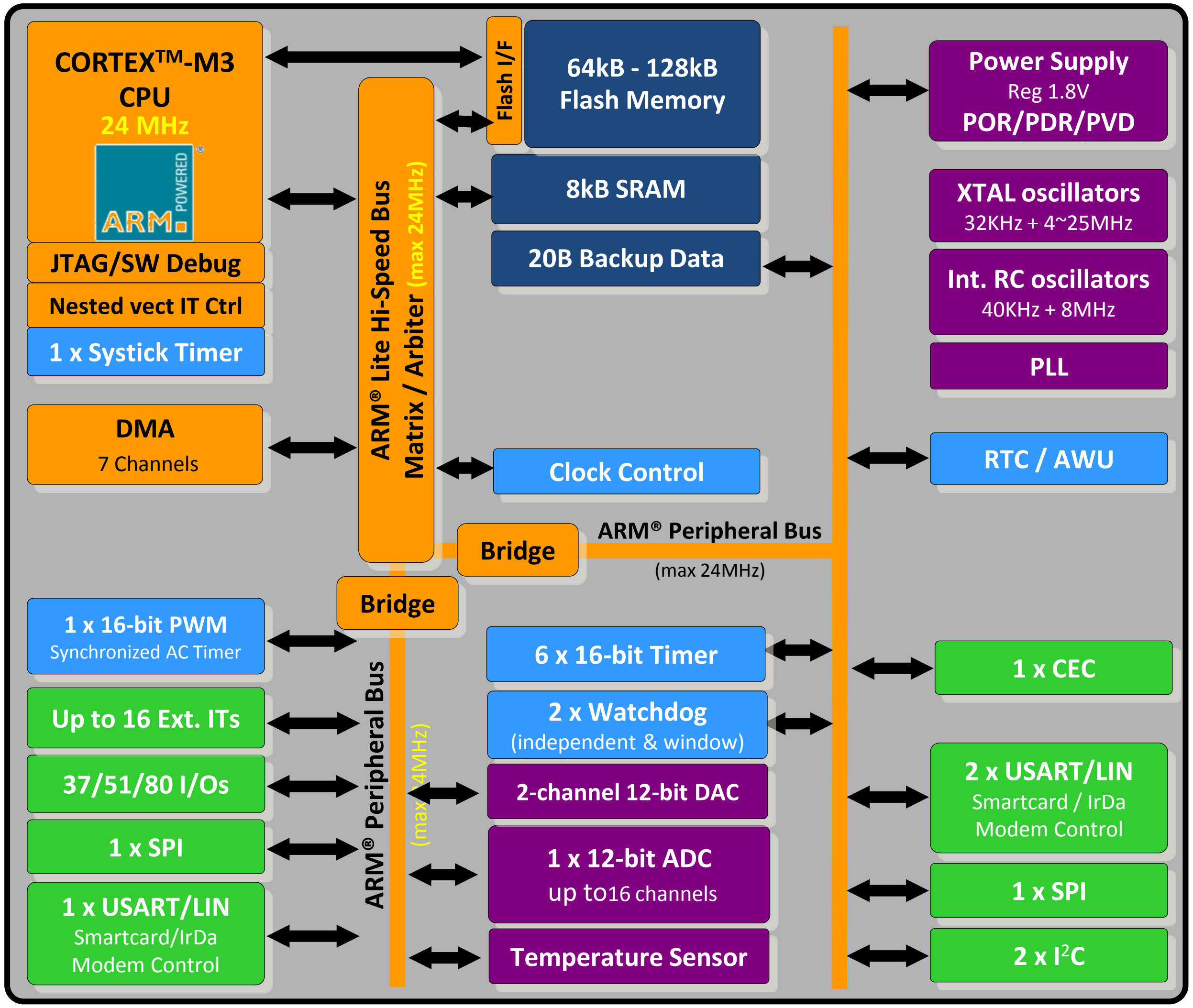
```
#include <stdio.h>
#include <inttypes.h>

#define REG_FOO 0x40000140

main () {
  uint32_t *reg = (uint32_t *)(REG_FOO);
  *reg += 3;

  printf(“0x%x\n”, *reg); // Prints out new value
}
```

"*reg += 3" is turned into a ld, add, str sequence

- Load instruction
  - A bus read operation commences
  - The CPU drives the address "reg" onto the address bus
  - The CPU indicated a read operation is in process (e.g. R/W#)
  - Some "handshaking" occurs
  - The target drives the contents of "reg" onto the data lines
  - The contents of "reg" is loaded into a CPU register (e.g. r0)
- Add instruction
  - An immediate add (e.g. add r0, #3) adds three to this value
- Store instruction
  - A bus write operation commences
  - The CPU drives the address "reg" onto the address bus
  - The CPU indicated a write operation is in process (e.g. R/W#)
  - Some "handshaking" occurs
  - The CPU drives the contents of "r0" onto the data lines
  - The target stores the data value into address "reg"

STM32 Value line 64K-128KBytes System Diagram

- **Core and operating conditions**
  - ARM® Cortex™-M3
  - 1.25 DMIPS/MHz up to 24 MHz
  - 2.0 V to 3.6 V range
  - -40 to +105 °C
- **Rich connectivity**
  - 8 communications peripherals
- **Advanced analog**
  - 12-bit1.2 µs conversion time ADC
  - Dual channel 12-bit DAC
- **Enhanced control**
  - 16-bit motor control timer
  - 6x 16-bit PWM timers
- **LQFP48, LQFP/BGA64, LQFP100**



On-Chip Buses

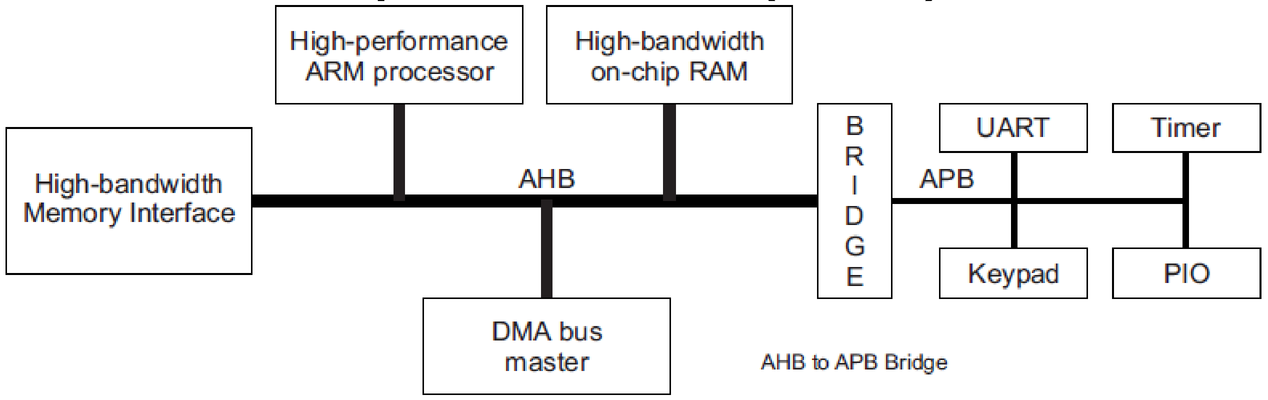
Why have so many busses?

- Many designs considerations
  - Master vs Slave
  - Internal vs External
  - Bridged vs Flat
  - Memory vs Peripheral
  - Synchronous vs Asynchronous
  - High-speed vs low-speed
  - Serial vs Parallel
  - Single master vs multi master
  - Single layer vs multi layer
  - Multiplexed A/D vs demultiplexed A/D

- Discussion: what are some of the tradeoffs?

Advanced Microcontroller Bus Architecture (AMBA)

- Advanced High-performance Bus (AHB)
- Advanced Peripheral Bus (APB)


High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth Memory Interface
AHB
DMA bus master
BRIDGE
APB
UART
Timer
Keypad
PIO
AHB to APB Bridge

AHB

- High performance
- Pipelined operation
- Burst transfers
- Multiple bus masters
- Split transactions

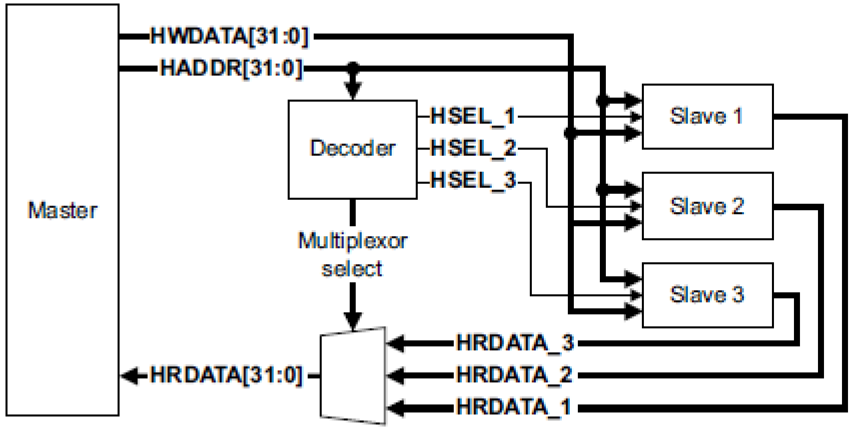
APB

- Low power
- Latched address/control
- Simple interface
- Suitable of many peripherals

