MOST®

Media **O**riented **S**ystems **T**ransport

Multimedia and Control
Networking Technology

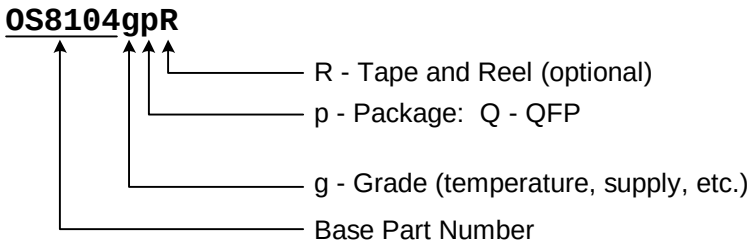
OS8104
MOST Network Transceiver
Final Product Data Sheet

DS8104FP4

Jan. 2003

OASIS SiliconSystems

Ordering Information



Valid Part Numbers:

Order Number	Grade		Package
	Temperature	Supply	
OS8104AQ	-40 to +85° C	4.5 to 5.5 V	44-pin TQFP
OS8104AQR	-40 to +85° C	4.5 to 5.5 V	44-pin TQFP, Tape and Reel

This table represents parts that were available at the time of printing and may not represent parts that are currently available. For the latest list of valid ordering numbers for this product, please contact your local sales office.

Support and Further Information

For more information on Oasis SiliconSystems products, including integrated circuits, software, and MOST development tools and modules, contact one of our offices below, or visit our web site:

www.oasis.com

Oasis SiliconSystems, Inc.
1120 Capital of Texas Hwy South
Building 2, Suite 100
Austin, Texas 78746 USA

Tel: (+1) 512 306-8450
Fax: (+1) 512 306-8442
OSS@oasis.com

Oasis SiliconSystems AG
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 6 25 37 - 0
Fax: (+49) (0) 721 6 25 37 - 119
OSS@oasis.de

Oasis SiliconSystems, Inc.
38600 Van Dyke Avenue
Suite 220
Sterling Heights, Michigan 48312 USA

Tel: (+1) 586 795-0545
Fax: (+1) 586 795-8950
Detroit@oasis.com

Oasis SiliconSystems AG Japan
Shin-Yokohama UU Bldg. 5F
2-5-2 Shin-Yokohama, Kohoku-ku
Yokohama 222-0033, Japan

Tel: (+81) 45-470 2240
Fax: (+81) 45-470 2242
Japan@oasis.com

Technical Support

For technical support please refer to one of the following e-mail addresses:

support@oasis.de
support@oasis.com

Intellectual Property

Trademarks

MOST is a registered trademark of Oasis SiliconSystems. All other trademarks used in this document are the property of their respective owners.

Patents

There are a number of patents and patents pending on the MOST technology. The rights to these patents are not granted without any specific Agreement between the users and the patent owners.

Conventions

Within this manual, the following abbreviations and symbols are used to improve readability.

Example	Description
BIT	Name of a single bit within a register
REG.BIT	Name of a single bit (BIT) in register REG
x..y	Range from x to y, inclusive
BITS[m:n]	Groups of bits (or pins) from m to n, inclusive
PIN	Pin Name
0xzzz	Hexadecimal number (value zzz)
zzh	Hexadecimal number (value zz)
rsvd	Reserved memory location. Must write 0, read value indeterminate
code	Instruciton code
Multi Word Name	Used for multiple words that are considered a single unit, such as: *Resource Allocate* message, or *Connection Label*, or *Decrement Stack Pointer* instruction.
Section Name	Section or Document name.
$\overline{\text{VAL}}$	Over-bar indicates active low pin or register bit
bREG	Single byte MOST register
mBUF	Multi-byte MOST buffer
x	Don't care

Revision History

Revision	Date	Description
1.1		Initial OS8104 Data Sheet
PP2	Sept. 2000	Fully revised.
FP1	Dec. 2000	Final Product Data Sheet
FP2	Dec. 2000	Final Product Data Sheet, revision 2
FP3	Jan. 2003	Final Product Data Sheet, revision 3
FP4	Jan. 2003	Final Product Data Sheet, revision 4

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The OS8104 is a complete MOST (Media Oriented Systems Transport) Network transceiver device capable of more than 24 Mbit/s data throughput when interfacing to a MOST Network. The architecture of the OS8104 is based on a RISC microcontroller to achieve the maximum performance and flexibility at the lowest possible price. All relevant Network management functions are handled on-chip, providing a complete Data-Link layer interface to the Physical layer components. Minimal additional components are required due to the high-level of integration on chip. An ultra-low jitter on-chip PLL guarantees high-quality audio and video transmission, and clock recovery over a wide frequency range.

A typical MOST Network node would consist of the OS8104 connected to a micro-controller that manages the higher-level Network functions. The MOST Network transceiver handles the Data-Link layer protocols and the micro-controller handles the Network layer and higher. Oasis SiliconSystems also offers a NetServices software stack, written in ANSI C, to ease implementation of the MOST Specification software protocol; thereby, drastically shrinking development time. NetServices can be ported to the target micro-controller, along with the application's code for the particular node. With the exception of some simple interface routines (Low-Level Driver), all Network management functions can be off-loaded from the programmer allowing full concentration on the application being developed. Figure 1-1 illustrates the OS8104 connected to a micro-controller, with the matching software protocol stack implementation. For simplicity, this example illustrates the OS8104 streaming source data between the MOST Network and local ADCs and DACs; however, the data could also be sent to DSPs or high-speed controllers using either a serial or parallel interface.

To minimize costs, the MOST Network supports a peer-to-peer topology, eliminating excessive hardware overhead such as a hub (although hub-based architectures can also be implemented). In addition to handling Network interface and communication management functions, the OS8104 also handles all of the important low-level Network management functions; such as node position sensing (Plug-and-Play), Network delay detection, start up and shut down, as well as error reporting, fail-safe operation, and channel allocation. The MOST Network consists of three simultaneous networks operating across a single low-cost plastic fiber, or a wired physical layer:

- a Control network to manage the Network and communicate control data across the Network,
- a Packet-based network to support nodes that communicate via packet data, and
- a Streaming network that supports high-speed synchronous data with extremely low overhead (such as audio and video).

Each of these three networks, or *data transfer methods*, operates independently (not affecting the other two), providing a robust, dependable, and deterministic system architecture. The OS8104 separates these three data transfer methods and directs them to the appropriate interface; as well as stores Control and Packet messages until the external system can retrieve them.

The OS8104 also supports a Stand-Alone mode in which a complete Network node consists of the OS8104 without any other local intelligence (no local micro-controller needed). In this mode, the OS8104 is controlled remotely through the Network, supporting the absolute lowest cost implementation.

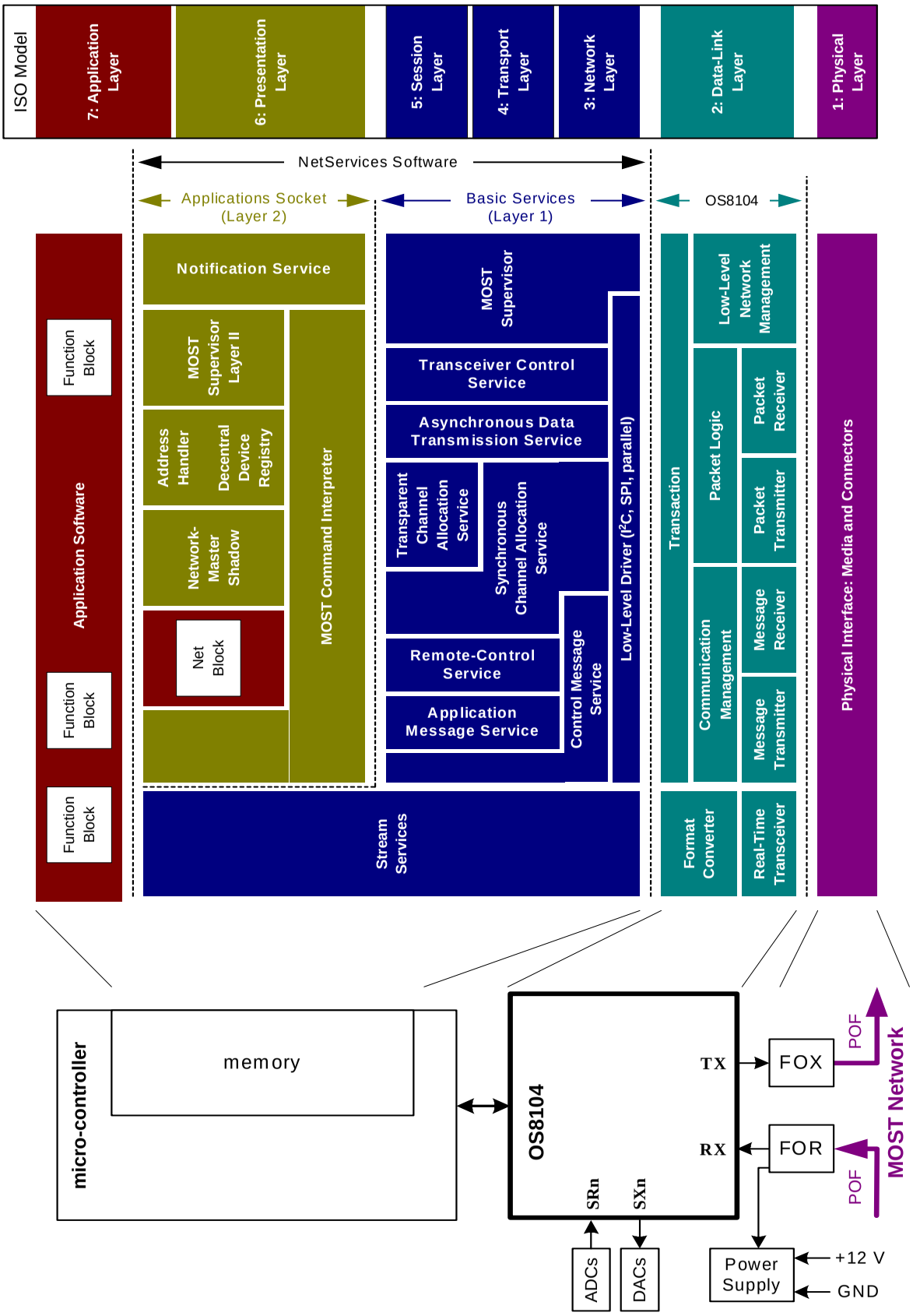


Figure 1-1: Typical MOST Hardware/Software System Overview

2 Functional Description

The OS8104 chip contains a RISC microcontroller-based MOST Routing Engine and several peripherals, including a clock manager, Source Ports, power management logic and an I^2C/SPI/parallel Control Port.

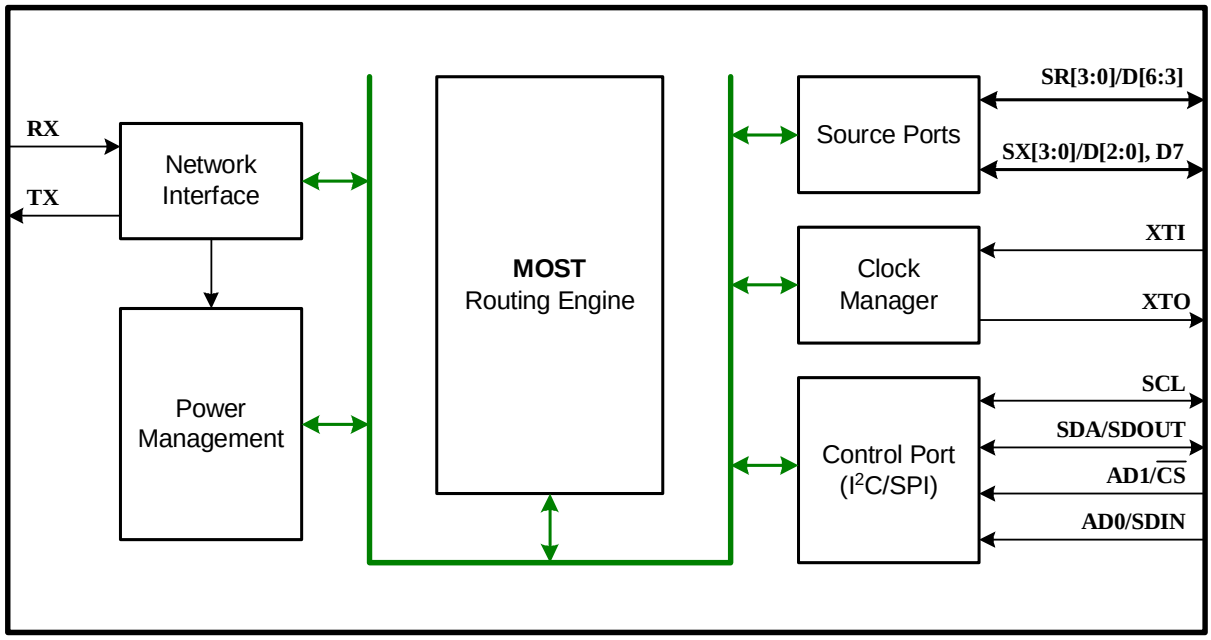


Figure 2-1: MOST Routing Engine

The Network interface includes an ultra-low jitter Phase Lock Loop (PLL) and a channel demodulator on the input side (**RX**). The Network output (**TX**) provides a channel-coded signal. Those signals can be connected directly to optical receiving/transmitting devices (FOR/FOX units) or balanced line drivers. The line quality can be monitored using the built-in error detection circuitry.

Each device can generate the Network clock or synchronize to it; however, only one device can generate the clock and the frame structure for a particular Network. Various A/D converters, D/A converters, digital signal processors, and Media Players (e.g., CD-audio/video/ROM) can be synchronized through one of the many clock and serial interface modes.

The Source Ports provide a synchronous data interface to the Network and are generally used as an interface to multimedia devices. Data sources and sinks can be connected to the Source Ports to transfer data to another node on the Network, or receive data from another node on the Network. The Source Ports support as many as four simultaneous inputs and four simultaneous outputs for transferring real-time data between external peripherals and the MOST Network.

2.1 Network Interface & Compatibility

The Source Ports support many different clock modes, providing an easy interface to single- and multiple-speed multimedia devices (i.e. I^2S, Sony, or Matsushita formats). In addition, one port can be configured for an S/PDIF (IEC 60958, parts 1 and 3) input and another as an S/PDIF output. The S/PDIF interface can operate as high as eight times the normal CD data rate; therefore, the maximum S/PDIF data rate at a Network frame rate of 48 kHz is:

$$(24 \text{ bit} \times 2) \times 8 \times Fs = 18{,}432 \text{ kbit/s} = 2.304 \times 10^6 \text{ byte/s (approx. 2.3 Mbytes/s)}$$

Real-time routing of source data through the Network is managed automatically by the on-chip routing algorithm. This algorithm handles the channel allocation and supplies *Connection Labels* for the requested channels (as a result of an allocation request). Unused channels are flagged and can be de-allocated in the event of unplugging or de-installation of a device. Real-time (stream) data is available to all nodes on the Network (broadcast) and can be retrieved by one or multiple sinks.

Each node has a Network address (logical) and a node position address (physical) for control purposes and system level Network management functions such as initialization, fault reporting, or delay detection. Control messages can be sent via single, group, or broadcast addresses.

The Control Port can be configured for serial access in I^2C or SPI formats. The highest data throughput for the OS8104 is achieved by configuring the part for parallel mode, which supports packet and/or real-time data access. Transferring data in parallel mode is ideal for burst type data and supports buffering of as many as eight bytes.

Remote access to control functions through the on-chip Control Port (configured as an I^2C-master interface) supports stand-alone operation and diagnostics. This allows entire nodes to be configured, controlled, and monitored remotely (from other Network nodes). External devices connected to the OS8104 can also be remotely controlled (via the Control Port) in this manner.

2.2 On-Chip Network Management

The core Network management functions in a MOST Network are handled on a distributed basis, and are embedded directly in the OS8104 MOST transceiver. Channel allocation, physical addressing, fault monitoring and power-down/wake-up are provided on-chip, providing a simple Network implementation and a high level of Network protection and reliability. Remote access allows for Network management functions, such as Network diagnostics, to be handled centrally or in a decentralized manner within each node, depending on the higher layer software structure.

Physical position sensing, MOST Network delay detection, and node alive supervision are some of the essential mechanisms provided on-chip. Part of the initialization of the Network is the node position sensing mechanism, which provides a unique physical address for every chip in the Network. This initialization procedure is part of the Network management and is done on-chip.

A device can also be configured and controlled with a logical address (usable as a DeviceID, per the *MOST Specification*) within the Network. The application can write the target logical address into the chip and the on-chip Network management verifies its unique existence. If the address is already used by another device, the address will be rejected and the application will be notified. All data requirements and channel connections are established during initialization, as well as dynamically in the background during normal operation.

An OS8104 device in a Network node can be either active or passive. During active operation, the request for data capacity comes from the application using a particular communication mode. As soon as a connection has been made, the required data capacity is provided by the Network.

The application does not have to deal with resource management within the Network, as long as no resource conflict is reported. Resource conflicts will be detected as soon as the maximum Network capacity of more than 24 Mbits/s is not sufficient to serve the applications running on the system. In this case, the unavailability of additional data capacity will be flagged. For the majority of applications, the large data capacity of the Network prevents the system from running into capacity conflicts, making high-level software administration and segmented transmission unnecessary.

The source data allocation algorithm is managed on the bit stream level. This results in fast response times (milliseconds) and high data efficiency at the same time. Once a synchronous connection is built, it will stay in place while the application needs the channels. Once the application is finished using the channels, it may de-allocate the channels to allow others to use them. Burst data channels are managed at the frame level, since their loading might change very quickly, and the latency time for that data transfer should be low.

If activities at several nodes must be synchronized and in phase (for example, different speaker connections) the inherent Network delay can be compensated by using the Network delay-reporting mechanism. Through this mechanism, the relative Network delay, with respect to each node, is available to every node.

All other bits within the Frame are for management purposes on the Network level. For example, the Preamble provides synchronization and clock regeneration; and the Parity bit indicates reliable data content and is used for error detection and Phase Lock Loop operation.

2.2.1 Channel Allocation

Within each MOST frame, 60 byte-wide physical channels are available for transporting source data (any data coming from or going to a Source Port). As many as eight physical channels (64 bits) can be allocated with a single allocation request, thus forming a logical channel.

For example, a stereo CD-audio channel requires four of the byte-wide physical channels, running at a frame rate of 44.1 kHz. Thus, as many as 15 stereo CD-audio channels can be supported simultaneously using the Network's 60 available byte-wide physical channels. Logical channels can be clustered into multi-channel streams in higher software layers.

Source data routing becomes an easy task, since all necessary functionality is supported by the bit stream structure and managed on-chip. Channel allocation (also referred to as resource allocation) is done by a Network-level channel-allocation algorithm, embedded in the OS8104. Each node contains a Channel Resource Allocation table (mCRA) containing labels associated with each logical channel (*Connection Labels*) available at any point in time. Since synchronous data channels (bytes) are quasi static, two functions for allocation and de-allocation are available. Nodes that want to place source data onto the Network can identify the existence of free channels and request allocation from the Network. Nodes that want to sink data can identify the correct channels by the Connection Label without needing to send extra messages to the source nodes. The channel allocation can be changed during runtime.

2.2.2 Physical Position Sensing

The node position relative to the timing-master (frame generator) node is available in each node, once the Network is running. The node position determines the node's unique location (and physical address) in the Network. The timing-master senses the number of nodes in the Network and provides this information to the application. Using the node position for configuration sensing enables “hot” plugging, supporting dynamic system reconfiguration.

2.2.3 Network Delay Detection

The delay in a Network, relative to the timing-master, is not necessarily directly related to the node position, since nodes can be active or passive. Only active nodes (Source Data Bypass inactive) add delay to source data. The Network delay detection provides an accurate number of frame delays and supports delay compensation. Similar to the Network position, every node also knows the maximum delay in the Network. Network delay compensation can be useful for noise canceling, speech recognition, and multi-channel sound applications.

2.2.4 Node Alive Supervision

The node-alive supervision mechanism is used for Network management, such as channel allocation, error management and power management. To avoid channel blocking, due to unplugged or defective devices, devices that are detected as not alive or sleeping can be taken out (de-allocated) of the resource request list. By this means, the corresponding channels will become available to other nodes.

2.3 On-Chip Power Management

Two different power saving modes are provided by the OS8104 transceiver chip. The Zero-Power mode is available when not using the fiber optic receiver (FOR) to manage system power, as described in the *MOST Specification* and illustrated in Chapter 20. Per the *MOST Specification*, the FOR powers down the entire node when no light is detected on the **RX** pin. When not using an FOR with power management capability (or not using FORs at all), the OS8104's Zero-Power mode can minimize node power (albeit not as low as when using the FOR to manage power).

The Zero-Power mode can be initiated via the Network by means of a remote Control message, or a Control Port access by the local application. During Zero-Power operation, the Network is continuously checked for activity. As soon as activity is detected, the OS8104 will power up.

When a particular node is not being used, Low-Power mode can place the device in the lowest power state possible while still keeping the Network running.

When in Low-Power mode, the MOST core enters a state where all functions except the transceiver are stopped. Normal activity is resumed after receipt of a wake-up signal. Setting the low-power wake-up bit of the timing-master node will cause a wake-up signal to be generated and sent around the Network.

2.4 Data Transfer Methods

The OS8104 supports three methods for transferring data:

- Synchronous, or Stream Data Transfer
- Control Message Data Transfer
- Asynchronous, or Packet Data Transfer

The synchronous data transfer method uses a circuit-switched approach for very low overhead support of streaming real-time data, such as audio or video. A typical application for synchronous data transfer is sending streaming audio from a CD transport unit to an amplifier that converts the audio data to analog and plays the data out of speakers. Once the logical connection is setup via Control messages, no overhead is wasted while streaming the data across the Network, thus providing the maximum efficiency for real-time data.

The Control message data transfer method operates simultaneously with the other transfer methods and supports control, status queries, and notification status among all nodes in the Network. This transport mechanism is supported by every node and is the backbone of the MOST Protocol. Messages can be sent to all devices (broadcast), to just a group of devices (groupcast), or to a specific node (using logical or physical addressing). A large set of standard functions and function classes are pre-defined to support peer-to-peer inter-operability. Control messages can turn on functions in other nodes (such as starting a CD player), as well as to get status from a node (such as the current track and time from the CD player).

The packet data transfer method uses a channel shared by all nodes to transfer asynchronous burst-type data. This method arbitrates for the channel and supports sending messages to individual nodes. The portion of the Network bandwidth allocated to asynchronous vs. synchronous data is user-selectable and can be optimized for each system. Examples of asynchronous packet data include internet data, GPS map data, or still video images that are only occasionally sent across the Network. The packet data transfer method is similar to Ethernet-style communication.

Once the Network is configured, the synchronous, asynchronous, and control data streams run concurrently, providing a very predictable, low-latency, high-performance network. High data loading on one transport method does not affect the other transport methods. Supporting these three methods simultaneously allows different types of data to operate on the same network, providing higher bandwidth utilization than with single-method networks.

2.5 Bandwidth

Bandwidth is divided into two categories: MOST Network bandwidth and OS8104 interface bandwidth. Each category applies to each of the three transport methods—synchronous, asynchronous, and control.

2.5.1 Control Messaging

A Control message requires 16 MOST frames (one MOST block) to be transported across the MOST Network. Therefore, the message rate depends upon the Network frame or sample rate (Fs). In addition, some Control message bandwidth is used for Network administration. The actual throughput available to the application on the MOST Network is determined by:

$\frac{62 \times Fs}{1024}$ Control messages per second = 2906 msg/s, when Fs is 48 kHz

Each Control message contains 17 bytes of usable data which leads to a net data rate of:

$\frac{62 \times Fs}{1024}$ msg/s $\times$ 17 bytes/msg $\times$ 8 bits/bytes = $\frac{8432 \times Fs}{1024}$ bits/s = 395.3 kbits/s, when Fs is 48 kHz

The Control message bandwidth is shared among all Network nodes by a fair arbitration mechanism. The OS8104 is capable of arbitrating for a new transmit message in the third control message after it finishes sending the last Control message. Therefore, assuming no other devices are arbitrating for the Control message channel, the maximum Control data rate for a single OS8104 node is:

$\frac{\left(\frac{8432 \times Fs}{1024}\right)}{3}$ bits/s = 131.8 kbits/s, when Fs is 48 kHz

The actual Control message throughput realized by a particular OS8104 device depends on how the Control Port interface is configured (SPI, I^2C, or through the parallel port).

2.5.2 Synchronous (Stream) Data

The synchronous bandwidth available in the MOST system at any particular time depends on the bSBC (synchronous bandwidth control) register value and the Network frame or sample frequency (Fs). bSBC is set in the timing-master node and sent to all the timing-slave nodes in the Network. bSBC determines the division of Network source data between synchronous and asynchronous transfer methods (for an overview of the MOST Network architecture, see Chapter 6 on page 35). The bSBC value is specified in quadlets (four bytes equals one quadlet), where the minimum is 6 (6 quadlets reserved for synchronous data, 9 quadlets for asynchronous data), and the maximum is 15 (15 quadlets for synchronous data and no asynchronous data). The MOST Network available synchronous bandwidth is:

bSBC quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s

Therefore, the minimum MOST Network synchronous bandwidth is:

6 quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s, or
192 $\times$ Fs bits/s = 9.216 Mbits/s, when Fs is 48 kHz

The maximum MOST Network synchronous bandwidth is:

15 quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s, or
480 $\times$ Fs bits/s = 23.04 Mbits/s, when Fs is 48 kHz

Since the bSBC unit is one quadlet, the step size for synchronous bandwidth is 1.54 Mbits/s. Synchronous bandwidth is allocated similar to a circuit-switched topology; therefore, no real-time bandwidth is used for addressing. Control messages set up the virtual circuit between the source and one or more destinations. Individual nodes can request (Allocation request) synchronous bandwidth in steps of one byte (one physical channel).

The volume of synchronous data that the OS8104 can transfer is dependent on the configuration of the Source Ports. The Source Ports can be configured in three modes. In serial mode, synchronous data is transferred serially into the chip on the **SR[3:0]** pins and out of the chip on the **SX[3:0]** pins. The format chosen for those pins also determines the maximum data rate. When the OS8104 is configured for parallel operation, in either Parallel-Synchronous or Parallel-Combined (Physical) modes, all the source data in an entire MOST frame can be transferred through the parallel port. When the OS8104 is configured for parallel operation in the Parallel-Asynchronous mode, no synchronous data is transferred.

When the Source Ports are configured for one of the standard serial formats (I^2S, S/PDIF, etc.), the maximum synchronous data rate, in each direction, is:

$4 \times 64 \times Fs$ bits/s = 12.288 Mbits/s, when Fs is 48 kHz

Using a special S/PDIF mode, this synchronous data rate can be doubled, although only one direction is supported (in or out):

$8 \times 64 \times Fs$ bits/s = 24.576 Mbits/s, when Fs is 48 kHz

which supports the maximum MOST Network synchronous bandwidth of 23 Mbits/s at Fs = 48 kHz.

When the Source Ports are configured in Parallel-Synchronous or Parallel-Combined (Physical) mode, the OS8104 parallel port can handle the maximum MOST Network synchronous bandwidth in each direction (along with one status quadlet in Parallel-Combined mode), supporting a bandwidth of:

15 quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s, for Parallel-Synchronous, or
$480 \times Fs$ bits/s = 23.04 Mbits/s, when Fs is 48 kHz

and

16 quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s, for Parallel-Combined, or
$512 \times Fs$ bits/s = 24.576 Mbits/s, when Fs is 48 kHz

2.5.3 Asynchronous (Packet) Data

As with the synchronous bandwidth, the asynchronous bandwidth available on the MOST Network at any particular time depends on the bSBC (synchronous bandwidth control) register value and the Network frame or sample frequency (Fs). bSBC determines the division of Network source data between synchronous and asynchronous transfer methods. The bSBC value is specified in quadlets (four bytes equals one quadlet), where the minimum is 6 (6 quadlets reserved for synchronous data, 9 quadlets for asynchronous data), and the maximum is 15 (15 quadlets for synchronous data and no asynchronous data). The MOST Network available asynchronous bandwidth is:

(15 – bSBC) quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s

Therefore the maximum MOST Network asynchronous bandwidth is:

9 quadlets $\times$ 4 bytes/quadlet $\times$ 8 bits/byte $\times$ Fs 1/s, or
$288 \times Fs$ bits/s = 13.8 Mbits/s, when Fs is 48 kHz

The minimum MOST Network asynchronous bandwidth is no asynchronous bandwidth, where all the source data quadlets are allocated to synchronous data. The maximum packet data length when the Source Ports are in Parallel-Combined mode is 1014 bytes. In all other modes, the maximum packet length is limited to the on-chip buffer size of 48 bytes.

Since asynchronous data transfer is done via packets, and all nodes share the same channel, packet headers (arbitration, target address, source address, CRC) lower the actual user data transfer rate by 10 bytes. Since packets can be variable lengths, the ratio of actual user data (net data) transfer to overall MOST Network asynchronous data transfer is not constant. However, the longer the packet, the higher the net data rate, since the packet header size is constant.

When a packet is ready to be sent out onto the MOST Network, the OS8104 must arbitrate for the asynchronous data channel. The chip can arbitrate for a new packet in the fifth frame after completion of a previous packet. When no other device is arbitrating for the channel, the OS8104 has a minimum delay of four frames between packets being sent out. The maximum net data rate that one device can achieve on the Network is defined as follows:

R	net data rate, bits/s
Pd	packet data length (net data), bytes
Ph	packet header, bytes
Tp	transmission time for packets, frames
Ta	delays between successive packets from one device, frames
Af	Asynchronous bytes per frame, bytes
Fs	MOST Network frame rate, sample frequency, 1/s
bSBC	synchronous bandwidth control, quadlets/frame

For the OS8104:

Ta = 4 frames
Pd maximum value defined by Source Port mode on the OS8104 (48 or 1014 bytes)
Ph = 10 bytes
Af = (15 – bSBC) × 4

Therefore, the needed frames to consider back-to-back packet transmission is:

$$Tp = \text{roundup}\left(\frac{Pd + Ph}{Af}\right) = \text{roundup}\left(\frac{Pd + Ph}{(15 - bSBC) \times 4}\right) = \text{roundup}\left(\frac{Pd + 10}{(15 - bSBC) \times 4}\right) \text{ frames}$$

The average net data rate = net data in a packet / (time for packet transmission + time between packets).

$$R = \frac{Pd \times 8}{\frac{Tp + Ta}{Fs}} = \frac{Pd \times 8 \times Fs}{Tp + Ta} = \frac{Pd \times 8 \times Fs}{\text{roundup}\left(\frac{Pd + 10}{(15 - bSBC) \times 4}\right) + Ta} \text{ bits/s}$$

The Ta value listed is the time required by the OS8104. If the external application needs more time to prepare and transfer the next packet to the chip, Ta will grow appropriately. Table 2-1 shows the net data rate for asynchronous transfer for different bSBC settings and different packet lengths (Pd). When the OS8104 Source Port is in Parallel-Combined mode, the maximum packet size is 1014 bytes. When the Source Ports are configured for any other mode, the maximum packet size is 48 bytes.

bSBC	Pd						Units
	1014	512	256	128	64	48	
6	11.799	10.348	8.192	6.144	3.511	3.072	Mbits/s
7	10.816	9.362	7.562	5.461	3.511	3.072	Mbits/s
8	9.497	8.548	7.022	5.461	3.511	2.633	Mbits/s
9	8.285	7.562	6.144	4.915	3.072	2.633	Mbits/s
10	6.953	6.342	5.461	4.468	3.072	2.633	Mbits/s
11	5.726	5.314	4.681	3.781	2.731	2.304	Mbits/s
12	4.326	4.096	3.641	3.072	2.234	2.048	Mbits/s
13	2.950	2.809	2.587	2.234	1.755	1.536	Mbits/s
14	1.498	1.456	1.385	1.260	1.069	0.970	Mbits/s
15	0	0	0	0	0	0	Mbits/s

Assumes Ta = 4 and a Network frame rate of Fs = 48 kHz

Table 2-1: Net Asynchronous Data Bandwidth (R)

2.6 MOST NetServices API

To accelerate development of applications using the OS8104, Oasis SiliconSystems offers the MOST NetServices API. The MOST NetServices API provides software access to the MOST Network. All services are available as a software library, including basic services like initialization, up to high-level communication tasks. MOST NetServices is modular and can be customized for the target system. The MOST NetServices API is implemented in ANSI C, which can be adapted to individual requirements through configuration files.

The MOST NetServices API is organized into two layers: Basic Services (Layer 1) and the Applications Socket (Layer 2). The Basic Services provides low-level services such as Network initialization, Control message management through the Control Port, Source Port configuration, synchronous channel allocation on the Network, and asynchronous data transmission services.

The Applications Socket (Layer 2) operates on top of Layer 1 and provides a command interpreter and the NetBlock function required on all Network devices. The command interpreter provides a simple API for developing new functions within a node. It also supports the *MOST Specification's* Notification Services and functional addressing.

With respect to the ISO communications model, the OS8104 chip supports the Data-Link layer. MOST NetServices Layer 1 supports the Network layer through the Session layer. MOST NetServices Layer 2 supports the Presentation and part of the Application layer. The ISO network model and MOST NetServices is depicted in Figure 2-1.

More information on MOST NetServices is available on the Oasis SiliconSystems web page:

http://www.oasis.com

3 Main Functional Blocks

The OS8104 has six main blocks that physically interface with the Network and the application:

- Configuration Interface - For general Source and Control Port mode selection at start-up.
- Network Interface - Providing control, status and connectivity to the MOST Network.
- Clock Manager - Synchronizes the entire OS8104 to various clock sources. It provides a scalable output of the system clock and an input for an external clock signal or a crystal.
- Source Ports - To source and sink either synchronous data (streams) in serial or parallel format, or asynchronous data (packets) in a parallel format.
- Control Port - This port exchanges control data (packets) in a serial or parallel format. It controls the chip locally by writing and reading internal registers. In addition, remote control of other chips via the Network is possible, as well as the requesting/allocating of system resources on the Network. This port provides the exchanging of control messages to other applications within the Network.
- Wake-up Logic & Power Management - Activates or deactivates power down and Low-Power mode controlled through Network activity or a dedicated command.

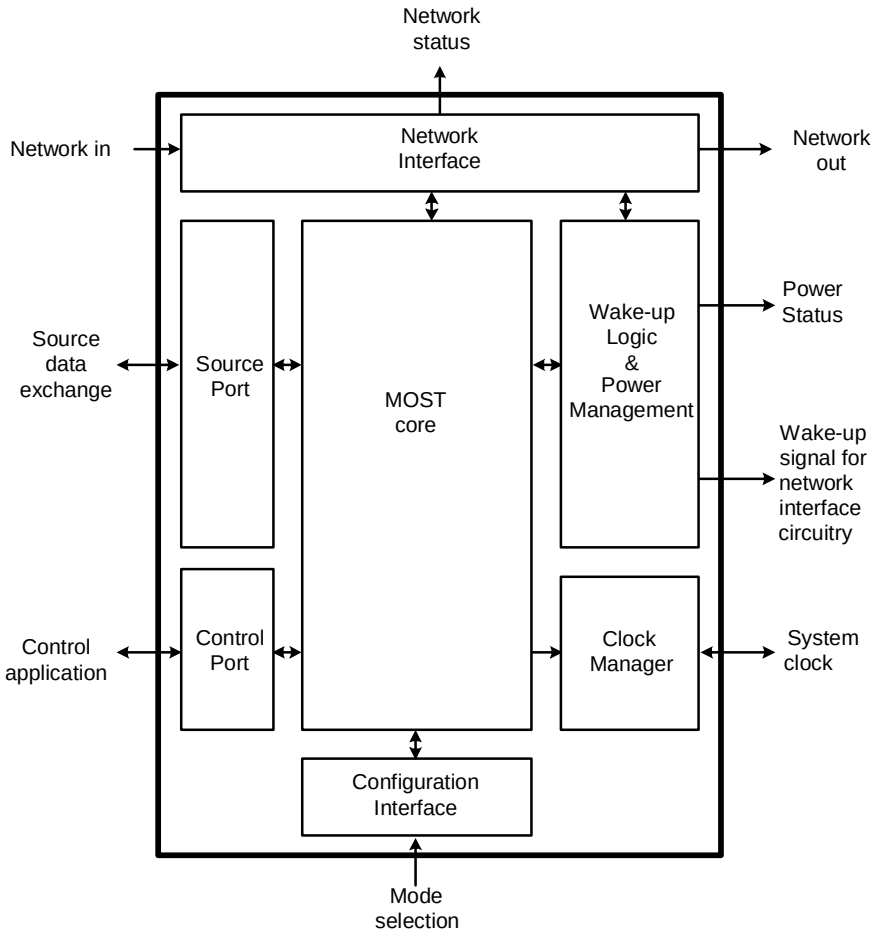


Figure 3-1: OS8104 Functional Blocks

4 Configuration

Before accessing the OS8104, the interface format must be configured. The Control Port provides access to the internal functions of the OS8104. The Source Port transports source data into and out of the chip. For both ports, either a serial or a parallel interface is available. Figure 4-1 illustrates the valid interface format combinations.

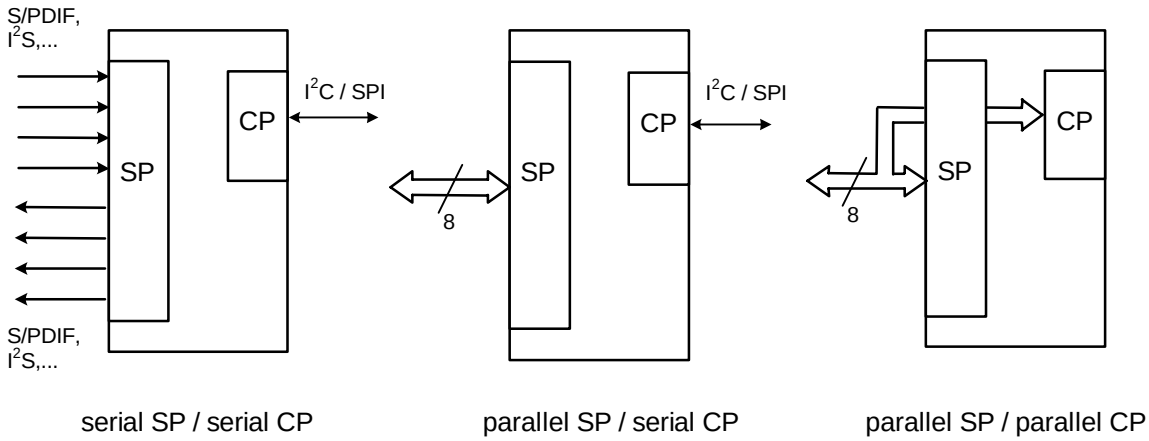


Figure 4-1: Source and Control Port Interface Options

The interface format for the Control Port and the Source Port is selected via the configuration interface, which consists of the following pins:

- **PAR_CP** (Parallel/serial mode configuration of Control Port)
- **PAR_SRC** (Parallel/serial mode configuration of Source Port)
- **ASYNC** (Source Port parallel mode: Parallel-Synchronous or Parallel-Asynchronous)
- **SCL** (At reset, Control Port serial mode: I^2C or SPI format)

In addition, $\overline{\textbf{RD}}$ and $\overline{\textbf{WR}}$ are used to configure the part in *Stand-Alone* mode.

PAR_CP, **PAR_SRC** and **ASYNC** must be valid before $\overline{\textbf{RS}}$ rises and cannot change during normal operation. The **SCL** pin selects the Control Port serial format on the rising edge of $\overline{\textbf{RS}}$. After initialization, **SCL** is the clock signal for the Control Port in serial mode.

The Control Port supports the following interfaces:

- Serial — **PAR_CP** tied low. The **SCL** pin selects the serial Control Port format.
  - A pull-up resistor on **SCL** selects the I^2C serial format. The pull-up on **SCL** is also needed since **SCL** is an open-drain bi-directional signal in the I^2C serial format.
  - A pull-down resistor on **SCL** (or **SCL** driven low) selects the SPI serial format.
- Parallel — **PAR_CP** tied high. The Source Port must also be in parallel mode (**PAR_SRC** high). Tie Control Port serial pins high when the Control Port is configured in parallel mode.

The Source Port uses the **SR[3:0]** and **SX[3:0]** pins in serial mode. In parallel mode, the **SRn** and **SXn** pins are configured for an 8-bit parallel port, **D[7:0]**. The Source Port supports the following interfaces:

- Serial — **PAR_SRC** tied low.
  - I^2S, Sony, Matsushita, S/PDIF and various others.
- Parallel-Synchronous — **PAR_SRC** tied high and **ASYNC** tied low.
  - Source (stream) data is handled in a fixed time scheme via the parallel port. Asynchronous data, if supported, must be handled through the Control Port.
- Parallel-Asynchronous — **PAR_SRC** tied high and **ASYNC** tied high.
  - No fixed timing scheme. Handshaking between the OS8104 and the external application is used. The Packet data buffers of OS8104 are accessible. No synchronous data is handled.
- Parallel-Combined — **PAR_SRC** tied high, **ASYNC** tied low, and the **bPCMA.APCM** bit is set.
  - Synchronous (stream) data and asynchronous (packet) data are handled in a fixed time scheme, at high speed, via the parallel port.

Table 4-1 shows the available pin configurations for the given signal combinations.

PAR_CP	PAR_SRC	ASYNC	SCL*	Description
0	0	x	0	SP in serial mode CP in serial SPI mode
0	0	x	1	SP in serial mode CP in serial I^2C mode
0	1	0	0	SP in Parallel-Synchronous† mode CP in serial SPI mode
0	1	0	1	SP in Parallel-Synchronous† mode CP in serial I^2C mode
0	1	1	0	SP in Parallel-Asynchronous mode CP in serial SPI mode
0	1	1	1	SP in Parallel-Asynchronous mode CP in serial I^2C mode
1	1	0	1	SP in Parallel-Synchronous† mode CP in parallel mode
1	1	1	1	SP in Parallel-Asynchronous mode CP in parallel mode
1	0	x	x	invalid combinations

* The **SCL** value is latched at the rising edge of $\overline{\textbf{RS}}$.

† The Parallel-Combined mode is derived by placing the part in Parallel-Synchronous mode and setting **bPCMA.APCM**

Table 4-1: Source Port and Control Port Configuration Options

Modes are configured when the chip comes out of reset. Once configured, the OS8104 will retain the mode selection until the next hardware reset or power-up reset.

The OS8104 can be controlled remotely by placing the part in Stand-Alone mode, which requires no external intelligence on the local node. The OS8104 receives all commands via the Network and can be used to control external peripherals. In this mode, the Control Port powers-up in serial mode as an I^2C master. The Stand-Alone mode is described in detail in Chapter 17 on page 147. The OS8104 is configured for Stand-Alone mode by tying $\overline{\textbf{RD}}$ and $\overline{\textbf{WR}}$ to ground before the rising edge of $\overline{\textbf{RS}}$, and they must remain at ground during Stand-Alone operation.

If not in Stand-Alone mode, the $\overline{\textbf{RD}}$ and $\overline{\textbf{WR}}$ pins must both remain high between the rising edge of $\overline{\textbf{RS}}$ and the first falling edge of $\overline{\textbf{INT}}$.

5 Control Port in Serial Mode

The Control Port (CP) provides access to all on-chip registers and operates in either serial or parallel mode. The Control Port in parallel mode is discussed in Section 8.1. Selecting the respective mode is done by using the configuration interface. Table 4-1 on page 28 shows all available modes. The respective signals for the Control Port are:

$\overline{RS}$	PAR_CP	SCL	Description
0	x	x	Chip in reset
↑	0	0	Control Port in serial SPI mode
↑	0	1	Control Port in serial I^2C mode
↑	*1	1	Control Port in parallel mode

* **PAR_SRC** must be high when **PAR_CP** is high.

Table 5-1: Control Port Configuration Interface

In serial mode, the Control Port can operate as fast as 400 kbit/s (clock rate), and supports either an I^2C or an SPI transmission protocol. The data is received, MSB-first, through the Control Port and is processed in bytes by the internal firmware of the OS8104. The processing time varies based on of the overall load of tasks. In I^2C mode, clock stretching provides an appropriate handshake mechanism to adapt the data transfer rate dynamically, thereby maximizing the transfer rate at any given time. In SPI mode, the transfer rate must be kept below the worst-case processing time, as specified in *Electrical Characteristics* (Section 18.5.6).

When not configured for Stand-Alone operation, the Control Port operates as a slave device (I^2C or SPI) that must be managed externally. A valid address range of any single access is within a memory page (00h through FFh). The default memory page is 0. Memory address FFh is a special reserved address for switching between memory pages. Therefore, writing the values 0 through 3 to address FFh places all successive memory accesses within one of the four memory pages 0 through 3, respectively.

5.1 I²C Mode

In I²C mode, the I²C address can be one of four address pairs (40h/41h, 42h/43h, 44h/45h, 46h/47h). This allows a maximum of four OS8104 devices on one I²C bus. The pins **AD0** and **AD1** specify the device address, **SCL** clocks data in and out, and **SDA** is the bi-directional data pin. An even address indicates a write access, an odd address a read access to the respective chip. Clock stretching is used as a hand-shake mechanism in I²C mode; therefore, the master device must monitor **SCL** when communicating with the chip.

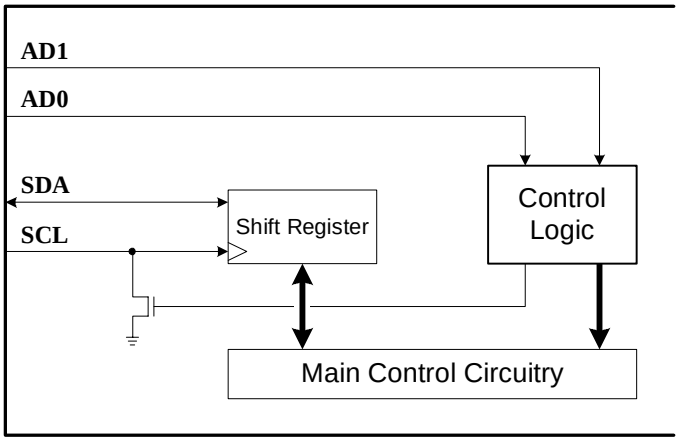


Figure 5-1: Control Port Block Diagram (I²C Mode)

For information about the I²C standard, please refer to the *I²C Specification* from Philips [1]. The following diagram illustrates the OS8104 in an I²C environment:

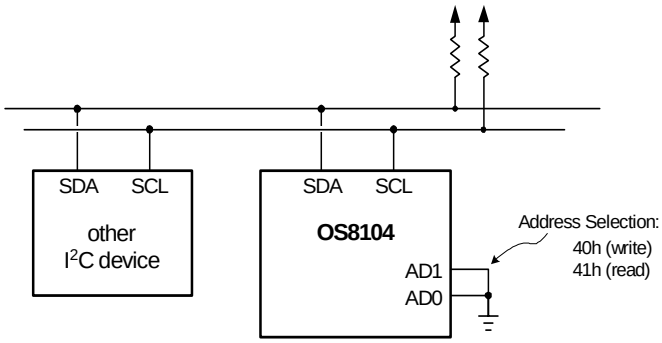


Figure 5-2: Control Port Pin Connections (I²C Mode)

5.1.1 Writing to the Control Port

When writing data to the OS8104 in I^{2}C mode, a pre-defined set of bytes must be sent to select the chip, and then indicate the location in the chip to access. The beginning of transmission is marked by a start condition. The first byte specifies the chip address as well as whether the operation is a read or a write. The address occupies the upper seven bits and the R/$\overline{W}$ bit is the LSB of the first byte. The upper five of the seven address bits are fixed at 01000, and the lower two address bits are configurable via the **AD1** and **AD0** pins.

The first byte of the write operation contains the address and the R/$\overline{W}$ bit set to zero (write operation). After the external system writes the first byte, the Control Port acknowledges the reception of the byte through an acknowledge bit. The second byte transmitted by the external system is the memory address pointer (MAP) which indicates which memory location will be written to (or read from) first.

The third byte is actual data which is written to the location within the OS8104 pointed to by the MAP. The MAP value, internally stored by the OS8104, is automatically incremented after writing a data byte. Therefore, succeeding bytes are written to increasing addresses, supporting efficient transfer of a continuous block of data. Data can be continually written until a stop condition occurs. If the MAP reaches FFh it will wrap back to 00h on the next increment.

Figure 5-3 illustrates the I^{2}C transmission write sequence. The characters “S” and “P” represent the start and stop conditions for messages in I^{2}C mode.

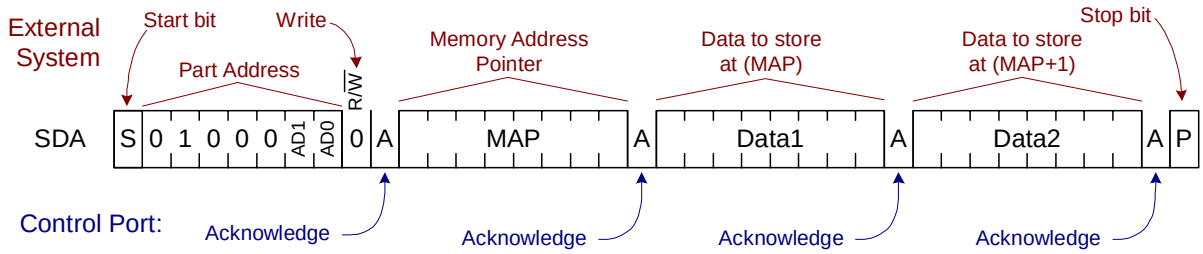


Figure 5-3: Control Port Write Sequence (I^{2}C Mode)

The bits AD1 and AD0 match the values derived from the pins **AD1** and **AD0,** and specify the chip address as follows:

AD[1:0] pins	First Byte Sent	
	Writing	Reading
00	40h	41h
01	42h	43h
10	44h	45h
11	46h	47h

Table 5-2: Control Port I^{2}C Addresses

The default memory page for writing and reading is 0. To switch to another memory page, memory location FFh must be written with the desired page number (0 through 3). This address is reserved on every page for page switching purposes.

5.1.2 Reading from the Control Port

In contrast to the write access, the read access consists of two parts. First, the target address (Memory Address Pointer, or MAP) must be sent to the chip using a separate transmission cycle (delineated by start conditions). Then the data can be read from the MAP address.

The beginning of the first access is marked by a start condition. The first byte (Address) specifies the chip address and the operation. The desired operation is encoded in the LSB, where 1 indicates a read operation, 0 a write operation. To write the MAP, the LSB is cleared in the first byte. The second byte contains the target address (MAP). This first transmission can be ended by a stop condition, or another start condition.

The second access is initiated with a start condition, followed by the address byte, which must specify a read operation (LSB set to 1). The OS8104 will then transmit byte after byte (auto-incremented), until a stop condition occurs. If the MAP reaches FFh it will wrap back to 00h on the next increment.

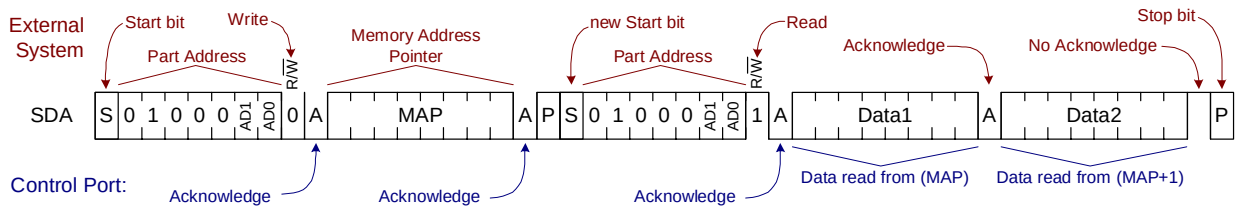


Figure 5-4: Control Port Read Sequence (I^2C Mode)

If the clock rate on the I^2C bus exceeds the maximum, clock stretching will occur.

The default memory page for writing and reading is 0. To switch to another memory page, memory location FFh must be written with the desired page number (0 through 3). This address is reserved on every page for page switching purposes.

5.2 SPI Mode

When the Control Port is in SPI mode, a unique chip select, $\overline{CS}$, is used in lieu of an address byte. SCL clocks data in and out of the chip, SDIN is the data input, and SDOUT is the data output. Similar to the I^2C format, reading and writing is controlled by the LSB of the first byte sent after $\overline{CS}$ goes low.

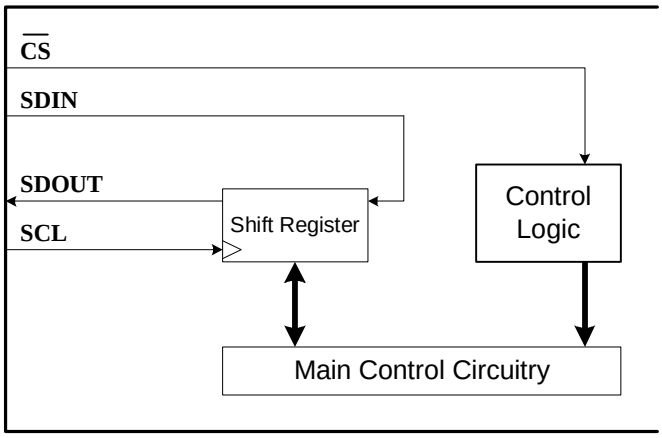


Figure 5-5: Control Port Block Diagram (SPI Mode)

5.2.1 Writing to the Control Port

The beginning of the transmission is marked by a falling edge at $\overline{CS}$. Then, an address byte must be sent with the LSB cleared (write operation), followed by the MAP byte. The upper seven bits of the address byte are ignored. Each byte sent after that sequence is interpreted as data bytes to be written to memory locations, until $\overline{CS}$ is set to high again. The target address for the memory location is automatically incremented. If the MAP reaches FFh it will wrap back to 00h on the next increment.

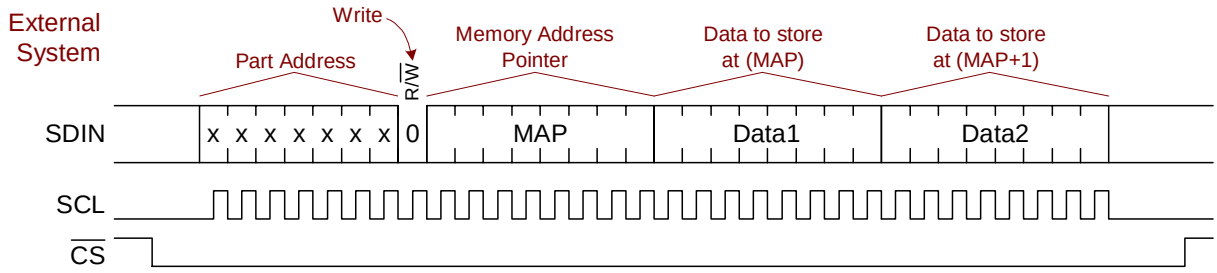


Figure 5-6: Control Port Write Sequence (SPI Mode)

Similar to I^2C mode, the default memory page for writing or reading is page 0. To switch to another memory page, memory location FFh must be written with the desired page number (0 through 3). This address is reserved on every page for page switching purposes.

5.2.2 Reading from the Control Port

Similar to the I^2C read, the SPI Control Port read operation consists of two separate cycles. The first transmits the MAP byte (memory address to read from) to the chip and the second cycle reads the data. The first cycle starts at the falling edge of $\overline{CS}$, and consists of a write operation to set the MAP value. The $\overline{CS}$ line must go high to indicate the end of the first cycle. The next falling edge of $\overline{CS}$ initiates the second cycle, and the address byte is set with the LSB set (read operation). The next byte transferred will be data read from the previously written MAP location, with each successive byte coming from increasing addresses. The MAP is auto-incremented by the OS8104. If the MAP reaches FFh it will wrap back to 00h on the next increment. When finished reading, $\overline{CS}$ is brought high, ending the second cycle.

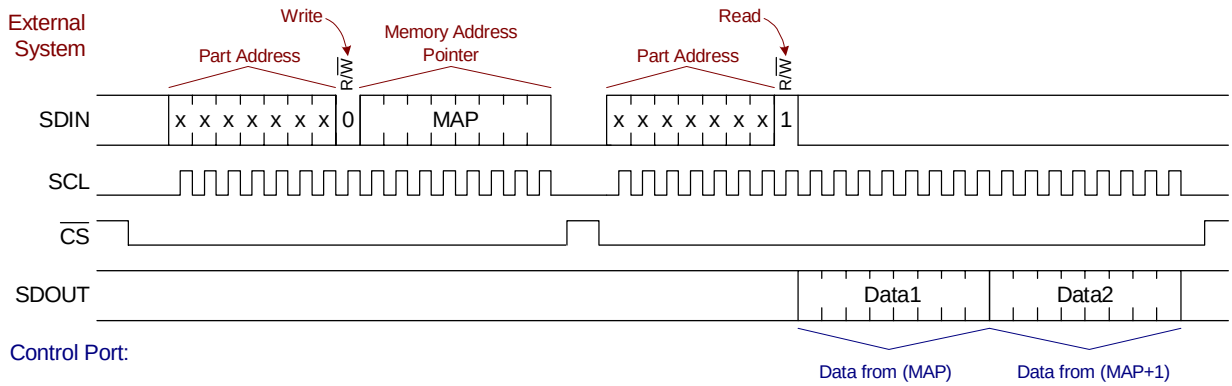


Figure 5-7: Control Port Read Sequence (SPI Mode)

The default memory page for writing or reading is page 0. To switch to another memory page, memory location FFh must be written with the desired page number (0 through 3). This address is reserved on every page for page switching purposes.

6 Network Interface

The MOST transceiver interfaces to the MOST Network. The **RX** (receive) and **TX** (transmit) pins are TTL compatible and can be directly interfaced with fiber optic receiver/transmitter devices (FOR/FOX units).

The main task of the Network interface is to decode and encode the bit stream. A master clock is recovered from the bit stream, and the node is synchronized on a Frame and Block level. The clock recovery and synchronization is accomplished by a high-precision Phase Lock Loop (PLL) in combination with the data-decoding circuit.

The Network interface can be configured as the timing-master or a timing-slave node. Only one timing-master node can exist in the Network, with all other Network nodes configured as timing-slaves. The timing-master node operates from its own clock source that can be derived from a crystal oscillator or an external clock. All the timing-slave nodes use a recovered clock from the Network (**RX** pin) as the internal master clock. In a Network, the single master node generates the timing for all other nodes (timing-slaves).

6.1 MOST Frame Structure

Data on the MOST Network are organized in frames. Frame generation is provided by a single device in the system, referred to as the timing-master or frame generator. Although this can be any device in the Network, it would typically be a control unit or human-machine interface (HMI) device.

The MOST frame structure provides maximum flexibility in terms of compatibility with a number of existing communication and data transport requirements, without any drawbacks in implementation cost or processing overhead. Built-in structures allow simple Network management on the lowest layers, avoiding overhead and cost shortcomings. The MOST frame is illustrated in Figure 6-1.

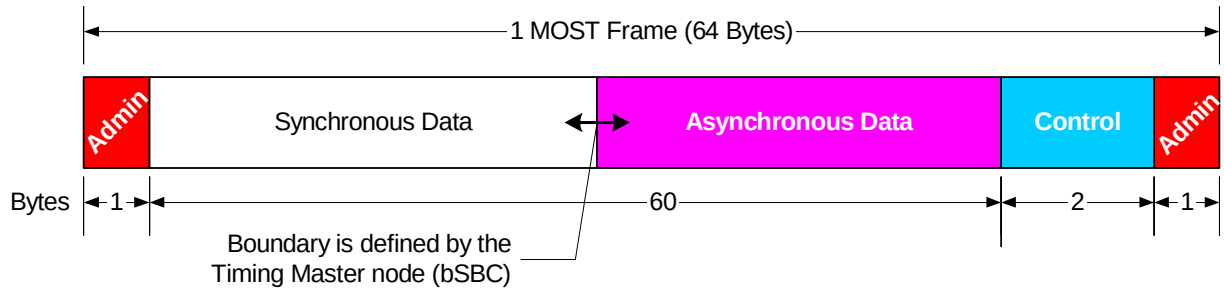


Figure 6-1: General MOST Frame Structure

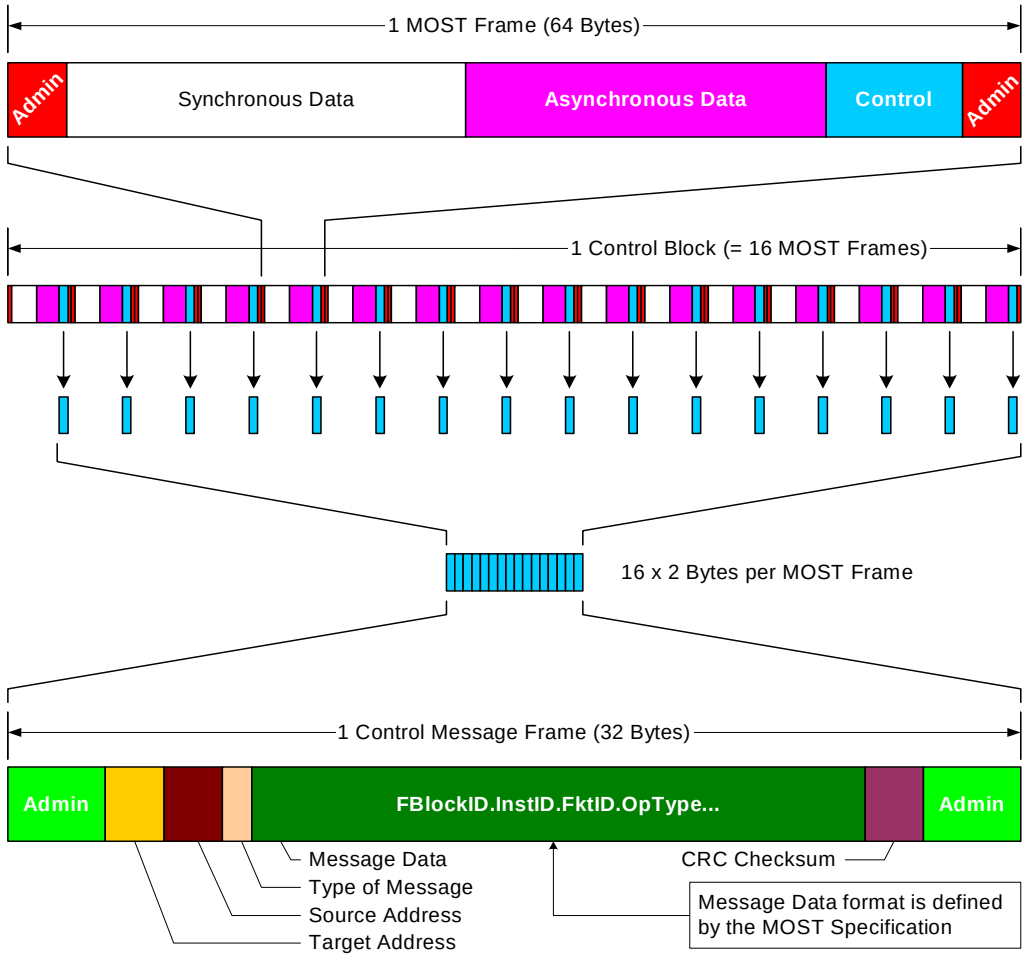


Figure 6-2: Control Message Frame

In addition to a small amount of administrative data, each MOST frame transports two bytes of Control data, along with 60 bytes of source data. A unit of 16 MOST frames is defined as a Control block. Since each frame transports two bytes of Control data, a block transports a total of 32 bytes of Control data, defined as a Control message frame or Control frame, see Figure 6-2.

Control messages are transported within a control frame. For more information on Control messages, see Chapter 13 on page 111.

6.2 Network Configuration

The MOST Network-related functions of the OS8104 are controlled via the following set of registers:

- bXCR — Transceiver Control Register. Control of bypasses, output enable, timing-master/slave selection.
- bXSR — Transceiver Status Register. Controls reporting of errors and reports error events.
- bSBC — Synchronous Bandwidth Control. Controls the number of bytes used for synchronous data transfer vs. the number of bytes used for asynchronous packet data transfer.
- bNDR — Node Delay Register. Reports source data delay between timing-master and local node.
- bNPR — Node Position Register. Reports physical position of a node, relative to the Network timing-master.
- bMPR — Maximum Position Register. Reports total number of active nodes in the Network.
- bMDR — Maximum Delay Register. Reports total synchronous data delay in the Network.

All registers are accessible via the Control Port. While the bXCR, bXSR, and bSBC (timing-master node only) registers are written by the application, the contents of bNDR, bNPR, bMPR, and bMDR registers are calculated automatically. The values in the bNDR, bNPR, bMPR, and bMDR registers are valid only when the Network is in a locked state and Network initialization has finished.

