# Overview

CMSIS-Stream is a scheduler of DSP/ML software components place in a graph to execute stream-based processing. CMSIS-Stream is open-source, and portable to to any 32bits or 64bits computing

Example of scheduled software components : image and voice codec, data conditioning, motion classifiers, data mixers, etc ...

From the developer point of view, it creates opaque memory interfaces to the input/output streams of a graph, and arranges data are exchanged in the developer’s desired formats. CMSIS-Stream manages the memory mapping, with speed constraints provided by the developer. The memory mapping and the scheduling tables are created off-line. CMSIS-Stream can schedule software components delivered in binary format. It is ready to integrate memory protection between software components.

From the system integrator view, it eases the tuning and the replacement of one component by another one, and it is made to ease processing split with multiprocessors. The stream is described with a graph (a text file) designed with a graphical tool. The development of DSP/ML processing will be possible without need to write code, and allow graph changes and tuning without recompilation.

Table of content

# Graph creation process

Compute Streaming Interface

CMSIS-Stream is scheduling the software components of a graph. The nodes of the graph are software components (“SWC ”) independent of the platform capabilities.

The graph description is a text file (example here) and is the result from the translation made in a GUI tool, using :

• a manifest of the platform (details on processors, memory, peripherals)

• a manifest of each SWC : description of the data formats of the interfaces

CMSIS-Stream is translating the graph description text file to a binary graph structure, with the help of the data in the manifests. This result is placed in shared memory area to all processes and processors.

This shared binary graph structure consists in :

• the linked list of arcs and nodes (the SWC) of the graph

• the arcs descriptors (read and write indexes to circular buffers)

• the memory of the CMSIS-Stream instances scheduled the graph.

• the structure describing the operations at the boundary of the graph (the graph “IOs”)

• registers used to synchronize the different CMSIS-Stream instances, if any

Two entry-points

CMSIS-Stream has two entry-points, one for controling and asking for services, and a second one used as callback for notifications of data transfers :

void arm\_stream (uint32\_t command, PTR\_INT ptr1, PTR\_INT ptr2, PTR\_INT ptr3);

void arm\_stream\_io (uint32\_t fw\_io\_idx, void \*data, uint32\_t length);

Calling sequence

The main CMSIS-Stream instance (the one located in the main process or processor) is called by the application to compute the amount of memory needed to execute the graph : the buffers of the arcs, the SWC instances of the graph, the buffers used for IOs (command “STREAM\_MEMREQ” below).

In a second step, the application provides the memory pointers to the requested memory banks. The CMSIS-Stream instances are now allowed to activate of the IOs at the boundary of the graph, do the memory initialization of all SWC (command “STREAM\_RESET” below).

Finally, the application lets the graph being scheduled by CMSIS-Stream (command “STREAM\_RUN” above).

# SYSTEM MANIFEST (FROM THE INTEGRATOR)

; ------------------------------------------------------------------------------------------------------------

; TOP MANIFEST :

; paths to the files

; processors manifests (memory and architecture)

; IO manifests to use for stream processing

; list of the nodes installed in the platform and their affinities with processors

; ------------------------------------------------------------------------------------------------------------

; list of paths for the included files

3 three file paths

../../stream\_platform/ "" path index 0 is local

../../stream\_platform/windows/manifest/ "" path index 1

../../stream\_nodes/ "" path index 2

; ------------------------------------------------------------------------------------------------------------

; PLATFORM DIGITAL, MIXED-SIGNAL AND IO MANIFESTS - max 32 IOs => iomask

1 procmap\_manifest\_computer.txt path index + file name

; path: path ID

; Manifest manifests file

; FW IO IDX index used in the graph

; ProcCtrl processor affinity bit-field

; ClockDomain provision for ASRC (clock-domain)

; some IO can be alternatively clocked from the system clock (0) or other ones. The system integrator decides

; with this field to manage the flow errors with buffer interpolation (0) or ASRC (other clock domain index)

; The clock domain index is just helping to group and synchronize the data flow per domain.

10 number of IO streams available aligned with struct

;Path Manifest fw\_io\_idx ProcCtrl clock-domain

1 io\_platform\_data\_in\_0.txt 0 1 0

1 io\_platform\_data\_in\_1.txt 1 1 0

1 io\_platform\_analog\_sensor\_0.txt 2 1 0

1 io\_platform\_motion\_in\_0.txt 3 1 0

1 io\_platform\_audio\_in\_0.txt 4 1 0

1 io\_platform\_2d\_in\_0.txt 5 1 0

1 io\_platform\_line\_out\_0.txt 6 1 0

1 io\_platform\_gpio\_out\_0.txt 7 1 0

1 io\_platform\_gpio\_out\_1.txt 8 1 0

1 io\_platform\_data\_out\_0.txt 9 1 0

; ------------------------------------------------------------------------------------------------------------

