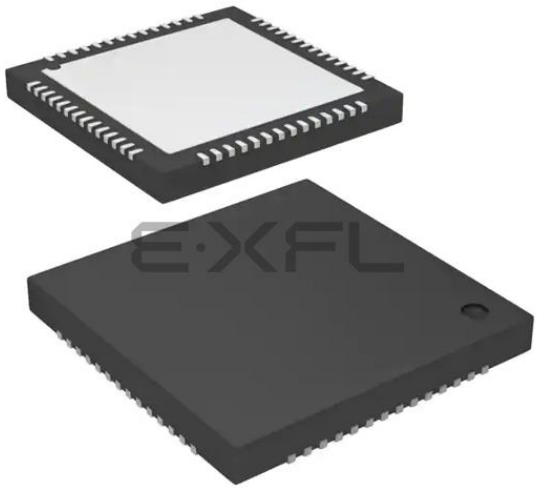


Welcome to E-XFL.COM

## Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

## What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

| Details | |
|---|---|
| Product Status | Active |
| Applications | USB Microcontroller |
| Core Processor | 8051 |
| Program Memory Type | ROMless |
| Controller Series | CY7C680xx |
| RAM Size | 16K x 8 |
| Interface | I²C, USB, USART |
| Number of I/O | 24 |
| Voltage - Supply | 3V ~ 3.6V |
| Operating Temperature | -40°C ~ 85°C |
| Mounting Type | Surface Mount |
| Package / Case | 56-VFQFN Exposed Pad |
| Supplier Device Package | 56-QFN-EP (8x8) |
| Purchase URL | https://www.e-xfl.com/product-detail/infineon-technologies/cy7c68013a-56ltxit |

## More Information

Cypress provides a wealth of data at www.cypress.com to help you to select the right device for your design, and to help you to quickly and effectively integrate the device into your design. For a comprehensive list of resources, see the application note AN65209 - Getting Started with FX2LP.

- Overview: USB Portfolio, USB Roadmap
- USB 2.0 Product Selectors: FX2LP, AT2LP, NX2LP-Flex, SX2
- Application notes: Cypress offers a large number of USB application notes covering a broad range of topics, from basic to advanced level. Recommended application notes for getting started with FX2LP are:
  - AN65209 - Getting Started with FX2LP
  - AN15456 - Guide to Successful EZ-USB® FX2LP™ and EZ-USB FX1™ Hardware Design and Debug
  - AN50963 - EZ-USB® FX1™/FX2LP™ Boot Options
  - AN66806 - EZ-USB® FX2LP™ GPIF Design Guide
  - AN61345 - Implementing an FX2LP™- FPGA Interface
  - AN57322 - Interfacing SRAM with FX2LP over GPIF
  - AN4053 - Streaming Data through Isochronous/Bulk Endpoints on EZ-USB® FX2 and EZUSB FX2LP
  - AN63787 - EZ-USB® FX2LP™ GPIF and Slave FIFO Configuration Examples using 8-bit Asynchronous Interface

For complete list of Application notes, click here.

- Code Examples:
  - USB Hi-Speed
- Technical Reference Manual (TRM):
  - EZ-USB FX2LP Technical Reference Manual
- Reference Designs:
  - CY4661 - External USB Hard Disk Drives (HDD) with Fingerprint Authentication Security
  - FX2LP DMB-T/H TV Dongle reference design
- Models: IBIS

### EZ-USB FX2LP Development Kit

The CY3684 EZ-USB FX2LP Development Kit is a complete development resource for FX2LP. It provides a platform to develop and test custom projects using FX2LP. The development kit contains collateral materials for the firmware, hardware, and software aspects of a design using FX2LP.

### GPIF™ Designer

FX2LP™ General Programmable Interface (GPIF) provides an independent hardware unit, which creates the data and control signals required by an external interface. FX2LP GPIF Designer allows users to create and modify GPIF waveform descriptors for EZ-USB FX2/ FX2LP family of chips using a graphical user interface. Extensive discussion of general GPIF discussion and programming using GPIF Designer is included in *FX2LP Technical Reference Manual* and *GPIF Designer User Guide*, distributed with GPIF Designer. *AN66806 - Getting Started with EZ-USB® FX2LP™ GPIF* can be a good starting point.

## Program/Data RAM

*SizeThe FX2LP has 16 KB of internal program/data RAM, where PSEN#/RD# signals are internally ORed to enable the 8051 to access it as both program and data memory. No USB control registers appears in this space.*

Two memory maps are shown in the following diagrams:

Figure 3 on page 10 shows the Internal Code Memory, EA = 0.

Figure 4 on page 11 shows the External Code Memory, EA = 1.

*Internal Code Memory, EA = 0*

This mode implements the internal 16 KB block of RAM (starting at 0) as combined code and data memory. When external RAM or ROM is added, the external read and write strobes are suppressed for memory spaces that exist inside the chip. This enables the user to connect a 64 KB memory without requiring address decodes to keep clear of internal memory spaces.

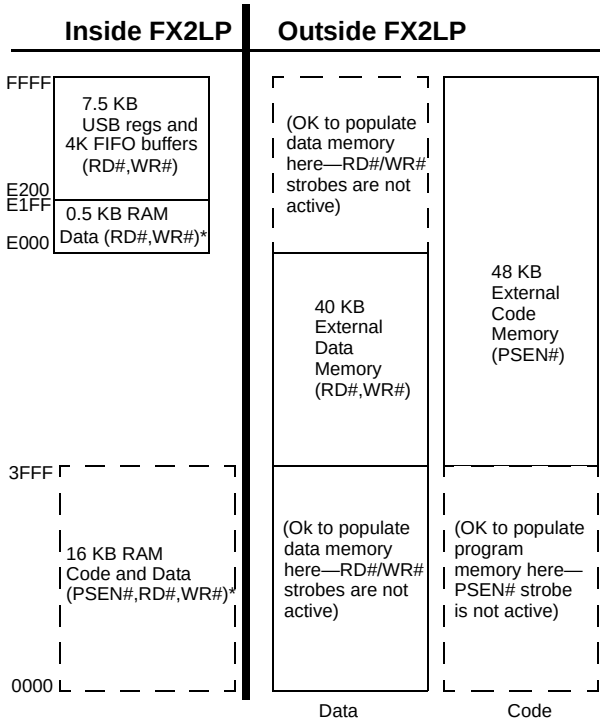
Only the internal 16 KB and scratch pad 0.5 KB RAM spaces have the following access:

- USB download
- USB upload
- Setup data pointer
- $I^2C$ interface boot load

*External Code Memory, EA = 1*

The bottom 16 KB of program memory is external and therefore the bottom 16 KB of internal RAM is accessible only as a data memory.

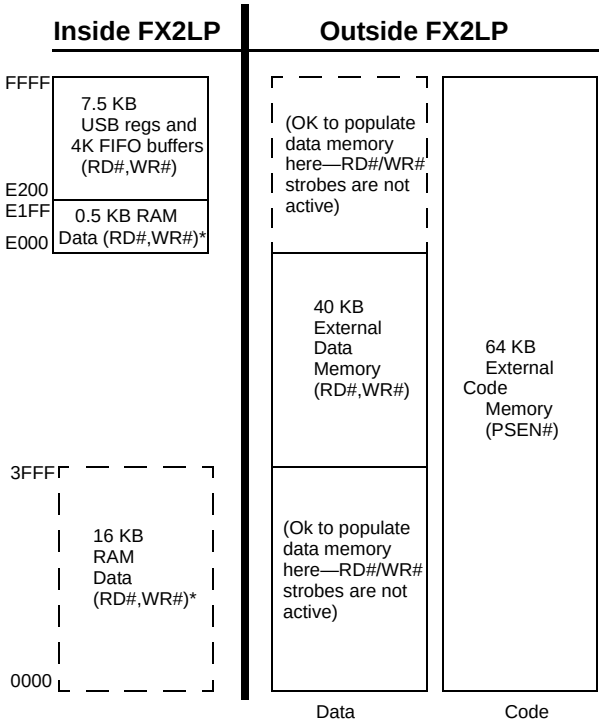
**Figure 3. Internal Code Memory, EA = 0**




*SUDPTR, USB upload/download, $I^2C$ interface boot access

**Note**
4. If the external clock is powered at the same time as the CY7C680xxA and has a stabilization wait period, it must be added to the 200 µs.

**Figure 4. External Code Memory, EA = 1**




*SUDPTR, USB upload/download, $I^2C$ interface boot access

## Register Addresses

| | |
|---|---|
| FFFF<br>F000 | 4 KB EP2-EP8 buffers (8 x 512) |
| EFFF<br>E800 | 2 KB RESERVED |
| E7FF<br>E7C0 | 64 BEP1IN |
| E7BF<br>E780 | 64 Bytes EP1OUT |
| E77F<br>E740 | 64 Bytes EP0 IN/OUT |
| E73F<br>E700 | 64 Bytes RESERVED |
| E6FF<br>E500 | 8051 Addressable Registers (512) |
| E4FF<br>E480 | Reserved (128) |
| E47F<br>E400 | 128 Bytes GPIF Waveforms |
| E3FF<br>E200 | Reserved (512) |
| E1FF<br>E000 | 512 Bytes 8051 xdata RAM |

*Default Full-Speed Alternate Settings*

**Table 6. Default Full Speed Alternate Settings**[5, 6]

| Alternate Setting | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ep0 | 64 | 64 | 64 | 64 |
| ep1out | 0 | 64 bulk | 64 int | 64 int |
| ep1in | 0 | 64 bulk | 64 int | 64 int |
| ep2 | 0 | 64 bulk out (2×) | 64 int out (2×) | 64 iso out (2×) |
| ep4 | 0 | 64 bulk out (2×) | 64 bulk out (2×) | 64 bulk out (2×) |
| ep6 | 0 | 64 bulk in (2×) | 64 int in (2×) | 64 iso in (2×) |
| ep8 | 0 | 64 bulk in (2×) | 64 bulk in (2×) | 64 bulk in (2×) |

*Default High Speed Alternate Settings*

**Table 7. Default Hi-Speed Alternate Settings**[5, 6]

| Alternate Setting | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ep0 | 64 | 64 | 64 | 64 |
| ep1out | 0 | 512 bulk[7] | 64 int | 64 int |
| ep1in | 0 | 512 bulk[7] | 64 int | 64 int |
| ep2 | 0 | 512 bulk out (2×) | 512 int out (2×) | 512 iso out (2×) |
| ep4 | 0 | 512 bulk out (2×) | 512 bulk out (2×) | 512 bulk out (2×) |
| ep6 | 0 | 512 bulk in (2×) | 512 int in (2×) | 512 iso in (2×) |
| ep8 | 0 | 512 bulk in (2×) | 512 bulk in (2×) | 512 bulk in (2×) |

## External FIFO Interface

*Architecture*

The FX2LP slave FIFO architecture has eight 512-byte blocks in the endpoint RAM that directly serve as FIFO memories and are controlled by FIFO control signals (such as IFCLK, SLCS#, SLRD, SLWR, SLOE, PKTEND, and flags).

