Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the TxFIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the TxFIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

OTG_FS device control OUT endpoint 0 control register (OTG_FS_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the OTG_FS_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
w	r			w	w					rs	rw	r	r	r		r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:

- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a Transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP:** Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

OTG_FS device endpoint-x control register (OTG_FS_DOEPCTLx) (x = 1..3, where x = Endpoint_number)

Address offset for OUT endpoints: 0xB00 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint disabled interrupt. The application must set this bit only if Endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

SODDFRM: Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous OUT endpoints only.

Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

OTG_FS device endpoint-x interrupt register (OTG_FS_DIEPINTx) (x = 0..3, where x = Endpoint_number)

Address offset: 0x908 + (Endpoint_number × 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 350*. The application must read this register when the IN endpoints interrupt bit of the Core interrupt register (IEPINT in OTG_FS_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_FS_DAINT) register to get the exact endpoint number for the Device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_FS_DAINT and OTG_FS_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
																								r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the OTG_FS_GAHBCFG register (TXFELVL bit in OTG_FS_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_FS_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITTXFE:** IN token received when TxFIFO is empty

Applies to non-periodic IN endpoints only.

Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Applies only to Control IN endpoints.

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 Reserved, must be kept at reset value..

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

OTG_FS device endpoint-x interrupt register (OTG_FS_DOEPINTx) (x = 0..3, where x = Endpoint_number)

Address offset: 0xB08 + (Endpoint_number × 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 350*. The application must read this register when the OUT Endpoints Interrupt bit of the OTG_FS_GINTSTS register (OEPINT bit in OTG_FS_GINTSTS) is set. Before the application can read this register, it must first read the OTG_FS_DAINT register to get the exact endpoint number for the OTG_FS_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_FS_DAINT and OTG_FS_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
																									rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received

Applies to control OUT endpoint only.

This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled

Applies only to control OUT endpoints.

Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done

Applies to control OUT endpoint only.

Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

OTG_FS device IN endpoint 0 transfer size register (OTG_FS_DIEPTSIZ0)

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_FS_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PKTCNT		Reserved												XFRSIZ						
											rw	rw													rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the TxFIFO.

OTG_FS device OUT endpoint 0 transfer size register (OTG_FS_DOEPTSIZ0)

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the Endpoint enable bit in the OTG_FS_DOEPCTL0 registers (EPENA bit in OTG_FS_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	rw	rw										rw													rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the RxFIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

OTG_FS device endpoint-x transfer size register (OTG_FS_DIEPTSIZx) (x = 1..3, where x = Endpoint_number)

Address offset: 0x910 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the Endpoint enable bit in the OTG_FS_DIEPCTLx registers (EPENA bit in OTG_FS_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bit 28:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.

Bits 18:0 **XFRSIZ:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the TxFIFO.

OTG_FS device IN endpoint transmit FIFO status register (OTG_FS_DTXFSTSx) (x = 0..3, where x = Endpoint_number)

Address offset for IN endpoints: 0x918 + (Endpoint_number × 0x20) This read-only register contains the free space information for the Device IN endpoint TxFIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INEPTFSAV															
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

31:16 Reserved, must be kept at reset value.

15:0 **INEPTFSAV:** IN endpoint TxFIFO space available

Indicates the amount of free space available in the Endpoint TxFIFO.
Values are in terms of 32-bit words:
0x0: Endpoint TxFIFO is full
0x1: 1 word available
0x2: 2 words available
0xn: n words available
Others: Reserved

OTG_FS device OUT endpoint-x transfer size register (OTG_FS_DOEPTSIZx) (x = 1..3, where x = Endpoint_number)

Address offset: 0xB10 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the OTG_FS_DOEPCTLx registers (EPENA bit in OTG_FS_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID:** Received data PID

Applies to isochronous OUT endpoints only.
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT: SETUP packet count

Applies to control OUT Endpoints only.

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bit 28:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.

Bits 18:0 **XFRSIZ:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

29.16.5 OTG_FS power and clock gating control register (OTG_FS_PCGCCTL)

Address offset: 0xE00

Reset value: 0x0000 0000

This register is available in host and device modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved			GATEHCLK	STPPCLK
																											rw			rw	rw

Bit 31:5 Reserved, must be kept at reset value.

Bit 4 **PHYSUSP:** PHY Suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit (bit 0).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

29.16.6 OTG_FS register map

The table below gives the USB OTG register map and reset values.

Table 155. OTG_FS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_FS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	OTG_FS_GOTGINT	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	OTG_FS_GAHBCFG	Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
	Reset value																								0	0							0
0x00C	OTG_FS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved									TRDT										HNPCAP	SRPCAP	PHYSEL	Reserved				TOCAL		
	Reset value													0	1	0	1							0	0	1					0	0	0
0x010	OTG_FS_GRSTCTL	AHBIDL	Reserved											TXFNUM															TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1												0	0	0									0				0	0	0	0	0	0
0x014	OTG_FS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_FS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	OTG_FS_GRXSTSR (host mode)	Reserved											PKTSTS				DPID		BCNT												CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSR (Device mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT												EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_FS_GRXSTSR (host mode)	Reserved											PKTSTS				DPID		BCNT												CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSPR (Device mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT												EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_FS_GRXFSIZ	Reserved																RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	**OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0**	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	**OTG_FS_HNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	**OTG_FS_GCCFG**	Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved																
	Reset value											0	0	0	0		0																
0x03C	**OTG_FS_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x100	**OTG_FS_HPTXFSIZ**	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0x104	**OTG_FS_DIEPTXF1**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	**OTG_FS_DIEPTXF2**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	**OTG_FS_DIEPTXF3**	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	**OTG_FS_HCFG**	Reserved																													FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	**OTG_FS_HFIR**	Reserved																FRIVL															
	Reset value																	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	**OTG_FS_HFNUM**	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	**OTG_FS_HPTXSTS**	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	**OTG_FS_HAINT**	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	**OTG_FS_HAINTMSK**	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	**OTG_FS_HPRT**	Reserved													PSPD	PTCTL					PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS	
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	**OTG_FS_HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	**OTG_FS_HCCHAR1**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	**OTG_FS_HCCHAR2**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x560	**OTG_FS_HCCHAR3**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	**OTG_FS_HCCHAR4**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	**OTG_FS_HCCHAR5**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	**OTG_FS_HCCHAR6**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	**OTG_FS_HCCHAR7**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_FS_HCINT0**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x528	**OTG_FS_HCINT1**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x548	**OTG_FS_HCINT2**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x568	**OTG_FS_HCINT3**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x588	**OTG_FS_HCINT4**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5A8	**OTG_FS_HCINT5**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5C8	**OTG_FS_HCINT6**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5E8	**OTG_FS_HCINT7**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x50C	**OTG_FS_HCINTMSK0**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x52C	**OTG_FS_HCINTMSK1**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54C	OTG_FS_HCINTMSK2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x56C	OTG_FS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x58C	OTG_FS_HCINTMSK4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5AC	OTG_FS_HCINTMSK5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5CC	OTG_FS_HCINTMSK6	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x5EC	OTG_FS_HCINTMSK7	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	OTG_FS_HCTSIZ0	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_FS_HCTSIZ1	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	OTG_FS_HCTSIZ2	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	OTG_FS_HCTSIZ3	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	OTG_FS_HCTSIZ4	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	OTG_FS_HCTSIZ5	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	OTG_FS_HCTSIZ6	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	OTG_FS_HCTSIZ7	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_FS_DCFG	Reserved																			PFIVL		DAD								Reserved	NZLSOHSK	DSPD
	Reset value																				0	0	0	0	0	0	0	0	0		0	0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804	**OTG_FS_DCTL**	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	**OTG_FS_DSTS**	Reserved										FNSOF														Reserved				EERR	ENUMSPD	SUSPSTS	
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	**OTG_FS_DIEPMSK**	Reserved																									INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM	
	Reset value																										0	0	0	0		0	0
0x814	**OTG_FS_DOEPMSK**	Reserved																											OTEPDM	STUPM	Reserved	EPDM	XFRCM	
	Reset value																												0	0		0	0
0x818	**OTG_FS_DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	**OTG_FS_DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	**OTG_FS_DVBUSDIS**	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	**OTG_FS_DVBUSPULSE**	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x834	**OTG_FS_DIEPEMPMSK**	Reserved																				INEPTXFEM											
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	**OTG_FS_DIEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0	0	0	0	0	0		0	0	0		1														0	0
0x918	**TG_FS_DTXFSTS0**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	**OTG_FS_DIEPCTL1**	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved								MPSIZ						
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0									0	0	0	0	0	0	0
0x938	**TG_FS_DTXFSTS1**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x940	**OTG_FS_DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved								MPSIZ						
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0									0	0	0	0	0	0	0
0x958	**TG_FS_DTXFSTS2**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x960	**OTG_FS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	**OTG_FS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB20	**OTG_FS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	**OTG_FS_DOEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	**OTG_FS_DOEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	**OTG_FS_DIEPINT0**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x928	**OTG_FS_DIEPINT1**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x948	**OTG_FS_DIEPINT2**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0x968	**OTG_FS_DIEPINT3**	Reserved																								TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																									1	0		0	0		0	0
0xB08	**OTG_FS_DOEPINT0**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0
0xB28	**OTG_FS_DOEPINT1**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																										0		0	0		0	0

Table 155. OTG_FS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB48	**OTG_FS_DOEPINT2**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																									0	0		0	0		0	0
0xB68	**OTG_FS_DOEPINT3**	Reserved																								Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																									0	0		0	0		0	0
0x910	**OTG_FS_DIEPTSIZ0**	Reserved											PKTCNT		Reserved												XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x930	**OTG_FS_DIEPTSIZ1**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	**OTG_FS_DIEPTSIZ2**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	**OTG_FS_DIEPTSIZ3**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB10	**OTG_FS_DOEPTSIZ0**	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	**OTG_FS_DOEPTSIZ1**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	**OTG_FS_DOEPTSIZ2**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	**OTG_FS_DOEPTSIZ3**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	**OTG_FS_PCGCCTL**	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

Refer to *Table 1 on page 50* for the register boundary addresses.

29.17 OTG_FS programming model

29.17.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_FS_GINTSTS (CMOD bit in OTG_FS_GINTSTS) reflects the mode. The OTG_FS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_FS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core's configuration:

1. Program the following fields in the OTG_FS_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - RxFIFO non-empty (RXFLVL bit in OTG_FS_GINTSTS)
   - Periodic TxFIFO empty level
2. Program the following fields in the OTG_FS_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - FS timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the OTG_FS_GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_FS_GINTSTS to determine whether the OTG_FS controller is operating in host or device mode.

29.17.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_FS_GINTMSK register to unmask
2. Program the OTG_FS_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_FS_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_FS_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_FS_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_FS_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_FS_HPRT.
9. Read the PSPD bit in OTG_FS_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_FS_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_FS_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_FS_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_FS_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

29.17.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_FS_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
2. Program the OTG_FS_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
3. Program the VBUSBSEN bit in the OTG_FS_GCCFG register to enable V_{BUS} sensing in “B” device mode and supply the 5 volts across the pull-up resistor on the DP line.
4. Wait for the USBRST interrupt in OTG_FS_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.

Wait for the ENUMDNE interrupt in OTG_FS_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_FS_DSTS

register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 1040*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

29.17.4 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_FS_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_FS_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_FS_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_FS_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_FS_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

Halting a channel

The application can disable any channel by programming the OTG_FS_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_FS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_FS_HCINTx before reallocating the channel for other transactions. The OTG_FS host does not interrupt the transaction that has already been started on the USB.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the Request queue is full (before disabling the channel), by programming the OTG_FS_HCCHARx register with the CHDIS bit set to 1, and the CHENA bit cleared to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_FS_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, Data, TXERR) for the same channel before receiving the halt.
2. When a DISCINT (Disconnect Device) interrupt in OTG_FS_GINTSTS is received. (The application is expected to disable all enabled channels).
3. When the application aborts a transfer before normal completion.

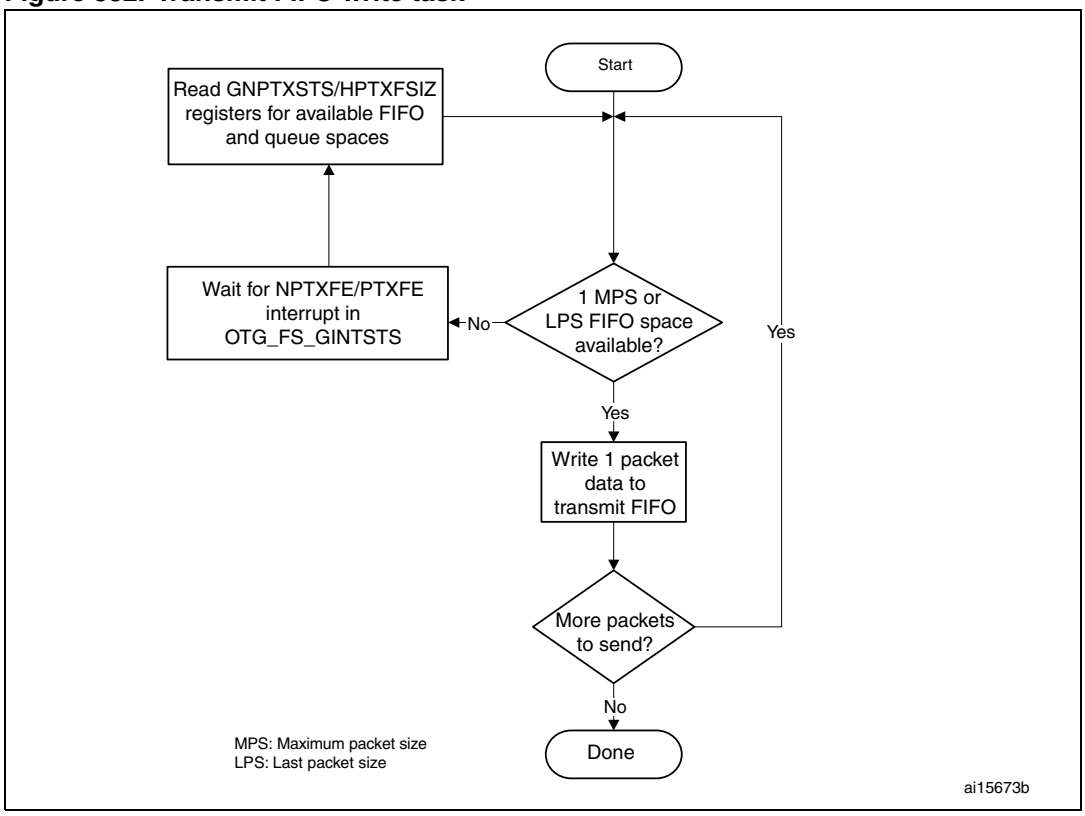
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG_FS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last DWORD write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in DWORDs. If the packet size is non-DWORD aligned, the application must use padding. The OTG_FS host determines the actual packet size based on the programmed maximum packet size and transfer size.

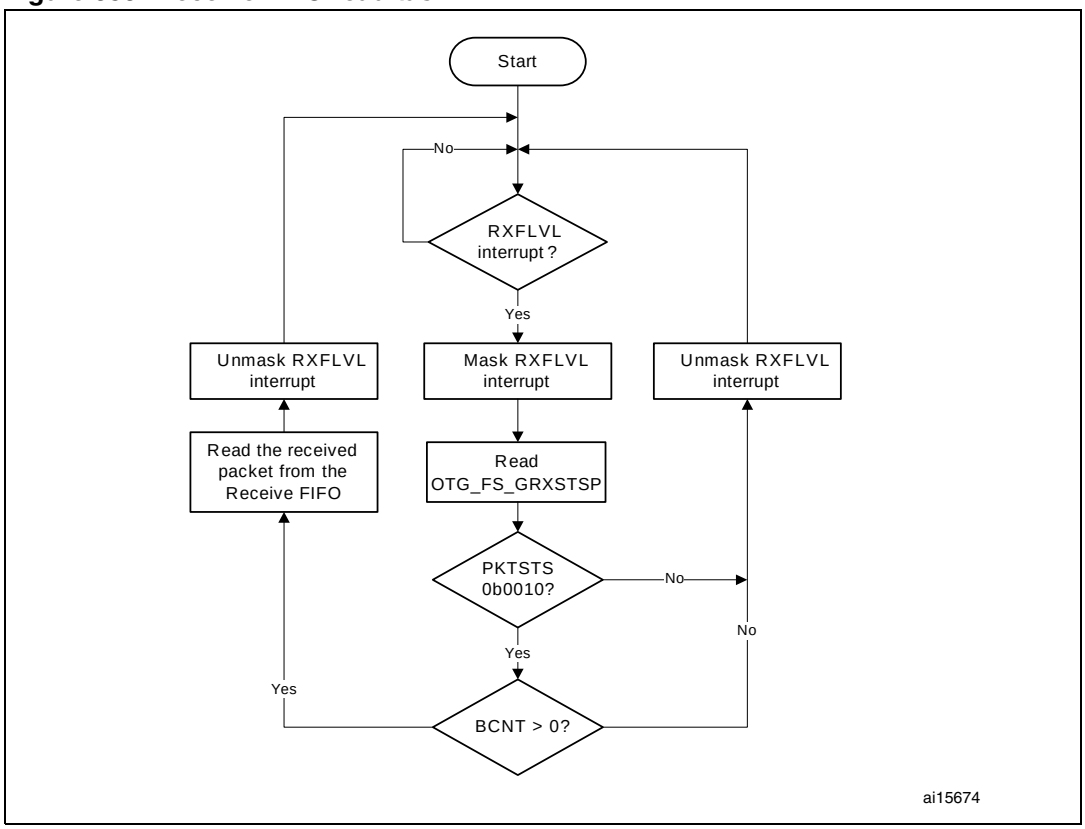
Figure 352. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 353. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 354*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control

SETUP transaction operates in the same way but has only one packet. The assumptions are:

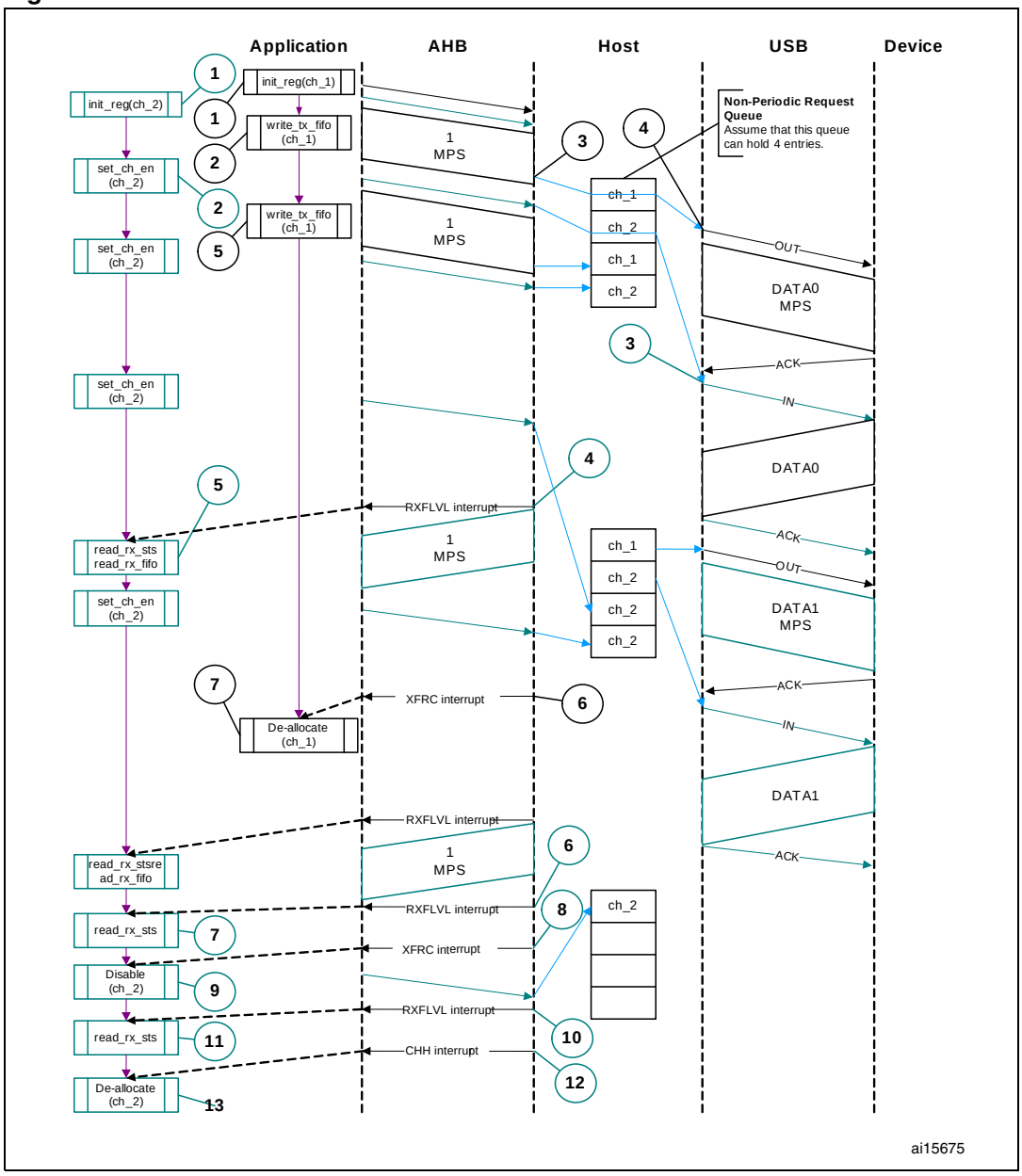
– The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
– The non-periodic transmit FIFO can hold two packets (128 bytes for FS).
– The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

  a) Initialize channel 1
  b) Write the first packet for channel 1
  c) Along with the last Word write, the core writes an entry to the non-periodic request queue
  d) As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
  e) Write the second (last) packet for channel 1
  f) The core generates the XFRC interrupt as soon as the last transaction is completed successfully
  g) In response to the XFRC interrupt, de-allocate the channel for other transfers
  h) Handling non-ACK responses

Figure 354. Normal bulk/control OUT/SETUP and bulk/control IN transactions



The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/Control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
```

```
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO as and when the space is available in the transmit FIFO and the Request queue. The application can make use of the NPTXFE interrupt in OTG_FS_GINTSTS to find the transmit FIFO space.

b) Bulk/Control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
```

```
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the Request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

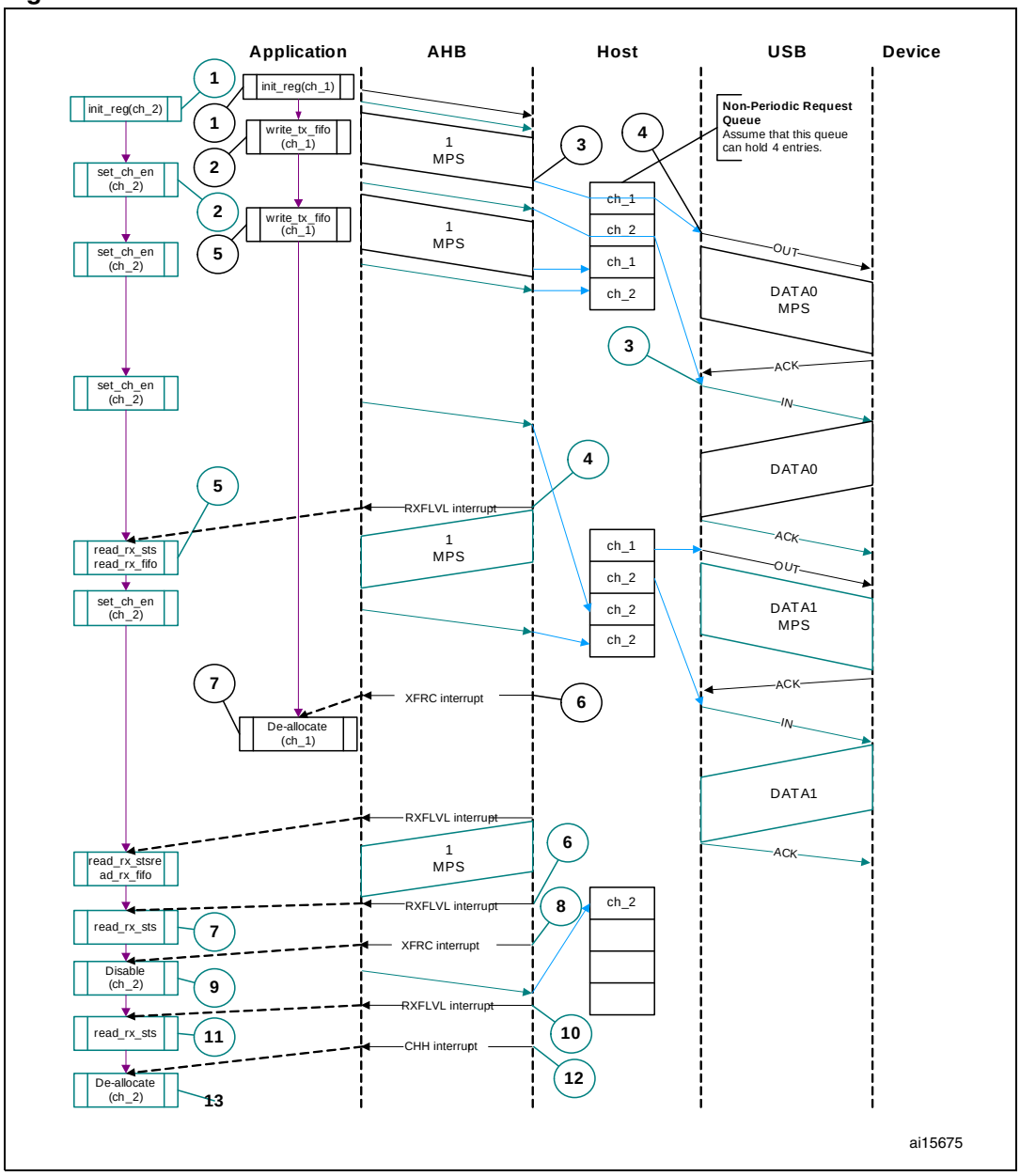
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 355*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status Words per packet (72 bytes for FS).

  – The non-periodic request queue depth = 4.

Figure 355. Bulk/control IN transactions



The sequence of operations is as follows:

a) Initialize channel 2.

b) Set the CHENA bit in HCCHAR2 to write an IN request to the non-periodic request queue.

c) The core attempts to send an IN token after completing the current OUT transaction.

d) The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.

e) In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.

f) The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.

g) The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).

h) The core generates the XFRC interrupt as soon as the receive packet status is read.

i) In response to the XFRC interrupt, disable the channel and stop writing the OTG_FS_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_FS_HCCHAR2 register is written.

j) The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.

k) Read and ignore the receive packet status.

l) The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.

m) In response to the CHH interrupt, de-allocate the channel for other transfers.

n) Handling non-ACK responses

- **Control transactions**

  Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup-, Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in OTG_FS_HCCHAR1 to Control. During the Setup stage, the application is expected to set the PID field in OTG_FS_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 356*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

  – The periodic transmit FIFO can hold one packet (1 KB)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

  a) Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_FS_HCCHAR1.

  b) Write the first packet for channel 1.

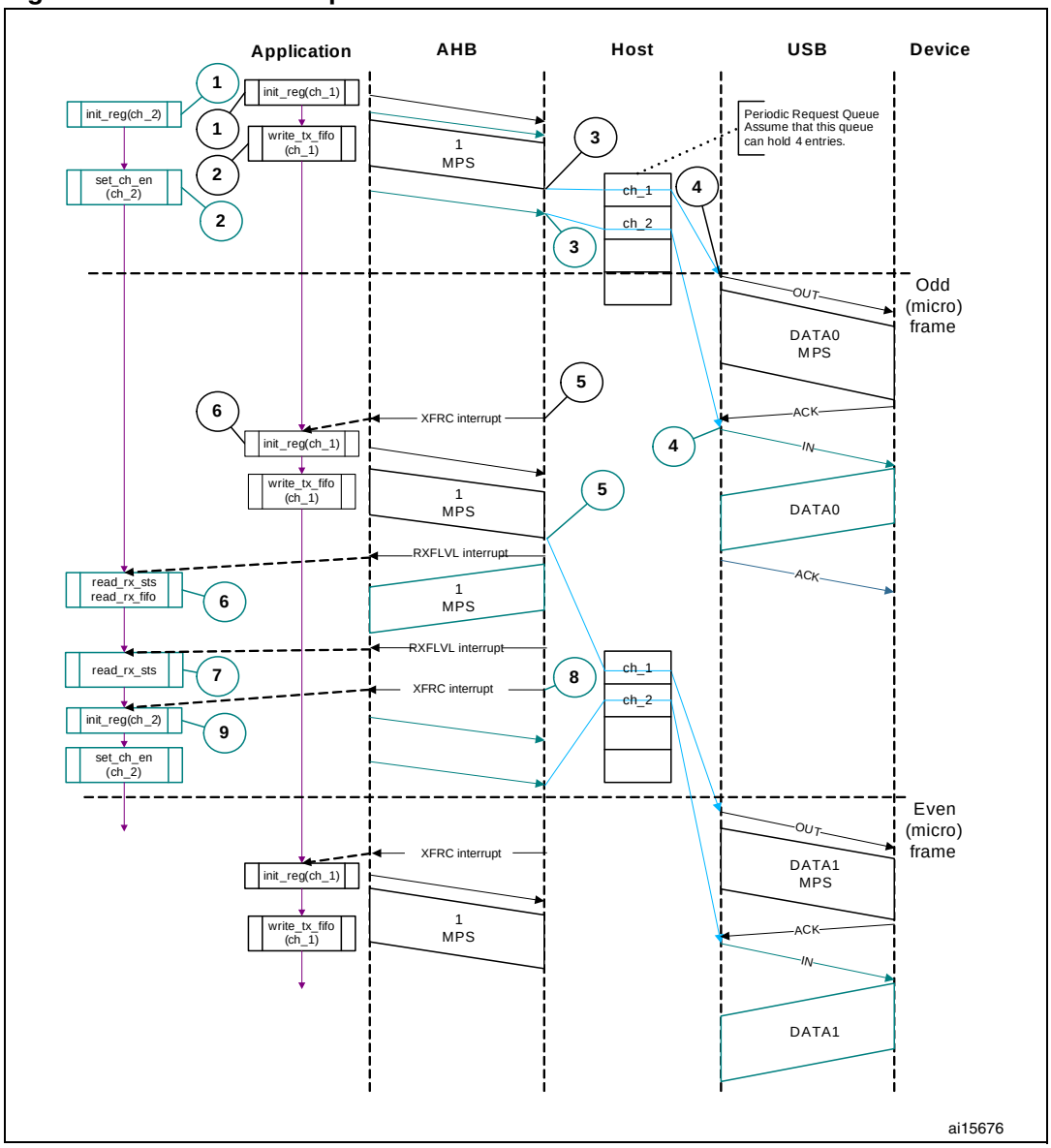
  c) Along with the last Word write of each packet, the OTG_FS host writes an entry to the periodic request queue.

  d) The OTG_FS host attempts to send an OUT token in the next (odd) frame.

  e) The OTG_FS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.

  f) In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 356. Normal interrupt OUT/IN transactions



- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
```

```
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

The application uses the NPTXFE interrupt in OTG_FS_GINTSTS to find the transmit FIFO space.

b) Interrupt IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_FS_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
```

```
      }
    }
  else
    if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
        {
        Reset Error Count
        Transfer Done = 1
        }
      else
        if (!FRMOR)
        {
        Reset Error Count
        }
      }
  else
    if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
  else
    if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
      else
        Re-initialize Channel (in next b_interval - 1 /Frame)
      }
    }
  else
    if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status Words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

- **Normal interrupt IN operation**

  The sequence of operations is as follows:

  a) Initialize channel 2. The application must set the ODDFRM bit in OTG_FS_HCCHAR2.

  b) Set the CHENA bit in OTG_FS_HCCHAR2 to write an IN request to the periodic request queue.

  c) The OTG_FS host writes an IN request to the periodic request queue for each OTG_FS_HCCHAR2 register write with the CHENA bit set.

  d) The OTG_FS host attempts to send an IN token in the next (odd) frame.

  e) As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.

  f) In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.

  g) The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).

  h) The core generates an XFRC interrupt as soon as the receive packet status is read.

  i) In response to the XFRC interrupt, read the PKTCNT field in OTG_FS_HCTSIZ2. If the PKTCNT bit in OTG_FS_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_FS_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_FS_HCCHAR2.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 357*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum packet size), starting with an odd frame. (transfer size = 1 024 bytes).

  – The periodic transmit FIFO can hold one packet (1 KB).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

  a) Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_FS_HCCHAR1.

  b) Write the first packet for channel 1.

  c) Along with the last Word write of each packet, the OTG_FS host writes an entry to the periodic request queue.

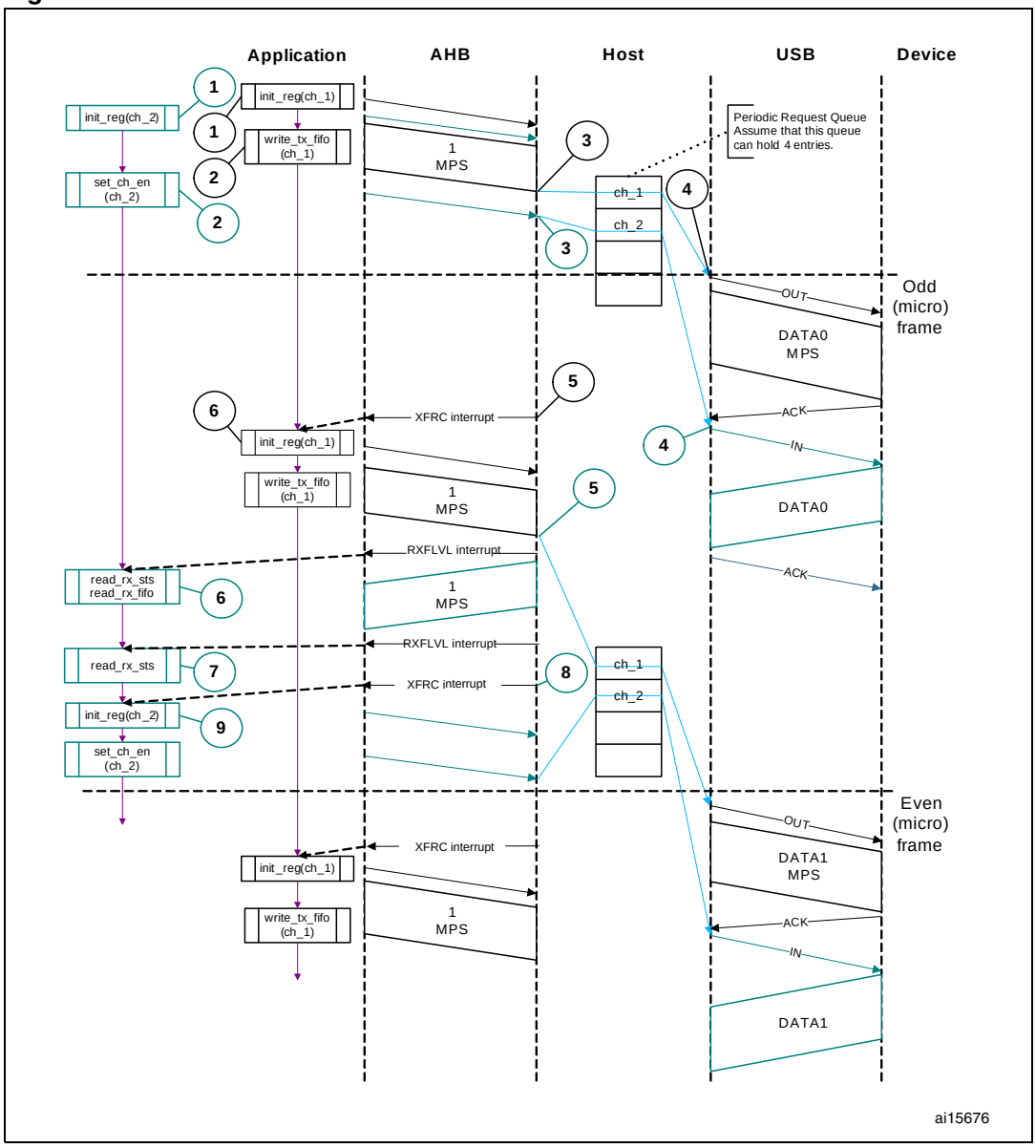
  d) The OTG_FS host attempts to send the OUT token in the next frame (odd).

  e) The OTG_FS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.

  f) In response to the XFRC interrupt, reinitialize the channel for the next transfer.

  g) Handling non-ACK responses

Figure 357. Normal isochronous OUT/IN transactions



- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: Isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
```

