

## USB2-SOFT USB 2.0 DEVICE SIE VHDL SOURCE CODE OVERVIEW

### Overview

**USB2-SOFT** is the VHDL source code to connect an FPGA-based USB device to a USB host. The code supports both high-speed (480 Mbits/s) and full-speed (12 Mbits/s) connections.

This FPGA code interfaces with a physical layer transceiver (PHY). The code is compatible with two industry standard interfaces commonly found in USB PHYs:

- 12-pin ULPI [2] (example: USB3300 from SMSC [5])
- 22-pin UTMI

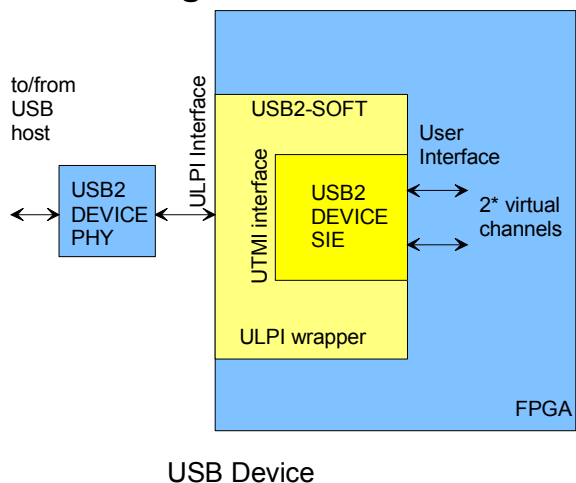
The code is structured so that the ULPI interface is a wrapper around the UTMI SIE.

The code is a Serial Interface Engine (SIE) implementing the Device side of the USB 2.0 standard protocol layer [1] including control and bulk endpoints.

**USB2-SOFT** does NOT support the following: Host-side protocol, OTG, low-speed, Interrupt and Isochronous endpoints.

The component's very efficient implementation makes it suitable for instantiation within a small FPGA. For example, it takes 22% of a small Spartan-6 XC6SLX16 [3][4].

### Block Diagram



(\*) The code is written for two bi-directional virtual channels (4 endpoints). Developers can easily change the number of channels/endpoints by editing the *USB20.vhd* source code.

### Target Hardware

The code is written in generic VHDL so that it can be ported to a variety of FPGAs. The code was developed and tested on a Xilinx Spartan-6 XC6SLX FPGA.

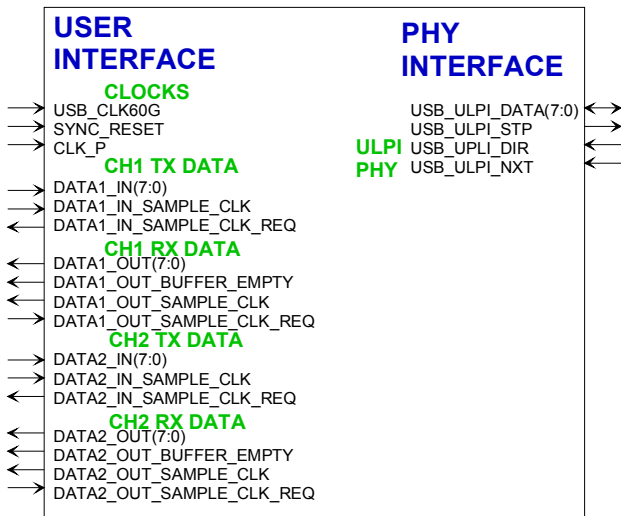
It can be easily ported to any Xilinx Virtex-5, Virtex-6, Spartan-6, Spartan-3 FPGAs and other FPGAs capable of running at 60 MHz.

### Device Utilization Summary

Device: Xilinx Spartan-6

Number of slices	516
Flip Flops	670
LUTs	1150
RAMB16BWERs	5
DSP48A1s	0
GCLKs	2
DCMs	1

## Interfaces

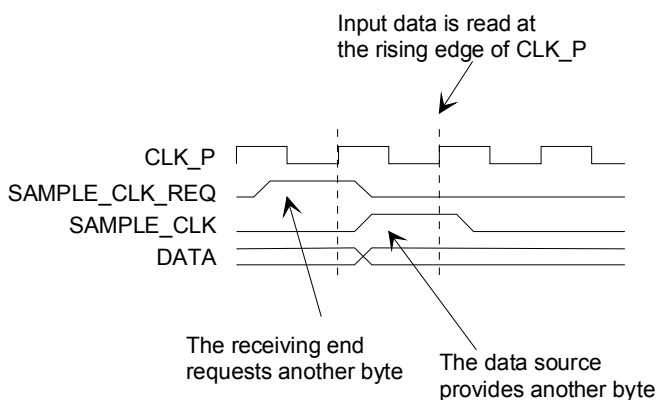


### User Interface

The user interface is synchronous with the user-supplied processing clock `CLK_P`. In order to meet the timing requirements, `CLK_P` must be defined as a global clock (i.e. a `BUFG` output).

`CLK_P` does not have to be the same as, or be related to the 60 MHz PHY clock.

Each data stream, whether input or output, follows the same timing diagram as illustrated below:



Each `DATA` byte is read at the rising edge of `CLK_P` when `SAMPLE_CLK` = '1'. In a sense, `SAMPLE_CLK` is really an 'enable' or 'valid' qualifying signal.

The stream data flow is controlled by means of the `SAMPLE_CLK_REQ` signal. '0' means that the receiving end does not have room for any more

data. The data source should not send data unless `SAMPLE_CLK_REQ` = '1'.

For maximum throughput, 16Kbit elastic buffers are included in both tx/rx directions at the user interface. This allows the USB engine to multi-task, sending data to the PHY while accepting subsequent data from the user and vice versa.

The USB protocol is invisible to the user. The user has no control, nor visibility of the data segmentation into `DATA0`/`DATA1` packets, frame check sequence (CRC) insertion and removal.

### PHY Interface

The PHY interface is synchronous with the PHY-supplied 60 MHz reference clock `USB_CLK60`.

Recommendation for best performance:

- At the time of PCB layout, connect the PHY 60 MHz reference clock to a FPGA global clock `GCLK` input port.
- Inside the FPGA, re-generate the 60 MHz reference clock through a `DCM` or `PLL` followed by a global buffer `BUFG`. An example is available in [6].

The most critical timing in this design is the time it takes for the `USB_ULPI_DIR` and `USB_ULPI_NXT` input signals to propagate through the FPGA and generate the appropriate `USB_ULPI_DATA` output by the next clock. It is thus important to define FPGA timing constraints as follows:

```
INST "USB_ULPI_NXT" TNM = USB_ULPI_IN;
INST "USB_ULPI_DIR" TNM = USB_ULPI_IN;
TIMEGRP "USB_ULPI_IN" OFFSET = IN 13.666 ns BEFORE USB_CLK60; #3ns at ULPI PHY output, period 16.6ns
```

### Configuration

There are no run-time configuration parameters. The two most likely customizations a developer may be tasked to implement are:

- adding/removing endpoints, and
- changing the descriptors.

