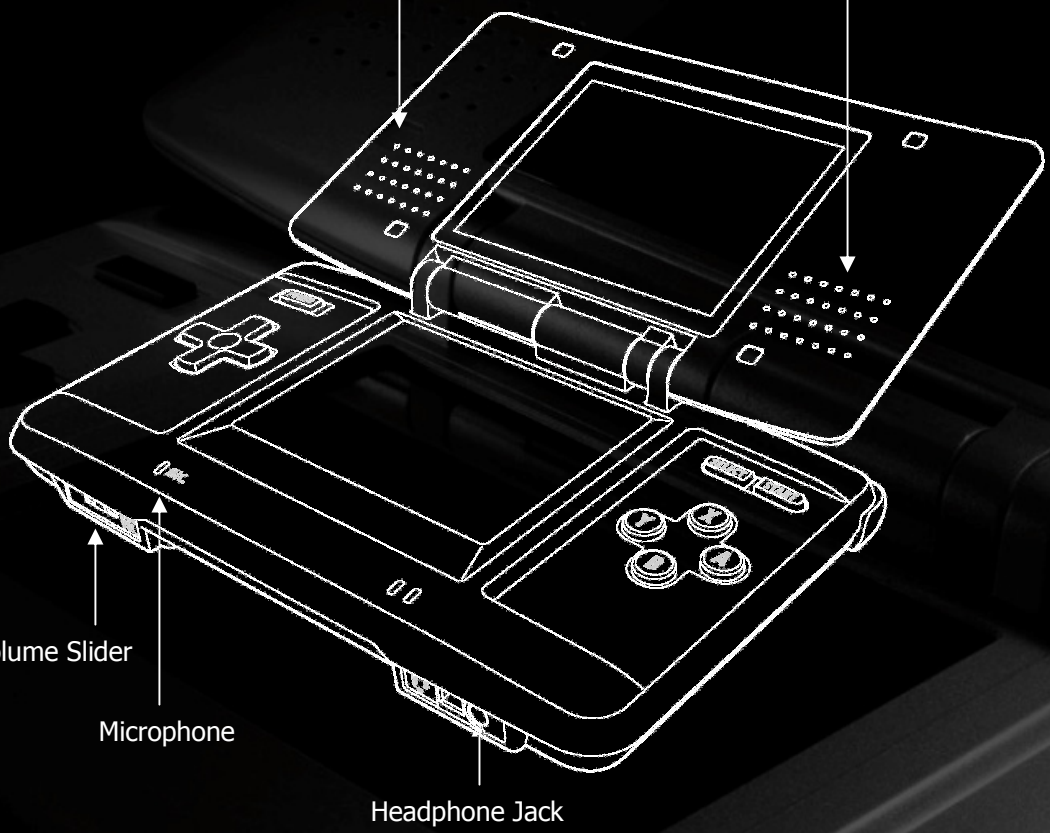
Nintendo DS Audio

Henry Cheng
Software Development Support Group
Nintendo of America

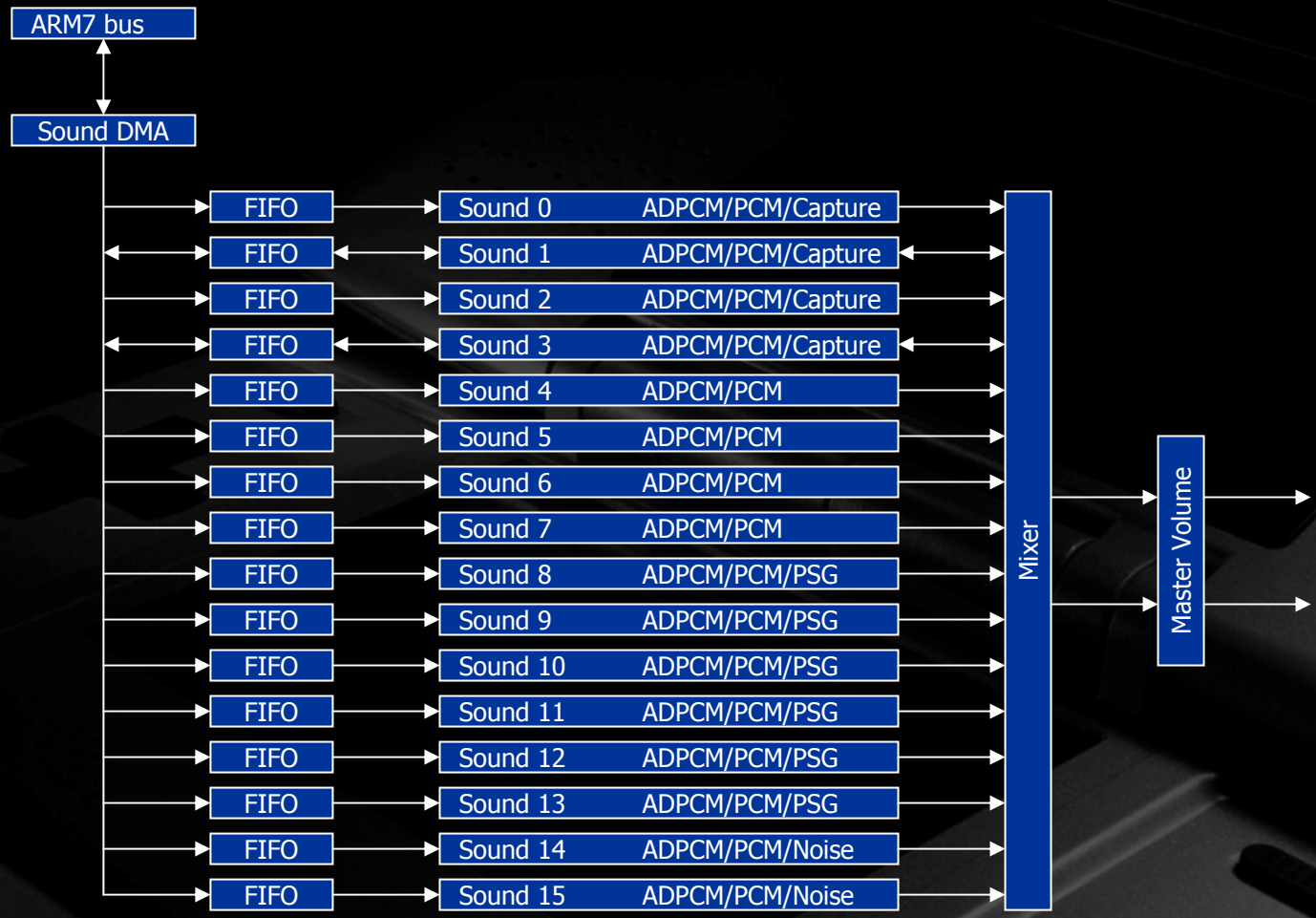
NINTENDO DS
DEVELOPERS CONFERENCE 2004

DS Audio Hardware



NINTENDO DS
DEVELOPERS CONFERENCE 2004

Sound Hardware



Sound Hardware

Sound Hardware

Sound Hardware

