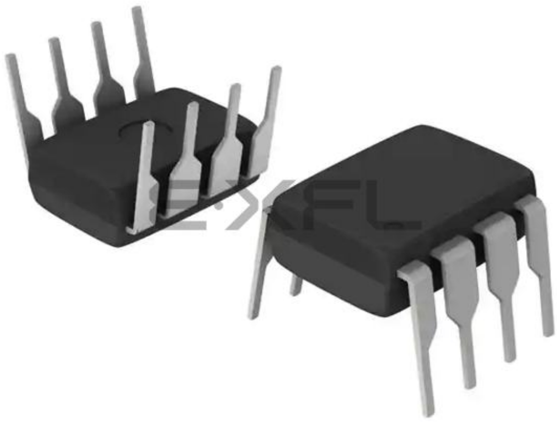
Zilog - Z8F042APB020SG Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	6
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 4x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	8-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f042apb020sg

List of Tables

CPU and Peripheral Overview

The eZ8 CPU, Zilog's latest 8-bit Central Processing Unit (CPU), meets the continuing demand for faster and more code-efficient microcontrollers. The eZ8 CPU executes a superset of the original Z8 instruction set. The features of eZ8 CPU include:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required program memory
- Software stack allows much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access of up to 4 KB
- New instructions improve execution efficiency for code developed using higher-level programming languages, including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the Register File
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

For more information about eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download on www.zilog.com.

10-Bit Analog-to-Digital Converter

The optional analog-to-digital converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC accepts inputs from eight different analog input pins in both single-ended and differential modes. The ADC also features a unity gain buffer when high input impedance is required.

Low-Power Operational Amplifier

The optional low-power operational amplifier (LPO) is a general-purpose amplifier primarily targeted for current sense applications. The LPO output may be routed internally to the ADC or externally to a pin.

Note: Asserting any power control bit disables the targeted block regardless of any enable bits contained in the target block's control registers.

General-Purpose Input/Output

The Z8 Encore! XP F082A Series products support a maximum of 25 port pins (Ports A–D) for general-purpose input/output (GPIO) operations. Each port contains control and data registers. The GPIO control registers determine data direction, open-drain, output drive current, programmable pull-ups, Stop Mode Recovery functionality and alternate pin functions. Each port pin is individually programmable. In addition, the Port C pins are capable of direct LED drive at programmable drive strengths.

GPIO Port Availability By Device

Table 14 lists the port pins available with each device and package type.

Table 14. Port Availability by Device and Package Type

Devices	Package	ADC	Port A	Port B	Port C	Port D	Total I/O
Z8F082ASB, Z8F082APB, Z8F082AQB Z8F042ASB, Z8F042APB, Z8F042AQB Z8F022ASB, Z8F022APB, Z8F022AQB Z8F012ASB, Z8F012APB, Z8F012AQB	8-pin	Yes	[5:0]	No	No	No	6
Z8F081ASB, Z8F081APB, Z8F081AQB Z8F041ASB, Z8F041APB, Z8F041AQB Z8F021ASB, Z8F021APB, Z8F021AQB Z8F011ASB, Z8F011APB, Z8F011AQB	8-pin	No	[5:0]	No	No	No	6
Z8F082APH, Z8F082AHH, Z8F082ASH Z8F042APH, Z8F042AHH, Z8F042ASH Z8F022APH, Z8F022AHH, Z8F022ASH Z8F012APH, Z8F012AHH, Z8F012ASH	20-pin	Yes	[7:0]	[3:0]	[3:0]	[0]	17
Z8F081APH, Z8F081AHH, Z8F081ASH Z8F041APH, Z8F041AHH, Z8F041ASH Z8F021APH, Z8F021AHH, Z8F021ASH Z8F011APH, Z8F011AHH, Z8F011ASH	20-pin	No	[7:0]	[3:0]	[3:0]	[0]	17
Z8F082APJ, Z8F082ASJ, Z8F082AHJ Z8F042APJ, Z8F042ASJ, Z8F042AHJ Z8F022APJ, Z8F022ASJ, Z8F022AHJ Z8F012APJ, Z8F012ASJ, Z8F012AHJ	28-pin	Yes	[7:0]	[5:0]	[7:0]	[0]	23
Z8F081APJ, Z8F081ASJ, Z8F081AHJ Z8F041APJ, Z8F041ASJ, Z8F041AHJ Z8F021APJ, Z8F021ASJ, Z8F021AHJ Z8F011APJ, Z8F011ASJ, Z8F011AHJ	28-pin	No	[7:0]	[7:0]	[7:0]	[0]	25