## Adding/Removing Endpoints

It is quite easy to add or remove USB endpoints (for example to create an application with more transmit streams than receive streams). To do so, one must cut/paste/edit sections of the *USB20.vhd* component.

To create an additional bulk-out endpoint, cut and paste the code between

```
--// ENDPOINT 2 TRANSFER STATE  
MACHINE  
and  
---//END OF ENDPOINT 2 TRANSFER STATE  
MACHINE
```

then rename the interface from IF0 to IF?.

Likewise, to create an additional bulk-in endpoint, cut and paste the code between

```
--// ENDPOINT 3 TRANSFER STATE  
MACHINE  
and  
---//END OF ENDPOINT 3 TRANSFER STATE  
MACHINE
```

then rename the interface from IF0 to IF?.

## Changing Descriptors

USB devices report their attributes using descriptors. Descriptor strings include manufacturer's name, product's name, number of interfaces and endpoints, etc.

Descriptors are stored in read-only block RAM according to the following memory map:  
0x00 - 0x11 Device descriptor  
0x12 - 0x1B Device\_Qualifier descriptor  
0x1C - 0x52 Configuration descriptor (includes interface and endpoints)  
0x53 - 0x89 Other-speed configuration descriptor (offset by x37 from configuration descriptor)

0xA0 - 0xA2 String0 descriptor  
0xA3 - 0xBC String1 descriptor  
0xBD - 0xCE String2 descriptor  
0xCF - 0xE4 String3 descriptor  
0xE5 - 0xFA String4 descriptor

Descriptors are formatted as specified in the USB 2.0 specifications, section 9.5.

## Exclusions

USB2-SOFT does NOT support the following:

- Host-side protocol
- OTG
- low-speed (1 Mbits/s)
- Isochronous endpoints
- Interrupt endpoints

## Software Licensing

USB2-SOFT is supplied under the following key licensing terms:

1. A nonexclusive, nontransferable license to use the VHDL source code internally, and
2. An unlimited, royalty-free, nonexclusive transferable license to make and use products incorporating the licensed materials, solely in bitstream format, on a worldwide basis.

The complete VHDL/IP Software License Agreement can be downloaded from <http://www.comblock.com/download/softwarelicense.pdf>

## Reference documents

- [1] Universal Serial Bus Specification, Revision 2.0, April 27,2000
- [2] UTMI+ Low Pin Interface (ULPI) Specification Revision 1.1 October 20, 2004
- [3] ComBlock COM-1600 FPGA + ARM + USB2.0+ DDR2 + NAND development platform [www.comblock.com/com1600.html](http://www.comblock.com/com1600.html)
- [4] COM-1600 development platform schematics
- [5] SMSC USB3300 Hi-Speed USB Host,Device or OTG PHY with ULPI Low Pin Interface
- [6] COM-1600 VHDL code template

## Configuration Management

The current software revision *rev* is 2. The software comprises:

[a] VHDL source code in directory  
USB20\_*rev*\src

[b] Xilinx .ucf constraint statements (to be copied to the project top level .ucf constraint file)  
USB20\_*rev*\src\USB20ULPI.ucf

[c] synthesized .ngc component:  
USB20\_*rev*\bin\USB20ULPI.ngc

where *rev* is the current revision number.

## VHDL development environment

The VHDL software was developed using the following development environment:

- (a) Xilinx ISE 13.4 with XST as synthesis tool and ISim simulator.
- (b) COM-1600 Spartan-6 FPGA development platform, including the USB3300 PHY

## Ready-to-use Hardware

The binary component (.ngc) is freely available for use on the following Comblock hardware modules:

- COM-1600 FPGA + ARM + DDR2 + NAND + USB2 development platform
- COM-1500 FPGA + ARM + DDR2 SODIMM development platform

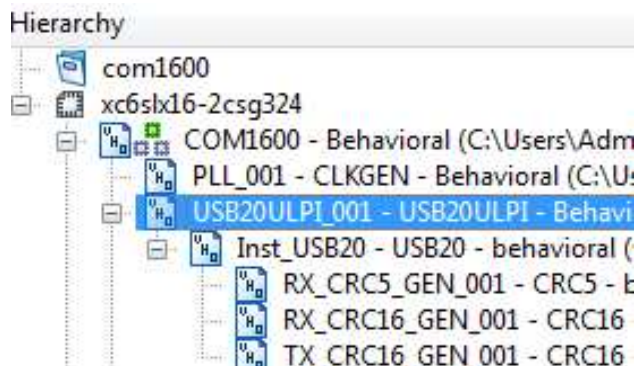
The schematics are available in this CD.

## Xilinx-specific code

The VHDL source code is written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- IOBUF
- IBUFG
- BUFG (global clocks)
- RAM block: RAMB16\_S9\_S9

## Top-Level VHDL hierarchy



The code is stored with one, and only one, component per file.

The root entity (highlighted above) is *USB20ULPI.vhd*. It is a wrapper that converts the natural PHY interface from UTMI to the lower-pin count ULPI. This ULPI wrapper can be removed if the FPGA interfaces with the external USB PHY over a UTMI interface.

The root also includes the following components:

- *USB20.vhd* is the Serial Interface Engine for a USB device. It can interface directly with a PHY over a UTMI interface.
- The *CRC5.vhd* verifies the CRC for incoming USB tokens.
- The *CRC16.vhd* computes the 16-bit CRC to be appended to tx packets and to check rx. The CRC computation is performed 8 data bits at a time.

## Test Environment

A testbench (*tusb20ulpi.vhd*) together with a simple PHY simulator (*USB\_PHY\_SIM.vhd*) are included. This allows to simulate the SIE and ULPI interface during high-speed negotiation and token exchange. All ULPI transactions are simulated: receive command, transmit command, data transmit, data receive, register read and register write.

To shorten the simulation, the constant *SIMULATION* in *usb20.vhd* should be set to '1'.