AHB-Lite

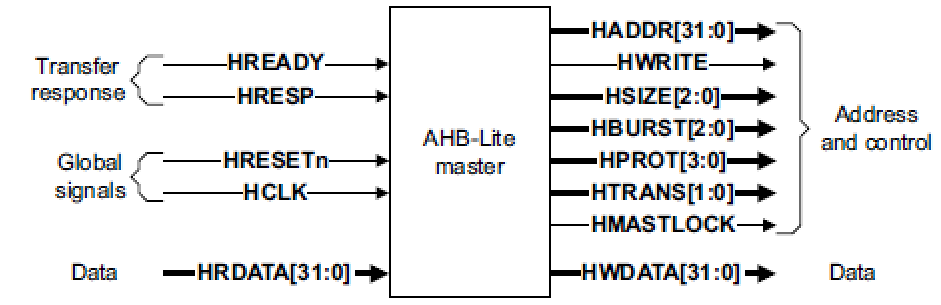
Supports single bus master and provides high-bandwidth operation

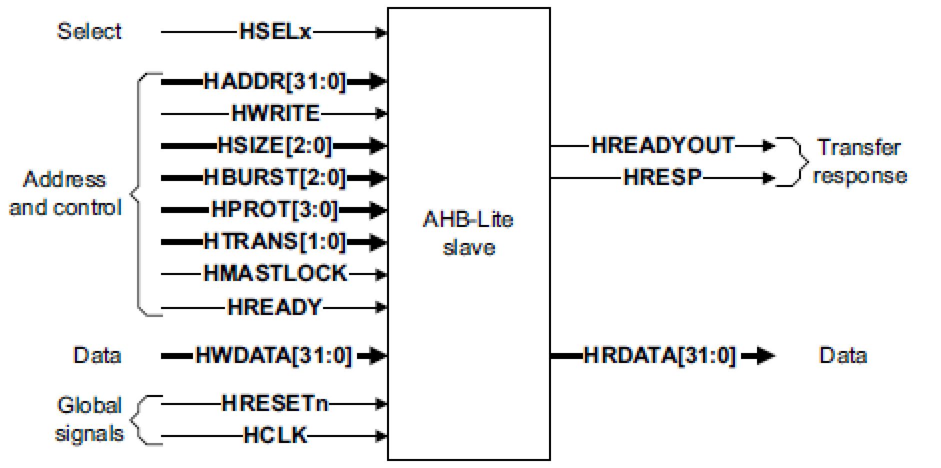
- Burst transfers
- Single clock-edge operation
- Non-tri-state implementation
- Configurable bus width



AHB-Lite bus master/slave interface

- Global signals
  - HCLK
  - HRESETn
- Master out/slave in
  - HADDR (address)
  - HWDATA (write data)
  - Control
    - HWRITE
    - HSIZE
    - HBURST
    - HPROT
    - HTRANS
    - HMASTLOCK
- Slave out/master in
  - HRDATA (read data)
  - HREADY
  - HRESP





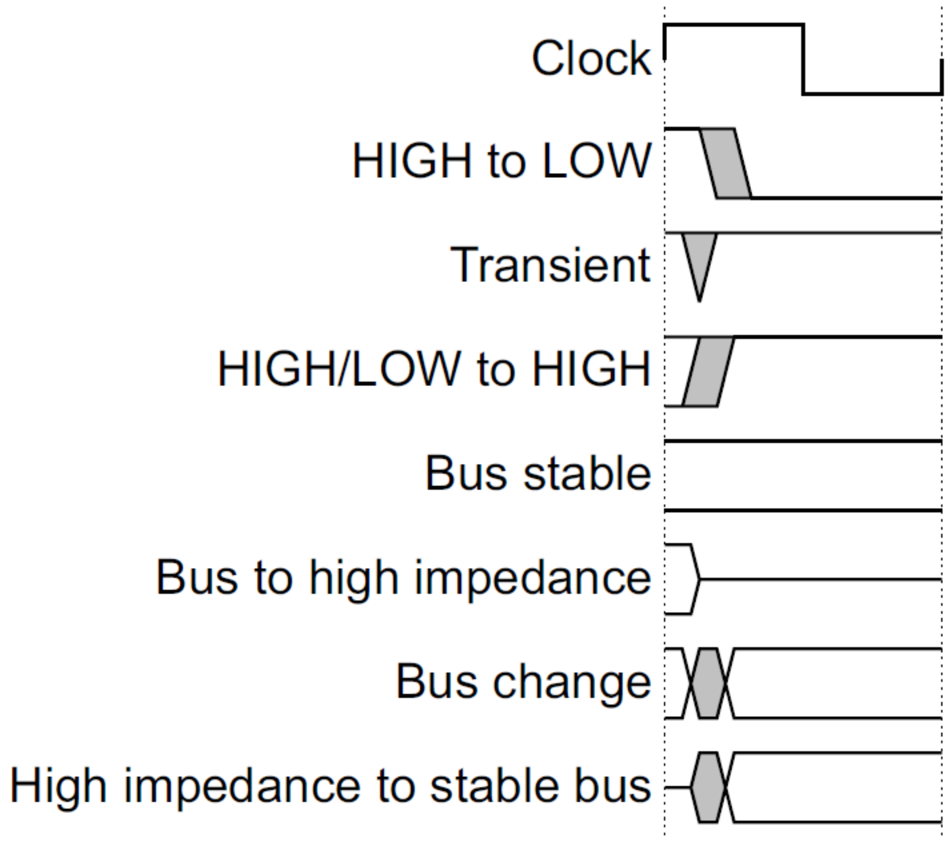
AHB-Lite signal definitions

- Global signals
  - HCLK: the bus clock source (rising-edge triggered)
  - HRESETn: the bus (and system) reset signal (active low)
- Master out/slave in
  - HADDR[31:0]: the 32-bit system address bus
  - HWDATA[31:0]: the system write data bus
  - Control
    - HWRITE: indicates transfer direction (Write=1, Read=0)
    - HSIZE[2:0]: indicates size of transfer (byte, halfword, or word)
    - HBURST[2:0]: indicates single or burst transfer (1, 4, 8, 16 beats)
    - HPROT[3:0]: provides protection information (e.g. I or D; user or handler)
    - HTRANS: indicates current transfer type (e.g. idle, busy, nonseq, seq)
    - HMASTLOCK: indicates a locked (atomic) transfer sequence
- Slave out/master in
  - HRDATA[31:0]: the slave read data bus
  - HREADY: indicates previous transfer is complete
  - HRESP: the transfer response (OKAY=0, ERROR=1)

Key to timing diagram conventions

- Timing diagrams
  - Clock
  - Stable values
  - Transitions
  - High-impedance

- Signal conventions
  - Lower case ‘n’ denote active low (e.g. RESETn)
  - Prefix ‘H’ denotes AHB
  - Prefix ‘P’ denotes APB