```
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
```

Code sample: Isochronous IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_FS_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
            else
               {
               Re-initialize Channel
               }
            }
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status Word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

  a) Initialize channel 2. The application must set the ODDFRM bit in OTG_FS_HCCHAR2.

  b) Set the CHENA bit in OTG_FS_HCCHAR2 to write an IN request to the periodic request queue.

  c) The OTG_FS host writes an IN request to the periodic request queue for each OTG_FS_HCCHAR2 register write with the CHENA bit set.

  d) The OTG_FS host attempts to send an IN token in the next odd frame.

  e) As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.

  f) In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.

  g) The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_FS_GRXSTSR ≠ 0b0010).

  h) The core generates an XFRC interrupt as soon as the receive packet status is read.

  i) In response to the XFRC interrupt, read the PKTCNT field in OTG_FS_HCTSIZ2. If PKTCNT≠ 0 in OTG_FS_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_FS_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_FS_HCCHAR2.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

  The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG_FS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for

the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When OTG_FS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

When OTG_FS controller detects a port babble, it flushes the RxFIFO and disables the port. The core then generates a Port disabled interrupt (HPRTINT in OTG_FS_GINTSTS, PENCHNG in OTG_FS_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the Port Disabled interrupt) by checking POCA in OTG_FS_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

29.17.5 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_FS_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_FS_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_FS_DAINTMSK (control 0 OUT endpoint)
   - STUP = 1 in DOEPMSK
   - XFRC = 1 in DOEPMSK
   - XFRC = 1 in DIEPMSK
   - TOC = 1 in DIEPMSK
3. Set up the Data FIFO RAM for each of the FIFOs
   - Program the OTG_FS_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 Words (for the status of the control OUT data packet) + 10 Words (for setup packets).
   - Program the OTG_FS_TX0FSIZ register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_FS_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_FS_GINTSTS), read the OTG_FS_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_FS_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_FS_DCFG register with the device address received in the SetAddress command
1. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_FS_DAINTMSK register.
5. Set up the Data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_FS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_FS_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - TxFIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_FS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_FS_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: 1 The application must meet the following conditions to set up the device core to handle traffic: NPTXFEM and RXFLVLM in the OTG_FS_GINTMSK register must be cleared.

29.17.6 Operational model

SETUP and OUT data transfers

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

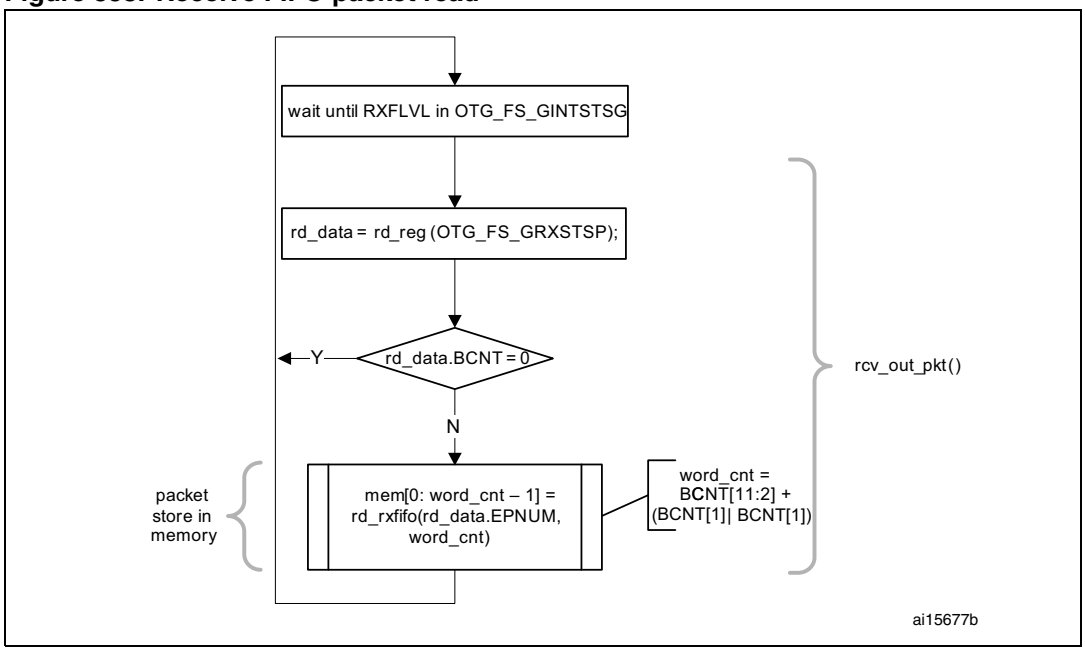
1. On catching an RXFLVL interrupt (OTG_FS_GINTSTS register), the application must read the Receive status pop register (OTG_FS_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_FS_GINTSTS) by writing to RXFLVL = 0 (in OTG_FS_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive Data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive FIFO's packet status readout indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Don't Care (0x0), DPID = Don't Care (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = D0.
   These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
   PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = Don't Care (0b00).
   These data indicate that the Setup stage for the specified endpoint has completed and the Data stage has started. After this entry is popped from the receive FIFO, the core asserts a Setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
   PKTSTS = DataOUT, BCNT = size of the received data OUT packet ($0 \leq$ BCNT $\leq$ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
   PKTSTS = Data OUT Transfer Done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = Don't Care (0b00).
   These data indicate that an OUT data transfer for the specified OUT endpoint has

completed. After this entry is popped from the receive FIFO, the core asserts a Transfer Completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_FS_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_FS_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 358 provides a flowchart of the above procedure.

Figure 358. Receive FIFO packet read



- **SETUP transactions**

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_FS_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_FS_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to

determine the correct number of SETUP packets received in the Setup stage of a control transfer.
   – STUPCNT = 3 in OTG_FS_DOEPTSIZx
2. The application must always allocate some extra space in the Receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 Words. Three Words are required for the first SETUP packet, 1 Word is required for the Setup stage done Word and 6 Words are required to store two extra SETUP packets among all control endpoints.
   – 3 Words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup packet pattern). The core reserves this space in the receive data.
   – FIFO to write SETUP data only, and never uses this space for data packets.
3. The application must read the 2 Words of the SETUP packet from the receive FIFO.
4. The application must read and discard the Setup stage done Word from the receive FIFO.

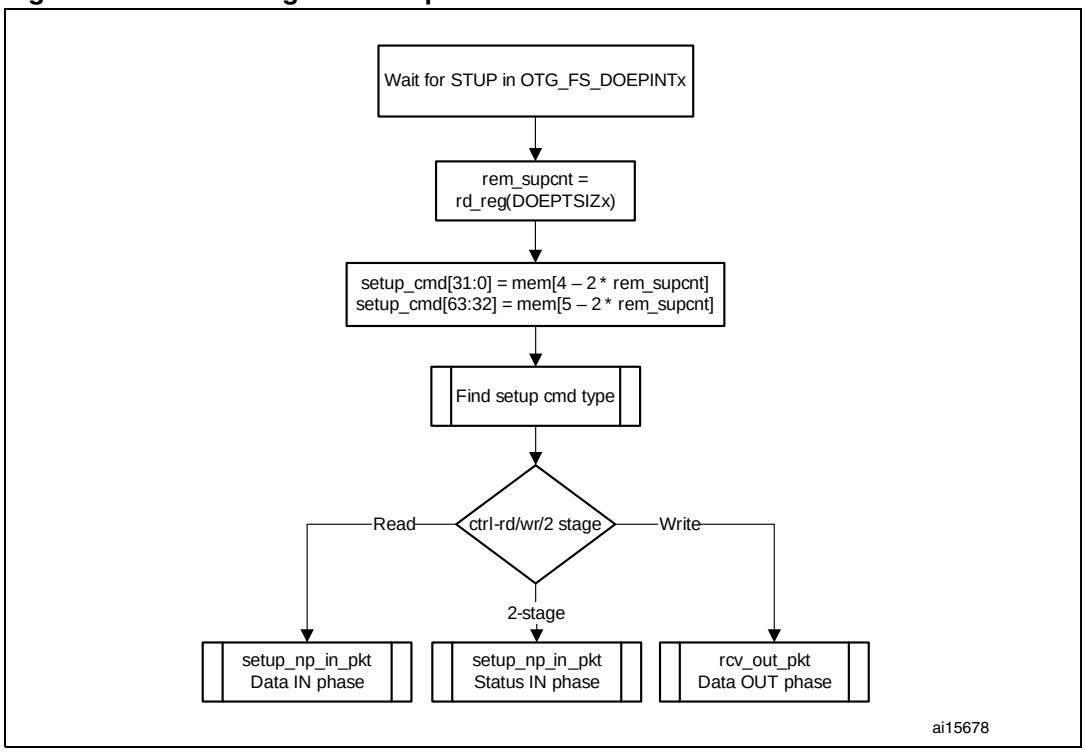
- **Internal data flow**

5. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
6. For every SETUP packet received on the USB, 3 Words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first Word contains control information used internally by the core
   – The second Word contains the first 4 bytes of the SETUP command
   – The third Word contains the last 4 bytes of the SETUP command
7. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup stage done Word) to the receive FIFO, indicating the completion of the Setup stage.
8. On the AHB side, SETUP packets are emptied by the application.
9. When the application pops the Setup stage done Word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_FS_DOEPINTx), indicating it can process the received SETUP packet.
   – The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_FS_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_FS_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_FS_DOEPINTx) marks a successful completion of the SETUP Data Transfer.
   – On this interrupt, the application must read the OTG_FS_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 359. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_FS controller generates an interrupt (B2BSTUP in OTG_FS_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_FS_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern Word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_FS_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_FS_DCTL.

Application programming sequence

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_FS_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_FS_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_FS_DCTL and before the core asserts the GONAKEFF interrupt (OTG_FS_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GINAKEFFM bit in the OTG_FS_GINTMSK register.
   – GINAKEFFM = 0 in the OTG_FS_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_FS_DCTL. This also clears the GONAKEFF interrupt (OTG_FS_GINTSTS).
   – OTG_FS_DCTL = 1 in CGONAK
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GINAKEFFM = 1 in GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_FS_DCTL
2. Wait for the GONAKEFF interrupt (OTG_FS_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_FS_DOEPCTLx
   – SNAK = 1 in OTG_FS_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_FS_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_FS_DOEPCTLx
   – EPENA = 0 in OTG_FS_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_FS_DCTL

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the Word boundary.
   – transfer size[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = n
   – $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   – OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   – After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   – If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   – In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.

6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   – The transfer size is 0 and the packet count is 0
   – The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_FS_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_FS_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_FS_DOEPCTLx
   – CNAK = 1 in OTG_FS_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_FS_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_FS_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_FS_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_FS_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_FS_GINTSTS) and before the SOF (OTG_FS_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_FS_DOEPCTLx) = SOFFN[0] (in OTG_FS_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in

OTG_FS_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_FS_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_FS_DOEPCTLx register with the endpoint characteristics and set the Endpoint Enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_FS_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_FS_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the IISOOXFRM interrupt in OTG_FS_GINTSTS.
6. Read the OTG_FS_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = D0 (in OTG_FS_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = D1 (in OTG_FS_DOEPTSIZx) and the number of USB packets in which this payload was received = 2
   – RXDPID = D2 (in OTG_FS_DOEPTSIZx) and the number of USB packets in which this payload was received = 3

   The number of USB packets in which this payload was received = Application programmed initial packet count – Core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_FS_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application could fail to detect the XFRC interrupt (OTG_FS_DOEPINTx) under the following circumstances:
   – When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   – When the isochronous OUT data packet is received with CRC errors
   – When the isochronous OUT token received by the core is corrupted
   – When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete Isochronous OUT data interrupt

(IISOOXFRM in OTG_FS_GINTSTS), indicating that an XFRC interrupt (in OTG_FS_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the IISOOXFRM interrupt (OTG_FS_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_FS_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an IISOOXFRM interrupt (in OTG_FS_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_FS_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_FS_DOEPCTLx) = SOFFN[0] (in OTG_FS_DSTS)
   – EPENA = 1 (in OTG_FS_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_FS_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_FS_DOEPCTLx.
6. Wait for the EPDIS interrupt (in OTG_FS_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_FS_DOEPCTL, set STALL = 1 (in OTG_FS_DOEPCTL).

   The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_FS_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

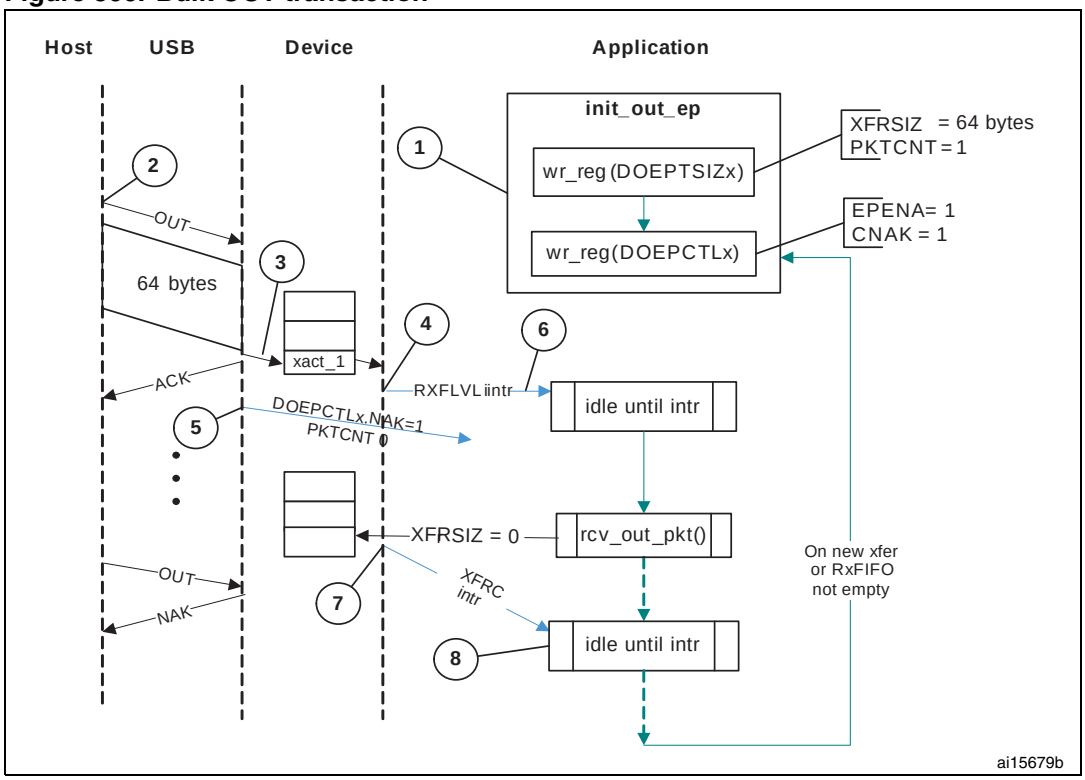
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 360 depicts the reception of a single Bulk OUT Data packet from the USB to the AHB and describes the events involved in the process.

Figure 360. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_FS_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_FS_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the RxFIFO because space is available there.
3. After writing the complete packet in the RxFIFO, the core then asserts the RXFLVL interrupt (in OTG_FS_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the RxFIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_FS_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_FS_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_FS_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - In interrupt mode, the application waits for the TXFE interrupt (in OTG_FS_DIEPINTx) and then reads the OTG_FS_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   - To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_FS_DIEPCTLx register to avoid modifying the contents of the register, except for setting the Endpoint Enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   - Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_FS_DIEPINTx in response to the SNAK bit in OTG_FS_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_FS_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_FS_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_FS_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in DIEPMSK.
   – INEPNEM = 0 in DIEPMSK
5. To exit Endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_FS_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_FS_DIEPINTx).
   – CNAK = 1 in OTG_FS_DIEPCTLx
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_FS_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_FS_DIEPINTx.
4. Set the following bits in the OTG_FS_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_FS_DIEPCTLx
   – SNAK = 1 in OTG_FS_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_FS_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_FS_DIEPCTLx
   – EPDIS = 0 in OTG_FS_DIEPCTLx
6. The application must read the OTG_FS_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the Endpoint transmit FIFO, by setting the following fields in the OTG_FS_GRSTCTL register:
   – TXFNUM (in OTG_FS_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_FS_GRSTCTL) = 1

The application must poll the OTG_FS_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:

     Transfer size[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp

     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = x
   – To transmit a single zero-length data packet:

     Transfer size[EPNUM] = 0

     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

     First transfer: transfer size[EPNUM] = $x \times$ MPSIZ[epnum]; packet count = n;

     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the Transfer size register. At the end of the IN transfer, the application must read the Transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – Core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK

handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.

5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when TxFIFO is empty” (ITTXFE) Interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_FS_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_FS_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (Endpoint Enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_FS_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_FS_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 1054* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

     transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
     (where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

     If (sp > 0), packet count[EPNUM] = *x* + 1
     Otherwise, packet count[EPNUM] = *x*;

     MCNT[EPNUM] = packet count[EPNUM]
   - The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
   - transfer size[EPNUM] = 0

     packet count[EPNUM] = 1

     MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_FS_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_FS_DIEPTSIZx, MPSIZ is in OTG_FS_DIEPCTLx, PKTCNT is in OTG_FS_DIEPTSIZx and XFERSIZ is in OTG_FS_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 Word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN token received when TxFIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_FS_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_FS_GINTSTS.

Application programming sequence:

1. Program the OTG_FS_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_FS_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_FS_DIEPINTx) with no ITTXFE interrupt in OTG_FS_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_FS_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_FS_DIEPINTx), with or without the ITTXFE interrupt (in OTG_FS_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_FS_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_FS_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_FS_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when TxFIFO empty interrupt in OTG_FS_DIEPINTx. The application can ignore this interrupt, as it eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_FS_GINTSTS) at the end of periodic frame.

      The core transmits a zero-length data packet on the USB in response to the received IN token.
2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the Endpoint Disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when TxFIFO empty interrupt in OTG_FS_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_FS_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_FS_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the Endpoint Control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_FS_DIEPCTLx register to disable the endpoint:
   - SNAK = 1 in OTG_FS_DIEPCTLx
   - EPDIS = 1 in OTG_FS_DIEPCTLx
6. The assertion of the Endpoint Disabled interrupt in OTG_FS_DIEPINTx indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_FS_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_FS_DIEPCTLx, when the endpoint is already enabled
   - STALL = 1 in OTG_FS_DIEPCTLx
   - The STALL bit always takes precedence over the NAK bit
3. Assertion of the Endpoint Disabled interrupt (in OTG_FS_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_FS_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_FS_DIEPINTx and the OTEPDIS interrupt in OTG_FS_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the

application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

29.17.7 Worst case response time

When the OTG_FS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_FS_GUSBCFG

The value in TRDT (OTG_FS_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_FS_GUSBCFG).

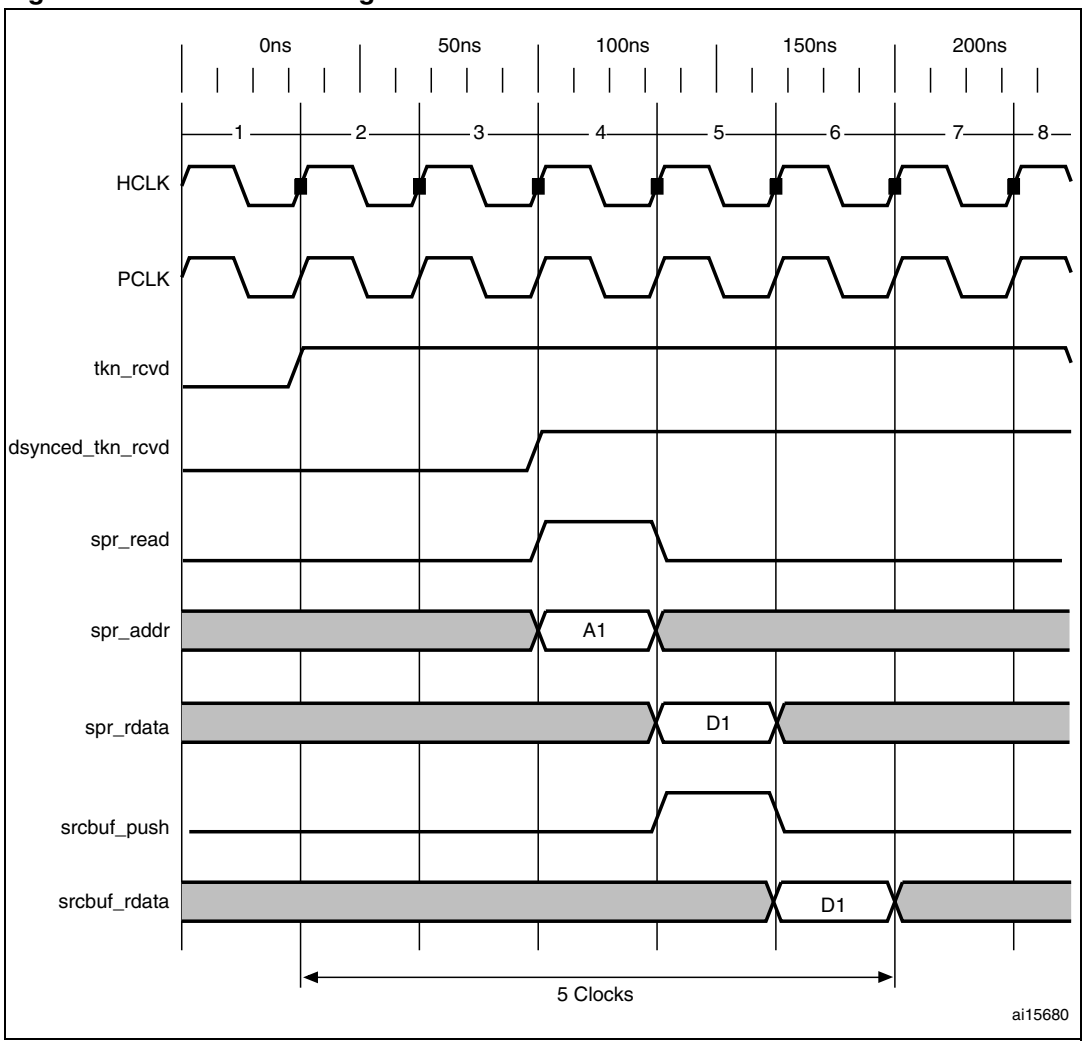
Figure 361 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

The application can use the following formula to calculate the value of TRDT:

4 × AHB clock + 1 PHY clock = (2 clock sync + 1 clock memory address + 1 clock memory data from sync RAM) + (1 PHY clock (next PHY clock MAC can sample the 2 clock FIFO outputs)

Figure 361. TRDT max timing case



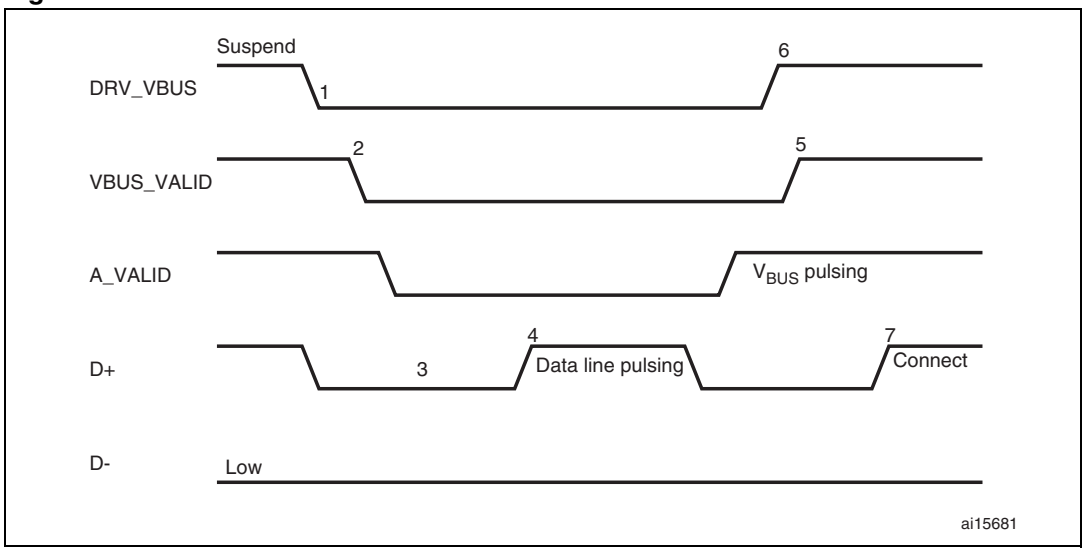
29.17.8 OTG programming model

The OTG_FS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG_FS controller turns off V_{BUS} to conserve power. SRP is a method by which the B-device signals the A-device to turn on V_{BUS} power. A device must perform both data-line pulsing and V_{BUS} pulsing, but a host can detect either data-line pulsing or V_{BUS} pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the Core USB configuration register. This enables the OTG_FS controller to detect SRP as an A-device.

Figure 362. A-device SRP



1. DRV_VBUS = V_{BUS} drive signal to the PHY
   VBUS_VALID = V_{BUS} valid signal from PHY
   A_VALID = A-peripheral V_{BUS} level signal to PHY
   D+ = Data plus line
   D- = Data minus line

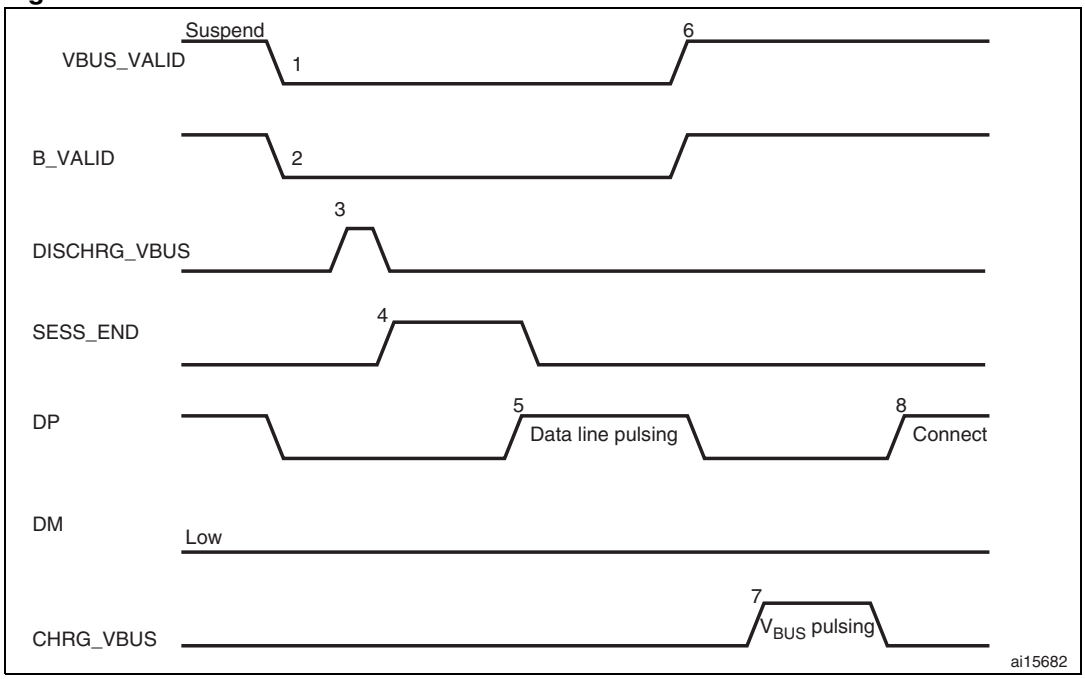
1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when V_{BUS} power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG_FS controller detects data-line pulsing.
5. The device drives V_{BUS} above the A-device session valid (2.0 V minimum) for V_{BUS} pulsing.

   The OTG_FS controller interrupts the application on detecting SRP. The Session request detected bit is set in Global interrupt status register (SRQINT set in OTG_FS_GINTSTS).
6. The application must service the Session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

B-device session request protocol

The application must set the SRP-capable bit in the Core USB configuration register. This enables the OTG_FS controller to initiate SRP as a B-device. SRP is a means by which the OTG_FS controller can request a new session from the host.

Figure 363. B-device SRP



1. VBUS_VALID = V_{BUS} valid signal from PHY
   B_VALID = B-peripheral valid session to PHY
   DISCHRG_VBUS = discharge signal to PHY
   SESS_END = session end signal to PHY
   CHRG_VBUS = charge V_{BUS} signal to PHY
   DP = Data plus line
   DM = Data minus line

1. To save power, the host suspends and turns off port power when the bus is idle.

   The OTG_FS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB suspend bit in the Core interrupt register.

   The OTG_FS controller informs the PHY to discharge V_{BUS}.

2. The PHY indicates the session's end to the device. This is the initial condition for SRP. The OTG_FS controller requires 2 ms of SE0 before initiating SRP.

   For a USB 1.1 full-speed serial transceiver, the application must wait until V_{BUS} discharges to 0.2 V after BSVLD (in OTG_FS_GOTGCTL) is deasserted. This

discharge time can be obtained from the transceiver vendor and varies from one transceiver to another.

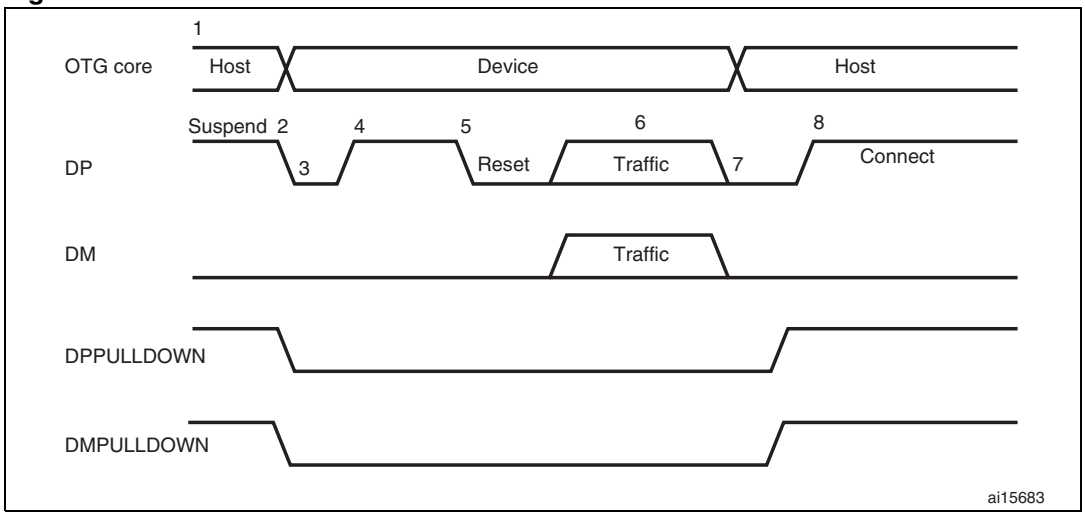
3. The USB OTG core informs the PHY to speed up V_{BUS} discharge.
4. The application initiates SRP by writing the session request bit in the OTG Control and status register. The OTG_FS controller perform data-line pulsing followed by V_{BUS} pulsing.
5. The host detects SRP from either the data-line or V_{BUS} pulsing, and turns on V_{BUS}. The PHY indicates V_{BUS} power-on to the device.
6. The OTG_FS controller performs V_{BUS} pulsing.

   The host starts a new session by turning on V_{BUS}, indicating SRP success. The OTG_FS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
7. When the USB is powered, the OTG_FS controller connects, completing the SRP process.

A-device host negotiation protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_FS controller to perform HNP as an A-device.

Figure 364. A-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. The OTG_FS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP Enable bit in the OTG Control

and status register to indicate to the OTG_FS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the Port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The OTG_FS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

   The OTG_FS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_FS_DP pull-up resistor to indicate a connect for B-device.

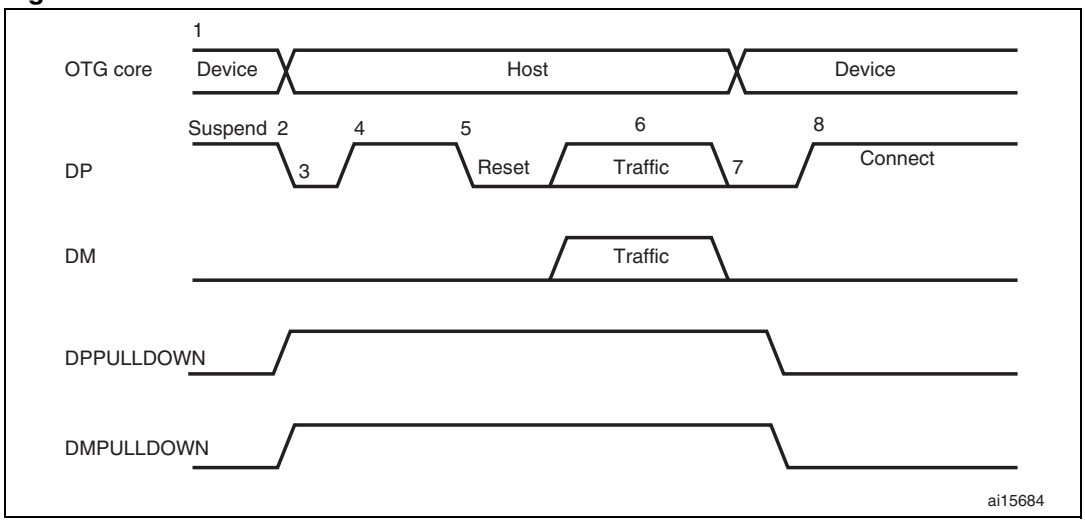
   The application must read the current mode bit in the OTG Control and status register to determine device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_FS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The OTG_FS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB Suspend bit in the Core interrupt register.
6. In Negotiated mode, the OTG_FS controller detects the suspend, disconnects, and switches back to the host role. The OTG_FS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_FS controller sets the Connector ID status change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the OTG_FS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

B-device host negotiation protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_FS controller to perform HNP as a B-device.

Figure 365. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG_FS controller's ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG Control and status register to indicate HNP support.

   The application sets the HNP request bit in the OTG Control and status register to indicate to the OTG_FS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the Port suspend bit in the host port control and status register.

   The OTG_FS controller sets the Early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB suspend bit in the Core interrupt register.

   The OTG_FS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG_FS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

   The A-device responds by activating its OTG_FS_DP pull-up resistor within 3 ms of detecting SE0. The OTG_FS controller detects this as a connect.

   The OTG_FS controller sets the host negotiation success status change interrupt in the OTG Interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG Control and status register to determine host

negotiation success. The application must read the current Mode bit in the Core interrupt register (OTG_FS_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG_FS_HPRT) and the OTG_FS controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG_FS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the Port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG_FS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the Core interrupt (OTG_FS_GINTSTS) register to determine the host mode operation.
7. The OTG_FS controller connects, completing the HNP process.

30 USB on-the-go high-speed (OTG_HS)

30.1 OTG_HS introduction

This section presents the architecture and the programming model of the OTG_HS controller.

The following acronyms are used throughout the section:

FS	full-speed
HS	High-speed
LS	Low-speed
USB	Universal serial bus
OTG	On-the-go
PHY	Physical layer
MAC	Media access controller
PFC	Packet FIFO controller
UTMI	USB Transceiver Macrocell Interface
ULPI	UTMI+ Low Pin Interface

References are made to the following documents:

- USB On-The-Go Supplement, Revision 1.3
- Universal Serial Bus Revision 2.0 Specification

The OTG_HS is a dual-role device (DRD) controller that supports both peripheral and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or peripheral-only controller, fully compliant with the *USB 2.0 Specification.* In host mode, the OTG_HS supports high-speed (HS, 480 Mbits/s), full-speed (FS, 12 Mbits/s) and low-speed (LS, 1.5 Mbits/s) transfers whereas in peripheral mode, it only supports high-speed (HS, 480Mbits/s) and full-speed (FS, 12 Mbits/s) transfers. The OTG_HS supports both HNP and SRP. The only external device required is a charge pump for VBUS in OTG mode.

30.2 OTG_HS main features

The main features can be divided into three categories: general, host-mode and peripheral-mode features.

30.2.1 General features

The OTG_HS interface main features are the following:

- It is USB-IF certified in compliance with the Universal Serial Bus Revision 2.0 Specification
- It supports 3 PHY interfaces
  - An on-chip full-speed PHY
  - An I^2C Interface for external full-speed I^2C PHY
  - An ULPI interface for external high-speed PHY.
- It supports the host negotiation protocol (HNP) and the session request protocol (SRP)
- It allows the host to turn V_{BUS} off to save power in OTG applications, with no need for external components
- It allows to monitor V_{BUS} levels using internal comparators
- It supports dynamic host-peripheral role switching
- It is software-configurable to operate as:
  - An SRP-capable USB HS/FS peripheral (B-device)
  - An SRP-capable USB HS/FS/low-speed host (A-device)
  - An USB OTG FS dual-role device
- It supports HS/FS SOFs as well as low-speed (LS) keep-alive tokens with:
  - SOF pulse PAD output capability
  - SOF pulse internal connection to timer 2 (TIM2)
  - Configurable framing period
  - Configurable end-of-frame interrupt
- It embeds an internal DMA with shareholding support and software selectable AHB burst type in DMA mode
- It has power saving features such as system clock stop during USB suspend, switching off of the digital core internal clock domains, PHY and DFIFO power management
- It features a dedicated 4-Kbyte data RAM with advanced FIFO management:
  - The memory partition can be configured into different FIFOs to allow flexible and efficient use of RAM
  - Each FIFO can contain multiple packets
  - Memory allocation is performed dynamically
  - The FIFO size can be configured to values that are not powers of 2 to allow the use of contiguous memory locations
- It ensures a maximum USB bandwidth of up to one frame without application intervention

30.2.2 Host-mode features

The OTG_HS interface features in host mode are the following:

- It requires an external charge pump to generate V_{BUS}
- It has up to 12 host channels (pipes), each channel being dynamically reconfigurable to support any kind of USB transfer
- It features a built-in hardware scheduler holding:
  - Up to 8 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 8 control plus bulk transfer requests in the nonperiodic hardware queue
- It manages a shared RX FIFO, a periodic TX FIFO, and a nonperiodic TX FIFO for efficient usage of the USB data RAM
- It features dynamic trimming capability of SOF framing period in host mode.

30.2.3 Peripheral-mode features

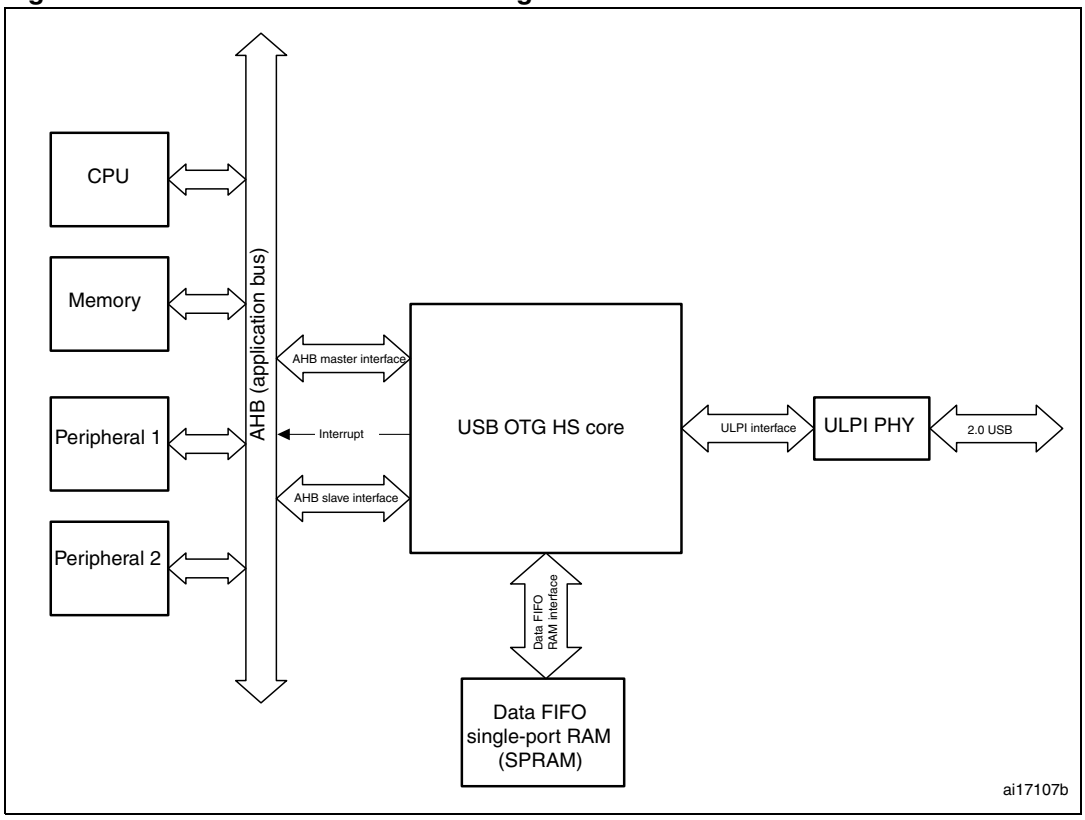
The OTG_HS interface main features in peripheral mode are the following:

- It has 1 bidirectional control endpoint 0
- It has 5 IN endpoints (EP) configurable to support bulk, interrupt or isochronous transfers
- It has 5 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- It manages a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- It manages up to 6 dedicated Tx-IN FIFOs (one for each IN-configured EP) to reduce the application load
- It features soft disconnect capability

30.3 OTG_HS functional description

Figure 366 shows the OTG_HS interface block diagram.

Figure 366. USB OTG interface block diagram



1. The USB DMA cannot directly address the internal Flash memory.

30.3.1 High-speed OTG PHY

The USB OTG HS core embeds an ULPI interface to connect an external HS phy.

30.3.2 External Full-speed OTG PHY using the I2C interface

The USB OTG HS core embeds an I^2C interface allowing to connect an external FS phy.

30.3.3 Embedded Full-speed OTG PHY

The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and Device. It directly drives transmission and reception on the single-ended USB lines.
- Integrated ID pull-up resistor used to sample the ID line for A/B Device identification.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_HS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as V_{BUS} is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both

DP/DM. Pull-up and pull-down resistors are dynamically switched when the peripheral role is changed via the host negotiation protocol (HNP).

- Pull-up/pull-down resistor ECN circuit

  The DP pull-up consists of 2 resistors controlled separately from the OTG_HS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows to achieve a better noise rejection and Tx/Rx signal quality.
- V_{BUS} sensing comparators with hysteresis used to detect VBUS_VALID, A-B Session Valid and session-end voltage thresholds. They are used to drive the session request protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the V_{BUS} supply during USB operations.
- V_{BUS} pulsing method circuit used to charge/discharge V_{BUS} through resistors during the SRP (weak drive).

Caution: To guarantee a correct operation for the USB OTG HS peripheral, the AHB frequency should be higher than 30 MHz.

30.4 OTG dual-role device

30.4.1 ID line detection

The host or peripheral (the default) role depends on the level of the ID input line. It is determined when the USB cable is plugged in and depends on which side of the USB cable is connected to the micro-AB receptacle:

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_HS conforms to the FSM standard described in section 6.8.2. On-The-Go B-device of the USB On-The-Go Supplement, Revision 1.3.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_HS issues an ID line status change interrupt (CIDSCHG bit in the OTG_HS_GINTSTS register) for host software initialization, and automatically switches to host role. In this configuration the OTG_HS conforms to the FSM standard described by section 6.8.1: On-The-Go A-Device of the USB On-The-Go Supplement, Revision 1.3.

30.4.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in the OTG_HS_GUSBCFG register) configures the OTG_HS core to dynamically change from A-host to A-device role and vice-versa, or from B-device to B-host role and vice-versa, according to the host negotiation protocol (HNP). The current device status is defined by the combination of the Connector ID Status bit in the Global OTG control and status register (CIDSTS bit in OTG_HS_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG_HS_GINTSTS).

The HNP programming model is described in detail in *Section 30.13: OTG_HS programming model*.

30.4.3 SRP dual-role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_HS_GUSBCFG) configures the OTG_HS core to switch V_{BUS} off for the A-device in

order to save power. The A-device is always in charge of driving V_{BUS} regardless of the OTG_HS role (host or peripheral). The SRP A/B-device program model is described in detail in *Section 30.13: OTG_HS programming model*.

30.5 USB functional description in peripheral mode

The OTG_HS operates as an USB peripheral in the following circumstances:

- OTG B-device

  OTG B-device default state if the B-side of USB cable is plugged in
- OTG A-device

  OTG A-device state after the HNP switches the OTG_HS to peripheral role
- B-Device

  If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG_HS_GUSBCFG) is cleared (see On-The-Go specification Revision 1.3 section 6.8.3).
- Peripheral only (see *Figure 344: USB peripheral-only connection*)

  The force peripheral mode bit in the Global USB configuration register (FDMOD in OTG_HS_GUSBCFG) is set to 1, forcing the OTG_HS core to operate in USB peripheral-only mode (see On-The-Go specification Revision 1.3 section 6.8.3). In this case, the ID line is ignored even if it is available on the USB connector.

Note: To build a bus-powered device architecture in the B-Device or peripheral-only configuration, an external regulator must be added to generate the V_{DD} supply voltage from V_{BUS}.

30.5.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG_HS_GUSBCFG) configures the OTG_HS to support the session request protocol (SRP). As a result, it allows the remote A-device to save power by switching V_{BUS} off when the USB session is suspended.

The SRP peripheral mode program model is described in detail in *Section : B-device session request protocol*.

30.5.2 Peripheral states

Powered state

The V_{BUS} input detects the B-session valid voltage used to put the USB peripheral in the Powered state (see USB2.0 specification section 9.1). The OTG_HS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host, and generates the session request interrupt (SRQINT bit in OTG_HS_GINTSTS) to notify the Powered state. The V_{BUS} input also ensures that valid V_{BUS} levels are supplied by the host during USB operations. If V_{BUS} drops below the B-session valid voltage (for example because power disturbances occurred or the host port has been switched off), the OTG_HS automatically disconnects and the session end detected (SEDET bit in OTG_HS_GOTGINT) interrupt is generated to notify that the OTG_HS has exited the Powered state.

In Powered state, the OTG_HS expects a reset from the host. No other USB operations are possible. When a reset is received, the reset detected interrupt (USBRST in OTG_HS_GINTSTS) is generated. When the reset is complete, the enumeration done interrupt (ENUMDNE bit in OTG_HS_GINTSTS) is generated and the OTG_HS enters the Default state.

Soft disconnect

The Powered state can be exited by software by using the soft disconnect feature. The DP pull-up resistor is removed by setting the Soft disconnect bit in the device control register (SDIS bit in OTG_HS_DCTL), thus generating a device disconnect detection interrupt on the host side even though the USB cable was not really unplugged from the host port.

Default state

In Default state the OTG_HS expects to receive a SET_ADDRESS command from the host. No other USB operations are possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_HS_DCFG). The OTG_HS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_HS peripheral constantly monitors the USB activity. When the USB remains idle for 3 ms, the early suspend interrupt (ESUSP bit in OTG_HS_GINTSTS) is issued. It is confirmed 3 ms later, if appropriate, by generating a suspend interrupt (USBSUSP bit in OTG_HS_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_HS_DSTS) and the OTG_HS enters the Suspended state.

The device can also exit from the Suspended state by itself. In this case the application sets the remote wakeup signaling bit in the device control register (WKUPINT bit in OTG_HS_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (RWUSIG bit in OTG_HS_GINTSTS) is generated and the device suspend bit is automatically cleared.

30.5.3 Peripheral endpoints

The OTG_HS core instantiates the following USB endpoints:

- Control endpoint 0

  This endpoint is bidirectional and handles control messages only.

  It has a separate set of registers to handle IN and OUT transactions, as well as dedicated control (OTG_HS_DIEPCTL0/OTG_HS_DOEPCTL0), transfer configuration (OTG_HS_DIEPTSIZ0/OTG_HS_DIEPTSIZ0), and status-interrupt

(OTG_HS_DIEPINTx/)OTG_HS_DOEPINT0) registers. The bits available inside the control and transfer size registers slightly differ from other endpoints.

- 5 IN endpoints
  - They can be configured to support the isochronous, bulk or interrupt transfer type.
  - They feature dedicated control (OTG_HS_DIEPCTLx), transfer configuration (OTG_HS_DIEPTSIZx), and status-interrupt (OTG_HS_DIEPINTx) registers.
  - The Device IN endpoints common interrupt mask register (OTG_HS_DIEPMSK) allows to enable/disable a single endpoint interrupt source on all of the IN endpoints (EP0 included).
  - They support incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_HS_GINTSTS). This interrupt is asserted when there is at least one isochronous IN endpoint for which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_HS_GINTSTS/EOPF).
- 5 OUT endpoints
  - They can be configured to support the isochronous, bulk or interrupt transfer type.
  - They feature dedicated control (OTG_HS_DOEPCTLx), transfer configuration (OTG_HS_DOEPTSIZx) and status-interrupt (OTG_HS_DOEPINTx) registers.
  - The Device Out endpoints common interrupt mask register (OTG_HS_DOEPMSK) allows to enable/disable a single endpoint interrupt source on all OUT endpoints (EP0 included).
  - They support incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_HS_GINTSTS). This interrupt is asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_HS_GINTSTS/EOPF).

Endpoint controls

The following endpoint controls are available through the device endpoint-x IN/OUT control register (DIEPCTLx/DOEPCTLx):

- Endpoint enable/disable
- Endpoint activation in current configuration
- Program the USB transfer type (isochronous, bulk, interrupt)
- Program the supported packet size
- Program the Tx-FIFO number associated with the IN endpoint
- Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
- Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
- Optionally program the NAK bit to always send a negative acknowledge to the host regardless of the FIFO status
- Optionally program the STALL bit to always stall host tokens to that endpoint
- Optionally program the Snoop mode for OUT endpoint where the received data CRC is not checked

Endpoint transfer

The device endpoint-x transfer size registers (DIEPTSIZx/DOEPTSIZx) allow the application to program the transfer size parameters and read the transfer status.

The programming operation must be performed before setting the endpoint enable bit in the endpoint control register.

Once the endpoint is enabled, these fields are read-only as the OTG FS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets constituting the overall transfer size.

Endpoint status/interrupt

The device endpoint-x interrupt registers (DIEPINTx/DOPEPINTx) indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_HS_GINTSTS or IEPINT bit in OTG_HS_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt register (OTG_HS_DAINT) to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the DAINT and GINTSTS registers.

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer has completed on both the application (AHB) and USB sides
- Setup stage done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge transmitted to the host (isochronous-in only)
- IN token received when Tx-FIFO was empty (bulk-in/interrupt-in only)
- OUT token received when endpoint was not yet enabled
- Babble error condition detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet dropped without generating an interrupt

30.6 USB functional description on host mode

This section gives the functional description of the OTG_HS in the USB host mode. The OTG_HS works as a USB host in the following circumstances:

- OTG A-host

  OTG A-device default state when the A-side of the USB cable is plugged in
- OTG B-host

  OTG B-device after HNP switching to the host role
- A-device

  If the ID line is present, functional and connected to the A-side of the USB cable, and the HNP-capable bit is cleared in the Global USB Configuration register (HNPCAP bit in OTG_HS_GUSBCFG). Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only (*Figure 345: USB host-only connection*).

  The force host mode bit in the global USB configuration register (FHMOD bit in OTG_HS_GUSBCFG) forces the OTG_HS core to operate in USB host-only mode. In this case, the ID line is ignored even if it is available on the USB connector. Integrated pull-down resistors are automatically set on the OTG_HS_FS_DP/OTG_HS_FS_DM lines.

Note: 1 On-chip 5 V V_{BUS} generation is not supported. As a result, a charge pump or a basic power switch (if a 5 V supply is available on the application board) must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

2 The V_{BUS} input ensures that valid V_{BUS} levels are supplied by the charge pump during USB operations while the charge pump overcurrent output can be input to any GPIO pin configured to generate port interrupts. The overcurrent ISR must promptly disable the V_{BUS} generation.

30.6.1 SRP-capable host

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_HS_GUSBCFG). When the SRP feature is enabled, the host can save power by switching off the V_{BUS} power while the USB session is suspended. The SRP host mode program model is described in detail in *Section : A-device session request protocol*.

30.6.2 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. As a result, a charge pump or a basic power switch (if a 5 V supply voltage is available on the application board) must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. When the application powers on V_{BUS} through the selected GPIO, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HS_HPRT).

V_{BUS} valid

The V_{BUS} input ensures that valid V_{BUS} levels are supplied by the charge pump during USB operations.

Any unforeseen V_{BUS} voltage drop below the V_{BUS} valid threshold (4.25 V) generates an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG_HS_GOTGINT). The application must then switch the V_{BUS} power off and clear the port power bit. The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input, and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Detection of peripheral connection by the host

Even if USB peripherals or B-devices can be attached at any time, the OTG_HS does not detect a bus connection until the end of the V_{BUS} sensing (V_{BUS} over 4.75 V).

When V_{BUS} is at a valid level and a remote B-device is attached, the OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG_HS_HPRT).

Detection of peripheral disconnection by the host

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_HS_GINTSTS).

Host enumeration

After detecting a peripheral connection, the host must start the enumeration process by issuing an USB reset and configuration commands to the new peripheral.

Before sending an USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG_HS_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on OTG_HS_FS_DP (full speed) or OTG_HS_FS_DM (low speed).

The application issues an USB reset (single-ended zero) via the USB by keeping the port reset bit set in the Host port control and status register (PRST bit in OTG_HS_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application monitors the time and then clears the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HS_HPRT) to inform the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HS_HPRT), and that the host is starting to drive SOFs (full speed) or keep-alive tokens (low speed). The host is then ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application can decide to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HS_HPRT). The OTG_HS core stops sending SOFs and enters the Suspended state.

The Suspended state can be exited on the remote device initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_HS_GINTSTS) is generated upon detection of a remote wakeup event, the port resume bit in the host port control and status

register (PRES bit in OTG_HS_HPRT) is set, and a resume signaling is automatically issued on the USB. The application must monitor the resume window duration, and then clear the port resume bit to exit the Suspended state and restart the SOF.

If the Suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, monitor the resume window duration and then clear the port resume bit.

30.6.3 Host channels

The OTG_HS core instantiates 12 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 8 transfer requests simultaneously. If more than 8 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support IN/OUT and any type of periodic/nonperiodic transaction. Each host channel has dedicated control (HCCHARx), transfer configuration (HCTSIZx) and status/interrupt (HCINTx) registers with associated mask (HCINTMSKx) registers.

Host channel controls

The following host channel controls are available through the host channel-x characteristics register (HCCHARx):

- Channel enable/disable
- Program the HS/FS/LS speed of target USB peripheral
- Program the address of target USB peripheral
- Program the endpoint number of target USB peripheral
- Program the transfer IN/OUT direction
- Program the USB transfer type (control, bulk, interrupt, isochronous)
- Program the maximum packet size (MPS)
- Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (HCTSIZx) allow the application to program the transfer size parameters, and read the transfer status.

The programming operation must be performed before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled, the packet count field is read-only as the OTG HS core updates it according to the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets constituting the overall transfer size
- Initial data PID

Host channel status/interrupt

The host channel-x interrupt register (HCINTx) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_HS_GINTSTS) is

set. Before the application can read these registers, it must first read the host all channels interrupt (HCAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the HAINT and GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HS_HCINTMSK-x register.

The host core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
- Channel stopped due to transfer completed, USB transaction error or disable command from the application
- Associated transmit FIFO half or completely empty (IN endpoints)
- ACK response received
- NAK response received
- STALL response received
- USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
- Babble error
- Frame overrun
- Data toggle error

30.6.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB the transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_HS_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG HS core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (HPTXSTS) and nonperiodic transmit FIFO and queue status register (HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx-FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the

USB for a maximum of 8 pending periodic transactions plus 8 pending nonperiodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HS_HNPTXSTS register or NPTQXSAV bits in the OTG_HS_HNPTXSTS register.

30.7 SOF trigger

The OTG FS core allows to monitor, track and configure SOF framing in the host and peripheral. It also features an SOF pulse output connectivity.

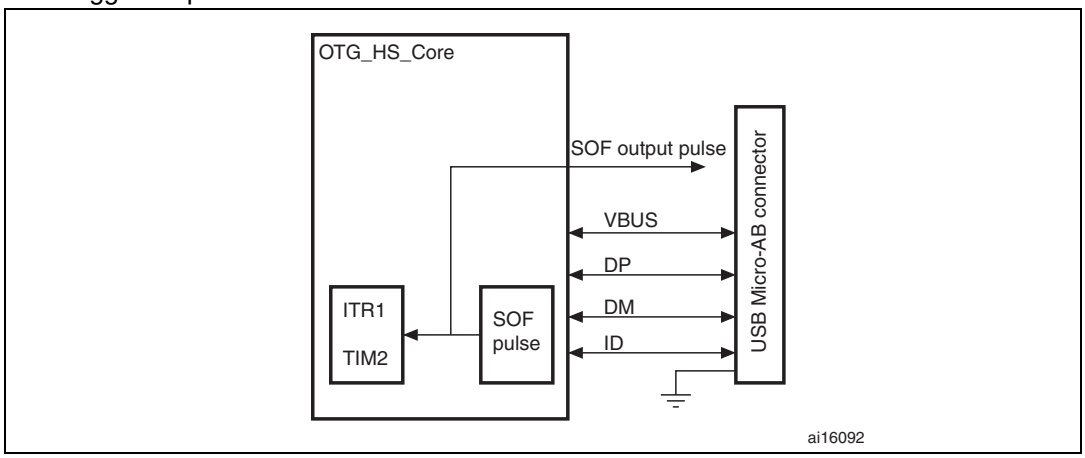
These capabilities are particularly useful to implement adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs trimming its framing rate according to the requirements of the audio peripheral.

30.7.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (FS) or keep-alive (LS) tokens is programmable in the host frame interval register (OTG_HS_HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_HS_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (OTG_HS_HFNUM).

An SOF pulse signal is generated at any SOF starting token and with a width of 12 system clock cycles. It can be made available externally on the SOF pin using the SOFOUTEN bit in the global control and configuration register. The SOF pulse is also internally connected to the input trigger of timer 2 (TIM2), so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse. The TIM2 connection is enabled through ITR1_RMP bits of TIM2_OR register.

SOF trigger output to TIM2 ITR1 connection



30.7.2 Peripheral SOFs

In peripheral mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_HS_GINTSTS). The corresponding frame number

can be read from the device status register (FNSOF bit in OTG_HS_DSTS). An SOF pulse signal with a width of 12 system clock cycles is also generated and can be made available externally on the SOF pin by using the SOF output enable bit in the global control and configuration register (SOFOUTEN bit in OTG_HS_GCCFG). The SOF pulse signal is also internally connected to the TIM2 input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse (see *Figure*). The TIM2 connection is enabled through ITR1_RMP bits of TIM2_OR register.

The end of periodic frame interrupt (GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_HS_DCFG).

This feature can be used to determine if all of the isochronous traffic for that frame is complete.

30.8 USB_HS power modes

The power consumption of the OTG PHY is controlled by three bits in the general core configuration register:

- PHY power down (GCCFG/PWRDWN)

  This bit switches on/off the PHY full-speed transceiver module. It must be preliminarily set to allow any USB operation.
- A-VBUS sensing enable (GCCFG/VBUSASEN)

  This bit switches on/off the V_{BUS} comparators associated with A-device operations. It must be set when in A-device (USB host) mode and during HNP.
- B-VBUS sensing enable (GCCFG/VBUSASEN)

  This bit switches on/off the V_{BUS} comparators associated with B-device operations. It must be set when in B-device (USB peripheral) mode and during HNP.

  Power reduction techniques are available in the USB suspended state, when the USB session is not yet valid or the device is disconnected.
- Stop PHY clock (STPPCLK bit in OTG_HS_PCGCCTL)
  - When setting the stop PHY clock bit in the clock gating control register, most of the clock domain internal to the OTG high-speed core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the clock input is kept running by the application
  - Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_HS_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_HS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.
- USB system stop
  - When the OTG_HS is in USB suspended state, the application can decide to drastically reduce the overall power consumption by shutting down all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY

clock bit and then configuring the system deep sleep mode in the powercontrol system module (PWR).

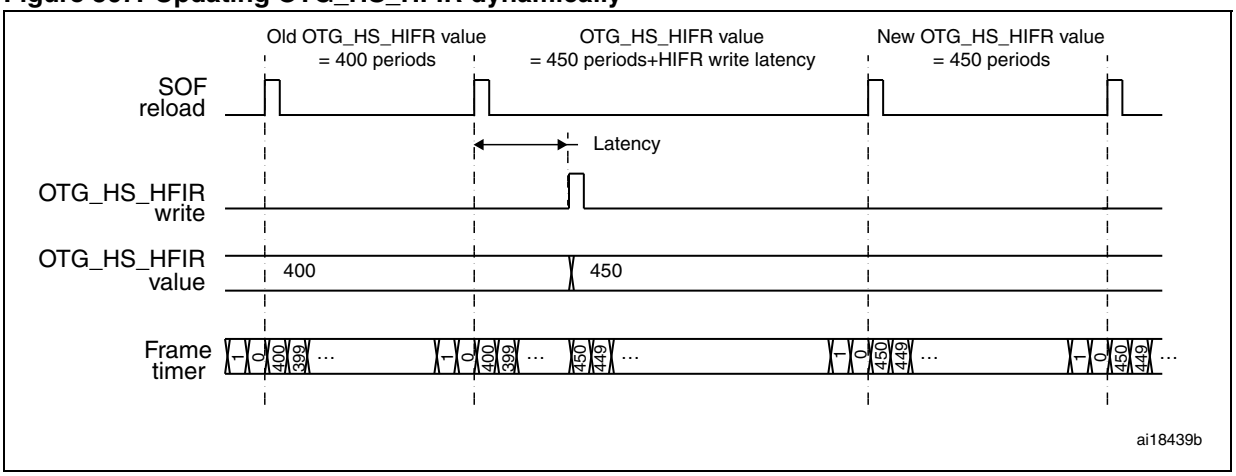
– The OTG_HS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a Device) signaling on the USB.

30.9 Dynamic update of the OTG_HS_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HS_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in *Figure 367*.

Figure 367. Updating OTG_HS_HFIR dynamically



30.10 FIFO RAM allocation

30.10.1 Peripheral mode

Receive FIFO RAM

For Receive FIFO RAM, the application should allocate RAM for SETUP packets: 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoints. These locations are reserved for SETUP packets and are not used by the core to write any other data.

One location must be allocated for Global OUT NAK. Status information are also written to the FIFO along with each received packet. Therefore, a minimum space of (Largest Packet Size / 4) + 1 must be allocated to receive packets. If a high-bandwidth endpoint or multiple isochronous endpoints are enabled, at least two spaces of (Largest Packet Size / 4) + 1 must be allotted to receive back-to-back packets. Typically, two (Largest Packet Size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet.

Along with each endpoints last packet, transfer complete status information are also pushed to the FIFO. Typically, one location for each OUT endpoint is recommended.

Transmit FIFO RAM

For Transmit FIFO RAM, the minimum RAM space required for each IN Endpoint Transmit FIFO is the maximum packet size for this IN endpoint.

Note: More space allocated in the transmit IN Endpoint FIFO results in a better performance on the USB.

30.10.2 Host mode

Receive FIFO RAM

For Receive FIFO RAM allocation, Status information are written to the FIFO along with each received packet. Therefore, a minimum space of (Largest Packet Size / 4) + 1 must be allocated to receive packets. If a high-bandwidth channel or multiple isochronous channels are enabled, at least two spaces of (Largest Packet Size / 4) + 1 must be allocated to receive back-to-back packets. Typically, two (Largest Packet Size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to AHB, the USB can receive the subsequent packet.

Along with each host channels last packet, transfer complete status information are also pushed to the FIFO. As a consequence, one location must be allocated to store this data.

Transmit FIFO RAM

For Transmit FIFO RAM allocation, the minimum amount of RAM required for the host nonperiodic Transmit FIFO is the largest maximum packet size for all supported nonperiodic OUT channels. Typically, a space corresponding to two Largest Packet Size is recommended, so that when the current packet is being transferred to the USB, the AHB can transmit the subsequent packet.

The minimum amount of RAM required for Host periodic Transmit FIFO is the largest maximum packet size for all supported periodic OUT channels. If there is at least one High Bandwidth Isochronous OUT endpoint, then the space must be at least two times the maximum packet size for that channel.

Note: 1 More space allocated in the Transmit nonperiodic FIFO results in better performance on the USB.

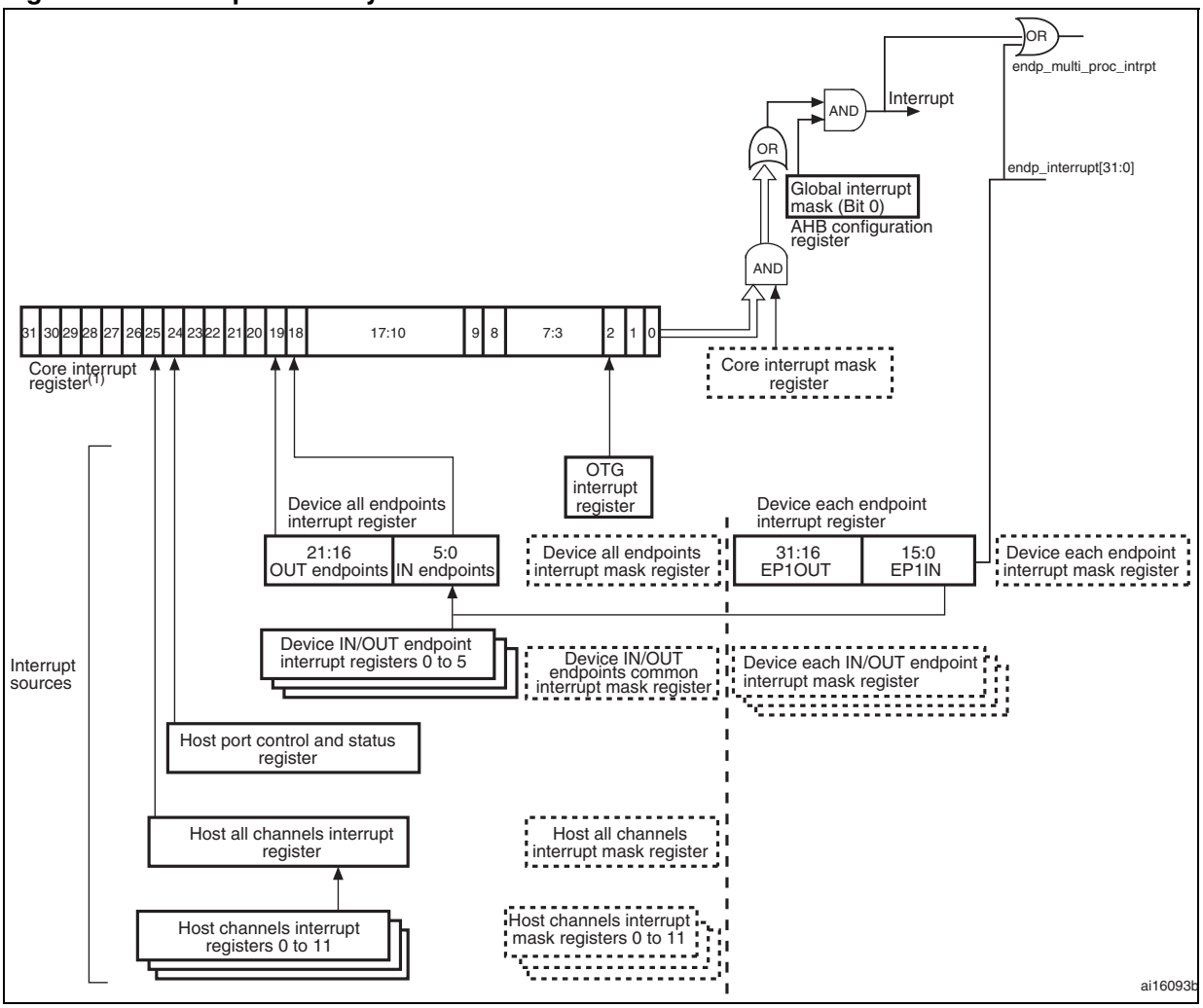
2 When operating in DMA mode, the DMA address register for each host channel (HCDMAn) is stored in the SPRAM (FIFO). One location for each channel must be reserved for this.

30.11 OTG_HS interrupts

When the OTG_HS controller is operating in one mode, either peripheral or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the Core interrupt register (MMIS bit in the OTG_HS_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 368 shows the interrupt hierarchy.

Figure 368. Interrupt hierarchy



1. The core interrupt register bits are shown in *OTG_HS core interrupt register (OTG_HS_GINTSTS) on page 1101*.

30.12 OTG_HS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_HS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. CSRs are classified as follows:

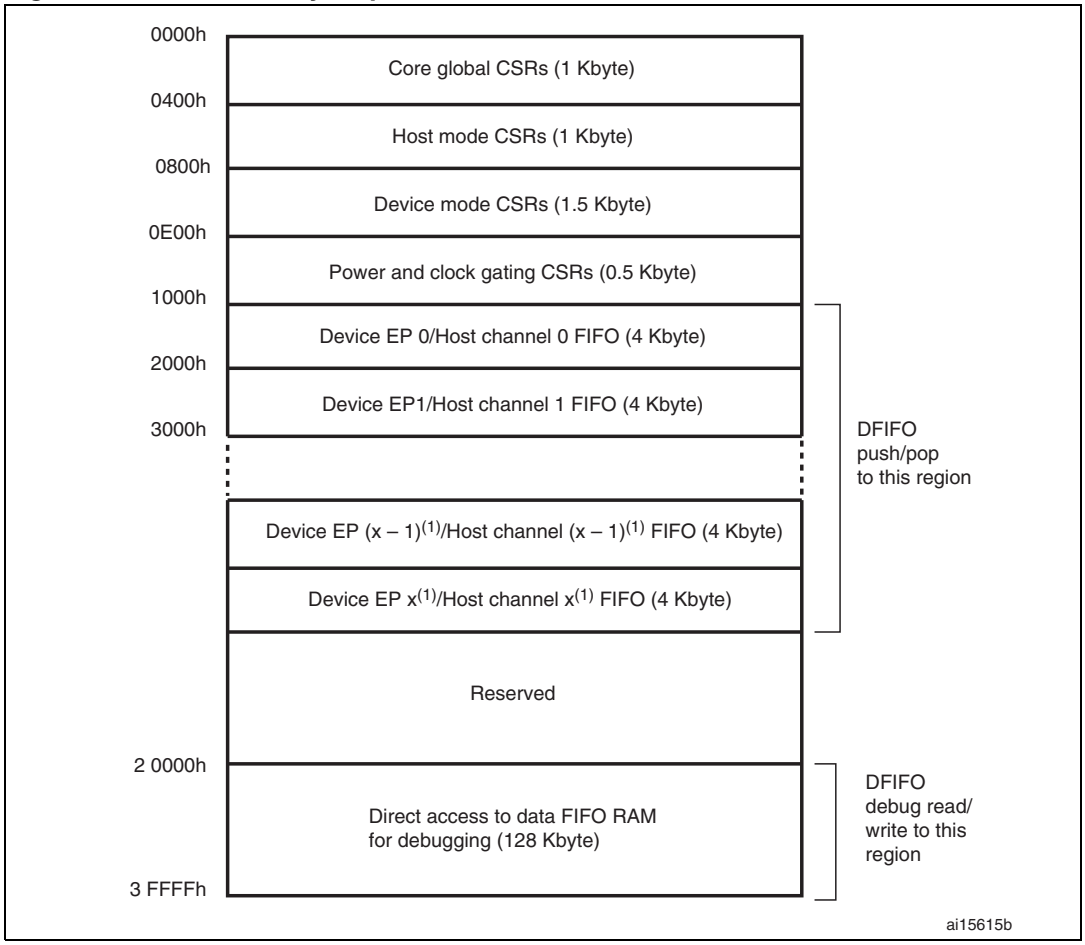
- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the Core global, Power and clock-gating, Data FIFO access, and host port control and status registers can be accessed in both host and peripheral modes. When the OTG_HS controller is operating in one mode, either peripheral or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the Core interrupt register (MMIS bit in the OTG_HS_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

30.12.1 CSR memory map

The host and peripheral mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Figure 369. CSR memory map



1. x = 5 in peripheral mode and x = 11 in host mode.

Global CSR map

These registers are available in both host and peripheral modes.

Table 156. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_HS_GOTGCTL	0x000	*OTG_HS control and status register (OTG_HS_GOTGCTL) on page 1091*
OTG_HS_GOTGINT	0x004	*OTG_HS interrupt register (OTG_HS_GOTGINT) on page 1093*
OTG_HS_GAHBCFG	0x008	*OTG_HS AHB configuration register (OTG_HS_GAHBCFG) on page 1094*
OTG_HS_GUSBCFG	0x00C	*OTG_HS USB configuration register (OTG_HS_GUSBCFG) on page 1095*
OTG_HS_GRSTCTL	0x010	*OTG_HS reset register (OTG_HS_GRSTCTL) on page 1098*

Table 156. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_HS_GINTSTS	0x014	*OTG_HS core interrupt register (OTG_HS_GINTSTS) on page 1101*
OTG_HS_GINTMSK	0x018	*OTG_HS interrupt mask register (OTG_HS_GINTMSK) on page 1105*
OTG_HS_GRXSTSR	0x01C	*OTG_HS Receive status debug read/OTG status read and pop registers (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP) on page 1108*
OTG_HS_GRXSTSP	0x020	
OTG_HS_GRXFSIZ	0x024	*OTG_HS Receive FIFO size register (OTG_HS_GRXFSIZ) on page 1109*
OTG_HS_GNPTXFSIZ/ OTG_HS_TX0FSIZ	0x028	*OTG_HS nonperiodic transmit FIFO size/Endpoint 0 transmit FIFO size register (OTG_HS_GNPTXFSIZ/OTG_HS_TX0FSIZ) on page 1110*
OTG_HS_GNPTXSTS	0x02C	*OTG_HS nonperiodic transmit FIFO/queue status register (OTG_HS_GNPTXSTS) on page 1110*
OTG_HS_GCCFG	0x038	*OTG_HS general core configuration register (OTG_HS_GCCFG) on page 1113*
OTG_HS_CID	0x03C	*OTG_HS core ID register (OTG_HS_CID) on page 1114*
OTG_HS_HPTXFSIZ	0x100	*OTG_HS Host periodic transmit FIFO size register (OTG_HS_HPTXFSIZ) on page 1114*
OTG_HS_DIEPTXFx	0x104 0x124 ... 0x13C	*OTG_HS device IN endpoint transmit FIFO size register (OTG_HS_DIEPTXFx) (x = 1..7, where x is the FIFO_number) on page 1114*

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 157. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HS_HCFG	0x400	*OTG_HS host configuration register (OTG_HS_HCFG) on page 1115*
OTG_HS_HFIR	0x404	*OTG_HS Host frame interval register (OTG_HS_HFIR) on page 1116*
OTG_HS_HFNUM	0x408	*OTG_HS host frame number/frame time remaining register (OTG_HS_HFNUM) on page 1116*
OTG_HS_HPTXSTS	0x410	*OTG_HS_Host periodic transmit FIFO/queue status register (OTG_HS_HPTXSTS) on page 1117*
OTG_HS_HAINT	0x414	*OTG_HS Host all channels interrupt register (OTG_HS_HAINT) on page 1118*
OTG_HS_HAINTMSK	0x418	*OTG_HS host all channels interrupt mask register (OTG_HS_HAINTMSK) on page 1118*

Table 157. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HS_HPRT	0x440	*OTG_HS host port control and status register (OTG_HS_HPRT) on page 1119*
OTG_HS_HCCHARx	0x500 0x520 ... 0x6E0	*OTG_HS host channel-x characteristics register (OTG_HS_HCCHARx) (x = 0..11, where x = Channel_number) on page 1121*
OTG_HS_HCSPLTx	0x504	*OTG_HS host channel-x split control register (OTG_HS_HCSPLTx) (x = 0..11, where x = Channel_number) on page 1123*
OTG_HS_HCINTx	0x508	*OTG_HS host channel-x interrupt register (OTG_HS_HCINTx) (x = 0..11, where x = Channel_number) on page 1124*
OTG_HS_HCINTMSKx	0x50C	*OTG_HS host channel-x interrupt mask register (OTG_HS_HCINTMSKx) (x = 0..11, where x = Channel_number) on page 1125*
OTG_HS_HCTSIZx	0x510	*OTG_HS host channel-x transfer size register (OTG_HS_HCTSIZx) (x = 0..11, where x = Channel_number) on page 1126*
OTG_HS_HCDMAx	0x514	*OTG_HS host channel-x DMA address register (OTG_HS_HCDMAx) (x = 0..11, where x = Channel_number) on page 1127*

Device-mode CSR map

These registers must be programmed every time the core changes to peripheral mode.

Table 158. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_HS_DCFG	0x800	*OTG_HS device configuration register (OTG_HS_DCFG) on page 1127*
OTG_HS_DCTL	0x804	*OTG_HS device control register (OTG_HS_DCTL) on page 1129*
OTG_HS_DSTS	0x808	*OTG_HS device status register (OTG_HS_DSTS) on page 1131*
OTG_HS_DIEPMSK	0x810	*OTG_HS device IN endpoint common interrupt mask register (OTG_HS_DIEPMSK) on page 1132*
OTG_HS_DOEPMSK	0x814	*OTG_HS device OUT endpoint common interrupt mask register (OTG_HS_DOEPMSK) on page 1133*
OTG_HS_DAINT	0x818	*OTG_HS device all endpoints interrupt register (OTG_HS_DAINT) on page 1134*
OTG_HS_DAINTMSK	0x81C	*OTG_HS all endpoints interrupt mask register (OTG_HS_DAINTMSK) on page 1134*
OTG_HS_DVBUSDIS	0x828	*OTG_HS device V_{BUS} discharge time register (OTG_HS_DVBUSDIS) on page 1135*
OTG_HS_DVBUSPULSE	0x82C	*OTG_HS device V_{BUS} pulsing time register (OTG_HS_DVBUSPULSE) on page 1135*

Table 158. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_HS_DIEPEMPMSK	0x834	*OTG_HS device IN endpoint FIFO empty interrupt mask register: (OTG_HS_DIEPEMPMSK) on page 1137*
OTG_HS_EACHHINT	0x838	*OTG_HS device each endpoint interrupt register (OTG_HS_DEACHINT) on page 1137*
OTG_HS_EACHHINTMSK	0x83C	*OTG_HS device each endpoint interrupt register mask (OTG_HS_DEACHINTMSK) on page 1138*
OTG_HS_DIEPEACHMSK1	0x840	*OTG_HS device each in endpoint-1 interrupt register (OTG_HS_DIEPEACHMSK1) on page 1138*
OTG_HS_DOEPEACHMSK1	0x880	*OTG_HS device each OUT endpoint-1 interrupt register (OTG_HS_DOEPEACHMSK1) on page 1139*
OTG_HS_DIEPCTLx	0x920 0x940 ... 0xAE0	*OTG device endpoint-x control register (OTG_HS_DIEPCTLx) (x = 0..7, where x = Endpoint_number) on page 1140*
OTG_HS_DIEPINTx	0x908	*OTG_HS device endpoint-x interrupt register (OTG_HS_DIEPINTx) (x = 0..7, where x = Endpoint_number) on page 1147*
OTG_HS_DIEPTSIZ0	0x910	*OTG_HS device IN endpoint 0 transfer size register (OTG_HS_DIEPTSIZ0) on page 1150*
OTG_HS_DIEPDMAx	0x914	*OTG_HS device endpoint-x DMA address register (OTG_HS_DIEPDMAx / OTG_HS_DOEPDMAx) (x = 1..5, where x = Endpoint_number) on page 1154*
OTG_HS_DTXFSTSx	0x918	*OTG_HS device IN endpoint transmit FIFO status register (OTG_HS_DTXFSTSx) (x = 0..5, where x = Endpoint_number) on page 1153*
OTG_HS_DIEPTSIZx	0x930 0x950 ... 0xAF0	*OTG_HS device endpoint-x transfer size register (OTG_HS_DOEPTSIZx) (x = 1..5, where x = Endpoint_number) on page 1153*
OTG_HS_DOEPCTL0	0xB00	*OTG_HS device control OUT endpoint 0 control register (OTG_HS_DOEPCTL0) on page 1143*
OTG_HS_DOEPCTLx	0xB20 0xB40 ... 0xCC0 0xCE0 0xCFD	*OTG device endpoint-x control register (OTG_HS_DIEPCTLx) (x = 0..7, where x = Endpoint_number) on page 1140*

Table 158. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_HS_DOEPINTx	0xB08	*OTG_HS device endpoint-x interrupt register (OTG_HS_DIEPINTx) (x = 0..7, where x = Endpoint_number) on page 1147*
OTG_HS_DOEPTSIZx	0xB10	*OTG_HS device endpoint-x transfer size register (OTG_HS_DOEPTSIZx) (x = 1..5, where x = Endpoint_number) on page 1153*

Data FIFO (DFIFO) access register map

These registers, available in both host and peripheral modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 159. Data FIFO (DFIFO) access register map

FIFO access register section	Address range	Access
Device IN Endpoint 0/Host OUT Channel 0: DFIFO Write Access Device OUT Endpoint 0/Host IN Channel 0: DFIFO Read Access	0x1000–0x1FFC	w r
Device IN Endpoint 1/Host OUT Channel 1: DFIFO Write Access Device OUT Endpoint 1/Host IN Channel 1: DFIFO Read Access	0x2000–0x2FFC	w r
...	...	...
Device IN Endpoint x(1)/Host OUT Channel x(1): DFIFO Write Access Device OUT Endpoint x(1)/Host IN Channel x(1): DFIFO Read Access	0xX000h–0xXFFCh	w r

1. Where x is 5 in peripheral mode and 11 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and peripheral modes.

Table 160. Power and clock gating control and status registers

Register name	Acronym	Offset address: 0xE00–0xFFF
Power and clock gating control register	PCGCR	0xE00-0xE04
Reserved		0xE05–0xFFF

30.12.2 OTG_HS global registers

These registers are available in both host and peripheral modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

OTG_HS control and status register (OTG_HS_GOTGCTL)

Address offset: 0x000

Reset value: 0x0000 0800

The OTG control and status register controls the behavior and reflects the status of the OTG function of the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
												r	r	r	r					rw	rw	rw	r							rw	r

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **BSVLD:** B-session valid

Indicates the peripheral mode transceiver status.
0: B-session is not valid.
1: B-session is valid.
In OTG mode, you can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in peripheral mode.

Bit 18 **ASVLD:** A-session valid

Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid

Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time

Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)

Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status

Indicates the connector ID status on a connect event.
0: The OTG_HS controller is in A-device mode
1: The OTG_HS controller is in B-device mode

Note: Accessible in both peripheral and host modes.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **DHNPEN:** Device HNP enabled

The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.
0: HNP is not enabled in the application
1: HNP is enabled in the application

Note: Only accessible in peripheral mode.

Bit 10 **HSHNPEN:** Host set HNP enable

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

0: Host Set HNP is not enabled

1: Host Set HNP is enabled

Note: Only accessible in host mode.

Bit 9 **HNPRQ:** HNP request

The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG interrupt register (HNSSCHG bit in OTG_HS_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

0: No HNP request

1: HNP request

Note: Only accessible in peripheral mode.

Bit 8 **HNGSCS:** Host negotiation success

The core sets this bit when host negotiation is successful. The core clears this bit when the HNP Request (HNPRQ) bit in this register is set.

0: Host negotiation failure

1: Host negotiation success

Note: Only accessible in peripheral mode.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SRQ:** Session request

The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG Interrupt register (HNSSCHG bit in OTG_HS_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

If you use the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until V_{BUS} discharges to 0.2 V, after the B-Session Valid bit in this register (BSVLD bit in OTG_HS_GOTGCTL) is cleared. This discharge time varies between different PHYs and can be obtained from the PHY vendor.

0: No session request

1: Session request

Note: Only accessible in peripheral mode.

Bit 0 **SRQSCS:** Session request success

The core sets this bit when a session request initiation is successful.

0: Session request failure

1: Session request success

Note: Only accessible in peripheral mode.

OTG_HS interrupt register (OTG_HS_GOTGINT)

Address offset: 0x04

Reset value: 0x0000 0000

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
												rc_w1	rc_w1	rc_w1								rc_w1	rc_w1						rc_w1		

Bits 31:20 Reserved, must be kept at reset value..

Bit 19 **DBCDNE:** Debounce done

The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the Core USB Configuration register (HNPCAP bit or SRPCAP bit in OTG_HS_GUSBCFG, respectively).

Note: Only accessible in host mode.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both peripheral and host modes.

Bit 17 **HNGDET:** Host negotiation detected

The core sets this bit when it detects a host negotiation request on the USB.

Note: Accessible in both peripheral and host modes.

Bits 16:10 Reserved, must be kept at reset value..

Bit 9 **HNSSCHG:** Host negotiation success status change

The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG Control and Status register (HNGSCS in OTG_HS_GOTGCTL) to check for success or failure.

Note: Accessible in both peripheral and host modes.

Bits 7:3 Reserved, must be kept at reset value..

Bit 8 **SRSSCHG:** Session request success status change

The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG Control and status register (SRQSCS bit in OTG_HS_GOTGCTL) to check for success or failure.

Note: Accessible in both peripheral and host modes.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-device session when $V_{BUS} < 0.8$ V.

Bits 1:0 Reserved, must be kept at reset value..

OTG_HS AHB configuration register (OTG_HS_GAHBCFG)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																							PTXFELVL	TXFELVL	Reserved	DMAEN	HBSTLEN				GINT
																							rw	rw		rw					rw

Bits 31:20 Reserved, must be kept at reset value..

Bit 8 **PTXFELVL:** Periodic TxFIFO empty level

Indicates when the periodic TxFIFO empty interrupt bit in the Core interrupt register (PTXFE bit in OTG_HS_GINTSTS) is triggered.

0: PTXFE (in OTG_HS_GINTSTS) interrupt indicates that the Periodic TxFIFO is half empty

1: PTXFE (in OTG_HS_GINTSTS) interrupt indicates that the Periodic TxFIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** TxFIFO empty level

In peripheral mode, this bit indicates when the IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_HS_DIEPINTx.) is triggered.

0: TXFE (in OTG_HS_DIEPINTx) interrupt indicates that the IN Endpoint TxFIFO is half empty

1: TXFE (in OTG_HS_DIEPINTx) interrupt indicates that the IN Endpoint TxFIFO is completely empty

Note: Only accessible in peripheral mode.

Bit 6 Reserved, must be kept at reset value.

Bits5 **DMAEN:** DMA enable

0: The core operates in slave mode

1: The core operates in DMA mode

Bits 4:1 **HBSTLEN:** Burst length/type

0000 Single

0001 INCR

0011 INCR4

0101 INCR8

0111 INCR16

Others: Reserved

Bit 0 **GINT:** Global interrupt mask

This bit is used to mask or unmask the interrupt line assertion to the application. Irrespective of this bit setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application

Note: Accessible in both peripheral and host modes.

OTG_HS USB configuration register (OTG_HS_GUSBCFG)

Address offset: 0x00C

Reset value: 0x0000 0A00

This register can be used to configure the core after power-on or a changing to host mode or peripheral mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

31	30	29	28 27 26	25	24	23	22	21	20	19	18	17	16	15	14	13 12 11 10	9	8	7 6 5 4 3	2 1 0
CTXPKT	FDMOD	FHMOD	Reserved	ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT	HNPCAP	SRPCAP	Reserved	TOCAL
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	r/rw	r/rw		rw

Bit 31 **CTXPKT:** Corrupt Tx packet

This bit is for debug purposes only. Never set this bit to 1.

Note: Accessible in both peripheral and host modes.

Bit 30 **FDMOD:** Forced peripheral mode

Writing a 1 to this bit forces the core to peripheral mode irrespective of the OTG_HS_ID input pin.

0: Normal mode

1: Forced peripheral mode

After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both peripheral and host modes.

Bit 29 **FHMOD:** Forced host mode

Writing a 1 to this bit forces the core to host mode irrespective of the OTG_HS_ID input pin.

0: Normal mode

1: Forced host mode

After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both peripheral and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **ULPIIPD:** ULPI interface protect disable

This bit controls the circuitry built in the PHY to protect the ULPI interface when the link tri-states stp and data. Any pull-up or pull-down resistors employed by this feature can be disabled. Please refer to the ULPI specification for more details.

0: Enables the interface protection circuit

1: Disables the interface protection circuit

Bit 24 **PTCI:** Indicator pass through

This bit controls whether the complement output is qualified with the internal V_{BUS} valid comparator before being used in the V_{BUS} state in the RX CMD. Please refer to the ULPI specification for more details.

0: Complement Output signal is qualified with the Internal V_{BUS} valid comparator

1: Complement Output signal is not qualified with the Internal V_{BUS} valid comparator

Bit 23 **PCCI:** Indicator complement

This bit controls the PHY to invert the ExternalVbusIndicator input signal, and generate the complement output. Please refer to the ULPI specification for more details.

0: PHY does not invert the ExternalVbusIndicator signal

1: PHY inverts ExternalVbusIndicator signal

Bit 22 **TSDPS**: TermSel DLine pulsing selection

This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).

0: Data line pulsing using utmi_txvalid (default)

1: Data line pulsing using utmi_termsel

Bit 21 **ULPIEVBUSI:** ULPI external V_{BUS} indicator

This bit indicates to the ULPI PHY to use an external V_{BUS} overcurrent indicator.

0: PHY uses an internal V_{BUS} valid comparator

1: PHY uses an external V_{BUS} valid comparator

Bit 20 **ULPIEVBUSD:** ULPI External V_{BUS} Drive

This bit selects between internal or external supply to drive 5 V on V_{BUS}, in the ULPI PHY.

0: PHY drives V_{BUS} using internal charge pump (default)

1: PHY drives V_{BUS} using external supply.

Bit 19 **ULPICSM:** ULPI Clock SuspendM

This bit sets the ClockSuspendM bit in the interface control register on the ULPI PHY. This bit applies only in the serial and carkit modes.

0: PHY powers down the internal clock during suspend

1: PHY does not power down the internal clock

Bit 18 **ULPIAR:** ULPI Auto-resume

This bit sets the AutoResume bit in the interface control register on the ULPI PHY.

0: PHY does not use AutoResume feature

1: PHY uses AutoResume feature

Bit 17 **ULPIFSLS:** ULPI FS/LS select

The application uses this bit to select the FS/LS serial interface for the ULPI PHY. This bit is valid only when the FS serial transceiver is selected on the ULPI PHY.

0: ULPI interface

1: ULPI FS/LS serial interface

Bit 16 Reserved, must be kept at reset value.

Bit 15 **PHYLPCS:** PHY Low-power clock select

This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.

0: 480 MHz internal PLL clock

1: 48 MHz external clock

In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 **Reserved, must be kept at reset value.**

Bits 13:10 **TRDT:** USB turnaround time

Sets the turnaround time in PHY clocks.

The formula below gives the value of TRDT:

TRDT = 4 × AHB clock frequency+ 1 PHY clock frequency.

For example:

If AHB clock frequency = 72 MHz (PHY Clock frequency = 48 MHz), the TRDT must be set to 9.

If AHB clock frequency = 48 Mhz (PHY Clock frequency = 48 MHz), the TRDT must be set to 5.

Note: Only accessible in peripheral mode.

Bit 9 **HNPCAP:** HNP-capable

The application uses this bit to control the OTG_HS controller's HNP capabilities.

0: HNP capability is not enabled

1: HNP capability is enabled

Note: Accessible in both peripheral and host modes.

Bit 8 **SRPCAP:** SRP-capable

The application uses this bit to control the OTG_HS controller's SRP capabilities. If the core operates as a nonSRP-capable B-device, it cannot request the connected A-device (host) to activate V_{BUS} and start a session.

0: SRP capability is not enabled

1: SRP capability is enabled

Note: Accessible in both peripheral and host modes.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

OTG_HS reset register (OTG_HS_GRSTCTL)

Address offset: 0x010

Reset value: 0x2000 0000

The application uses this register to reset various hardware features inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	DMAREQ	Reserved																			TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r	r																				rw					rs	rs		rs	rs	rs

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both peripheral and host modes.

Bit 30 **DMAREQ:** DMA request signal

This bit indicates that the DMA request is in progress. Used for debug.

Bits 29:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM:** TxFIFO number

This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit.

- 00000:
  - Nonperiodic TxFIFO flush in host mode
  - Tx FIFO 0 flush in peripheral mode
- 00001:
  - Periodic TxFIFO flush in host mode
  - TXFIFO 1 flush in peripheral mode
- 00010: TXFIFO 2 flush in peripheral mode

...

- 00101: TXFIFO 15 flush in peripheral mode
- 10000: Flush all the transmit FIFOs in peripheral or host mode.

Note: Accessible in both peripheral and host modes.