Shared Debug Pin

On the 8-pin version of this device only, the Debug pin shares function with the PA0 GPIO pin. This pin performs as a general purpose input pin on power-up, but the debug logic monitors this pin during the reset sequence to determine if the unlock sequence occurs. If the unlock sequence is present, the debug function is unlocked and the pin no longer functions as a GPIO pin. If it is not present, the debug feature is disabled until/unless another reset event occurs. For more details, see the On-Chip Debugger chapter on page 180**.**

Crystal Oscillator Override

For systems using a crystal oscillator, PA0 and PA1 are used to connect the crystal. When the crystal oscillator is enabled, the GPIO settings are overridden and PA0 and PA1 are disabled. See the Oscillator Control Register Definitions section on page 196 for details.

5V Tolerance

All six I/O pins on the 8-pin devices are 5V-tolerant, unless the programmable pull-ups are enabled. If the pull-ups are enabled and inputs higher than V_{DD} are applied to these parts, excessive current flows through those pull-up devices and can damage the chip.

Note: In the 20- and 28-pin versions of this device, any pin which shares functionality with an ADC, crystal or comparator port is not 5 V-tolerant, including PA[1:0], PB[5:0] and PC[2:0]. All other signal pins are 5 V-tolerant and can safely handle inputs higher than V_{DD} except when the programmable pull-ups are enabled.

External Clock Setup

For systems using an external TTL drive, PB3 is the clock source for 20- and 28-pin devices. In this case, configure PB3 for alternate function CLKIN. Write the Oscillator Control (OSCCTL) Register such that the external oscillator is selected as the system clock. See the Oscillator Control Register Definitions section on page 196 for details. For 8-pin devices, use PA1 instead of PB3.

> **!** **Caution:** To avoid retriggerings of the Watchdog Timer interrupt after exiting the associated interrupt service routine, Zilog recommends that the service routine continues to read from the RSTSTAT Register until the WDT bit is cleared as shown in the following example.

```
CLEARWDT:
 LDX r0, RSTSTAT ; read reset status register to clear wdt bit
 BTJNZ 5, r0, CLEARWDT   ; loop until bit is cleared
```

Interrupt Control Register Definitions

For all interrupts other than the Watchdog Timer interrupt, the Primary Oscillator Fail Trap and the Watchdog Oscillator Fail Trap, the interrupt control registers enable individual interrupts, set interrupt priorities and indicate interrupt requests.

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) Register, shown in Table 35, stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 Register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 0 Register to determine if any interrupt requests are pending.

Table 35. Interrupt Request 0 Register (IRQ0)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	T1I	T0I	U0RXI	U0TXI	Reserved	Reserved	ADCI
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FC0H							

Bit	**Description**
[7]	**Reserved** This bit is reserved and must be programmed to 0.
[6] T1I	**Timer 1 Interrupt Request** 0 = No interrupt request is pending for Timer 1. 1 = An interrupt request from Timer 1 is awaiting service.
[5] T0I	**Timer 0 Interrupt Request** 0 = No interrupt request is pending for Timer 0. 1 = An interrupt request from Timer 0 is awaiting service.

Table 72. UART Baud Rates (Continued)

Acceptable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	1	223.72	–10.51
115.2	2	111.9	–2.90
57.6	4	55.9	–2.90
38.4	6	37.3	–2.90
19.2	12	18.6	–2.90
9.60	23	9.73	1.32
4.80	47	4.76	–0.83
2.40	93	2.41	0.23
1.20	186	1.20	0.23
0.60	373	0.60	–0.04
0.30	746	0.30	–0.04

Acceptable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A
250.0	N/A	N/A	N/A
115.2	1	115.2	0.00
57.6	2	57.6	0.00
38.4	3	38.4	0.00
19.2	6	19.2	0.00
9.60	12	9.60	0.00
4.80	24	4.80	0.00
2.40	48	2.40	0.00
1.20	96	1.20	0.00
0.60	192	0.60	0.00
0.30	384	0.30	0.00

Receiving IrDA Data

Data received from the infrared transceiver using the IR_RXD signal through the RXD pin is decoded by the infrared endec and passed to the UART. The UART's baud rate clock is used by the infrared endec to generate the demodulated signal (RXD) that drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 18 displays data reception. When the infrared endec is enabled, the UART's RXD signal is internal to the Z8 Encore! XP F082A Series products while the IR_RXD signal is received through the RXD pin.

