CES*741: Development of Scientific Computing Software
School of Computational Engineering and Science
McMaster University

sDFT:

simple Discrete Fourier Transform
command-line tool
for
UNIX systems
on single- or multi-core (SMP) computers

Modules Interface Specification

sDFT version α

Created: 2009-03-26 Gonzalo Sánchez
Updated: 2009-04-21 Gonzalo Sánchez
Doc. ID: sDFT_MIS_doc 58

TABLE OF CONTENTS

[1] REFERENCE MATERIAL 6
[1.1] Related Documentation 6
[1.2] Abbreviations and Acronyms 6
[1.3] Symbols 6
[1.4] Index of Modules 6

[2] INTRODUCTION 7
[2.1] Purpose of the Document 7
[2.2] Organization of the Document 7

[3] MODULES INTERFACE SPECIFICATION 8
[3.1] Hardware Abstraction Modules 8
[3.2] Behaviour Abstraction Modules 8
[3.2.1] M1: sDFT module 8
[3.2.1.1] USES 8
[3.2.1.1.1] Imported constants 8
[3.2.1.1.2] Imported data types 8
[3.2.1.1.3] Imported access programs 8
[3.2.1.2] SYNTAX 9
[3.2.1.2.1] Exported constants 9
[3.2.1.2.2] Exported data types 9
[3.2.1.2.3] Exported access programs 9
[3.2.1.3] SEMANTICS 9
[3.2.1.3.1] State variables 9
[3.2.1.3.2] State invariants 9
[3.2.1.3.3] Assumptions 10
[3.2.1.3.4] Local types 10
[3.2.1.3.5] Local constants 10
[3.2.1.3.6] Local functions 10
[3.2.1.3.7] Access programs 10
[3.2.1.3.8] Considerations 11
[3.2.2] M2: Configuration module 12
[3.2.2.1] USES 12
[3.2.2.1.1] Imported constants 12
[3.2.2.1.2] Imported data types 12
[3.2.2.1.3] Imported access programs 12
[3.2.2.2] SYNTAX 12
[3.2.2.2.1] Exported constants 12
[3.2.2.2.2] Exported data types 12
[3.2.2.2.3] Exported access programs 12
[3.2.2.3] SEMANTICS 13
[3.2.2.3.1] State variables 13
[3.2.2.3.2] State invariants 13
[3.2.2.3.3] Assumptions 13
[3.2.2.3.4] Local types 13
[3.2.2.3.5] Local constants 13
[3.2.2.3.6] Local functions 14
[3.2.2.3.7] Access programs 14
[3.2.2.3.8] Considerations 14
[3.2.3] M3: Input module 15

[3.2.3.1] USES 15
[3.2.3.1.1] Imported constants 15
[3.2.3.1.2] Imported data types 15
[3.2.3.1.3] Imported access programs 15
[3.2.3.2] SYNTAX 15
[3.2.3.2.1] Exported constants 15
[3.2.3.2.2] Exported data types 15
[3.2.3.2.3] Exported access programs 15
[3.2.3.3] SEMANTICS 15
[3.2.3.3.1] State variables 16
[3.2.3.3.2] State invariants 16
[3.2.3.3.3] Assumptions 16
[3.2.3.3.4] Local types 16
[3.2.3.3.5] Local constants 16
[3.2.3.3.6] Local functions 16
[3.2.3.3.7] Access programs 16
[3.2.3.3.8] Considerations 17
[3.2.4] M4: Output module 18
[3.2.4.1] USES 18
[3.2.4.1.1] Imported constants 18
[3.2.4.1.2] Imported data types 18
[3.2.4.1.3] Imported access programs 18
[3.2.4.2] SYNTAX 18
[3.2.4.2.1] Exported constants 18
[3.2.4.2.2] Exported data types 18
[3.2.4.2.3] Exported access programs 18
[3.2.4.3] SEMANTICS 18
[3.2.4.3.1] State variables 19
[3.2.4.3.2] State invariants 19
[3.2.4.3.3] Assumptions 19
[3.2.4.3.4] Local types 19
[3.2.4.3.5] Local constants 19
[3.2.4.3.6] Local functions 19
[3.2.4.3.7] Access programs 19
[3.2.4.3.8] Considerations 20
[3.3] Software Decision Modules **21**
[3.3.1] M5: Sequence data module 21
[3.3.1.1] USES 21
[3.3.1.1.1] Imported constants 21
[3.3.1.1.2] Imported data types 21
[3.3.1.1.3] Imported access programs 21
[3.3.1.2] SYNTAX 21
[3.3.1.2.1] Exported constants 21
[3.3.1.2.2] Exported data types 21
[3.3.1.2.3] Exported access programs 21
[3.3.1.3] SEMANTICS 22
[3.3.1.3.1] State variables 22
[3.3.1.3.2] State invariants 22
[3.3.1.3.3] Assumptions 22
[3.3.1.3.4] Local types 22
[3.3.1.3.5] Local constants 22
[3.3.1.3.6] Local functions 22
[3.3.1.3.7] Access programs 23
[3.3.1.3.8] Considerations 23
[3.3.2] M6: Stream data module 24
[3.3.2.1] USES 24
[3.3.2.1.1] Imported constants 24
[3.3.2.1.2] Imported data types 24