Basic read and write transfers with no wait states

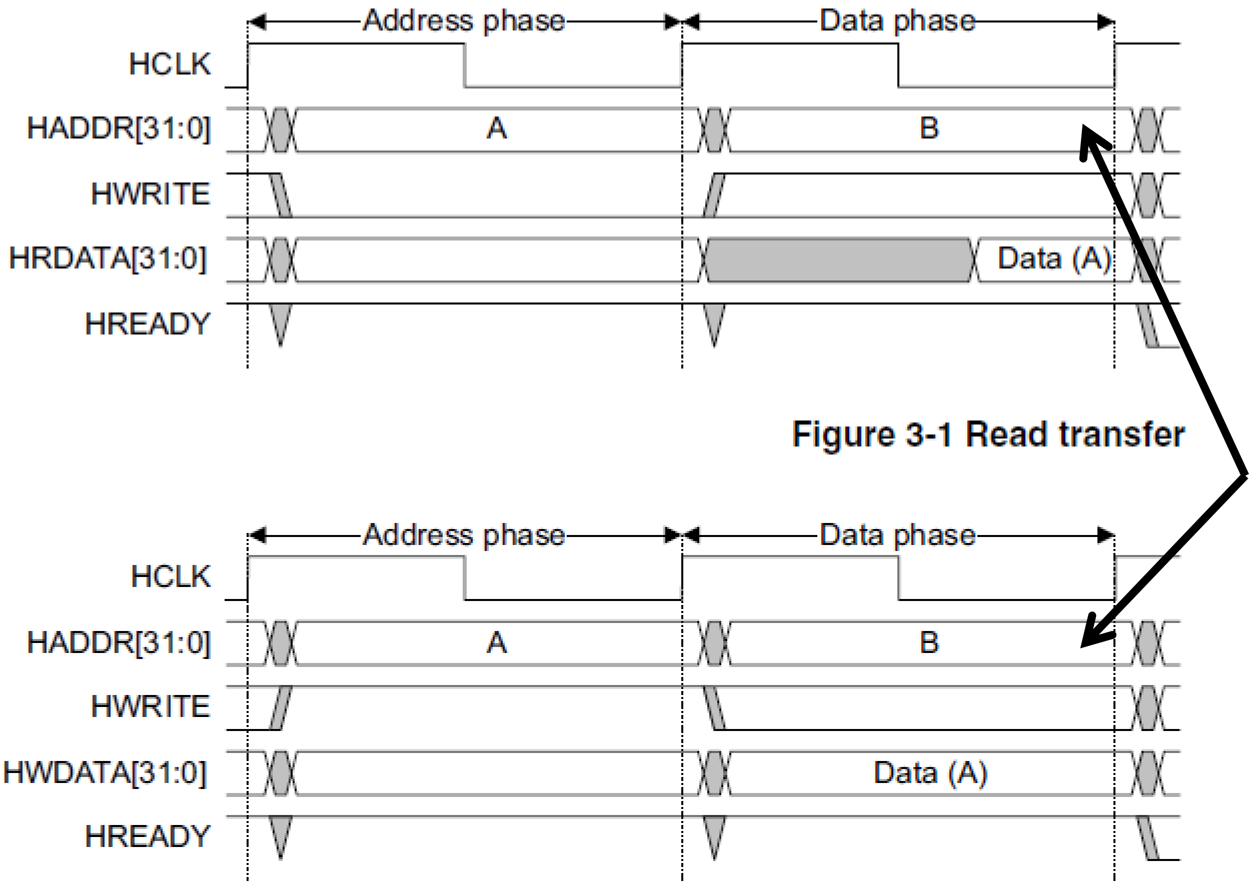
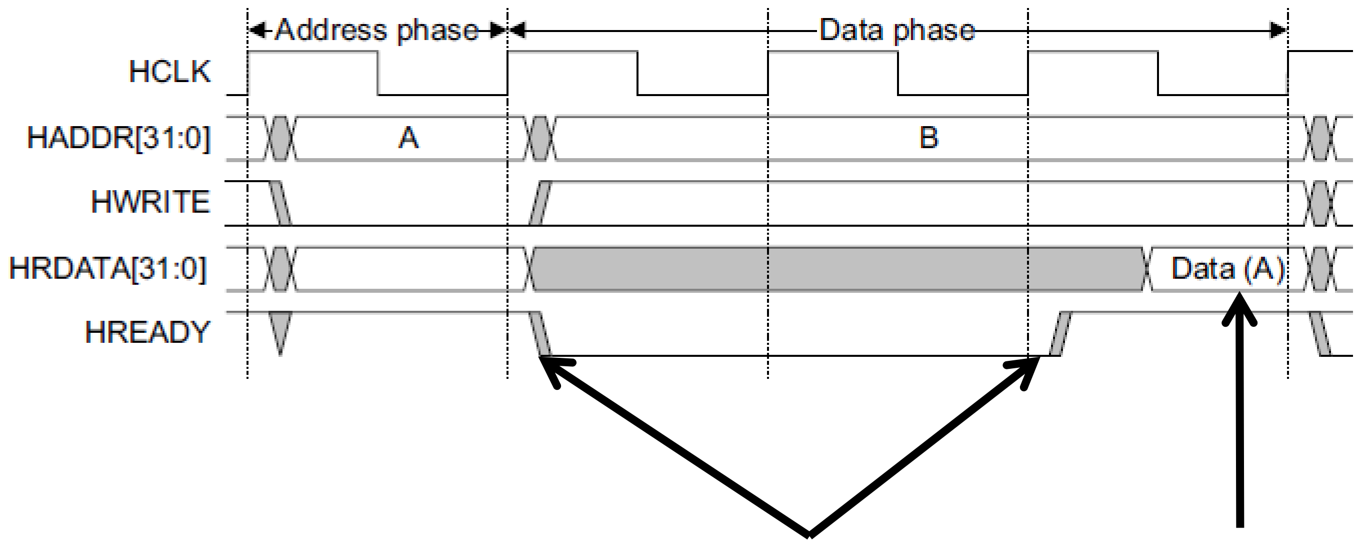


Figure 3-1 Read transfer

Pipelined Address & Data Transfer

Figure 3-2 Write transfer

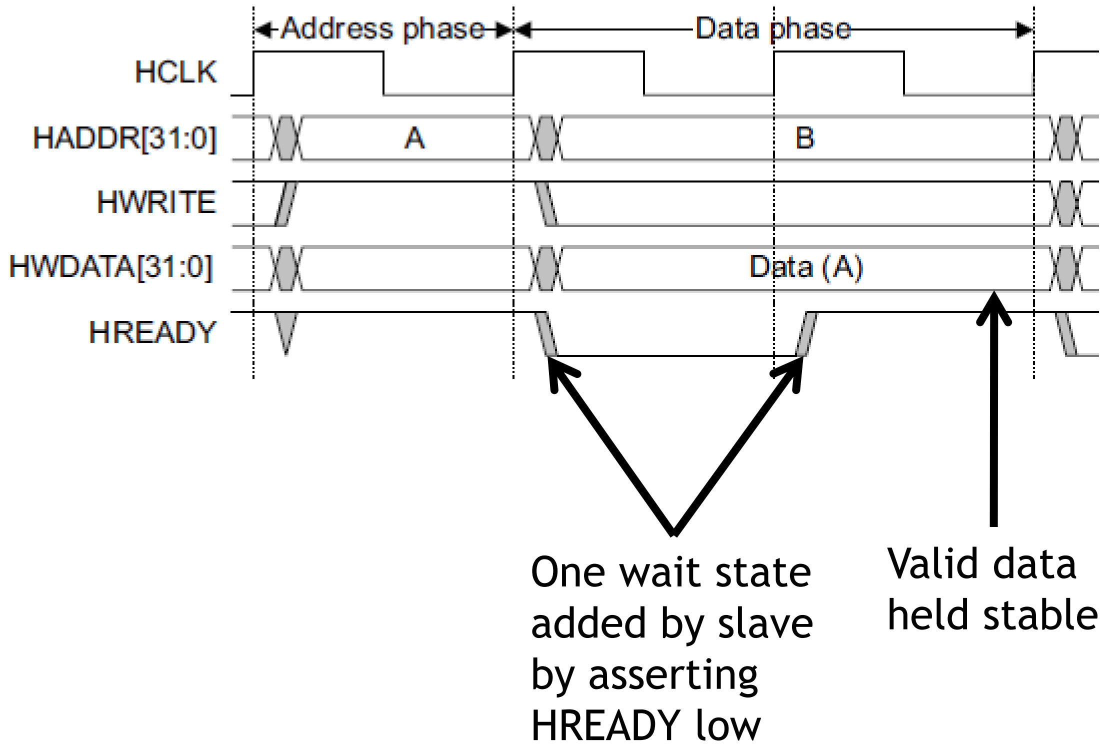
Read transfer with two wait states



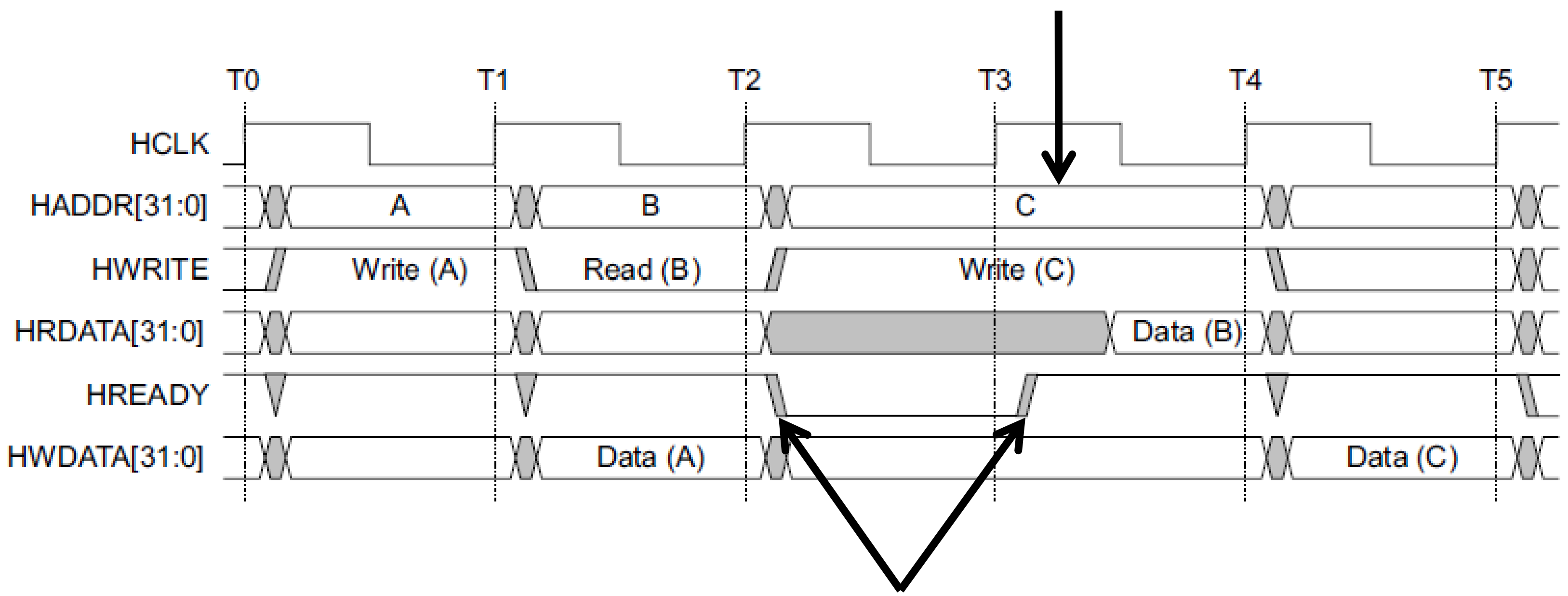
Two wait states added by slave by asserting HREADY low