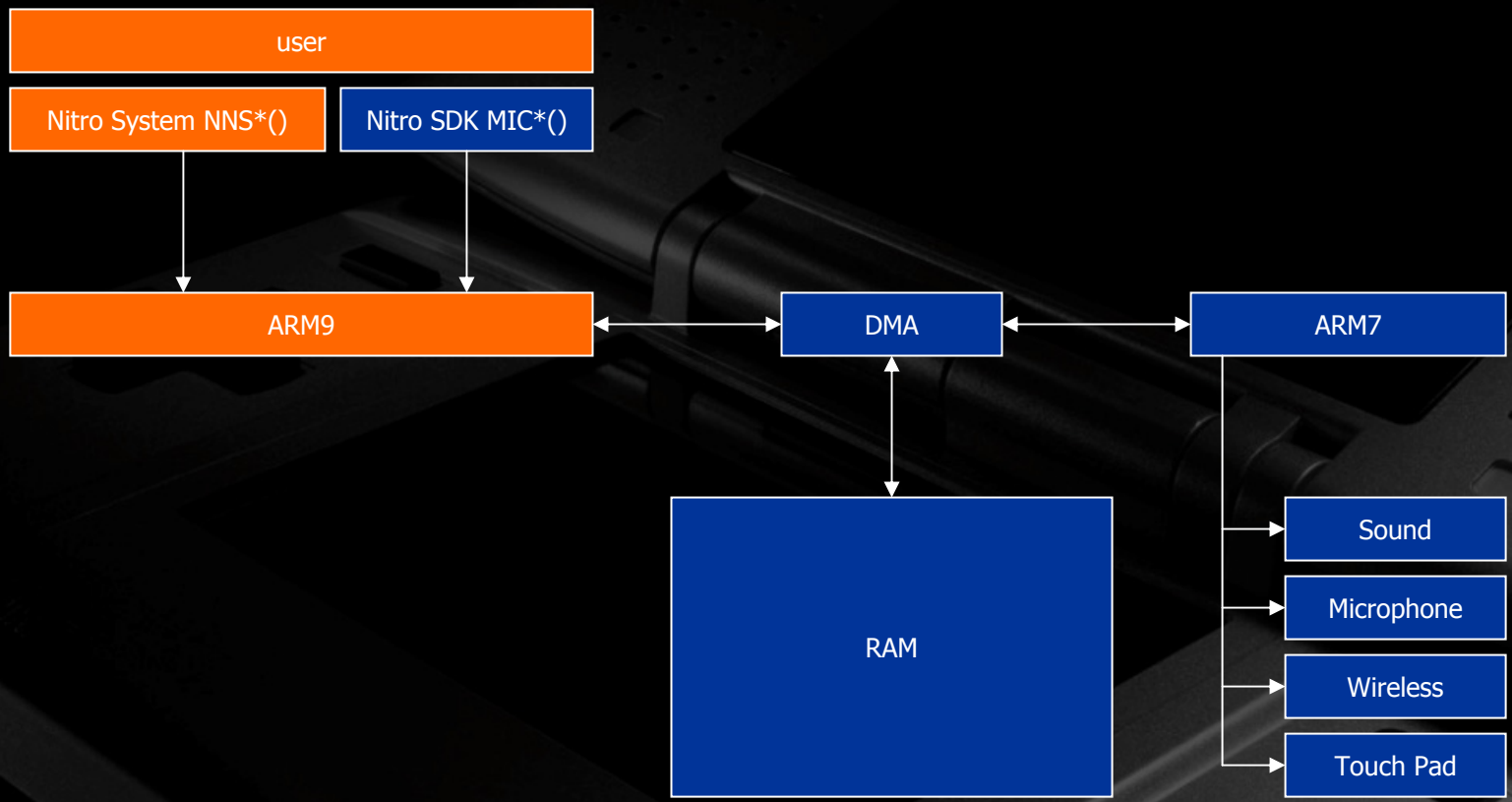
Microphone Hardware

Microphone Hardware

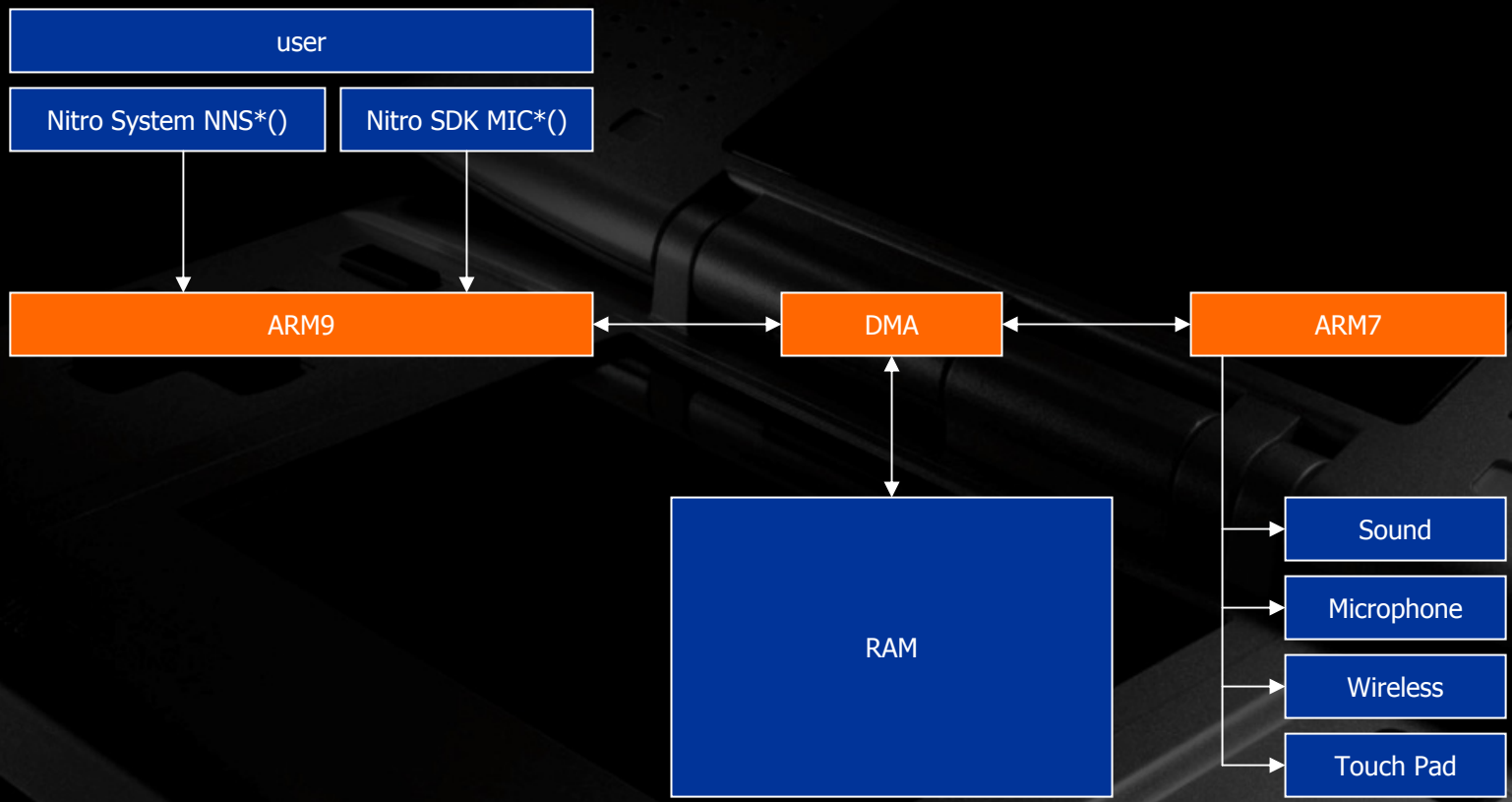
Microphone Hardware

Microphone Hardware

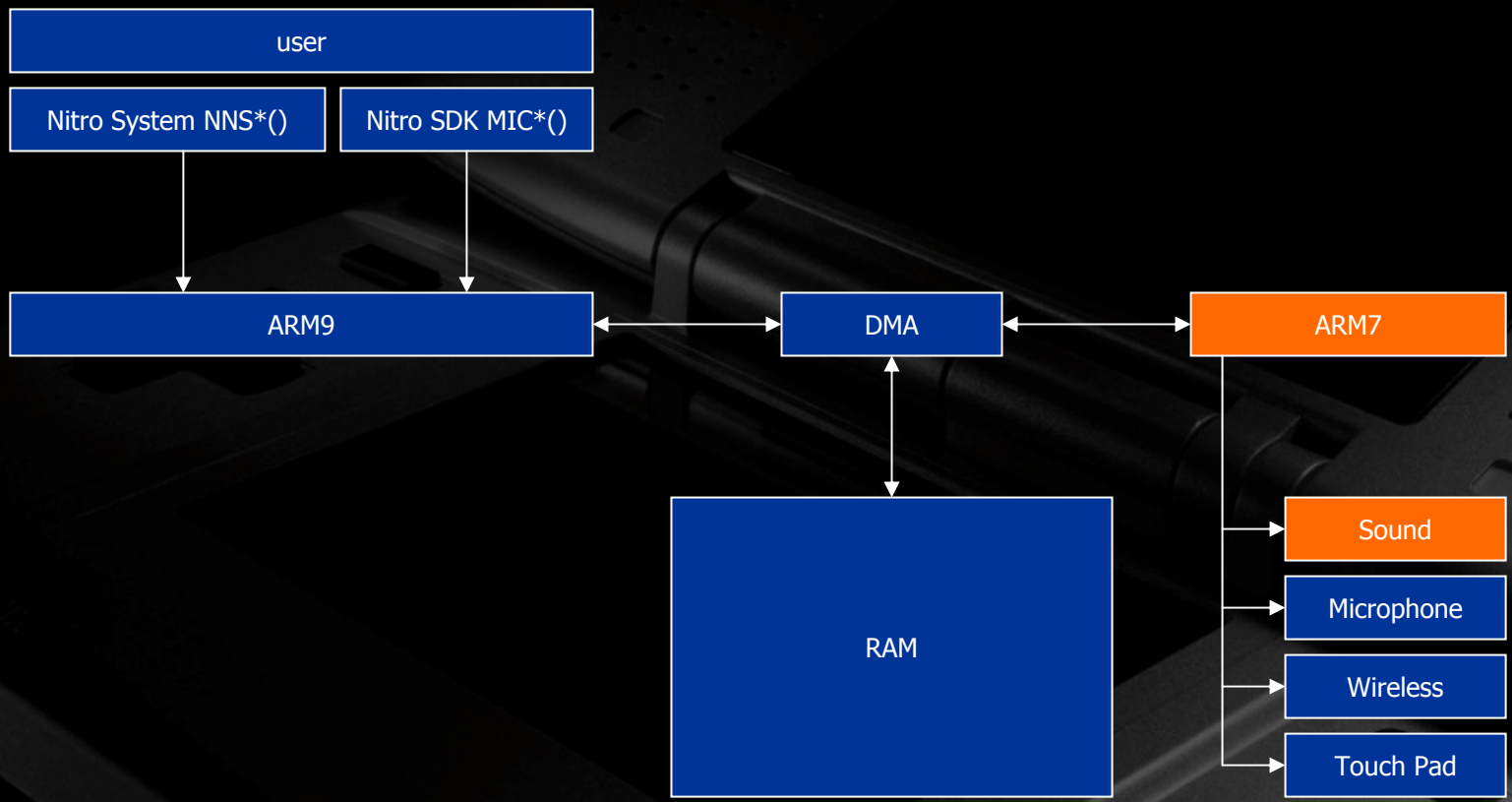
Audio System Processing Model



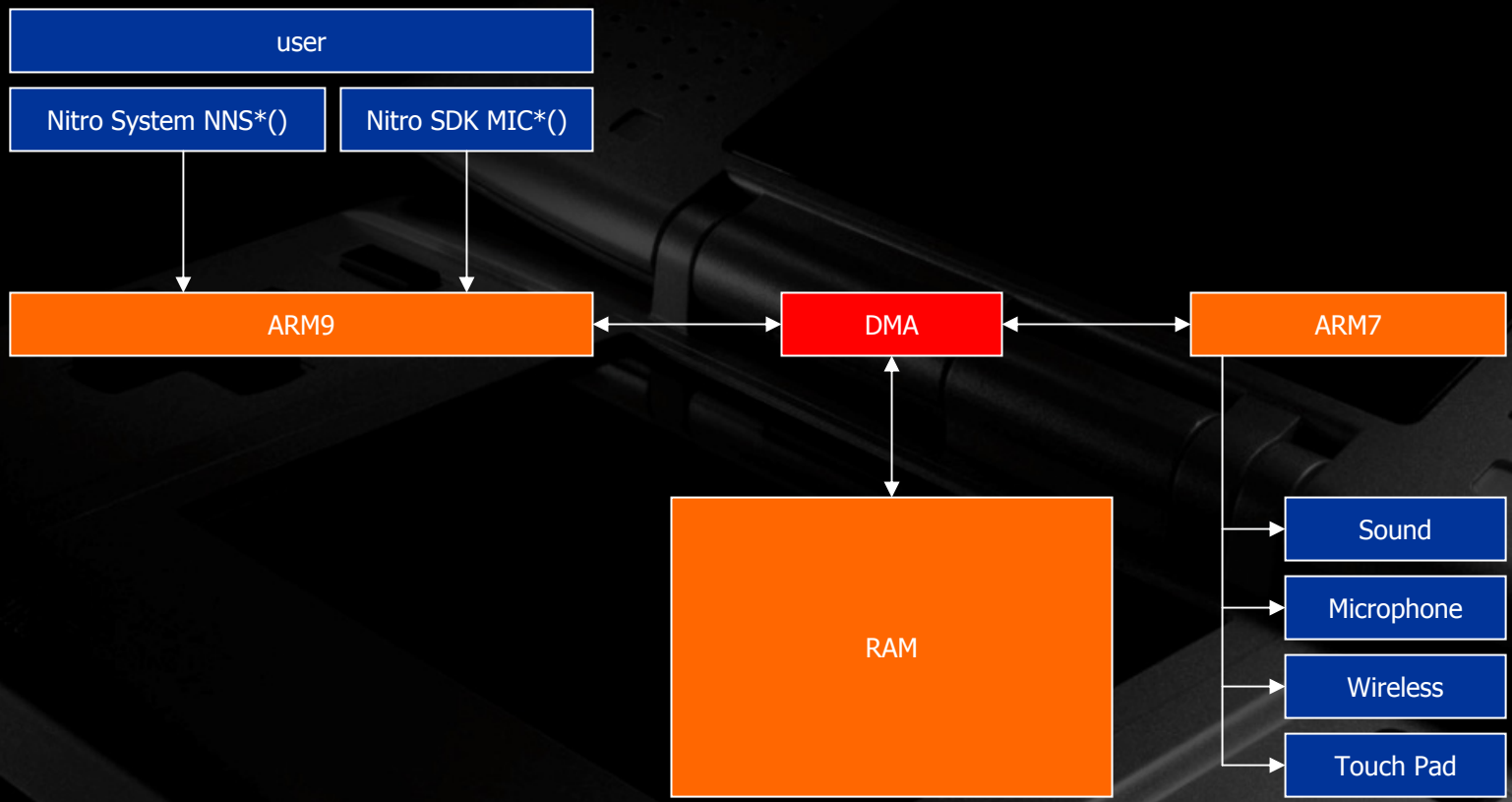
Audio System Processing Model



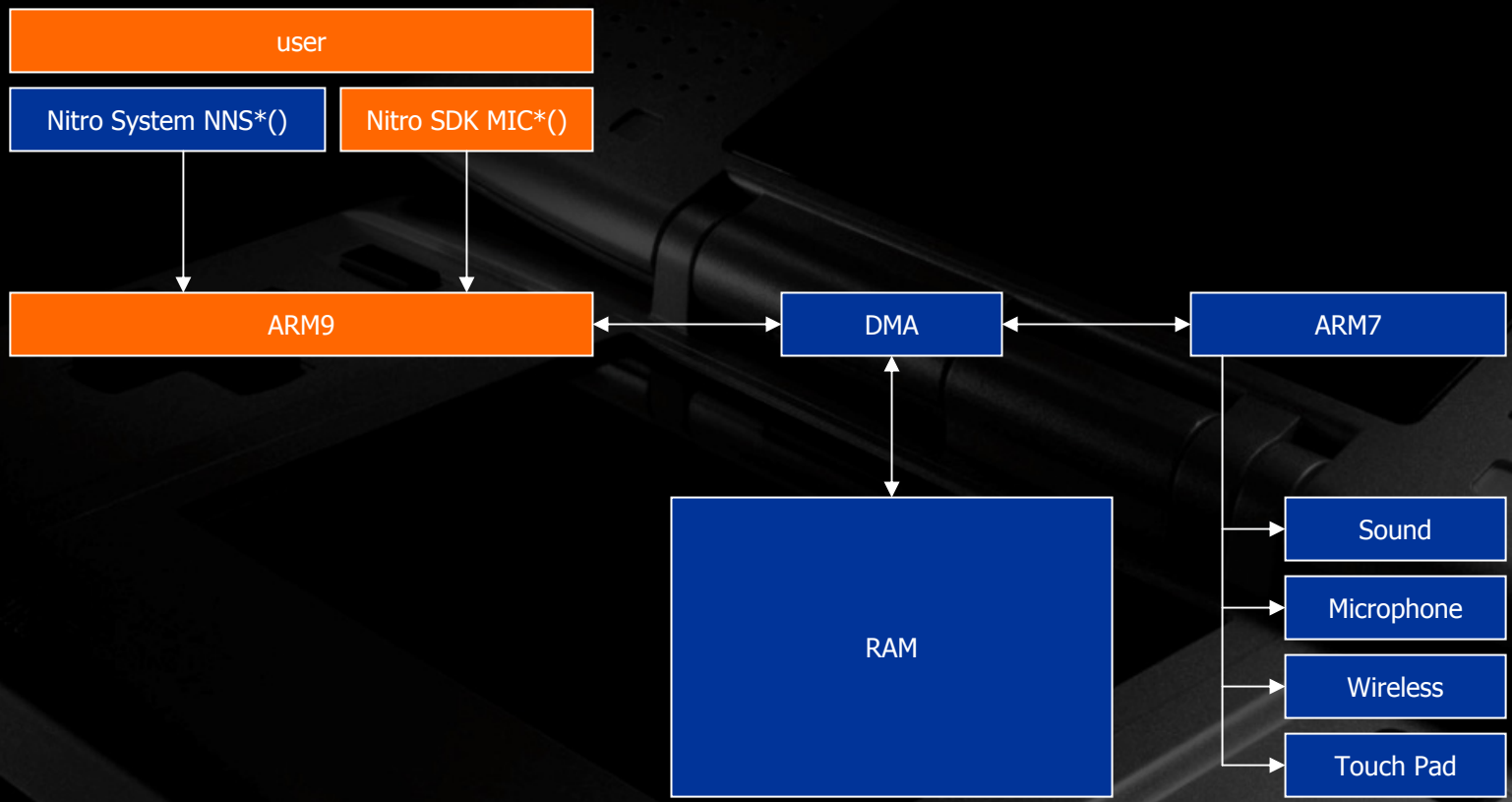
