To all our customers

Regarding the change of names mentioned in the document, such as Mitsubishi Electric and Mitsubishi XX, to Renesas Technology Corp.

The semiconductor operations of Hitachi and Mitsubishi Electric were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Mitsubishi Electric, Mitsubishi Electric Corporation, Mitsubishi Semiconductors, and other Mitsubishi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Note : Mitsubishi Electric will continue the business operations of high frequency & optical devices and power devices.

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

RENESAS
Renesas Technology Corp.

Renesas Technology Corp.

ADVANCED AND EVER ADVANCING *SOCIO-TECH*

MITSUBISHI DIGITAL ASSP M66290A USB APPLICATION NOTE

Ver 1.0 Rev.C '00-11-20

MITSUBISHI ELECTRIC CORPORATION

MITSUBISHI ELECTRIC SEMICONDUCTOR SYSTEMS CORPORATION

-- Contents --

1. Overview of the Manual

1.1. Overview

This manual (referred to as the Application Note henceforth) describes application technology manual for controlling the MITSUBISHI Digital ASSP general purpose USB device controller [M66290 Series] (referred to as M66290 henceforth) that is connected to the memory extension of the control CPU, and which does not specify any particular CPUs.

1.2. Features of the Application Note

The Application Note has the following features.

- Describes the techniques for M66290 control using actual coding examples.
- Describes notes on using the M66290.
- Links with M66290 control F/W.
- Describes abundant practical functional purposes.

The M66290 control F/W V.1.01 does not handle control of USB special signals. This manual simply shows examples of the M66290 control method. The actual method must be modified according to the customer's application.

1.3. Related Documents

[1] M66290AGP/FP data sheet
[http://www.mitsubishichips.com/data/datasheets/assps/index.html]

[2] M66290A USB Firmware Ver1.01
[MESC Release '99.12.16]

[3] Base of USB (in Japanese)
[http://www.infomicom.mesc.co.jp/usb/usbtop.htm]

[4] USB Device Catalog
[http://www.infomicom.mesc.co.jp/usb/usbmaine.htm]

[5] USB Specification Version1.1
[http://www.usb.org/developers/docs.html]

2. Overview of USB

2.1. What is USB?

- ·Standard communication specification of PC peripheral
- ·Handles 2 speeds; Full/Low
- •Four transfer types
- ·Packet communication of 1 frame per 1ms
- ·Polling response from Host PC
- ·LSB fast, BitStuff, NRZI

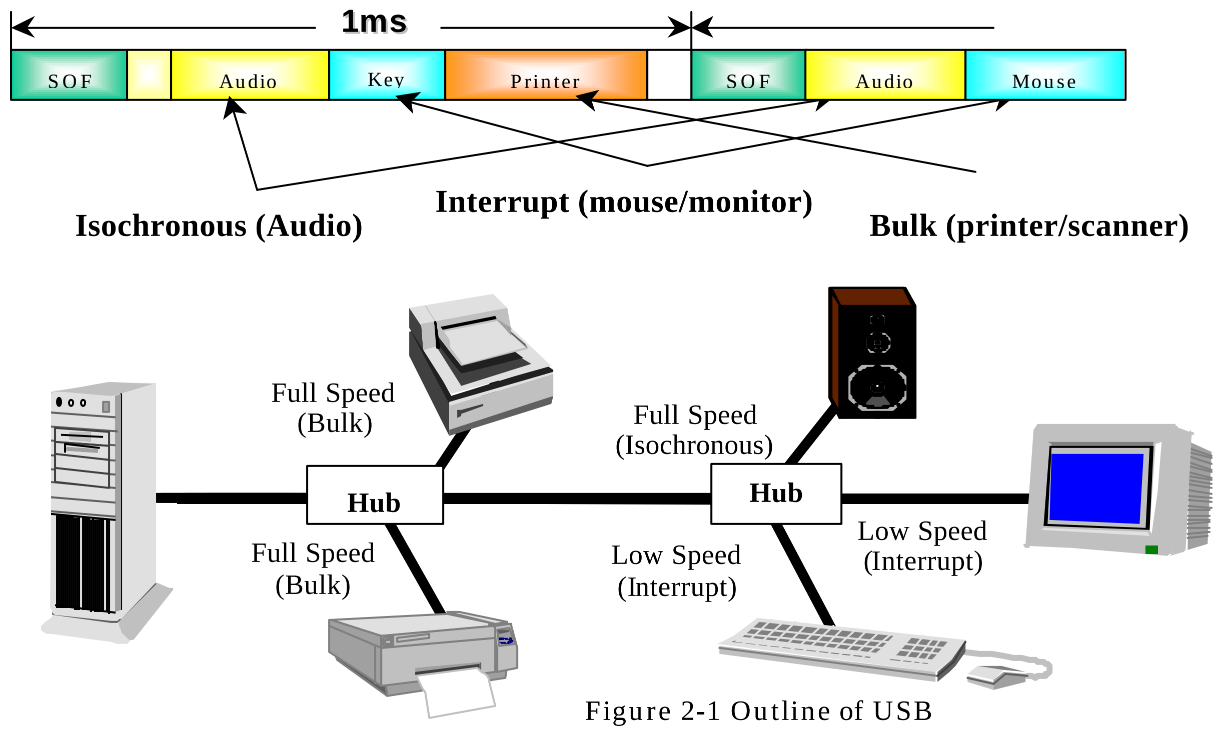


Figure 2-1 Outline of USB

2.2. Transfer Type

[Control] Control Transfer	• Transfer that is mainly used at setup • Consists of 3 stages (each stage consists of multiple packets)
[Data transfer] Interrupt Transfer Low Speed	• Suitable for real time transfer although the amount of data is small such as keyboard and mouse operations • Consists of 3 packets(Token, Data, Handshake)
Bulk Transfer	• Suitable for transfer of large volume of data such as printer output although the real time feature is low • Consists of 3 packets(Token, Data, Handshake)
Isochronous Transfer Full Speed	• Suitable for transfer of a relatively large volume of data that requires a certain transfer rate such as voice and animation. The data transfer width and the data transfer time are guaranteed. Data error guarantee is not supported. Consists of 2 packets(Token, Data)

Figure 2-2 USB Transfer type

2.3. USB Protocol

USB performs data communication in packet units. USB forms a communication unit called a transaction by grouping multiple packets. One transfer is formed by grouping multiple transactions.

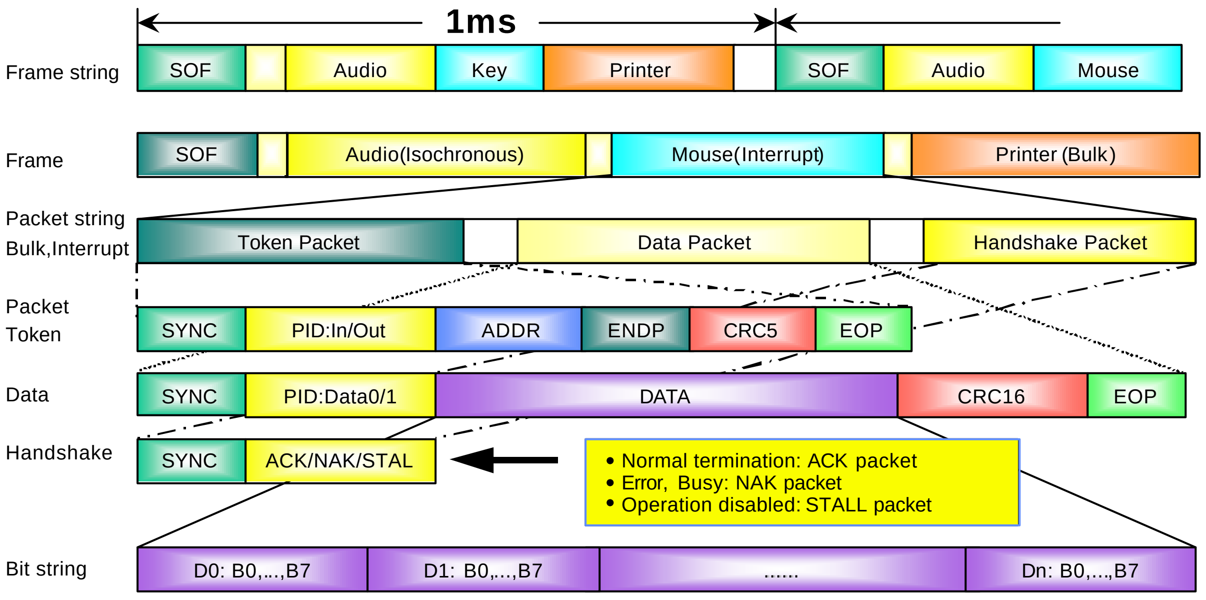


Figure 2-3 USB Communication Protocol

2.4. USB Packet

All bus transactions involve the transmission of up to three packets. Figure 2-4 shows the packet format. A packet consists of fields starting from the SYNC field followed by the PID field that indicates the packet type.

Four types of packets are used by USB.

SOF Packet — Frame start packet that is used by the host every 1ms.

SYNC	PID	Frame Number	CRC5	EOP
8bit	8bit	11bit	5bit	2bit

Token packet — Packet issued by the host at the start of a transaction

SYNC	PID	ADDR	ENDP	CRC5	EOP
8bit	8bit	7bit	4bit	5bit	2bit

Data packet — Packet used at data transfer

SYNC	PID	DATA...	CRC16	EOP
8bit	8bit	0~1023 byte	16bit	2bit

Handshake packet — Packet used by a transaction that performs flow control

SYNC	PID
8bit	8bit

Figure 2-4 Packet Format

3. Overview of the M66290

3.1. Features of the M66290

See below for the features of the M66290. See the data sheet for details.

- USB specification 1.1 compliant
- Built-in USB transceiver circuit
- Supports Full Speed (12 Mbps) transmission
- Handles all the USB transfer types
(Control, bulk, isochronous, and interrupt transfer)
- Built-in FIFO (3 KBytes) for Endpoint
- Up to 6 endpoint (EP0 to EP5) selectable
- Data transfer condition selectable for each Endpoints (EP1 to EP5)
  - •Data transfer type(Bulk, Isochronous and Interrupt)
  - •Transfer direction (IN/OUT)
  - • Buf fer size of FIFO (maximum 1024 Bytes)
Double (Toggle) buffer configuration
  - • Continuous transfer mode(Buffering up to 1 Kbyte *2)
  - •Maximum packet size

- Supports 4 input clock frequencies
  - • Input clock : 6/12/24/48 MHz
  - • Built-in PLL which has an oscillation buffer and outputs at 48 MHz
- Supports both 8-bit and 16-bit DMA transfers
- 16-bit CPU bus interface
- 3.3V single power source
- Built-in JTAG

APPLICATION
- •Printer, scanner
- •Display monitor, multimedia speaker
- • PC camera , Terminal adapter etc.
- • Support all PC peripheral using Full Speed USB

Outline
M66290AGP :48P6Q-A (LQFP)
M66290AFP :48P6X-A (TQFP)

Figure 3-1 Features of M66290

3.2. Example of Connection with Control CPU

Figure 3-2 shows a connection example when 16-bit microcomputer is used as the CPU for controlling the M66290. Data bus D0 to D15 are set to an input state when /CS is not selected, so when the input level were indefinite, lead-through current would flow. Therefore, pull-up or pull-down resistance is required at the designing of hardware.

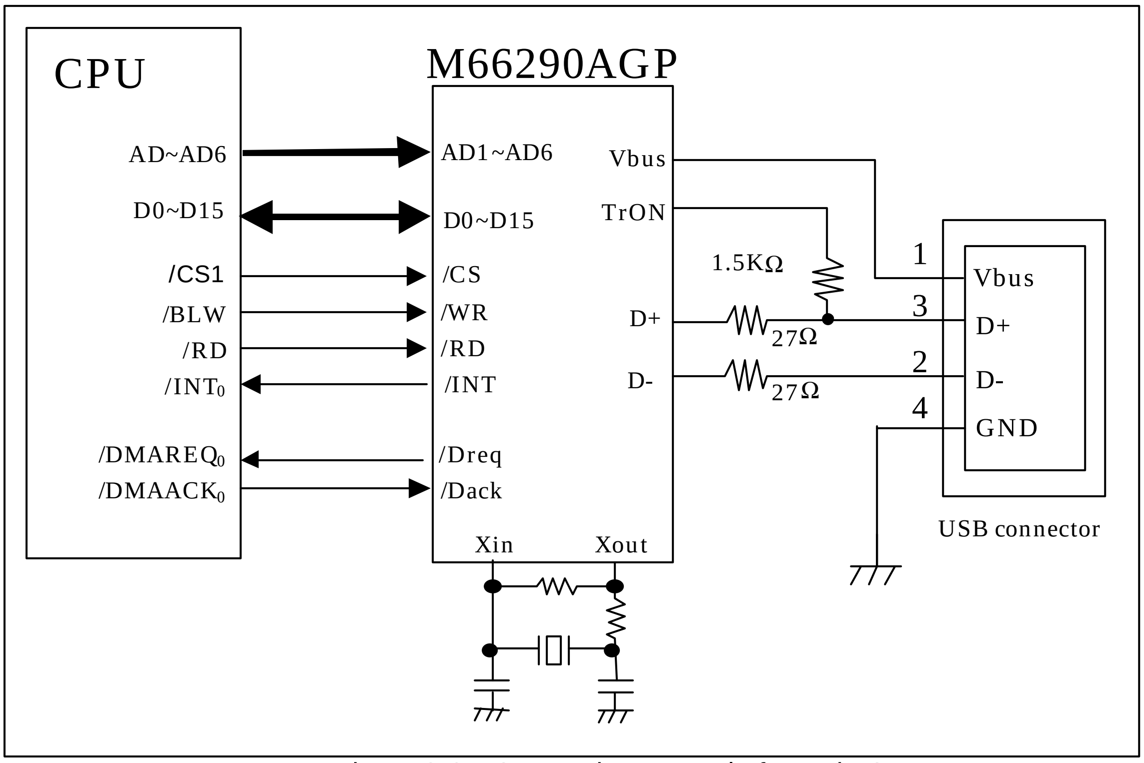


Figure 3-2 Connection Example for Main CPU

3.3. Global Flow of Control Program

The M66290 notifies to CPU USB data transmit/receive processing and special signal processing using interrupt signals. Therefore, the control program consists of an initialization program and a USB interrupt processing program.

The control program checks interrupt factors using the external interrupt program and executes related processing.

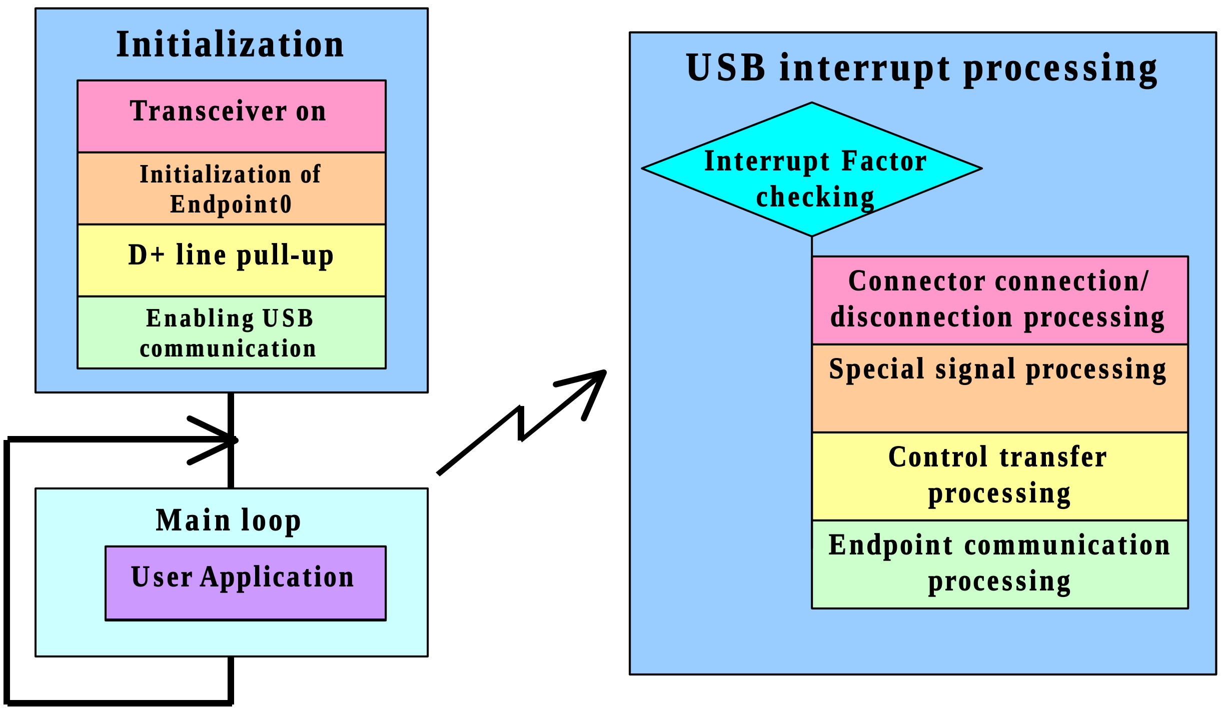
(1) Special signal processing
  Vbus interrupt, resume interrupt, SOF detection interrupt, and device state transition interrupt

(2) Control transfer processing
  Control transfer stage transition interrupt and device state transition interrupt

(3) Endpoint transfer processing
  Empty/size error interrupt, buffer/not ready interrupt, and buffer ready interrupt

The M66290 notifies various USB factors to the control CPU through interrupts. The following diagram shows the outline of the system flow when various USB factors that can be checked by the M66290 are posted by interrupt processing.

The main function requires initialization of the control CPU and registers that are related to the M66290.



Initialization
Transceiver on
Initialization of Endpoint0
D+ line pull-up
Enabling USB communication
Main loop
User Application
USB interrupt processing
Interrupt Factor checking
Connector connection/ disconnection processing
Special signal processing
Control transfer processing
Endpoint communication processing

Figure 3-3 Example of M66290 Control Program Configuration

3.4. Interrupt Processing

Control CPU identifies USB interrupts from the /INT signals from the M66290. Control CPU can identify factors from the contents of the following four types of interrupt status registers.

Interrupt status 0 register (Address 18h) Interrupt type identification
Interrupt status 1 register (Address 1Ah) Buffer ready identification
Interrupt status 2 register (Address 1Ch) Buffer not ready identification
Interrupt status 3 register (Address 1Eh) Buffer empty/size error identification

The interrupt status 0 register is used to identify a M66290 interrupt. The control program identifies the interrupt factor according to the bit status corresponding to each interrupt and performs related interrupt processing.

The interrupt status 1 register, interrupt status 2 register, and interrupt status 3 register are used to identify endpoint numbers that correspond to the endpoint buffer status interrupts.

See below for the flow of processing performed when a USB interrupt is detected.

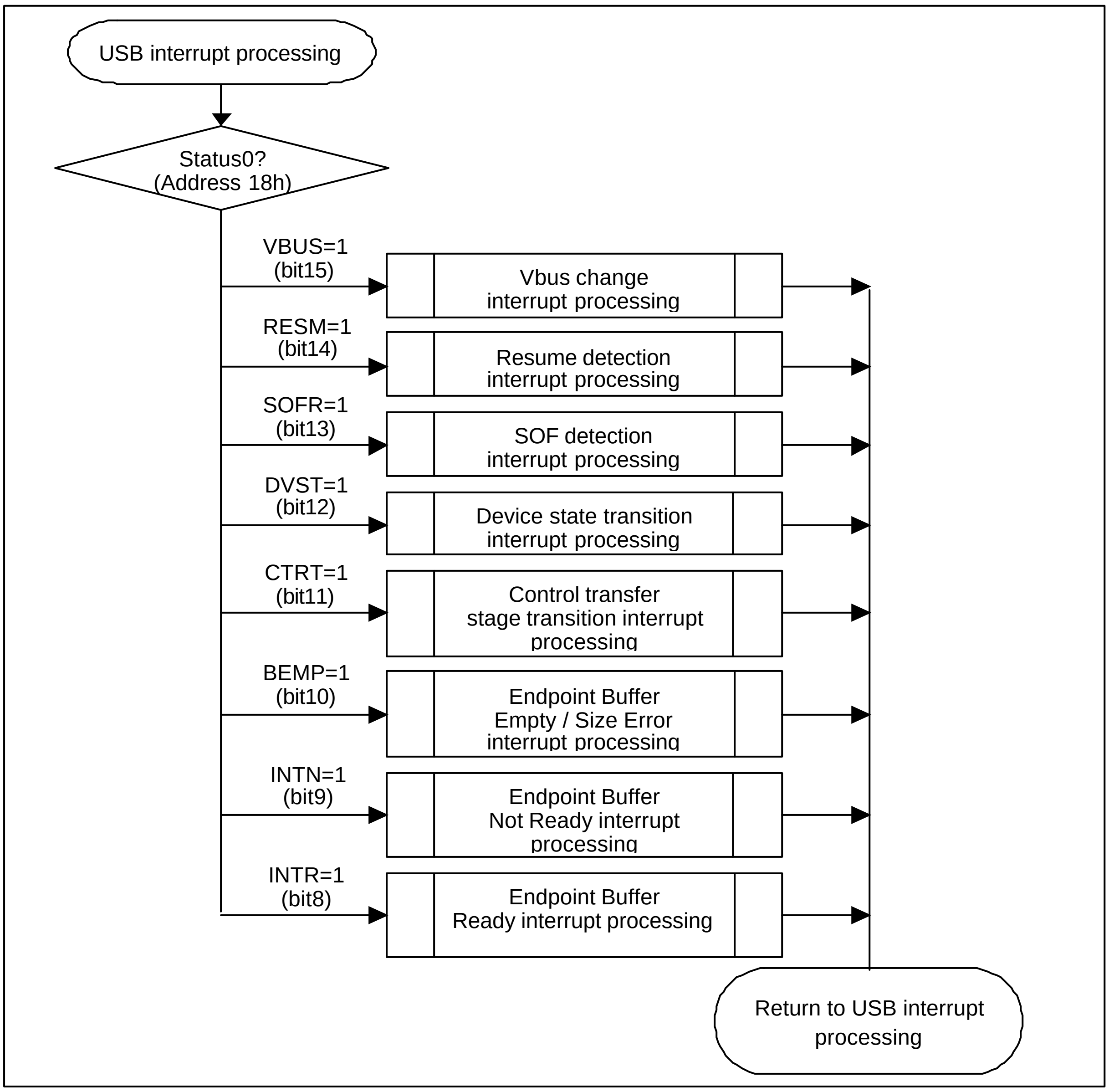

USB interrupt processing
Status0? (Address 18h)
VBUS=1 (bit15)
Vbus change interrupt processing
RESM=1 (bit14)
Resume detection interrupt processing
SOFR=1 (bit13)
SOF detection interrupt processing
DVST=1 (bit12)
Device state transition interrupt processing
CTRT=1 (bit11)
Control transfer stage transition interrupt processing
BEMP=1 (bit10)
Endpoint Buffer Empty / Size Error interrupt processing
INTN=1 (bit9)
Endpoint Buffer Not Ready interrupt processing
INTR=1 (bit8)
Endpoint Buffer Ready interrupt processing
Return to USB interrupt processing

Figure 3-4 Example of M66290 Interrupt Processing

3.5. Interrupt of Data Transfer

The M66290 is equipped with three types of data transfer interrupts and users can select each data transfer interrupt type according to the purpose. The endpoint from which the interrupt factor of the interrupt occurred can be identified from one of the status registers 1 to 3. Note that the occurrence factor of a data transfer interrupt varies according to the transfer type, transfer direction, and access method.

See the M66290 data sheet [(2) Interrupts] for clearing each interrupt factor and the detail function.

Table 3-1 Interrupt Factors According to the Data Transfer Types

Features of M66290 interrupt functions			
Transfer type	INTR interrupt	INTN interrupt	BEMP interrupt
Outline of interrupt factor	Read/Write enable (Level interrupt)	Transmit/receive unable (Edge interrupt)	Buffer Empty (Edge interrupt)
Control Read transfer	Does not occur	"NAK" response for IN token	Transmission completion
Control Write transfer	Receive data Read enable	"NAK" response for OUT token	Exceeded packet size
Data transfer IN (CPU access)	Transmit data Write enable	"NAK" response for IN token	Transmission completion (Buffer Empty)
Data transfer IN (DMA access)	Does not occur	"NAK" response for OUT token	Transmission completion (Buffer Empty)
Data transfer OUT (CPU access)	Receive data Read enable	"NAK" response for IN token	Exceeded packet size
Data transfer OUT (DMA access)	Received short packet	"NAK" response for OUT token	Exceeded packet size

* When the FIFO buffer is cleared while the endpoint is in the IN direction, the INTR interrupt occurs continuously.

* Note that when FIFO is used by a double buffer, the BEMP interrupt occurs at transmission completion of opposite side of the double buffer while data is written.

(See 3.6, "FIFO Control" also.)

The following interrupt is recommended for each USB data transfer.

Table 3-2 Recommended Interrupt for Each Transfer

Transfer	Recommended interrupt	Double buffer	Continuous transfer
Control Read	Control transfer transition BufferEmpty(non-continuous transfer)	—	O
Control Write	Control transfer transition Buffer Ready (non-continuous transfer)	—	O
Bulk IN	Buffer Ready	O	O
Bulk OUT	Buffer Ready	O	O
Interrupt IN	Buffer Empty	—	—
Interrupt OUT	Buffer Ready	—	—
Isochronous IN	Buffer Ready	O	—
Isochronous OUT	Buffer Ready	O	—

O ------------ Used

— ------------ Not used

3.6. FIFO Control

The M66290 is equipped with a large capacity FIFO buffer of 3K bytes and uses the buffer by allocating a separate buffer area to each of endpoints EP0 to EP5. The FIFO buffer consists of 64 bytes * 48 blocks. Specify the first block number (EPi_Buf_Nmb) of EPi configuration register 0 (60h, 64h, 68h, 6Ch, 70h) , buffer size (EPi_Buf_Siz), and the buffer mode (EPi_DBLB) to allocate a FIFO buffer area to each endpoint. See the setting example that is described later for the method of allocating an area. An error does not occur even if the areas overlap. Note that the latest data written to the overlapped areas becomes valid.

Three FIFO registers are used to access each endpoint buffer. The FIFO registers include one EP0 dedicated FIFO register (CPU access) and two EP1~EP5 common FIFO data registers. The common FIFO data registers are classified into the register for CPU access and the register for DMA transfer according to the access method. The setting of the FIFO selection register determines the endpoint to be accessed among EP1 to EP5.

Do not specify the endpoint number with DMA transfer mode specified in the CPU_FIFO selection register.