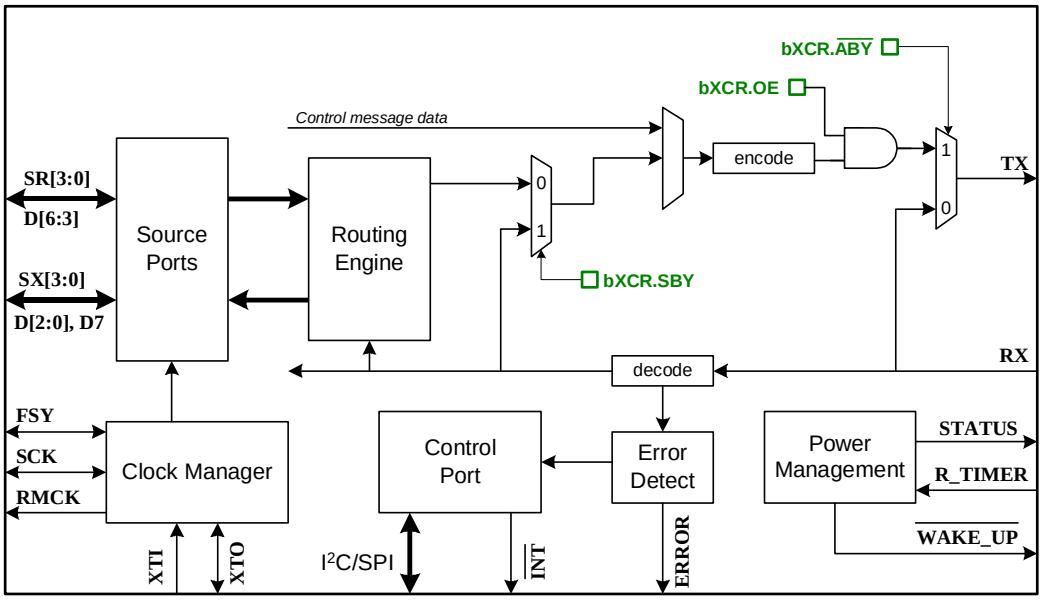


Figure 6-3: Network Interface Overview

6.2.1 bXCR (Transceiver Control Register)

The bXCR register controls the main MOST Network-related functions of the OS8104. After reset/power-up reset, the transceiver comes up as a timing-slave in *All-bypass* mode where received data (**RX**) is directly passed to **TX**, unaltered.

80h bXCR Transceiver Control Register

Bit	Name	Description	Default
7	MTR	Timing Master/Slave select	0
6	OE	Transmitter output enable	0
5	rsvd	Reserved; Write as 1	1
4	LPW	Low-Power wake-up (write only)	0
3	SAN	Stand-Alone mode; Write as 0	0
2	SBY	Source data bypass	0
1	$\overline{\text{ABY}}$	All-bypass mode	0
0	$\overline{\text{REN}}$	RMCK enable	0

Table 6-1: bXCR (Transceiver Control Register)

MTR Timing Master/Slave select. When the **MTR** bit is set, the transceiver acts as the timing-master for the whole Network. When the node is configured as a timing-master, it also handles the resource allocation, Network supervision, etc. The **bCM1.MX[1:0]** bits select the timing source.

OE Transmitter output enable. If the transceiver is configured as an active part in the Network (**$\overline{\text{ABY}}$** set), the **OE** bit controls the **TX** output pin. When **OE** is set, the transmitter pin (**TX**) is enabled. When **OE** is cleared, **TX** is forced low.

LPW Low-Power wake-up. The **LPW** bit is functional only when the **MTR** bit is set (in the timing-master). When **LPW** is set, the transceiver generates a wake-up signal that wakes up all Network nodes that are in Low-Power mode. This bit is write-only and always reads as 0.

SAN — Stand-Alone mode. When the **SAN** bit is read as 1, it indicates that the chip is running in Stand-Alone mode (described in Chapter 17 on page 147). When writing to this register, the **SAN** bit must not be modified (written as 1 in Stand-Alone mode, or written as 0 otherwise). Stand-Alone mode is set via hardware at power-up by tying the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins low at reset.

SBY — Synchronous Source Data Bypass. Nodes that do not need to receive or transmit synchronous data can set **SBY** to minimize the delay through the Network. When the **SBY** bit is cleared, the node increments the node delay counter (bNDR) by one and the delay through the node for synchronous data is two frames. When **SBY** is clear, the node can control synchronous routing via the MRT registers.

$\overline{\text{ABY}}$ — All-bypass mode enable

0 — When the $\overline{\text{ABY}}$ bit is cleared, the signal received at **RX** from the Network is directly connected to the **TX** pin and is unaltered by the node. This keeps the overall locking time in the Network small since all PLLs come up simultaneously and also ensures that only nodes with a running application can enter into the Network.

1 — To disable the All-bypass mode and activate the node on the Network, the $\overline{\text{ABY}}$ bit must be set, where the node increments the Node Position Register (bNPR).

$\overline{\text{REN}}$ — **RMCK** enable

0 — When the $\overline{\text{REN}}$ bit is cleared, the **RMCK** pin outputs a clock based on the **bCM1.RD[2:0]** bits.

1 — When the $\overline{\text{REN}}$ bit is set, the **RMCK** output is high impedance. Changing $\overline{\text{REN}}$ can cause glitches on **RMCK** in the OS8104.

6.2.2 bXSR (Transceiver Status Register)

Register bXSR indicates errors and status of the MOST Network. Errors are described in more detail in Section 6.2.4 on page 39.

81h bXSR Transceiver Status Register

Bit	Name	Description	Default
7	rsvd	Reserved; Write as 0	0
6	MSL	Mask *S/PDIF lock error*	1
5	MXL	Mask *transceiver lock error*	0
4	ME	Mask *coding error*	1
3	ERR	All error capture	0
2	rsvd	Reserved; Write as 0	0
1	ESL	Error capture - *S/PDIF lock error*	0
0	EXL	Error capture - *transceiver lock error*	0

Table 6-2: bXSR (Transceiver Status Register)

MSL — Mask *S/PDIF lock error*. When the **MSL** bit is set, the **ERR** bit will not be set when an S/PDIF lock error occurs.

MXL — Mask *transceiver lock error*. When the **MXL** bit is set, the **ERR** bit will not be set when a transceiver lock error occurs.

ME — Mask *coding error*. When the **ME** bit is set, the **ERR** bit will not be set when a coding (bi-phase or parity) error occurs on the Network (**RX**) bit stream.

ERR — All error capture. The **ERR** bit is set by the transceiver whenever an unmasked-error has occurred (based on the **MSL**, **MXL**, and **ME** bits). To generate an interrupt ($\overline{\text{INT}}$ low), the **bIE.IERR** bit must be set. The **ERR** bit is sticky and must be written to 0, to clear it. The **ERROR** pin is not sticky and only indicates an error for the length of the event causing the error (see Figure 6-4).

ESL — Error capture - *S/PDIF lock error*. The **ESL** bit flags the occurrence of S/PDIF lock errors. The **ESL** bit is sticky and can be cleared/re-triggered for a new capture by clearing it. If multiple events are connected to the **ERR** bit, the **ESL** and **EXL** bits can help determine the cause of **ERR** being set.

EXL — Error capture - *transceiver lock error*. The **EXL** bit flags the occurrence of transceiver lock errors. The **EXL** bit is sticky and can be cleared/re-triggered for a new capture by writing a zero to it. If multiple events are connected to the **ERR** bit, the **ESL** and **EXL** bits can help determine the cause of **ERR** being set.

6.2.3 bXSR2 (Transceiver Status Register 2)

97h bXSR2 Transceiver Status Register 2

Bit	Name	Description	Default
7..2	rsvd	Reserved; Write as 0	000000
1	INV	**RX** Inversion Control. Sets the polarity of the Network received data (**RX**), which affects the polarity of pulse width distortion. This value should be optimized for the respective FOR device used.	0
0	rsvd	Reserved; Write as 0	0

Table 6-3: bXSR2 (Transceiver Status Register 2)

6.2.4 Transceiver Error Events

The **bXSR.ERR** bit is set when the transceiver obtains lock or when S/PDIF or line-coding errors occur. Each of these events can be masked out by an individual mask bit (**MSL**, **MXL** and **ME**) in the bXSR register. Figure 6-4 shows how the mask bits, the **ERROR** pin, and the respective flags are related.

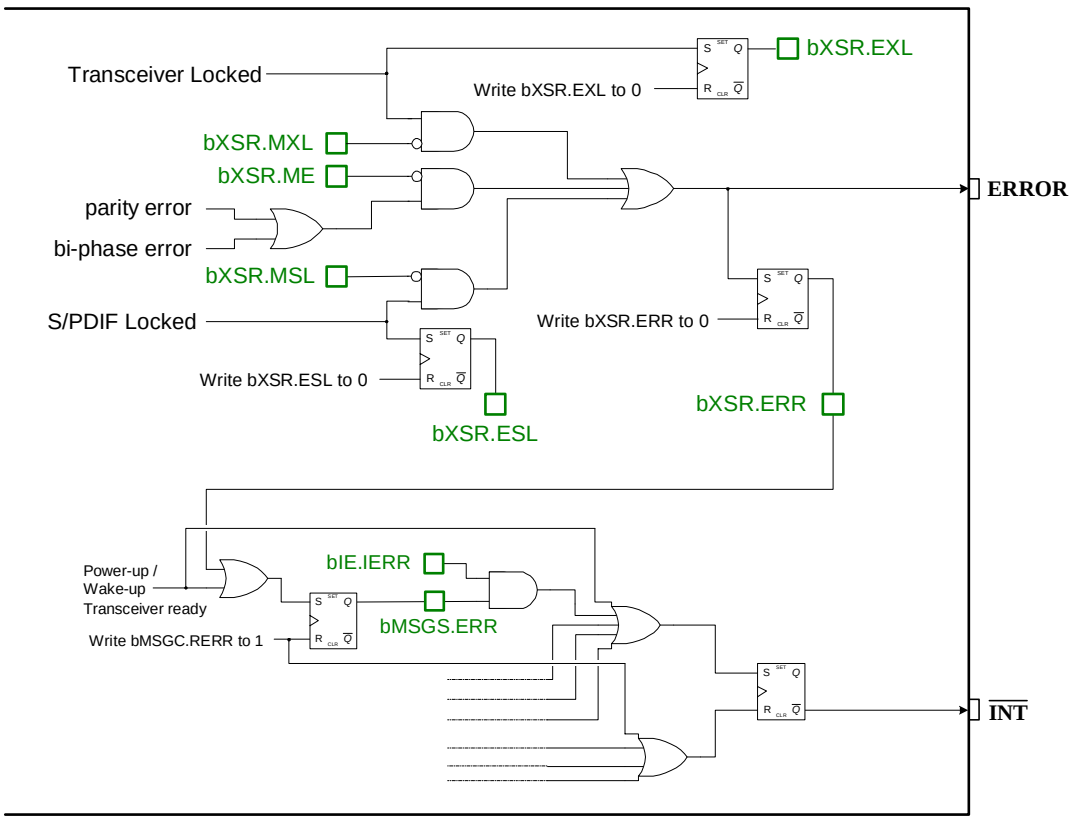


Figure 6-4: Error Flags and Error Masks

Once the **bXSR.ERR** bit is set, it can be cleared for a new capture by writing it to 0. The same applies to **bXSR.ESL** and **bXSR.EXL**. An interrupt is generated when **bXSR.ERR** is set, if the **bIE.IERR** bit is also set (See Section 11.1 on page 95). This error interrupt can be reset by setting **bMSGC.RERR**.

If the interrupt service routine is finished before the error event disappears, another interrupt is generated. It is possible either to:

- Mask the respective error event in bXSR, clear **bXSR.ERR**, and then set **bMSGC.RERR**.
- Clear **bIE.IERR** and then set **bMSGC.RERR**.

The **ERROR** pin is active for the period of time an error event lasts; as a result, short pulses can occur on the **ERROR** pin. Since the **bXSR.ERR** bit is sticky, applications may find the bit more convenient than the **ERROR** pin.

6.2.5 bSBC (Synchronous Bandwidth Control Register)

In each MOST frame, 60 bytes are available for transporting source data. These bytes are referred to as *physical channels*, and are organized in units of four, called *quadlets*. Therefore, the OS8104 provides 15 quadlets of source data. Two transport methods use the source data: Synchronous (stream) data transfer, and Asynchronous (packet) data transfer. The bSBC register determines how the quadlets are divided between the two services.

The division between the two transfer types is controlled by the timing-master node by specifying the number of quadlets used for synchronous data transfer, in the bSBC register. The bSBC register in timing-slave nodes is read-only and is updated automatically.

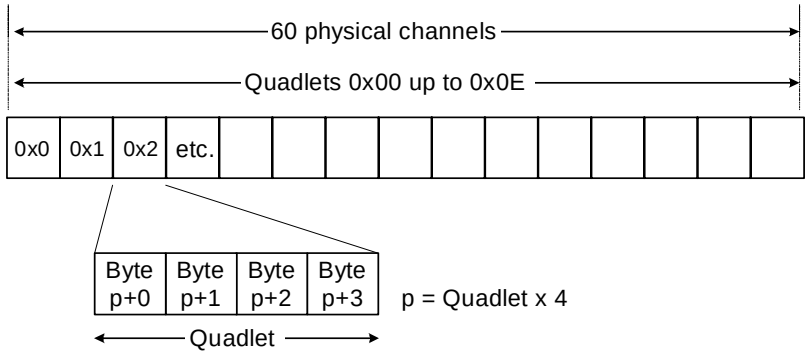


Figure 6-5: Source Data Quadlets

Between 6 and 15 quadlets can be assigned to synchronous data transfer, providing anywhere from 24 to 60 physical channels for synchronous data transfer. When assigning the maximum of 60 physical channels (15 quadlets) to synchronous data transfer, no asynchronous data transfer is possible. Although the OS8104 supports all 15 quadlets reserved for Synchronous data, the *MOST Specification* requires that at least one quadlet be reserved for Packet data. The number of quadlets available for asynchronous data transfer is calculated as follows:

Number of asynchronous quadlets = 15 – contents of bSBC

The organization of physical channels in quadlets is only relevant for specifying the bSBC value. Physical channels assigned to synchronous data transfer are independent of each other and can be allocated individually for resource management or *routing* of synchronous data. The synchronous bytes are numbered from 00h to 3Bh (if all channels are assigned to synchronous data transfer).

The byte number is calculated by using the following formula:

Byte Number = (Number of quadlet × 4) + Byte number within quadlet

> The maximum number of asynchronous bytes per frame is 36 (9 quadlets). The minimum number of synchronous quadlets is 6.

During normal operating conditions, bSBC will be written once during the general initialization phase of the MOST Network. Every time bSBC is changed, the complete allocation mechanism of the Network must be re-initialized by sending a *De-allocate All* message to the timing-master. This also applies to reset or power-up reset. For more information about de-allocating Network resources, see Section 13.5.2.5.

96h bSBC Synchronous Bandwidth Control Register

Bit	Name	Description	Default
7..4	rsvd	Reserved; Write as 0	0000
3..0	SAC[3:0]	Synchronous Area Count. Number of synchronous quadlets. Read-only when **bXCR.MTR** is clear (timing-slave device). Read/write when **bXCR.MTR** is set.	0000

Table 6-4: bSBC (Synchronous Bandwidth Control Register)

After being configured as the timing-master, the application must set bSBC to an appropriate value in the timing-master node. Timing-slave nodes extracts bSBC information from the incoming MOST frame. The value in bSBC is only valid if the chip is in lock state.

> The default value in bSBC is not valid. The timing-master node must initialize **SAC[3:0]** to a value between 6 and 15. (The *MOST Specification* allows no more than 14 synchronous quadlets.)

6.2.6 bNDR (Node Delay Register)

8Fh bNDR Node Delay Register

Bit	Name	Description	Default
7..0	NDR[7:0]	Network node delay (read-only)	00h

Table 6-5: bNDR (Node Delay Register)

The Node Delay register determines how many node delays are between this node and the timing-master node, for synchronous data. The node delay value in the timing-master is 0 and is incremented by one in every node that has its **bXCR.SBY** bit cleared. In addition, the node will delay synchronous data by two frames before transmitting to the next node. If the **bXCR.SBY** bit of a node is set, the node does not handle synchronous data to or from the Network. Therefore, the received synchronous data will be directly retransmitted incurring no frame delays, and the bNDR register will not be incremented. The value in the bNDR register is only valid after lock is established and Network initialization has finished.

6.2.7 bNPR (Node Position Register)

87h bNPR Node Position Register

Bit	Name	Description	Default
7..0	NPR[7:0]	Network node position (read-only)	00h

Table 6-6: bNPR (Node Position Register)

The Node Position register indicates the physical position of a node in the Network, or the physical address, which can be used when sending Control messages. The node position value in the timing-master node is 0. The timing-slave node connected to the **TX** output of the timing-master node has a bNPR of 1. Nodes in All-bypass mode (**bXCR.$\overline{\text{ABY}}$** clear) are invisible to the Network and do not increment bNPR. bNPR is only valid after lock is established and Network initialization has finished.

6.2.8 bMPR (Maximum Position Register)

90h bMPR Maximum Position Register

Bit	Name	Description	Default
7..0	MPR[7:0]	Number of active nodes in the Network (read-only)	00h

Table 6-7: bMPR (Maximum Position Register)

The Maximum Position register indicates the total number of active nodes in the MOST Network (nodes not in All-bypass mode). If this register changes, **bMSGS.ALC** is set. In addition, an interrupt is generated if **bIE.IALC** is set. bMPR is only valid after lock is established and Network initialization has finished. The maximum number of nodes in a MOST Network is 64. If 64 nodes exist, bMPR will read 00h.

6.2.9 bMDR (Maximum Delay Register)

91h bMDR Maximum Delay Register

Bit	Name	Description	Default
7..0	MDR[7:0]	Total delay in the Network (read-only)	00h

Table 6-8: bMDR (Maximum Delay Register)

The Maximum Delay Register indicates the total number of node delays for synchronous data within the Network. If this register changes, **bMSGS.ALC** is set. In addition, an interrupt is generated if **bIE.IALC** is set. bMDR is only valid after lock is established and Network initialization has finished.

6.2.10 Network Registers After Lock

The contents of bNPR, bNDR, bMPR, bMDR, and mCRA are updated automatically after the Network achieves lock. Table 6-9 indicates when the register contents will be valid.

Register/ Buffer	Best Case	Worst Case	Comments
bNPR	$\frac{2}{Fs}$	$\frac{4}{Fs}$	Any node
bNDR	$\frac{2}{Fs}$	$\frac{1024}{Fs}$	Any node
bMPR†	$\frac{1024}{Fs}$	$\frac{(NPR+2)\times 1024}{Fs}$	Any node
bMDR†	$\frac{1024}{Fs}$	$\frac{(NPR+2)\times 1024}{Fs}$	Any node
mCRA		$\frac{(NPR+2)\times 1024}{Fs}$	Timing-Slave nodes
		$\frac{(MPR+1)\times 1024}{Fs}$	Timing-Master node

† After lock of the complete network

Table 6-9: Network Register Update Times

7 Source Ports in Serial Mode

The term *source data* refers to any data which is transmitted, transported, and received in a continuous stream, meeting real-time requirements. A typical application for source data is audio data transmitted from an A/D converter to an amplifier. Therefore, the hardware interface to external applications is called the *Source Data Ports*, or *Source Ports*.

The OS8104 can receive and transmit data between external applications and the MOST Network simultaneously. The **SRn** pins (n = 0 to 3) receive source data externally for transmission onto the Network, and the **SXn** pins transmit data to external applications from the Network receiver. The Source Ports support many different data formats and can operate in serial or parallel modes. Serial formats such as I^2S, Matsushita, Sony, and S/PDIF are supported. In parallel mode, both synchronous data and asynchronous (packet) data can be transferred. The parallel access mode of the Source Ports is described in Chapter 8.

After a reset, **SCK** should be driven to a known state before Network lock (**bCM2.$\overline{\text{LOK}}$** low). The following recommendations are offered as a means to ensure **SCK** is non-floating at the time of lock.

- **SCK** may be configured as an input and always driven by an external source.
- A pull-down resistor at **SCK** may be used.
- **SCK** may be configured as an output before lock by performing a reset pulse (of at least 1 ms), waiting for the falling edge of **$\overline{\text{INT}}$**, then setting **bSDC1.IO** within 1.2 ms at 44.1 kHz (1.5 ms at 48 kHz). This can be accomplished through the `Most_Reset()` NetServices callback function.

> For transporting any data via the Source Ports, the chip must be in lock state. In un-lock state, synchronization to the Network is lost; therefore, no source data exchange is possible. The external interface itself is still active and does not need to be deactivated by the application.

In serial mode, the Source Ports can be used for synchronous or transparent (asynchronous) data input and output. In addition, the **SR0** and **SX0** pins can be configured for S/PDIF (IEC-60958 parts 1 and 3) format. These pins can decode and encode the bi-phase data, handle the preambles, and generate an S/PDIF data stream, based on internal timing. The **SR0** and **SX0** pins can operate at rates from 1 to 8 times the standard S/PDIF data rate (Fs). In S/PDIF mode, the **SX0** pin outputs the block start preamble by receiving a block bit from the received bit stream or by automatic insertion in case the block bit is not in the bit stream (i.e., received data stream is not S/PDIF).

The transparent (asynchronous) mode uses **SR1** and **SX1**, where **SR1** is configured as an input port for sampling incoming signals. For more information about transparent data transport and routing, see Section 7.5 (*Transparent Data Transport*) and Section 12.5 (*Transparent Channel Data Routing*).

Source data control registers select the serial mode format for the Source Port.

- bSDC1 - Source Data Control register 1. Controls the polarity of different interface signals, muting, and S/PDIF selection.
- bSDC2 - Source Data Control register 2. Controls the clock rates (**SCK**, S/PDIF, Transparent channels, and multi-speed **FSY**).
- bSDC3 - Source Data Control register 3. Special S/PDIF control (synchronization source, and I/O direction in S/PDIF 8x mode).

7.1 Serial Source Port Interface

The Source Ports are typically connected to multimedia sources and/or sinks that handle audio, video or other data streams. The **SR[3:0]** pins receive data from external sources, and the **SX[3:0]** pins transmit data to external sources. The **SCK** pin is the bit clock for all four inputs and all four outputs. The **FSY** pin is the Frame Sync delineating the word/channel (left vs. right channel) boundaries for all the data pins. **FSY** and **SCK** can be configured as outputs or inputs. When **FSY** is an output, it is synchronized to the MOST Network Frame; therefore, when lock is established, **FSY** will be resynchronized to the Network, which can cause an abnormally short or long **FSY** pulse.

For I^2S format compatibility, a delay bit in bSDC1 is available to shift the data relative to **FSY** by one bit (one **SCK** cycle). Possible clock modes are 8/16/32/48/64/128/256 times the sampling frequency (Fs).

Each Source Port data pin is connected to an 8-bit shift register. Figure 7-1 shows how the shift registers are arranged.

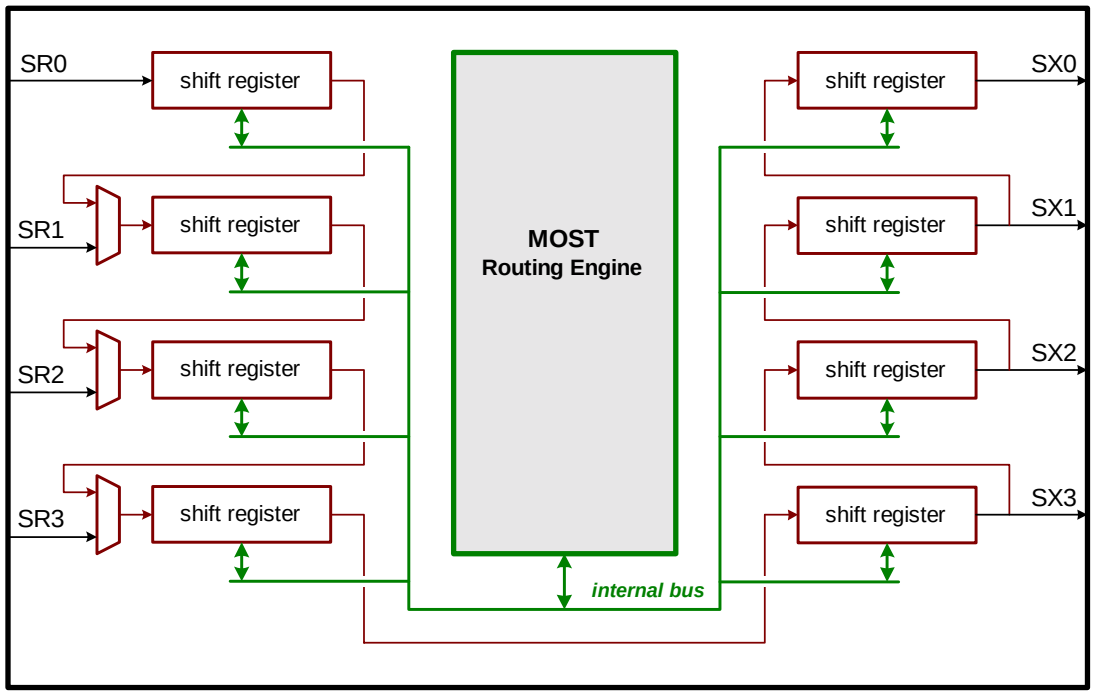


Figure 7-1: Source Port Block Diagram (Serial Mode)

To support higher data rates, the shift registers can be cascaded. A typical bit rate for serial Source Ports is 64Fs, in which **SCK** runs 64 times faster than the MOST Network frames. Therefore, as many as 64 bits per frame are shifted into each shift register. The internal Routing Engine reads each Source Port eight times (when set to 64Fs) per frame to get all 64 bits.

Figure 7-2 shows the transfer of synchronous serial data for one port running at 64Fs:

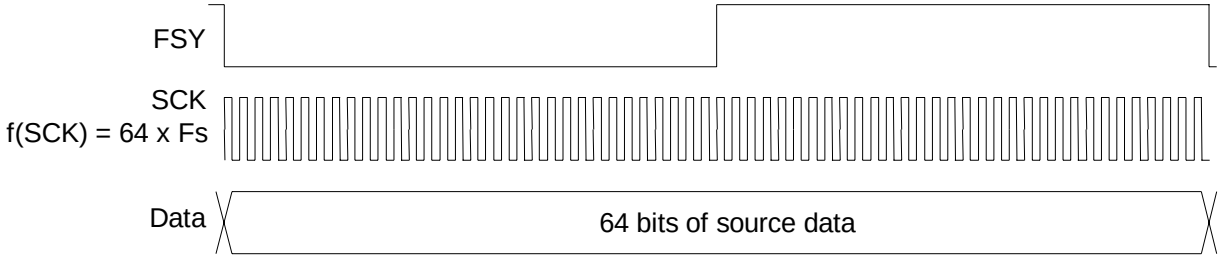


Figure 7-2: Source Port at 64Fs

When cascading shift registers, the number of available Source Port pins will be reduced. Cascading all the shift registers produces a 64-bit shift register. Since the Source Port is read eight times per Fs interval, 512 bits can be shifted into or out of the chip. In this mode, only one Source Port pin is available in one direction. Either **SR0** can input 512 bits or **SX0** can output 512 bits. This mode is only available when **SR0**/**SX0** is configured for S/PDIF mode running at 8x standard speed.

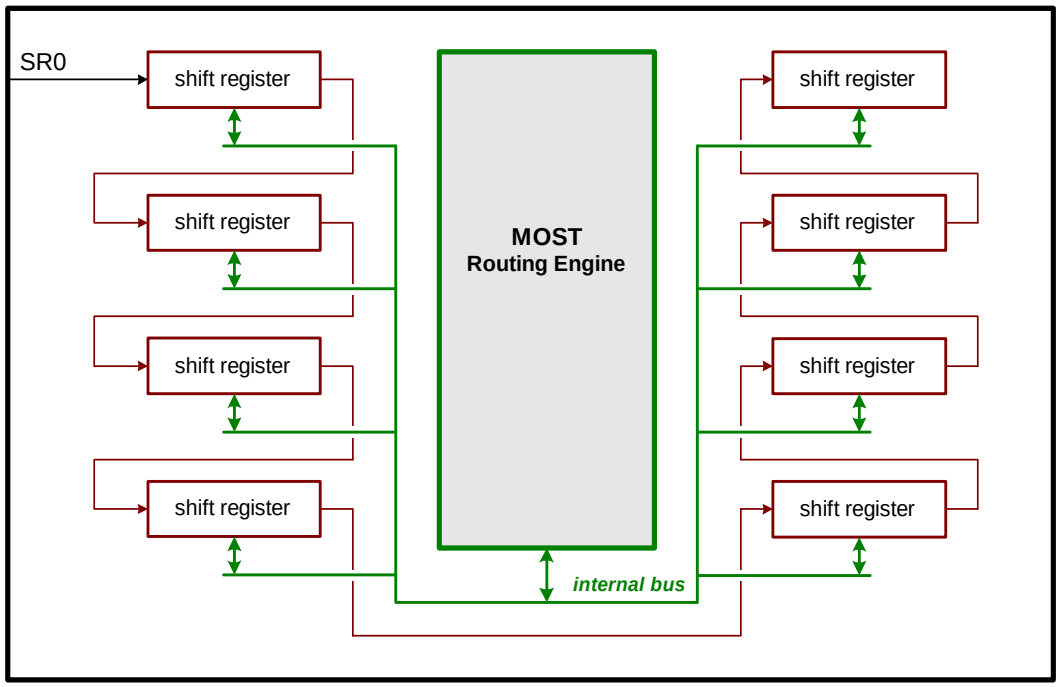


Figure 7-3: Source Port Cascaded (Serial 512Fs Mode)

7.2 Source Port Configuration (Serial)

The general configuration (whether the Source Ports work in serial or parallel mode) is done when the OS8104 exits the reset state. When transmitting data onto the MOST Network, the MSB is transmitted first. To comply with the *MOST Specification*, when transmitting multi-byte words, the most significant byte should be transmitted first. In addition, when transmitting stereo data, left should be sent before right.

7.2.1 bSDC1 (Source Data Control Register 1)

The Source Ports support a variety of serial data formats. The Source Data Control Register 1 (bSDC1) controls the serial data format.

82h bSDC1 Source Data Control 1 Register

Bit	Name	Description	Default
7	EDG	Active edge of **SCK**	0
6	DEL	Delay first bit against **FSY**	0
5	POL	Polarity of **FSY**	0
4	IO	Input/Output select of **FSY** and **SCK**	0
3	NBR	Number of **SCK** cycles per frame	0
2	SPD	S/PDIF port enable	0
1	$\overline{\text{MT}}$	Mute Source Port outputs	0
0	$\overline{\text{TCE}}$	Transparent channel enable	0

Table 7-1: bSDC1 (Source Data Control Register 1)

EDG — Active edge of **SCK**. When the **EDG** bit is cleared, the **SRn** pins are sampled on the falling edge of **SCK**, and the **SXn** pins transition on the rising edge. When the **EDG** bit is set, the **SRn** pins are sampled on the rising edge of **SCK**, and the **SXn** pins transition on the falling edge.

DEL — Delay first bit relative to **FSY**. If the **DEL** bit is set, a one **SCK** cycle delay exists between the change of **FSY** (Frame Sync) and the first data bit sampled (I^2S format). When the **DEL** bit is cleared, the first data bit is at the **FSY** edge (no delay).

POL — Polarity of **FSY**. When the **POL** bit is set, the rising edge of **FSY** indicates the start of frame. When the **POL** bit is cleared, the start of frame is indicated by the falling edge of **FSY**.

IO — Input/Output select of **FSY** and **SCK**. When the **SPD** bit is set, the **IO** bit is a don't care (**FSY** and **SCK** are outputs). When **SPD** is cleared, **IO** clear configures the **FSY** and **SCK** pins as inputs, **IO** set configures the **FSY** and **SCK** pins as outputs.

When Network lock is achieved, the **SCK** pin should not be floating. One way to accomplish this is by configuring **SCK** as an output (**IO** set) after the power-on interrupt, but prior to lock. See the beginning of this Chapter for more information on ensuring non-floating values of **SCK**.

FSY and **SCK** must be supplied at all times when they are set as inputs (**IO** clear and **SPD** clear).

NBR — Number of **SCK** cycles per frame. The **NBR** bit is used only when the clock rate of the Source Ports is set to 64Fs (**bSDC2.SPR[2:0]**). If **SPR[2:0]** is 64Fs and the **NBR** bit is 0, **SCK** is set to 64Fs. If **SPR[2:0]** is 64Fs and the **NBR** bit is 1, **SCK** is set to 48Fs. When **SCK** is an output at 48Fs, the actual clock frequency is 64Fs with 16 clock cycles removed. In each half of the **FSY** cycle, the third (out of four) set of 8 clocks are removed, as illustrated in Figure 7-4. When **SCK** is an input at 48Fs, it must be continuous (non-gated).

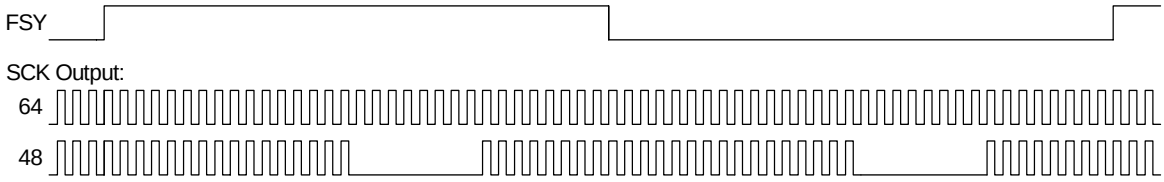


Figure 7-4: Source Port SCK Output Timing (bSDC1.EDG = 0)

SPD — S/PDIF port enable. When the **SPD** bit is set, the **SR0** and **SX0** pins are configured for S/PDIF format, and the **FSY** and **SCK** pins are forced to outputs. The S/PDIF speed is selected by **bSDC2.SDR[1:0]**. When **SPD** is clear, standard serial modes for **SR0** and **SX0** are enabled.

$\overline{\text{MT}}$ — Mute Source Port outputs. When the $\overline{\textbf{MT}}$ bit is cleared, **SX[3:0]** pins are forced low. When the $\overline{\textbf{MT}}$ bit is set, the **SX[3:0]** pins output data.

$\overline{\text{TCE}}$ — Transparent Channel enable. When the $\overline{\textbf{TCE}}$ bit is cleared, **SR1** and **SX1** are configured for transparent mode operation, and the signal at **SR1** is sampled with a frequency calculated as follows:

$$\text{sample clock rate} = \frac{\textbf{bSDC2.SPR[2:0]} \text{ setting}}{\textbf{bSDC2.TCR[1:0]} \text{ setting}}$$

Transparent mode is not available when the **SCK** clock rate is 128Fs or 256Fs.

7.2.1.1 I²S (Philips) Source Data Format

To support an I²S format [11], bSDC1 should be configured as follows (Figure 7-5 illustrates the timing):

- bSDC1 bits **EDG** = **DEL** = 1
- bSDC1 bits **POL** = **NBR** = 0

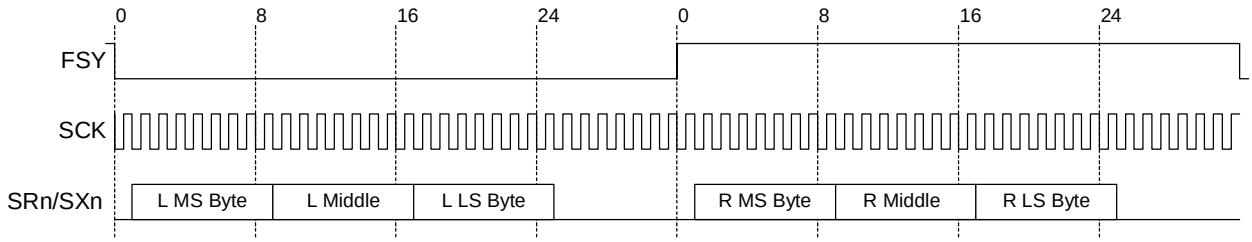


Figure 7-5: I²S Source Data Format

In Figure 7-5, data is arranged MSB first, high byte first.

7.2.1.2 Sony Source Data Format

To support a Sony format, bSDC1 should be configured as follows (Figure 7-6 illustrates the timing):

- bSDC1 bits **EDG** = **POL** = **IO** = **NBR** = 1
- **bSDC1.DEL** = 0

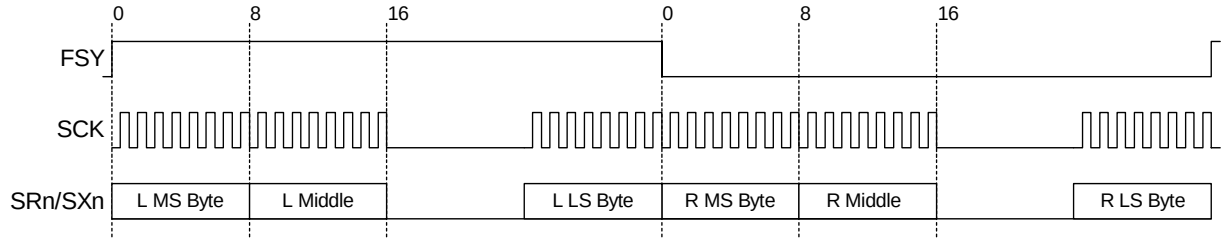


Figure 7-6: Sony Source Data Format

In Figure 7-6, data is arranged MSB first, high byte first.

7.2.1.3 Matsushita Source Data Format

To support a Matsushita or left-justified format, bSDC1 should be configured as follows (Figure 7-7 illustrates the timing):

- bSDC1 bits **EDG** = **POL** = 1
- bSDC1 bits **DEL** = **NBR** = 0

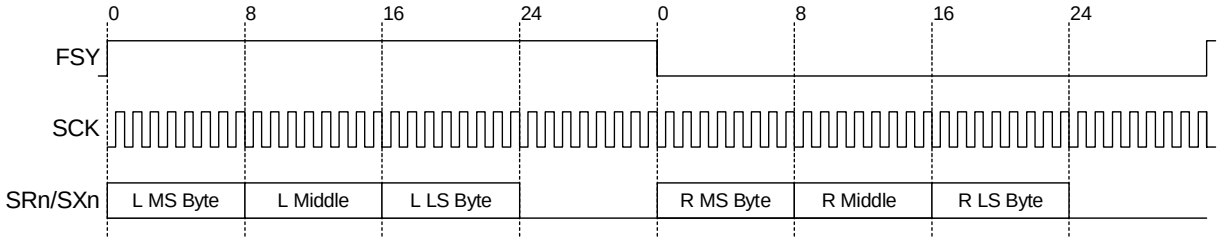


Figure 7-7: Matsushita Source Data Format

In Figure 7-7, data is arranged MSB first, high byte first.

7.2.2 bSDC2 (Source Data Control Register 2)

The bSDC2 register contains the control bits for Source Port **SCK** rate, Transparent Channel clock rate, Multi-speed **FSY**, and the S/PDIF speed rate.

8Ch bSDC2 Source Data Control 2 Register

Bit	Name	Description	Default
7..5	SPR[2:0]	Source Port **SCK** rate	011
4	MFSY	Multi-speed **FSY** enable	0
3, 2	TCR[1:0]	Transparent channel clock rate	00
1, 0	SDR[1:0]	S/PDIF speed rate	00

Table 7-2: bSDC2 (Source Data Control Register 2)

SPR[2:0] **SCK** rate. These bits select the **SCK** clock (bit) rate of the Source Ports in serial mode. When the **SCK** rate is higher than 64Fs, the shift registers in the Source Ports are cascaded, thus decreasing the number of active Source Port pins. When the Source Port is configured for parallel operation, the **SPR[2:0]** bits must be set to 101.
000 — 8Fs
001 — 16Fs
010 — 32Fs
011 — 64Fs
100 — 128Fs. Ports **SR1**, **SX1**, **SR3**, and **SX3** are not available.
101 — 256Fs. Ports **SR[3:1]** and **SX[3:1]** are not available.

MFSY Multi-speed **FSY** enable. The **MFSY** bit depends on the **SCK** clock rate (set via **SPR[2:0]**), and is only enabled at 128Fs or 256Fs, when **FSY** is an output. At **SCK** rates below 128Fs, the **FSY** output is always 1Fs. As an input, **FSY** must always be 1Fs.
If the **MFSY** bit is set and **SCK** rate is set to 128Fs, **FSY** is 2Fs.
If the **MFSY** bit is set and **SCK** rate is set to 256Fs, **FSY** is 4Fs.

TCR[1:0] Transparent Channel **SCK** divider. The **TCR[1:0]** bits select the sample rate of **SX1** and **SR1** when configured in transparent mode (**bSDC1.$\overline{\text{TCE}}$** cleared). The **TCR[1:0]** bits control a clock divider that scales down the selected **SCK** clock rate (**SPR[2:0]**). Transparent mode is not available when **SPR[2:0]** is 128Fs or 256Fs.
00 — **SCK** divided by 1
01 — **SCK** divided by 2
10 — **SCK** divided by 4
11 — **SCK** divided by 8

SDR[1:0] S/PDIF speed rate. These bits specify the speed (and bit rate) at which Source Port 0 is running when configured in S/PDIF mode (**bSDC1.SPD** set). At clock rates higher than 64Fs (double-speed S/PDIF and faster), the shift registers in the Source Ports are cascaded, thus decreasing the number of active Source Port pins.
00 — 64Fs (1x)
01 — 128Fs (2x). Ports **SR1** and **SX1** are not available
10 — 256Fs (4x). Ports **SR[3:1]** and **SX[3:1]**are not available
11 — 512Fs (8x). Ports **SR[3:1]** and **SX[3:1]** are not available. In addition, the S/PDIF stream is in only one direction (in or out), specified by **bSDC3.SIO**.

7.2.3 bSDC3 (Source Data Control Register 3)

The bSDC3 register contains the control bits for S/PDIF operation.

8Dh bSDC3 Source Data Control 3 Register

Bit	Name	Description	Default
7	SIO	S/PDIF 8x mode direction (either input or output)	0
6..4	rsvd	Reserved; Write as 0	000
3	SPS	S/PDIF sync source	0
2..0	rsvd	Reserved; Write as 0	000

Table 7-3: bSDC3 (Source Data Control Register 3)

SIO S/PDIF 8x mode direction. When 8x S/PDIF is selected (**SDR[1:0]** = 11), **SIO** cleared supports S/PDIF data in on **SR0**, and **SIO** set supports S/PDIF data out on **SX0**.

SPS S/PDIF sync source. When cleared, the S/PDIF output is synchronized to the S/PDIF input data stream. For more information, see Section 7.4.1 on page 55.

7.3 Serial Source Port Modes

The OS8104 provides 11 serial modes, which differ in bit rates and number of available ports. In the following descriptions, *Port 0* indicates **SR0** and **SX0**, except where **ot**herwise stated. Table 7-4 shows an overview of the eleven serial modes:

Mode	Comment	
1	Ports 0,1,2,3	in serial port format; **SCK** bit rate 8Fs, 16Fs, 32Fs or 64Fs
2	Ports 0,2,3	in serial port format; **SCK** bit rate 8Fs, 16Fs, 32Fs or 64Fs
	Port 1	in transparent mode; Sample rate = (**SCK** bit rate) / div [div = 1, 2, 4, or 8]
3	Ports 0, 2	in serial port format; **SCK** bit rate 128Fs
4	Port 0	in serial port format; **SCK** bit rate 256Fs
5	Port 0	in S/PDIF format (1x speed)
	Ports 1,2,3	in serial port format; **SCK** bit rate 8Fs, 16Fs, 32Fs or 64Fs
6	Port 0	in S/PDIF format (1x speed)
	Port 1	in transparent mode; Sample rate = (**SCK** bit rate) / div [div = 1, 2, 4, or 8]
	Ports 2,3	in serial port format; **SCK** bit rate 8Fs, 16Fs, 32Fs or 64Fs
7	reserved	
8	Port 0	in S/PDIF format (2x speed)
	Ports 2,3	in serial port format; **SCK** bit rate 8Fs, 16Fs, 32Fs or 64Fs
9	Port 0	in S/PDIF format (2x speed)
	Port 2	in serial port format; **SCK** bit rate 128Fs
10	Port 0	in S/PDIF format (4x speed)
11	Port 0	in S/PDIF format (8x speed) as input (**SR0** only)
12	Port 0	in S/PDIF format (8x speed) as output (**SX0** only)

Table 7-4: Serial Source Port Modes

The term *serial port format* stands for an industry-standard format such as I^2S or Sony, configured using bits in bSDC1.

For the Serial Source Port modes described in the following tables, Table 7-5 lists the symbols and abbreviations used in the descriptions. Table 7-6 gives an overview of the mode configuration registers.

Symbol	Comment
NnnnFs	Represents a serial port format (I^2S, Sony, etc., selected by bSDC1) and the **SCK** bit rate 'nnn' of the particular port.
N(a)	Represents a serial port format (I^2S, Sony, etc., selected by bSDC1). The **SCK** bit rate of the port is determined by 'a': **bSDC2.SPR[2:0]** 000 001 010 011 a = 8Fs 16Fs 32Fs 64Fs
Snx	S/PDIF format at a speed of 'n', where n = 1, 2, 4, or 8
T(b)	Represents a transparent format with a sample rate of **SCK** bit rate divided by 'b': **bSDC2.TCR[1:0]** 00 01 10 11 b = 1 2 4 8
---	Port not available
x	Don't care

Table 7-5: Symbols and Abbreviations

The mapping of the respective bytes from the serial Source Port pins to and from the MOST Network is described in Chapter 12 on page 97.

Source Port Format Combinations										
Mode	Ports				Control Registers					
					bSDC1		bSDC2			bSDC3
	SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
1	N(a)	N(a)	N(a)	N(a)	0	1	a	x	x	x
2	N(a)	T(b)	N(a)	N(a)	0	0	a	b	x	x
3	N128Fs	---	N128Fs	---	0	x	100	x	x	x
4	N256Fs	---	---	---	0	x	101	x	x	x
5	S1x	N(a)	N(a)	N(a)	1	1	a	x	00	x
6	S1x	T(b)	N(a)	N(a)	1	0	a	b	00	x
7	reserved									
8	S2x	---	N(a)	N(a)	1	x	a	x	01	x
9	S2x	---	N128Fs	---	1	x	100	x	01	x
10	S4x	---	---	---	1	x	x	x	10	x
11	S8x in	---	---	---	1	x	x	x	11	0
12	S8x out	---	---	---	1	x	x	x	11	1

Table 7-6: Serial Source Port Modes vs. Registers

7.3.1 Source Port Mode 1

Ports				Control Registers					
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	bSDC1		bSDC2			bSDC3
				SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
N(a)	N(a)	N(a)	N(a)	0	1	a	x	x	x

Table 7-7: Source Port Mode 1 Register Settings

In Mode 1, all four Source Ports are configured for the same serial data format and SCK rate. The selection of a serial data format is described in Section 7.2.1. N(a) indicates that the SCK rate is specified by setting **bSDC2.SPR[2:0]** (shown in Table 7-8). This mode is enabled by clearing **bSDC1.SPD** and setting **bSDC.$\overline{\text{TCE}}$**.

a	8Fs	16Fs	32Fs	64Fs
SPR[2:0]	000	001	010	011

Table 7-8: Source Port SCK Rates

7.3.2 Source Port Mode 2

Ports				Control Registers					
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	bSDC1		bSDC2			bSDC3
				SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
N(a)	T(b)	N(a)	N(a)	0	0	a	b	x	x

Table 7-9: Source Port Mode 2 Register Settings

In Mode 2, the Source Ports 0, 2 and 3 are configured for the same serial data format and SCK rate. The selection of a serial data format is described in Section 7.2.1. N(a) indicates that the SCK rate is specified by setting **bSDC2.SPR[2:0]** (shown in Table 7-8). This mode is enabled by clearing **bSDC1.SPD** and **bSDC1.$\overline{\text{TCE}}$**.

Source Port 1 is configured for Transparent mode, where the sample clock rate is set via **bSDC2.SPR[2:0]** (shown in Table 7-8) and **bSDC2.TCR[1:0]** (shown in Table 7-10).

Scale factor b	1	2	4	8
TCR[1:0]	00	01	10	11

Table 7-10: Source Port Transparent Channel Clock Rate

The sample clock rate is calculated as follows:

$$\text{sample clock rate} = \frac{a}{b} = \frac{\textbf{bSDC2.SPR[2:0]} \text{ setting}}{\textbf{bSDC2.TCR[1:0]} \text{ setting}}$$

For an overview of Transparent mode, see Section 7.5.

7.3.3 Source Port Mode 3

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
N128Fs	---	N128Fs	---	0	x	100	x	x	x

Table 7-11: Source Port Mode 3 Register Settings

In Mode 3, Source Ports 0 and 2 are configured for the same serial data format at a fixed **SCK** rate of 128Fs. Section 7.2.1 describes the selection of a serial data format. Source Ports 1 and 3 are not available. This mode is enabled by clearing **bSDC1.SPD** and setting **bSDC2.SPR[2:0]** to 100.

7.3.4 Source Port Mode 4

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
N256Fs	---	---	---	0	x	101	x	x	x

Table 7-12: Source Port Mode 4 Register Settings

In Mode 4, only Source Port 0 is operational, at an **SCK** rate of 256Fs. Section 7.2.1 describes the selection of a serial data format. Source Ports 1 through 3 are not available. This mode is enabled by clearing **bSDC1.SPD** and setting **bSDC2.SPR[2:0]** to 101.

7.3.5 Source Port Mode 5

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3S/X3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S1x	N(a)	N(a)	N(a)	1	1	a	x	00	x

Table 7-13: Source Port Mode 5 Register Settings

In Mode 5, Source Port 0 is configured for S/PDIF format, at single S/PDIF speed. Source Ports 1 through 3 are configured for the same serial data format and **SCK** rate. Section 7.2.1 describes the selection of a serial data format. N(a) indicates that the **SCK** rate is specified by setting **bSDC2.SPR[2:0]** (shown in Table 7-8). This mode is enabled by setting **bSDC1.SPD** and **bSDC1.$\overline{\text{TCE}}$**, and setting the **bSDC2.SDR[1:0]** bits to 00.

7.3.6 Source Port Mode 6

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S1x	T(b)	N(a)	N(a)	1	0	a	b	00	x

Table 7-14: Source Port Mode 6 Register Settings

In Mode 6, Source Port 0 is configured for S/PDIF format, at single S/PDIF speed. Source Ports 2 and 3 are configured for the same serial data format and SCK rate. Section 7.2.1 describes the selection of a serial data format. N(a) indicates that the SCK rate is specified by setting **bSDC2.SPR[2:0]** (shown in Table 7-8). This mode is enabled by setting **bSDC1.SPD**, and clearing the **bSDC1.$\overline{\text{TCE}}$** and **bSDC2.SDR[1:0]** bits.

Source Port 1 is configured for Transparent mode, where the sample clock rate is set via **bSDC2.SPR[2:0]** (shown in Table 7-8) and **bSDC2.TCR[1:0]** (shown in Table 7-10).

The sample clock rate is calculated as follows:

$$\text{sample clock rate} = \frac{a}{b} = \frac{\textbf{bSDC2.SPR[2:0]}\ \text{setting}}{\textbf{bSDC2.TCR[1:0]}\ \text{setting}}$$

For an overview of Transparent mode, see Section 7.5.

7.3.7 Source Port Mode 8

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S2x	---	N(a)	N(a)	1	x	a	x	01	x

Table 7-15: Source Port Mode 8 Register Settings

In Mode 8, Source Port 0 is configured for S/PDIF format at double S/PDIF speed. Source Port 1 is not available. Source Ports 2 and 3 are configured for the same serial data format and SCK rate. The selection of a serial data format is described in Section 7.2.1. N(a) indicates that the SCK rate is determined by **bSDC2.SPR[2:0]** (shown in Table 7-8). This mode is enabled by setting **bSDC1.SPD** and setting the **bSDC2.SDR[1:0]** bits to 01.

7.3.8 Source Port Mode 9

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S2x	---	N128Fs	---	1	x	100	x	01	x

Table 7-16: Source Port Mode 9 Register Settings

In Mode 9, Source Port 0 is configured for S/PDIF format, at double S/PDIF speed. Source Ports 1 and 3 are not available. Source Port 2 operates in the standard serial data formats at a fixed SCK rate of 128Fs. Section 7.2.1 describes the selection of a serial data format. This mode is enabled by setting **bSDC1.SPD** and setting the **bSDC2.SPR[2:0]** and **bSDC2.SDR[1:0]** bits to 100 and 01, respectively.

7.3.9 Source Port Mode 10

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0/SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S4x	---	---	---	1	x	x	x	10	x

Table 7-17: Source Port Mode 10 Register Settings

In Mode 10, Source Port 0 is configured for S/PDIF format, at quadruple S/PDIF speed. Source Ports 1 through 3 are unavailable. This mode is enabled by setting **bSDC1.SPD** and setting the **bSDC2.SDR[1:0]** bits to 10.

7.3.10 Source Port Mode 11

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SR0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S8x in	---	---	---	1	x	x	x	11	0

Table 7-18: Source Port Mode 11 Register Settings

In Mode 11, Source Port 0 is configured for S/PDIF format at octal S/PDIF speed, and can only receive S/PDIF data on **SR0** (**SX0** not available). Source Ports 1 through 3 are unavailable. This mode is enabled by setting **bSDC1.SPD**, setting the **bSDC2.SDR[1:0]** bits to 11, and clearing **bSDC3.SIO**.

7.3.11 Source Port Mode 12

Ports				Control Registers					
				bSDC1		bSDC2			bSDC3
SX0	SR1/SX1	SR2/SX2	SR3/SX3	SPD	$\overline{\text{TCE}}$	SPR[2:0]	TCR[1:0]	SDR[1:0]	SIO
S8x out	---	---	---	1	x	x	x	11	1

Table 7-19: Source Port Mode 12 Register Settings

In Mode 12, Source Port 0 is configured for S/PDIF format at octal S/PDIF speed, and can only transmit S/PDIF data on **SX0** (**SR0** not available). Source Ports 1 through 3 are unavailable. This mode is enabled by setting **bSDC1.SPD** and **bSDC3.SIO**, and setting bits **bSDC2.SDR[1:0]** to 11.

7.4 S/PDIF (IEC-60958)

The OS8104 can handle S/PDIF data from single-speed mode to 8x speed mode through Source Ports **SR0** and **SX0**.

S/PDIF mode is enabled by setting **bSDC1.SPD**. When enabled, the **FSY** and **SCK** pins are always configured as outputs. The diagram below shows single-speed S/PDIF data, and how **FSY** is aligned to the data:

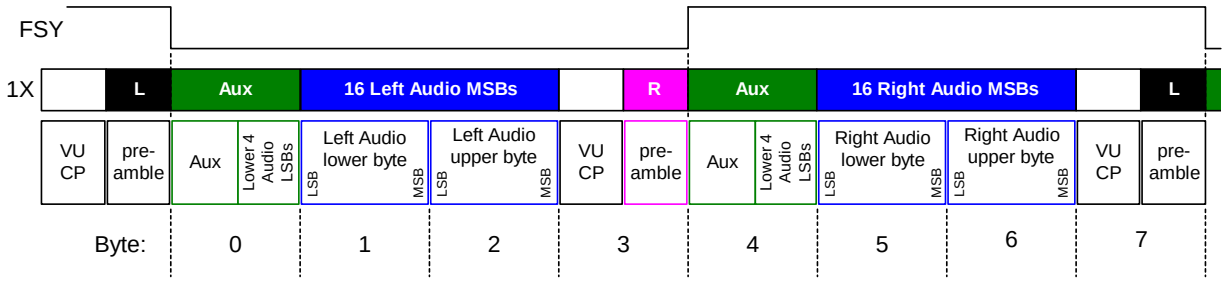


Figure 7-8: Standard S/PDIF Data Stream

Audio data in S/PDIF format is transported LSB first. The OS8104 changes the sequence of bits, within a byte, automatically to MSB first when receiving S/PDIF data via **SR0**. However, the data byte order is Low Byte/High Byte in the input buffer, so the MRT must be adapted to change the sequence on a byte level. Incoming single-speed S/PDIF data is handled like any serial data received at 64Fs **SCK** clock rate. The respective address references (used for routing) can be derived from the tables in Chapter 12 on page 97.

The VUC bits of the S/PDIF data stream and the respective preambles are received in the same manner as other data. This makes it possible to receive S/PDIF data, transport the entire stream via the MOST Network, and restore the S/PDIF data at a different node.

The OS8104 provides three ways to process S/PDIF data:

- S/PDIF data to S/PDIF data
  Receiving S/PDIF data, transmitting the entire data stream to another node, restoring of the original S/PDIF data stream at the destination node, and outputting the data stream on the **SX0** pin in S/PDIF mode.
- S/PDIF data to non-S/PDIF data
  Receiving S/PDIF data and transmitting only the audio related parts of the data stream.
- Non-S/PDIF data to S/PDIF data
  Routing any non-S/PDIF audio data to the **SX0** pin and outputting the data as an S/PDIF data stream. Dummy VUC and S/PDIF timing are generated by the OS8104 automatically.

7.4.1 Synchronizing To S/PDIF

When configured as the timing-master node (**bXCR.MTR** set), the PLL can be synchronized to the incoming S/PDIF data stream. Since the timing-master node generates the timing for the entire Network, all nodes will be synchronized to that S/PDIF data stream as well.

As an alternative, the timing-master's PLL can be locked to a crystal connected to the **XTI/XTO** pins. In this scenario, the source of S/PDIF data at the input of the timing-master node must be synchronized to the timing-master's clock by using the **RMCK** output pin.

When configured as a timing-slave node, the Network input (**RX**) must be chosen as the synchronization source. An S/PDIF device connected to a timing-slave node must always be synchronized to the Network by using the **RMCK** output pin.

Setting the **bSDC3.SPS** bit separates the internal timing generator from the S/PDIF data stream at **SR0**. Otherwise the timing is synchronized to the S/PDIF data stream at **SR0**. Using the **bSDC3.SPS** bit can resolve two issues:

- If an external S/PDIF device synchronizes its output to the incoming S/PDIF data and the same is done within the MOST transceiver, a closed loop is generated. No S/PDIF transfer is possible, since both devices keep trying to re-synchronize their frame positions. This closed loop is avoided by setting the **bSDC3.SPS** bit, illustrated in Figure 7-9.
- A corrupt signal at the S/PDIF input will corrupt S/PDIF output as well. Switching synchronization from S/PDIF input to internal synchronization by setting the **bSDC3.SPS** bit solves this issue as well.

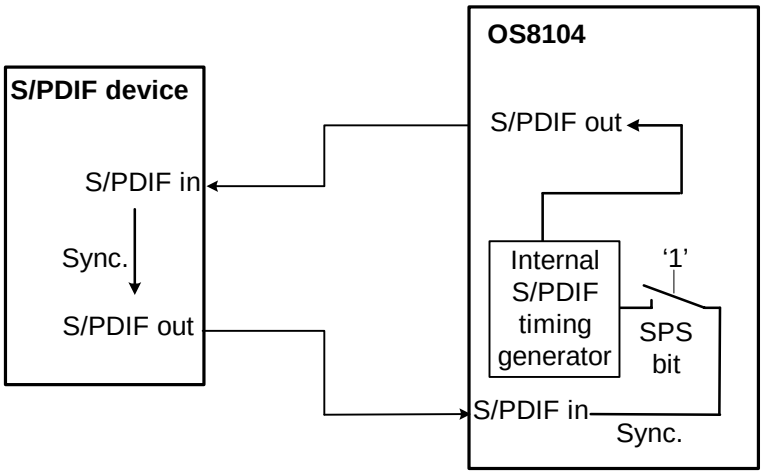


Figure 7-9: SPS Bit Block Diagram

7.4.2 Routing S/PDIF Data

Source data is transmitted across the MOST Network MSB-first. Since S/PDIF data is transmitted LSB-first, each byte is bit-reversed before transmission. Therefore, data bytes received from **SR0** are bit-reversed when received. Likewise, bytes are bit-reversed before transmitting out the **SX0** pin.

The S/PDIF receiver checks for parity and bi-phase coding errors. If an error occurs, the validity bit (V) of the erred sub-frame is automatically set.

- MOST Network synchronous data that is greater than one byte (channel) generally places the most significant byte in the lowest MRT location. To place the 16 most significant bits of S/PDIF left data from the received **SR0** pin on the MOST Network, channels 0 and 1 and the right S/PDIF data on channels 2 and 3; the received-S/PDIF most-significant-byte address 50h would be placed in the MRT table location 00h and the middle-byte address 48h would be placed in the MRT location 01h. For the right channel, the received-S/PDIF address 70h would be placed in MRT location 02h, and S/PDIF address 68h placed in MRT location 03h.

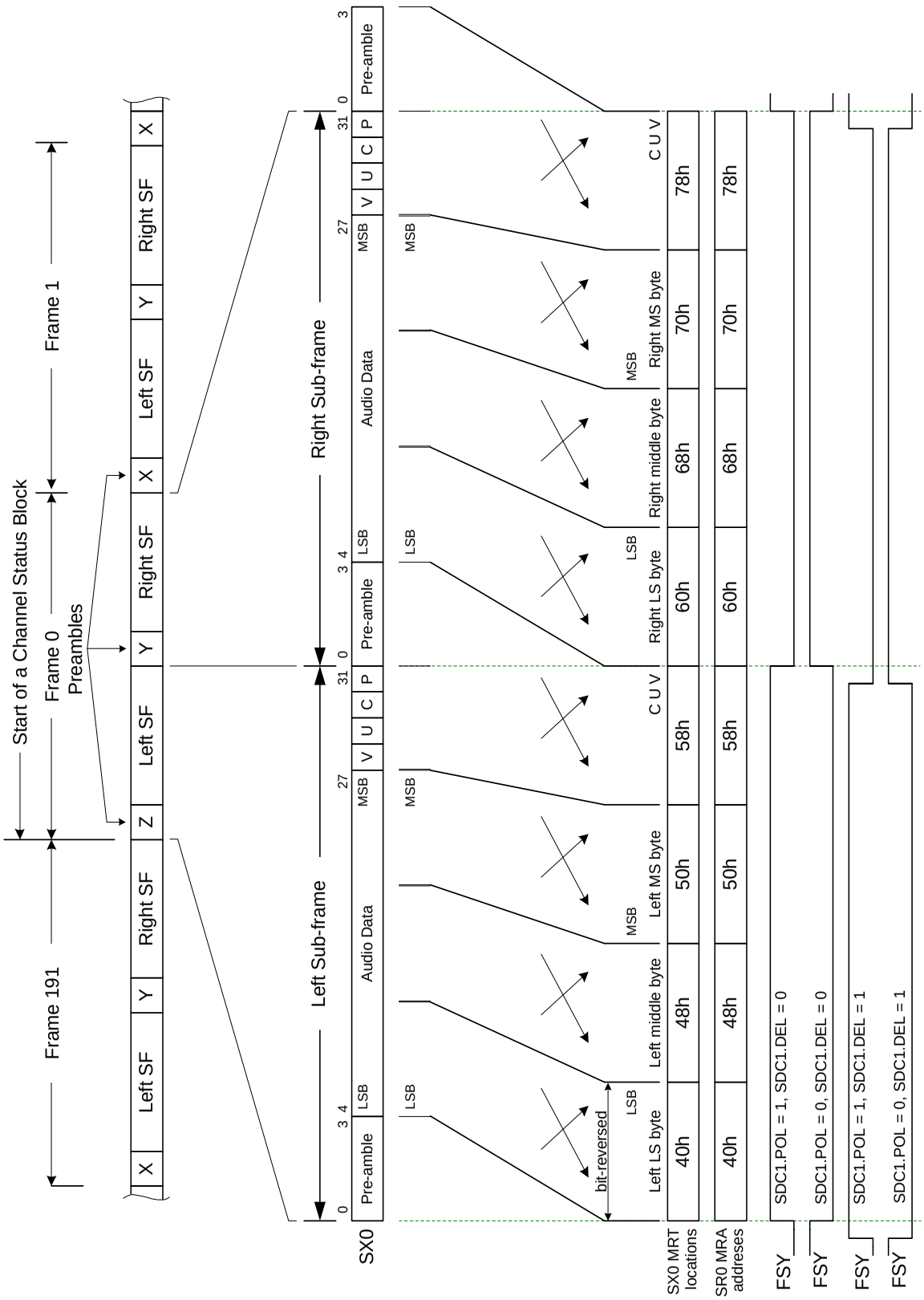


Figure 7-10: Source Port S/PDIF Format MRT/MRA

7.4.3 S/PDIF Speed Modes

The OS8104 supports several different S/PDIF speeds. Table 7-20 lists the possible S/PDIF speed selections, using **bSDC2.SDR[1:0]**.

bSDC2.SDR[1:0]	Speed	Bytes/Frame	Clock rate [Fs]
00	1x	8	64
01	2x	16	128
10	4x	32	256
11	8x	64	512

Table 7-20: S/PDIF Mode Speeds

If 8x-S/PDIF mode is selected, all the Source Port shift registers (in both directions) are cascaded; therefore, only one direction can be supported (input or output). Clearing **bSDC3.SIO** configures **SR0** as 8x-S/PDIF *input only* and setting **bSDC3.SIO** configures **SX0** as 8x-S/PDIF *output only*. **SIO** is not used for S/PDIF speeds slower than 8x, where **SR0** and **SX0** are both available as Source Ports. Routing information for S/PDIF data can be found in the tables of Chapter 12 on page 97.

In S/PDIF modes, the **FSY** signal is aligned to the first data byte after the left preamble, as illustrated in Figure 7-11.