[3.3.2.1.3] Imported access programs....24
[3.3.2.2] SYNTAX....24
[3.3.2.2.1] Exported constants....24
[3.3.2.2.2] Exported data types....24
[3.3.2.2.3] Exported access programs....24
[3.3.2.3] SEMANTICS....24
[3.3.2.3.1] State variables....25
[3.3.2.3.2] State invariants....25
[3.3.2.3.3] Assumptions....25
[3.3.2.3.4] Local types....25
[3.3.2.3.5] Local constants....25
[3.3.2.3.6] Local functions....25
[3.3.2.3.7] Access programs....25
[3.3.2.3.8] Considerations....26
[3.3.3] M7: Window functions module....27
[3.3.3.1] USES....27
[3.3.3.1.1] Imported constants....27
[3.3.3.1.2] Imported data types....27
[3.3.3.1.3] Imported access programs....27
[3.3.3.2] SYNTAX....27
[3.3.3.2.1] Exported constants....27
[3.3.3.2.2] Exported data types....27
[3.3.3.2.3] Exported access programs....27
[3.3.3.3] SEMANTICS....27
[3.3.3.3.1] State variables....28
[3.3.3.3.2] State invariants....28
[3.3.3.3.3] Assumptions....28
[3.3.3.3.4] Local types....28
[3.3.3.3.5] Local constants....28
[3.3.3.3.6] Local functions....28
[3.3.3.3.7] Access programs....28
[3.3.3.3.8] Considerations....29
[3.3.4] M8: Sequence DFT module....30
[3.3.4.1] USES....30
[3.3.4.1.1] Imported constants....30
[3.3.4.1.2] Imported data types....30
[3.3.4.1.3] Imported access programs....30
[3.3.4.2] SYNTAX....30
[3.3.4.2.1] Exported constants....30
[3.3.4.2.2] Exported data types....30
[3.3.4.2.3] Exported access programs....30
[3.3.4.3] SEMANTICS....31
[3.3.4.3.1] State variables....31
[3.3.4.3.2] State invariants....31
[3.3.4.3.3] Assumptions....31
[3.3.4.3.4] Local types....31
[3.3.4.3.5] Local constants....31
[3.3.4.3.6] Local functions....31
[3.3.4.3.7] Access programs....31
[3.3.4.3.8] Considerations....32
[3.3.5] M9: Stream DFT module....33
[3.3.5.1] USES....33
[3.3.5.1.1] Imported constants....33
[3.3.5.1.2] Imported data types....33
[3.3.5.1.3] Imported access programs....33
[3.3.5.2] SYNTAX....33
[3.3.5.2.1] Exported constants....33
[3.3.5.2.2] Exported data types....33