; SOFTWARE COMPONENTS MANIFESTS

20 nodes path index + file name, in the same order o

; p\_stream\_node node\_entry\_point\_table[NB\_NODE\_ENTRY\_POI

2 swc\_manifest\_none.txt

2 Basic/arm/script/swc\_manifest\_graph\_control.txt

2 Basic/arm/script/swc\_manifest\_script.txt

2 Basic/arm/router/swc\_manifest\_router.txt

2 Basic/arm/converter/swc\_manifest\_converter.txt

2 Basic/arm/amplifier/swc\_manifest\_amplifier.txt

2 Basic/arm/mixer/swc\_manifest\_mixer.txt

2 Audio/arm/filter/swc\_manifest\_filter.txt

2 Audio/arm/detector/swc\_manifest\_detector.txt

2 Basic/arm/rescaler/swc\_manifest\_rescaler.txt

2 Audio/arm/compressor/swc\_manifest\_compressor.txt

2 Audio/arm/decompressor/swc\_manifest\_decompressor.txt

2 Basic/arm/modulator/swc\_manifest\_modulator.txt

2 Basic/arm/demodulator/swc\_manifest\_demodulator.txt

2 Basic/arm/resampler/swc\_manifest\_resampler.txt

2 Basic/arm/qos/swc\_manifest\_qos.txt

2 Basic/arm/split/swc\_manifest\_split.txt

2 image/arm/detector2D/swc\_manifest\_detector2D.txt

2 image/arm/filter2D/swc\_manifest\_filter2D.txt

2 Basic/arm/analysis/swc\_manifest\_analysis.txt

; ------------------------------------------------------------------------------------------------------------

; ------------------------------------------------------------------------------------------------------------

; Processor and memory configuration + default memory mapping

; ------------------------------------------------------------------------------------------------------------

;

1 1 9 number of architectures, number of processors, number of memory banks

; memory banks:

; - ID base offset ID reference above

; - VID virtual ID used in the graph for manual mapping, must stay below 99 for swap controls (see NodeTemplate.txt)

; - S 0=any/1=normal/2=fast/3=critical-Fast,

; - W static0/working1/retention2,

; - P shared0/private1,

; - H DMAmemHW1

; - D Data0/Prog1/Both2

; - Size minimum sizes guaranteed per VID starting from @[ID]+offset below

; - Offset maximum offset from the base offset ID, (continuous banks means = previous size + previous offset)

; the memory is further split in the graph "top\_memory\_mapping" to ease mapping and overlays

; ID VID S W P H D Size offset from offsetID

0 0 1 0 0 0 0 95526 10 VID0=DEFAULT flat memory bank, can overlap with the others

0 1 0 0 0 0 0 65526 10 SRAM0 static, hand tuned memory banks

0 2 0 0 0 0 0 30000 65536 SRAM1 static

0 3 0 1 0 0 0 15000 95536 SRAM1 working at application level

0 4 0 1 0 0 0 256000 262144 DDR working at application level

2 5 3 1 1 0 0 1024 262144 DTCM Private memory of processor 1

1 10 0 2 0 0 0 1024 524288 Retention memory

3 20 0 0 0 0 0 200000 10 Data in Flash

2 8 3 1 1 0 1 16384 0 ITCM Private memory of processor 1

; memory offsets ID used by all processors and physical address seen from the main processor

; 0 h20000000 image of "platform\_specific\_long\_offset(intPtr\_t long\_offset[])"

; 1 h28000000 in stream\_al/platform\_XXXXX.c

; 2 h2C000000 TCM Private memory of processor 1

; 3 h08000000 Internal Flash

;----------------------------------------------------------------------------------------

; all architectures

; all processors (processor IDs >0)

;------------------------------------------------------------

1 1 15 processor ID, boolean "I am the main processor" allowed to boot the graphs

; Bit-field computation firmware extensions, on top of the basic one, embedded in Stream services

; EXT\_SERVICE\_MATH 1, EXT\_SERVICE\_DSPML 2, EXT\_SERVICE\_AUDIO 3, EXT\_SERVICE\_IMAGE 4

;===========================================================================================================================

; TOP GRAPH

; 0: location of the binary graph is all in ram (default)

; 1: keep the graph in Flash and copy in RAM the portion starting from the node linked-list

; 2: keep the graph in Flash and copy in RAM the portion starting from the arc descriptors

graph\_location 0 ; 3: the graph is already in RAM provided by the application

; LSB set means "call the debug script before each naonoAppsRT is called"

; bit 1 (2) set means "call the debug script after each naonoAppsRT is called"

; bit 2 (4) set means "call the debug script at the end of the loop"

; bit 3 (8) set means "call the debug script is called when starting the graph scheduling"

; bit 4 (16) set means "call the debug script is called when returning of the graph scheduling"

debug\_script\_fields 0 ; no bit is set (default) the debug script is not called (default 0)

; 1: return to caller after each SWC calls

; 2: return to caller once all SWC are parsed

scheduler\_return 2 ; 3: return to caller when all SWC are starving (default 3)

allowed\_processors 1 ; bit-field of the processors alloed to execute this graph, (default = 1 main processor)

set\_file\_path 0 ./ ; index and its file path, used for sub graphs and scripts

;============================

; FORMATS

;

format 2 ; index used to start the declaration of a new format

format\_raw\_data 17 ; raw data of this format (17 : S16 is the default)

format\_frame\_length 160 ; frame length in number of bytes (default :1)

format\_interleaving 0 ; 0 means interleaved data, 1 means deinterleaved data by packets of "frame size"

format\_nbchan 1 ; number of channels in the stream (default 1)

format\_time\_stamp 0 ; time-stamp format 0:none, 1:absolute time-stamp, 2:relative time, 3:simple counter

format\_time\_stamp\_size 0 ; 0:16bits 1:32bits 2:64bits (see "STREAM\_TIME16D" for example)

format\_sdomain 0 ; subdomain type (for example see stream\_unit\_physical used for analog sensors)

format\_domain 2 ; IO\_DOMAIN defined in the platform IO manifest (0 means "any")

; general (a)synchronous sensor + rescaling, .. remote data, compressed streams, JSON, SensorThings