Audio System Processing Model



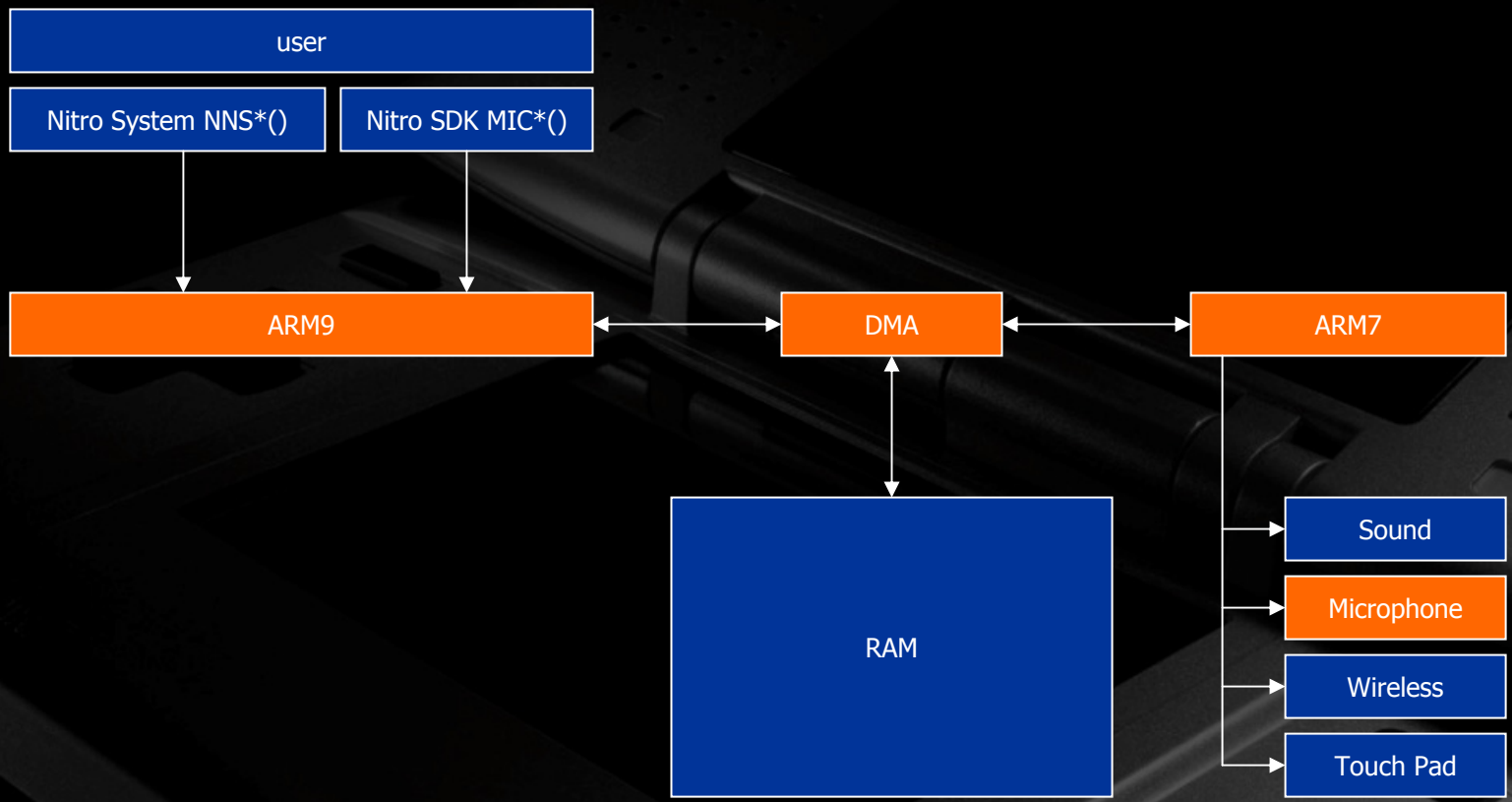
Audio System Processing Model



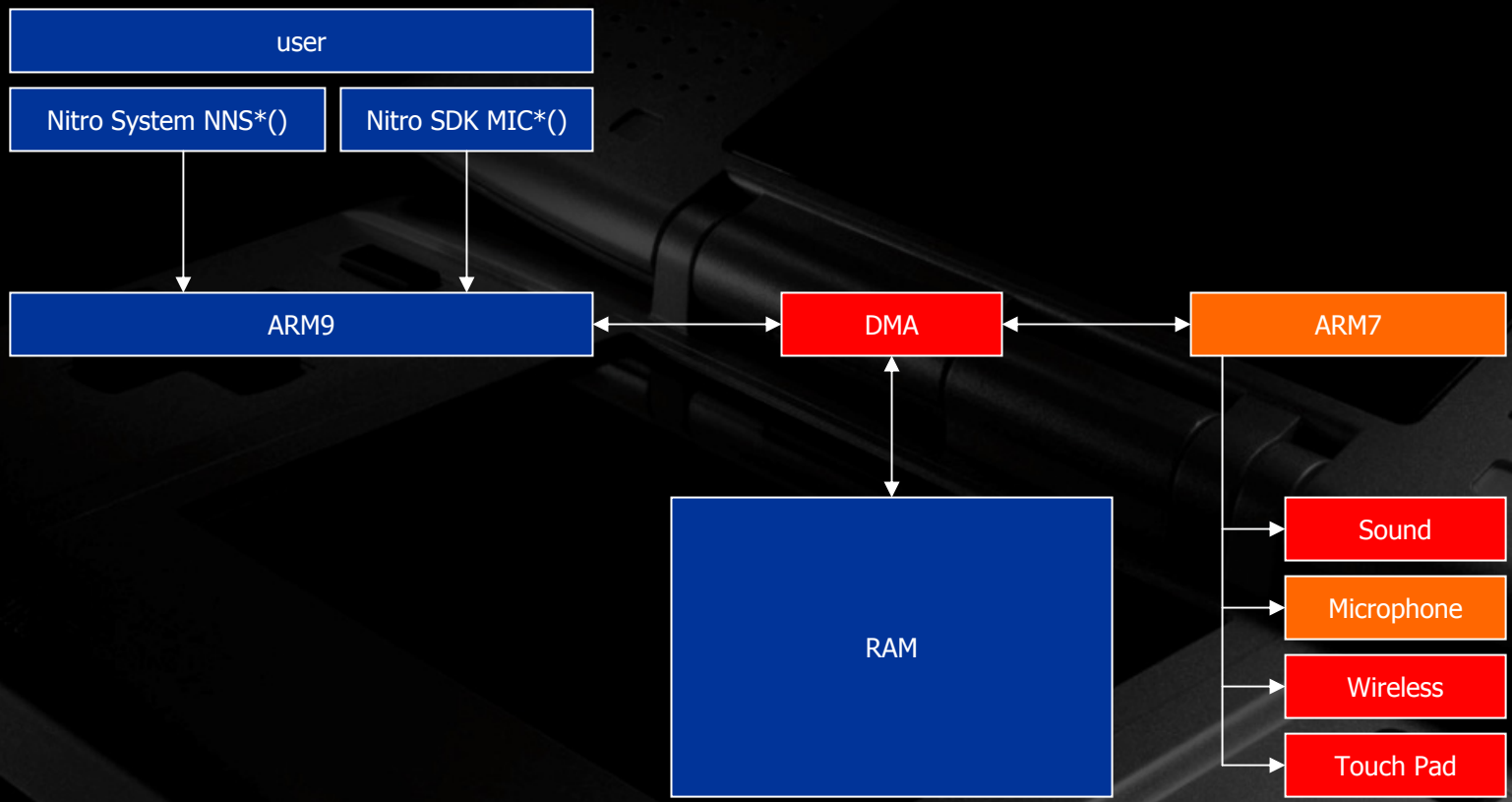
Audio System Processing Model



Audio System Processing Model



Audio System Processing Model



Get it!

Nitro SDK

- Can't build without it!
- MIC API and libraries!
- API man pages!