Bit 5 **TXFFLSH:** TxFIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers:

- Read: the NAK effective interrupt ensures the core is not reading from the FIFO
- Write: the AHBIDL bit in OTG_HS_GRSTCTL ensures that the core is not writing anything to the FIFO

Note: Accessible in both peripheral and host modes.

Bit 4 **RXFFLSH:** RxFIFO flush

The application can flush the entire RxFIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the RxFIFO nor writing to the RxFIFO.

The application must wait until the bit is cleared before performing any other operation. This bit requires 8 clocks (slowest of PHY or AHB clock) to be cleared.

Note: Accessible in both peripheral and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

Note: Only accessible in host mode.

Bit 1 **HSRST:** HCLK soft reset

The application uses this bit to flush the control logic in the AHB Clock domain. Only AHB Clock Domain pipelines are reset.

FIFOs are not flushed with this bit.

All state machines in the AHB clock domain are reset to the Idle state after terminating the transactions on the AHB, following the protocol.

CSR control bits used by the AHB clock domain state machines are cleared.

To clear this interrupt, status mask bits that control the interrupt status and are generated by the AHB clock domain state machine are cleared.

Because interrupt status bits are not cleared, the application can get the status of any core events that occurred after it set this bit.

This is a self-clearing bit that the core clears after all necessary logic is reset in the core. This can take several clocks, depending on the core's current state.

Note: Accessible in both peripheral and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PCLK domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

– RSTPDMODL bit in OTG_HS_PCGCCTL
– GAYEHCLK bit in OTG_HS_PCGCCTL
– PWRCLMP bit in OTG_HS_PCGCCTL
– STPPCLK bit in OTG_HS_PCGCCTL
– FSLSPCS bit in OTG_HS_HCFG
– DSPD bit in OTG_HS_DCFG

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when you dynamically change the PHY selection bits in the above listed USB configuration registers. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both peripheral and host modes.

OTG_HS core interrupt register (OTG_HS_GINTSTS)

Address offset: 0x014

Reset value: 0x0400 0020

This register interrupts the application for system-level events in the current mode (peripheral mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in peripheral mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_HS_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r			rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wakeup detected interrupt

In peripheral mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

Note: Accessible in both peripheral and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In peripheral mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-device device. Accessible in both peripheral and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both peripheral and host modes.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **PTXFE:** Periodic TxFIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic TxFIFO empty level bit in the Core AHB configuration register (PTXFELVL bit in OTG_HS_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the host all channels interrupt (OTG_HS_HAINT) register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding host channel-x interrupt (OTG_HS_HCINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HS_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_HS controller ports in host mode. The application must read the host port control and status (OTG_HS_HPRT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the host port control and status register to clear this bit.

Note: Only accessible in host mode.

Bits 23 Reserved, must be kept at reset value.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints
- Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_FS_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

Note: Only accessible in host mode.

INCOMPISOOUT: Incomplete isochronous OUT transfer

In peripheral mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in peripheral mode.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in peripheral mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in peripheral mode). The application must read the device all endpoints interrupt (OTG_HS_DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding device OUT Endpoint-x Interrupt (OTG_HS_DOEPINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_HS_DOEPINTx register to clear this bit.

Note: Only accessible in peripheral mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in peripheral mode). The application must read the device All Endpoints Interrupt (OTG_HS_DAINT) register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding device IN Endpoint-x interrupt (OTG_HS_DIEPINTx) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_HS_DIEPINTx register to clear this bit.

Note: Only accessible in peripheral mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the device configuration register (PFIVL bit in OTG_HS_DCFG) has been reached in the current frame.

Note: Only accessible in peripheral mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in peripheral mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the device Status (OTG_HS_DSTS) register to obtain the enumerated speed.

Note: Only accessible in peripheral mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in peripheral mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the data lines for a period of 3 ms.

Note: Only accessible in peripheral mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in peripheral mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the Device control register (SGONAK bit in OTG_HS_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the Device control register (CGONAK bit in OTG_HS_DCTL).

Note: Only accessible in peripheral mode.

Bit 6 **GINAKEFF:** Global IN nonperiodic NAK effective

Indicates that the Set global nonperiodic IN NAK bit in the Device control register (SGINAK bit in OTG_HS_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global nonperiodic IN NAK bit in the Device control register (CGINAK bit in OTG_HS_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in peripheral mode.

Bit 5 **NPTXFE:** Nonperiodic TxFIFO empty

This interrupt is asserted when the nonperiodic TxFIFO is either half or completely empty, and there is space in at least one entry to be written to the nonperiodic transmit request queue. The half or completely empty status is determined by the nonperiodic TxFIFO empty level bit in the OTG_HS_GAHBCFG register (TXFELVL bit in OTG_HS_GAHBCFG).

Note: Only accessible in host mode.

Bit 4 **RXFLVL:** RxFIFO nonempty

Indicates that there is at least one packet pending to be read from the RxFIFO.

Note: Accessible in both host and peripheral modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In peripheral mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: Accessible in both host and peripheral modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG Interrupt Status (OTG_HS_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HS_GOTGINT register to clear this bit.

Note: Accessible in both host and peripheral modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

A host mode register, when the core is operating in peripheral mode

A peripheral mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and peripheral modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Peripheral mode

1: Host mode

Note: Accessible in both host and peripheral modes.

OTG_HS interrupt mask register (OTG_HS_GINTMSK)

Address offset: 0x018

Reset value: 0x0000 0000

This register works with the Core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the Core Interrupt (OTG_HS_GINTSTS) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	r		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wakeup detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and peripheral modes.

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and peripheral modes.

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and peripheral modes.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and peripheral modes.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **PTXFEM:** Periodic TxFIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **FSUSPM:** Data fetch suspended mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.
IISOOXFRM: Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 17 **EPMISM:** Endpoint mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bits 9:8 Reserved, must be kept at reset value..

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 6 **GINAKEFFM:** Global nonperiodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in peripheral mode.

Bit 5 **NPTXFEM:** Nonperiodic TxFIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both peripheral and host modes.

Bit 4 **RXFLVLM:** Receive FIFO nonempty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both peripheral and host modes.

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both peripheral and host modes.

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both peripheral and host modes.

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both peripheral and host modes.

Bit 0 Reserved, must be kept at reset value.

OTG_HS Receive status debug read/OTG status read and pop registers (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP)

Address offset for Read: 0x01C

Address offset for Pop: 0x020

Reset value: 0x0000 0000

A read to the Receive status debug read register returns the contents of the top of the Receive FIFO. A read to the Receive status read and pop register additionally pops the top data entry out of the RxFIFO.

The receive status contents must be interpreted differently in host and peripheral modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the Receive Status FIFO when the Receive FIFO nonempty bit of the Core interrupt register (RXFLVL bit in OTG_HS_GINTSTS) is asserted.

Host mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PKTSTS				DPID		BCNT												CHNUM			
											r				r		r												r			

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID:** Data PID
Indicates the Data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM:** Channel number
Indicates the channel number to which the current received packet belongs.

Peripheral mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRMNUM				PKTSTS				DPID		BCNT													EPNUM		
							r				r				r		r													r		

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID:** Data PID

Indicates the Data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

OTG_HS Receive FIFO size register (OTG_HS_GRXFSIZ)

Address offset: 0x024

Reset value: 0x0000 0200

The application can program the RAM size that must be allocated to the RxFIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																RXFD															
																r/rw															

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD:** RxFIFO depth

This value is in terms of 32-bit words.
Minimum value is 16
Maximum value is 1024
The power-on reset value of this register is specified as the largest Rx data FIFO depth.

OTG_HS nonperiodic transmit FIFO size/Endpoint 0 transmit FIFO size register (OTG_HS_GNPTXFSIZ/OTG_HS_TX0FSIZ)

Address offset: 0x028

Reset value: 0x0000 0200

Host mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD																NPTXFSA															
r/rw																r/rw															

Bits 31:16 **NPTXFD:** Nonperiodic TxFIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Maximum value is 1024

Bits 15:0 **NPTXFSA:** Nonperiodic transmit RAM start address
This field contains the memory start address for nonperiodic transmit FIFO RAM.

Peripheral mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX0FD																TX0FSA															
r/rw																r/rw															

Bits 31:16 **T0XFD:** Endpoint 0 TxFIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Maximum value is 256

Bits 15:0 **TX0FSA:** Endpoint 0 transmit RAM start address
This field contains the memory start address for Endpoint 0 transmit FIFO RAM.

OTG_HS nonperiodic transmit FIFO/queue status register (OTG_HS_GNPTXSTS)

Address offset: 0x02C

Reset value: 0x0008 0200

Note: In peripheral mode, this register is not valid.

This read-only register contains the free space information for the nonperiodic TxFIFO and the nonperiodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	r							r								r															

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP:** Top of the nonperiodic transmit request queue

Entry in the nonperiodic Tx request queue that is currently being processed by the MAC.
Bits [30:27]: Channel/endpoint number
Bits [26:25]:

- 00: IN/OUT token
- 01: Zero-length transmit packet (device IN/host OUT)
- 10: PING/CSPLIT token
- 11: Channel halt command

Bit [24]: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV:** Nonperiodic transmit request queue space available

Indicates the amount of free space available in the nonperiodic transmit request queue. This queue holds both IN and OUT requests in host mode. Peripheral mode has only IN requests.
00: Nonperiodic transmit request queue is full
01: dx1 location available
10: dx2 locations available
bxn: dx*n* locations available (0 ≤ *n* ≤ dx8)
Others: Reserved

Bits 15:0 **NPTXFSAV:** Nonperiodic TxFIFO space available

Indicates the amount of free space available in the nonperiodic TxFIFO.
Values are in terms of 32-bit words.
00: Nonperiodic TxFIFO is full
01: dx1 word available
10: dx2 words available
0xn: dx*n* words available (where 0 ≤ *n* ≤ dx1024)
Others: Reserved

OTG_HS I²C access register (OTG_HS_GI2CCTL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSYDNE	RW	Reserved	I2CDATSE0	I2CDEV ADR		Reserved	ACK	I2CEN	ADDR							REGADDR								RWDATA							
rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BSYDNE**: I2C Busy/Done

The application sets this bit to 1 to start a request on the I^2C interface. When the transfer is complete, the core deasserts this bit to 0. As long as the bit is set indicating that the I^2C interface is busy, the application cannot start another request on the interface.

Bit 30 **RW**: Read/Write Indicator

This bit indicates whether a read or write register transfer must be performed on the interface.
0: Write
1: Read

Note: Read/write bursting is not supported for registers.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **I2CDATSE0**: I^2C DatSe0 USB mode

This bit is used to select the full-speed interface USB mode.
0: VP_VM USB mode
1: DAT_SE0 USB mode

Bits 27:26 **I2CDEVADR**: I^2C Device Address

This bit selects the address of the I^2C slave on the USB 1.1 full-speed serial transceiver corresponding to the one used by the core for OTG signalling.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ACK**: I^2C ACK

This bit indicates whether an ACK response was received from the I^2C slave. It is valid when BSYDNE is cleared by the core, after the application has initiated an I^2C access.
0: NAK
1: ACK

Bit 23 **I2CEN**: I^2C Enable

This bit enables the I^2C master to initiate transactions on the I^2C interface.

Bits 22:16 **ADDR**: I^2C Address

This is the 7-bit I^2C device address used by the application to access any external I^2C slave, including the I^2C slave on a USB 1.1 OTG full-speed serial transceiver.

Bits 15:8 **REGADDR**: I^2C Register Address

These bits allow to program the address of the register to be read from or written to.

Bits 7:0 **RWDATA**: I^2C Read/Write Data

After a register read operation, these bits hold the read data for the application.
During a write operation, the application can use this register to program the data to be written to a register.

OTG_HS general core configuration register (OTG_HS_GCCFG)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	I2CPADEN	.PWRDWN	Reserved															
										rw	rw	rw	rw	rw	rw																

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **NOVBUSSENS**: V_{BUS} sensing disable option

When this bit is set, V_{BUS} is considered internally to be always at V_{BUS} valid level (5 V). This option removes the need for a dedicated V_{BUS} pad, and leave this pad free to be used for other purposes such as a shared functionality. V_{BUS} connection can be remapped on another general purpose input pad and monitored by software.

This option is only suitable for host-only or device-only applications.

0: V_{BUS} sensing available by hardware

1: V_{BUS} sensing not available by hardware.

Bit 20 **SOFOUTEN:** SOF output enable

0: SOF pulse not available on PAD

1: SOF pulse available on PAD

Bit 19 **VBUSBSEN:** Enable the V_{BUS} sensing “B” device

0: V_{BUS} sensing “B” disabled

1: V_{BUS} sensing “B” enabled

Bit 18 **VBUSASEN:** Enable the V_{BUS} sensing “A” device

0: V_{BUS} sensing “A” disabled

1: V_{BUS} sensing “A” enabled

Bit 17 **I2CPADEN**: Enable I^2C bus connection for the external I^2C PHY interface.

0: I^2C bus disabled

1: I^2C bus enabled

Bit 16 **PWRDWN:** Power down

Used to activate the transceiver in transmission/reception

0: Power down active

1: Power down deactivated (“Transceiver active”)

Bits 15:0 Reserved, must be kept at reset value..

OTG_HS core ID register (OTG_HS_CID)

Address offset: 0x03C

Reset value:0x0000 1200

This is a read only register containing the Product ID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID:** Product ID field
Application-programmable ID field.

OTG_HS Host periodic transmit FIFO size register (OTG_HS_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0200 0600

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFD																PTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

Bits 31:16 **PTXFD:** Host periodic TxFIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Maximum value is 512

Bits 15:0 **PTXSA:** Host periodic TxFIFO start address
The power-on reset value of this register is the sum of the largest Rx data FIFO depth and largest nonperiodic Tx data FIFO depth.

OTG_HS device IN endpoint transmit FIFO size register (OTG_HS_DIEPTXFx) (x = 1..7, where x is the FIFO_number)

Address offset: 0x104 + (FIFO_number – 1) × 0x04

Reset value: 0x02000400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFD																INEPTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

Bits 31:16 **INEPTXFD:** IN endpoint TxFIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Maximum value is 512
The power-on reset value of this register is specified as the largest IN endpoint FIFO number depth.

Bits 15:0 **INEPTXSA:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx.

30.12.3 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

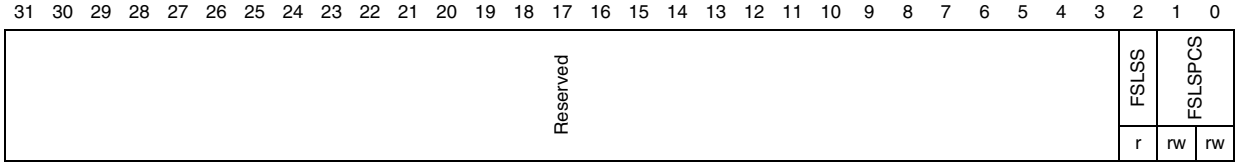
Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in peripheral mode, as the results are undefined. Host mode registers can be categorized as follows:

OTG_HS host configuration register (OTG_HS_HCFG)

Address offset: 0x400

Reset value: 0x0000 0000

This register configures the core after power-on. Do not change to this register after initializing the host.



Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support
The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.
0: HS/FS/LS, based on the maximum speed supported by the connected device
1: FS/LS-only, even if the connected device can support HS (read-only)

Bits 1:0 **FSLSPCS:** FS/LS PHY clock select
When the core is in FS host mode
01: PHY clock is running at 48 MHz
Others: Reserved
When the core is in LS host mode
01: PHY clock is running at 48 MHz.
Others: Reserved

OTG_HS Host frame interval register (OTG_HS_HFIR)

Address offset: 0x404

Reset value: 0x0000 EA60

This register stores the frame interval information for the current speed to which the OTG_HS controller has enumerated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FRIVL															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **FRIVL:** Frame interval

The value that the application programs to this field specifies the interval between two consecutive SOFs (FS), micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the Port enable bit of the host port control and status register (PENA bit in OTG_HS_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY Clock Select field of the Host configuration register (FSLSPCS in OTG_HS_HCFG):

frame duration × PHY clock frequency

Note: The FRIVL bit can be modified whenever the application needs to change the Frame interval time.

OTG_HS host frame number/frame time remaining register (OTG_HS_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTREM																FRNUM															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

OTG_HS_Host periodic transmit FIFO/queue status register (OTG_HS_HPTXSTS)

Address offset: 0x410

Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic TxFIFO and the periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXQTOP								PTXQSAV								PTXFSAVL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **PTXQTOP:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit [31]: Odd/Even frame

- 0: send in even (micro) frame
- 1: send in odd (micro) frame

Bits [30:27]: Channel/endpoint number

Bits [26:25]: Type

- 00: IN/OUT
- 01: Zero-length packet
- 11: Disable channel command

Bit [24]: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full

01: dx1 location available

10: dx2 locations available

bxn: dxn locations available (0 ≤ dxn ≤ PTXFD)

Others: Reserved

Bits 15:0 **PTXFSAVL:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic TxFIFO.

Values are in terms of 32-bit words

0000: Periodic TxFIFO is full

0001: dx1 word available

0010: dx2 words available

bxn: dxn words available (where 0 ≤ dxn ≤ dx512)

Others: Reserved

OTG_HS Host all channels interrupt register (OTG_HS_HAINT)

Address offset: 0x414

Reset value: 0x0000 000

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the Core interrupt register (HCINT bit in OTG_HS_GINTSTS). This is shown in *Figure 368*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HAINT															
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT:** Channel interrupts

One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

OTG_HS host all channels interrupt mask register (OTG_HS_HAINTMSK)

Address offset: 0x418

Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HAINTM															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM:** Channel interrupt mask

0: Masked interrupt

1: Unmasked interrupt

One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

OTG_HS host port control and status register (OTG_HS_HPRT)

Address offset: 0x440

Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 368*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_HS_GINTSTS). On a Port Interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
													r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD:** Port speed

Indicates the speed of the device attached to this port.
00: High speed
01: Full speed
10: Low speed
11: Reserved

Bits 16:13 **PTCTL:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
0: Power off
1: Power on

Bits 11:10 **PLSTS:** Port line status

Indicates the current logic level USB data lines
Bit [10]: Logic level of OTG_HS_FS_DP
Bit [11]: Logic level of OTG_HS_FS_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in Suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the Port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the Port reset bit or Port resume bit in this register or the Resume/remote wakeup detected interrupt bit or Disconnect detected interrupt bit in the Core interrupt register (WKUINT or DISCINT in OTG_HS_GINTSTS, respectively).

0: Port not in Suspend mode

1: Port in Suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the Port resume/remote wakeup detected interrupt bit of the Core interrupt register (WKUINT bit in OTG_HS_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the Port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the Port enable bit [2] in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the Core interrupt register (HPRTINT bit in OTG_HS_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

OTG_HS host channel-x characteristics register (OTG_HS_HCCHARx) (x = 0..11, where x = Channel_number)

Address offset: 0x500 + (Channel_number × 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even (micro) frame

1: Odd (micro) frame

Bits 28:22 **DAD:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MC:** Multi Count (MC) / Error Count (EC)

– When the split enable bit (SPLITEN) in the host channel-x split control register (OTG_HS_HCSPLTx) is reset (0), this field indicates to the host the number of transactions that must be executed per micro-frame for this periodic endpoint. For nonperiodic transfers, this field specifies the number of packets to be fetched for this channel before the internal DMA engine changes arbitration.

00: Reserved This field yields undefined results
01: 1 transaction
b10: 2 transactions to be issued for this endpoint per micro-frame
11: 3 transactions to be issued for this endpoint per micro-frame.

– When the SPLITEN bit is set (1) in OTG_HS_HCSPLTx, this field indicates the number of immediate retries to be performed for a periodic split transaction on transaction errors. This field must be set to at least 01.

Bits 19:18 **EPTYP:** Endpoint type

Indicates the transfer type selected.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.
0: OUT
1: IN

Bits 14:11 **EPNUM:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

OTG_HS host channel-x split control register (OTG_HS_HCSPLTx) (x = 0..11, where x = Channel_number)

Address offset: 0x504 + (Channel_number × 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
rw															rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPLITEN:** Split enable

The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT:** Do complete split

The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS:** Transaction position

This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.

11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)

10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)

00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)

01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR:** Hub address

This field holds the device address of the transaction translator's hub.

Bits 6:0 **PRTADDR:** Port address

This field is the port number of the recipient transaction translator.

OTG_HS host channel-x interrupt register (OTG_HS_HCINTx) (x = 0..11, where x = Channel_number)

Address offset: 0x508 + (Channel_number × 0x20)

Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 368*. The application must read this register when the host channels interrupt bit in the Core interrupt register (HCINT bit in OTG_HS_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HS_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HS_HAINT and OTG_HS_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR:** Data toggle error

Bit 9 **FRMOR:** Frame overrun

Bit 8 **BBERR:** Babble error

Bit 7 **TXERR:** Transaction error

Indicates one of the following errors occurred on the USB.

CRC check failure
Timeout
Bit stuff error
False EOP

Bit 6 **NYET:** Response received interrupt

Bit 5 **ACK:** ACK response received/transmitted interrupt

Bit 4 **NAK:** NAK response received interrupt

Bit 3 **STALL:** STALL response received interrupt

Bit 2 **AHBERR**: AHB error

This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH:** Channel halted

Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

Bit 0 **XFRC:** Transfer completed

Transfer completed normally without any errors.

OTG_HS host channel-x interrupt mask register (OTG_HS_HCINTMSKx) (x = 0..11, where x = Channel_number)

Address offset: 0x50C + (Channel_number × 0x20)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous section.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
																					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **NYET:** response received interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERR:** AHB error

This is generated only in Internal DMA mode when there is an AHB error during AHB read/write. The application can read the corresponding channel's DMA address register to get the error address.

Bit 1 **CHHM:** Channel halted mask

0: Masked interrupt

1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask

0: Masked interrupt

1: Unmasked interrupt

OTG_HS host channel-x transfer size register (OTG_HS_HCTSIZx) (x = 0..11, where x = Channel_number)

Address offset: 0x510 + (Channel_number × 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DPID		PKTCNT										XFRSIZ																		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPING:** Do ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

Note: Do not set this bit for IN transfers. If this bit is set for IN transfers it disables the channel.

Bits 30:29 **DPID:** Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0

01: DATA2

10: DATA1

11: MDATA (noncontrol)/SETUP (control)

Bits 28:19 **PKTCNT:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and nonperiodic).

OTG_HS host channel-x DMA address register (OTG_HS_HCDMAx) (x = 0..11, where x = Channel_number)

Address offset: 0x514 + (Channel_number × 0x20)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR:** DMA address

This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

30.12.4 Device-mode registers

OTG_HS device configuration register (OTG_HS_DCFG)

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in peripheral mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
						rw	rw												rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **PERSCHIVL:** Periodic scheduling interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro)frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro)frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame
01: 50% of (micro)frame
10: 75% of (micro)frame
11: Reserved

Bits 23:13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL:** Periodic (micro)frame interval

Indicates the time within a (micro) frame at which the application must be notified using the end of periodic (micro) frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD:** Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Nonzero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage.

1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.

0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD:** Device speed

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

00: High speed
01: Reserved
10: Reserved
11: Full speed (USB 1.1 transceiver clock is 48 MHz)

OTG_HS device control register (OTG_HS_DCTL)

Address offset: 0x804

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
																				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

A write to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

A write to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the Core interrupt register (GONAKEFF bit in OTG_HS_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

A write to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

A write to this field sets the Global nonperiodic IN NAK.The application uses this bit to send a NAK handshake on all nonperiodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the Core interrupt register (GINAKEFF bit in OTG_HS_GINTSTS) is cleared.

Bits 6:4 **TCTL:** Test control

000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status

0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings.
1: No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status

0: A handshake is sent out based on the data availability in the transmit FIFO.
1: A NAK handshake is sent out on all nonperiodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.
0: Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.
1: The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

Table 161 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 161. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
High speed	Not Idle or Suspended (Performing transactions)	125 µs
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or Suspended (Performing transactions)	2.5 µs

OTG_HS device status register (OTG_HS_DSTS)

Address offset: 0x808

Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_HS_DAINT) register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
										r	r	r	r	r	r	r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:8 **FNSOF:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.

Due to erratic errors, the OTG_HS controller goes into Suspended state and an interrupt is generated to the application with Early suspend bit of the Core interrupt register (ESUSP bit in OTG_HS_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD:** Enumerated speed

Indicates the speed at which the OTG_HS controller has come up after speed detection through a chirp sequence.

00: High speed

01: Reserved

10: Reserved

11: Full speed (PHY clock is running at 48 MHz)

Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In peripheral mode, this bit is set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines

– When the application writes to the Remote wakeup signaling bit in the Device control register (RWUSIG bit in OTG_HS_DCTL).

OTG_HS device IN endpoint common interrupt mask register (OTG_HS_DIEPMSK)

Address offset: 0x810

Reset value: 0x0000 0000

This register works with each of the Device IN endpoint interrupt (OTG_HS_DIEPINTx) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_HS_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																						rw	rw		rw	rw	rw	rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BIM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when TxFIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (nonisochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

OTG_HS device OUT endpoint common interrupt mask register (OTG_HS_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the Device OUT endpoint interrupt (OTG_HS_DOEPINTx) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_HS_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																						BOIM	OPEM	Reserved	B2BSTUP	Reserved	OTEPDM	STUPM	Reserved	EPDM	XFRCM
																						rw	rw		rw		rw	rw		rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BOIM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **OPEM:** OUT packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** SETUP phase done mask
Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

OTG_HS device all endpoints interrupt register (OTG_HS_DAINT)

Address offset: 0x818

Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a device all endpoints interrupt register interrupts the application using the Device OUT endpoints interrupt bit or Device IN endpoints interrupt bit of the Core interrupt register (OEPINT or IEPINT in OTG_HS_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-x interrupt register (OTG_HS_DIEPINTx/OTG_HS_DOEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEPINT																IEPINT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 31 for OUT endpoint 15

Bits 15:0 **IEPINT:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 15 for endpoint 15

OTG_HS all endpoints interrupt mask register (OTG_HS_DAINTMSK)

Address offset: 0x81C

Reset value: 0x0000 0000

The device endpoint interrupt mask register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the device all endpoints interrupt (OTG_HS_DAINT) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEPM																IEPM															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 18 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

OTG_HS device V_{BUS} discharge time register (OTG_HS_DVBUSDIS)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VBUSDT															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VBUSDT:** Device V_{BUS} discharge time
Specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP. This value equals:
V_{BUS} discharge time in PHY clocks / 1 024
Depending on your V_{BUS} load, this value may need adjusting.

OTG_HS device V_{BUS} pulsing time register (OTG_HS_DVBUSPULSE)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register specifies the V_{BUS} pulsing time during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				DVBUSP											
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DVBUSP:** Device V_{BUS} pulsing time
Specifies the V_{BUS} pulsing time during SRP. This value equals:
V_{BUS} pulsing time in PHY clocks / 1 024

OTG_HS Device threshold control register (OTG_HS_DTHRCTL)

Address offset: 0x0830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ARPEN	Reserved	RXTHRLEN									RXTHREN	Reserved					TXTHRLEN									ISOTHREN	NONISOTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **ARPEN:** Arbiter parking enable

This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.

Bit 26 Reserved, must be kept at reset value.

Bits 25: 17 **RXTHRLEN:** Receive threshold length

This field specifies the receive thresholding size in DWORDS. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_HS_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable

When this bit is set, the core enables thresholding in the receive direction.

Bits 15: 11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN:** Transmit threshold length

This field specifies the transmit thresholding size in DWORDS. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmitting on the USB. The threshold length has to be at least eight DWORDS. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_HS_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable

When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable

When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

OTG_HS device IN endpoint FIFO empty interrupt mask register: (OTG_HS_DIEPEMPMSK)

Address offset: 0x834

Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_HS_DIEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INEPTXFEM															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_HS_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 15 for IN endpoint 15
0: Masked interrupt
1: Unmasked interrupt

OTG_HS device each endpoint interrupt register (OTG_HS_DEACHINT)

Address offset: 0x0838

Reset value: 0x0000 0000

There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OEP1INT	Reserved															IEP1INT	Reserved
														rw																rw	

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INT:** OUT endpoint 1 interrupt bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INT:** IN endpoint 1interrupt bit

Bit 0 Reserved, must be kept at reset value.

OTG_HS device each endpoint interrupt register mask (OTG_HS_DEACHINTMSK)

Address offset: 0x083C

Reset value: 0x0000 0000

There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OEP1INTM	Reserved															IEP1INTM	Reserved

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **OEP1INTM:** OUT Endpoint 1 interrupt mask bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 **IEP1INTM:** IN Endpoint 1 interrupt mask bit

Bit 0 Reserved, must be kept at reset value.

OTG_HS device each in endpoint-1 interrupt register (OTG_HS_DIEPEACHMSK1)

Address offset: 0x840

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAKM	Reserved			BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																		rw				rw	rw		rw	rw	rw	rw		rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: unmasked interrupt

Bit 12:10 Reserved, must be kept at reset value.

Bit 9 **BIM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when TxFIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (nonisochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

OTG_HS device each OUT endpoint-1 interrupt register (OTG_HS_DOEPEACHMSK1)

Address offset: 0x880

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	NYETM	NAKM	BERRM	Reserved		BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
																	rw	rw	rw			rw	rw		rw	rw	rw	rw		rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETM**: NYET interrupt mask
0: Masked interrupt
1: unmasked interrupt

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM:** Bubble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 11:10 Reserved, must be kept at reset value.

Bit 9 **BIM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **OPEM:** OUT packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

OTG device endpoint-x control register (OTG_HS_DIEPCTLx) (x = 0..7, where x = Endpoint_number)

Address offset: 0x900 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/ rs		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint disabled interrupt. The application must set this bit only if Endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM:** TxFIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to noncontrol, nonisochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting nonNAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For nonisochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the TxFIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the TxFIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

OTG_HS device control OUT endpoint 0 control register (OTG_HS_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the device control OUT endpoint 0 control register. Nonzero control endpoints use registers for endpoints 1–15.

31	30	29 28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
EPENA	EPDIS	Reserved	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	Reserved	USBAEP	Reserved	MPSIZ
w	r		w	w		rs	rw	r r	r		r		r r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.
The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a Transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP:** Endpoint type

Hardcoded to 2'b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting nonNAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

OTG_HS device endpoint-x control register (OTG_HS_DOEPCTLx) (x = 1..3, where x = Endpoint_number)

Address offset for OUT endpoints: 0xB00 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint disabled interrupt. The application must set this bit only if Endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to noncontrol, nonisochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting nonNAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

OTG_HS device endpoint-x interrupt register (OTG_HS_DIEPINTx) (x = 0..7, where x = Endpoint_number)

Address offset: 0x908 + (Endpoint_number × 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 368*. The application must read this register when the IN endpoints interrupt bit of the Core interrupt register (IEPINT in OTG_HS_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_HS_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HS_DAINT and OTG_HS_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
																								r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK interrupt

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit10 Reserved, must be kept at reset value.

Bit 9 **BNA:** Buffer not available interrupt

The core generates this interrupt when the descriptor accessed is not ready for the Core to process, such as host busy or DMA done.

Bit 8 **TXFIFOUDRN**: Transmit Fifo Underrun (TxfifoUndrn) The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint.

Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO empty level bit in the Core AHB configuration register (TXFELVL bit in OTG_HS_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_HS_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **ITTXFE:** IN token received when TxFIFO is empty

Applies to nonperiodic IN endpoints only.

Indicates that an IN token was received when the associated TxFIFO (periodic/nonperiodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Applies only to Control IN endpoints.

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

OTG_HS device endpoint-x interrupt register (OTG_HS_DOEPINTx) (x = 0..7, where x = Endpoint_number)

Address offset: 0xB08 + (Endpoint_number × 0x20)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 368*. The application must read this register when the OUT Endpoints Interrupt bit of the Core interrupt register (OEPINT bit in OTG_HS_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_HS_DAINT) register to get the exact endpoint number for the device Endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HS_DAINT and OTG_HS_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
																									rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYET:** NYET interrupt
The core generates this interrupt when a NYET response is transmitted for a nonisochronous OUT endpoint.

Bits 13:7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received
Applies to Control OUT endpoint only.
This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled
Applies only to control OUT endpoint.
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done
Applies to control OUT endpoints only.
Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD:** Endpoint disabled interrupt
This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

OTG_HS device IN endpoint 0 transfer size register (OTG_HS_DIEPTSIZ0)

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_HS_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–15.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											PKTCNT		Reserved												XFRSIZ						
											rw	rw													rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the TxFIFO.

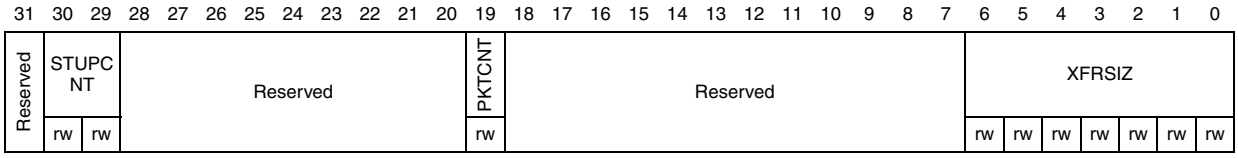
OTG_HS device OUT endpoint 0 transfer size register (OTG_HS_DOEPTSIZ0)

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the Endpoint enable bit in the device control endpoint 0 control registers (EPENA bit in OTG_HS_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–15.



Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the RxFIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

OTG_HS device endpoint-x transfer size register (OTG_HS_DIEPTSIZx) (x = 1..3, where x = Endpoint_number)

Address offset: 0x910 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the Endpoint enable bit in the device endpoint-x control registers (EPENA bit in OTG_HS_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bit 28:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.

Bits 18:0 **XFRSIZ:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the TxFIFO.

OTG_HS device IN endpoint transmit FIFO status register (OTG_HS_DTXFSTSx) (x = 0..5, where x = Endpoint_number)

Address offset for IN endpoints: 0x918 + (Endpoint_number × 0x20) This read-only register contains the free space information for the Device IN endpoint TxFIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INEPTFSAV															
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

31:16 Reserved, must be kept at reset value.

15:0 **INEPTFSAV:** IN endpoint TxFIFO space avail ()
Indicates the amount of free space available in the Endpoint TxFIFO.
Values are in terms of 32-bit words:
0x0: Endpoint TxFIFO is full
0x1: 1 word available
0x2: 2 words available
0xn: n words available (0 < n < 512)
Others: Reserved

OTG_HS device endpoint-x transfer size register (OTG_HS_DOEPTSIZx) (x = 1..5, where x = Endpoint_number)

Address offset: 0xB10 + (Endpoint_number × 0x20)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the device endpoint-x control registers (EPENA bit in OTG_HS_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the Endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID:** Received data PID
Applies to isochronous OUT endpoints only.
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT: SETUP packet count

Applies to control OUT Endpoints only.

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bit 28:19 **PKTCNT:** Packet count

Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.

Bits 18:0 **XFRSIZ:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.

OTG_HS device endpoint-*x* DMA address register (OTG_HS_DIEPDMAx / OTG_HS_DOEPDMAx) (x = 1..5, where x = Endpoint_number)

Address offset for IN endpoints: 0x914 + (Endpoint_number × 0x20)

Reset value: 0xXXXX XXXX

Address offset for OUT endpoints: 0xB14 + (Endpoint_number × 0x20)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR																															

Bits 31:0 **DMAADDR:** DMA address

This bit holds the start address of the external memory for storing or fetching endpoint data.

Note: For control endpoints, this field stores control OUT data packets as well as SETUP transaction data packets. When more than three SETUP packets are received back-to-back, the SETUP data packet in the memory is overwritten. This register is incremented on every AHB transaction. The application can give only a DWORD-aligned address.

30.12.5 OTG_HS power and clock gating control register (OTG_HS_PCGCCTL)

Address offset: 0xE00

Reset value: 0x0000 0000

This register is available in host and peripheral modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved			GATEHCLK	STPPCLK
																											rw			rw	rw

Bit 31:5 Reserved, must be kept at reset value.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit (bit 0).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

30.12.6 OTG_HS register map

The table below gives the USB OTG register map and reset values.

Table 162. OTG_HS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_HS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	OTG_HS_GOTGINT	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	OTG_HS_GAHBCFG	Reserved																							PTXFELVL	TXFELVL	Reserved						GINT
	Reset value																								0	0							0
0x00C	OTG_HS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved			ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT				HNPCAP	SRPCAP	Reserved					TOCAL		
	Reset value	0	0	0				0	0	0	0	0	0	0	0	0		0		0	0	1	0	1	0						0	0	0
0x010	OTG_HS_GRSTCTL	AHBIDL	DMAREQ	Reserved																TXFNUM									TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1	0																	0	0	0	0	0					0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	**OTG_HS_GINTSTS**	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0		0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	**OTG_HS_GINTMSK**	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0		0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	**OTG_HS_GRXSTSR (Host mode)**	Reserved											PKTSTS				DPID		BCNT												CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSR (peripheral mode)	Reserved								FRMNUM			PKTSTS				DPID		BCNT												EPNUM		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**OTG_HS_GRXSTSP (Host mode)**	Reserved											PKTSTS				DPID		BCNT												CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSP (peripheral mode)	Reserved								FRMNUM			PKTSTS				DPID		BCNT												EPNUM		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**OTG_HS_GRXFSIZ**	Reserved																RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	**OTG_HS_GNPTXFSIZ (Host mode)**	NPTXFD																NPTXFSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	OTG_HS_GNPTXFSIZ (peripheral mode)	TX0FD																TX0FSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	**OTG_HS_GNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x030	**OTG_HS_GI2CCTL**	BSYDNE	RW	Reserved	I2CDATSE0	I2CDEVADR		Reserved	ACK	I2CEN	ADDR							REGADDR								RWDATA								
	Reset value	0	0		0	0	0		0	0	0	0	0	0	0	0	0																
0x038	**OTG_HS_GCCFG**	Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	.I2CPADEN	.PWRDWN	Reserved																
	Reset value											0	0	0	0	0	0																
0x03C	**OTG_HS_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	OTG_HS_HPTXFSIZ	PTXFD																PTXSA															
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0
0x104	OTG_HS_DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	OTG_HS_DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	OTG_HS_DIEPTXF3	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x110	OTG_HS_DIEPTXF4	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_HS_HCFG	Reserved																													FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	OTG_HS_HFIR	Reserved																FRIVL															
	Reset value																	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_HS_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_HS_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	OTG_HS_HAINT	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_HS_HAINTMSK	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_HS_HPRT	Reserved													PSPD	PTCTL			PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS			
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x500	OTG_HS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_HS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	OTG_HS_HCCHAR2	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	OTG_HS_HCCHAR3	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	OTG_HS_HCCHAR4	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	OTG_HS_HCCHAR5	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM					MPSIZ									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C0	**OTG_HS_HCCHAR6**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	**OTG_HS_HCCHAR7**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x600	**OTG_HS_HCCHAR8**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x620	**OTG_HS_HCCHAR9**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x640	**OTG_HS_HCCHAR10**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	**OTG_HS_HCCHAR11**	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	**OTG_HS_HCSPLT0**	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_HS_HCINT0**	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x524	**OTG_HS_HCSPL1**	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	**OTG_HS_HCINT1**	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x544	**OTG_HS_HCSPLT2**	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x548	**OTG_HS_HCINT2**	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x564	**OTG_HS_HCSPLT3**	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x568	OTG_HS_HCINT3	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x584	OTG_HS_HCSPLT4	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x588	OTG_HS_HCINT4	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5A4	OTG_HS_HCSPLT5	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A8	OTG_HS_HCINT5	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5C4	OTG_HS_HCSPLT6	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8	OTG_HS_HCINT6	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5E4	OTG_HS_HCSPLT7	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E8	OTG_HS_HCINT7	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x604	OTG_HS_HCSPLT8	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x608	OTG_HS_HCINT8	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x624	OTG_HS_HCSPLT9	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x628	OTG_HS_HCINT9	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x644	OTG_HS_HCSPLT10	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x648	OTG_HS_HCINT10	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_HS_HCSPLT11	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	OTG_HS_HCINT11	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_HS_HCINTMSK0	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM		CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x52C	OTG_HS_HCINTMSK1	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x54C	OTG_HS_HCINTMSK2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x56C	OTG_HS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x58C	OTG_HS_HCINTMSK4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5AC	OTG_HS_HCINTMSK5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5CC	OTG_HS_HCINTMSK6	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5EC	OTG_HS_HCINTMSK7	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60C	OTG_HS_HCINTMSK8	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x62C	OTG_HS_HCINTMSK9	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x64C	OTG_HS_HCINTMSK10	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x66C	OTG_HS_HCINTMSK11	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERR	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x510	OTG_HS_HCTSIZ0	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_HS_HCTSIZ1	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	OTG_HS_HCTSIZ2	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	OTG_HS_HCTSIZ3	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	OTG_HS_HCTSIZ4	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	OTG_HS_HCTSIZ5	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	OTG_HS_HCTSIZ6	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	OTG_HS_HCTSIZ7	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x610	OTG_HS_HCTSIZ8	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x630	OTG_HS_HCTSIZ9	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x650	OTG_HS_HCTSIZ10	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x670	OTG_HS_HCTSIZ11	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HS_HCDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x524	OTG_HS_HCDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x544	OTG_HS_HCDMA2	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x564	OTG_HS_HCDMA3	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x584	OTG_HS_HCDMA4	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A4	OTG_HS_HCDMA5	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C4	OTG_HS_HCDMA6	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E4	OTG_HS_HCDMA7	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x604	OTG_HS_HCDMA8	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x624	OTG_HS_HCDMA9	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x644	OTG_HS_HCDMA10	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_HS_HCDMA11	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_HS_DCFG	Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
	Reset value							1	0												0	0	0	0	0	0	0	0	0		0	0	0
0x804	OTG_HS_DCTL	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	OTG_HS_DSTS	Reserved										FNSOF														Reserved				EERR	ENUMSPD	SUSPSTS	
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	
0x810	OTG_HS_DIEPMSK	Reserved																						BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																							0	0	0	0	0	0	0		0	0
0x814	OTG_HS_DOEPMSK	Reserved																						BOIM	OPEM	Reserved	B2BSTUP	Reserved	OTEPDM	STUPM	Reserved	EPDM	XFRCM
	Reset value																							0	0		0	0	0	0		0	0
0x818	OTG_HS_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_HS_DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x828	OTG_HS_DVBUSDIS	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	OTG_HS_DVBUSPULSE	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x830	OTG_HS_DTHRCTL	Reserved				ARPEN	Reserved	RXTHRLEN									RXTHREN	Reserved				TXTHRLEN										ISOTHREN	NONISOTHREN
	Reset value					0		0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_HS_DIEPEMPMSK	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x838	OTG_HS_DEACHINT	Reserved																Reserved															Reserved
	Reset value															0																0	
0x83C	OTG_HS_DEACHINTMSK	Reserved																Reserved															Reserved
	Reset value															0																0	
0x840	OTG_HS_DIEPEACHMSK1	Reserved																		NAKM	Reserved		BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																			0			0	0		0	0	0	0		0	0
0x880	OTG_HS_DOEPEACHMSK1	Reserved																	NYETM	NAKM	BERRM	Reserved	BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																		0	0	0		0	0		0	0	0	0		0	0
0x900	OTG_HS_DIEPCTL0	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved			MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0
0x918	TG_FS_DTXFSTS0	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_HS_DIEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved			MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0
0x938	TG_FS_DTXFSTS1	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x940	OTG_HS_DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved			MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0
0x958	TG_FS_DTXFSTS2	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x960	**OTG_HS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x980	**OTG_HS_DIEPCTL4**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9A0	**OTG_HS_DIEPCTL5**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9C0	**OTG_HS_DIEPCTL6**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9E0	**OTG_HS_DIEPCTL7**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB00	**OTG_HS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB20	**OTG_HS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	**OTG_HS_DOEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB60	OTG_HS_DOEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ											
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_HS_DIEPINT0	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x928	OTG_HS_DIEPINT1	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x948	OTG_HS_DIEPINT2	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x968	OTG_HS_DIEPINT3	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x988	OTG_HS_DIEPINT4	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9A8	OTG_HS_DIEPINT5	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9C8	OTG_HS_DIEPINT6	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0x9E8	OTG_HS_DIEPINT7	Reserved																		NAK	BERR	PKTDRPSTS	Reserved	BNA	TXFIFOUDRN	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0	0	0		0	0	1	0		0	0		0	0
0xB08	OTG_HS_DOEPINT0	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	1		0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB28	**OTG_HS_DOEPINT1**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB48	**OTG_HS_DOEPINT2**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB68	**OTG_HS_DOEPINT3**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xB88	**OTG_HS_DOEPINT4**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBA8	**OTG_HS_DOEPINT5**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBC8	**OTG_HS_DOEPINT6**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0xBE8	**OTG_HS_DOEPINT7**	Reserved																	NYET	Reserved							B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																		0								0		0	0		0	0
0x910	**OTG_HS_DIEPTSIZ0**	Reserved											PKTCNT		Reserved												XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x930	**OTG_HS_DIEPTSIZ1**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x934	**OTG_HS_DIEPDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x93C	**OTG_HS_DIEPDMAB1**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	**OTG_HS_DIEPTSIZ2**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x954	**OTG_HS_DIEPDMA2**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x95C	**OTG_HS_DIEPDMAB2**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	**OTG_HS_DIEPTSIZ3**	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x974	**OTG_HS_DIEPDMA3**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x97C	**OTG_HS_DIEPDMAB3**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 162. OTG_HS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB10	**OTG_HS_DOEPTSIZ0**	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	**OTG_HS_DOEPTSIZ1**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	**OTG_HS_DOEPDMA1**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB3C	**OTG_HS_DOEPDMAB1**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	**OTG_HS_DOEPTSIZ2**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB54	**OTG_HS_DOEPDMA2**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB5C	**OTG_HS_DOEPDMAB2**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	**OTG_HS_DOEPTSIZ3**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB74	**OTG_HS_DOEPDMA3**	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB7C	**OTG_HS_DOEPDMAB3**	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	**OTG_HS_PCGCCTL**	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

Refer to *Table 1 on page 50* for the register boundary addresses.

30.13 OTG_HS programming model

30.13.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the Core interrupt register (CMOD bit in OTG_HS_GINTSTS) reflects the mode. The OTG_HS controller enters host mode when an “A” plug is connected or peripheral mode when a “B” plug is connected.

This section explains the initialization of the OTG_HS controller after power-on. The application must follow the initialization sequence irrespective of host or peripheral mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the Global AHB configuration (OTG_HS_GAHBCFG) register:
   - DMA mode bit
   - AHB burst length field
   - Global interrupt mask bit GINT = 1
   - RxFIFO nonempty (RXFLVL bit in OTG_HS_GINTSTS)
   - Periodic TxFIFO empty level
2. Program the following fields in OTG_HS_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - FS timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_HS_GINTSTS to determine whether the OTG_HS controller is operating in host or peripheral mode.

30.13.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in GINTMSK to unmask
2. Program the OTG_HS_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HS_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_HS_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HS_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HS_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HS_HPRT.
9. Read the PSPD bit in OTG_HS_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1.
11. Program the FSLSPCS field in OTG_FS_HCFG register according to the speed of the detected device read in step 9. If FSLSPCS has been changed, reset the port.
12. Program the OTG_HS_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HS_GNPTXFSIZ register to select the size and the start address of the nonperiodic transmit FIFO for nonperiodic transactions.
14. Program the OTG_HS_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

30.13.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_HS_DCFG register:
   - Device speed
   - Nonzero-length status OUT handshake
2. Program the OTG_HS_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
3. Program the VBUSBSEN bit in the OTG_HS_GCCFG register to enable V_{BUS} sensing in “B” peripheral mode and supply the 5 volts across the pull-up resistor on the DP line.
4. Wait for the USBRST interrupt in OTG_HS_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.

Wait for the ENUMDNE interrupt in OTG_HS_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_HS_DSTS register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 1198*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

30.13.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (HCDMAx register in host mode and DIEPDMAx/DOEPDMAx register in peripheral mode) to access the data buffers.

30.13.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the GINTMSK register to unmask the following:
2. Channel interrupt
   – Nonperiodic transmit FIFO empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the packet count field programmed with more than one).
   – Nonperiodic transmit FIFO half-empty for OUT transactions (applicable for Slave mode that operates in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HS_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HS_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HS_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the selected channels in the OTG_HS_HCSPLTx register(s) with the hub and port addresses (split transactions only).
7. Program the selected channels in the HCDMAx register(s) with the buffer start address.
8. Program the OTG_HS_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

Halting a channel

The application can disable any channel by programming the OTG_HS_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_HS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HS_HCINTx before reallocating the channel for other transactions. The OTG_HS host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG_HS host checks for space to write the disable request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the nonperiodic request queue (when disabling a nonperiodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the Request queue is full (before disabling the channel), by programming the OTG_HS_HCCHARx register with the CHDIS bit set to 1, and the CHENA bit cleared to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an XFRC interrupt in OTG_HS_HCINTx is received during a nonperiodic IN transfer or high-bandwidth interrupt IN transfer (Slave mode only)
2. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HS_HCINTx is received for an IN or OUT channel (Slave mode only). For high-bandwidth interrupt INs in Slave mode, once the application has received a DTERR interrupt it must disable the channel

and wait for a channel halted interrupt. The application must be able to receive other interrupts (DTERR, NAK, Data, TXERR) for the same channel before receiving the halt.

3. When a DISCINT (Disconnect Device) interrupt in OTG_HS_GINTSTS is received. (The application is expected to disable all enabled channels
4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG_HS host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints.

The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the HS_OTG host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO).

This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in HCTSIZx before enabling the channel or by just writing the HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the HS_OTG host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in HCTSIZx for a NAK/NYET response in case of Bulk/Control OUT. The OTG_HS host automatically sets the DOPING bit in HCTSIZx, and issues the ping tokens for Bulk/Control OUT. The HS_OTG host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

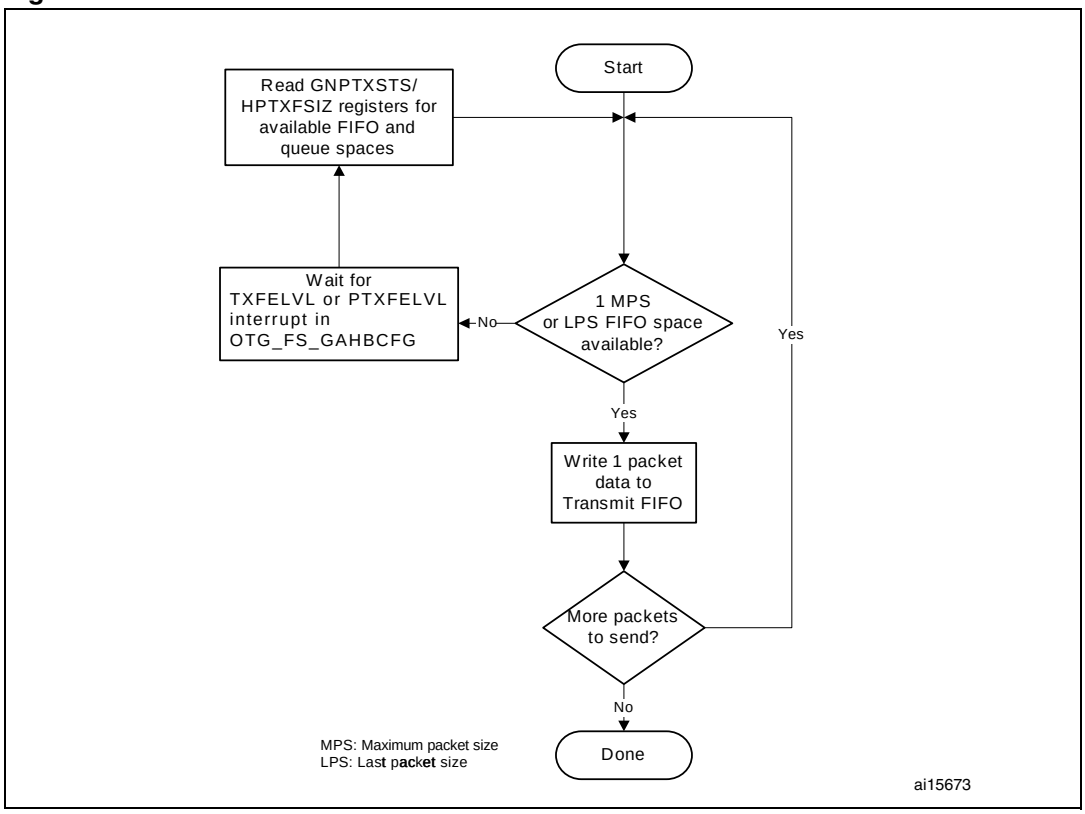
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

The OTG_HS host automatically writes an entry (OUT request) to the periodic/nonperiodic request queue, along with the last DWORD write of a packet. The application must ensure that at least one free space is available in the periodic/nonperiodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in DWORDs. If the packet size is nonDWORD aligned, the application must use padding. The OTG_HS host determines the actual packet size based on the programmed maximum packet size and transfer size.

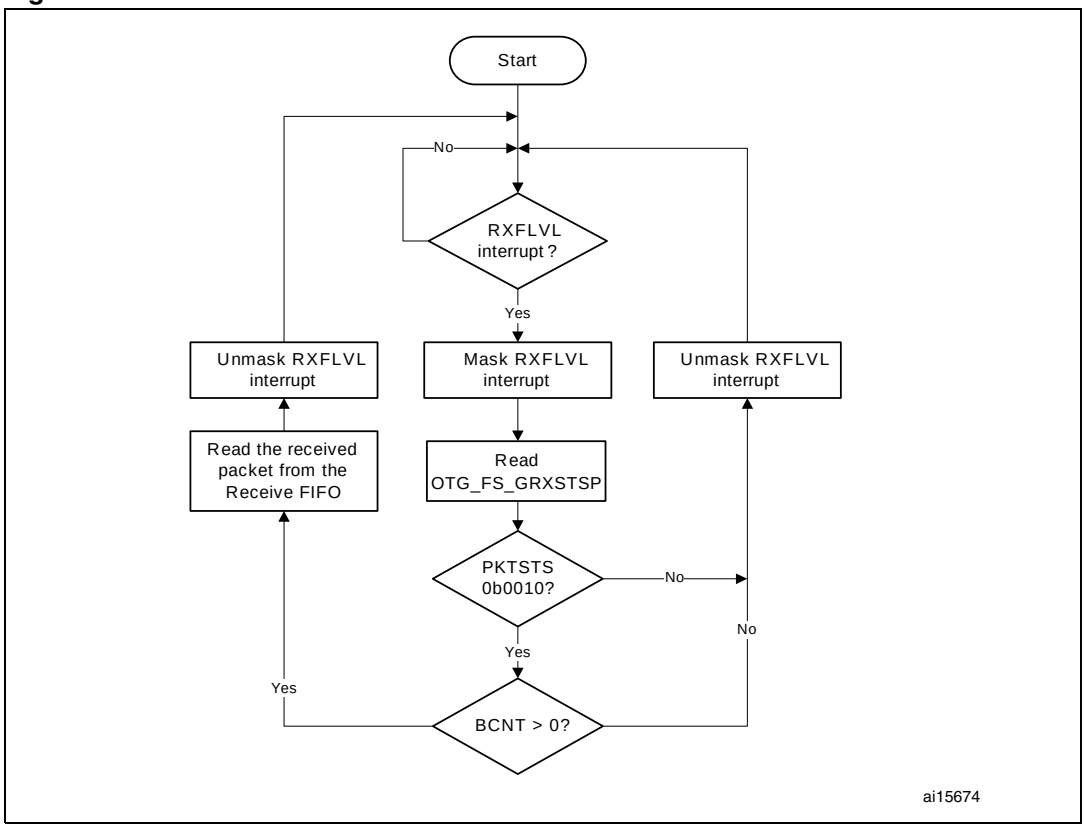
Figure 370. Transmit FIFO write task



- **Reading the receive FIFO**

The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 371. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 372*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control

SETUP transaction operates in the same way but has only one packet. The assumptions are:

– The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
– The nonperiodic transmit FIFO can hold two packets (128 bytes for FS).
– The nonperiodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations for channel 1 is as follows:

  a) Initialize channel 1
  b) Write the first packet for channel 1
  c) Along with the last DWORD write, the core writes an entry to the nonperiodic request queue
  d) As soon as the nonperiodic queue becomes nonempty, the core attempts to send an OUT token in the current frame
  e) Write the second (last) packet for channel 1
  f) The core generates the XFRC interrupt as soon as the last transaction is completed successfully
  g) In response to the XFRC interrupt, de-allocate the channel for other transfers
  h) Handling nonACK responses

Figure 372. Normal bulk/control OUT/SETUP and bulk/control IN transactions - DMA mode

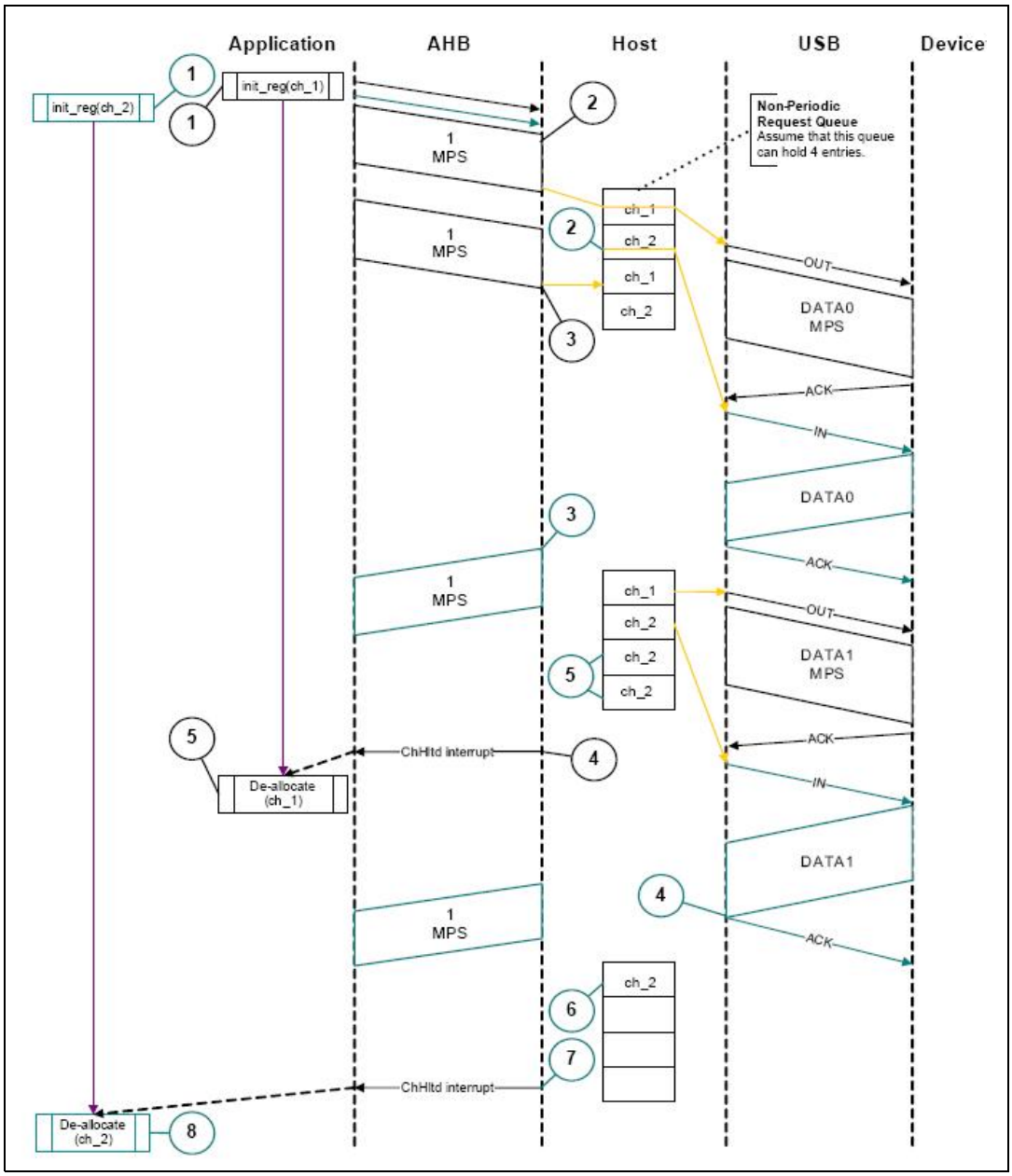
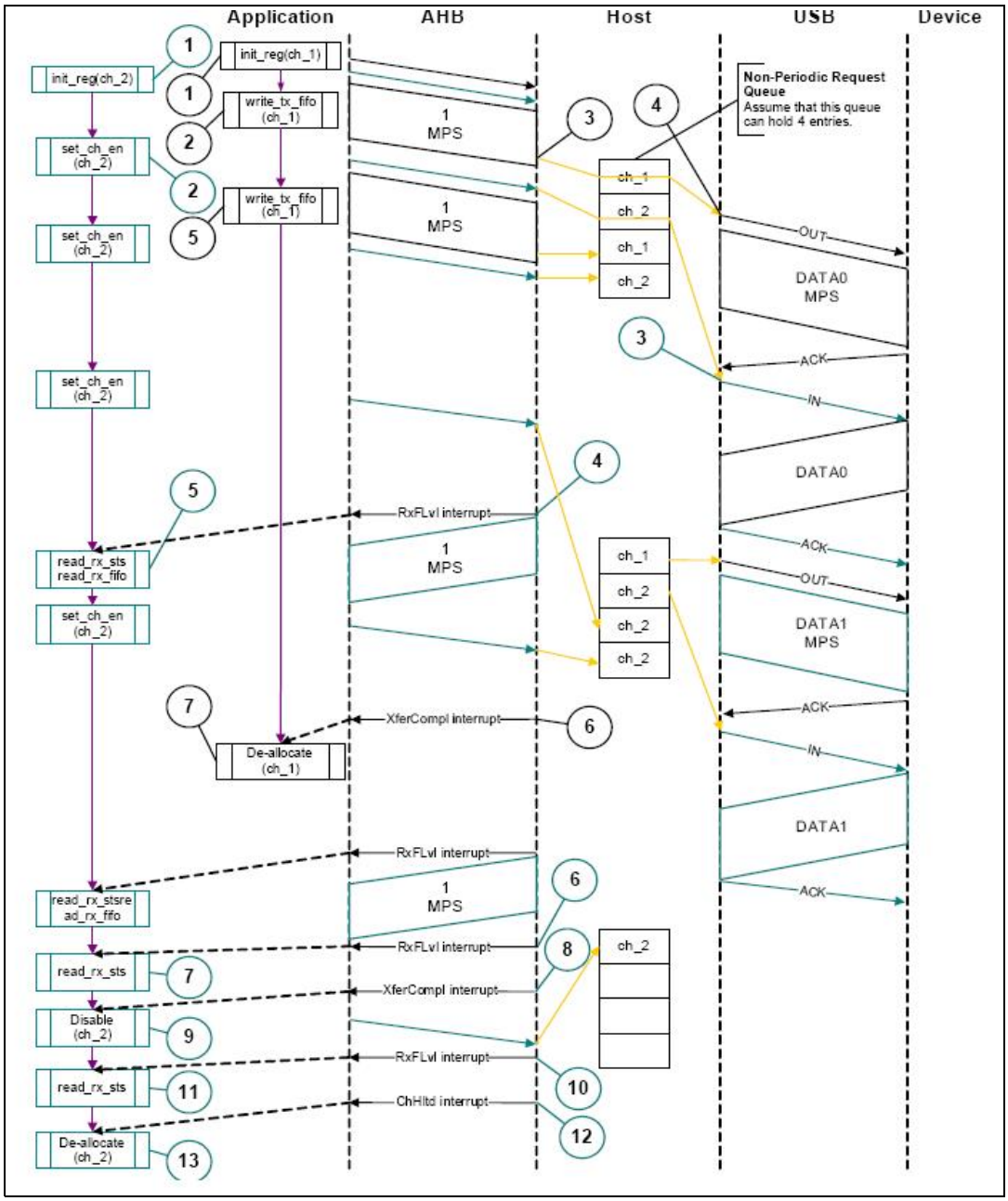


Figure 373. Normal bulk/control OUT/SETUP and bulk/control IN transactions - Slave mode



The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions in Slave mode is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

a) Bulk/Control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
```

```
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO as and when the space is available in the transmit FIFO and the Request queue. The application can make use of the NPTXFE interrupt in OTG_HS_GINTSTS to find the transmit FIFO space.

b) Bulk/Control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
```

```
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the Request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 374*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status DWORDs per packet (72 bytes for FS).

  – The nonperiodic request queue depth = 4.

Figure 374. Bulk/control IN transactions - DMA mode

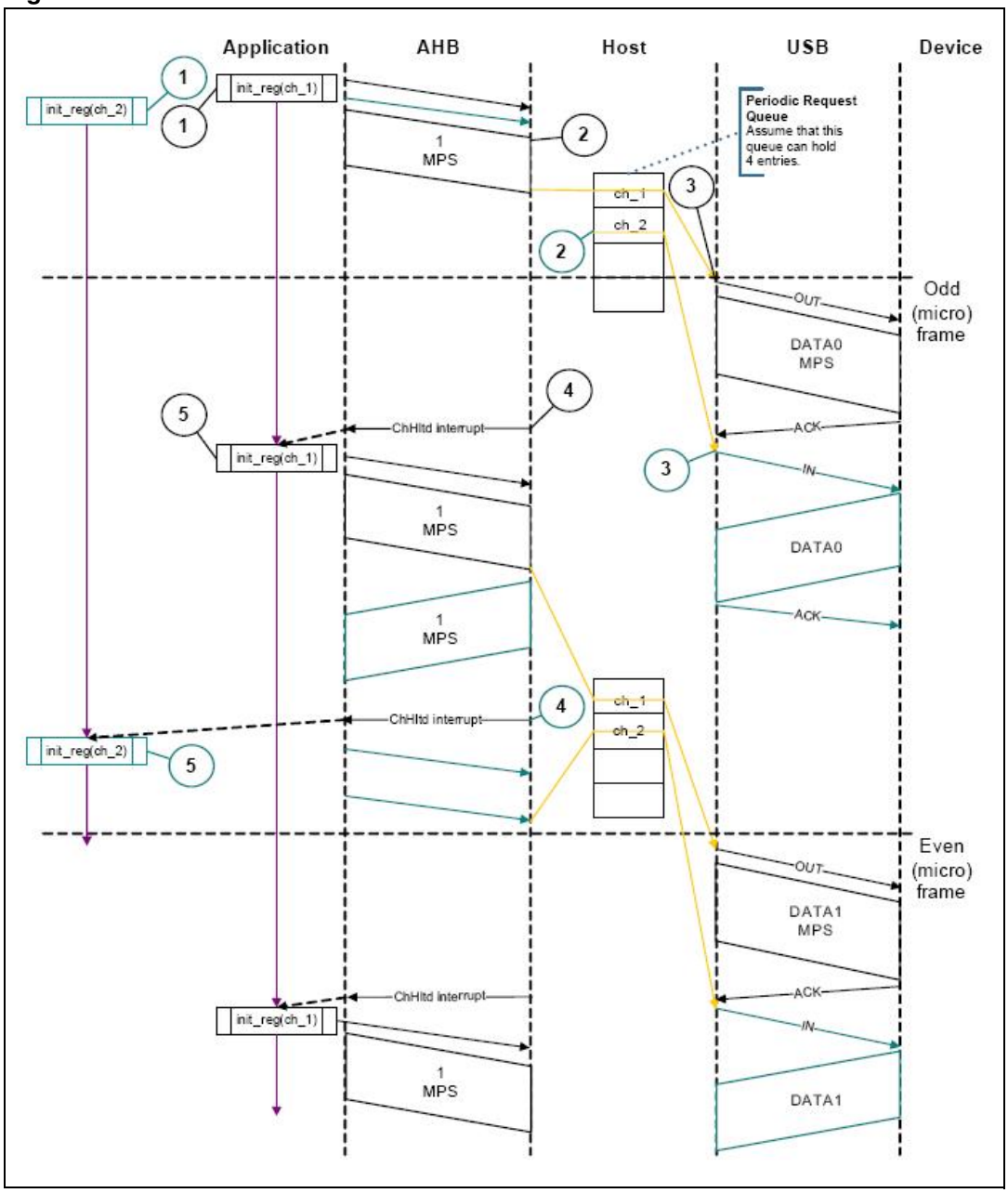
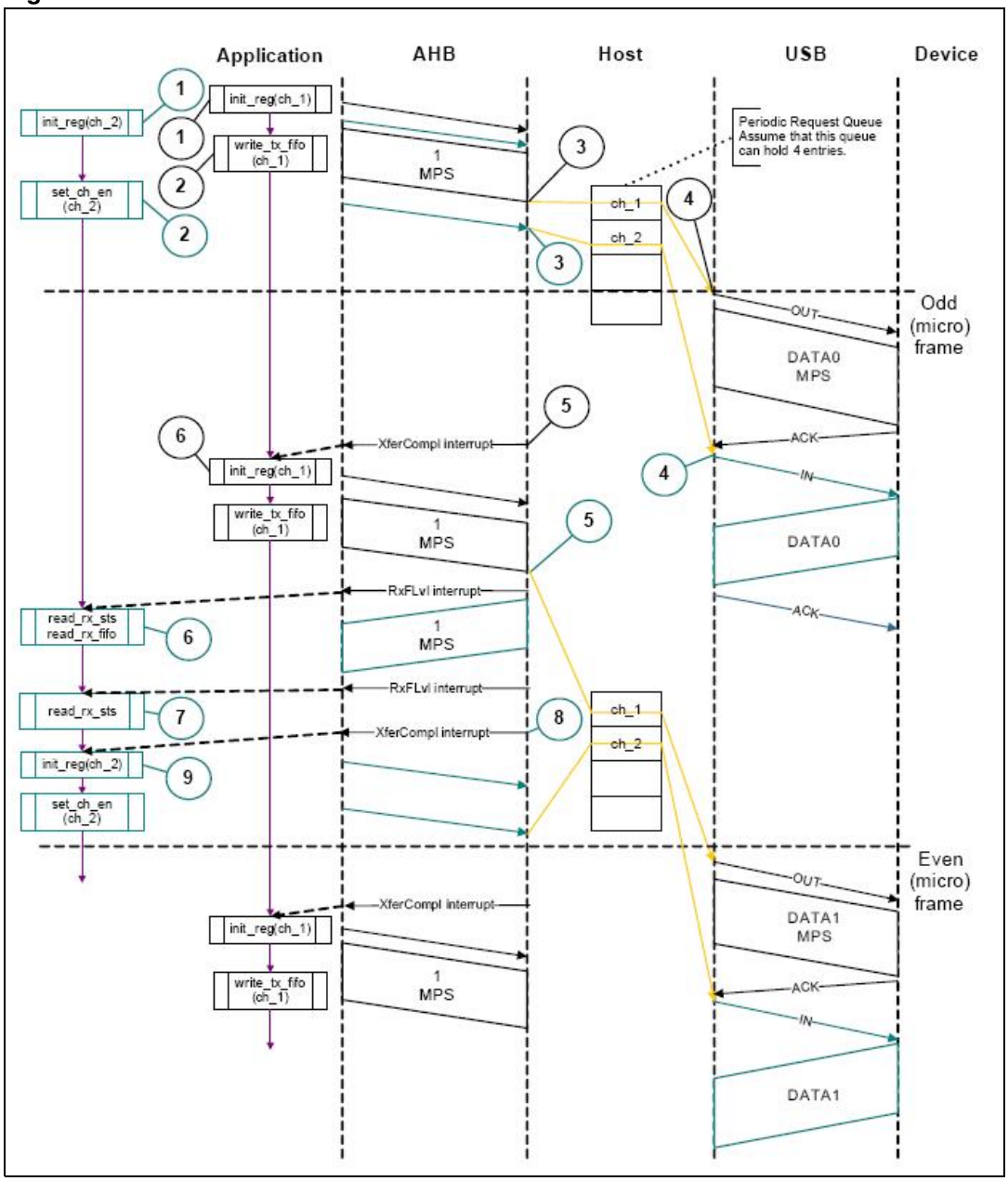


Figure 375. Bulk/control IN transactions - Slave mode



The sequence of operations is as follows:

a) Initialize channel 2.

b) Set the CHENA bit in HCCHAR2 to write an IN request to the nonperiodic request queue.

c) The core attempts to send an IN token after completing the current OUT transaction.

d) The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.

e) In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the

receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.

f) The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.

g) The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).

h) The core generates the XFRC interrupt as soon as the receive packet status is read.

i) In response to the XFRC interrupt, disable the channel and stop writing the OTG_HS_HCCHAR2 register for further requests. The core writes a channel disable request to the nonperiodic request queue as soon as the OTG_HS_HCCHAR2 register is written.

j) The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.

k) Read and ignore the receive packet status.

l) The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.

m) In response to the CHH interrupt, de-allocate the channel for other transfers.

n) Handling nonACK responses

- **Control transactions in slave mode**

  Setup, Data, and Status stages of a control transfer must be performed as three separate transfers. Setup-, Data- or Status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or Status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in OTG_HS_HCCHAR1 to Control. During the Setup stage, the application is expected to set the PID field in OTG_HS_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation in Slave mode is shown in *Figure 376*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

  – The periodic transmit FIFO can hold one packet (1 KB)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

  a) Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HS_HCCHAR1.

  b) Write the first packet for channel 1. For a high-bandwidth interrupt transfer, the application must write the subsequent packets up to MCNT (maximum number of packets to be transmitted in the next frame times) before switching to another channel.

  c) Along with the last DWORD write of each packet, the OTG_HS host writes an entry to the periodic request queue.

  d) The OTG_HS host attempts to send an OUT token in the next (odd) frame.

  e) The OTG_HS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.

  f) In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 376. Normal interrupt OUT/IN transactions - DMA mode

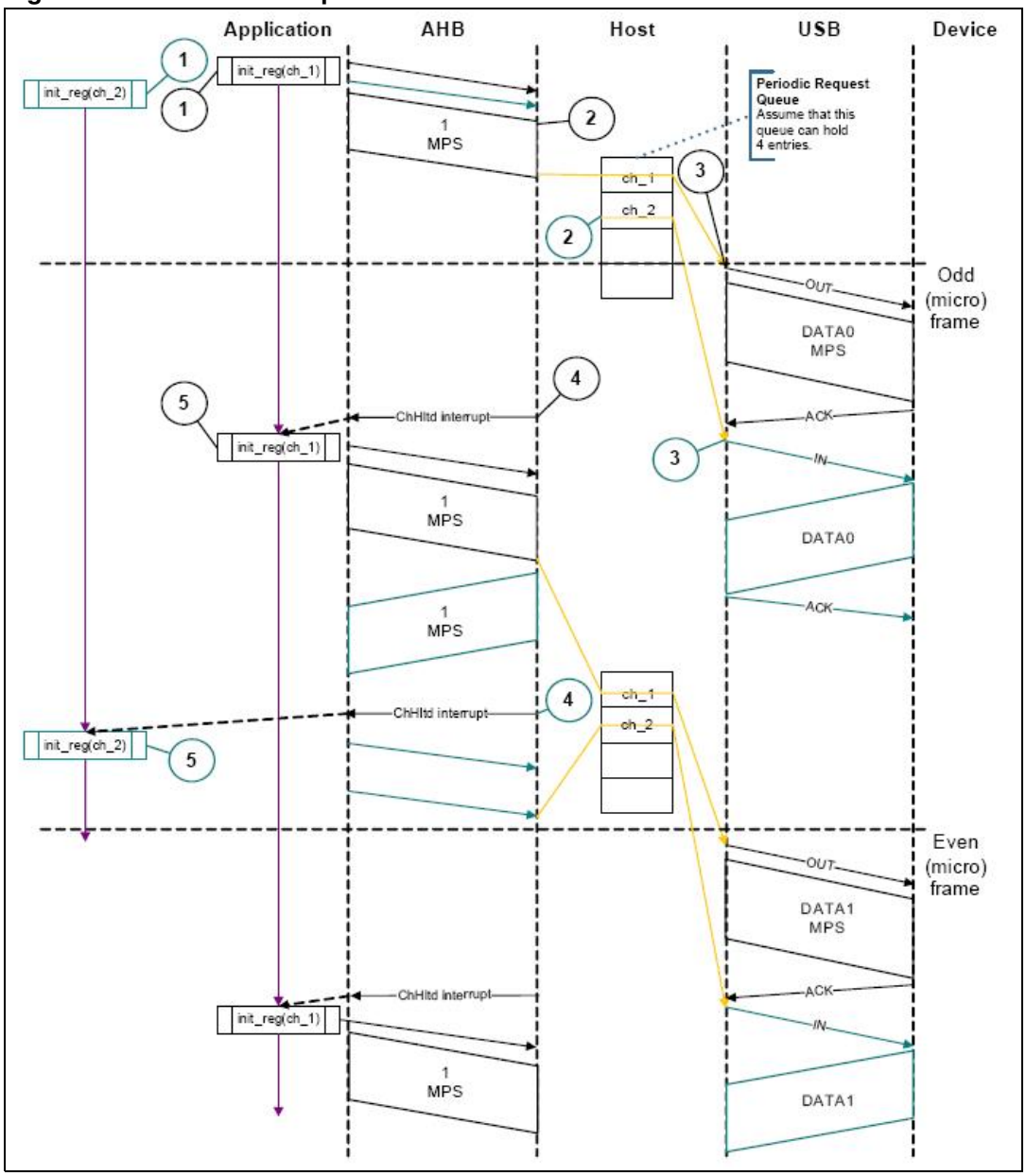
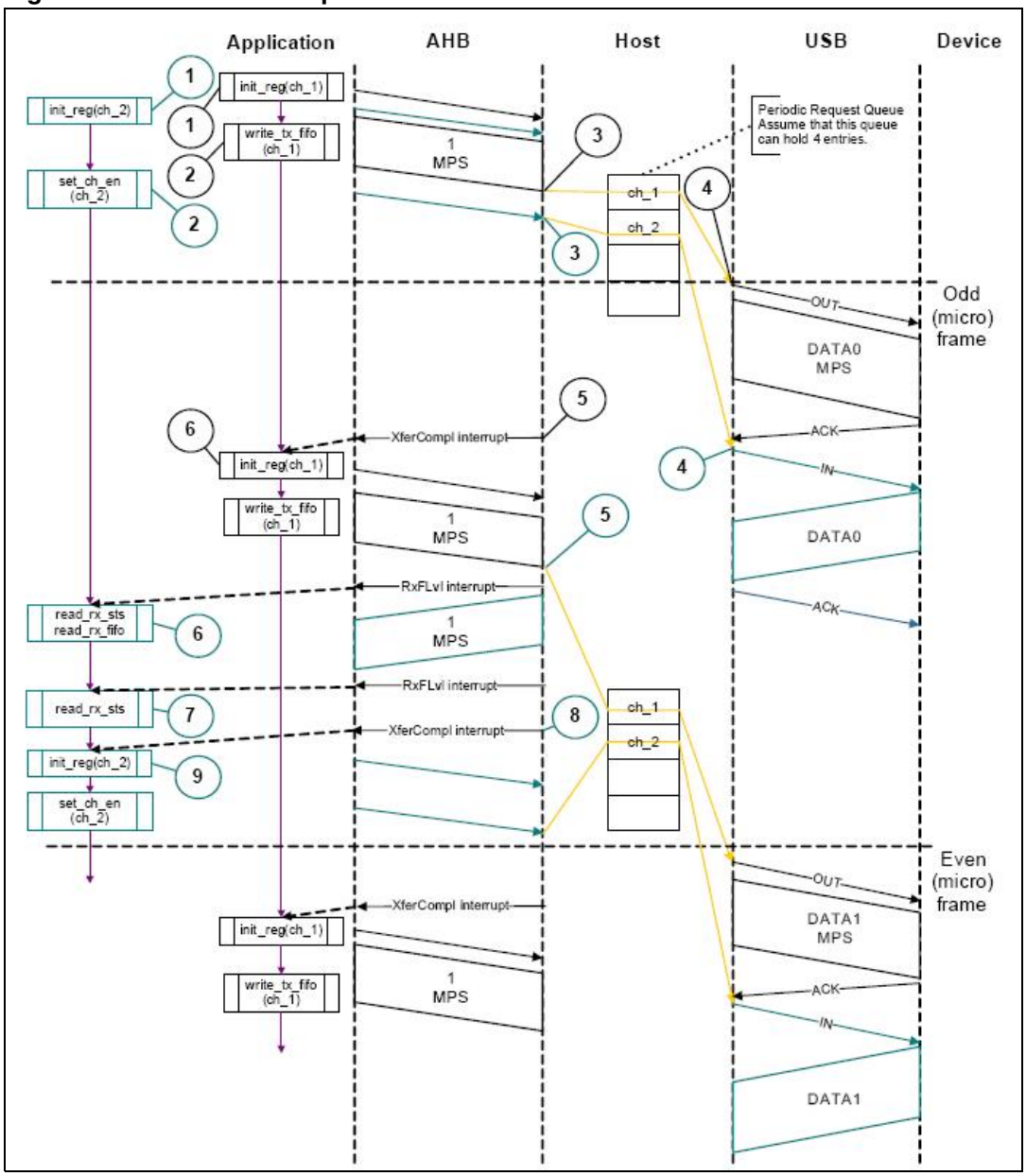


Figure 377. Normal interrupt OUT/IN transactions - Slave mode



- **Interrupt service routine for interrupt OUT/IN transactions**

a) Interrupt OUT

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
```

```
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
        {
        Transfer Done = 1
        }
      }
    else
      if (NAK or TXERR)
        {
        Rewind Buffer Pointers
        Reset Error Count
        Mask ACK
        Unmask CHH
        Disable Channel
        }
      else
        if (CHH)
        {
        Mask CHH
        if (Transfer Done or (Error_count == 3))
          {
          De-allocate Channel
          }
        else
          {
          Re-initialize Channel (in next b_interval - 1 Frame)
          }
        }
      else
        if (ACK)
        {
        Reset Error Count
        Mask ACK
        }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the Request queue up to the count specified in the MCNT field before switching to another channel. The application uses the NPTXFE interrupt in OTG_HS_GINTSTS to find the transmit FIFO space.

b) Interrupt IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
  {
  Reset Error Count
  Mask ACK
  if (OTG_HS_HCTSIZx.PKTCNT == 0)
    {
    De-allocate Channel
    }
  else
    {
```

```
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
```