; audio\_in microphone, line-in, I2S, PDM RX

; audio\_out line-out, earphone / speaker, PDM TX, I2S,

; gpio\_in generic digital IO , control of relay,

; gpio\_out generic digital IO , control of relay,

; motion accelerometer, combined or not with pressure and gyroscope

; 2d\_in camera sensor

; 2d\_out display, led matrix,

; analog\_in with aging control

; analog\_out D/A, position piezzo, PWM converter

; rtc ticks sent from a programmable timer

; user\_interface\_in button, slider, rotary button

; user\_interface\_out LED, digits, display,

; platform\_3

; platform\_2 platform-specific #2, decoded with callbacks

; platform\_1 platform-specific #1, decoded with callbacks

;

; SPECIFIC OF DOMAINS

;

format\_sampling\_rate 16000.0 ;

format\_audio\_mapping

format\_motion\_mapping

format\_2d\_height

format\_2d\_width

format\_2d\_border

;========================

; MAIN IO

;

; list of HW IOs from "stream\_tools\_files\_manifests\_PLATFORMNAME.txt" + IO arc patched with this IO.

stream\_io 2 ; index used to start the declaration of a new IO

stream\_io\_format 0 ; index to the stream format (Index of the above table) (default #0)

stream\_io\_hwid 0 ; ID of the interface given in "files\_manifests\_computer" (default #0)

stream\_io\_setting1 0 ; setting word32 (SETTINGS\_IOFMT2), the format depends on the IO domain (default #0)

;============================

; MAPPING (TBC)

;

; Split the memory mapping to ease memory overlays between nodes and arcs

; format : original memory bank ID

; new ID to use in the node/arc declarations

; start within the original ID

; length of the new memory bank

;

; ORIGINAL\_ID NEW\_ID START LENGTH

; memory\_mapping 2 100 1024 32700

;

;============================

; DEBUG TRACE

; top\_trace\_node

; ./subgraph\_name/node\_name #instanceID

; \_end\_

;

;==============================

; SUBGRAPHS

;

; subgraph name, used for name mangling of the nodes and arcs

; path ID (set\_file\_path) and file name

; list of indexes from "top\_graph\_interface" (or indexes if we are already in a subgraph)

; memory\_mapping partitions, list of VIDs used in the subgraph

;

subgraph

sub1 ; subgraph name, used for name mangling

3 sub\_graph\_0.txt ; path and file name

5 i16: 0 1 2 3 4 ; 5 streaming interfaces data\_in\_0, data\_out\_0 ..

3 i16: 0 0 0 ; 3 partitions here assigned to VID0 : fast-working slow-working slow-static

;==============================

; NODE IN GRAPHS

;

node <node\_name> <instance\_index> ; instance index in the (sub)graph, the compilation listing gives the physical index

node\_preset 1 ; parameter preset used at boot time, default = #0

node\_malloc\_E 12 ; "E" parameter used in "Memory Size Bytes", default = #0

node\_map\_hwblock 2 3 ; index of the memory block "node\_mem" and the VID indexes from "procmap\_manifest\_xxxx.txt"

; where to map it. Default VID's is 0.

node\_map\_copy 2 3 ; copy the indexed "node\_mem" to VID 3 (faster memory) before run

node\_map\_swap 2 3 ; swap the indexed "node\_mem" to VID 3 (faster memory) before run, and restored after

node\_map\_proc 0 ; execute this nanoApp on this processor (0: any possible, default)

node\_map\_arch 0 ; execute this nanoApp on this architecture (0: any possible, default)

node\_map\_rtos 0 ; execute this nanoApp on this thread index (0: any possible, default)

node\_map\_verbose 0 ; level of debug trace, default = #0

node\_script <0..127> ; index of the script to call before and after execution of this node

node\_parameters ; node parameters example (default : no parameter)

; Set\_parameter : the array of parameters starts on 32bits-aligned addresses

; The programmer must arrange the data are aligned with respect to the way parameters are read in

; the nanoApp (using pointers to 8/16/32bits fields).

1 i8; 0 ; TAG= 0 "load all parameters"

7 i8; 2 3 4 5 6 7 8 ; parameters

include 1 filter\_parameters.txt ; path + text file-name using parameter syntax

\_end\_

;==============================

; SCRIPTING NODE IN GRAPHS (SCHEDULED LIKE OTHER NODES)

;

; Checks if the data it needs is available and returns to the scheduler

; Its single arc (TX) is always empty

node arm\_stream\_script 1 ; instance index of arm\_stream\_script

script\_registers 2 ; numer of registers used in this script , default 2

script\_pointers 2 ; numer of pointers used in this script , default 2

script\_stack 12 ; size of the stack in word64 (default = 0)

script\_mem\_shared 1 ; Is it a private RAM(0) or can it be shared with other scripts(1)

script\_mem\_map 0 ; Memory mapping to VID #0 (default)

; this declaration creates the transmit arc of the script-node pointing to the stack/buffer area

script\_code

2 h16; 2002 0001 ; movi int16 r0 1

1 h16; e810 ; equ r1,r0

1 h16; 0381 ; ccallsys 1

1 h16; C000 ; ret

\_end\_

script\_assembler ; start of assembler language (@@@ TBD)

pshc int8 1

gtr

cjmp #1

pshc int16

cals readparam

labl #1

ret

\_end\_

node\_parameters <ID2> ; node parameters and index to let the code addressing it

; Set\_parameter : the array of parameters starts on 32bits-aligned addresses

; The programmer must arrange the data are aligned with respect to the way parameters are read in

; the nanoApp (using pointers to 8/16/32bits fields).