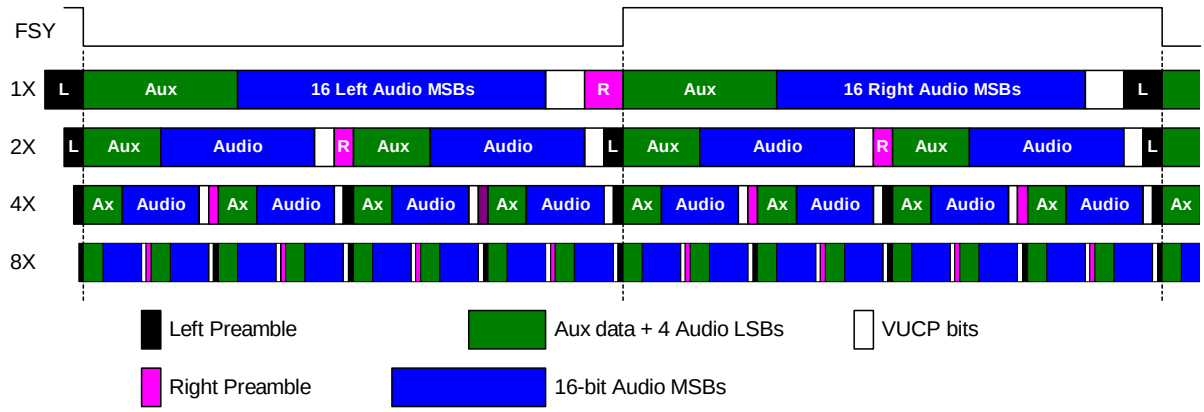


Figure 7-11: FSY – S/PDIF Alignment

7.4.4 S/PDIF Data To S/PDIF Data

When transmitting S/PDIF data on a MOST Network, the VUC data bits and the preambles must also be transmitted if the final data output at the receiver node is to be in the S/PDIF format.

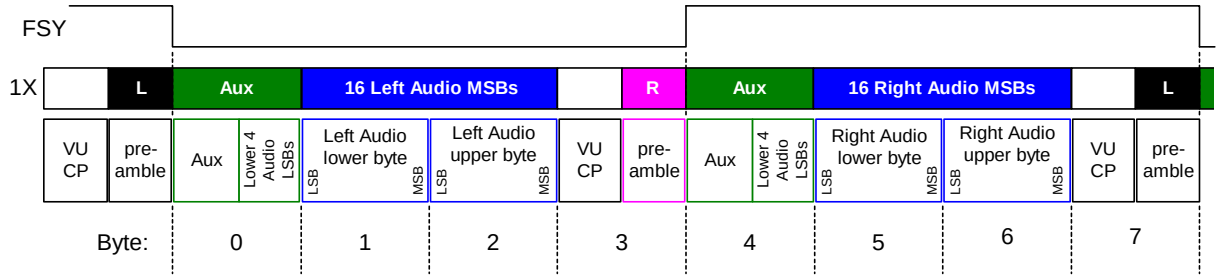


Figure 7-12: S/PDIF Byte Sequence

At the target node, the correct sequence of bytes must be restored. Although changing the order of audio data bytes is not absolutely necessary when sending pure S/PDIF data from an S/PDIF source to an S/PDIF sink, the byte order, within a word or channel, should be reversed on the MOST Network to conform to the *MOST Specification*. Then any other node may use the data traversing the MOST Network, even if they are not outputting S/PDIF streams. The MOST Network standard is left channel before right (same as S/PDIF). However, the MOST standard is MSbyte first, which is opposite to S/PDIF. This byte-reversal is handled easily through re-configuration of the MRT. As previously mentioned, the bit reversal of each byte in the S/PDIF stream is handled automatically by the OS8104 (S/PDIF data bytes are LSB first and the MOST standard is MSB first).

7.4.5 S/PDIF Data To Non-S/PDIF Data

When building a MOST standard data stream based upon data from a Source Port in S/PDIF mode, the preambles and VUC bits can be omitted. In this scenario, only the bytes listed below must be transmitted in the given sequence:

- 16-bit audio: byte 2, byte 1; then byte 6, byte 5
- 24-bit audio: byte 2, byte 1, byte 0; then byte 6, byte 5, byte 4

where the byte numbers refer to Figure 7-12.

7.4.6 Non-S/PDIF Data To S/PDIF Data

When building an S/PDIF data stream based upon audio data coming from a Source Port (running in one of the standard MOST modes), no S/PDIF specific preambles (left, right and block) or VUC bits are transmitted with the data. In this case, the OS8104 automatically adds these bits to the outgoing S/PDIF stream.

The non-S/PDIF data must be routed to the respective byte positions onto Source Port **SX0**, by configuring the MRT appropriately. For setting the VUC bits to a defined value of 000, the address reference MRA0xF8 must be placed at the appropriate location in the MRT. MRA0xF8 is a special address reference that outputs 00h for that particular byte (see Chapter 12 for more information on the MRT). The OS8104 will automatically insert the correct channel and block boundary preambles.

All of the C bits will be set to zero. The majority of S/PDIF receiving devices will properly accept this data; however, this does not comply with the IEC-60958 specification.

7.5 Transparent Data Transport

Two of the Source Port pins (**SR1**, **SX1**) can be configured for transparent mode. In this mode, the input port **SR1** samples the incoming signal at regular intervals. The sample rate depends on the current **SCK** rate and an additional scale factor. By adjusting the **SCK** rate and scale factor, the appropriate sample rate for a given application can be selected. Since the sampled data is handled via the standard **SR1** and **SX1** ports, it can be routed onto and off of the Network similar to the other Source Ports (using the MRT).

The following diagram shows the effect of the sampling on a transmitted signal.

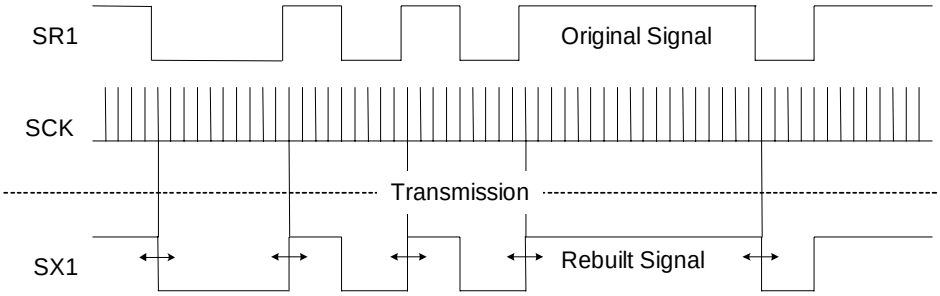


Figure 7-13: Transparent Signal Transmission

The Network source data channel capacity used depends on the over-sampling ratio. Over-sampling ratios can be programmed with respect to the signal frequency and the maximum acceptable jitter on the receiver's **SX1** port, with allowable values between 1Fs and 64Fs. This transparent transport mechanism can handle different application protocols and formats such as RS-232, DAB, ITTS, GSM AT, GPS, ISDN, and UNILINK.

8 Parallel Access

Both the Source Ports (SP) and the Control Port (CP) can be configured for parallel operation. The Source Port can be configured for parallel mode while the Control Port still operates in serial mode; however, to use the Control Port in parallel mode, the Source Ports must be in parallel mode. The **bSDC2.SPR[2:0]** bits must be set to 101 when the Source Ports are configured for parallel mode.

In parallel mode, the Source Port pins (**SR[3:0]** and **SX[3:0]**) form an 8-bit parallel interface, and are defined as **D[7:0]**, as shown in Table 8-1. All **D[7:0]** lines should have pull-ups, if not driven.

D7	D6	D5	D4	D3	D2	D1	D0
SX1	SR1	SR3	SR2	SR0	SX3	SX2	SX0

Table 8-1: Source Port Pins Configured for 8-bit Parallel I/O

The Source Port shift registers are re-configured to an 8-byte FIFO. All source data is handled via this FIFO, with the exception of reading the status registers bCP and bSP. If the Control Port is configured for parallel operation, it is also independent of the source data FIFO.

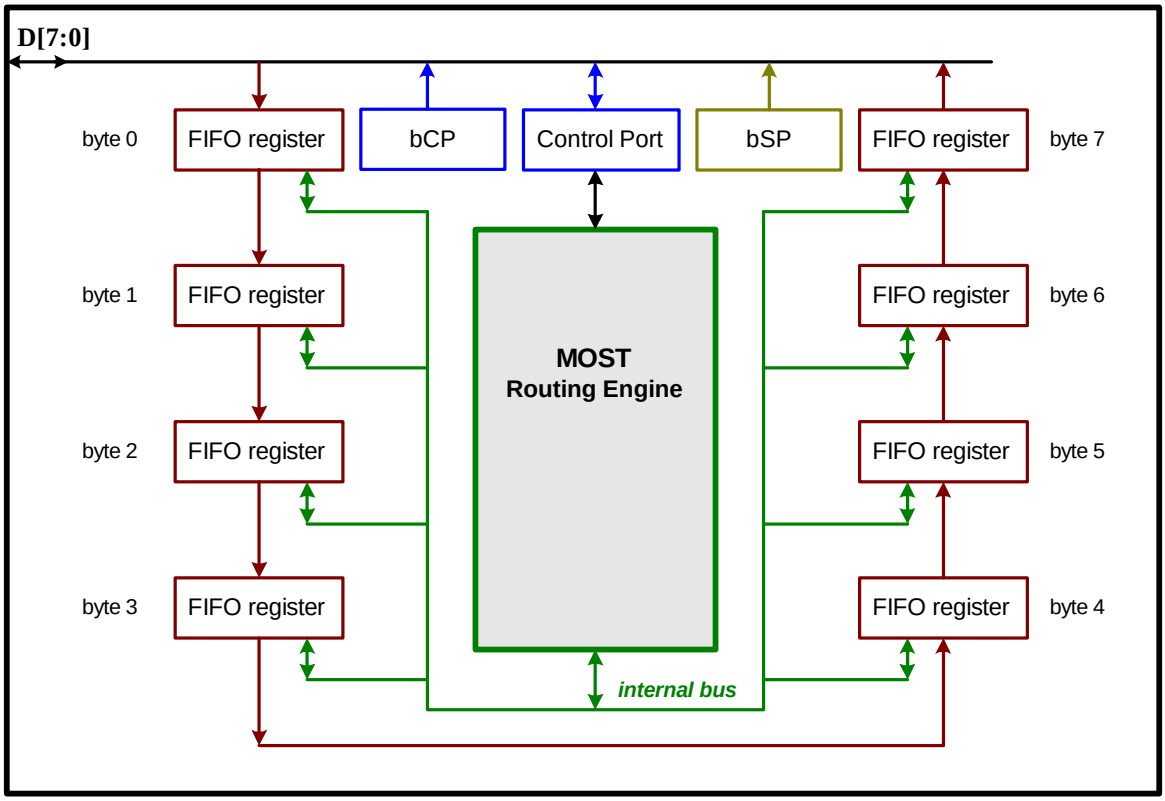


Figure 8-1: Source Ports (FIFO) in Parallel Mode

For source data, there are three parallel configuration options: Parallel-Synchronous, Parallel-Asynchronous, or Parallel-Combined (Physical) mode. Regardless of the particular Source Port parallel mode, the Control Port can also be configured to use the parallel port (or the Control Port can remain in serial mode).

- Parallel-Synchronous (Stream data)
  - Synchronous data handled in fixed time scheme controlled by **FSY** and **SRC_FLOW**
  - 48-byte Packet data support through the Control Port only

- Parallel-Asynchronous (Packet data)
  - No fixed time scheme
  - **SRC_FLOW** provides data transfer handshaking (OS8104 busy signal)
  - No synchronous data handling available

- Parallel-Combined, Physical mode (Packet and Stream)
  - Synchronous and packet data handled in fixed time scheme controlled by **FSY** and **SRC_FLOW**

All parallel port data formats use a common set of signals and registers. Control and data flow in parallel mode is controlled by the following pins:

- $\overline{\text{RD}}$ — Read access
- $\overline{\text{WR}}$ — Write access
- **PAD[1:0]** — Parallel address, determines the type of operation
- **CP_FLOW** — Control message handshaking (Control Port busy)
- **SRC_FLOW** — Fixed timing scheme for Parallel-Synchronous and Parallel-Combined modes, and packet data handshaking (port busy) for Parallel-Asynchronous mode.
- **FSY** — Synchronization to the Network frame in Parallel-Synchronous and Parallel-Combined modes

Status is available through special status registers that do not reside in the standard memory area of the OS8104 (similar to the other registers). These registers are only available through the parallel interface.

- bCP (Control Port parallel status)
- bSP (Source Port parallel status)

8.1 Control Port in Parallel Mode

The Control Port can only be configured in parallel mode, when the Source Port is also configured in parallel mode. Table 4-1 shows all available modes. The respective signals for configuring the Control Port in parallel mode are:

$\overline{\text{RS}}$	PAR_CP	PAR_SRC	Description
0	x	x	Chip is being reset
↑	1	1	Control Port in parallel mode

Table 8-2: Control Port Configuration Pins

The **PAD[1:0]** pins specify the type of operation to be performed, and $\overline{\text{RD}}$ and $\overline{\text{WR}}$ specify a read or write operation, respectively. Before the first data can be read or written, a MAP (Memory Address Pointer) byte must be written, which specifies where the following data will be read from or written to. Table 8-3 shows how various operations in parallel mode are selected.

PAD[1:0]	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Selected Operation
00	↓	1	Read Control Port Status Register bCP
00	1	↑	Write Control Port MAP Data Register
01	↓	1	Read Control Port Data Register
01	1	↑	Write Control Port Data Register

Table 8-3: Control Port Control Signal Overview in Parallel Mode

The **CP_FLOW** signal indicates when the chip is ready to receive/transmit data via the parallel port. The Control Port Status register (bCP) indicates the current status of the Control Port.

When the Control Port is configured in parallel mode, tie serial Control Port pins high.

8.1.1 Writing to the Control Port MAP Data Register

The Control Port MAP data register contains the address that specifies where a read or write operation starts. The MAP value must be written before the first access to the Control Port. This operation is selected as shown in Table 8-4.

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
00	1	↑	Write Control Port MAP Data Register

Table 8-4: Writing to the Control Port MAP Data Register in Parallel Mode

Data is written into the register at the rising edge of $\overline{WR}$.

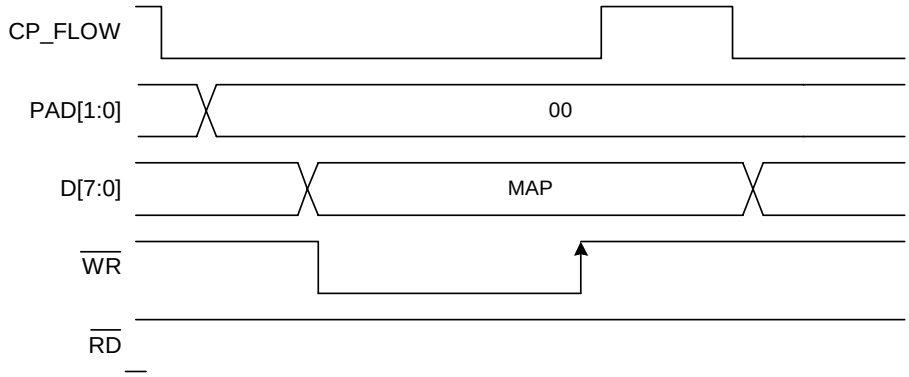


Figure 8-2: Control Port MAP Data Register Write Timing (Parallel Mode)

The MAP value only needs to be written the first time, as it is auto-incremented. As long as data is read from or written to sequential addresses within a memory page, a new MAP is not needed.

8.1.2 Writing to the Control Port

Before writing data to the Control Port, the destination address for the data must be specified by writing the MAP byte. The MAP specifies the address within the current memory page (default page is 0). To change to another memory page, the page number (0 to 3) must be written to memory location FFh.

The write operation is selected by setting the control pins as follows:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
01	1	↑	Write Control Port Data Register

Table 8-5: Writing to the Control Port in Parallel Mode

The data is internally latched at the rising edge of $\overline{WR}$. The first write cycle after the specification of MAP writes data directly to the target address specified by MAP. Each write access auto-increments MAP, so MAP only needs to be specified once when transferring large blocks of data to the same memory page.

Figure 8-3 shows a sample data transfer for writing N bytes. $\overline{\mathbf{RD}}$ is constantly high in this example:

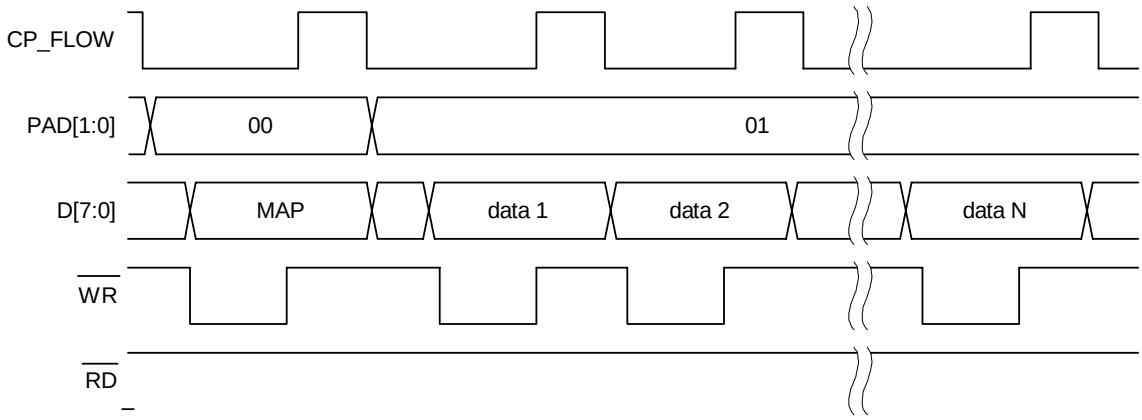


Figure 8-3: Control Port Write Timing (Parallel Mode)

When data is latched, **CP_FLOW** goes high indicating that the Control Port is busy (also indicated by **bCP.CP_BUSY[2:0]**). The Control Port is ready for another access when either **CP_FLOW** returns to zero, or all **bCP.CP_BUSY[2:0]** bits are clear.

8.1.3 Reading from the Control Port

Before reading data from the Control Port, the target address of the data must be specified by writing the MAP byte. The MAP specifies the address within the current memory page (default page is 0). To change to another memory page, the page number (0 to 3) must be written to memory location FFh.

The read operation is selected by setting the control pins as follows:

PAD[1:0]	$\overline{\mathbf{RD}}$	$\overline{\mathbf{WR}}$	Selected Operation
01	↓	1	Read Control Port Data Register

Table 8-6: Reading from the Control Port in Parallel Mode

Figure 8-4 shows the reading of N bytes:

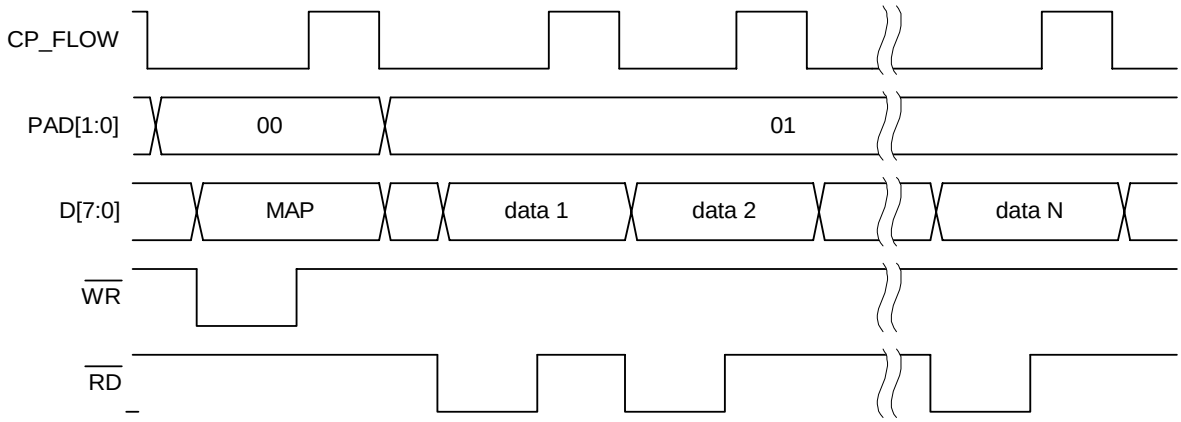


Figure 8-4: Control Port Read Timing (Parallel Mode)

When reading data, **CP_FLOW** goes high to indicate that the Control Port is busy (also indicated by **bCP.CP_BUSY[2:0]**). The Control Port is ready for another access when either **CP_FLOW** returns to zero, or all **bCP.CP_BUSY[2:0]** bits are clear.

8.1.4 bCP (Control Port Status Register)

Since bCP is handled independently of the FIFO, it is always available for reading, regardless of the state of **CP_FLOW**. For reading bCP, the following signals must be set:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
00	↓	1	Read Control Port Data Register bCP

Table 8-7: Reading Control Port Status in Parallel Mode (bCP)

bCP Control Port Status Register (in Parallel Mode, Read-Only)

Bit	Name	Description	Default
7	ZP	Zero-Power mode	0
6	LP	Low-Power mode	0
5	rsvd	Reserved	0
4	AINT	Asynchronous Packet interrupt (opposite polarity from $\overline{AINT}$ pin)	0
3	INT	Control and Error interrupt (opposite polarity from $\overline{INT}$ pin)	0
2..0	CP_BUSY[2:0]	Control Port busy (when non-zero)	000

Table 8-8: bCP (Control Port Status Register — Parallel Mode)

ZP Zero-Power Mode. When **ZP** is set, the chip is in Zero-Power mode.

LP Low-Power Mode. When **LP** is set, the chip is in Low-Power mode.

AINT Status of $\overline{AINT}$ pin used for Packet data transfer. The **AINT** bit set indicates either the reception or transmission of a data packet. **AINT** reflects an inverted version of the $\overline{AINT}$ pin. If an asynchronous packet is received or transmitted, **AINT** is set and the $\overline{AINT}$ pin is driven low.

INT Status of $\overline{INT}$ pin used for Control and error handling. When initially powered up, **INT** set indicates that the OS8104 is ready to be accessed. External hardware should wait for **INT** before any other accesses are made to the OS8104. **INT** reflects an inverted version of the $\overline{INT}$ pin. If a Control message has been received or an enabled error has occurred, **INT** is set and the $\overline{INT}$ pin is driven low. The bIE register enables the various error events that cause **INT** to be set.

CP_BUSY[2:0]
If any of these bits are set, the OS8104 is busy and cannot respond to Control Port accesses. If all three bits are clear, the Control Port is ready for another access.

8.2 Source Ports in Parallel Mode

The Source Ports support three parallel modes, two of which are selected with the Parallel-Synchronous (**PAR_SRC**) and Parallel-Asynchronous (**ASYNC**) configuration pins, as shown in Table 8-9. The third mode, Parallel-Combined, is derived from the Parallel-Synchronous mode and the setting of **bPCMA.APCM** (see Section 8.3).

$\overline{\text{RS}}$	PAR_SRC	ASYNC	Description
0	x	x	Chip is being reset
↑	0	x	Source Port (and Control Port) in serial mode
↑	1	0	Source Port in Parallel-Synchronous mode (or Parallel-Combined)
↑	1	1	Source Port in Parallel-Asynchronous mode

Table 8-9: Source Port Configuration Pins

In Parallel-Asynchronous mode, **SRC_FLOW** indicates whether the chip is ready to receive/transmit data via the parallel port. In Parallel-Synchronous and Parallel-Combined mode, **SRC_FLOW** defines frame intervals where synchronous data is read from and written to the FIFO. **SRC_FLOW** is active whenever the OS8104 is in lock state (**bCM2.$\overline{\text{LOK}}$** clear). **FSY** provides synchronization with the Network frame in Parallel-Synchronous and Parallel-Combined mode, and will synchronize to the MOST Network once lock is established.

The Source Port Status register (bSP) provides Source Port status when in parallel mode.

> Data is only transported via the Source Port interface when the chip is in the lock state. When the chip is in the unlock state (**bCM2.$\overline{\text{LOK}}$** set), synchronization to the Network is lost and no source data exchange is possible (control data can still be transferred).

PAD[1:0]	$\overline{\text{RD}}$	$\overline{\text{WR}}$	Selected Operation
10	↓	1	Read Source Port Status Register bSP
10	1	↑	Write MAP to FIFO / Write Data to FIFO (1 byte)
11	↓	1	Read Data from FIFO
11	1	↑	Write Data into FIFO (8 bytes)

Table 8-10: Source Port Control Signal Overview in Parallel Mode

8.2.1 Parallel-Synchronous Mode

In Parallel-Synchronous mode, synchronous data is transferred into and out of the chip via a FIFO. Synchronous data is transferred continuously; therefore, data transfer in this mode follows a fixed timing scheme.

Each Network frame period is divided into eight time intervals (SF0 to SF7), which are identified by **SRC_FLOW**. A falling edge on **FSY** indicates the start of SF0. Within each SF interval, eight bytes of synchronous data can be read and eight bytes can be written (16 bytes transferred in all). Therefore, a total of 64 bytes of synchronous data can be transferred in each direction during each Network frame period. Even if no data is needed for a particular SF interval, at least one read or write access is required during each SF interval. A routing address location is associated with each data byte and is determined by the SF interval and the FIFO byte position (see Chapter 12 on page 97).

In reset, **SRC_FLOW** is held high and will go low approximately 3 ms after $\overline{\text{RS}}$ is released. **SRC_FLOW** stays low until the OS8104 is in lock. Once in lock, **SRC_FLOW** and **FSY** determine the particular SF interval. If the OS8104 loses lock, **SRC_FLOW** goes low and stays low until lock is reacquired. **FSY** is synchronized to the MOST Network Frame; therefore, when lock is established, **FSY** will be resynchronized to the Network and can cause an abnormally short or long **FSY** pulse.

Just prior to the beginning of each SF interval, the Routing Engine fills the 8-byte FIFO with received data. At the rising edge of **SRC_FLOW**, data is ready to be read from the FIFO. Data must be read from the FIFO before writing to the FIFO, as writing to the FIFO overwrites the read data. If received data is not needed, the FIFO can be written immediately after **SRC_FLOW** rises. There is no requirement to read or write all eight bytes; however, at least one byte must be accessed (read or written) during each SF interval.

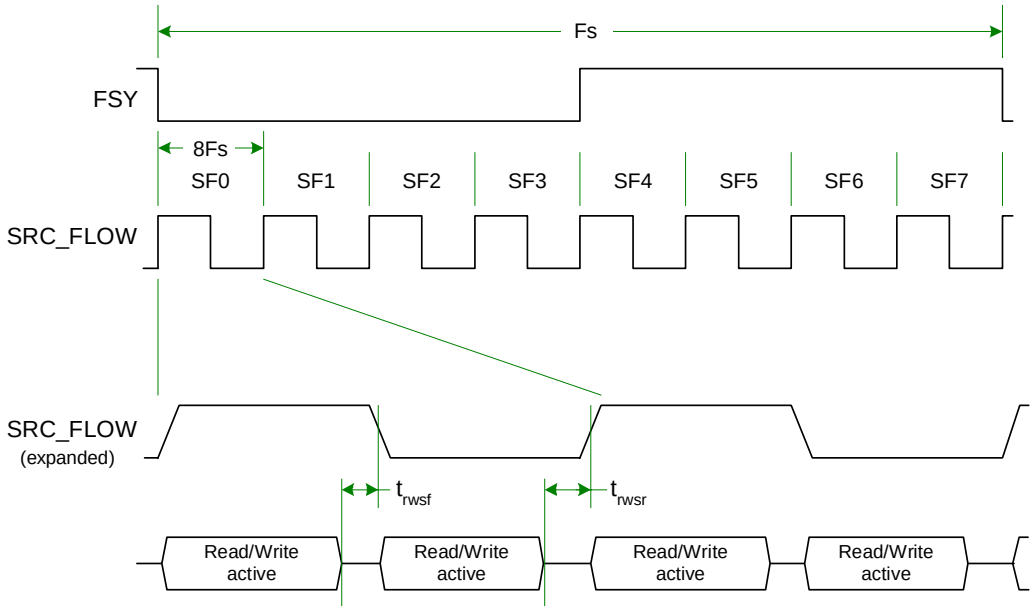


Figure 8-5: Source Port Parallel-Synchronous Mode Timing Overview

Accesses to the Source Port section of the parallel port must not be made t_{rwsf} before the falling edge of **SRC_FLOW** or t_{rwsr} before the rising edge of **SRC_FLOW**. Violating these timing restrictions could corrupt data within the FIFO.

The **FSY** signal indicates the start of a frame and aligns the Routing Engine data with a particular SF interval. SF0 is defined as the first **SRC_FLOW** interval after the falling edge of **FSY**. The **SRC_FLOW** signal is time-aligned with the **FSY** signal and toggles at an 8Fs rate.

8.2.1.1 Reading from the FIFO

For reading data from the FIFO, the **PAD[1:0]** and $\overline{\textbf{WR}}$ pins must be high, as illustrated in Figure 8-6. A falling edge at $\overline{\textbf{RD}}$ starts the output of data:

PAD[1:0]	$\overline{\textbf{RD}}$	$\overline{\textbf{WR}}$	Selected Operation
11	↓	1	Read Data from FIFO

Table 8-11: Reading from the FIFO in Parallel-Synchronous Mode

Figure 8-6 shows the timing diagram for reading from the FIFO.

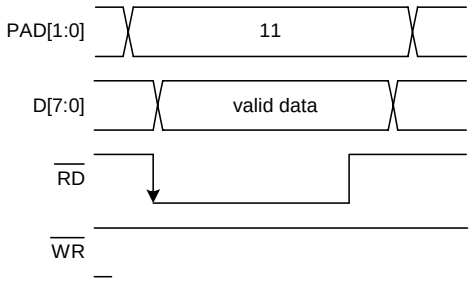


Figure 8-6: Source Port Read Timing (Parallel-Synchronous Mode)

8.2.1.2 Writing into the FIFO

For writing data into the FIFO, the **PAD[1:0]** and **$\overline{RD}$** pins must be high, as illustrated in Figure 8-7. A rising edge at **$\overline{WR}$** stores the data in the FIFO register:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
11	1	↑	Write Data into FIFO

Table 8-12: Writing into the FIFO in Parallel-Synchronous Mode

Figure 8-7 shows the timing diagram for writing into the FIFO.

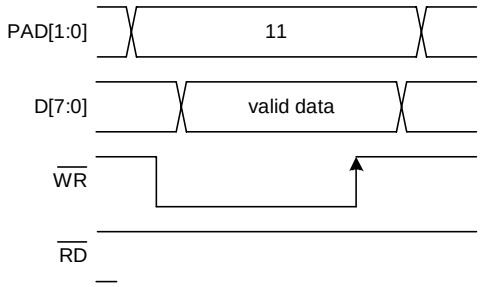


Figure 8-7: Source Port Write Timing (Parallel-Synchronous Mode)

8.2.2 Parallel-Asynchronous Mode

Parallel-Asynchronous data transfer is not restricted to a fixed timing scheme synchronized to the frame on the MOST Network. The Parallel-Asynchronous mode provides access to the registers of the OS8104. By sending an appropriate Memory Address Pointer (MAP), the asynchronous transmit and receive buffers can be read or written at a faster speed than can be accomplished through the Control Port. In this mode, **SRC_FLOW** is a hand-shaking signal that indicates when the FIFO is busy, or when it can be accessed.

When in Parallel-Asynchronous data transfer mode, access to synchronous data is not available. To access both synchronous and asynchronous data, see the Parallel-Combined/Physical transfer mode.

With the exception of the Source Port Status register (bSP), eight bytes must always be read or written. When the eight bytes are transferred, **SRC_FLOW** goes high. While **SRC_FLOW** is high, no parallel accesses (other than reading bSP) are permitted. When **SRC_FLOW** goes low, the FIFO is ready for another transfer.

The OS8104 provides three data transfer methods in Parallel-Asynchronous mode:
- Read data block of 8 bytes
- Write data block of 8 bytes
- Write a single data byte (8 bytes written, but only one byte is stored)

When transmitting packets, the **$\overline{AINT}$** pin goes low when a packet has been transmitted, the transmit status is available, and the Asynchronous Transmit Packet buffer (mAXP) is available. When receiving packets, the **$\overline{AINT}$** pin goes low when a valid packet has been received, where the entire packet is available in the Asynchronous Receive Packet buffer (mARP). A valid received packet is one for which the target address matches the logical address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL) of the receiving node, and the message has a valid CRC.

The OS8104 must be in lock (**bCM2.$\overline{LOK}$** clear) when writing asynchronous data. Otherwise, internal data could be corrupted when re-lock occurs.

8.2.2.1 Memory Address Pointer (MAP)

The start address must be written before any data accesses can occur. The start address, or memory address pointer (MAP), consists of two bytes: MAP1 and MAP2.

- MAP1 — Contains the offset within a page (00h to FFh).
- MAP2 — Contains the page (00h to 03h) and the control bit specifying a single byte transfer (bit 7). Therefore, when performing single byte transfers, 80h must be added to the memory page value.

For example, to write the first eight data bytes of the Asynchronous Transmit Packet buffer (mAXP) (locations 1C0h through 1F1h), the MAP value would be:

MAP1: C0h
MAP2: 01h

Byte(n) in the FIFO will be written to/read from the address location MAP + n. Therefore, byte0 – the first byte entered will be written to location 1C0h in the previous example.

To write only the first byte of mAXP, the MAP values would be:

MAP1: C0h
MAP2: 81h

Even though only the last byte is stored, eight bytes must still be written to the FIFO due to the hardware handshaking mechanism.

8.2.2.2 Writing into the FIFO

When the **SRC_FLOW** signal changes from high to low, the FIFO is ready to be accessed. Eight bytes must always be transferred even if only the last byte is actually stored.

The **PAD[1:0]** signals specify the type of operation to be performed, and $\overline{\textbf{RD}}$ and $\overline{\textbf{WR}}$ define a read or a write access, respectively. Before writing to the FIFO, either the **SRC_FLOW** signal must be low, or the **bSP.SRCF** bit must be zero.

When the **SRC_FLOW** signal is low, the **PAD[1:0]** signals select the operation, data is setup, and a rising edge of $\overline{\textbf{WR}}$ latches the data in the chip.

After eight write cycles, the **SRC_FLOW** signal changes from low to high, indicating that the FIFO is busy being processed. The **SRC_FLOW** signal stays high until the chip has finished processing the FIFO, at which time the **SRC_FLOW** signal goes low and more FIFO transfers can occur. As shown in Figure 8-8, the last data byte written to the FIFO is stored in *byte 0* of the FIFO. For more information about the internal structure of the FIFO, refer to Figure 8-1.

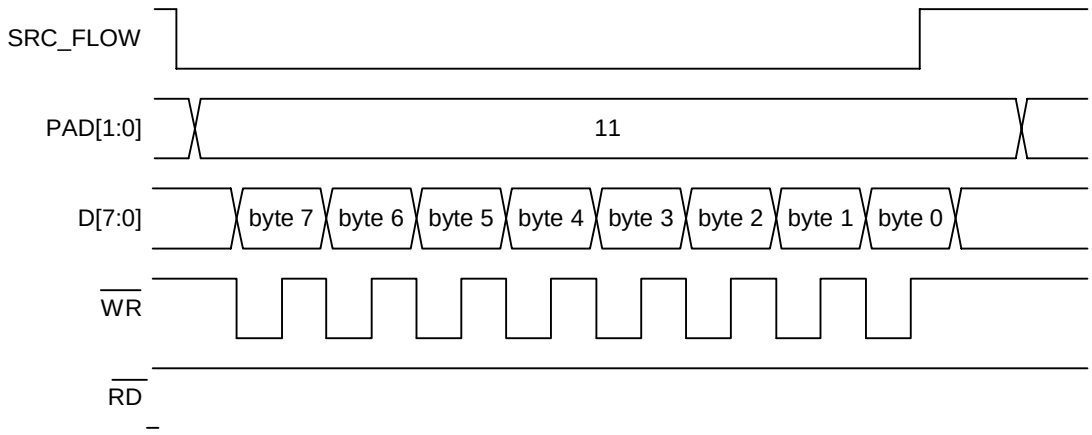


Figure 8-8: Source Port Parallel-Asynchronous Mode Write Timing Overview

Figure 8-9 illustrates how the bytes in FIFO are associated with the bytes in memory.

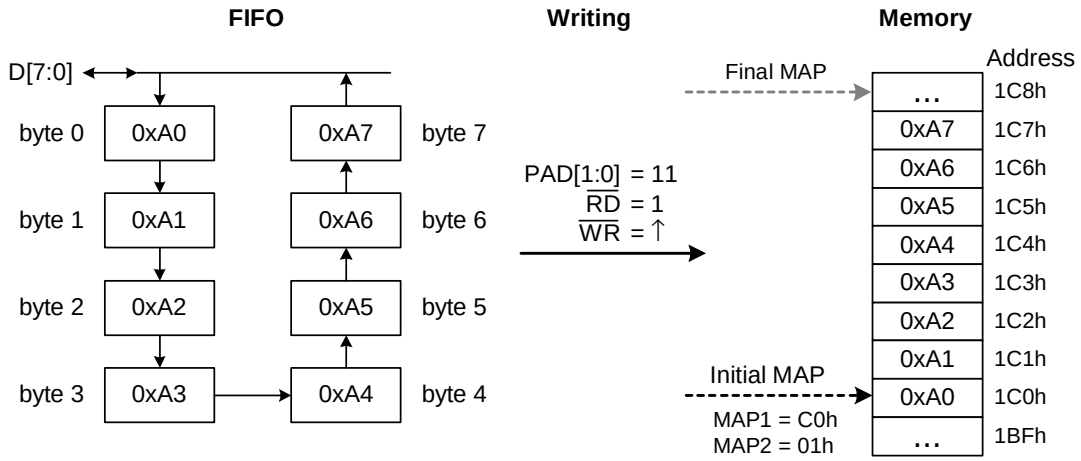


Figure 8-9: Source Port Parallel-Asynchronous Write Data Mapping Example

8.2.2.3 Writing the MAP

For writing the MAP value into the chip, the following signals must be set:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
10	1	↑	Write MAP to FIFO

Table 8-13: Writing MAP in Parallel-Asynchronous Mode

A rising edge at $\overline{WR}$ stores the data into the FIFO. The order of bytes (xx = don't care) within the FIFO is:

FIFO	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
Contents	MAP1	MAP2	xx	xx	xx	xx	xx	xx

Table 8-14: Contents of FIFO for Writing MAP

To achieve this order, six arbitrary bytes must be written to the FIFO first. Then MAP2 and finally MAP1 must be written as shown in Figure 8-10.

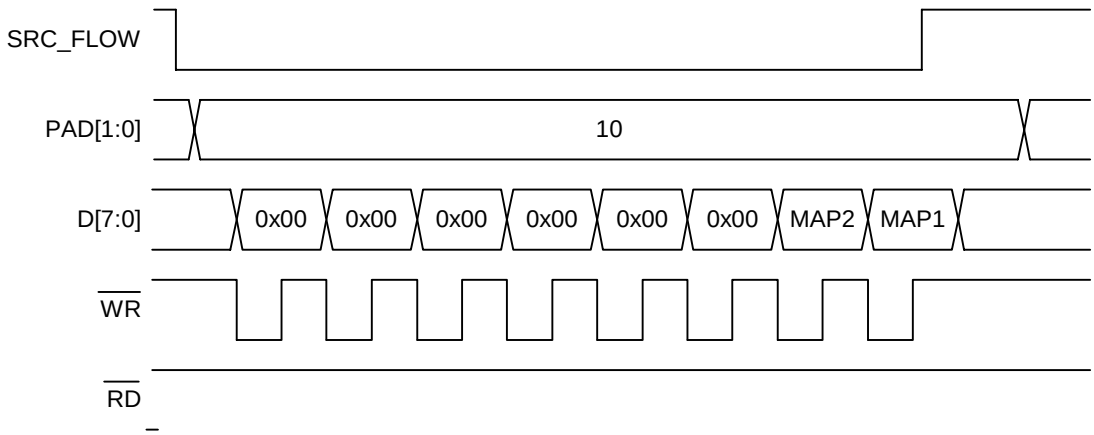


Figure 8-10: Source Port Writing MAP Timing (Parallel-Asynchronous Mode)

8.2.2.4 Writing 8 Bytes into the FIFO

Before writing data to the FIFO the first time, the MAP data must be initialized. Then data can be continually transferred up to a page boundary. To cross a page boundary, the new page value must be written into the MAP. After writing a new MAP, the application must wait for **SRC_FLOW** to transition low before data can be written to the FIFO. For this operation, the following signals must be set:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
11	1	↑	Write DATA to FIFO

Table 8-15: Writing 8 Bytes into the FIFO in Parallel-Asynchronous Mode

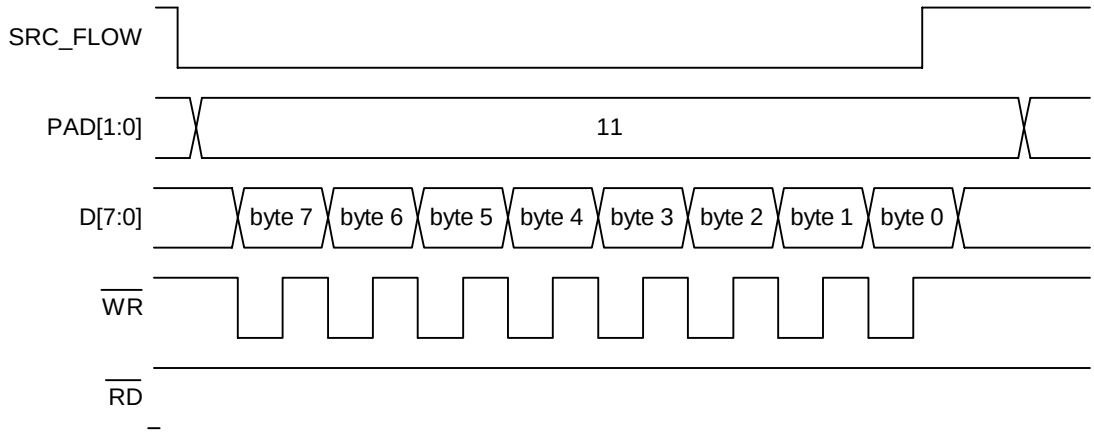


Figure 8-11: Source Port Writing 8 Bytes Timing (Parallel-Asynchronous Mode)

The sequence of data in the FIFO needed for this operation is described in Section 8.2.2.2. MAP is auto-incremented, so that it needs to be written only once when writing larger blocks of data. After the next falling edge at **SRC_FLOW**, the next set of eight bytes can be written (if desired).

8.2.2.5 Writing 1 Byte into the FIFO

To write only one byte to memory, the data byte is sent along with the MAP value. The MSB of MAP2 must be set to indicate a single-byte transfer. Therefore, the FIFO write sequence consists of writing five arbitrary bytes followed by the data byte, followed by the two MAP bytes, as shown in Figure 8-12:

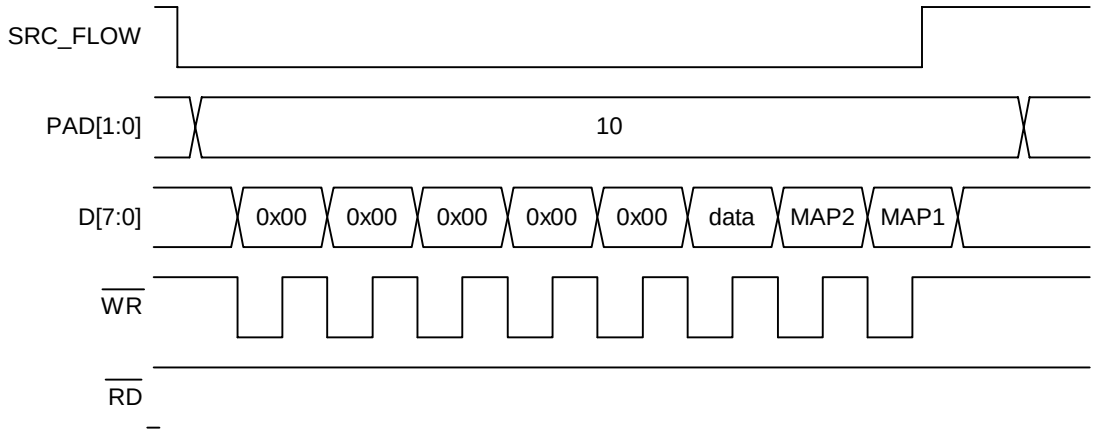


Figure 8-12: Source Port Writing One Byte Timing (Parallel-Asynchronous Mode)

8.2.2.6 Reading 8 Bytes from the FIFO

As with the write sequence, the MAP value must be initialized before any data can be read from the FIFO. After writing MAP, the application must wait for **SRC_FLOW** to transition low before data can be read from the FIFO. For this operation, the following signals must be set:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
11	↓	1	Read Data from FIFO

Table 8-16: Reading 8 Bytes from the FIFO in Parallel-Asynchronous Mode

Eight bytes must always be read from the FIFO. Figure 8-13 illustrates the signal flow for reading data in Parallel-Asynchronous data transfer mode:

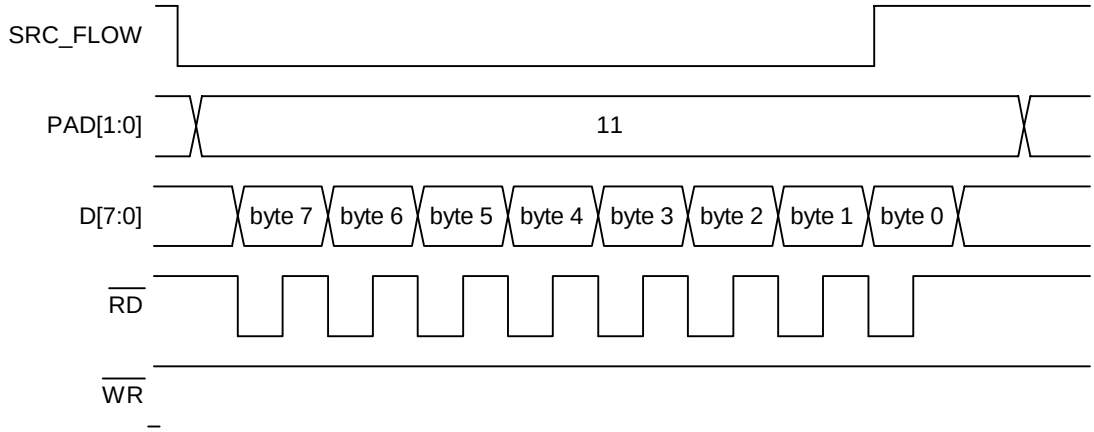


Figure 8-13: Source Port Reading 8 Bytes Timing (Parallel-Asynchronous Mode)

After **SRC_FLOW** has transitions low, **PAD[1:0]** must equal 11. Then a falling edge at $\overline{RD}$ makes the data accessible at **D[7:0]**. After eight read cycles, **SRC_FLOW** changes from low to high again, indicating that the FIFO is busy being processed. MAP is auto-incremented, so it only needs to be written once when reading large blocks of data. After the next falling edge of **SRC_FLOW**, the next set of eight bytes can be read (if desired). The last byte of the 8-byte FIFO is read out first. The first byte in the FIFO contains the first byte read from memory (the data MAP pointed at).

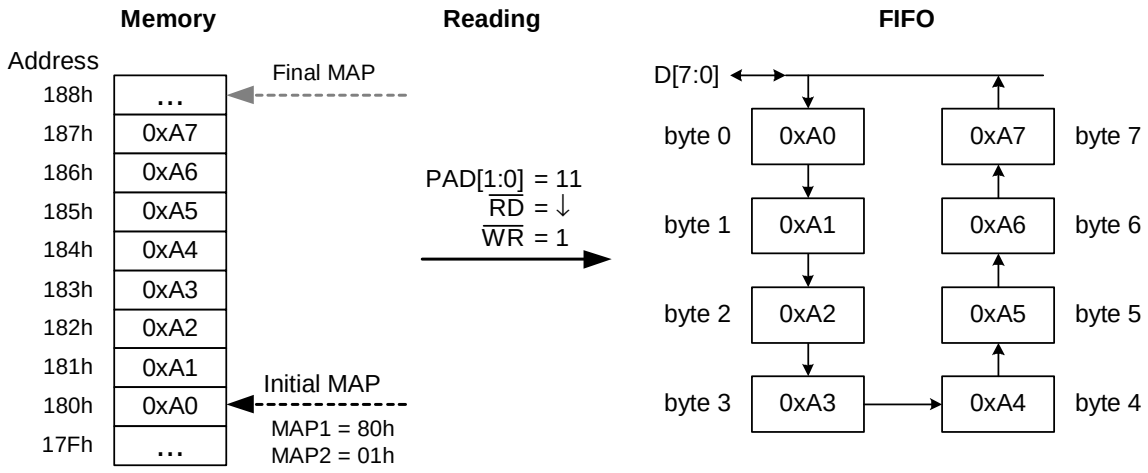


Figure 8-14: Source Port Parallel-Asynchronous Read Data Mapping Example

8.2.3 (bSP) Source Port Status Register

Current status information about the Source Ports in parallel mode can be obtained by reading the Source Port Status register (bSP). The bSP register is a special register that is only available through the parallel interface and does not reside in the OS8104 internal memory map. In order to read the bSP register, the following signals must be set:

PAD[1:0]	$\overline{RD}$	$\overline{WR}$	Selected Operation
10	↓	1	Read Source Port Status Register bSP

Table 8-17: Reading Source Port Status in Parallel Mode (bSP)

A single read cycle retrieves the contents of bSP, as opposed to the eight cycles required to transfer data.

bSP Source Port Status Register (in Parallel Mode, Read-Only)

Bit	Name	Description	Default
7	ZP	Zero-Power mode	0
6	LP	Low-Power mode	0
5	rsvd	Reserved	0
4	AINT	Asynchronous (Packet) data interrupt (opposite polarity from $\overline{AINT}$ pin)	0
3	rsvd	Reserved	0
2	FIFO Empty	FIFO Empty	0
1	FSYNC	Frame Sync (**FSY** pin)	0
0	SRCF	Source Port Busy (**SRC_FLOW** pin)	0

Table 8-18: bSP (Source Port Status Register — Parallel Mode)

ZP Zero-Power Mode. When **ZP** is set, indicates that the chip is in Zero-Power mode. When the chip is in Zero-Power mode, **two** writes to any register (in parallel mode) will wake up the chip.

LP Low-Power Mode. When **LP** is set, indicates that the chip is in Low-Power mode. When the chip is in Low-Power mode, **two** writes to any register (in parallel mode) will wake up the chip.

AINT Asynchronous (Packet) Data interrupt. The **AINT** bit reflects an inverted version of the $\overline{AINT}$ pin. If an asynchronous interrupt is active, **AINT** is set and the $\overline{AINT}$ pin is driven low.

FIFO Empty When set, indicates that the FIFO is ready to receive the next 8 bytes. This bit can be used to re-synchronize the application to the FIFO in Parallel-Asynchronous mode.

FSYNC Frame Sync. **FSYNC** reflects the status of the **FSY** pin. A falling edge of **FSYNC** indicates the start of the SF0 interval.

SRCF **SRC_FLOW** indicator. **SRCF** reflects the status of the **SRC_FLOW** pin. In Parallel-Synchronous (or Parallel-Combined) mode, a rising edge of **SRCF** (along with **FSYNC**) indicates a particular SF interval. In Parallel-Asynchronous mode, **SRCF** set indicates the FIFO is busy being processed and cannot transfer data. When **SRCF** is clear, the FIFO is ready for more transfers.

8.3 Parallel-Combined/Physical Mode

The Parallel-Combined mode is a high-speed parallel interface mode designed to support the maximum bandwidth for both synchronous (stream) data and asynchronous (packet) data. All 60 source data bytes (divided between synchronous and asynchronous data) of the MOST frame are accessible in this mode. In addition, status and control information are provided to manage the asynchronous (packet) data transfers externally. Since the data throughput is very high, the device controlling the OS8104 must be able to handle large amounts of high-speed data.

For Parallel-Combined mode, the OS8104 must be initially configured for Parallel-Synchronous mode. Then setting the **bPCMA.APCM** bit changes the OS8104 parallel interface from Parallel-Synchronous to Parallel-Combined mode. All source data transfers are handled via the FIFO. The synchronous data routing is handled as in the Parallel-Synchronous mode. The FIFO is processed eight times per frame in each direction for source data. The frame is divided into eight intervals of identical length, labeled SF0 to SF7. During each interval, at least one read or write access must be performed. In addition, the transmit status quadlet must be fully written (all four bytes) during the proper SF interval.

Figure 8-15 illustrates the eight SF intervals and reading and writing through the parallel port. During an SF interval, the FIFO must first be read from, then written to. Also illustrated in Figure 8-15 are control data accesses after all source data is transferred (assuming the Control Port is configured to use the parallel interface).

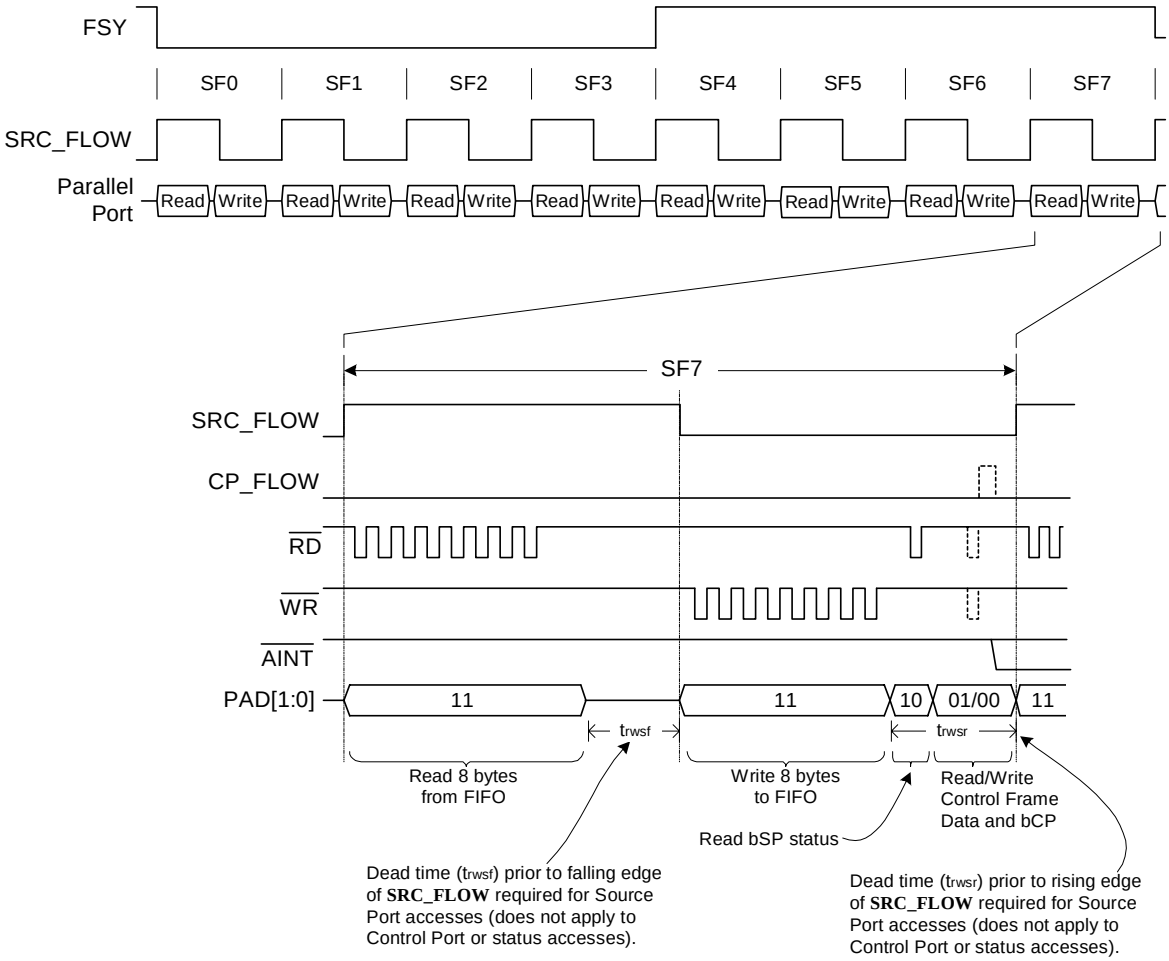


Figure 8-15: Parallel-Combined Mode Timing

As in the Parallel-Synchronous mode, accesses to the Source Port section of the parallel port must not be made t_{rwsf} before the falling edge of **SRC_FLOW** or t_{rwsr} before the rising edge of **SRC_FLOW**. Violating these timing restrictions could corrupt data within the FIFO.

Although asynchronous packets containing as many as 1014 data bytes can be transferred in this mode, the transmitting node must ensure that receiving nodes can accept packet data lengths greater than 48 bytes. For asynchronous data handling in the serial and Parallel-Asynchronous modes, the OS8104 buffers the packet on chip, which limits the packet data size to the internal buffer size of 48 bytes. In Parallel-Combined mode, the asynchronous data is transferred through the external port and the packets must be assembled by the external device. The OS8104 only acts as a transport mechanism and all packet management and control must be handled by the external device. The OS8104 does, however, add CRC data to the end of transmitted packets, and checks the CRC on received packets.

8.3.1 Configuring Parallel-Combined (Physical) Mode

Since the Parallel-Combined (Physical) mode is based on the Source Port's Parallel-Synchronous mode, the OS8104 must be configured in this mode first (see Section 8.2). Setting **bPCMA.APCM** will then place the device in Parallel-Combined mode.

Once in Parallel-Combined mode, the MOST Routing Table (MRT) registers MRT0x40 to MRT0x7F must be filled with address references MRA0x00 to MRA0x3F, as illustrated in Table 8-19. This will map all 60 bytes of Network source data to the parallel port, along with four additional status bytes.

MRT	+0	+1	+2	+3	+4	+5	+6	+7	SF:
0x40	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	← **0**
0x48	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	← **1**
0x50	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	← **2**
0x58	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	← **3**
0x60	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	← **4**
0x68	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F	← **5**
0x70	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	← **6**
0x78	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F	← **7**
	↑	↑	↑	↑	↑	↑	↑	↑	
FIFO Byte:	**byte 0**	**byte 1**	**byte 2**	**byte 3**	**byte 4**	**byte 5**	**byte 6**	**byte 7**	

Table 8-19: Upper Half of MRT in Parallel-Combined Mode

Table 8-19 shows the location of each source data byte of the incoming frame is found in the FIFO, for the respective SF interval.

For sending data out to the Network, the lower half of the MRT (MRT0x00 to MRT0x3F) must be configured. By default (after reset), the lower half of the MRT is configured for *direct throughput*, where the bytes received from **RX** are retransmitted on **TX**, in the same channel. For routing synchronous data entered through the parallel port onto the Network, the lower half of the MRT must be modified as described in Section 12.4 on page 106, replacing the *direct throughput* data with the externally-sourced data address references (MRA).

E6h bPCMA Parallel-Combined Mode Activate Register

Bit	Name	Description	Default
7..1	rsvd	Reserved; Write as 0	0000000
0	APCM	Activate Parallel-Combined mode	0

Table 8-20: bPCMA (Parallel-Combined Mode Activate Register)

APCM Activate Parallel-Combined/Physical Mode. If the Source Port is in Parallel-Synchronous mode, setting **APCM** enables Parallel-Combined mode.

8.3.2 Asynchronous Data Packets

In Parallel-Combined mode, an asynchronous data packet has the following structure:

Byte	Function	
0	Arbitration	
1	Destination address (high byte)	
2	Destination address (low byte)	
3	Length of data (in quadlets) valid values 01h to FEh	
4	Source address (high byte)	These bytes are relevant for length calculation.
5	Source address (low byte)	
6	Data byte 0	
7	Data byte 1	
...	...	
N	Data byte V	
N+1 .. N+4	CRC	

Table 8-21: Data Packet Architecture

Bytes 4 through N are relevant for the packet length calculation. For transmit packets, the CRC is added by the OS8104 hardware. For receive packets, the CRC is validated by the OS8104, with the result available in the last status byte (source data byte 63).

The packet length value is in quadlets (4 bytes) and is calculated as described in Section 15.1.3 on page 136. The maximum length is FEh, as opposed to 0Dh for Packet Data transfer when not in Parallel-Combined mode. The number of data bytes transmitted in a packet is calculated as follows:

$$\text{number of data bytes} = (\text{Length} \times 4) - 2$$

Two bytes must be subtracted since two bytes of the first quadlet, used in the length calculation, are the source address. The maximum number of data bytes per packet in Parallel-Combined mode is:

$$(\text{FEh} \times 4) - 2 = (254 \times 4) - 2 = 1014 \text{ bytes}$$

8.3.3 Receiving Source Data

Based on the configuration of the MRT shown in Table 8-19, all the Network-received source data (60 bytes) is sent out the parallel port, via the FIFO, followed by four status bytes. The division between the synchronous and asynchronous data is contained in the bSBC register, and is also part of the status bytes at the end of the FSY period. The external device would generally collect the entire frame of data into a buffer, and then use the SBC data in the status to determine the start of asynchronous data. The external device does not have to read synchronous data that it does not need; however, at least one read or write access must occur during each SF interval, and the full transmit status quadlet must be written. Figure 8-16 illustrates reading the source data from the OS8104's FIFO.

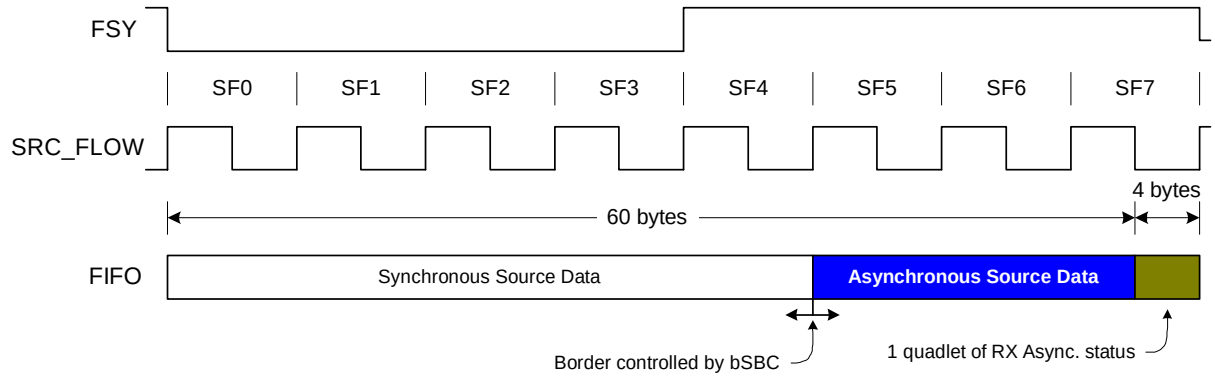


Figure 8-16: Parallel-Combined Mode Data Output (Network Received Data)

8.3.3.1 Received Asynchronous Status Bytes

Starting with the first frame source data byte at 0, the first status byte is 60. Table 8-22 shows the status bytes and their function.

Byte Number	Name	Inactive	Function
60	Synchronous Bandwidth Control (SBC)	—	The lowest 4 bits of SBC contain information about the number of bytes used for synchronous data transfer (in quadlets). See Section 6.2.5 on page 40.
61	Start	00h	If non-zero, this byte indicates that this frame contains the start of a packet.
62	rsvd	rsvd	Reserved
63	Error	00h	If non-zero, this byte indicates that an error (incorrect CRC or packet ended prematurely) occurred during reception of the current packet.

Table 8-22: Asynchronous Received Status Bytes

When receiving a packet, the position of the destination address and packet length (contained in the packet) can be derived from the current SBC value:

Destination Address, high byte location = (SBC × 4) + 1
Destination Address, low byte location = (SBC × 4) + 2
Packet length, in quadlets = (SBC × 4) + 3

The Destination Address and Packet length are only available in the first buffer of a packet (when the *Start* status flag is non-zero).

8.3.3.2 Moving Received Asynchronous Status Bytes

The asynchronous received status bytes have been previously illustrated in a fixed position at the end of the FSY period. These bytes may also be placed between the asynchronous and the synchronous data; however, their position would not be fixed and would now be determined by the SBC value. Assuming the application can manage the dynamic nature of this location, having the status quadlet before the asynchronous data bytes helps minimize the need for external buffering.

The upper half of the MRT must be modified whenever the SBC value changes. The special address references MRA0x3C through MRA0x3F correspond to the four received status bytes. To move the status bytes to the synchronous/asynchronous boundary, the following MRT locations must be changed to contain address references MRA0x3C through MRA0x3F, as follows:

MRT[0x40 + (SBC × 4)] = MRA0x3C
MRT[0x41 + (SBC × 4)] = MRA0x3D
MRT[0x42 + (SBC × 4)] = MRA0x3E
MRT[0x43 + (SBC × 4)] = MRA0x3F

Then the successive MRT locations must be filled with the Network received asynchronous data address references as follows:

MRT[0x44 + (SBC × 4) + n] = (SBC × 4) + n

where n starts at 0 and ends when all MRT locations are filled (MRT0x7F is the last location). Placing the status before the asynchronous data changes the position of the destination address and packet length to:

Destination Address, high byte location = (SBC × 4) + 5
Destination Address, low byte location = (SBC × 4) + 6
Packet length, in quadlets = (SBC × 4) + 7

The following example uses a SBC value of 0Ah, where 10 quadlets are reserved for synchronous data and five quadlets reserved for asynchronous packet data. To place the status between the synchronous and asynchronous data, the first status byte is placed in MRT0x68 as shown in Table 8-23.

MRT	+0	+1	+2	+3	+4	+5	+6	+7	SF
0x40	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	← **0**
0x48	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	← **1**
0x50	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	← **2**
0x58	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	← **3**
0x60	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	← **4**
0x68	**0x3C**	**0x3D**	**0x3E**	**0x3F**	0x28	0x29	0x2A	0x2B	← **5**
0x70	0x2C	0x2D	0x2E	0x2F	0x30	0x31	0x32	0x33	← **6**
0x78	0x34	0x35	0x36	0x37	0x38	0x39	0x3A	0x3B	← **7**
FIFO Byte:	**byte 0**	**byte 1**	**byte 2**	**byte 3**	**byte 4**	**byte 5**	**byte 6**	**byte 7**	

Table 8-23: Upper Half of MRT (Moving Received Status Bytes Example)

The address references (MRA0x28 through MRA0x3B) are shifted up in the MRT to make room for the received status bytes (MRA0x3C though MRAox3F). As a result of moving the status bytes, the destination address and packet length (in the buffer where the *Start* status is set) are now moved to (first asynchronous byte is arbitration):

Destination Address, high byte location = (bSBC × 4) + 5 = 2Dh
Destination Address, low byte location = (bSBC × 4) + 6 = 2Eh
Packet length in quadlets = (bSBC × 4) + 7 = 2Fh

Figure 8-17 illustrates the received data flow with the status bytes moved.