}

The application is expected to write the requests for the same channel when the Request queue space is available up to the count specified in the MCNT field before switching to another channel (if any).

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1 031 bytes).

  – Periodic request queue depth = 4.

- **Normal interrupt IN operation**

  The sequence of operations is as follows:

  a) Initialize channel 2. The application must set the ODDFRM bit in OTG_HS_HCCHAR2.

  b) Set the CHENA bit in OTG_HS_HCCHAR2 to write an IN request to the periodic request queue. For a high-bandwidth interrupt transfer, the application must write the OTG_HS_HCCHAR2 register MCNT (maximum number of expected packets in the next frame times) before switching to another channel.

  c) The OTG_HS host writes an IN request to the periodic request queue for each OTG_HS_HCCHAR2 register write with the CHENA bit set.

  d) The OTG_HS host attempts to send an IN token in the next (odd) frame.

  e) As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.

  f) In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.

  g) The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).

  h) The core generates an XFRC interrupt as soon as the receive packet status is read.

  i) In response to the XFRC interrupt, read the PKTCNT field in OTG_HS_HCTSIZ2. If the PKTCNT bit in OTG_HS_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer, if any). If PKTCNT bit in

OTG_HS_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HS_HCCHAR2.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation in Slave mode is shown in *Figure 378*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum packet size), starting with an odd frame. (transfer size = 1 024 bytes).