Valid data produced

Write transfer with one wait state



Wait states extend the address phase of next transfer



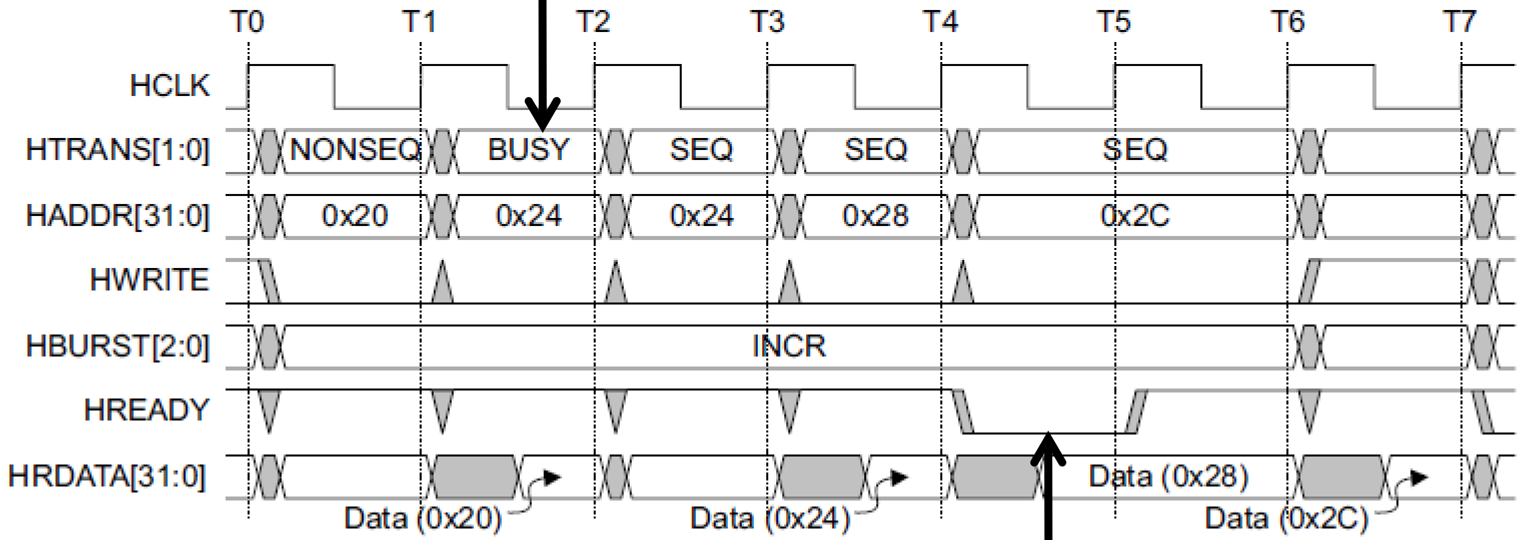
One wait state added by slave by asserting HREADY low

Transfers can be of four types (HTRANS[1:0])

- IDLE (b00)
  - No data transfer is required
  - Slave must OKAY w/o waiting
  - Slave must ignore IDLE
- BUSY (b01)
  - Insert idle cycles in a burst
  - Burst will continue afterward
  - Address/control reflects next transfer in burst
  - Slave must OKAY w/o waiting
  - Slave must ignore BUSY
- NONSEQ (b10)
  - Indicates single transfer or first transfer of a burst
  - Address/control unrelated to prior transfers
- SEQ (b11)
  - Remaining transfers in a burst
  - Addr = prior addr + transfer size

A four beat burst with master busy and slave wait

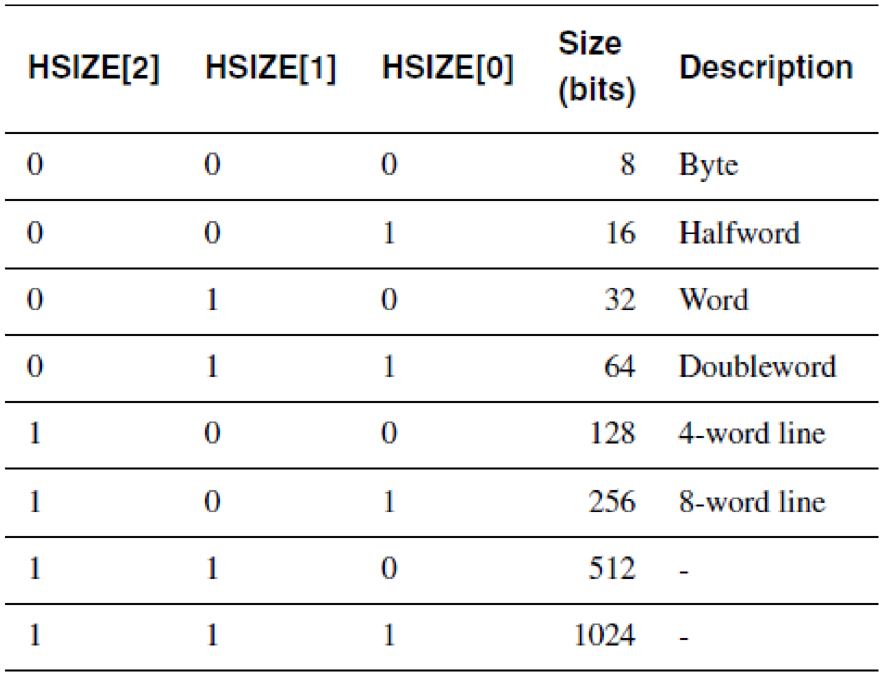
Master busy indicated by HTRANS[1:0]



One wait state added by slave by asserting HREADY low

Controlling the size (width) of a transfer

- HSIZE[2:0] encodes the size
- The cannot exceed the data bus width (e.g. 32-bits)
- HSIZE + HBURST is determines wrapping boundary for wrapping bursts
- HSIZE must remain constant throughout a burst transfer



HSIZE[2]	HSIZE[1]	HSIZE[0]	Size (bits)	Description
0	0	0	8	Byte
0	0	1	16	Halfword
0	1	0	32	Word
0	1	1	64	Doubleword
1	0	0	128	4-word line
1	0	1	256	8-word line
1	1	0	512	-
1	1	1	1024	-

Controlling the burst beats (length) of a transfer

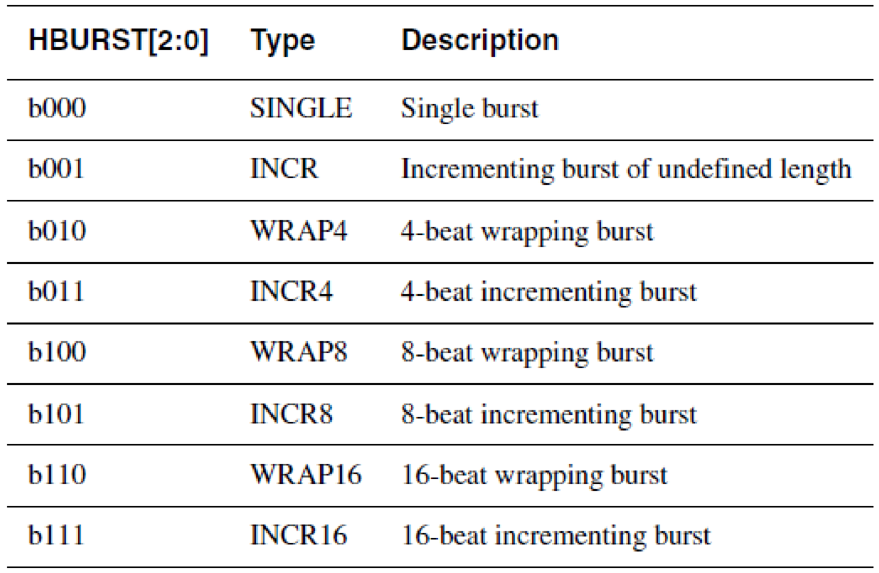
- Burst of 1, 4, 8, 16, and undef

- HBURST[2:0] encodes the type

- Incremental burst

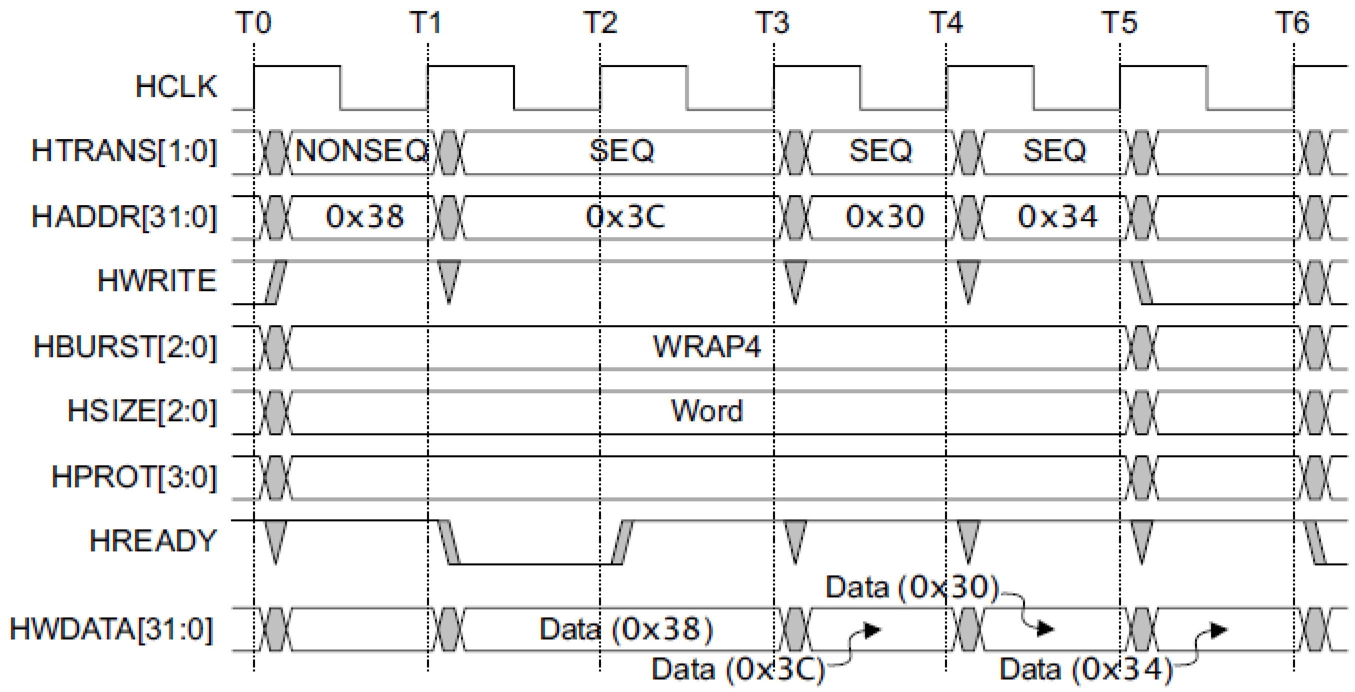
- Wrapping bursts
  - 4 beats x 4-byte words wrapping
  - Wraps at 16 byte boundary
  - E.g. 0x34, 0x38, 0x3c, 0x30,…