In operation, some of the eight RAM blocks fill or empty from the SIE, while the others are connected to the I/O transfer logic. The transfer logic takes two forms: the GPIF for internally generated control signals and the slave FIFO interface for externally controlled transfers.

*Master/Slave Control Signals*

The FX2LP endpoint FIFOs are implemented as eight physically distinct 256×16 RAM blocks. The 8051/SIE can switch any of the RAM blocks between two domains, the USB (SIE) domain and the 8051-I/O Unit domain. This switching is done virtually instantaneously, giving essentially zero transfer time between "USB FIFOs" and "Slave FIFOs." Because they are physically the same memory, no bytes are actually transferred between buffers.

At any time, some RAM blocks are filling/emptying with the USB data under SIE control, while other RAM blocks are available to the 8051, the I/O control unit, or both. The RAM blocks operates as single-port in the USB domain, and dual-port in the 8051-I/O domain. The blocks can be configured as single-, double-, triple-, or quad-buffered as previously shown.

The I/O control unit implements either an internal master (M for Master) or external master (S for Slave) interface.

In Master (M) mode, the GPIF internally controls FIFOADR[1..0] to select a FIFO. The RDY pins (two in the 56-pin package, six in the 100-pin and 128-pin packages) can be used as flag inputs from an external FIFO or other logic if desired. The GPIF can be run from either an internally derived clock or externally supplied clock (IFCLK), at a rate that transfers data up to 96 MBytes/s (48

**Notes**
5. "0" means "not implemented."
6. "2×" means "double buffered."
7. Even though these buffers are 64 bytes, they are reported as 512 for USB 2.0 compliance. The user must never transfer packets larger than 64 bytes to EP1.

## Autopointer Access

FX2LP provides two identical autopointers. They are similar to the internal 8051 data pointers but with an additional feature: they can optionally increment after every memory access. This capability is available to and from both internal and external RAM. Autopointers are available in external FX2LP registers under the control of a mode bit (AUTOPTRSET-UP.0). Using the external FX2LP autopointer access (at 0xE67B – 0xE67C) enables the autopointer to access all internal and external RAM to the part.

Also, autopointers can point to any FX2LP register or endpoint buffer space. When the autopointer access to external memory is enabled, locations 0xE67B and 0xE67C in XDATA and code space cannot be used.

## I²C Controller

FX2LP has one I²C port that is driven by two internal controllers, the one that automatically operates at boot time to load VID/PID/DID and configuration information, and another that the 8051 uses when running to control external I²C devices. The I²C port operates in master mode only.

### *I²C Port Pins*

The I²C pins SCL and SDA must have external 2.2-kΩ pull-up resistors even if no EEPROM is connected to the FX2LP. External EEPROM device address pins must be configured properly. See Table 8 for configuring the device address pins.

**Table 8. Strap Boot EEPROM Address Lines to These Values**

| Bytes | Example EEPROM | A2 | A1 | A0 |
|---|---|---|---|---|
| 16 | 24LC00[10] | N/A | N/A | N/A |
| 128 | 24LC01 | 0 | 0 | 0 |
| 256 | 24LC02 | 0 | 0 | 0 |
| 4K | 24LC32 | 0 | 0 | 1 |
| 8K | 24LC64 | 0 | 0 | 1 |
| 16K | 24LC128 | 0 | 0 | 1 |

### *I²C Interface Boot Load Access*

At power-on reset, the I²C interface boot loader loads the VID/PID/DID configuration bytes and up to 16 KB of program/data. The available RAM spaces are 16 KB from 0x0000–0x3FFF and 512 bytes from 0xE000–0xE1FF. The 8051 is in reset. I²C interface boot loads only occur after power-on reset.

### *I²C Interface General-Purpose Access*

The 8051 can control peripherals connected to the I²C bus using the I2CTL and I2DAT registers. FX2LP provides I²C master control only; it is never an I²C slave.

## Compatible with Previous Generation EZ-USB FX2

The EZ-USB FX2LP is form-, fit-, and with minor exceptions, functionally-compatible with its predecessor, the EZ-USB FX2.

This makes for an easy transition for designers wanting to upgrade their systems from the FX2 to the FX2LP. The pinout and package selection are identical and a vast majority of firmware previously developed for the FX2 functions in the FX2LP.

For designers migrating from the FX2 to the FX2LP, a change in the bill of material and review of the memory allocation (due to increased internal memory) is required. For more information about migrating from EZ-USB FX2 to EZ-USB FX2LP, see the application note titled *Migrating from EZ-USB FX2 to EZ-USB FX2LP* available in the Cypress web site.

**Table 9. Part Number Conversion Table**

| EZ-USB FX2 Part Number | EZ-USB FX2LP Part Number | Package Description |
|---|---|---|
| CY7C68013-56PVC | CY7C68013A-56PVXC or CY7C68014A-56PVXC | 56-pin SSOP |
| CY7C68013-56PVCT | CY7C68013A-56PVXCT or CY7C68014A-56PVXCT | 56-pin SSOP – Tape and Reel |
| CY7C68013-56LFC | CY7C68013A-56LFXC or CY7C68014A-56LFXC | 56-pin QFN |
| CY7C68013-100AC | CY7C68013A-100AXC or CY7C68014A-100AXC | 100-pin TQFP |
| CY7C68013-128AC | CY7C68013A-128AXC or CY7C68014A-128AXC | 128-pin TQFP |

## CY7C68013A/14A and CY7C68015A/16A Differences

CY7C68013A is identical to CY7C68014A in form, fit, and functionality. CY7C68015A is identical to CY7C68016A in form, fit, and functionality. CY7C68014A and CY7C68016A have a lower suspend current than CY7C68013A and CY7C68015A respectively and are ideal for power-sensitive battery applications.

CY7C68015A and CY7C68016A are available in 56-pin QFN package only. Two additional GPIO signals are available on the CY7C68015A and CY7C68016A to provide more flexibility when neither IFCLK or CLKOUT are needed in the 56-pin package.

USB developers wanting to convert their FX2 56-pin application to a bus-powered system directly benefit from these additional signals. The two GPIOs give developers the signals they need for the power-control circuitry of their bus-powered application without pushing them to a high-pincount version of FX2LP.

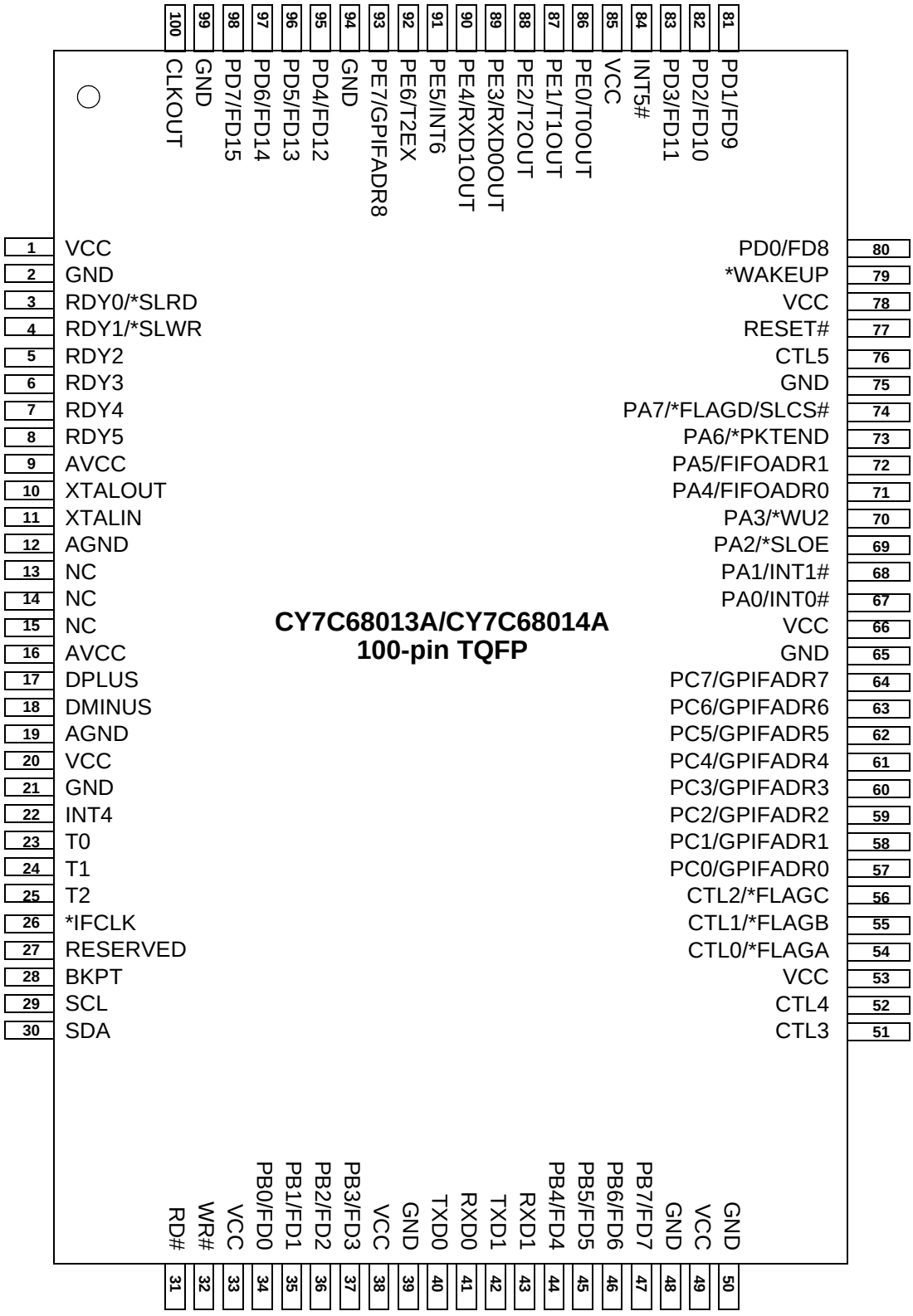
The CY7C68015A is only available in the 56-pin QFN package