  – The periodic transmit FIFO can hold one packet (1 KB).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

  a) Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HS_HCCHAR1.

  b) Write the first packet for channel 1. For a high-bandwidth isochronous transfer, the application must write the subsequent packets up to MCNT (maximum number of packets to be transmitted in the next frame times before switching to another channel.

  c) Along with the last DWORD write of each packet, the OTG_HS host writes an entry to the periodic request queue.

  d) The OTG_HS host attempts to send the OUT token in the next frame (odd).

  e) The OTG_HS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.

  f) In response to the XFRC interrupt, reinitialize the channel for the next transfer.

  g) Handling nonACK responses

Figure 378. Normal isochronous OUT/IN transactions - DMA mode

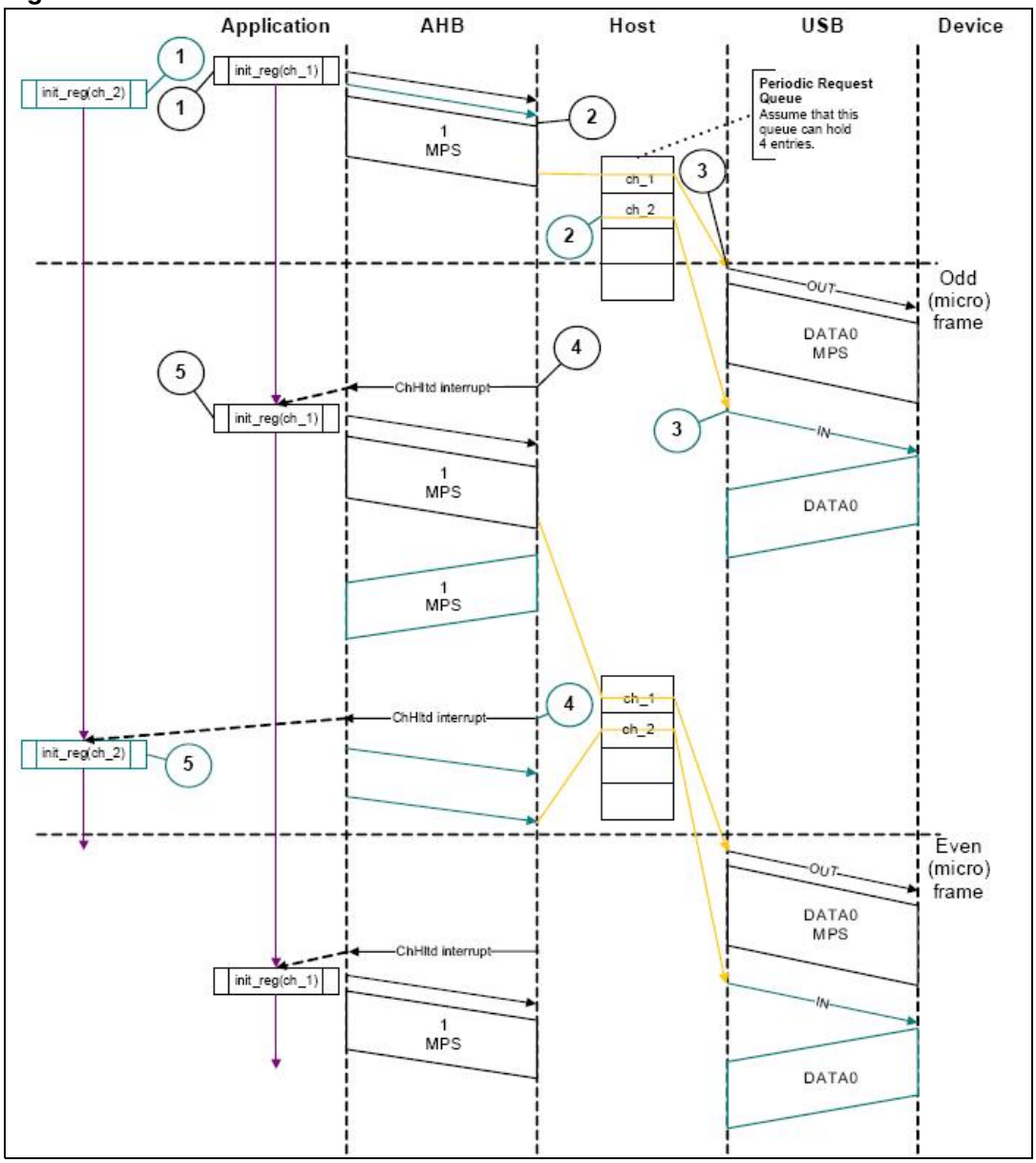
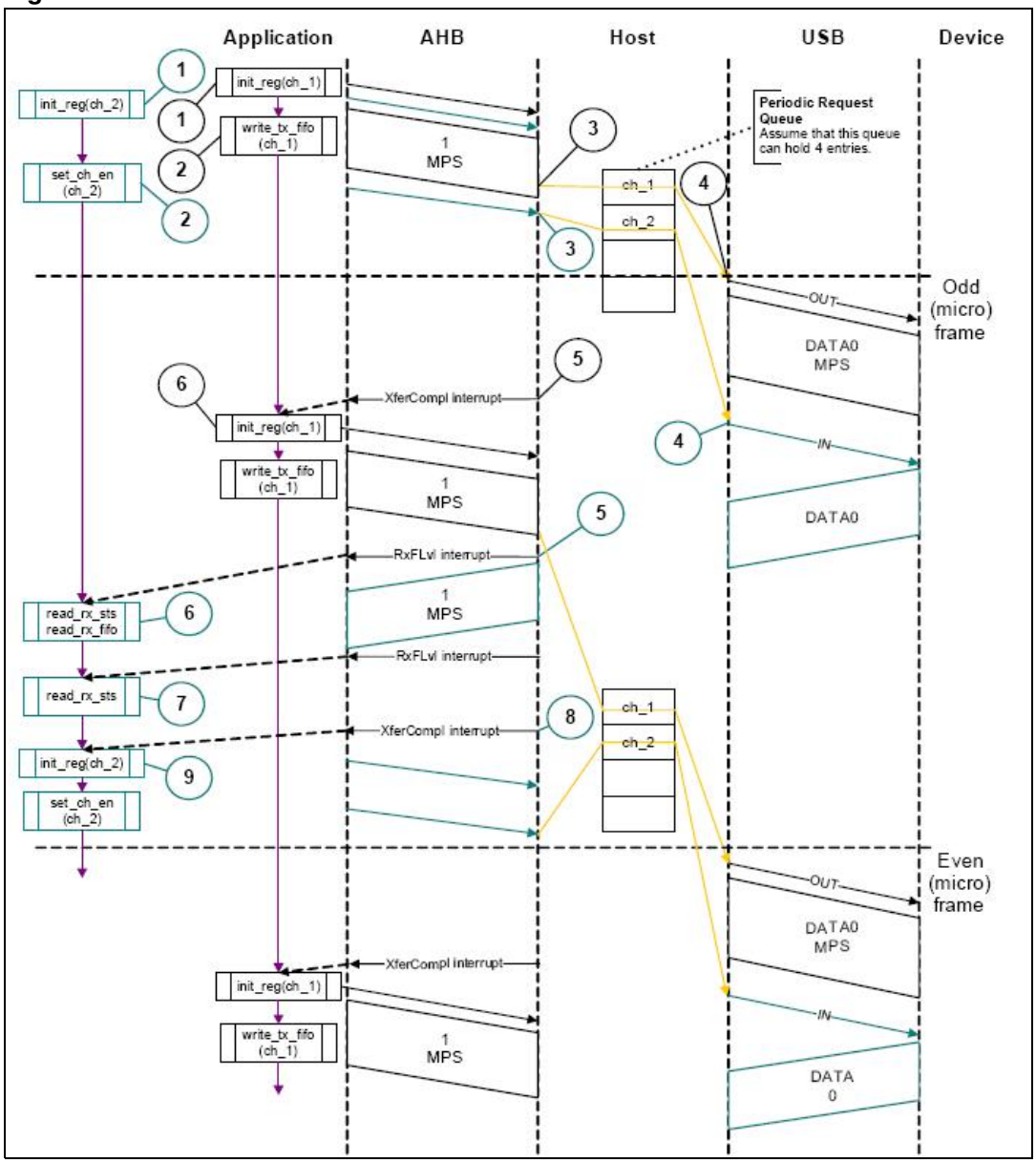


Figure 379. Normal isochronous OUT/IN transactions - Slave mode



- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: Isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
```