- Microphone demos with source in spi!

NITRO-System

- Sound API and libs!
- Docs + API man pages!
- Sound demos with source!
- NITRO-Composer!

NINTENDO DS
DEVELOPERS CONFERENCE 2004

Get it!

Nitro SDK

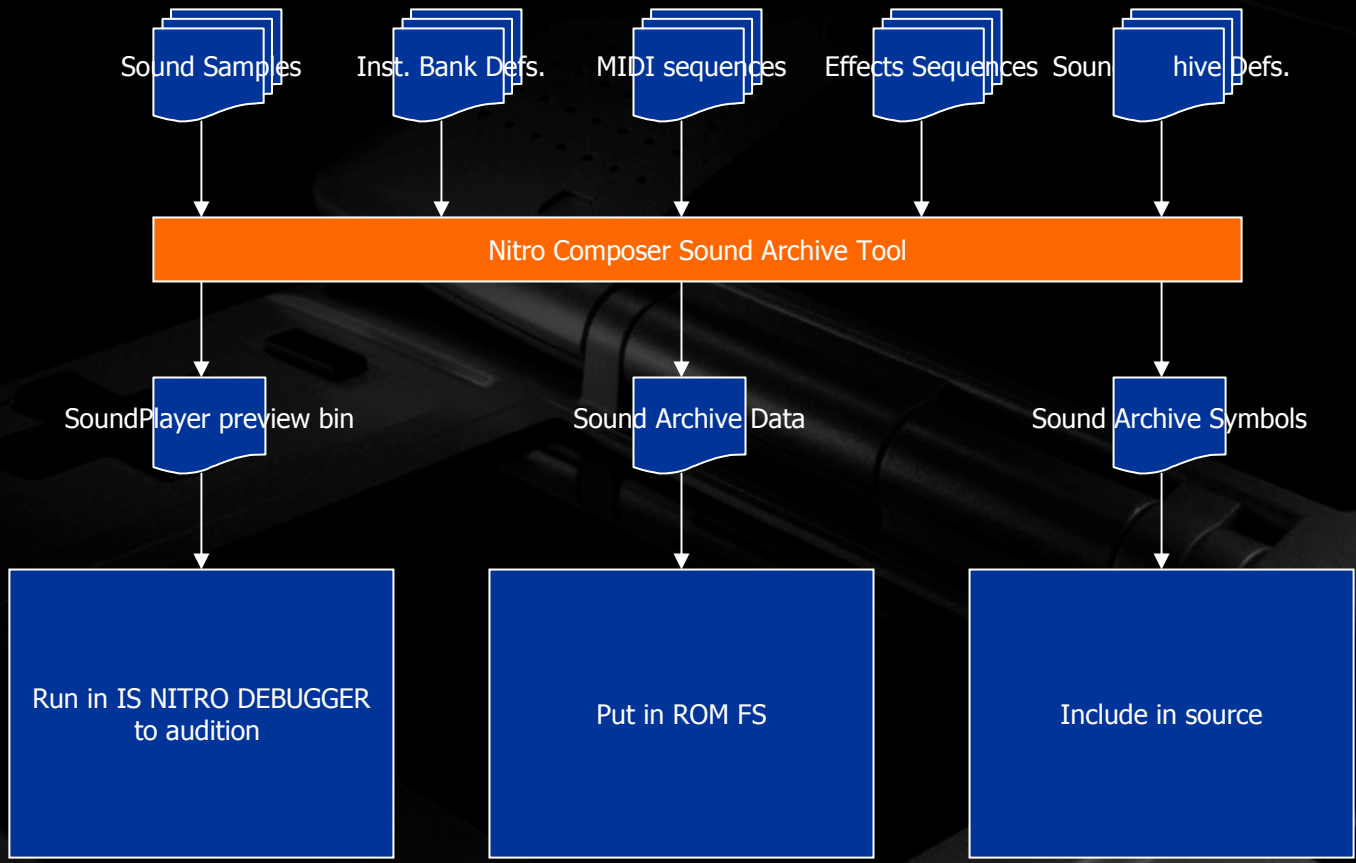
- Can't build without it!
- MIC API and libraries!
- API man pages!
- Microphone demos with source in spi!