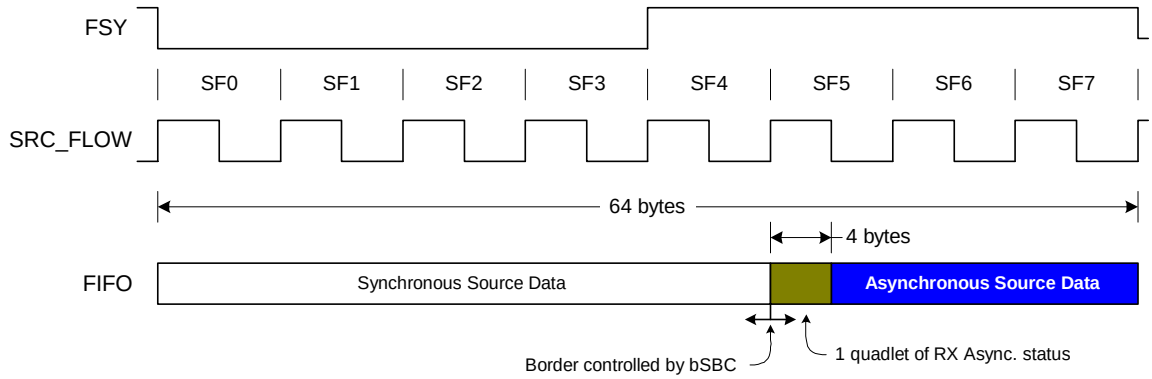


Figure 8-17: Parallel Port Data Output (Network Received Data) — Moved Status

8.3.3.3 Handling Received Data

Figure 8-18 illustrates a program-flow example for handling received data in Parallel-Combined mode. In this example, all 60 data bytes of a frame, plus the four status bytes are read. In the OS8104, the asynchronous packet status bytes are not available until after the synchronous data; therefore, the four status bytes must be the last four bytes of the frame, or placed between the synchronous and asynchronous data. The routine described in this example uses two additional variables:

- *Packet_reception* — Indicates when the routine expects the current frame to contain an intermediate part of the packet (*Packet_reception* set).
- *Length* — Contains the number of data bytes to be received.

After receiving a complete frame, the status byte containing SBC should be read first, then the division of synchronous data and packet data can be calculated. After the synchronous data is transferred to the appropriate sections of the application, packet data handling is performed.

The start of a packet reception is indicated by status byte *Start* being set to a non-zero value. When a start of packet is received, the destination address must be compared with the local node address. This is the task of the application, since the chip does not perform address comparison in Parallel-Combined mode. If the address does not match, packet handling is finished. If the address matches, *Packet_reception* must be set and the length of the packet (in bytes) must be calculated, based on the value contained in the packet (see Section 15.1.3 on page 136). This value is stored in the *Length* variable. The read pointer must then be adjusted to the start of packet data. If the number of bytes to be received is greater than the data contained by the current frame, the local variable *Packet_reception* remains set, indicating succeeding frames contain the rest of the packet. *Packet_reception* should be reset after the last byte of the packet is read.

An error exists if *Packet_reception* is active at the same time *Start* is received, since the reception of the preceding packet was not finished correctly. In this case, an error must be reported, *Packet_reception* must be cleared, and packet data handling is finished.

The received status also includes an *Error* status byte, which should be checked. If the *Error* byte is zero, data can be read until the last byte of the packet is read, or the end of the buffer is reached. If the *Error* status byte is set, the error should be reported, *Packet_reception* should be cleared, and Packet handling is finished.

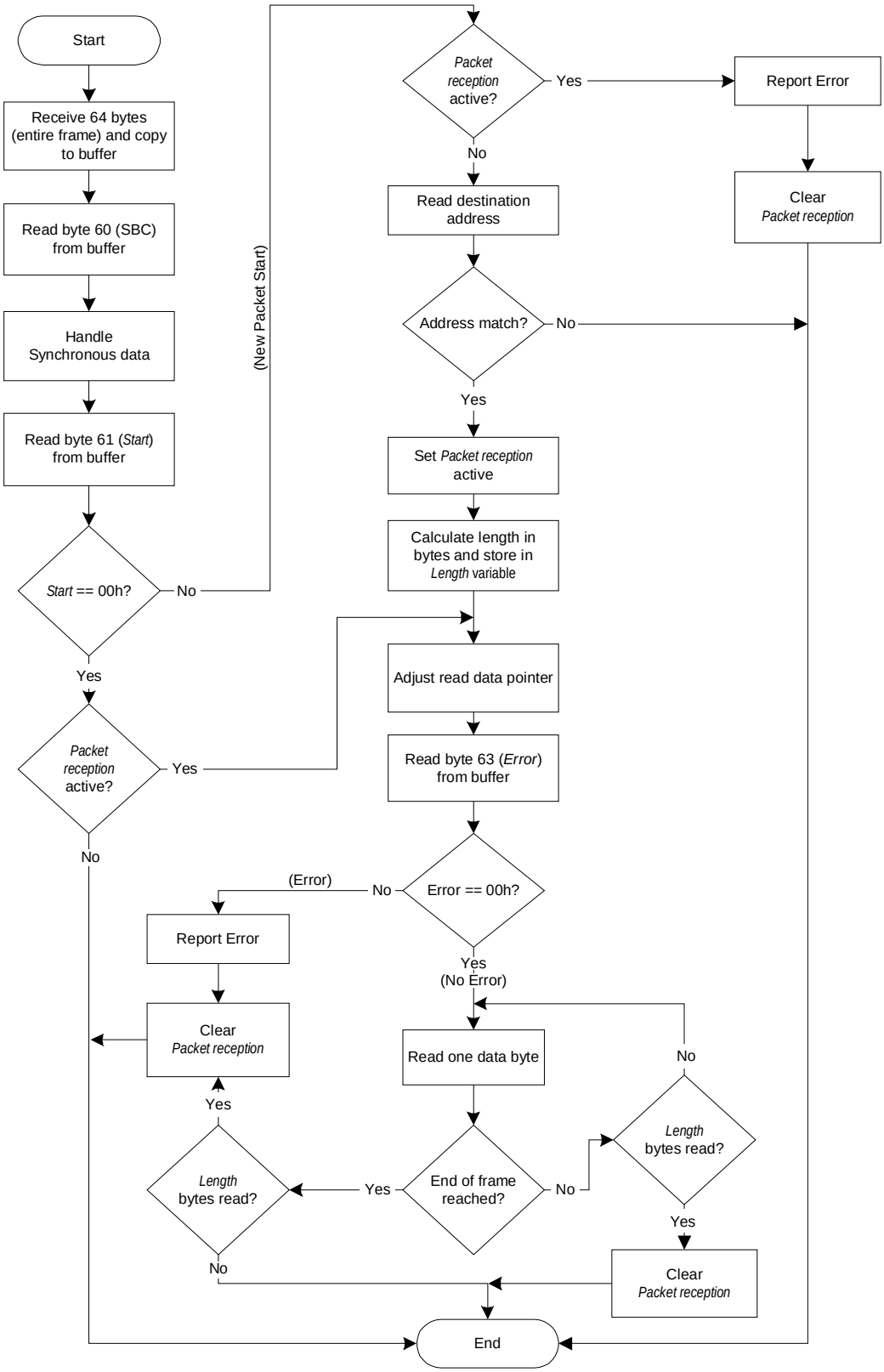


Figure 8-18: Program Flow for Receiving Packet Data (Parallel-Combined Mode)

8.3.4 Transmitting Data

For transmitting synchronous data, the same rules used for the Source Port in Parallel-Synchronous data mode apply. The address references for routing synchronous data are identical, and are described in Chapter 12 on page 97.

When sending out synchronous data and packet data in Parallel-Combined mode, the synchronous data must be transmitted first, followed by four status bytes (1 quadlet), and then the packet data.

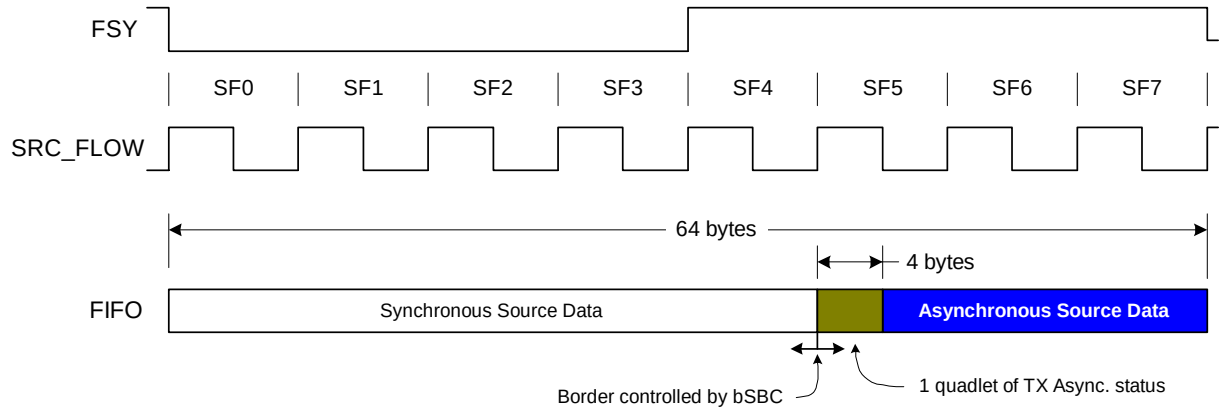


Figure 8-19: Parallel Port Data Input (Network Transmit Data)

8.3.4.1 Transmit Asynchronous Status Bytes

When transmitting packet data, one quadlet contains status information for the Routing Engine and must be prepared by the external application.

Byte Number	Name	Inactive	Function
(SBCx4)+0	Start Transmit	00h	Indicates to the chip that the external application wants to start to transmit a packet of data.
(SBCx4)+1	rsvd	rsvd	Reserved; Write as 00h
(SBCx4)+2	Ack	00h	Indicates to the chip that the application has recognized the activation of the $\overline{\text{AINT}}$ pin and is sending the fourth frame of data.
(SBCx4)+3	rsvd	rsvd	Reserved; Write as 00h

Table 8-24: Asynchronous Transmit Status Bytes

All four transmit status bytes must be sent in each frame, even if no packet data is being sent.

8.3.4.2 Preparing Packet Data

Table 8-21 illustrates Packet structure. The CRC is calculated automatically by the chip and added to the end of the packet. The Arbitration value must be calculated by using the following formula (the Node Position is in register bNPR):

Arbitration = (Node Position × 2) +1

The destination address is the logical node address or the alternate node address of the node that the packet is being sent to. The length is a one-byte value that contains the length of the packet data (bytes 4 through N) in quadlets. The length calculation includes the two source address bytes and must be rounded up to whole quadlets.

$$\text{length} = \text{roundup}\left(\frac{\text{number of data bytes} + 2}{4}\right) = \text{roundup}\left(\frac{\text{number of packet length bytes (bytes 4 through N)}}{4}\right)$$

The maximum length value is FEh, as opposed to 0Dh for packets when not in Parallel-Combined mode. The Source address is the logical node address of the node transmitting the packet, and is included in the length calculation. The external application must fill in the source address bytes with the node's logical address (bNAH/bNAL).

Packets must not exceed 48 data bytes when sending to nodes that cannot accept larger packets.

When the external application has a packet ready for transmission, it starts transmission by setting the *Start Transmit* status byte to a value other than zero. Then the rest of the frame contains the first buffer of the packet data, stored in one of the internal frame buffers. When the frame of data is received by the OS8104 with the *Start Transmit* status flag set, the chip starts to arbitrate for the MOST Network asynchronous data channel.

If the packet fits into three frames or less, the packet data can be written frame by frame and no further action is needed. When the chip wins arbitration, the **AINT** pin is driven low, indicating that arbitration is won. If more than three frames are needed for the packet, then the external device must send the fourth buffer of data in the frame after **AINT** goes low, and continue sending portions frame by frame until the entire packet is sent. The fourth buffer must have *Ack* set to acknowledge that this buffer includes the fourth buffer of asynchronous data. When the final buffer of packet data is transmitted by the chip, the **AINT** pin is driven high, indicating to the external device that the OS8104 is ready to arbitrate/send a new packet, if needed.

In Figure 8-20, 2 quadlets per frame are reserved for asynchronous data transfer. Therefore, bSBC has the value 0Dh, which reserves 52 bytes (13 quadlets) for synchronous data transfer.

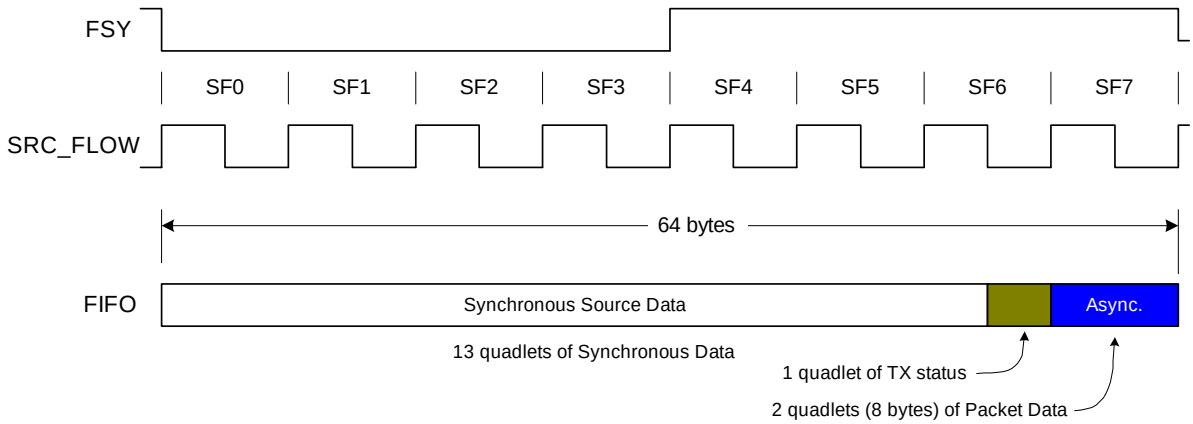


Figure 8-20: Packet Example — Source Data Allocation

In this example, the packet is comprised of nine data bytes sent, from logical node address 0222h (at node position 03h), to destination address 0123h. The assembled packet would look like:

Byte	Value	Function	
0	07h	Arbitration (two times node position plus one)	
1	01h	Destination address (high byte)	
2	23h	Destination address (low byte)	
3	03h	Length of data (in quadlets). Valid values are 01h through FEh	
4	02h	Source address (high byte)	These bytes relavent for the length calculation.
5	22h	Source address (low byte)	
6	10h	Data byte 0	
7	11h	Data byte 1	
8	12h	Data byte 2	
9	13h	Data byte 3	
10	14h	Data byte 4	
11	15h	Data byte 5	
12	16h	Data byte 6	
13	17h	Data byte 7	
14	18h	Data byte 8	

Table 8-25: Sample Asynchronous Data Packet

Since 8 bytes per frame are allocated to asynchronous data, the packet takes two frames to load into the chip and three frames to transmit on the Network, as the total packet size is 18 bytes. One byte is unused, since the packet length is in quadlets, and two bytes are for the CRC, added by the OS8104 when transmitting the packet. Since the CRC is added by the OS8104 hardware, only two frames need to be copied to the OS8104 internal buffers. SBC does not change dynamically, so the *Start Transmit* status byte will always be located at byte position 52 of the frame. The top portion of Figure 8-21 illustrates the data sent by the external device, through the parallel port, and into the FIFO. The bottom portion of Figure 8-21 illustrates the FIFO being read by the Routing Engine and stored in an internal Frame buffer.

Frame 1:

Parallel Port:

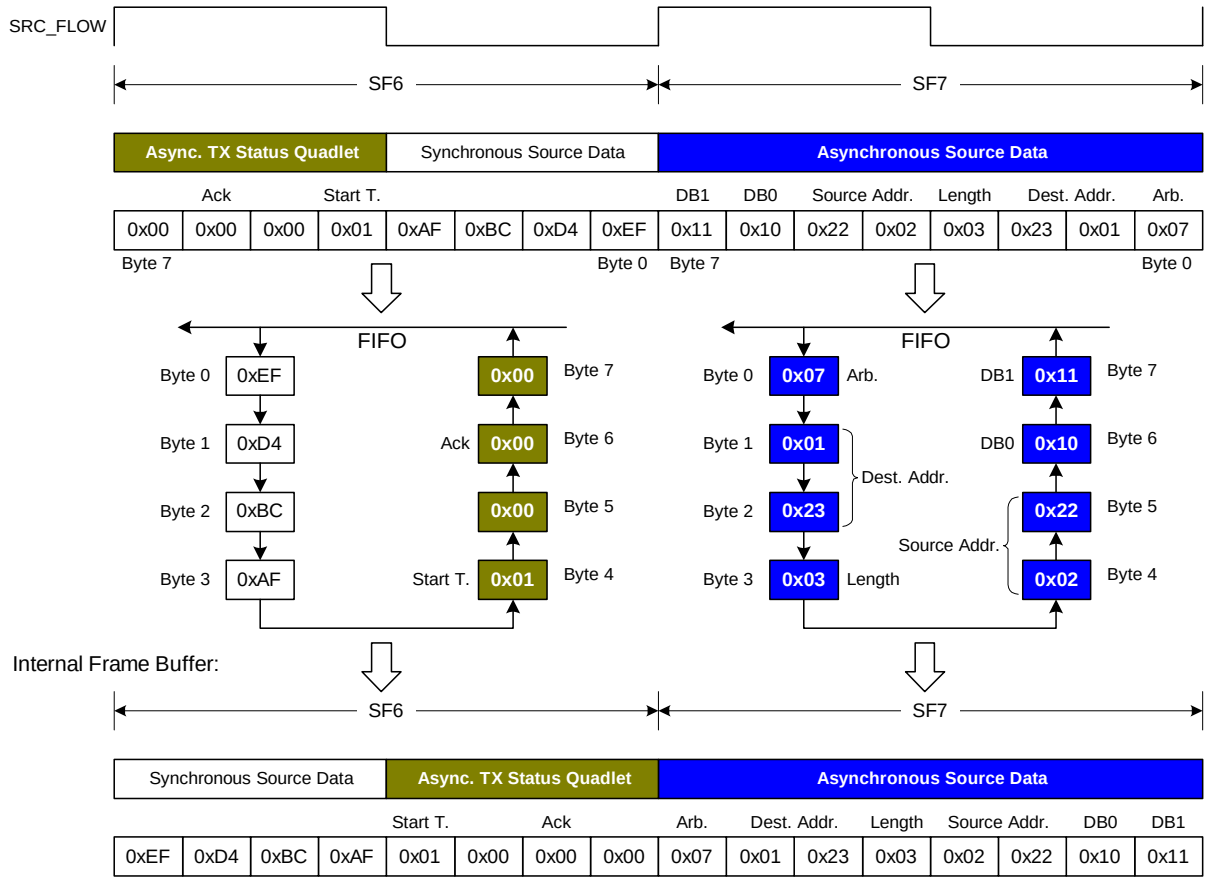


Figure 8-21: Asynchronous Packet Example — Frame 1

The frame containing the first buffer (Frame 1 - includes the initial portion of the asynchronous packet), must have the *Start Transmit* flag set (shown as *Start T.* in Figure 8-21). Frame 2 will contain the rest of the packet. The *Start Transmit* flag must be zero for all buffers/frames except the first one.

In SF6 of the first frame, the external device sends eight bytes to the FIFO. These bytes are half synchronous data and half asynchronous transmit status bytes. The FIFO is filled in reverse order (Byte 7 to Byte 0), placing the *Ack* status flag at the second byte sent to the FIFO and the *Start T.* flag at the fourth byte sent to the FIFO. The Routing Engine reads the data out of the FIFO in reverse order, placing the *Start T.* as the first asynchronous status byte in the internal frame buffer.

SF7 contains the start of the asynchronous packet. The first byte of the packet is the arbitration byte, which is sent to the parallel port last (due to the FIFO byte-reversal). The last byte of the packet, in SF7 of this frame, is data byte 1 (DB1).

Frame 2 of this example is illustrated in Figure 8-22. This frame sends the second (and last) buffer for the example presented. Half of SF6 contains the asynchronous transmit status bytes. The *Start T.* status byte is cleared in this frame, since it is not the first buffer of the asynchronous packet. SF7 contains the last seven data bytes of the asynchronous packet. One extra byte is added to complete the eight-byte buffer since the length must be rounded up to full quadlets.

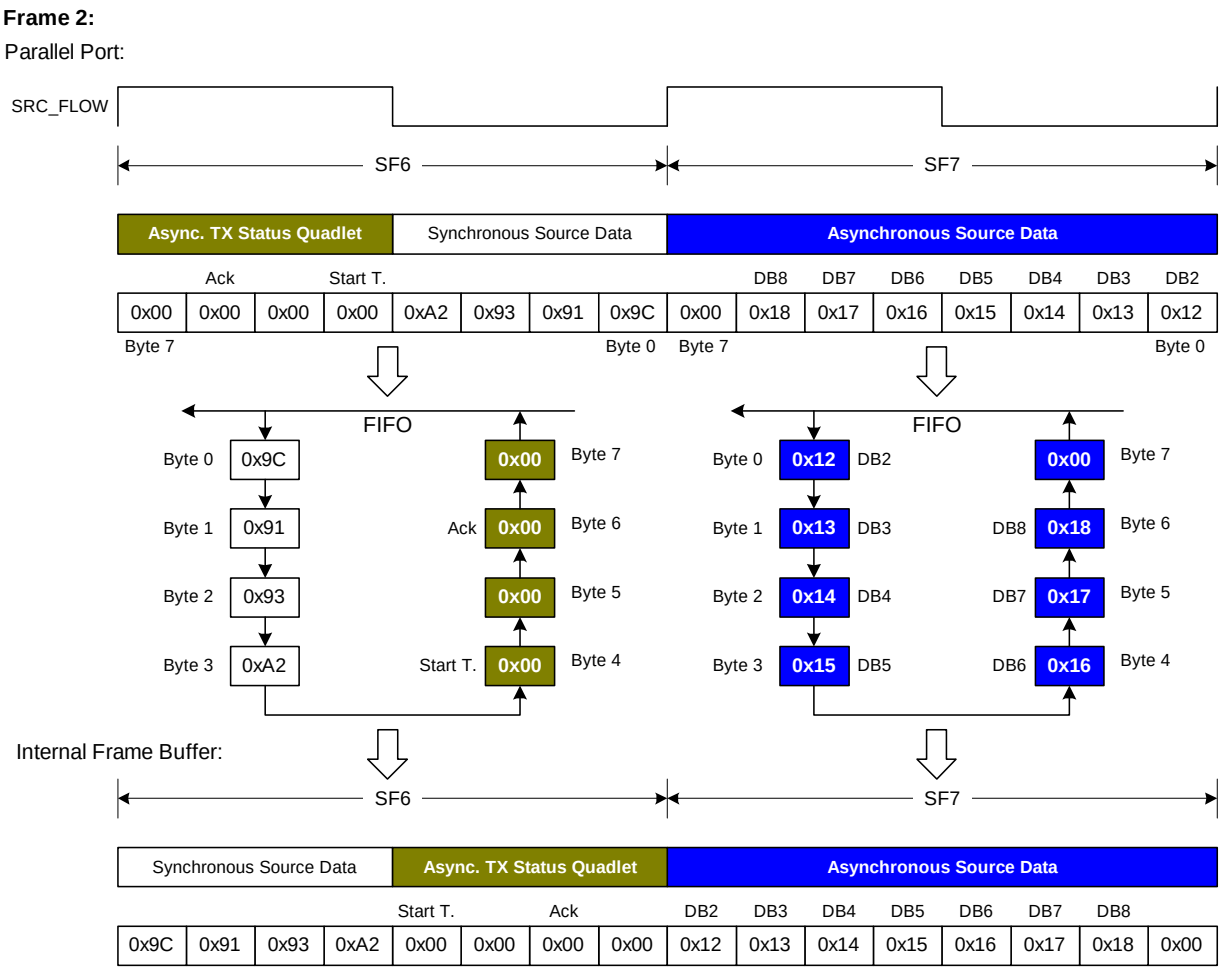


Figure 8-22: Asynchronous Packet Example — Frame 2

The two internal frame buffers used in this example are illustrated in Figure 8-23. The frame buffers are 64 bytes long and store a combination of synchronous and asynchronous data, along with the four asynchronous transmit status bytes. The asynchronous buffer size is 8 bytes (set by the bSBC register value). Since the OS8104 has three internal frame buffers, an asynchronous message that fits into three asynchronous buffers (24 bytes in this example) can be written in back-to-back frames. Asynchronous packets that take more than three buffers must use the **AINT** pin and the *Ack* status byte for hand-shaking with the OS8104.

The synchronous data stored in the internal frame buffers is only transmitted onto the MOST Network if the lower half of the MRT is configured as shown in Table 12-6 on page 106.

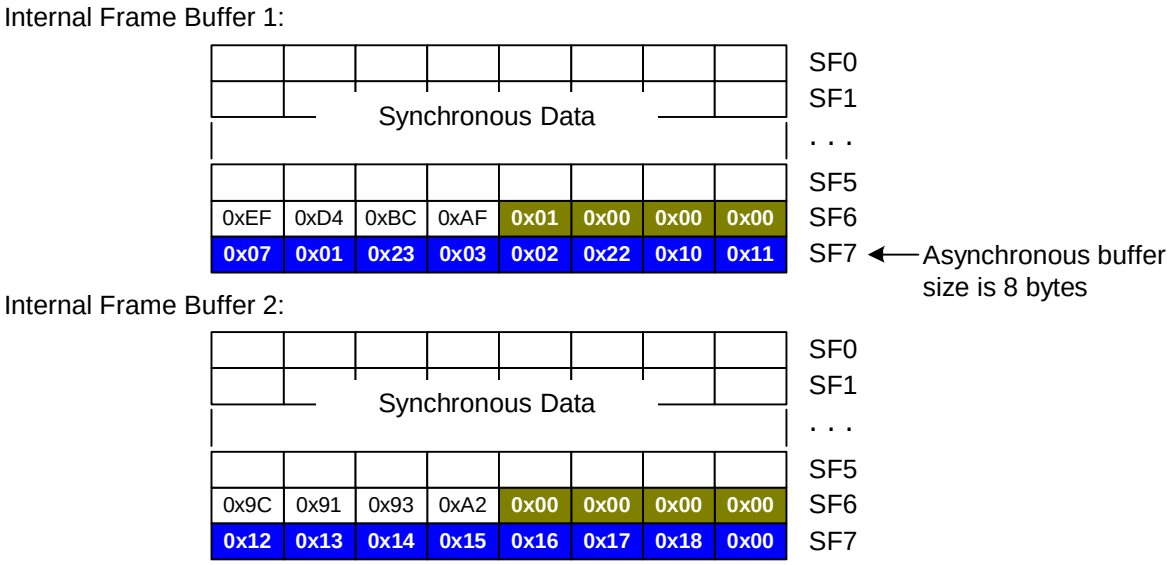


Figure 8-23: Internal Frame Buffers

8.3.4.3 Multi-Frame Packets and $\overline{\text{AINT}}$ Handshaking

To start transmitting a packet, the OS8104 has to arbitrate for the asynchronous data channel. The time between the start of transmission and when the node wins arbitration can be delayed; therefore, a certain amount of data must be stored in internal data buffers. The OS8104 contains three frame buffers that can each hold one complete frame of source data. The asynchronous buffer size is that part of the frame dedicated to asynchronous data (determined by the bSBC register).

If a packet fits into three frames or less, the packet data can be written frame by frame and no further action is required. When the chip wins arbitration, the $\overline{\text{AINT}}$ pin is driven low. $\overline{\text{AINT}}$ goes high at the end of the message transmission, indicating a new message can be sent.

In the following example, the packet being sent fits into two internal buffers (Short Packet). When the first buffer is loaded into the chip (in Frame 1), with the *Start Transmit* (*Start T.*) status byte set, the OS8104 starts arbitrating for the asynchronous channel in the next frame (Frame 2). The second and final buffer is loaded in Frame 2, as illustrated in Figure 8-24 as b2. If the asynchronous channel is free, the node wins arbitration in Frame 2 and drives $\overline{\text{AINT}}$ low. However, if the asynchronous channel is busy, the node may not win arbitration until a later frame. In Figure 8-24, arbitration is won in the 4th frame where the chip drives $\overline{\text{AINT}}$ low. The first asynchronous buffer, b1, (loaded into the chip in Frame 1) is output on the Network in the Frame 4 time period. Then the second (and last) asynchronous buffer for this message, b2, is output on the Network in the Frame 5 time period. Since only two buffers were used for the asynchronous packet, $\overline{\text{AINT}}$ is driven high at the end of Frame 5, indicating the message has been sent. In Frame 6, a new packet can be started.

The time $\overline{\text{AINT}}$ changes with respect to **FSY** is based on the SBC value; however, $\overline{\text{AINT}}$ will change at least a minimum of $\frac{5}{128\text{Fs}}$ before **FSY** changes, or 0.88 ms when Fs is 44.1 kHz.

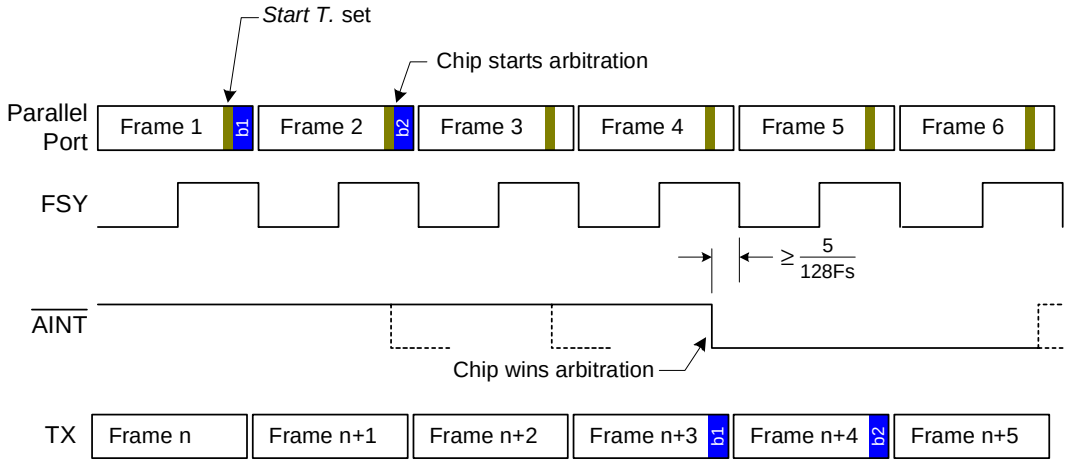


Figure 8-24: Asynchronous Packet Example Arbitration

If the packet is larger than can be contained in the three internal buffers, then the external device should load the first three buffers in three consecutive frames. If the **AINT** pin is low by the beginning of the fourth frame, the external device can continually load buffers until the complete message is transferred. The *Ack* status byte in the Frame containing the fourth asynchronous buffer must be non-zero or the OS8104 will ignore the asynchronous data buffer. This scenario is illustrated in Figure 8-25, where the node wins arbitration while the second or third buffer is being sent to the parallel port. At Frame 4, the **AINT** pin is low so the external device continually sends asynchronous buffers.

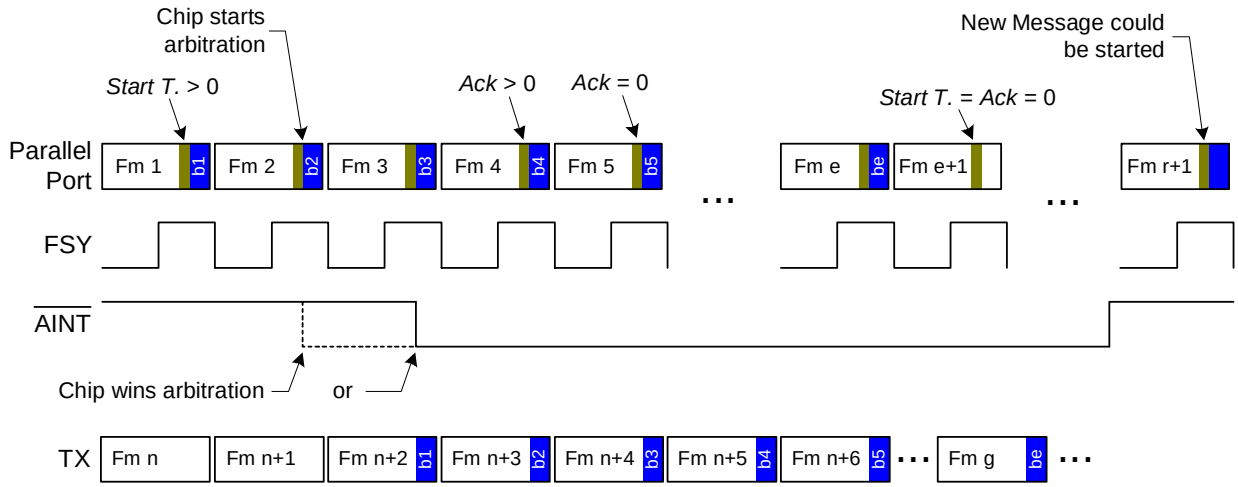


Figure 8-25: Packet Buffering with No Delay

The **AINT** pin goes high after the OS8104 has transmitted the last frame of the packet. Due to internal buffering, this packet could be sent to the OS8104 as many as three frames before it gets transmitted out the **TX** pin onto the Network. The frames sent to the OS8104 after the last packet (*Fm e* in Figure 8-25), but before the **AINT** pin goes high, must not have the *Start Transmit* status flag set and must not have the *Ack* status flag set. (shown as *Fm e+1 …* in Figure 8-25).

If the AINT pin is still high by the beginning of the fourth frame, then the asynchronous channel on the MOST Network is in use and the external device must wait until the AINT pin goes low before loading any more asynchronous data buffers. Synchronous data should still be sent since asynchronous data transfer operates independently of synchronous data transfer. Typically, when an external device is waiting for the AINT pin to go low, it just continually repeats sending the third asynchronous data buffer since at least one access in each SF interval is required for normal operation. The actual data sent is irrelevant since it is not stored in an internal buffer.

Figure 8-26 illustrates sending a Packet larger than three buffers, where the Network asynchronous channel is in use. The Network asynchronous channel frees up at parallel port Frame 6, where the local node wins arbitration. The external device notices that the AINT pin is low, and starting in the 7th frame, sends the fourth asynchronous buffer (b4) with the *Ack* status byte greater than 0. The fifth asynchronous buffer (b5) is sent in Frame 8 with the *Ack* status byte reset to 0, and buffers are continually transmitted until the entire packet is sent. A new message can be sent in the frame following the one in which the AINT pin goes high.

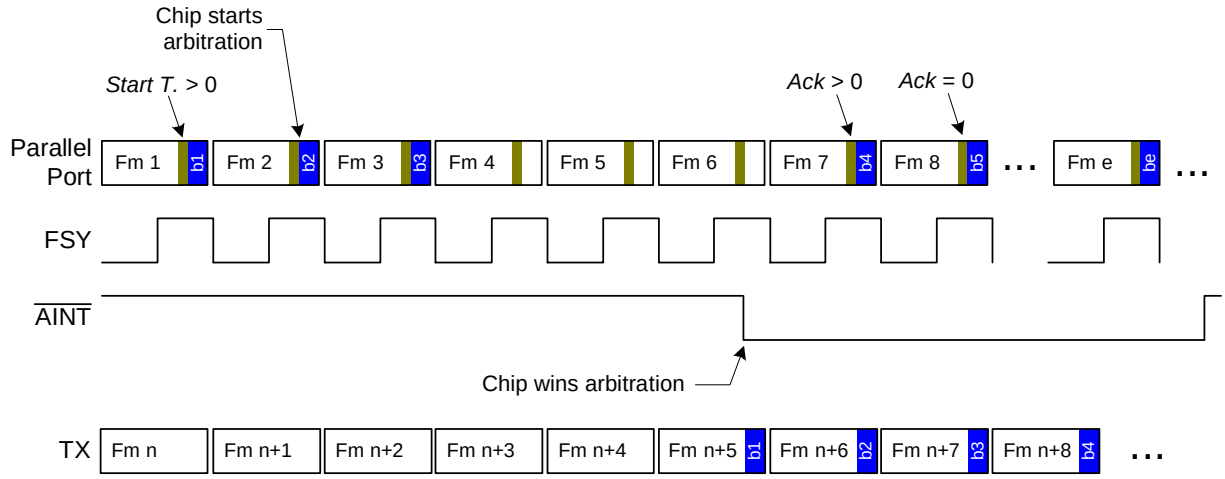


Figure 8-26: Packet Buffering with Delay

The fourth packet buffer (with the *Ack* status byte greater than 0) must be sent in the frame following the one in which the AINT pin went low. Otherwise, invalid data may be transmitted.

8.3.4.4 Priority

Packet priority is specified in the bPPI register (see Section 15.1.4 on page 136).

8.3.4.5 Idle SF Intervals

If no data transfer is required during an SF interval, it is sufficient to perform a single read or write access. However, all four bytes of the asynchronous transmit status quadlet must always be written (to all zeros if no status).

9 Clock Manager

The Clock Manager generates all internal clocks and the **RMCK** output clock, which can drive external devices such as CD and DVD drives. The Clock Manager consists of an input mux, an analog PLL, output dividers, and a network activity detector. The input mux allows the PLL to be driven by one of several possible sources:

- Crystal oscillator (or external clock source)
- Bit clock (**SCK**)
- S/PDIF data stream (through **SR0**)
- Incoming data from MOST Network (**RX**)

The Clock Manager, depicted in Figure 9-1, is controlled through three registers: bCM1, bCM2, and bCM4. The PLL always provides a clock source; however, when the PLL is unlocked (i.e. no data on the **RX** pin), the PLL will drift to its lowest frequency, typically 7 to 10 kHz. The OS8104 is always accessible through the Control Port, although source data is not transferred when unlocked.

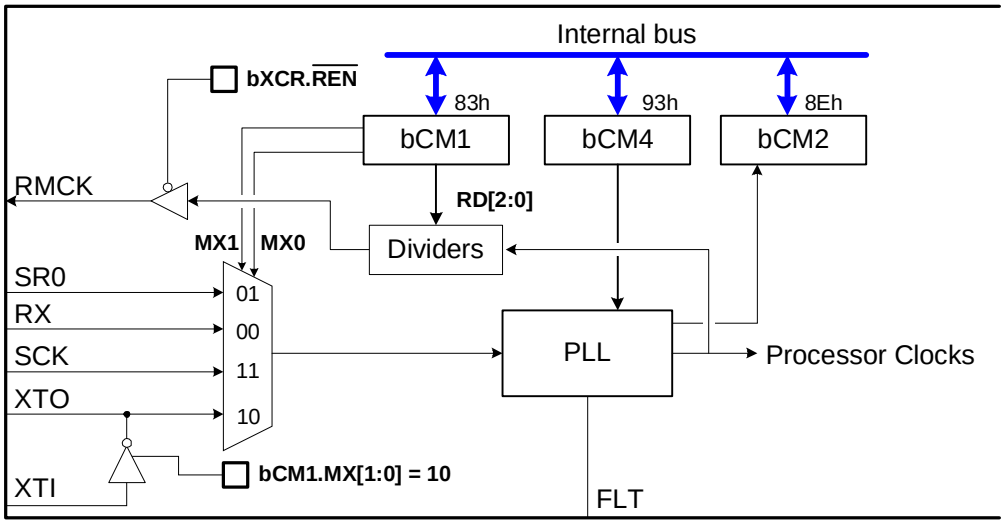


Figure 9-1: Clock Manager

9.1 Clock Manager Registers

83h bCM1 Clock Manager 1 Register

Bit	Name	Description	Default
7	PLD	PLL disable	0
6..4	RD[2:0]	RMCK divider	000
3, 2	XTL[1:0]	Oscillator divider	00
1, 0	MX[1:0]	PLL input select	00

Table 9-1: bCM1 (Clock Manager 1 Register)

PLD PLL disable. When **PLD** is set, the PLL stops trying to lock to the selected source and drifts to its lowest frequency (typically 7 to 10 kHz).

RD[2:0] **RMCK** Divider. When **RMCK** is enabled (**bXCR.$\overline{\text{REN}}$** clear), these bits determine the output frequency as a ratio to the locked sample frequency. Changing **RD[2:0]** can cause glitches on **RMCK** in the OS8104. In addition, an unlock event can cause glitches on **RMCK**, when the **RMCK** frequency is less than 1024Fs.
000 – 384Fs
001 – 256Fs
010 – 128Fs
011 – 64Fs
100 – 1536Fs
101 – 1024Fs (33 % duty cycle)
110 – 768Fs
111 – 512Fs

XTL[1:0] Crystal oscillator divider. These bits allow the crystal oscillator to be one of three frequencies for a given Network sample frequency (Fs).
00 – 256Fs
01 – 384Fs
10 – 512Fs
11 – Reserved

MX[1:0] PLL input select. These bits determine the PLL locking source. **RX** is the default source after reset. If **bXCR.MTR** is clear (timing-slave), then **MX[1:0]** should be 00. If **bXCR.MTR** is set (timing-master), then **MX[1:0]** should be configured for one of the other three choices.
00 – **RX** (MOST)
01 – **SR0** (S/PDIF)
10 – Crystal Oscillator
11 – **SCK**

8Eh bCM2 Clock Manager 2 Register

Bit	Name	Description	Default
7	$\overline{\text{LOK}}$	PLL lock status (read-only)	1
6	NAC	Network activity (read-only)	0
5	ZP	Zero-Power mode enable	0
4	LP	Low-Power mode enable	0
3..0	rsvd	Reserved; Write as 0	0000

Table 9-2: bCM2 (Clock Manager Register 2)

$\overline{\text{LOK}}$ PLL lock status. When clear, indicates the PLL is locked to the selected source. When set, the PLL is unlocked. If **RMCK** is less than 1024Fs, lock/unlock events can cause glitches on the **RMCK** output.

NAC Network Activity detection. When **NAC** is set, it indicates activity on the Network (rising or falling edges detected on **RX**), but does not indicate whether data is being transferred correctly or not. When **NAC** is cleared, it indicates that no activity is detected. **NAC** is cleared approximately 128 ns after activity ceases on **RX**.

ZP Zero-Power mode enable. When set, the OS8104 is placed in Low-Power mode and enters Zero-Power if Network activity ceases. The OS8104 powers up again after Network activity detection, or the chip is accessed via the Control Port or parallel port.

LP Low-Power mode enable. When set, the OS8104 is placed in Low-Power mode. The OS8104 powers up after access to the chip via the Control Port or parallel port, or reception of a wake-up signal from the Network.

93h	bCM4	Clock Manager 4 Register	Host

Bit	Label	Description	Default
7..0	D[7:0]	Affects the PWD tolerance and should be configured for optimal operation (see Table 9-4). PWD tolerance listed in the *Electrical Characteristics* Section will not be achieved if this value is set incorrectly.	C0h

Table 9-3: bCM4 (Clock Manager 4 Register)

Node Type	bCM4 Settings	
	Revision D	Revision C
Timing-slave node	C0h	C3h
Timing-master node	C3h	C0h

Table 9-4: bCM4 Settings

9.2 PLL Lock Status

The **bCM2.$\overline{\text{LOK}}$** bit indicates the current lock status of the PLL. When **$\overline{\text{LOK}}$** is clear, the PLL is locked, and when **$\overline{\text{LOK}}$** is set the PLL is unlocked. When **$\overline{\text{LOK}}$** is set, the **bXSR.EXL** bit indicates a lock error occurred since the last time it was cleared. **EXL** is sticky and can be cleared (armed for capturing a new lock error) by writing **EXL** to zero.

When the OS8104 initially obtains lock, it takes three frames for the OS8104 to synchronize source data. Therefore, if the OS8104 is not in *All-bypass* mode (**bXCR.$\overline{\text{ABY}}$** set) and not in *source-data bypass* (**bXCR.SBY** clear), then synchronous data will not be transferred properly between the Network and the Source Ports for three frames.

For timing-slave nodes, if **RX** is low (no light is present), the PLL drifts to its lowest frequency, typically 7 to 10 kHz. Source data cannot be transferred to the OS8104 while the PLL is unlocked. Only Control data can be transferred (via the Control Port or the parallel interface).

9.3 VREF and FLT Pins

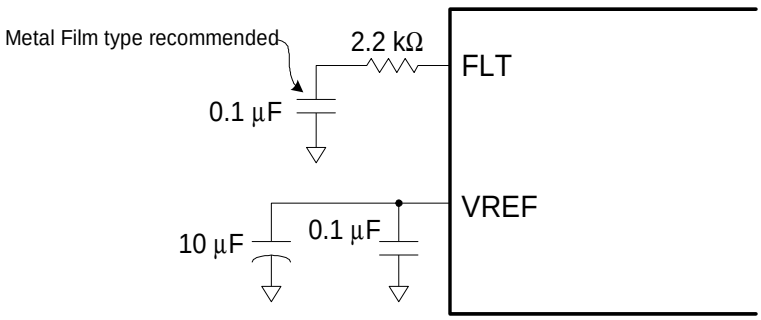


Figure 9-2: Standard Application for VREF and FLT Pins

The circuitry illustrated in Figure 9-2 is the optimum setup for the **FLT** and **VREF** pins. The **VREF** 0.1 µF capacitor should be of a ceramic type, and should be placed as close as possible to the **VREF** and **AGND** pins. The **FLT** capacitor and resistor should be placed as close as possible to the **FLT** and **AGND** pins. To minimize vibration and shock effects on PLL locking, the **FLT** capacitor should be a metal film type (such as Panasonic ECP-U1C104MA5), as ceramic capacitors are more sensitive to shock and could cause unlock events in high-vibration environments. The **FLT** pin is a high impedance node; therefore, leakage currents should be kept below 1 µA or average pulse-width distortion tolerance could be adversely affected. Conformal coating is recommended for systems where condensation can occur.

9.4 Crystal Pins (XTI/XTO)

A crystal oscillator can be used by the timing-master node to set the timing for the entire Network. Timing-slave nodes generally have a crystal oscillator to allow the node to run diagnostics when the Network is down.

The crystal oscillator should be in a fundamental mode, parallel resonant with a load capacitor specified as mentioned below. Figure 9-3 depicts the external circuitry connected to the OS8104 oscillator circuit. Since the internal inverter/amplifier is operated in its linear region, external series resistors should not be used, as they will lower the gain and could cause start-up problems.

When the crystal oscillator is not selected as the timing source for the node (**bCM1.MX[1:0]** not set to 10), the oscillator is powered down to minimize noise and power.

If an external clock is used in lieu of a crystal oscillator, it must support CMOS drive levels and be connected to **XTO**, with **XTI** grounded (see Figure 9-3). For the timing-master node, this clock must be jitter free. For a timing-slave node, it is recommended that this clock also be jitter free to support ring-down diagnostics where the timing-slave might be the timing-master for the rest of a broken ring.

OR

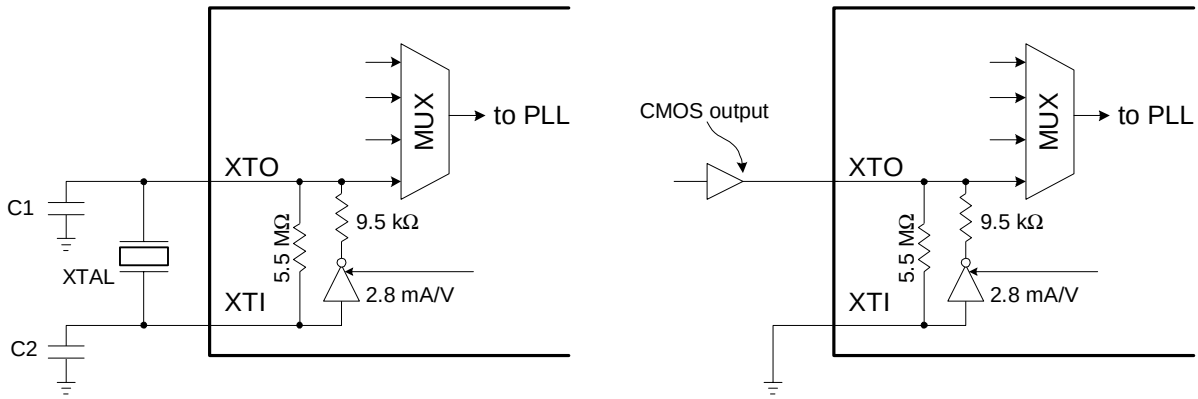


Figure 9-3: Crystal Oscillator Input

The crystal frequency can be 256Fs, 384Fs, or 512Fs, in the OS8104. The selection is made via the **bCM1.XTL[1:0]** bits. Several factors must be considered when selecting a crystal. These include load capacitance, oscillator margin, cut, and operating temperature to name a few.

Table 9-5 show the possible crystal oscillator frequencies, given various Network frame rates.

bCM1.XTL[1:0]	00	01	10	Units
Fs	256x	384x	512x	
38 kHz	9.728	14.592	19.456	MHz
44.1 kHz	11.2896	16.9344	22.5792	MHz
48 kHz	12.288	18.432	24.576	MHz

Table 9-5: Crystal Oscillator Frequencies

For information on crystal oscillator selection, see Section 20.1 on page 180.

10 Power Management

Two power-saving modes are available in the OS8104 to minimize the node power when no activity exists on the Network or from the external application:

- Low-Power mode
- Zero-Power mode

The power saving modes are enabled by writing the appropriate bit in the bCM2 register. The power-saving mode is exited when activity is detected on the Network (**RX** pin), the Control Port is accessed by the external application, or a special wake-up preamble is issued by the timing-master. Table 10-1 describes the function of the power management pins.

Pin	Comment
STATUS	When high, indicates that the chip is in Zero-Power mode. The **STATUS** pin can be used to control the external application's power supply.
$\overline{\text{WAKE\_UP}}$	This signal enables or disables the power supply of the external signal reception circuitry (e.g., a FOR unit). If the chip pulls the **WAKE_UP** pin low, the reception circuitry should be active.
R_TIMER	Connector for external resistor (390 kΩ or 400 kΩ). Used by the internal clock section of the wake-up logic.

Table 10-1: Power Management Pins

Power and Network activity status, as well as the enable bits for Zero-Power mode and Low-Power mode, can be accessed in the bCM2 register.

10.1 Low-Power Mode

The Low-Power mode will place the device in the lowest power state where the Network is still active, but the particular node is not being used. When configured as the timing-master, the **bXCR.LPW** bit (see Section 6.2.1 on page 37) initiates the transmission of a wakeup-frame, sent out onto the Network to wake up nodes in Low-Power mode. As of the writing of this Data Sheet, the Low-Power mode is not supported by the existing *MOST Specification*.

10.1.1 Entering Low-Power Mode

The chip switches to Low-Power mode when the **bCM2.LP** bit is set via the Control Port. The following list shows the characteristic features of this mode:

- Only the transceiver section of the MOST chip and the wake-up logic are active. The rest of the chip (e.g., Source Ports) is powered down.
- The **bXCR.SBY** bit is activated:
  - Node is still visible to the Network
  - Node delay count for synchronous data is reduced

Since the transceiver is still active, the device is visible to the rest of the Network and counted during Node Position counting. The Node Delay count is NOT incremented since **bXCR.SBY** is set (synchronous data bypass).

10.1.2 Leaving Low-Power Mode

The chip wakes up whenever one of the following events occur:

- Wake-up frame is received (generated by the timing-master node, using **bXCR.LPW**).
- Two falling edges are detected on **SDA** of the Control Port in I^2C mode.
- Two falling edges are detected on **$\overline{\text{CS}}$** of the Control Port in SPI mode.
- Two write accesses are detected on **$\overline{\text{WR}}$** when in parallel mode.

The **bXCR.SBY** bit stays active after the waking up procedure is finished.

At the end of the wake-up procedure, the **bMSGS.ERR** bit is set and causes the **INT** interrupt pin to go low. The register values bNAH, bNAL, bGA, bXTIM, bXRTY, bAPAH, and bAPAL are unaltered. All the other registers are reset to default values. Since the Network configuration may have changed during Low-Power mode, the MRT is reset to its default values.

10.2 Zero-Power Mode

The Zero-Power mode is available when not using the FOR to manage system power, as described in the *MOST Specification* and illustrated in Chapter 20. Per the *MOST Specification*, the FOR generally powers down the entire node when no light is detected on the **RX** pin. When not using an FOR with power management capability (or when not using FORs at all), the Zero-Power mode can minimize power when the Network is inactive (although it will not be as low as when using the FOR to control power). When the Control Port is configured in parallel mode, the **AD0** and **AD1** pins must be tied to VDDD, in order to enter Zero-Power mode.

10.2.1 Entering Zero-Power Mode

Zero-Power mode is enabled when the **bCM2.ZP** bit is set. When **ZP** is set, the chip activates Low-Power mode first. Once in Low-Power mode, if no activity is detected on the Network, the chip then automatically goes to Zero-Power mode. If the **ZP** bit is set while there is still Network activity, the chip waits for Network activity to cease. As long as the chip is in Low-Power mode, it will wake up when a wake-up signal is sent via the Network. The following list shows the characteristic features of Zero-Power mode:

- Low-Power mode is entered immediately after setting the **bCM2.ZP** bit.
- Zero-Power mode is entered only if no Network activity detected.
- In Zero-Power mode, only the wake-up logic is active.
- The wake-up logic monitors the Network (**RX**) for activity.

10.2.2 WAKE_UP and R_TIMER Pins

In Zero-Power mode, the wake-up logic scans the Network input for activity every 50 ms (20 Hz, if the resistor at **R_TIMER** is 400 kΩ). Therefore, the **WAKE_UP** pin is driven low for approximately 150 µs. It is possible to save more power when switching the external receiving circuitry's power off when **WAKE_UP** is high. If no activity is detected, the OS8104 will return to Zero-Power mode.

Changing the resistor at **R_TIMER** to 390 kΩ has no significant effect on the scan interval (20.5 Hz/48.78 ms instead of 20 Hz/50 ms). If the wake-up logic and Zero-Power mode are not used, **R_TIMER** must be tied to **VDDA** (analog power supply pin).

10.2.3 Leaving Zero-Power Mode

The chip wakes up whenever one of the following events occurs:

- Network activity is detected on **RX**.
- Two falling edges are detected on **SDA** of the Control Port in I^2C mode.
- Two falling edges are detected on **CS** of the Control Port in SPI mode.
- Two write accesses are detected on **WR** when configured for parallel operation.

After finishing the wake-up procedure, the chip will set **bMSGS.ERR** and generate a power-on reset interrupt (**INT** driven low), to indicate that the chip's internal clocks are running and the external microprocessor can access internal registers via the Control Port. The registers bNAH, bNAL, bGA, bXTIM, bXRTY, bAPAH, bAPAL are unaltered. All the other registers, including the MRT, are reset to default values.

11 Interrupt Handling

The OS8104 has two interrupt pins:

- $\overline{\text{AINT}}$ — Used when sending asynchronous (packet) data. The $\overline{\text{AINT}}$ pin is described in Section 8.3.4.3 and Chapter 15.
- $\overline{\text{INT}}$ — Indicates power-up initialization complete, Control message received/transmitted, or occurrence of an error.

The $\overline{\text{INT}}$ pin signals a variety of events. Interrupts can be generated by a *Network configuration change*, the transmission or reception of a Control message, or the occurrence of an unmasked error (Section 6.2.4 on page 39 describes which error events can affect the $\overline{\text{INT}}$ pin). The bIE register controls which events are unmasked (allowed to generate interrupts). When an unmasked interrupt event occurs, writing to the appropriate bit in the bMSGC register can clear the interrupt event. Figure 13-2 on page 116 illustrates the logic associated with the $\overline{\text{INT}}$ and **ERROR** pins.

11.1 bIE (Interrupt Enable Register)

88h bIE Interrupt Enable Register

Bit	Label	Description	Default
7..4	rsvd	Reserved; Write as 0	0000
3	IALC	Network configuration changed	0
2	IERR	Interrupt on Error Event	0
1	IMTX	Message transmitted	0
0	IMRX	Message received	0

Table 11-1: bIE (Interrupt Enable Register)

IALC Interrupt on: Network configuration changed. When **IALC** is set, an interrupt is generated if **bMSGS.ALC** is also set. The **bMSGS.ALC** bit indicates that bMPR or bMDR has changed.

IERR Interrupt on: Error Event. When **IERR** is set, an interrupt is generated if **bMSGS.ERR** is also set. At power-up or wake-up, an interrupt is generated and the $\overline{\text{INT}}$ pin always goes low after initialization is complete. This occurs regardless of the value of **IERR**.

IMTX Interrupt on: Message transmitted. When **IMTX** is set, an interrupt is generated if **bMSGS.MTX** is also set. The **bMSGS.MTX** bit indicates that a Control message has been transmitted. The **bMSGS.TXR** bit indicates the transmission status, with the bXTS register storing transmission error codes, if a transmission problem was encountered.

IMRX Interrupt on: Message received. When **IMRX** is set, an interrupt is generated if **bMSGS.MRX** is also set. The **bMSGS.MRX** bit indicates that a Control message has been received. The received message type is indicated in the bRTYP register.

All interrupts can be cleared by setting the respective reset bits in the bMSGC register (see Section 13.2.1 on page 115).

11.2 Power-On Interrupt

When the OS8104 is powered up (or reset), the $\overline{\text{INT}}$ pin is driven low after the chip has finished initialization. The interrupt can be cleared (and $\overline{\text{INT}}$ brought high) by the application by writing to the bMSGC register. The chip should not be accessed until this power-up interrupt is completed. The power-on interrupt occurs regardless of the **bIE.IERR** setting.

11.3 Behavior on Multiple Interrupts

When multiple interrupt events are enabled (through the bIE register), more than one event can be pending at any given time. Figure 11-1 illustrates the occurrence of two interrupt events. **bMSGS.ALC** is triggered first, which generates an interrupt (falling edge of $\overline{\text{INT}}$). During the time the **bMSGS.ALC** interrupt is pending, another interrupt event (**bMSGS.MTX**) occurs.

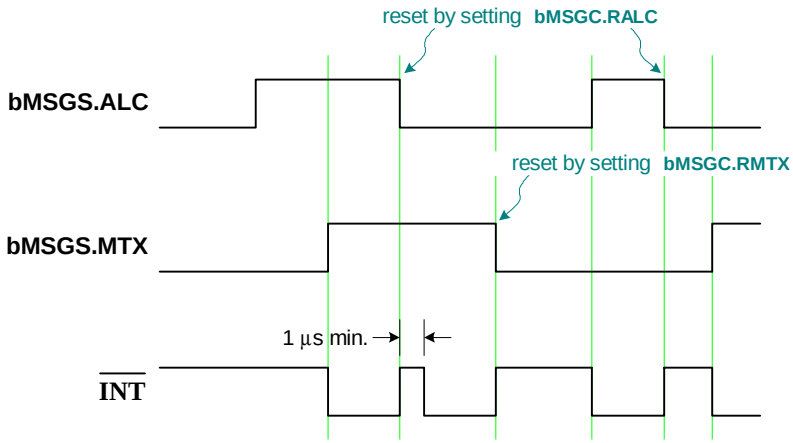


Figure 11-1: Multiple-Interrupt Events

When the first event (**bMSGS.ALC**) is cleared, the $\overline{\text{INT}}$ pin returns high (inactive) for a short period of time. Since a second event is pending, a new interrupt occurs, and is active until the second event is cleared.

12 MOST Routing Table

Synchronous data can be routed from the Source Port inputs (**SRn** or **D[7:0]** pins) to the MOST Network transmit pin (**TX**). Likewise, synchronous data can be routed from the MOST Network receive pin (**RX**) to the Source Port outputs (**SXn** or **D[7:0]** pins). Lastly, data can be routed directly from the Source Port inputs to the Source Port outputs. The OS8104 contains a Routing Engine that acts like a cross-point switch to manage the transfer of synchronous data from inputs to outputs, see Figure 12-1.

The Routing Engine (RE) contains a set of registers, referred to as the MOST Routing Table (MRT) (MRT0x00 to MRT0x7F), that determines how data is routed through the chip. This set of registers is divided into two halves. The lower half (MRT0x00 to MRT0x3B) represents the MOST Network transmitter output locations and routes data onto the Network (**TX**). The upper half (MRT0x40 to MRT0x7F) of the MRT represents the Source Port output locations and routes data to the Source Port outputs (**SXn**) or to the parallel port **D[7:0]** (Source Port in Parallel-Synchronous or Parallel-Combined mode).

The MRT is filled with MOST Routing Addresses (MRA) or *address references* that determine the input data to be sent to a particular output location. Address references are also divided into two halves. Address references MRA0x00 through MRA0x3B (as many as 60 bytes) are associated with synchronous data coming from the MOST Network receiver (**RX**). The upper half of the address references (MRA0x40 to MRA0x7F) are associated with the data arriving on the Source Port input pins (**SRn**) or from the parallel port **D[7:0]** when the Source Port is in Parallel-Synchronous or Parallel-Combined mode.

When address references are placed in the MRT, the Routing Engine transfers data from the address-referenced source (incoming Network frame, **SRn** pins, or the parallel port) to the MRT destination (outgoing Network frame, **SXn** pins, or the parallel port).

After reset, the MRT is configured such that the Network frame data received is passed directly to the Network transmitter. Therefore, physical channel 1 of the incoming Network data stream on **RX** is sent out as physical channel 1 of the outgoing data stream on **TX** (channel 2 is sent out as channel 2, etc.). See Section 12.7 for more details.

All incoming data is buffered before being transmitted. If the node is placing data onto the Network (**bXCR.SBY** clear), then the incoming source data is delayed two frames before being transmitted on the Network or the Source Port outputs. In Figure 12-2, the arrows show how data is routed from inputs to outputs as dictated by the contents of the MRT. Note that any input can be routed to any output and one input can be routed to multiple outputs.

Some of the address references for the MOST Network Receiver (MRA0x00 through MRA0x3B) can not be used if Network source data bandwidth is allocated to asynchronous packet data. The number of usable synchronous data bytes per frame is specified via register bSBC (Section 6.2.5 on page 40). Therefore, MRT locations associated with source data not reserved for synchronous data must not be used.

To minimize confusion, the following text conventions are used when discussing routing. The memory address for a MRT location is written as MRT0xnn. All address references for data sources are written as MRA0xnn, but are entered into the MRT simply as 0xnn.