The three FIFO data registers have the following functions and perform options according to the setting of the “FIFO selection registor and FIFO control register”.

(1) Short packet transmission function (IVAL: IN buffer status bit)
When sending a short packet, write IVAL after writing data in the FIFO register.

(2) Transmit/receive buffer clear function (BCLR: Buffer clear bit)
When a double buffer is set, the side (half of buffer size) that can be access from the control CPU is cleared. When clearing the all of buffer, use EPi_ACLR.

(3) Null data (length 0 data) transmission function (IVAL & BCLR)
By writing IVAL=”1” and BCLR=”1”, a Null packet of length “0” can be sent.

(4) Data length (8/16 bits) setting function (Octl: Register 8-bit mode bit)
The mode can be changed from a 16-bit mode to an 8-bit mode during FIFO buffer Read/Write processing. During odd number byte access, data is read or written to the last byte of 16 bits, the mode is switched from the 16-bit mode to an 8-bit mode, and the last byte is read or written.

(5) Receive data length count-down function (RCNT: Read count mode bit)
The receive data length count-down function is valid for CPU access only.

When the FIFO selection is accessed for write processing while data is being written to or read from the FIFO buffer, the FIFO buffer internal counter is initialized and the data that has been written (read) becomes invalid. Also, when this situation happens during FIFO data Write processing, the system assumes “FIFO Not Ready” (the host cannot respond to the IN token) and Empty interrupt and Null addition function (valid at continuous transfer setting) operate.

When FIFO Write processing is too slow, the above incident occurs. FIFO Write processing is terminated by the following processing.

(1) IVAL=1 was written and a short packet was specified.
(2) Data was written until the FIFO buffer becomes full.

* At continuous transfer setting also, write processing is terminated by the processing indicated in (2) and the system cannot respond to the IN token even if data with the length between the maximum packet length and FIFO buffer length has been written. In this case, processing is terminated by writing IVAL=1.

See Chapter 6, “Data Transfers” for a FIFO buffer access example.

Table 3-3 Example of Setting Each Endpoint Configuration Register

Transfer type	Bulk IN	Interrupt IN	Isochronous IN
Example of using endpoint			
Maximum packet size	64 bytes	32 bytes	100 bytes
FIFO buffer size	512 bytes	64 bytes	256 bytes
Double buffer structure	256*2	—	128*2
FIFO number	08h~0Fh	20h~20h	22h~25h
FIFO address	200h~3FFh	800h~83Fh	880h~97Fh
Example of setting configuration register 0/1			
Transfer type	01	10	11
Transfer direction	1	1	1
Toggle mode	0	0	0
Buffer size	0011(256 bytes)	0000(64 bytes)	0001(128 bytes)
Double buffer mode	1(256*2 bytes)	0	1(128*2 bytes)
Continuous transmit/receive mode	1	0	0
First buffer number	08h	20h	22h
ResponsePID	—	—	—
DMA transfer mode	0	0	0
Null addition mode	1	0	0
Auto-clear mode	0	0	0
FIFO access mode	0	0	0
Maximum packet size	040h	020h	064h
Transfer type	Bulk OUT	Interrupt OUT	Isochronous OUT
Example of using endpoint			
Maximum packet size	64 bytes	32 bytes	100 bytes
FIFO buffer size	1024 bytes	64 bytes	256 bytes
Structure of double buffer	512*2	—	128*2
FIFO number	10h~1Fh	21h~21h	26h~29h
FIFO address	400h~7FFh	840h~87Fh	980h~A7Fh
Example of setting configuration register 0/1			
Transfer type	01	10	11
Transfer direction	0	0	0
Toggle mode	0	0	0
Buffer size	0111(1024 bytes)	0000(64 bytes)	0001(128 bytes)
Double buffer mode	1(1024*2 bytes)	0	1(128*2 bytes)
Continuous transmit/receive mode	1	0	0
First buffer number	10h	21h	26h
Response PID	—	—	—
DMA transfer mode	0	0	0
Null addition mode	0	0	0
Auto-clear mode	0	0	0
FIFO access mode	0	0	0
Maximum packet size	040h	020h	064h

* Notes on using a Null data transmit mode

When a double FIFO buffer is set and data is still being written from control CPU to one buffer after transmission of data in the other buffer to PC is completed, the M66290 assumes a buffer empty state and sends a NULL packet autoamtically. Therefore, a Null addition transmit mode must be set after the entire transmission data has been written to the FIFO buffer.

See Section 9.6, “Q & A Relating to the Null Addition Function.

3.7. DMA Access

The M66290 can transfer data to FIFO buffers of endpoints EP1 to EP5 through DMA access. By DMA access, DMA FIFO data registers are accessed with a 16-bit width or an 8-bit width. In DMA access, the M66290 allocates buffer areas using endpoint configuration registers, specifies DMA transfer, and transfers data using DMA access FIFO data registers in the same way as for CPU access.

In DMA transfer by the M66290, data is transferred by handshaking between external DMAC, /Dreq, and /Dack signals, and the /Dreq signal is asserted when the buffer of the endpoint that is set to the DMA transfer mode is set to a Ready state. The Buffer Ready state enables indicates a Read Enable state if the endpoint transfer direction is set to OUT (receiving data from the host) and a Write Enable state if the transfer direction of the endpoint is set to IN (transmitting data to the host). The /Dack signal is handled in the same way as for the specification of the /CS signal and DMA FIFO data register address. The Read operation/Write operation is specified in the /RD signal or the /WR signal.

The M66290 negates /Dreq when detecting the /Dack signal ("L" level) by external DMAC after asserting the /Dreq signal. For assertion of the /Dreq signal for sending the next data, rising of the /RD or /WR signal (high-speed transfer mode) and rising of the /Dack signal (one-word transfer mode) can be selected. DMA transfer corresponds to only the single transfer that performs single-word transfer (16 or 18 bits) by one /Dreq activation.

* The M66290 sets the data bus to an output state regardless of the access method when the /RD signal is input while the /Dack signal is asserted (data of the DMA FIFO register is output). Caution on the hardware design is necessary to avoid data collision. Since the M66290 is in an input state in other states, pull-up or pull-down processing must be performed externally to avoid internal lead-through current.

* Note that the M66290 may stop the /Dreq signal according to the timing of /Dack in single-word transfer operation mode. In single-word transfer mode, the hardware must be designed so that the maximum period from rising of /RD or /WR to rising of Dack will be within 83ns. (See Figure 3-4..)

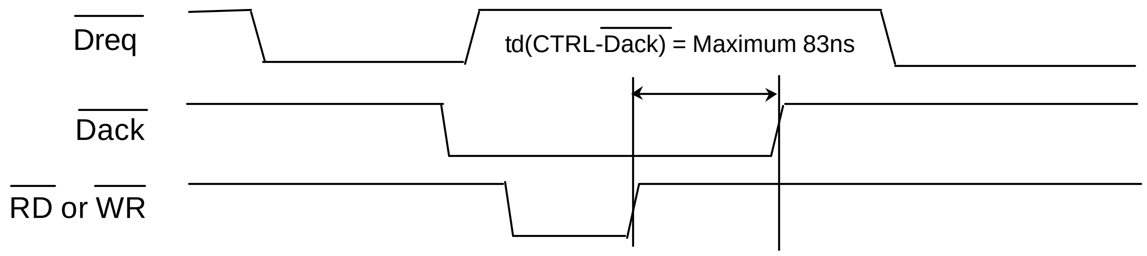


Figure 3-4 M66290 DMA Transfer Timing (Single-word mode)

In DMA transfer, an endpoint Buffer Not Ready interrupt and an Endpoint Buffer Empty interrupt occur according to the endpoint buffer state in the same way as for CPU access. However, there are the following differences with the CPU access for the Endpoint Buffer Ready interrupt.

(1) An Endpoint Buffer Ready interrupt does not occur when the transfer direction is IN.

(2) When the transfer direction is OUT, the M66290 receives a short data packet and an endpoint buffer ready interrupt occurs when transfer of the entire data that was received by DMA transfer has been transferred.

Reception of a short data packet can be recognized by the occurrence of an endpoint buffer ready interrupt and by referencing DMA_DTLN (DMA FIFO receive data length register). DMA_DTLN (DMA FIFO receive data length register) indicates the number of bytes of a short data packet or the number of bytes of received data up to the short data packet in continuous receive mode.

4. Initialization

4.1. Activating the M66290

Perform initialization processing after oscillation of the M66290 is stabilized. After PLL operation permission is set, a wait time (within 1ms) is required to avoid unstable operation by the clock. The host (upstream side) detects device connection when "H" is output from the Tr-On port.

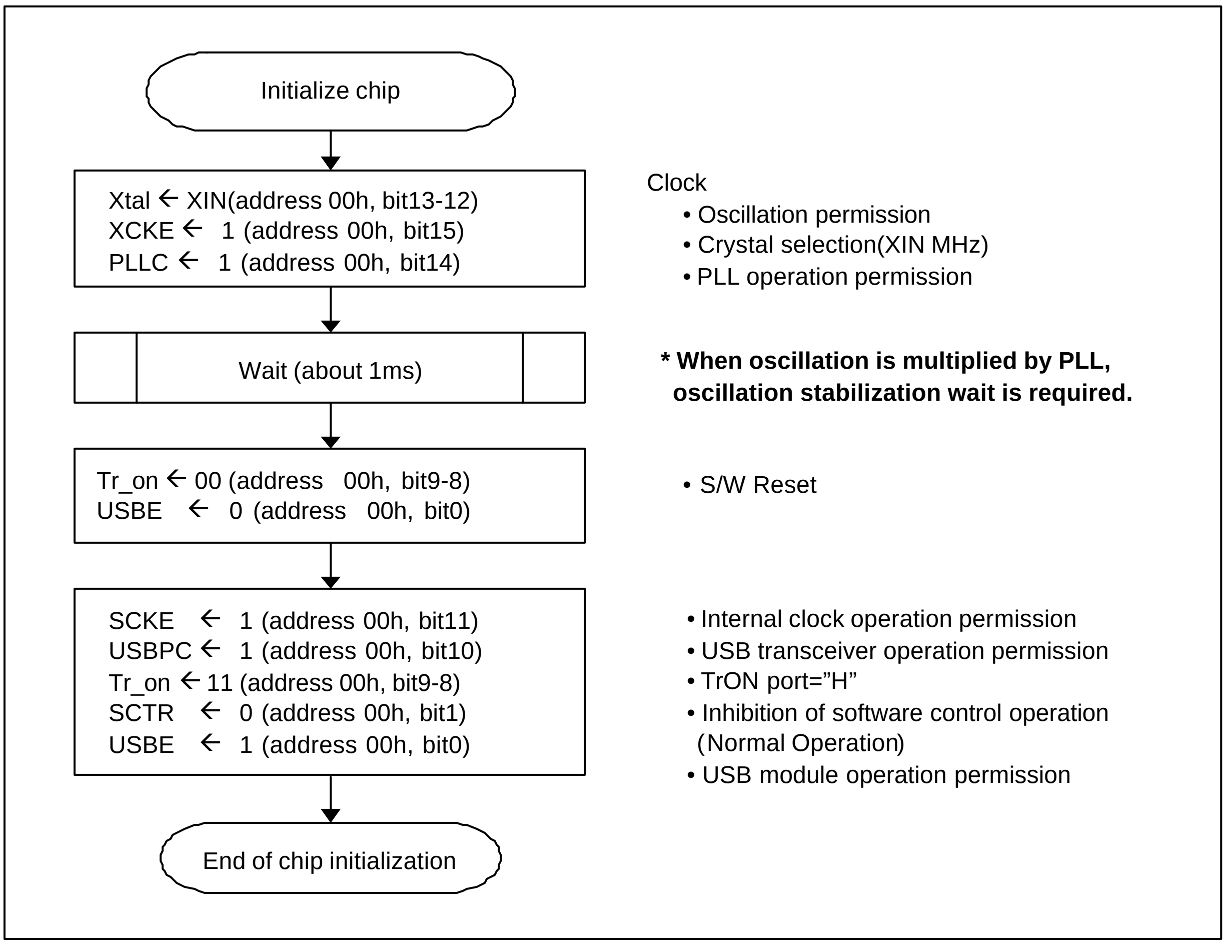
See the "Table of registers" of the M66290 Datasheet for the status of each register at execution of software reset (USBE="0").

M66290 activation example

- The activation example shows the case where oscillation is supplied from a crystal oscillator.
- The software reset after a wait period also functions as notification of non-connection to the host for receiving the USB bus reset signal.
- This setting example shows activation of the M66290 (initialization) performed when Vbus is supplied from the host.

(Tr_on and USBE bit setting : See Section 7.6, "Vbus Detection" for details of Vbus.)

See below for the initialization flow of the M66290.



Initialize chip
Xtal ← XIN(address 00h, bit13-12)
XCKE ← 1 (address 00h, bit15)
PLLC ← 1 (address 00h, bit14)
Clock
• Oscillation permission
• Crystal selection(XIN MHz)
• PLL operation permission
Wait (about 1ms)
* When oscillation is multiplied by PLL, oscillation stabilization wait is required.
Tr_on ← 00 (address 00h, bit9-8)
USBE ← 0 (address 00h, bit0)
• S/W Reset
SCKE ← 1 (address 00h, bit11)
USBPC ← 1 (address 00h, bit10)
Tr_on ← 11 (address 00h, bit9-8)
SCTR ← 0 (address 00h, bit1)
USBE ← 1 (address 00h, bit0)
• Internal clock operation permission
• USB transceiver operation permission
• TrON port="H"
• Inhibition of software control operation (Normal Operation)
• USB module operation permission
End of chip initialization

Figure 4-1 Example of M66290 Initialization

4.2. Preparation for Control Transfers (Initialization of Endpoint0)

When the length of the data to be transmitted or received at the data stage is 256 bytes or more, continuous transfer cannot be used.

See Chapter 5, "Control Transfers" for the status transition for which auto-response is effective.

Specify the address so that the transmit/receive FIFO buffers do not overlap.

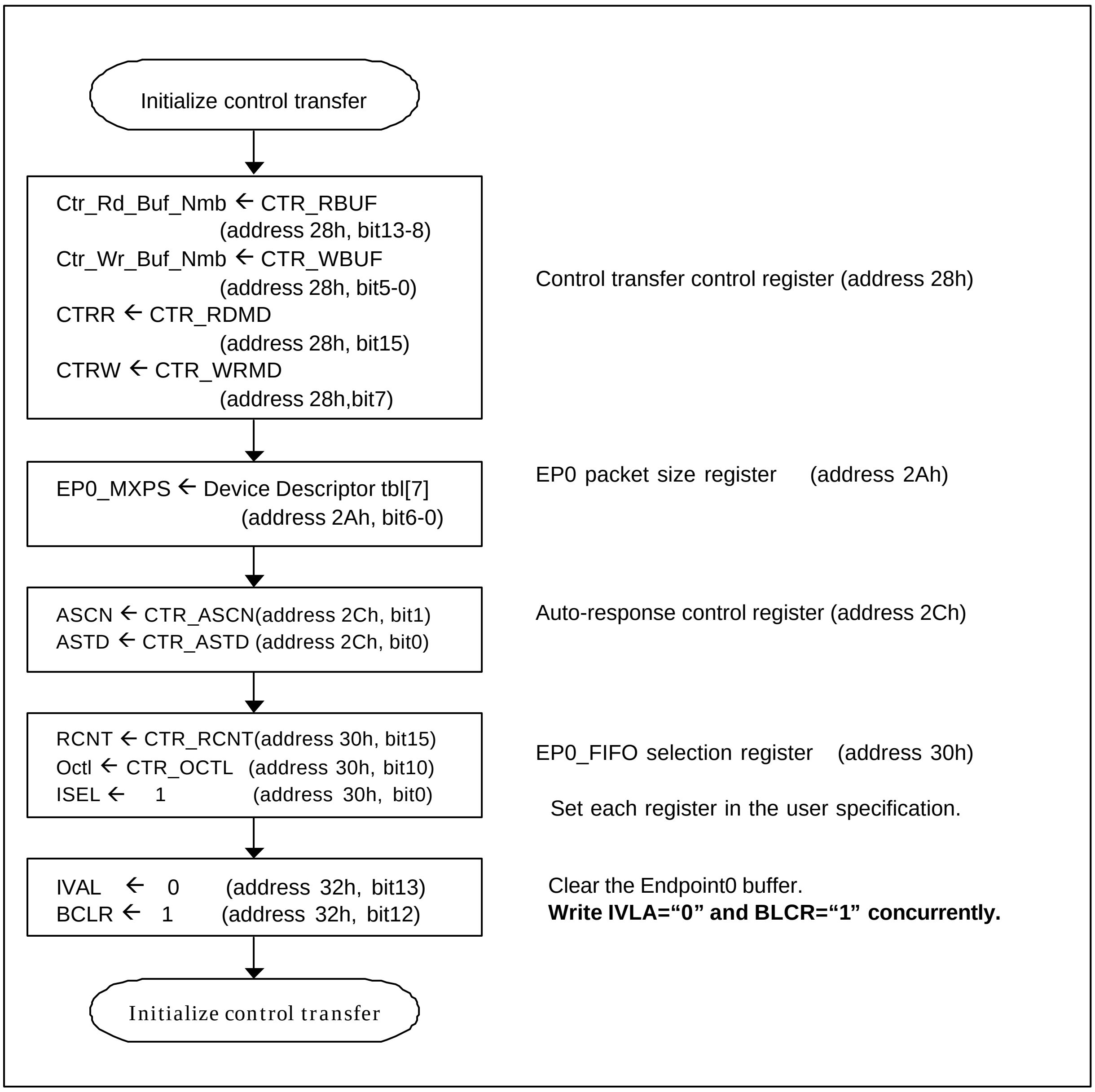
Initialize Endpoint0 after starting the M66290.

To clear the FIFO buffer, write IVLA = "0" and BLCR = "1" concurrently. In control transfer, the data packet in the Data stage uses the FIFO buffer. The reception of the data packet of the Setup stage is stored in a dedicated register. A size of "8 bytes" is recommended as the maximum packet size to correspond to the host controllers of the old type and the host OS of the old type.

Example of Endpoint0 initialization

•Each setting of Endpoint0 indicates the case where the macros are defined.

See below for the flow of Endpoint0 initialization processing.



Initialize control transfer
Ctr_Rd_Buf_Nmb ← CTR_RBUF (address 28h, bit13-8)
Ctr_Wr_Buf_Nmb ← CTR_WBUF (address 28h, bit5-0)
CTRR ← CTR_RDMD (address 28h, bit15)
CTRW ← CTR_WRMD (address 28h,bit7)
Control transfer control register (address 28h)
EP0_MXPS ← Device Descriptor tbl[7] (address 2Ah, bit6-0)
EP0 packet size register (address 2Ah)
ASCN ← CTR_ASCN(address 2Ch, bit1)
ASTD ← CTR_ASTD (address 2Ch, bit0)
Auto-response control register (address 2Ch)
RCNT ← CTR_RCNT(address 30h, bit15)
Octl ← CTR_OCTL (address 30h, bit10)
ISEL ← 1 (address 30h, bit0)
EP0_FIFO selection register (address 30h)
Set each register in the user specification.
IVAL ← 0 (address 32h, bit13)
BCLR ← 1 (address 32h, bit12)
Clear the Endpoint0 buffer.
Write IVLA="0" and BLCR="1" concurrently.
Initialize control transfer

4.3. USB Interrupt Enable

Set M66290 interrupt permission after starting the M66290 and initializing Endpoint0.
Set the timing of the M66290 interrupt permission taking into consideration the timing of the control CPU and other chips according to the system configuration and the purposes.

A device state transition interrupt, control transfer, and a control transfer stage transition interrupt other than USB bus reset and suspend detection are performed after detection of the USB bus reset signal.
Endpoint buffer interrupts (BEMPE, INTNE, INTRE) are issued after reception of a Set Configuration request.

Example of interrupt permission when Vbus is supplied (USB device has already been connected to the PC)

- A VBUS interrupt is not used in this setting example.
- A SUSP interrupt is not used in this setting example.
- This example shows the setting at activation of the M66290 (initialization) when Vbus is supplied from the host.

(Tr_on and USBE bit setting : See Section 7.6, "Vbus Detection" for details.)
See below for the flow of USB interrupt permission processing.

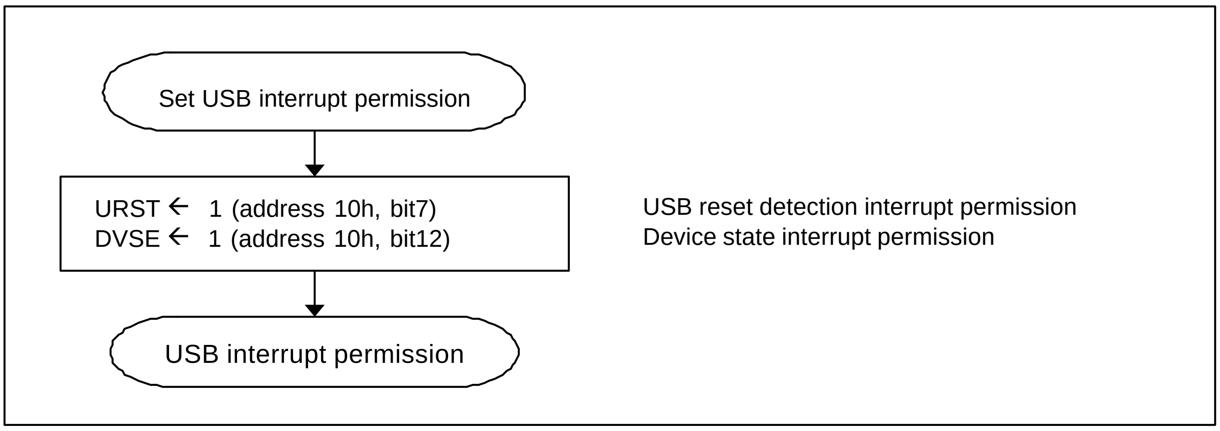

Set USB interrupt permission
URST ← 1 (address 10h, bit7)
DVSE ← 1 (address 10h, bit12)
USB reset detection interrupt permission
Device state interrupt permission
USB interrupt permission

Figure 4-3 Example of USB Interrupt Permission Setting

5. Control Transfers

5.1. Outline of Control Transfers

Control transfer is mainly bi-directional transfer that is used at setup and consists of at least two transaction stages (setup stage and status stage).
A data stage can be inserted between the two stages as optional.

In USB communication, control transfer by Endpoint0 and responses to a number of requests that are transmitted from a host to a device by control transfer of Endpoint0 are mandatory functions.

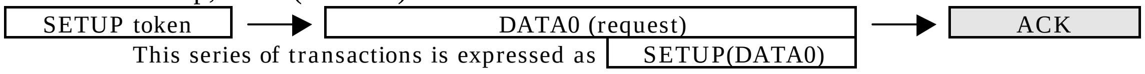
The M66290 is equipped with the following four functions to facilitate control transfer by Endpoint0 and the coding of a setup processing (bus enumeration) control program.

(1) Device state transition management function	[See Sections 5.2 and 5.3 of this manual]
(2) Data stage transition management function	[See Sections 5.4 and 5.5 of this manual]
(3) Continuous transmit/receive function	[See Sections 5.7 and 5.8 of this manual]
(4) Auto-response function	[See Section 5.9 of this manual]

The following transaction types are available for control transfer.

Setup stage :

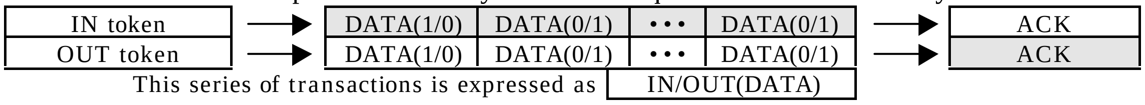
Consists of setup, data (DATA0) and Handshake Packet.



This series of transactions is expressed as SETUP(DATA0)

Data stage :

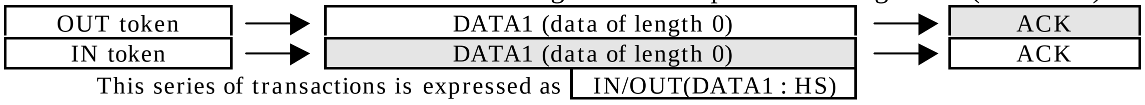
A data transaction is repeated as many times as required in the same way as for bulk transfer.



This series of transactions is expressed as IN/OUT(DATA)

Status stage :

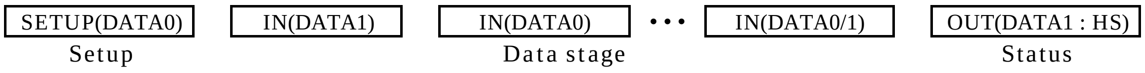
Consists of the reverse tokens of the data stage and data packet of length “0” (“DATA1”)



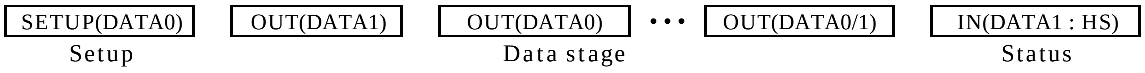
This series of transactions is expressed as IN/OUT(DATA1 : HS)

The following types of control transfers are available.

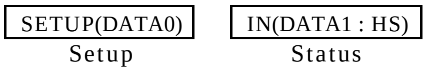
Control read transfer :



Control write transfer :



Control write no data transfer :



The M66290 is equipped with a stage management function and notifies stage transition through a control transfer transition interrupt. CPU performs stage processing related to the control transfer according to the interrupt factor. The contents of the data acquired at the setup stage clearly indicate to the device request type that is required by the host, and the device can perform related control transfer processing.

The M66290 is equipped with a continuous transmit/receive function. The continuous transmit/receive function in control transfer sends and receive data continuously up to 255 bytes without issuing an interrupt. By using the continuous transmit/receive function, the data stage no longer requires FIFO access for each transaction, thereby enhancing the CPU performance.

5.2. USB Device State Transition

The M66290 manages device state transition of USB device states, which are Powered, Default, Address, Configured, Suspended, with hardware. Device state transition is notified to the CPU by a device state transition interrupt (DVST). When an auto-response is set, transfer of Set Address and Set Configuration requests can be completed without issuing an interrupt to the CPU.