[3.3.5.2.3] Exported access programs....33
[3.3.5.3] SEMANTICS....33
[3.3.5.3.1] State variables....34
[3.3.5.3.2] State invariants....34
[3.3.5.3.3] Assumptions....34
[3.3.5.3.4] Local types....34
[3.3.5.3.5] Local constants....34
[3.3.5.3.6] Local functions....34
[3.3.5.3.7] Access programs....34
[3.3.5.3.8] Considerations....35

[4] REFERENCES....**36**

[1] REFERENCE MATERIAL

[1.1] Related Documentation

ID	*Document description*
sDFT_SRS	sDFT Software Requirements Specification.
sDFT_MG	sDFT Module Guide.
sDFT_MIS	sDFT Module Interface Specification (***this document***).
sDFT_SVTG	sDFT System Validation Tests Guide.
sDFT_FL	sDFT Files List.

Table 1: Related documentation

[1.2] Abbreviations and Acronyms

Refer to the Table of Abbreviations and Acronyms in the SRS document (see: Table 1: Related documentation)

[1.3] Symbols

Refer to the Table of Symbols in the SRS document (see: Table 1: Related documentation)

[1.4] Index of Modules

Hardware Abstraction Modules	*Section*	*Page*
M0: N/A	N/A	N/A
Behaviour Abstraction Modules	***Section***	***Page***
M1: sDFT module	3.2.1	8
M2: Configuration module	3.2.2	12
M3: Input module	3.2.3	15
M4: Output module	3.2.4	18
Software Decision Modules	***Section***	***Page***
M5: Sequence data module	3.3.1	21
M6: Stream data module	3.3.2	24
M7: Window functions module	3.3.3	27
M8: Sequence DFT module	3.3.4	30
M9: Stream DFT module	3.3.5	33

Table 2: Modules

[2] INTRODUCTION

This section of the MIS (Modules Interface Specification) document introduces its purpose, and outlines the organization of the document.

[2.1] Purpose of the Document

This document provides the modules interface specification for the sDFT (simple Discrete Fourier Transform) tool.

It is intended for programmers with knowledge of the UNIX environments, who would either be interested in:

- performing maintenance tasks on the software
- enhancing the software with added features, or to broaden its scope of application
- using modules of the software for inclusion in other software products

[2.2] Organization of the Document

The sDFT modules interface specification document is structured following the MIS template for scientific software proposed in [1] and its extension in [2] (see also: [3], [4], [5], [6]). The complete layout can be appreciated in the Table of Contents starting at page 2.

The first part of the document includes the **Table of Contents**, the **Reference Material** tables, and this **Introduction**.

The rest of the document, which constitutes the core of the MIS, is organized as follows:

- **Modules Interface Specification**: This section is the actual specification of the modules interface. A subsection is devoted to each module, and the information therein is presented in tabular format.

To the end of the document, the list of bibliographical **References** is included.

[3] MODULES INTERFACE SPECIFICATION

This section is the actual specification of the modules interface. A subsection is devoted to each module, and the information therein is presented in tabular format.

NOTE: This MIS is for sDFT version α.

[3.1] Hardware Abstraction Modules

All of the Hardware Abstraction Modules are already implemented by the underlying operating system and/or programming language runtime environment and function libraries. There are no Hardware Abstraction Modules that need to be defined within the sDFT tool.

[3.2] Behaviour Abstraction Modules

[3.2.1] M1: sDFT module

[3.2.1.1] USES

[3.2.1.1.1] Imported constants

Constant	Type	Module
EXIT_SUCCESS	integer	(UNIX)

[3.2.1.1.2] Imported data types

Data Type	–	Module
sdft_cfgT	–	M2: Configuration module

[3.2.1.1.3] Imported access programs

Access Program	–	Module
set_cfg	–	M2: Configuration module
get_cfg	–	M2: Configuration module
seq_init	–	M5: Sequence data module
get_input	–	M3: Input module
win_rect	–	M7: Window functions module
win_hanning	–	M7: Window functions module
sdft_seqDFT	–	M8: Sequence DFT module
get_output	–	M4: Output module