- Bursts must not cross 1KB address boundaries

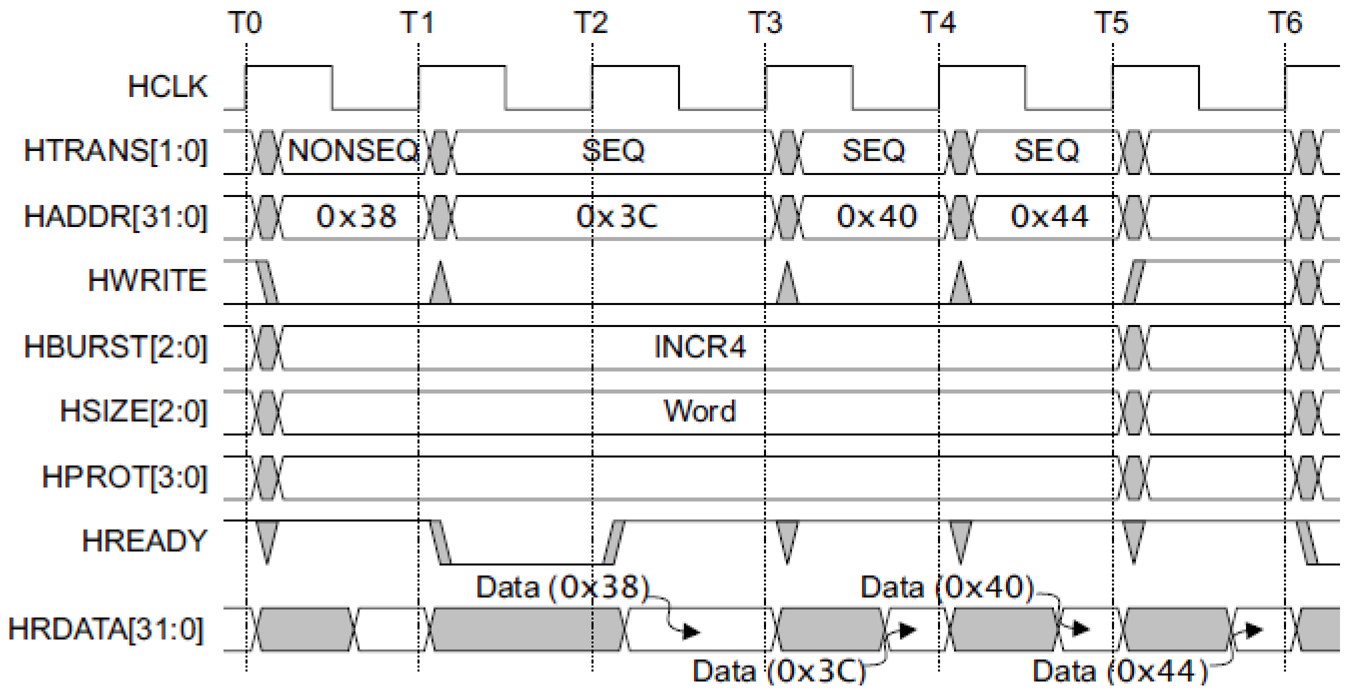


HBURST[2:0]	Type	Description
b000	SINGLE	Single burst
b001	INCR	Incrementing burst of undefined length
b010	WRAP4	4-beat wrapping burst
b011	INCR4	4-beat incrementing burst
b100	WRAP8	8-beat wrapping burst
b101	INCR8	8-beat incrementing burst
b110	WRAP16	16-beat wrapping burst
b111	INCR16	16-beat incrementing burst

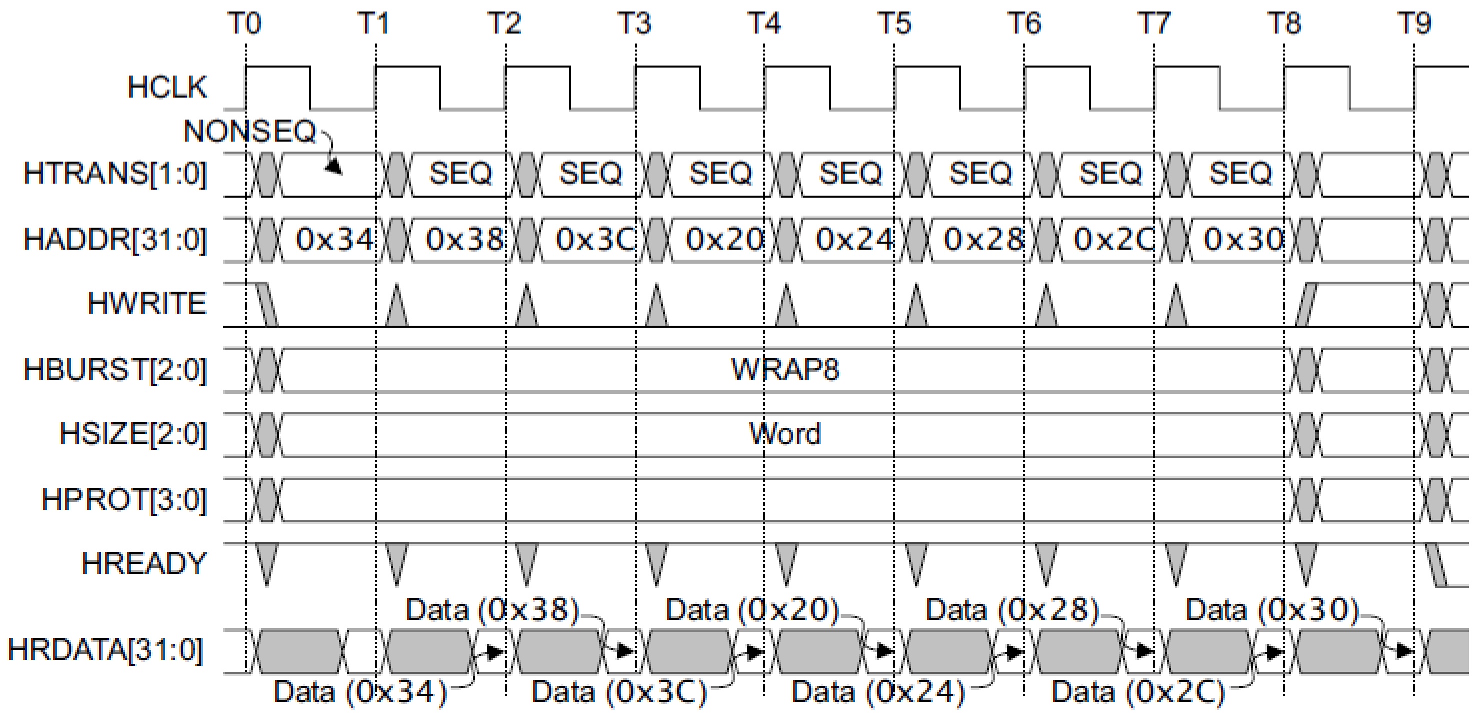
A four beat wrapping burst (WRAP4)