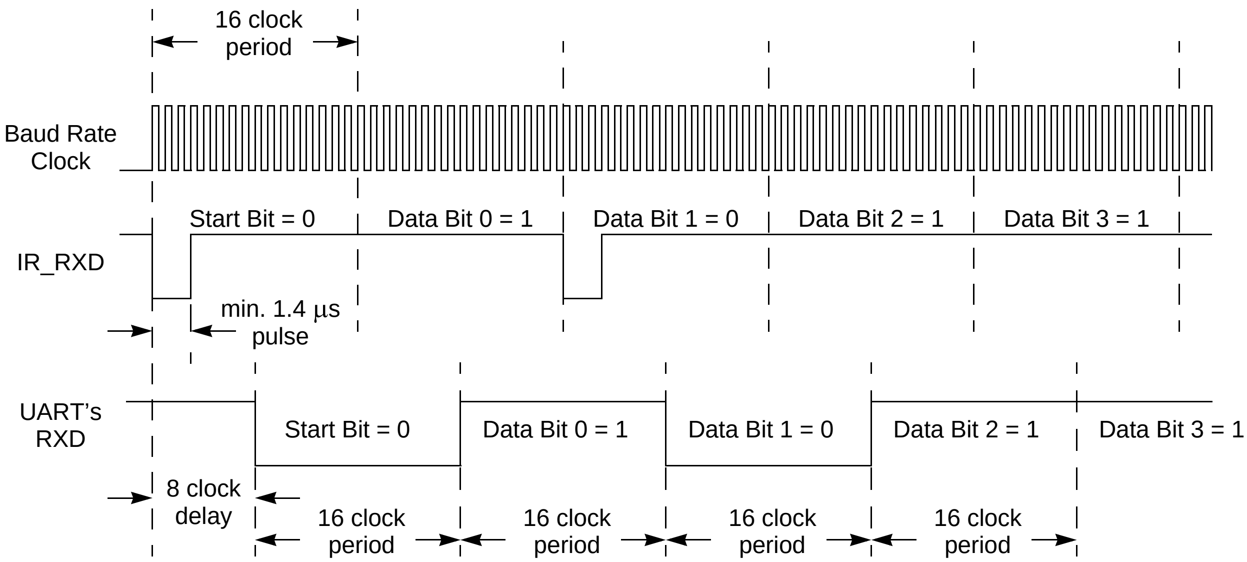


Figure 18. IrDA Data Reception

Infrared Data Reception

! **Caution:** The system clock frequency must be at least 1.0 MHz to ensure proper reception of the 1.4 µs minimum width pulses allowed by the IrDA standard.

Endec Receiver Synchronization

The IrDA receiver uses a local baud rate clock counter (0 to 15 clock periods) to generate an input stream for the UART and to create a sampling window for detection of incoming pulses. The generated UART input (UART RXD) is delayed by 8 baud rate clock periods with respect to the incoming IrDA data stream. When a falling edge in the input data stream is detected, the Endec counter is reset. When the count reaches a value of 8, the UART RXD value is updated to reflect the value of the decoded data. When the count reaches 12 baud clock periods, the sampling window for the next incoming pulse opens.

Table 75. ADC Data High Byte Register (ADCD_H)

Bit	7	6	5	4	3	2	1	0
Field	ADCDH							
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	F72H							
X = Undefined.								

Bit	Description
[7:0] ADCDH	**ADC Data High Byte** This byte contains the upper eight bits of the ADC output. These bits are not valid during a single-shot conversion. During a continuous conversion, the most recent conversion output is held in this register. These bits are undefined after a Reset.

ADC Data Low Byte Register

The ADC Data Low Byte (ADCD_L) Register contains the lower bits of the ADC output plus an overflow status bit. The output is a 13-bit two's complement value. During a single-shot conversion, this value is invalid. Access to the ADC Data Low Byte Register is read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 76. ADC Data Low Byte Register (ADCD_L)

Bit	7	6	5	4	3	2	1	0
Field	ADCDL					Reserved		OVF
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Address	F73H							
X = Undefined.								

Bit	Description
[7:3] ADCDL	**ADC Data Low Bits** These bits are the least significant five bits of the 13-bits of the ADC output. These bits are undefined after a Reset.

Flash Page Select Register

The Flash Page Select (FPS) Register shares address space with the Flash Sector Protect Register. Unless the Flash controller is unlocked and written with 5EH, writes to this address target the Flash Page Select Register.