```
   else
   if (CHH)
     {
     Mask CHH
     De-allocate Channel
     }
```

Code sample: Isochronous IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HS_HCTSIZx.PKTCNT == 0))
     {
     Reset Error Count
     De-allocate Channel
     }
   else
     {
     Unmask CHH
     Disable Channel
     }
   }
   else
     if (TXERR or BBERR)
       {
       Increment Error Count
       Unmask CHH
       Disable Channel
       }
     else
       if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
           {
           De-allocate Channel
           }
         else
           {
           Re-initialize Channel
           }
         }
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status DWORDs per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

  a) Initialize channel 2. The application must set the ODDFRM bit in OTG_HS_HCCHAR2.

  b) Set the CHENA bit in OTG_HS_HCCHAR2 to write an IN request to the periodic request queue. For a high-bandwidth isochronous transfer, the application must write the OTG_HS_HCCHAR2 register MCNT (maximum number of expected packets in the next frame times) before switching to another channel.

  c) The OTG_HS host writes an IN request to the periodic request queue for each OTG_HS_HCCHAR2 register write with the CHENA bit set.

  d) The OTG_HS host attempts to send an IN token in the next odd frame.

  e) As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.

  f) In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.

  g) The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_HS_GRXSTSR ≠ 0b0010).

  h) The core generates an XFRC interrupt as soon as the receive packet status is read.

  i) In response to the XFRC interrupt, read the PKTCNT field in OTG_HS_HCTSIZ2. If PKTCNT≠ 0 in OTG_HS_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HS_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HS_HCCHAR2.

- **Selecting the queue depth**

  Choose the periodic and nonperiodic request queue depths carefully to match the number of periodic/nonperiodic endpoints accessed.

  The nonperiodic request queue depth affects the performance of nonperiodic transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline nonperiodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

  The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a micro-frame. In Slave mode, however, the application must also take into account the disable entry that must be put into the queue. So, if there are two nonhigh-bandwidth periodic endpoints, the periodic request queue depth must be at least 4. If at least one high-bandwidth endpoint is supported, the queue depth must be

8. If the periodic request queue depth is smaller than the periodic transfers scheduled in a micro-frame, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG_HS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_HS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

  When OTG_HS controller detects a port babble, it flushes the RxFIFO and disables the port. The core then generates a Port disabled interrupt (HPRTINT in OTG_HS_GINTSTS, PENCHNG in OTG_HS_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the Port Disabled interrupt) by checking POCA in OTG_HS_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**

  The sequence of operations is as follows:

  a) Initialize and enable channel 1 as explained in *Section : Channel initialization*.

  b) The HS_OTG host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG_HS host uses the programmed DMA address to fetch the packet.

  c) After fetching the last DWORD of the second (last) packet, the OTG_HS host masks channel 1 internally for further arbitration.

  d) The HS_OTG host generates a CHH interrupt as soon as the last packet is sent.

  e) In response to the CHH interrupt, de-allocate the channel for other transfers.

- **NAK and NYET handling with internal DMA**

  a) The OTG_HS host sends a bulk OUT transaction.

  b) The device responds with NAK or NYET.

  c) If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to

service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.

d) The core automatically issues a ping token.

e) When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.

The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**

  The sequence of operations is as follows:

  a) Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).

  c) The OTG_HS host starts writing the received data to the system memory as soon as the last byte is received with no errors.

  d) When the last packet is received, the OTG_HS host sets an internal flag to remove any extra IN requests from the request queue.

  e) The OTG_HS host flushes the extra requests.

  f) The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.

  g) The OTG_HS host generates the CHH interrupt as soon as the disable request comes to the top of the queue.

  h) In response to the CHH interrupt, de-allocate the channel for other transfers.

- **Interrupt OUT transactions in DMA mode**

  a) Initialize and enable channel x as explained in *Section : Channel initialization*.

  b) The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last DWORD fetch. In high-bandwidth

transfers, the HS_OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

c) The OTG_HS host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.

d) After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.

e) In response to the CHH interrupt, reinitialize the channel for the next transfer.

- **Interrupt IN transactions in DMA mode**

  The sequence of operations (channelx) is as follows:

  a) Initialize and enable channel x as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host writes consecutive writes up to MC times.

  c) The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

  d) As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.

  e) In response to the CHH interrupt, reinitialize the channel for the next transfer.

- **Isochronous OUT transactions in DMA mode**

  a) Initialize and enable channel x as explained in *Section : Channel initialization*.

  b) The OTG_HS host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last DWORD fetch. In high-bandwidth transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

  c) The OTG_HS host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.

  d) After successfully transmitting the packet, the HS_OTG host generates a CHH interrupt.

  e) In response to the CHH interrupt, reinitialize the channel for the next transfer.

- **Isochronous IN transactions in DMA mode**

  The sequence of operations ((channel x) is as follows:

  a) Initialize and enable channel x as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-

bandwidth transfers, the OTG_HS host performs consecutive write operations up to MC times.

c) The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

d) As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.

e) In response to the CHH interrupt, reinitialize the channel for the next transfer.

- **Bulk and control OUT/SETUP split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

  a) Initialize and enable channel x for start split as explained in *Section : Channel initialization*.

  b) The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last DWORD fetch.

  c) After successfully transmitting start split, the OTG_HS host generates the CHH interrupt.

  d) In response to the CHH interrupt, set the COMPLSPLT bit in HCSPLT1 to send the complete split.

  e) After successfully transmitting complete split, the OTG_HS host generates the CHH interrupt.

  f) In response to the CHH interrupt, de-allocate the channel.

- **Bulk/Control IN split transactions in DMA mode**

  The sequence of operations (channel x) is as follows:

  a) Initialize and enable channel x as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG_HS host masks the channel x internally for the arbitration after writing the request.

  c) As soon as the IN token is transmitted, the OTG_HS host generates the CHH interrupt.

  d) In response to the CHH interrupt, set the COMPLSPLT bit in HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.

  e) The OTG_HS host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.

  f) The OTG_HS host starts writing the packet to the system memory after receiving the packet successfully.

  g) As soon as the received packet is written to the system memory, the OTG_HS host generates a CHH interrupt.

  h) In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

  a) Initialize and enable channel 1 for start split as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in HCCHAR1.

  b) The HS_OTG host starts reading the packet.

  c) The HS_OTG host attempts to send the start split transaction.

  d) After successfully transmitting the start split, the OTG_HS host generates the

CHH interrupt.

e) In response to the CHH interrupt, set the COMPLSPLT bit in HCSPLT1 to send the complete split.

f) After successfully completing the complete split transaction, the OTG_HS host generates the CHH interrupt.

g) In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

  a) Initialize and enable channel x for start split as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.

  c) The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.

  d) The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.

  e) In response to the CHH interrupt, set the COMPLSPLT bit in HCSPLT2 to send the complete split.

  f) As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

  g) The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory.

  h) In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**

  The sequence of operations (channel x) is as follows:

  a) Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in HCCHAR1. Program the MPS field.

  b) The HS_OTG host starts reading the packet.

  c) After successfully transmitting the start split (begin), the HS_OTG host generates the CHH interrupt.

  d) In response to the CHH interrupt, reinitialize the registers to send the start split (end).

  e) After successfully transmitting the start split (end), the OTG_HS host generates a CHH interrupt.

  f) In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**

  The sequence of operations (channel x) is as follows:

  a) Initialize and enable channel x for start split as explained in *Section : Channel initialization*.

  b) The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.

  c) The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.

d) The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.

e) In response to the CHH interrupt, set the COMPLSPLT bit in HCSPLT2 to send the complete split.

f) As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

g) The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.

30.13.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   – SNAK = 1 in OTG_HS_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   – INEP0 = 1 in OTG_HS_DAINTMSK (control 0 IN endpoint)
   – OUTEP0 = 1 in OTG_HS_DAINTMSK (control 0 OUT endpoint)
   – STUP = 1 in DOEPMSK
   – XFRC = 1 in DOEPMSK
   – XFRC = 1 in DIEPMSK
   – TOC = 1 in DIEPMSK
3. Set up the Data FIFO RAM for each of the FIFOs
   – Program the OTG_HS_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets).
   – Program the OTG_HS_TX0FSIZ register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   – STUPCNT = 3 in OTG_HS_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. In DMA mode, the DOEPDMA0 register should have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_HS_GINTSTS), read the OTG_HS_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_HS_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. In DMA mode, program the DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.
   – EPENA bit in DOEPCTL0 = 1

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_HS_DCFG register with the device address received in the SetAddress command
1. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_HS_DAINTMSK register.
5. Set up the Data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_HS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_HS_DOEPCTLx register (for OUT or bidirectional endpoints).
   – Maximum packet size
   – USB active endpoint = 1
   – Endpoint start data toggle (for interrupt and bulk endpoints)
   – Endpoint type
   – TxFIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_HS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_HS_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: 1 The application must meet the following conditions to set up the device core to handle traffic: NPTXFEM and RXFLVLM in GINTMSK must be cleared.

30.13.7 Operational model

SETUP and OUT data transfers

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO in Slave mode.

1. On catching an RXFLVL interrupt (OTG_HS_GINTSTS register), the application must read the Receive status pop register (OTG_HS_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_HS_GINTSTS) by writing to RXFLVL = 0 (in GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive Data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive FIFO's packet status readout indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Don't Care (0x0), DPID = Don't Care (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = D0.

These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.

c) Setup stage done pattern:
PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = Don't Care (0b00).
These data indicate that the Setup stage for the specified endpoint has completed and the Data stage has started. After this entry is popped from the receive FIFO, the core asserts a Setup interrupt on the specified control OUT endpoint.

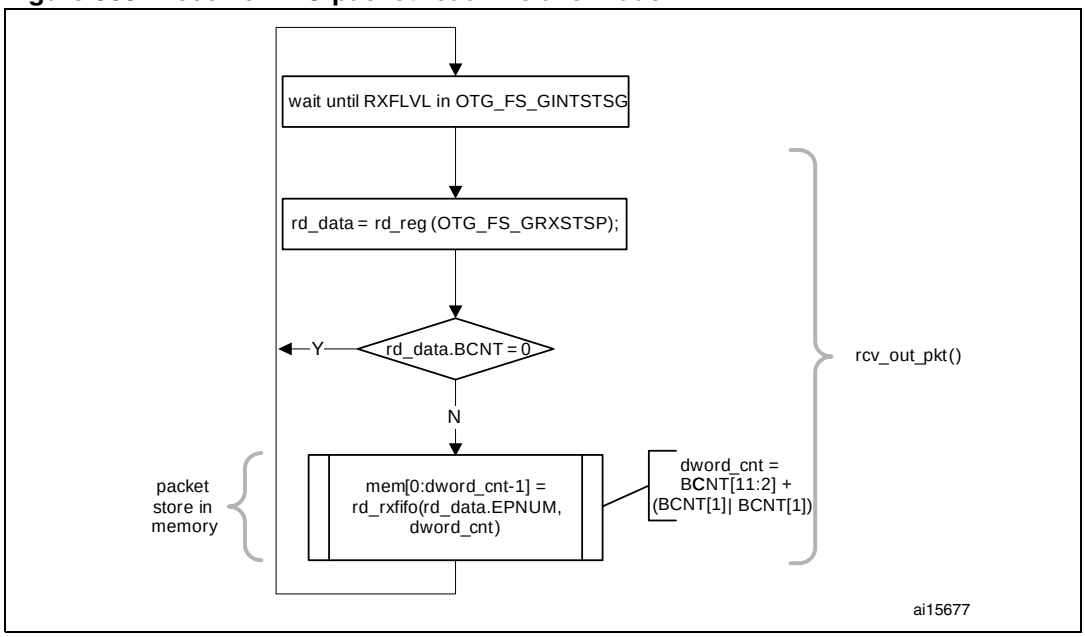
d) Data OUT packet pattern:
PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.

e) Data transfer completed pattern:
PKTSTS = Data OUT Transfer Done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = Don't Care (0b00).
These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a Transfer Completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_HS_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_HS_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 380 provides a flowchart of the above procedure.

Figure 380. Receive FIFO packet read in slave mode



- **SETUP transactions**

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_HS_DOEPTSIZx) in a control OUT endpoint must be programmed to a nonzero value. When the application programs the STUPCNT field to a nonzero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_HS_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the Setup stage of a control transfer.
   – STUPCNT = 3 in OTG_HS_DOEPTSIZx
2. The application must always allocate some extra space in the Receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 DWORDs. Three DWORDs are required for the first SETUP packet, 1 DWORD is required for the Setup stage done DWORD and 6 DWORDs are required to store two extra SETUP packets among all control endpoints.
   – 3 DWORDs per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup packet pattern). The core reserves this space in the receive data.
   – FIFO to write SETUP data only, and never uses this space for data packets.
3. The application must read the 2 DWORDs of the SETUP packet from the receive FIFO.
4. The application must read and discard the Setup stage done DWORD from the receive FIFO.

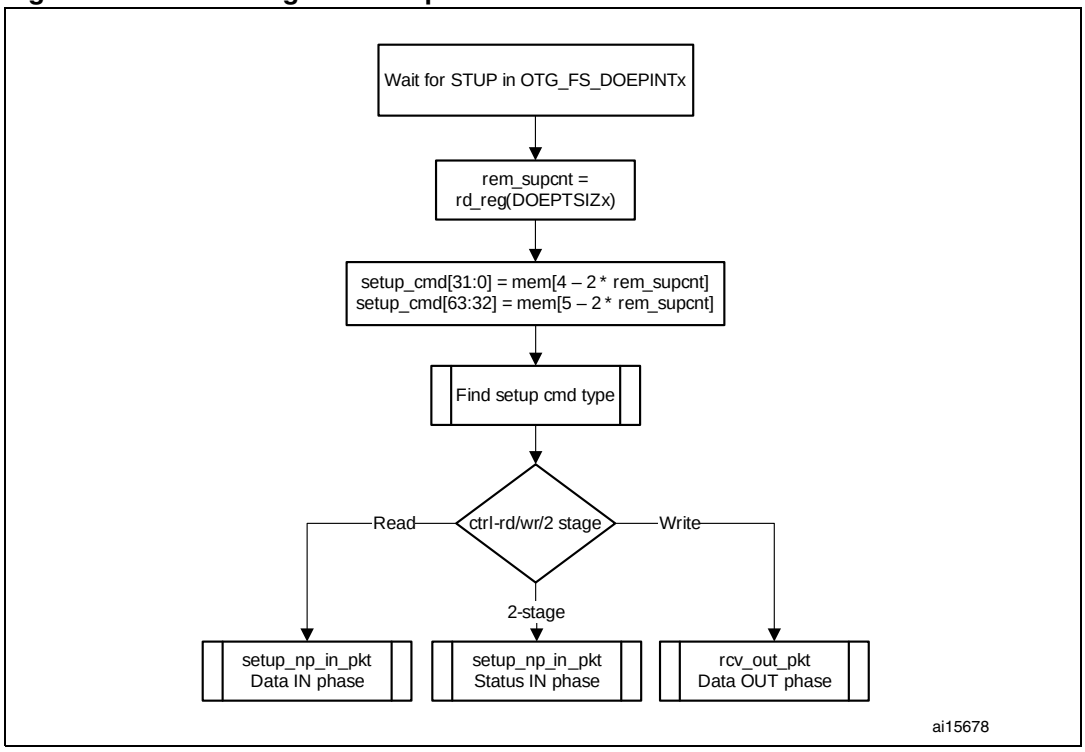
- **Internal data flow**

5. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
6. For every SETUP packet received on the USB, 3 DWORDs of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first DWORD contains control information used internally by the core
   – The second DWORD contains the first 4 bytes of the SETUP command
   – The third DWORD contains the last 4 bytes of the SETUP command
7. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup stage done DWORD) to the receive FIFO, indicating the completion of the Setup stage.
8. On the AHB side, SETUP packets are emptied by the application.
9. When the application pops the Setup stage done DWORD from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_HS_DOEPINTx), indicating it can process the received SETUP packet.
   – The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_HS_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_HS_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_HS_DOEPINTx) marks a successful completion of the SETUP Data Transfer.
   – On this interrupt, the application must read the OTG_HS_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 381. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_HS controller generates an interrupt (B2BSTUP in OTG_HS_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_HS_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, nonisochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.

3. When the application pops the Global OUT NAK pattern DWORD from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_HS_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_HS_DCTL.

Application programming sequence

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_HS_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_HS_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_HS_DCTL and before the core asserts the GONAKEFF interrupt (OTG_HS_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GINAKEFFM bit in GINTMSK.
   – GINAKEFFM = 0 in GINTMSK
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_HS_DCTL. This also clears the GONAKEFF interrupt (OTG_HS_GINTSTS).
   – OTG_HS_DCTL = 1 in CGONAK
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GINAKEFFM = 1 in GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_HS_DCTL
2. Wait for the GONAKEFF interrupt (OTG_HS_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_HS_DOEPCTLx
   – SNAK = 1 in OTG_HS_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_HS_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_HS_DOEPCTLx
   – EPENA = 0 in OTG_HS_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other nondisabled OUT endpoints.
   – SGONAK = 0 in OTG_HS_DCTL

- **Generic nonisochronous OUT data transfers**

This section describes a regular nonisochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary.
   - transfer size[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   - packet count[EPNUM] = n
   - $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   - Payload size in memory = application programmed initial transfer size – core updated final transfer size
   - Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards nonisochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or nonisochronous data packets are ignored and not written to the receive FIFO. Additionally, nonisochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or nonisochronous data packets are ignored and not written to the receive FIFO, and nonisochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.

6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   – The transfer size is 0 and the packet count is 0
   – The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_HS_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_HS_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_HS_DOEPCTLx
   – CNAK = 1 in OTG_HS_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_HS_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_HS_DOEPINTx) marks a successful completion of the nonisochronous OUT data transfer.
5. Read the OTG_HS_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for nonisochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_HS_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_HS_GINTSTS) and before the SOF (OTG_HS_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for nonisochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_HS_DOEPCTLx) = SOFFN[0] (in OTG_HS_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in

OTG_HS_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_HS_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_HS_DOEPCTLx register with the endpoint characteristics and set the Endpoint Enable, ClearNAK, and Even/Odd frame bits.
   - EPENA = 1
   - CNAK = 1
   - EONUM = (0: Even/1: Odd)
3. In Slave mode, wait for the RXFLVL interrupt (in OTG_HS_GINTSTS) and empty the data packets from the receive FIFO
   - This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_HS_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the IISOOXFRM interrupt in OTG_HS_GINTSTS.
6. Read the OTG_HS_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   - RXDPID = D0 (in OTG_HS_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   - RXDPID = D1 (in OTG_HS_DOEPTSIZx) and the number of USB packets in which this payload was received = 2
   - RXDPID = D2 (in OTG_HS_DOEPTSIZx) and the number of USB packets in which this payload was received = 3

     The number of USB packets in which this payload was received = Application programmed initial packet count – Core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_HS_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application could fail to detect the XFRC interrupt (OTG_HS_DOEPINTx) under the following circumstances:
   - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   - When the isochronous OUT data packet is received with CRC errors
   - When the isochronous OUT token received by the core is corrupted
   - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete Isochronous OUT data interrupt

(IISOOXFRM in OTG_HS_GINTSTS), indicating that an XFRC interrupt (in OTG_HS_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the IISOOXFRM interrupt (OTG_HS_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_HS_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an IISOOXFRM interrupt (in OTG_HS_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_HS_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current micro-frame. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_HS_DOEPCTLx) = SOFFN[0] (in OTG_HS_DSTS)
   – EPENA = 1 (in OTG_HS_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_HS_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_HS_DOEPCTLx.
6. Wait for the EPDIS interrupt (in OTG_HS_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a nonisochronous OUT endpoint**

This section describes how the application can stall a nonisochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_HS_DOEPCTL, set STALL = 1 (in OTG_HS_DOEPCTL).

   The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_HS_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

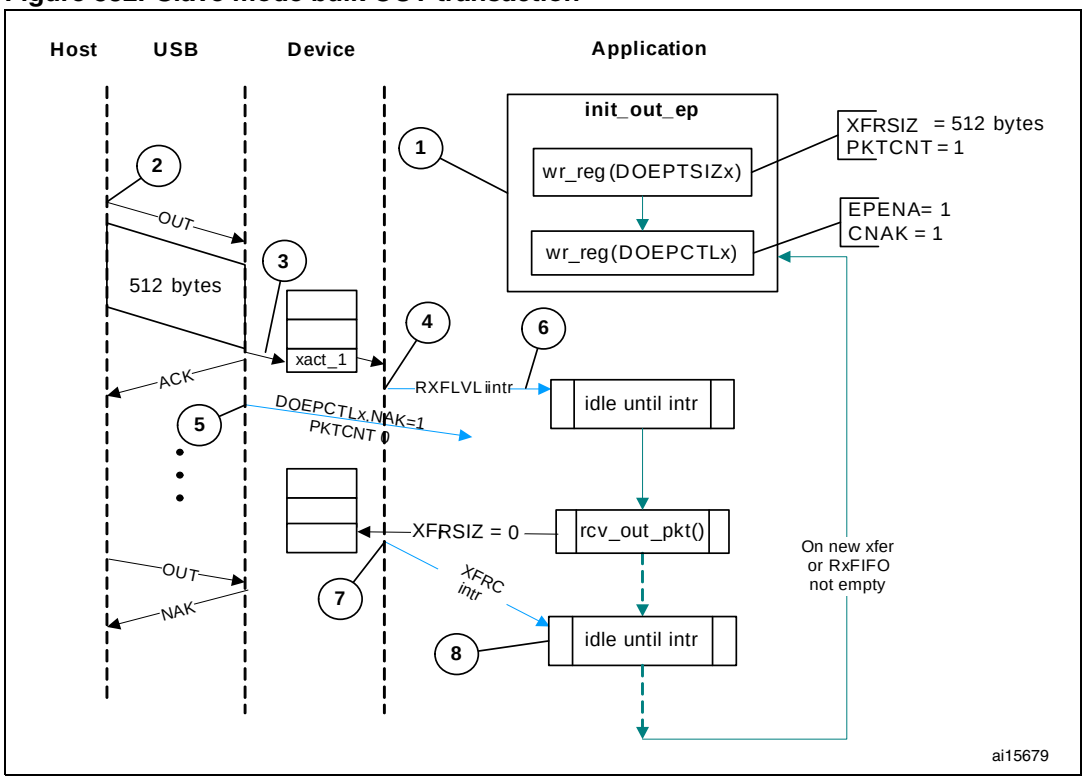
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Slave mode bulk OUT transaction

Figure 382 depicts the reception of a single Bulk OUT Data packet from the USB to the AHB and describes the events involved in the process.

Figure 382. Slave mode bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_HS_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_HS_DOEPTSIZx register.

1. Host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the RxFIFO because space is available there.
3. After writing the complete packet in the RxFIFO, the core then asserts the RXFLVL interrupt (in OTG_HS_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the RxFIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_HS_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_HS_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO in Slave mode when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_HS_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - In interrupt mode, the application waits for the TXFE interrupt (in OTG_HS_DIEPINTx) and then reads the OTG_HS_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   - To write a single nonzero length data packet, there must be space to write the entire packet in the data FIFO.
   - To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_HS_DIEPCTLx register to avoid modifying the contents of the register, except for setting the Endpoint Enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one micro-frame. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Nonisochronous IN tokens receive a NAK handshake reply
   - Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_HS_DIEPINTx in response to the SNAK bit in OTG_HS_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_HS_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_HS_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_HS_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in DIEPMSK.
   – INEPNEM = 0 in DIEPMSK
5. To exit Endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_HS_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_HS_DIEPINTx).
   – CNAK = 1 in OTG_HS_DIEPCTLx
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_HS_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_HS_DIEPINTx.
4. Set the following bits in the OTG_HS_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_HS_DIEPCTLx
   – SNAK = 1 in OTG_HS_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_HS_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_HS_DIEPCTLx
   – EPDIS = 0 in OTG_HS_DIEPCTLx
6. The application must read the OTG_HS_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the Endpoint transmit FIFO, by setting the following fields in the OTG_HS_GRSTCTL register:
   – TXFNUM (in OTG_HS_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_HS_GRSTCTL) = 1

The application must poll the OTG_HS_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Generic nonperiodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:

     Transfer size[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp

     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = x
   - To transmit a single zero-length data packet:

     Transfer size[EPNUM] = 0

     Packet count[EPNUM] = 1
   - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.

     First transfer: transfer size[EPNUM] = $x \times$ MPSIZ[epnum]; packet count = n;

     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the Transfer size register. At the end of the IN transfer, the application must read the Transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count – Core updated final packet count) × MPSIZ[EPNUM]
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every nonisochronous IN data packet transmitted with an ACK

handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.

5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when TxFIFO is empty” (ITTXFE) Interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for nonisochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_HS_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_HS_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (Endpoint Enable) bits.
3. When transmitting nonzero length data packet, the application must poll the OTG_HS_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_HS_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic nonperiodic IN data transfers on page 1210* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
(where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = *x* + 1
Otherwise, packet count[EPNUM] = *x*;

MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_HS_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_HS_DIEPTSIZx, MPSIZ is in OTG_HS_DIEPCTLx, PKTCNT is in OTG_HS_DIEPTSIZx and XFERSIZ is in OTG_HS_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 DWORD of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints
4. For a high-bandwidth IN endpoint with three packets in a frame, the application can program the endpoint FIFO size to be 2 × max_pkt_size and have the third packet loaded in after the first packet has been transmitted on the USB.

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the frame is not present in the FIFO, then the core generates an IN token received when TxFIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:

- For isochronous endpoints, when a zero- or nonzero-length data packet is transmitted
- For interrupt endpoints, when an ACK handshake is transmitted
- When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.

6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_HS_DCFG), when the core finds nonempty any of the isochronous IN endpoint FIFOs scheduled for the current frame nonempty, the core generates an IISOIXFR interrupt in OTG_HS_GINTSTS.

Application programming sequence:

1. Program the OTG_HS_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_HS_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_HS_DIEPINTx) with no ITTXFE interrupt in OTG_HS_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_HS_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_HS_DIEPINTx), with or without the ITTXFE interrupt (in OTG_HS_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_HS_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_HS_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_HS_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when TxFIFO empty interrupt in OTG_HS_DIEPINTx. The application can ignore this interrupt, as it eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_HS_GINTSTS) at the end of periodic frame.

      The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the Endpoint Disable interrupt for the endpoint.

Application programming sequence

1. The application can ignore the IN token received when TxFIFO empty interrupt in OTG_HS_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_HS_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_HS_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the Endpoint Control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_HS_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_HS_DIEPCTLx
   – EPDIS = 1 in OTG_HS_DIEPCTLx
6. The assertion of the Endpoint Disabled interrupt in OTG_HS_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next micro-frame. To flush the data, the application must use the OTG_HS_GRSTCTL register.

- **Stalling nonisochronous IN endpoints**

This section describes how the application can stall a nonisochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_HS_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_HS_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the Endpoint Disabled interrupt (in OTG_HS_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the nonperiodic or periodic transmit FIFO, depending on the endpoint type. In case of a nonperiodic endpoint, the application must re-enable the other nonperiodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_HS_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_HS_DIEPINTx and the OTEPDIS interrupt in OTG_HS_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

30.13.8 Worst case response time

When the OTG_HS controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_HS_GUSBCFG

The value in TRDT (OTG_HS_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_HS_GUSBCFG).

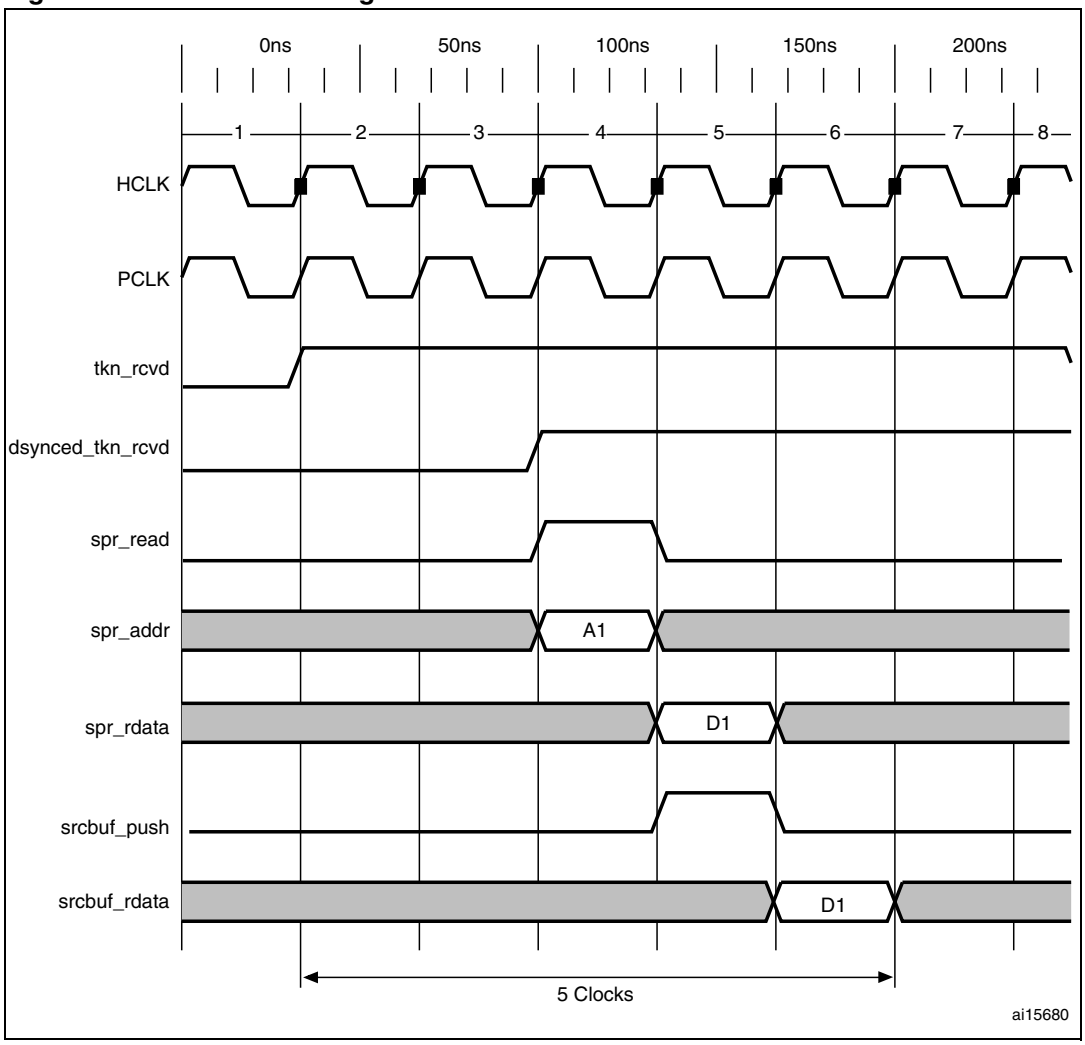
Figure 383 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

The application can use the following formula to calculate the value of TRDT:

4 × AHB clock + 1 PHY clock = (2 clock sync + 1 clock memory address + 1 clock memory data from sync RAM) + (1 PHY clock (next PHY clock MAC can sample the 2 clock FIFO outputs)

Figure 383. TRDT max timing case



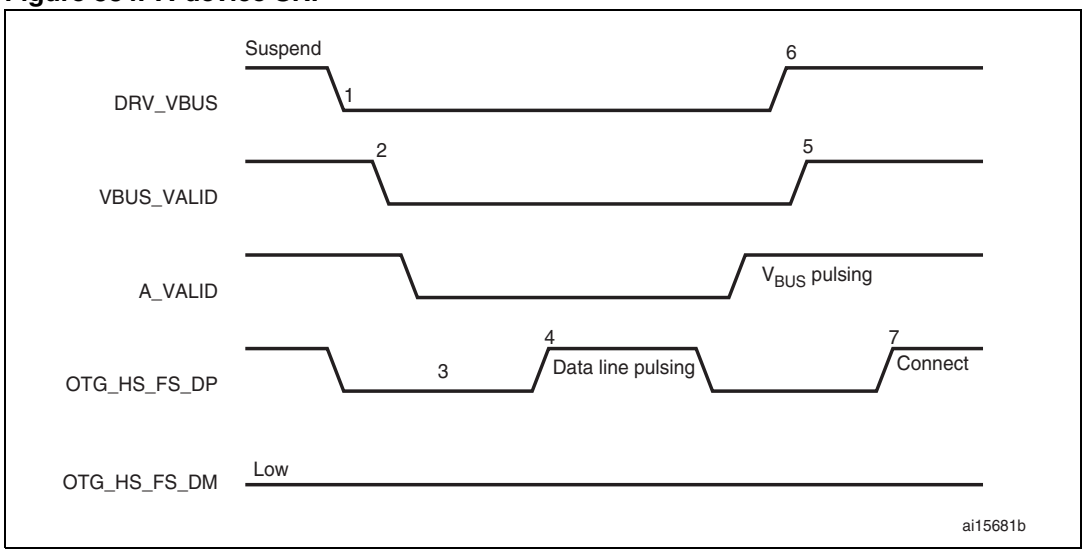
30.13.9 OTG programming model

The OTG_HS controller is an OTG device supporting HNP and SRP. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device. In host mode, the OTG_HS controller turns off V_{BUS} to conserve power. SRP is a method by which the B-device signals the A-device to turn on V_{BUS} power. A device must perform both data-line pulsing and V_{BUS} pulsing, but a host can detect either data-line pulsing or V_{BUS} pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the Core USB configuration register. This enables the OTG_HS controller to detect SRP as an A-device.

Figure 384. A-device SRP



1. DRV_VBUS = V_{BUS} drive signal to the PHY
   VBUS_VALID = V_{BUS} valid signal from PHY
   A_VALID = A-device V_{BUS} level signal to PHY
   DP = Data plus line
   DM = Data minus line

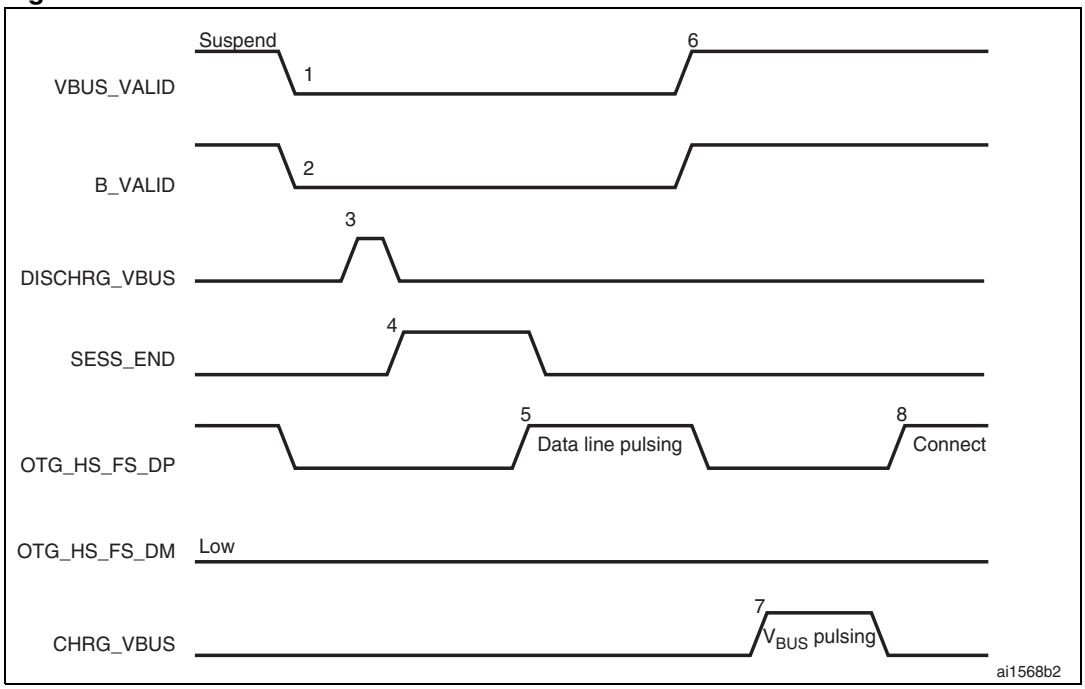
1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when V_{BUS} power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG_HS controller detects data-line pulsing.
5. The device drives V_{BUS} above the A-device session valid (2.0 V minimum) for V_{BUS} pulsing.

   The OTG_HS controller interrupts the application on detecting SRP. The Session request detected bit is set in Global interrupt status register (SRQINT set in OTG_HS_GINTSTS).
6. The application must service the Session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

B-device session request protocol

The application must set the SRP-capable bit in the Core USB configuration register. This enables the OTG_HS controller to initiate SRP as a B-device. SRP is a means by which the OTG_HS controller can request a new session from the host.

Figure 385. B-device SRP



1. VBUS_VALID = V_{BUS} valid signal from PHY
   B_VALID = B-device valid session to PHY
   DISCHRG_VBUS = discharge signal to PHY
   SESS_END = session end signal to PHY
   CHRG_VBUS = charge V_{BUS} signal to PHY
   DP = Data plus line
   DM = Data minus line

1. To save power, the host suspends and turns off port power when the bus is idle.

   The OTG_HS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the Core interrupt register.

   The OTG_HS controller informs the PHY to discharge V_{BUS}.

2. The PHY indicates the session's end to the device. This is the initial condition for SRP. The OTG_HS controller requires 2 ms of SE0 before initiating SRP.

   For a USB 1.1 full-speed serial transceiver, the application must wait until V_{BUS} discharges to 0.2 V after BSVLD (in OTG_HS_GOTGCTL) is deasserted. This

discharge time can be obtained from the transceiver vendor and varies from one transceiver to another.

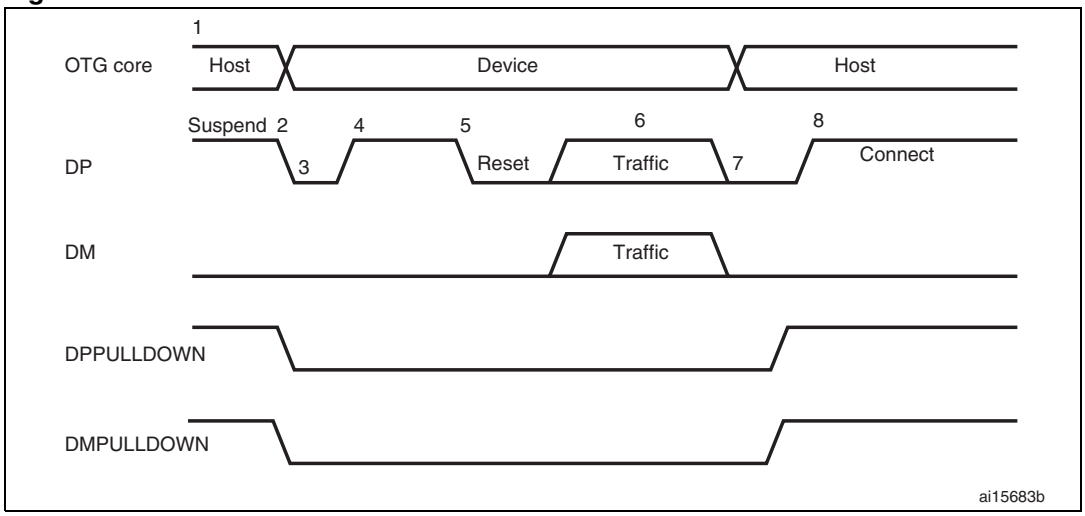
3. The USB OTG core informs the PHY to speed up V_{BUS} discharge.
4. The application initiates SRP by writing the session request bit in the OTG Control and status register. The OTG_HS controller perform data-line pulsing followed by V_{BUS} pulsing.
5. The host detects SRP from either the data-line or V_{BUS} pulsing, and turns on V_{BUS}. The PHY indicates V_{BUS} power-on to the device.
6. The OTG_HS controller performs V_{BUS} pulsing.

   The host starts a new session by turning on V_{BUS}, indicating SRP success. The OTG_HS controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
7. When the USB is powered, the OTG_HS controller connects, completing the SRP process.

A-device host negotiation protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_HS controller to perform HNP as an A-device.

Figure 386. A-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. The OTG_HS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP Enable bit in the OTG Control

and status register to indicate to the OTG_HS controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the Port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The OTG_HS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

   The OTG_HS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_HS_DP pull-up resistor to indicate a connect for B-device.

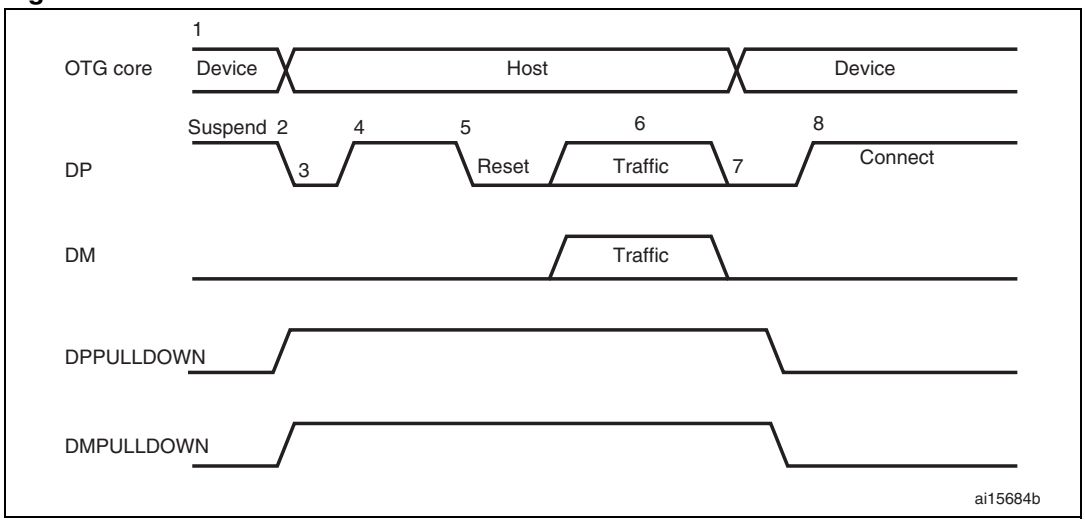
   The application must read the current mode bit in the OTG Control and status register to determine peripheral mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_HS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The OTG_HS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB Suspend bit in the Core interrupt register.
6. In Negotiated mode, the OTG_HS controller detects the suspend, disconnects, and switches back to the host role. The OTG_HS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_HS controller sets the Connector ID status change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the OTG_HS controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

B-device host negotiation protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_HS controller to perform HNP as a B-device.

Figure 387. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY. DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG_HS controller's ACK response indicates that it supports HNP. The application must set the Device HNP enable bit in the OTG Control and status register to indicate HNP support.

   The application sets the HNP request bit in the OTG Control and status register to indicate to the OTG_HS controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the Port suspend bit in the host port control and status register.

   The OTG_HS controller sets the Early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_HS controller sets the USB suspend bit in the Core interrupt register.

   The OTG_HS controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG_HS controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

   The A-device responds by activating its OTG_HS_DP pull-up resistor within 3 ms of detecting SE0. The OTG_HS controller detects this as a connect.

   The OTG_HS controller sets the host negotiation success status change interrupt in the OTG Interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG Control and status register to determine

host negotiation success. The application must read the current Mode bit in the Core interrupt register (OTG_HS_GINTSTS) to determine host mode operation.

3. The application sets the reset bit (PRST in OTG_HS_HPRT) and the OTG_HS controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG_HS controller continues the host role of initiating traffic, and when done, suspends the bus by writing the Port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG_HS controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the Core interrupt (OTG_HS_GINTSTS) register to determine the host mode operation.
7. The OTG_HS controller connects, completing the HNP process.

31 Flexible static memory controller (FSMC)

31.1 FSMC main features

The FSMC block is able to interface with synchronous and asynchronous memories and 16-bit PC memory cards. Its main purpose is to:

- Translate the AHB transactions into the appropriate external device protocol
- Meet the access timing requirements of the external devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FSMC performs only one access at a time to an external device.

The FSMC has the following main features:

- Interfaces with static memory-mapped devices including:
  - Static random access memory (SRAM)
  - Read-only memory (ROM)
  - NOR Flash memory/OneNAND Flash memory
  - PSRAM (4 memory banks)
- Two banks of NAND Flash with ECC hardware that checks up to 8 Kbytes of data
- 16-bit PC Card compatible devices
- Supports burst mode access to synchronous devices (NOR Flash and PSRAM)
- 8- or 16-bit wide databus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices, in particular:
  - Programmable wait states (up to 15)
  - Programmable bus turnaround cycles (up to 15)
  - Programmable output enable and write enable delays (up to 15)
  - Independent read and write timings and protocol, so as to support the widest variety of memories and timings
- Write enable and byte lane select outputs for use with PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A Write FIFO, 2 words long, each word is 32 bits wide, only stores data and not the address. Therefore, this FIFO only buffers AHB write burst transactions. This makes it possible to write to slow memories and free the AHB quickly for other operations. Only one burst at a time is buffered: if a new AHB burst or single transaction occurs while an operation is in progress, the FIFO is drained. The FSMC will insert wait states until the current memory access is complete).
- External asynchronous wait control

The FSMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up. However, it is possible to change the settings at any time.

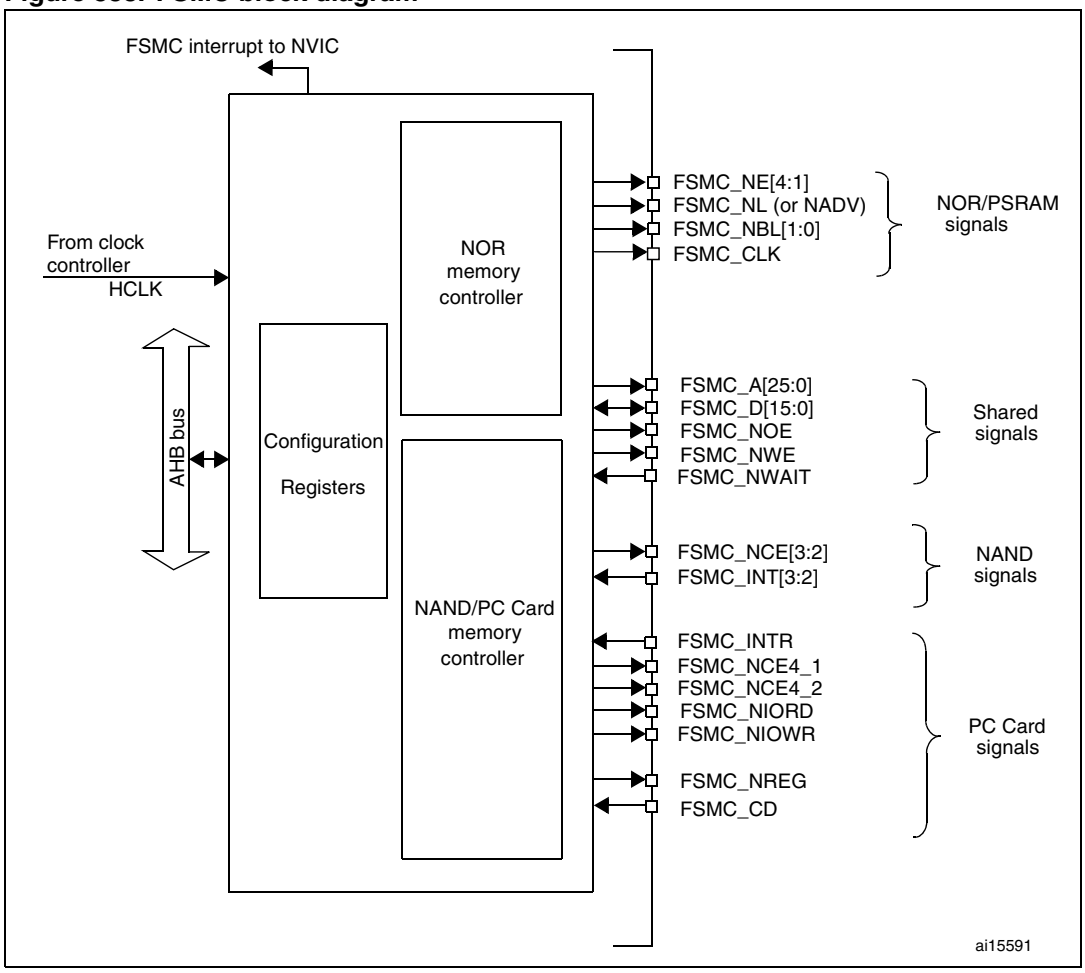
31.2 Block diagram

The FSMC consists of four main blocks:

- The AHB interface (including the FSMC configuration registers)
- The NOR Flash/PSRAM controller
- The NAND Flash/PC Card controller
- The external device interface

The block diagram is shown in *Figure 388*.

Figure 388. FSMC block diagram



31.3 AHB interface

The AHB slave interface enables internal CPUs and other bus master peripherals to access the external static memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16 or 8 bits wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The Chip Select toggles for each access.

The FSMC generates an AHB error in the following conditions:

- When reading or writing to an FSMC bank which is not enabled
- When reading or writing to the NOR Flash bank while the FACCEN bit is reset in the FSMC_BCRx register.
- When reading or writing to the PC Card banks while the input pin FSMC_CD (Card Presence Detection) is low.

The effect of this AHB error depends on the AHB master which has attempted the R/W access:

- If it is the Cortex™-M4F CPU, a hard fault interrupt is generated
- If is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FSMC.

31.3.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal
  There is no issue in this case.
- AHB transaction size is greater than the memory size
  In this case, the FSMC splits the AHB transaction into smaller consecutive memory accesses in order to meet the external data width.
- AHB transaction size is smaller than the memory size
  Asynchronous transfers may or not be consistent depending on the type of external device.
  - Asynchronous accesses to devices that have the byte select feature (SRAM, ROM, PSRAM).

    a) FSMC allows write transactions accessing the right data through its byte lanes NBL[1:0]

    b) Read transactions are allowed (the controller reads the entire memory word and uses the needed byte only). The NBL[1:0] are always kept low during read transactions.
  - Asynchronous accesses to devices that do not have the byte select feature (NOR and NAND Flash 16-bit).
    This situation occurs when a byte access is requested to a 16-bit wide Flash memory. Clearly, the device cannot be accessed in byte mode (only 16-bit words can be read from/written to the Flash memory) therefore:

    a) Write transactions are not allowed

    b) Read transactions are allowed (the controller reads the entire 16-bit memory word and uses the needed byte only).

Configuration registers

The FSMC can be configured using a register set. See *Section 31.5.6*, for a detailed description of the NOR Flash/PSRAM controller registers. See *Section 31.6.8*, for a detailed description of the NAND Flash/PC Card registers.

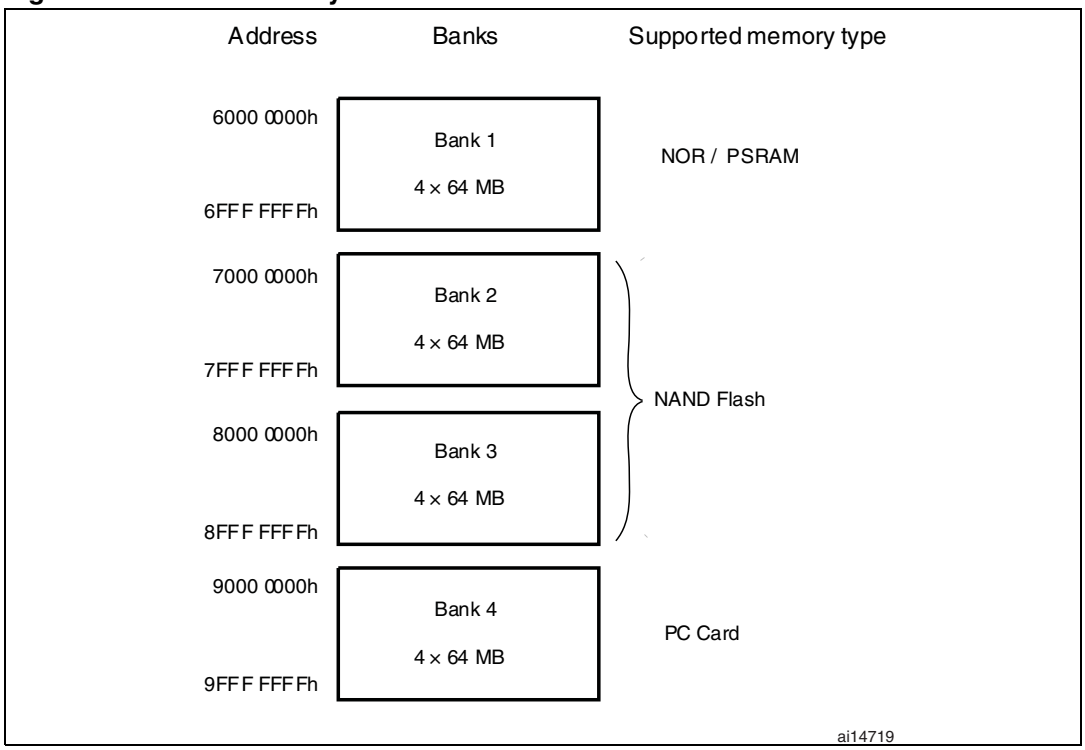
31.4 External device address mapping

From the FSMC point of view, the external memory is divided into 4 fixed-size banks of 256 Mbytes each (Refer to *Figure 389*):

- Bank 1 used to address up to 4 NOR Flash or PSRAM memory devices. This bank is split into 4 NOR/PSRAM regions with 4 dedicated Chip Select.
- Banks 2 and 3 used to address NAND Flash devices (1 device per bank)
- Bank 4 used to address a PC Card device

For each bank the type of memory to be used is user-defined in the Configuration register.

Figure 389. FSMC memory banks



31.4.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 163*.

Table 163. NOR/PSRAM bank selection

HADDR[27:26](1)	Selected bank
00	Bank 1 NOR/PSRAM 1
01	Bank 1 NOR/PSRAM 2

Table 163. NOR/PSRAM bank selection (continued)

HADDR[27:26](1)	Selected bank
10	Bank 1 NOR/PSRAM 3
11	Bank 1 NOR/PSRAM 4

1. HADDR are internal AHB address lines that are translated to external memory.

HADDR[25:0] contain the external memory address. Since HADDR is a byte address whereas the memory is addressed in words, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 164. External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FSMC will internally use HADDR[25:1] to generate the address for external memory FSMC_A[24:0].
Whatever the external memory width (16-bit or 8-bit), FSMC_A[0] should be connected to external memory address A[0].

Wrap support for NOR Flash/PSRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in linear burst mode of undefined length.

31.4.2 NAND/PC Card address mapping

In this case, three banks are available, each of them divided into memory spaces as indicated in *Table 165*.

Table 165. Memory mapping and timing registers

Start address	End address	FSMC Bank	Memory space	Timing register
0x9C00 0000	0x9FFF FFFF	Bank 4 - PC card	I/O	FSMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FSMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FSMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FSMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FSMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Bank 2- NAND Flash	Attribute	FSMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FSMC_PMEM2 (0x68)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 166* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 166. NAND bank selections

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

The application software uses the 3 sections to access the NAND Flash memory:

- **To send a command to NAND Flash memory**: the software must write the command value to any memory location in the command section.
- **To specify the NAND Flash address that must be read or written**: the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive writes to the address section are needed to specify the full address.
- **To read or write data**: the software reads or writes the data value from or to any memory location in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

31.5 NOR Flash/PSRAM controller

The FSMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8-bit
  - 16-bit
  - 32-bit
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode
  - Multiplexed or nonmultiplexed
- NOR Flash
  - Asynchronous mode or burst mode
  - Multiplexed or nonmultiplexed

The FSMC outputs a unique chip select signal NE[4:1] per bank. All the other signals (addresses, data and control) are shared.

For synchronous accesses, the FSMC issues the clock (CLK) to the selected external device. This clock is a submultiple of the HCLK clock. The size of each bank is fixed and equal to 64 Mbytes.

Each bank is configured by means of dedicated registers (see *Section 31.5.6*).

The programmable memory parameters include access timings (see *Table 167*) and support for wait management (for PSRAM and NOR Flash accessed in burst mode).

Table 167. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	AHB clock cycle (HCLK)	0	15
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	1	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

31.5.1 External memory interface signals

Table 168, *Table 169* and *Table 170* list the signals that are typically used to interface NOR Flash, SRAM and PSRAM.

Note: Prefix "N". specifies the associated signal as active low.

NOR Flash, nonmultiplexed I/Os

Table 168. Nonmultipled I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

NOR Flash, multiplexed I/Os

Table 169. Multiplexed I/O NOR Flash

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR Flash devices)
NWAIT	I	NOR Flash wait input signal to the FSMC

NOR-Flash memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM/SRAM, nonmultiplexed I/Os

Table 170. Nonmultiplexed I/Os PSRAM/SRAM

FSMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous burst)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

PSRAM, multiplexed I/Os

Table 171. Multiplexed I/O PSRAM

FSMC signal name	I/O	Function
CLK	O	Clock (for synchronous burst)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FSMC
NBL[1]	O	Upper byte enable (memory signal name: NUB)
NBL[0]	O	Lowed byte enable (memory signal name: NLB)

PSRAM memories are addressed in 16-bit words. The maximum capacity is 512 Mbit (26 address lines).

31.5.2 Supported memories and transactions

Table 172 below displays an example of the supported devices, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM. Transactions not allowed (or not supported) by the FSMC in this example appear in gray.

Table 172. NOR Flash/PSRAM supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR Flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	

Table 172. NOR Flash/PSRAM supported memories and transactions (continued)

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and nonmultiplexed I/Os)	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	
	Synchronous	R	16	16	Y	
	Synchronous	R	32	16	Y	
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses

31.5.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In synchronous read and write mode, the output data changes on the falling edge of the memory clock (FSMC_CLK).

31.5.4 NOR Flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR Flash, SRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FSMC always samples the data before de-asserting the chip select signal NE. This guarantees that the memory data-hold timing constraint is met (chip enable high to data transition, usually 0 ns min.)
- When extended mode is set, it is possible to mix modes A, B, C and D in read and write (it is for instance possible to read in mode A and write in mode B).

Mode 1 - SRAM/CRAM

Figure 390. Mode1 read accesses

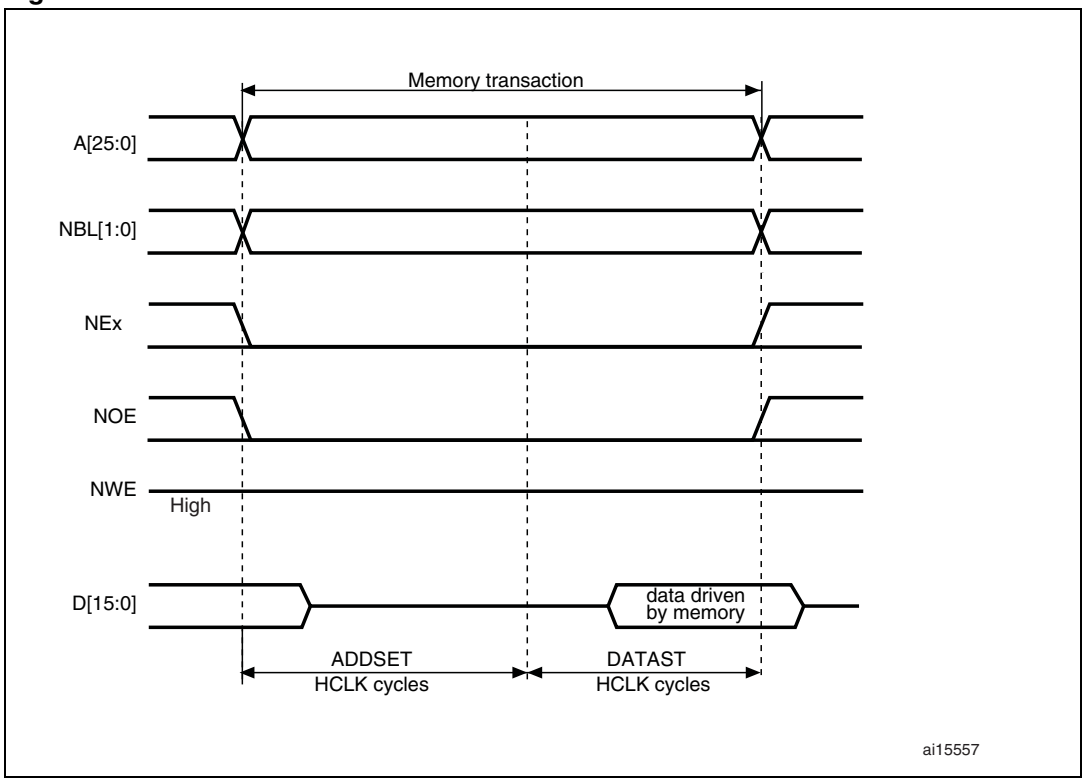
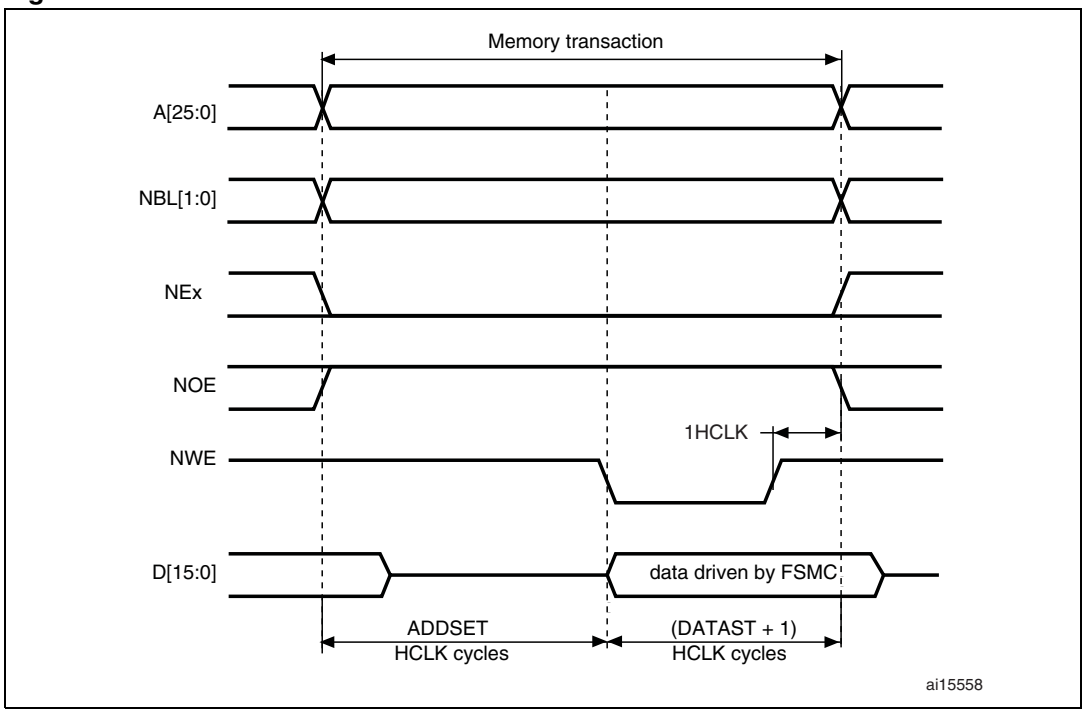


Figure 391. Mode1 write accesses



1. NBL[1:0] are driven low during read access.

The one HCLK cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this one HCLK cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 173. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 174. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-20		0x0000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles for write accesses, DATAST HCLK cycles for read accesses).
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 392. ModeA read accesses

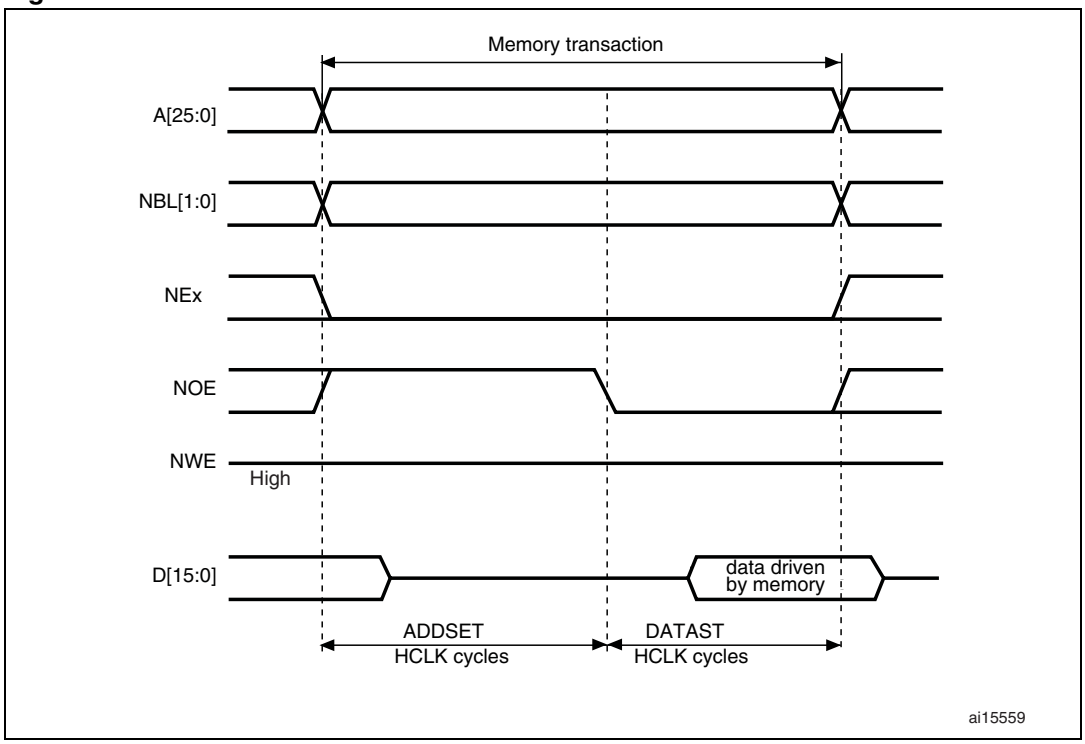
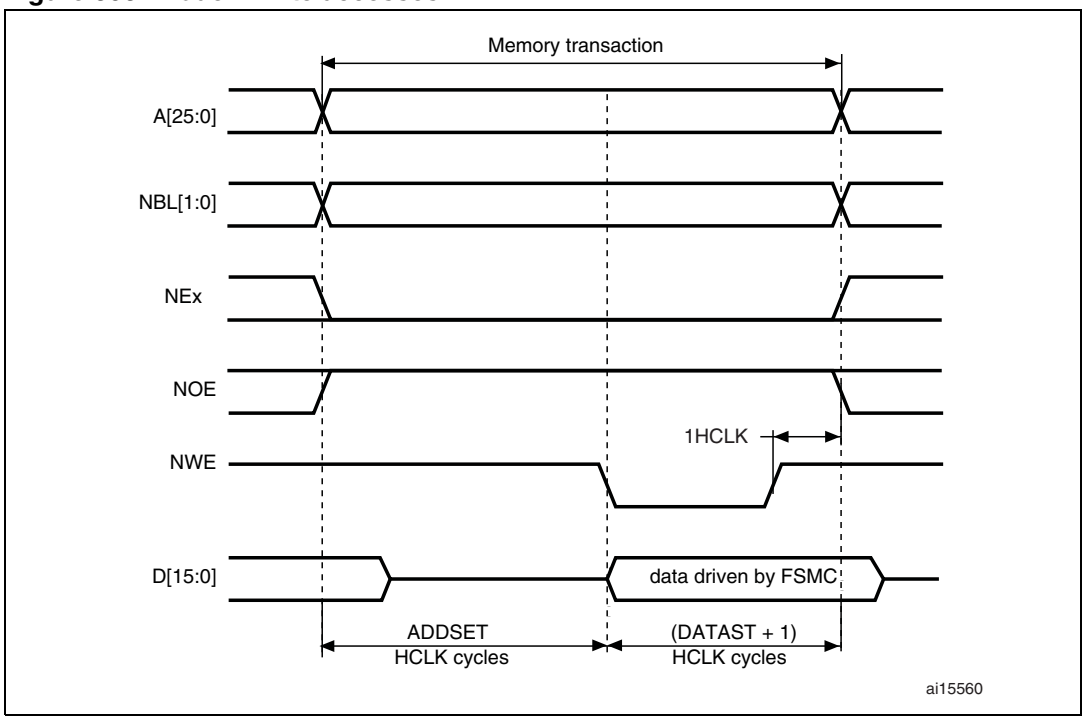


Figure 393. ModeA write accesses



1. NBL[1:0] are driven low during read access.

The differences compared with mode1 are the toggling of NOE and the independent read and write timings.

Table 175. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	-