**Table 10. CY7C68013A/14A and CY7C68015A/16A Pin Differences**

| CY7C68013A/CY7C68014A | CY7C68015A/CY7C68016A |
|---|---|
| IFCLK | PE0 |
| CLKOUT | PE1 |

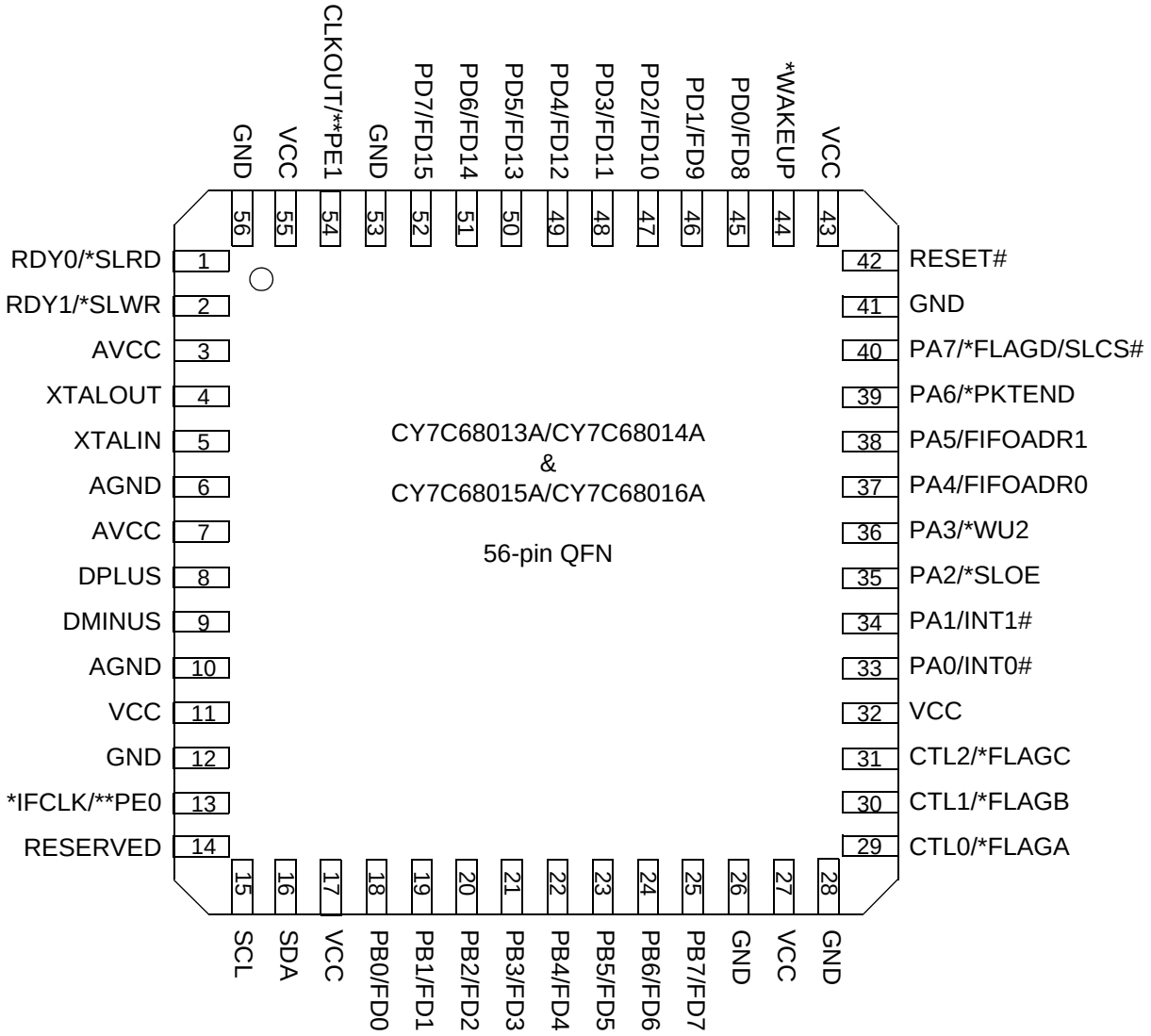
**Note**
10. This EEPROM does not have address pins.