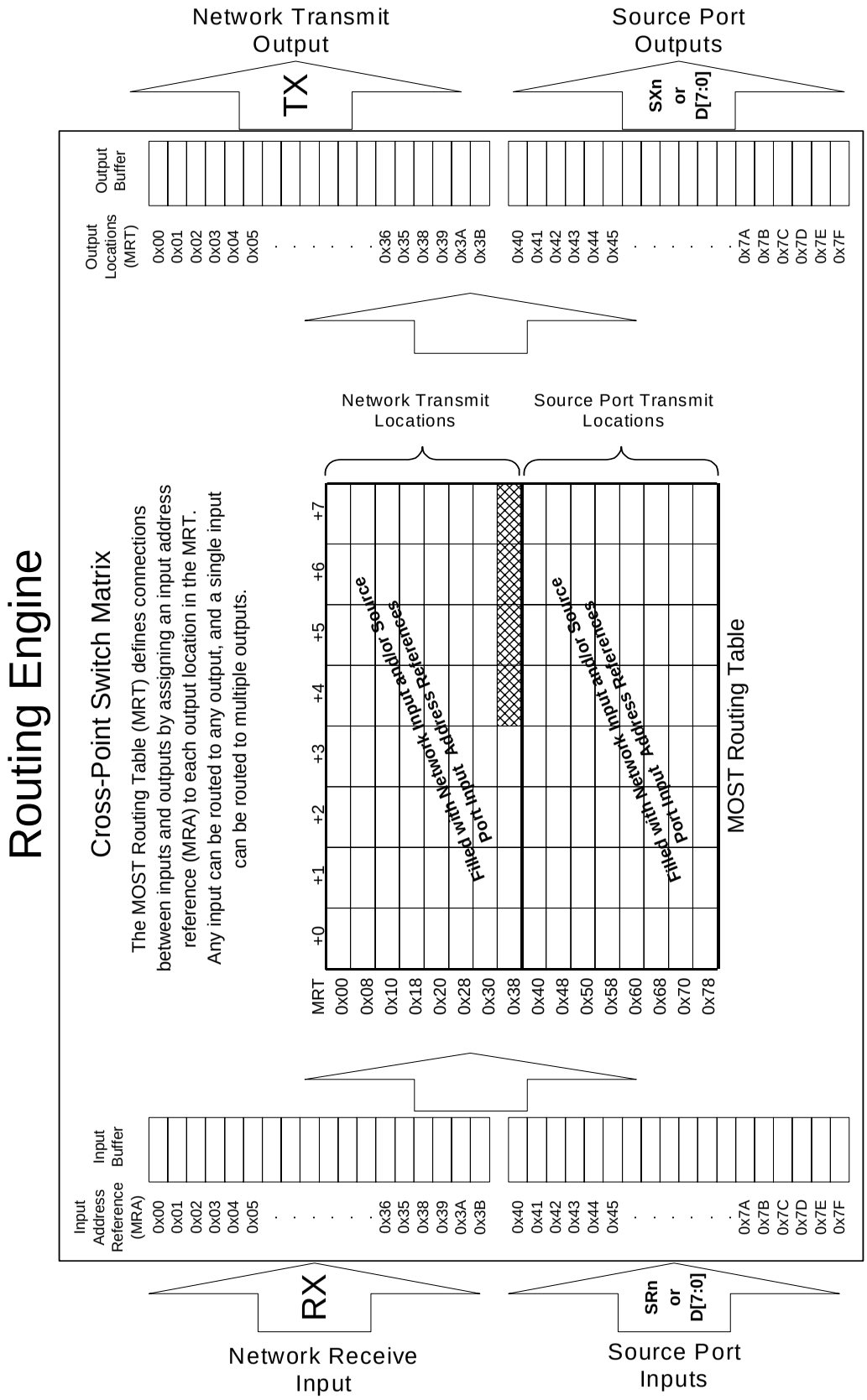


Figure 12-1: Routing Engine

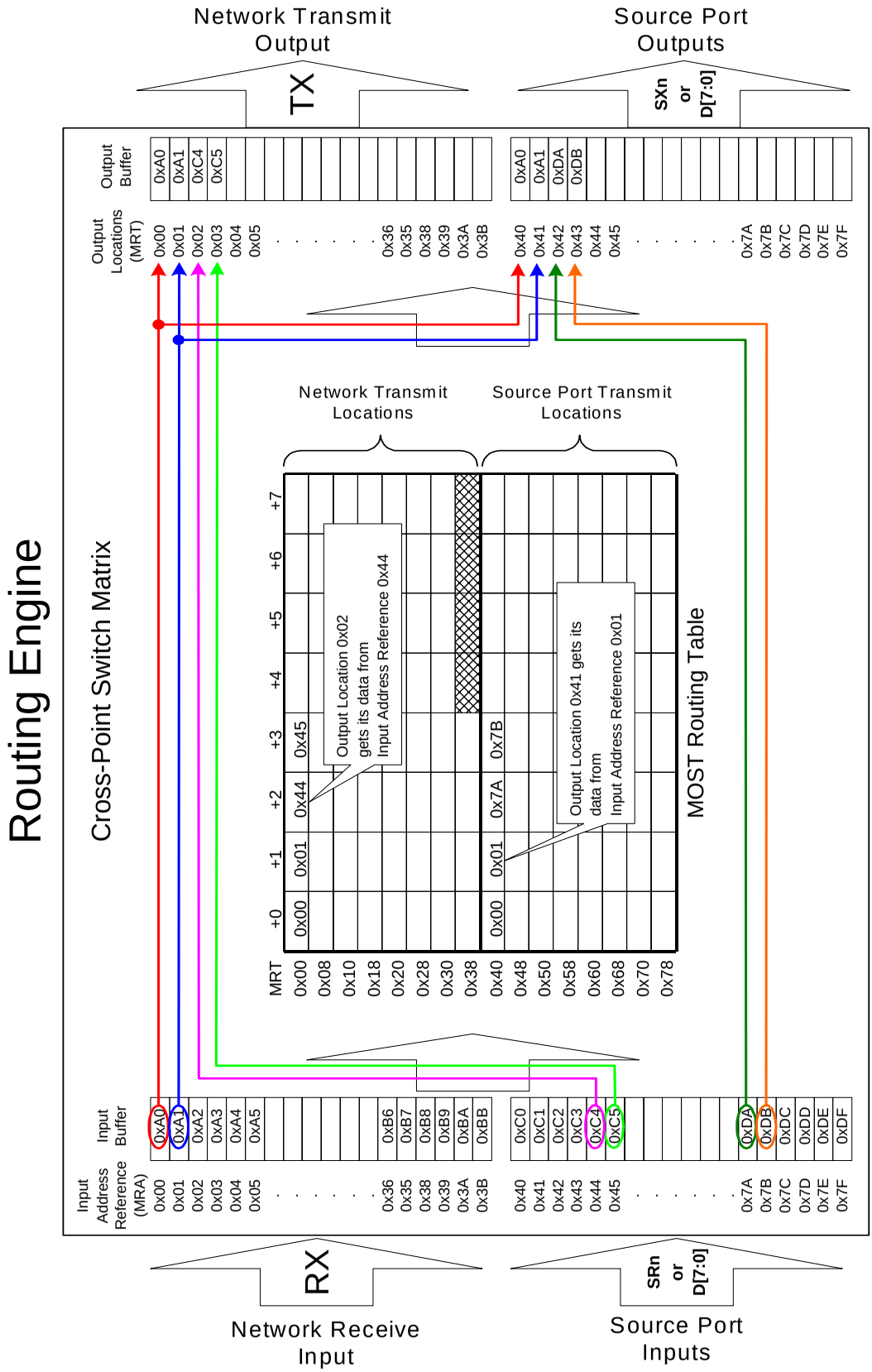


Figure 12-2: Routing Process Example

12.1 Incoming Network Data to Outgoing Network Data

Each MOST frame contains a maximum of 60 bytes of source data. When routing data from the incoming frame to the outgoing frame, the respective address reference value (MRA) is equal to the MRT location (valid numbers are 0x00 through 0x3B). For example, to output byte 8 of the received data to byte 8 of the transmitted data, address reference MRA0x08 must be written to MRT location MRT0x08 (transmitted location).

Table 12-1 illustrates the lower half of the MRT at power-up. The locations MRT0x00 to MRT0x3B reference synchronous data locations on the outgoing MOST Network frame. In this table, each MRT location is filled with the corresponding input address reference (MRA), such that the MOST Network source data is passed from input (**RX**) to output (**TX**) unaltered.

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7
0x00	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
0x08	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x10	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
0x18	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
0x20	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27
0x28	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F
0x30	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
0x38	0x38	0x39	0x3A	0x3B				

Table 12-1: Lower Half of MRT (Network Transmit Locations)

For example, to send byte 0 of the incoming frame to byte 17 in the outgoing frame:

- Determine the address reference of byte 0 in the incoming frame (address reference MRA0x00)
- Determine the MRT location that refers to byte 17 of the outgoing frame (MRT location MRT0x11)
- Write the address reference MRA0x00 to MRT location MRT0x11.

Originally, incoming channel 17 was transmitted on outgoing channel 17. After writing MRA0x00 to MRT0x11, the content of incoming channel 0 is transmitted on outgoing channel 17, and the incoming channel 17 is discarded (unless it is routed elsewhere).

12.2 Serial Source Port Inputs to Network

Since the Serial Source Port inputs are *data sources*, they are associated with address references that need to be placed into the MRT to transfer data. The address references used for the Source Port inputs (**SRn** pins) are dependent on the selected speed of the Source Ports. Table 12-2 is used to determine address references for a given Source Port configuration.

Source Port Inputs — Address References (MRA)									
Bytes/Frame	**64**	**32**	**16**	**8**	**6**		**4**	**2**	**1**
Bit Rate (Fs)	512	256	128	64	48†		32	16	8
FSY Direction	In	In/Out	In/Out	In/Out	In	Out	In/Out	In/Out	In/Out
SR n	0*	0	0,2	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3	0,1,2,3
Addressing Index x (hex)	47/46	43	41	40	40	40	48	58	78
	45/44	42							
	43/42	41	40						
	41/40	40							
	4F/4E	4B	49	48	48	48			
	4D/4C	4A							
	4B/4A	49	48						
	49/48	48							
	57/56	53	51	50			58		
	55/54	52							
	53/52	51	50		50	58			
	51/50	50							
	5F/5E	5B	59	58					
	5D/5C	5A							
	5B/5A	59	58						
	59/58	58							
	67/66	63	61	60	58	60	68	78	
	65/64	62							
	63/62	61	60						
	61/60	60							
	6F/6E	6B	69	68	60	68			
	6D/6C	6A							
	6B/6A	69	68						
	69/68	68							
	77/76	73	71	70	68	78	78		
	75/74	72							
	73/72	71	70						
	71/70	70							
	7F/7E	7B	79	78					
	7D/7C	7A							
	7B/7A	79	78						
	79/78	78							
Address Reference (MRA) = x + n									

* **SR0** in 8x S/PDIF input mode (**bSDC3.SIO** clear). All numbers are in hexadecimal.

† If **bSDC1.IO** is set, the *out* column applies; otherwise, the *in* column applies.

Table 12-2: Source Port Input (SRn) Address References (MRA)

The first two rows specify the speed (bytes per frame), and the bit rate. The third row specifies the direction of **FSY**. The fourth row specifies which of the Source Port input pins are available at the respective clock rate. The address reference used for a particular **SRn** location is calculated using the formula at the bottom of Table 12-2, and is equal to the number in the table, offset by the particular Source Port pin number (**SRn**).

Network resources should be reserved (allocated) before routing source data to the Network. Reserving source data resources is accomplished through the *Resource Allocate* Control message. See Section 13.5.2.4 on page 126 for more information.

Since the synchronous data bytes of the outgoing Network frame are assigned to MRT locations MRT0x00 to MRT0x3B, an address reference written to one of these locations routes the respective source data input byte to the outgoing frame.

Using an example where the Source Ports are configured in Mode 3 (Source Ports 0 and 2 running at 128Fs), if bytes 0 to 2 of the incoming data at Source Port 2 should be transmitted in byte position 57 to 59 in the outgoing frame, the following steps must be performed:

- Determine the address reference of byte 0 to 2 at Source Port 2 in Mode 3.
  - The correct values are MRA0x43, MRA0x42, MRA0x4B
- Determine which MRT locations refer to bytes 57 to 59 of the outgoing frame.
  - The correct values are MRT0x39, MRT0x3A, MRT0x3B

Write the address references to the respective MRT locations as follows:

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7
0x00								
0x08								
0x10								
0x18								
0x20								
0x28								
0x30								
0x38		0x43	0x42	0x4B				

Table 12-3: Lower Half of MRT (Example)

The number of available synchronous data bytes per frame is specified in register bSBC (Section 6.2.5 on page 40). If all source bytes of the MOST frame are not reserved for synchronous data transfer, the respective bytes in the MRT must not be used.

12.3 Network to Serial Source Port Outputs

Similar to the previous section, the speed of the Source Ports determines the address reference (MRA) used for routing incoming Network data out of the serial Source Ports (**SXn**). When routing data to the Source Port output pins, two values must be determined: the address reference for the data to be shifted out, and the appropriate MRT location the address reference should be written to. Table 12-4 is used to determine the MRT locations for a specific Source Port configuration.

<table>
<tr><th colspan="11">Source Port Outputs — MRT Locations</th></tr>
<tr><td>Bytes/Frame</td><td>64</td><td>32</td><td>16</td><td>8</td><td colspan="2">6</td><td>4</td><td>2</td><td>1</td></tr>
<tr><td>Bit Rate (Fs)</td><td>512</td><td>256</td><td>128</td><td>64</td><td colspan="2">48†</td><td>32</td><td>16</td><td>8</td></tr>
<tr><td>FSY Direction</td><td>Out</td><td>In/Out</td><td>In/Out</td><td>In/Out</td><td>In</td><td>Out</td><td>In/Out</td><td>In/Out</td><td>In/Out</td></tr>
<tr><td>SX n</td><td>0*</td><td>0</td><td>0,2</td><td>0,1,2,3</td><td>0,1,2,3</td><td>0,1,2,3</td><td>0,1,2,3</td><td>0,1,2,3</td><td>0,1,2,3</td></tr>
<tr><td rowspan="32">Addressing Index x (hex)</td><td>47/46</td><td>47</td><td rowspan="2">47</td><td rowspan="4">47</td><td rowspan="8">47</td><td rowspan="8">47</td><td rowspan="16">47</td><td rowspan="32">47</td><td rowspan="32">47</td></tr>
<tr><td>45/44</td><td>46</td></tr>
<tr><td>43/42</td><td>45</td><td rowspan="2">46</td></tr>
<tr><td>41/40</td><td>44</td></tr>
<tr><td>4F/4E</td><td>4F</td><td rowspan="2">4F</td><td rowspan="4">4F</td></tr>
<tr><td>4D/4C</td><td>4E</td><td rowspan="8">4F</td><td rowspan="8">4F</td></tr>
<tr><td>4B/4A</td><td>4D</td><td rowspan="2">4E</td></tr>
<tr><td>49/48</td><td>4C</td></tr>
<tr><td>57/56</td><td>57</td><td rowspan="2">57</td><td rowspan="4">57</td><td rowspan="16">57</td></tr>
<tr><td>55/54</td><td>56</td></tr>
<tr><td>53/52</td><td>55</td><td rowspan="2">56</td></tr>
<tr><td>51/50</td><td>54</td></tr>
<tr><td>5F/5E</td><td>5F</td><td rowspan="2">5F</td><td rowspan="4">5F</td><td rowspan="8">57</td><td rowspan="8">5F</td></tr>
<tr><td>5D/5C</td><td>5E</td></tr>
<tr><td>5B/5A</td><td>5D</td><td rowspan="2">5E</td></tr>
<tr><td>59/58</td><td>5C</td></tr>
<tr><td>67/66</td><td>67</td><td rowspan="2">67</td><td rowspan="4">67</td><td rowspan="8">67</td><td rowspan="8">67</td><td rowspan="16">67</td></tr>
<tr><td>65/64</td><td>66</td></tr>
<tr><td>63/62</td><td>65</td><td rowspan="2">66</td></tr>
<tr><td>61/60</td><td>64</td></tr>
<tr><td>6F/6E</td><td>6F</td><td rowspan="2">6F</td><td rowspan="4">6F</td></tr>
<tr><td>6D/6C</td><td>6E</td><td rowspan="8">67</td><td rowspan="8">6F</td></tr>
<tr><td>6B/6A</td><td>6D</td><td rowspan="2">6E</td></tr>
<tr><td>69/68</td><td>6C</td></tr>
<tr><td>77/76</td><td>77</td><td rowspan="2">77</td><td rowspan="4">77</td><td rowspan="16">77</td></tr>
<tr><td>75/74</td><td>76</td></tr>
<tr><td>73/72</td><td>75</td><td rowspan="2">76</td></tr>
<tr><td>71/70</td><td>74</td></tr>
<tr><td>7F/7E</td><td>7F</td><td rowspan="2">7F</td><td rowspan="4">7F</td><td rowspan="8">6F</td><td rowspan="8">7F</td></tr>
<tr><td>7D/7C</td><td>7E</td></tr>
<tr><td>7B/7A</td><td>7D</td><td rowspan="2">7E</td></tr>
<tr><td>79/78</td><td>7C</td></tr>
<tr><td colspan="11">MRT Location = x – n</td></tr>
</table>

* **SX0** in 8x S/PDIF output mode (**bSDC3.SIO** set). All numbers are in hexadecimal.

† If **bSDC1.IO** is set, the *out* column applies; otherwise, the *in* column applies.

Table 12-4: Source Port Output (SXn) MRT Locations

The first row contains the number of bytes shifted out per frame. The second row contains the respective serial clock rate. The third row specifies the direction of **FSY**, and the fourth row specifies which of the Source Port output are available at the respective clock rate. The MRT location used for a particular **SXn** location is calculated using the formula at the bottom of the Table 12-4, and is equal to the number in the table, offset by the particular Source Port pin number (**SXn**).

The address reference that must be written to the MRT depends on the source of the data. If the source data comes from a serial Source Port input pin (**SRn**), Table 12-2 must be used. If the data comes from the incoming MOST frame, the address references MRA0x00 to MRA0x3B are relevant (see Table 12-1).

Figure 12-3 shows the conjunction between the entries in Table 12-2 and the incoming bytes at a serial Source Port input pin when **SCK** is configured for 64Fs. The table is turned counterclockwise (90 degrees) and the two left-most data columns are removed for readability.

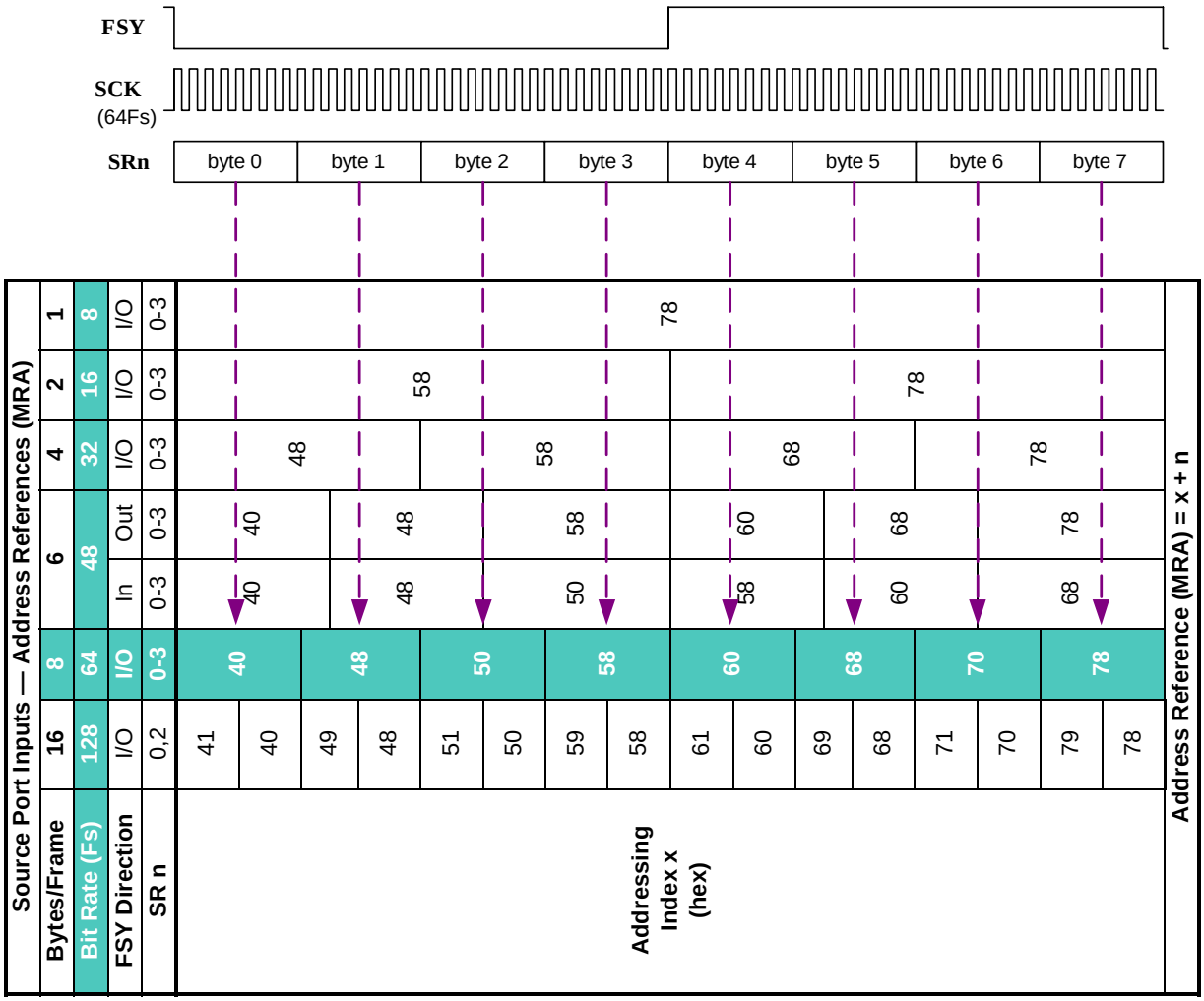


Figure 12-3: Source Port Serial Format: Table vs. SRn Example

The same applies to Table 12-4. When the table is turned counterclockwise (90 degrees), the sequence of entries in the table corresponds to the sequence of data bytes in the serial Source Port output frame (**SCK** speed must always be taken into consideration).

For example, consider the Source Ports configured in Mode 1 and running at a 64Fs bit rate. If byte 4 of the incoming MOST frame is to be the second byte shifted out of **SX3**, the MRT location (relative to the second byte of **SX3**) must be calculated at the appropriate speed.

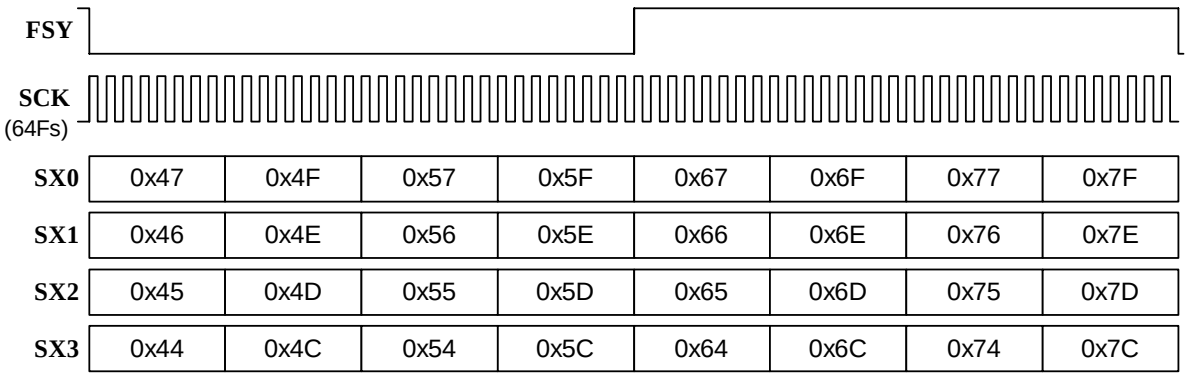
Using Table 12-4, the MRT location is MRT0x4C (0x4F – 3). Since byte 4 of the incoming MOST frame has address reference MRA0x04, the MRT would look like:

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7
0x40								
0x48					0x04			
0x50								
0x58								
0x60								
0x68								
0x70								
0x78								
					SX3	SX2	SX1	SX0

Table 12-5: Upper Half of MRT (Example)

Table 12-5 illustrates the MRT locations used to route data out of Source Port 3 (**SX3**), when configured in Mode 1 (**SCK** speed of 64Fs). The eight locations for **SX3**, starting at the beginning of **FSY**, are MRT0x44 for the first byte, MRT0x4C for the second byte, MRT0x54 for the third byte, up to MRT0x7C for the eighth byte.

Figure 12-4 illustrates the MRT locations associated with each byte of the Source Port outputs (**SXn**) in Mode 1 (**SCK** speed of 64Fs).



SX0	0x47	0x4F	0x57	0x5F	0x67	0x6F	0x77	0x7F
SX1	0x46	0x4E	0x56	0x5E	0x66	0x6E	0x76	0x7E
SX2	0x45	0x4D	0x55	0x5D	0x65	0x6D	0x75	0x7D
SX3	0x44	0x4C	0x54	0x5C	0x64	0x6C	0x74	0x7C

Figure 12-4: Source Port Output Routing Example (MRT Locations)

12.4 Synchronous Parallel Port Data Transfers

When the Source Ports are configured in Parallel-Synchronous or Parallel-Combined mode, the Source Port pins form an 8-bit parallel port that transfers data through a FIFO that is eight bytes deep. Each frame is divided into eight intervals of identical length, designated SF0 to SF7. During each SF interval, a total of 16 bytes can be transferred (8 received, 8 sent), for a total of 128 bytes per frame. (For more detailed descriptions of the Parallel-Synchronous and Parallel-Combined modes, see Section 8.2.1 on page 66 and Section 8.3 on page 74). For source data transferred from an external device, through the parallel port, and onto the Network, address references for each of the parallel bytes are needed in the lower half of the MRT to determine what is transmitted onto the MOST Network (TX). For source data transferred from the MOST Network receiver, through the parallel port, and to an external device, the upper half of the MRT must be programmed with the address references of the received Network data. Each location in the upper half of the MRT corresponds to one of the 64 parallel port bytes that can be transferred each frame.

12.4.1 Synchronous Parallel Data To Network

When receiving data from an external device, through the parallel port, in Parallel-Synchronous or Parallel-Combined mode, a maximum of 64 bytes per frame can be written by the external device. For assigning a particular parallel port byte to one of the available positions within the MOST Network transmit frame (the maximum number of source data bytes in the frame is 60), address references for the parallel port data are needed. Table 12-6 shows the relation between the position of a byte in the FIFO, the SF interval during which the byte is transferred, and the address reference (MRA) needed in the lower half of the MRT.

SRC_FLOW Interval	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
SF0	0x40	0x41	0x42	0x43	0x44	0x45	0x46	0x47
SF1	0x48	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F
SF2	0x50	0x51	0x52	0x53	0x54	0x55	0x56	0x57
SF3	0x58	0x59	0x5A	0x5B	0x5C	0x5D	0x5E	0x5F
SF4	0x60	0x61	0x62	0x63	0x64	0x65	0x66	0x67
SF5	0x68	0x69	0x6A	0x6B	0x6C	0x6D	0x6E	0x6F
SF6	0x70	0x71	0x72	0x73	0x74	0x75	0x76	0x77
SF7	0x78	0x79	0x7A	0x7B	0x7C	0x7D	0x7E	0x7F

Table 12-6: Parallel Port Synchronous Data Input Address References (MRA)

For example, to route the data positioned in *byte 5* during SF interval 6 to the outgoing frame, the correct address reference is MRA0x75. To send this byte onto the first byte of the outgoing Network frame, MRA0x75 must be written to MRT0x00 (first MRT location).

12.4.2 Network to Synchronous Parallel Data

Likewise, when sending data to an external device, through the parallel port, in Parallel-Synchronous or Parallel-Combined mode, a maximum of 64 bytes per frame can be read by the external device. The upper half of the MRT is mapped (using SF intervals and FIFO byte position) to each of the parallel port bytes. Filling these upper MRT locations with the address references of the Network receive data maps the data to the parallel port.

Table 12-7 shows the relationship between the MRT location and parallel port bytes, based on the SF interval and the particular FIFO byte number. Generally, all these locations are filled with the full Network bandwidth (address references MRA0x00 through MRA0x3B). In Parallel-Combined mode, the last four locations are generally used for the asynchronous status bytes (MRA0x3C through MRA0x3F), see Section 8.3.3.1 on page 77 for more information.

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7	SF:
0x40									← 0
0x48									← 1
0x50									← 2
0x58									← 3
0x60									← 4
0x68									← 5
0x70									← 6
0x78									← 7
	↑	↑	↑	↑	↑	↑	↑	↑	
FIFO byte:	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	

Table 12-7: Upper Half of MRT for Parallel Port Output Data

For example, to route the first eight bytes of the received MOST frame to the Source Port in Parallel-Synchronous mode, one byte will be sent per SF interval. This meets the requirement that at least one read or write access must occur during each SF interval. When reading from the parallel port, FIFO *byte 7* is the first byte read. Therefore, FIFO *byte 7* of each SF interval will be used to transfer the received MOST Network data. The bottom of Table 12-7 specifies which column refers to which byte within the FIFO. The right-most column gives information about the respective SF interval.

Using the FIFO *byte 7* column of Table 12-7, the address references for the first eight Network receive bytes (MRA0x00-MRA0x07) are placed in the MRT as shown in Table 12-8.

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7	SF:
0x40								0x00	← 0
0x48								0x01	← 1
0x50								0x02	← 2
0x58								0x03	← 3
0x60								0x04	← 4
0x68								0x05	← 5
0x70								0x06	← 6
0x78								0x07	← 7

Table 12-8: Parallel-Synchronous Source Data Routing Example

12.5 Transparent Channel Data Routing

When the transparent channel is enabled, the selected sample rate will determine the number of bits sampled at **SR1** or transmitted on **SX1**. The sample rate is calculates as follows:

$$\text{Sample rate} = \frac{\textbf{bSDC2.SPR[2:0]} \text{ setting}}{\textbf{bSDC2.TCR[1:0]} \text{ setting}}$$

At the highest sample rate, 8 bytes are available per frame. Only one byte per frame is available at the lowest sample rate. When routing transparent data from **SR1** to the outgoing MOST frame, Table 12-9 lists the address references for the selected sample rate.

Transparent Channel Input (SR1) to MOST Network				
Bytes/Frame	8	4	2	1
Sample rate (Fs)	64	32	16	8
Address Reference (MRA)	0x41	0x49	0x59	0x79
	0x49			
	0x51	0x59		
	0x59			
	0x61	0x69	0x79	
	0x69			
	0x71	0x79		
	0x79			

Table 12-9: SR1 Transparent Channel Address References (MRA)

When routing transparent data from the incoming MOST frame to **SX1**, the Network receive data address reference must be programmed into the appropriate MRT location. Table 12-10 lists the MRT locations for the selected sample rate.

Transparent Channel Output (SX1) from MOST Network				
Bytes/Frame	8	4	2	1
Sample rate (Fs)	64	32	16	8
MRT Location	0x46	0x46	0x46	0x46
	0x4E			
	0x56	0x56		
	0x5E			
	0x66	0x66	0x66	
	0x6E			
	0x76	0x76		
	0x7E			

Table 12-10: SX1 Transparent Channel MRT Locations

For example, in two MOST nodes the transparent channel is configured to run at an 8Fs sample rate (one byte of transparent data per frame). The sample rate of 8Fs is a result of **SCK** running at 64Fs (**bSDC2.SPR[2:0]** = 011) and the **SCK** divider set to 8 (**bSDC2.TCR[1:0]** = 11). The first node (Node 1) receives a signal via the **SR1** pin, and puts the data onto physical channel 0 of the Network frame. The second node (Node 2) takes the data from the received physical channel 0, and restores the signal at **SX1**.

Using the 8Fs sample rate column in Table 12-9 and Table 12-10, Node 1 must program MRT location MRT0x00 (physical channel 0) to address reference MRA0x79. Node 2 must program MRT location MRT0x46 to address reference MRA0x00 (physical channel 0 of incoming Network frame). With this setup, data on the **SR1** pin of Node 1 will be transmitted across the Network and output on the **SX1** pin of Node 2.

12.6 Address Reference 0xF8

The address reference MRA0xF8 sets the respective target byte to zero. To force a zero to be sent to a destination (sink device), set the appropriate MRT location to MRA0xF8. An MRA0xF8 in the lower half of the MRT transmits a zero to the appropriate byte output on the MOST Network. A MRA0xF8 in the upper half of the MRT transmits a zero to the appropriate Source Port output pin (**SXn**) in serial mode, or to the appropriate FIFO byte in parallel mode.

12.7 MRT Power-up Defaults

The power-up default values for the MRT are illustrated in Figure 12-5. The MOST Network transmit section of the MRT (lower half) is filled with the corresponding MOST receive bytes. Therefore, when the transceiver is taken out of *All-bypass* mode, all the bytes received from the MOST Network will be retransmitted in the same location. The Source Port section of the MRT (upper half) contains MRA0xF8 by default, which sends zeros to all Source Ports that are enabled.

MRT Locations	+0	+1	+2	+3	+4	+5	+6	+7	
0x00	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	MOST Network Transmit Locations
0x08	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	
0x10	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	
0x18	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F	
0x20	0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	
0x28	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F	
0x30	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	
0x38	0x38	0x39	0x3A	0x3B					
0x40	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	Source Port (**SXn** and **D[7:0]**) Transmit Locations
0x48	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x50	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x58	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x60	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x68	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x70	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	
0x78	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	0xF8	

Figure 12-5: MRT Power-up Defaults

12.8 Routing Limitations

Certain routing limitations exist when configuring the MRT to route directly from **SRn** to **SXn** pins. No limitations exist when routing **SRn** pins to the MOST Network, routing the MOST Network to **SXn** pins, or routing from the receive to transmit portions of the MOST Network. The routing limitations exist due to the asynchronous nature of **FSY** (input or output), which **SRn** and **SXn** are synchronized to, and the MOST Network receive and transmit start-of-frame.

To guarantee that all transmitted bytes are equally delayed (enter the chip at the same time - or are coherent), MRT locations MRT0x40h to MRT0x47 (associated with **SXn** pins) should not contain any **SRn** address references between MRT0x78 and MRT0x7F. For systems that do not comply with the *MOST Specification of Physical Layer*, the delay from transmit start-of-frame (**TX**) to receive start-of-frame (**RX**) should be limited to less then 9/16 of a frame. This condition applies to the timing-master node. If the *MOST Specification of Physical Layer* is met, a Network delay this large cannot occur.

13 Control Messages

In addition to transferring synchronous and asynchronous data, a MOST Network supports a special messaging service. Control messages can get status information about devices connected to the MOST Network, and can control devices remotely. Traffic on the Control message channel does not influence the bandwidth of source data transfer. The Control message service supports many different message types. Figure 13-1 shows the existing types of messages.

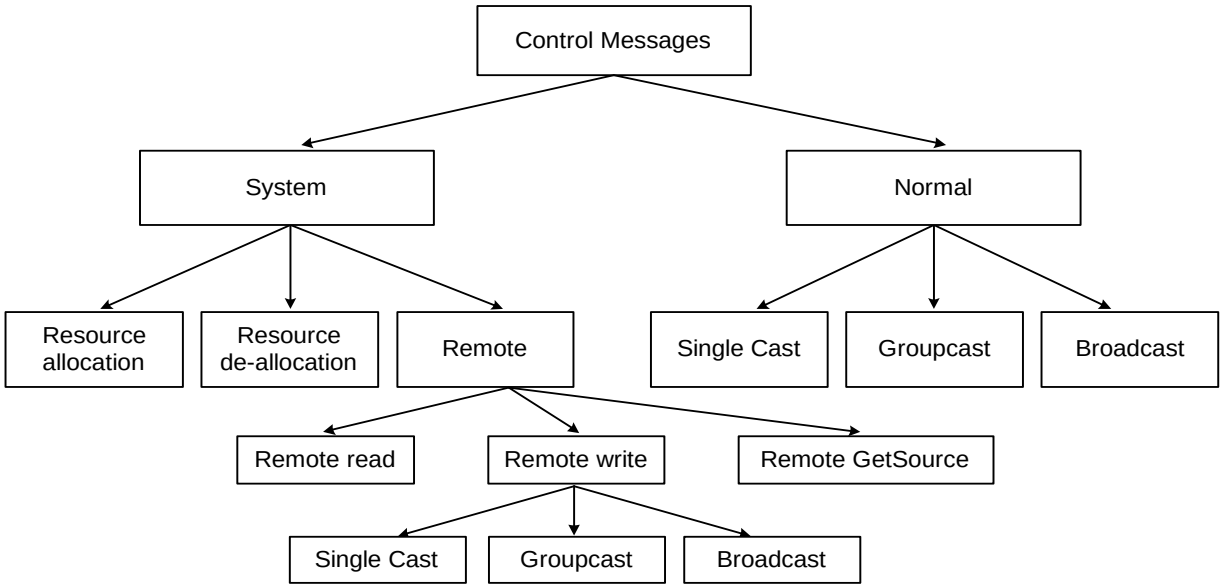


Figure 13-1: Control Message Types

For example, normal messages can control the functionality of a CD player. Normal messages can be sent to a single node (single cast), to a group of nodes (groupcast), or to all nodes in a Network (broadcast).

System messages handle MOST related functions such as resource management (channel allocation/de-allocation) and remote operation of a MOST transceiver (remote read/write, finding nodes routing source data on allocated channels).

The kind of transmission (single cast, groupcast or broadcast) is determined through special addresses or address ranges. System messages of type *Resource Allocate* or *Resource De-allocate* are always directed to the timing-master node and should not be sent to other nodes.

Control messages have a maximum length of 17 bytes. The maximum number of retries, the retry time interval, and the message priority can be configured through internal registers. The built-in arbitration guarantees fair handling of the messages. An acknowledge mechanism helps track the status of a message, and the contents of a Control message are protected by CRC (Cyclic Redundancy Check).

The following registers are used when transmitting or receiving Control messages:

- bNAH, bNAL — Node Address registers (Logical Address register)
- bGA — Group Address register (determines Groupcast address)
- bMSGC — Message Control register
- bMSGS — Message Status register
- bXTS — Transmit Transfer Status register
- bXTRY — Transmit Retry register
- bXTIM — Transmit Retry Time register
- mRCMB — Receive Control Message Buffer
- mXCMB — Transmit Control Message Buffer
- bIE — Interrupt Enable register

13.1 Transmit Message Addressing

Transmit messages can be sent to the target node using the following address types:

- Node Address (Logical Addressing)
- Node Position Address (Physical Addressing)
- Group Address/Groupcast
- Special Group Address/Broadcast

13.1.1 Node Address (Logical Addressing)

Each OS8104 has a node address (logical address), which is stored in two registers: Node Address High (bNAH) and Node Address Low (bNAL). After reset, the default node address is 0FFFh. Valid logical addresses are 0001h through 02FFh, and 0500h through FFFFh. The gap in the address space is due to other addressing modes.

The device address can either be set manually by application software, or through the SAI (Start Address Initialization) procedure. The SAI procedure provides automatic testing to determine whether the desired address is unique, and automatic setting of bNAH and bNAL if the address is okay. When using SAI, the low byte of the desired address must be written to C3h (bXTAL register in mXCMB) and the high byte of the desired address must be written to C2h (bXTAH register in mXCMB), then **bMSGC.SAI** must be set. After verification, the OS8104 sets **bMSGS.MTX** (Message transmitted), and reports the result of the SAI procedure by using **bMSGS.TXR**. If **TXR** is set, it indicates that the address was unique and successfully stored in the bNAH and bNAL registers. If **TXR** is clear, it indicates that an error occurred during transmission and the respective error code is stored in bXTS.

13.1.2 Node Position Address (Physical Addressing)

Each node in a Network has a unique node position address (physical address), which is determined by its position in the Network. Counting starts at the timing-master node with node position 00h. The Node Position address register (bNPR) is automatically configured by the OS8104 after lock is established.

The valid address range for Node Position addressing is 0400h through 04FFh. The destination address for sending Control messages to a single node by accessing its physical address, is calculated as follows:

Node physical address = 0400h + bNPR

The bNPR register stores the Node Position Address, which is the lower byte used in the physical address calculation (see Section 6.2.7 on page 41).

13.1.3 Group Address/Groupcast

Several nodes in the MOST Network can be combined into a group by setting their Group Address register bGA to the same value. The bGA register stores the lower byte of the Groupcast address and the upper byte of the Groupcast address is 03h. For sending a Control message to a group of nodes, the groupcast address in the sending node is calculated as follows:

Node groupcast address = 0300h + bGA

The valid address range for Group Addressing is 0300h through 03C7h and 03C9h through 03FFh. Address 03C8h is reserved for a Network-wide broadcast address.

When using groupcast addressing, the number of available bytes to send is 16. One byte is used to secure groupcast.

13.1.4 Special Broadcast Address

The address value 03C8h is reserved for sending messages to all nodes in the MOST Network. During the sending of a broadcast message, all other Control message transmissions are delayed until the end of the broadcast process, to ensure reception of the message at each node. Therefore, excessive use of broadcast messages is discouraged as it decreases the overall bandwidth of the Control message service.

13.1.5 Address Ranges vs. Addressing Modes

When transmitting messages, there are four addressing modes available for reaching individual nodes as well as groups of nodes. The addressing modes differ by the address range:

Address	Addressing Mode
0001h..02FFh; 0500h..FFFFh	Sending messages to a single node. (Logical addressing) Address 0000h is not valid.
0300h..03C7h; 03C9h..03FFh	Sending messages to a group of nodes. (Groupcast addressing)
03C8h	Sending messages to all nodes. (Broadcast addressing)
0400h..04FFh	Sending messages to a single node based on node position. (Physical addressing) The Node Position Register contains the current node position number. The timing-master node always has position number 00h, so that the timing-master's address would be 0400h.

Table 13-1: Address Ranges vs. Addressing Modes

When sending Control messages via Broadcast addressing, the sending of other Control messages is disabled until all nodes have received the broadcast message. Excessive use of broadcast addressing keeps the Network from sending normal Control messages.

13.1.6 bNAH, bNAL (Node Address Registers)

The bNAH and bNAL registers contain the node logical address used for Control messages. After reset, the default address is 0FFFh. A node address of 0000h is not allowed.

8Ah bNAH Node Address High Register

Bit	Name	Description	Default
7..0	NA[15:8]	Logical Node address high byte	0Fh

Table 13-2: bNAH (Node Address High Register)

8Bh bNAL Node Address Low Register

Bit	Name	Description	Default
7..0	NA[7:0]	Logical Node address low byte	FFh

Table 13-3: bNAL (Node Address Low Register)

13.1.7 bGA (Group Address Register)

89h bGA Group Address Register

Bit	Name	Description	Default
7..0	GA[7:0]	Lower byte of the Group address. 03h is the upper address byte.	*00h

* When coming out of reset in Stand-Alone mode, the default value of bGA is F0h.

Table 13-4: bGA (Group Address Register)

The value in the bGA register specifies the group a node belongs to. The OS8104 provides 254 different addresses (group address C8h is reserved for broadcast messages that all Network nodes respond to). A message sent to a group address is received by all nodes that have that particular group address (bGA value). If the Network supports the Broadcast message type, the transmit retry time in register bXTIM must be the same for all nodes.

13.2 Managing Control Messages

MOST Control messages are handled via two buffers. The Receive Control Message Buffer (mRCMB) contains received Control message data along with information about the sending node and the addressing mode used. The Transmit Control Message Buffer (mXCMB) is used when preparing messages to be sent out onto the Network. Along with the actual message, the addressing mode, the message type (normal or system), and the message priority must be specified.

The Message Control register (bMSGC) is used to manage Control messages, with the results stored in the Message Status register (bMSGS). Additional information about the transmit status is provided by the Transmit Status register (bXTS). The Transmit Retry register (bXRTY) and the Transmit Retry Time register (bXTIM) control the retry limit and retry interval for re-sending transmissions that previously failed.

13.2.1 bMSGC (Message Control Register)

This register controls sending and receiving of Control messages. The status of the operations triggered by bMSGC is reported in the Message Status register bMSGS. Figure 13-2 illustrates the logic associated with the **INT** and **ERROR** pins.

85h bMSGC Message Control Register (Write-only)

Bit	Name	Description	Default
7	STX	Start transmission	0
6	RBE	Receive buffer enable	0
5	rsvd	Reserved; Write as 0	0
4	SAI	Start address initialization	0
3	RALC	Reset Network configuration changed interrupt	0
2	RERR	Reset Error or Power-on interrupt	0
1	RMTX	Reset Message transmitted interrupt	0
0	RMRX	Reset Message received interrupt	0

Table 13-5: bMSGC (Message Control Register)

STX Start Transmission. When setting the **STX** bit, the message currently stored in the Transmit Control Message Buffer (mXCMB) is transmitted. The transceiver writes the result of the transmission operation into the **bMSGS.TXR** bit and to register bXTS.

RBE Receive Buffer Enable. Incoming Control messages are stored in the Receive Control Message Buffer (mRCMB). Once a message has been received, the reception of further messages is blocked until the **RBE** bit is set, releasing mRCMB. Therefore, once the contents of mRCMB are read, or if the application does not care about the message, the **RBE** bit must be set to allow further reception of Control messages. The **bMSGS.RBS** bit provides the current status of mRCMB. When the **RBE** bit is set, the **bMSGS.RBS** bit is cleared automatically after the transceiver finishes the releasing procedure.

SAI Start Address Initialization. The **SAI** bit can be used to validate a logical address. The desired node address must be written to mXCMB (bytes bXTAH/bXTAL) before the **SAI** bit is set. The transceiver then checks whether the desired address is valid and unique within the Network. If so, it is written to the Node Address registers (bNAH/bNAL), and the **bMSGS.TXR** bit is set to indicate success. **SAI** is cleared automatically.

RALC Reset *Network configuration changed* interrupt. Setting **RALC** resets **bMSGS.ALC** and the **INT** interrupt pin. The **RALC** bit is cleared automatically.

RERR Reset *Error or Power-on after start-up* interrupt. Setting **RERR** resets **bMSGS.ERR** and the **INT** interrupt pin. The **RERR** bit is cleared automatically.

RMTX Reset *Message transmitted* interrupt. Setting **RMTX** resets **bMSGS.MTX** and the **INT** interrupt pin. The **RMTX** bit is cleared automatically.

RMRX Reset *Message received* interrupt. Setting **RMRX** resets **bMSGS.MRX** and the **INT** interrupt pin. The **RMRX** bit is cleared automatically.

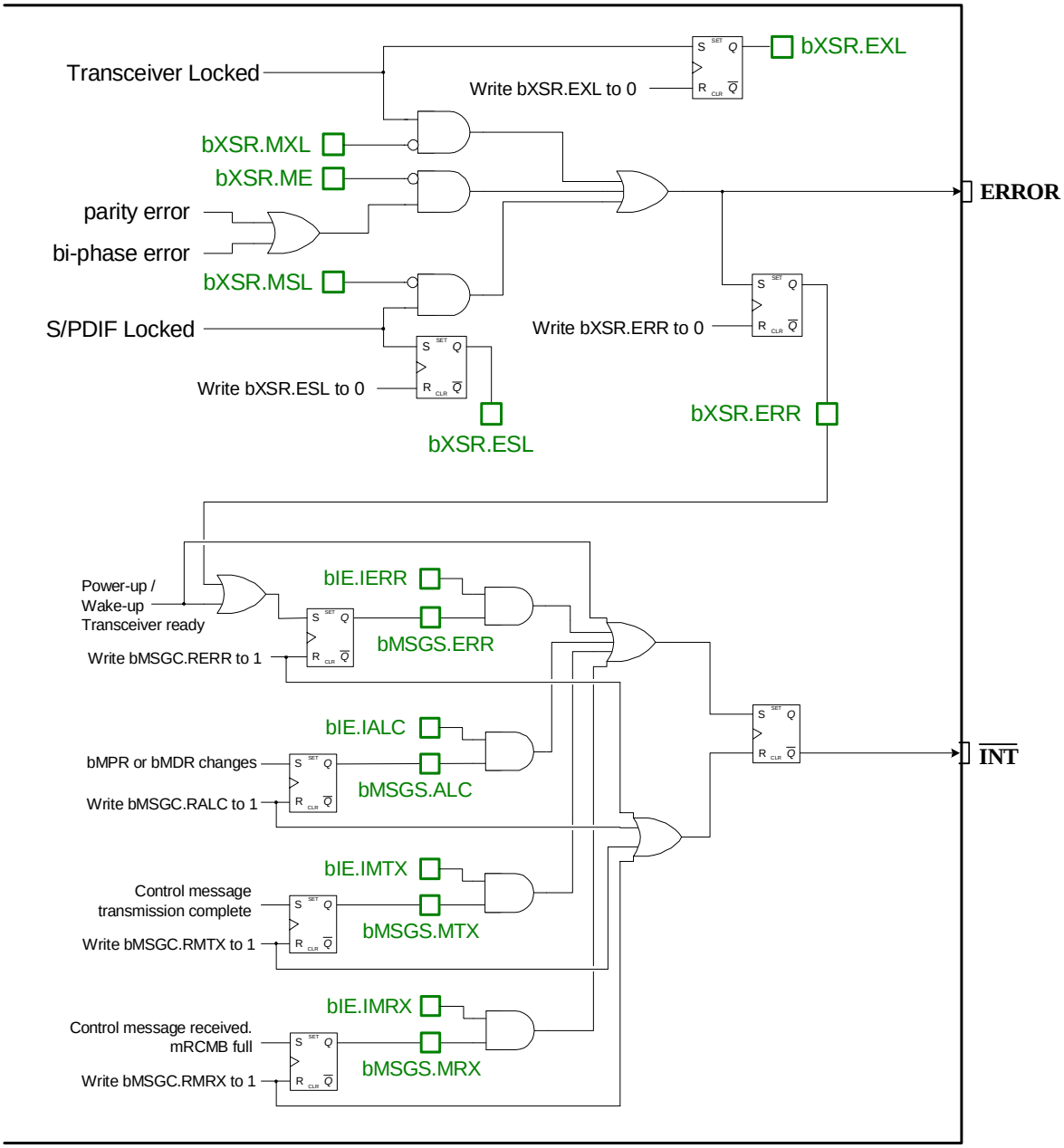


Figure 13-2: $\overline{INT}$ and ERROR Pins Logical Schematic

13.2.2 bMSGS (Message Status Register)

The Message Status Register bMSGS contains status of operations associated with the sending and receiving of Control messages. Figure 13-2 illustrates the logic associated with the **INT** and **ERROR** pins.

86h bMSGS Message Status Register (Read-only)

Bit	Name	Description	Default
7	RBS	Receive buffer status	0
6	TXR	Transmission result	0
5, 4	rsvd	Reserved; Write as 0	00
3	ALC	Network configuration changed	0
2	ERR	Error or Power-on initialization complete	1
1	MTX	Message transmitted	0
0	MRX	Message received	0

Table 13-6: bMSGS (Message Status Register)

RBS Receive Buffer Status. When clear, indicates that the Receive Control Message Buffer mRCMB is ready to receive a new Control message. When the **RBS** bit is set, mRCMB contains a message and no further messages will be received. mRCMB status is controlled via the **bMSGC.RBE** bit.

TXR Transmission Result. The **TXR** bit flags the status of all transmissions once **MTX** is set. The **TXR** bit set, indicates a successful transmission. If an error occurs, the **TXR** bit is cleared and the respective error code is stored in the bXTS register.

ALC Network Configuration Changed. The **ALC** bit set indicates that either bMPR (Maximum Position Register) or bMDR (Maximum Delay Register) has changed (or both). If **bIE.IALC** is set, an interrupt is generated when **ALC** is set. The **ALC** bit is cleared by setting the **bMSGC.RALC** bit.

ERR Error or Power-on initialization (or wake-up) complete. The **ERR** bit set indicates either the completion of power-on initialization, or an error event. If the **bIE.IERR** bit is set, an interrupt is generated when **ERR** is set. The bXSR register determines which error events affect **ERR**. The **ERR** bit is reset by setting **bMSGC.RERR**.

MTX Message Transmitted. The **MTX** bit set indicates that a message has been transmitted. When the **MTX** bit is set, the **TXR** bit indicates whether the transmission was successful. If an error occurs, the bXTS register indicates the actual error condition. If **bIE.IMTX** is set, an interrupt is generated when **MTX** is set. The **MTX** bit is cleared by setting **bMSGC.RMTX**.

MRX Message Received. The **MRX** bit set indicates that a Control message has been received. If **bIE.IMRX** is set, an interrupt is generated when **MRX** is set. The **MRX** bit is cleared by setting **bMSGC.RMRX**.

13.2.3 bXTS (Transmit Status Register)

This register contains the result of a transmission, including the error code in case of a transmission failure. For Groupcast and Broadcast messages, the transmission result stored in bXTS is the ORed acknowledge values of the last retry and all other receiving nodes.

D5h bXTS Xmit Transfer Status Register (Read-only)

Bit	Name	Description	Default
7..0	XTS[7:0]	Transmission result	00h

Table 13-7: bXTS (Transmit Status Register)

XTS[7:0] Transmission result. Possible values of the bXTS register are:
00h — Transmission failed. No response from the target node.
10h — Transmission successful.
11h — Transmission successful; however, receiving node does not support the message type used. Nodes with **bNC.RWD** set will return this status in response to a *Remote Write* request.
20h — Transmission failed because of a bad CRC at the receiving node.
21h — Transmission failed because the receiving node's receive buffer was full.

30h — For Groupcast and Broadcast messages, this is a possible result of ORing varying acknowledge values between receiving nodes (e.g. one receiving node acknowledge is 10h and a second receiving node acknowledge is 20h).
31h — For Groupcast and Broadcast messages, this is a possible result of ORing varying acknowledge values between receiving nodes (e.g. one receiving node acknowledge is 11h and a second receiving node acknowledge is 20h).

13.2.4 bXRTY (Transmit Retry Register)

BFh bXRTY Transmit Retry Register

Bit	Name	Description	Default
7..0	XRTY[7:0]	Total transmission attempts	06h

Table 13-8: bXRTY (Transmit Retry Register)

This register defines the total number of times the transceiver can attempt a given transmission. This value includes the first attempt and any retransmission attempts after the first transmission fails. The default value of 06h will lead to a maximum of five additional attempts, if the original transmission fails. The minimum value is 01h and the maximum value is FFh.

13.2.5 bXTIM (Transmit Retry Time Register)

BEh bXTIM Transmit Retry Time Register

Bit	Name	Description	Default
7..0	XTIM[7:0]	Time between transmission retries. Valid values are 03h to FDh.	0Bh

Table 13-9: bXTIM (Transmit Retry Time Register)

The bXTIM register specifies the delay between transmission retries when an initial transmission fails. The value represents the number of time units (CF block times) between the retries, where the time unit depends on the Network's frame rate (Fs). The formula for calculating the retry time is:

Retry time = <Time Unit> × bXTIM

The time units are approximately:

421 µs at Fs = 38 kHz
363 µs at Fs = 44.1 kHz
333 µs at Fs = 48 kHz

The minimum value in bXTIM is 03h. The maximum value is FDh. In a Network running at a frame rate of 44.1 kHz, the retry time for the default value 0Bh is:

363 µs × 11 = 3.99 ms

Therefore, if transmission of a message fails, the OS8104 waits approximately 4 ms before retrying the transmission.

All nodes in the Network must have the same bXTIM value for proper operation of Broadcast Messages.

13.2.6 mRCMB (Receive Control Message Buffer)

When a Control message is addressed to the local node, and has the correct CRC, it is stored in the Receive Control Message Buffer (mRCMB), and **bMSGS.RBS** is set.

A0h mRCMB Receive Control Message Buffer

Address	Name	Description	Default
A0h	bRTYP	Type of received Control Message	00h
A1h	bRSAH	Source address high	00h
A2h	bRSAL	Source address low	00h
A3h	bRCD0	Received Control data byte 0	00h
...	...	Received Control data bytes 1 through 15	00h
B3h	bRCD16	Received Control data byte 16	00h

Table 13-10: mRCMB (Receive Control Message Buffer)

bRTYP Received Control Message type. bRTYP indicates the address type of message received. The receive message types are described in more detail in Section 13.5.1. The definition of bRTYP is different from the definition of bXTYP in mXCMB.
00h — Logical Addressing (based on bNAH/bNAL)
01h — Physical Addressing (based on bNPR)
02h — Broadcast Addressing (using address 3C8h)
03h — Groupcast Addressing (based on bGA)

bRSAH Source Address High. bRSAH contains the high byte of the sending node's logical address (the node who sent the message).

bRSAL Source Address Low. bRSAL contains the low byte of the sending node's logical address.

bRCDn Received Control data byte n. The bytes bRCD0 to bRCD16 contain the currently received message. For Groupcast and Broadcast, the data bytes are bRCD4 to bRCD15.

13.2.7 mXCMB (Transmit Control Message Buffer)

The Transmit Control Message Buffer mXCMB contains the entire Control message to be sent.

C0h mXCMB Transmit Control Message Buffer

Address	Name	Description	Default
C0h	bXPRI	Priority for Xmit Control message	01h
C1h	bXTYP	Type of Xmit Control message	00h
C2h	bXTAH	Target address high	00h
C3h	bXTAL	Target address low	00h
C4h	bXCD0	Xmit Control data byte 0	00h*
...	...	Xmit Control data bytes 1 through 15	00h
D4h	bXCD16	Xmit Control data byte 16	00h

* After reset, bXCD0 through bXCD2 contain the OS8104 version number (refer to Section 16.4 on page 146)

Table 13-11: mXCMB (Transmit Control Message Buffer)

bXPRI Transmit Priority. bXPRI specifies the priority at which the transmit arbitration handles the message being sent. The value 00h represents the lowest, and 0Fh the highest priority. If several nodes try to send a message on the same priority level, the MOST Network grants access to the Control message channel based on a fair-arbitration algorithm. If a node sends a message on a higher priority level than all the other nodes, the MOST Network will immediately grant access to the next time slot for that Control message.

bXTYP Transmit Message Type. bXTYP determines the type of message to be sent. The meaning of bXTYP is different from that of bRTYP (in mRCMB) for received messages. bXTYP is described in more detail is Section 13.5.2.

00h — Normal message, using any of the various addressing options
01h — System message: *Remote Read*
02h — System message: *Remote Write* (can be blocked by nodes with **bNC.RWD** set)
03h — System message: *Resource Allocate*
04h — System message: *Resource De-allocate*
05h — System message: *Remote GetSource*

bXTAH Target Address high. bXTAH contains the high byte of the target address. This value determines the addressing mode used (see Section 13.1).

bXTAL Target Address low. bXTAL contains the low byte of the target address. This value determines the addressing mode used (see Section 13.1).

bXCDn Transmit Control data byte n. The bytes bXCD0 to bXCD16 contain the data of the message to send. When using Broadcast or Groupcast addressing, only 16 data bytes are available for messages (bXCD16 is not available).

13.3 Control Message Reception

The maximum number of received data bytes per message is 17 (16 when using broadcast or groupcast addressing). Either polling or interrupts can be used to determine when a Control message has been received.

The left side of Figure 13-3 illustrates the flow for an interrupt service routine for receiving Control messages. When using interrupts, the **bIE.IMRX** bit enables received message interrupts. The right side of Figure 13-3 illustrates the use of polling to determine receive message status. When using the polling method, **bIE.IMRX** must be cleared to disable interrupts.

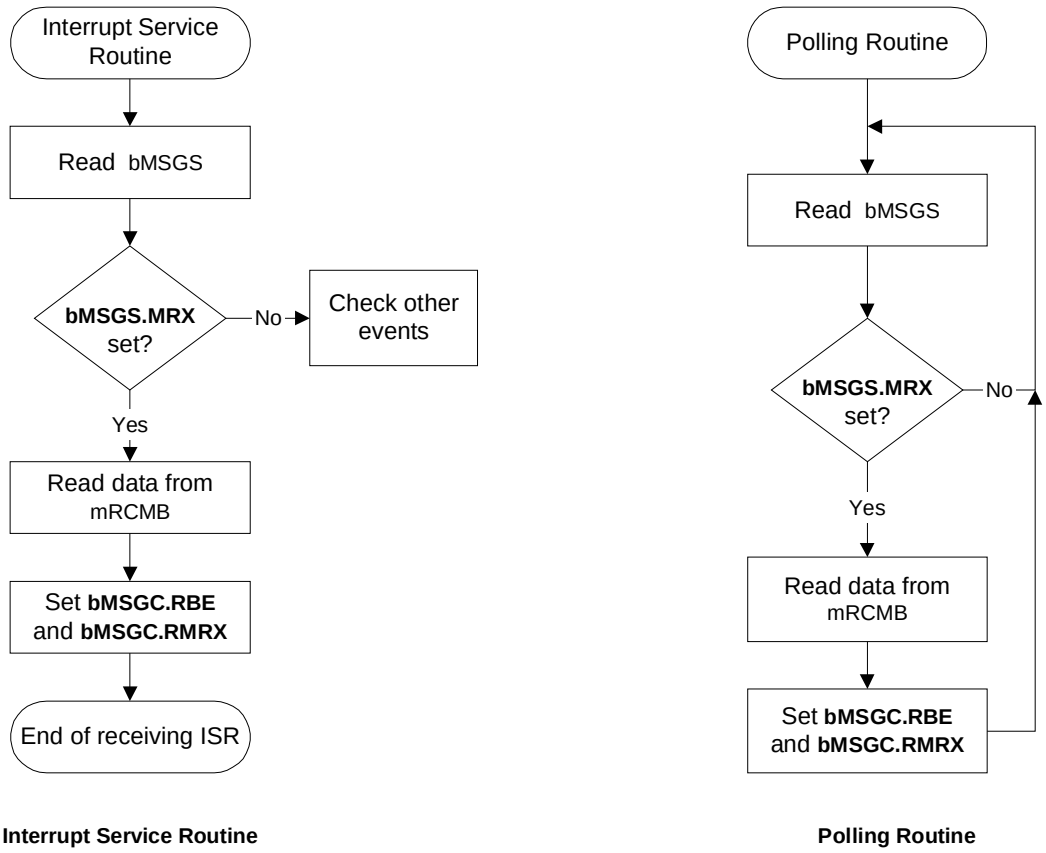


Figure 13-3: Control Message Reception Flow

If the **bMSGC.RBE** bit is not set after reading the message, further Control messages will be blocked from being received.

13.4 Control Message Transmission

Similar to Control message reception, Control message transmission can be done through polling or the use of interrupts. To use the interrupt method, the **bIE.IMTX** bit must be set. Figure 13-4 shows the program flow for transmitting a Control message.

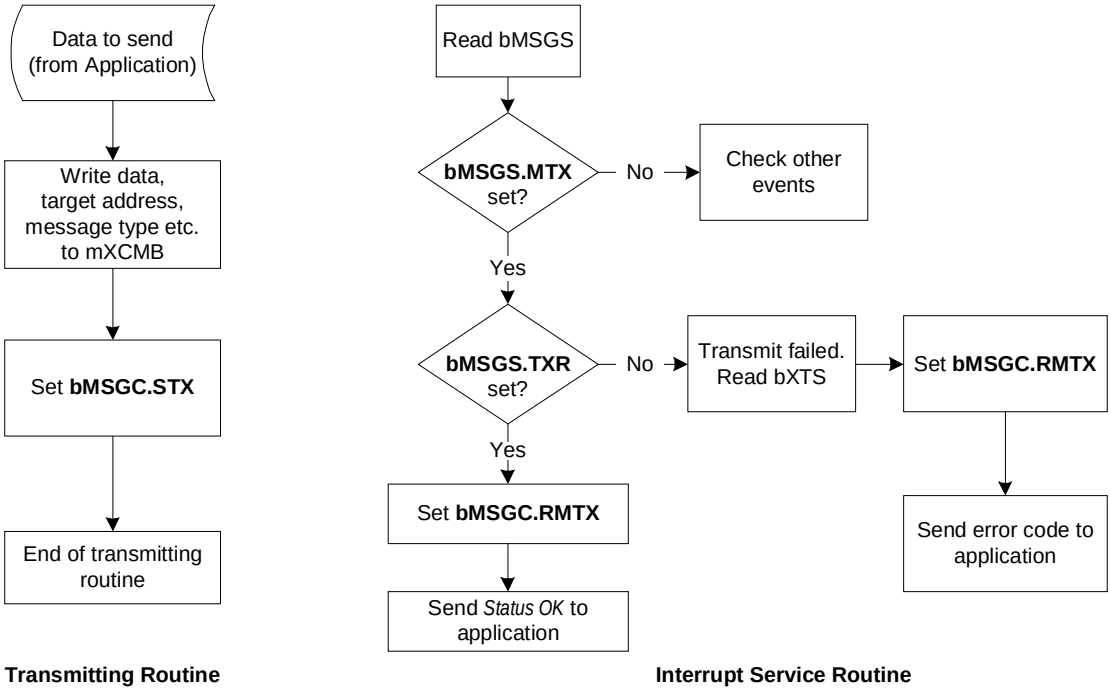


Figure 13-4: Sending MOST Control Messages: Interrupt Service Routine

To use the polling method, the **bIE.IMTX** bit should be cleared to disable interrupts, and the Message Status Register (bMSGS) must be polled in regular intervals. Figure 13-5 illustrates the program flow for sending Control messages based on polling.

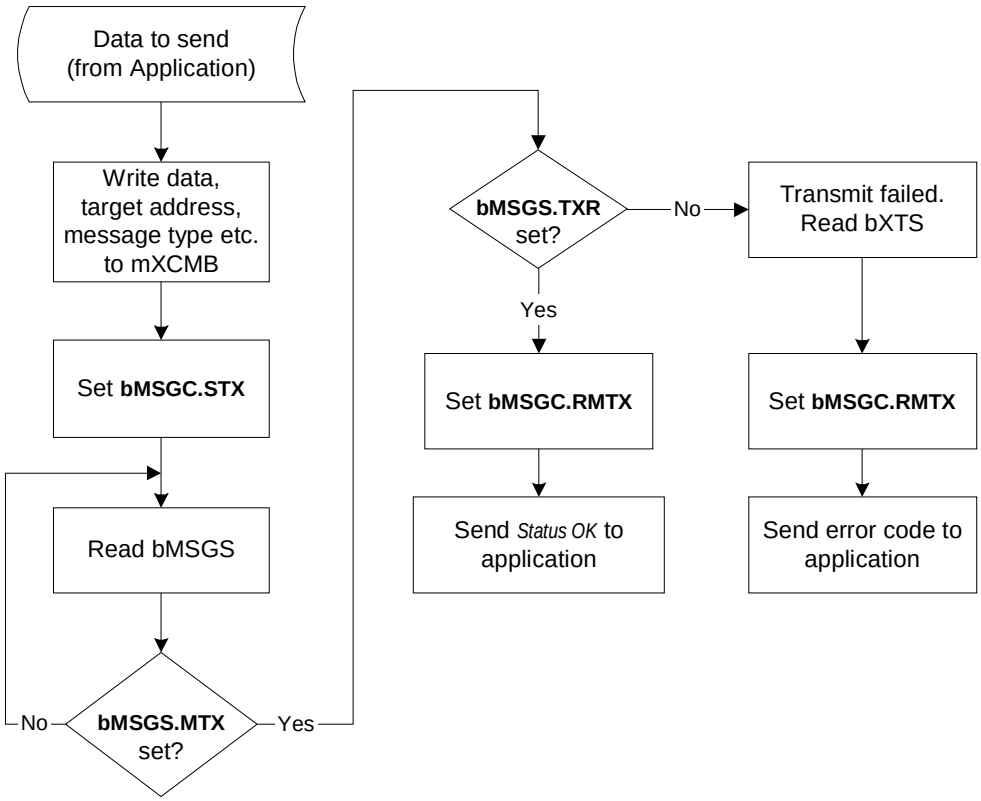


Figure 13-5: Sending MOST Control Messages: Polling

13.5 Control Message Types

13.5.1 Received Control Message Types

When receiving Control messages, the bRTYP register in mRCMB specifies the addressing mode used for the received message.

bRTYP	Description
00h	Message addressed to logical address (bNAH/bNAL register values)
01h	Message addressed to physical address/node position (bNPR register value)
02h	Message broadcasted (Address = 03C8h)
03h	Message groupcasted (bGA register value)

Table 13-12: bRTYP (Received Control Message Types)

13.5.2 Transmit Control Message Types

The following types of messages are available when transmitting Control messages:

- Normal messages:
  - Single cast (Logical or Physical Addressing)
  - Groupcast
  - Broadcast

- System messages:
  - Resource Allocate
  - Resource De-allocate
  - Remote Read
  - Remote Write (can be blocked by nodes with **bNC.RWD** set):
    Single cast (logical or physical addressing)
    Groupcast
    Broadcast
  - Remote GetSource

The transmit message type must be written to bXTYP in mXCMB.

bXTYP	Description
Standard message:	
00h	Message sent as normal message.
System messages:	
01h	Message sent as *Remote Read* message.
02h	Message sent as *Remote Write* message.
03h	Message sent as *Resource Allocate* message.
04h	Message sent as *Resource De-allocate* message.
05h	Message sent as *Remote GetSource* message.