Access Program	–	Module
seq_destroy	–	M5: Sequence data module

[3.2.1.2] SYNTAX

[3.2.1.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.2.1.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.2.1.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
main	integer ptr(ptr)	integer	–

[3.2.1.3] SEMANTICS

[3.2.1.3.1] State variables

State Variable	Type	Comments
exit_val	integer	Reassigned; captures the exit value of called access programs, if available
sdft_cfg	sdft_cfgT = Tuple	Tuple that holds the configuration for the actual invocation of the program (the values of command-line switches, environment variables, and keywords in configuration files, after being sorted by precedence and exclusion rules)

[3.2.1.3.2] State invariants

#	State Invariant
1	sdft_cfg: The program configuration does not change at runtime

[3.2.1.3.3] Assumptions

#	Assumption
1	sDFT Module logic depends on the configuration: different user choices (*e.g.* change of algorithm) might require different management of resources and coordination.

[3.2.1.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.2.1.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.2.1.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.2.1.3.7] Access programs

name	
Transition:	–
Exceptions:	–

name	
Output:	–
Exceptions:	–

[3.2.1.3.8] Considerations

#	Consideration
1	sDFT is the top-level module of the application. Its task is the management of resources and coordination of the other modules. In sDFT version α the resources allocation and module coordination is simple: only one sequence of data needs to be allocated, and the tasks of the software are virtually sequential: • Request the evaluation of the configuration for the actual invocation of the program, to set up accordingly • Retrieve the configuration • Request allocation for the necessary sequence (length) • Request the input of data • Request the application of a 'windowing' function to weigh the input data, according to the user's configuration • Request the computation of the DFT or inverse DFT on the (weighed) input data, also according to the user's configuration • Request the publication of the results • Release the allocated resources

[3.2.2] M2: Configuration module

[3.2.2.1] USES

[3.2.2.1.1] Imported constants

Constant	Type	Module
N/A	–	–

[3.2.2.1.2] Imported data types

Data Type	–	Module
N/A	–	–

[3.2.2.1.3] Imported access programs

Access Program	–	Module
N/A	–	–
N/A	–	–

[3.2.2.2] SYNTAX

[3.2.2.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.2.2.2.2] Exported data types

Data Type	–	–
sdft_cfgT	–	–

[3.2.2.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
set_cfg	integer ptr(ptr)	integer	ENOFILE EBADFILE EBADVAR EBADSW
get_cfg	-	sdft_cfgT	-

[3.2.2.3] SEMANTICS

[3.2.2.3.1] State variables

State Variable	Type	Comments
config.SDFT_ALG	string	See SRS Table 3: sDFT command-line switches, environment variables, and configuration parameters
config.SDFT_CFILE	string	
config.SDFT_NCPUS	natural	
config.SDFT_DTYPE	string	
config.SDFT_DIM	natural	
config.SDFT_IDNFMT	string	
config.SDFT_ODNFMT	string	
config.SDFT_IFILE	string	
config.SDFT_NDATA	natural	
config.SDFT_NNODES	natural	
config.SDFT_OFILE	string	
config.SDFT_IFTYPE	string	
config.SDFT_OFTYPE	string	
config.SDFT_INV	boolean	
config.SDFT_WIN	string	

[3.2.2.3.2] State invariants

#	State Invariant
N/A	–

[3.2.2.3.3] Assumptions

#	Assumption
1	set_cfg is called before any calls to get_cfg

[3.2.2.3.4] Local types

Local Type	Type	–
config	sdft_cfgT	–

[3.2.2.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.2.2.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
prn_help	-	integer	-
prn_cfg	-	integer	-
prn_ver	-	integer	-