**Figure 8. CY7C68013A/CY7C68014A 100-Pin TQFP Pin Assignment**



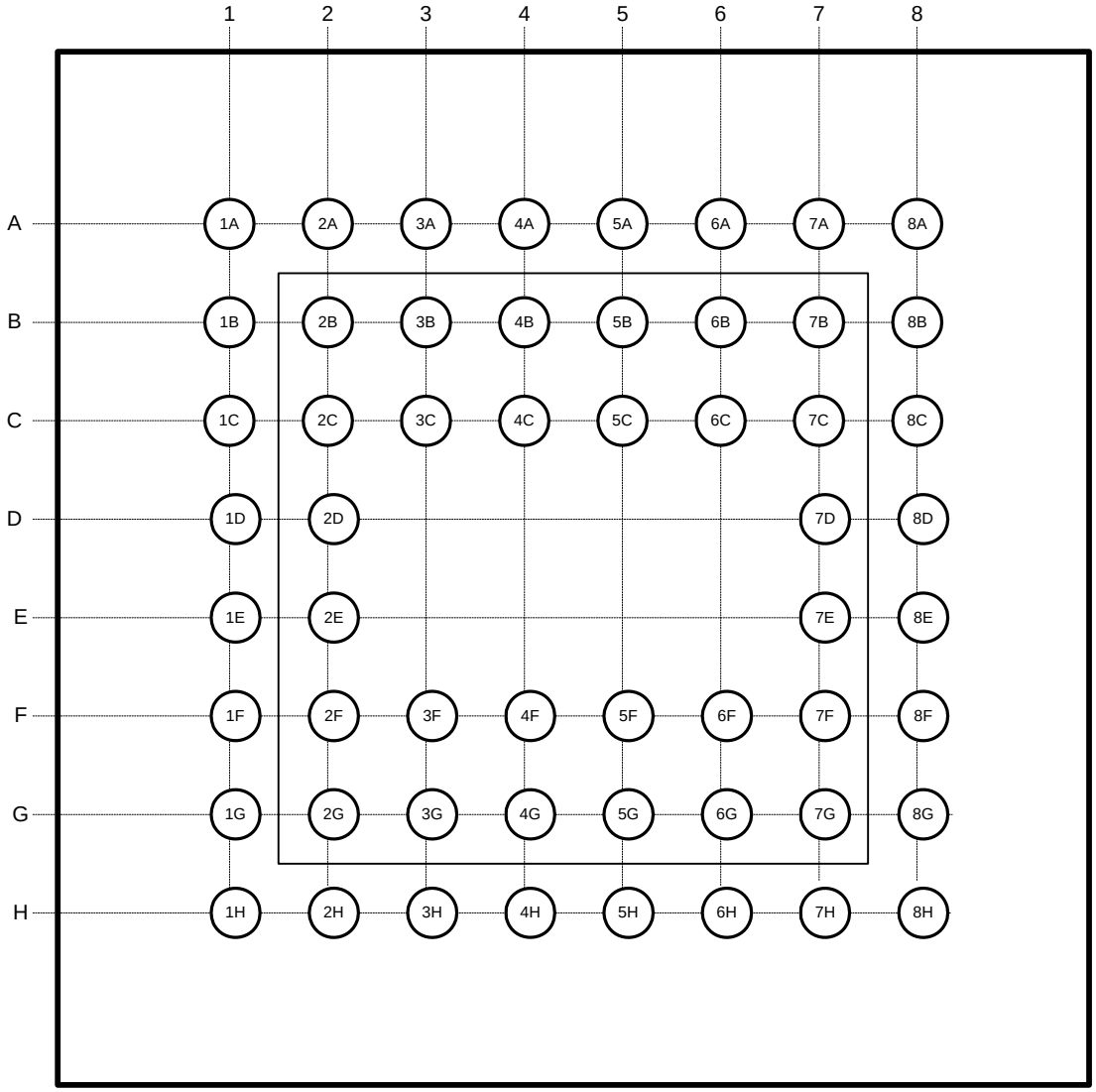

* denotes programmable polarity

**Figure 10. CY7C68013A/14A/15A/16A 56-Pin QFN Pin Assignment**




* denotes programmable polarity
** denotes CY7C68015A/CY7C68016A pinout

**Figure 11. CY7C68013A 56-pin VFBGA Pin Assignment – Top View**



## CY7C68013A/15A Pin Descriptions

**Table 11. FX2LP Pin Descriptions**[11]

| 128 TQFP | 100 TQFP | 56 SSOP | 56 QFN | 56 VFBGA | Name | Type | Default | Reset[12] | Description |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | 10 | 3 | 2D | AVCC | Power | N/A | N/A | **Analog VCC**. Connect this pin to the 3.3 V power source. This signal provides power to the analog section of the chip. |
| 17 | 16 | 14 | 7 | 1D | AVCC | Power | N/A | N/A | **Analog VCC**. Connect this pin to the 3.3 V power source. This signal provides power to the analog section of the chip. |
| 13 | 12 | 13 | 6 | 2F | AGND | Ground | N/A | N/A | **Analog Ground**. Connect to ground with as short a path as possible. |
| 20 | 19 | 17 | 10 | 1F | AGND | Ground | N/A | N/A | **Analog Ground**. Connect to ground with as short a path as possible. |
| 19 | 18 | 16 | 9 | 1E | DMINUS | I/O/Z | Z | N/A | **USB D– Signal**. Connect to the USB D– signal. |
| 18 | 17 | 15 | 8 | 2E | DPLUS | I/O/Z | Z | N/A | **USB D+ Signal**. Connect to the USB D+ signal. |
| 94 | – | – | – | – | A0 | Output | L | L | **8051 Address Bus**. This bus is driven at all times. When the 8051 is addressing internal RAM it reflects the internal address. |
| 95 | – | – | – | – | A1 | Output | L | L | |
| 96 | – | – | – | – | A2 | Output | L | L | |
| 97 | – | – | – | – | A3 | Output | L | L | |
| 117 | – | – | – | – | A4 | Output | L | L | |
| 118 | – | – | – | – | A5 | Output | L | L | |
| 119 | – | – | – | – | A6 | Output | L | L | |
| 120 | – | – | – | – | A7 | Output | L | L | |
| 126 | – | – | – | – | A8 | Output | L | L | |
| 127 | – | – | – | – | A9 | Output | L | L | |
| 128 | – | – | – | – | A10 | Output | L | L | |
| 21 | – | – | – | – | A11 | Output | L | L | |
| 22 | – | – | – | – | A12 | Output | L | L | |
| 23 | – | – | – | – | A13 | Output | L | L | |
| 24 | – | – | – | – | A14 | Output | L | L | |
| 25 | – | – | – | – | A15 | Output | L | L | |
| 59 | – | – | – | – | D0 | I/O/Z | Z | Z | **8051 Data Bus**. This bidirectional bus is high impedance when inactive, input for bus reads, and output for bus writes. The data bus is used for external 8051 program and data memory. The data bus is active only for external bus accesses, and is driven LOW in suspend. |
| 60 | – | – | – | – | D1 | I/O/Z | Z | Z | |
| 61 | – | – | – | – | D2 | I/O/Z | Z | Z | |
| 62 | – | – | – | – | D3 | I/O/Z | Z | Z | |
| 63 | – | – | – | – | D4 | I/O/Z | Z | Z | |
| 86 | – | – | – | – | D5 | I/O/Z | Z | Z | |
| 87 | – | – | – | – | D6 | I/O/Z | Z | Z | |
| 88 | – | – | – | – | D7 | I/O/Z | Z | Z | |
| 39 | – | – | – | – | PSEN# | Output | H | H | **Program Store Enable**. This active LOW signal indicates an 8051 code fetch from external memory. It is active for program memory fetches from 0x4000–0xFFFF when the EA pin is LOW, or from 0x0000–0xFFFF when the EA pin is HIGH. |

**Notes**
11. Unused inputs must not be left floating. Tie either HIGH or LOW as appropriate. Outputs should only be pulled up or down to ensure signals at power up and in standby. Note also that no pins should be driven while the device is powered down.
12. The Reset column indicates the state of signals during reset (RESET# asserted) or during Power on Reset (POR).

**Table 12. FX2LP Register Summary** (continued)

| Hex | Size | Name | Description | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | Default | Access |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E62B | 1 | ECC1B1 | ECC1 Byte 1 Address | LINE7 | LINE6 | LINE5 | LINE4 | LINE3 | LINE2 | LINE1 | LINE0 | 00000000 | R |
| E62C | 1 | ECC1B2 | ECC1 Byte 2 Address | COL5 | COL4 | COL3 | COL2 | COL1 | COL0 | LINE17 | LINE16 | 00000000 | R |
| E62D | 1 | ECC2B0 | ECC2 Byte 0 Address | LINE15 | LINE14 | LINE13 | LINE12 | LINE11 | LINE10 | LINE9 | LINE8 | 00000000 | R |
| E62E | 1 | ECC2B1 | ECC2 Byte 1 Address | LINE7 | LINE6 | LINE5 | LINE4 | LINE3 | LINE2 | LINE1 | LINE0 | 00000000 | R |
| E62F | 1 | ECC2B2 | ECC2 Byte 2 Address | COL5 | COL4 | COL3 | COL2 | COL1 | COL0 | 0 | 0 | 00000000 | R |
| E630 H.S. | 1 | EP2FIFOPFH[13] | Endpoint 2 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | IN:PKTS[2] OUT:PFC12 | IN:PKTS[1] OUT:PFC11 | IN:PKTS[0] OUT:PFC10 | 0 | PFC9 | PFC8 | 10001000 | bbbbbrbb |
| E630 F.S. | 1 | EP2FIFOPFH[13] | Endpoint 2 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | OUT:PFC12 | OUT:PFC11 | OUT:PFC10 | 0 | PFC9 | IN:PKTS[2] OUT:PFC8 | 10001000 | bbbbbrbb |
| E631 H.S. | 1 | EP2FIFOPFL[13] | Endpoint 2 / slave FIFO Programmable Flag L | PFC7 | PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E631 F.S | 1 | EP2FIFOPFL[13] | Endpoint 2 / slave FIFO Programmable Flag L | IN:PKTS[1] OUT:PFC7 | IN:PKTS[0] OUT:PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E632 H.S. | 1 | EP4FIFOPFH[13] | Endpoint 4 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | 0 | IN: PKTS[1] OUT:PFC10 | IN: PKTS[0] OUT:PFC9 | 0 | 0 | PFC8 | 10001000 | bbrbbrrb |
| E632 F.S | 1 | EP4FIFOPFH[13] | Endpoint 4 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | 0 | OUT:PFC10 | OUT:PFC9 | 0 | 0 | PFC8 | 10001000 | bbrbbrrb |
| E633 H.S. | 1 | EP4FIFOPFL[13] | Endpoint 4 / slave FIFO Programmable Flag L | PFC7 | PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E633 F.S | 1 | EP4FIFOPFL[13] | Endpoint 4 / slave FIFO Programmable Flag L | IN: PKTS[1] OUT:PFC7 | IN: PKTS[0] OUT:PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E634 H.S. | 1 | EP6FIFOPFH[13] | Endpoint 6 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | IN:PKTS[2] OUT:PFC12 | IN:PKTS[1] OUT:PFC11 | IN:PKTS[0] OUT:PFC10 | 0 | PFC9 | PFC8 | 00001000 | bbbbbrbb |
| E634 F.S | 1 | EP6FIFOPFH[13] | Endpoint 6 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | OUT:PFC12 | OUT:PFC11 | OUT:PFC10 | 0 | PFC9 | IN:PKTS[2] OUT:PFC8 | 00001000 | bbbbbrbb |
| E635 H.S. | 1 | EP6FIFOPFL[13] | Endpoint 6 / slave FIFO Programmable Flag L | PFC7 | PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E635 F.S | 1 | EP6FIFOPFL[13] | Endpoint 6 / slave FIFO Programmable Flag L | IN:PKTS[1] OUT:PFC7 | IN:PKTS[0] OUT:PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E636 H.S. | 1 | EP8FIFOPFH[13] | Endpoint 8 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | 0 | IN: PKTS[1] OUT:PFC10 | IN: PKTS[0] OUT:PFC9 | 0 | 0 | PFC8 | 00001000 | bbrbbrrb |
| E636 F.S | 1 | EP8FIFOPFH[13] | Endpoint 8 / slave FIFO Programmable Flag H | DECIS | PKTSTAT | 0 | OUT:PFC10 | OUT:PFC9 | 0 | 0 | PFC8 | 00001000 | bbrbbrrb |
| E637 H.S. | 1 | EP8FIFOPFL[13] | Endpoint 8 / slave FIFO Programmable Flag L | PFC7 | PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| E637 F.S | 1 | EP8FIFOPFL[13] | Endpoint 8 / slave FIFO Programmable Flag L | IN: PKTS[1] OUT:PFC7 | IN: PKTS[0] OUT:PFC6 | PFC5 | PFC4 | PFC3 | PFC2 | PFC1 | PFC0 | 00000000 | RW |
| | 8 | reserved | | | | | | | | | | | |
| E640 | 1 | EP2ISOINPKTS | EP2 (if ISO) IN Packets per frame (1-3) | AADJ | 0 | 0 | 0 | 0 | 0 | INPPF1 | INPPF0 | 00000001 | brrrrrbb |
| E641 | 1 | EP4ISOINPKTS | EP4 (if ISO) IN Packets per frame (1-3) | AADJ | 0 | 0 | 0 | 0 | 0 | INPPF1 | INPPF0 | 00000001 | brrrrrrr |
| E642 | 1 | EP6ISOINPKTS | EP6 (if ISO) IN Packets per frame (1-3) | AADJ | 0 | 0 | 0 | 0 | 0 | INPPF1 | INPPF0 | 00000001 | brrrrrbb |
| E643 | 1 | EP8ISOINPKTS | EP8 (if ISO) IN Packets per frame (1-3) | AADJ | 0 | 0 | 0 | 0 | 0 | INPPF1 | INPPF0 | 00000001 | brrrrrrr |
| E644 | 4 | reserved | | | | | | | | | | | |
| E648 | 1 | INPKTEND[13] | Force IN Packet End | Skip | 0 | 0 | 0 | EP3 | EP2 | EP1 | EP0 | xxxxxxxx | W |
| E649 | 7 | OUTPKTEND[13] | Force OUT Packet End | Skip | 0 | 0 | 0 | EP3 | EP2 | EP1 | EP0 | xxxxxxxx | W |
| | | INTERRUPTS | | | | | | | | | | | |
| E650 | 1 | EP2FIFOIE[13] | Endpoint 2 slave FIFO Flag Interrupt Enable | 0 | 0 | 0 | 0 | EDGEPF | PF | EF | FF | 00000000 | RW |
| E651 | 1 | EP2FIFOIRQ[13,14] | Endpoint 2 slave FIFO Flag Interrupt Request | 0 | 0 | 0 | 0 | 0 | PF | EF | FF | 00000000 | rrrrrbbb |
| E652 | 1 | EP4FIFOIE[13] | Endpoint 4 slave FIFO Flag Interrupt Enable | 0 | 0 | 0 | 0 | EDGEPF | PF | EF | FF | 00000000 | RW |
| E653 | 1 | EP4FIFOIRQ[13,14] | Endpoint 4 slave FIFO Flag Interrupt Request | 0 | 0 | 0 | 0 | 0 | PF | EF | FF | 00000000 | rrrrrbbb |
| E654 | 1 | EP6FIFOIE[13] | Endpoint 6 slave FIFO Flag Interrupt Enable | 0 | 0 | 0 | 0 | EDGEPF | PF | EF | FF | 00000000 | RW |
| E655 | 1 | EP6FIFOIRQ[13,14] | Endpoint 6 slave FIFO Flag Interrupt Request | 0 | 0 | 0 | 0 | 0 | PF | EF | FF | 00000000 | rrrrrbbb |
| E656 | 1 | EP8FIFOIE[13] | Endpoint 8 slave FIFO Flag Interrupt Enable | 0 | 0 | 0 | 0 | EDGEPF | PF | EF | FF | 00000000 | RW |
| E657 | 1 | EP8FIFOIRQ[13,14] | Endpoint 8 slave FIFO Flag Interrupt Request | 0 | 0 | 0 | 0 | 0 | PF | EF | FF | 00000000 | rrrrrbbb |
| E658 | 1 | IBNIE | IN-BULK-NAK Interrupt Enable | 0 | 0 | EP8 | EP6 | EP4 | EP2 | EP1 | EP0 | 00000000 | RW |
| E659 | 1 | IBNIRQ[14] | IN-BULK-NAK interrupt Request | 0 | 0 | EP8 | EP6 | EP4 | EP2 | EP1 | EP0 | 00xxxxxx | rrbbbbbb |
| E65A | 1 | NAKIE | Endpoint Ping-NAK / IBN Interrupt Enable | EP8 | EP6 | EP4 | EP2 | EP1 | EP0 | 0 | IBN | 00000000 | RW |
| E65B | 1 | NAKIRQ[14] | Endpoint Ping-NAK / IBN Interrupt Request | EP8 | EP6 | EP4 | EP2 | EP1 | EP0 | 0 | IBN | xxxxxx0x | bbbbbbrb |
| E65C | 1 | USBIE | USB Int Enables | 0 | EP0ACK | HSGRANT | URES | SUSP | SUTOK | SOF | SUDAV | 00000000 | RW |