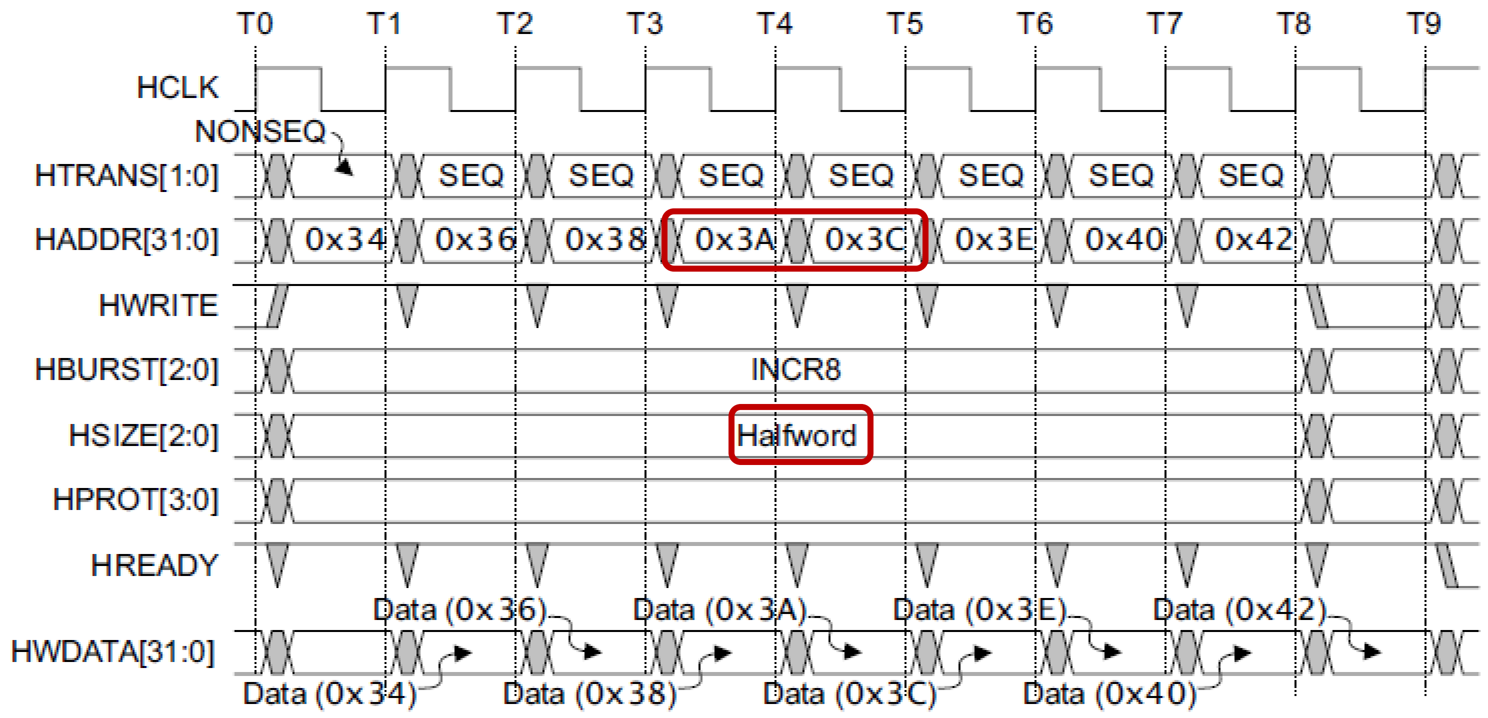
A four beat incrementing burst (INCR4)



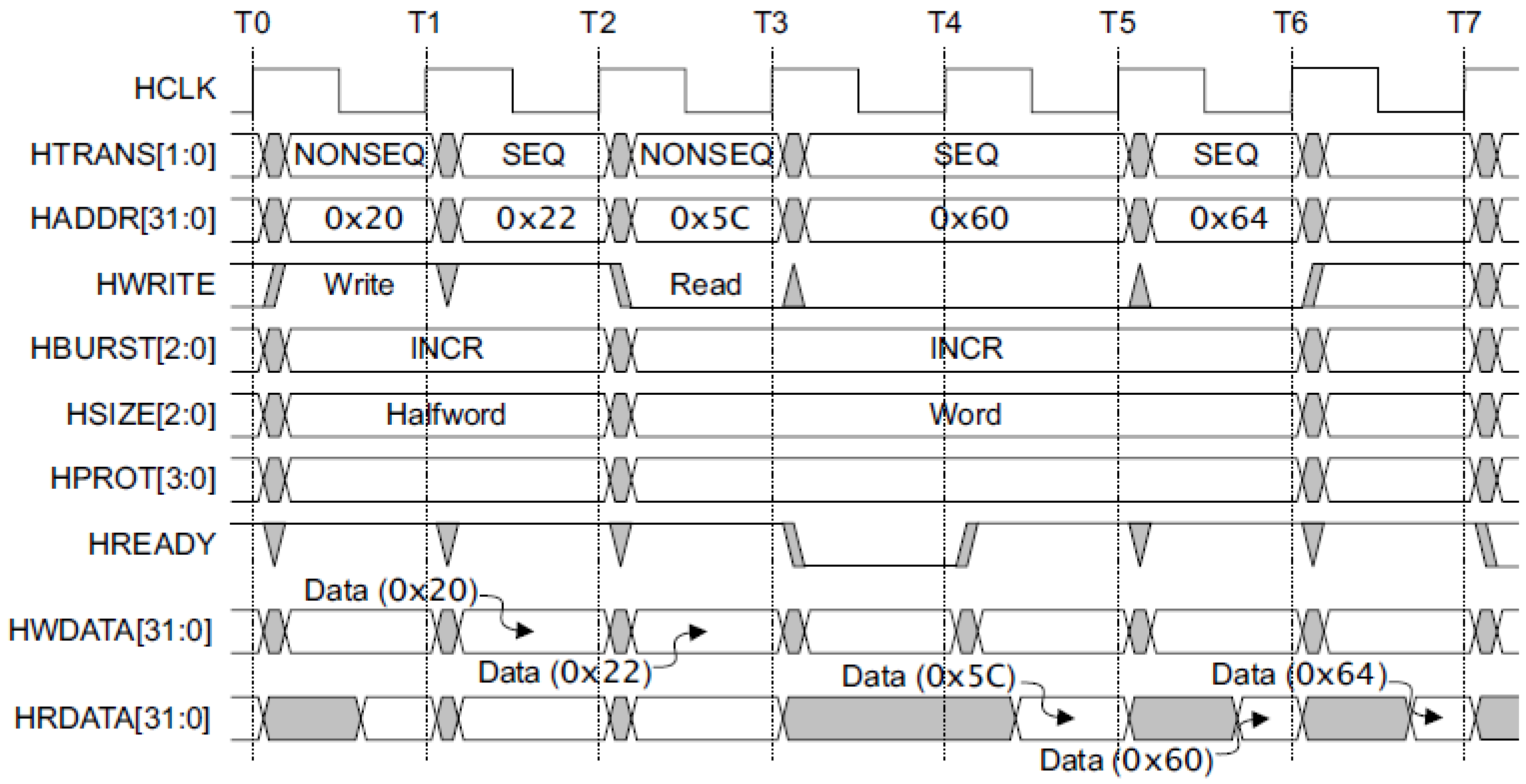
An eight beat wrapping burst (WRAP8)



An eight beat incrementing burst (INCR8) using half-word transfers

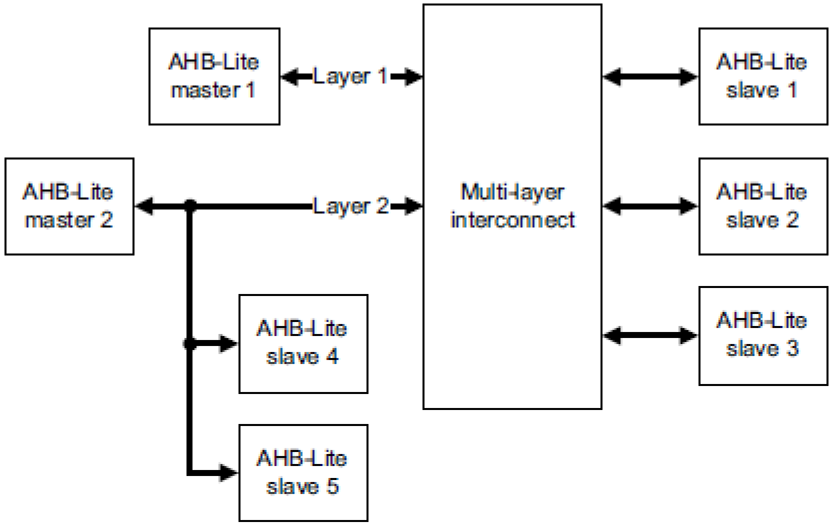


An undefined length incrementing burst (INCR)



Multi-master AHB-Lite requires a multi-layer interconnect

- AHB-Lite is single-master
- Multi-master operation
  - Must isolate masters
  - Each master assigned to layer
  - Interconnect arbitrates slave accesses
- Full crossbar switch often unneeded
  - Slaves 1, 2, 3 are shared
  - Slaves 4, 5 are local to Master 1



APB is simpler interface than AHB

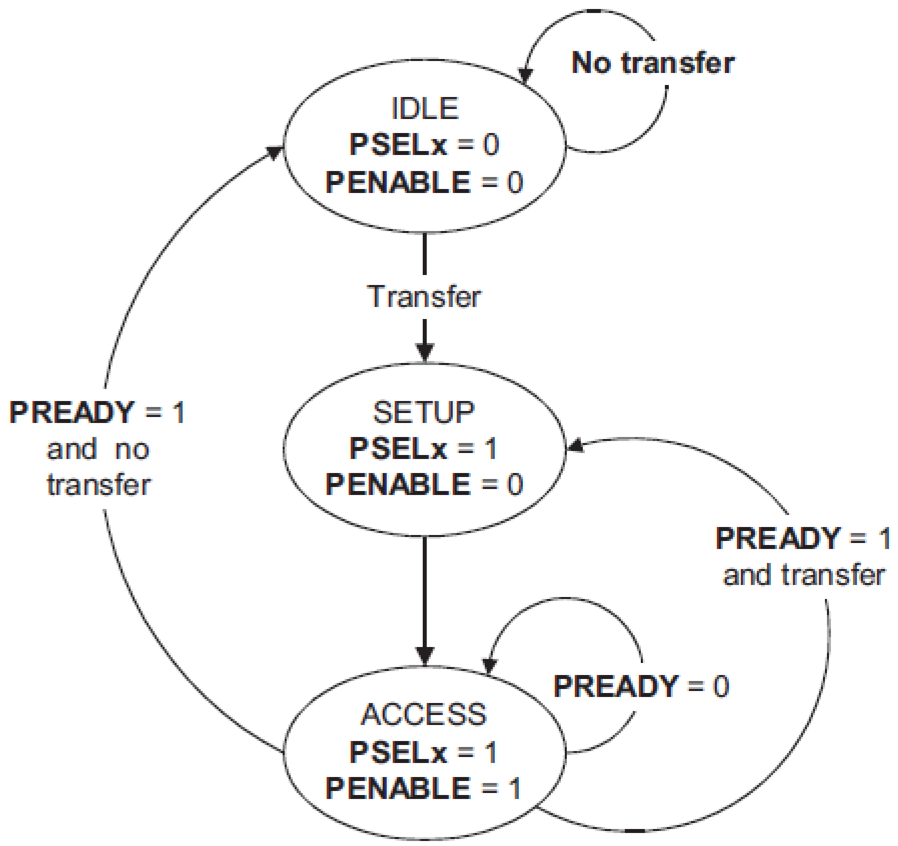
- Low-cost
- Low-power
- Low-complexity
- Low-bandwidth
- Non pipelined
- Ideal for peripherals