The register is used to select one of the available Flash memory pages to be programmed or erased. Each Flash Page contains 512 bytes of Flash memory. During a Page Erase operation, all Flash memory having addresses with the most significant 7 bits given by FPS[6:0] are chosen for program/erase operation.

Table 82. Flash Page Select Register (FPS)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	INFO_EN	PAGE						
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	FF9H							

Bit	**Description**
[7] INFO_EN	**Information Area Enable** 0 = Information Area us not selected. 1 = Information Area is selected. The Information Area is mapped into the Program Memory address space at addresses `FE00H` through `FFFFH`.
[6:0] PAGE	**Page Select** This 7-bit field identifies the Flash memory page for Page Erase and page unlocking. Program Memory Address[15:9] = PAGE[6:0]. For the Z8F08xx devices, the upper 3 bits must be zero. For the Z8F04xx devices, the upper 4 bits must be zero. For Z8F02xx devices, the upper 5 bits must always be 0. For the Z8F01xx devices, the upper 6 bits must always be 0.

Table 105. Randomized Lot ID Locations (Continued)

Info Page Address	Memory Address	Usage
6A	FE6A	Randomized Lot ID Byte 13.
6B	FE6B	Randomized Lot ID Byte 12.
6D	FE6D	Randomized Lot ID Byte 11.
6E	FE6E	Randomized Lot ID Byte 10.
70	FE70	Randomized Lot ID Byte 9.
71	FE71	Randomized Lot ID Byte 8.
73	FE73	Randomized Lot ID Byte 7.
74	FE74	Randomized Lot ID Byte 6.
76	FE76	Randomized Lot ID Byte 5.
77	FE77	Randomized Lot ID Byte 4.
79	FE79	Randomized Lot ID Byte 3.
7A	FE7A	Randomized Lot ID Byte 2.
7C	FE7C	Randomized Lot ID Byte 1.
7D	FE7D	Randomized Lot ID Byte 0 (least significant).

Nonvolatile Data Storage

The Z8 Encore! XP F082A Series devices contain a nonvolatile data storage (NVDS) element of up to 128 bytes. This memory can perform over 100,000 write cycles.

Operation

The NVDS is implemented by special purpose Zilog software stored in areas of program memory, which are not user-accessible. These special-purpose routines use the Flash memory to store the data. The routines incorporate a dynamic addressing scheme to maximize the write/erase endurance of the Flash.

Note: Different members of the Z8 Encore! XP F082A Series feature multiple NVDS array sizes; see the Part Selection Guide section on page 2 for details. Devices containing 8KB of Flash memory do not include the NVDS feature.

NVDS Code Interface

Two routines are required to access the NVDS: a write routine and a read routine. Both of these routines are accessed with a CALL instruction to a predefined address outside of the user-accessible program memory. Both the NVDS address and data are single-byte values. Because these routines disturb the working register set, user code must ensure that any required working register values are preserved by pushing them onto the stack or by changing the working register pointer just prior to NVDS execution.

During both read and write accesses to the NVDS, interrupt service is NOT disabled. Any interrupts that occur during the NVDS execution must take care not to disturb the working register and existing stack contents or else the array may become corrupted. Disabling interrupts before executing NVDS operations is recommended.

Use of the NVDS requires 15 bytes of available stack space. Also, the contents of the working register set are overwritten.

For correct NVDS operation, the Flash Frequency registers must be programmed based on the system clock frequency (see **the** Flash Operation Timing Using the Flash Frequency Registers **section on page 149**).

Byte Read

To read a byte from the NVDS array, user code must first push the address onto the stack. User code issues a `CALL` instruction to the address of the byte-read routine (`0x1000`). At the return from the sub-routine, the read byte resides in working register R0 and the read status byte resides in working register R1. The contents of the status byte are undefined for read operations to illegal addresses. Also, the user code must pop the address byte off the stack.

The read routine uses 9 bytes of stack space in addition to the one byte of address pushed by the user. Sufficient memory must be available for this stack usage.

Because of the Flash memory architecture, NVDS reads exhibit a nonuniform execution time. A read operation takes between 44 µs and 489 µs (assuming a 20 MHz system clock). Slower system clock speeds result in proportionally higher execution times.

NVDS byte reads from invalid addresses (those exceeding the NVDS array size) return 0xff. Illegal read operations have a 2 µs execution time.