**Note**
14. The register can only be reset; it cannot be set.

**Table 12. FX2LP Register Summary** (continued)

| Hex | Size | Name | Description | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | Default | Access |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E6CF | 1 | GPIFTCB2[13] | GPIF Transaction Count Byte 2 | TC23 | TC22 | TC21 | TC20 | TC19 | TC18 | TC17 | TC16 | 00000000 | RW |
| E6D0 | 1 | GPIFTCB1[13] | GPIF Transaction Count Byte 1 | TC15 | TC14 | TC13 | TC12 | TC11 | TC10 | TC9 | TC8 | 00000000 | RW |
| E6D1 | 1 | GPIFTCB0[13] | GPIF Transaction Count Byte 0 | TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 | 00000001 | RW |
| | 2 | reserved | | | | | | | | | | 00000000 | RW |
| | | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| E6D2 | 1 | EP2GPIFFLGSEL[13] | Endpoint 2 GPIF Flag select | 0 | 0 | 0 | 0 | 0 | 0 | FS1 | FS0 | 00000000 | RW |
| E6D3 | 1 | EP2GPIFPFSTOP | Endpoint 2 GPIF stop transaction on prog. flag | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FIFO2FLAG | 00000000 | RW |
| E6D4 | 1 | EP2GPIFTRIG[13] | Endpoint 2 GPIF Trigger | x | x | x | x | x | x | x | x | xxxxxxxx | W |
| | 3 | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| E6DA | 1 | EP4GPIFFLGSEL[13] | Endpoint 4 GPIF Flag select | 0 | 0 | 0 | 0 | 0 | 0 | FS1 | FS0 | 00000000 | RW |
| E6DB | 1 | EP4GPIFPFSTOP | Endpoint 4 GPIF stop transaction on GPIF Flag | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FIFO4FLAG | 00000000 | RW |
| E6DC | 1 | EP4GPIFTRIG[13] | Endpoint 4 GPIF Trigger | x | x | x | x | x | x | x | x | xxxxxxxx | W |
| | 3 | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| E6E2 | 1 | EP6GPIFFLGSEL[13] | Endpoint 6 GPIF Flag select | 0 | 0 | 0 | 0 | 0 | 0 | FS1 | FS0 | 00000000 | RW |
| E6E3 | 1 | EP6GPIFPFSTOP | Endpoint 6 GPIF stop transaction on prog. flag | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FIFO6FLAG | 00000000 | RW |
| E6E4 | 1 | EP6GPIFTRIG[13] | Endpoint 6 GPIF Trigger | x | x | x | x | x | x | x | x | xxxxxxxx | W |
| | 3 | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| | | reserved | | | | | | | | | | | |
| E6EA | 1 | EP8GPIFFLGSEL[13] | Endpoint 8 GPIF Flag select | 0 | 0 | 0 | 0 | 0 | 0 | FS1 | FS0 | 00000000 | RW |
| E6EB | 1 | EP8GPIFPFSTOP | Endpoint 8 GPIF stop transaction on prog. flag | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FIFO8FLAG | 00000000 | RW |
| E6EC | 1 | EP8GPIFTRIG[13] | Endpoint 8 GPIF Trigger | x | x | x | x | x | x | x | x | xxxxxxxx | W |
| | 3 | reserved | | | | | | | | | | | |
| E6F0 | 1 | XGPIFSGLDATH | GPIF Data H (16-bit mode only) | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | xxxxxxxx | RW |
| E6F1 | 1 | XGPIFSGLDATLX | Read/Write GPIF Data L & trigger transaction | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| E6F2 | 1 | XGPIFSGLDATLNOX | Read GPIF Data L, no transaction trigger | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | R |
| E6F3 | 1 | GPIFREADYCFG | Internal RDY, Sync/Async, RDY pin states | INTRDY | SAS | TCXRDY5 | 0 | 0 | 0 | 0 | 0 | 00000000 | bbbrrrrr |
| | | | | | | | | | | | | | |
| E6F4 | 1 | GPIFREADYSTAT | GPIF Ready Status | 0 | 0 | RDY5 | RDY4 | RDY3 | RDY2 | RDY1 | RDY0 | 00xxxxxx | R |
| E6F5 | 1 | GPIFABORT | Abort GPIF Waveforms | x | x | x | x | x | x | x | x | xxxxxxxx | W |
| E6F6 | 2 | reserved | | | | | | | | | | | |
| | | ENDPOINT BUFFERS | | | | | | | | | | | |
| E740 | 64 | EP0BUF | EP0-IN/-OUT buffer | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| E780 | 64 | EP10UTBUF | EP1-OUT buffer | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| E7C0 | 64 | EP1INBUF | EP1-IN buffer | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| E800 | 2048 | reserved | | | | | | | | | | | RW |
| F000 | 1024 | EP2FIFOBUF | 512/1024 byte EP 2 / slave FIFO buffer (IN or OUT) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| F400 | 512 | EP4FIFOBUF | 512 byte EP 4 / slave FIFO buffer (IN or OUT) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| F600 | 512 | reserved | | | | | | | | | | | |
| F800 | 1024 | EP6FIFOBUF | 512/1024 byte EP 6 / slave FIFO buffer (IN or OUT) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| FC00 | 512 | EP8FIFOBUF | 512 byte EP 8 / slave FIFO buffer (IN or OUT) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | xxxxxxxx | RW |
| FE00 | 512 | reserved | | | | | | | | | | | |

## PORTC Strobe Feature Timings

The RD# and WR# are present in the 100-pin version and the 128-pin package. In these 100-pin and 128-pin versions, an 8051 control bit can be set to pulse the RD# and WR# pins when the 8051 reads from or writes to PORTC. This feature is enabled by setting PORTCSTB bit in CPUCS register.

The RD# and WR# strobes are asserted for two CLKOUT cycles when PORTC is accessed.

The WR# strobe is asserted two clock cycles after PORTC is updated and is active for two clock cycles after that, as shown in Figure 16.

As for read, the value of PORTC three clock cycles before the assertion of RD# is the value that the 8051 reads in. The RD# is pulsed for two clock cycles after three clock cycles from the point when the 8051 has performed a read function on PORTC.

The RD# signal prompts the external logic to prepare the next data byte. Nothing gets sampled internally on assertion of the RD# signal itself; it is just a prefetch type signal to get the next data byte prepared. So, using it with that in mind easily meets the setup time to the next read.

The purpose of this pulsing of RD# is to allow the external peripheral to know that the 8051 is done reading PORTC and the data was latched into PORTC three CLKOUT cycles before asserting the RD# signal. After the RD# is pulsed, the external logic can update the data on PORTC.