1 i8; 0 ; TAG= 0 "load all parameters"

7 i8; 2 3 4 5 6 7 8 ; parameters

node\_parameters <ID2> ;

include 1 binary\_code.txt

\_end\_

\_end\_

;==============================

; COMMON SCRIPTS IN GRAPHS (AND SCRIPT ID IN THE NODE HEADER "SCRIPT\_LW0")

;

; script instance #0 is the "main script" of the subgraph

script 0 ; index of the script, to be mapped to an index in the graph compiler

script\_registers 2 ; same as arm\_stream\_script

script\_pointers 2 ;

script\_stack 12 ;

script\_mem\_shared 1 ;

script\_mem\_map 0 ;

script\_code ; start of byte-codes of the script

2 h16; 2002 0001 ;

1 f64; 3.14159265359 ;

1 h16; e810 ;

1 i8 ; 0 ; parameters embedded in the code and addressed with Labels

7 i8 ; 2 3 4 5 6 7 8 ;

\_end\_ ;

;====================================

; ARCS IN GRAPHS

;----ARC CONNECTED TO GRAPH INTERFACE

; in a subgraph the IDX interfaces are sequential 1,2,3.. and documented like function parameters

; in the main graph the "top\_graph\_interface" have the indexes to use use in the first column

; IO-ID NAME INST IO FORMAT

arc\_input 1 node\_name 2 0 0 ; input arc index #1 connected to "node\_name" instance #2 and its arc index #0, Format #0

..

; IO-ID NAME INST IO FORMAT

arc\_output 2 node\_name 3 1 0 ; output arc index #2 connected to "node\_name" instance #3 and its arc index #1, Format #0

..

;----ARC CONNECTION BETWEEN TWO NANOAPPS

; NAME INST IO FMT NAME INST IO FMT

arc node1 1 2 0 node2 3 4 1 ; arc between node1 instance #1 arc index #2, producer format #0 to

; node2 instance #3 and its arc index #4, consumer format #1

..

;----ARC PARAMETERS

arc\_flow\_error 1 ; #1 do something depending on domain when a flow error occurs, default #0 (no interpolation)

arc\_debug\_cmd 1 ; debug action "ARC\_INCREMENT\_REG", default = #0 (no debug)

arc\_debug\_reg 3 ; index of the 64bits result, default = #0

arc\_debug\_page 0 ; debug registers base address + 64bits x 16 registers = 32 word32 / page, default = #0

arc\_flush 0 ; control of register "MPFLUSH\_ARCW1" : forced flush of data in MProcessing and shared tasks

arc\_extend\_addr 1 ; address range extension-mode of the arc descriptor "EXTEND\_ARCW2" for large NN models, default = #0 (no extension)

arc\_map\_hwblock 0 ; mapping VID index from "procmap\_manifest\_xxxx.txt" to map the buffer, default = #0 (VID0)

arc\_jitter\_ctrl 1.5 ; factor to apply to the minimum size between the producer and the consumer, default = 1.0 (no jitter)

;====================================

;----------------------------------------------------------------------------------------

; List of nodes

;----------------------------------------------------------------------------------------

;

1 arm\_stream\_graph\_control, /\* scheduler control : lock, bypass, loop, if-then \*/

2 arm\_stream\_script, /\* byte-code interpreter, index "arm\_stream\_script\_INDEX" \*/

3 arm\_stream\_router, /\* copy input arcs and subchannel and output arcs and subchannels \*/

4 arm\_stream\_converter, /\* raw data format converter \*/

5 arm\_stream\_amplifier, /\* amplifier mute and un-mute with ramp and delay control \*/

6 arm\_stream\_mixer, /\* multichannel mixer with mute/unmute and ramp control \*/

7 arm\_stream\_filter, /\* cascade of DF1 filters \*/

8 arm\_stream\_detector, /\* estimates peaks/floor of the mono input and triggers a flag on high SNR \*/

9 arm\_stream\_rescaler, /\* raw data values remapping using "interp1" \*/

10 arm\_stream\_compressor, /\* raw data compression with adaptive prediction \*/

11 arm\_stream\_decompressor, /\* raw data decompression \*/

12 arm\_stream\_modulator, /\* signal generator with modulation \*/

13 arm\_stream\_demodulator, /\* signal demodulator, frequency estimator \*/

14 arm\_stream\_resampler, /\* asynchronous sample-rate converter \*/

15 arm\_stream\_qos, /\* raw data interpolator with synchronization to one HQoS stream \*/

16 arm\_stream\_split, /\* let a buffer be used by several nodes \*/

17 arm\_stream\_detector2D, /\* activity detection, pattern detection \*/

18 arm\_stream\_filter2D, /\* Filter, rescale, rotate, exposure compensation \*/

19 arm\_stream\_analysis, /\* arm\_stream\_analysis, \*/

;----------------------------------------------------------------------------------------

;1. arm\_stream\_graph\_control

;----------------------------------------------------------------------------------------

; mix of scheduler controls : lock, bypass, loop, if-then, UART remote controler

;

; lock the next N nodes to let them be executed on the same processor

; or have scripts placed before/after processing (ASRC drift control for example)

;

; conditional execution of some nodes from use-case value (if-then) [debug\_reg + page\_index]

; using a stack and the scripting language

;

; minimal serial interface protocol with stream header detection,

; 6b->8b conversion, scheduler call to set parameters, change use-case,

; reload a new graph and optionnaly restart the graph using application\_callbacks[]

;

; Use-case : dispatch debug commands and allow a new graph to be downloaded

;

node

arm\_stream\_graph\_control 0 ; instance index 0

script\_stack 12 ; size of the stack, default = 0

map\_verbose 0 ; level of debug trace, default = #0

; ; codes

1 h16; 0211 PSHC INT8 S0=2 > R0?