APB signal definitions

- PCLK: the bus clock source (rising-edge triggered)
- PRESETn: the bus (and typically system) reset signal (active low)
- PADDR: the APB address bus (can be up to 32-bits wide)
- PSELx: the select line for each slave device
- PENABLE: indicates the 2nd and subsequent cycles of an APB xfer
- PWRITE: indicates transfer direction (Write=H, Read=L)
- PWDATA: the write data bus (can be up to 32-bits wide)
- PREADY: used to extend a transfer
- PRDATA: the read data bus (can be up to 32-bits wide)
- PSLVERR: indicates a transfer error (OKAY=L, ERROR=H)

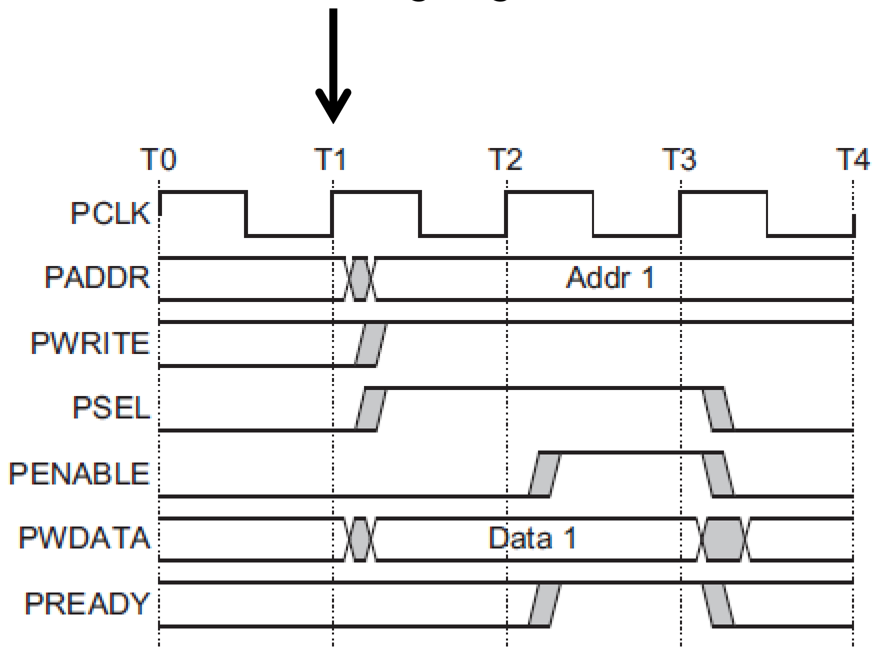
APB state machine

- IDLE
  - Default APB state
- SETUP
  - When transfer required
  - PSELx is asserted
  - Only one cycle
- ACCESS
  - PENABLE is asserted
  - Addr, write, select, and write data remain stable
  - Stay if PREADY = L
  - Goto IDLE if PREADY = H and no more data
  - Goto SETUP is PREADY = H and more data pending