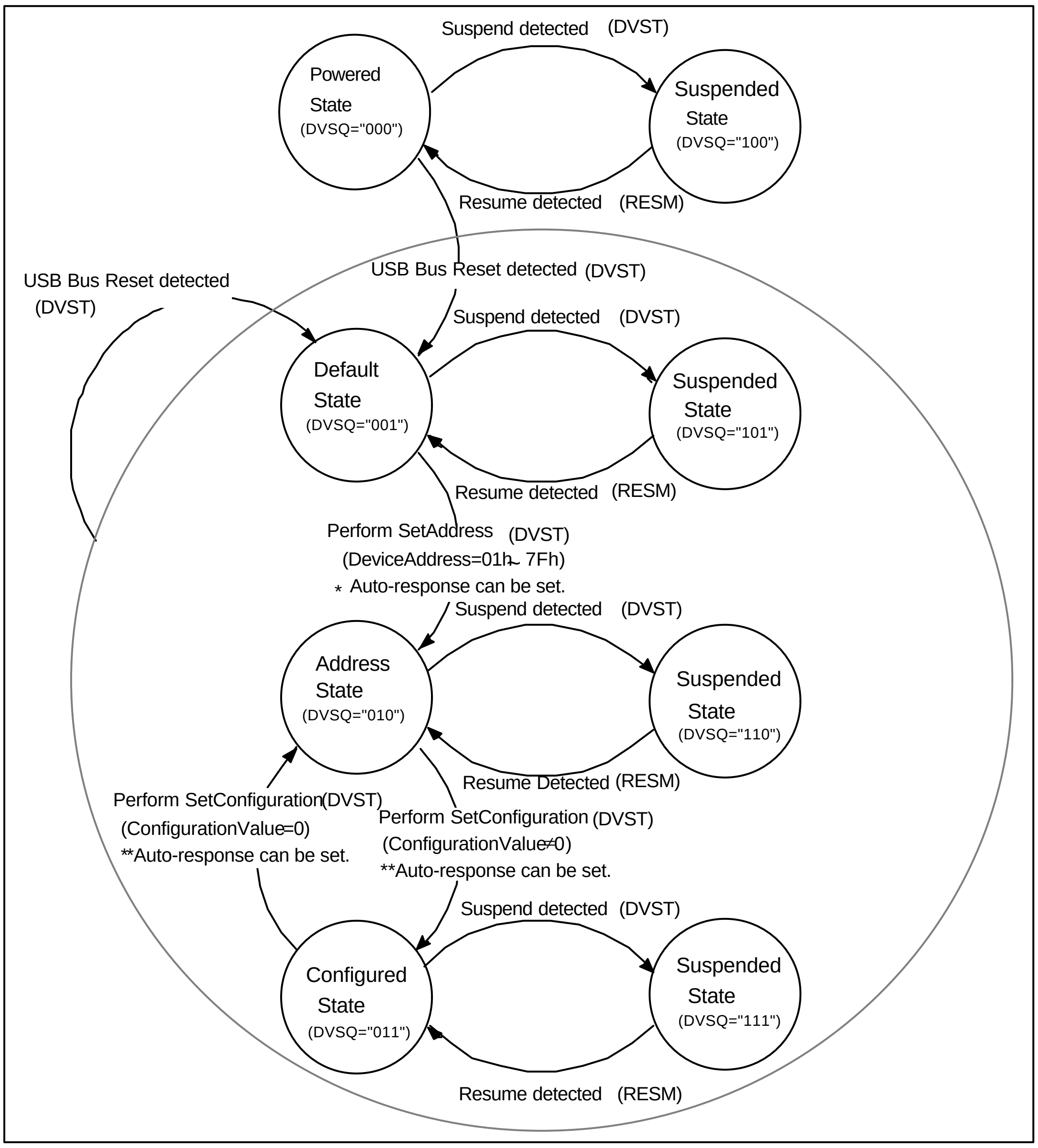

Suspend detected (DVST)
Powered State (DVSQ="000")
Suspended State (DVSQ="100")
Resume detected (RESM)
USB Bus Reset detected (DVST)
USB Bus Reset detected (DVST)
Suspend detected (DVST)
Default State (DVSQ="001")
Suspended State (DVSQ="101")
Resume detected (RESM)
Perform SetAddress (DVST)
(DeviceAddress=01h～7Fh)
* Auto-response can be set.
Suspend detected (DVST)
Address State (DVSQ="010")
Suspended State (DVSQ="110")
Resume Detected (RESM)
Perform SetConfiguration(DVST)
(ConfigurationValue=0)
**Auto-response can be set.
Perform SetConfiguration (DVST)
(ConfigurationValue≠0)
**Auto-response can be set.
Suspend detected (DVST)
Configured State (DVSQ="011")
Suspended State (DVSQ="111")
Resume detected (RESM)

Figure 5-1 Device State Transition

5.3. Device State Transition Processing

When device state transition is executed normally, the M66290 generates a device state transition interrupt (DVST). However, the device state transition interrupt (DVST) will not occur, unless the bit corresponding to Interrupt Enable Register0 (address 10h) of the Detect USB Reset Interrupt, Set Address Interrupt, Set Configuration interrupt, and Suspend Detection Interrupt is set. In this case, the related value is set in the device state bit (DVSQ) of Interrupt Status Register0(address 18h) by the device state management function. Processing of the device state that was changed is performed and the user program is executed according to the state transition.

The M66290 changes states by detecting the USB reset signal, suspend signal, or Set Address/Set Configuration request. Note that the device state bit (DVSQ) has been changed at the occurrence of an interrupt.

(1) Power state	software Reset	No processing required
(2) Default state	USB bus reset detection processing	Control transfer enable
(3) Address state	Set Address request detection	Address retention, etc.
(4) Configured state	Set Configuration request detection	Configuration number retention, etc.
(5) Suspend state	Suspend signal detection	Suspend processing

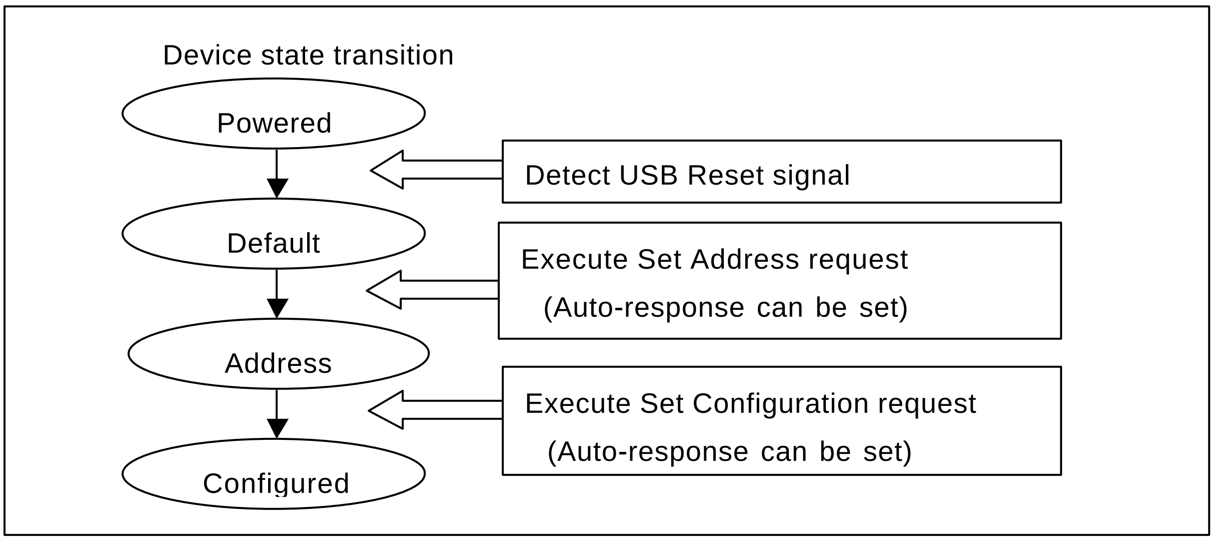

Device state transition
Powered
Default
Address
Configured
Detect USB Reset signal
Execute Set Address request
(Auto-response can be set)
Execute Set Configuration request
(Auto-response can be set)

Figure 5-2 Device State Transition

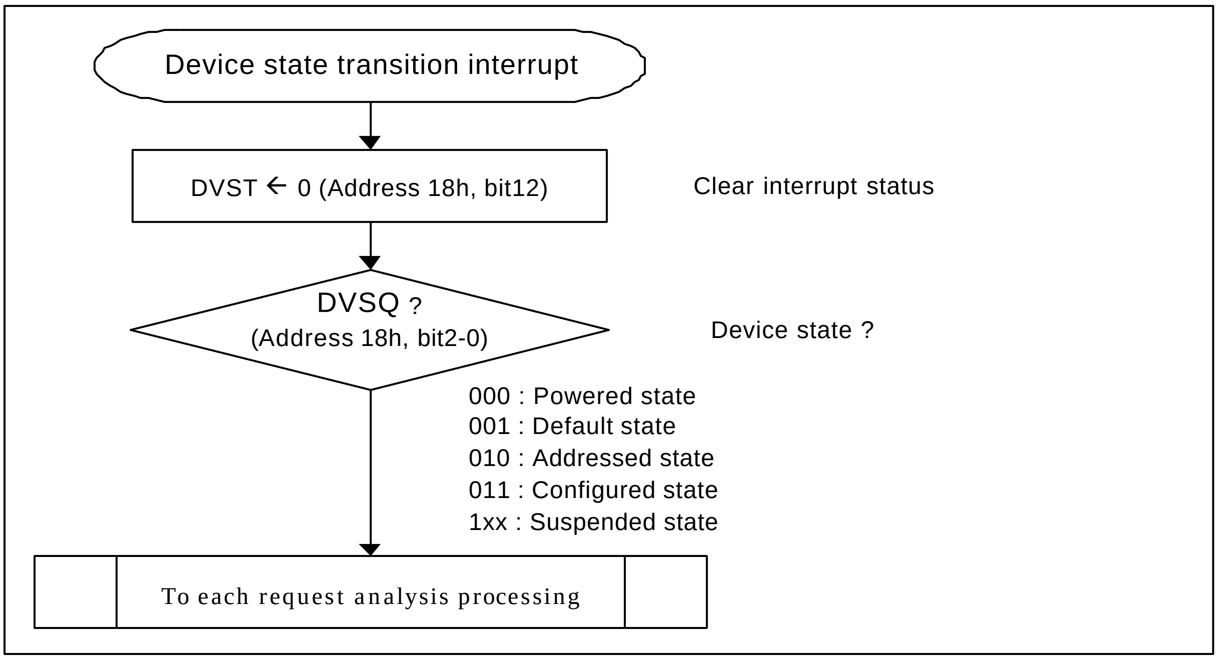


Figure 5-3 Example of a Setup Stage Control Procedure

5.4. Control Transfer Stage Transition

The M66290 manages transition of control transfer stages, namely a setup stage, a data stage, and a status stage through hardware. Control transfer stage transition is notified to the CPU by a control transfer stage transition interrupt (CTRT). When auto-response is set, transfer can be completed without generating an interrupt to CPU for the Set Address and Set Configuration requests. There are five CTRT interrupt factors; setup stage completion, Control Write transfer status stage transition, Control Read transfer status stage transition, Control Transfer completion, and control transfer sequence error. The following seven Control Transfer sequence errors can be detected by hardware.

When hardware detects a control transfer sequence error, the response PID is automatically set to STALL(“1x”) and the contents are retained until the next setup packet is received.

1. Receiving the IN token packet at the Control Write data stage
   (Receiving the IN token packet when no ACK response has been issued for the OUT token packet of the data stage)
2. Receiving the OUT token packet at the Control Write status stage
3. Receiving the OUT token packet at the Control Read data stage
   (Receiving the OUT token packet when no data has been transferred to the IN token packet of the data stage)
4. Receiving the IN token packet at the Control Read status stage
5. Receiving a Data Packet other than Null data at the Control Read status stage
6. Receiving the OUT token packet at the Control Write no data status stage
7. Receiving data that exceeds the maximum packet size

When a data packet exceeding the wLength value of the request is received in the control data stage, the M66290 cannot recognize the event as a sequence error.

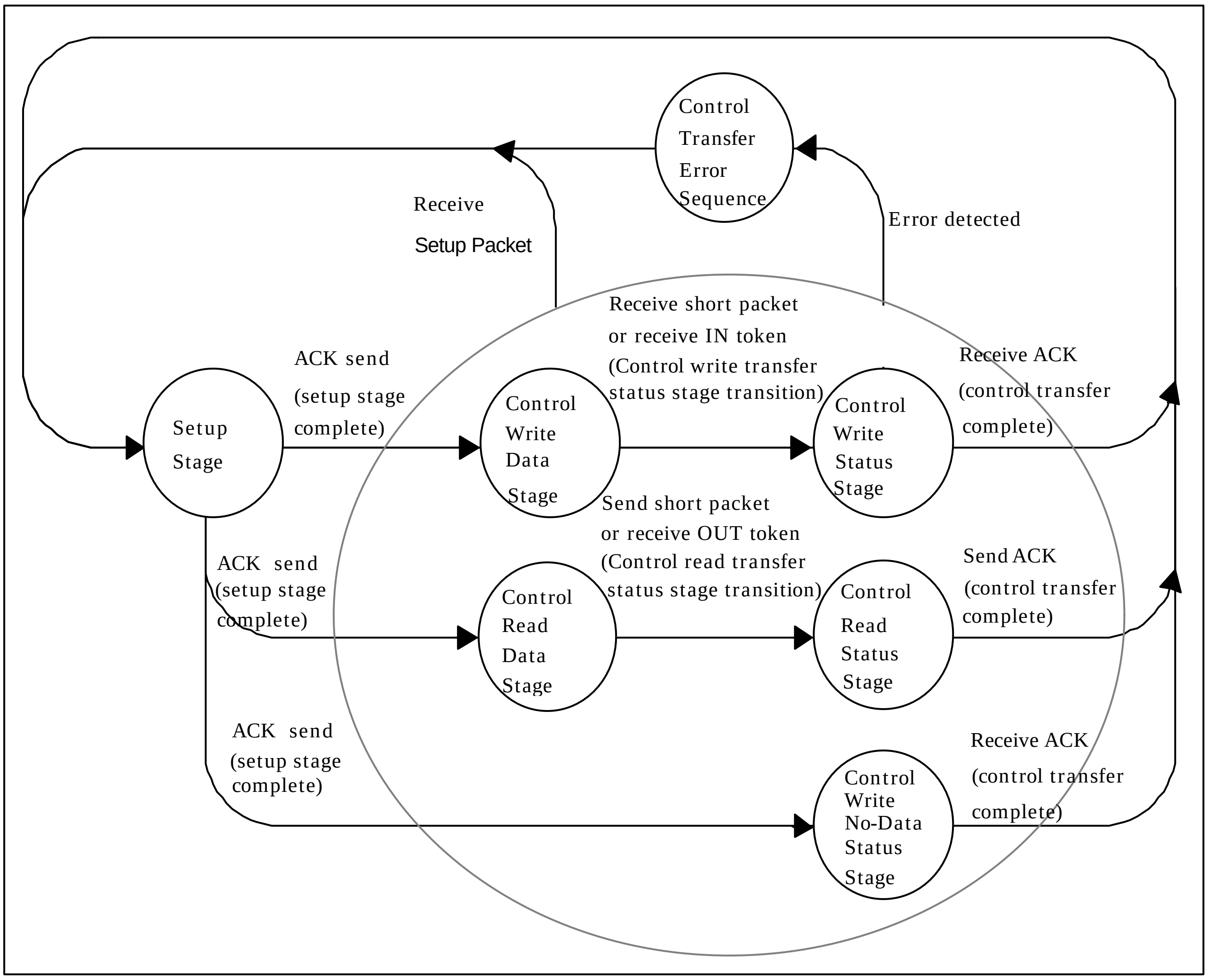

Control Transfer Error Sequence
Receive Setup Packet
Error detected
Receive short packet or receive IN token (Control write transfer status stage transition)
ACK send (setup stage complete)
Setup Stage
Control Write Data Stage
Control Write Status Stage
Receive ACK (control transfer complete)
Send short packet or receive OUT token (Control read transfer status stage transition)
ACK send (setup stage complete)
Control Read Data Stage
Control Read Status Stage
Send ACK (control transfer complete)
ACK send (setup stage complete)
Control Write No-Data Status Stage
Receive ACK (control transfer complete)

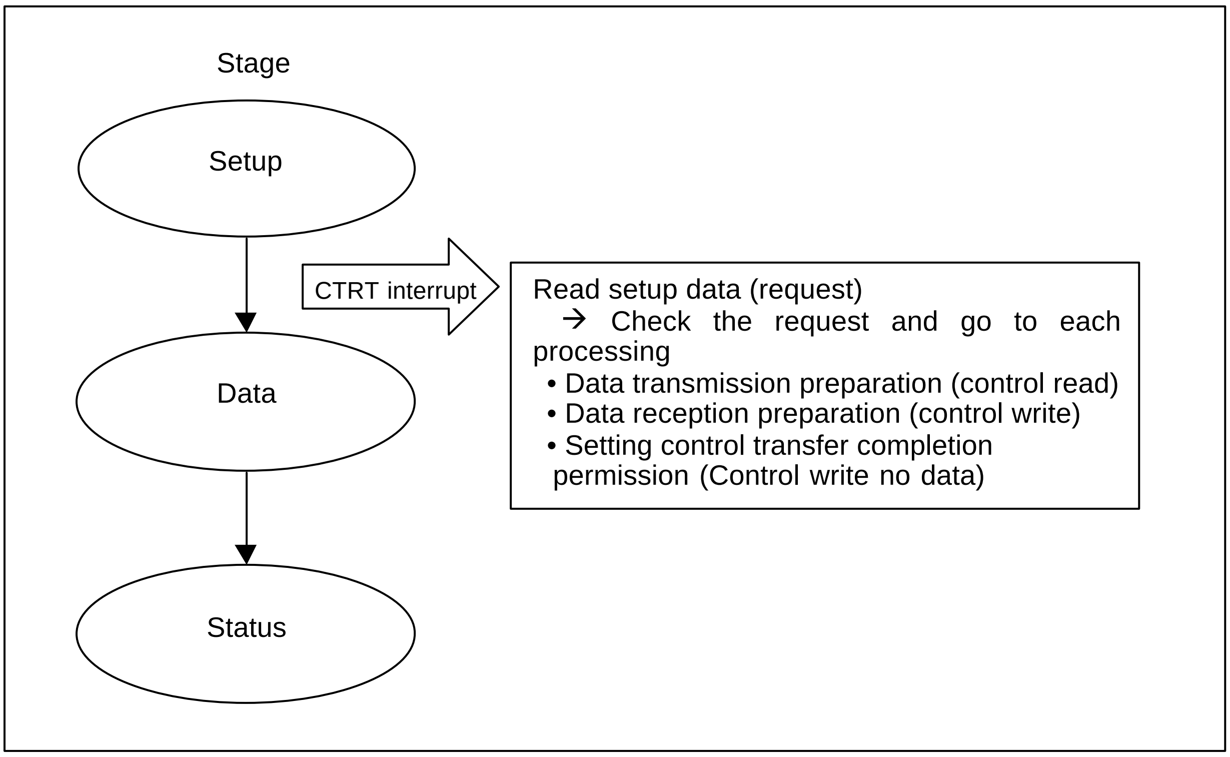
Figure 5-4 Control Transfer State Transition

5.5. Stage Transition Processing

The CPU that detected control stage transition of the M66290 through a CTRT interrupt checks the transfer stage and performs related processing. The M66290 stores the data packet (request from the host PC) of the setup stage in the dedicated register. The data packet of the data stage uses Endpoint0 FIFO.

Note that the stage of 66290 control transfer stage bit (CTSQ) has been shifted at the occurrence of a CTRT interrupt.

(1) Idle or setup stage	No processing is required
(2) Control read transfer data stage	Data (Request) analysis and preparation for data transmission
(3) Control write transfer data stage	Data (Request) analysis and preparation for data reception
(4) Control write no data transfer status stage	Data (Request) analysis
(5) Control read transfer status stage	No processing is required
(6) Control write transfer status stage	Data analysis
(7) Control transfer sequence error	Preparation for setup re-reception



Read setup data (request)
→ Check the request and go to each processing
• Data transmission preparation (control read)
• Data reception preparation (control write)
• Setting control transfer completion permission (Control write no data)

Figure 5-5 Status Stage Transition Processing

5.6. Setup Stage

A setup transaction transfers a request composed 8bytes(data packet of the setup stage) from the host and stores it in the following four registers.

Table 5-1 Request Storage Register

Register name	Address	Bit	Value stored
Request	20h	15-8	bRequest
		7-0	bmRequest
Value	22h	15-0	wValue
Index	24h	15-0	wIndex
Length	26h	15-0	wLength

When the request is stored in the registers normally, the M66290 generates a control transfer stage transition interrupt (CTRT). In this case, the setup packet detect bit (VALID) of the interrupt status register 0 is set to “1”. The control transfer stage management function sets a related value in the control transfer stage bit (CTSQ) in the register. After sets “0” in the setup packet detect bit (VALID), reads the request from the registers, checks any errors, performs request analysis processing, and prepares for the data stage.

When receiving a setup transaction normally, the M66290 sends an “ACK” response. After receiving a setup packet, the M66290 automatically sets PID of Endpoint0 to NAK.

See below for the flow of control transfer transition processing.

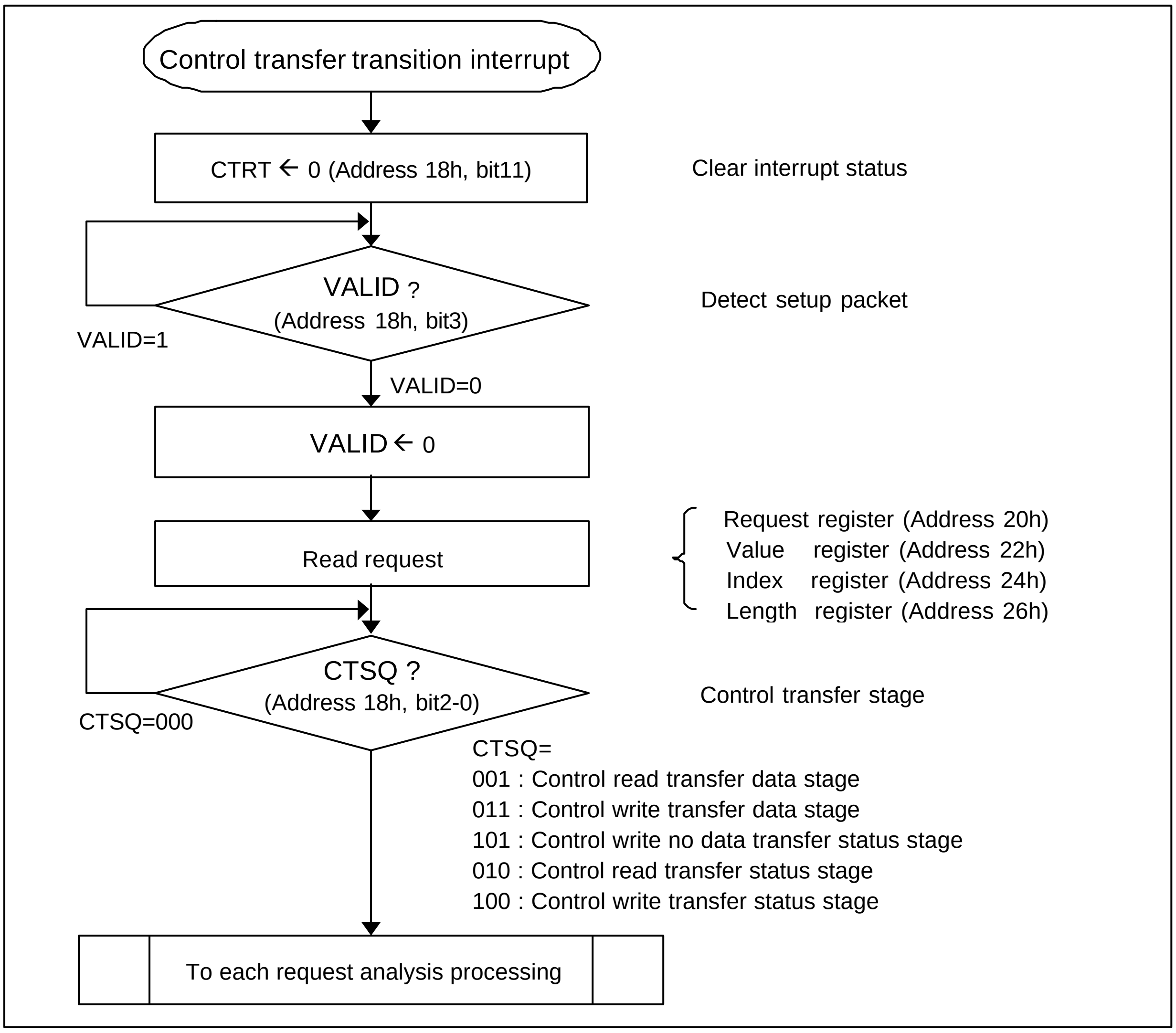

Control transfer transition interrupt
CTRT ← 0 (Address 18h, bit11)
Clear interrupt status
VALID ?
(Address 18h, bit3)
Detect setup packet
VALID=1
VALID=0
VALID ← 0
Read request
Request register (Address 20h)
Value register (Address 22h)
Index register (Address 24h)
Length register (Address 26h)
CTSQ ?
(Address 18h, bit2-0)
Control transfer stage
CTSQ=000
CTSQ=
001 : Control read transfer data stage
011 : Control write transfer data stage
101 : Control write no data transfer status stage
010 : Control read transfer status stage
100 : Control write transfer status stage
To each request analysis processing

Figure 5-6 Example of Control Transfer Transition Interrupt

5.7. Control Read Transfer Processing (Data Stage)

When the request that was received at the setup stage is a Control Read request, the data that was requested to the host PC in the data stage is transmitted.

The M66290 is equipped with a continuous transfer function and can transmit multiple packets continuously at data transmission (data stage) in control read. The maximum length of continuous transmission data in control transfer is 255 bytes. Set non-continuous transmission,when transmitting data exceeding 255 bytes. When sending a short packet in the data stage, the M66290 automatically changes the stage to a status stage. When the length of the data to be sent is an exact multiple of the maximum packet size, the M6290 automatically adds a Null packet (handled as a short packet), as specified in the USB Specification after sending the entire data and changes the stage. For transmission of data with a size less than the FIFO size, send a short packet by setting “IVAL=1” after writing the last data.

See below for the flow of Control Read transfer processing.0.