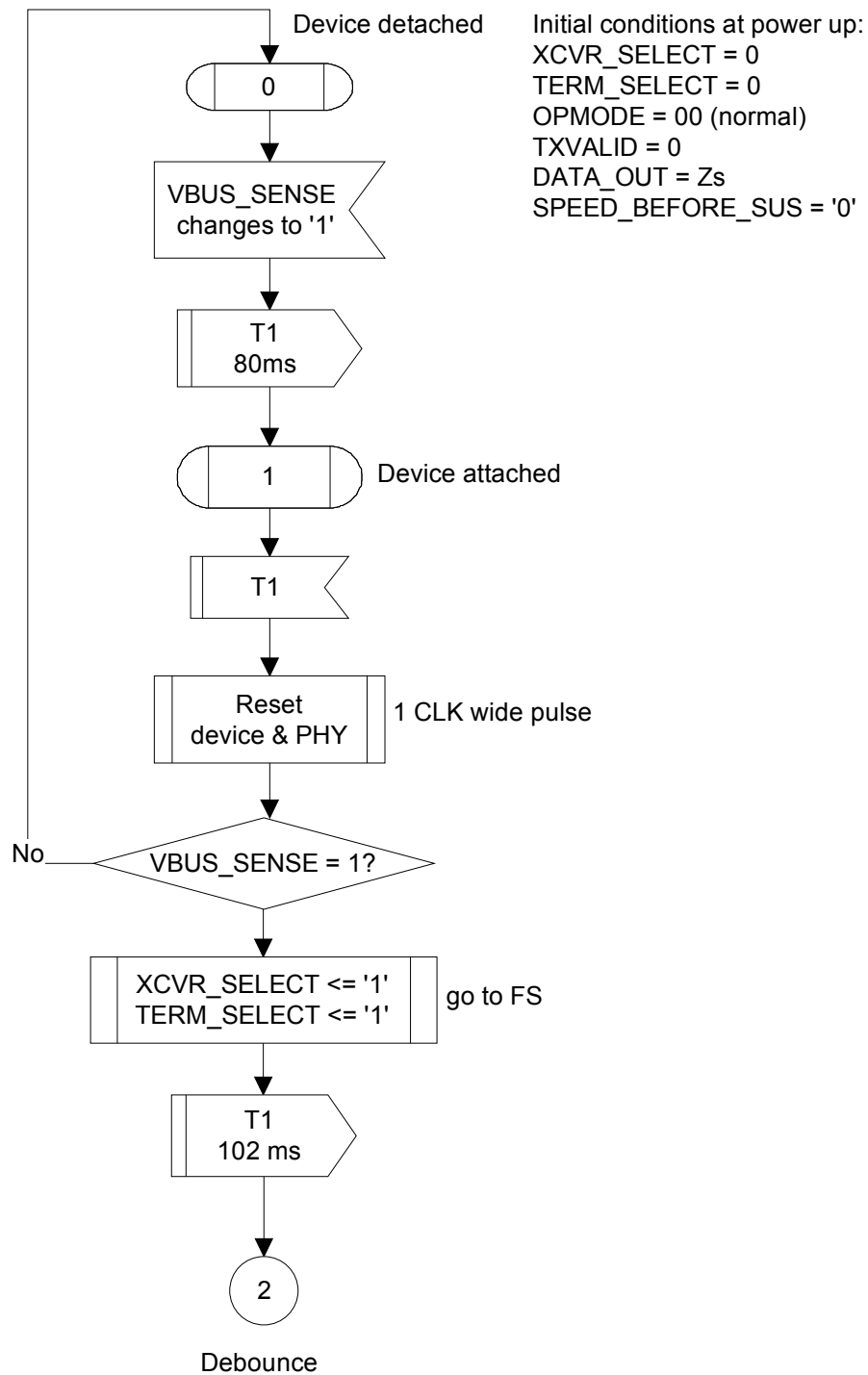
## ***Clock / Timing***

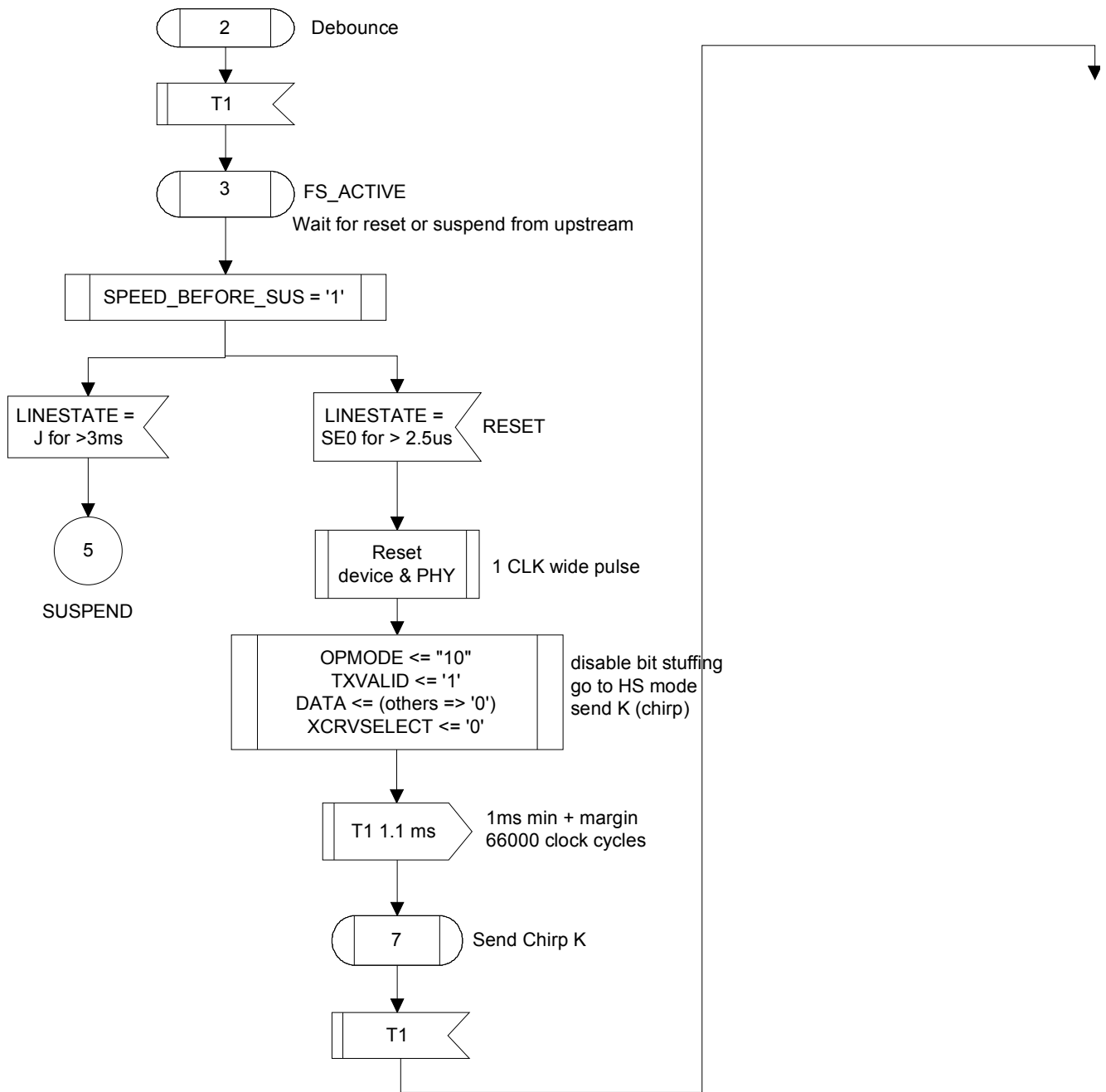
The software uses two main clocks:

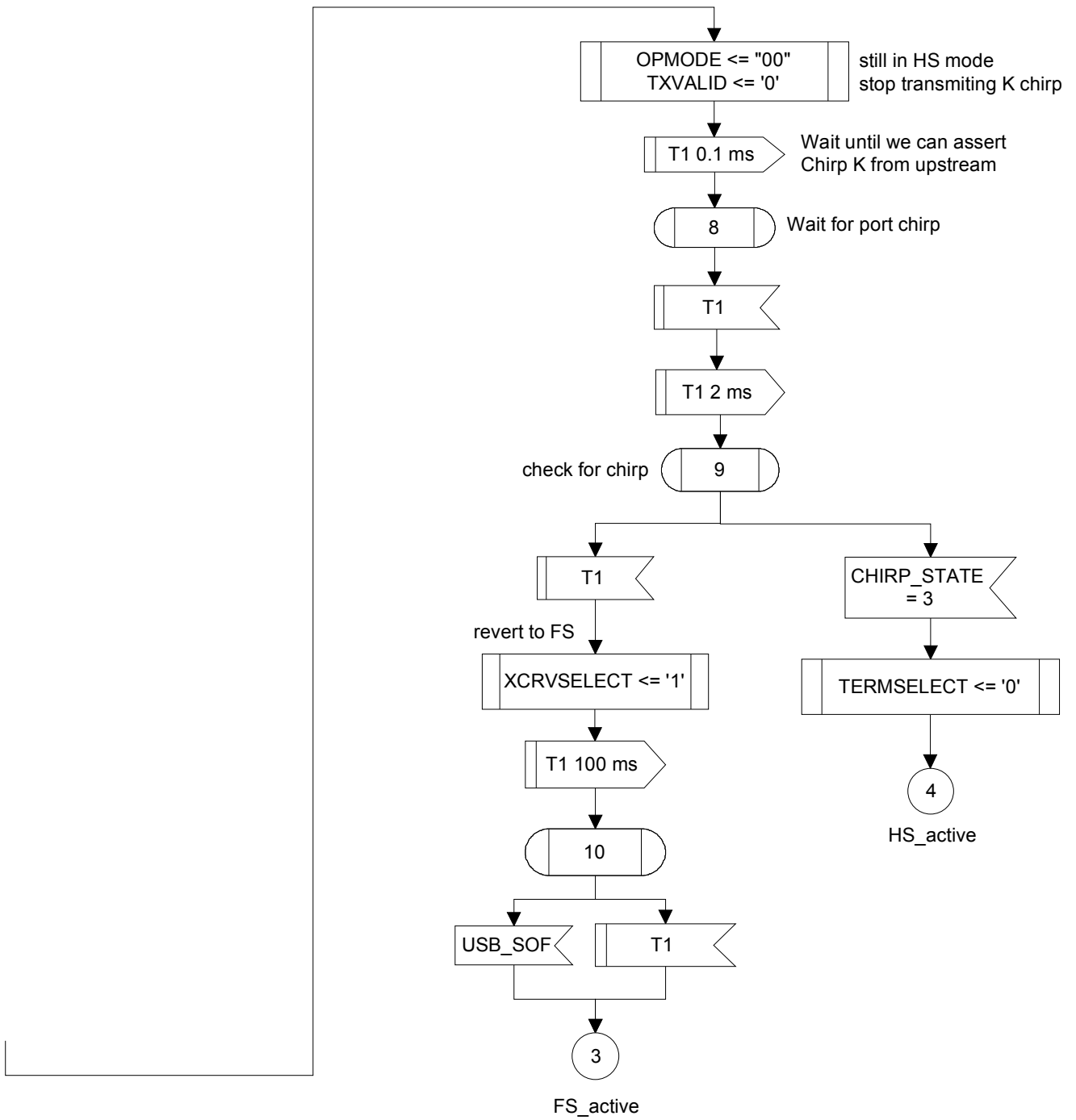
- a 60 MHz reference clock (USB\_CLK60G) generated by the PHY and 'cleaned' through a DCM or PLL.
- A user-selected processing clock (CLK\_P) to send and receive data from the user application.

# State Machine

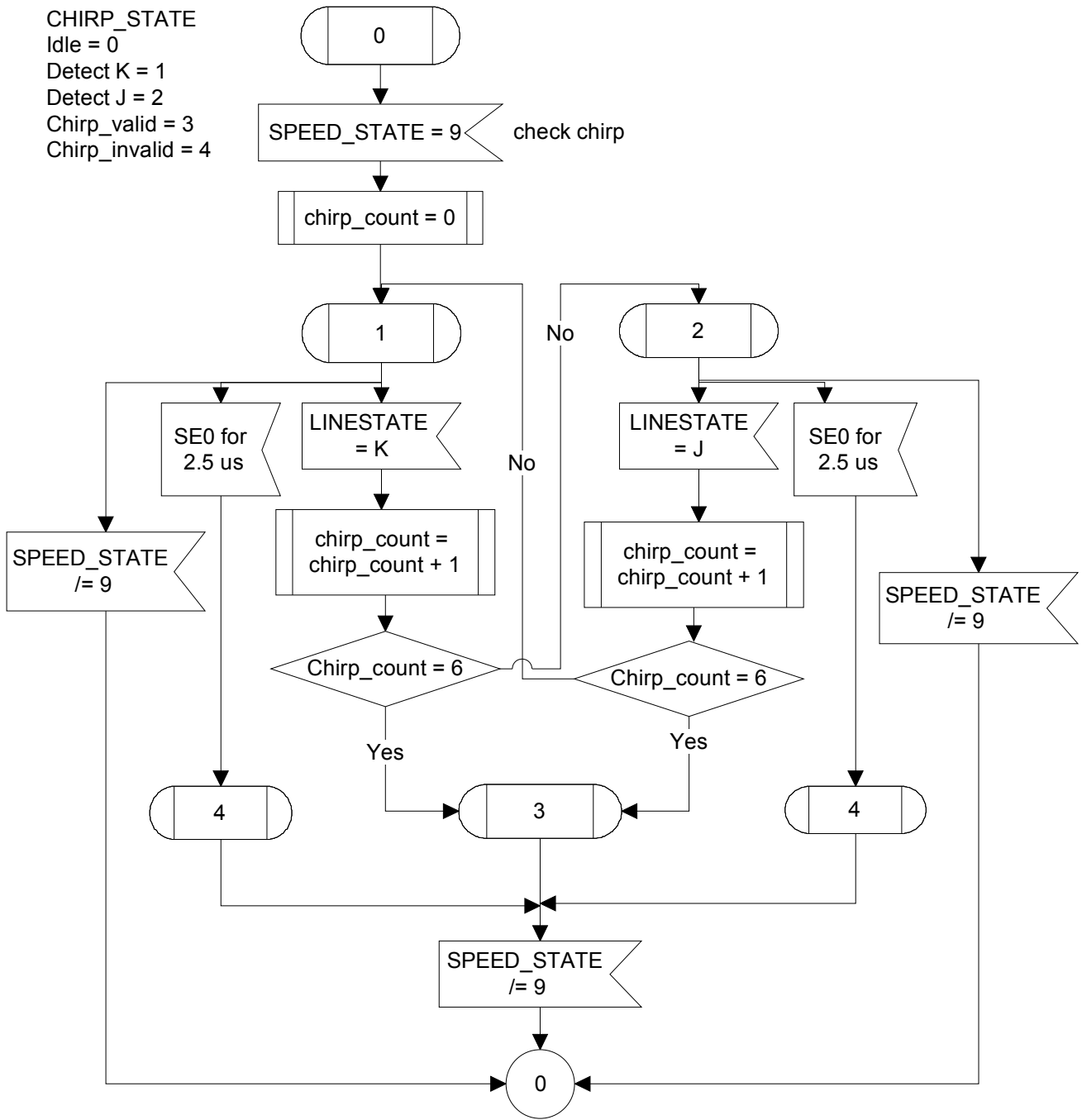
The state machine is described by the SDL flowcharts below:

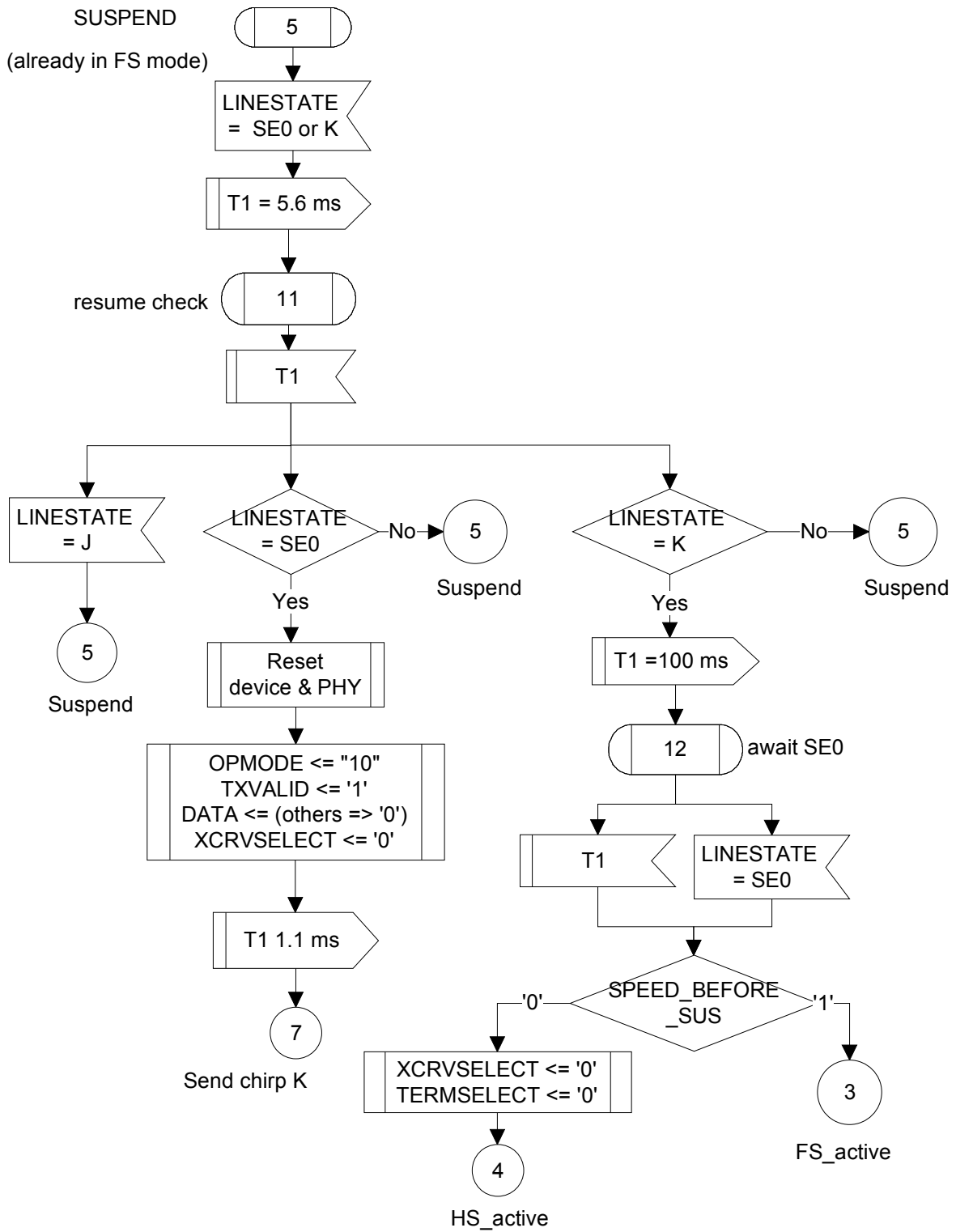


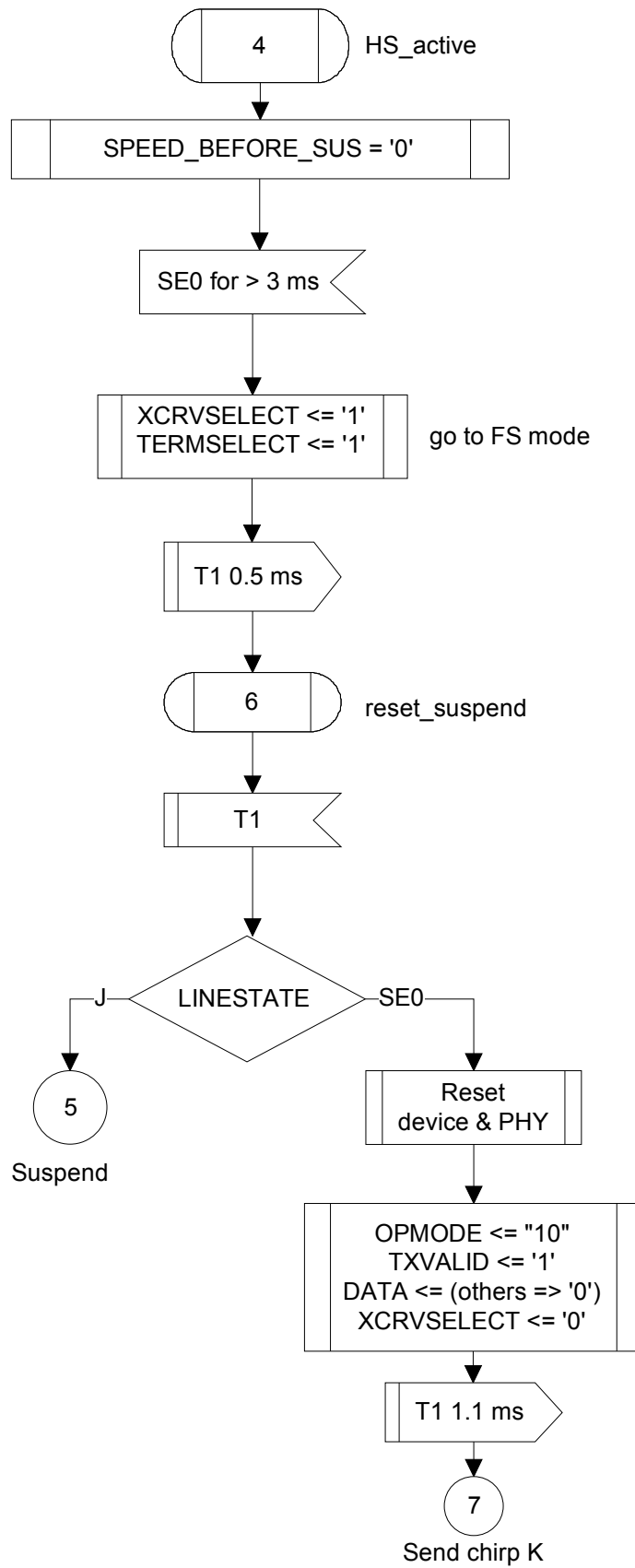




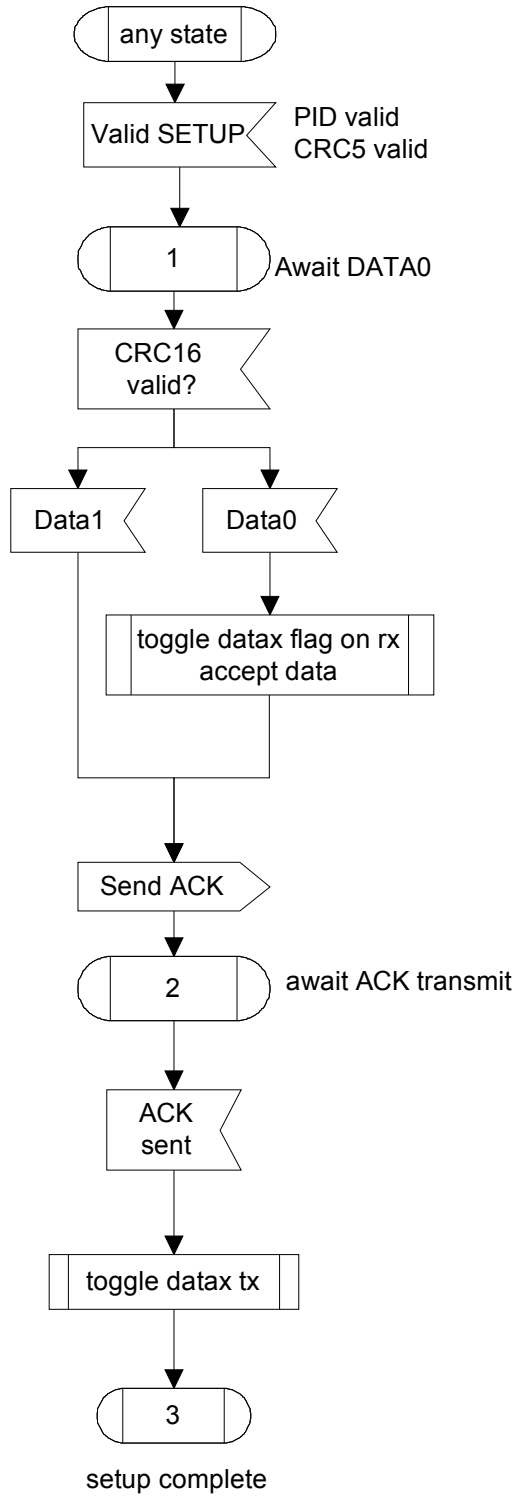
CHIRP\_STATE  
 Idle = 0  
 Detect K = 1  
 Detect J = 2  
 Chirp\_valid = 3  
 Chirp\_invalid = 4

