Summary - File Write

Name	file_write
Worker Type	Application
Version	v1.3
Release Date	Febuary 2018
Component Library	ocpi.core
Workers	file_write.rcc file_write.hdl
Tested Platforms	isim, xsim, modelsim, xilinx13_3, centos6, centos7

Functionality

The File_Write component writes application data to a file. It is normally used by specifying an instance of the File_Write component, and connecting its input port to an output port of the component produces the data. The name of the file to be written is specified in a property. This component has one input port whose name is in, which carries the messages to be written to the file. There is no protocol associated with the file, enabling it to be agnostic as to the protocol of the file data and the connected output port.

Operating Modes

Data Streaming Mode

In data streaming mode, the contents of the file becomes the payloads of the stream of messages arriving at the input port. No message lengths or opcodes are recorded in the output file.

Messaging Mode

In messaging mode, the contents of the output file is written as a sequence of defined messages, with an 8-byte header in the file itself preceding the data for each message written to the file. This header contains the length and opcode of the message, with the data contents of the message following the header. The length can be zero, meaning that a header will be written but no data will follow the header in the file.

The first 32-bit word of the header is interpreted as the message length in bytes, little-endian. The next 8-bit byte is the opcode of the message, followed by 3 padding bytes. If the end of the file is encountered while reading a message header, or while reading the header-specified length of the message payload, an error will be reported and the component will terminate. The following Messaging File Field Layout shows the layout of a file:

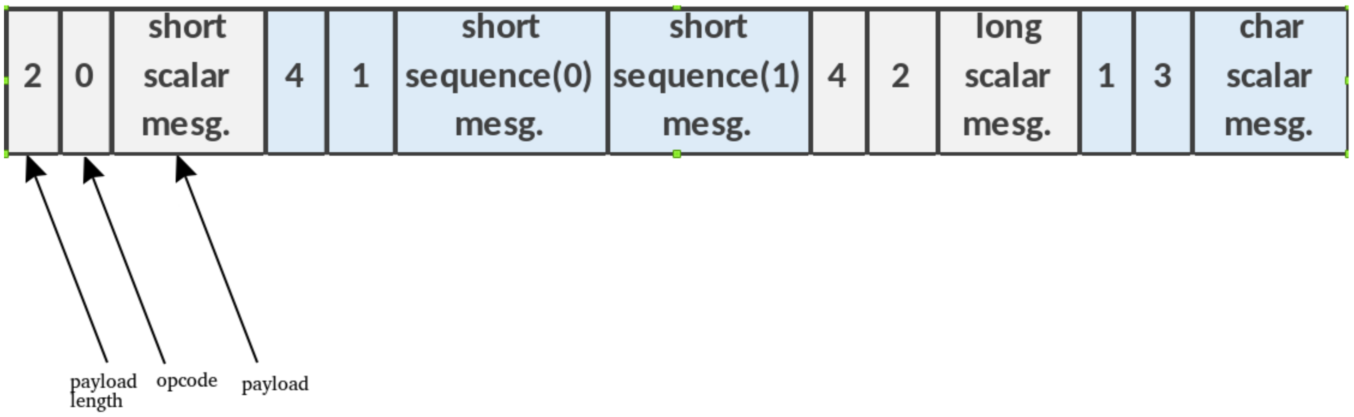

Figure 1: Messaging File Field Layout

No Protocol

The port on the component has no protocol specified in order to support interfacing with any protocol. This means that the data file that is created is formatted to match the protocol of the output port of the connected worker.

End of File Handling

When the File_Write component receives a zero-length message, it will interpret it as the end of data if the *stopOnEOF* property is true (the default). In this case it will declare itself "done, and not write any further messages to the file (and not write the zero-length message to the file either). If *stopOnEOF* is false, and it is in messaging mode, it will write the zero-length message to the file like any other message (with a header, but with no data).

Zero Length Messages

A zero-length message (ZLM) is any message that has zero payload (data) associated with it but the rest of the message signaling and opcode are passed through. The default operation for end-of-file for File_Read is to pass along a ZLM on the opcode set by the property *opcode* (default is 0) which is then advanced to the downstream worker. The ZLM is processed by the File_Write component at the end of the application chain and File_Write sets its state to"done". In file based applications it is common practice to set an application to complete when the File write component is done to ensure all data is processed. This means that all other workers in the path must process and pass on ZLMs for this to function properly.

Worker Implementation Details

file_write.hdl

This worker will only run on simulator platforms. This includes isim, xsim, and modelsim and will not run on or be built for any other hardware platforms. This is because it conatins code that cannot be realized into RTL.

file_write.rcc

This worker in implemented in the C language version of the RCC model. Most new workers are implemented in the C++ language version of the RCC model.

Source Dependencies

file_write.rcc

- file_write.c
- file_write.h

file_write.hdl

- file_write.vhd

Component Spec Properties

Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
fileName	string length:1024	-	-	Initial	-	-	The name of the file that is written to disk from the input port.
messagesInFile	bool	-	-	Readable, Initial	-	false	The flag that is used to turn on and off message mode. See section on Message Mode.
bytesWritten	uLongLong	-	-	Volatile	-	-	The number of bytes written to file. Useful for debugging data flow issues.
messagesWritten	uLongLong	-	-	Volatile	-	-	The number of messages written to file. Useful for debugging data flow issues.
stopOnEOF	bool	-	-	Initial	-	true	The flag used to enable or disable the zero length message that puts the component into the done state.

Worker Properties

file_write.hdl

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
Spec Property	fileName	string	-	-	Readable	-	-	added Readable
Property	CWD_MAX_LENGTH	ulong	-	-	Paramater	-	256	parameter for max string length for the cwd property
Property	cwd	string length:CWD_MAX_LENGTH	-	-	Volatile	-	-	the current working directory of the application (this is required for the hdl version of this worker and cannot be detiermined automaticly)

file_write.rcc

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
Spec Property	fileName	string	-	-	Readable	-	-	added Readable
Spec Property	suppressEOF	string	-	-	Readable	-	-	added Readable
Spec Property	messageSize	string	-	-	Readable	-	-	added Volatile

Component Ports

Name	Producer	Protocol	Optional	Advanced	Usage
in	False	None	False	-	-

Worker Interfaces

file_write.hdl

Type	Name	DataWidth	Advanced	Usage
StreamInterface	in	32	-	-

file_write.rcc

Type	Name	DataWidth	Advanced	Usage
Port	in	-	buffersize=8k	-

Test and Verification

All test benches use this worker as part of the verification process. A unit-test does not exist for this component.