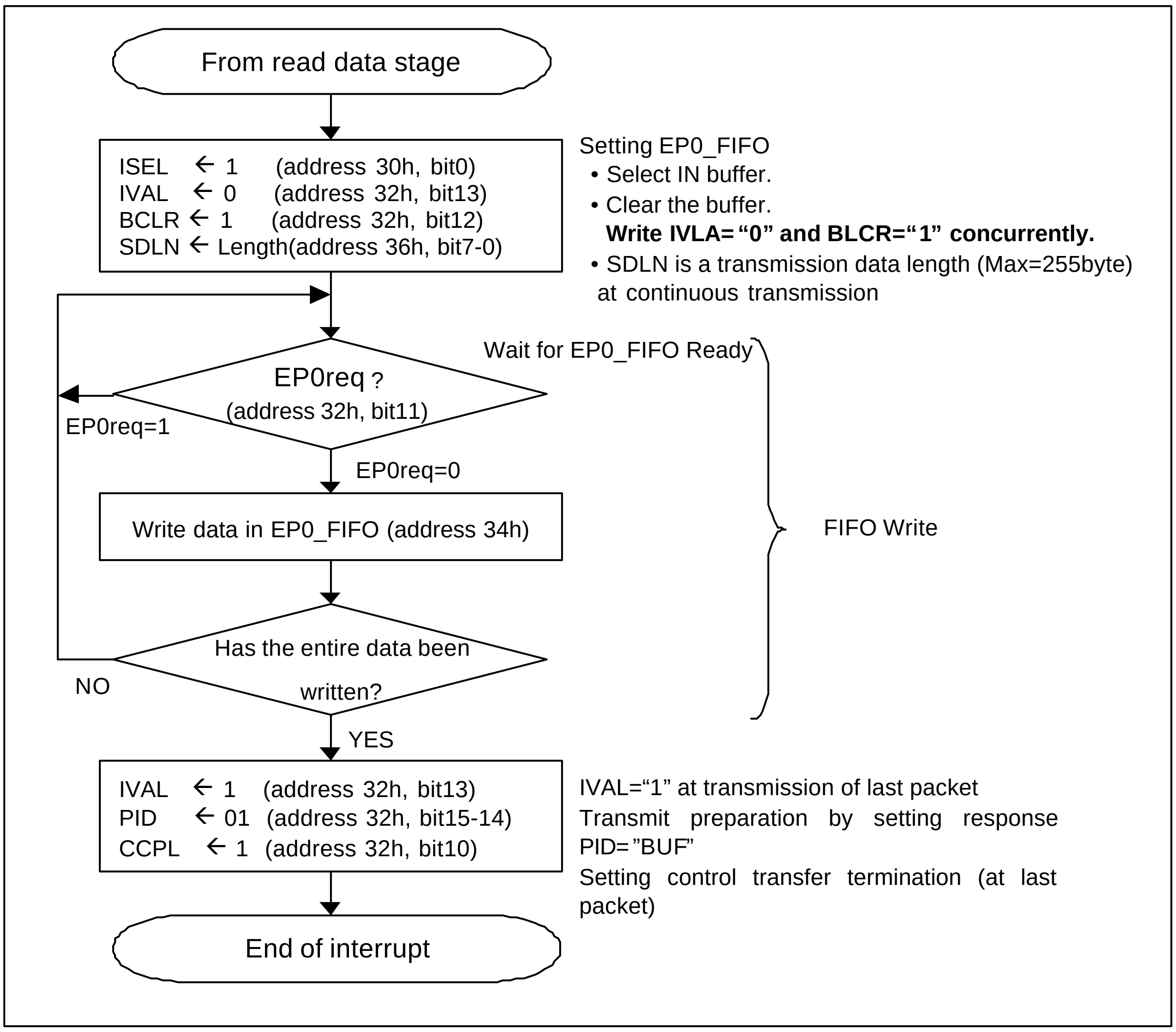

From read data stage
ISEL ← 1 (address 30h, bit0)
IVAL ← 0 (address 32h, bit13)
BCLR ← 1 (address 32h, bit12)
SDLN ← Length(address 36h, bit7-0)
Setting EP0_FIFO
• Select IN buffer.
• Clear the buffer.
Write IVLA=“0” and BLCR=“1” concurrently.
• SDLN is a transmission data length (Max=255byte) at continuous transmission
Wait for EP0_FIFO Ready
EP0req ? (address 32h, bit11)
EP0req=1
EP0req=0
Write data in EP0_FIFO (address 34h)
FIFO Write
Has the entire data been written?
NO
YES
IVAL ← 1 (address 32h, bit13)
PID ← 01 (address 32h, bit15-14)
CCPL ← 1 (address 32h, bit10)
IVAL=“1” at transmission of last packet
Transmit preparation by setting response PID=”BUF”
Setting control transfer termination (at last packet)
End of interrupt

Figure 5-7 Example of Control Read Transfer Processing

* When CCPL is set at a status stage, NAK packet for the OUT token is inserted.
* When the ISEL bit is accessed during FIFO write processing, the processing becomes invalid.
* When writing data of odd number bytes, use FIFO access 8bit mode bit (Octl) function. The mode can be changed to “Octl” when the last byte is written.

5.8. Control Write Transfer Processing (Data Stage)

When the request that was received at the setup stage is a Control Write request, the data that was requested to the host PC is received at the data stage.

The M66290 is equipped with a continuous transfer function at data reception (data stage) in Control Write and can receive multiple packets continuously. The maximum length of continuous reception data in control transfer is 255 bytes. Set a non-continuous mode when receiving data more than 256 bytes.

When receiving a short packet in data stage, the M66290 automatically changes the stage to a status stage. The M66290 also changes the stage when receiving a Null packet (handled as a short packet) as specified in the USB specification.

See below for the flow of Control Write transfer processing.

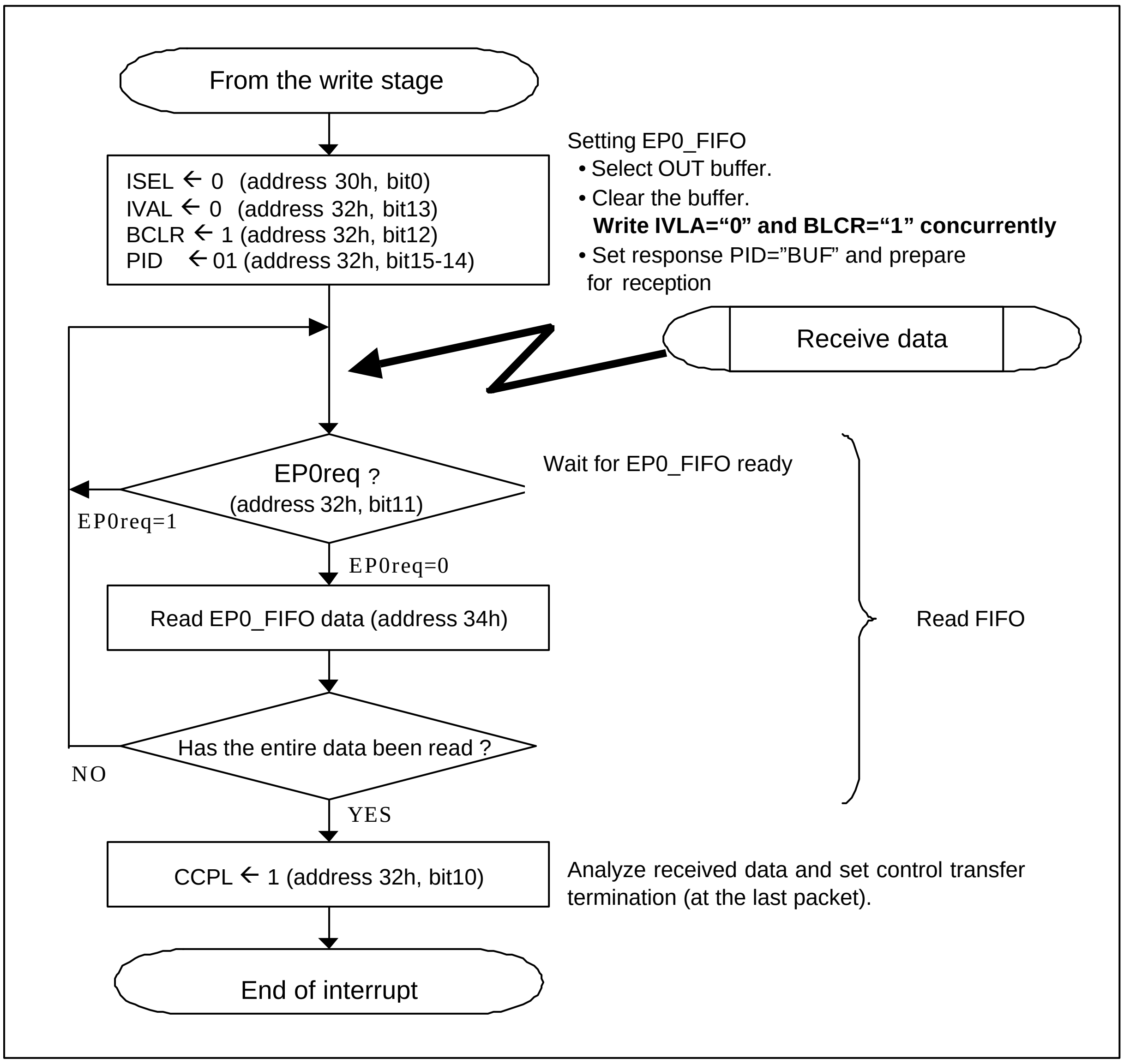

From the write stage
ISEL ← 0 (address 30h, bit0)
IVAL ← 0 (address 32h, bit13)
BCLR ← 1 (address 32h, bit12)
PID ← 01 (address 32h, bit15-14)
Setting EP0_FIFO
• Select OUT buffer.
• Clear the buffer.
Write IVLA="0" and BLCR="1" concurrently
• Set response PID="BUF" and prepare for reception
Receive data
EP0req ? (address 32h, bit11)
Wait for EP0_FIFO ready
EP0req=1
EP0req=0
Read EP0_FIFO data (address 34h)
Read FIFO
Has the entire data been read ?
NO
YES
CCPL ← 1 (address 32h, bit10)
Analyze received data and set control transfer termination (at the last packet).
End of interrupt

Figure 5-8 Example of Control Write Transfer Processing

* Set CCPL after analyzing the data that was received at the status data stage.
* When the ISEL bit is accessed during FIFO read processing, the read counter becomes invalid.
* When reading data of odd number of bytes, use FIFO access 8bit mode bit (Octl) function. The mode can be changed to "Octl" when the last byte is read.
* When the last packet is received in non-continuous control receive mode, an INTR interrupt occurs after a stage transition interrupt. Therefore, data must be read by checking the INTR status during status stage transition interrupt processing.

5.9. Control Write No Data Transfer Processing (Status Stage)

When the request that was received at a setup stage is a Control Write No Data request, the data does not transmit and receive at a data stage.

When receiving a Control Write No Data request, the M66290 automatically changes the stage to a status stage. The bmReqeustType and wLength field are used for checking a Control Write No Data request.

The M66290 can complete transfer without generating an interrupt in the CPU for the Set Address and Set Configuration standard requests in Control Write No Data transfer.

- In Address auto-response mode, the M66290 sends automatically responses for the Set Address Request (DeviceAddress=01h~7Fh)in the Default state, but it dose not response when the device state is another state or DeviceAddress 01h~7Fh for the Set Address Request.
- In Configuration auto-response mode, the M66290 sends automatically a response for the Set Configuration request (ConfigurationValue 0) when the device state is the Address state and for the Set Configuration request (Configuration Value=0) in the Configured state. The M66290 does not response automatically for requests in other states and for the Set Configuration requests with values different from the value indicated above.

See below for the flow of data transfer stage processing.

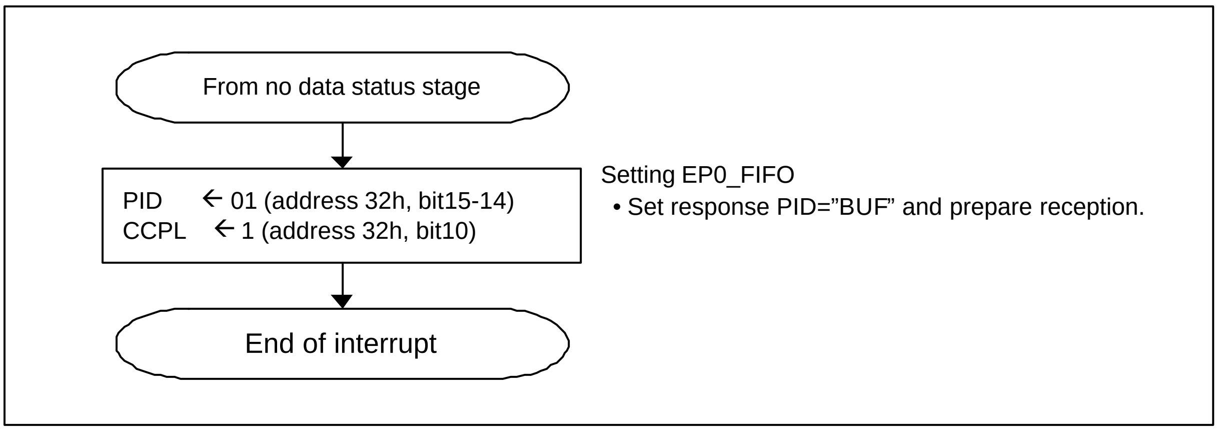

From no data status stage
PID ← 01 (address 32h, bit15-14)
CCPL ← 1 (address 32h, bit10)
Setting EP0_FIFO
• Set response PID=”BUF” and prepare reception.
End of interrupt

Figure 5-9 Example of Control No Data Transfer Processing

6. Data Transfers

6.1. Outline of Data Transfers

Data transfer is single-direction transfer and consists of at least two packets (token packet and data packet). In the transfer that guarantees data transfer between a host and a device by the retransmit function, a handshake packet is added following the data packet, involving 3 packets. Three transfer types are available; isochronous transfer consisting of 2 packets that does not detect communication errors between a host and a device and guarantees a transfer rate, bulk transfer and interrupt transfer consisting of 3 packets that detects communication errors, but does not guarantee a transfer rate.

See below for the features of each data transfer type.

Table 6-1 List of Data Transfer Types

Transfer type	Bulk transfer	Interrupt transfer	Isochronous transfer
Real time feature	Low	Real time	High
Transfer rate	Not guaranteed	Not guaranteed	Guaranteed
Volume of transfer data	Large	Small	Small – large
Number of packets	3	3	2
Data packet	Data0/Data1	Data0/Data1	Data0
Handshake	Yes	Yes	No
Maximum packet size	8, 16, 32, 64	0~64	0~1023
Features of M66290 function			
Continuous transmit function	Valid	Invalid	Invalid
Double buffer function	Valid	Valid	Valid
Null addition function	Valid (at continuous transfer)	Invalid	Invalid
DMA access	Valid	Valid	Valid

A data transfer transaction is specified as follows.

Bulk IN and interrupt IN :

Consists of IN token, data (DATA0/1), and handshake packets.

$$\begin{tikzcd}\boxed{\text{IN token}} \arrow[r] & \boxed{\text{DATA(0/1)}} \arrow[r] & \boxed{\text{ACK}}\end{tikzcd}$$

Isochronous IN :

Consists of IN token and data (DATA0).

$$\begin{tikzcd}\boxed{\text{IN token}} \arrow[r] & \boxed{\text{DATA(0)}}\end{tikzcd}$$

Bulk OUT and interrupt OUT :

Consists of OUT token, data (DATA0/1) and handshake packets.

$$\begin{tikzcd}\boxed{\text{OUT token}} \arrow[r] & \boxed{\text{DATA(0/1)}} \arrow[r] & \boxed{\text{ACK}}\end{tikzcd}$$

Isochronous OUT :

Consists of OUT token and data (DATA0).

$$\begin{tikzcd}\boxed{\text{OUT token}} \arrow[r] & \boxed{\text{DATA(0)}}\end{tikzcd}$$

This series of transactions is expressed as $\begin{array}{|c|}\hline \text{IN(DATA0)} \\ \hline \text{OUT(DATA0)} \\ \hline\end{array}$

Communication of each data transfer is performed in the following transaction structure.

Bulk IN transfers and interrupt IN transfer :

$\boxed{\text{IN(DATA0)}}$ $\boxed{\text{IN(DATA1)}}$ $\boxed{\text{IN(DATA0)}}$ $\boxed{\text{IN(DATA1)}}$ ･･･ $\boxed{\text{IN(DATA0/1)}}$

Isochronous IN transfer :

$\boxed{\text{IN(DATA0)}}$ $\boxed{\text{IN(DATA0)}}$ $\boxed{\text{IN(DATA0)}}$ $\boxed{\text{IN(DATA0)}}$ ･･･ $\boxed{\text{IN(DATA0)}}$

Bulk OUT transfer and interrupt OUT transfer :

$\boxed{\text{OUT(DATA0)}}$ $\boxed{\text{OUT(DATA1)}}$ $\boxed{\text{OUT(DATA0)}}$ $\boxed{\text{OUT(DATA1)}}$ ･･･ $\boxed{\text{OUT(DATA0/1)}}$

Isochronous OUT transfer :

$\boxed{\text{OUT(DATA0)}}$ $\boxed{\text{OUT(DATA0)}}$ $\boxed{\text{OUT(DATA0)}}$ $\boxed{\text{OUT(DATA0)}}$ ･･･ $\boxed{\text{OUT(DATA0)}}$

6.2. Features of Data Transfers by the M66290

The M66290 can transfer data of all types that are specified in USB communication. The M66290 is equipped with the following functions to facilitate the coding of data transfer programs.

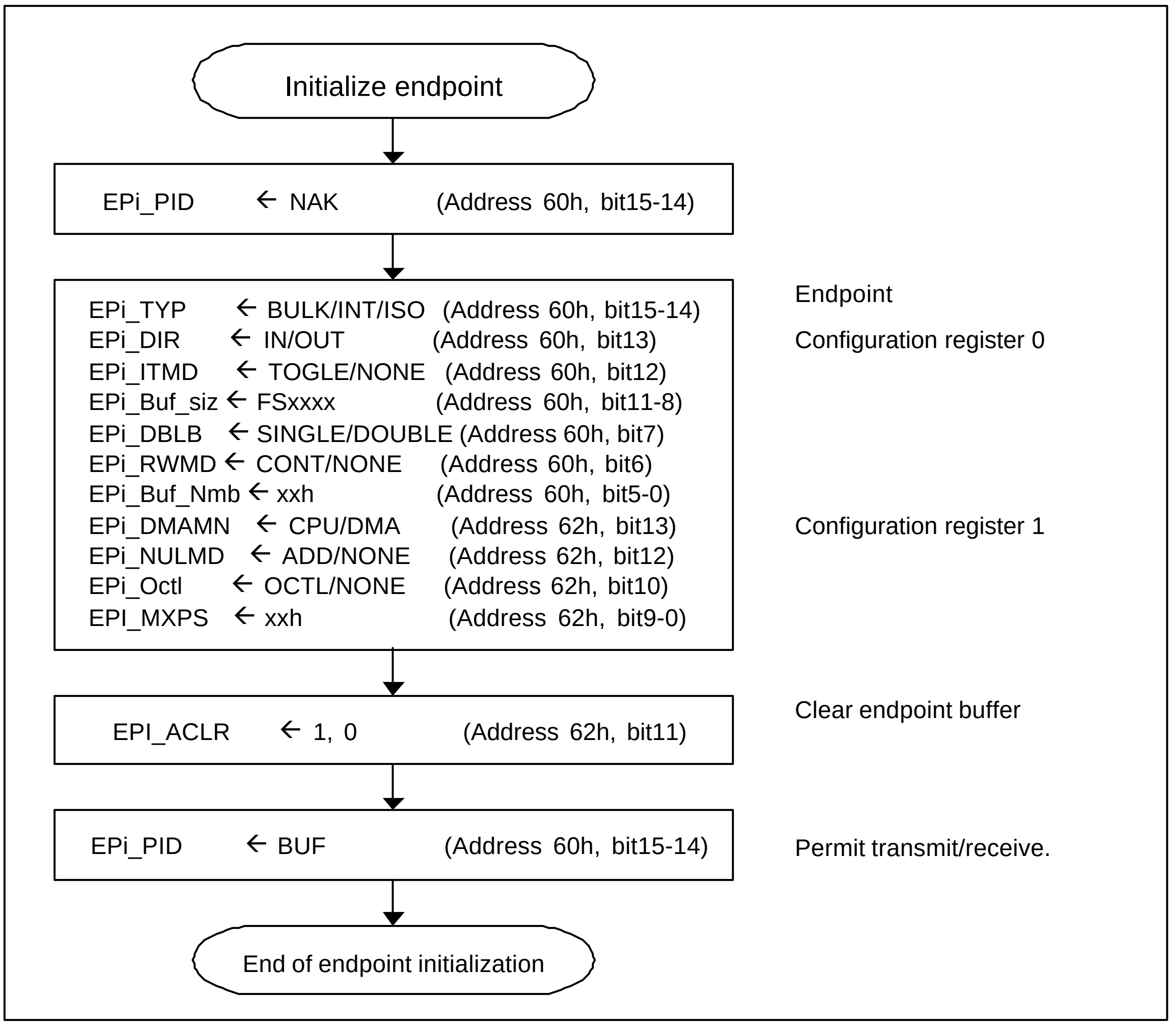
(1) Abundant endpoint interrupts [See 3.5 and 6.3 of this manual]
(2) FIFO buffer of a total of 3K bytes [See 3.6, 6.4, and 6.5 of this manual]
(3) Continuous transmit/receive function (bulk transfer only) [See 3.6]
(4) Enables transfer conditions for each endpoint [See 3.6 of this manual]

The M66290 is equipped with a double buffer function. Therefore, buffers can be accessed without having to wait for a free FIFO buffer while the device is communicating with the host. By using the double buffer function, data transfer of high USB bus efficiency can be achieved.

The M66290 is also equipped with a continuous transmit/receive function for bulk transfer only. In data transfer, the M66290 has a function that transmits and receives data continuously up to 1K bytes without issuing an interrupt. In bulk transfer, if the continuous transmit/receive function is used, FIFO access is not required for each transfer, enhancing the CPU performance.

*The M66290 automatically issues a “NAK” response when receiving a Token packet from the host if the FIFO buffer cannot be accessed due to Full/Empty.

See below for the flow of endpoint initialization processing.



Initialize endpoint
EPi_PID ← NAK (Address 60h, bit15-14)
EPi_TYP ← BULK/INT/ISO (Address 60h, bit15-14)
EPi_DIR ← IN/OUT (Address 60h, bit13)
EPi_ITMD ← TOGLE/NONE (Address 60h, bit12)
EPi_Buf_siz ← FSxxxx (Address 60h, bit11-8)
EPi_DBLB ← SINGLE/DOUBLE (Address 60h, bit7)
EPi_RWMD ← CONT/NONE (Address 60h, bit6)
EPi_Buf_Nmb ← xxh (Address 60h, bit5-0)
EPi_DMAMN ← CPU/DMA (Address 62h, bit13)
EPi_NULMD ← ADD/NONE (Address 62h, bit12)
EPi_Octl ← OCTL/NONE (Address 62h, bit10)
EPI_MXPS ← xxh (Address 62h, bit9-0)
Endpoint
Configuration register 0
Configuration register 1
EPI_ACLR ← 1, 0 (Address 62h, bit11)
Clear endpoint buffer
EPi_PID ← BUF (Address 60h, bit15-14)
Permit transmit/receive.
End of endpoint initialization

6.3. Data Transmit/Receive Interrupt

The M66290 is equipped with three types of data transfer interrupt functions; they are Buffer Ready, Buffer Not Ready, and Buffer Empty. For each interrupt, the endpoint from which the interrupt factor occurred can be recognized from status registers 1 to 3. Note that the occurrence factor of a data transfer interrupt varies according to the tranfser type, transfer direction, and access method. (For the difference of factors, see [3.5. Data Transfer Interrupt].)

See below for the flow of data transmit/receive processing.