Following is the timing diagram of the read and write strobing function on accessing PORTC. Refer to Data Memory Read[21] and Data Memory Write[23] for details on propagation delay of RD# and WR# signals.

**Figure 16. WR# Strobe Function when PORTC is Accessed by 8051**

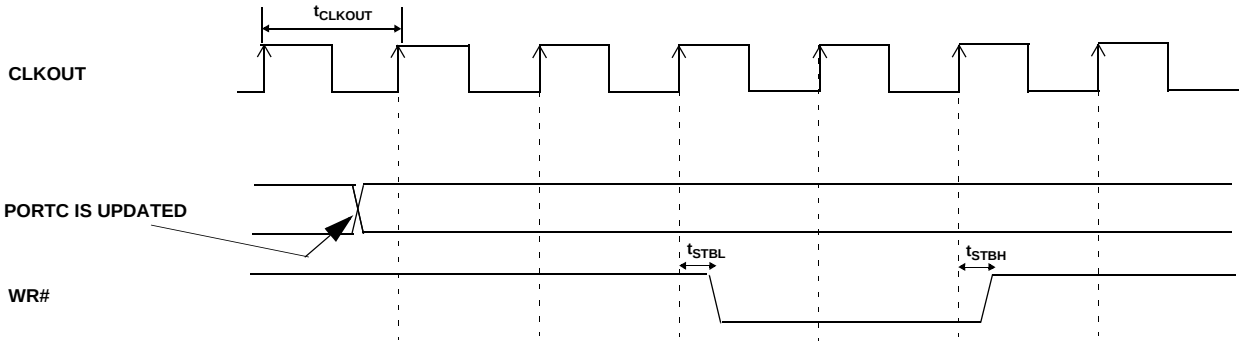
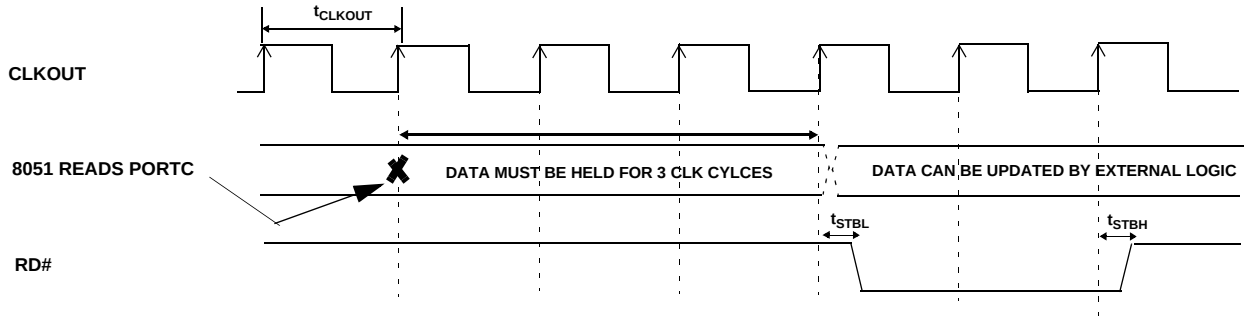



**Figure 17. RD# Strobe Function when PORTC is Accessed by 8051**



## Slave FIFO Synchronous Read

**Figure 19. Slave FIFO Synchronous Read Timing Diagram**[24]

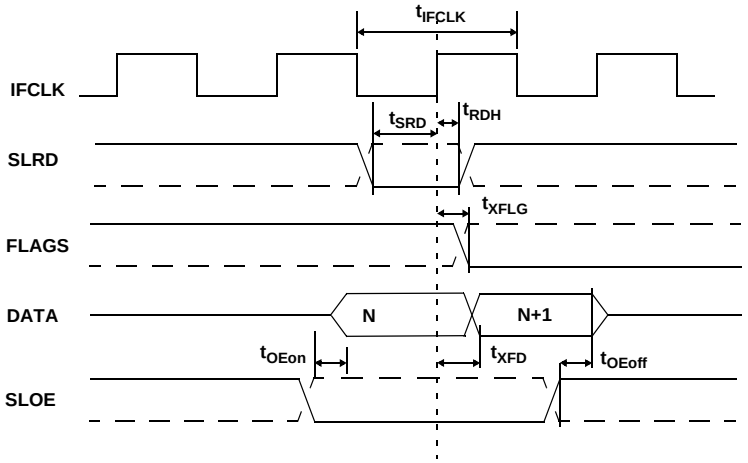



**Table 20. Slave FIFO Synchronous Read Parameters with Internally Sourced IFCLK**[25]

| Parameter | Description | Min | Max | Typ | | Unit |
|---|---|---|---|---|---|---|
| | | | | Min | Max | |
| $t_{IFCLK}$ | IFCLK period | 20.83 | – | – | – | ns |
| $t_{SRD}$ | SLRD to clock setup time | 18.7 | – | – | – | ns |
| $t_{RDH}$ | Clock to SLRD hold time | 0 | – | – | – | ns |
| $t_{OEon}$ | SLOE turn on to FIFO data valid | – | 10.5 | – | – | ns |
| $t_{OEoff}$ | SLOE turn off to FIFO data hold | – | 10.5 | – | – | ns |
| $t_{XFLG}$ | Clock to FLAGS output propagation delay | – | 9.5 | – | – | ns |
| $t_{XFD}$ | Clock to FIFO data output propagation delay | – | 11 | – | – | ns |
| $t_{IFCLKR}$ | IFCLK rise time | – | – | – | 900 | ps |
| $t_{IFCLKF}$ | IFCLK fall time | – | – | – | 900 | ps |
| $t_{IFCLKOD}$ | IFCLK output duty cycle | – | – | 49 | 51 | % |
| $t_{IFCLKJ}$ | IFCLK jitter peak to peak | – | – | – | 300 | ps |

**Table 21. Slave FIFO Synchronous Read Parameters with Externally Sourced IFCLK[25]**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| $t_{IFCLK}$ | IFCLK period | 20.83 | 200 | ns |
| $t_{SRD}$ | SLRD to clock setup time | 12.7 | – | ns |
| $t_{RDH}$ | Clock to SLRD hold time | 3.7 | – | ns |
| $t_{OEon}$ | SLOE turn on to FIFO data valid | – | 10.5 | ns |
| $t_{OEoff}$ | SLOE turn off to FIFO data hold | – | 10.5 | ns |
| $t_{XFLG}$ | Clock to FLAGS output propagation delay | – | 13.5 | ns |
| $t_{XFD}$ | Clock to FIFO data output propagation delay | – | 15 | ns |

## Slave FIFO Asynchronous Read

**Figure 20. Slave FIFO Asynchronous Read Timing Diagram[24]**

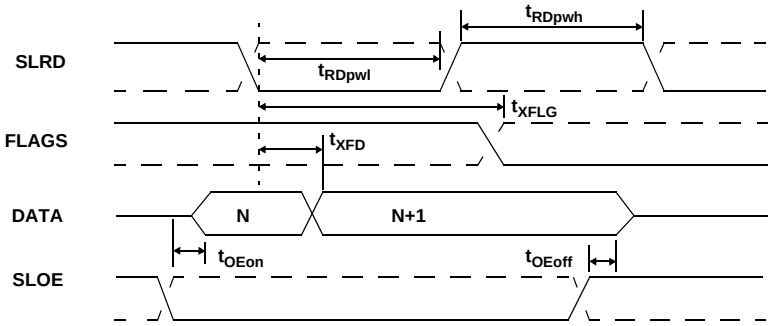



**Table 22. Slave FIFO Asynchronous Read Parameters[27]**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| $t_{RDpwl}$ | SLRD pulse width LOW | 50 | – | ns |
| $t_{RDpwh}$ | SLRD pulse width HIGH | 50 | – | ns |
| $t_{XFLG}$ | SLRD to FLAGS output propagation delay | – | 70 | ns |
| $t_{XFD}$ | SLRD to FIFO data output propagation delay | – | 15 | ns |
| $t_{OEon}$ | SLOE turn-on to FIFO data valid | – | 10.5 | ns |
| $t_{OEoff}$ | SLOE turn-off to FIFO data hold | – | 10.5 | ns |

**Note**
27. Slave FIFO asynchronous parameter values use internal IFCLK setting at 48 MHz.

### Slave FIFO Address to Flags/Data

**Figure 27. Slave FIFO Address to Flags/Data Timing Diagram[24]**

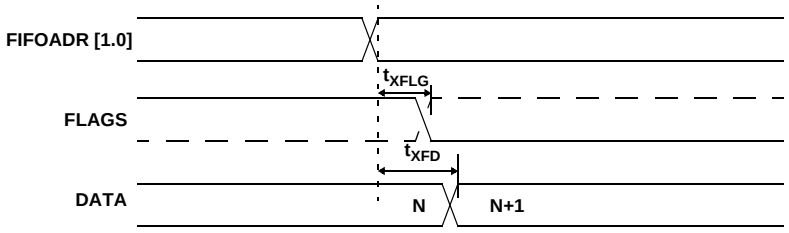



**Table 30. Slave FIFO Address to Flags/Data Parameters**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| $t_{XFLG}$ | FIFOADR[1:0] to FLAGS output propagation delay | – | 10.7 | ns |
| $t_{XFD}$ | FIFOADR[1:0] to FIFODATA output propagation delay | – | 14.3 | ns |