NITRO-System

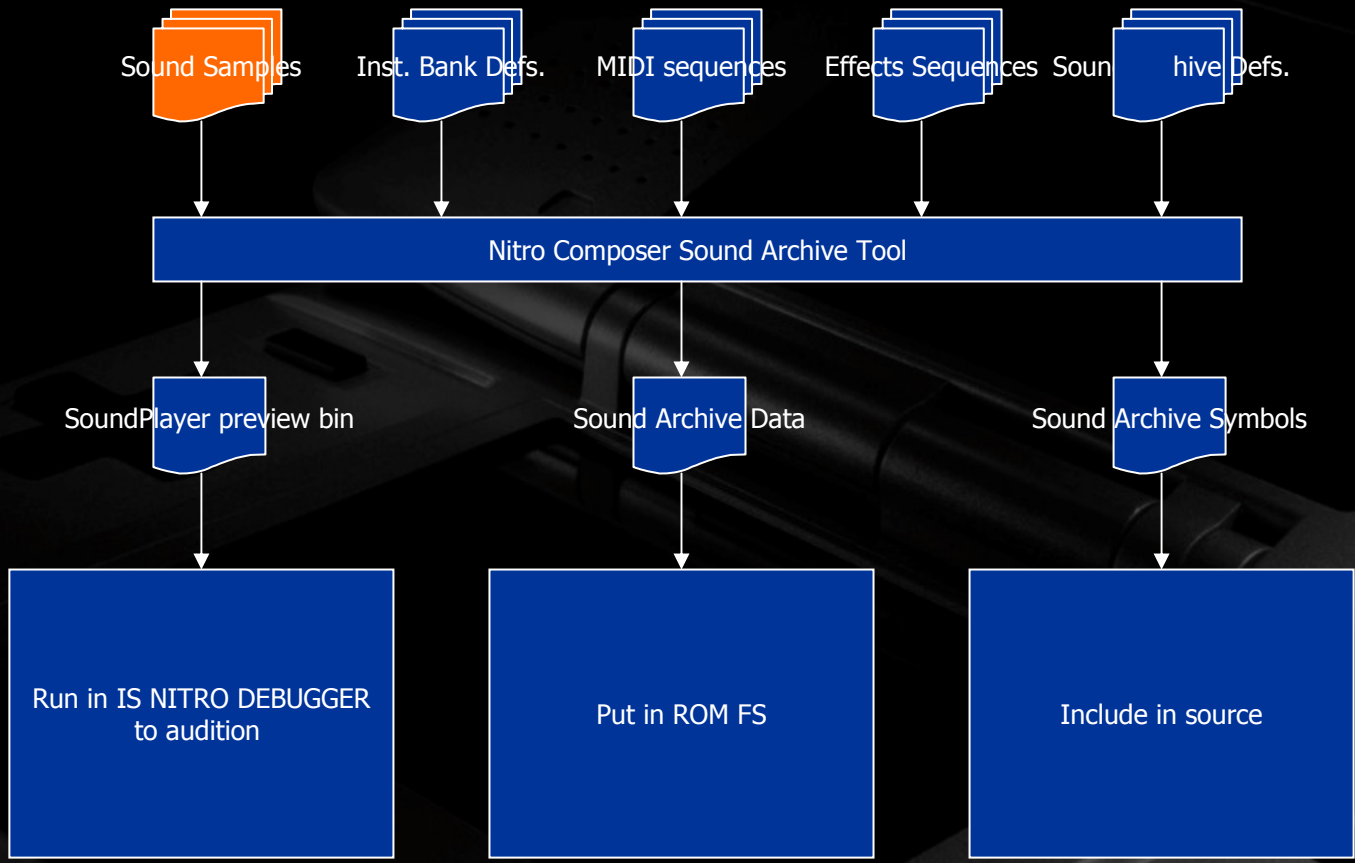
- Sound API and libs!
- Docs + API man pages!
- Sound demos with source!
- NITRO-Composer!