[3.2.2.3.7] Access programs

set_cfg()	
Transition:	Fills the tuple of configuration parameters for the actual invocation of sDFT, computing it (priority-wise) from the system defaults, the configuration file, the environment variables, and the pertinent command-line switches(tuple type: sdft_cfgT).
Exceptions:	Invalid command-line switch or value

get_cfg()	
Outputs:	Tuple containing the configuration parameters for the actual invocation of sDFT (tuple type:sdft_cfgT). Can be accessed by fields.
Exceptions:	-

[3.2.2.3.8] Considerations

#	Consideration
N/A	-

[3.2.3] M3: Input module

[3.2.3.1] USES

[3.2.3.1.1] Imported constants

Constant	Type	Module
EXIT_SUCCESS	integer	(UNIX)
EXIT_FAILURE	integer	(UNIX)

[3.2.3.1.2] Imported data types

Data Type	–	Module
complex	–	(Application-wide)

[3.2.3.1.3] Imported access programs

Access Program	–	Module
seq_setval	–	M5: Sequence data module

[3.2.3.2] SYNTAX

[3.2.3.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.2.3.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.2.3.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
get_input	natural	integer	EBADFORMAT EBADTYPE

[3.2.3.3] SEMANTICS

[3.2.3.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.2.3.3.2] State invariants

#	State Invariant
1	N: sequence length

[3.2.3.3.3] Assumptions

#	Assumption
1	Input data is: • ASCII text, conformant to the numerical format for 'floating point' type as used by scanning and printing standard library routines `scanf()`, `fprintf()` and `printf()` of the C programming language; • unidimensional • complex-valued • received as a single columns of numbers, where the first number is the real part of the first data value, and the second number its imaginary part, and so on

[3.2.3.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.2.3.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.2.3.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.2.3.3.7] Access programs

get_input()	
Transition:	Reads input data from the input specified in the field `SDFT_IFILE` of the configuration tuple and stores it in the sequence
Exceptions:	–

[3.2.3.3.8] Considerations

#	Consideration
N/A	–

[3.2.4] M4: Output module

[3.2.4.1] USES

[3.2.4.1.1] Imported constants

Constant	Type	Module
EXIT_SUCCESS	integer	(UNIX)

[3.2.4.1.2] Imported data types

Data Type	–	Module
complex	–	(Application-wide)

[3.2.4.1.3] Imported access programs

Access Program	–	Module
seq_getval	–	M5: Sequence data module

[3.2.4.2] SYNTAX

[3.2.4.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.2.4.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.2.4.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
get_output	natural	integer	–

[3.2.4.3] SEMANTICS

[3.2.4.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.2.4.3.2] State invariants

#	State Invariant
1	N: sequence length

[3.2.4.3.3] Assumptions

#	Assumption
N/A	–

[3.2.4.3.4] Local types

Local Type	Type	
N/A	–	–

[3.2.4.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.2.4.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.2.4.3.7] Access programs

get_output()	
Transition/Output:	Reads the data from the sequence and writes it to the output specified in the field SDFT_OFILE of the configuration tuple (file|stdout) in the format specified in the SDFT_ODNFMT field of the configuration tuple.
Exceptions:	–

name	
Output:	–
Exceptions:	–

[3.2.4.3.8] Considerations

#	Consideration
N/A	–

[3.3] Software Decision Modules

[3.3.1] M5: Sequence data module

[3.3.1.1] USES

[3.3.1.1.1] Imported constants

Constant	Type	Module
ENOMEM	integer	(UNIX)
EXIT_SUCCESS	integer	(UNIX)

[3.3.1.1.2] Imported data types

Data Type	–	Module
complex	–	(Application-wide)

[3.3.1.1.3] Imported access programs

Access Program	–	Module
N/A	–	–

[3.3.1.2] SYNTAX

[3.3.1.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.3.1.2.2] Exported data types

Data Type	–	–
–	–	–

[3.3.1.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
seq_init	natural	integer	ENOMEM
seq_setval	complex natural	integer	-

Access Program	Inputs	Outputs	Exceptions
seq_getval	natural	complex	-
seq_getlen	-	natural	-
seq_destroy	-	integer	-