### Slave FIFO Synchronous Address

**Figure 28. Slave FIFO Synchronous Address Timing Diagram[24]**

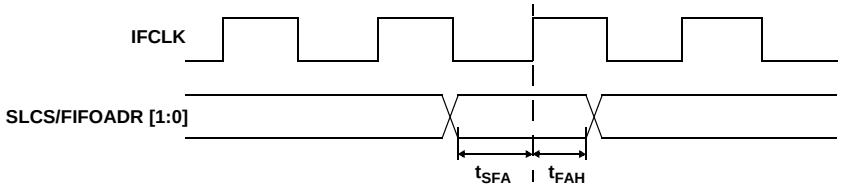



**Table 31. Slave FIFO Synchronous Address Parameters[25]**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| $t_{IFCLK}$ | Interface clock period | 20.83 | 200 | ns |
| $t_{SFA}$ | FIFOADR[1:0] to clock setup time | 25 | – | ns |
| $t_{FAH}$ | Clock to FIFOADR[1:0] hold time | 10 | – | ns |

### Slave FIFO Asynchronous Address

**Figure 29. Slave FIFO Asynchronous Address Timing Diagram[24]**

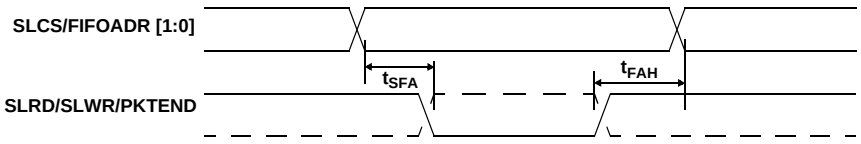



**Table 32. Slave FIFO Asynchronous Address Parameters[27]**

| Parameter | Description | Min | Max | Unit |
|---|---|---|---|---|
| $t_{SFA}$ | FIFOADR[1:0] to SLRD/SLWR/PKTEND setup time | 10 | – | ns |
| $t_{FAH}$ | RD/WR/PKTEND to FIFOADR[1:0] hold time | 10 | – | ns |

## Sequence Diagram

*Single and Burst Synchronous Read Example*

**Figure 30. Slave FIFO Synchronous Read Sequence and Timing Diagram**[24]

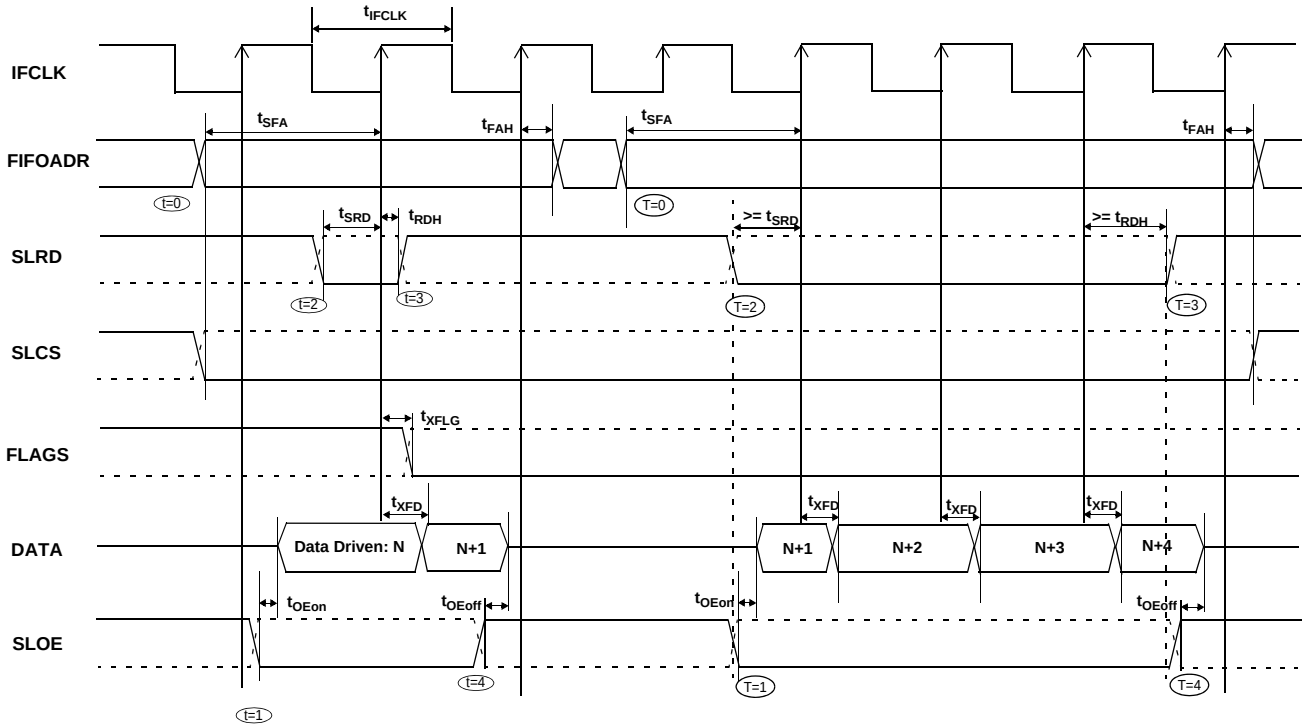



**Figure 31. Slave FIFO Synchronous Sequence of Events Diagram**

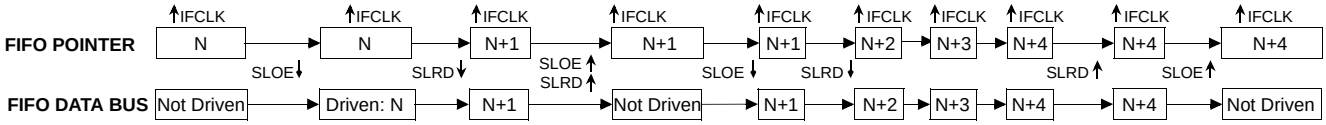



Figure 30 on page 52 shows the timing relationship of the SLAVE FIFO signals during a synchronous FIFO read using IFCLK as the synchronizing clock. The diagram illustrates a single read followed by a burst read.

- At t = 0, the FIFO address is stable and the signal SLCS is asserted (SLCS may be tied LOW in some applications). Note that $t_{SFA}$ has a minimum of 25 ns. This means that when IFCLK is running at 48 MHz, the FIFO address setup time is more than one IFCLK cycle.
- At t = 1, SLOE is asserted. SLOE is an output enable only, whose sole function is to drive the data bus. The data that is driven on the bus is the data that the internal FIFO pointer is currently pointing to. In this example it is the first data value in the FIFO. **Note**: the data is prefetched and is driven on the bus when SLOE is asserted.
- At t = 2, SLRD is asserted. SLRD must meet the setup time of $t_{SRD}$ (time from asserting the SLRD signal to the rising edge of the IFCLK) and maintain a minimum hold time of $t_{RDH}$ (time from the IFCLK edge to the deassertion of the SLRD signal). If the SLCS signal is used, it must be asserted before SLRD is asserted (The SLCS and SLRD signals must both be asserted to start a valid read condition).
- The FIFO pointer is updated on the rising edge of the IFCLK, while SLRD is asserted. This starts the propagation of data from the newly addressed location to the data bus. After a propagation delay of $t_{XFD}$ (measured from the rising edge of IFCLK) the new data value is present. N is the first data value read from the FIFO. To have data on the FIFO data bus, SLOE MUST also be asserted.

The same sequence of events are shown for a burst read and are marked with the time indicators of T = 0 through 5.

**Note** For the burst mode, the SLRD and SLOE are left asserted during the entire duration of the read. In the burst read mode, when SLOE is asserted, data indexed by the FIFO pointer is on the data bus. During the first read cycle, on the rising edge of the clock, the FIFO pointer is updated and incremented to point to address N+1. For each subsequent rising edge of IFCLK, while the SLRD is asserted, the FIFO pointer is incremented and the next data value is placed on the data bus.