5-4	MWID	As needed
3-2	MTYP	As needed, exclude 10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 176. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 177. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x0
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).

Table 177. FSMC_BWTRx bit fields (continued)

Bit number	Bit name	Value to set
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write Minimum value for ADDSET is 1.

Mode 2/B - NOR Flash

Figure 394. Mode2/B read accesses

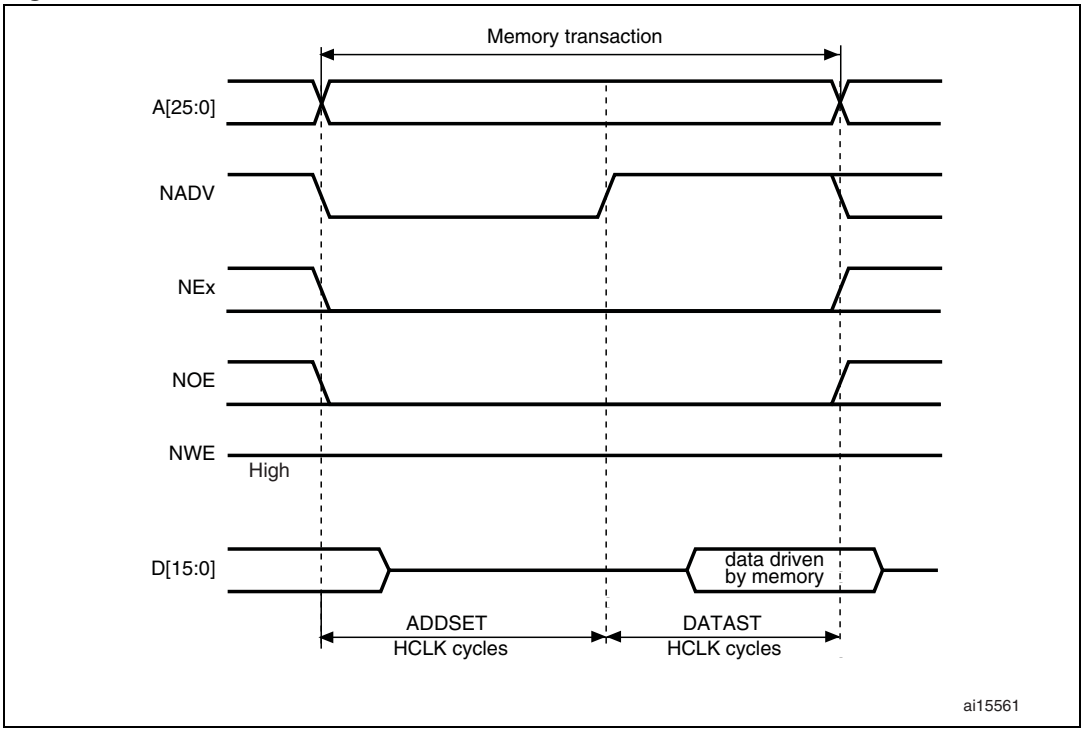


Figure 395. Mode2 write accesses

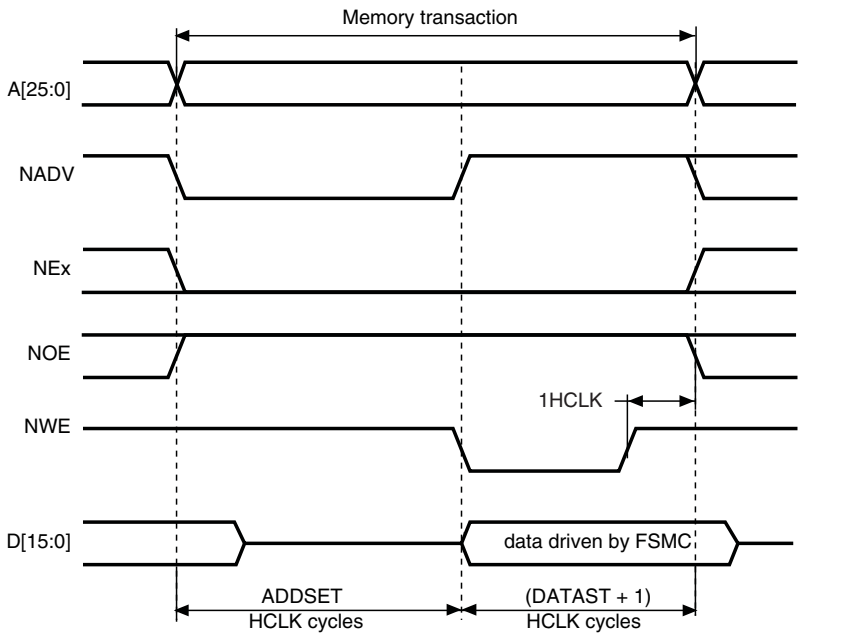
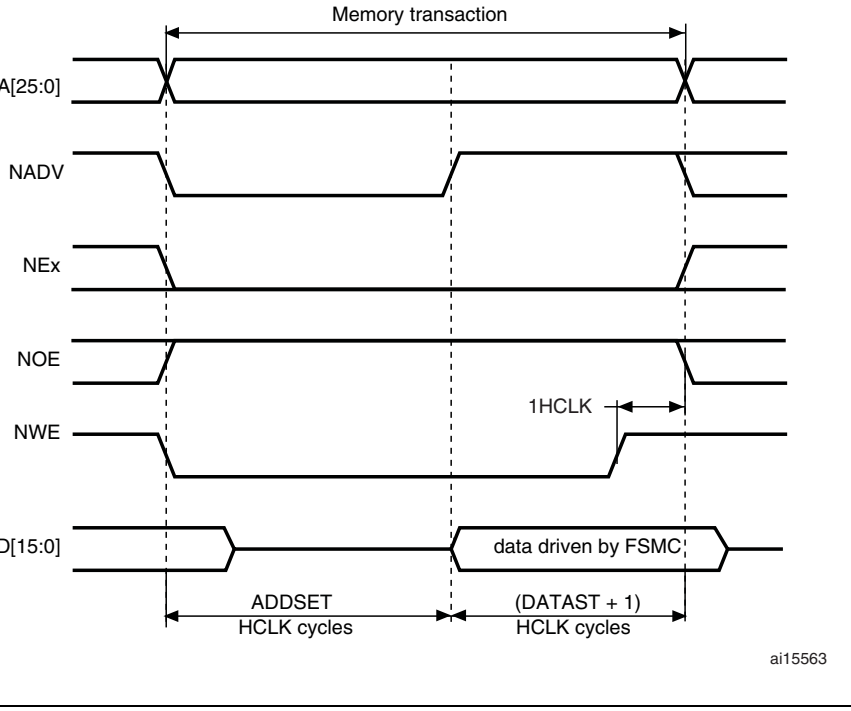


Figure 396. ModeB write accesses



The differences with mode1 are the toggling of NADV and the independent read and write timings when extended mode is set (Mode B).

Table 178. FSMC_BCRx bit fields

Bit number	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	10 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 179. FSMC_BTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in read. Minimum value for ADDSET is 1.

Table 180. FSMC_BWTRx bit fields

Bit number	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x1 if extended mode is set
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the access second phase (DATAST+1 HCLK cycles) in write.

Table 180. FSMC_BWTRx bit fields (continued)

Bit number	Bit name	Value to set
7-4		0x0
3-0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Note: The FSMC_BWTRx register is valid only if extended mode is set (mode B), otherwise all its content is don't care.

Mode C - NOR Flash - OE toggling

Figure 397. ModeC read accesses

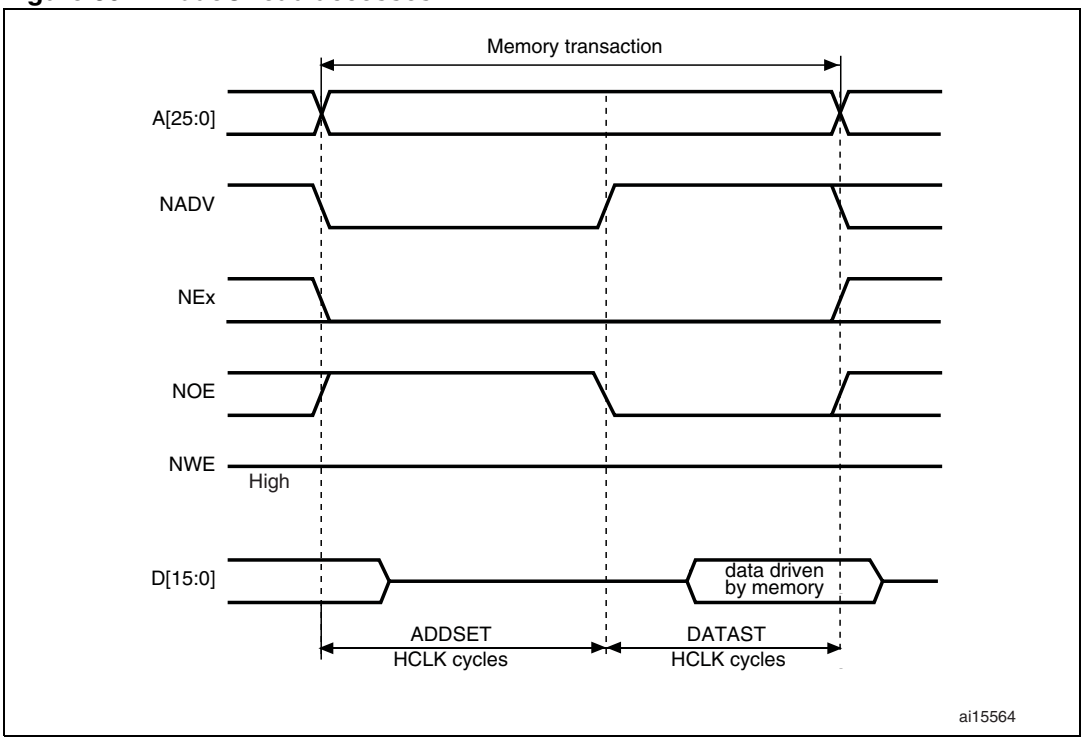
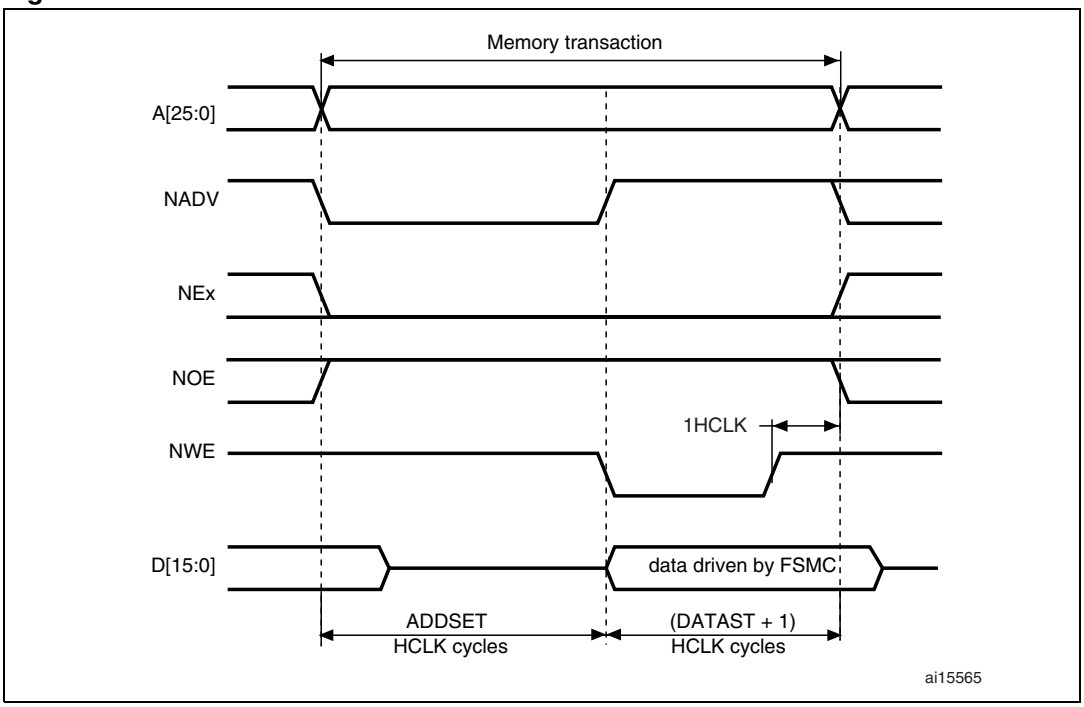


Figure 398. ModeC write accesses



The differences compared with mode1 are the toggling of NOE and NADV and the independent read and write timings.

Table 181. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	1
5-4	MWID	As needed
3-2	MTYP	0x02 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Table 182. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSETHCLK cycles) in read. Minimum value for ADDSET is 1.

Table 183. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.
7-4		0x0
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Mode D - asynchronous access with extended address

Figure 399. ModeD read accesses

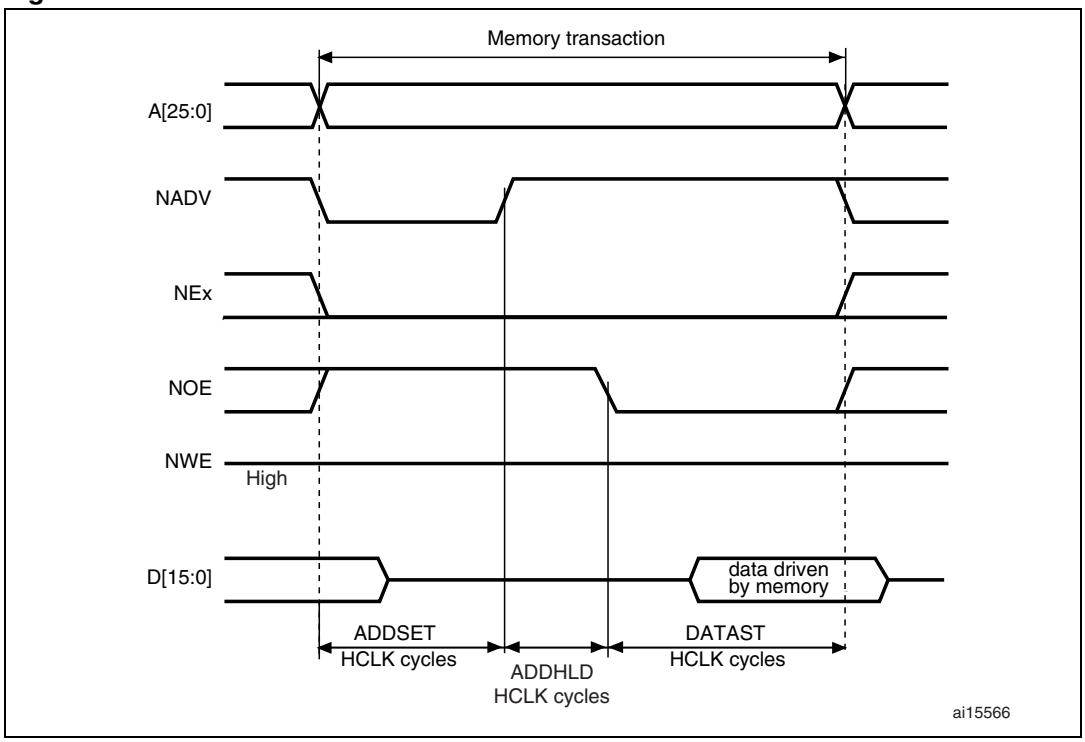
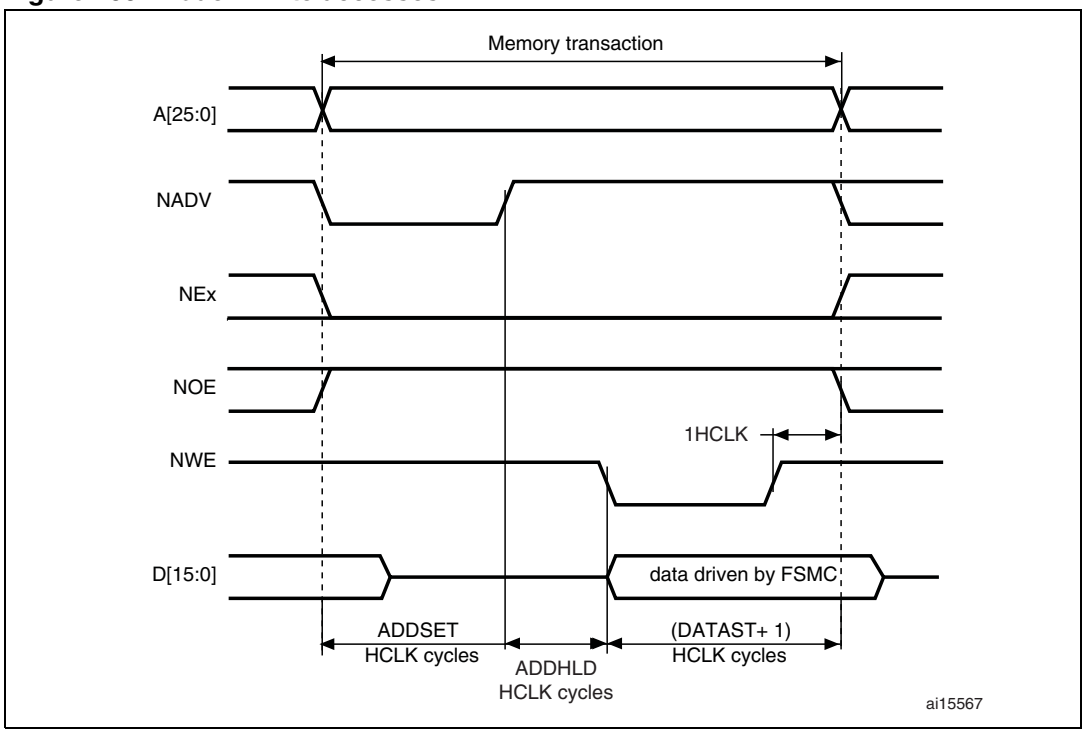


Figure 400. ModeD write accesses



The differences with mode1 are the toggling of NADV, NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 184. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 185. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles) in read.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSETHCLK cycles) in read. Minimum value for ADDSET is 1.

Table 186. FSMC_BWTRx bit fields

Bit No.	Bit name	Value to set
31-30		0x0
29-28	ACCMOD	0x2
27-20		0x000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	Duration of the second access phase (DATAST+1 HCLK cycles) in write.

Table 186. FSMC_BWTRx bit fields (continued)

Bit No.	Bit name	Value to set
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) in write. Minimum value for ADDSET is 1.

Mode muxed - asynchronous access muxed NOR Flash

Figure 401. Multiplexed read accesses

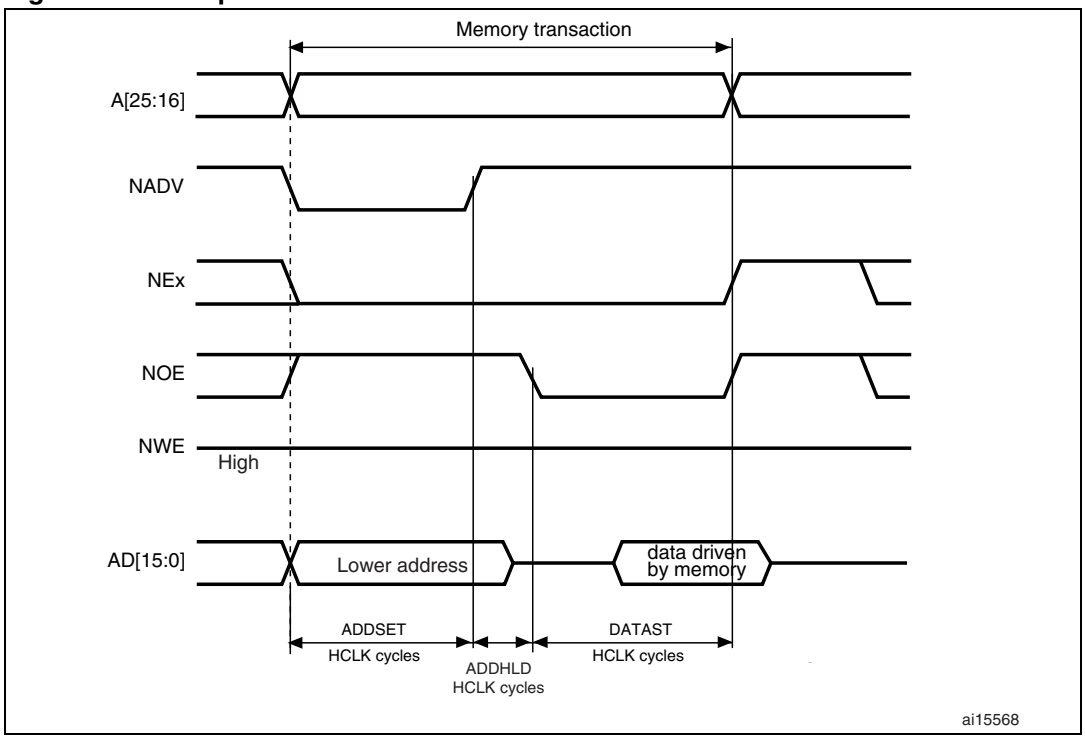
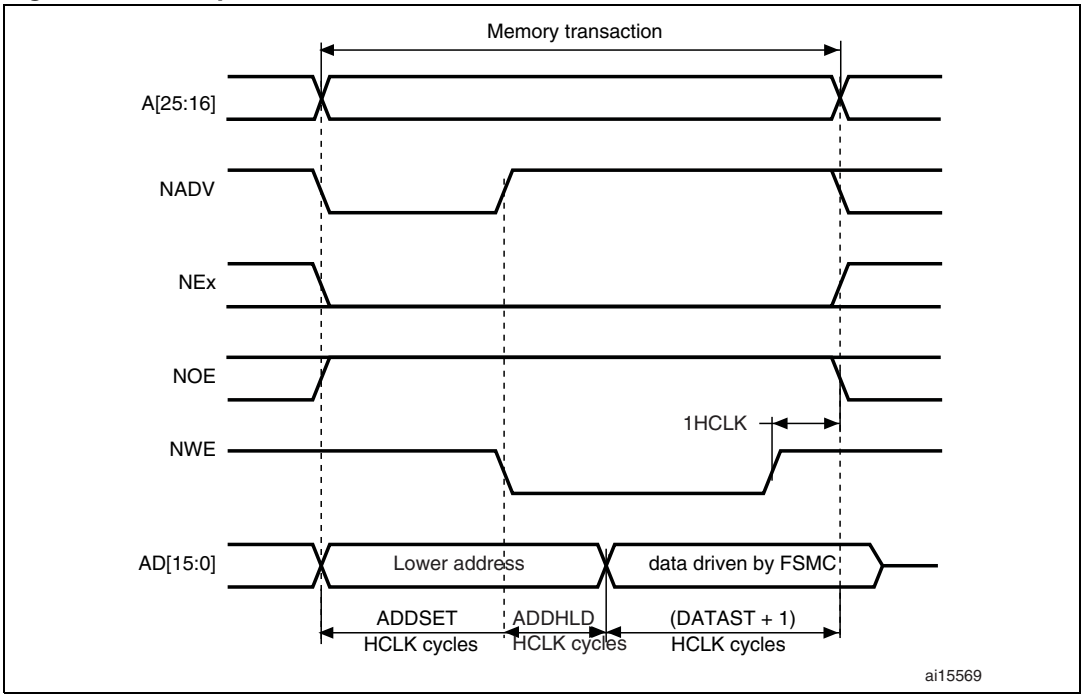


Figure 402. Multiplexed write accesses



The difference with mode D is the drive of the lower address byte(s) on the databus.

Table 187. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-16		0x0000
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13-10		0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7		-
6	FACCEN	0x1
5-4	MWID	As needed
3-2	MTYP	0x2 (NOR)
1	MUXEN	0x1
0	MBKEN	0x1

Table 188. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)

Table 188. FSMC_BTRx bit fields (continued)

Bit No.	Bit name	Value to set
15-8	DATAST	Duration of the second access phase (DATAST HCLK cycles for read accesses and DATAST+1 HCLK cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts a WAIT signal to advise that it's not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FSMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase) programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data phase must be programmed so that WAIT can be detected 4 HCLK cycles before the data sampling. The following cases must be considered:

1. Memory asserts the WAIT signal aligned to NOE/NWE which toggles:

   data_setup phase >= 4 * HCLK + max_wait_assertion_time

2. Memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):

   if max_wait_assertion_time > (address_phase + hold_phase)

   data_setup phase >= 4 * HCLK + (max_wait_assertion_time - address_phase - hold_phase)

   otherwise

   data_setup phase >= 4 * HCLK

Where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

The *Figure 403* and *Figure 404* show the number of HCLK clock cycles that memory access is extended after WAIT is removed by the asynchronous memory (independently of the above cases).

Figure 403. Asynchronous wait during a read access

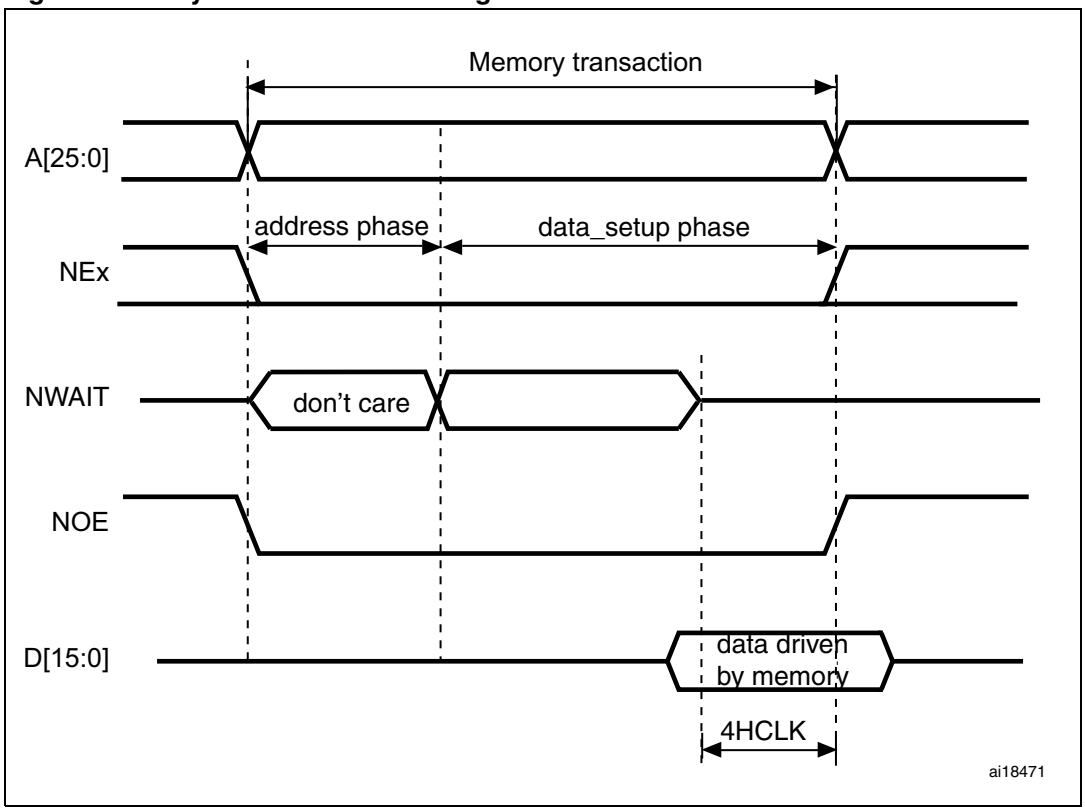
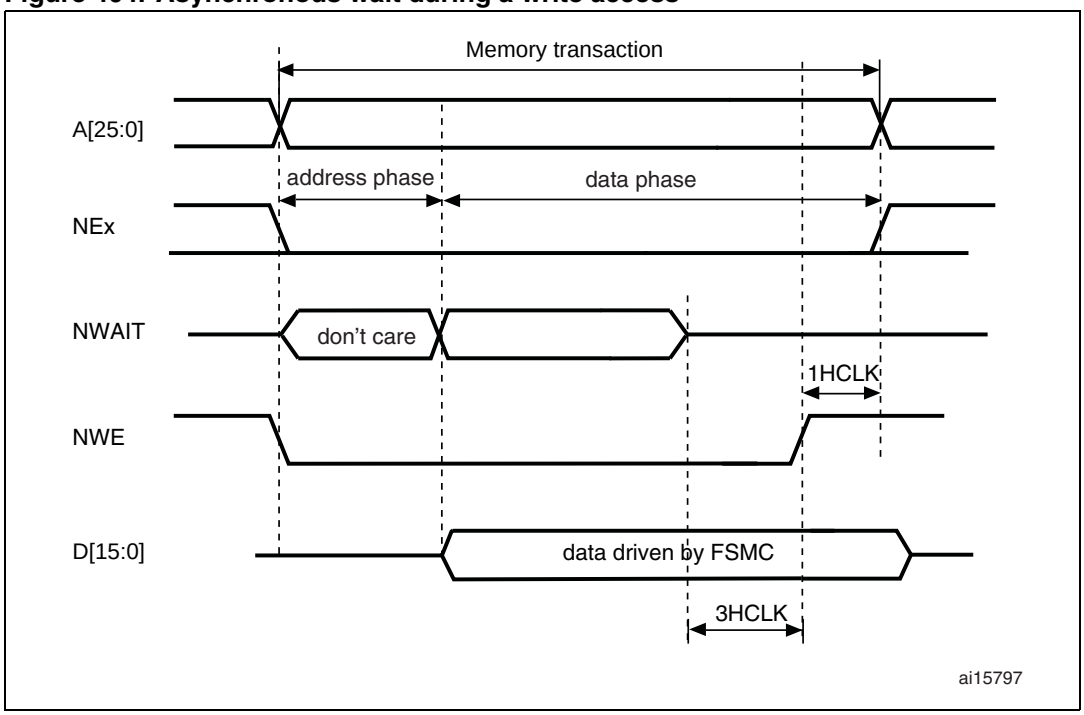


Figure 404. Asynchronous wait during a write access



31.5.5 Synchronous burst transactions

The memory clock, CLK, is a submultiple of HCLK according to the value of parameter CLKDIV.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FSMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs *in the middle* of the NADV low pulse.

Data latency versus NOR Flash latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR Flash configuration register. The FSMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR Flash memories include the NADV Low cycle in the data latency count, so the exact relation between the NOR Flash latency and the FMSC DATLAT parameter can be either of:

- NOR Flash latency = DATLAT + 2
- NOR Flash latency = DATLAT + 3

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FSMC samples the data and waits long enough to evaluate if the data are valid. Thus the FSMC detects when the memory exits latency and real data are taken.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FSMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in synchronous burst mode, if an AHB single-burst transaction is requested, the FSMC performs a burst transaction of length 1 (if the AHB transfer is 16-bit), or length 2 (if the AHB transfer is 32-bit) and de-assert the chip select signal when the last data is strobed.

Clearly, such a transfer is not the most efficient in terms of cycles (compared to an asynchronous read). Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Wait management

For synchronous burst NOR Flash, NWAIT is evaluated after the programmed latency period, (DATALAT+2) CLK clock cycles.

If NWAIT is sensed active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is sensed inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid, and does not consider the data valid.

There are two timing configurations for the NOR Flash NWAIT signal in burst mode:

- Flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset)
- Flash memory asserts the NWAIT signal during the wait state

These two NOR Flash wait state configurations are supported by the FSMC, individually for each chip select, thanks to the WAITCFG bit in the FSMC_BCRx registers (x = 0..3).

Figure 405. Wait configurations

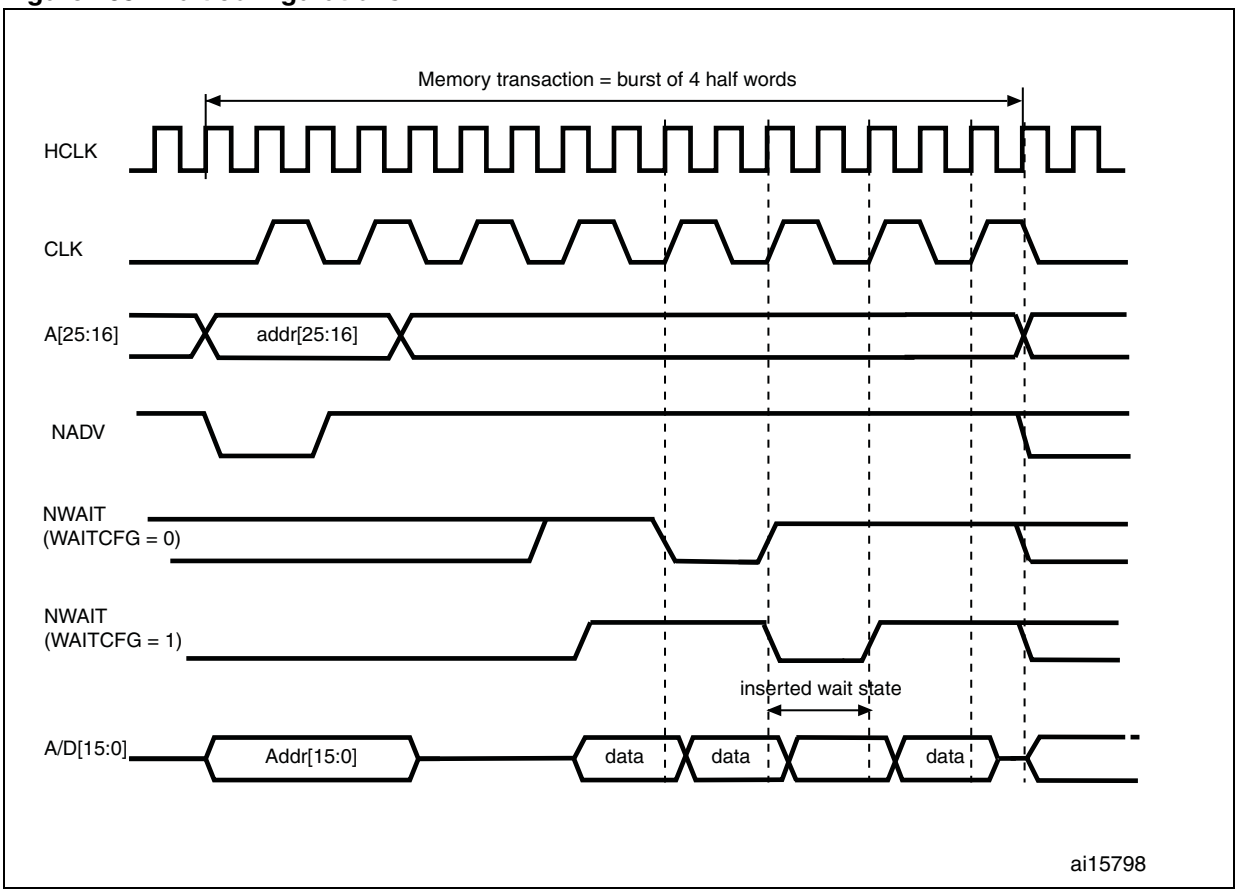
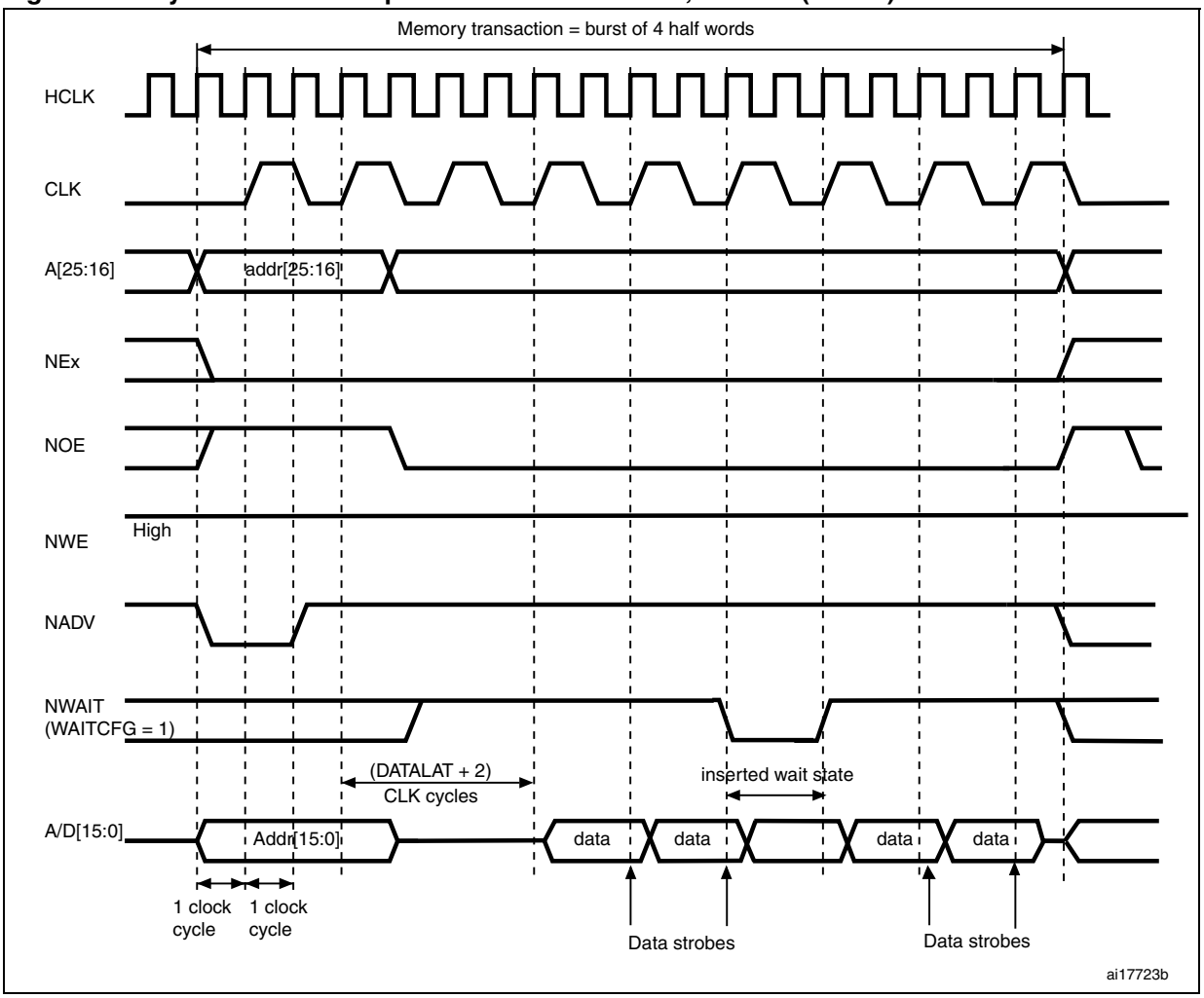


Figure 406. Synchronous multiplexed read mode - NOR, PSRAM (CRAM)



1. Byte lane outputs BL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 189. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	No effect on synchronous read
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory

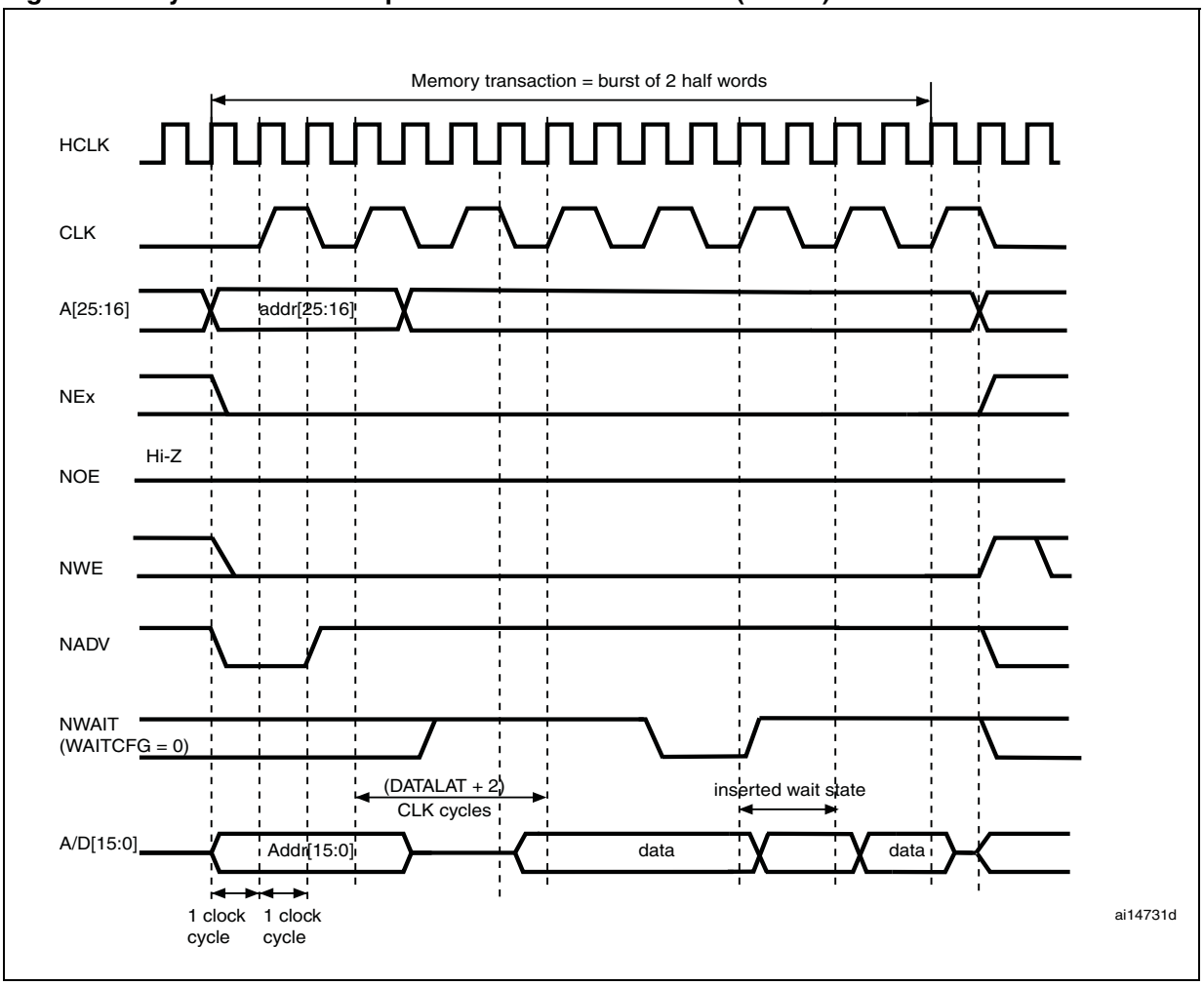
Table 189. FSMC_BCRx bit fields (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x1
7	FWPRLVL	Set to protect memory from accidental write access
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 190. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK)
15-8	DATAST	no effect
7-4	ADDHLD	no effect
3-0	ADDSET	no effect

Figure 407. Synchronous multiplexed write mode - PSRAM (CRAM)



1. Memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 191. FSMC_BCRx bit fields

Bit No.	Bit name	Value to set
31-20		0x0000
19	CBURSTRW	0x1
18-15		0x0
14	EXTMOD	0x0
13	WAITEN	When high, the first data after latency period is taken as always valid, regardless of the wait from memory value
12	WREN	no effect on synchronous read
11	WAITCFG	0x0
10	WRAPMOD	no effect
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	FWPRLVL	Set to protect memory from accidental writes
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 192. FSMC_BTRx bit fields

Bit No.	Bit name	Value to set
31-30	-	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0 to get CLK = HCLK (not supported) 1 to get CLK = 2 × HCLK
19-16	BUSTURN	No effect
15-8	DATAST	No effect
7-4	ADDHLD	No effect
3-0	ADDSET	No effect

31.5.6 NOR/PSRAM controller registers

SRAM/NOR-Flash chip-select control registers 1..4 (FSMC_BCR1..4)

Address offset: 0xA000 0000 + 8 * (x – 1), x = 1...4

Reset value: 0x0000 30DX

This register contains the control information of each memory bank, used for SRAMs, ROMs and asynchronous or burst NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	Reserved			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 19 **CBURSTRW:** Write burst enable.

For Cellular RAM, the bit enables synchronous burst protocol during write operations. For Flash memory access in burst mode, this bit enables/disables the wait state insertion via the NWAIT signal. The enable bit for the synchronous burst protocol during read access is the BURSTEN bit in the FSMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables the FSMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)

1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 **EXTMOD:** Extended mode enable.

This bit enables the FSMC to program inside the FSMC_BWTR register, so it allows different timings for read and write.

0: values inside FSMC_BWTR register are not taken into account (default after reset)

1: values inside FSMC_BWTR register are taken into account

Bit 13 **WAITEN:** Wait enable bit.

For Flash memory access in burst mode, this bit enables/disables wait-state insertion via the NWAIT signal:

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed Flash latency period)

1: NWAIT signal is enabled (its level is taken into account after the programmed Flash latency period to insert wait states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit.

This bit indicates whether write operations are enabled/disabled in the bank by the FSMC:

0: Write operations are disabled in the bank by the FSMC, an AHB error is reported,

1: Write operations are enabled for the bank by the FSMC (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration.

For memory access in burst mode, the NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:
0: NWAIT signal is active one data cycle before wait state (default after reset),
1: NWAIT signal is active during wait state (not for Cellular RAM).

Bit 10 **WRAPMOD:** Wrapped burst mode support.

Defines whether the controller will or not split an AHB burst wrap access into two linear accesses. Valid only when accessing memories in burst mode
0: Direct wrapped burst is not enabled (default after reset),
1: Direct wrapped burst is enabled.

Note: This bit has no effect as the CPU and DMA cannot generate wrapping burst transfers.

Bit 9 **WAITPOL:** Wait signal polarity bit.

Defines the polarity of the wait signal from memory. Valid only when accessing the memory in burst mode:
0: NWAIT active low (default after reset),
1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit.

Enables the burst access mode for the memory. Valid only with synchronous burst memories:
0: Burst access mode disabled (default after reset)
1: Burst access mode enable

Bit 7 Reserved, must be kept at reset value..

Bit 6 **FACCEN:** Flash access enable

Enables NOR Flash memory access operations.
0: Corresponding NOR Flash memory access is disabled
1: Corresponding NOR Flash memory access is enabled (default after reset)

Bits 5:4 **MWID:** Memory databus width.

Defines the external memory device width, valid for all type of memories.
00: 8 bits,
01: 16 bits (default after reset),
10: reserved, do not use,
11: reserved, do not use.

Bits 3:2 **MTYP:** Memory type.

Defines the type of external memory attached to the corresponding memory bank:
00: SRAM, ROM (default after reset for Bank 2...4)
01: PSRAM (Cellular RAM: CRAM)
10: NOR Flash/OneNAND Flash (default after reset for Bank 1)
11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit.

When this bit is set, the address and data values are multiplexed on the databus, valid only with NOR and PSRAM memories:
0: Address/Data nonmultiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit.

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.
0: Corresponding memory bank is disabled
1: Corresponding memory bank is enabled

SRAM/NOR-Flash chip-select timing registers 1..4 (FSMC_BTR1..4)

Address offset: 0xA000 0000 + 0x04 + 8 * (x – 1), x = 1..4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. If the EXTMOD bit is set in the FSMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FSMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 29:28 **ACCMOD:** Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.

00: access mode A

01: access mode B

10: access mode C

11: access mode D

Bits 27:24 **DATLAT**: Data latency for synchronous burst NOR Flash memory

For NOR Flash with synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:

0000: Data latency of 2 CLK clock cycles for first burst access

1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (CLK) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In the case of CRAM, this field must be set to '0'.

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal)

Defines the period of CLK clock output signal, expressed in number of HCLK cycles:

0000: Reserved

0001: CLK period = 2 × HCLK periods

0010: CLK period = 3 × HCLK periods

1111: CLK period = 16 × HCLK periods (default value after reset)

In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN:** Bus turnaround phase duration

These bits are written by software to insert the bus turnaround delay after a read access only from multiplexed NOR Flash memory to avoid bus contention if the controller needs to drive addresses on the databus for the next side-by-side transaction. BUSTURN can be set to the minimum if the slowest memory does not take more than 6 HCLK clock cycles to put the databus in Hi-Z state.

These bits are written by software to add a delay at the end of a write/read transaction. This delay allows to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (tEHQZ):

(BUSTRUN + 1)HCLK period ≥ $t_{EHELmin}$ and (BUSTRUN + 2)HCLK period ≥ $t_{EHQZmax}$ if EXTMOD = '0'

(BUSTRUN + 2)HCLK period ≥ max ($t_{EHELmin}$, $t_{EHQZmax}$) if EXTMOD = '1'.

0000: BUSTURN phase duration = 0 HCLK clock cycle added

...

1111: BUSTURN phase duration = 15 × HCLK clock cycles (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 390* to *Figure 402*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000 0000: Reserved

0000 0001: DATAST phase duration = 1 × HCLK clock cycles

0000 0010: DATAST phase duration = 2 × HCLK clock cycles

...

1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

For each memory type and access mode data-phase duration, please refer to the respective figure (*Figure 390* to *Figure 402*).

Example: Mode1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 399* to *Figure 402*), used in mode D and multiplexed accesses:

0000: Reserved

0001: ADDHLD phase duration =1 × HCLK clock cycle

0010: ADDHLD phase duration = 2 × HCLK clock cycle

...

1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

For each access mode address-hold phase duration, please refer to the respective figure (*Figure 399* to *Figure 402*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 390* to *Figure 402*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

1111: ADDSET phase duration = 1615 × HCLK clock cycles (default value after reset)

For each access mode address setup phase duration, please refer to the respective figure (refer to *Figure 390* to *Figure 402*).

Note: In synchronous accesses, this value is don't care.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
With PSRAMs (CRAMs) the filed DATLAT must be set to 0, so that the FSMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous Flash memories that issue the NWAIT signal, unlike older Flash memories (check the datasheet of the specific Flash memory being used).

SRAM/NOR-Flash write timing registers 1..4 (FSMC_BWTR1..4)

Address offset: 0xA000 0000 + 0x104 + 8 * (x – 1), x = 1...4

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, ROMs and NOR Flash memories. When the EXTMOD bit is set in the FSMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 29:28 **ACCMOD:** Access mode.
Specifies the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FSMC_BCRx register is 1.
00: access mode A
01: access mode B
10: access mode C
11: access mode D

Bits 27:24 **DATLAT:** Data latency (for synchronous burst NOR Flash).
For NOR Flash with Synchronous burst mode enabled, defines the number of memory clock cycles (+2) to issue to the memory before getting the first data:
0000: (0x0) Data latency of 2 CLK clock cycles for first burst access
...
1111: (0xF) Data latency of 17 CLK clock cycles for first burst access (default value after reset)
*Note: This timing parameter is not expressed in HCLK periods, but in Flash clock (**CLK**) periods. In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care. In case of CRAM, this field must be set to 0*

Bits 23:20 **CLKDIV:** Clock divide ratio (for CLK signal).
Defines the period of CLK clock output signal, expressed in number of HCLK cycles:
0000: Reserved
0001 CLK period = 2 × HCLK periods
0010 CLK period = 3 × HCLK periods
1111: CLK period = 16 × HCLK periods (default value after reset)
In asynchronous NOR Flash, SRAM or ROM accesses, this value is don't care.

Bits 19:16 **BUSTURN**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (t_{EHEL} from ENx high to ENx low):
(BUSTRUN + 1) HCLK period ≥ $t_{EHELmin}$.
0000: BUSTURN phase duration = 0 HCLK clock cycle added
...
1111: BUSTURN phase duration = 15 HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 390* to *Figure 402*), used in SRAMs, ROMs and asynchronous NOR Flash accesses:
0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 399* to *Figure 402*), used in SRAMs, ROMs and asynchronous multiplexed NOR Flash accesses:
0000: Reserved
0001: ADDHLD phase duration = 1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is not used, the address hold phase is always 1 Flash clock period duration.

Bits 3:0 **ADDSET:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to *Figure 399* to *Figure 402*), used in SRAMs, ROMs and asynchronous NOR Flash accessed:
0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR Flash accesses, this value is don't care.

31.6 NAND Flash/PC Card controller

The FSMC generates the appropriate signal timings to drive the following types of device:

- NAND Flash
  - 8-bit
  - 16-bit
- 16-bit PC Card compatible devices

The NAND/PC Card controller can control three external banks. Bank 2 and bank 3 support NAND Flash devices. Bank 4 supports PC Card devices.

Each bank is configured by means of dedicated registers (*Section 31.6.8*). The programmable memory parameters include access timings (shown in *Table 193*) and ECC configuration.

Table 193. Programmable NAND/PC Card access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	256
Memory wait	Minimum duration (HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	256
Memory hold	Number of clock cycles (HCLK) to hold the address (and the data in case of a write access) after the command de-assertion	Read/Write	AHB clock cycle (HCLK)	1	255
Memory databus high-Z	Number of clock cycles (HCLK) during which the databus is kept in high-Z state after the start of a write access	Write	AHB clock cycle (HCLK)	0	255

31.6.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND Flash and PC Card.

Caution: When using a PC Card or a CompactFlash in I/O mode, the NIOS16 input pin must remain at ground level during the whole operation, otherwise the FSMC may not operate properly. This means that the NIOS16 input pin must *not* be connected to the card, but directly to ground (only 16-bit accesses are allowed).

Note: Prefix “N”. specifies the associated signal as active low.

8-bit NAND Flash

Table 194. 8-bit NAND Flash

FSMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip select, x = 2, 3
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy input signal to the FSMC

There is no theoretical capacity limitation as the FSMC can manage as many address cycles as needed.

16-bit NAND Flash

Table 195. 16-bit NAND Flash

FSMC signal name	I/O	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip select, x = 2, 3
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy input signal to the FSMC

There is no theoretical capacity limitation as the FSMC can manage as many address cycles as needed.

Table 196. 16-bit PC Card

FSMC signal name	I/O	Function
A[10:0]	O	Address bus
NIOS16	I	Data transfer in I/O space. It must be shorted to GND (16-bit transfer only)
NIORD	O	Output enable for I/O space
NIOWR	O	Write enable for I/O space
NREG	O	Register signal indicating if access is in Common or Attribute space
D[15:0]	I/O	Bidirectional databus
NCE4_1	O	Chip select 1
NCE4_2	O	Chip select 2 (indicates if access is 16-bit or 8-bit)
NOE	O	Output enable in Common and in Attribute space
NWE	O	Write enable in Common and in Attribute space
NWAIT	I	PC Card wait input signal to the FSMC (memory signal name IORDY)
INTR	I	PC Card interrupt to the FSMC (only for PC Cards that can generate an interrupt)
CD	I	PC Card presence detection. Active high. If an access is performed to the PC Card banks while CD is low, an AHB error is generated. Refer to *Section 31.3: AHB interface*

31.6.2 NAND Flash / PC Card supported memories and transactions

Table 197 below shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND Flash / PC Card controller appear in gray.

Table 197. Supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FSMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FSMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FSMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FSMC accesses
NAND 16-bit	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FSMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FSMC accesses

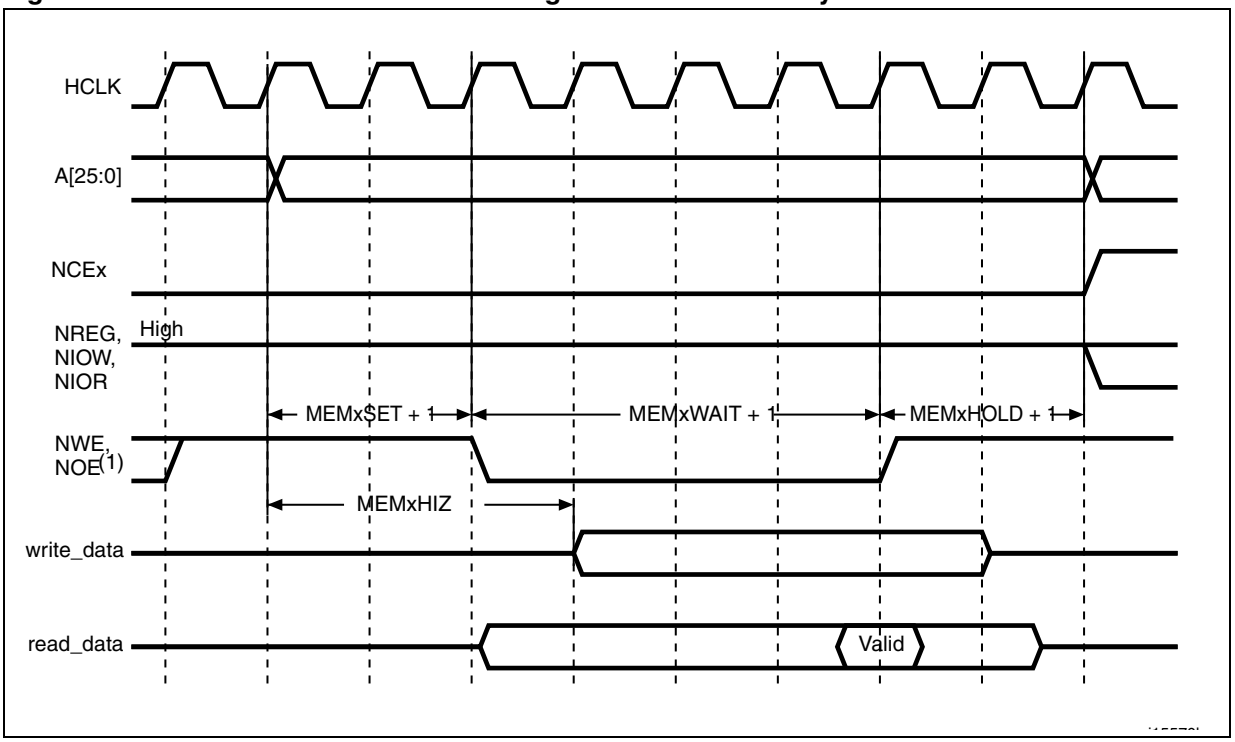
31.6.3 Timing diagrams for NAND and PC Card

Each PC Card/CompactFlash and NAND Flash memory bank is managed through a set of registers:

- Control register: FSMC_PCRx
- Interrupt status register: FSMC_SRx
- ECC register: FSMC_ECCRx
- Timing register for Common memory space: FSMC_PMEMx
- Timing register for Attribute memory space: FSMC_PATTx
- Timing register for I/O space: FSMC_PIOx

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any PC Card/CompactFlash or NAND Flash access, plus one parameter that defines the timing for starting driving the databus in the case of a write. *Figure 408* shows the timing parameter definitions for common memory accesses, knowing that Attribute and I/O (only for PC Card) memory space access timings are similar.

Figure 408. NAND/PC Card controller timing for common memory access



1. NOE remains high (inactive) during write access. NWE remains high (inactive) during read access.

31.6.4 NAND Flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash device are driven by some address signals of the FSMC controller. This means that to send a command or an address to the NAND Flash memory, the CPU has to perform a write to a certain address in its memory space.

A typical page read operation from the NAND Flash device is as follows:

1. Program and enable the corresponding memory bank by configuring the FSMC_PCRx and FSMC_PMEMx (and for some devices, FSMC_PATTx, see *Section 31.6.5: NAND Flash pre-wait functionality on page 1266*) registers according to the characteristics of the NAND Flash (PWID bits for the databus width of the NAND Flash, PTYP = 1, PWAITEN = 1, PBKEN = 1, see section *Common memory space timing register 2..4 (FSMC_PMEM2..4) on page 1272* for timing configuration).
2. The CPU performs a byte write in the common memory space, with data byte equal to one Flash command byte (for example 0x00 for Samsung NAND Flash devices). The CLE input of the NAND Flash is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND Flash. Once the command is latched by the NAND Flash device, it does not need to be written for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], then STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] for 64 Mb x 8 bit NAND Flash) in the common memory or attribute space. The ALE input of the NAND Flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it

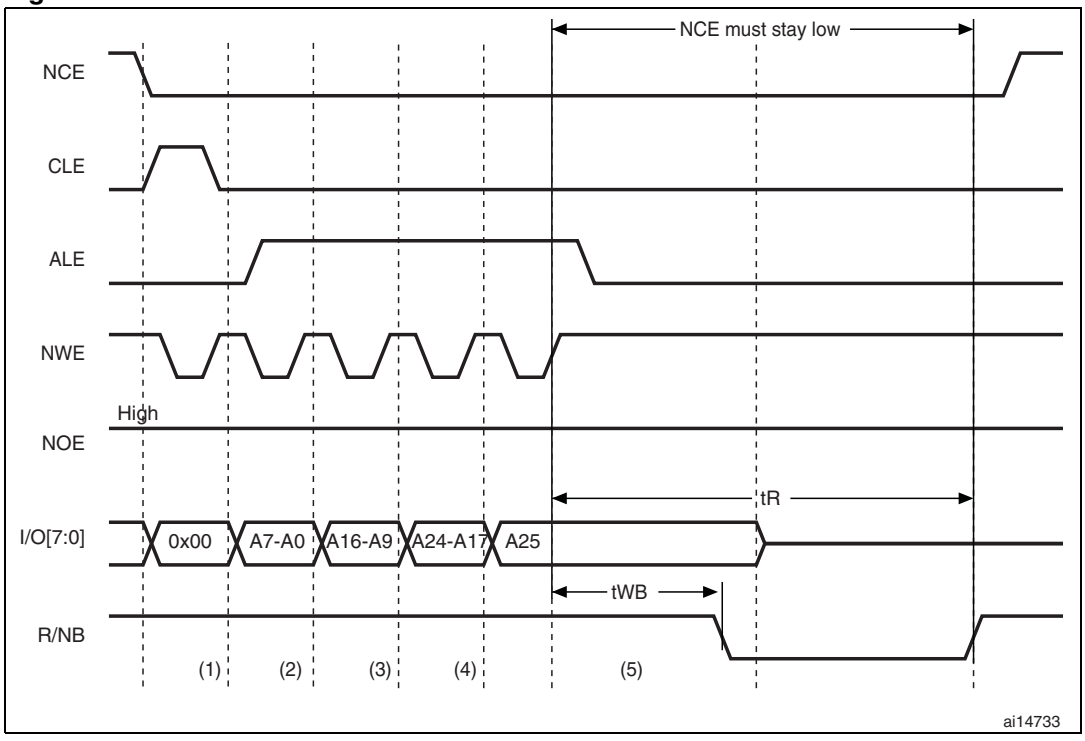
possible to use a different timing configuration of the FSMC, which can be used to implement the prewait functionality needed by some NAND Flash memories (see details in *Section 31.6.5: NAND Flash pre-wait functionality on page 1266*).

4. The controller waits for the NAND Flash to be ready (R/NB signal high) to become active, before starting a new access (to same or another memory bank). While waiting, the controller maintains the NCE signal active (low).
5. The CPU can then perform byte read operations in the common memory space to read the NAND Flash page (data field + Spare field) byte by byte.
6. The next NAND Flash page can be read without any CPU command or address write operation, in three different ways:
   - by simply performing the operation described in step 5
   - a new random address can be accessed by restarting the operation at step 3
   - a new command can be sent to the NAND Flash device by restarting at step 2

31.6.5 NAND Flash pre-wait functionality

Some NAND Flash devices require that, after writing the last part of the address, the controller wait for the R/NB signal to go low as shown in *Figure 409*.

Figure 409. Access to non 'CE don't care' NAND-Flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x7802 0000: FSMC performs a write access using FSMC_PATT2 timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × HCLK = 112 ns > t_{WB} max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND Flash memories where NCE is not don't care).

When this functionality is needed, it can be guaranteed by programming the MEMHOLD value to meet the t_{WB} timing, however any CPU read or write access to the NAND Flash then has the hold delay of (MEMHOLD + 1) HCLK cycles inserted from the rising edge of the NWE signal to the next access.

To overcome this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and leaving the MEMHOLD value at its minimum. Then, the CPU must use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where the CPU must write to the attribute memory space.

31.6.6 Error correction code computation ECC (NAND Flash)

The FSMC PC-Card controller includes two error correction code computation hardware blocks, one per memory bank. They are used to reduce the host CPU workload when processing the error correction code by software in the system.

These two registers are identical and associated with bank 2 and bank 3, respectively. As a consequence, no hardware ECC computation is available for memories connected to bank 4.

The error correction code (ECC) algorithm implemented in the FSMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read from or written to NAND Flash.

The ECC modules monitor the NAND Flash databus and read/write signals (NCE and NWE) each time the NAND Flash memory bank is active.

The functional operations are:

- When access to NAND Flash is made to bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When access to NAND Flash occurs at any other address, the ECC logic is idle, and does not perform any operation. Thus, write operations for defining commands or addresses to NAND Flash are not taken into account for ECC computation.

Once the desired number of bytes has been read from/written to the NAND Flash by the host CPU, the FSMC_ECCR2/3 registers must be read in order to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to zero. To compute a new data block, the ECCEN bit must be set to one in the FSMC_PCR2/3 registers.

31.6.7 PC Card/CompactFlash operations

Address spaces & memory accesses

The FSMC supports Compact Flash storage or PC Cards in Memory Mode and I/O Mode (True IDE mode is not supported).

The Compact Flash storage and PC Cards are made of 3 memory spaces:

- Common Memory Space
- Attribute Space
- I/O Memory Space

The nCE2 and nCE1 pins (FSMC_NCE4_2 and FSMC_NCE4_1 respectively) select the card and indicate whether a byte or a word operation is being performed: nCE2 accesses

the odd byte on D15-8 and nCE1 accesses the even byte on D7-0 if A0=0 or the odd byte on D7-0 if A0=1. The full word is accessed on D15-0 if both nCE2 and nCE1 are low.

The memory space is selected by asserting low nOE for read accesses or nWE for write accesses, combined with the low assertion of nCE2/nCE1 and nREG.

- If pin nREG=1 during the memory access, the common memory space is selected
- If pin nREG=0 during the memory access, the attribute memory space is selected

The I/O Space is selected by asserting low nIORD for read accesses or nIOWR for write accesses [instead of nOE/nWE for memory Space], combined with nCE2/nCE1. Note that nREG must also be asserted low during accesses to I/O Space.

Three type of accesses are allowed for a 16-bit PC Card:

- Accesses to Common Memory Space for data storage can be either 8-bit accesses at even addresses or 16 bit AHB accesses.

  Note that 8-bit accesses at odd addresses are not supported and will not lead to the low assertion of nCE2. A 32-bit AHB request is translated into two 16-bit memory accesses.
- Accesses to Attribute Memory Space where the PC Card stores configuration information are limited to 8-bit AHB accesses at even addresses.

  Note that a 16-bit AHB access will be converted into a single 8-bit memory transfer: nCE1 will be asserted low, NCE2 will be asserted high and only the even Byte on D7-D0 will be valid. Instead a 32-bit AHB access will be converted into two 8-bit memory transfers at even addresses: nCE1 will be asserted low, NCE2 will be asserted high and only the even bytes will be valid.
- Accesses to I/O Space must be limited to AHB 16 bit accesses.

Table 198. 16-bit PC-Card signals and access type

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access Type	Allowed/not Allowed
1	0	1	0	1	X	X	X-X	X	Common Memory Space	Read/Write byte on D7-D0	YES
	YES										YES
0	1	1	0	1	X	X	X-X	X		Read/Write byte on D15-D8	Not supported
0	0	1	0	1	X	X	X-X	0		Read/Write word on D15-D0	YES
X	0	0	0	1	0	1	X-X	0	Attribute Space	Read or Write Configuration Registers	YES
X	0	0	0	1	0	0	X-X	0		Read or Write CIS (Card Information Structure)	YES
1	0	0	0	1	X	X	X-X	1	Invalid Attribute Space	Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Read or Write (odd address)	YES

Table 198. 16-bit PC-Card signals and access type (continued)

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access Type	Allowed/not Allowed
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	Not supported
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	Not supported
0	0	0	1	0	X	X	X-X	0		Read Word on D15-0	YES
0	0	0	1	0	X	X	X-X	0		Write word on D15-0	YES
0	1	0	1	0	X	X	X-X	X		Read Odd Byte on D15-8	Not supported
0	1	0	1	0	X	X	X-X	X		Write Odd Byte on D15-8	Not supported

The FSMC Bank 4 gives access to those 3 memory spaces as described in *Section 31.4.2: NAND/PC Card address mapping* - *Table 165: Memory mapping and timing registers*

Wait Feature

The CompactFlash Storage or PC Card may request the FSMC to extend the length of the access phase programmed by MEMWAITx/ATTWAITx/IOWAITx bits, asserting the nWAIT signal after nOE/nWE or nIORD/nIOWR activation if the wait feature is enabled through the PWAITEN bit in the FSMC_PCRx register. In order to detect the nWAIT assertion correctly, the MEMWAITx/ATTWAITx/IOWAITx bits must be programmed as follows:

xxWAITx >= 4 + max_wait_assertion_time/HCLK

Where max_wait_assertion_time is the maximum time taken by NWAIT to go low once nOE/nWE or nIORD/nIOWR is low.

After the de-assertion of nWAIT, the FSMC extends the WAIT phase for 4 HCLK clock cycles.

31.6.8 NAND Flash/PC Card controller registers

PC Card/NAND Flash control registers 2..4 (FSMC_PCR2..4)

Address offset: 0xA0000000 + 0x40 + 0x20 * (x – 1), x = 2..4

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												ECCPS			TAR				TCLR				Res.	ECCEN	PWID		PTYP	PBKEN	PWAITEN	Reserved
												rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bits 19:17 **ECCPS:** ECC page size.
Defines the page size for the extended ECC:
000: 256 bytes
001: 512 bytes
010: 1024 bytes
011: 2048 bytes
100: 4096 bytes
101: 8192 bytes

Bits 16:13 **TAR:** ALE to RE delay.
Sets time from ALE low to RE low in number of AHB clock cycles (HCLK).
Time is: t_ar = (TAR + SET +) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles
Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR:** CLE to RE delay.
Sets time from CLE low to RE low in number of AHB clock cycles (HCLK).
Time is t_clr = (TCLR + SET +) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles
Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value..

Bits 6 **ECCEN:** ECC computation logic enable bit
0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

Bits 5:4 **PWID:** Databus width.
Defines the external memory device width.
00: 8 bits (default after reset)
01: 16 bits (mandatory for PC Card)
10: reserved, do not use
11: reserved, do not use

Bit 3 **PTYP:** Memory type.
Defines the type of device attached to the corresponding memory bank:
0: PC Card, CompactFlash, CF+ or PCMCIA
1: NAND Flash (default after reset)

Bit 2 **PBKEN:** PC Card/NAND Flash memory bank enable bit.
Enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus
0: Corresponding memory bank is disabled (default after reset)
1: Corresponding memory bank is enabled

Bit 1 **PWAITEN:** Wait feature enable bit.
Enables the Wait feature for the PC Card/NAND Flash memory bank:
0: disabled
1: enabled

Note: For a PC Card, when the wait feature is enabled, the MEMWAITx/ATTWAITx/IOWAITx bits must be programmed to a value as follows:
xxWAITx ≥ 4 + max_wait_assertion_time/HCLK
Where max_wait_assertion_time is the maximum time taken by NWAIT to go low once nOE/nWE or nIORD/nIOWR is low.

Bit 0 Reserved, must be kept at reset value..

FIFO status and interrupt register 2..4 (FSMC_SR2..4)

Address offset: 0xA000 0000 + 0x44 + 0x20 * (x-1), x = 2..4

Reset value: 0x0000 0040

This register contains information about FIFO status and interrupt. The FSMC has a FIFO that is used when writing to memories to store up to16 words of data from the AHB.
This is used to quickly write to the AHB and free it for transactions to peripherals other than the FSMC, while the FSMC is draining its FIFO into the memory. This register has one of its bits that indicates the status of the FIFO, for ECC purposes.
The ECC is calculated while the data are written to the memory, so in order to read the correct ECC the software must wait until the FIFO is empty.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
																									r	rw	rw	rw	rw	rw	rw

Bit 6 **FEMPT:** FIFO empty.
Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty

Bit 5 **IFEN:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 4 **ILEN:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 3 **IREN:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

Bit 2 **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

Bit 1 **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status
The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

Common memory space timing register 2..4 (FSMC_PMEM2..4)

Address offset: Address: 0xA000 0000 + 0x48 + 0x20 * (x – 1), x = 2..4

Reset value: 0xFCFC FCFC

Each FSMC_PMEMx (x = 2..4) read/write register contains the timing information for PC Card or NAND Flash memory bank x, used for access to the common memory space of the 16-bit PC Card/CompactFlash, or to access the NAND Flash for command, address write access and data read/write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MEMHIZx:** Common memory x databus HiZ time
Defines the number of HCLK clock cycles during which the databus is kept in HiZ after the start of a PC Card/NAND Flash write access to common memory space on socket x. Only valid for write transaction:
0000 0000: (0x00) 0 HCLK cycle (for PC Card)
1111 1111: (0xFF) 255 HCLK cycles (for PC Card) - (default value after reset)

Bits 23:16 **MEMHOLDx:** Common memory x hold time
Defines the number of HCLK clock cycles to hold address (and data for write access) after the command deassertion (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x:
0000 0000: reserved
0000 0001: 1 HCLK cycle
1111 1111: 255 HCLK cycles (default value after reset)

Bits 15:8 **MEMWAITx:** Common memory x wait time
Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
0000 0000: reserved
0000 0001: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: 256 HCLK cycles (+ wait cycle introduced by the Card deasserting NWAIT) (default value after reset)

Bits 7:0 **MEMSETx:** Common memory x setup time

Defines the number of HCLK () clock cycles to set up the address before the command assertion (NWE, NOE), for PC Card/NAND Flash read or write access to common memory space on socket x:
0000 0000: 1 HCLK cycle (for PC Card) / HCLK cycles (for NAND Flash)
1111 1111: 256 HCLK cycles (for PC Card) / 257 HCLK cycles (for NAND Flash) - (default value after reset)

Attribute memory space timing registers 2..4 (FSMC_PATT2..4)

Address offset: 0xA000 0000 + 0x4C + 0x20 * (x – 1), x = 2..4

Reset value: 0xFCFC FCFC

Each FSMC_PATTx (x = 2..4) read/write register contains the timing information for PC Card/CompactFlash or NAND Flash memory bank x. It is used for 8-bit accesses to the attribute memory space of the PC Card/CompactFlash or to access the NAND Flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to *Section 31.6.5: NAND Flash pre-wait functionality*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ATTHIZx:** Attribute memory x databus HiZ time

Defines the number of HCLK clock cycles during which the databus is kept in HiZ after the start of a PC CARD/NAND Flash write access to attribute memory space on socket x. Only valid for write transaction:
0000 0000: 0 HCLK cycle
1111 1111: 255 HCLK cycles (default value after reset)

Bits 23:16 **ATTHOLDx:** Attribute memory x hold time

Defines the number of HCLK clock cycles to hold address (and data for write access) after the command deassertion (NWE, NOE), for PC Card/NAND Flash read or write access to attribute memory space on socket x
0000 0000: reserved
0000 0001: 1 HCLK cycle
1111 1111: 255 HCLK cycles (default value after reset)

Bits 15:8 **ATTWAITx:** Attribute memory x wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for PC Card/NAND Flash read or write access to attribute memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
0000 0000: reserved
0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
1111 1111: 256 HCLK cycles (+ wait cycle introduced by the card deasserting NWAIT) (default value after reset)

Bits 7:0 **ATTSETx:** Attribute memory x setup time

Defines the number of HCLK (+1) clock cycles to set up address before the command assertion (NWE, NOE), for PC CARD/NAND Flash read or write access to attribute memory space on socket x:
0000 0000: 1 HCLK cycle
1111 1111: 256 HCLK cycles (default value after reset)

I/O space timing register 4 (FSMC_PIO4)

Address offset: 0xA000 0000 + 0xB0
Reset value: 0xFCFCFCFC

The FSMC_PIO4 read/write registers contain the timing information used to gain access to the I/O space of the 16-bit PC Card/CompactFlash.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOHIZx								IOHOLDx								IOWAITx								IOSETx							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IOHIZx:** I/O x databus HiZ time

Defines the number of HCLK clock cycles during which the databus is kept in HiZ after the start of a PC Card write access to I/O space on socket x. Only valid for write transaction:
0000 0000: 0 HCLK cycle
1111 1111: 255 HCLK cycles (default value after reset)

Bits 23:16 **IOHOLDx:** I/O x hold time

Defines the number of HCLK clock cycles to hold address (and data for write access) after the command deassertion (NWE, NOE), for PC Card read or write access to I/O space on socket x:
0000 0000: reserved
0000 0001: 1 HCLK cycle
1111 1111: 255 HCLK cycles (default value after reset)

Bits 15:8 **IOWAITx:** I/O x wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (SMNWE, SMNOE), for PC Card read or write access to I/O space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
0000 0000: reserved, do not use this value
0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
1111 1111: 256 HCLK cycles (+ wait cycle introduced by the Card deasserting NWAIT) (default value after reset)

Bits 7:0 **IOSETx:** I/O x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for PC Card read or write access to I/O space on socket x:
0000 0000: 1 HCLK cycle
1111 1111: 256 HCLK cycles (default value after reset)

ECC result registers 2/3 (FSMC_ECCR2/3)

Address offset: 0xA000 0000 + 0x54 + 0x20 * (x – 1), x = 2 or 3

Reset value: 0x0000 0000

These registers contain the current error correction code value computed by the ECC computation modules of the FSMC controller (one module per NAND Flash memory bank). When the CPU reads the data from a NAND Flash memory page at the correct address (refer to *Section 31.6.6: Error correction code computation ECC (NAND Flash)*), the data read from or written to the NAND Flash are processed automatically by ECC computation module. At the end of X bytes read (according to the ECCPS field in the FSMC_PCRx registers), the CPU must read the computed ECC value from the FSMC_ECCx registers, and then verify whether these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it if applicable. The FSMC_ECCRx registers should be cleared after being read by setting the ECCEN bit to zero. For computing a new data block, the ECCEN bit must be set to one.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECCx																															
r																															

Bits 31:0 **ECCx:** ECC result

This field provides the value computed by the ECC computation logic. *Table 199* hereafter describes the contents of these bit fields.

Table 199. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

31.6.9 FSMC register map

The following table summarizes the FSMC registers.

Table 200. FSMC register map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA000 0000	FSMC_BCR1	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG		WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
	Reset value																																
0xA000 0008	FSMC_BCR2	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0010	FSMC_BCR3	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0018	FSMC_BCR4	Reserved												CBURSTRW	Reserved			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID		MTYP		MUXEN	MBKEN
0xA000 0004	FSMC_BTR1	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 000C	FSMC_BTR2	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0014	FSMC_BTR3	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 001C	FSMC_BTR4	Res.		ACCMOD		DATLAT				CLKDIV				BUSTURN				DATAST								ADDHLD				ADDSET			
0xA000 0104	FSMC_BWTR1	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 010C	FSMC_BWTR2	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 0114	FSMC_BWTR3	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 011C	FSMC_BWTR4	Res.		ACCMOD		DATLAT				CLKDIV				Reserved				DATAST								ADDHLD				ADDSET			
0xA000 0060	FSMC_PCR2	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0080	FSMC_PCR3	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP	PBKEN	PWAITEN	Reserved
0xA000 00A0	FSMC_PCR4	Reserved												ECCPS				TAR				TCLR			Res.	ECCEN		PWID		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0064	FSMC_SR2	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 0084	FSMC_SR3	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 00A4	FSMC_SR4	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 0068	FSMC_PMEM2	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0xA000 0088	FSMC_PMEM3	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							

Table 200. FSMC register map (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA000 00A8	FSMC_PMEM4	MEMHIZx								MEMHOLDx								MEMWAITx								MEMSETx							
0xA000 006C	FSMC_PATT2	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 008C	FSMC_PATT3	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 00AC	FSMC_PATT4	ATTHIZx								ATTHOLDx								ATTWAITx								ATTSETx							
0xA000 00B0	FSMC_PIO4	IOHIZx								IOHOLDx								IOWAITx								IOSETx							
0xA000 0074	FSMC_ECCR2	ECCx																															
0xA000 0094	FSMC_ECCR3	ECCx																															

Refer to *Table 1 on page 50* for the register boundary addresses.

32 Debug support (DBG)

32.1 Overview

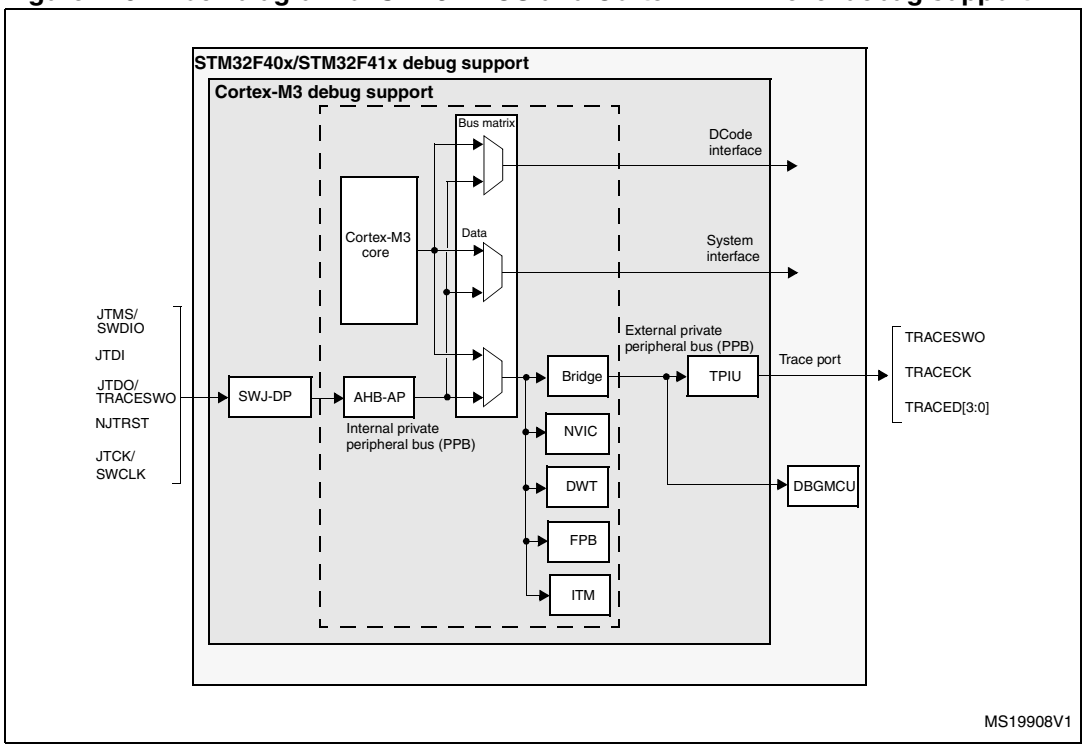
The STM32F40x and STM32F41x are built around a Cortex™-M4F core which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state may be examined. Once examination is complete, the core and the system may be restored and program execution resumed.

The debug features are used by the debugger host when connecting to and debugging the STM32F40x and STM32F41x MCUs.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 410. Block diagram of STM32 MCU and Cortex™-M4F-level debug support



Note: The debug features embedded in the Cortex™-M4F *core are a subset of the ARM CoreSight Design Kit.*

The ARM Cortex™-M4F core provides integrated on-chip debug support. It is comprised of:

- SWJ-DP: Serial wire / JTAG debug port
- AHP-AP: AHB access port
- ITM: Instrumentation trace macrocell
- FPB: Flash patch breakpoint
- DWT: Data watchpoint trigger
- TPUI: Trace port unit interface (available on larger packages, where the corresponding pins are mapped)
- ETM: Embedded Trace Macrocell (available on larger packages, where the corresponding pins are mapped)

It also includes debug features dedicated to the STM32F40x and STM32F41x:

- Flexible debug pinout assignment
- MCU debug box (support for low-power modes, control over peripheral clocks, etc.)

Note: For further information on debug functionality supported by the ARM Cortex™-M4F core, refer to the Cortex™-M4F-r0p1 Technical Reference Manual and to the CoreSight Design Kit-r0p1 TRM (see Section 32.2: Reference ARM documentation).

32.2 Reference ARM documentation

- Cortex™-M4F r0p1 Technical Reference Manual (TRM)
(see Related documents on page 1)
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r0p1 Technical Reference Manual

32.3 SWJ debug port (serial wire and JTAG)

The STM32F40x and STM32F41x core integrates the Serial Wire / JTAG Debug Port (SWJ-DP). It is an ARM standard CoreSight debug port that combines a JTAG-DP (5-pin) interface and a SW-DP (2-pin) interface.

- The JTAG Debug Port (JTAG-DP) provides a 5-pin standard JTAG interface to the AHP-AP port.
- The Serial Wire Debug Port (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP port.

In the SWJ-DP, the two JTAG pins of the SW-DP are multiplexed with some of the five JTAG pins of the JTAG-DP.

Figure 411. SWJ debug port

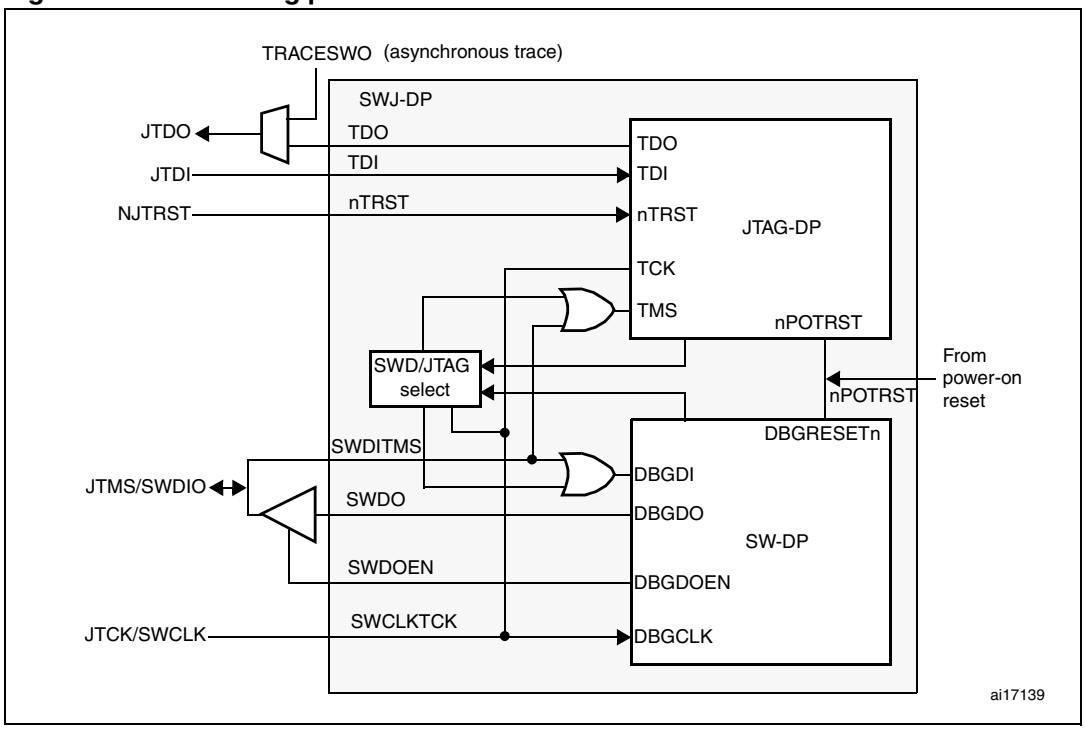


Figure 411 shows that the asynchronous TRACE output (TRACESWO) is multiplexed with TDO. This means that the asynchronous trace can only be used with SW-DP, not JTAG-DP.

32.3.1 Mechanism to select the JTAG-DP or the SW-DP

By default, the JTAG-Debug Port is active.

If the debugger host wants to switch to the SW-DP, it must provide a dedicated JTAG sequence on TMS/TCK (respectively mapped to SWDIO and SWCLK) which disables the JTAG-DP and enables the SW-DP. This way it is possible to activate the SWDP using only the SWCLK and SWDIO pins.

This sequence is:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111100111100111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

32.4 Pinout and debug port pins

The STM32F40x and STM32F41x MCUs are available in various packages with different numbers of available pins. As a result, some functionality (ETM) related to pin availability may differ between packages.

32.4.1 SWJ debug port pins

Five pins are used as outputs from the STM32F40x and STM32F41x for the SWJ-DP as *alternate functions* of general-purpose I/Os. These pins are available on all packages.

Table 201. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

32.4.2 Flexible SWJ-DP pin assignment

After RESET (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are assigned as dedicated pins immediately usable by the debugger host (note that the trace outputs are not assigned except if explicitly programmed by the debugger host).

However, the STM32F40x and STM32F41x MCUs offer the possibility of disabling some or all of the SWJ-DP ports and so, of releasing the associated pins for general-purpose IO (GPIO) usage. For more details on how to disable SWJ-DP port pins, please refer to *Section 6.3.2: I/O pin multiplexer and mapping*.

Table 202. Flexible SWJ-DP pin assignment

Available debug ports	SWJ IO pin assigned				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Full SWJ (JTAG-DP + SW-DP) - Reset State	X	X	X	X	X
Full SWJ (JTAG-DP + SW-DP) but without NJTRST	X	X	X	X	
JTAG-DP Disabled and SW-DP Enabled	X	X			
JTAG-DP Disabled and SW-DP Disabled	Released				

Note: When the APB bridge write buffer is full, it takes one extra APB cycle when writing the GPIO_AFR register. This is because the deactivation of the JTAGSW pins is done in two cycles to guarantee a clean level on the nTRST and TCK input signals of the core.