1 h16; AD10 GTR

1 h16; 6111 JMPC #1 2 > R0 : exit

1 h16; 0211 PSHC INT16 ARC1

1 h16; 0011

1 h16; 7111 CALS READPARAM (param\_id, /mySWC /toArc, arcID)

1 h16; B111 LABL #1

1 h16; 9011 OPRC RETS

\_end\_ end of byte codes

\_end\_

;----------------------------------------------------------------------------------------

; 2 arm\_stream\_script

;----------------------------------------------------------------------------------------

; scripts are special nodes, the byte code in indexed in the SCRIPTS section of the graph

; data RAM is placed in a shared arc for all script (instance reg+stack parameters)

; constants are placed after the byte-codes

; The default memory configuration is "shared": the TX\_FIFO descriptors associated with

; the script are sharing the same memory buffer (BASEIDXOFFSCRARCW0). To have

; individual static memory associated to a script the "private memory flag" must be set.

;

node

arm\_stream\_script 0 ; instance index 0

script\_stack 12 ; size of the stack in word64 (default = 0)

script\_minimum 1 ; reduced number of registers configuration (default = 0)

; minimum state memory = (2 registers + 1 pointer + 0 special) x 8 Bytes = 24 Bytes

; standard state memory = (6 registers + 6 pointers + 3 special) x 8 Bytes = 120 Bytes

script\_mem\_shared 1 ; Is it a private RAM(0) or can it be shared with other scripts(1)

script\_mem\_map 0 ; Memory mapping to VID #0 (default)

script\_assembler ; start of assembler language (@@@ TBD)

PSHC INT8 1

GTR

CJMP #1

PSHC INT16

CALS READPARAM

LABL #1

RET

\_end\_

\_end\_ end of byte codes

parameters 14 ; preloaded parameters, label 14, for router nanoApp

2 i8; 1 2 ; preset=1 load only the second index (Tag=2)

4 i16; 0 0 2 0 ; move arc0,chan0, to arc2,chan0

4 i16; 0 0 2 0 ; move arc0,chan0, to arc2,chan0

\_end\_

parameters 98 ; preloaded parameters, label 98, for router nanoApp

2 i8; 1 2 ; preset=1 load only the second index (Tag=2)

4 i16; 0 0 2 0 ; move arc0,chan0, to arc2,chan0

4 i16; 0 0 2 0 ; move arc0,chan0, to arc2,chan0

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 3 arm\_stream\_router

;----------------------------------------------------------------------------------------

; I/O streams have the same RAW format size, if time-stamped then only with frame size of 1

;

; table of routing : a list of 16bits words (source 8b + destination 8b)

; format: router arc index (3b) up to 8 arcs

; sub-channel index -1 (5b) up to 32 channels

;

; example with the router\_2x2 used to merge 2 stereo input in a 4-channels output

;

node

arm\_stream\_router i

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

4 i16; 0 1 2 1 move arc0,chan1, to arc2,chan1

4 i16; 1 0 2 2 move arc1,chan0, to arc2,chan2

4 i16; 1 1 2 3 move arc1,chan1, to arc2,chan3

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 4 arm\_stream\_converter

;----------------------------------------------------------------------------------------

; Operation : convert input format to the format of the output arc.

; For 2D data this is image rescaling, sampling-rate is not used

; Converter do not check Framesize

; Converter is changing RAW data and Units, interleaving, time-stamp format, number of channels

; and sampling-rate (subroutines shared with arm\_stream\_resampler)

;

; Operations :

; - check the output buffer size with the new format and sampling rate

; new size = frame-input duration x new data rate/s

; - process nchan, interleaving and time-stamps (as 1..8 extra channels for time64)

; routing input buffer to a scratch buffer, merge with raw data change

; for images: use an intermediate format separating color/monochrome and RGB/YUV

; nchan changes are done with masks on channels for the averaging

; - 2D resize and 1D resampling : filter + interpolator

; TBD : creating a synchronous stream from an input using time-stamps (IMU data for ex.)

;

; Parameters : domains of operations to correctly interpret the Word2 of the FORMAT (SETTINGS\_IOFMT2)

;

arm\_stream\_converter (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 4 5 index of the input and output formats

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 5 arm\_stream\_amplifier

;----------------------------------------------------------------------------------------

; Operation : control of the amplitude of the input stream with controlled time of ramp-up/ramp-down.

; The gain control “mute” is used to store the current gain setting, being reloaded with the command “unmute”

; Option : either the same gain/controls for all channels or list of parameters for each channel

;

; Parameters : new gain/mute/unmute, ramp-up/down slope, delay before starting the slope.

; Use-cases :

; Features : adaptive gain control (compressor, expander, AGC) under a script control with energy polling

; Metadata features : "saturation occured" "energy"

; Mixed-Signal glitches : remove the first seconds of an IR sensor until it was self-calibrated (same for audio Class-D)

;

;

; parameters of amplifier (variable size):

; TAG\_CMD = 1, uint8\_t, 1st-order shifter slope time (as stream\_mixer, 0..75k samples)

; TAG\_CMD = 2, uint16\_t, desired gain FP\_8m4e, 0dB=0x0805

; TAG\_CMD = 3, uint8\_t, set/reset mute state

; TAG\_CMD = 4, uint16\_t, delay before applying unmute, in samples

; TAG\_CMD = 5, uint16\_t, delay before applying mute, in samples

;

node

arm\_stream\_amplifier (i)

arm\_stream\_router

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 1 load only rising/falling coefficient slope

1 h16; 805 gain -100dB .. +36dB (+/- 1%)

1 i8; 0 muted state

2 i16; 0 0 delay-up/down

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 6 arm\_stream\_mixer

;----------------------------------------------------------------------------------------

; Operation : receives several mono or multichannel streams and produces one output arc.