[3.3.1.3] SEMANTICS

[3.3.1.3.1] State variables

State Variable	Type	Comments
seq	ptr	Sequence of complex-valued data

[3.3.1.3.2] State invariants

#	State Invariant
1	N: sequence length

[3.3.1.3.3] Assumptions

#	Assumption
1	seq_init must be called to allocate memory for a sequence prior to using other access programs of the module

[3.3.1.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.3.1.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.3.1.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.1.3.7] Access programs

seq_init()	
Transition:	Allocates memory for an instance of a sequence of N real (“R”) or complex (“C”) numbers.
Exceptions:	ENOMEM: Memory allocation failure

seq_destroy()	
Transition:	Releases the allocated memory by `seq_init()`
Exceptions:	–

seq_setval(z,i)	
Transition:	Assigns the complex value ‘z’ at position ‘i’ of the sequence
Exceptions:	–

seq_getval(i)	
Output:	Retrieves the value ‘z’ at position ‘i’ of the sequence
Exceptions:	–

seq_getlen()	
Output:	Retrieves the length of the sequence
Exceptions:	–

[3.3.1.3.8] Considerations

#	Consideration
N/A	–

[3.3.2] M6: Stream data module

NOTE: not implemented in sDFT v. α

[3.3.2.1] USES

[3.3.2.1.1] Imported constants

Constant	Type	Module
N/A	–	–

[3.3.2.1.2] Imported data types

Data Type		Module
N/A	–	–

[3.3.2.1.3] Imported access programs

Access Program	–	Module
N/A	–	–

[3.3.2.2] SYNTAX

[3.3.2.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.3.2.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.3.2.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.2.3] SEMANTICS

[3.3.2.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.3.2.3.2] State invariants

#	State Invariant
N/A	–

[3.3.2.3.3] Assumptions

#	Assumption
N/A	–

[3.3.2.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.3.2.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.3.2.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.2.3.7] Access programs

name	
Transition:	–
Exceptions:	–

name	
Output:	–
Exceptions:	–

[3.3.2.3.8] Considerations

#	Consideration
–	–

[3.3.3] M7: Window functions module

[3.3.3.1] USES

[3.3.3.1.1] Imported constants

Constant	Type	Module
EXIT_SUCCESS	integer	(UNIX)

[3.3.3.1.2] Imported data types

Data Type	–	Module
complex	–	(Application-wide)

[3.3.3.1.3] Imported access programs

Access Program	–	Module
seq_getval	–	M5: Sequence data module
seq_setval	–	M5: Sequence data module

[3.3.3.2] SYNTAX

[3.3.3.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.3.3.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.3.3.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
win_rect	natural	integer	–
win_hanning	natural	integer	–

[3.3.3.3] SEMANTICS

[3.3.3.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.3.3.3.2] State invariants

#	State Invariant
1	N: Sequence length

[3.3.3.3.3] Assumptions

#	Assumption
N/A	–

[3.3.3.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.3.3.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.3.3.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.3.3.7] Access programs

win_rect()	
Transition:	Does nothing: applying a 'rect' window is the same as leaving the sequence 'as is'.
Exceptions:	-

win_hanning()	
Transition:	Applies a Hanning (or Hann) window to the sequence (weights the elements of the sequence).
Exceptions:	-

[3.3.3.3.8] Considerations

#	Consideration
1	Designed only for real-valued input data. Application to complex valued data with nonzero imaginary components is responsibility of the user.