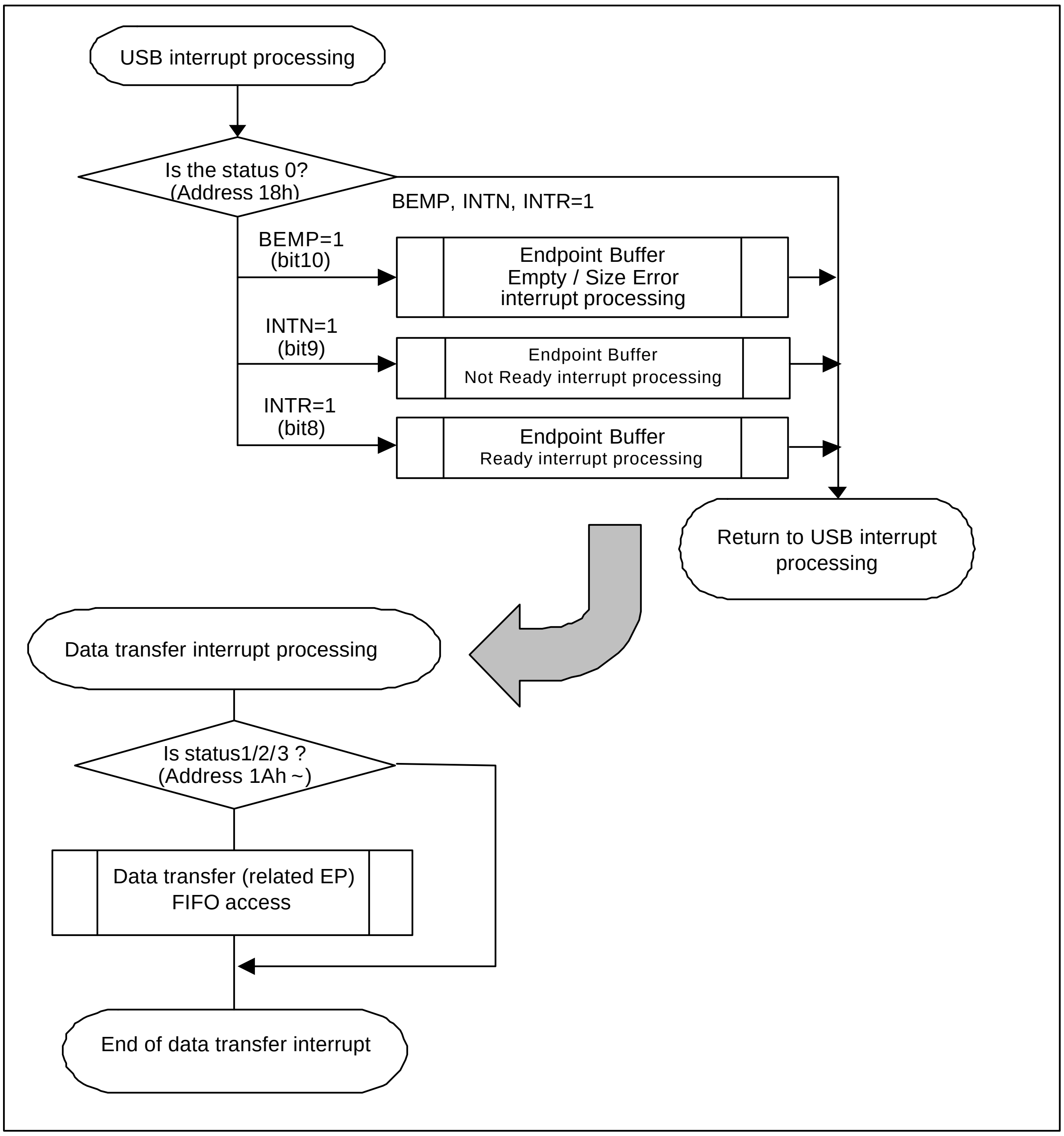
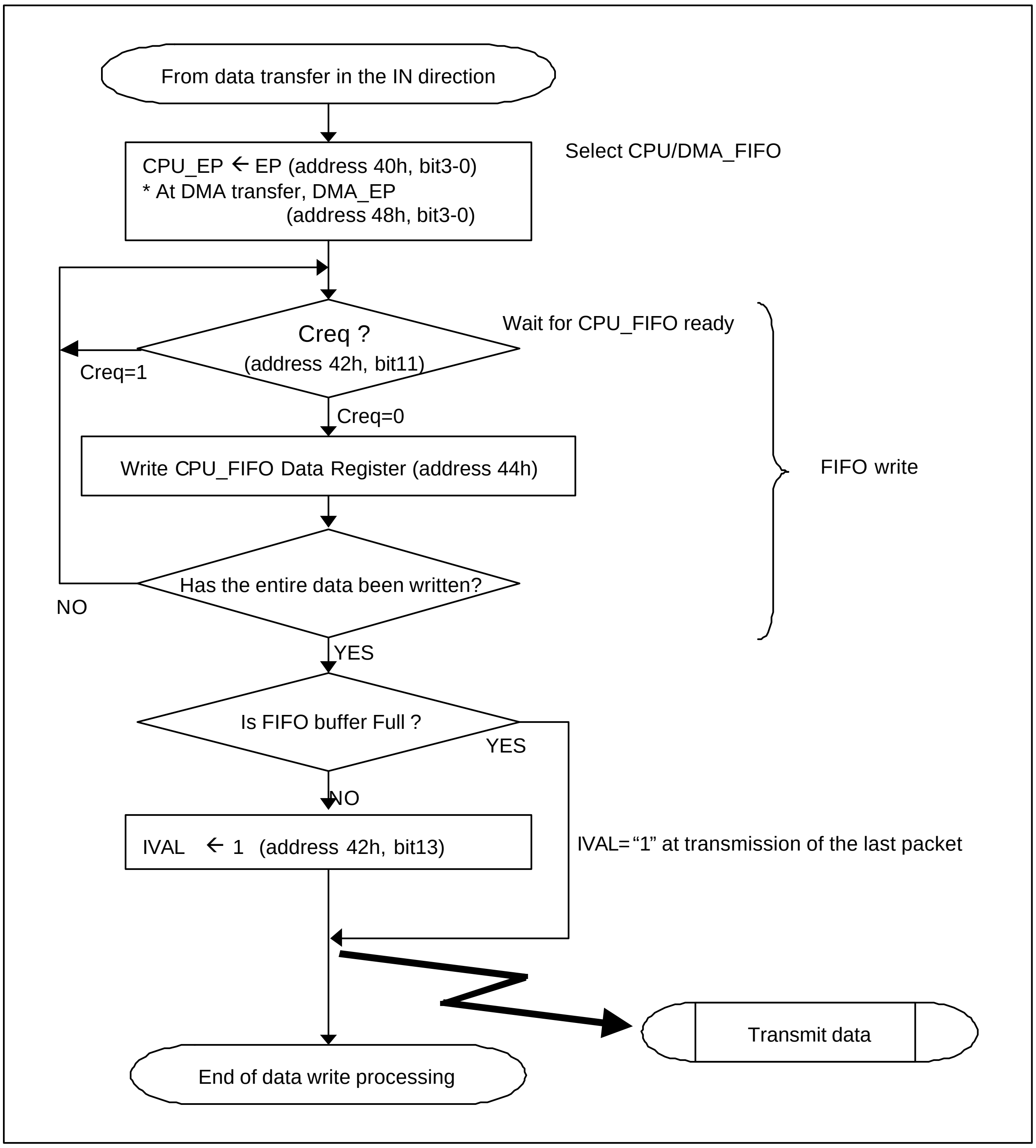

USB interrupt processing
Is the status 0? (Address 18h)
BEMP, INTN, INTR=1
BEMP=1 (bit10)
Endpoint Buffer Empty / Size Error interrupt processing
INTN=1 (bit9)
Endpoint Buffer Not Ready interrupt processing
INTR=1 (bit8)
Endpoint Buffer Ready interrupt processing
Return to USB interrupt processing
Data transfer interrupt processing
Is status1/2/3 ? (Address 1Ah ~)
Data transfer (related EP) FIFO access
End of data transfer interrupt

Figure 6-2 Example of Data Transmit/Receive Control Procedure

6.4. FIFO Access (Data Transmit: IN Direction)

When there is data to be transmitted to the device side, the data is transferred to the host by the IN transaction. In USB communication, data is transferred by host PC sampling. Therefore, the device writes data to be transmitted to the FIFO buffer of the M66290 in advance and waits for the IN token from the host. The M66290 notifies that the write processing to the FIFO buffer is enabled by termination of data transfer, by generating an interrupt. (For details of the factor, see [3.5. Data Tranfser Interrupt].)

See below for the flow of data write processing.



From data transfer in the IN direction
CPU_EP ← EP (address 40h, bit3-0)
* At DMA transfer, DMA_EP (address 48h, bit3-0)
Select CPU/DMA_FIFO
Creq ?
(address 42h, bit11)
Wait for CPU_FIFO ready
Creq=1
Creq=0
Write CPU_FIFO Data Register (address 44h)
FIFO write
Has the entire data been written?
NO
YES
Is FIFO buffer Full ?
YES
NO
IVAL ← 1 (address 42h, bit13)
IVAL="1" at transmission of the last packet
Transmit data
End of data write processing

Figure 6-3 Example of Data Transmission Setting Control Procedure

6.5. FIFO Access (Data Receive: OUT Direction)

When a device receives data from a host PC, data is transferred by the OUT transaction. In USB communication, data is transferred by host PC sampling. Therefore, the device waits for an OUT token from the host by emptying the FIFO buffer of the M66290. The M66290 notifies that the FIFO buffer contents can be read by termination of data transfer. (For details of factors, see [3.5 Data Transfer Interrupt].)

See below for the flow of data read processing.

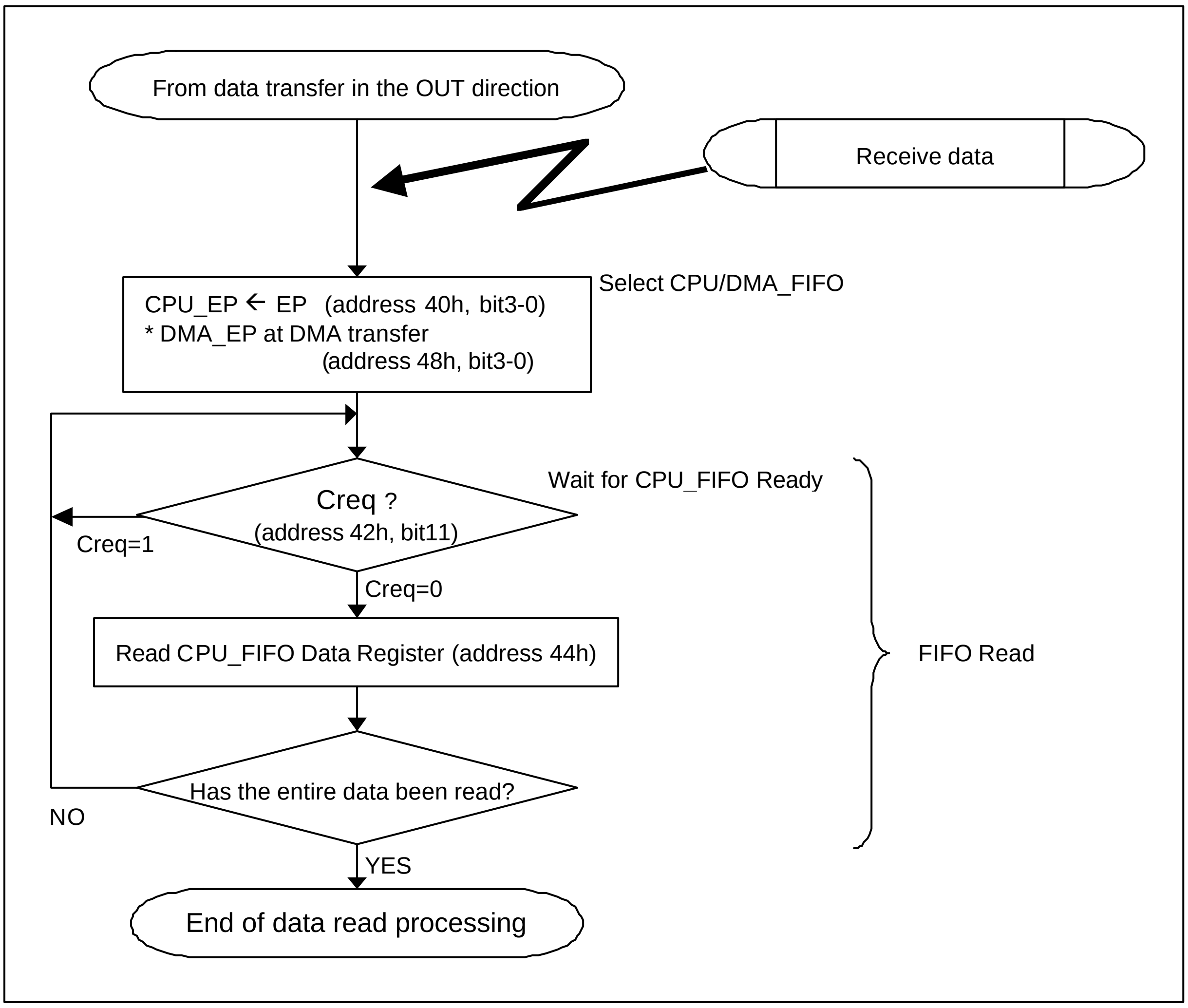

From data transfer in the OUT direction
Receive data
Select CPU/DMA_FIFO
CPU_EP ← EP (address 40h, bit3-0)
* DMA_EP at DMA transfer
(address 48h, bit3-0)
Wait for CPU_FIFO Ready
Creq ?
(address 42h, bit11)
Creq=1
Creq=0
Read CPU_FIFO Data Register (address 44h)
FIFO Read
Has the entire data been read?
NO
YES
End of data read processing

Figure 6-4 Example of Data Receive Setting Control Procedure

7. USB Special Signal Processing

7.1. USB Bus Reset Signal

If an SE0 continues for 2.5µs or more on a USB bus, the M66290 geneartes a device state transition interrupt (DVSQ=001) of the Default state. When detecting a USB bus reset signal, the device must change the address that has been assigned to the Default address and at the same time perform the device state transition according to the USB specification such as changing of the state to the Default state. The M66290 manages device states through hardware. When detecting a USB bus reset signal, the M66290 automatically rewrites the address register to the Default address. After the USB bus reset is detected, the device must prepare for issuing a response for the request (bus enumeration) from the host.

When the internal clock of the M66290 is stopped, a device state interrupt cannot be generated since the USB block is not operating. Consequently, a USB bus reset signal cannot be detected while supply of an internal clock is stopped.

For initialization of M66290 registers as a result of detection of USB bus reset, see the M66290 Datasheet [Register List] .

See below for the processing performed when a USB bus reset signal is detected.

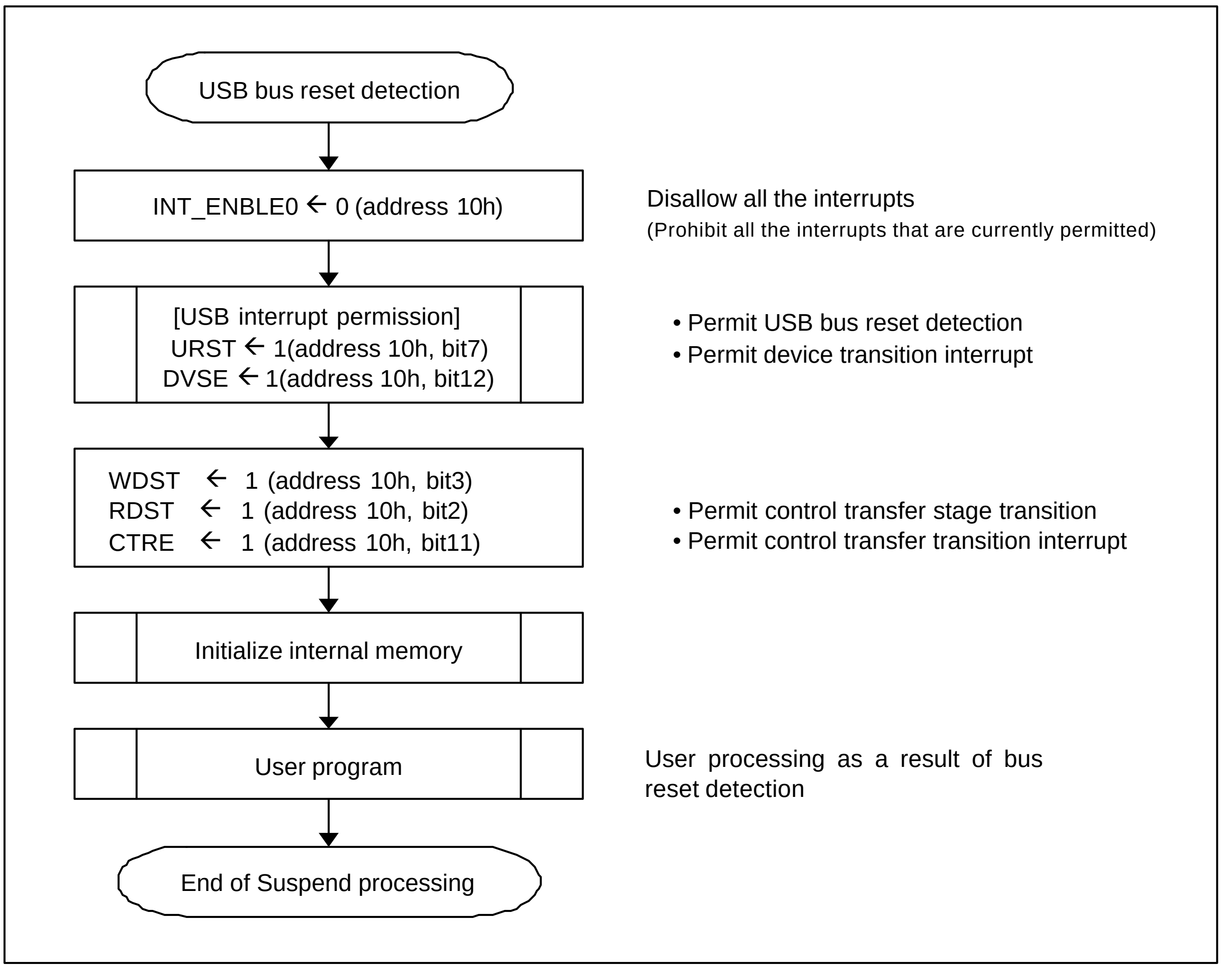

USB bus reset detection
INT_ENBLE0 ← 0 (address 10h)
Disallow all the interrupts
(Prohibit all the interrupts that are currently permitted)
[USB interrupt permission]
URST ← 1(address 10h, bit7)
DVSE ← 1(address 10h, bit12)
• Permit USB bus reset detection
• Permit device transition interrupt
WDST ← 1 (address 10h, bit3)
RDST ← 1 (address 10h, bit2)
CTRE ← 1 (address 10h, bit11)
• Permit control transfer stage transition
• Permit control transfer transition interrupt
Initialize internal memory
User program
User processing as a result of bus reset detection
End of Suspend processing

Figure 7-1 Example of USB Bus Reset Signal Detection Processing

7.2. Suspend Signal

When a USB bus is in an Idle state for 3.0ms or more, the M66290 generates a device state transition interrupt (DVSQ=1xx) of the Suspend state. The device must be changed from any of the USB specified states to a Suspend state. The device in the Suspend state can draw current of less than 500µA only from the USB bus. Therefore, when a Bus power is used, the M66290 must change the mode to a low power consumption mode after detecting a suspend signal.

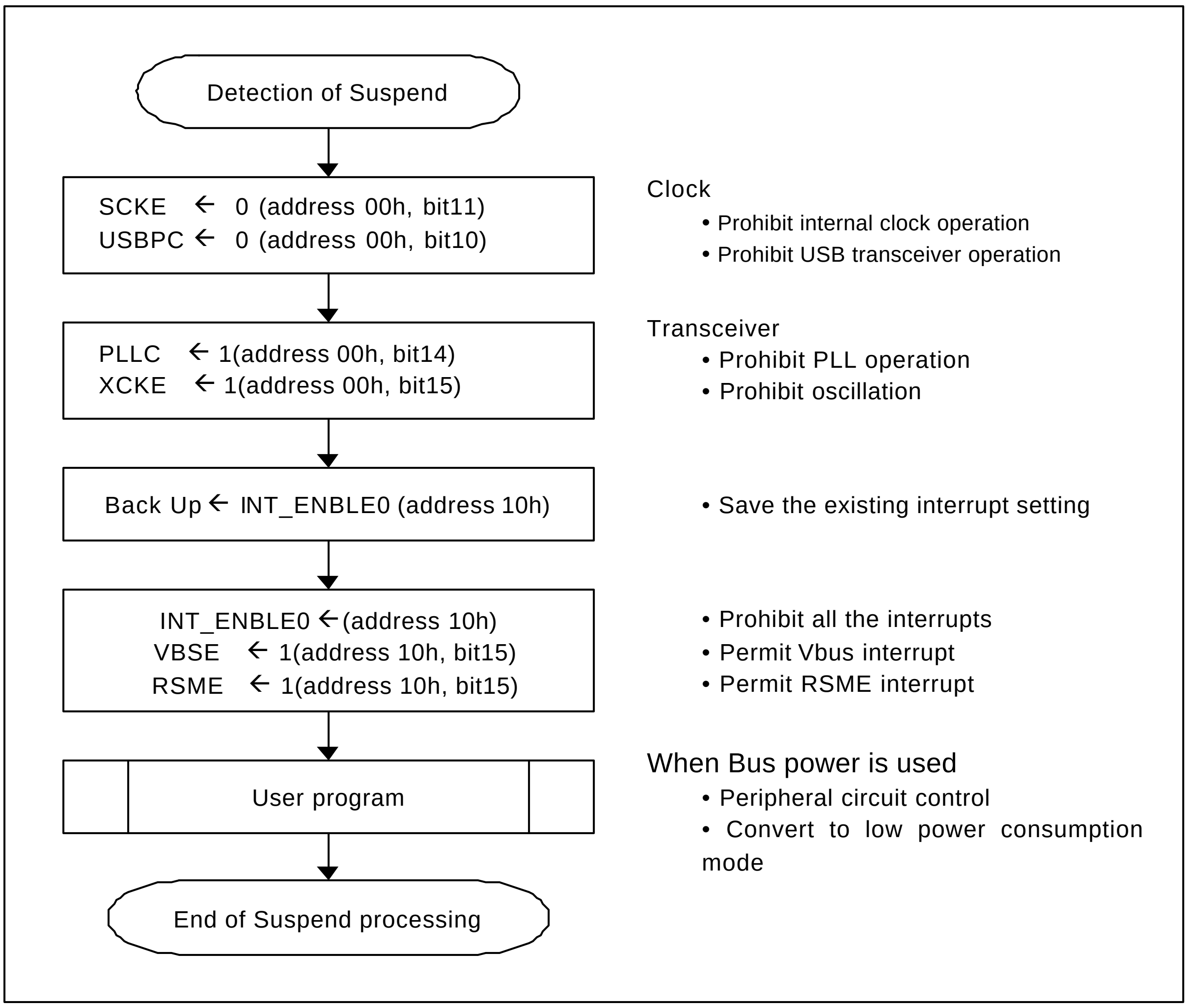
For the device to restart from the Suspend state, some bus activity must be generated or a request must be issued to the host using a remote wakeup function.

The M66290 can detect a bus activity from the RESM interrupt even if supply of an internal clock is stopped. (See [7.3. Resume Detection (Bus Activity)] and [7.4. USB Reset Detection of Suspend State] .)

When the device supports a remote wakeup function, the device sends a remote wakeup signal according to the system specification. (See [7.5 Remote Wakeup].)

When the state of the device is changed to a Suspend state, the setting must be changed so that return detection is enabled (permission of VBUS and RESM interrupts).

See below for the flow of processing performed when a Suspend signal is detected.



Detection of Suspend
SCKE ← 0 (address 00h, bit11)
USBPC ← 0 (address 00h, bit10)
Clock
• Prohibit internal clock operation
• Prohibit USB transceiver operation
PLLC ← 1(address 00h, bit14)
XCKE ← 1(address 00h, bit15)
Transceiver
• Prohibit PLL operation
• Prohibit oscillation
Back Up ← INT_ENBLE0 (address 10h)
• Save the existing interrupt setting
INT_ENBLE0 ←(address 10h)
VBSE ← 1(address 10h, bit15)
RSME ← 1(address 10h, bit15)
• Prohibit all the interrupts
• Permit Vbus interrupt
• Permit RSME interrupt
User program
When Bus power is used
• Peripheral circuit control
• Convert to low power consumption mode
End of Suspend processing

Figure 7-2 Example of Suspend Detection Processing

7.3. Resume Detection (Bus Activity)

When detecting a USB bus status change (from “J” to “K”, or “SE0”) while the device state is a Suspend state, the M66290 generates a Resume detect interrupt (RESM).

The Resume interrupt detects a change of a USB bus status, and the interrupt occurs even if clock supply (internal clock) to the USB block of the M66290 is stopped. An internal clock must be supplied to clear the status flag.

7.4. USB Bus Reset Detection of Suspend State

When supply of an internal clock of the M66290 is stopped, a device state interrupt cannot be generated since the USB block is not operating. Consequently, a USB bus reset signal cannot be detected in a Suspend state while internal clock supply is stopped.

In this case, the USB bus reset signal is returned with a Resume detect interrupt and the signal can be detected by checking the device state (DVSQ=001: Default) after an internal clock is supplied.

See below for the flow of processing performed when a bus activity is detected.

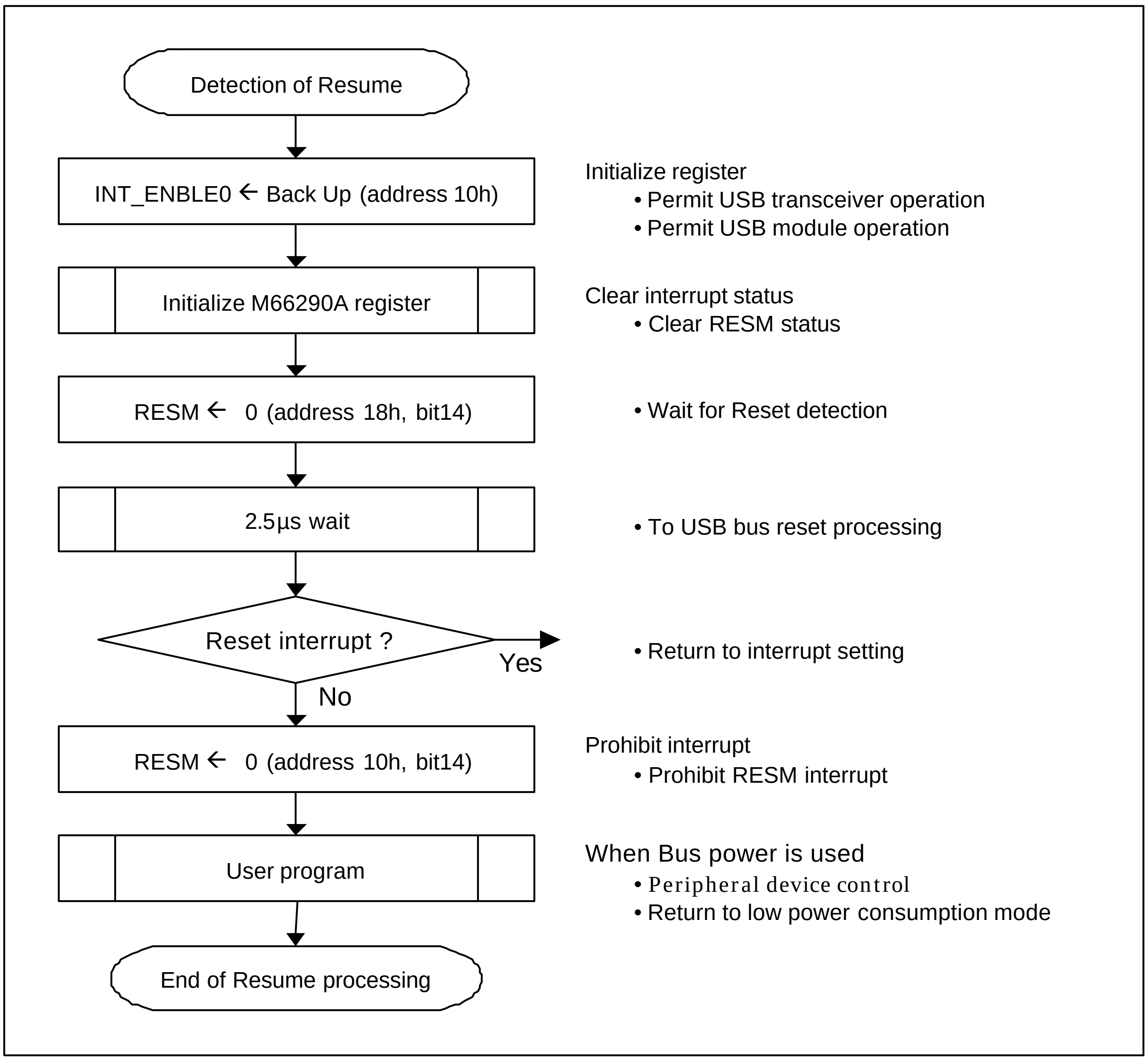

Detection of Resume
INT_ENBLE0 ← Back Up (address 10h)
Initialize register
• Permit USB transceiver operation
• Permit USB module operation
Initialize M66290A register
Clear interrupt status
• Clear RESM status
RESM ← 0 (address 18h, bit14)
• Wait for Reset detection
2.5µs wait
• To USB bus reset processing
Reset interrupt ?
Yes
No
• Return to interrupt setting
RESM ← 0 (address 10h, bit14)
Prohibit interrupt
• Prohibit RESM interrupt
User program
When Bus power is used
• Peripheral device control
• Return to low power consumption mode
End of Resume processing

Figure 7-3 Example of Resume Detection Processing

7.5. Remote Wakeup

When restarting a device in a Suspend state through the factor on the device side, issue a restart request to the host by sending a remote wakeup signal.