- Cycle 1: the JTAGSW input signals to the core are tied to 1 or 0 (to 1 for nTRST, TDI and TMS, to 0 for TCK)
- Cycle 2: the GPIO controller takes the control signals of the SWJTAG IO pins (like controls of direction, pull-up/down, Schmitt trigger activation, etc.).

32.4.3 Internal pull-up and pull-down on JTAG pins

It is necessary to ensure that the JTAG input pins are not floating since they are directly connected to flip-flops to control the debug mode features. Special care must be taken with the SWCLK/TCK pin which is directly connected to the clock of some of these flip-flops.

To avoid any uncontrolled IO levels, the device embeds internal pull-ups and pull-downs on the JTAG input pins:

- NJTRST: Internal pull-up
- JTDI: Internal pull-up
- JTMS/SWDIO: Internal pull-up
- TCK/SWCLK: Internal pull-down

Once a JTAG IO is released by the user software, the GPIO controller takes control again. The reset states of the GPIO control registers put the I/Os in the equivalent state:

- NJTRST: Input pull-up
- JTDI: Input pull-up
- JTMS/SWDIO: Input pull-up
- JTCK/SWCLK: Input pull-down
- JTDO: Input floating

The software can then use these I/Os as standard GPIOs.

Note: The JTAG IEEE standard recommends to add pull-ups on TDI, TMS and nTRST but there is no special recommendation for TCK. However, for JTCK, the device needs an integrated pull-down.

Having embedded pull-ups and pull-downs removes the need to add external resistors.

32.4.4 Using serial wire and releasing the unused debug pins as GPIOs

To use the serial wire DP to release some GPIOs, the user software must change the GPIO (PA15, PB3 and PB4) configuration mode in the GPIO_MODER register. This releases PA15, PB3 and PB4 which now become available as GPIOs.

When debugging, the host performs the following actions:

- Under system reset, all SWJ pins are assigned (JTAG-DP + SW-DP).
- Under system reset, the debugger host sends the JTAG sequence to switch from the JTAG-DP to the SW-DP.
- Still under system reset, the debugger sets a breakpoint on vector reset.
- The system reset is released and the Core halts.
- All the debug communications from this point are done using the SW-DP. The other JTAG pins can then be reassigned as GPIOs by the user software.

Note: For user software designs, note that:

To release the debug pins, remember that they will be first configured either in input-pull-up (nTRST, TMS, TDI) or pull-down (TCK) or output tristate (TDO) for a certain duration after reset until the instant when the user software releases the pins.

When debug pins (JTAG or SW or TRACE) are mapped, changing the corresponding IO pin configuration in the IOPORT controller has no effect.

32.5 STM32F40x and STM32F41x JTAG TAP connection

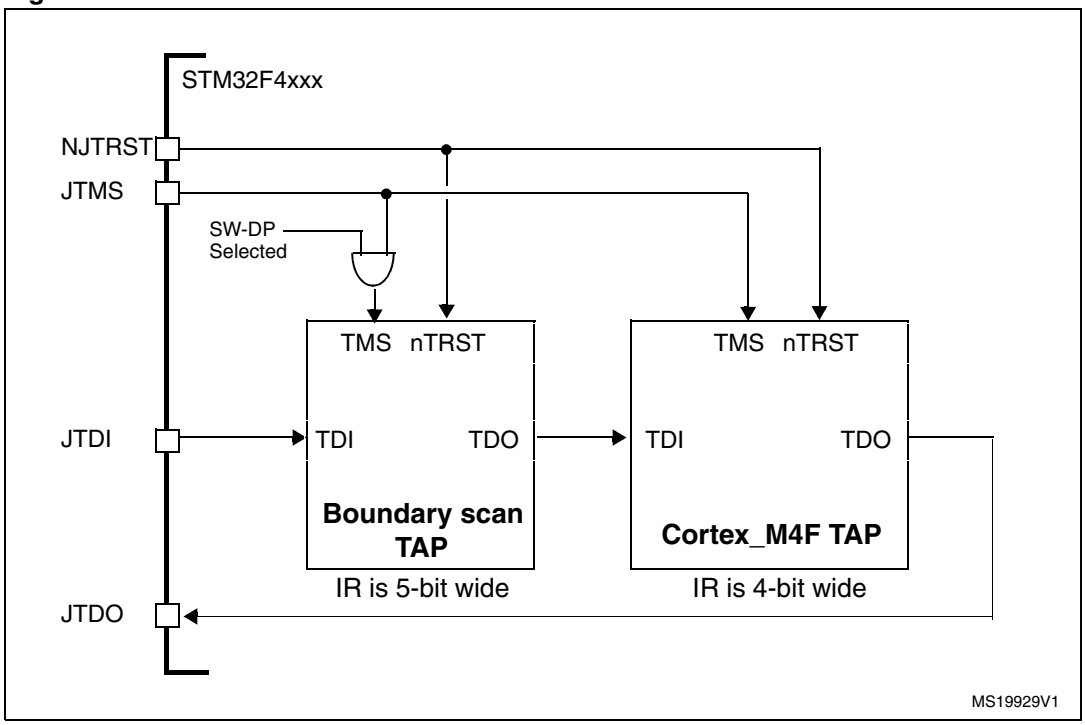
The STM32F40x and STM32F41x MCUs integrate two serially connected JTAG TAPs, the boundary scan TAP (IR is 5-bit wide) and the Cortex™-M4F TAP (IR is 4-bit wide).

To access the TAP of the Cortex™-M4F for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the boundary scan TAP.
2. Then, for each IR shift, the scan chain contains 9 bits (=5+4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

*Note: **Important**: Once Serial-Wire is selected using the dedicated ARM JTAG sequence, the boundary scan TAP is automatically disabled (JTMS forced high).*

Figure 412. JTAG TAP connections



32.6 ID codes and locking mechanism

There are several ID codes inside the STM32F40x and STM32F41x MCUs. ST strongly recommends tools designers to lock their debuggers using the MCU DEVICE ID code located in the external PPB memory map at address 0xE0042000.

32.6.1 MCU device ID code

The STM32F40x and STM32F41x MCUs integrate an MCU ID code. This ID identifies the ST MCU part-number and the die revision. It is part of the DBG_MCU component and is mapped on the external PPB bus (see *Section 32.16 on page 1296*). This code is accessible using the JTAG debug port (4 to 5 pins) or the SW debug port (two pins) or by the user software. It is even accessible while the MCU is under system reset.

DBGMCU_IDCODE

Address: 0xE004 2000

Only 32-bits access supported. Read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DEV_ID											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID(15:0)** Revision identifier
This field indicates the revision of the device:
0x1000 = Revision A

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID(11:0)**: Device identifier
The device ID is 0x413

32.6.2 Boundary scan TAP

JTAG ID code

The TAP of the STM32F40x and STM32F41x BSC (boundary scan) integrates a JTAG ID code equal to 0x06413041.

32.6.3 Cortex™-M4F TAP

The TAP of the ARM Cortex™-M4F integrates a JTAG ID code. This ID code is the ARM default one and has not been modified. This code is only accessible by the JTAG Debug Port.
This code is **0x4BA00477** (corresponds to Cortex™-M4F r0p1, see *Section 32.2: Reference ARM documentation*).

Only the DEV_ID(11:0) should be used for identification by the debugger/programmer tools.

32.6.4 Cortex™-M4F JEDEC-106 ID code

The ARM Cortex™-M4F integrates a JEDEC-106 ID code. It is located in the 4KB ROM table mapped on the internal PPB bus at address 0xE00FF000_0xE00FFFFF.

This code is accessible by the JTAG Debug Port (4 to 5 pins) or by the SW Debug Port (two pins) or by the user software.

32.7 JTAG debug port

A standard JTAG state machine is implemented with a 4-bit instruction register (IR) and five data registers (for full details, refer to the Cortex™-M4Fr0p1 Technical Reference Manual (TRM), for references, please see *Section 32.2: Reference ARM documentation*).

Table 203. JTAG debug port data registers

IR(3:0)	Data register	Details
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	*ID CODE* 0x4BA00477 (ARM Cortex™-M4F r0p1 ID Code)

Table 203. JTAG debug port data registers (continued)

IR(3:0)	Data register	Details
1010	DPACC [35 bits]	*Debug port access register* This initiates a debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved Refer to *Table 204* for a description of the A(3:2) bits
1011	APACC [35 bits]	*Access port access register* Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit address (sub-address AP registers). Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT OTHER = reserved There are many AP Registers (see AHB-AP) addressed as the combination of: – The shifted value A[3:2] – The current value of the DP SELECT register
1000	ABORT [35 bits]	*Abort register* – Bits 31:1 = Reserved – Bit 0 = DAPABORT: write 1 to generate a DAP abort.

Table 204. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x0	00	Reserved, must be kept at reset value.
0x4	01	DP CTRL/STAT register. Used to: – Request a system or debug power-up – Configure the transfer operation for AP accesses – Control the pushed compare and pushed verify operations. – Read some status flags (overrun, power-up acknowledges)

Table 204. 32-bit debug port registers addressed through the shifted value A[3:2]

Address	A(3:2) value	Description
0x8	10	DP SELECT register: Used to select the current access port and the active 4-words register window. – Bits 31:24: APSEL: select the current AP – Bits 23:8: reserved – Bits 7:4: APBANKSEL: select the active 4-words register window on the current AP – Bits 3:0: reserved
0xC	11	DP RDBUFF register: Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation)

32.8 SW debug port

32.8.1 SW protocol introduction

This synchronous serial protocol uses two pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional

The protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written to.

Bits are transferred LSB-first on the wire.

For SWDIO bidirectional management, the line must be pulled-up on the board (100 KΩ recommended by ARM).

Each time the direction of SWDIO changes in the protocol, a turnaround time is inserted where the line is not driven by the host nor the target. By default, this turnaround time is one bit time, however this can be adjusted by configuring the SWCLK frequency.

32.8.2 SW protocol sequence

Each sequence consist of three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target
3. Data transfer phase (33 bits) transmitted by the host or the target

Table 205. Packet request (8-bits)

Bit	Name	Description
0	Start	Must be “1”
1	APnDP	0: DP Access 1: AP Access
2	RnW	0: Write Request 1: Read Request

Table 205. Packet request (8-bits) (continued)

Bit	Name	Description
4:3	A(3:2)	Address field of the DP or AP registers (refer to *Table 204*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by the host. Must be read as “1” by the target because of the pull-up

Refer to the Cortex™-M4Fr0p1 *TRM* for a detailed description of DPACC and APACC registers.

The packet request is always followed by the turnaround time (default 1 bit) where neither the host nor target drive the line.

Table 206. ACK response (3 bits)

Bit	Name	Description
0..2	ACK	001: FAULT 010: WAIT 100: OK

The ACK Response must be followed by a turnaround time only if it is a READ transaction or if a WAIT or FAULT acknowledge has been received.

Table 207. DATA transfer (33 bits)

Bit	Name	Description
0..31	WDATA or RDATA	Write or Read data
32	Parity	Single parity of the 32 data bits

The DATA transfer must be followed by a turnaround time only if it is a READ transaction.

32.8.3 SW-DP state machine (reset, idle states, ID code)

The State Machine of the SW-DP has an internal ID code which identifies the SW-DP. It follows the JEP-106 standard. This ID code is the default ARM one and is set to **0x2BA01477** (corresponding to Cortex™-M4F r0p1).

Note: Note that the SW-DP state machine is inactive until the target reads this ID code.

- The SW-DP state machine is in RESET STATE either after power-on reset, or after the DP has switched from JTAG to SWD or after the line is high for more than 50 cycles
- The SW-DP state machine is in IDLE STATE if the line is low for at least two cycles after RESET state.
- After RESET state, it is **mandatory** to first enter into an IDLE state AND to perform a READ access of the DP-SW ID CODE register. Otherwise, the target will issue a FAULT acknowledge response on another transactions.

Further details of the SW-DP state machine can be found in the *Cortex™-M4F* r0p1 *TRM* and the *CoreSight Design Kit r0p1 TRM*.

32.8.4 DP and AP read/write accesses

- Read accesses to the DP are not posted: the target response can be immediate (if ACK=OK) or can be delayed (if ACK=WAIT).
- Read accesses to the AP are posted. This means that the result of the access is returned on the next transfer. If the next access to be done is NOT an AP access, then the DP-RDBUFF register must be read to obtain the result.
  The READOK flag of the DP-CTRL/STAT register is updated on every AP read access or RDBUFF read request to know if the AP read access was successful.
- The SW-DP implements a write buffer (for both DP or AP writes), that enables it to accept a write operation even when other transactions are still outstanding. If the write buffer is full, the target acknowledge response is “WAIT”. With the exception of IDCODE read or CTRL/STAT read or ABORT write which are accepted even if the write buffer is full.
- Because of the asynchronous clock domains SWCLK and HCLK, two extra SWCLK cycles are needed after a write transaction (after the parity bit) to make the write effective internally. These cycles should be applied while driving the line low (IDLE state)
  This is particularly important when writing the CTRL/STAT for a power-up request. If the next transaction (requiring a power-up) occurs immediately, it will fail.

32.8.5 SW-DP registers

Access to these registers are initiated when APnDP=0

Table 208. SW-DP registers

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is not set to ST code. **0x2BA01477** (identifies the SW-DP)
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.

Table 208. SW-DP registers (continued)

A(3:2)	R/W	CTRLSEL bit of SELECT register	Register	Notes
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

32.8.6 SW-AP registers

Access to these registers are initiated when APnDP=1

There are many AP Registers (see AHB-AP) addressed as the combination of:

- The shifted value A[3:2]
- The current value of the DP SELECT register

32.9 AHB-AP (AHB access port) - valid for both JTAG-DP and SW-DP

Features:

- System access is independent of the processor status.
- Either SW-DP or JTAG-DP accesses AHB-AP.
- The AHB-AP is an AHB master into the Bus Matrix. Consequently, it can access all the data buses (Dcode Bus, System Bus, internal and external PPB bus) but the ICode bus.
- Bitband transactions are supported.
- AHB-AP transactions bypass the FPB.

The address of the 32-bits AHP-AP resisters are 6-bits wide (up to 64 words or 256 bytes) and consists of:

c) Bits [7:4] = the bits [7:4] APBANKSEL of the DP SELECT register

d) Bits [3:2] = the 2 address bits of A(3:2) of the 35-bit packet request for SW-DP.

The AHB-AP of the Cortex™-M4F includes 9 x 32-bits registers:

Table 209. Cortex™-M4F AHB-AP registers

Address offset	Register name	Notes
0x00	AHB-AP Control and Status Word	Configures and controls transfers through the AHB interface (size, hprot, status on current transfer, address increment type
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data 0	Directly maps the 4 aligned data words without rewriting the Transfer Address Register.
0x14	AHB-AP Banked Data 1	
0x18	AHB-AP Banked Data 2	
0x1C	AHB-AP Banked Data 3	
0xF8	AHB-AP Debug ROM Address	Base Address of the debug interface
0xFC	AHB-AP ID Register	

Refer to the *Cortex™-M4F r0p1 TRM* for further details.

32.10 Core debug

Core debug is accessed through the core debug registers. Debug access to these registers is by means of the *Advanced High-performance Bus* (AHB-AP) port. The processor can access these registers directly over the internal *Private Peripheral Bus* (PPB).

It consists of 4 registers:

Table 210. Core debug registers

Register	Description
DHCSR	*The 32-bit Debug Halting Control and Status Register* This provides status information about the state of the processor enable core debug halt and step the processor
DCRSR	*The 17-bit Debug Core Register Selector Register:* This selects the processor register to transfer data to or from.
DCRDR	*The 32-bit Debug Core Register Data Register:* This holds data for reading and writing registers to and from the processor selected by the DCRSR (Selector) register.
DEMCR	*The 32-bit Debug Exception and Monitor Control Register:* This provides Vector Catching and Debug Monitor Control. This register contains a bit named ***TRCENA*** which enable the use of a TRACE.

Note: ***Important****: these registers are not reset by a system reset. They are only reset by a power-on reset.*

Refer to the *Cortex™-M4F r0p1 TRM* for further details.

To Halt on reset, it is necessary to:

- enable the bit0 (VC_CORRESET) of the Debug and Exception Monitor Control Register
- enable the bit0 (C_DEBUGEN) of the Debug Halting Control and Status Register.

32.11 Capability of the debugger host to connect under system reset

The STM32F40x and STM32F41x MCUs' reset system comprises the following reset sources:

- POR (power-on reset) which asserts a RESET at each power-up.
- Internal watchdog reset
- Software reset
- External reset

The Cortex™-M4F differentiates the reset of the debug part (generally PORRESETn) and the other one (SYSRESETn)

This way, it is possible for the debugger to connect under System Reset, programming the Core Debug Registers to halt the core when fetching the reset vector. Then the host can release the system reset and the core will immediately halt without having executed any instructions. In addition, it is possible to program any debug features under System Reset.

Note: It is highly recommended for the debugger host to connect (set a breakpoint in the reset vector) under system reset.

32.12 FPB (Flash patch breakpoint)

The FPB unit:

- implements hardware breakpoints
- patches code and data from code space to system space. This feature gives the possibility to correct software bugs located in the Code Memory Space.

The use of a Software Patch or a Hardware Breakpoint is exclusive.

The FPB consists of:

- 2 literal comparators for matching against literal loads from Code Space and remapping to a corresponding area in the System Space.
- 6 instruction comparators for matching against instruction fetches from Code Space. They can be used either to remap to a corresponding area in the System Space or to generate a Breakpoint Instruction to the core.

32.13 DWT (data watchpoint trigger)

The DWT unit consists of four comparators. They are configurable as:

- a hardware watchpoint or
- a trigger to an ETM or
- a PC sampler or
- a data address sampler

The DWT also provides some means to give some profiling informations. For this, some counters are accessible to give the number of:

- Clock cycle
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- CPI (clock per instructions)
- Interrupt overhead

32.14 ITM (instrumentation trace macrocell)

32.14.1 General description

The ITM is an application-driven trace source that supports *printf* style debugging to trace *Operating System* (OS) and application events, and emits diagnostic system information. The ITM emits trace information as packets which can be generated as:

- **Software trace.** Software can write directly to the ITM stimulus registers to emit packets.
- **Hardware trace.** The DWT generates these packets, and the ITM emits them.
- **Time stamping.** Timestamps are emitted relative to packets. The ITM contains a 21-bit counter to generate the timestamp. The Cortex™-M4F clock or the bit clock rate of the *Serial Wire Viewer* (SWV) output clocks the counter.

The packets emitted by the ITM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to TPIU) and then output the complete packets sequence to the debugger host.

The bit TRCEN of the Debug Exception and Monitor Control Register must be enabled before you program or use the ITM.

32.14.2 Time stamp packets, synchronization and overflow packets

Time stamp packets encode time stamp information, generic control and synchronization. It uses a 21-bit timestamp counter (with possible prescalers) which is reset at each time stamp packet emission. This counter can be either clocked by the CPU clock or the SWV clock.

A synchronization packet consists of 6 bytes equal to 0x80_00_00_00_00_00 which is emitted to the TPIU as 00 00 00 00 00 80 (LSB emitted first).

A synchronization packet is a timestamp packet control. It is emitted at each DWT trigger.

For this, the DWT must be configured to trigger the ITM: the bit CYCCNTENA (bit0) of the DWT Control Register must be set. In addition, the bit2 (SYNCENA) of the ITM Trace Control Register must be set.

Note: If the SYNENA bit is not set, the DWT generates Synchronization triggers to the TPIU which will send only TPIU synchronization packets and not ITM synchronization packets.

An overflow packet consists is a special timestamp packets which indicates that data has been written but the FIFO was full.

Table 211. Main ITM registers

Address	Register	Details
@E0000FB0	ITM lock access	Write 0xC5ACCE55 to unlock Write Access to the other ITM registers
@E0000E80	ITM trace control	Bits 31-24 = Always 0
		Bits 23 = Busy
		Bits 22-16 = 7-bits ATB ID which identifies the source of the trace data.
		Bits 15-10 = Always 0
		Bits 9:8 = TSPrescale = Time Stamp Prescaler
		Bits 7-5 = Reserved
		Bit 4 = SWOENA = Enable SWV behavior (to clock the timestamp counter by the SWV clock).
		Bit 3 = DWTENA: Enable the DWT Stimulus
		Bit 2 = SYNCENA: this bit must be to 1 to enable the DWT to generate synchronization triggers so that the TPIU can then emit the synchronization packets.
		Bit 1 = TSENA (Timestamp Enable)
		Bit 0 = ITMENA: Global Enable Bit of the ITM
@E0000E40	ITM trace privilege	Bit 3: mask to enable tracing ports31:24
		Bit 2: mask to enable tracing ports23:16
		Bit 1: mask to enable tracing ports15:8
		Bit 0: mask to enable tracing ports7:0
@E0000E00	ITM trace enable	Each bit enables the corresponding Stimulus port to generate trace.
@E0000000-E000007C	Stimulus port registers 0-31	Write the 32-bits data on the selected Stimulus Port (32 available) to be traced out.

Example of configuration

To output a simple value to the TPIU:

- Configure the TPIU and assign TRACE I/Os by configuring the DBGMCU_CR (refer to *Section 32.17.2: TRACE pin assignment* and *Section 32.16.3: Debug MCU configuration register*)
- Write 0xC5ACCE55 to the ITM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00010005 to the ITM Trace Control Register to enable the ITM with Sync enabled and an ATB ID different from 0x00
- Write 0x1 to the ITM Trace Enable Register to enable the Stimulus Port 0
- Write 0x1 to the ITM Trace Privilege Register to unmask stimulus ports 7:0
- Write the value to output in the Stimulus Port Register 0: this can be done by software (using a printf function)

32.15 ETM (Embedded trace macrocell)

32.15.1 General description

The ETM enables the reconstruction of program execution. Data are traced using the Data Watchpoint and Trace (DWT) component or the Instruction Trace Macrocell (ITM) whereas instructions are traced using the Embedded Trace Macrocell (ETM).

The ETM transmits information as packets and is triggered by embedded resources. These resources must be programmed independently and the trigger source is selected using the Trigger Event Register (0xE0041008). An event could be a simple event (address match from an address comparator) or a logic equation between 2 events. The trigger source is one of the fourth comparators of the DWT module, The following events can be monitored:

- Clock cycle matching
- Data address matching

For more informations on the trigger resources refer to *Section 32.13: DWT (data watchpoint trigger)*.

The packets transmitted by the ETM are output to the TPIU (Trace Port Interface Unit). The formatter of the TPIU adds some extra packets (refer to *Section 32.17: TPIU (trace port interface unit)*) and then outputs the complete packet sequence to the debugger host.

32.15.2 Signal protocol, packet types

This part is described in the chapter 7 ETMv3 Signal Protocol of the ARM IHI 0014N document.

32.15.3 Main ETM registers

For more information on registers refer to the chapter 3 of the ARM IHI 0014N specification.

Table 212. Main ETM registers

Address	Register	Details
0xE0041FB0	ETM Lock Access	Write 0xC5ACCE55 to unlock the write access to the other ETM registers.
0xE0041000	ETM Control	This register controls the general operation of the ETM, for instance how tracing is enabled.
0xE0041010	ETM Status	This register provides information about the current status of the trace and trigger logic.
0xE0041008	ETM Trigger Event	This register defines the event that will control trigger.
0xE004101C	ETM Trace Enable Control	This register defines which comparator is selected.
0xE0041020	ETM Trace Enable Event	This register defines the trace enabling event.
0xE0041024	ETM Trace Start/Stop	This register defines the traces used by the trigger source to start and stop the trace, respectively.

32.15.4 Configuration example

To output a simple value to the TPIU:

- Configure the TPIU and enable the I/IO_TRACEN to assign TRACE I/Os in the STM32F40x and STM32F41x debug configuration register.
- Write 0xC5ACCE55 to the ETM Lock Access Register to unlock the write access to the ITM registers
- Write 0x00001D1E to the control register (configure the trace)
- Write 0000406F to the Trigger Event register (define the trigger event)
- Write 0000006F to the Trace Enable Event register (define an event to start/stop)
- Write 00000001 to the Trace Start/stop register (enable the trace)
- Write 0000191E to the ETM Control Register (end of configuration)

32.16 MCU debug component (DBGMCU)

The MCU debug component helps the debugger provide support for:

- Low-power modes
- Clock control for timers, watchdog, I2C and bxCAN during a breakpoint
- Control of the trace pins assignment

32.16.1 Debug support for low-power modes

To enter low-power mode, the instruction WFI or WFE must be executed.

The MCU implements several low-power modes which can either deactivate the CPU clock or reduce the power of the CPU.

The core does not allow FCLK or HCLK to be turned off during a debug session. As these are required for the debugger connection, during a debug, they must remain active. The MCU integrates special means to allow the user to debug software in low-power modes.

For this, the debugger host must first set some debug configuration registers to change the low-power mode behavior:

- In Sleep mode, DBG_SLEEP bit of DBGMCU_CR register must be previously set by the debugger. This will feed HCLK with the same clock that is provided to FCLK (system clock previously configured by the software).
- In Stop mode, the bit DBG_STOP must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in STOP mode.

32.16.2 Debug support for timers, watchdog, bxCAN and I^2C

During a breakpoint, it is necessary to choose how the counter of timers and watchdog should behave:

- They can continue to count inside a breakpoint. This is usually required when a PWM is controlling a motor, for example.
- They can stop to count inside a breakpoint. This is required for watchdog purposes.

For the bxCAN, the user can choose to block the update of the receive register during a breakpoint.

For the I^2C, the user can choose to block the SMBUS timeout during a breakpoint.

32.16.3 Debug MCU configuration register

This register allows the configuration of the MCU under DEBUG. This concerns:

- Low-power mode support
- Timer and watchdog counter support
- bxCAN communication support
- Trace pin assignment

This DBGMCU_CR is mapped on the External PPB bus at address 0xE0042004

It is asynchronously reset by the PORESET (and not the system reset). It can be written by the debugger under system reset.

If the debugger host does not support these features, it is still possible for the user software to write to these registers.

DBGMCU_CR

Address: 0xE004 2004

Only 32-bit access supported

POR Reset: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw			rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:5 **TRACE_MODE[1:0] and TRACE_IOEN**: Trace pin assignment control

– *With TRACE_IOEN=0:*

TRACE_MODE=xx: TRACE pins not assigned (default state)

– *With TRACE_IOEN=1:*

- TRACE_MODE=00: TRACE pin assignment for Asynchronous Mode
- TRACE_MODE=01: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 1
- TRACE_MODE=10: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 2
- TRACE_MODE=11: TRACE pin assignment for Synchronous Mode with a TRACEDATA size of 4

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY:** Debug Standby mode

0: (FCLK=Off, HCLK=Off) The whole digital part is unpowered.
From software point of view, exiting from Standby is identical than fetching reset vector (except a few status bit indicated that the MCU is resuming from Standby)
1: (FCLK=On, HCLK=On) In this case, the digital part is not unpowered and FCLK and HCLK are provided by the internal RC oscillator which remains active. In addition, the MCU generate a system reset during Standby mode so that exiting from Standby is identical than fetching from reset

Bit 1 **DBG_STOP:** Debug Stop mode

0: (FCLK=Off, HCLK=Off) In STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is identical to the one after RESET (CPU clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable the PLL, the Xtal, etc.
1: (FCLK=On, HCLK=On) In this case, when entering STOP mode, FCLK and HCLK are provided by the internal RC oscillator which remains active in STOP mode. When exiting STOP mode, the software must reprogram the clock controller to enable the PLL, the Xtal, etc. (in the same way it would do in case of DBG_STOP=0)

Bit 0 **DBG_SLEEP:** Debug Sleep mode

0: (FCLK=On, HCLK=Off) In Sleep mode, FCLK is clocked by the system clock as previously configured by the software while HCLK is disabled.
In Sleep mode, the clock controller configuration is not reset and remains in the previously programmed state. Consequently, when exiting from Sleep mode, the software does not need to reconfigure the clock controller.
1: (FCLK=On, HCLK=On) In this case, when entering Sleep mode, HCLK is fed by the same clock that is provided to FCLK (system clock as previously configured by the software).

32.16.4 Debug MCU APB1 freeze register (DBGMCU_APB1_FZ)

The DBGMCU_APB1_FZ register is used to configure the MCU under Debug. It concerns APB1 peripherals. It is mapped on the external PPB bus at address 0xE004 2008.

The register is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address : 0xE004 2008

Only 32-bits access are supported.

Power on reset (POR): 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved				
					rw	rw		rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
				rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **DBG_CAN2_STOP:** Debug CAN2 stopped when Core is halted
0: Same behavior as in normal mode
1: The CAN2 receive registers are frozen

Bit 25 **DBG_CAN1_STOP:** Debug CAN2 stopped when Core is halted
0: Same behavior as in normal mode
1: The CAN2 receive registers are frozen

Bit 24 Reserved, must be kept at reset value.

Bit 23 **DBG_I2C3_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bit 22 **DBG_I2C2_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bit 21 **DBG_I2C1_SMBUS_TIMEOUT:** SMBUS timeout mode stopped when Core is halted
0: Same behavior as in normal mode
1: The SMBUS timeout is frozen

Bit 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP:** Debug independent watchdog stopped when core is halted
0: The independent watchdog counter clock continues even if the core is halted
1: The independent watchdog counter clock is stopped when the core is halted

Bit 11 **DBG_WWDG_STOP:** Debug Window Watchdog stopped when Core is halted
0: The window watchdog counter clock continues even if the core is halted
1: The window watchdog counter clock is stopped when the core is halted

Bit 10 **DBG_RTC_STOP:** RTC stopped when Core is halted
0: The RTC counter clock continues even if the core is halted
1: The RTC counter clock is stopped when the core is halted

Bit 9 Reserved, must be kept at reset value.

Bits 8:0 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=2..7, 12..14)
0: The clock of the involved Timer Counter is fed even if the core is halted
1: The clock of the involved Timer counter is stopped when the core is halted

32.16.5 Debug MCU APB2 Freeze register (DBGMCU_APB2_FZ)

The DBGMCU_APB2_FZ register is used to configure the MCU under Debug. It concerns APB2 peripherals.

This register is mapped on the external PPB bus at address 0xE004 200C

It is asynchronously reset by the POR (and not the system reset). It can be written by the debugger under system reset.

Address: 0xE004 200C

Only 32-bit access is supported.

POR: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DBG_TIM8_STOP	DBG_TIM1_STOP
														rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **DBG_TIMx_STOP:** TIMx counter stopped when core is halted (x=9..11)
0: The clock of the involved Timer Counter is fed even if the core is halted
1: The clock of the involved Timer counter is stopped when the core is halted

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM8_STOP**: TIM8 counter stopped when core is halted
0: The clock of the involved Timer Counter is fed even if the core is halted
1: The clock of the involved Timer counter is stopped when the core is halted

Bit 0 **DBG_TIM1_STOP:** TIM1 counter stopped when core is halted
0: The clock of the involved Timer Counter is fed even if the core is halted
1: The clock of the involved Timer counter is stopped when the core is halted

32.17 TPIU (trace port interface unit)

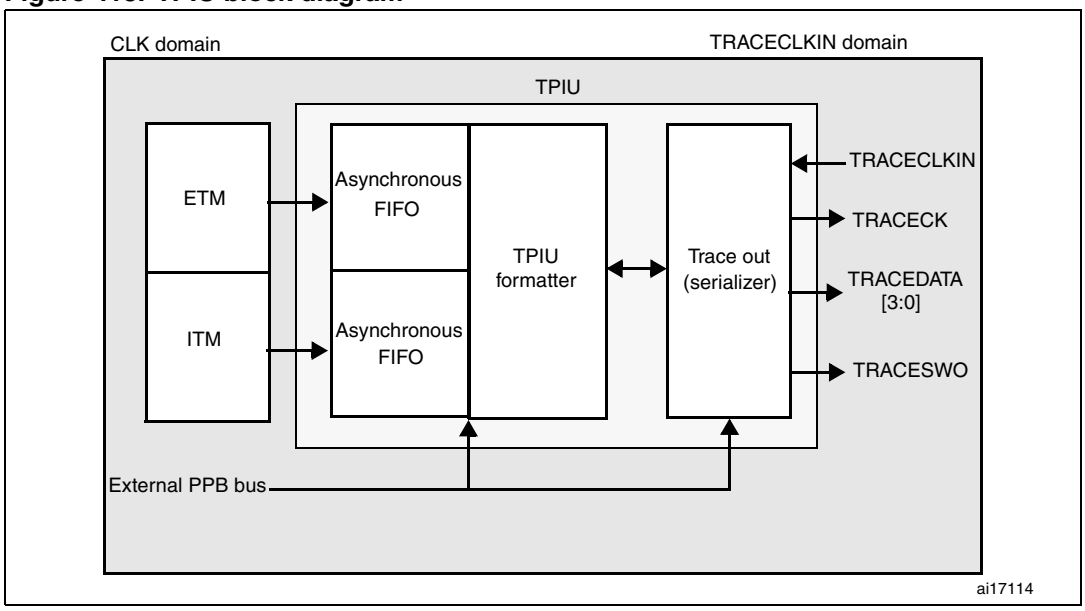
32.17.1 Introduction

The TPIU acts as a bridge between the on-chip trace data from the ITM and the ETM.

The output data stream encapsulates the trace source ID, that is then captured by a *trace port analyzer* (TPA).

The core embeds a simple TPIU, especially designed for low-cost debug (consisting of a special version of the CoreSight TPIU).

Figure 413. TPIU block diagram



32.17.2 TRACE pin assignment

- Asynchronous mode

  The asynchronous mode requires 1 extra pin and is available on all packages. It is only available if using Serial Wire mode (not in JTAG mode).

Table 213. Asynchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F40x and STM32F41x pin assignment
	Type	Description	
TRACESWO	O	TRACE Async Data Output	PB3

- Synchronous mode

  The synchronous mode requires from 2 to 6 extra pins depending on the data trace size and is only available in the larger packages. In addition it is available in JTAG mode and in Serial Wire mode and provides better bandwidth output capabilities than asynchronous trace.

Table 214. Synchronous TRACE pin assignment

TPUI pin name	Trace synchronous mode		STM32F40x and STM32F41x pin assignment
	Type	Description	
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

TPUI TRACE pin assignment

By default, these pins are NOT assigned. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the ***MCU Debug component configuration register***. This configuration has to be done by the debugger host.

In addition, the number of pins to assign depends on the trace configuration (asynchronous or synchronous).

- **Asynchronous mode**: 1 extra pin is needed
- **Synchronous mode**: from 2 to 5 extra pins are needed depending on the size of the data trace port register (1, 2 or 4):
  - TRACECK
  - TRACED(0) if port size is configured to 1, 2 or 4
  - TRACED(1) if port size is configured to 2 or 4
  - TRACED(2) if port size is configured to 4
  - TRACED(3) if port size is configured to 4

To assign the TRACE pin, the debugger host must program the bits TRACE_IOEN and TRACE_MODE[1:0] of the Debug MCU configuration Register (DBGMCU_CR). By default the TRACE pins are not assigned.

This register is mapped on the external PPB and is reset by the PORESET (and not by the SYSTEM reset). It can be written by the debugger under SYSTEM reset.

Table 215. Flexible TRACE pin assignment

DBGMCU_CR register		Pins assigned for:	TRACE IO pin assigned					
TRACE_ IOEN	TRACE_ MODE[1:0]		PB3 / JTDO/ TRACESWO	PE2 / TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	No Trace (default state)	Released (1)					
1	00	Asynchronous Trace	TRACESWO			Released (usable as GPIO)		
1	01	Synchronous Trace 1 bit	Released (1)	TRACECK	TRACED[0]			
1	10	Synchronous Trace 2 bit		TRACECK	TRACED[0]	TRACED[1]		
1	11	Synchronous Trace 4 bit		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. When Serial Wire mode is used, it is released. But when JTAG is used, it is assigned to JTDO.

Note: By default, the TRACECLKIN input clock of the TPIU is tied to GND. It is assigned to HCLK two clock cycles after the bit TRACE_IOEN has been set.

The debugger must then program the Trace Mode by writing the PROTOCOL[1:0] bits in the SPP_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Trace Port Mode (synchronous)
- PROTOCOL=01 or 10: Serial Wire (Manchester or NRZ) Mode (asynchronous mode). Default state is 01

It then also configures the TRACE port size by writing the bits [3:0] in the CPSPS_R (Current Sync Port Size Register) of the TPIU:

- 0x1 for 1 pin (default state)
- 0x2 for 2 pins
- 0x8 for 4 pins

32.17.3 TPUI formatter

The formatter protocol outputs data in 16-byte frames:

- seven bytes of data
- eight bytes of mixed-use bytes consisting of:
  - 1 bit (LSB) to indicate it is a DATA byte (‘0) or an ID byte (‘1).
  - 7 bits (MSB) which can be data or change of source ID trace.
- one byte of auxiliary bits where each bit corresponds to one of the eight mixed-use bytes:
  - if the corresponding byte was a data, this bit gives bit0 of the data.
  - if the corresponding byte was an ID change, this bit indicates when that ID change takes effect.

Note: Refer to the ARM CoreSight Architecture Specification v1.0 (ARM IHI 0029B) for further information

32.17.4 TPUI frame synchronization packets

The TPUI can generate two types of synchronization packets:

- The Frame Synchronization packet (or Full Word Synchronization packet)

  It consists of the word: 0x7F_FF_FF_FF (LSB emitted first). This sequence can not occur at any other time provided that the ID source code 0x7F has not been used.

  It is output periodically ***between*** frames.

  In continuous mode, the TPA must discard all these frames once a synchronization frame has been found.
- The Half-Word Synchronization packet

  It consists of the half word: 0x7F_FF (LSB emitted first).

  It is output periodically ***between or within*** frames.

  These packets are only generated in continuous mode and enable the TPA to detect that the TRACE port is in IDLE mode (no TRACE to be captured). When detected by the TPA, it must be discarded.

32.17.5 Transmission of the synchronization frame packet

There is no Synchronization Counter register implemented in the TPIU of the core. Consequently, the synchronization trigger can only be generated by the **DWT**. Refer to the registers DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI Frame synchronization packet (0x7F_FF_FF_FF) is emitted:

- after each TPIU reset release. This reset is synchronously released with the rising edge of the TRACECLKIN clock. This means that this packet is transmitted when the TRACE_IOEN bit in the DBGMCU_CFG register is set. In this case, the word 0x7F_FF_FF_FF is not followed by any formatted packet.
- at each DWT trigger (assuming DWT has been previously configured). Two cases occur:
  - If the bit SYNENA of the ITM is reset, only the word 0x7F_FF_FF_FF is emitted without any formatted stream which follows.
  - If the bit SYNENA of the ITM is set, then the ITM synchronization packets will follow (0x80_00_00_00_00_00), formatted by the TPUI (trace source ID added).

32.17.6 Synchronous mode

The trace data output size can be configured to 4, 2 or 1 pin: TRACED(3:0)

The output clock is output to the debugger (TRACECK)

Here, TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

Note: In this synchronous mode, it is not required to provide a stable clock frequency.

The TRACE I/Os (including TRACECK) are driven by the rising edge of TRACLKIN (equal to HCLK). Consequently, the output frequency of TRACECK is equal to HCLK/2.

32.17.7 Asynchronous mode

This is a low cost alternative to output the trace using only 1 pin: this is the asynchronous output pin TRACESWO. Obviously there is a limited bandwidth.

TRACESWO is multiplexed with JTDO when using the SW-DP pin. This way, this functionality is available in all STM32F40x and STM32F41x packages.

This asynchronous mode requires a constant frequency for TRACECLKIN. For the standard UART (NRZ) capture mechanism, 5% accuracy is needed. The Manchester encoded version is tolerant up to 10%.

32.17.8 TRACECLKIN connection inside the STM32F40x and STM32F41x

In the STM32F40x and STM32F41x, this TRACECLKIN input is internally connected to HCLK. This means that when in asynchronous trace mode, the application is restricted to use to time frames where the CPU frequency is stable.

Note: ***Important:*** *when using asynchronous trace: it is important to be aware that:*

The default clock of the STM32F40x and STM32F41x MCUs is the internal RC oscillator. Its frequency under reset is different from the one after reset release. This is because the RC calibration is the default one under system reset and is updated at each system reset release.

Consequently, the trace port analyzer (TPA) should not enable the trace (with the TRACE_IOEN bit) under system reset, because a Synchronization Frame Packet will be issued with a different bit time than trace packets which will be transmitted after reset release.

32.17.9 TPIU registers

The TPIU APB registers can be read and written only if the bit TRCENA of the Debug Exception and Monitor Control Register (DEMCR) is set. Otherwise, the registers are read as zero (the output of this bit enables the PCLK of the TPIU).

Table 216. Important TPIU registers

Address	Register	Description
0xE0040004	Current port size	Allows the trace port size to be selected: Bit 0: Port size = 1 Bit 1: Port size = 2 Bit 2: Port size = 3, not supported Bit 3: Port Size = 4 Only 1 bit must be set. By default, the port size is one bit. (0x00000001)
0xE00400F0	Selected pin protocol	Allows the Trace Port Protocol to be selected: Bit1:0= 00: Sync Trace Port Mode 01: Serial Wire Output - manchester (default value) 10: Serial Wire Output - NRZ 11: reserved
0xE0040304	Formatter and flush control	Bit 31-9 = always ‘0 Bit 8 = TrigIn = always ‘1 to indicate that triggers are indicated Bit 7-4 = always 0 Bit 3-2 = always 0 Bit 1 = EnFCont. In Sync Trace mode (Select_Pin_Protocol register bit1:0=00), this bit is forced to ‘1: the formatter is automatically enabled in continuous mode. In asynchronous mode (Select_Pin_Protocol register bit1:0 <> 00), this bit can be written to activate or not the formatter. Bit 0 = always 0 The resulting default value is 0x102 **Note:** In synchronous mode, because the TRACECTL pin is not mapped outside the chip, the formatter is always enabled in continuous mode -this way the formatter inserts some control packets to identify the source of the trace packets).
0xE0040300	Formatter and flush status	Not used in Cortex™-M4F, always read as 0x00000008

32.17.10 Example of configuration

- Set the bit TRCENA in the Debug Exception and Monitor Control Register (DEMCR)
- Write the TPIU Current Port Size Register to the desired value (default is 0x1 for a 1-bit port size)
- Write TPIU Formatter and Flush Control Register to 0x102 (default value)
- Write the TPIU Select Pin Protocol to select the sync or async mode. Example: 0x2 for async NRZ mode (UART like)
- Write the DBGMCU control register to 0x20 (bit IO_TRACEN) to assign TRACE I/Os for async mode. A TPIU Sync packet is emitted at this time (FF_FF_FF_7F)
- Configure the ITM and write the ITM Stimulus register to output a value

32.18 DBG register map

The following table summarizes the Debug registers

Table 217. DBG register map and reset values

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	**DBGMCU_IDCODE**	REV_ID																Reserved				DEV_ID											
	Reset value[1]	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0xE004 2004	**DBGMCU_CR**	Reserved											DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT	Reserved								TRACE_MODE[1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value												0	0	0	0	0									0	0	0			0	0	0
0xE004 2008	**DBGMCU_APB1_FZ**	Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved								DBG_IWDG_STOP	DBG_WWDG_STOP	Reserved	DBG_RTC_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value						0	0	0	0	0	0									0	0		0	0	0	0	0	0	0	0	0	0
0xE004 200C	**DBGMCU_APB2_FZ**	Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	Reserved													DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value														0	0	0														0	0

1. The reset value is product dependent. For more information, refer to *Section 32.6.1: MCU device ID code*.

33 Device electronic signature

The electronic signature is stored in the System memory area in the Flash memory module, and can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32F40x and STM32F41x microcontroller.

33.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x1FFF 7A10

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(32:0)																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(31:0):** 31:0 unique ID bits

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(63:32):** 63:32 unique ID bits

33.2 Flash size

Base address: 0x1FFF 7A10

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

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

Bits 15:0 F_ID(15:0): Flash memory size

This bitfield indicates the size of the device Flash memory expressed in Kbytes. As an example, 0x4000 corresponds to 1024 Kbytes.

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **U_ID(95:64):** *95:64 Unique ID bits.*

Revision history

Table 218. Document revision history

Date	Version	Changes
16-Sep-2011	1	Initial release.

Index

A

ADC_CCR 245
ADC_CDR 247
ADC_CR1 233
ADC_CR2 235
ADC_CSR 244
ADC_DR 242
ADC_HTR 239
ADC_JDRx 242
ADC_JOFRx 239
ADC_JSQR 241
ADC_LTR 239
ADC_SMPR1 238
ADC_SMPR2 238
ADC_SQR1 240
ADC_SQR2 240
ADC_SQR3 241
ADC_SR 232

C

CAN_BTR 796
CAN_ESR 795
CAN_FA1R 806
CAN_FFA1R 806
CAN_FiRx 807
CAN_FM1R 805
CAN_FMR 804
CAN_FS1R 805
CAN_IER 794
CAN_MCR 786
CAN_MSR 789
CAN_RDHxR 803
CAN_RDLxR 803
CAN_RDTxR 802
CAN_RF0R 792
CAN_RF1R 793
CAN_RIxR 801
CAN_TDHxR 800
CAN_TDLxR 800
CAN_TDTxR 799
CAN_TIxR 798
CAN_TSR 790
CRC_DR 61
CRC_IDR 61
CRYP_CR 501
CRYP_DIN 504
CRYP_DMACR 506
CRYP_DOUT 505
CRYP_IMSCR 506
CRYP_IV0LR 510
CRYP_IV0RR 510
CRYP_IV1LR 511
CRYP_IV1RR 511
CRYP_K0LR 508
CRYP_K0RR 508
CRYP_K1LR 508
CRYP_K1RR 509
CRYP_K2LR 509
CRYP_K2RR 509
CRYP_K3LR 509
CRYP_K3RR 510
CRYP_MISR 507
CRYP_RISR 507
CRYP_SR 503

D

DAC_CR 261
DAC_DHR12L1 265
DAC_DHR12L2 266
DAC_DHR12LD 267
DAC_DHR12R1 264
DAC_DHR12R2 265
DAC_DHR12RD 266
DAC_DHR8R1 265
DAC_DHR8R2 266
DAC_DHR8RD 267
DAC_DOR1 268
DAC_DOR2 268
DAC_SR 268
DAC_SWTRIGR 264
DBGMCU_APB1 1299
DBGMCU_APB2_FZ 1300
DBGMCU_CR 1297
DBGMCU_IDCODE 1284
DCMI_CR 281
DCMI_CWSIZE 290
DCMI_CWSTRT 290
DCMI_DR 291
DCMI_ESCR 287
DCMI_ESUR 288
DCMI_ICR 287
DCMI_IER 285
DCMI_MIS 286
DCMI_RIS 284
DCMI_SR 283

DMA_HIFCR 184
DMA_HISR 182
DMA_LIFCR 183
DMA_LISR 181
DMA_SxCR 185
DMA_SxFCR 190
DMA_SxM0AR 188
DMA_SxM1AR 189
DMA_SxNDTR 188
DMA_SxPAR 188

E

ETH_DMABMR 911
ETH_DMACHRBAR 925
ETH_DMACHRDR 924
ETH_DMACHTBAR 924
ETH_DMACHTDR 924
ETH_DMAIER 921
ETH_DMAMFBOCR 923
ETH_DMAOMR 918
ETH_DMARDLAR 914
ETH_DMARPDR 914
ETH_DMARSWTR 923
ETH_DMASR 915
ETH_DMATDLAR 915
ETH_DMATPDR 913
ETH_MACA0HR 894
ETH_MACA0LR 895
ETH_MACA1HR 895
ETH_MACA1LR 896
ETH_MACA2HR 896
ETH_MACA2LR 897
ETH_MACA3HR 897
ETH_MACA3LR 898
ETH_MACCR 880
ETH_MACDBGR 891
ETH_MACFCR 887
ETH_MACFFR 883
ETH_MACHTHR 884
ETH_MACHTLR 885
ETH_MACIMR 894
ETH_MACMIIAR 885
ETH_MACMIIDR 886
ETH_MACPMTCSR 890
ETH_MACRWUFFR 889
ETH_MACSR 893
ETH_MACVLANTR 888
ETH_MMCCR 899
ETH_MMCRFAECR 904
ETH_MMCRFCECR 903
ETH_MMCRGUFCR 904
ETH_MMCRIMR 901
ETH_MMCRIR 899
ETH_MMCTGFCR 903
ETH_MMCTGFMSCCR 903
ETH_MMCTGFSCCR 902
ETH_MMCTIMR 902
ETH_MMCTIR 900
ETH_PTPPPSCR 911
ETH_PTPSSIR 906
ETH_PTPTSAR 909
ETH_PTPTSCR 904
ETH_PTPTSHR 907
ETH_PTPTSHUR 908
ETH_PTPTSLR 907
ETH_PTPTSLUR 909
ETH_PTPTSSR 910
ETH_PTPTTHR 910
ETH_PTPTTLR 910
EXTI_EMR 203
EXTI_FTSR 204
EXTI_IMR 203
EXTI_PR 205
EXTI_RTSR 204
EXTI_SWIER 205

F

FSMC_BCR1..4 1256
FSMC_BTR1..4 1258
FSMC_BWTR1..4 1260

G

GPIOx_AFRH 153
GPIOx_AFRL 152
GPIOx_BSRR 150
GPIOx_IDR 150
GPIOx_LCKR 151
GPIOx_MODER 148
GPIOx_ODR 150
GPIOx_OSPEEDR 149
GPIOx_OTYPER 148
GPIOx_PUPDR 149

H

HASH_CR 527
HASH_CSRx 534
HASH_DIN 529
HASH_HR0 531
HASH_HR1 531
HASH_HR2 531
HASH_HR3 531

HASH_HR4 532
HASH_IMR 532
HASH_SR 533
HASH_STR 530

I

I2C_CCR 603
I2C_CR1 593
I2C_CR2 595
I2C_DR 598
I2C_OAR1 597
I2C_OAR2 597
I2C_SR1 598
I2C_SR2 602
I2C_TRISE 604
IWDG_KR 469
IWDG_PR 470
IWDG_RLR 470
IWDG_SR 471

O

OTG_FS_CID 978, 1114
OTG_FS_DAINT 996, 1134
OTG_FS_DAINTMSK 997, 1134
OTG_FS_DCFG 991
OTG_FS_DCTL 992, 1129
OTG_FS_DIEPCTL0 999
OTG_FS_DIEPEMPMSK 998, 1137
OTG_FS_DIEPINTx 1007, 1147
OTG_FS_DIEPMSK 994, 1132
OTG_FS_DIEPTSIZ0 . . . . . .1009, 1150
OTG_FS_DIEPTSIZx 1011, 1152
OTG_FS_DIEPTXFx 980, 1114
OTG_FS_DOEPCTL0 . . . . . .1003, 1143
OTG_FS_DOEPCTLx 1004, 1144
OTG_FS_DOEPINTx 1008, 1149
OTG_FS_DOEPMSK 995, 1133
OTG_FS_DOEPTSIZ0 . . . . . .1010, 1151
OTG_FS_DOEPTSIZx 1012, 1153
OTG_FS_DSTS 993, 1131
OTG_FS_DTXFSTSx 1012, 1153
OTG_FS_DVBUSDIS 997, 1135
OTG_FS_DVBUSPULSE 998, 1135
OTG_FS_GAHBCFG 962, 1094
OTG_FS_GCCFG 977, 1113
OTG_FS_GINTMSK 971, 1105
OTG_FS_GINTSTS 967, 1101
OTG_FS_GNPTXFSIZ 976, 1110
OTG_FS_GNPTXSTS 976, 1110
OTG_FS_GOTGCTL 958, 1091
OTG_FS_GOTGINT 960, 1093
OTG_FS_GRSTCTL 965, 1098
OTG_FS_GRXFSIZ 975, 1109
OTG_FS_GRXSTSP 974, 1108
OTG_FS_GRXSTSR 974, 1108
OTG_FS_GUSBCFG 963, 1095
OTG_FS_HAINT 983, 1118
OTG_FS_HAINTMSK 984, 1118
OTG_FS_HCCHARx 987, 1121
OTG_FS_HCFG 980, 1115
OTG_FS_HCINTMSKx 989, 1125
OTG_FS_HCINTx 988, 1124
OTG_FS_HCTSIZx 990, 1126
OTG_FS_HFIR 981, 1116
OTG_FS_HFNUM 982, 1116
OTG_FS_HPRT 984, 1119
OTG_FS_HPTXFSIZ 979, 1114
OTG_FS_HPTXSTS 982, 1117
OTG_FS_PCGCCTL 1013, 1154
OTG_HS_DCFG 1127
OTG_HS_DEACHINTMSK 1138
OTG_HS_DIEPDMAx 1154
OTG_HS_DOEPDMAx 1154
OTG_HS_DTHRCTL 1136
OTG_HS_HCSPLTx 1123

P

PWR_CR 78
PWR_CSR 79

R

RCC_AHB1ENR 110
RCC_AHB1LPENR 119
RCC_AHB1RSTR 102
RCC_AHB2ENR 112
RCC_AHB2LPENR 121
RCC_AHB2RSTR 104
RCC_AHB3ENR 113
RCC_AHB3LPENR 122
RCC_AHB3RSTR 105
RCC_APB1ENR 113
RCC_APB1LPENR 123
RCC_APB1RSTR 105
RCC_APB2ENR 117
RCC_APB2LPENR 126
RCC_APB2RSTR 108
RCC_BDCR 128
RCC_CFGR 97
RCC_CIR 99
RCC_CR 93
RCC_CSR 129
RCC_PLLCFGR 95, 132

RCC_SSCGR 131
RNG_CR 515
RNG_DR 516
RNG_SR 515
RTC_ALRMAR 561
RTC_ALRMBR 562
RTC_ALRMBSSR 572
RTC_BKxR 573
RTC_CALIBR 561
RTC_CALR 568
RTC_CR 555
RTC_DR 554
RTC_ISR 557
RTC_PRER 559
RTC_SHIFTR 565
RTC_SSR 563
RTC_TR 553
RTC_TSDR 567
RTC_TSSSR 567
RTC_TSTR 566
RTC_WPR 566
RTC_WUTR 560

S

SDIO_CLKCR 751
SDIO_DCOUNT 757
SDIO_DCTRL 756
SDIO_DLEN 755
SDIO_DTIMER 755
SDIO_FIFO 764
SDIO_FIFOCNT 763
SDIO_ICR 759
SDIO_MASK 761
SDIO_POWER 751
SDIO_RESPCMD 754
SDIO_RESPx 754
SDIO_STA 758
SPI_CR1 701
SPI_CR2 703
SPI_CRCPR 705
SPI_DR 705
SPI_I2SCFGR 707
SPI_I2SPR 708
SPI_RXCRCR 706
SPI_SR 704
SPI_TXCRCR 706
SYSCFG_EXTICR1 156
SYSCFG_EXTICR2 157
SYSCFG_EXTICR3 157
SYSCFG_EXTICR4 158
SYSCFG_MEMRMP 155

T

TIM2_OR 411
TIM5_OR 412
TIMx_ARR 406, 444, 453, 465
TIMx_BDTR 355
TIMx_CCER 348, 404, 443, 452
TIMx_CCMR1 . . . . . 344, 400, 440, 450
TIMx_CCMR2 . . . . . 347, 403
TIMx_CCR1 . . . . . 353, 407, 445, 454
TIMx_CCR2 . . . . . 354, 407, 445
TIMx_CCR3 . . . . . 354, 408
TIMx_CCR4 . . . . . 355, 408
TIMx_CNT 352, 406, 444, 453, 464
TIMx_CR1 . . . . . 333, 392, 434, 447, 462
TIMx_CR2 . . . . . 334, 394, 435, 463
TIMx_DCR 357, 409
TIMx_DIER 339, 396, 437, 448, 463
TIMx_DMAR 358, 409
TIMx_EGR 342, 399, 439, 449, 464
TIMx_PSC 352, 406, 444, 453, 465
TIMx_RCR 353
TIMx_SMCR 337, 395, 436
TIMx_SR 341, 397, 438, 448, 464

U

USART_BRR 649
USART_CR1 649
USART_CR2 652
USART_CR3 653
USART_DR 648
USART_GTPR 656
USART_SR 646

W

WWDG_CFR 477
WWDG_CR 476
WWDG_SR 477