NINTENDO DS
DEVELOPERS CONFERENCE 2004

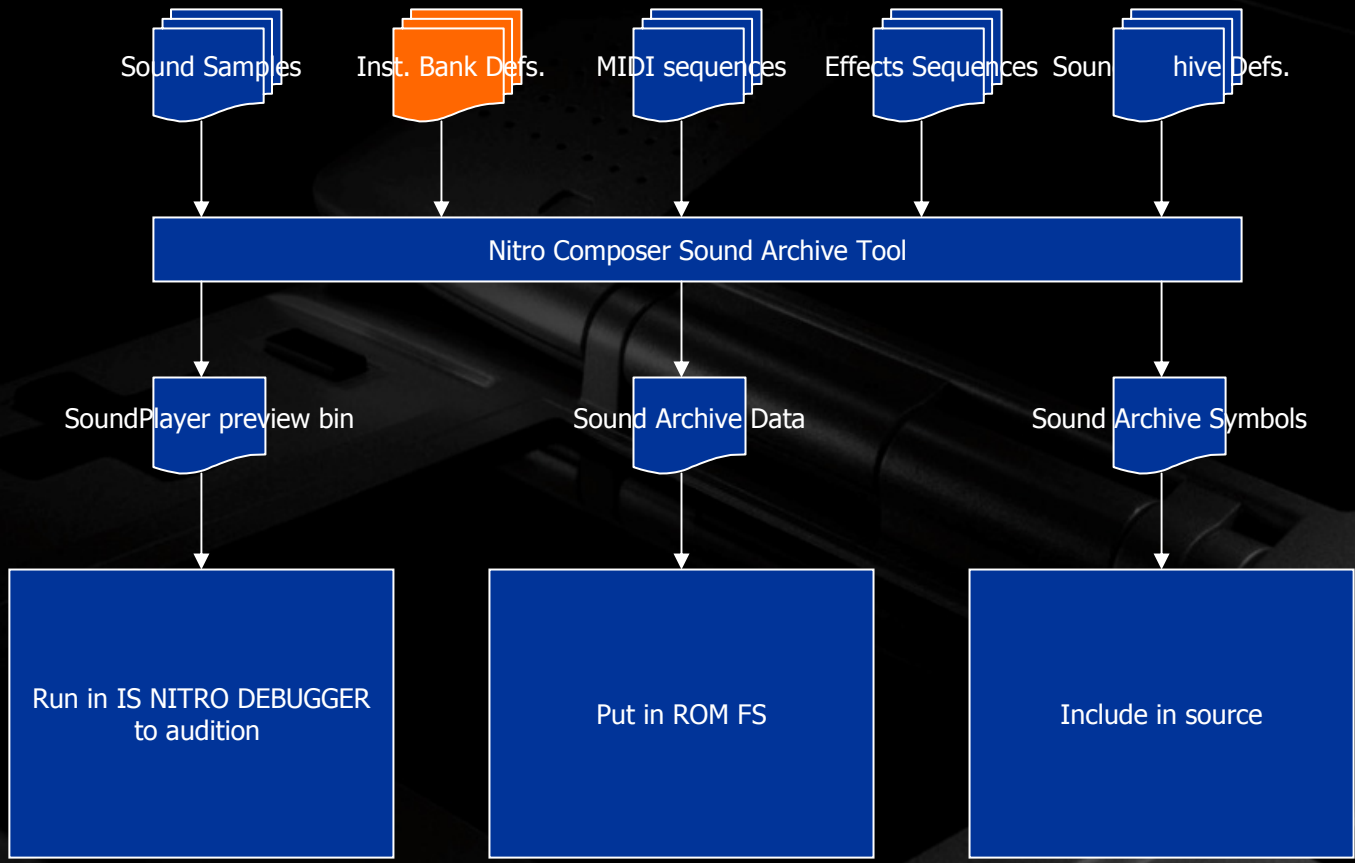
NITRO-Composer



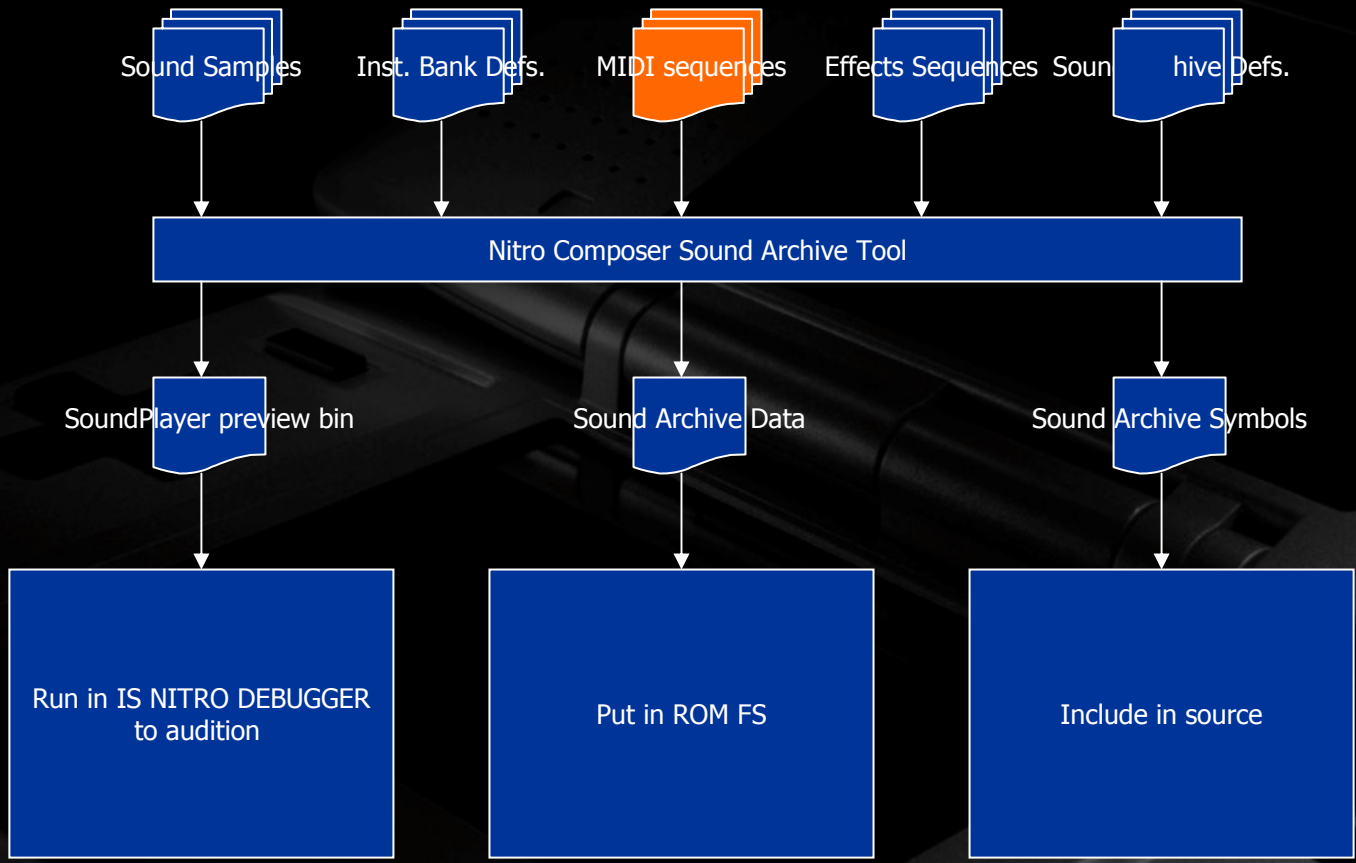
NITRO-Composer



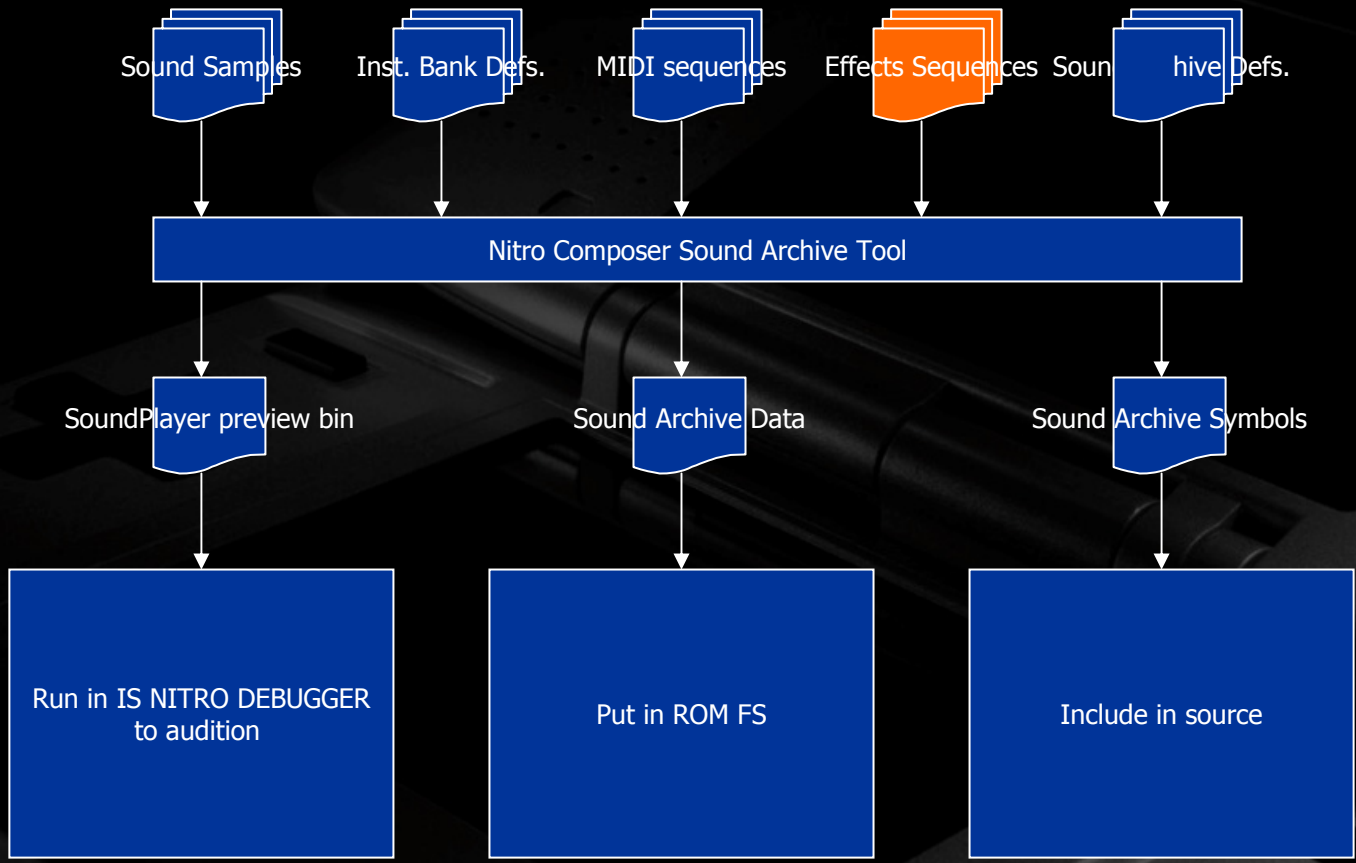
NITRO-Composer



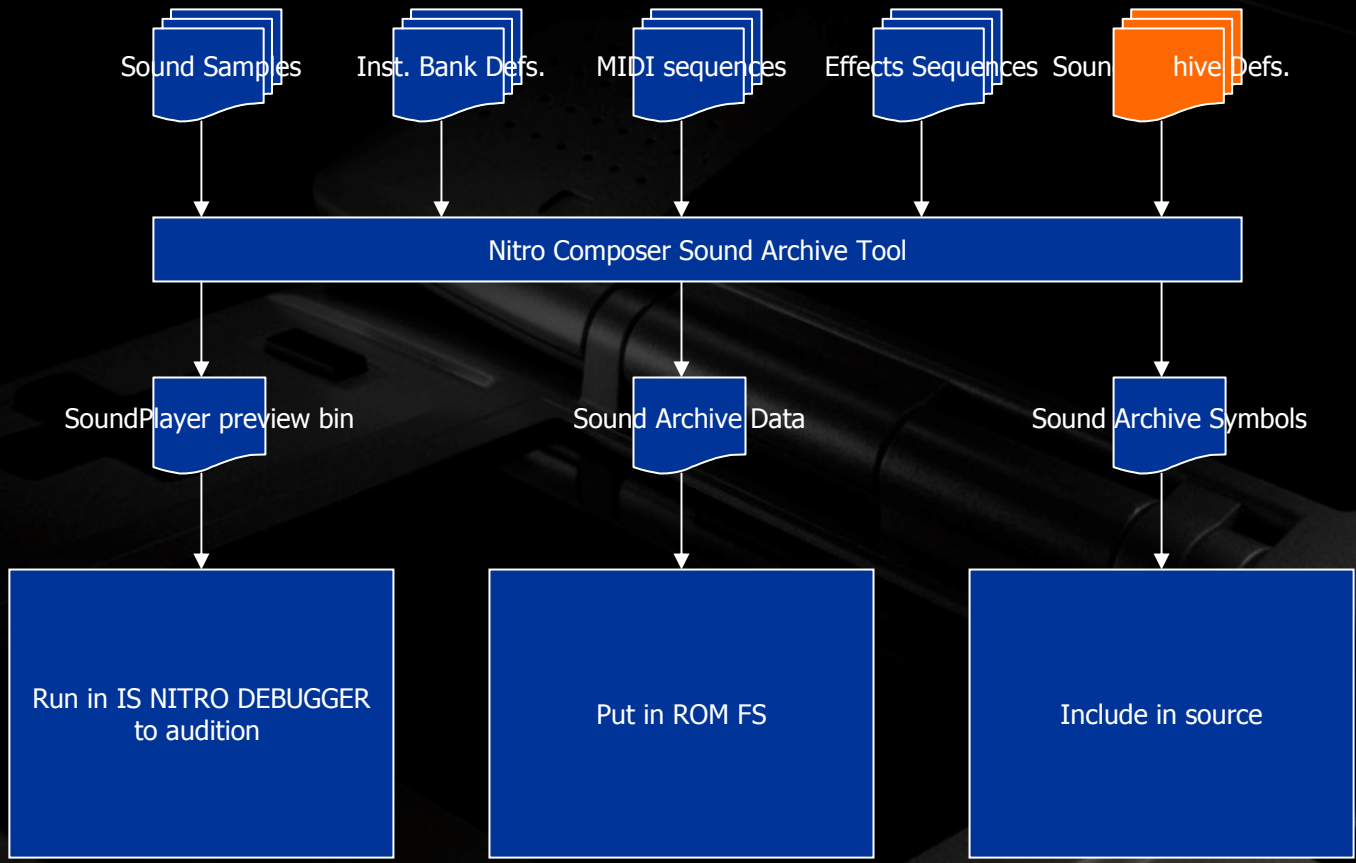
NITRO-Composer



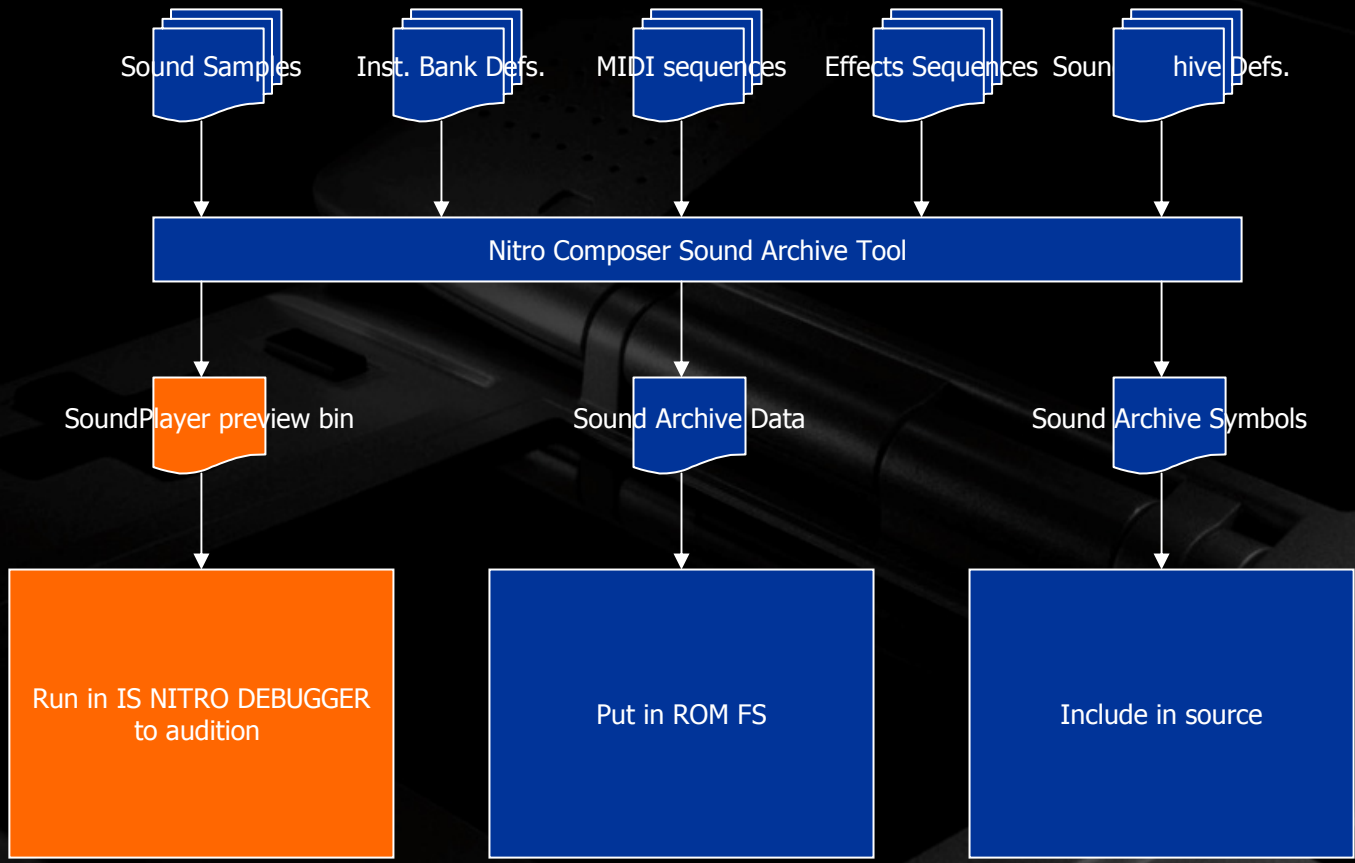
NITRO-Composer



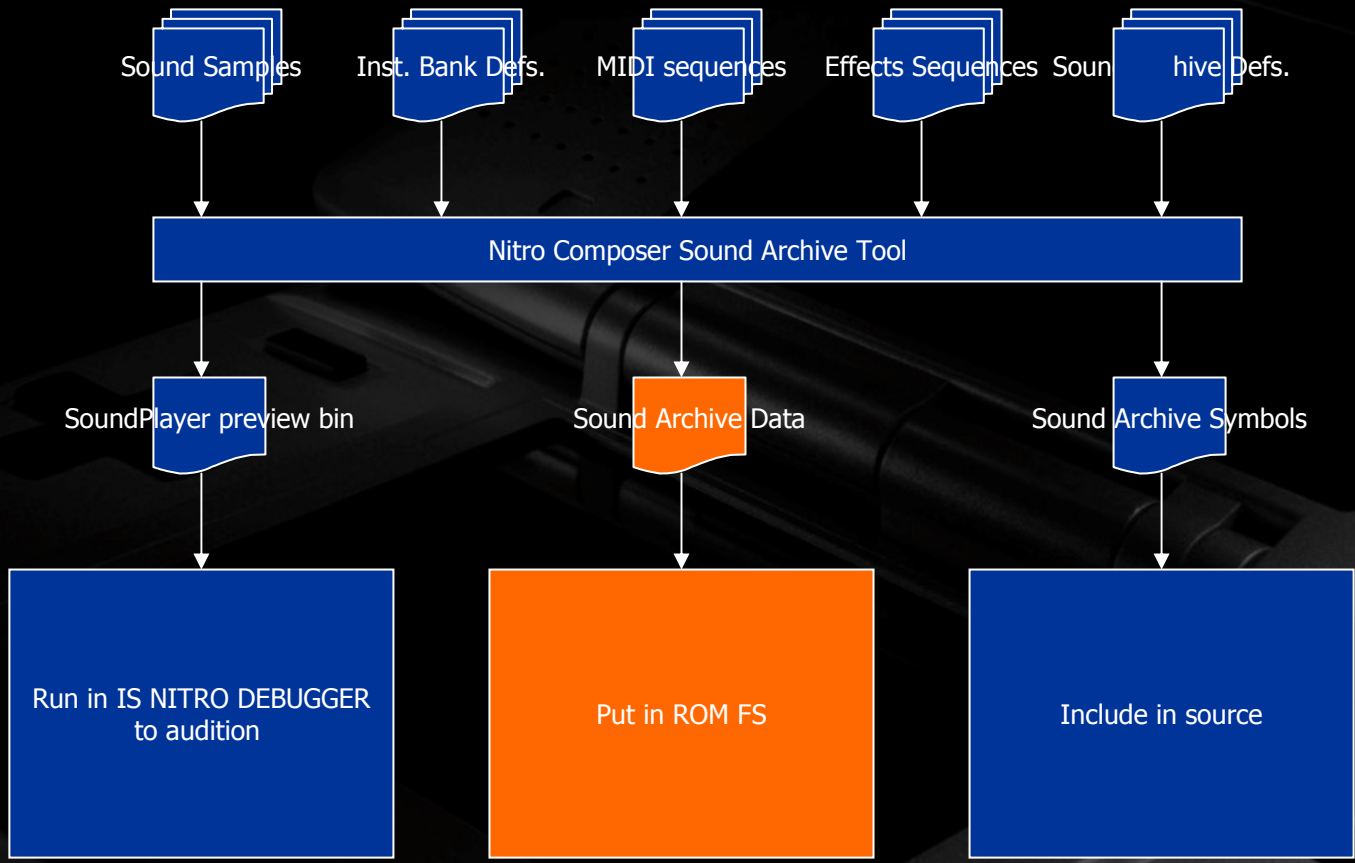
NITRO-Composer



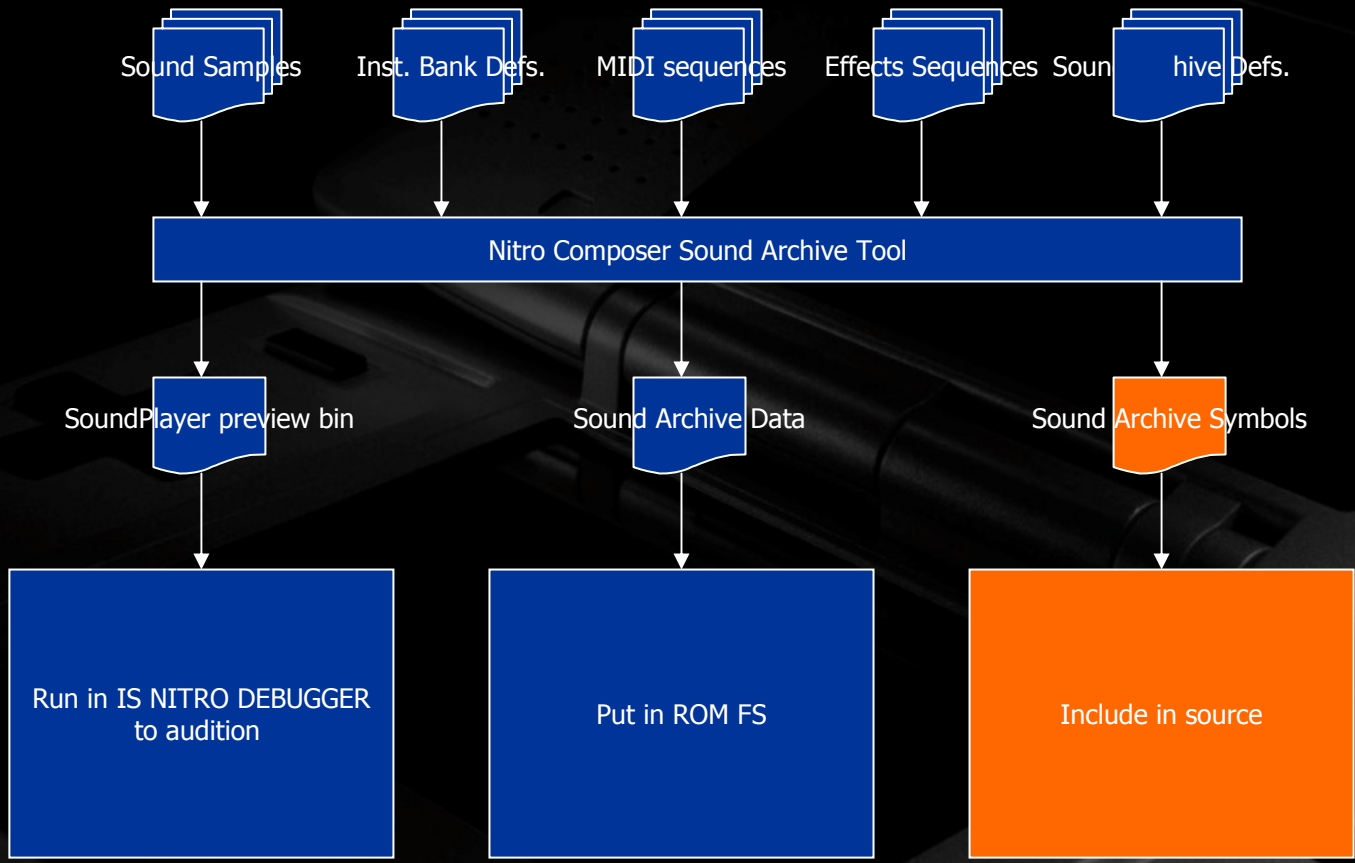
NITRO-Composer



NITRO-Composer



NITRO-Composer

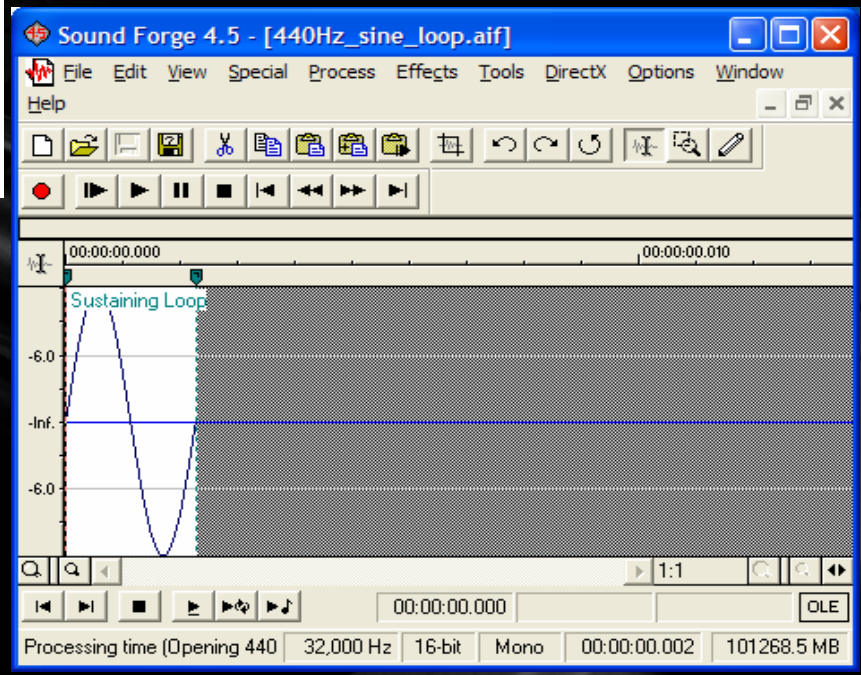


Demo – DTMF tones

		Upper Band			
		1209 Hz	1336 Hz	1477 Hz	1633 Hz
Lower Band	697 Hz	1	2	3	A
	770 Hz	4	5	6	B
	852 Hz	7	8	9	C
	941 Hz	*	0	#	D

Calculation for Equal-Tempered tuning [A3 = 440Hz]

	Hertz	Octave=0	Octave=1	Octave=2	Octave=3	Octave=4	Octave=5
0	A	55.000	110.000	220.000	440.000	880.000	1,760.000
1	A#/Bb	58.270	116.541	233.082	466.164	932.328	1,864.655
2	B	61.735	123.471	246.942	493.883	987.767	1,975.533
3	C	65.406	130.813	261.626	523.251	1,046.502	2,093.005
4	C#/Db	69.296	138.591	277.183	554.365	1,108.731	2,217.461
5	D	73.416	146.832	293.665	587.330	1,174.659	2,349.318
6	D#/Eb	77.782	155.563	311.127	622.254	1,244.508	2,489.016
7	E	82.407	164.814	329.628	659.255	1,318.510	2,637.020
8	F	87.307	174.614	349.228	698.456	1,396.913	2,793.826