A write transfer with no wait states

Setup phase begins
with this rising edge

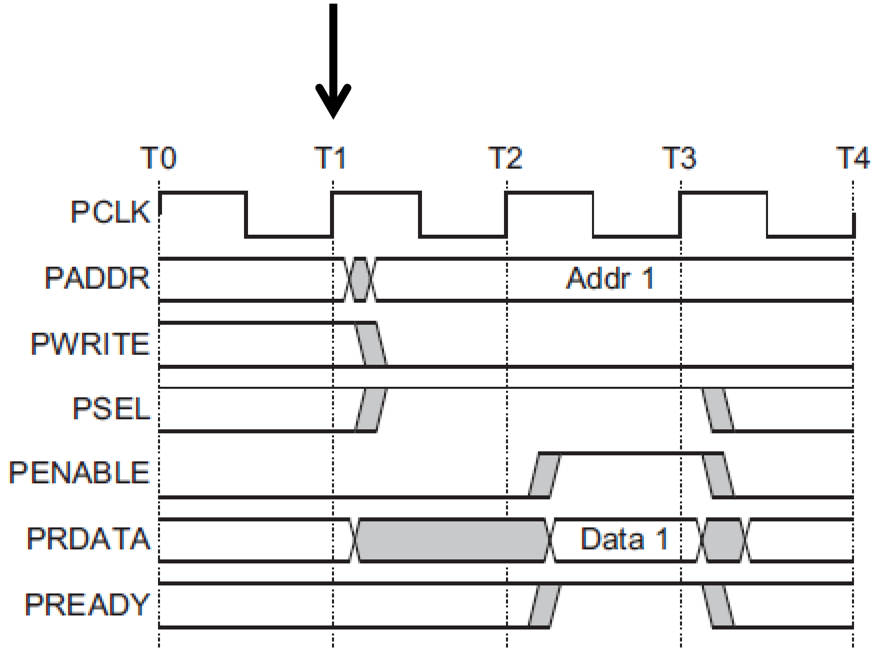


Setup Phase

Access Phase

A read transfer with no wait states

Setup phase begins with this rising edge

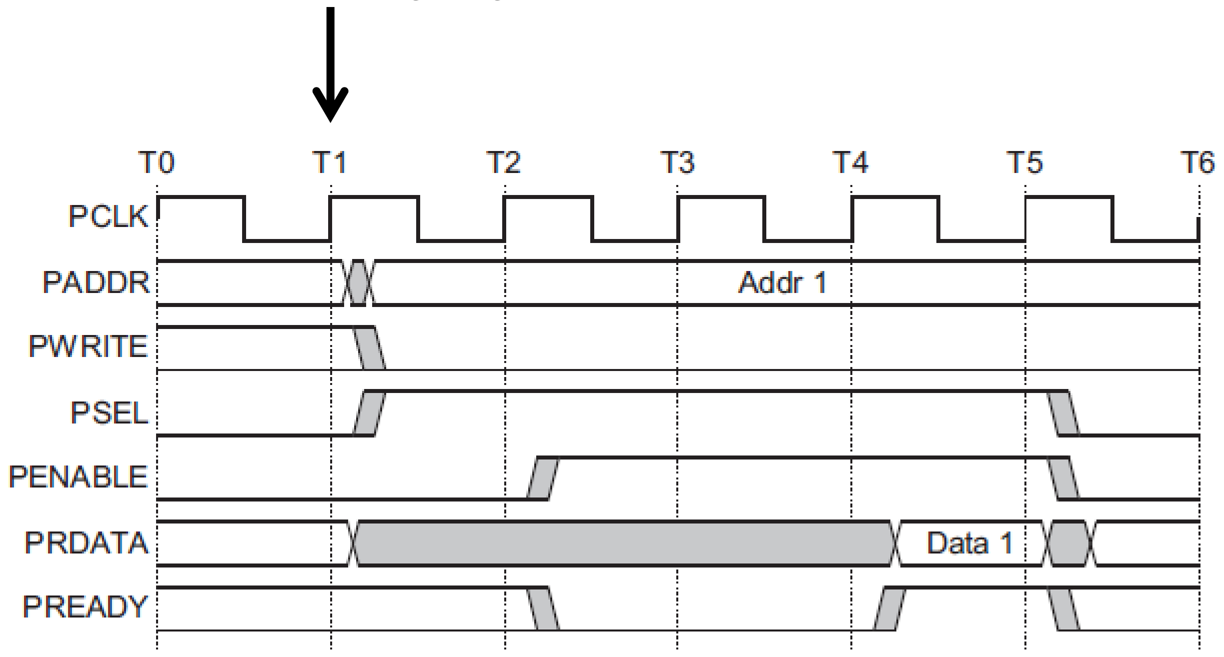


Setup Phase

Access Phase

A read transfer with wait states

Setup phase begins
with this rising edge



Setup Phase

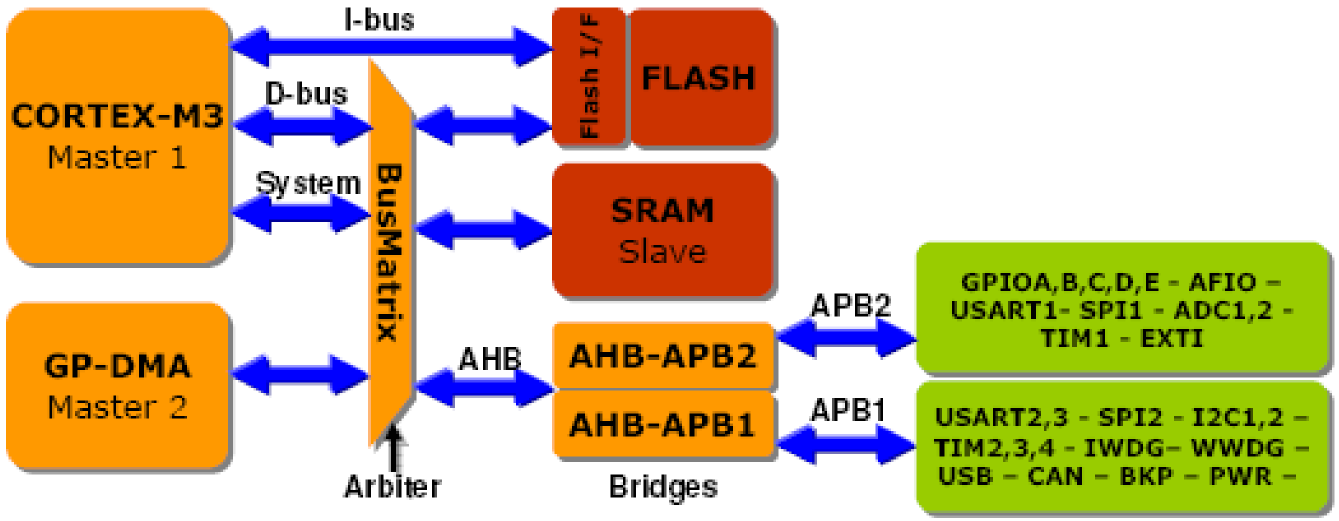
Wait State

Wait State

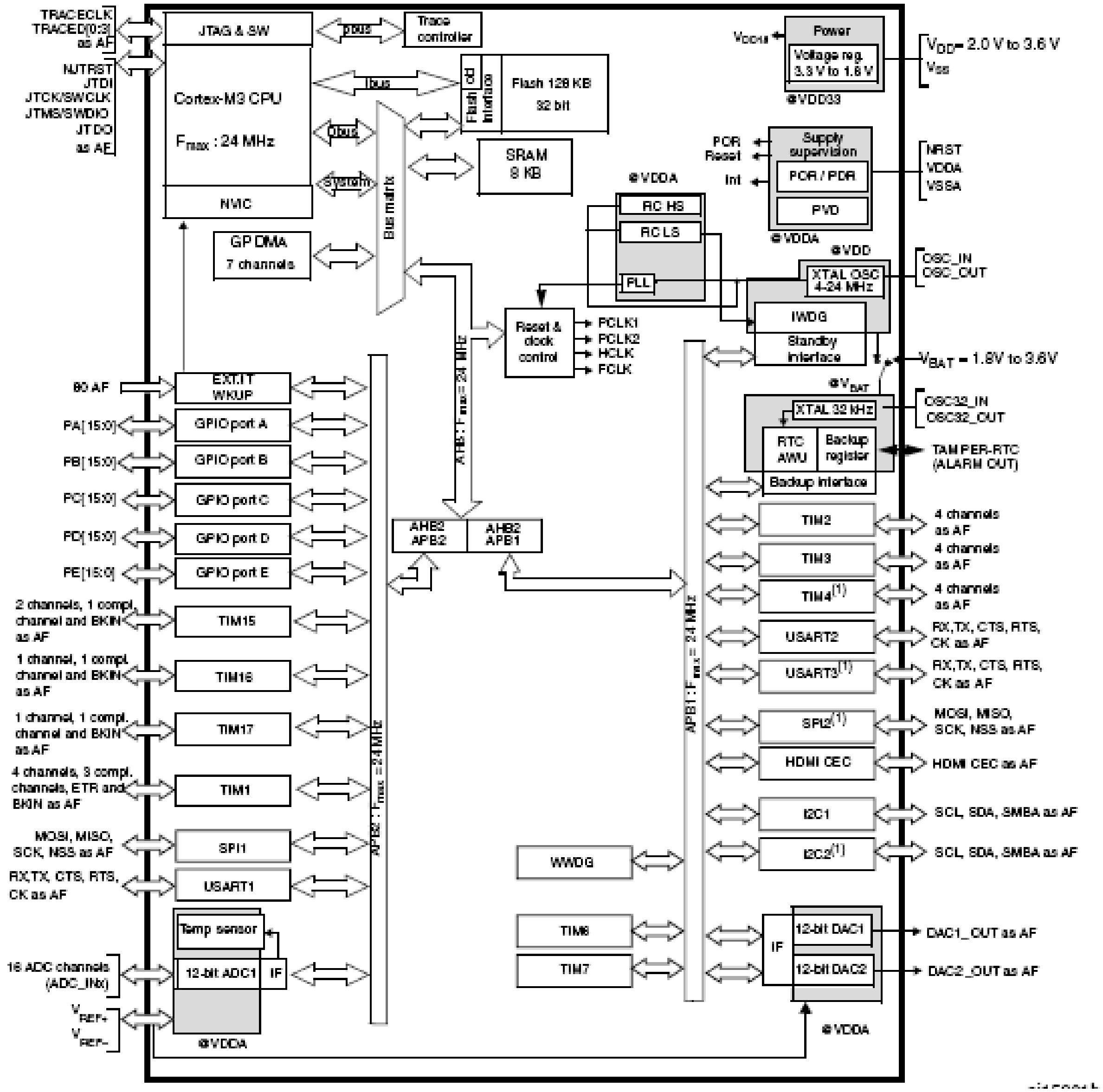
Access Phase

Summing Up: Cortex M3 Core IF

- Multiply possibilities of bus accesses to SRAM, Flash, Peripherals, DMA
  - BusMatrix added to Harvard architecture allows parallel access
- Efficient DMA and Rapid data flow
  - Direct path to SRAM through arbiter, guarantees alternating access
  - Harvard architecture + BusMatrix allows Flash execution in parallel with DMA transfer
- Increase Peripherals Speed for better performance
  - Dual Advanced Peripheral buses (APB) architecture w/ High Speed APB (APB2) up to 72MHz and Low Speed APB (APB1) up to 36MHz
  - Allows to optimize use of peripherals (18MHz SPI, 4.5Mbps USART, 72MHz PWM Timer, 18MHz toggling I/Os)


CORTEX-M3
Master 1
GP-DMA
Master 2
I-bus
D-bus
System
BusMatrix
Arbiter
AHB
Flash I/F
FLASH
SRAM
Slave
AHB-APB2
AHB-APB1
Bridges
APB2
APB1
GPIOA,B,C,D,E - AFIO - USART1- SPI1 - ADC1,2 - TIM1 - EXTI
USART2,3 - SPI2 - I2C1,2 - TIM2,3,4 - IWDG- WWDG - USB - CAN - BKP - PWR -

Details: STM32F100RBT6B



TRACECLK
TRACED[0:3] as AF
NJTRST
JTDI
JTCK/SWCLK
JTMS/SWDIO
JTDO as AF
JTAG & SW
pbus
Trace controller
Cortex-M3 CPU
Fmax : 24 MHz
NVIC
Ibus
Dbus
System
Bus matrix
Flash obl interface
Flash 128 KB 32 bit
SRAM 8 KB
GP DMA 7 channels
Power
Voltage reg. 3.3 V to 1.8 V
@VDD33
VDD18
VDD = 2.0 V to 3.6 V
VSS
Supply supervision
POR / PDR
PVD
@VDDA
POR Reset
Int
NRST
VDDA
VSSA
@VDDA
RC HS
RC LS
PLL
@VDD
XTAL OSC 4-24 MHz
OSC_IN
OSC_OUT
IWDG
Standby interface
VBAT = 1.8V to 3.6V
Reset & clock control
PCLK1
PCLK2
HCLK
FCLK
AHB : Fmax = 24 MHz
@VBAT
XTAL 32 kHz
OSC32_IN
OSC32_OUT
RTC AWU
Backup register
Backup interface
TAMPER-RTC (ALARM OUT)
80 AF
EXT.IT WKUP
PA[15:0]
GPIO port A
PB[15:0]
GPIO port B
PC[15:0]
GPIO port C
PD[15:0]
GPIO port D
PE[15:0]
GPIO port E
AHB2 APB2
AHB2 APB1
2 channels, 1 compl. channel and BKIN as AF
TIM15
1 channel, 1 compl. channel and BKIN as AF
TIM16
1 channel, 1 compl. channel and BKIN as AF
TIM17
4 channels, 3 compl. channels, ETR and BKIN as AF
TIM1
MOSI, MISO, SCK, NSS as AF
SPI1
RX,TX, CTS, RTS, CK as AF
USART1
Temp sensor
16 ADC channels (ADC_INx)
12-bit ADC1
IF
VREF+
VREF-
@VDDA
APB2 : Fmax = 24 MHz
APB1 : Fmax = 24 MHz
TIM2
4 channels as AF
TIM3
4 channels as AF
TIM4(1)
4 channels as AF
USART2
RX,TX, CTS, RTS, CK as AF
USART3(1)
RX,TX, CTS, RTS, CK as AF
SPI2(1)
MOSI, MISO, SCK, NSS as AF
HDMI CEC
HDMI CEC as AF
I2C1
SCL, SDA, SMBA as AF
I2C2(1)
SCL, SDA, SMBA as AF
WWDG
TIM6
TIM7
IF
12-bit DAC1
DAC1_OUT as AF
12-bit DAC2
DAC2_OUT as AF
@VDDA