Table 13-13: bXTYP (Transmit Control Message Type)

13.5.2.1 Normal Messages (Type 00h)

By using this message type, data of any kind can be sent to any node within the network. Table 13-14 shows the contents of mXCMB for sending a normal message:

Address	Contents	Description
C0h	01h	Priority; Default 01h
C1h	00h	Normal message
C2h	XTAH	Target address high
C3h	XTAL	Target address low
C4h..D4h	D0..D16	17 data bytes

Table 13-14: mXCMB for Sending a Normal Control message

Since the type of addressing is encoded in the actual address, the target address can be set for Logical or Physical addressing to a single node, Groupcast addressing to multiple nodes, or Broadcast addressing to all nodes.

13.5.2.2 Remote Read Message (Type 01h)

This message type provides reading of registers on any other node, without influencing the target node's transmit and receive buffer or the respective bits. Therefore, the *Remote Read* is hidden from the application running on the target node. For the sending node, the transmission of a *Remote Read* message appears as a normal transmission; however, the result (the received data) will be stored in the transmit buffer (mXCMB). *Remote Read* accesses are limited to the target node's memory page 0. Table 13-15 shows the contents of mXCMB after having sent a *Remote Read* message.

Address	Contents	Description
C0h	01h	Priority; Default 01h
C1h	01h	*Remote Read*
C2h	XTAH	Target address high
C3h	XTAL	Target address low
C4h	rsvd	Reserved; Write as 00h
C5h	MAP	Address to read from
C6h	rsvd	Reserved
C7h..CEh	D0..D7	8 data bytes read from target node (after the target node has answered the *Remote Read* request). D0 contains data from MAP, D1 contains MAP+1, etc.
CFh..D4h	rsvd	Reserved; Write as 00h

Table 13-15: mXCMB after a Remote Read message

The result of the *Remote Read* message transmission is indicated by the **bMSGS.MTX** and **bMSGS.TXR** bits. If a successful transmission is indicated, the data read from the remote node is available in mXCMB bytes C7h..CEh (D0..D7).

13.5.2.3 Remote Write Message (Type 02h)

This message type allows modification of the remote node's memory locations on memory page 0. Like the *Remote Read* message, the *Remote Write* access is a special mode of operation, since the target node's transmit and receive buffers are NOT influenced in any way. Therefore, a *Remote Write* access is hidden from the application running on the target node. If receiving nodes have their **bNC.RWD** bit set, then *Remote Write* operations are disabled, and the node will respond with a status of 11h in bXTS (OS8104 revisions prior to D do not support disabling of *Remote Write* Control messages). For the sending node, the transmission of a *Remote Write* message is a normal Control message transmission. One to eight bytes in the remote node can be written starting at the memory address pointer (MAP). Table 13-16 shows the contents of mXCMB when sending a *Remote Write* message:

Address	Contents	Description
C0h	01h	Priority; Default 01h
C1h	02h	*Remote Write*
C2h	XTAH	Target address high
C3h	XTAL	Target address low
C4h	rsvd	Reserved; Write as 00h
C5h	MAP	Address to write to
C6h	LENGTH	Number of data bytes to be written (max. is 8)
C7h..CEh	D0..D7	As many as 8 data bytes can be written to the target node. The contents of C7h will be written into address MAP, C8h into MAP+1, etc.
CFh..D4h	rsvd	Reserved

Table 13-16: mXCMB when Sending a Remote Write message

The result of a *Remote Write* message transmission is indicated by the **bMSGS.MTX** and **bMSGS.TXR** bits. If a successful transmission is indicated, the data was written successfully to the remote node.

84h bNC Network Control Register

Bit	Label	Description	Default
7..1	rsvd	Reserved; Write to 0	0000000
0	RWD	*Remote Write* disable	0

Table 13-17: bNC (Network Control Register)

RWD *Remote Write* Disable. Revisions B and C of the OS8104 do not support *Remote Write* disable; however, writes to this bit location on revision B or C have no adverse affects (don't cares).

0 – *Remote Write* Control messages are supported. Other nodes are allowed to write to page 0 registers of the chip.

1 – *Remote Write* Control messages are blocked. Other nodes cannot write to any of the on-chip registers; however, *Remote Reads* are allowed. Nodes attempting to send *Remote Write* messages will get the bXTS response of 11h (transmit message type not supported).

13.5.2.4 Resource Allocate Message (Type 03h)

Before routing any synchronous data to the MOST Network, the synchronous channels (physical channels) should be allocated. Allocation of synchronous data channels, managed by the timing-master node, guarantees that other nodes wanting to route synchronous data to the Network do not use the same physical channels (causing collisions). When a node wants to place synchronous data onto the Network, it should first send a *Resource Allocate* message to the timing-master node.

For the node requesting synchronous resources, the transmission of a *Resource Allocate* message appears as a normal transmission; however, the response from the timing-master node is stored in the transmit buffer. When an OS8104 is configured as the timing-master, it internally manages the incoming *Resource Allocate* messages and responds without added software support.

Resource Allocate messages can either be sent using logical addressing or physical addressing. Using physical (node position) addressing is the most effective approach, since the timing-master in a MOST Network always has node position 00h. Therefore, the physical address is always 0400h, where bXTAH is set to 04h, and bXTAL to 00h.

One to eight physical channels can be allocated per *Resource Allocate* request. When the timing-master responds, the **bMSGS.MTX** and **bMSGS.TXR** bits are set, and Answer1 and Answer2 in mXCMB indicate the response. If the Answer1 result is 01h, then the allocation request was granted and C9h..D0h of mXCMB contain the physical channel numbers to use. For the node requesting the channels, these physical channel numbers are the MRT registers into which the address reference (MRA) of the source data should be loaded. This will connect the local node's source data to the MOST Network. The first channel that was granted (located in P0 at C9h, is also referred to as a *Connection Label* (CL) which is stored in the allocation table (mCRA) and represents the entire set of channels allocated from this request. This CL is used to de-allocate this set of channels. Resource allocation and the mCRA are discussed in more detail in Chapter 14 on page 131. Table 13-18 shows the contents of mXCMB when sending a *Resource Allocate* message.

Address	Contents	Transmit Control Message Buffer mXCMB
C0h	01h	Priority; Default 01h
C1h	03h	*Resource Allocate*
C2h	04h	Target address high (for physical addressing = 04h)
C3h	00h	Target address low (for physical addressing = 00h)
C4h	rsvd	Reserved; Write as 00h
C5h	Request	Number of channels to be allocated (maximum value is 8)
C6h	rsvd	Reserved; Write as 00h
C7h	Answer1	Allocation status from timing-master (see Table 13-19)
C8h	Answer2	Allocation status from timing-master (see Table 13-19)
C9h..D0h	P0..P7	†MRT locations. The Connection Label (CL) in byte P0 is used when de-allocating this logical channel. A byte value of FFh indicates *not usable*.
D1h..D4h	rsvd	Reserved

† The value returned by the timing-master refers to the physical channel address within the MOST frame. These address are the MRT locations used to transmit data across the Network. The source data address references (MRA) should be placed in these MRT locations.

Table 13-18: mXCMB when Sending a Resource Allocate message

Answer1	Comment	Answer2	Comment
01h	ALLOC_GRANT; Enough free channels to fill request.	AL_FREE	Total number of free locations at that time (after current request filled)
02h	ALLOC_BUSY; Timing-master is busy working on allocation requests. Try again later.	AL_FREE	Total number of free locations at that time
03h	ALLOC_DENY; Request was denied. Not enough free channels to fill request.	AL_FREE	Total number of free locations at that time
04h	ALLOC_WRONG; The *Request* value was set to 0 or to a value greater than 8.	AL_FREE	Total number of free locations at that time
05h	WRONG_TARGET; Request was directed to a timing-slave node instead of the timing-master.	00h	

Table 13-19: Resource Allocate Message Responses

The result of the current *Resource Allocate* request is returned by the timing-master node and written into the bytes Answer1 (C7h) and Answer2 (C8h) of mXCMB. The flow chart shown in Figure 13-6 illustrates how to send allocate requests and how to respond to the answers.

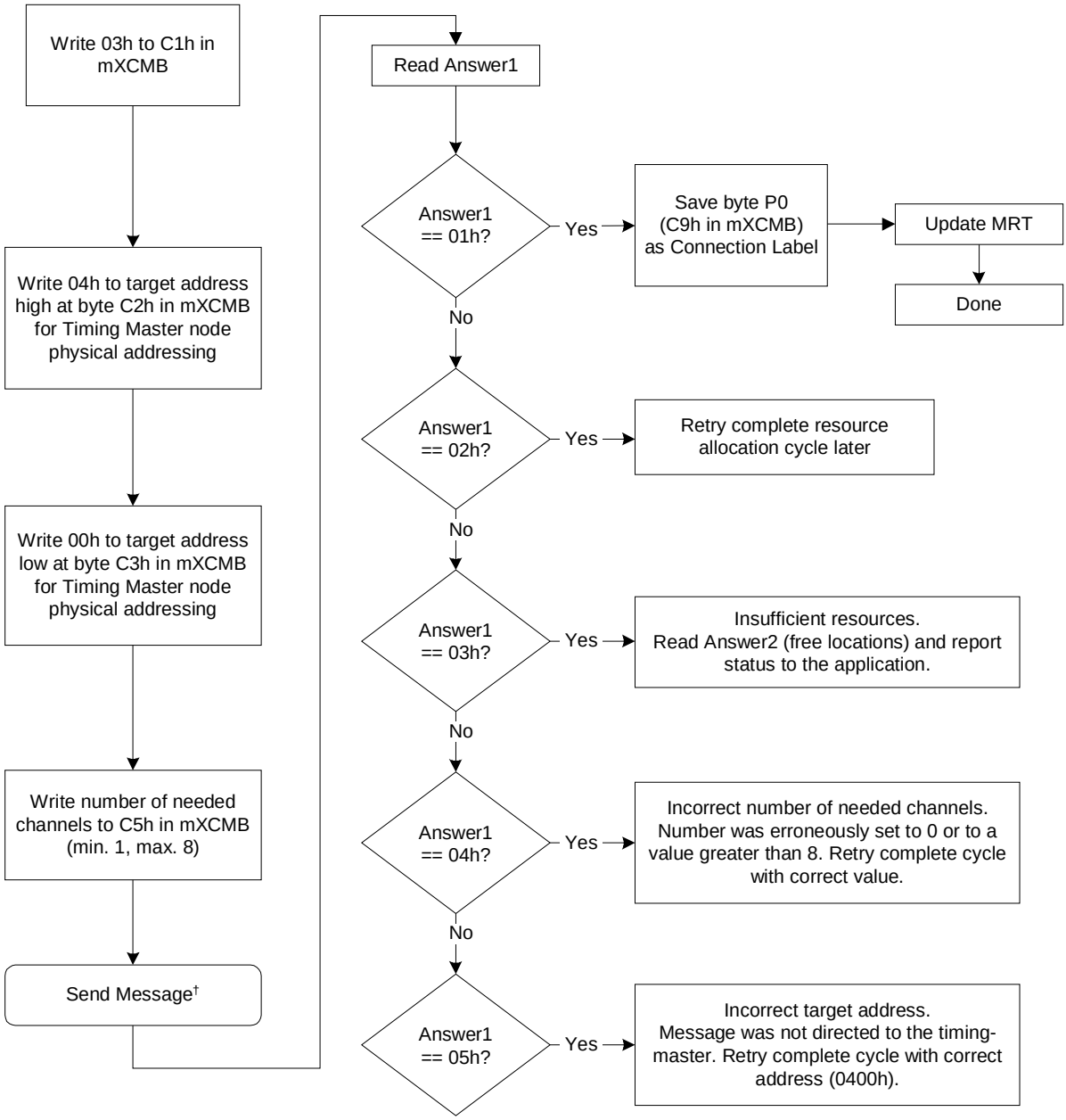


Figure 13-6: Resource Allocation Flow

† The sending of a message is described in Section 13.4.

13.5.2.5 Resource De-Allocate Message (Type 04h)

If Network synchronous data is no longer needed by the node, the resources should be de-allocated, so they are available for other applications. De-allocating allocated resources is a simple task using the *Resource De-allocate* message.

The Connection Label, provided by the timing-master node when the resources were allocated, is sent in the de-allocate message to indicate to the timing-master which logical channel to de-allocate.

The Connection Label (CL) is the physical channel ID of the first channel and is stored in the Channel Allocation table (mCRA) for all channels assigned to this ID. The mCRA is described in more detail in Chapter 14 on page 131.

One special CL (7Fh) forces the timing-master to de-allocate all resources that have previously been allocated and is labeled *De-allocate All*. After power-up or reset, the application of the timing-master node must send a *De-allocate All* message to physical address 0400h (himself), or to its own logical address to initialize the mCRA table. In addition, after every change of bSBC, the allocation mechanism must be initialized by sending the *De-allocate All* message to the timing-master node.

Table 13-20 shows the contents of mXCMB for sending a *Resource De-allocate* message:

Address	Contents	Transmit Control Message Buffer mXCMB
C0h	01h	Priority; Default 01h
C1h	04h	*Resource De-allocate*
C2h	04h	Target address (for physical addressing = 04h)
C3h	00h	Target address (for physical addressing = 00h)
C4h	rsvd	Reserved; Write as 00h
C5h	CL	Connection Label assigned during allocation process†
C6h	rsvd	Reserved; Write as 00h
C7h	Answer1	De-Allocation status from timing-master (see Table 13-21)
C8h	00h	Constant 00h
C9h..D4h	rsvd	Reserved

† See the *Resource Allocate* message

Table 13-20: mXCMB when Sending a Resource De-Allocate message

Answer1	Comment
01h	DEALLOC_GRANT; De-allocation successful.
02h	DEALLOC_BUSY; Timing-master is busy working on allocation/de-allocation requests.
04h	DEALLOC_WRONG; The Connection Label had a value greater than 7Fh.
05h	WRONG_TARGET; Request was directed to a timing-slave node instead of the timing-master.

Table 13-21: Resource De-Allocate Message Responses

13.5.2.6 Remote GetSource Message (Type 05h)

This message finds the addresses (logical, physical, and group) of the node that is placing source data onto the Network, using a particular physical channel. Using the assumption that a node is using all physical channels as a group (the logical channel), the Connection Label (CL) for the logical channel can be used to check the entire logical set. If nodes do not use the logical channel as a set, then the individual physical channels must be checked. Similar to other system messages, this message does not interfere with the target node's transmit or receive buffers, and is handled in the background, without interfering with the normal Control message traffic.

The *Remote GetSource* message is sent as a broadcast message (address 03C8h) to all nodes. This system message is unique since it does not block the Control message channel even though it uses broadcast addressing. The Connection Label associated with a particular source channel is placed in the mXCMB buffer at location C5h. The node associated with that Connection Label will respond with its addresses.

If the **bMSGS.MTX** and **bMSGS.TXR** bits are set, then the addresses sought are stored back in the mXCMB. Transmit buffer byte CAh contains the node position (part of the physical address), byte CCh contains the group address, and bytes CDh and CEh contain the node's logical address.

If **bMSGS.TXR** is cleared, no node responded; therefore, no current node in the Network is associated with that Connection Label.

Address	Contents	Description
C0h	01h	Priority; Default 01h
C1h	05h	*Remote GetSource* message code
C2h	03h	Broadcast address high. Set to 03h
C3h	C8h	Broadcast address low. Set to C8h
C4h	rsvd	Reserved; Write as 00h
C5h	CL	Connection Label (or particular physical channel to check)
C6h..C9h	rsvd	Reserved; Write as 00h
CAh	NPR	Returned Node Position of the node that routes stream data to the channels specified by CL
CBh	rsvd	Reserved; Write as 00h
CCh	GA	Returned Group Address of the node that routes stream data to the channels specified by CL
CDh	NAH	Returned high byte of logical address of the node that routes stream data to the channels specified by CL
CEh	NAL	Returned low byte of logical address of the node that routes stream data to the channels specified by CL
CFh..D4h	rsvd	Reserved; Write as 00h

Table 13-22: mXCMB when Sending a Remote GetSource message

14 Resource Administration

The MOST Network supports a maximum of 60 bytes per frame for transporting source data. These bytes are divided between synchronous data and asynchronous packet data. The bSBC register (see Section 6.2.5 on page 40) controls the division between the two, and contains the number of quadlets (four bytes) reserved for synchronous data. Once the bSBC register is initialized, the timing-master knows how many synchronous channels (physical channels) are available, and where to start asynchronous channel arbitration.

To manage the synchronous data on a Network-wide basis, the timing-master node maintains a Channel Resource Allocation table (mCRA) to keep track of the status of each physical channel. This table is distributed to all nodes on the Network at regular intervals (every 1024 frames). Having one node (the timing-master) manage the allocation of synchronous data, keeps other Network nodes from placing data on the same physical channels and causing collisions.

14.1 mCRA (Channel Resource Allocation Table)

The Channel Resource Allocation table (mCRA) contains the current allocation status of every synchronous data byte in the MOST frame. In each node, mCRA is located at memory locations 0380h to 03BBh. Since the number of synchronous data bytes per frame is variable, the length of mCRA is also variable. The maximum memory address of the mCRA is calculated as follows:

last mCRA address = 0380h + (**bSBC.SAC[3:0]** $\times$ 4) – 1

The following example shows the structure of mCRA when 14 quadlets of the available 15 quadlets are assigned to synchronous data. The variables CRAnn in Table 14-1 refer to the physical byte position of the synchronous data in the MOST frame (CRA0F stands for the physical channel number 0Fh in the MOST frame). The last memory address of the mCRA in this example is:

0380h + (14 $\times$ 4) – 1 = 03B7h

Therefore, the contents of addresses 03B8h to 03BBh (shown in the shaded area of Table 14-1) must be ignored since they do not represent valid synchronous bytes.

mCRA	+0	+1	+2	+3	+4	+5	+6	+7
380h	CRA00	CRA01	CRA02	CRA03	CRA04	CRA05	CRA06	CRA07
388h	CRA08	CRA09	CRA0A	CRA0B	CRA0C	CRA0D	CRA0E	CRA0F
390h	CRA10	CRA11	CRA12	CRA13	CRA14	CRA15	CRA16	CRA17
398h	CRA18	CRA19	CRA1A	CRA1B	CRA1C	CRA1D	CRA1E	CRA1F
3A0h	CRA20	CRA21	CRA22	CRA23	CRA24	CRA25	CRA26	CRA27
3A8h	CRA28	CRA29	CRA2A	CRA2B	CRA2C	CRA2D	CRA2E	CRA2F
3B0h	CRA30	CRA31	CRA32	CRA33	CRA34	CRA35	CRA36	CRA37
3B8h	**CRA38**	**CRA39**	**CRA3A**	**CRA3B**				

Table 14-1: mCRA Built for 56 bytes of Synchronous Data

After reset, all values in the mCRA table are set to 70h, indicating that the byte is not allocated.

14.1.1 mCRA in Timing-Slave Nodes

The allocation status of a byte is encoded in the value stored to its associated location within the mCRA. The following values are valid in the mCRA of a timing-slave node:

Value*	Description
70h	Corresponding byte in the frame is not allocated.
00h..3Bh	Corresponding byte in the frame is allocated. The value specifies the Connection Label.

* Assumes the MSB is masked off before interpreting value.

Table 14-2: Valid mCRA Values in Timing-Slave Nodes

The most significant bit of the values in mCRA must be ignored (masked off) in all timing-slave nodes (all nodes except the timing-master). The Connection Label is the first physical channel number returned from an allocation request and is stored at every physical channel location in the mCRA associated with that request.

14.1.2 mCRA in the Timing-Master Node

In a timing-master node, the most significant bit of the values in mCRA indicates whether a byte is actually in use, and the lower seven bits indicate whether a byte is allocated.

Value	MSB	Bit[6:0]	Description
70h	0	70h	Corresponding byte in the frame is not allocated and not in use.
F0h	1	70h	Corresponding byte in frame is not allocated, but is being used by a node within the Network. This is an error, since the byte was never allocated before being used by a node.
80h	1	00h	Corresponding byte in the frame is allocated to Connection Label 00h and is in use.
00h	0	00h	Corresponding byte in the frame is allocated to Connection Label 00h, but it is not in use. The node which allocated this resource is not using the channel at this time.

Table 14-3: Valid mCRA Values in the Timing-Master Node

14.2 Allocating Network Resources

To reduce the complexity of application software, the timing-master node manages the synchronous data resources automatically. When the timing-master is initializing and setting the **bSBC.SAC[3:0]** bits to define the SAC (Synchronous Area Count) value, a *De-allocate All* Control message must be sent to initialize the mCRA.

Each application that needs synchronous channels should send a *Resource Allocate* Control message to the timing-master. If there are enough free channels to fill the request, the timing-master returns the physical channel numbers that were allocated for this request. If not enough physical channels are free to fill the request, the timing-master sends an error message to the requesting node. Each physical channel number corresponds to an 8-bit channel in the frame that is reserved for the requesting-node's application. These physical channel numbers are also the MRT locations that the application fills with the address references of the data to be routed onto the Network. When the application places address references into the lower half of the MRT, the OS8104 moves the source data associated with that address reference onto the MOST Network in the position (physical channel number) indicated by the MRT location. Chapter 12 on page 97 explains the routing of synchronous data in detail.

The first physical channel number returned by the timing-master is also defined as the Connection Label (CL), and is an ID for all the channels (logical channel) allocated on this particular request. Therefore, the mCRA only stores CLs for allocated channels (each physical channel location in the mCRA contains the Connection Label associated with that allocation request).

For example, if an application requests eight channels, the timing-master could grant the request and might return the physical channel numbers 00h through 07h. The CL for this request is 00h, and must be stored by the application. The CL 00h would be stored in the mCRA at the first eight locations (those locations associated with physical channel numbers 00h through 07h), and used to de-allocate these channel when no longer needed.

When the application is finished using these channels, it de-allocates the channels by sending the CL (00h) to the timing-master, and removes the address references from the corresponding MRT locations. The MRT should then be filled with the corresponding received Network data to allow those synchronous channels to pass through the node unaltered.

After numerous allocate/de-allocate operations, the mCRA may be fragmented since the physical channels associated with a CL need not be contiguous. Table 14-4 illustrates an example of an mCRA table, in a timing-slave node, where 24 bytes are reserved for synchronous data (6 quadlets). The values in the table are shown after the MSB is masked off.

Byte	+0	+1	+2	+3	+4	+5	+6	+7
00h	00h	00h	00h	00h	00h	00h	00h	00h
08h	08h	08h	08h	0Bh	0Bh	0Dh	0Bh	0Dh
10h	0Dh	0Dh	0Dh	0Dh	0Dh	0Dh	70h	70h

Table 14-4: mCRA Example

The physical channels of the MOST frame are assigned as follows:

Connection Label (CL)	Physical Source Channels in MOST frame
00h	00h through 07h (8 physical channels)
08h	08h through 0Ah (3 physical channels)
0Bh	0Bh, 0Ch, and 0Eh (3 physical channels)
0Dh	0Dh, 0Fh, and 10h through 15h (8 physical channels)
70h	16h and 17h are unallocated channels

Table 14-5: Connection Label Example

The maximum number of channels that can be allocated per request is eight. The number of requests per node is not limited. Therefore, a node could allocate eight bytes at once, or could send eight requests for a single channel (although the application would now have to track eight Connection Labels). Since the maximum number of channels that can be allocated per node is only limited by the number of synchronous data bytes (bSBC), a single node could allocate all 60 bytes.

14.3 De-Allocating Network Resources

When a node no longer needs allocated synchronous channels, the channels should be de-allocated to allow other nodes to allocate them. This is done by sending a *Resource De-allocate* Control message to the timing-master of the Network. The *Resource De-allocate* message is described in Section 13.5.2.5 on page 129.

For de-allocating resources, the Connection Label, given when the resources were allocated, must be sent to the timing-master in the de-allocation request. Using the example from the previous section, Table 14-6 shows the mCRA before the de-allocation request.

mCRA	+0	+1	+2	+3	+4	+5	+6	+7
380h	00h	00h	00h	00h	00h	00h	00h	00h
388h	08h	08h	08h	0Bh	0Bh	0Dh	0Bh	0Dh
390h	0Dh	0Dh	0Dh	0Dh	0Dh	0Dh	70h	70h

Table 14-6: mCRA Example before Resource De-allocation

The node sends a de-allocation request to the timing-master using the Connection Label 0Bh.

Table 14-7 illustrates the mCRA after the resources have been de-allocated. Their status (not allocated) is indicated by the value 70h, as shown in the shaded areas in Table 14-7.

mCRA	+0	+1	+2	+3	+4	+5	+6	+7
380h	00h	00h	00h	00h	00h	00h	00h	00h
388h	08h	08h	08h	**70h**	**70h**	0Dh	**70h**	0Dh
390h	0Dh	0Dh	0Dh	0Dh	0Dh	0Dh	70h	70h

Table 14-7: mCRA Example after Resource De-allocation

15 Packet Data Transfer

The *Packet Data Transfer* service uses the portion of the MOST Network reserved for the asynchronous channel, and is useful for applications that can transfer data in bursts (e.g,. Internet data, GPS map data, email), instead of in a continuous data stream (e.g., video or audio). In all Source Port modes, except Parallel-Combined, the maximum packet size supports 48 data bytes. When the Source Ports are configured for Parallel-Combined mode, the maximum packet data size is 1014. The packet is protected by a trailing CRC (Cyclic Redundancy Check), which is automatically generated/checked by the OS8104.

15.1 Packet Transfer Registers

The following registers are associated with asynchronous packets:

- bAPAH — Alternate Packet Address High register
- bAPAL — Alternate Packet Address Low register
- bPLDT — Packet Transmit Length register
- bPPI — Packet Priority register
- bPCTC — Packet Control register
- bPSTX — Packet Start Tx register
- bPCTS — Packet Status register
- mARP — Asynchronous Receive Packet buffer
- mAXP — Asynchronous Transmit Packet buffer

In addition, the logical node address in the bNAH and bNAL registers are sent as the source address in packets.

When sending a packet, the target address can either be the target node's logical address (bNAH/bNAL registers) or the target node's alternate packet address (bAPAH/bAPAL registers). Sending packets to the logical address uses the same addressing as used for the Control messages. If separate addresses for Control and Packet messages are required, the alternate packet address registers (bAPAH/bAPAL) can be used.

15.1.1 bAPAH (Alternate Packet Address High Register)

E8h bAPAH Alternate Packet Address High Register

Bit	Name	Description	Default
7..0	APA[15:8]	Alternate Packet Address High. This value cannot be the same as bNAH.	0Fh

Table 15-1: bAPAH (Alternate Packet Address High Register)

The bAPAH register stores bits 15 through 8 of the alternate address for packets received. The default alternate packet address after reset is 0F0Fh. The bAPAH register cannot be the same value as bNAH. When a node receives a packet, it checks the target address high byte against its bNAH value. If they do not match, the alternate packet address registers are checked. bAPAH and bAPAL can be used as an asynchronous packet groupcast address to allow multiple nodes to receive the same message. However, using this register for groupcast addressing does not have the same protection and acknowledges that are associated with the Control message groupcast addressing.

15.1.2 bAPAL (Alternate Packet Address Low Register)

E9h bAPAL Alternate Packet Address Low Register

Bit	Name	Description	Default
7..0	APA[7:0]	Alternate Packet Address Low. Lower 8 bits of Alternate Packet Address.	0Fh

Table 15-2: bAPAL (Alternate Packet Address Low Register)

15.1.3 bPLDT (Packet Transmit Length Register)

ECh bPLDT Packet Transmit Length Register

Bit	Name	Description	Default
7..0	PLDT[7:0]	Packet length for data transfer in quadlets	00h

Table 15-3: bPLDT (Packet Length Register)

The value in this register specifies the length of the data (in quadlets) sent when transmitting a packet, and includes the data bytes along with the two source address bytes. Therefore, the length value is calculated as follows:

$$bPDLT = \text{roundup}\left(\frac{\text{number of data bytes} + 2}{4}\right)$$

Valid values for bPLDT are 01h to 0Dh. If the length value is rounded up (fractional), then the extra filler bytes at the end of the packet should be set to 00h. For example, sending three data bytes takes two quadlets, because two source address bytes are used. Three filler bytes exist at the end of the packet, which should be filled with zeros. bPLDT does not include the CRC bytes, which is automatically generated by the OS8104. Table 15-4 shows the bPLDT value for various data byte lengths.

Data Bytes	Quadlets (bPLDT value)
1	01h
2	01h
3	02h
4	02h
5	02h
6	02h
7	03h
...	...
48	0Dh

Table 15-4: bPLDT Length Values

15.1.4 bPPI (Packet Priority Register)

F2h bPPI Packet Priority Register

Bit	Name	Description	Default
7..0	PPI[7:0]	Packet Priority. Valid values are 01h (highest) to 07h (lowest).	01h

Table 15-5: bPPI (Packet Priority Register)

The value in bPPI determines the priority of a transmitted packet, used in arbitration of the asynchronous portion of the MOST Network source data. Valid values are 01h to 07h, where 01h represents the highest priority level.

15.1.5 bPCTC (Packet Control Register)

E2h bPCTC Packet Control Register

Bit	Name	Description	Default
7..5	rsvd	Reserved; Write as 0	000
4	RAF	Reset *Packet rejected status (mARP full)* bit **bPCTS.AF**	0
3	RAC	Reset *Packet rejected status (CRC failed)* bit **bPCTS.AC**	0
2	rsvd	Reserved; Write as 0	0
1	RATX	Clear *Packet Transmitted* interrupt	0
0	RARX	Clear *Packet Received* interrupt. (Unlocks the Receive Packet Buffer mARP)	0

Table 15-6: bPCTC (Packet Control Register)

RAF Reset *Packet Rejected status (mARP full)* bit **bPCTS.AF**. When the **RAF** bit is set, the **bPCTS.AF** bit is cleared. **RAF** must not be set unless **bPCTS.AF** is set. The **RAF** bit is cleared automatically.

RAC Reset *Packet Rejected status (CRC failed)* bit **bPCTS.AC**. When the **RAC** bit is set, the **bPCTS.AC** bit is cleared. **RAC** must not be set unless **bPCTS.AC** is set. The **RAC** bit is cleared automatically.

RATX Clear *Packet Transmitted* interrupt. When the **RATX** bit is set, the **bPCTS.ATX** bit is cleared. Additionally, the interrupt is set to inactive, and the $\overline{\text{AINT}}$ pin goes high. **RATX** must not be set unless **bPCTS.ATX** is set. The **RATX** bit is cleared automatically.

RARX Clear *Packet Received* interrupt and unlock the Asynchronous Receive Packet Buffer (mARP). Setting the **RARX** bit clears the **bPCTS.ARX** bit and unlocks the Asynchronous Receive Packet Buffer (mARP). When the **bPCTS.ARX** bit is set, mARP contains a packet and is locked until the **RARX** bit unlocks it. While locked, mARP will not receive any other packets. **RARX** must not be set unless **bPCTS.ARX** is set. The **RARX** bit is cleared automatically.

Buffer mARP should be read before being unlocked; otherwise, the data could be overwritten by a new packet.

15.1.6 bPSTX (Packet Start Tx Register)

EAh bPSTX Packet Start Tx Register

Bit	Name	Description	Default
7	ASTX	Start packet transmission	0
6..0	rsvd	Reserved; Write as 0	0000000

Table 15-7: bPSTX (Packet Start Tx Register)

ASTX Start packet transmission. When the **ASTX** bit is set, the OS8104 starts arbitrating for the asynchronous channel to transmit the current packet in mARP. The **ASTX** bit is cleared automatically when the transmission is finished.

15.1.7 bPCTS (Packet Status Register)

E3h bPCTS Packet Status Register

Bit	Name	Description	Default
7..5	rsvd	Reserved	000
4	AF	Packet rejected — mARP full	0
3	AC	Packet rejected — CRC failed	0
2	rsvd	Reserved	0
1	ATX	Packet transmitted	0
0	ARX	Packet received	0

Table 15-8: bPCTS (Packet Status Register)

AF Packet rejected — mARP full. When the **AF** bit is set, it indicates that the last reception failed because the packet receive buffer mARP is full (locked). mARP is unlocked by setting the **RARX** bit after having read its contents. The **AF** bit is cleared by setting the **bPCTC.RAF** bit.

AC Packet rejected — CRC failed. When the **AC** bit is set, it indicates that the last reception failed due to a bad CRC value. The **AC** bit is cleared by setting the **bPCTC.RAC** bit.

ATX Packet transmitted event. When the **ATX** bit is set, it indicates that a packet transmission is completed. **ATX** will be set after the last byte of a packet is processed. The $\overline{\text{AINT}}$ pin will go low shortly after the **ATX** bit is set. The **ATX** bit is cleared by setting the **bPCTC.RATX** bit.

ARX Packet received event. When the **ARX** bit is set, it indicates that a packet has been received and that mARP is locked. Shortly after the **ARX** bit is set, the $\overline{\text{AINT}}$ pin will go low. The **ARX** bit is cleared by setting the **bPCTC.RARX** bit.

As long as the **bPCTS.ARX** bit is set, no reception of further packets is possible. When the **bPCTS.ARX** bit is cleared (by setting **bPCTC.RARX**), mARP is unlocked. The contents of mARP can then be overwritten by the next packet.

15.1.8 mARP (Asynchronous Receive Packet Buffer)

The Asynchronous Receive Packet buffer contains the last packet which was successfully received.

180h mARP Asynchronous Receive Packet Buffer

Byte	Name	Description	Default
180h	bARTH	Received Target address high	00h
181h	bARTL	Received Target address low	00h
182h	bASAH	Source address high	00h
183h	bASAL	Source address low	00h
184h	bARD0	Asynchronous receive data byte 0	00h
185h ... 1B2h	bARD1 ... bARD46	Asynchronous receive data byte 1 to 46	00h
1B3h	bARD47	Asynchronous receive data byte 47	00h

Table 15-9: mARP (Asynchronous Receive Packet Buffer)

bARTH/bARTL

Received Target Address High/Received Target Address Low. This byte contains the logical address of the node to which the received packet was sent. The target address is either the logical node address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL).

bASAH/bASAL
: Source Address High/Source Address Low. These bytes contain the logical address of the node from which the packet was transmitted (origin of the packet data).

bARD0..47 Asynchronous Receive Data Bytes 0 to 47. These bytes contain the actual packet data that was received.

15.1.9 mAXP (Asynchronous Transmit Packet Buffer)

1C0h mAXP Asynchronous Transmit Packet Buffer

Byte	Name	Description	Default
1C0h	bATAH	Target address high	0Fh
1C1h	bATAL	Target address low	FFh
1C2h	bAXD0	Asynchronous Transmit data byte 0	00h
1C3h ... 1F0h	bAXD1 ... bAXD46	Asynchronous Transmit data byte 1 to 46	00h
1F1h	bAXD47	Asynchronous Transmit data byte 47	00h

Table 15-10: mAXP (Asynchronous Transmit Packet Buffer)

bATAH/bATAL
: Target Address High/Target Address Low. These bytes contain the target address of the node to which the packet will be sent to. This should be the logical node address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL) of the receiving/target node.

bAXD0..47 Asynchronous Transmit Data Bytes 0 to 47. These bytes contain the packet data to be sent.

If the number of bytes to be sent is not divisible by 4, adding filler bytes is recommended (generally 00h).

15.2 Asynchronous Interrupt Pin ($\overline{\text{AINT}}$)

The asynchronous interrupt pin ($\overline{\text{AINT}}$) indicates either the reception or transmission of packet data. For transmitted packets, $\overline{\text{AINT}}$ goes low when the packet is completely transferred, the transmit status is available, and the Packet Transmit Buffer (mAXP) is available. For received packets, $\overline{\text{AINT}}$ goes low when a valid packet is received, with the entire packet available in the Packet Receive Buffer (mARP). A valid received packet is one for which the logical address (bNAH/bNAL) or the alternate packet address (bAPAH/bAPAL) is correct, and the message has a valid CRC.

15.3 Packet Data Handling

The chip transmits packet data using the portion of the MOST frame reserved for asynchronous packet data transfer. The amount of source data reserved for asynchronous packet data is defined in the bSBC register. If there is not sufficient space for sending a packet within a single frame, data is segmented automatically and sent in smaller portions until the transmission is finished.

15.3.1 Preparing Packet Data for Transmission

When sending packet data, a maximum of 48 data bytes can be sent per packet. The packet size can be increased by running the Source Ports in Parallel-Combined mode (see Section 8.3 on page 74).

As preparatory work, the destination address and the priority level must be written to the respective registers. The length of data (in quadlets) must be calculated according to Section 15.1.3 and placed in the bPLDT register. When receiving a packet, the actual data length in the buffer is not provided. Therefore, the number of user data bytes per packet must be reported from the sending node to the receiving node. This is the task of the controlling application software and must be done before starting transmission. Alternately, the packet length can be included in the first few bytes when using a predefined packet protocol. This is the method used by the *MOST High Protocol* [8].

After length is calculated and written to the bPLDT register, user data can be written to the Asynchronous Transmit Packet Buffer (mAXP).

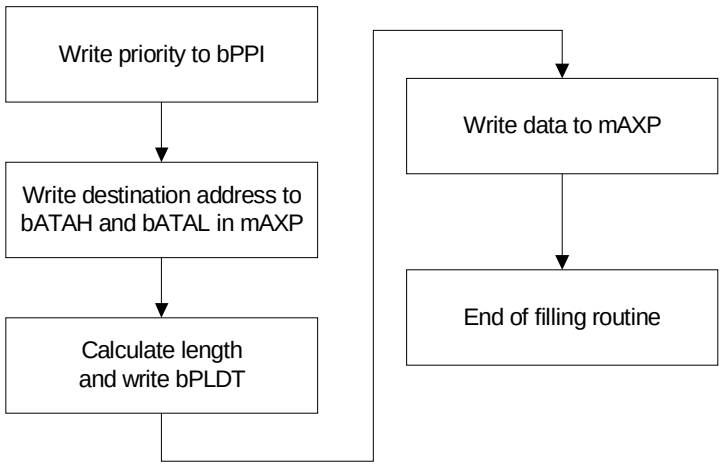


Figure 15-1: Preparing Packet Data for Transmission

The small routine shown in Figure 15-1 is used when describing the handling of packet data transfer, noted as "†" in the Figures to follow.

15.3.2 Polling-Based Packet Data Handling

The following routine must be called periodically by the main routine of the application. First, the Packet Control register (bPCTC) must be read and compared with 00h. If the content of this register is zero, no operation is pending, and the OS8104 is ready for further operations. The bPCTS is then checked for the completion of a packet transmission or packet reception. If either one or both of the **bPCTS.ATX** and **bPCTS.ARX** bits are set, then packet data handling is required. The error information (**bPCTS.AF** and **bPCTS.AC**) can be stored for later use in the main loop, if needed.

If the **bPCTS.ARX** bit is set, a packet has been received and is available in the Asynchronous Receive Packet Buffer (mARP). After having read mARP, the buffer must be unlocked to allow receiving of the next packet. This is done by setting the **bPCTC.RARX** bit. If the **bPCTS.AF** or **bPCTS.AC** error bits have been set, they should also be cleared to prepare them for the next message. This is done by setting **bPCTC.RAF** and **bPCTC.RAC**.

If the **bPCTS.ATX** bit is set, a packet transmitted event occurred and a new packet can be prepared for transmission if more packets need to be sent. If a new packet needs to be transmitted, the new packet data and all the header information should be loaded into the appropriate registers (see Figure 15-1). The **bPCTC.RATX** bit should be set to clear the *Packet Transmitted* status bit (**bPCTS.ATX**), and a new transmission is initiated by setting **bPSTX.ASTX**. Once the packet is sent, **bPSTX.ASTX** is cleared automatically. If no new packets need to be transmitted, the *Packet Transmitted* status bit should simply be cleared by setting **bPCTC.RATX**. This ends the handling of a packet transmitted event.

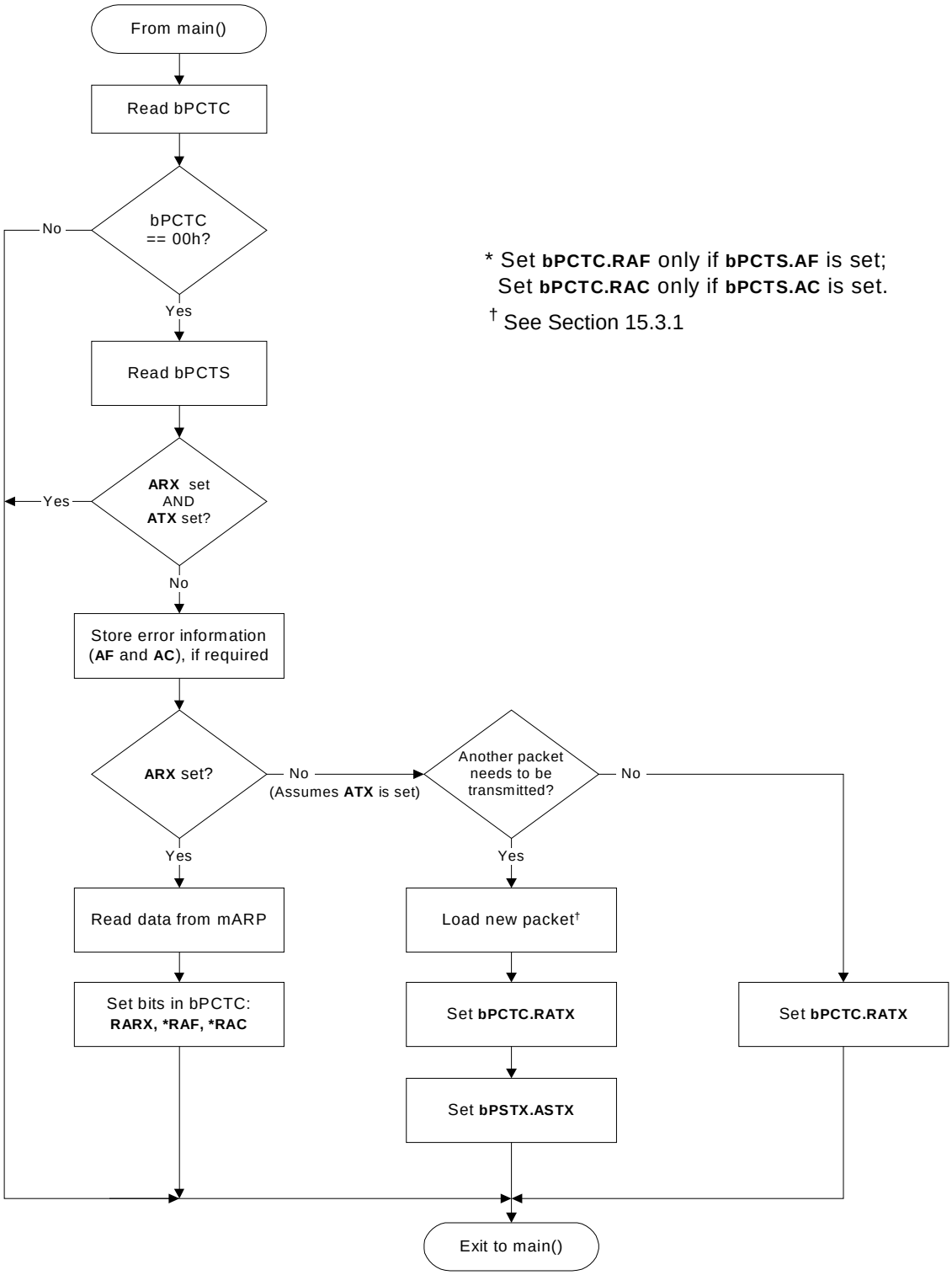


Figure 15-2: Handling Packet Data Transfer by Polling

15.3.3 Interrupt-Based Packet Data Handling

As mentioned in Section 15.2, an interrupt occurs on the $\overline{\text{AINT}}$ pin when a packet transmission is finished or a valid packet has been received (correct target address and CRC). After having entered the interrupt service routine, the Packet Status Register (bPCTS) must be read. If needed, the error information (**bPCTS.AF** and **bPCTS.AC**) can be stored for later use in the main loop of the application.

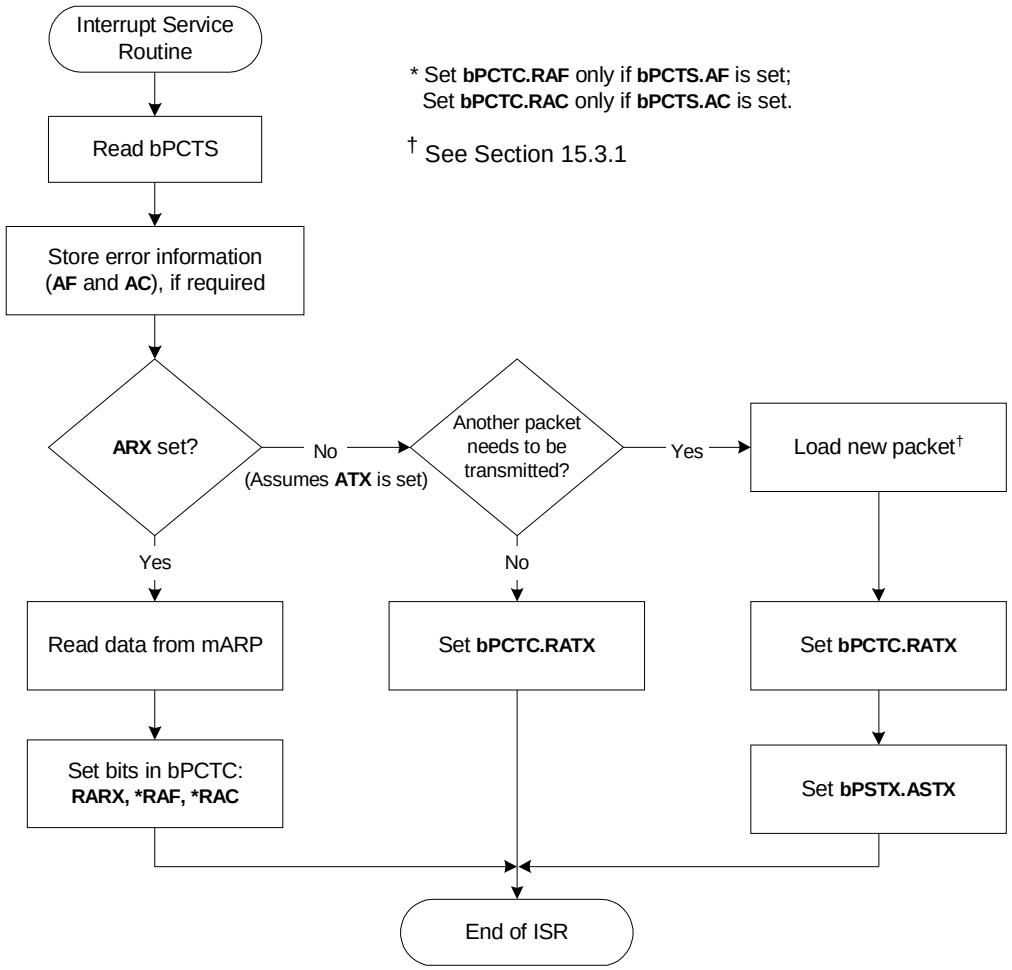


Figure 15-3: Handling Packet Data Transfer via Interrupt

If the **bPCTS.ARX** bit is set, a packet has been received, and the received packet data is in the Asynchronous Receive Packet Buffer (mARP). After having read mARP, the buffer must be released (unlocked) to allow receiving the next packet. This is done by setting the **bPCTC.RARX** bit. If the **bPCTS.AF** or **bPCTS.AC** error bits have been set, they should also be cleared to prepare them for the next message. This is done by setting **bPCTC.RAF** and **bPCTC.RAC**.This ends the handling of a packet received event.

If the interrupt was caused by a packet transmit event (**bPCTS.ATX** set), a new packet can be prepared for transmission if more packets need to be sent. If a new packet needs to be transmitted, the new packet data and all the header information should be loaded into the appropriate registers (see Figure 15-1). The Packet Transmit interrupt is then cleared by setting the **bPCTC.RATX**, and a new transmission is initiated by setting **bPSTX.ASTX**. Once the asynchronous channel is arbitrated for and the packet is sent, the $\overline{\text{AINT}}$ pin goes low and **bPSTX.ASTX** is cleared automatically. If no new packets need to be transmitted, the Packet Transmit Interrupt should simply be cleared (**bPCTC.RATX** reset). This ends the handling of a packet transmitted event.

16 OS8104 Startup

This section describes the start-up procedure for the OS8104. The flow charts (Figure 16-1, Figure 16-2, and Figure 16-3), provided as examples, describe the basic steps that must be performed to set up the standard configuration of the OS8104. The first flow chart must be followed every time the OS8104 is powered up. After having processed the main flow, the other two flow charts illustrate the steps necessary to configure the chip either as the timing-master or as a timing-slave mode. In this example, the Control Port is configured in I^2C mode (set by tying **PAR_CP** low and placing a pull-up on **SCL**).

After power-up is complete and the power supplies have stabilized, the OS8104 must be reset by a rising edge on the **RS** pin. At this time, the Control Port and Source Port configuration pins must be set properly for the intended modes of operation. The configuration options determine whether the Ports are used in serial or parallel mode, along with the format of the Control Port when used in serial mode. The configuration options are described in Chapter 4 on page 27.

To configure the Source Ports in serial mode and the Control Port in I^2C serial mode, the **PAR_CP** and **PAR_SRC** pins must be tied to ground, and the **SCL** pin must have a pull-up to VDDD (**SCL** is an open-drain pin in I^2C mode and requires a pull-up anyway). In addition, the **SCK** pin should be driven at the time lock is achieved (see Chapter 7 for information on ensuring **SCK** is driven at the time of lock).

After reset, the **bXCR.$\overline{\text{ABY}}$** bit is set, configuring the part for *All-bypass* mode (**RX** data is routed directly to **TX**, unaltered). The node is not *visible* to the Network (it has no valid node position number and is not addressable by other nodes).

In timing-slave mode, there is always a system clock. If there is no signal at **RX**, the system clock frequency is determined by the PLL running at its lowest frequency.

> After reset, the chip is configured in timing-slave mode; therefore, the crystal oscillator is disabled. It will not oscillate until the Clock Manager is configured for using the crystal as the clock source.

The start-up procedures listed in this section are useful for understanding the OS8104 requirements; however, these procedures do not cover all the start-up requirements of the *MOST Specification*. For more information about MOST requirements, see the *MOST Specification*, *Net Interface* Section [7].

16.1 Set Up After Power Up Reset

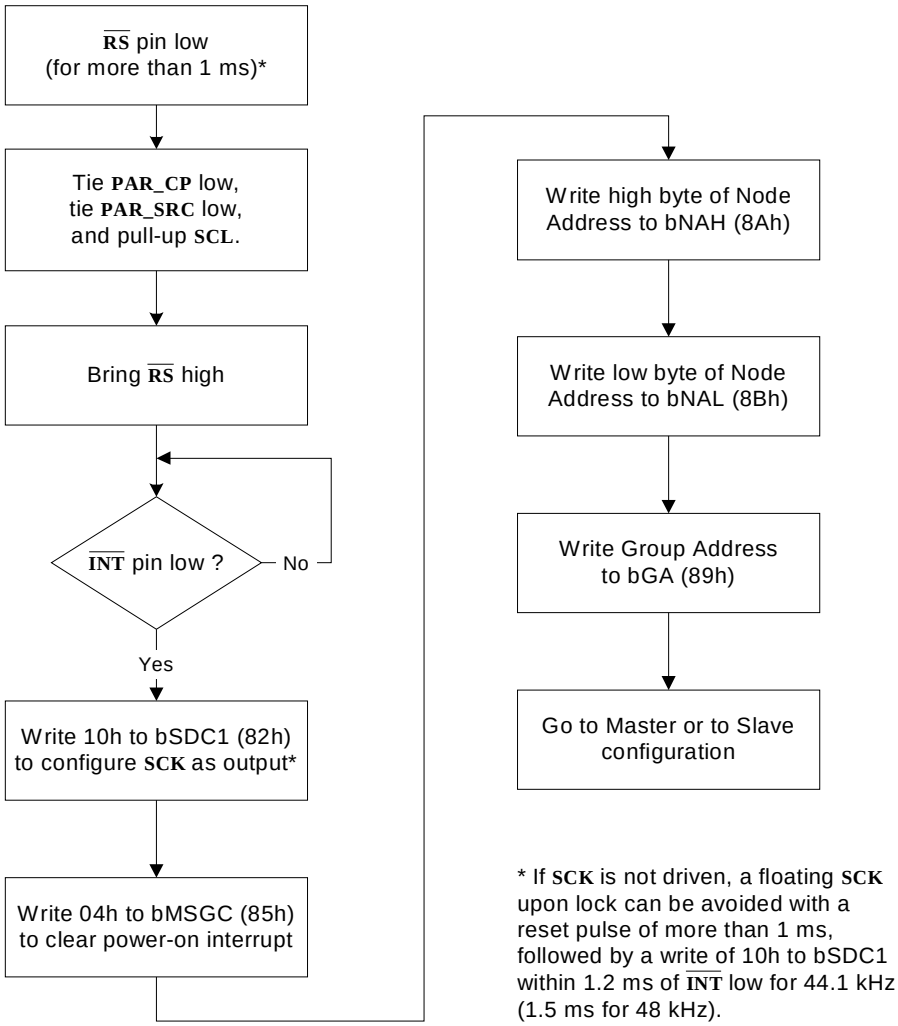


Figure 16-1: Starting up - Initial Flow

16.2 Timing-Master Mode

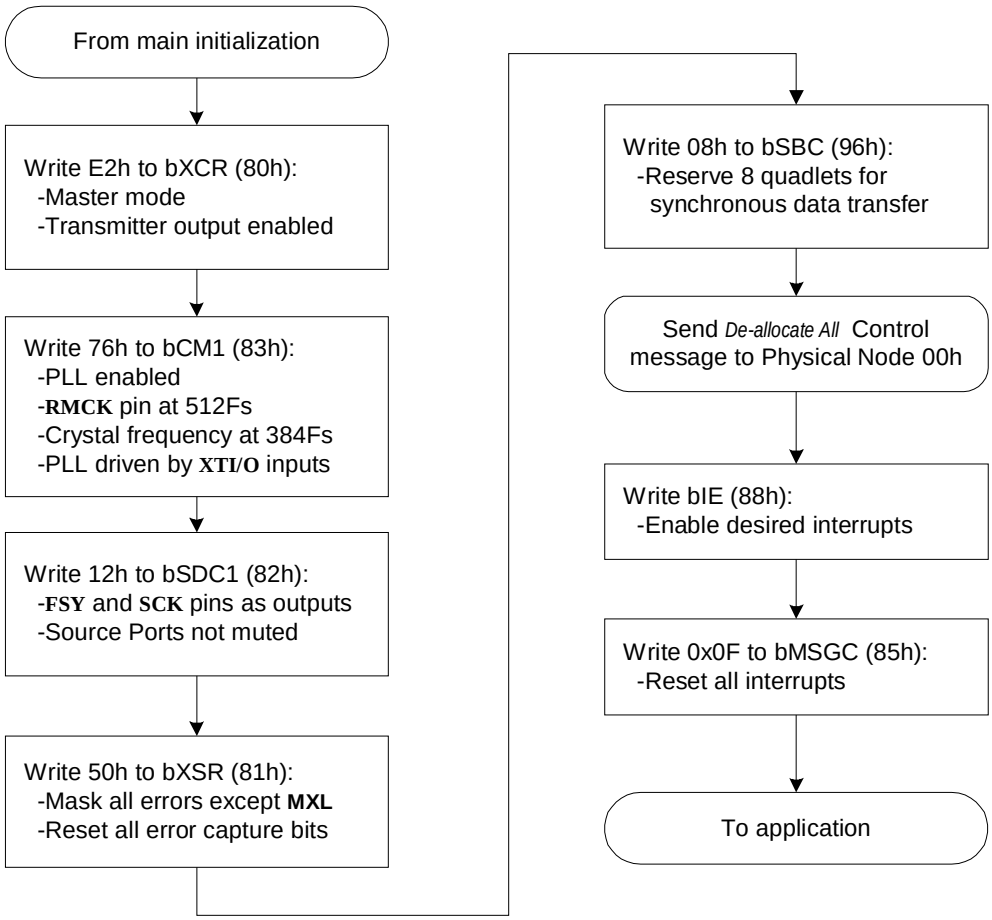


Figure 16-2: Starting up - Timing-Master Configuration

Sending a *De-allocate All* message is described in Section 13.5.2.5 on page 129.

16.3 Timing-Slave Mode

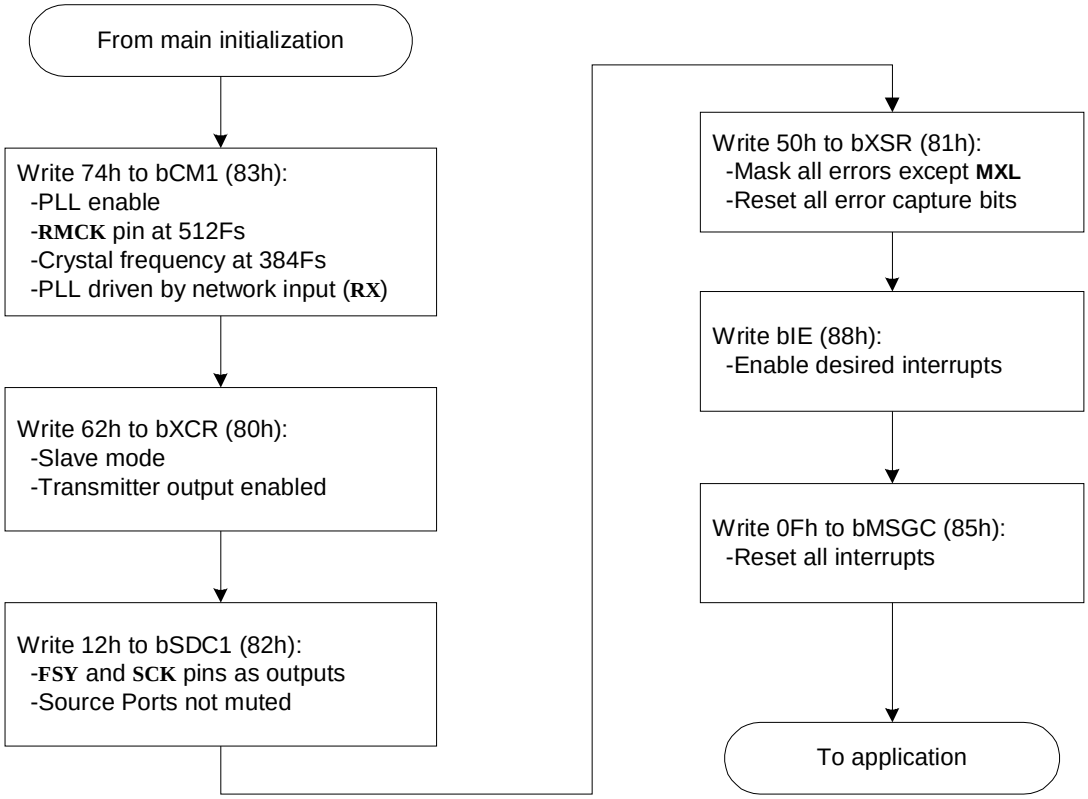


Figure 16-3: Starting up - Timing-Slave Configuration

16.4 Version Number

The current chip version is provided after hardware reset ($\overline{RS}$) or power-up reset by reading memory locations C4h, C5h and C6h, as shown in Table 16-1.

OS8104 Revision	Part Marking		Month	Year
		C4h	C5h	C6h
A	TEGEDAA	10h	12h	97h
B	TEGEDAB	11h	10h	98h
C	TEGFCAA	12h	06h	99h
D	TEGFCBP	13h	07h	00h

Table 16-1: OS8104 Version Numbers

The version number will be overwritten by the first write access to mXCMB.

17 Stand-Alone Mode

The OS8104 can be configured in *Stand-Alone* or *Remote-Controlled* mode, where no local controlling hardware is needed. In this mode, the OS8104 receives all commands via the Network and can control peripherals through the Control Port, which is automatically configured as an I^2C master. Another Network node can then configure and control the OS8104 using *Remote Read* or *Remote Write* Control messages (described in Section 13.5.2 on page 124). The OS8104 can also transmit MOST Control messages when interrupts occur. Currently, the *MOST Specification* does not support Stand-Alone mode.

17.1 Entering Stand-Alone mode

To configure the OS8104 for Stand-Alone mode, the pins $\overline{\mathbf{RD}}$ and $\overline{\mathbf{WR}}$ must be tied to GND before the rising edge of $\overline{\mathbf{RS}}$ and stay low through the completion of the power-up initialization procedure.

The chip starts up with the following configuration:

- MOST timing-slave mode (clock recovered from the Network)
- **TX** output enabled
- Source bypass bit **bXCR.SBY** active (all synchronous data transferred directly from **RX** to **TX**)
- All-bypass bit **bXCR.$\overline{\text{ABY}}$** inactive (bit is set)
- Automatic mute on error
  Whenever an error occurs, the outputs of the Source Ports are automatically muted by setting them constantly to 0. Upon start-up, only coding errors and transceiver errors are enabled. The S/PDIF lock error condition is masked.
- Control Port configured as I^2C master. The OS8104 only supports single I^2C master operation. The I^2C speed in this mode is kept below 100 kHz to minimize clock stretching.
- Automatic Zero-Power mode on missing Network activity
- The $\overline{\mathbf{INT}}$ pin is configured as an input for receiving external application interrupts.
- Responds to Group Addresses 03F0h (bGA = F0h)

17.2 Registers

When Stand-Alone mode is active, the following registers are accessible:

- bCM1 - Clock Manager register 1 (Section 9.1 on page 89)
- bCM2 - Clock Manager register 2 (Section 9.1 on page 90)
- bSDC1 - Source Data Control register 1 (Section 7.2.1 on page 46)
- bSDC2 - Source Data Control register 2 (Section 7.2.2 on page 48)
- bSDC3 - Source Data Control register 3 (Section 7.2.3 on page 49)
- bNDR - Node Delay register (Section 6.2.6 on page 41)
- bNPR - Node Position register (Section 6.2.7 on page 41)
- bXCR - Transceiver Control register - bit **SBY** only (Section 6.2.1 on page 37)
- mSIMB - Stand-Alone I^2C Messaging Buffer
- bXRTY - Transmit Retry register (Section 13.2.4 on page 118)
- bXTIM - Transmit Retry Time register (Section 13.2.5 on page 118)
- mXCMB - Transmit Control Message Buffer (Section 13.2.7 on page 120)

17.3 mSIMB (Stand-Alone Control Port Message Buffer)

The Stand-Alone Control Port Messaging buffer (mSIMB) is a unique message buffer provided to support I^2C-master communication in Stand-Alone mode.

EBh mSIMB Stand-Alone Control Port Message Buffer

Byte	Name	Description	Default
EBh	bSIMC	Number of bytes sent (byte count)	00h
ECh	bSITA	I^2C target address	00h
EDh	bSIMA	I^2C MAP	00h
EEh	bSITD0	I^2C transfer data (byte0)	00h
EFh	bSITD1	I^2C transfer data (byte1)	00h
F0h	bSITD2	I^2C transfer data (byte2)	00h
F1h	bSITD3	I^2C transfer data (byte3)	00h
F2h	bSITC	I^2C transfer control/status byte	00h

Table 17-1: mSIMB (Stand-Alone CP Message Buffer)

bSIMC Byte count. bSIMC contains the number of bytes (after bSIMC) to be sent via the Control Port:
Minimum value is 2 (only bSITA and bSIMA are sent)
Maximum value is 6 (bSITA to bSITD3 are sent)

bSITA I^2C target address. bSITA contains the address of the particular external I^2C device being accessed. I^2C devices are connected to the Control Port (pins **SCL** and **SDA**), which is configured as an I^2C master. The correct setting of the LSB in the I^2C address (for read or write access) is done automatically by the OS8104.

bSIMA I^2C MAP. bSIMA contains the Memory Address Pointer for the I^2C peripheral.

bSITD[0:3] I^2C transfer data bytes 0 to 3. These bytes contain data to be transferred to the external peripheral.

bSITC I^2C control/status byte. bSITC controls and monitors the Control Port data transfer.
Command values (when writing to bSITC):
06h: Execute I^2C write operation
07h: Execute I^2C read operation
Status values (when reading from bSITC):
00h: Operation finished; Ready for new command
Other: Busy

17.4 Writing to External Peripherals

The Stand-Alone Message buffer (mSIMB) must be filled before the node can write to the external peripheral. The first value in the buffer is the number of bytes to be transmitted and includes all bytes from bSITA through bSITD3 (inclusive). The count is 6 to transfer all data bytes through bSITD3. The minimum count is 2 to transfer only the target address of the device (bSITA) and the MAP (bSIMA).

In the following example, four data bytes are transmitted, starting at memory location 33h on the external I^2C peripheral. The I^2C address of the external device is 50h, and the node position of the OS8104 in Stand-Alone mode is 05h.