*Sequence Diagram of a Single and Burst Asynchronous Read*

**Figure 33. Slave FIFO Asynchronous Read Sequence and Timing Diagram**[24]

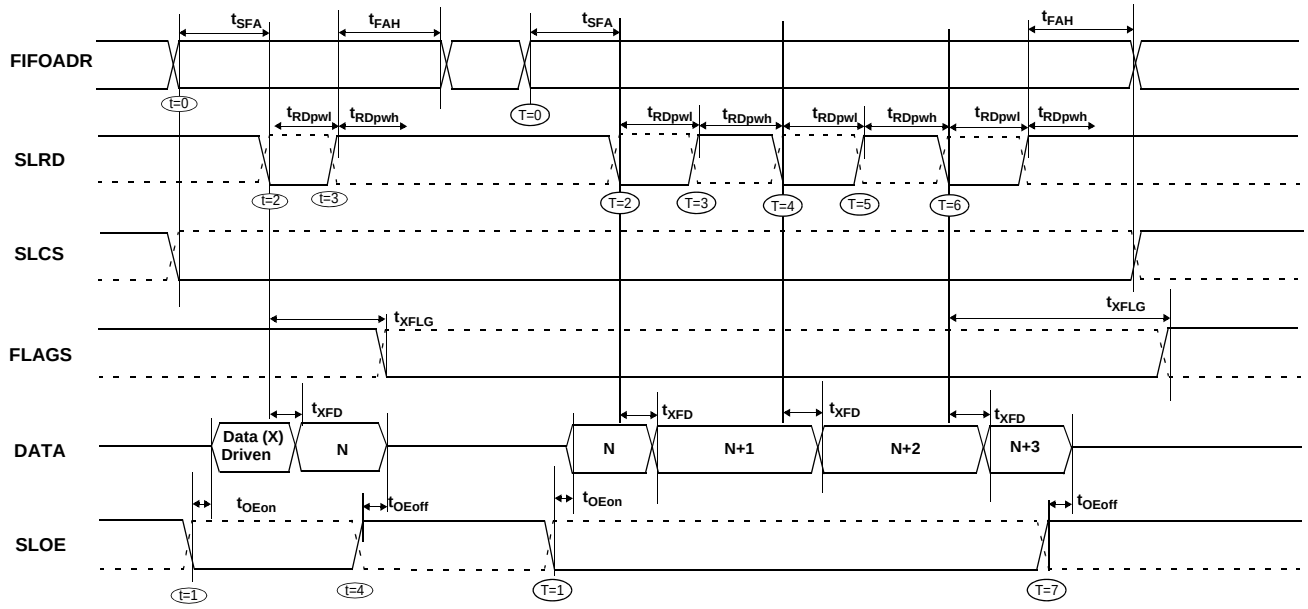



**Figure 34. Slave FIFO Asynchronous Read Sequence of Events Diagram**

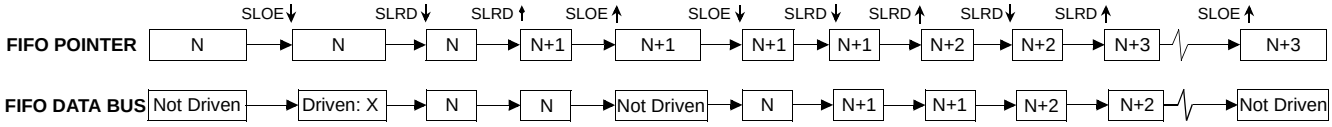


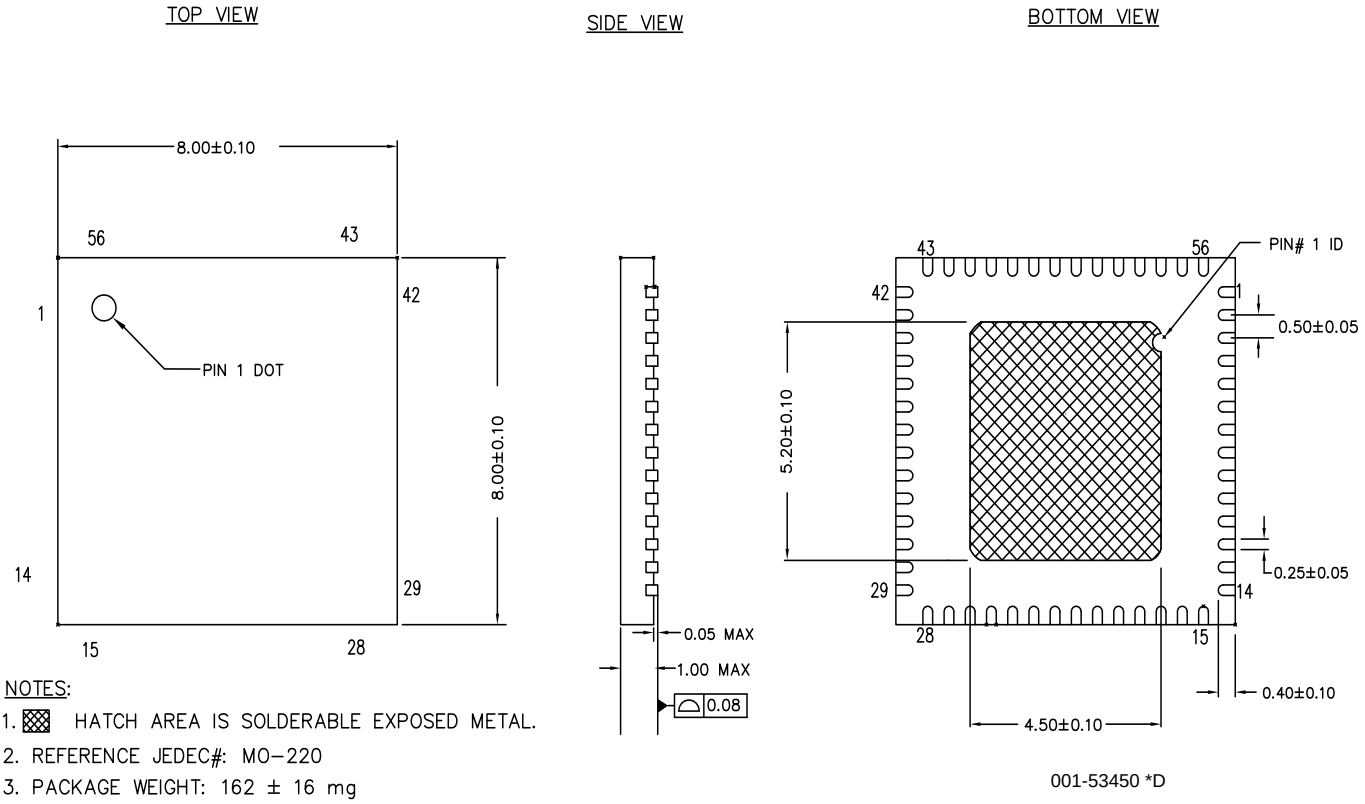

Figure 33 shows the timing relationship of the SLAVE FIFO signals during an asynchronous FIFO read. It shows a single read followed by a burst read.

- At t = 0, the FIFO address is stable and the SLCS signal is asserted.
- At t = 1, SLOE is asserted. This results in the data bus being driven. The data that is driven on to the bus is the previous data, the data that was in the FIFO from anearlier read cycle.
- At t = 2, SLRD is asserted. The SLRD must meet the minimum active pulse of $t_{RDpwl}$ and minimum de-active pulse width of $t_{RDpwh}$. If SLCS is used, then SLCS must be asserted before SLRD is asserted (The SLCS and SLRD signals must both be asserted to start a valid read condition.)
- The data that is driven, after asserting SLRD, is the updated data from the FIFO. This data is valid after a propagation delay of $t_{XFD}$ from the activating edge of SLRD. In Figure 33, data N is the first valid data read from the FIFO. For data to appear on the data bus during the read cycle (SLRD is asserted), SLOE must be in an asserted state. SLRD and SLOE can also be tied together.

The same sequence of events is also shown for a burst read marked with T = 0 through 5.

**Note** In the burst read mode, during SLOE is asserted, the data bus is in a driven state and outputs the previous data. After SLRD is asserted, the data from the FIFO is driven on the data bus (SLOE must also be asserted) and then the FIFO pointer is incremented.

**Figure 37. 56-Pin QFN 8 × 8 mm Sawn Version (001-53450)**




NOTES:
1. HATCH AREA IS SOLDERABLE EXPOSED METAL.
2. REFERENCE JEDEC#: MO–220
3. PACKAGE WEIGHT: 162 ± 16 mg
4. ALL DIMENSIONS ARE IN MILLIMETERS

## Quad Flat Package No Leads (QFN) Package Design Notes

Electrical contact of the part to the PCB is made by soldering the leads on the bottom surface of the package to the PCB. Therefore, special attention is required to the heat transfer area below the package to provide a good thermal bond to the circuit board. Design a copper (Cu) fill in the PCB as a thermal pad under the package. Heat is transferred from the FX2LP through the device's metal paddle on the bottom side of the package. Heat from here is conducted to the PCB at the thermal pad. It is then conducted from the thermal pad to the PCB inner ground plane by a 5 × 5 array of via. A via is a plated-through hole in the PCB with a finished diameter of 13 mil. The QFN's metal die paddle must be soldered to the PCB's thermal pad. Solder mask is placed on the board top side over each via to resist solder flow into the via. The mask on the top side also minimizes outgassing during the solder reflow process.

For further information on this package design, refer to application notes for Surface Mount Assembly of Amkor's MicroLeadFrame (MLF) Packages. You can find this on Amkor's website http://www.amkor.com.

This application note provides detailed information about board mounting guidelines, soldering flow, rework process, etc.

Figure 41 shows a cross-sectional area underneath the package. The cross section is of only one via. The solder paste template should be designed to allow at least 50% solder coverage. The thickness of the solder paste template should be 5 mil. Use the No Clean type 3 solder paste for mounting the part. Nitrogen purge is recommended during reflow.

Figure 42 is a plot of the solder mask pattern and Figure 43 displays an X-Ray image of the assembly (darker areas indicate solder).

**Figure 41. Cross-section of the Area Underneath the QFN Package**

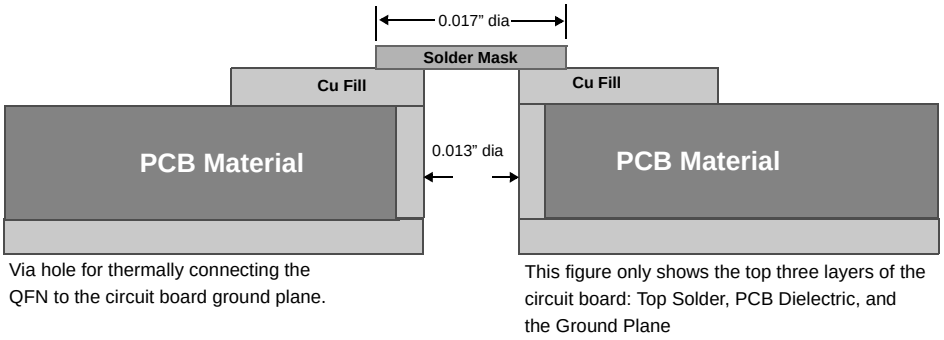



**Figure 42. Plot of the Solder Mask (White Area)**

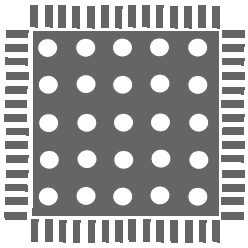
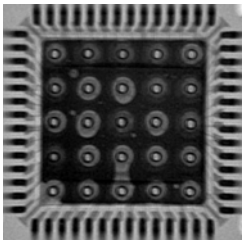


**Figure 43. X-ray Image of the Assembly**



# Sales, Solutions, and Legal Information

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

### Products

| | |
|---|---|
| ARM® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Lighting & Power Control | cypress.com/powerpsoc |
| Memory | cypress.com/memory |
| PSoC | cypress.com/psoc |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless/RF | cypress.com/wireless |

### PSoC®Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

### Cypress Developer Community

Forums | Projects | Video | Blogs | Training | Components

### Technical Support

cypress.com/support