[3.3.4] M8: Sequence DFT module

[3.3.4.1] USES

[3.3.4.1.1] Imported constants

Constant	Type	Module
EXIT_SUCCESS	integer	(UNIX)
EXIT_FAILURE	integer	(UNIX)

[3.3.4.1.2] Imported data types

Data Type	–	Module
complex	–	(Application-wide)

[3.3.4.1.3] Imported access programs

Access Program	–	Module
seq_getval	–	M5: Sequence data module
seq_setval	–	M5: Sequence data module

[3.3.4.2] SYNTAX

[3.3.4.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.3.4.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.3.4.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
sdft_seqDFT	natural boolean	integer	EXIT_FAILURE

[3.3.4.3] SEMANTICS

[3.3.4.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.3.4.3.2] State invariants

#	State Invariant
1	N: Sequence length

[3.3.4.3.3] Assumptions

#	Assumption
N/A	–

[3.3.4.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.3.4.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.3.4.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.4.3.7] Access programs

sdft_seqDFT()	
Transition:	Computes the DFT of the sequence, and produces unscrambled (the algorithm includes a bit-reversal section) normalized results.
Exceptions:	-

name	
Output:	-
Exceptions:	-

[3.3.4.3.8] Considerations

#	Consideration
1	The algorithm improves on the original taken from *Numerical Recipes in C*, in that: • Checks for N (sequence length) is a power of two • Includes a section for normalizing the results of an inverse DFT • Recasts all loops to `for` constructs, which makes them amenable for loop level parallelization with OpenMP constructs
2	In spite of reduced efficiency, the algorithm uses calls to the data-structure hiding module. This can have the advantage of 'bounds checking' by that module, a fact that the usual FFT algorithms -working at the array level- do not normally check.

[3.3.5] M9: Stream DFT module

NOTE: not implemented in sDFT v. α

[3.3.5.1] USES

[3.3.5.1.1] Imported constants

Constant	Type	Module
N/A	–	–

[3.3.5.1.2] Imported data types

Data Type	–	Module
N/A	–	–

[3.3.5.1.3] Imported access programs

Access Program	–	Module
N/A	–	–

[3.3.5.2] SYNTAX

[3.3.5.2.1] Exported constants

Constant	Type	Value
N/A	–	–

[3.3.5.2.2] Exported data types

Data Type	–	–
N/A	–	–

[3.3.5.2.3] Exported access programs

Access Program	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.5.3] SEMANTICS

[3.3.5.3.1] State variables

State Variable	Type	Comments
N/A	–	–

[3.3.5.3.2] State invariants

#	State Invariant
N/A	–

[3.3.5.3.3] Assumptions

#	Assumption
N/A	–

[3.3.5.3.4] Local types

Local Type	Type	–
N/A	–	–

[3.3.5.3.5] Local constants

Local Constant	Type	Value
N/A	–	–

[3.3.5.3.6] Local functions

Local Function	Inputs	Outputs	Exceptions
N/A	–	–	–

[3.3.5.3.7] Access programs

name	
Transition:	–
Exceptions:	–

name	
Output:	–
Exceptions:	–

[3.3.5.3.8] Considerations

#	Consideration
–	–

[4] REFERENCES

[1] "*Software Design, Automated Testing, and Maintenance – A Practical Approach*"; D. Hoffman (University of Victoria, Canada); P. Strooper (University of Queensland, Australia). © 1995 International Thomson Computer Press. QA76.758.H64_1995; ISBN: 1-850-32206-6

[2] "*Overview of Module Interface Specification*"; S. Smith (Computing and Software Department, McMaster University, Canada); Slides for CAS/CES*741 course: "Development of Scientific Computing Software".

[3] "*Module Interface Specification*"; S. Smith (Computing and Software Department, McMaster University, Canada); February 17, 2009. Slides for CAS/CES*741 course: "Development of Scientific Computing Software".

[4] "*Generic MIS*"; S. Smith (Computing and Software Department, McMaster University, Canada); February 2, 2009. Slides for CAS/CES*741 course: "Development of Scientific Computing Software".

[5] "*Abstract Data Types*"; S. Smith (Computing and Software Department, McMaster University, Canada); January 26, 2009. Slides for CAS/CES*741 course: "Development of Scientific Computing Software".

[6] "*Example MIS*"; S. Smith (Computing and Software Department, McMaster University, Canada); January 28, 2009. Slides for CAS/CES*741 course: "Development of Scientific Computing Software".