Two devices are involved in transferring data:

- RCC - The chip running in Stand-Alone mode.
- CMS - The node sending the *Remote Read* and *Remote Write* Control messages.

The bSITD[0:3] bytes contain the four data bytes, bSITA contains the I^2C target address of 50h, and bSIMA contains the MAP address 33h, which specifies where to start storing the data. The count is bSIMC = 4 + 2 = 6.

The correct content of mSIMB in the RCC node is:

mSIMB	Value	Stand-Alone Control Port Message Buffer
bSIMC	06h	Count of bytes to transmit
bSITA	50h	I^2C address of target device
bSIMA	33h	I^2C MAP
bSITD0	48h	any data
bSITD1	75h	any data
bSITD2	68h	any data
bSITD3	75h	any data
bSITC	06h	I^2C control/status

Table 17-2: mSIMB when Writing to I^2C in Stand-Alone Mode

In order to fill mSIMB of the RCC node, the CMS node must send a *Remote Write* Control message. Table 17-3 shows the correct content of the Transmit Control Message Buffer (mXCMB) of the CMS node. Since mSIMB is located at memory location EBh in the RCC node, this value must be written to the MAP byte in mXCMB (address C5h) of the CMS node.

mXCMB Addr.	Contents	Transmit Control Port Message Buffer
C0h	01h	Priority; Default 01h
C1h	02h	*Remote Write* operation
C2h	04h	Target address high (of RCC in MOST Network)
C3h	05h	Target address low
C4h	00h	rsvd
C5h	EBh	Memory location in RCC to write to (mSIMB buffer)
C6h	08h	Count of data bytes to be written
C7h	06h	D0 of Control Message (bSIMC)
C8h	50h	D1 of Control Message (bSITA)
C9h	33h	D2 of Control Message (bSIMA)
CAh	48h	D3 of Control Message (bSITD0)
CBh	75h	D4 of Control Message (bSITD1)
CCh	68h	D5 of Control Message (bSITD2)
CDh	75h	D6 of Control Message (bSITD3)
CEh	06h	D7 of Control Message (bSITC)

Table 17-3: Stand-Alone Mode: Write Example

The mechanism for sending the *Remote Write* Control message is described in Section 13.5.2.3 on page 126.

When the RCC node receives this message, the I^2C data transmission will start immediately. The current status of this transmission can be obtained by reading bSITC (address F2h) via a *Remote Read* Control message. As long as the chip is busy transmitting through the Control Port, bSITC will be non-zero.

Reading a zero from bSITC (STATUS) indicates the conclusion of the remote node's Control Port transmission. The *Remote Read* Control message for checking the transfer status would be:

mXCMB Addr.	Contents		Transmit Control Message Buffer
	Sent	Returned	
C0h	01h	01h	Priority; Default 01h
C1h	01h	01h	*Remote Read* operation
C2h	04h	04h	Target address high (of RCC in MOST Network)
C3h	05h	05h	Target address low
C4h	00h	00h	rsvd
C5h	EBh	EBh	Memory location in RCC to read from (mSIMB buffer)
C6h	00h	00h	rsvd
C7h	x	x	D0 of Control Message (bSIMC)
C8h	x	x	D1 of Control Message (bSITA)
C9h	x	x	D2 of Control Message (bSIMA)
CAh	x	x	D3 of Control Message (bSITD0)
CBh	x	x	D4 of Control Message (bSITD1)
CCh	x	x	D5 of Control Message (bSITD2)
CDh	x	x	D6 of Control Message (bSITD3)
CEh	x	STATUS	D7 of Control Message (bSITC) Contains current status

Table 17-4: Stand-Alone Mode: Write Example Status Check

More information about handling *Remote Read* messages can be found in Section 13.5.2.2 on page 125. The flow chart shown in Figure 17-1 shows how a Control Port transmission can be done:

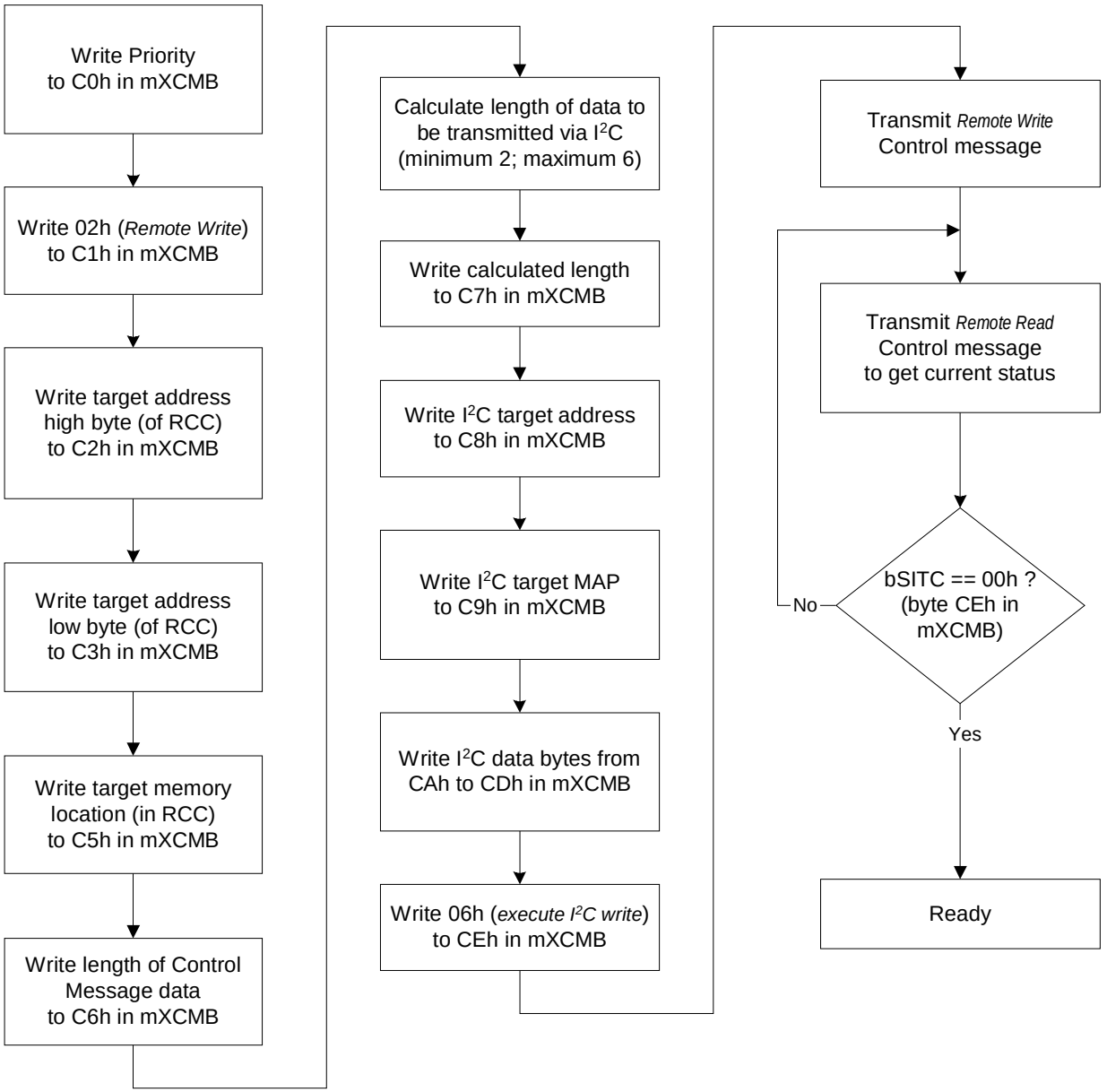


Figure 17-1: Stand-Alone Mode: Write Flow

17.5 Reading from External Peripherals

When reading data from an external peripheral via the I^2C bus, several MOST Control messages are needed. The controlling node must send one *Remote Write* message to transfer the I^2C address, I^2C MAP, and the number of bytes to be read (bytes are always read from the external peripheral in increments of four). The read cycles between the OS8104 and the external peripheral are handled automatically, such that only one *Remote Write* access is needed for initializing the data transfer via the I^2C bus. The controlling node must then send one or more *Remote Read* messages to transport data (read via I^2C) and status back.

In the following example, data is read from an OS8104 running in Stand-Alone mode (RCC device as in the last example), with logical address of 0123h.

For reading I^2C data, only the I^2C target address and the I^2C MAP must be transferred to the external peripheral. Since the number of bytes to be read from the external peripheral is always four, the number of bytes to transmit (bSIMC) is always six. The external peripheral has an I^2C address of 50h, and data will be read starting from memory location 33h. Table 17-5 shows the correct data to transfer to mSIMB for this example.

mSIMB	Value	Stand-Alone Control Port Message Buffer
bSIMC	06h	Number of bytes to transmit (always 06h)
bSITA	50h	I^2C address of target device
bSIMA	33h	I^2C MAP
bSITD0	x	fill up data (don't care)
bSITD1	x	fill up data (don't care)
bSITD2	x	fill up data (don't care)
bSITD3	x	fill up data (don't care)
bSITC	07h	I^2C read operation

Table 17-5: Stand-Alone Mode: mSIMB Read Example

To get mSIMB of the RCC node filled, the controlling node (CMS) must send a *Remote Write* Control message. Table 17-6 shows the correct contents of the Transmit Control Message Buffer (mXCMB) of the CMS node. Since mSIMB is located at memory location EBh in the RCC node, this value must be written to the MAP byte in mXCMB (address C5h) of the CMS node.

mXCMB Addr.	Value	Transmit Control Message Buffer
C0h	01h	Priority; Default 01h
C1h	02h	*Remote Write* operation
C2h	01h	Target address high (of RCC)
C3h	23h	Target address low
C4h	00h	rsvd
C5h	EBh	Memory location in RCC to write (mSIMB buffer)
C6h	08h	Number of data bytes to be written
C7h	06h	D0 of Control Message (bSIMC) — Must be 06h
C8h	50h	D1 of Control Message (bSITA)
C9h	33h	D2 of Control Message (bSIMA)
CAh	x	D3 of Control Message (don't care)
CBh	x	D4 of Control Message (don't care)
CCh	x	D5 of Control Message (don't care)
CDh	x	D6 of Control Message (don't care)
CEh	07h	D7 of Control Message (bSITC) — *Execute I^2C Read*

Table 17-6: Stand-Alone Mode: mXCMB Read Example

When the RCC node receives this message, the Control Port data transmission will start immediately. The current status of this transmission can be obtained by reading bSITC (address F2h) using a *Remote Read* Control message. As long as the chip is busy with the Control Port, bSITC will be non-zero. Reading 00h from bSITC indicates the completion of the Control Port read and that the data bytes are valid.

The data read from the external peripheral is stored in bytes bSITD[0:3]; therefore, reading the data and status can be accomplished using a single *Remote Read* message. Table 17-7 illustrates possible results of a *Remote Read*, when checking bSITC status.

mXCMB Addr.	Contents		Transmit Control Message Buffer
	Not Valid	Valid	
C0h	01h	01h	Priority; Default 01h
C1h	01h	01h	*Remote Read* operation
C2h	01h	01h	Target address high (of RCC)
C3h	23h	23h	Target address low
C4h	00h	00h	rsvd
C5h	EBh	EBh	Memory location in RCC to read from (mSIMB buffer)
C6h	00h	00h	rsvd
C7h	x	06h	D0 of Control Message (bSIMC)
C8h	x	50h	D1 of Control Message (bSITA)
C9h	x	33h	D2 of Control Message (bSIMA)
CAh	x	ddh	D3 of Control Message (Data byte0 from I^2C device)
CBh	x	ddh	D4 of Control Message (Data byte1 from I^2C device)
CCh	x	ddh	D5 of Control Message (Data byte2 from I^2C device)
CDh	x	ddh	D6 of Control Message (Data byte3 from I^2C device)
CEh	07h	00	D7 of Control Message (bSITC)

Table 17-7: Stand-Alone Mode: mXCMB Read Example Status

More information about handling *Remote Read* messages can be found in Section 13.5.2.2 on page 125. The flow chart in Figure 17-2 shows the reading of Control Port data on a remote node.

17.6 Message on Interrupt

When the OS8104 is configured for Stand-Alone mode, it can send a Control message when the external application pulls the $\overline{\text{INT}}$ pin low (which is configured as an input in Stand-Alone mode).

The minimum external interrupt pulse width is 2 µs.

This mechanism works in *single-shot* mode. Therefore, after an interrupt event has been reported, the interrupt mechanism must be re-armed. All Control message addressing modes and transmit message types are supported.

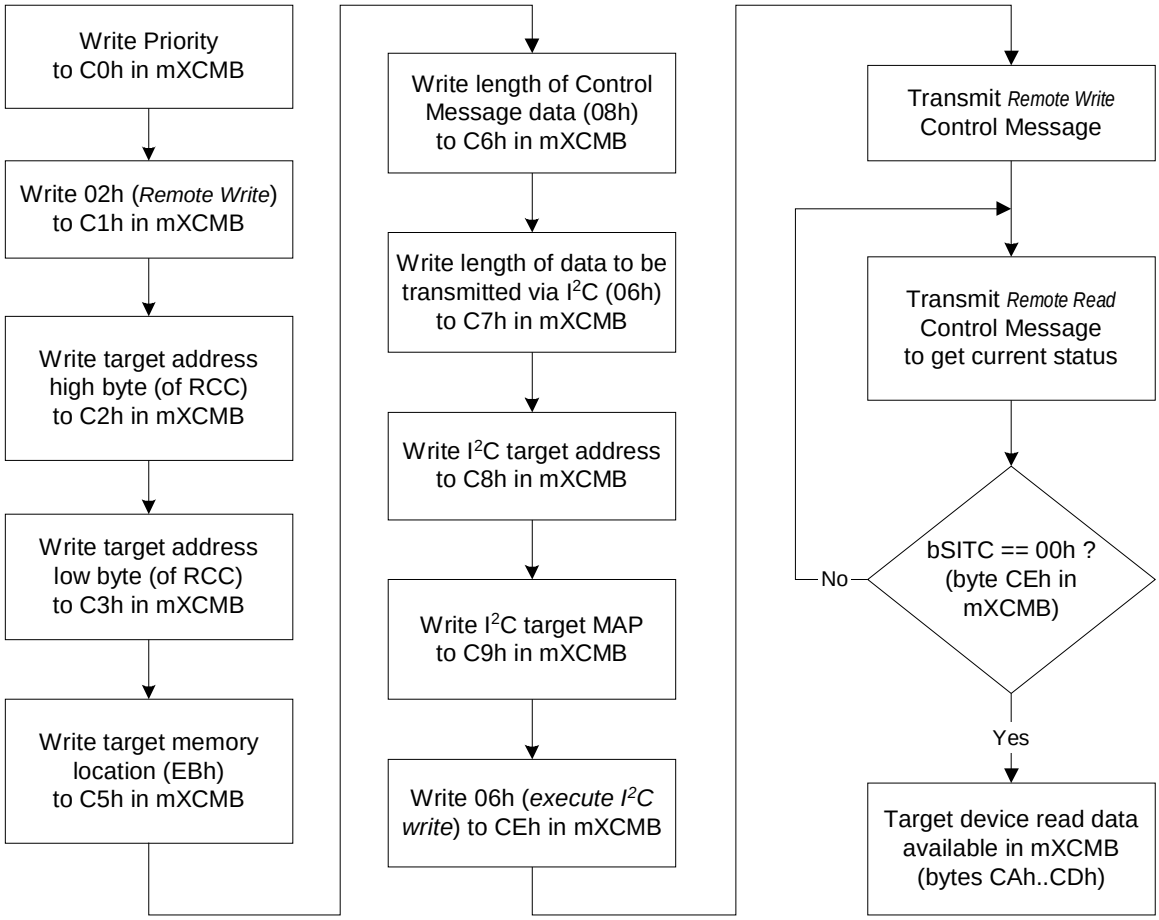


Figure 17-2: Stand-Alone Mode: Read Flow

First, the Transmit Control Message Buffer (mXCMB) must be filled with the message to be sent out when an interrupt occurs. Since the remote interrupt mechanism is armed by setting the MSB in bXTYP, the remote mXCMB should be filled, starting from location C2h (bXTAH). Once mXCMB on the remote node is filled, using the *Remote Write* Control message type, the interrupt mechanism can be armed by writing the remote bXTYP byte in mXCMB. The bXTYP on the remote node is filled with the appropriate message type, ORed with 80h, thereby setting the MSB. For example, to send a normal message (type 00h), the value in bXTYP would be 80h.

Priority, number of retries, and retry time can also be changed, if required (registers bXPRI, bXRTY, bXTIM).

As soon as an interrupt occurs, the OS8104 in Stand-Alone mode sends the prepared message and disarms the interrupt mechanism.

No errors are reported if the transmission is a failure.

The message-on-interrupt mechanism is re-armed by writing to bXTYP again.

For more information on sending Control messages, see Section 13.4 on page 122.

18 Electrical Characteristics

18.1 Absolute Maximum Ratings

Parameter (Note 1)	Min	Max	Unit
Storage Temperature	–65	150	°C
Ambient Temperature Under Bias	–55	125	°C
Junction Temperature		150	°C
Power Supply Voltage	–0.5	6.0	V
Power Dissipation		990	mW
DC Current to Any Pin Except Power and **RX** (Note 2)		±10	mA
Voltage on Any Pin	–0.3	VDD+0.3	V

Notes:

1. Operation at or above these limits may damage the device.
2. The **RX** pin can handle ±30 mA for as long as 500 ms.

18.2 Guaranteed Operating Conditions

The OS8104 operation is only guaranteed to function, as specified in this document, within the limits listed below.

Parameter	Min	Max	Unit
Operating Ambient Temperature	–40	85	°C
Power Supply Voltage	4.5	5.5	V
Operating Network Sample Frequency (Fs)	37.9	48.1	kHz

18.3 Thermal Characteristics

Parameter	Symbol	Value	Unit
Junction to case	θ_{JC}	9.3	°C/W
Junction to ambient, single-layer PCB (no power/ground plane)	θ_{JA}	68	°C/W
Junction to ambient, multi-layer PCB (power and ground planes)	θ_{JA}	45	°C/W

18.4 DC Characteristics

T_A = –40 to 85 °C; VDDD,VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Low Level Input Voltage: **XTO** Other Input pins	V_{IL}			0.1 × VDDD 0.8	V V	(Note 1)
High Level Input Voltage: **XTO** Other Input pins	V_{IH}	0.9 × VDDD 2.0			V V	(Note 1)
Low Level Output Voltage: **RMCK** Other Output pins	V_{OL}			0.4 0.4	V V	I_{OL} = 2.0 mA I_{OL} = 2.4 mA
High Level Output Voltage: **RMCK** Other Output pins (Note 2)	V_{OH}	VDDD – 0.5 VDDD – 0.5			V V	I_{OH} = –1.0 mA I_{OH} = –2.4 mA
Input Leakage Current	I_L			±10	µA	0 < Vin < VDDD
Digital Input Pin Capacitance: **RX** Other pins				5 10	pF pF	
Analog Supply Current: Normal Low-Power mode	I_A			20 20	mA mA	Fs = 48.1 kHz
Digital Supply Current: Normal Low-Power mode	I_D			160 50	mA mA	Fs = 48.1 kHz
Zero-Power mode: Analog and Digital Supplies combined	$I_A + I_D$			400	µA	

Notes:

1. When driven externally (not using a crystal).
2. Except for open-drain outputs: **$\overline{\text{WAKE\_UP}}$**, **$\overline{\text{AINT}}$**, **$\overline{\text{INT}}$**; and in I^2C mode: **SDA** and **SCL**.

18.5 Switching Characteristics

18.5.1 Clocks

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF; unless otherwise stated.

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Crystal Oscillator	f_{xtal}	9.7024		 24.6272	MHz MHz	256x(Fs = 37.9 kHz) 512x(Fs = 48.1 kHz)
Recovered Master Clock, **RMCK** (PLL locked to Fs)	f_{rmck}	2.4256		 73.8816	MHz MHz	64x(Fs = 37.9 kHz) 1536x(Fs = 48.1 kHz)
RMCK rise/fall time	t_{rmck}, t_{fmck}		7	10	ns	Load capacitance of 20 pF
Network Frame Frequency	Fs	37.9 5	44.1 10	48.1	kHz kHz	PLL locked PLL unlocked — **RX** low
Jitter Tolerance: (Note 7) (timing-master) **XTO**, **SR0**, **SCK** **RX** - revision B, C **RX** - revisions greater than C	t_{jit}	 0.8 $\frac{1}{3072Fs}$ 25			 ns (pp) ns (pp) ns (pp)	 Note 1 Note 2
Pulse Width Variation: **RX**	t_{pwmn} t_{pwmx}	0.7		 1.4	UI UI	Note 3
Average Pulse Width Distortion: **RX** - revision B, C **RX** - revisions greater than C	t_{apwd}	 0 -0.15		 0.36 0.36	 UI UI	Notes 3, 4
Pulse Width Distortion, **SR0**	t_{pwds}	–0.2		+0.2	UI	Note 5
RX input rise/fall time	t_{rrx}, t_{frx}		10		ns	Note 6
TX rise/fall time	t_{rtx}, t_{ftx}		2	7	ns	

Notes:

1. When locking to a crystal, **SR0**, or any **SCK** frequency, bCM4 (address 93h) must be set according to Table 9-4. The MOST Specification requires each node have a crystal-based master-clock source to support ring-break diagnostics, where nodes normally configured as timing-slaves are reconfigured as the timing-master.
2. Worse case over all **RX**-to-**TX** phase variations. The actual jitter tolerance could be as high as 30 ns (pp). For Fs = 44.1 kHz, **RX** Jitter Tolerance is 7.38 ns (pp) max.
3. **bXSR2.INV** = 0. When the MOST Network frequency is Fs = 44.1 kHz, one UI is 22.1 ns, which is $\frac{t_{rxbp}}{2}$. The pulse width variation is defined as the sum of the average Pulse Width Distortion plus high-frequency jitter. When configured as a timing-master, t_{pwmx} is 1.36 UI.
4. The average PWD (APWD) spec only applies to timing-slave nodes (**bXCR.MTR** clear), and is defined as $t_{apwd} = \frac{t_{pwmx} + t_{pwmn} - t_{rxbp}}{2}$ (see Figure 18-1). In addition, the bCM4 (address 93h) must be set according to Table 9-4. The **FLT** pin is a high-impedance node; therefore, leakage must be kept below 1 µA or APWD tolerance can be adversely affected.
5. **SR0** configured for S/PDIF and not as the timing-master node. One UI (unit interval) is defined as a single bi-phase period (one half of a bit period). Therefore, for 1xSPDIF mode with Fs = 44.1 kHz, one UI is 177 ns.
6. Recommended. **RX** rise and fall time should be as short as possible to minimize jitter and PWD.
7. Jitter Tolerance: This defines the amount of jitter that the part will tolerate without errors.

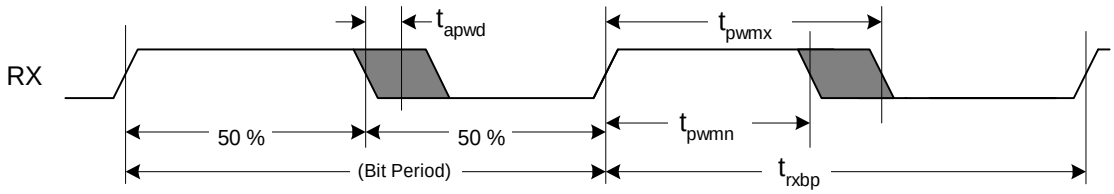


Figure 18-1: RX Pulse-Width Distortion

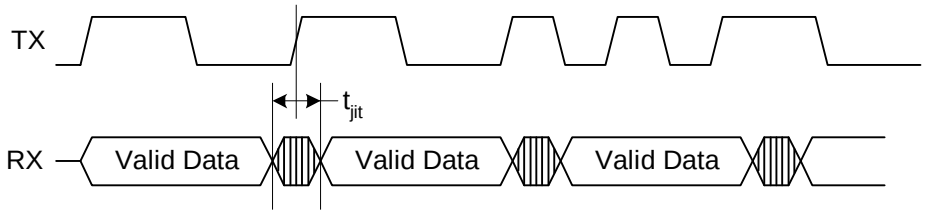


Figure 18-2: RX Jitter

18.5.2 Reset

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, Load Capacitance = 40 pF; unless otherwise stated.

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Reset Pulse Width	t_{rspw}	10			ns	Notes 1, 2
Power-up Access	t_{puac}			1	ms	Note 3
Configuration pin setup to **RS** rising: **SCL**, **PAR_CP**, **PAR_SRC**, **ASYNC**, **WR**, **RD**	t_{cpsrs}	20			ns	
Config. pin hold from **RS** rising: **SCL** **PAR_CP**, **PAR_SRC**, **ASYNC** **WR**, **RD**	t_{cphrs}	20 (Note 4) 1			ns ms	 Until t_{puac} time expires.

Notes:

1. The OS8104 must be reset after the power supplies have stabilized for proper operation.
2. If **INT** was low prior to **RS** going low, short t_{rspw} times might not give **INT** time to rise properly, with **INT** still appearing low after **RS** rises. **INT** should be checked for a high-to-low edge before accessing the chip.
3. Indicated via the power-on interrupt, **INT** pin. All accesses to the OS8104 must wait until the power-on interrupt occurs before accessing the chip via the serial or parallel interfaces.
4. **PAR_CP**, **PAR_SRC**, and **ASYNC** can only change when **RS** is asserted, and must remain constant when **RS** is de-asserted.

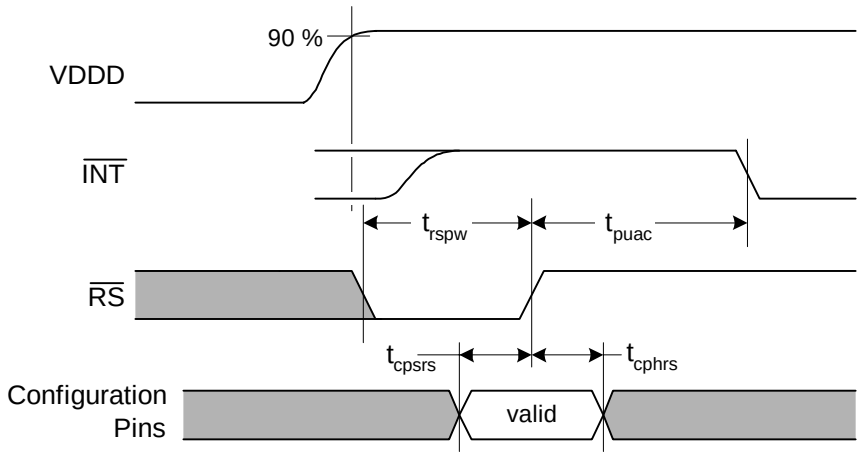


Figure 18-3: Reset Pulse Width

18.5.3 Parallel Interface

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF

Parameter	Symbol	Min	Typ	Max	Unit	Comments
Read Operation:						
$\overline{\textbf{RD}}$ low time	t_{rdp}	50			ns	
$\overline{\textbf{RD}}$ high time	t_{rdh}	20			ns	
$\overline{\textbf{RD}}$ low to **D[7:0]** valid	t_{drd}			27	ns	
D[7:0] hold from $\overline{\textbf{RD}}$ high	t_{dhd}	0	2	10	ns	
PAD[1:0] valid to $\overline{\textbf{RD}}$ low	t_{asur}	10			ns	
PAD[1:0] hold from $\overline{\textbf{RD}}$ high	t_{ahdr}	6			ns	
Write Operation:						
$\overline{\textbf{WR}}$ low time	t_{wrp}	50			ns	
$\overline{\textbf{WR}}$ high time	t_{wrh}	20			ns	
D[7:0] valid to $\overline{\textbf{WR}}$ high	t_{dsu}	10			ns	
D[7:0] hold from $\overline{\textbf{WR}}$ high	t_{dhd}	6			ns	
PAD[1:0] valid to $\overline{\textbf{WR}}$ low	t_{asuw}	10			ns	
PAD[1:0] hold from $\overline{\textbf{WR}}$ high	t_{ahdw}	6			ns	
Parallel-Synchronous and Parallel-Combined modes:						
SRC_FLOW period	t_{ssf}	2.598		3.298	µs µs	8x(Fs = 48.1 kHz) 8x(Fs = 37.9 kHz)
$\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ inactive to **SRC_FLOW** low (Source Port accesses only)	t_{rwsf}	30			ns	Note 1
$\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ inactive to **SRC_FLOW** high (Source Port accesses only)	t_{rwsr}	730			ns	Note 1
SRC_FLOW high to $\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ low	t_{ssfh}	0			ns	
SRC_FLOW rise/fall time	t_{rsf}, t_{fsf}		7	10	ns	
$\overline{\textbf{WR}}$ Control Port high to $\overline{\textbf{WR}}$ Source Port low	t_{wrcpsp}	$\frac{1}{128Fs}$			s	Note 2 Revisions prior to D only
Parallel-Asynchronous mode:						
$\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ high to **SRC_FLOW** high	t_{asfd}			40	ns	
SRC_FLOW high time	t_{asfh}		$\frac{1}{16Fs}$	$\frac{1.1}{8Fs}$	s	
Control Port Parallel Access:						
$\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ high to **CP_FLOW** high	t_{cfd}			20	ns	
CP_FLOW high time	t_{cfh}		8		µs	
$\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ high (CP access) to $\overline{\textbf{WR}}$, $\overline{\textbf{RD}}$ high (non-CP access)	t_{cpd}	$\frac{1.02}{512Fs}$				Note 3

Notes:
1. When accessing the Source Port in Parallel-Synchronous and Parallel-Combined modes, $\overline{\textbf{WR}}$ and $\overline{\textbf{RD}}$ must be inactive for t_{rwsf} before the falling edge of **SRC_FLOW** and t_{rwsr} before the rising edge of **SRC_FLOW**. If these timing violations are violated, data within the FIFO could be corrupted. These timing restrictions do not apply to parallel Control Port accesses.
2. In revisions prior to D that are configured for Parallel-Synchronous or Parallel-Combined mode, there must be a delay of $\frac{1}{128Fs}$ between the rising edge of $\overline{\textbf{WR}}$ for a Control Port write and the falling edge of $\overline{\textbf{WR}}$ for a Source Port write; otherwise, the Control Port data written will be corrupted. This timing parameter is not required when using revision D or higher silicon.
3. When the Network frame frequency (Fs) drifts outside of the PLL locked frequency specification, each Control Port access must be followed by a t_{cpd} delay before a seceding non-Control Port access can occur. This timing restriction does not apply to back-to-back Source Port, bCP, or bSP accesses.

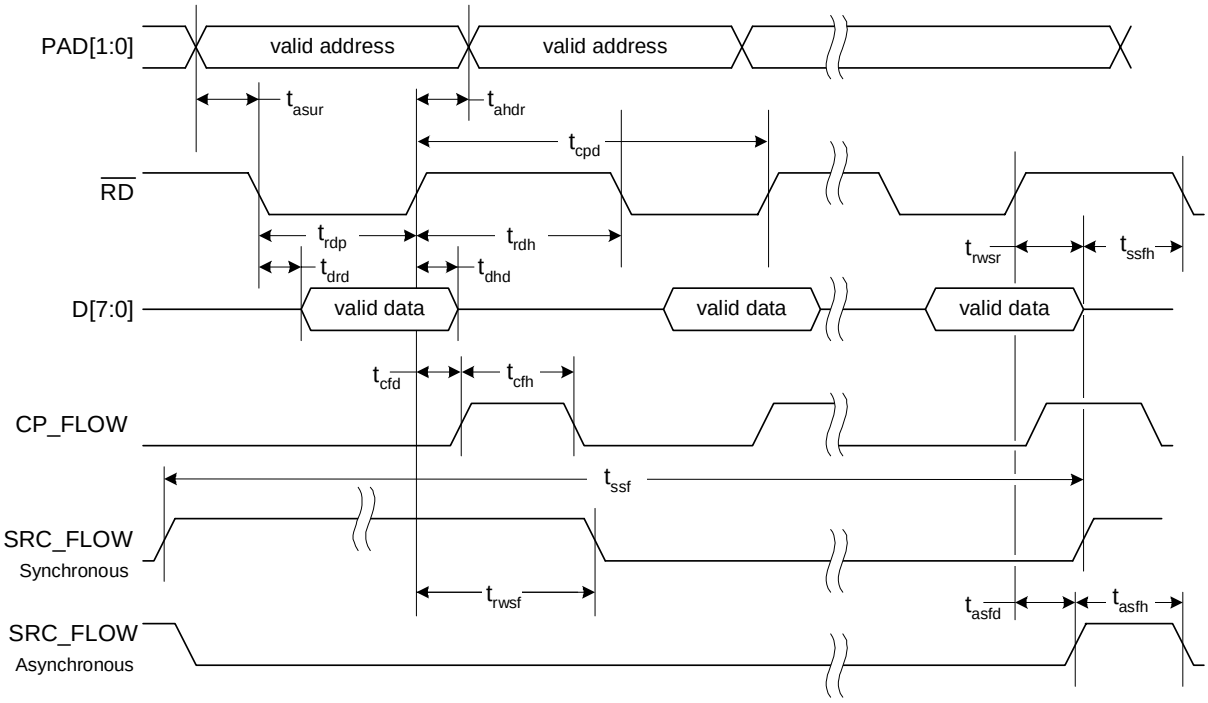


Figure 18-4: Parallel Read Operation

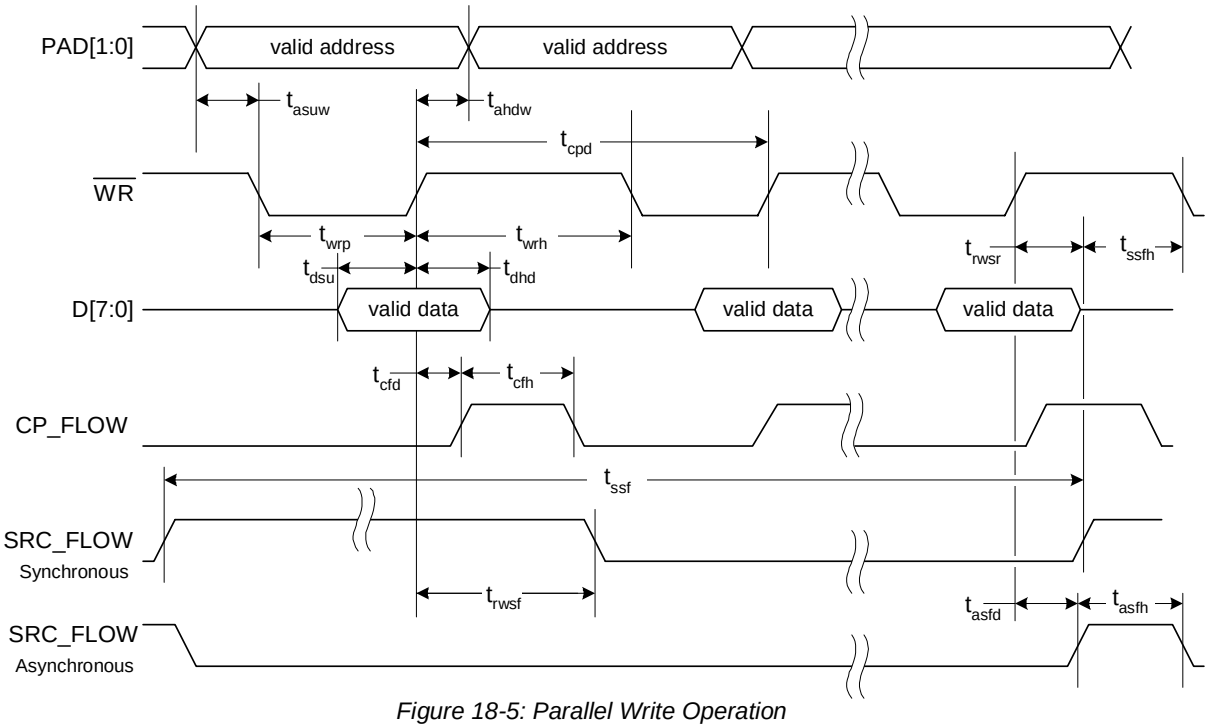


Figure 18-5: Parallel Write Operation

18.5.4 Source Ports (Serial) External Clocking

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF

Parameter	Symbol	Min	Typ	Max	Unit	Comments
FSY frequency (Note 1)	f_{fsy}	37.9	44.1	48.1	kHz	
SCK frequency (Note 1)	f_{sck}	0.3032		12.3136	MHz MHz	Min.: 8Fs @ 37.9 kHz Max.: 256Fs @ 48.1 kHz
SCK low time	t_{sckl}	25			ns	
SCK high time	t_{sckh}	25			ns	
FSY valid to **SCK** rising	t_{fsys}	25			ns	Notes 2, 3
FSY hold from **SCK** rising	t_{fsyh}	25			ns	Notes 2, 3
SR[3:0] valid to **SCK** rising	t_{srs}	25			ns	Notes 2, 4
SR[3:0] hold from **SCK** rising	t_{srh}	25			ns	Notes 2, 4
SCK falling to **SX[3:0]** valid	t_{sxv}			30	ns	Notes 2, 4

Notes:

1. **SCK** and **FSY** inputs must be frequency locked to the OS8104 master clock (**RMCK** output clock).
2. **SCK** active edge (edge where data is stable and not changing) is determined by the **bSDC1.EDG** bit. The parameters and Figure 18-6 are illustrated with **EDG** set. If **EDG** is clear, reverse the edge in the parameters listed above and invert **SCK** in the diagram below.
3. **FSY** polarity is determined by the **bSDC1.POL** bit.
4. The MSB of **SR[3:0]** and **SX[3:0]** is either the first or the second bit after **FSY** changes, based on the **bSDC1.DEL** bit.

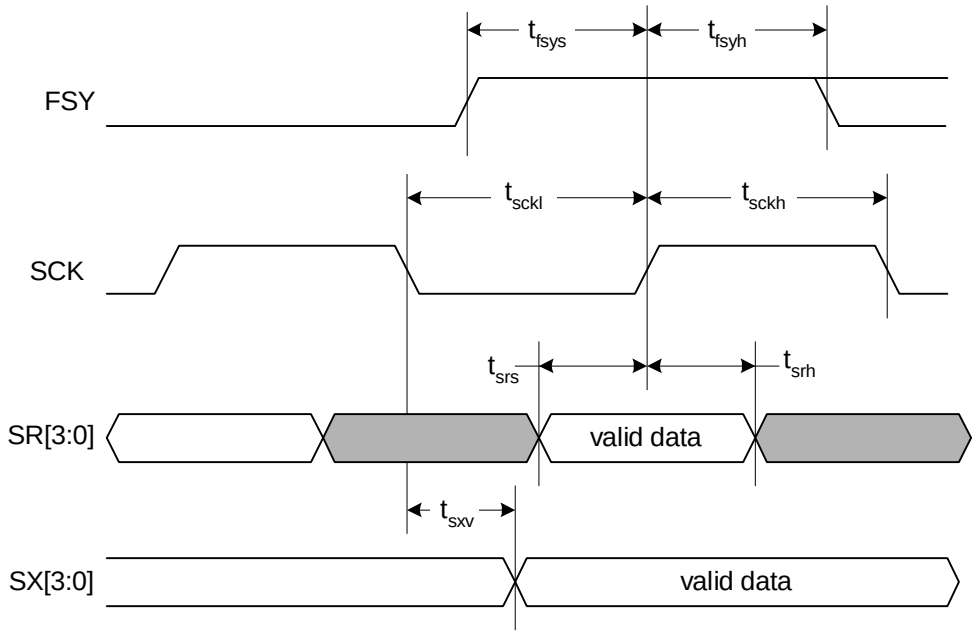


Figure 18-6: Source Port External Timing

18.5.5 Source Ports (Serial) Internal Clocking

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF

Parameter	Symbol	Min	Typ	Max	Unit	Comments
FSY frequency	f_{fsy}	37.9	44.1	48.1 96.2 192.4	kHz kHz kHz	**MFSY** = 0 **MFSY** = 1, **SCK** = 128Fs **MFSY** = 1, **SCK** = 256Fs
SCK frequency	f_{sck}	0.3032		 12.3136	MHz MHz	Min.: 8Fs @ 37.9 kHz Max.: 256Fs @ 48.1 kHz
SCK low time	t_{sckl}	25			ns	
SCK high time	t_{sckh}	25			ns	
SCK falling to **FSY** valid	t_{fsyv}	–25		25	ns	Notes 1, 2
SCK/FSY rise/fall time	t_{rsp}, t_{fsp}		7	10	ns	
SR[3:0] valid to **SCK** rising	t_{srs}	25			ns	Notes 1, 3
SR[3:0] hold from **SCK** rising	t_{srh}	25			ns	Notes 1, 3
SCK falling to **SX[3:0]** valid	t_{sxv}	–25		30	ns	Notes 1, 3

Notes:

1. **SCK** active edge (edge where data is stable and not changing) is determined by the **bSDC1.EDG** bit. The parameters and Figure 18-7 are illustrated with **EDG** set. If **EDG** is clear, reverse the edge in the parameters listed above and invert **SCK** in the diagram below.
2. **FSY** polarity is determined by the **bSDC1.POL** bit. **FSY** is a 50 % duty cycle clock where the number of **SCK** cycles per **FSY** period is defined by the **bSDC2.SPR[2:0]** bits and the **bSDC1.NBR** bit.
3. The MSB of **SR[3:0]** and **SX[3:0]** is either the first or the second bit after **FSY** changes, based on the **bSDC1.DEL** bit.

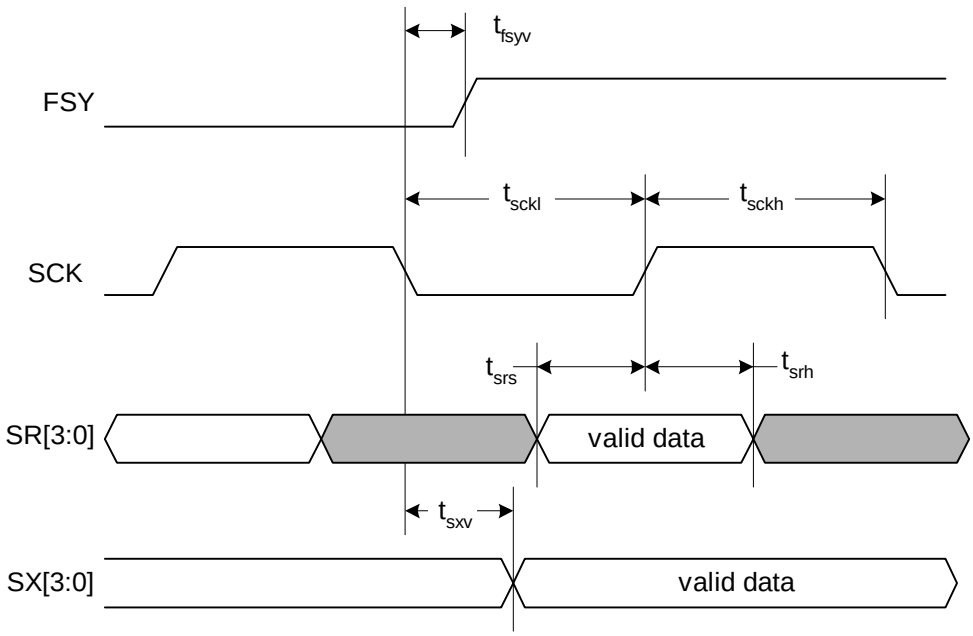


Figure 18-7: Source Port Internal Timing

18.5.6 Control Port in SPI Mode

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF.

Parameter	Symbol	Min	Typ	Max	Unit
SCL frequency	f_{scl}			200	kHz
SCL low time	t_{scll}	1			µs
SCL high time	t_{sclh}	1			µs
SCL low to $\overline{\textbf{CS}}$ high	t_{sklcsh}	1			µs
$\overline{\textbf{CS}}$ low to **SCL** rising	t_{css}	1			µs
$\overline{\textbf{CS}}$ high time	t_{cht}	1			µs
SDIN valid to **SCL** rising	t_{sds}	500			ns
SDIN hold from **SCL** rising	t_{sdh}	500			ns
SCL fall to **SDOUT** valid	t_{sdv}			1	µs
$\overline{\textbf{CS}}$ low to **SDOUT** driven	t_{cdv}			500	ns
$\overline{\textbf{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}			1	µs

PLL unlocked, no data at **RX** input (Note 1).

Parameter	Symbol	Min	Typ	Max	Unit
SCL frequency	f_{scl}			140	kHz
SCL low time	t_{scll}	1.2			µs
SCL high time	t_{sclh}	1.2			µs
SCL low to $\overline{\textbf{CS}}$ high	t_{sklcsh}	1			µs
$\overline{\textbf{CS}}$ low to **SCL** rising	t_{css}	1.2			µs
$\overline{\textbf{CS}}$ high time	t_{cht}	1.2			µs
SDIN valid to **SCL** rising	t_{sds}	500			ns
SDIN hold from **SCL** rising	t_{sdh}	500			ns
SCL fall to **SDOUT** valid	t_{sdv}			2.9	µs
$\overline{\textbf{CS}}$ low to **SDOUT** driven	t_{cdv}			1.75	µs
$\overline{\textbf{CS}}$ high to **SDOUT** Hi-Z	t_{sdz}			1.75	µs

Note:
1. This table is only useful for systems that are not referenced to **RMCK**, which scales with the PLL.

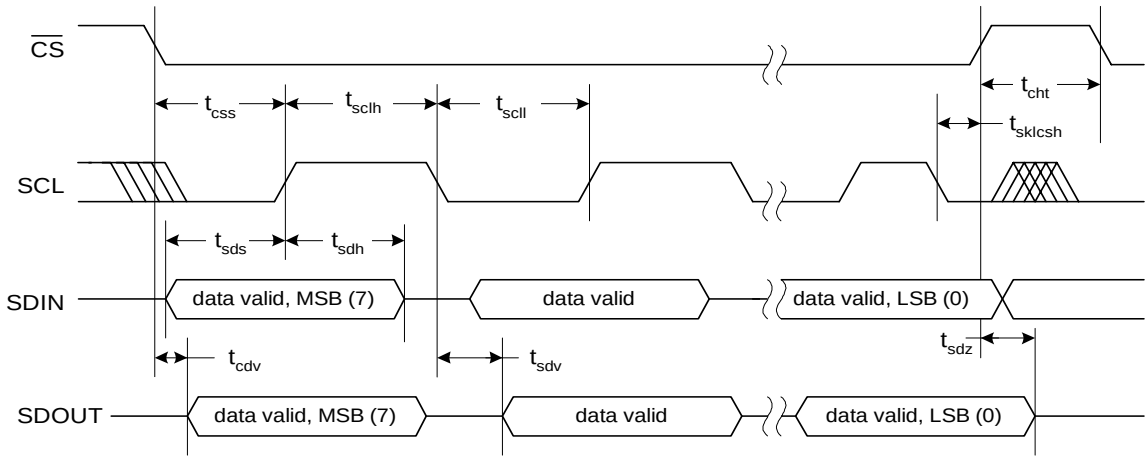


Figure 18-8: Control Port Timing in SPI Mode

18.5.7 Control Port in I²C Mode

T_A = –40 to 85 °C; VDDD, VDDA = 5 V ±10 %; GNDD,GNDA = 0.0 V, I²C Master (Note 1), PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF.

Parameter	Symbol	Min	Typ	Max	Unit
SCL frequency	f_{scl}			100	kHz
Bus free between transmissions (**SDA** and **SCL** high time between stop and start)	t_{buf}	4.7			µs
Start condition hold time (**SDA** falling to **SCL** falling)	t_{stah}	4			µs
SCL low time	t_{scll}	4.7			µs
SCL high time	t_{sclh}	4			µs
SCL falling to **SDA** input hold time	t_{sdah}	0			µs
SDA input setup time to **SCL** rising	t_{sdas}	250			ns
(repeated) start condition setup time	t_{stas}	4.7			µs
SDA and **SCL** rise time	t_{riic}			1	µs
SDA and **SCL** fall time	t_{fiic}			300	ns
Stop condition setup time (**SCL** rising to **SDA** rising)	t_{stps}	4			µs

Note:

1. The I²C Master format is only available when in Stand-Alone mode (see Chapter 17 on page 147).

I²C Slave, PLL locked at Fs = 44.1 kHz, Load Capacitance = 40 pF.

Parameter		Symbol	Min	Typ	Max	Unit
SCL frequency	(Notes 1, 2)	f_{scl}			400	kHz
Bus free between transmissions (**SDA** and **SCL** high time between stop and start)		t_{buf}	1.3			µs
Start condition hold time (**SDA** falling to **SCL** falling)		t_{stah}	0.6			µs
SCL low time	(Note 2)	t_{scll}	1.3			µs
SCL high time	(Note 2)	t_{sclh}	0.6			µs
SCL falling to **SDA** input hold time		t_{sdah}	0		0.9	µs
SDA input setup time to **SCL** rising	(Note 2)	t_{sdas}	100			ns
(repeated) start condition setup time		t_{stas}	0.6			µs
SDA and **SCL** rise time		t_{riic}			300	ns
SDA and **SCL** fall time		t_{fiic}			300	ns
Stop condition setup time (**SCL** rising to **SDA** rising)		t_{stps}	0.6			µs

Notes:

1. Above 200 kHz, clock stretching may occur.
2. When the PLL is unlocked at its lowest frequency, t_{sclh} is 1.18 µs minimum, t_{scll} is 2.15 µs minimum, t_{sdas} is 400 ns minimum, and t_{stah} is 0.8 µs. This data is only useful for systems that are not referenced to **RMCK** (which scales with the PLL frequency).

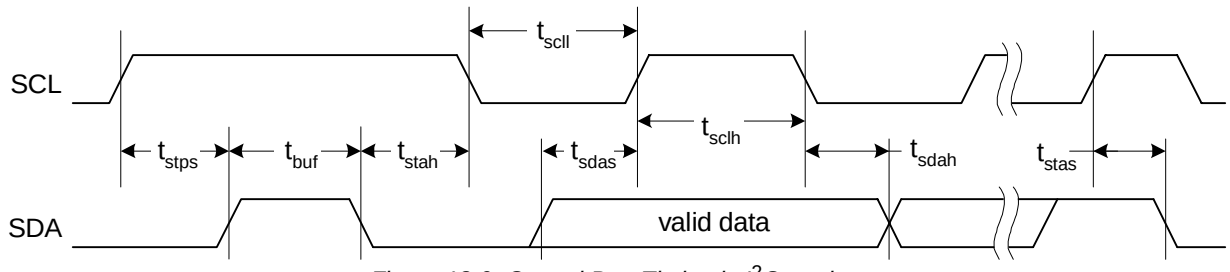


Figure 18-9: Control Port Timing in I²C mode.

19 Packaging and Pinout

All inputs must not be left floating; therefore the input pins must be driven, pulled up or down, or tied to one of the power pins.

19.1 Pinout List

Pin	Serial Name	Parallel Name	Type	Pin Description
1	**PAR_SRC**	**PAR_SRC**	D_{IN}	Parallel Source Data Port select input.
2	Tie to **GNDD**	**PAD0**	D_{IN}	Parallel address select input bit 0.
3	**TX**	**TX**	D_{OUT}	MOST transmitter output
4	**RMCK**	**RMCK**	D_{OUTZ}	Recovered master clock output
5	**VDDD**	**VDDD**		Digital power supply
6	**GNDD**	**GNDD**		Digital ground
7	Tie to **VDDD**	$\overline{\textbf{RD}}$	D_{IN}	Parallel mode read enable input.
8	Tie to **VDDD**	$\overline{\textbf{WR}}$	D_{IN}	Parallel mode write enable input.
9	**SX0**	**D0**	D_{OUT} $D_{I/O}$	Source Port data output 0 Bit 0 of parallel data I/O
10	**SX2**	**D1**	D_{OUT} $D_{I/O}$	Source Port data output 2 Bit 1 of parallel data I/O
11	Tie to **GNDD**	**PAD1**	D_{IN}	Parallel address select input bit 1.
12	**SX3**	**D2**	D_{OUT} $D_{I/O}$	Source Port data output 3 Bit 2 of parallel data I/O
13	**FSY**	**FSY**	$D_{I/O}$	Frame sync I/O
14	**SCK***	**SRC_FLOW**	$D_{I/O}$ D_{OUT}	Serial bit clock I/O Parallel data FIFO flow control output
15	**SR0**	**D3**	D_{IN} $D_{I/O}$	Source Port data input 0 Bit 3 of parallel data I/O
16	**SR2**	**D4**	D_{IN} $D_{I/O}$	Source Port data input 2 Bit 4 of parallel data I/O
17	**SR3**	**D5**	D_{IN} $D_{I/O}$	Source Port data input 3 Bit 5 of parallel data I/O
18	**VDDP**	**VDDP**		Periphery power supply
19	**GNDP**	**GNDP**		Periphery ground
20	**XTI**	**XTI**	CMOS IN	Crystal oscillator input
21	**XTO**	**XTO**	CMOS I/O	Crystal oscillator output.
22	$\overline{\textbf{WAKE\_UP}}$	$\overline{\textbf{WAKE\_UP}}$	D_{OODP}	Zero-Power mode wake-up time control
23	**STATUS**	**STATUS**	D_{OUT}	Power mode status.
24	**R_TIMER**	**R_TIMER**	A	Wake-up timer resistor input
25	**VREF**	**VREF**	A	Voltage reference buffer
26	**FLT**	**FLT**	A	PLL loop filter input
27	**GNDA**	**GNDA**		Analog ground
28	**GNDA**	**GNDA**		Analog ground

* Pull-down resistor recommended if **SCK** is not driven immediately upon exiting reset.

† Requires pull-up resistor to VDDD for I^2C mode.

†† Requires pull-up resistor to VDDD for all modes.

Table 19-1: Pinout List

Pin	Serial Name	Parallel Name	Type	Pin Description
29	**VDDA**	**VDDA**		Analog power supply
30	**VDDA**	**VDDA**		Analog power supply
31	**$\overline{\text{RS}}$**	**$\overline{\text{RS}}$**	D_{IN}	Hardware reset input
32	**$\overline{\text{CS}}$** **AD1**	Pull up to **VDDD**	D_{IN}	Chip select input for SPI mode Address bit 1 input for I^2C mode
33	Do not connect	**CP_FLOW**	D_{OUT}	Control port flow control indicator output
34	**ASYNC**	**ASYNC**	D_{IN}	Asyncronous mode configuration pin.
35	**SR1**	**D6**	D_{IN} $D_{I/O}$	Source Port data input 1 Bit 6 of parallel data I/O
36	**SX1**	**D7**	D_{IN} $D_{I/O}$	Source Port data output 1 Bit 7 of parallel data I/O
37	**ERROR**	**ERROR**	D_{OUT}	Error indicator output
38	**SDIN** **AD0**	Pull up to **VDDD**	D_{IN}	Serial data input in SPI mode Address bit 0 input for I^2C mode
39	**SCL**†	Pull up to **VDDD**	$D_{I/O+CONF}$	I^2C/SPI clock
40	**SDA**† **SDOUT**	Pull up to **VDDD**	$D_{I/O+CONF}$ D_{OUT}	I^2C data I/O SPI data out
41	**$\overline{\text{INT}}$**††	**$\overline{\text{INT}}$**††	$D_{I/O + OD}$	Control message and power-on interrupt pin
42	**RX**	**RX**	D_{IN}	MOST receiver input
43	**$\overline{\text{AINT}}$**††	**$\overline{\text{AINT}}$**††	D_{OOD}	Asynchronous message interrupt
44	**PAR_CP**	**PAR_CP**	D_{IN}	Parallel Control Port select input.

* Pull-down resistor recommended if **SCK** is not driven immediately upon exiting reset.

† Requires pull-up resistor to VDDD for I^2C mode.

†† Requires pull-up resistor to VDDD for all modes.

Table 19-1: Pinout List (Continued)

19.2 Low-Power/Zero-Power Mode Pin State

Pin	Serial Name	Parallel Name	Low-Power	Zero-Power
1	**PAR_SRC**	**PAR_SRC**	input active	input active
2	Tie to **GNDD**	**PAD0**	input inactive	input inactive
3	**TX**	**TX**	output active	output, driven to **RX** value
4	**RMCK**	**RMCK**	output active	output inactive (Hi-Z, should be pulled high or low)
5	**VDDD**	**VDDD**	---	---
6	**GNDD**	**GNDD**	---	---
7	Tie to **VDDD**	**$\overline{\text{RD}}$**	input active (Note 2)	input active (Note 2)
8	Tie to **VDDD**	**$\overline{\text{WR}}$**	input active (Notes 1 and 2)	input active (Note 1)
9	**SX0**	**D0**	output, driven low IO (Note 5)	output, driven low IO (Note 5)
10	**SX2**	**D1**	output, driven low IO (Note 5)	output, driven low IO (Note 5)
11	Tie to **GNDD**	**PAD1**	input inactive	input inactive
12	**SX3**	**D2**	output, driven low IO (Note 5)	output, driven low IO (Note 5)
13	**FSY**	**FSY**	input inactive	input inactive
14	**SCK**	**SRC_FLOW**	output, driven low	output, driven low
15	**SR0**	**D3**	input inactive IO (Note 5)	input inactive IO (Note 5)
16	**SR2**	**D4**	input inactive IO (Note 5)	input inactive IO (Note 5)
17	**SR3**	**D5**	input inactive IO (Note 5)	input inactive IO (Note 5)
18	**VDDP**	**VDDP**	---	---
19	**GNDP**	**GNDP**	---	---
20	**XTI**	**XTI**	normally inactive (Note 3)	inactive
21	**XTO**	**XTO**	normally inactive (Note 3)	inactive
22	**$\overline{\text{WAKE\_UP}}$**	**$\overline{\text{WAKE\_UP}}$**	output, driven low	output strobing
23	**STATUS**	**STATUS**	output, driven low	output, driven high
24	**R_TIMER**	**R_TIMER**	---	---
25	**VREF**	**VREF**	---	---
26	**FLT**	**FLT**	---	---
27	**GNDA**	**GNDA**	---	---
28	**GNDA**	**GNDA**	---	---
29	**VDDA**	**VDDA**	---	---
30	**VDDA**	**VDDA**	---	---
31	**$\overline{\text{RS}}$**	**$\overline{\text{RS}}$**	input active	input active

Notes:

1. These signals can be used to wake up the chip in Low-Power or Zero-Power mode. **$\overline{\text{CS}}$** can only wake up in SPI mode, **SDA** only in I^2C mode, **$\overline{\text{WR}}$** can only wake up in parallel mode.
2. In parallel mode, status registers can be read, but normal functionality disabled.
3. In timing-slave mode, generally disabled (clock recovered from **RX**). In timing-master mode, active.
4. Based on bXSR settings, this pin can reflect bXCR lock status.
5. **D[7:0]** are Hi-Z when **$\overline{\text{RD}}$** is high and driven when **$\overline{\text{RD}}$** is low.

Table 19-2: Low-Power and Zero-Power Pin State

Pin	Serial Name	Parallel Name	Low-Power	Zero-Power
32	$\overline{\mathbf{CS}}$ **AD1**	$\overline{\mathbf{CS}}$ **AD1**	input active (Note 1) input inactive	input active (Note 1) input inactive
33	Do not connect	**CP_FLOW**	output keeps last value	output 0
34	**ASYNC**	**ASYNC**	input active	input active
35	**SR1**	 **D6**	input inactive IO (Note 5)	input inactive IO (Note 5)
36	**SX1**	 **D7**	output, driven low IO (Note 5)	output, driven low IO (Note 5)
37	**ERROR**	**ERROR**	output active (Note 4)	output, driven high
38	**SDIN** **AD0**	**SDIN** **AD0**	input inactive	input inactive
39	**SCL**	**SCL**	input active	input active
40	**SDA** **SDOUT**	**SDA** **SDOUT**	input active (Note 1) Hi-Z	input active (Note 1) Hi-Z
41	$\overline{\mathbf{INT}}$	$\overline{\mathbf{INT}}$	keeps last value	Hi-Z (pulled up by ext. resistor)
42	**RX**	**RX**	input inactive	input inactive
43	$\overline{\mathbf{AINT}}$	$\overline{\mathbf{AINT}}$	Hi-Z (pulled up by ext. resistor)	Hi-Z (pulled up by ext. resistor)
44	**PAR_CP**	**PAR_CP**	input active	input active

Notes:

1. These signals can be used to wake up the chip in Low-Power or Zero-Power mode. $\overline{\mathbf{CS}}$ can only wake up in SPI mode, **SDA** only in I^2C mode, $\overline{\mathbf{WR}}$ can only wake up in parallel mode.
2. In parallel mode, status registers can be read, but normal functionality disabled.
3. In timing-slave mode, generally disabled (clock recovered from **RX**). In timing-master mode, active.
4. Based on bXSR settings, this pin can reflect bXCR lock status.
5. **D[7:0]** are Hi-Z when $\overline{\mathbf{RD}}$ is high and driven when $\overline{\mathbf{RD}}$ is low.