; Mixer manages up to 8 multichannels input arcs, the number is given at STREAM\_RESET stage.

;

; Parameters : input arcs names, their associated gain and time-constant in [ms], name of the

; output arc and an extra mixing gain

; + index of the HQoS input arc to use for the frame size to use, if the other arcs

; have flow-errors it will be managed with their respective OVERFLRD\_ARCW2 index

;

; Synchronization operation : arranges multiple input streams are delivered at the same time.

; Use-case: multi-channel audio stream, each channel is processed with independent

; signal processing feature. The problem is to avoid (on MP devices) one channel to

; be delivered to the final mixer ahead and desynchronized from the others

; (should be OK when the arcs have the sma buffer size / framesize)

;

; preset :

; #1 (default) : mixer-4 with all arcs unmuted with gain = 0dB, slopes of ~1000 samples

; #2 (shut-down) : mixer-4 with all arcs with gain = -96dB, slopes of ~100 samples

; >2 : number of input channels

;

; parameters of mixer (variable size):

; - slopes of rising and falling gains, identical to all channels

; slope coefficient = 0..15 (iir\_coef = 1-1/2^coef = 0 .. 0.99)

; Convergence time to 90% of the target in samples:

; slope nb of samples to converge

; 0 0

; 1 3

; 2 8

; 3 17

; 4 36

; 5 73

; 6 146

; 7 294

; 8 588

; 9 1178

; 10 2357

; 11 4715

; 12 9430

; 13 18862

; 14 37724

; 15 75450

; convergence in samples = abs(round(1./abs(log10(1-1./2.^[0:15])'))

;

; - output mixer gain format FP

;

; format to be packed in 32bits words:

; input arc index (4b) up to 15 arcs

; sub-channel index -1 (5b) up to 32 channels per arc of the input arc

; sub-channel index -1 (5b) up to 32 channels per arc of the output arc

; gain (16b) format FP

; muted (1b) 1 = "muted"

;

node

arm\_stream\_mixer (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

; Example: 4 input one mono, one stereo, output is stereo

parameters <optional label for scripts>

1 i8; 0 load all parameters

2 i8; 6 6 rising/falling slope on all arcs

1 h32; 807 -12dB output gain in FP\_8m4e format;

4 i16; 0 0 0 arcID0 inputChan0 outputChan0

1 h32; 805 0dB gain; mono mixed on the left output

4 i16; 0 0 1 arcID0 inputChan0 outputChan1

1 h32; 805 0dB gain; mono mixed on the right output

4 i16; 1 0 0 arcID1 inputChan0 outputChan0

1 h32; 805 0dB gain; stereo input left to the left

4 i16; 1 1 1 arcID1 inputChan0 outputChan1

1 h32; 803 +12dB gain; stereo input right to the right

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 7 arm\_stream\_filter

;----------------------------------------------------------------------------------------

; Operation : receives one multichannel stream and produces one filtered multichannel stream.

; Parameters : biquad filters coefficients used in cascade. Implementation is 2 Biquads max.

; (see www.w3.org/TR/audio-eq-cookbook)

; Option : either the same coefficients for all channels or list of coefficients for each channel

;

; presets:

; #1 : bypass

; #2 : LPF fc=fs/4

; #3 : DC-filter (use-case: audio, XYZ gravity compensation/estimation)

;

; parameter of filter :

; - number of biquads in cascade (1 or 2)

; - coefficients in FP32

;

; Simple format : S + Preset + Use-case

; Extended : X + Preset, pre/post process script, arch, proc, priority, verbose level

; followed by the list of VID virtual memory banks defined in "platform\_manifest\_computer.txt"

; of each memory bank defined in swc\_manifest\_filter.txt

; + one segment for the code (0=default VID segment: Flash/no Swap)

;

node

arm\_stream\_filter 0 node subroutine name + instance ID

node\_preset 1 ; parameter preset used at boot time, default = #0

node\_map\_hwblock 0 0 ; list of "nb\_mem\_block" VID indexes of "procmap\_manifest\_xxxx.txt" where to map the allocated memory

; default = #0

node\_map\_verbose 1 ; level #1 of debug trace

parameters optional\_label

1 i8; 0 0="load all parameters"

1 u8; 2 Two biquads

1 i8; 0 postShift

5 f32; 0.284277f 0.455582f 0.284277f 0.780535f -0.340176f b0/b1/b2/-a1/-a2 ellip(4, 1, 40, 3600/8000, 'low')

5 f32; 0.284277f 0.175059f 0.284277f 0.284669f -0.811514f

; or \_include 1 arm\_stream\_filter\_parameters\_x.txt (path + file-name)

\_end\_

\_end\_

;

;

;----------------------------------------------------------------------------------------

; 8 arm\_stream\_detector

;----------------------------------------------------------------------------------------

; Operation : provides a boolean output stream from the detection of a rising

; edge above a tunable signal to noise ratio.

; A tunable delay allows to maintain the boolean value for a minimum amount of time

; Use-case example 1: debouncing analog input and LED / user-interface.