The status byte returned by the NVDS read routine is zero for successful read, as determined by a CRC check. If the status byte is nonzero, there was a corrupted value in the NVDS array at the location being read. In this case, the value returned in R0 is the byte most recently written to the array that does not have a CRC error.

Power Failure Protection

The NVDS routines employ error checking mechanisms to ensure a power failure endangers only the most recently written byte. Bytes previously written to the array are not perturbed.

A system reset (such as a pin reset or Watchdog Timer reset) that occurs during a write operation also perturbs the byte currently being written. All other bytes in the array are unperturbed.

Optimizing NVDS Memory Usage for Execution Speed

NVDS read time can vary drastically. This discrepancy is a trade-off for minimizing the frequency of writes that require post-write page erases, as indicated in Table 107. The NVDS read time of address N is a function of the number of writes to addresses other than N since the most recent write to address N, plus the number of writes since the most recent page erase. Neglecting effects caused by page erases and results caused by the initial condition in which the NVDS is blank, a rule of thumb is that every write since the most recent page erase causes read times of unwritten addresses to increase by 1 µs up to a maximum of (511-NVDS_SIZE) µs.

eZ8 CPU Instruction Set

This chapter describes the following features of the eZ8 CPU instruction set:

Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without concern for actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

Table 128. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LDX dst, src	dst ← src	r	ER	84	–	–	–	–	–	–	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ← src + X	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
OR dst, src	dst ← dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

General Purpose I/O Port Output Timing

Figure 35 and Table 144 provide timing information for GPIO port pins.

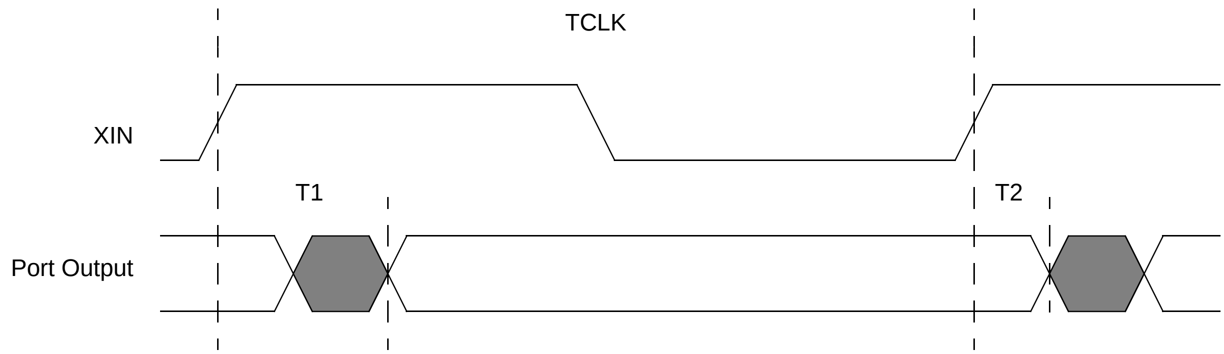


Figure 35. GPIO Port Output Timing

Table 144. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
GPIO port pins			
T_1	X_{IN} Rise to Port Output Valid Delay	–	15
T_2	X_{IN} Rise to Port Output Hold Time	2	–

Part Number Suffix Designations

Zilog part numbers consist of a number of components, as indicated in the following example.

Example. Part number Z8F042ASH020SG is an 8-bit Flash MCU with 4KB of Program Memory, equipped with advanced analog peripherals in a 20-pin SOIC package, operating within a 0ºC to +70ºC temperature range and built using lead-free solder.

Z8 F 04 2A S H 020 S G

Environmental Flow
G = Green Plastic Packaging Compound

Temperature Range
S = Standard, 0°C to 70°C
E = Extended, –40°C to +105°C

Speed
020 = 20MHz

Pin Count
B = 8
H = 20
J = 28

Package
H = SSOP
P = PDIP
Q = QFN
S = SOIC

Device Type
2A = Contains Advanced Analog Peripherals
1A = Does Not Contain Advanced Analog Peripherals

Memory Size
08 = 8KB Flash, 1KB RAM, 0 B NVDS
04 = 4KB Flash, 1KB RAM, 128 B NVDS
02 = 2KB Flash, 512 B RAM, 64 B NVDS
01 = 1KB Flash, 256 B RAM, 16 B NVDS

Memory Type
F = Flash

Device Family
Z8 = Zilog's 8-Bit Microcontroller

J

L

M

N

O

Z