The remote wakeup function of the M66290 maintains an idle state for a period of 2ms and outputs a K state for a period of 10ms. (An idle state of 5 ms must be maintained for a remote wakeup signal after a bus state is changed to an idle state. In the USB Specification, since a period of 3ms is required to detect a Suspend signal, a bus idle period of 5ms can be maintained according to the standard even if a remote wakeup signal is sent immediately after detection of a Suspend signal.)

The remote wakeup function is used in a Suspend state in a configuration. Do not use the function in each state in a non-configured state. (See USB Specification Ver.1.1 Chapter 9.4.5 Get Status.)

When a remote wakeup signal is output by this function, the device state is changed to an Address state. Therefore, the device state must be reset to the Configured state by software.

See below for the flow of the processing when a device is restarted by a device factor.

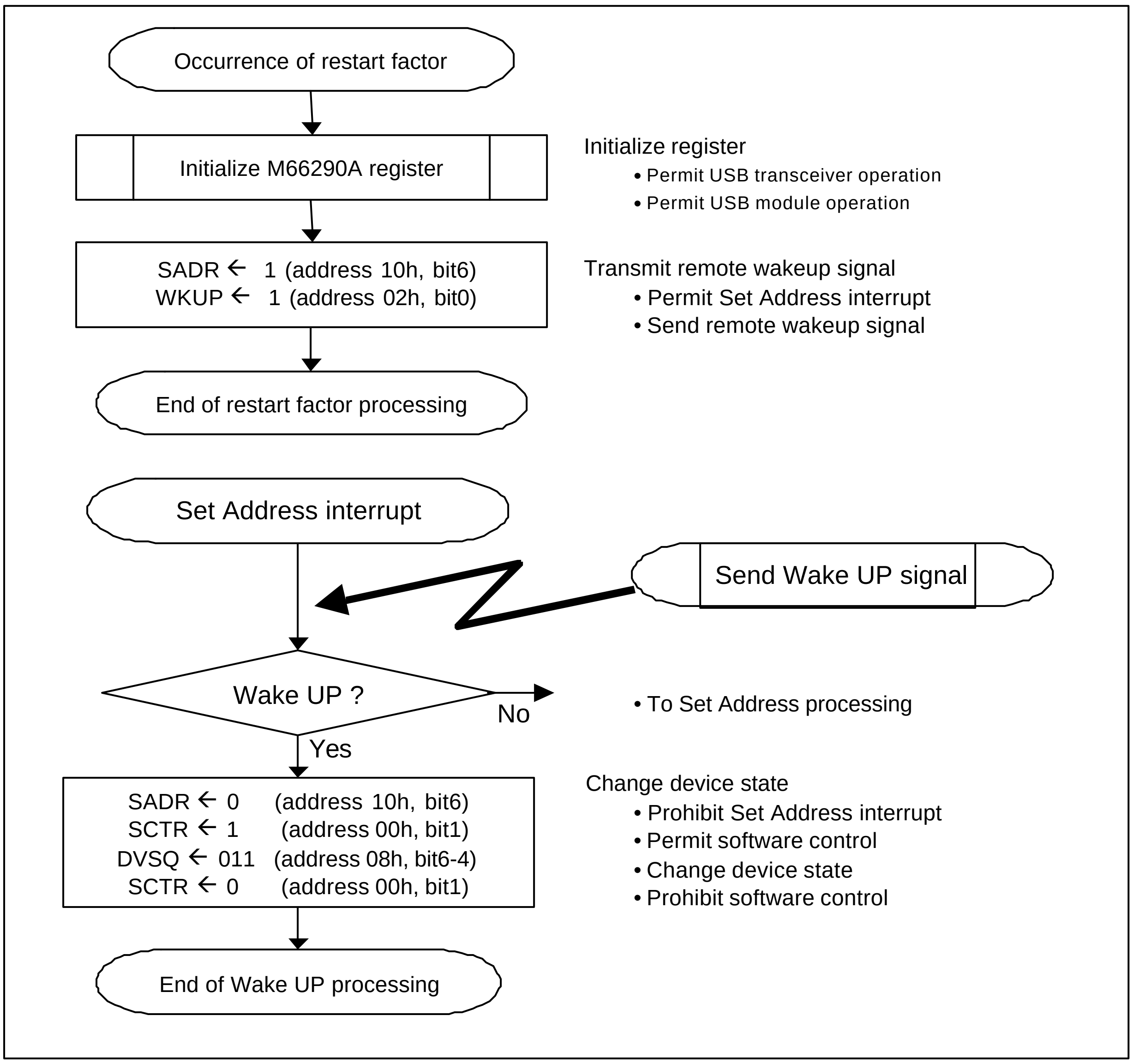

Occurrence of restart factor
Initialize M66290A register
Initialize register
• Permit USB transceiver operation
• Permit USB module operation
SADR ← 1 (address 10h, bit6)
WKUP ← 1 (address 02h, bit0)
Transmit remote wakeup signal
• Permit Set Address interrupt
• Send remote wakeup signal
End of restart factor processing
Set Address interrupt
Send Wake UP signal
Wake UP ?
No
• To Set Address processing
Yes
SADR ← 0 (address 10h, bit6)
SCTR ← 1 (address 00h, bit1)
DVSQ ← 011 (address 08h, bit6-4)
SCTR ← 0 (address 00h, bit1)
Change device state
• Prohibit Set Address interrupt
• Permit software control
• Change device state
• Prohibit software control
End of Wake UP processing

Figure 7-4 Example of Remote Wakeup Processing

7.6. Vbus Detection

When detecting a status change of the Vbus input terminal (can detect both "H"→"L" and "L"→"H" edges) , the M66290 generates a Vbus interrupt (VBUS). Check the Vbus input port bit to identify if a leading edge or trailing edge was detected. A Vbus interrupt occurs even if internal clock supply of the M66290 is stopped.

The status cannot be checked by the Vbus port input while the M66290 internal clock supply is stopped. For Vbus port input in a Suspend state while internal clock supply is stopped, check the port after returning control due to a VBUS interrupt and supplying an internal clock. For the Vbus port, the state must be checked after removing chattering.

See below for the flow of processing performed when Vbus is detected.

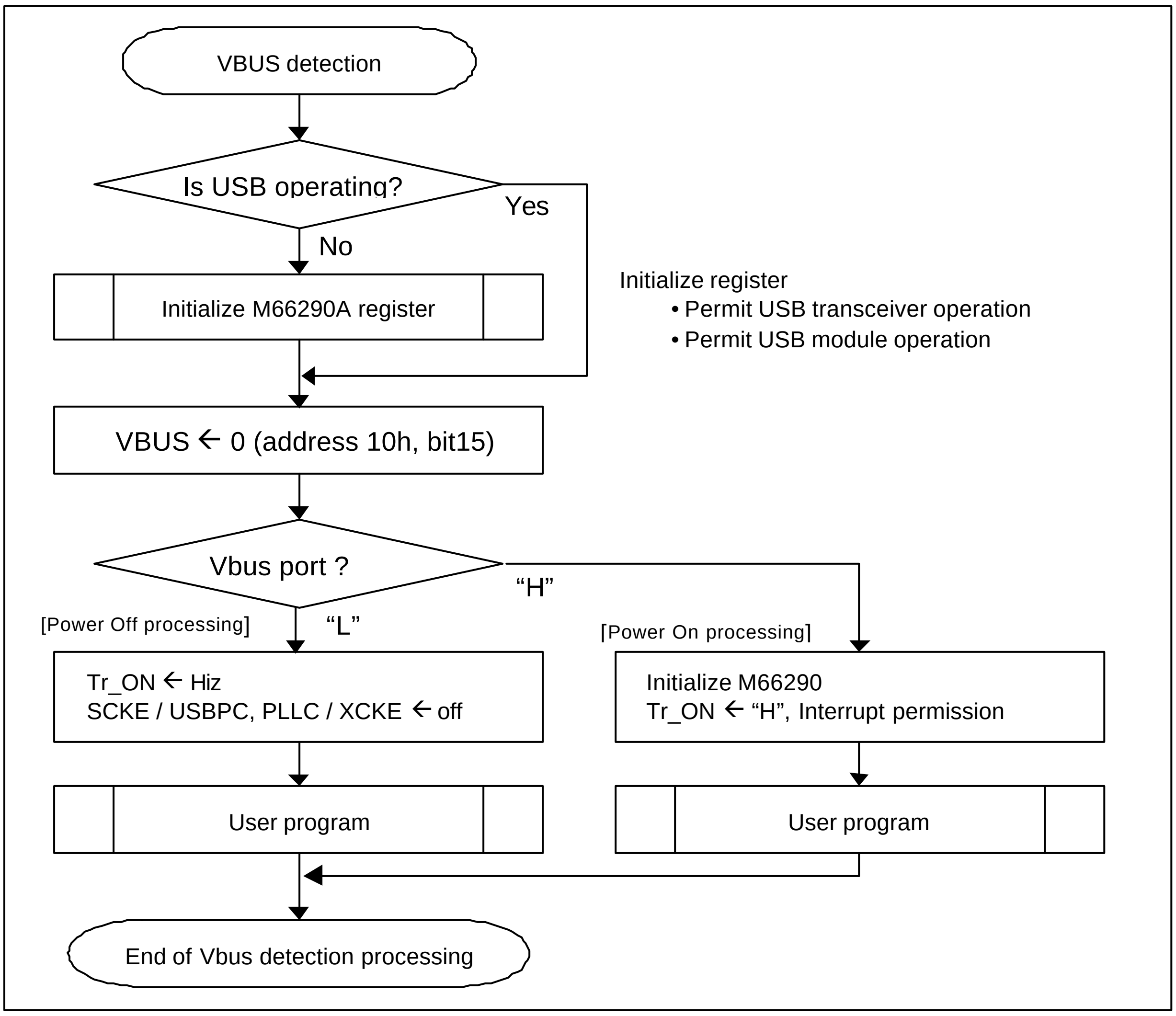

VBUS detection
Is USB operating?
Yes
No
Initialize M66290A register
Initialize register
• Permit USB transceiver operation
• Permit USB module operation
VBUS ← 0 (address 10h, bit15)
Vbus port ?
"H"
[Power Off processing]
"L"
[Power On processing]
Tr_ON ← Hiz
SCKE / USBPC, PLLC / XCKE ← off
Initialize M66290
Tr_ON ← "H", Interrupt permission
User program
User program
End of Vbus detection processing

Figure 7-5 Example of Vbus Detection Processing

Table 7-1 M66290 Initialization Processing by Vbus Detection and Powered Status

Power	Vbus	Vcc	M66290	Peripheral		VBUS
Bus	Off-On	On	Main initialization	Configured	Reset by cable insertion	Not required
Bus	On-Off	On	Stop	Power off	Power Off by cable removal	Required
Self	Off-On	On	Vbus initialization	Main	Cable insertion detection by VBUS	Required
Self	On-Off	On	Stop		Cable removal detection by VBUS	Required
	----	Off	----	----	Not exist	

8. Application Examples

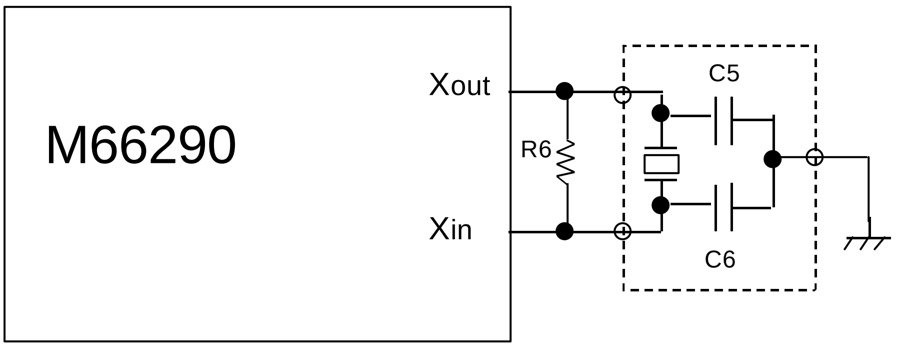
8.1. Oscillation Circuit

This circuit example shows of an oscillation circuit of the substrate that was manufactured by Mitsubishi. The optimum circuit consatnt may change according to the substrate. Check with the oscillator manufacturer for details.

It is recommended to use a stable power supply and a bypass capacitor between the Vcc and GND of the M66920 when configuring a circuit.

Figure 8-1 shows an example when ceramic oscillator that were manufactured by Murata Seisakusho are used.

Figure 8-2 shows an example when a crystal oscillator that was manufactured by Daishinku is used.



Specification and circuit constant of ceramic oscillator

Manufacturer	Murata Manufacturing
Item name	CSTCW4800MX41( )
Oscillation frequency	48MHz
R6	1MΩ
C5	(6pF)
C6	(6pF)

C5 and C6 are incorporated in oscillator

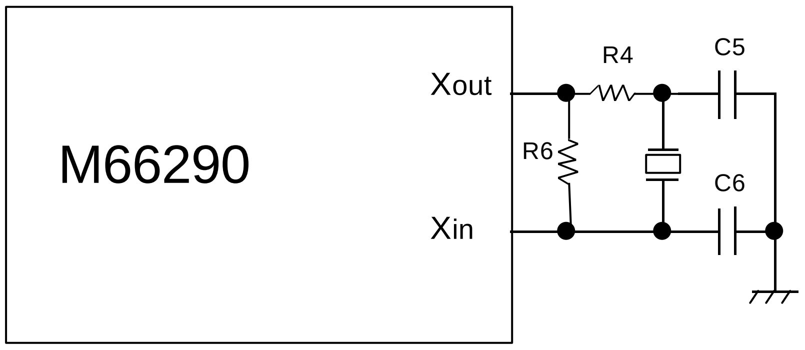
Figure 8-1 Oscillation Circuit Example 1 (Ceramic Oscillator Manufactured by Murata Manufacturing)

*Note

"41( )" at the end of the item name shown above indicates the frequency narrow deviation type.

Note that since a high precision oscillation frequency is required when the oscillator is used for USB, the oscillator must be evaluated for each substrate used (whenever the substrate is changed).

It is recommended for the manufacturer to attach the custom item number to the item name ( ) for each substrate by selection of an oscillator frequency combination. Check with the oscillator manufacturer for details.



Specification of crystal oscillators and circuit constants

Manufacturer	Daishinku		
Item name	SMD-49	DSX630G	DSX630G
Oscillation frequency	6MHz	12MHz	24MHz
Load size	16pF	12pF	12pF
R4	1.0k Ω	680 Ω	330 Ω
R6	1M Ω	1M Ω	1M Ω
C5	22pF	15pF	15pF
C6	27pF	15pF	15pF

Figure 8-2 Oscillation Circuit Example 2 (Crystal Oscillator Manufactured by Daishinku)

8.2. CPU Connection Method

When access between the M66290 and CPU is examined, CPU Endians must be considered. See Table 8-1 lists Endians of the CPUs that are assumed to be connected to the M66290.

Table 8-1 CPUs that Are Assumed to be Connected to M66290 and the Endians

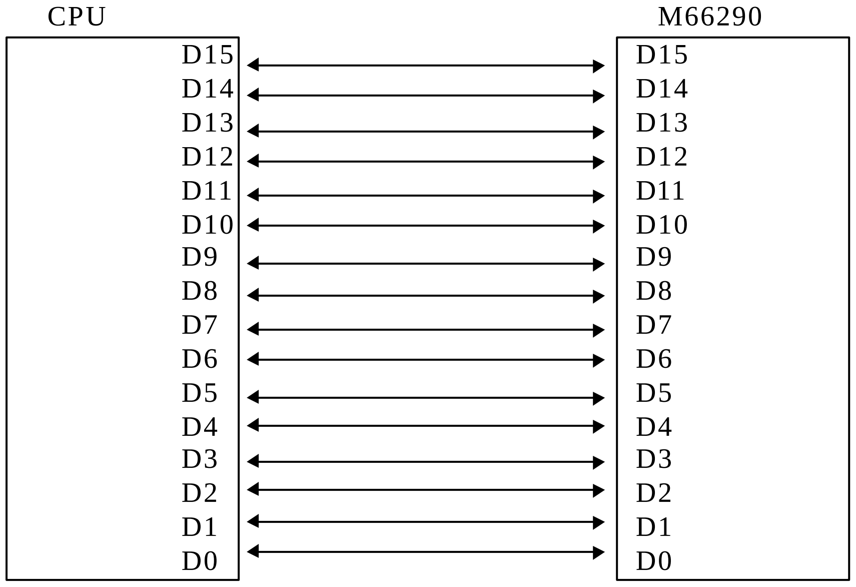
No.	Bit Endian	Byte Endian	CPU
1	Little	Little	7700 Series, 7900 Series, and M16C family
2	Little	Big	Other manufacturers' microcomputer
3	Big	Little	Examination is omitted since it is rare
4	Big	Big	M32R family

This section describes connection with each CPU and the method of connecting with the M66290 and the associated problems. The M66290 can be accessed by setting access through CPU or through DMA transfer. Problems in the access methods are also discussed below.

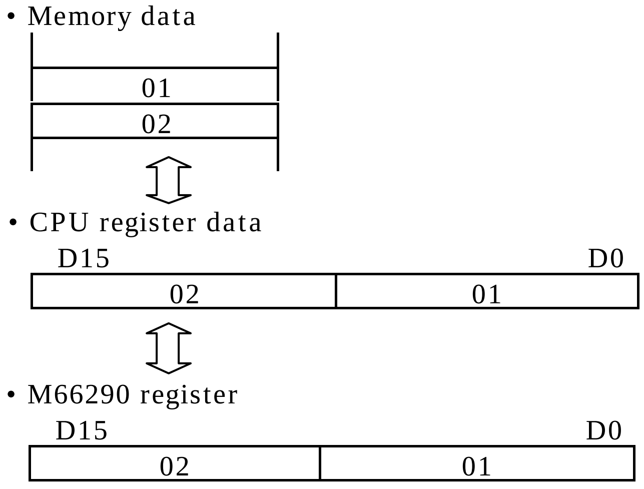
(1) When the bit Endian of the CPU is little and the byte Endian of the CPU is little

The Mitsubishi 7700 Series, 7900 Series, and M16C Family belong to this CPU type.

< Connection >



< Data transfer example >



< Access method >

• CPU access to the M66290 registers

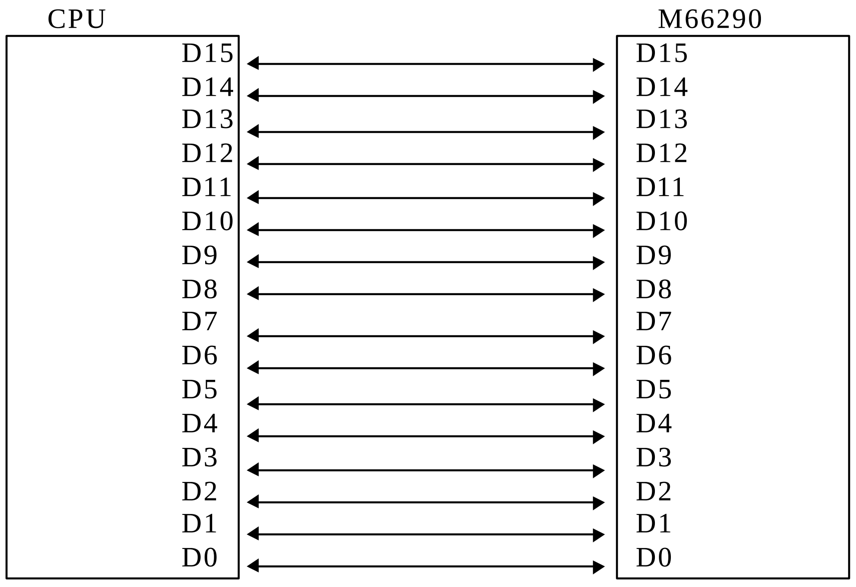
The data in the M66290 registers are read to the CPU registers or memory as it is. Data in CPU registers or memory is written to registers of the M66290 as it is.

• CPU access to FIFO of the M66290
The data that was read from FIFO of the M66290 is sequentially stored in the memory as it is.
The data that is stored in the memory is sequentially written to FIFO in the M66290 as it is.

• DMA access to FIFO of the M66290
The data that was read from FIFO of the M66290 is transferred to the memory as it is.
The data that is stored in the memory is transferred to FIFO of the M66290 as it is.

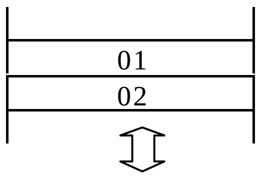
(2) When the bit Endian is little and the byte Endian is big 1
Mitsubishi does not use this CPU type for the microcomputers, however, other manufacturers use this CPU type.

< Connection >

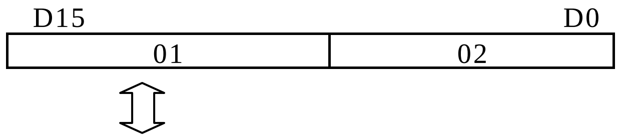

CPU
M66290
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

< Data transfer example >

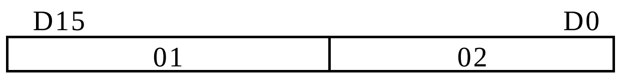
• Memory data



• CPU registers



• M66290 registers



< Access methods >

• CPU access to M66290 registers
The data of the M66290 registers is read to registers of CPU or the memory as it is.
The data in CPU registers or memory is written to registers of the M66290 as it is.

• CPU access to FIFO of the M66290
The data that was read from FIFO of the M66290 is sequentially stored in the memory by being reversed in byte units.
The data that is stored in memory is sequentially written to FIFO of the M66290 by

being reversed in byte units.

- DMA access to FIFO of the M66290

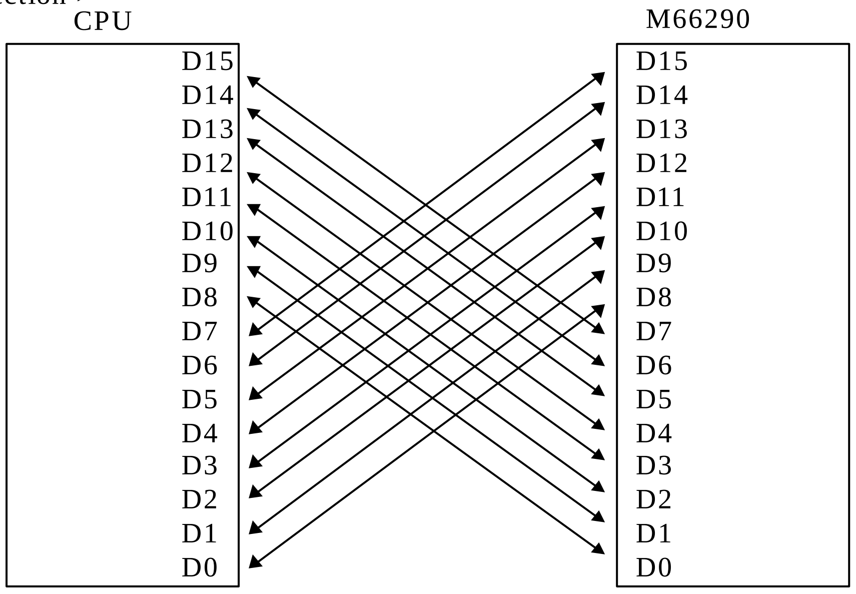
  The data that was read from FIFO of the M66290 is transferred to the memory by being reversed in byte units.

  The data that is stored in memory is transferred to FIFO of the M66290 by being reversed in byte units.

(3) When the bit Endian is little and the byte Endian is big 2

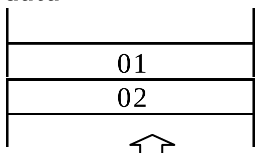
Mitsubishi does not use this CPU type for the microcomputers, however, other manufacturers use this CPU type.

< Connection >

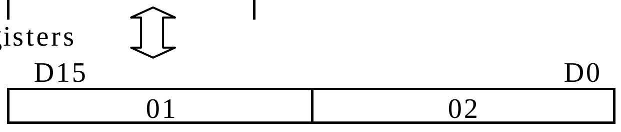


< Data transfer example >

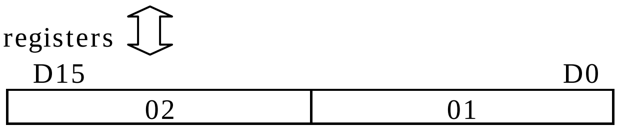
- Memory data



- CPU registers



- M66290 registers



< Access methods >

- CPU access to registers of the M66290

  Data in M66290 registers is read to CPU registers or the memory by being reversed in byte units.

  Data in CPU registers or memory is written to M66290 registers by being reversed in byte units.

- CPU access to FIFO of the M66290

  The data that was read from FIFO of the M66290 is sequentially stored in the memory as it is.

  The data that is stored in memory is sequentially written to FIFO of the M66290.

- DMA access to FIFO of the M66290

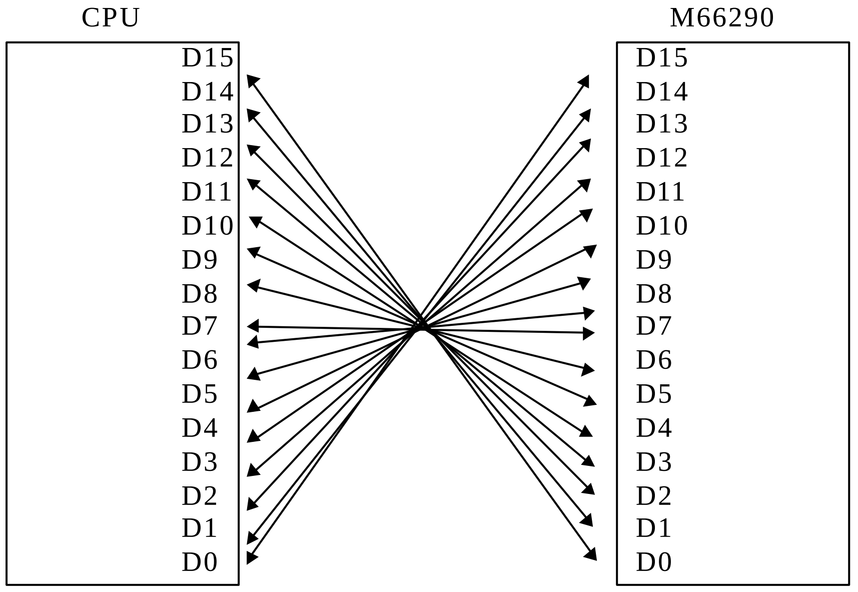
  The data that was read from FIFO of the M66290 is transferred to memory as it is.