; Use-case example 2: IMU and voice activity detection (VAD)

; Parameters : time-constant to gate the output, sensitivity of the use-case

;

; presets control

; #1 : no HPF pre-filtering, fast and high sensitivity detection (button debouncing)

; #2 : VAD with HPF pre-filtering, time constants tuned for ~10kHz

; #3 : VAD with HPF pre-filtering, time constants tuned for ~44.1kHz

; #4 : IMU detector : HPF, slow reaction time constants

; #5 : IMU detector : HPF, fast reaction time constants

;

; Metadata information can be extracted with the command "TAG\_CMD" from parameter-read:

; 0 read the floor noise level

; 1 read the current signal peak

; 2 read the signal to noise ratio

;

; Simple format : S + Preset

; Extended : X + Preset, pre/post process script, arch, proc, priority, verbose level

; followed by the list of VID virtual memory banks defined in "platform\_manifest\_computer.txt"

; of each memory bank defined in swc\_manifest\_filter.txt

; + one segment for the code (0=default VID segment: Flash/no Swap)

;

node

arm\_stream\_detector 0 node name + instance ID

preset 1 ; parameter preset used at boot time, default = #0

map\_verbose 1 ; level #1 of debug trace = #0

parameters <optional label for scripts>

2 i8; 0 0="load all parameters"

8; i8; 1 2 3 4 5 6 7 8 the 8 bytes of "struct detector\_parameters"

\_end\_

\_end\_

;

;

;----------------------------------------------------------------------------------------

; 9 arm\_stream\_rescaler

;----------------------------------------------------------------------------------------

; Operation : applies vq = interp1(x,v,xq)

; Following https://fr.mathworks.com/help/matlab/ref/interp1.html

; linear of polynomial interpolation (implementation)

; Parameters : X,V vectors, size max = 32 points

;

; no preset ('0')

;

node

arm\_stream\_rescaler (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 load all parameters

; input output

2; f32; -1 1

2; f32; 0 0 ; this table creates the abs(x) conversion

2; f32; 1 1

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 10 arm\_stream\_compressor

;----------------------------------------------------------------------------------------

; Operation : wave compression using IMADPCM(4bits/sample)

; Parameters : coding scheme

;

; presets (provision codes):

; #1 : encoder IMADPCM

; #2 : encoder LPC

; #3 : encoder CVSD for BT speech

; #4 : encoder LC3

; #5 : encoder SBC

; #6 : encoder mSBC

; #7 : encoder OPUS Silk

; #7 : encoder MP3

; #8 : encoder MPEG-4 aacPlus v2

; #9 : encoder OPUS CELT

;

node

arm\_stream\_compressor 0

S 1 simple format

X 1 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

4; i32; 0 0 0 0 provision for extra parameters in other codecs

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 11 arm\_stream\_decompressor

;----------------------------------------------------------------------------------------

; Operation : wave decompression of MONO encoded data

; Parameters : coding scheme and a block of 16 parameter bytes for codecs

;

; WARNING : if the output format can change (mono/stereo, sampling-rate, ..)

; the variation is detected by the node and reported to the scheduler with

; "STREAM\_SERVICE\_INTERNAL\_FORMAT\_UPDATE", the "uint32\_t \*all\_formats" must be

; mapped in a RAM for dynamic updates with "COPY\_CONF\_GRAPH0\_COPY\_ALL\_IN\_RAM"

;

; Example of data to share with the application

; outputFormat: AndroidOutputFormat.MPEG\_4,

; audioEncoder: AndroidAudioEncoder.AAC,

; sampleRate: 44100,

; numberOfChannels: 2,

; bitRate: 128000,

;

; presets provision

; #1 : decoder IMADPCM

; #2 : decoder LPC

; #3 : MIDI player / tone sequencer

; #4 : decoder CVSD for BT speech

; #5 : decoder LC3

; #6 : decoder SBC

; #7 : decoder mSBC

; #7 : decoder OPUS Silk

; #8 : decoder MP3

; #9 : decoder MPEG-4 aacPlus v2

; #10: decoder OPUS CELT

; #11: decoder JPEG

;

node

arm\_stream\_decompressor (i)

S 1 simple format

X 1 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

1 i8; 1 state = off (generates zeros) / on with ramp-up

4; i32; 0 0 0 0 provision for extra parameters in other codecs

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 12 arm\_stream\_modulator

;----------------------------------------------------------------------------------------

; Operation : sine, noise, square, saw tooth with amplitude or frequency modulation

; use-case : ring modulator, sweep generation with a cascade of a ramp generator and

; a frequency modulator

;

; Parameters types and Tags:

; u8 #1 wave type : 1=sine 2=square 3=white noise 4=pink noise

; 5=sawtooth 6=triangle 7=pulse

; 8=prerecorded pattern playback from arc

; 9=sigma-delta with OSR control for audio on PWM ports and 8b DAC

; 10=PWM 11=ramp 12=step

;

; parameter format for wave generation:

; u16 #2 linear amplitude, format UQ16 0dB .. -96dB

; u16 #3 amplitude offset, format Q15 [-1 .. +1]

; f32 #4 wave frequency [Hz] [0.1uHz .. 250kHz], 0Hz translates to DC level.

; s16 #5 starting phase, format S15 [-pi .. +pi]/pi

; u8 #6 modulation type, 0:amplitude, 1:frequency (reserved 2:FSK, ..)

; u8 #7 modulation, 0:none 1=from arc bit stream, 2=from arc q15 words

; u16 #8 modulation index (a) and offset (b), in q15 as fraction of frequency or amplitude

; modulation y=ax+b, x=input data.

; f32 #9 modulation frequency [Hz] separating two data bits/samples from the arc

;

node

arm\_stream\_modulator (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

1 u8; 1 sinewave

2 h16; FFFF 0 full-scale, no offset

1 f32; 1200 1200Hz

1 s16; 0 initial phase

2 u8; 1 1 frequency modulation from bit-stream

2 h16; 8000 0 full amplitude modulation with sign inversion of the bit-stream

1 f32; 300 300Hz modulation => (900Hz .. 1500Hz modulation)

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 13 arm\_stream\_demodulator

;----------------------------------------------------------------------------------------

; Operation : decode a bit-stream from analog data. Use-case: IR decoder, CAN/UART on

; SPI/I2S audio.