Table 19-2: Low-Power and Zero-Power Pin State (Continued)

19.3 Reset Pin State

Pin	Serial Name	Parallel Name	State during Reset ($\overline{RS}$ active)
1	**PAR_SRC**	**PAR_SRC**	input
2	Tie to **GNDD**	**PAD0**	input
3	**TX**	**TX**	output, driven to value on **RX**
4	**RMCK**	**RMCK**	output, driven low
5	**VDDD**	**VDDD**	---
6	**GNDD**	**GNDD**	---
7	Tie to **VDDD**	$\overline{\textbf{RD}}$	input
8	Tie to **VDDD**	$\overline{\textbf{WR}}$	input
9	**SX0**	**D0**	Hi-Z
10	**SX2**	**D1**	Hi-Z
11	Tie to **GNDD**	**PAD1**	input
12	**SX3**	**D2**	Hi-Z
13	**FSY**	**FSY**	Hi-Z output, driven low
14	**SCK**	**SRC_FLOW**	Hi-Z output, driven opposite polarity to **ASYNC**
15	**SR0**	**D3**	Hi-Z
16	**SR2**	**D4**	Hi-Z
17	**SR3**	**D5**	Hi-Z
18	**VDDP**	**VDDP**	---
19	**GNDP**	**GNDP**	---
20	**XTI**	**XTI**	input
21	**XTO**	**XTO**	output, driven to inverse of **XTI**
22	$\overline{\textbf{WAKE\_UP}}$	$\overline{\textbf{WAKE\_UP}}$	output, driven low
23	**STATUS**	**STATUS**	output, driven low
24	**R_TIMER**	**R_TIMER**	---
25	**VREF**	**VREF**	---
26	**FLT**	**FLT**	---
27	**GNDA**	**GNDA**	---
28	**GNDA**	**GNDA**	---
29	**VDDA**	**VDDA**	---
30	**VDDA**	**VDDA**	---
31	$\overline{\textbf{RS}}$	$\overline{\textbf{RS}}$	input, must be driven low externally
32	$\overline{\textbf{CS}}$ **AD1**	$\overline{\textbf{CS}}$ **AD1**	input
33	Do not connect	**CP_FLOW**	output, driven high (Note 1)
34	Tie to **GNDD**	**ASYNC**	input
35	**SR1**	**D6**	Hi-Z
36	**SX1**	**D7**	Hi-Z
37	**ERROR**	**ERROR**	output, driven high
38	**SDIN** **AD0**	**SDIN** **AD0**	input
39	**SCL**	**SCL**	Hi-Z

Note:

1. Revisions B and C drove **CP_FLOW** low in the reset state.

Table 19-3: Pin Reset Value

Pin	Serial Name	Parallel Name	State during Reset ($\overline{RS}$ active)
40	**SDA** **SDOUT**	**SDA** **SDOUT**	Hi-Z
41	**$\overline{INT}$**	**$\overline{INT}$**	Hi-Z
42	**RX**	**RX**	input
43	**$\overline{AINT}$**	**$\overline{AINT}$**	Hi-Z
44	**PAR_CP**	**PAR_CP**	input

Note:

1. Revisions B and C drove **CP_FLOW** low in the reset state.

Table 19-3: Pin Reset Value (Continued)

19.4 Equivalent Schematics For Pins

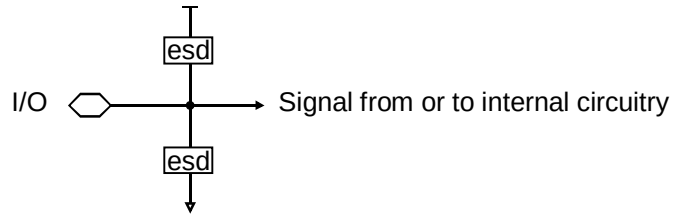


Figure 19-1: Pin Equivalent for Analog Input Pins and Output Pins (A)

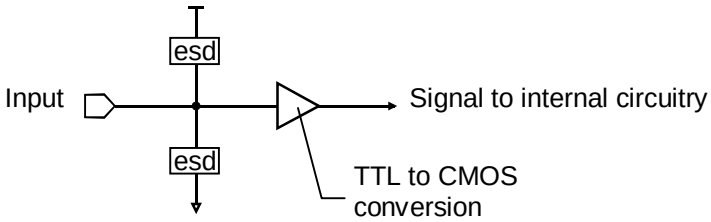


Figure 19-2: Pin Equivalent for Digital Input Pin (D_{IN})

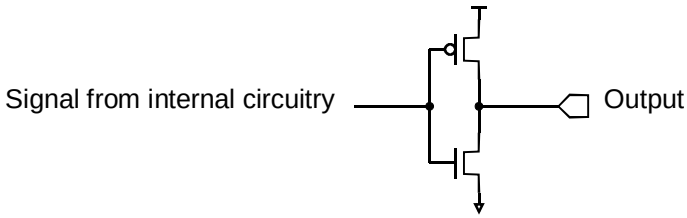


Figure 19-3: Pin Equivalent for Digital Output Pin (D_{OUT})

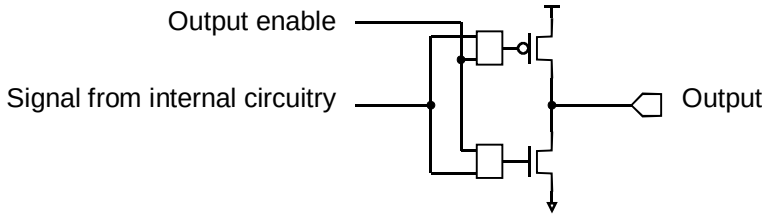


Figure 19-4: Pin Equivalent for Digital Output Pin with Tri-State (D_{OUTZ})

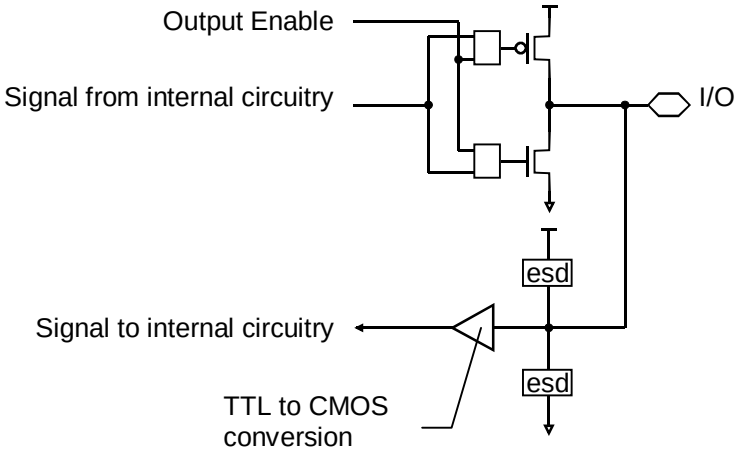


Figure 19-5: Pin Equivalent for Bi-directional Digital Output Pin ($D_{I/O}$)

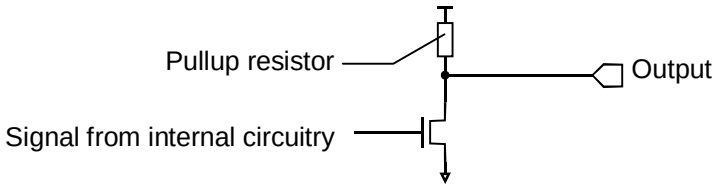


Figure 19-6: Pin Equivalent for Digital Open-Drain Output Pin with Internal Pull-up (D_{OODP})

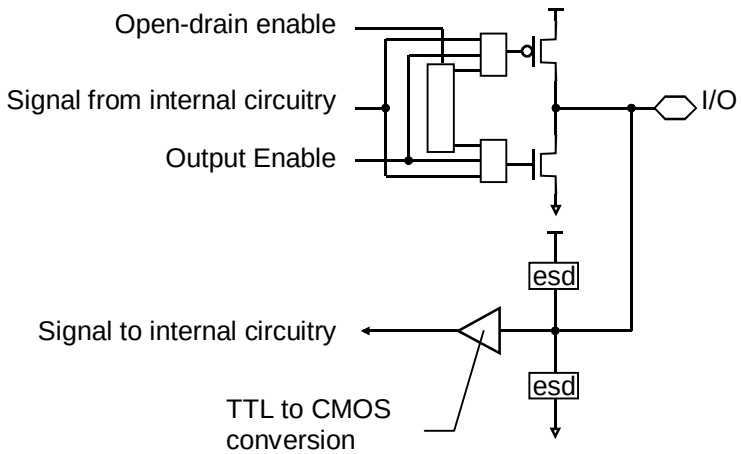


Figure 19-7: Pin Equivalent for Bi-directional Digital and Configurable Pin ($D_{I/O+CONF}$)

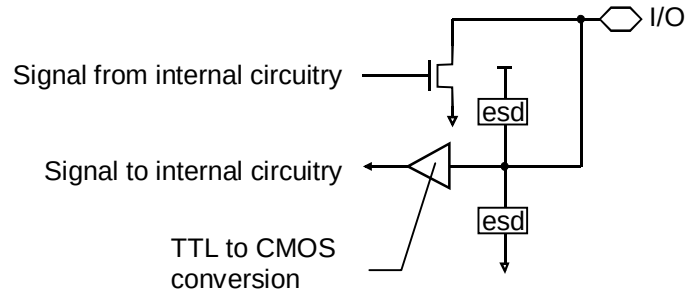


Figure 19-8: Pin Equivalent for Bi-directional Digital Pin with Open-Drain ($D_{I/O+OD}$)

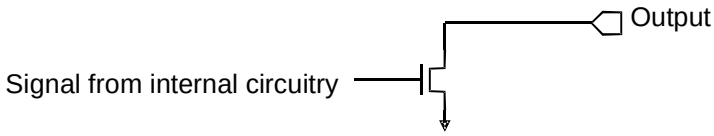


Figure 19-9: Pin Equivalent for Digital Open-Drain Output Pin (D_{OOD})

19.5 Block Diagram

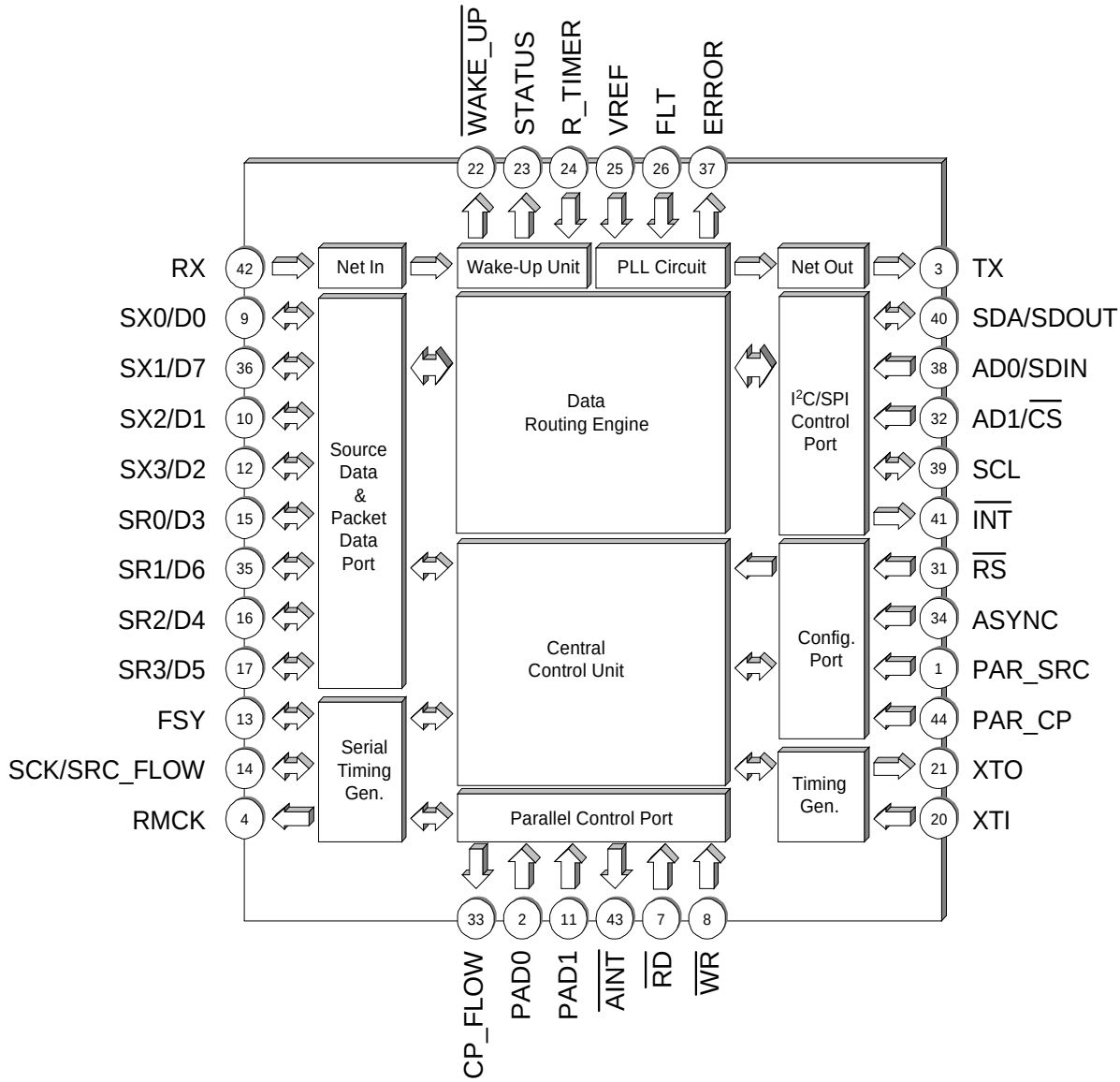


Figure 19-10: Pinout Block Diagram

19.6 Pinout

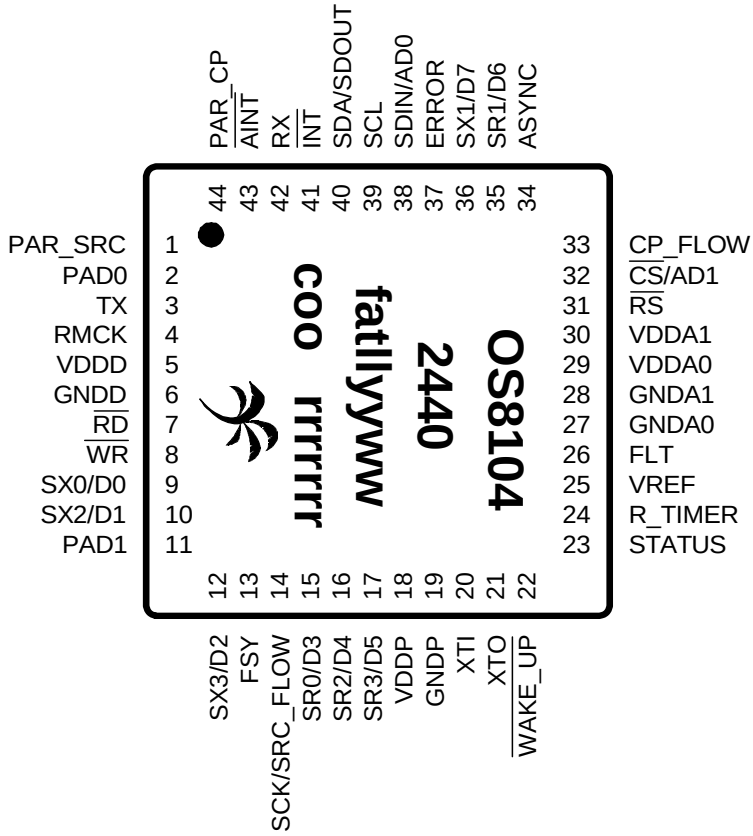


Figure 19-11: Pinout

The package designators are:

- f - Fab
- a - Assembly Site
- t -Test Site
- ll - Lot Sequence Code
- yy - last two digits of assembly Year. Revision A and revision B devices have only one character for assembly Year (last digit of year).
- ww - Assembly Work Week

- coo - Country of Origin. MAL = Malta
- rrrrrrr - Chip Revision Field

rrrrrrr	Revision
TEGEDAA	A
TEGEDAB	B
TEGFCAA	C
TEGFCBP	D

Table 19-4: Package Marking vs. Revisions

19.7 Package Outline (TQFP 44)

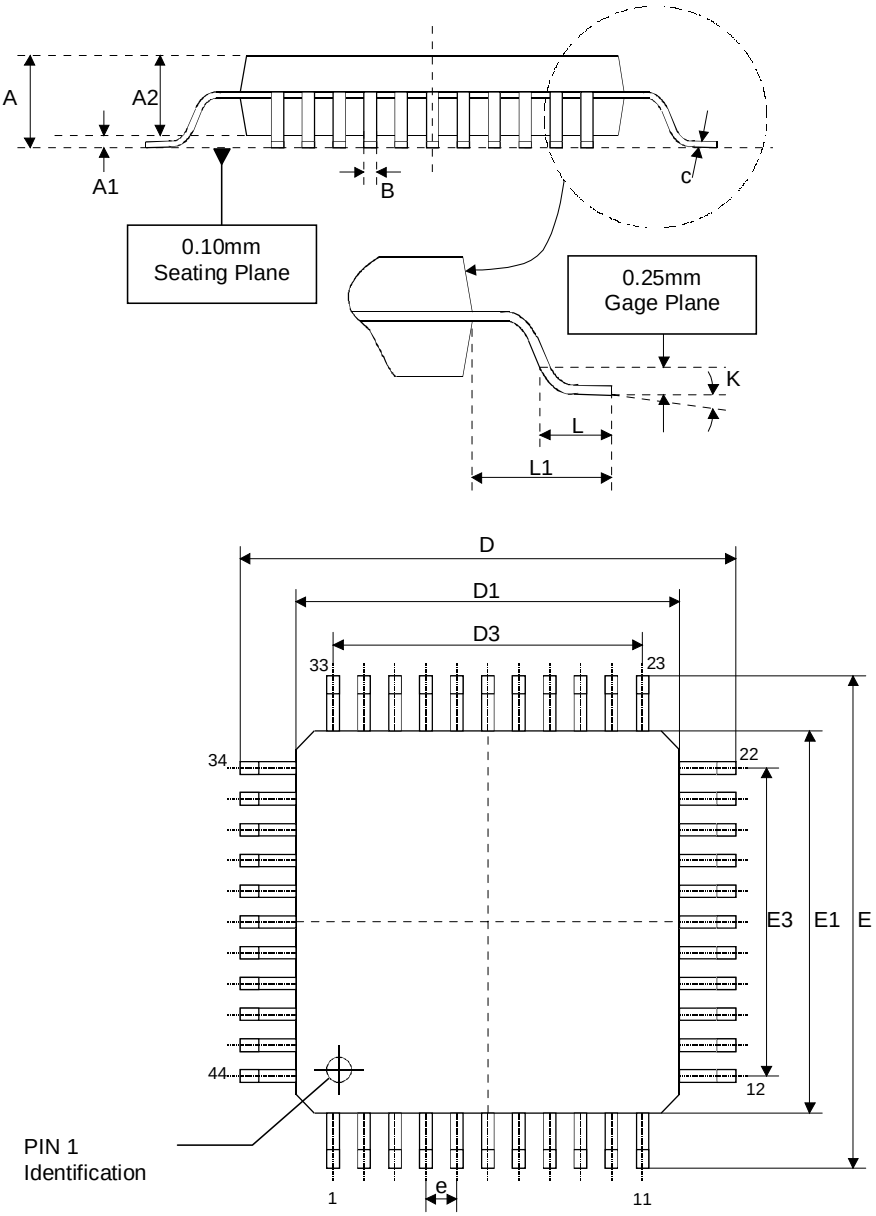


Figure 19-12: Package Outline

	A	A1	A2	B	c	D	D1	D3	e	E	E1	E3	L	L1	K
Min		0.05	1.35	0.30	0.09	11.80	9.80			11.80	9.80		0.45		0°
Typ			1.40	0.37		12.00	10.00	8.00	0.80	12.00	10.00	8.00	0.60	1.00	3.5°
Max	1.60	0.15	1.45	0.45	0.20	12.20	10.20			12.20	10.20		0.75		7°

Table 19-5: Package Outline Dimensions in mm and degrees

20 Application Information

As a general rule, all traces should be kept as short as possible to avoid capacitive loads at driver outputs. Placing any capacitance directly on the output pin should be avoided as the capacitor increases the output drive current, thereby increasing EMI radiation. All components should be placed as close to the OS8104 as possible. The most critical signals with respect to EMI are:

- **RMCK**
- **TX**
- **RX**
- **SX[3:0]**, **SR[3:0]**
- **SCK**

The reduction of EMI is most effective if the current is minimized at the signals' source. Efficient ways of lowering radiation include the use of a series-termination resistor to impedance-match the trace (and lengthen rise/fall times) and the use of ferrite beads to remove some of the higher-frequency energy.

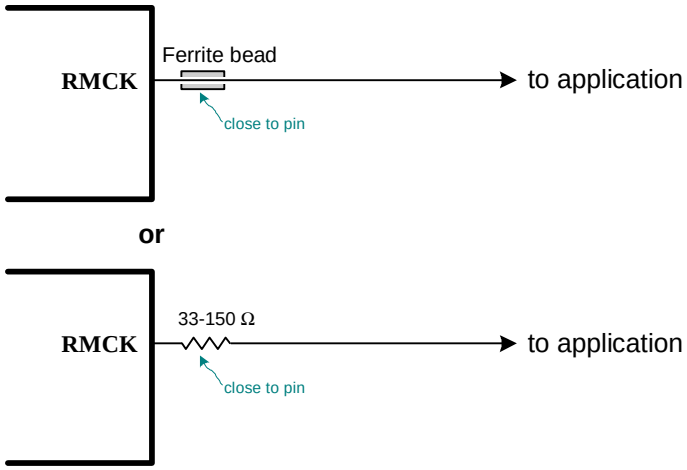


Figure 20-1: EMI-Reducing Circuits

The series-termination resistor or ferrite bead must be placed as close as possible to the output pin to minimize output pin capacitance and unterminated signal energy. The value of the resistor will depend directly on the application (capacitive load, slew rate, trace impedance, etc.); however, values will generally range from 33 Ω to 150 Ω.

20.1 Crystal Oscillator Selection

Several factors must be considered when selecting a crystal. These include load capacitance, oscillator margin, cut, and operating temperature.

Oscillator margin is a measure of the stability of an oscillator circuit, and is defined as the ratio of the oscillator's negative resistance (R_{NEG}) to the crystal's ESR (R_{ESR}), or:

$$Margin = \frac{|R_{NEG}|}{R_{ESR}} = \frac{|R_{VAR}| + R_{ESR}}{R_{ESR}}$$

The negative resistance can be measured by placing a variable resistor (R_{VAR}) in series with the crystal and finding the largest resistor value where the crystal still starts up properly. This point would be just below where the oscillator does not start-up or where the start-up time is excessively long.

Ideally oscillator margin should be greater than or equal to 10, and should be at least 5. Smaller oscillator margin can affect the ability of the oscillator to start up.

The load capacitor, specified when ordering the crystal, is the series combination of the capacitance on each leg of the crystal. This capacitance includes not only the added capacitors, but also PCB trace (shunt) capacitance and chip pin capacitance. Larger capacitors also have a negative affect on oscillator margin. In general, the external capacitors on each leg (C1 and C2) should be in the 12 to 22 pF range.

Name	Value	Description
Correlation	Parallel Resonant	Mode of oscillation
Osc. Mode	Fundamental	Oscillation mode or Operation mode.
C_L	16-20 pF	Recommended Load Capacitance.
ESR	40 Ω 20 Ω	Recommended Maximum Equivalent Series Resistance: When crystal frequency is 256Fs or 384Fs When crystal frequency is 512Fs
Drive Level	50 µW	Typical Drive Level
T_A	-40 to 85 °C	Operating temperature range
cut	AT	AT cut produces the best temperature stability.
Tolerance	±50 ppm	Frequency tolerance at 25 °C. Typical value.

Table 20-1: Crystal Oscillator Specifications

The crystal cut and tolerance value listed in Table 20-1 are typical values and may be changed to suit differing system requirements. Higher ESR values, than those listed in the Table, run the risk of having start-up problems and should be thoroughly tested before being used. Contact the crystal manufacturer for more information.

20.2 Power Supplies and Analog Components

Figure 20-2 illustrates the standard power arrangement for the OS8104. The 0.1 µF (±10%) capacitors must be of ceramic type (except for the **FLT** capacitor) and should be placed as close as possible to the OS8104.

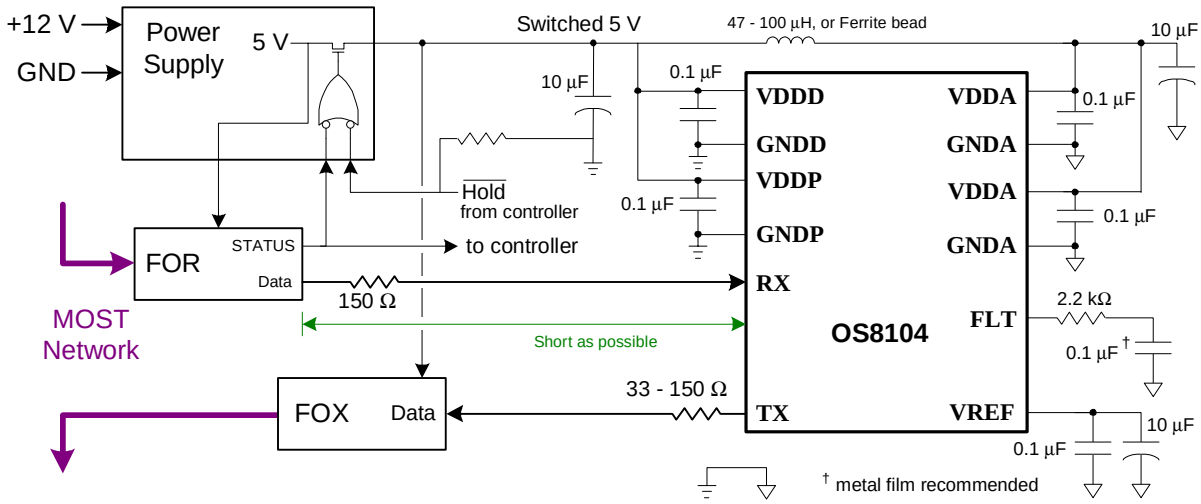


Figure 20-2: Typical Power Supply Connection Diagram

The **FLT** and **VREF** analog components should be placed as close as possible to the **GNDA** pins to minimize loop currents. To minimize vibration and shock effects on PLL locking, a metal-film type capacitor, such as Panasonic ECP-U1C104MA5, is recommended for the **FLT** pin, since ceramic capacitors are more sensitive to shock and could cause unlock events in high-vibration environments. The **FLT** pin is a high-impedance node; therefore, leakage should be kept below 1 µA or average pulse-width distortion tolerance could be adversely affected. Conformal coating is recommended for systems where condensation can occur.

The analog and digital grounds, if separated, should be tied together at only one point on the board.

The distance between the fiber-optic receiver (FOR) unit and the OS8104 should be as short as possible to minimize capacitance on the Data line. This will shorten transition times out of the FOR thereby minimizing pulse width distortion and jitter.

The series resistor in the **TX** path for the FOX is for series-termination and should be as close as possible to the OS8104. The chosen value is based on the board layout and should match the line impedance. This resistor will help minimize reflections and lower EMI.

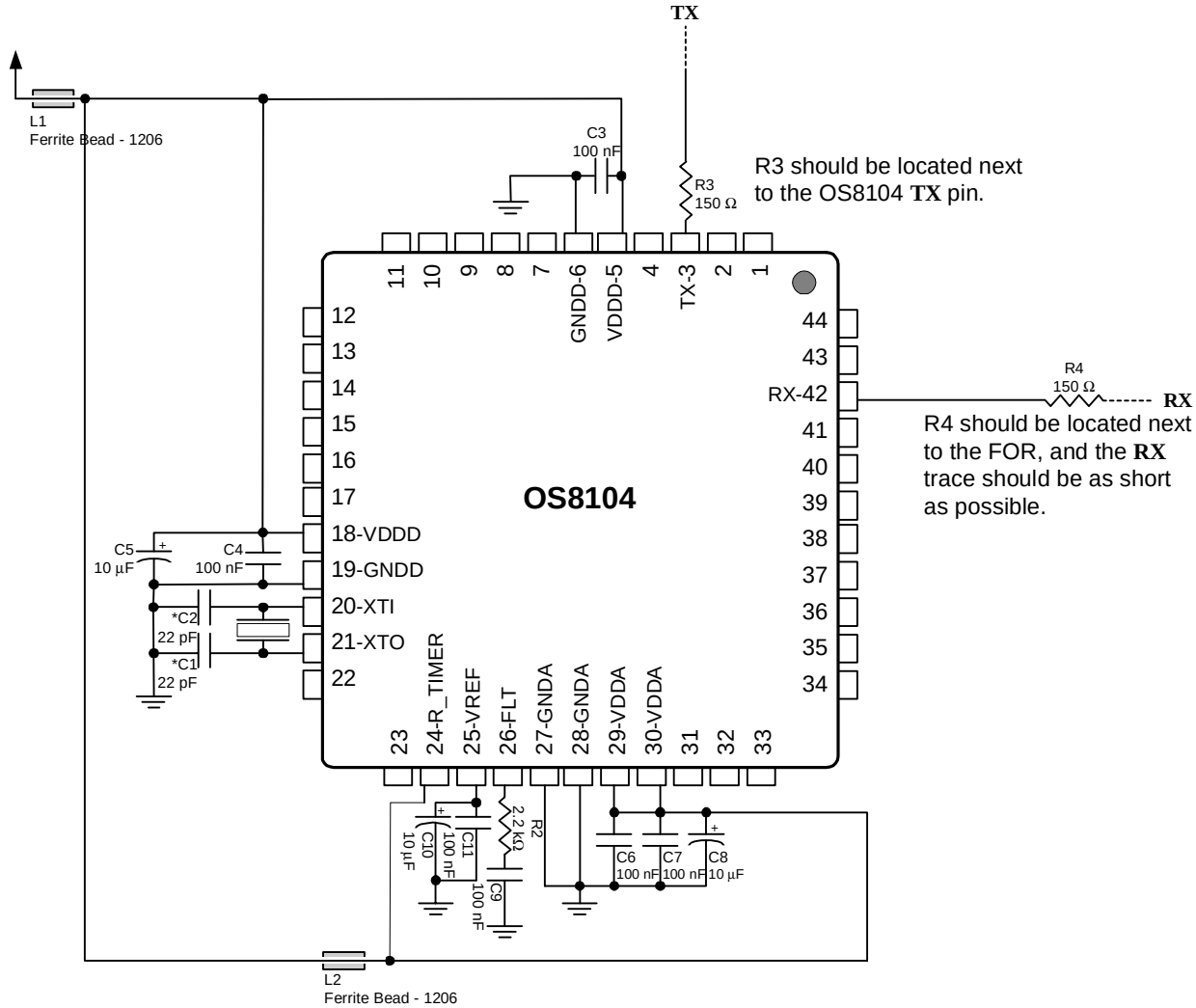
In the example above, the FOR controls the power to the node. All power supplies, except the FOR are on the *Switched 5 V* circuit. When there is no light at the FOR input, the STATUS output turns off the switched power supply to minimize the power drain. The external controller has the option to delay power-down to allow an orderly shut-down of the local node. The controller can apply power control when normal operation has commenced. When the FOR loses light, STATUS goes high telling the controller to perform a shut-down. When the controller has finished the orderly shut-down, it releases power control causing the node to go into full power-down. Having the FOR control the node's power complies with the *MOST Specification*. The circuit illustrated is a conceptual schematic and doesn't include the decoupling needed between powered-off and powered devices.

When light is initially detected at the FOR, the STATUS output turns on the *Switched 5 V* supply thereby powering up the node. The delay from STATUS low to power-up should be less than 50 ms. The series resistor on the Data output provides current limiting for the short period of time between the FOR driving data out and the Power Supply switch-on time.

20.3 Layout Guidelines

OS8104 board designs should follow the basic guidelines outlined throughout this chapter, with special consideration given to capacitor selection and placement. In order to reduce trace impedance, decoupling capacitors should be positioned close to the OS8104. In situations where large electrolytic and small ceramic capacitors are used, the smaller capacitor should be closest to OS8104 pins. The smaller capacitors should also be on the same layer to avoid the additional impedance of vias.

Figure 20-4 illustrates an OS8104 pin subset, with component choices that are typical for most OS8104 applications.



* C1 and C2 are typical values.
Actual value is determined by
PCB and crystal characteristics.

Figure 20-3: Typical Schematic

Figure 20-4 illustrates a recommended layout for the OS8104 schematic shown Figure 20-3, representing good component placement and routing for typical OS8104 applications. As mentioned above, the high frequency decoupling capacitors are placed closest to the OS8104 and on the same layer.

Ground-plane fill is used under the part to minimize the ground impedance between all the ground pins, which also helps minimize EMI. If the ground-plane fill must be broken by a trace, then vias should connect each set of pins to the ground plane on another layer. Vias that connect decoupling and high frequency capacitors to the ground plane should also be close to the OS8104 to minimize loop area back to the chip ground pins.

Signals between the chip and the FOT should be as short as possible, and should have an unbroken ground path between the two. The series-termination resistors should be close to the driving source: the OS8104's **TX** pin and the FOT **RX** Data pin. Ground-plane fill should not be too close to the **RX** data path, as it adds capacitance which can negatively impact PWD performance.

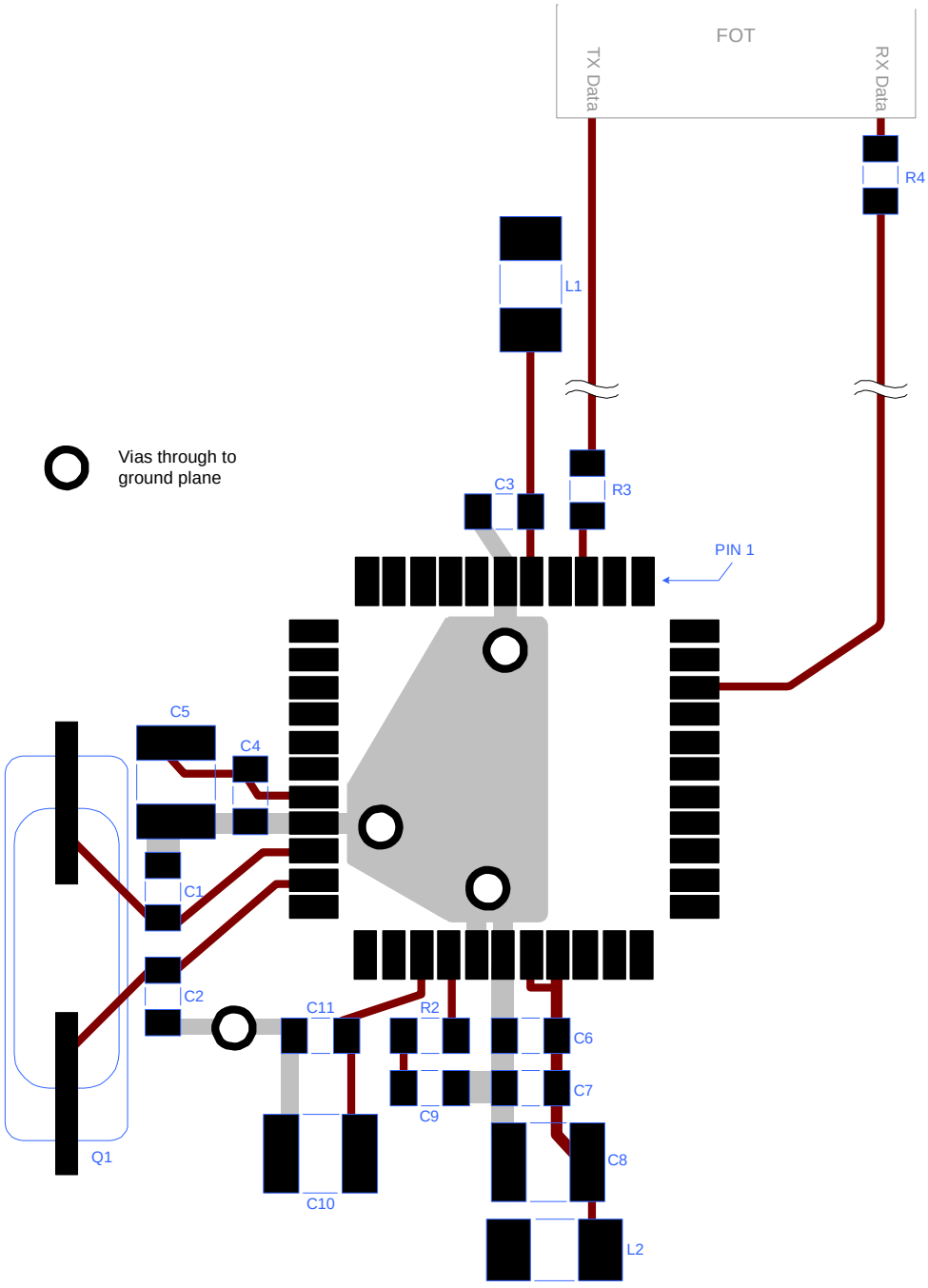


Figure 20-4: Typical Layout

20.4 Control Port in I²C Serial Mode

The following diagram assumes that the FOR manages power and the OS8104 power management modes are not used. The pull-up on **SCL** selects the I²C Control Port format at power-up.

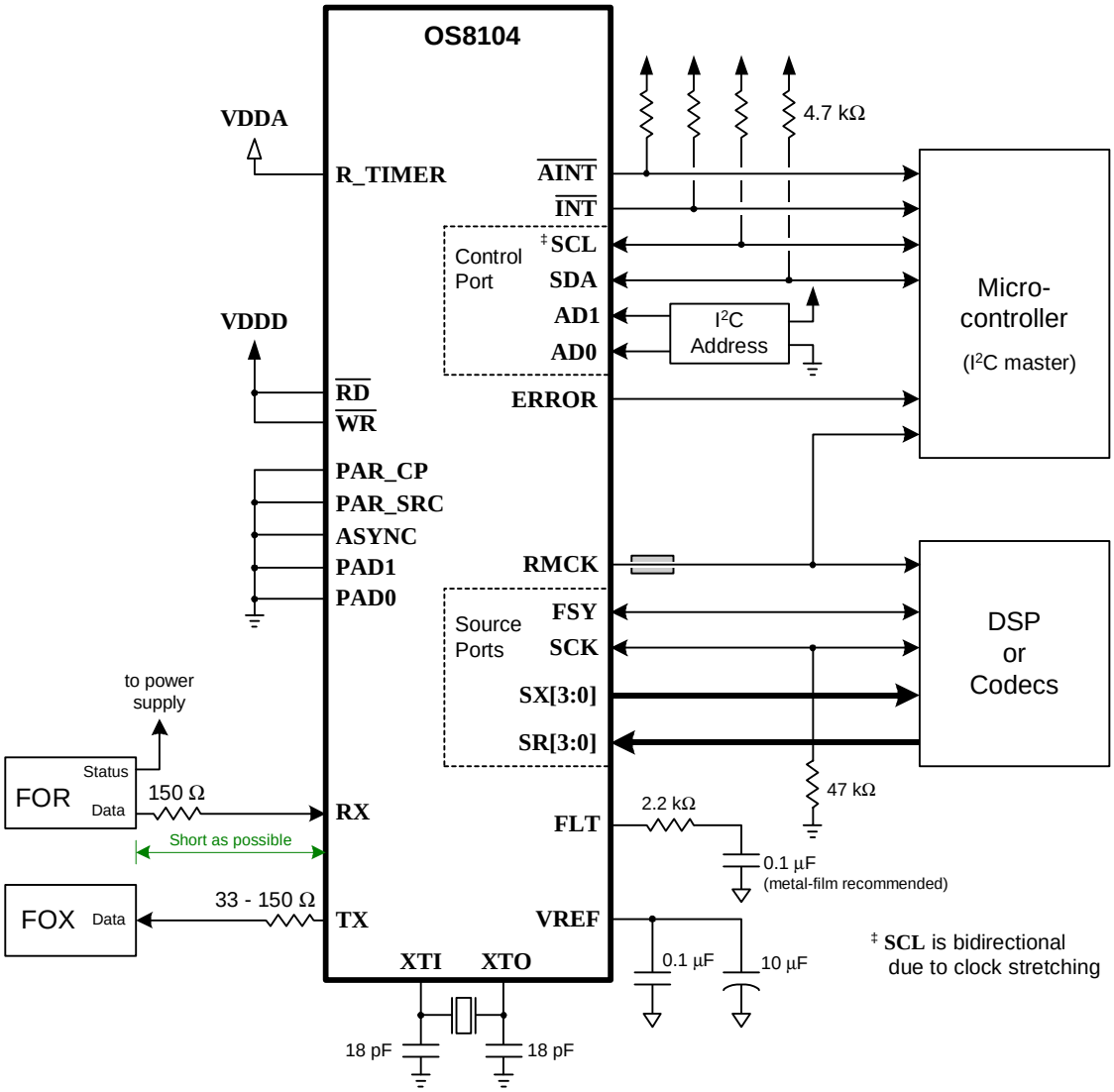


Figure 20-5: Typical Application — Source Ports in Serial Mode, Control Port in I²C Mode

20.5 Control Port in SPI Serial Mode

The following diagram assumes that the FOR manages power and the OS8104 power management modes are not used. The **SCL** pin must be low at power-up to select SPI mode for the Control Port. If **SCL** is not driven (low) at power-up, it should be connected to a weak pull-down resistor.

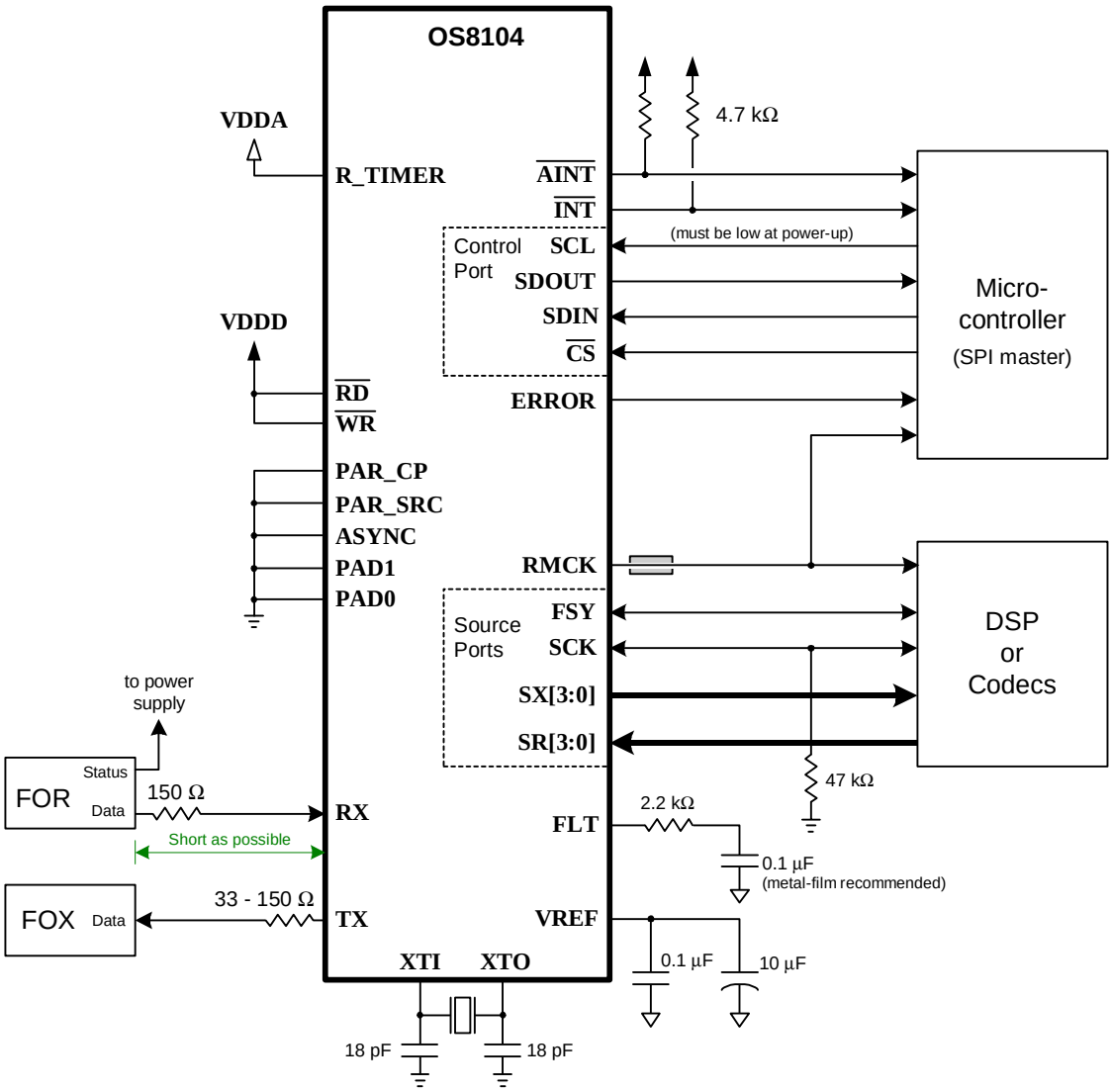


Figure 20-6: Typical Application — Source Ports in Serial Mode, Control Port in SPI Mode

20.6 Parallel Source Ports, I²C Serial Control Port

The following diagram assume that the FOR manages power and that the OS8104 power management modes are not used.

The **PAR_SRC** pin high selects parallel mode for the Source Ports. The **FSY** and **SRC_FLOW** signals are available as pins or in the parallel port status register. The **$\overline{RD}$** and **$\overline{WR}$** pins must be held high from power-up until the **$\overline{INT}$** pin goes low indicating the OS8104 is ready to be accessed. In addition, the value of **SCL** at power-up determines the Control Port's serial format.

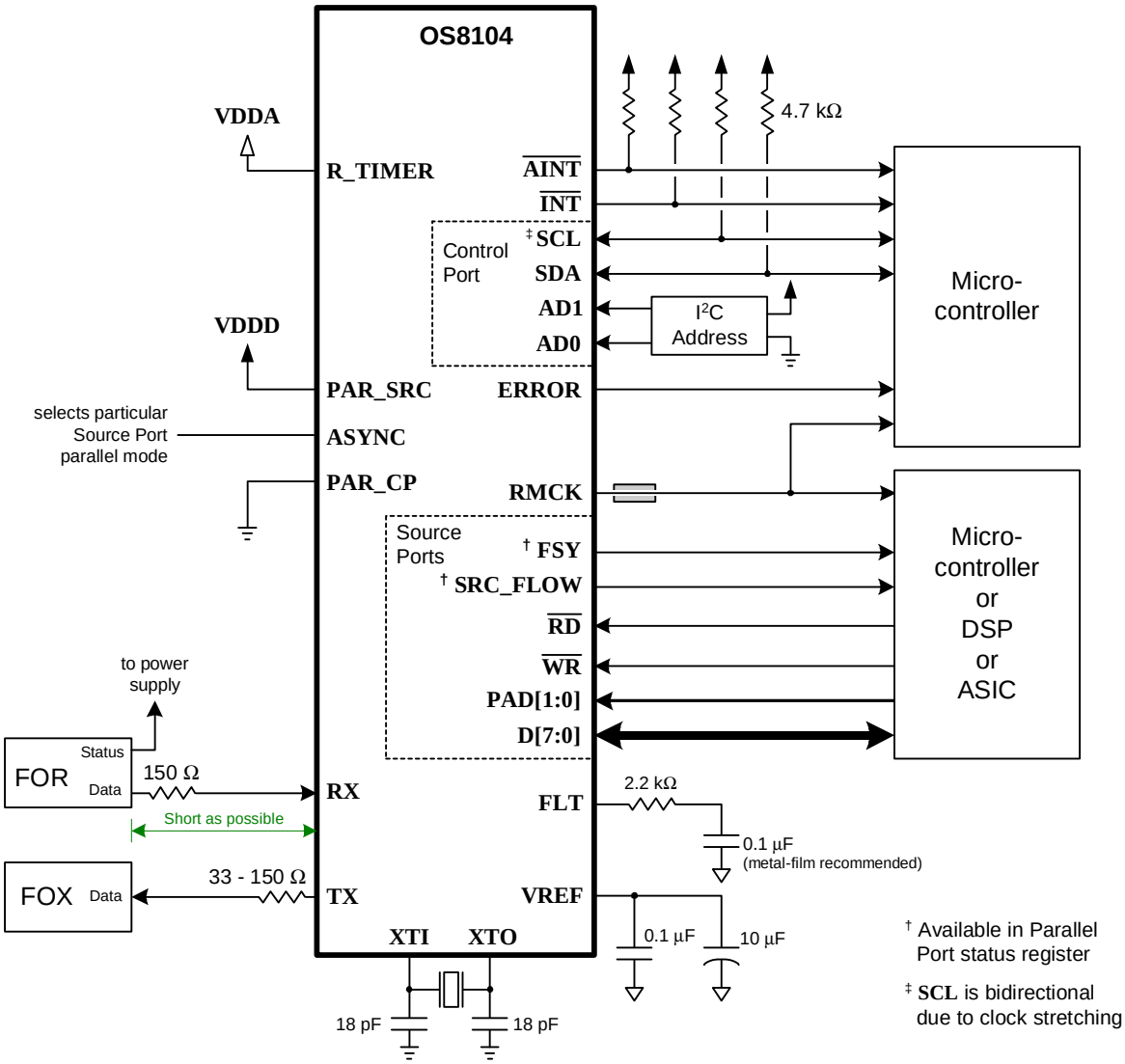


Figure 20-7: Typical Application — Source Ports in Parallel, Control Port in I²C Mode

20.7 Source Ports and Control Port in Parallel

The following diagram assume that the FOR manages power and that the OS8104 power management modes are not used.

The **PAR_SRC** pin high selects parallel mode for the Source Ports, and the **PAR_CP** pin high selects parallel mode for the Control Port. As previously mentioned, the Source Ports have to be in parallel mode to use the Control Port in parallel mode, since the Control Port uses the same data bus pins as the Source Ports. The **FSY**, **SRC_FLOW**, and **CP_FLOW** signals are available as pins or in the parallel port status register. **ASYNC** selects between Parallel-Asynchronous and Parallel-Synchronous/Parallel-Combined mode.

The $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins must be held high from power-up until the $\overline{\text{INT}}$ pin goes low indicating the OS8104 is ready to be accessed. The pull-up resistors on $\overline{\text{AINT}}$, $\overline{\text{INT}}$, **SCL** and **SDA** are required regardless of whether the pins are used. The **SCL** and **SDA** pins must be inactive when the Control Port is configured for parallel operation.

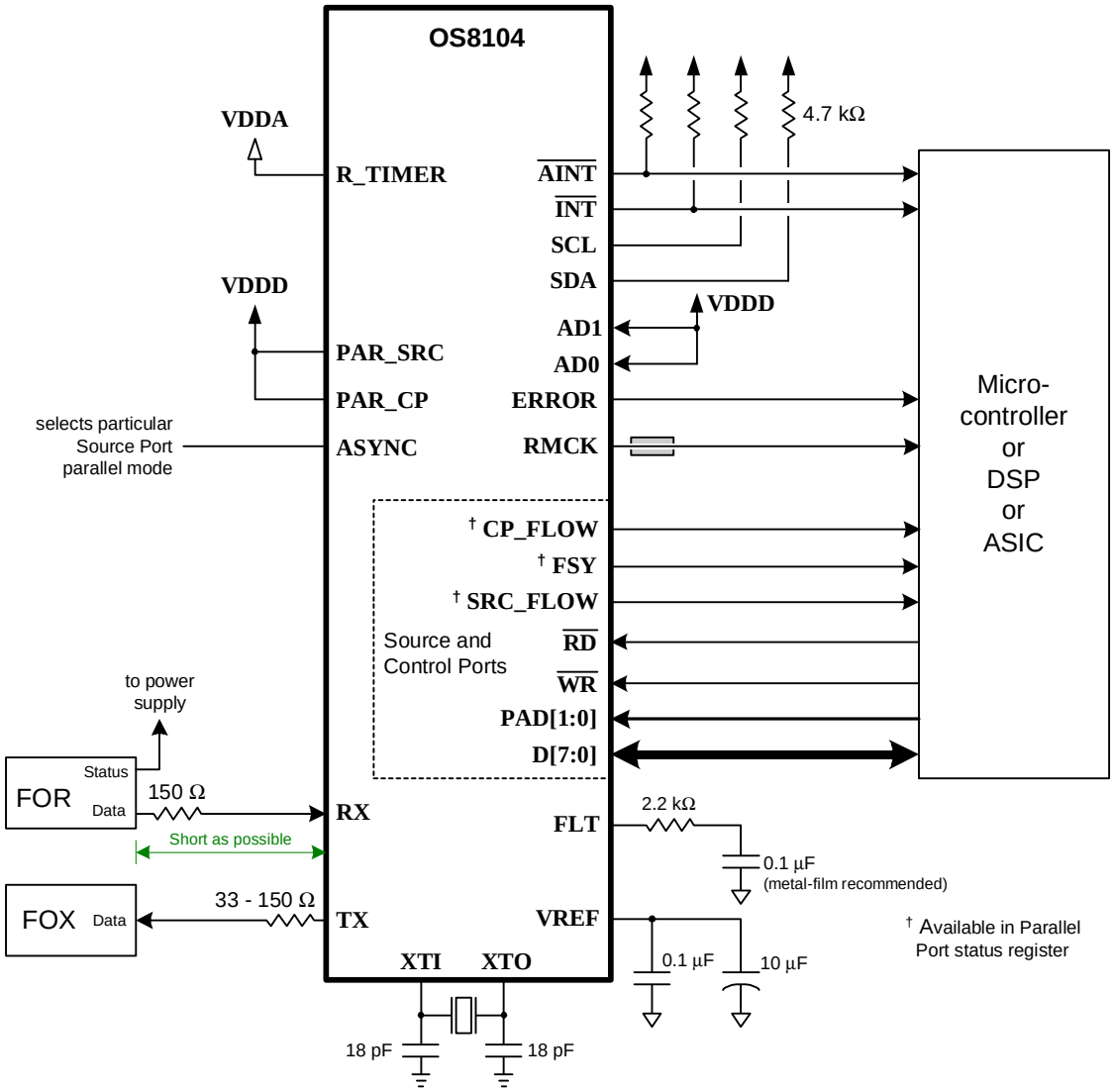


Figure 20-8: Typical Application — Source Port and Control Port in Parallel Mode

20.8 Stand-Alone Mode

Stand-Alone mode is selected by holding the **RD** and **WR** pins low. The diagram below illustrates using either the FOR, or the OS8104 **STATUS** pin, to manage local power.

If the OS8104 **STATUS** pin is used, then the resistor on **R_TIMER** is needed, and the **WAKE_UP** pin can control the power to the FOR to periodically check for Network activity. When activity is detected on **RX**, the chip will exit Zero-Power mode and **STATUS** will go low, indicating to the power supply to power up all peripheral components. When no activity exists on **RX** for a period of time, the chip will enter Zero-Power mode and the **STATUS** pin will go high, indicating to the power supply to power down all peripheral components. Stand-Alone mode is described in more detail in Chapter 17 on page 147.

The crystal oscillator is not needed since the clock is always recovered from the MOST Network **RX** pin.

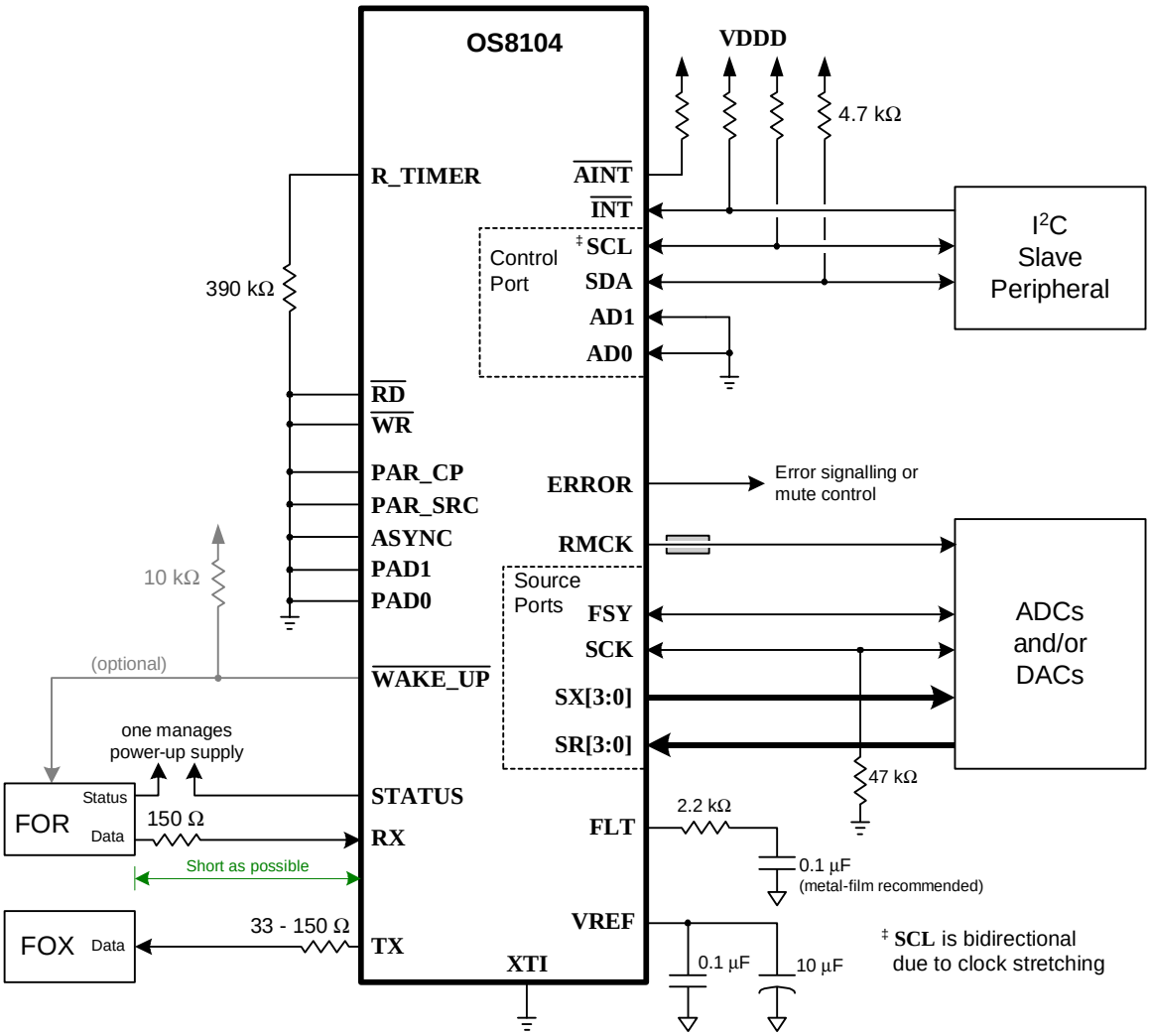


Figure 20-9: Typical Application — Stand-Alone Mode

Appendix A: References

1. Philips Semiconductors. (January 2000) The I²C-Bus Specification (ver. 2.1) [Online]. Available: http://www.semiconductors.philips.com/acrobat/various/I2C_BUS_SPECIFICATION_3.pdf
2. General I²C information can be found at http://www.semiconductors.philips.com/i2c
3. IEC-60958-1 “Digital Audio Interface - Part 1: General,” (Ed. 1), December, 1999. {also known as “S/PDIF”}
4. IEC-60958-3 “Digital Audio Interface - Part 3: Consumer Applications” (Ed. 1), December, 1999. {also known as “S/PDIF”}
5. General MOST Cooperation information can be found at http://www.mostcooperation.com/
6. MOST Cooperation. (October 2001) MOST Specification Framework (rev 1.1) [Online]. Available: http://www.mostcooperation.com/downloads/Specifications/
7. MOST Cooperation. (February 2001) MOST Specification (rev. 2.1) [Online]. Available: http://www.mostcooperation.com/downloads/Specifications/
8. MOST Cooperation. (February 2001) MOST High Protocol Specification (rev. 2.1) [Online]. Available: http://www.mostcooperation.com/downloads/Specifications/
9. MOST NetServices “Application Layer”, http://www.oasis.com/software/index.htm
10. MOST NetServices “Basic Layer”, http://www.oasis.com/software/index.htm
11. Philips Semiconductors. (June 5, 1996) I²S Bus Specification [Online]. Available: http://www.semiconductors.philips.com/acrobat/various/i2sbus.pdf

Appendix B: Register Overview

Addr.	Name	B7	B6	B5	B4	B3	B2	B1	B0	Register Name	Page
		Routing Section									
00h ... 3Fh	MRT0x00 ... MRT0x3F									Lower Half of Routing Table	100
40h ... 7Fh	MRT0x40 ... MRT0x7F									Upper Half of Routing Table	105
		Hardware Control Section									
80h	bXCR	MTR	OE	1	LPW	SAN	SBY	$\overline{ABY}$	$\overline{REN}$	Transceiver Control	37
81h	bXSR		MSL	MXL	ME	ERR		ESL	EXL	Transceiver Status	38
82h	bSDC1	EDG	DEL	POL	IO	NBR	SPD	$\overline{MT}$	$\overline{TCE}$	Source Data Control 1	46
83h	bCM1	PLD	RD2	RD1	RD0	XTL1	XTL0	MX1	MX0	Clock Manager 1	89
84h	bNC								RWD	Network Control	126
85h	bMSGC	STX	RBE		SAI	RALC	RERR	RMTX	RMRX	Message Control	115
86h	bMSGS	RBS	TXR			ALC	ERR	MTX	MRX	Message Status	117
87h	bNPR									Node Position	41
88h	bIE					IALC	IERR	IMTX	IMRX	Interrupt Enable	95
89h	bGA									Group Address	114
8Ah	bNAH									Node Address High	114
8Bh	bNAL									Node Address Low	114
8Ch	bSDC2	SPR2	SPR1	SPR0	MFSY	TCR1	TCR0	SDR1	SDR0	Source Data Control 2	48
8Dh	bSDC3	SIO				SPS				Source Data Control 3	49
8Eh	bCM2	$\overline{LOK}$	NAC	ZP	LP					Clock Manager 2	90
8Fh	bNDR									Node Delay	41
90h	bMPR									Maximum Position	42
91h	bMDR									Maximum Delay	42
93h	bCM4									Clock Manager 4	91
96h	bSBC					SAC3	SAC2	SAC1	SAC0	Synchronous Bandwidth Control	41
97h	bXSR2							INV		Transceiver Status 2	39
	mRCMB - Receive Control Message Buffer										119
A0h	bRTYP									Receive Message Type	
A1h	bRSAH									Source Address High	
A2h	bRSAL									Source Address Low	
A3h … B3h	bRCD0 ... bRCD16									Receive Control Data 0 ... Receive Control Data 16	
		Control Message Transmit Section (1)									
BEh	bXTIM									Transmit Retry Time	118
BFh	bXRTY									Transmit Retries	118
	mXCMB - Transmit Control Message Buffer										120
C0h	bXPRI									Transmit Priority	
C1h	bXTYP									Transmit Message Type	
C2h	bXTAH									Target Address High	
C3h	bXTAL									Target Address Low	

Addr.	Name	B7	B6	B5	B4	B3	B2	B1	B0	Register Name	Page
C4h … D4h	bXCD0 … bXCD16									Transmit Control Data 0 … Transmit Control Data 16	
	Control Message Transmit Section (2)										
D5h	bXTS									Transmit Transfer Status	118
	Packet Control Section										
E2h	bPCTC				RAF	RAC		RATX	RARX	Packet Control	137
E3h	bPCTS				AF	AC		ATX	ARX	Packet Status	138
E6h	bPCMA								APCM	Parallel-Combined mode Activate	75
E8h	bAPAH									Alternate Packet Addr. High	135
E9h	bAPAL									Alternate Packet Addr. Low	136
EAh	bPSTX	ASTX								Packet Start Tx	137
ECh	bPLDT									Packet Length For Data Transmit	136
F2h	bPPI									Packet Priority	136
	mARP - Packet Receive Buffer										138
180h	bARTH									Received Target Addr. High	
181h	bARTL									Received Target Addr. Low	
182h	bASAH									Source Address High	
183h	bASAL									Source Address Low	
184h … 1B3h	bARD0 … bARD47									Packet Receive Data 00 … Packet Receive Data 47	
	mAXP - Packet Transmit Buffer										139
1C0h	bATAH									Target Address High	
1C1h	bATAL									Target Address Low	
1C2h … 1F1h	bAXD0 … bAXD47									Packet Transmit Data 00 … Packet Transmit Data 47	
	mCRA - Channel Resource Allocation Buffer										131
380h … 3BBh	CRA00 … CRA3B									Channel Allocation Table	
	†***mSIMB - Stand-Alone Message Buffer***										148
EBh	bSIMC									Count of bytes to send	
ECh	bSITA									I^2C target address	
EDh	bSIMA									I^2C MAP	
EEh … F1h	bSITD0 … bSITD3									I^2C transfer data byte 0 … I^2C transfer data byte 3	
F2h	bSITC									I^2C control/status byte	

† Available only in Stand-Alone mode

Appendix C: Data Sheet Revision History

Section/Page	Change
DS8104PP2 Changes:	
Text was edited throughout the Data Sheet. The following are the major changes:	
Section 2.5	Expanded to include chip bandwidth numbers for each data transfer method
Section 6.2.3	Added bXSR2 register
Section 8.3	Expanded Parallel-Combined mode Section
Section 9.5	Added Crystal Oscillator Section
Section 12	Clarified Routing Synchronous Data Section
Tables 12-2, 12-4	Added 48Fs Routing Data to Source Port In and Source Port Out Tables.
Section 16.4	Expanded chip version number table
Section 18	Updated all timing diagrams in Electrical Characteristics Section
Section 18.2	Added Recommended Operating Conditions Section Fs range changed to 37.9 kHz to 48.1 kHz
Section 18.3	Added Low- and Zero-Power mode current values Added thermal resistances
Section 18.4.1	Added **RMCK** rise/fall time Added Pulse Width Distortion and Jitter numbers Added configuration pin setup and hold times
Section 18.4.2	Changed t_{drd} to 23 ns (was 10 ns) Added **RD**, **WR** minimum high time Changed **PAD[1:0]** and data hold times Added t_{dhd} hold time maximum Added **SRC_FLOW** specifications Added **CP_FLOW** specifications
Section 18.4.3	Changed t_{sxv} to 30 ns (was 25 ns) Added **FSY** frequency values
Section 18.4.4	Changed t_{sxv} to 30 ns (was 25 ns) Added **FSY** frequency values Added **FSY**, **SCK** rise/fall times
Section 18.4.5	Added t_{sklcsl} and t_{sklcsh} SPI timing parameters (note - t_{sklcsl} removed in later Data Sheet) Added t_{cdv} **SDOUT** time
DS8104FP1 Changes:	
	Converted *Preliminary Product* Data Sheet to *Final Product* Data Sheet
Sections 8 - 8.2.3 Section 8.2.2	Updated many of the Figures in these sections. Added note that OS8104 must be in lock when writing data in Parallel Asynchronous mode.
Section 9.1, 9.2	Added **RMCK** glitch note to lock/unlock events
Section 18.1	Added **RX** absolute maximum input current specification note
Section 18.5.1	Added Network Frame Frequency for PLL unlocked Changed **PWD** and Jitter numbers. Added t_{apwd}
Section 18.5.2	Changed t_{drd} to 27 ns (was 23 ns) Changed t_{rdh} and t_{wrh} from 25 ns to 20 ns Changed t_{ahdr}, t_{dhd}, and t_{ahdw} from 4 ns to 6 ns Added t_{wrcpsp} spec for Parallel-Combined and Parallel-Synchronous modes
Section 18.5.5	Added SPI timing table for an unlocked state
Section 18.5.6	Added I^2C unlock timing note 2
Appendix A	Updated Figure A-1 and text to describe external controller's delayed power-down control
DS8104FP2 Changes:	
Section 18.4	Added separate specification for **RX** input capacitance

Section/Page	Change
DS8104FP3 Changes:	
Revision D silicon information was added, and text, and some Figures, were reformatted throughout the document. The following are the major changes:	
Chapter 4 Section 8.1	Added clarification of serial CP pins when the Control Port is configured in parallel mode
Chapter 5	Control Port operation clarified
Section 6.2.10	Changed maximum valid times for bMPR, bMDR, bNPR, and bNDR Added maximum latency for a valid mCRA table
Chapter 7	Added recommendation for driving **SCK** before lock is achieved
Section 7.4.2	Added information about Routing S/PDIF data
Section 9.1	bCM4 register and settings documented
Chapter 12	Routing operation clarified
Section 12.7	Added MRT power-up defaults
Section 12.8	Added Routing Limitations
Section 13.2.5	Updated valid range of bXTIM (03h-FDh)
Section 13.5.2.3	bNC register (**RWD** bit) documented (revision D only)
Section 15.1	Changed Alternate Packet Address default value to 0F0Fh
Section 18.1	Added Juction Temperature specification
Section 18.5.1	Updated jitter tolerance Added bCM4 setting (Note 4) to t_{apwd} specification Pulse width variation specification, t_{pwmn} and t_{pwmx}, tightened slightly Jitter and APWD specifications split out between chip revisions
Section 18.5.3	Added t_{rswr} and t_{rswf} timing
Section 18.5.6	Removed timing t_{sklcsl} requirement, **SCL** low to $\overline{\textbf{CS}}$ low setup The t_{cdv} timing moved from the minimum column to the maximum column. In unlock (lower table), t_{cdv} was changed to 1.75 from 1.2 µs
Section 18.5.7	Note 2 in the lower table was changed for the PLL unlock case
Section 20.1	Crystal parameters clarified
Section 20.2	Added recommendation for conformal coating on **FLT** capacitor Added recommendation that **FLT** capacitor is a metal film type as opposed to a ceramic type
Section 20.3	Added Layout Recommendations section
Appendix B	Added References Appendix
DS8104FP4 Changes:	
Section 18.5.1	Pulse width variation specification, t_{pwmn} and t_{pwmx} tightened slightly Updated APWD specification, t_{apwd} (revisions greater than C)
Section 18.5.3	Added t_{cpd} specification, for parallel Control Port accesses

INDEX

A

B

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

V

W

X

Z