  The data that is stored in memory is transferred to FIFO of the M66290 as it is.

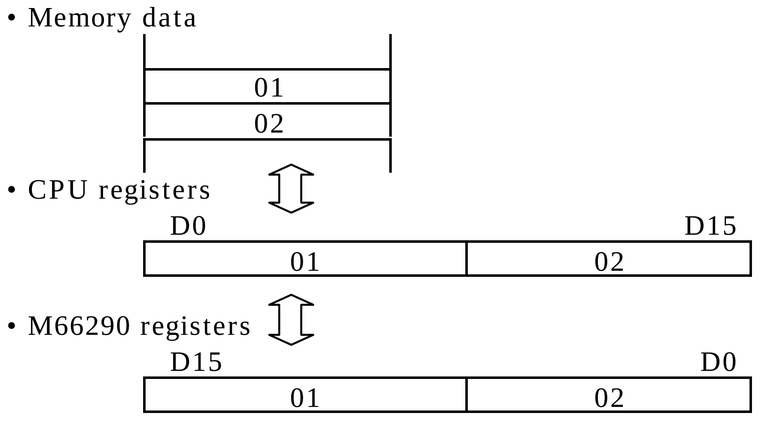
(4) When the bit Endian is big and the byte Endian is big 1

The Mitsubishi's M32R family belongs to this CPU type.

< Connection >



< Data transfer example >



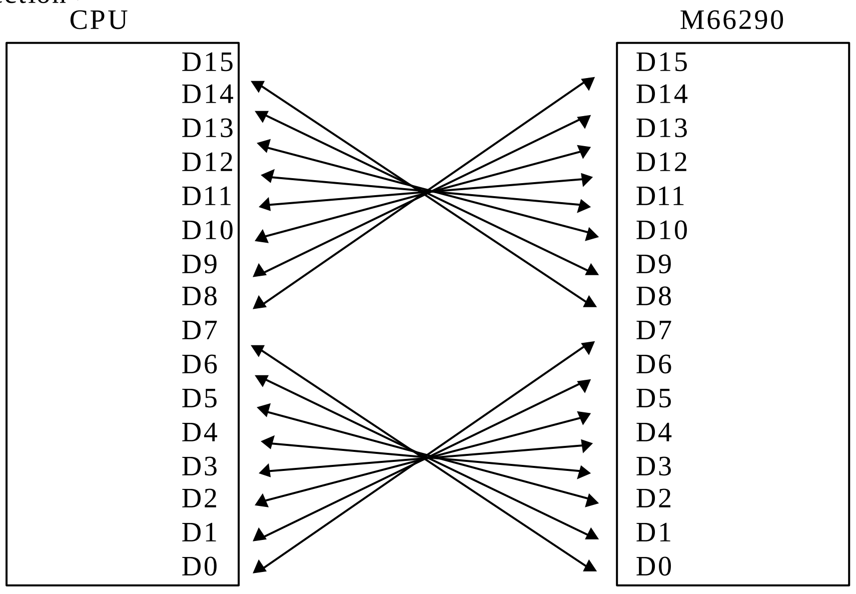
< Access methods >

- CPU access to registers of M66290
  Data in M66290 registers is read to CPU registers or memory as it is.
  Data in CPU registers or memory is written to M66290 registers as it is.
- CPU access to FIFO of the M66290
  The data that was read from FIFO of the M66290 is sequentially stored in memory by being reversed in byte units.
  The data that is stored in memory is sequentially written to FIFO of the M66290 by being reversed in byte units.
- DMA access to FIFO of the M66290
  The data that was read from FIFO of the M66290 is transferred to the memory by being reversed in byte units.
  The data that is stored in memory is transferred to FIFO of the M66290 by being reversed in memory units.

(5) When the bit Endian is big and the byte Endian is big 2

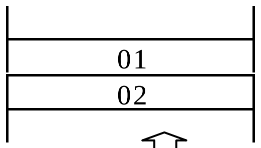
The Mitsubishi's M32R family belongs to this CPU type.

< Connection >

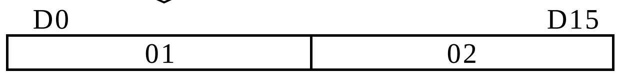


< Data transfer example >

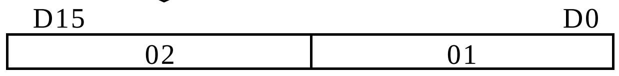
• Memory data



• CPU registers



• M66290 registers



< Access methods >

• CPU access to registers of the M66290

Data in M66290 registers are read to CPU registers or the memory by being reversed in byte units.

Data in CPU registers or the memory is written to M66290 registers by being reversed in byte units.

• CPU access to FIFO of the M66290

The data that was read from FIFO of the M66290 is sequentially stored in the memory as it is.

That data that is stored in memory is sequentially written to FIFO of the M66290 as it is.

• DMA access to FIFO of the M66290

The data that was read from FIFO of the M66290 is transferred to the memory as it is.

The data that is stored in the memory is transferred to FIFO of the M66290 as it is.

Table 8-2 summarizes the connectibility between CPUs of various Endians and the M66290.

Table 8-2 Connectability of the M66290 According to the CPU Endians

No.	Bit Endian	Byte Endian	Connection method	CPU access	DMA access
1	Little	Little	Connection method (1)	O	O
2	Little	Big	Connection method (2)	O	X
3	Little	Big	Connection method (3)	X	O
4	Big	Big	Connection method (4)	O	X
5	Big	Big	Connection method (5)	X	O

O : The data can be read or written as it is.
X : The data can be read or written by reversing in byte units.

The major objective in connection between the M66290 and CPU is to transfer data in DMA access as it is without reversing the data in byte units. This objective can be achieved by applying measures in programs in CPU access.
When the M66290 is actually used, the connection must be selected according to the customer's usage method.

9. Q & A

9.1. Q & A Relating to Substrate Design

1	[Q]	Are there any cautions in designing patterns?
	[A]	1) Avoid cross over of D+ and D- lines as much as possible when designing. 2) General FR4 can be used as the substrate material and no special care is required. 3) In general, the line is about 0.8mm in width and the length to the connector is about 50mm.
2	[Q]	How can EMI measures be applied?
	[A]	Insert a common mode choke coil in the signal lines (D+, D-). However, as a result the quality of the D+ and D- signals deteriorates.
3	[Q]	How can EMC measures be applied?
	[A]	Insert ferrite beads in the Vbus line and GND line. Since an inductor is inserted in the GND line and Vbus serially, a component with low serial resistance and rapid acceleration of the inductance feature must be selected. However, as a result, the quality of the D+ and D- signals deteriorates.
4	[Q]	Is it necessary to insert a protection circuit in D+ and D-?
	[A]	The current USB Specification does not discuss a protection circuit. However, since USB specifies hot line connection, many actual products introduce ESD protection.
5	[Q]	Is there any problem in using VBUS to a USB ASPP by connecting to +5.0V on the substrate, not from a cable?
	[A]	There is no problem. However, Vbus interrupt cannot be checked by inserting and removing the plug.
6	[Q]	How can JTAG pins be processed when JTAG is not used?
	[A]	_TRST(8pin) : Connect the same hardware reset signal as the 46pin _RST signal. TCK:NC TMS : NC TDI : NC TDO : NC
7	[Q]	Can the M66290 run at 2.7V?
	[A]	The USB Specification 1.1 specifies the D+/D- signal output level(V_{OH}) to be Min.:2.8V. The M66290 cannot run at voltage 2.7V since it does not satisfy the VOH specification.
8	[Q]	Is 3.3V input allowed for a M66290 Vbus terminal? *5V or 3.3V may be input from a USB connector Vbus terminal.
	[A]	The USB Specification specifies the Vbus voltage to be 5V(4.75-5.25V) . Since the M66290 complies with this specification, 5V (4.75-5.25V) , the M66290 does not run with 3.3V. See the Suggestion 1) and 2) below. 1) Input 5V of the system to the M66290 Vbus terminal and use the TrON function(D+ terminal pull-up function). The Vbus detection circuit (cable insertion/removal detection circuit) is configured by the external hardware and generates an interrupt different from that of the M66290. 2) with the external hardware, form an electrical switch that supplies 5V of the system to a Vbus terminal by input of the Vbus signal (5V or 3.3V) from the USB connector. By using this method, both the TrON function and Vbus detection circuit of the M66290 can be used. Note: When there are INT terminals that are remaining for the control CPU, 1) is simpler in terms of hardware configuration. If there are no remaining INT terminals, 2) is recommended since 1) causes complicated circuit configuration.
9	[Q]	What minimum voltage and mA does the Vbus terminal require?
	[A]	Input 4.75V or more to Vbus. The current consumption of Vbus is 2.2mA during operation when TrON output is connected to the D+ pull-up resistance. (D+ is connected t the 0V drive.)
10	[Q]	Is any care necessary for the M66290 data bus, address, and control signal lines in circuit design?
	[A]	It is recommended to apply a pull-up (pull-down) resistance for the M66290 data bus, address, and control signal line. When CS is not selected, the M66290 is set to a state where it does not drive a data bus (input stage) and the terminal is set to a floating state. There will be no problem if the potential of the data bus has been defined. However, if none of the devices that are connected to the data bus drive, the potential becomes undefined, causing flow of lead-through current. Since the address bus is always in an input stage, a pull-up or pull-down resistance is recommended. The same concept applies to the control signal line.

9.2. Q & A Relating to Control Transfer

1	[Q]	What response is issued if the host issues Clear Feature of EP0 after response PID of Endpoint0 is set to STALL?
	[A]	When STALL is automatically responded from Endpoint0, EP0_PID=STALL is set until the next SETUP token is received. When receiving a new SETUP token, the M66290 automatically issues an ACK response, setting EP0_PID=NAK. Therefore, ACK response is issued for Clear Feature and EP0_PID=NAK is set automatically.
2	[Q]	Is a status stage terminated with EP0_PID=BUF and CCPL=1 after receiving Clear Feature when Endpoint0 is in a STATL state?
	[A]	After receiving Clear Feature, the M66290 must terminate EP0_PID with BUF and CCPL=1.
3	[Q]	Is the response PID remain STALL after a STALL response is issued for the status stage with response PID=STALL, CCPL=1 of Endpoint0 or is the hardware automatically set to BUF?
	[A]	When the M66290 sets EP0_PID=STALL and responds STALL from Endpoint0, EP0_PID=STALL remains set until the next SETUP token is received. When receiving a new SETUP token, the M66290 automatically responds with ACK, setting EP0_PID=NAK.
4	[Q]	Is return to BUF to be performed manually after STALL is set in response PID regardless of whether a STALL response is issued to the host by hardware automatically or the response is issued manually by software?
	[A]	No. In either case, hardware automatically releases STALL after STALL is set in response PID and the next SETUP token is received and sets NAK for the response PID after responding with ACK.
5	[Q]	When release of STALL PID is attempted manually by software, STALL should be responded to the host in the stage only. STALL should have been released in the next stage (probably setup stage). What is the release timing for this USB ASSP? (Is there any method for detecting the return of STALL for the host?)
	[A]	The timing of returning STALL after the setting of a STALL PID response cannot be detected. Response PID of Endpoint0 need not be released after it is set to STALL. See the answer to question in 9.2.4.
6	[Q]	In Control Write, response PID is set to NAK automatically after completion of a setup stage. When data of 255 bytes is received in the data stage subsequently, the NAK should be issued to subsequent data as the response (the response PID remains BUF). Is ACK responded again when the buffer comes empty?
	[A]	In continuous control transfer (for both Write and Read), only the data of 255 bytes can be transmitted or received according to the specification. Set a non-continuous transmit/receive mode when transmitting or receiving data of 256 bytes or more.
7	[Q]	In Control Read, is it possible to assume that the response PID remains BUF unless a sequence error occurs when the IN token in the length is less than the continuous transmit data length register or the IN token in the length exceeding the continuous transmit data length register is received? Is the response PID changed to NAK at the next setup stage?
	[A]	Yes
8	[Q]	For only the release of STALL when EP0 is specified, the response PID is set to BUF after being set to NAK. Is it necessary?
	[A]	Since EP0 processing (Control transfer) is running, NAK response is set until processing is terminated.
9	[Q]	What actual processing such as STALL is performed when the specified Endpoint does not exist or the request type cannot be checked?
	[A]	EP0_PID = STALL is set and STALL is responded at the status stage.
10	[Q]	What actual processing such as STALL is performed when Interface is specified as the request type?
	[A]	EP0_PID = STALL is set and STALL is responded at the status stage.

11	[Q]	What response is issued when the IN token that was issued exceeded the length of the continuous transmit data length register in continuous transfer mode?
	[A]	When the IN token in the length exceeding the continuous transmit data length register is issued, the response is determined by the response PID. When the response PID is BUF, a NAK response is issued. (In this case, since a BEMP interrupt occurs at completion of the entire data transmission, the response can be checked by the software before NAK response.)
12	[Q]	What response is issued when the IN token in the length that is less than the continuous transmit data length register is issued in continuous transfer mode?
	[A]	In control data stage, the stage is changed to a status stage when the M66290 transmits a short packet or receives an OUT packet. Therefore, when the M66290 receives the OUT token following the IN token in the length that is less than the continuous transmission data length register, the stage is changed to a status stage. (A control transfer stage transition interrupt occurs.)
13	[Q]	Response PID is set (BUF) once only in continuous transfer mode. Is it set in packet units?
	[A]	Response PID is set (BUF) once only even if multiple packets are sent in continuous transfer mode.
14	[Q]	When is response PID reset to NAK in continuous transfer mode?
	[A]	Response PID is set to NAK in control transfer only when a SETUP token is received.
15	[Q]	No continuous receive data length register is available in continuous receive mode. Does an interrupt occur only in continuous reception of data of 255 bytes or reception of a short packet?
	[A]	When a continuous receive mode is set and data of 255 bytes or a short packet is received, an INTR interrupt occurs. The USB specification specifies that a data stage of control transfer is to be terminated with a short packet (including a NULL packet).
16	[Q]	If the buffer size is set to 256 in continuous receive mode and data of 64 bytes is receive (not short packet), does an interrupt occur?
	[A]	Since stage management is performed by hardware control transfer, a control transfer stage transition interrupt and an INTR interrupt occurs as a result of transition from a data stage to a status stage. The USB Specification specifies that a data stage in control transfer is to end with a short packet (including a NULL packet).
17	[Q]	If the buffer size is set to 64 in continuous receive mode and data of 64 bytes is received (not a short packet), does an interrupt occur?
	[A]	In continuous receive mode of control transfer, a buffer size cannot be set to 64 bytes.
18	[Q]	In continuous receive mode, response PID is set (BUF) once only. Is it set in packet units?
	[A]	It is not necessary to set response PID for each packet. When receiving a SETUP token, the hardware automatically sets NAK. However, in a setup stage, the M66290 stores the request received through an ACK response in the registers as specified in the USB specification unless a packet error occurs. Transmit/receive processing of a data stage is started by setting BUF with the software after completion of request analysis and preparation for data transmission/reception.
19	[Q]	When response PID is reset to NAK in continuous receive mode?
	[A]	A response in a status stage is controlled by a control transfer complete enable bit. When a SETUP token is received, the hardware automatically sets the control transfer complete enable bit (CCPL) to 0. In this stage, NAK is returned in a status stage regardless of the response PID. To terminate control transfer, reflect the processing result of the data stage in the response PID and set the control transfer complete enable bit (CCPL) to 1.
20	[Q]	VALID of an interrupt status register is not used as an interrupt factor. An interrupt occurs at completion of a setup stage (CTRT=1). Can a USB request be fetched by referencing VALID, which is not an interrupt factor?
	[A]	Fetch the request by referencing VALID.

21	[Q]	When the data length that was received continuously in a Control Write data stage exceeded 255 bytes, does a sequence error occur in the hardware?
	[A]	When data that is received in continuous receive mode exceeds 255 bytes, a sequence error does not occur in the hardware and no sequence error interrupt occurs. Consequently, when data exceeding 255 bytes is to be received, the wLength field of the setup data must be checked. If the data contains 256 bytes or more, switch the mode to a non-continuous receive mode and receive the data in the unit of maximum packet size by using a Buffer Ready interrupt.
22	[Q]	Did the data that was received exceed the buffer size in continuous transfer?
	[A]	When continuous control transfer is set, Endpoint0 requests a buffer size of up to 256 bytes. Therefore, when continuous control transfer is specified, a buffer of up to 256 bytes must be allocated in advance.
23	[Q]	When data of the size equal to the packet size or data of less than the buffer size is received in Control Write continuous transfer (no short packet), does an INTR interrupt occur?
	[A]	Since stage management is performed in control continuous transfer, a Buffer Ready is set when a data stage is changed to a status stage. Consequently, in control transfer, a Buffer Ready is set as a result of reception of a short packet, Buffer Full (255 bytes), or transition to a status stage. When the stage is changed to a status stage, the M66290 does not check that the necessary number of bytes has been received.
24	[Q]	Does setup reception in control transfer always interrupt CPU? (Other than SetAddress and SetConfiguration of automatic response control)
	[A]	When a control transfer stage transition interrupt is permitted, a transfer stage transition interrupt of control transfer always occurs at completion of a setup stage.
25	[Q]	What processing is performed for CLEAR_FETURE?
	[A]	Releases STALL from the specified Endpoint and clears the sequence toggle bit.
26	[Q]	What processing is performed for SET_INTERFACE?
	[A]	Performs alternate setting of the specified interface, releases STALL from the Endpoint used in the new alternate setting, and clears the toggle bit.
27	[Q]	In what timing, response PID is set to BUF and written to FIFO in Control Read mode?
	[A]	Either FIFO write processing or BUF setting can be performed first.
28	[Q]	When Control Write is used, is it possible to assume that data is read from FIFO of EP0 when an EP0 Buffer Ready interrupt occurs?
	[A]	Although the concept is not wrong, a simpler way is to permit WDST (3, 10h) in continuous receive mode and read FIFO when the Control Write stage is terminated.
29	[Q]	How can OUT/IN be checked when EP0 INTR occurs? Since EP0 is bi-directional, how can it be checked whether data was received from the host (read enable) or data can be transmitted (buffer empty)? Does only INTR in the direction that is set in ISEL occur?
	[A]	Endpoint0 is a Control transfer dedicated pipe used as a default pipe. In Control transfer, the direction of the Data stage is determined specifically by the request that is received in the Setup stage. (Reception of data in the reverse direction of the request causes a Control transfer error.) The Setup stage of Control transfer is a response from a request from a host. The data stage in the IN direction is a response for the request. The Data stage in the OUT direction is a request parameter. In either case, request analysis is required.
30	[Q]	Is any processing required by the software when a control transfer sequence error occurs?
	[A]	When a sequence error that can be detected by HW occurs, the HW sends STALL. When a data analysis error such as a request error or a request that has not been handled is detected, PID=STALL is output.

31	[Q]	Although the stage of the M66290 is changed to a status stage by Control_Read, an IN token occurred from the host and STALL was output due to a sequence error. According to the USB specification [1.0] 8.4.5.1 (IN transaction response) , NAK is responded. Is there any problem if STALL is output? According to the specification [1.0] 8.5.2.2 (last transaction error processing) , the M66290 detects an OUT setup transaction and proceeds with a status phase. The IN token seems to be output by BIOS after a short packet.
	[A]	Section 8.5.2.2 of Specification 1.1 indicates that a data stage is to end at transmission of a short packet. Since it is interpreted that STALL is returned for the IN that requests data transmission after transmission of a short packet, there is no problem.
32	[Q]	Is a control transfer sequence error issued after STALL is responded to the host?
	[A]	Assume that a control transfer sequence error occurs at the same timing as for a STALL response.

9.3. Q & A Related to Data Transfer

1	[Q]	An error interrupt occurs when the data that was received exceeds the Maximum packe size in the BEMP interrupt of the OUT direction. In this case, is the data that was received written to FIFO? (Is it necessary to clear the error data with software?)
	[A]	Although the data is written to FIFO, the status indicates that there is no data. The data is automatically cleared by hardware.
2	[Q]	Does an INTR interrupt occur when the data that was received exceeds the Maximum packet size in the BEMP interrupt of the OUT direction?
	[A]	Since there is no data, the buffer is not set to Ready. In the BEMP interrupt, processing to FIFO is not required.
3	[Q]	Does INTR occur again unless data is written to the buffer even if the interrupt status i cleared at the occurrence of the INTR interrupt of the OUT direction?
	[A]	The INTR interrupt status cannot be cleared even if 0 is written to the register. For the OUT direction, INTR occurs when the buffer becomes full or a short packet is received INTR occurs unless the entire data is read from the buffer. For the IN direction, INTR occurs in a buffer write enabled state (empty state). INTR occurs unless transmission data is written to the buffer. In this case, prohibit the INTR interrupt of the Endpoint.
4	[Q]	Doesn’t an interrupt occur when response PID is set to NAK?
	[A]	Although the interrupt type in the question is not clear, I assume that the interrupt is an Endpoint 0-5 Buffer Not Ready interrupt. When the response PID is set to NAK, a Buffer Not Ready interrupt will not occur even if a NAK response is issued to the IN/OUT token. In a setup stage, a request is automatically received and an ACK response is issued regardless of the setting of the response PID. In this case, a Buffer Not Ready interrupt will not occur either.
5	[Q]	Please explain the definition of “Ready” of Buffer Ready in each case of IN, OUT, buffer size, Maximum packet size, and EP, if there are any differences.
	[A]	Ready indicates a Read/Write enabled state for a FIFO buffer. Assume that the M66290 buffer has the following two states. •State where data can be transmitted to and received from a USB •State where data can be read or written from CPU or DMAC (Ready state) <IN direction> Ready indicates a state where the buffer that was set is empty and data can be written to the buffer. By writing data of the buffer size that was set, the buffer is set to a transmission enable state for a USB bus and is no longer in a Ready state. When data with a size less than the buffer size is sent, the buffer is set to a transmission enable state by setting the IVAL bit after data is written and as a result the buffer is no longer in a Ready state. In non-continuous transfer mode, the buffer is changed from a Ready state regardless of the setting of the buffer size if one packet is written. For double buffers, either buffer indicates a write enabled state. Even if data has been written to one buffer, the state is a Ready state if the other buffer is empty. <OUT direction> Ready indicates a read enabled state after data is received in the buffer that was set. Two cases can be assumed; data of the specified buffer size is received and a short packet is received. In non-continuous transfer mode, the buffer is set to a Ready state when receiving one packet regardless of the setting of the buffer size. For double buffers, a Ready state is assumed when either buffer receives data and is set to a read enabled state. Even if only one buffer receives data and can be read, the state i a Ready state. An Endpoint Buffer Not Ready interrupt (INTN) occurs when no response can be issued for the USB bus. In this case only, the buffer is set to Ready for the USB bus.

6	[Q]	Is a Buffer Ready interrupt to be used basically as the trigger of fetching data from a buffer and writing data to a buffer regardless of the transfer protocol and transfer direction?
	[A]	The interrupt type that is used is determined mainly by the method of using the FIFO buffer. The INTR interrupt alone does not reveal whether FIFO is empty when double buffers in the IN direction are set. In the OUT direction, since overrun cannot be identified, INTR and BEMP must be used. See below for the summary of interrupt summary occurrence factors for each buffer use method. Use method — I/O — INTR — INTN — BEMP Non-continuous single — IN — 1 → 0PKT — 0 → 0PKT — 1 → 0PKT — OUT — 0 → 1PKT — 1 → 1PKT — Over Non-continuous double — IN — 2 → 1PKT — 0 → 0PKT — 1 → 0PKT — OUT — 0 → 1PKT — 2 → 2PKT — Over Continuous single — IN — n … 1 → 0PKT — 0 → 0PKT — n … 1 → 0PKT — OUT — 0 → 1 … nPKT — n → nPKT — Over Continuous double — IN — 2n → n+1 → nPKT — 0 → 0PKT — n … 1 → 0PKT — OUT — 0 → 1 … nPKT — 2n → 2nPKT — Over In the case of continuous double IN, when one side of FIFO buffer is sent, an INTR interrupt occurs, the FIFO buffer becomes empty (0PKT remaining), and a BEMP interrupt occurs. * In the case of OUT, an INTR interrupt occurs when a short packet is received.
7	[Q]	In the case of data transmission (IN), I assume that a Ready interrupt is issued after transmission is completed and ACK is returned from the host. What is the timing of changing to Not Ready interrupt?
	[A]	An INTN interrupt occurs when the device issues a NAK response. (e.g. When FIFO is empty at IN, an INTN interrupt occurred due to reception of the IN token.)
8	[Q]	Multiple interrupt factors may be issued according to the setting of interrupt permission. Please indicate all the factors that may occur concurrently as the trigger for processing the same contents. To simplify processing, I wished to group interrupt factors for permission.
	[A]	Permit only the necessary interrupts according to the purpose. In the following case, only interrupts regarding data communication are extracted and competes. The INTR and BEMP interrupts may occur concurrently. (e.g. In the case of single buffer and IN, the interrupts occur concurrently at completion of packet transmission.)
9	[Q]	Can STALL for Endpoint be released simply by setting the response PID to BUF?
	[A]	By setting PID = BUF, communication between the FIFO and host is enabled at the occurrence of the IN/OUT token.
10	[Q]	Does EP0 Buffer Ready interrupt occur when the EP0-FIFO selection register to Control Write is set (ISEL = 0) ?
	[A]	The interrupt occurs when an INTR interrupt is permitted in ISEL = ”0” and ControlWrite transfer and the following event occurs. a)Received a short packet in the Data stage. b)Received data of a Maximum packet size in non-continuous setting. The interrupt also occurs when ISEL is changed from ‘1’ to ‘0’ and the interrupt condition is satisfied.
11	[Q]	Does a Buffer Empty interrupt occur when the EP0_FIFO selection register is set to Control Read (ISEL=1) ?
	[A]	The BEMP interrupt condition is satisfied when the entire FIFO data is sent. However, in control transfer, only the data stage is targeted. The interrupt does not occur as a result of NULL transmission in a status stage.
12	[Q]	The FIFO counter is reset when FIFO access is restarted after it is stopped temporarily and another processing is performed. To what function does this specification correspond?
	[A]	Setting change during write processing (IN) /read processing (OUT) is not permitted.The FIFO counter is shared and no feature is available for storing intermediate values.