; Parameters : clock and parity setting or let the algorithm discover the frame setting after

; some time. https://en.wikipedia.org/wiki/Universal\_asynchronous\_receiver-transmitter

;

; presets control :

; #1 .. 10: provision for demodulators

;

; Metadata information can be extracted with the command "parameter-read":

; TAG\_CMD = 1 read the signal amplitude

; TAG\_CMD = 2 read the signal to noise ratio

;

node

arm\_stream\_demodulator (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 14 arm\_stream\_resampler

;----------------------------------------------------------------------------------------

; Operation : convert multichannel input data rate to the rate of the output arcs

; + synchronous rate convertion with +/- 100ppm adjustment

; + conversion of time-stamp asynchronous to synchronous stream

;

; SSRC synchronous rate converter, FS in/out are exchanged during STREAM\_RESET

; ASRC asynchronous rate converter using time-stamps (in) to synchronous FS (out)

; pre-LP-filtering tuned from Fout/Fin ratio + Lagrange polynomial interpolator

;

; drift compensation managed with STREAM\_SET\_PARAMETER command:

; TAG\_CMD = 0 to stop drift compensation

; TAG\_CMD = 1 to continuously apply a drift compensation of +100ppm

; TAG\_CMD = 2 to continuously apply a drift compensation of -100ppm

;

; The script associated to the node is used to read the in/out arcs filling state

; to tune the drift control

;

node

arm\_stream\_resampler (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 15 arm\_stream\_qos

;----------------------------------------------------------------------------------------

;

; Operation : align the content of arc buffers with respect to an high-priority one. (HQOS)

; Parameters : the list of arcs and the one taken as time-alignment reference

;

node

arm\_stream\_qos (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

4 i16; 0 1 2 1 move arc0,chan1, to arc2,chan1

4 i16; 1 0 2 2 move arc1,chan0, to arc2,chan2

4 i16; 1 1 2 3 move arc1,chan1, to arc2,chan3

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 16 arm\_stream\_split

;----------------------------------------------------------------------------------------

;

; Operation : share the input arc to several output arcs, to let several nodes have

; access to the same buffer without data duplication. Consolidate the output read

; indexes before updating the input arc read index.

; There is no priority managed between output arcs.

; The base, size and write index of the output arcs are identical to the input arc

;

; Parameters : the list of arcs , no memory for instances is needed

;

node

arm\_stream\_split (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

4 i16; 0 1 2 1 move arc0,chan1, to arc2,chan1

4 i16; 1 0 2 2 move arc1,chan0, to arc2,chan2

4 i16; 1 1 2 3 move arc1,chan1, to arc2,chan3

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 17 arm\_stream\_detector2D MOTION DETECTOR

;----------------------------------------------------------------------------------------

;

; Motion and pattern detector (lines)

; Operation : detection of movement(s) and computation of the movement map

; Parameters : sensitivity, floor-noise smoothing factors

; Metadata : decimated map of movement detection

;

; presets control

; #1 : no HPF pre-filtering, fast and high sensitivity detection (button debouncing)

;

; preset parameter :

;

; Metadata information can be extracted with the command "TAG\_CMD" from parameter-read:

; 0 read the floor noise level

; 1 read the current signal peak

; 2 read the signal to noise ratio

;

node

arm\_stream\_detector2D (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

4 i16; 0 1 2 1 move arc0,chan1, to arc2,chan1

4 i16; 1 0 2 2 move arc1,chan0, to arc2,chan2

4 i16; 1 1 2 3 move arc1,chan1, to arc2,chan3

\_end\_

\_end\_

;

;

;----------------------------------------------------------------------------------------

; 18 arm\_stream\_filter2D

;----------------------------------------------------------------------------------------

; Filter, rescale, rotate, exposure compensation

; Operation : 2D filters

; Parameters : convolutional kernels, decimation, distortion, color mapping/effect

;

; presets:

; #1 : bypass

;

; parameter of filter :

;

;

node

arm\_stream\_filter2D (i)

S 0 simple format

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

2 i8; 2 2 nb input/output arcs

4 i16; 0 0 2 0 move arc0,chan0, to arc2,chan0

4 i16; 0 1 2 1 move arc0,chan1, to arc2,chan1

4 i16; 1 0 2 2 move arc1,chan0, to arc2,chan2

4 i16; 1 1 2 3 move arc1,chan1, to arc2,chan3

\_end\_

\_end\_

;

;----------------------------------------------------------------------------------------

; 19 arm\_stream\_analysis

;----------------------------------------------------------------------------------------

;

; Operation : spectrum and energy analysis

; Use-case : debug with script control

; results are used with "read\_parameter"

;

; presets :

; #1 : frequency estimator, counting the number of periods

; Operation : input signal frequency estimator with second-order autoregression

; Use-case example: measure 50Hz from power lines on the three phases (a -1% deviation means "power-down").

; Parameters : none. output data estimate sampling rate is provided at reset

; #2 : energy estimation and DC offset

; #3 : energy analysis on 3 frequency points (Goertzel)

; #4 : full spectrum 64 points in decibels

; #5 : gyroscope rotation drifts on XYZ

;

arm\_stream\_analysis (i)

S 0 simple format = preset

X 0 2 0 0 0 0 4 preset VAD

0 0 list of VIDs

parameters <optional label for scripts>

1 i8; 0 0=load all parameters

\_end\_

\_end\_

;----------------------------------------------------------------------------------------