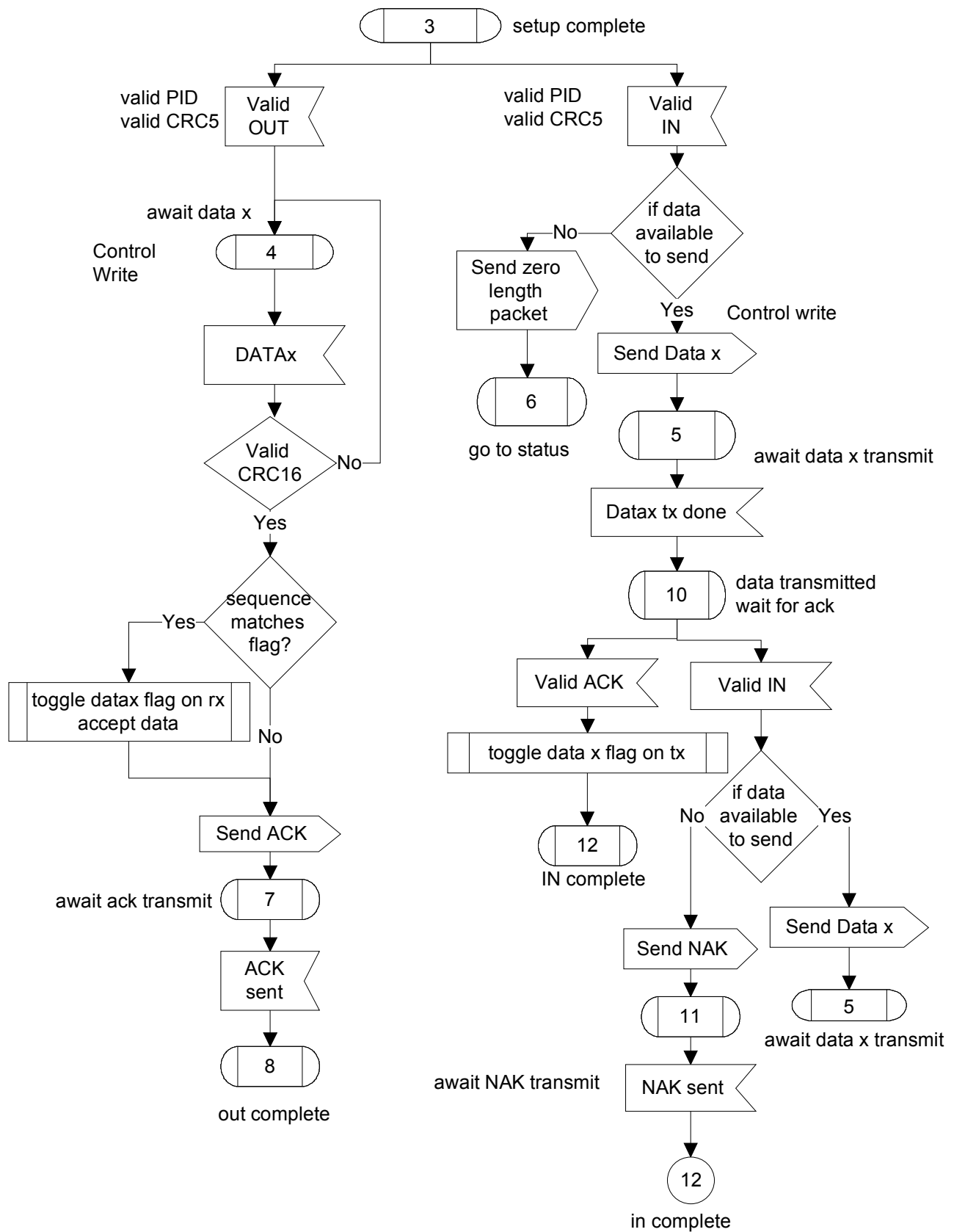


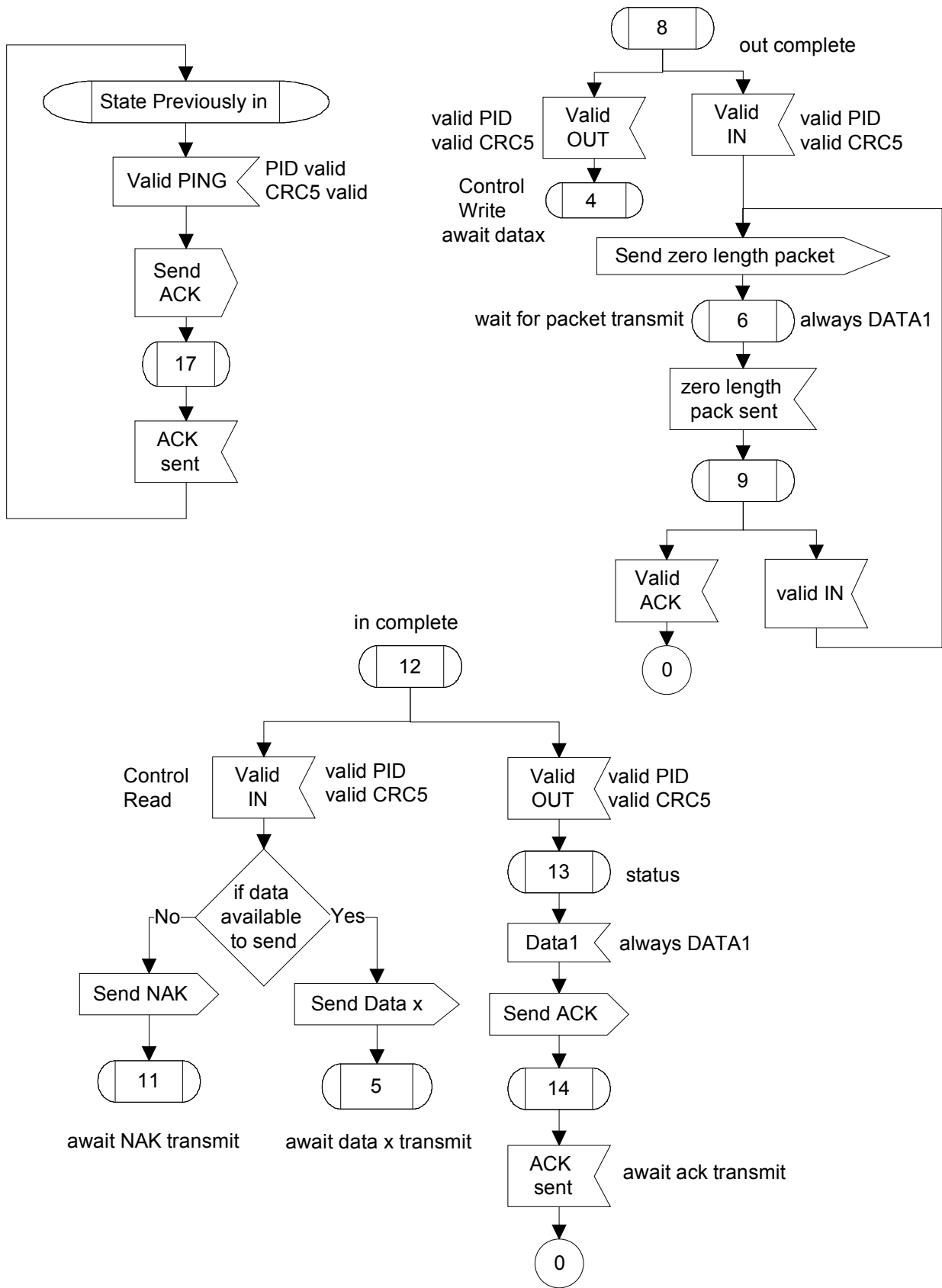


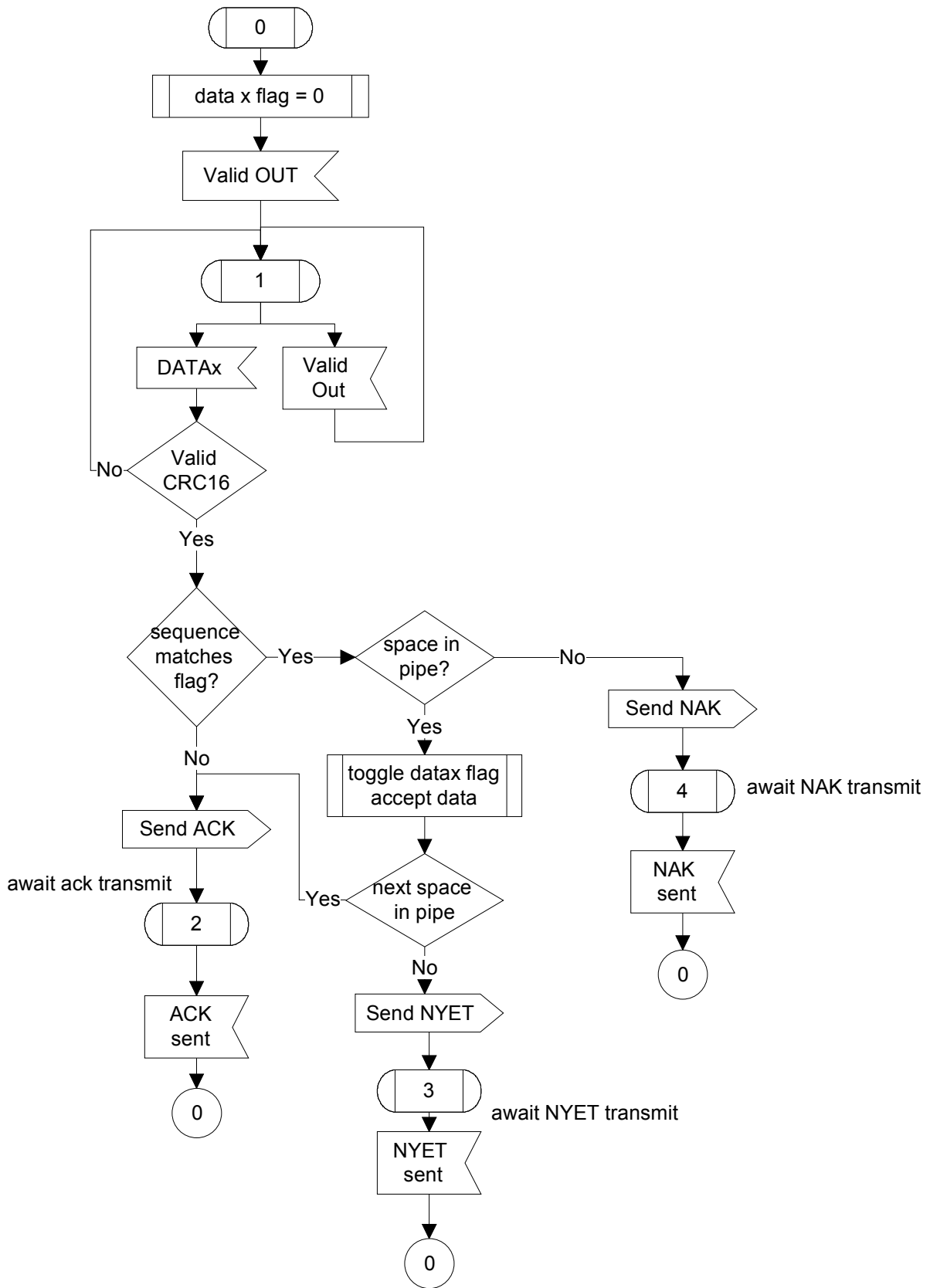


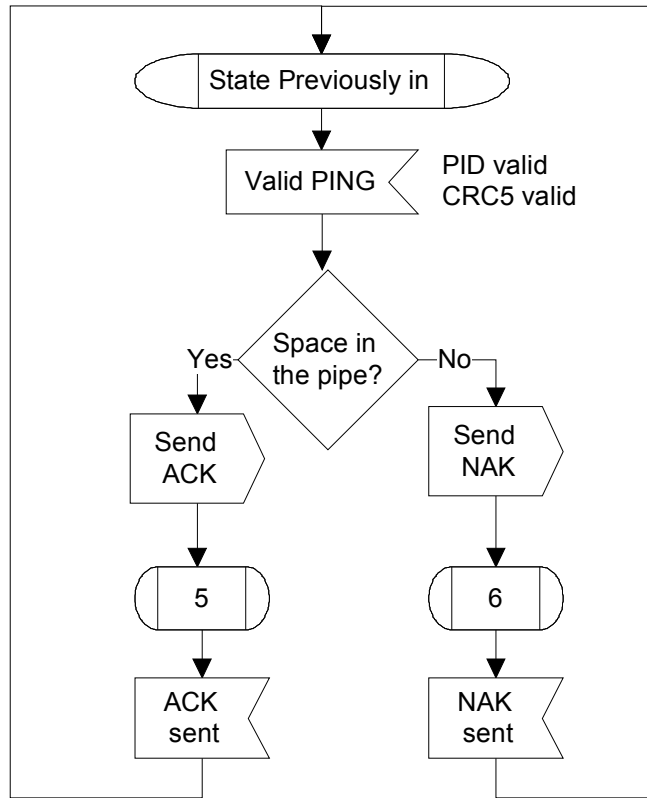
CONTROL PIPE      Initialize datax flags for tx and rx to 0

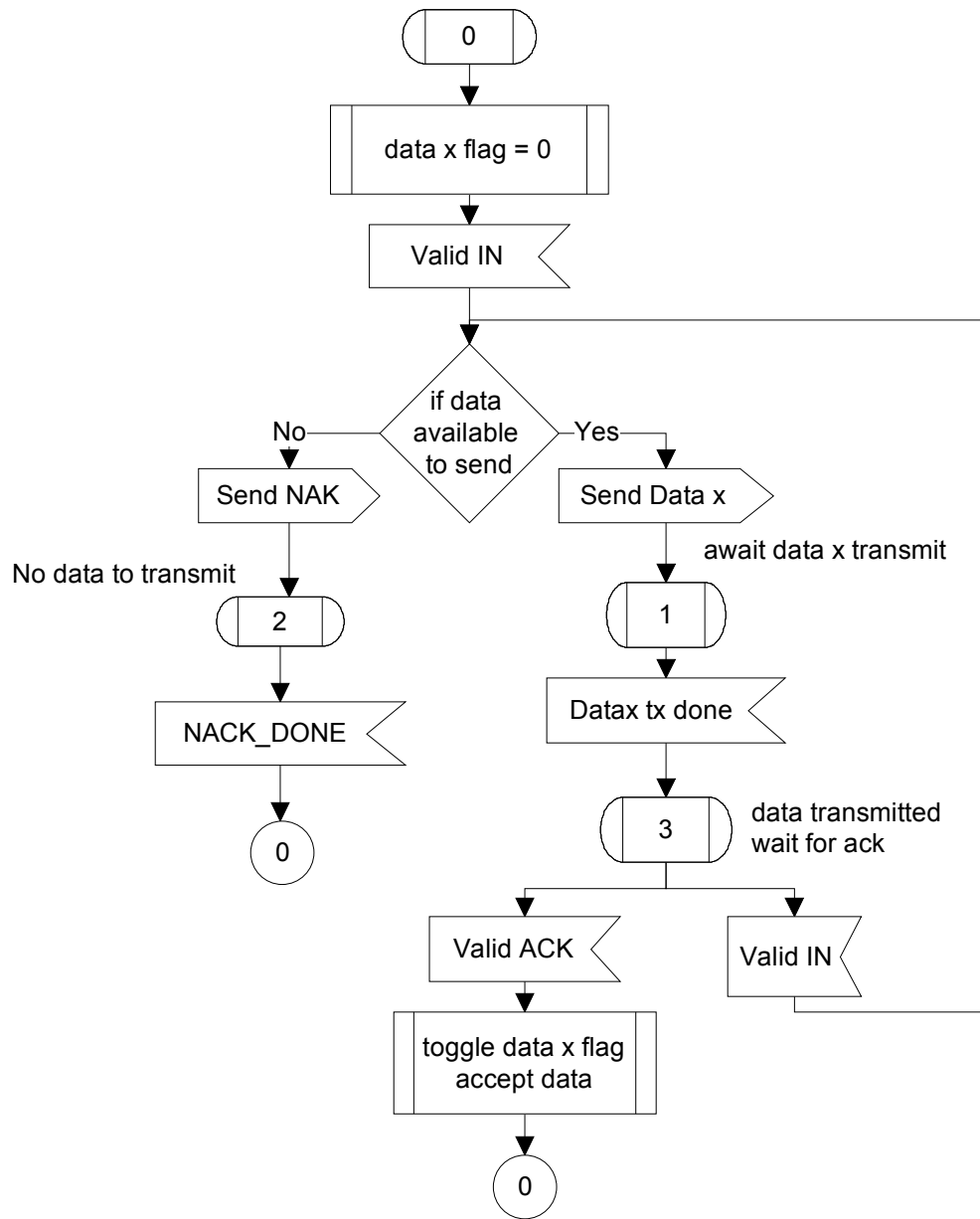












## ***ComBlock Compatibility List***

<b>FPGA development platform</b>
<a href="#">COM-1600</a> FPGA + ARM + DDR2 + USB2 + NAND development platform
<a href="#">COM-1500</a> FPGA + DDR2 SODIMM socket + ARM development platform

## ***ComBlock Ordering Information***

USB2-SOFT USB2.0 DEVICE SIE, VHDL SOURCE CODE

## ***Contact Information***

MSS • 18221-A Flower Hill Way •  
Gaithersburg, Maryland 20879 • U.S.A.  
Telephone: (240) 631-1111  
Facsimile: (240) 631-1676  
E-mail: [info@comblock.com](mailto:info@comblock.com)