9.4. Q & A Relating to DMA Transfer

1	[Q]	Is the DMA Enable function available?
	[A]	The DMA permission bit is available.
2	[Q]	Is the /Dreq signal asserted with 200ns cycle or less?
	[A]	Asserted. However, in DMA transfer, the tw(CYCLE) Specification is not required and Write/Read processing can be accessed without setting a wait to DMAC after /Dreq is asserted. See also the description of DMA transfer timing in the Specification.
3	[Q]	Can a Buffer Ready interrupt be interpreted as reception of a short packet (not necessary be the last packet?) when DMA of BULKOUT is used?
	[A]	The interrupt occurs as a result of reception of a short packet. When DMA is used, an interrupt for each packet is not used (does not occur due to DMA) and only when the last packet is a short packet, an interrupt occurs by transferring DMA. In this case, the data length is read and the DMA transfer capacity is set. * When continuous transfer is set in CPU access, the interrupt occurs at reception of a short packet or Buffer Full. When the last packet is not a short packet, a Null packet must be sent from the host.
4	[Q]	In this case (9.4.3), does DMA_DTLN indicate the received data length?
	[A]	Use DMA_DTLN.
5	[Q]	Minimum value timing chart of Td(Dack-Dreq) in one-word transfer mode (data sheet p48) and switching feature (data sheet P43)
	[A]	In one-word transmission mode, since /Dreq is asserted after /Dack="H" is detected, the minimum time td(Dackh-Dreq) up to the next /Dreq assertion is 20ns (one cycle of 48MHz).

9.5. Q & A Relating to FIFO Control

1	[Q]	How the FIFO buffer is cleared?
	[A]	The FIFO buffer can be cleared by using the BCLR bit (writing "1") of the CPU_FIFO control register or the DMA_FIFO register. Only the buffers that can be written from CPU can be cleared by BCLR and buffer toggle operation is not performed after the buffer is cleared. Therefore, when double buffers are set, the buffer that cannot be written from the CPU (buffer on the rear side) cannot be cleared. Clear the FIFO with double buffer specified using ACLR of EPi configuration register 1. However, when the buffer is cleared using ACLR during data transfer, the data in the buffer on the rear side is also cleared.
2	[Q]	Can't a packet be received from the host unless the FIFO is entirely empty?
	[A]	In continuous transfer mode, data can be received continuously up to the FIFO size that was set or until a short packet is received. In non-continuous transmit/receive mode also, data packets can be received continuously even if the buffer on one side is not entirely empty as long as a double buffer mode is set.
3	[Q]	Is FIFO on one side used even if a short packet is received?
	[A]	Even if a short packet is received, a FIFO buffer on one side is used. For instance, when a 1Kbyte double buffer is used for the FIFO buffer in continuous transfer setting, the area on one side (1Kbyte) is used when a short packet is received.
4	[Q]	At transmission on the USB ASSP side, is the FIFO data retained until ACK is issued from the host?
	[A]	Yes.
5	[Q]	About how long is it required to access registers and FIFO buffer from release of Reset/Suspend? (Warmup time considering oscillation stabilization time and PLL-Lock time)
	[A]	Consider an oscillation stabilization time and a PLL oscillation stabilization time of the oscillator. •Since the oscillation stabilization time of an oscillator varies according to the oscillator, check the data sheet of the oscillator that is used. •The PLL oscillation stabilization time is about 1ms.
6	[Q]	When receiving data by setting a mode to a double buffer continuous transfer (OUT) mode, can the data be stored in the FIFO buffer during reception read?
	[A]	In double buffer continuous transfer mode, the buffer in the process of reception (not in a Buffer Full state) cannot be read. The buffer toggle condition is Buffer Full or reception of a short packet. The program does not support a buffer toggle function.
7	[Q]	Combinations between IVAL and Creq are restricted for CPU_EP[3:0] of the CPU_FIFO selection register. At what timing are the IVAL and Creq states changed?
	[A]	•Timing of Creq state change When the Endpoint is set to the IN direction, IVAL is set to "1" when the FIFO buffer becomes full. Creq is set to "0" when the CPU_FIFO data register is set to a Write Enable state. When the Endpoint is set to the OUT direction, Creq is set to "1" when the FIFO buffer becomes full or a data/short packet data is received. Creq is set to "0" when CPU_FIFO data registers can be read. •Necessity for checking IVAL and Creq when switching Endpoint When switching an Endpoint with software, check completion of data write processing (IN direction)/data read processing (OUT direction) to the FIFO of the Endpoint currently being used. If EP is switched without checking it, a hardware operation error may occur. To check the states, check the values of IVAL and Creq.

8	[Q]	Is the state of IVAL and Creq sometimes changed when change of CPU_EP[3:0] is attempted after checking IVAL and Creq?
	[A]	Setting the IN direction for Endpoint IVAL='0', Creq='0' : Indicates that the FIFO buffer of the EP currently being used became empty. IVAL='1', Creq='1' : Indicates that the FIFO buffer of the EP currently being used is Full or in the process of data output. Setting the OUT direction for Endpoint IVAL='1', Creq='0' : Indicates that the FIFO buffer of the EP currently being used became full. IVAL='0', Creq='1' : Indicates that the FIFO buffer of the EP currently being used became empty or is receiving data.
9	[Q]	When the data that was written exceeded the buffer size, IVAL may be automatically set to 1. Is IVAL set to 1 automatically also if the data size is equal to the buffer size?
	[A]	It is automatically set to “1”.
10	[Q]	In continuous transfer mode, data is set even if the size exceeds the maximum packet size. Is IVAL set once only?
	[A]	In continuous transmission, IVAL need not be set for each packet.
11	[Q]	When the data size exceeds the maximum packet size, is IVAL set to 1 automatically?
	[A]	In continuous transmit/receive mode, IVAL is automatically set to 1 when the data that was written exceeded the buffer size that was set, not the Maximum packet size. When the data that is transmitted is shorter than the buffer size that was set, set IVAL to 1 and validate the buffer. For instance, in double buffers, when data has been written to one side, the buffer (one side) becomes valid for the USB bus. When data is written continuously, it is written to the buffer on the rear side. If the buffer on the rear side is not still full after the data to be transmitted has been written, set IVAL with software. The buffer on the rear side becomes valid and a response is sent to the IN packet.
12	[Q]	Is there any continuous transmit/receive data length register for EP1 to EP5? In Control transfer, there is such register. How can the number of bytes transmitted be recognized?
	[A]	In EP1~EP5, the number of bytes written to the buffer is the number of bytes transmitted.
13	[Q]	Interrupt status 1 (Buffer Ready Interrupt bit) cannot be cleared. * Although BCLR was executed, the state was not changed from “1” to “0”. When the same Endpoint is switched from IN to OUT (assuming Alternate), a Buffer ready interrupt occurs, a Creq state is set, and the program falls into a permanent loop although there is no data.
	[A]	When double buffers are set, the buffers cannot be completely cleared by BCLR. Use ACLR.
14	[Q]	In continuous transfer mode in the OUT direction, doesn’t interrupt occur in CPU even if the data size is less than the buffer size and MaximumPacketSize is received? If so, how can CPU recognize termination of continuous transfer?
	[A]	When the size of the packet that was received is equal to MaximumPacketSize and the size is less than the buffer size that was set, an interrupt does not occur. When using a continuous transfer mode, set the number of bytes to be transferred to the buffer size or use a short packet (or Null packet) for the last packet. This specification must be agreed with the software on the host side.
15	[Q]	Is it necessary to set Buffer Select (ISEL) when reading data from or writing data to EP0_FIFO?
	[A]	Necessary
16	[Q]	What is the definition of FIFO access cycle time tw(CYCLE)?
	[A]	In CPU access, FIFO access cycle means from the edge of /WR or /RD to the other edge.

17	[Q]	For change of IVAL and Creq, are both IVAL and Creq sometimes set to “1” or “0” at transition from IVAL='1' and Creq='0' to IVAL='0' and Creq='1'?
	[A]	See below for the state transition of IVAL and Creq. When checking values of both IVAL and Creq, check the value of IVAL when checking the setting of Creq to “0” twice. When the OUT direction is set, the state is set to IVAL='1', Creq='1' for one 48MHz clock. The state is not set to IVAL='0', Creq='0'. In the same way, when the IN direction is set, the state is set to IVAL='0', Creq='1' for one 48MHz clock. The state is not set to IVAL='1', Creq='0'. Neither the state that allows data read processing perform (IVAL='0',Creq='0' in the OUT direction setting) when there is no data nor the state that allows write processing (IVAL='1',Creq='0' in the IN direction setting) when there is data will not be set.
18	[Q]	What is the difference between the BCLR of the CPU_FIFO control register (address 42h) and EPi_ACLR of the EPi configuration register?
	[A]	When ‘1’ is written to BCLR, the FIFO buffer of the selected Endpoint is cleared. The FIFO buffers of the corresponding Endpoint are continuously cleared when ‘1’ is written to ACLR. Consequently, when FIFO buffers are cleared by ACLR, ACLR must be set to ‘0’ after being set to ‘1’. •When a signal buffer is set, both ACLR and BCLR perform the same function. The function clears all the FIFO areas that were allocated for each Endpoint. •When double buffers are set, ACLR and BCLR perform different functions. BCLR clears only the buffers to which data can be written from CPU. → Clears one side of the double buffers. ACLR clears all the FIFO areas that were allocated for each Endpoint. → Clears both sides of the double buffers. * Note that the buffer in the process of transfer processing is also cleared if buffers are cleared by ACLR during data transfer.

9.6. Q & A Relating to the Null Addition Function

1	[Q]	Is Null added automatically if data is written until FIFO becomes full? ForEx) If data of 192 bytes is transferred when FIFO is 128byte, is the data set as 2 data +Null+1 data +Null?
	[A]	The NULL addition conditions are as follows. Set a continuous mode or a Null addition mode. Write data that ends a multiple of the Maximum packet size. 1. If the buffer is not full, NULL is added. 2. Buffer full 2.1 Double buffers NULL is added when the writing of data to the buffer of the other side is not completed before transmission from the buffer that became full is completed. 2.2 Single buffer NULL is added.
2	[Q]	What is the image of the purpose of the Null addition mode? * Is it like a continuous receive mode on the host side?
	[A]	The Specification indicates that a data stage ends when receiving a short packet containing NULL at the data stage of control transfer. In other transfers also (particularly, BULK), the same specification may be used for some purpose. •The data size is not clear before transfer. •The volume of data is to be minimized by transferring raw data only without data information.
3	[Q]	When double buffers are used, is a Null addition mode applicable to the write side only? ForEx) When FIFO rear data is written and Null is added while FIFO front data is being transmitted is Null sent at termination transmission of FIFO front data?
	[A]	See the NULL addition conditions.
4	[Q]	A Null packet cannot be stopped when Null is added at Bulk IN transfer. * A Null packet is sent for every IN token, not once only. BulkIN (RDY interrupt) is permitted by a NR interrupt and the RDY interrupt is prohibited after the entire data on the memory is transferred to FIFO. In Null addition mode, the NR interrupt does not occur from the second time so that data cannot be transferred.
	[A]	A NULL packet is sent once only unless re-transmission operation is performed due to ACK corruption, etc. When the Maximum packet size is set to 0, NULL is continuously sent automatically. In retransmission or when the Maximum packet size is set to 0, the INTR interrupt does not occur since there is data to be transmitted.
5	[Q]	In what occasion is Null data transmission addition mode used?
	[A]	When a host such as the M66290 supports a continuous transfer mode, Null data can be transmitted at the end to indicate the end of transfer.

9.7. Q & A Relating to Communication Errors

1	[Q]	In received data abnormality (PID abnormality, CRC error, or EOP detection error), is data field discard processing performed automatically on the USB ASSP side? (Is any intervention necessary from the connected micro computer?)
	[A]	Yes. When the data packet from the host is abnormal, no data is received and no response is sent until normal data is received.
2	[Q]	How does the USB ASSP operate in each abnormality state? (does not return NAK, interrupt, etc.)
	[A]	The USB ASSP neither sends a response nor sends NAK to the host. An interrupt to the control microcomputer does not occur. However, when isochronous transfer is set, th function for notifying a CRC error is provided.
3	[Q]	Is CRC created by the USB ASSP automatically?
	[A]	Yes

9.8. Q & A Relating to Other Functions

1	[Q]	When the M66290 is the data receiving side, is there any setting (operation mode) for returning NAK to the host in a state other than FIFO FULL state?
	[A]	By setting the response PID of each Endpoint to NAK or STALL, NAK or STALL can be returned for the host in a state other than a FIFO FULL state.
2	[Q]	When USB is not used, the clock will be stopped to reduce current consumption. In this case, does the M66290 automatically change the mode to the one that can perform USB communication by starting clock oscillation as a result of USB connection or reception of an external command?
	[A]	The M66290 can stop oscillation of the clock or prohibit USB transceiver operation by the operation of the "USB operation enable register". (The current consumption can be reduced.) In this case, the mode cannot be changed automatically at USB connection. An interrupt occurs as a result of a Vbus interrupt or USB special signals such as Reset and Resume. Therefore, USB communication can be resumed by permitting USB module operation by operating the "USB operation enable register" in a program. Use the following procedure when starting USB communication by a Vbus interrupt. 1) Permit Vbus interrupt after stopping oscillation and prohibiting USB transceiver operation. 2) When USB is connected, a Vbus interrupt occurs. 3) Select a crystal, and start oscillation and PLL operation. 4) Wait about 1ms until the oscillation becomes stable. 5) Set the SCKE bit and USBPC bit of "USB operation enable register". 6) If the Vbus port in the interrupt program is se to "H", execute 7). If it is set to "L", return to 1) and wait for USB connection. 7)Initialize various USB registers. 8) Set the TrON bit and USBE bit of the "USB operation enable register". 9) Permit a status transition interrupt. Subsequently, normal USB communication starts.
3	[Q]	Terminal states in hardware Reset and software Reset
	[A]	Hardware reset: A6-A1, /CS, /RD, /WR, /Dack, and /RST pins are in an input state. D15-D0 are in an input state. However, /CS and /RD are inactive. The Tron terminal is Hi-Z output. The /INT and /Dreq terminals are "H" output. software reset: The state before resetting is retained. However, software reset is operated during interrupt, the interrupt status cleared and as a result the output of the /INT terminal is changed.
4	[Q]	Is there any problem if Reset is asserted for a long time by keeping the power supplied?
	[A]	No problem
5	[Q]	Are there any terminals that are cared internally such as by Pull up/down?
	[A]	Vbus terminal – Pulled down with 100kΩ. TEST1,TEST2, and TCK terminals – Pulled down with 50k Ω. /TRST, TMS, and TDI terminals – Pulled up with 50kΩ.
6	[Q]	Is a tolerance (5V) provided for the CPU interface specification?
	[A]	The CPU interface can handle 3.3V only. (Only 5V input is allowed for the Vbus terminal only.)

7	[Q]	Is the response PID (Endpoints 1 to 5) ever set to NAK on the USB ASSP side?
	[A]	Three settings are available for response PID; NAK response, STALL response, and the response by the buffer state (BUF). In NAK/STALL setting, the M66290 issues a NAK/STALL response to the IN/OUT token, ignoring the buffer state and does not transmit or receive data. In BUF response setting, the M66290 response as follows. 1) When the IN direction transmission and OUT direction reception are enabled, the M66290transmits and receives data and issues an ACK response. 2) When transmission in the IN direction and reception in the OUT direction are not possible, the M66290 issues a NAK response for IN/OUT. However, at NAK response in 2) also, the PID remains in the BUF setting.
8	[Q]	When does the IC side change the setting of response PID?
	[A]	All the response PIDs of Endpoint1-5 are switched by a program except for the case where the PID response is automatically set to STALL when a data packet exceeding the maximum packet size is received in the OUT direction setting.
9	[Q]	Does a Vbus detection interrupt occur when is Vbus already set to “H” when it is started from reset?
	[A]	A Vbus change detection interrupt detects leading/trailing of Vbus. Therefore, changes occurred before USB was permitted (USBE = '1') cannot be detected. When USB is enabled from a disable state such as after power supply or immediately after reset, an interrupt does not occur. Therefore, check the state using the Vbus status.
10	[Q]	Can the D+ pull-up resistance control function control cable hot line insertion with a register?
	[A]	In Full Speed USB, connection is notified to the upstream side (HUB) by pulling up D+ at cable connection. The D+ pull-up resistance control function enables control of pull-up/not pull-up D+ through register setting.
11	[Q]	Does an interrupt occur if a USB cable is connected when a Vbus interrupt is prohibited and the interrupt is permitted later? Does an interrupt occur when a USB cable is inserted and removed when Vbus is prohibited and then the interrupt is permitted?
	[A]	In either case, the VBUS interrupt status is set to 1. An interrupt signal is generated if a Vbus interrupt is permitted as long as a VBUS interrupt status is not cleared.
12	[Q]	Is EPi_PID set to NAK automatically in EP1~EP5 continuous receive mode when the buffer is not Ready?
	[A]	When the response PID is set to BUF, a NAK response is issued automatically.
13	[Q]	Is it necessary to permit oscillation of the “Oscillation Buffer Enable” bit of the USB operation enable register even if external clock input is used?
	[A]	Oscillation must be permitted to the “Oscillation Buffer Enable” bit of the USB operation enable register even if external clock input is used.
14	[Q]	To achieve a low power consumption mode, the CPU of the main unit is set to a sleep mode. I assume that an interrupt can be issued by reception of a packet from USB to handle this operation. Please describe the specification.
	[A]	In low power consumption mode, the device is set to a Suspend state. (When the device is set to a USB bus idle state, the state is changed to a suspend state.) When a USB packet transmit /receive request is issued from the host in a Device Suspend state, a packet transmit/receive request is notified to the device from the host after Suspend is released (transmitting a Resume or Reset signal). The M66290 detects a line state change and generates a Resume interrupt even if oscillation is stopped due to a low power consumption mode. * See Chapter 9 of the USB Specification 1.1.
15	[Q]	When an interrupt is asserted from Inactive to Active, is the active level maintained until the CPU clears the interrupt factor?
	[A]	Yes

16	[Q]	Is there any time limit for a response after release of sleep to a USB ASSP?
	[A]	1) When the M66290 is resumed from a Suspend state, the host (PC) transmits a resume signal for at least 20ms. Therefore, preparation for a response must be completed within 20ms. 2) When the M66290 is reset from a Suspend state, the host (PC) transmits a Reset signal for at least 10ms. Therefore, preparation for a response must be completed within 10ms. * See Chapter 7 of the USB Specification 1.1. The M66290 detects a change of differential signal to return from Suspend. The interrupt varies according to the state of the CPU for control, however, it occurs for a shor period.
17	[Q]	Reception of USB Bus Reset during a Suspend state
	[A]	Since an “SE0” continues for 10ms or more, the M66290 detects a reset signal again after transceiver operation. 1) Operation such as transceiver as a result of occurrence of a RESUME interrupt 2) Verification of a device state in 2.5μs or more after PLL operation 3)-1 Continuation of a Suspend state in Suspend state ...... Suspend processing 3)-2 Detection of a Reset signal in Default state ...... Reset processing 3)-3 Detection of a Resume signal in other cases ...... Resume processing
18	[Q]	How are communication and D+ waveform observed in Suspend/Resume?
	[A]	Click on the Start during USB operation on the Windows98 desktop, move the cursor to the End of Windows, and set the system to Standby by clicking on Standby. In this state, USB peripheral devices are set to a Suspend state. Then, resume Windows98 by entering a command from a keyboard, etc. Measure again the waveform from the USB communication log at the time (at return from Standby) and D+ M66290 terminal section. Note: The M66290 detects Resume at D+ leading edge and generates a Resume interrupt. Bus Reset is also included. The D- value is not related. The M66290 checks if the state is a K state (Resume) or an SE0 (Bus Reset) after clock oscillation. As a result, the M66290 changes the device state to the original state or Default and generates a device state interrupt. Consequently, a clock must be 2.5μs before the Resume and Bus Reset signals are terminated. When the clock could not oscillated, the M66290 cannot detect Resume or Bus Reset as the device state.
19	[Q]	Please explain STALL processing.
	[A]	Basically, STALL is used by a request error response from a host. STALL can also be used as STALL processing by the SET_FEATURE /CLEAR_FEATURE command and device class specific method (storage class). In the M66290, a STALL response can be issued by setting response PID = STALL.
20	[Q]	In Endpoint0, when is the state returned to BUF after STALL is specified for PID?
	[A]	The M66290 issues an ACK response to the SETUP token of Endpoint0(default pipe) when receiving a request, regardless of the existing response PID setting. In this case, when receiving a request normally, the M66290 automatically set EP0_PID = NAK. Therefor, if STALL response is issued for Endpoint0 due to a request error, the PID setting is left to hardware until a new SETUP token is received.
21	[Q]	In Endpoint1-5, is the procedure to specify STALL for PID, wait after resetting the sequence toggle, and release STALL of the specified EP using CLEAR_FEATURE from the host?
	[A]	In Endpoint1-5, after clearing the sequence bit, wait for CLEAR_FEATURE. In the M66290, STALL can be released by setting PID=BUF/NAK.
22	[Q]	Is level output used for the interrupt output from the M66290 as long as the factor that permits the interrupt exists?
	[A]	Level output is used until the factor that is permitted is avoided.
23	[Q]	Is a status flag set even for the factor that prohibits an interrupt?
	[A]	Status setting is performed also for the factor that prohibits an interrupt.

24	[Q]	I assume that a good method is to define a processing method of the interrupt factors that are issued concurrently and use an edge interrupt for the interrupt on the CPU side. Is there any problem in this method? (Is it better to permit an interrupt in level by permitted a null interrupt?)
	[A]	Either method is acceptable. In either case, the presence or absence of the factor that occurred during interrupt processing must be checked before control is returned from the interrupt.
25	[Q]	At USB Reset interrupt, a control transfer stage interrupt is permitted. Does a USB reset interrupt occur even in the system that detects VBUS (insertion) and setting TrON to H?
	[A]	By setting TrON = H, the host verifies connection of the device and sends a Reset signal. The USB device detects USB Reset and initializes various registers and internal status.
26	[Q]	Are there any differences in STALL release methods due to the differences of BULK anc INTERRUPT?
	[A]	There are no differences due to the difference of transfer types. However, note that the STALL use method varies according to the device class. In the M66290, STALL can be released by setting PID=BUF/NAK.
27	[Q]	When data that was received in the OUT direction exceeds the packet size, an error can be detected. In this case, what response does the packet issue to the host?
	[A]	The processing performed when the data exceeds the packet size in the OUT transactior is described in Chapter 5. It depends on how the Specification is interpreted. The M66290 discards (no response) the transaction with a size error, sets the Endpoint to a STALL state, and returns STALL to subsequent transactions. (However, no response is issued to the transaction for which a size error occurred.) This specification is neither completely out of track nor incorrect. The problem is the specification (current) on the application side. In particular, the problem is the degree of handling error processing. Some hosts seem to stop as a result of return of STALL response or some hosts do not handle STALL (in the specification, STALL is released). (Operation has not been verified using the actual machine.)
28	[Q]	Please explain STALL of Endpoint0. Since EPO_PID is set to NAK after EP0 is terminated with Stall and the next setup stage is completed, the host cannot reference information of EP0_PID when transmitting GetStatus to EP0 and a STALL response cannot be issued to GetStatus. What is the solution for this problem?
	[A]	A STALL response of Endpoint0 is different from that of normal data transfer. Since a default pipe (EP0) must always be formed, it will not be set to a disabled state. *See Chapter 5.3 USB Communication Flow. In USB communication, the following rules are applied to control transfer. •When data is received by a Setup transaction, an ACK response is issued. •No response is issued when data is corrupted in a Setup transaction. *See Chapter 8.5.2 Control Transfers •When a command sequence is terminated normally, a Null packet is sent. •When a command sequence is terminated abnormally, a STALL response is issued. •A NAK response is issued during execution of a command sequence. *See Chapter 8.5.2.1 Reporting Results Consequently, STALL operation of EP0 is different from that of normal Endpoint. For Endpoint0, an ACK response must be issued for a new request even during STALL (the previous response is STALL), in the same way as the case where STALL is not issued. In the M66290, this function is supported by hardware and an ACK response is issued to the SETUP token for any EP0_PID type and PID is changed to NAK.

29	[Q]	In which timing from the following does a Set Address interrupt occur when a remote wakeup function is used? 1)After the M66290A transmits a remote signal 2)After a Resume signal of the host is detected In 1), is Suspend conversion processing necessary? In 2), Is reset signal checking necessary? Reason: Since the DVSQ set value at the occurrence of a Set_Address is unavailable, DVSQ=111 will be set in the case of 1) and in the case of 2), DVSQ=011 will be set.
	[A]	The interrupt occurs at the timing of 1). As the concept of a device state, the meaning of the interrupt changes according to the state from which the state was changed. In this case, the meaning is as follows. Default state → Address state indicates a Set Address interrupt (control completion interrupt also occurs). Suspended (Suspend from Address) state → Address state indicates a resume synchronous interrupt (resume interrupt (by D+ change) also occurs) or a remote wakeup termination interrupt (resume interrupt also occurs). Resume can be recognized from a device state interrupt also. This requires a clock. It can be used in a system that does not stop a clock. A bit-14 resume interrupt in an interrupt status does not require a clock. When the above interrupt occurs, the DVSQ value is set to 010. Address state → Suspended (Suspend from Address) state indicates a suspend interrupt. The DVSQ value is set to 110. The DVSQ value is changed as follows. 011 → 110 → 010 ↑ ↑ ↑ • Suspend • Remote wakeup • End When Remote wakeup is generated from Suspend of the Configured state, the state is changed to an Address state. Consequently, it is necessary to rewrite from the Address state to Configured state with F/W in this case only. The DVSQ value is changed as follows. 011 → 111 → 010 → 011 ↑ ↑ ↑ ↑ • Suspend • Remote wakeup • End • F/W rewrite
30	[Q]	Is the process of the M66290A CMOS or NMOS?
	[A]	CMOS.

10. Revision History

Version	Data	Contents
0.9	'00.09.01	Release
1.0	'00.11.20	1.3 Related Documents URL 3.2 Example of Connection with Control CPU CPU's name Figure 3-2 Connection Example for Main CPU CPU's name

Mitsubishi Digital ASSP M66290A
USB Application Note
Ver 1.0 Rev.C ‘00-11-20