9	F#/Gb	92.499	184.997	369.994	739.989	1,479.978	2,959.955
10	G	97.999	195.998	391.995	783.991	1,567.982	3,135.963
11	G#/Ab	103.826	207.652	415.305	830.609	1,661.219	3,322.438
12	A	110.000	220.000	440.000	880.000	1,760.000	3,520.000



NINTENDO DS DEVELOPERS CONFERENCE 2004

Bank

```
;
;    dtmf.instrument_bank
;
;    Sine loop for DTMF tones
;
;------------------------------------------------------------------------
@PATH "../samples"
@INSTLIST

    DTMF_ROOT : PCM16, "440Hz_sine_loop.aif", an2, 126, 127, 127, 126, 64
```

Bank file

Instrument Label

Data format, this specifies how to encode this data, PCM16, PCM8, ADPCM.

File location. Notice AIFF is used to support looping.

MIDI unity note. This is the note that will playback this data at original sample rate.

ADSR envelope, time and volume table in docs.

Default pan value, 0 – 127 per MIDI spec.

Sequence Archive

```
; dtmf.sequence_archive

@SEQ_TABLE
#define bank   0
#define volume 127
#define sp     127
#define pp     127
#define pn     0
;-------------------------------------------------------------------------
SEQ_DTMF_1          : dtmf_1,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_2          : dtmf_2,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_3          : dtmf_3,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_4          : dtmf_4,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_5          : dtmf_5,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_6          : dtmf_6,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_7          : dtmf_7,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_8          : dtmf_8,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_9          : dtmf_9,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_0          : dtmf_0,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_astricks   : dtmf_astricks,   bank,    volume,    sp,    pp,    pn
SEQ_DTMF_pound      : dtmf_pound,      bank,    volume,    sp,    pp,    pn
SEQ_DTMF_A          : dtmf_A,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_B          : dtmf_B,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_C          : dtmf_C,          bank,    volume,    sp,    pp,    pn
SEQ_DTMF_D          : dtmf_D,          bank,    volume,    sp,    pp,    pn
;-------------------------------------------------------------------------
```

Sequence Archive

Sequence Label

Data Label

Bank Number

Volume

Sound Priority

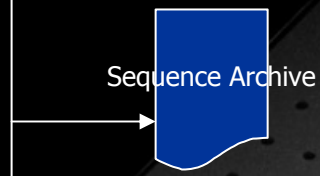
Player Priority

Player Number

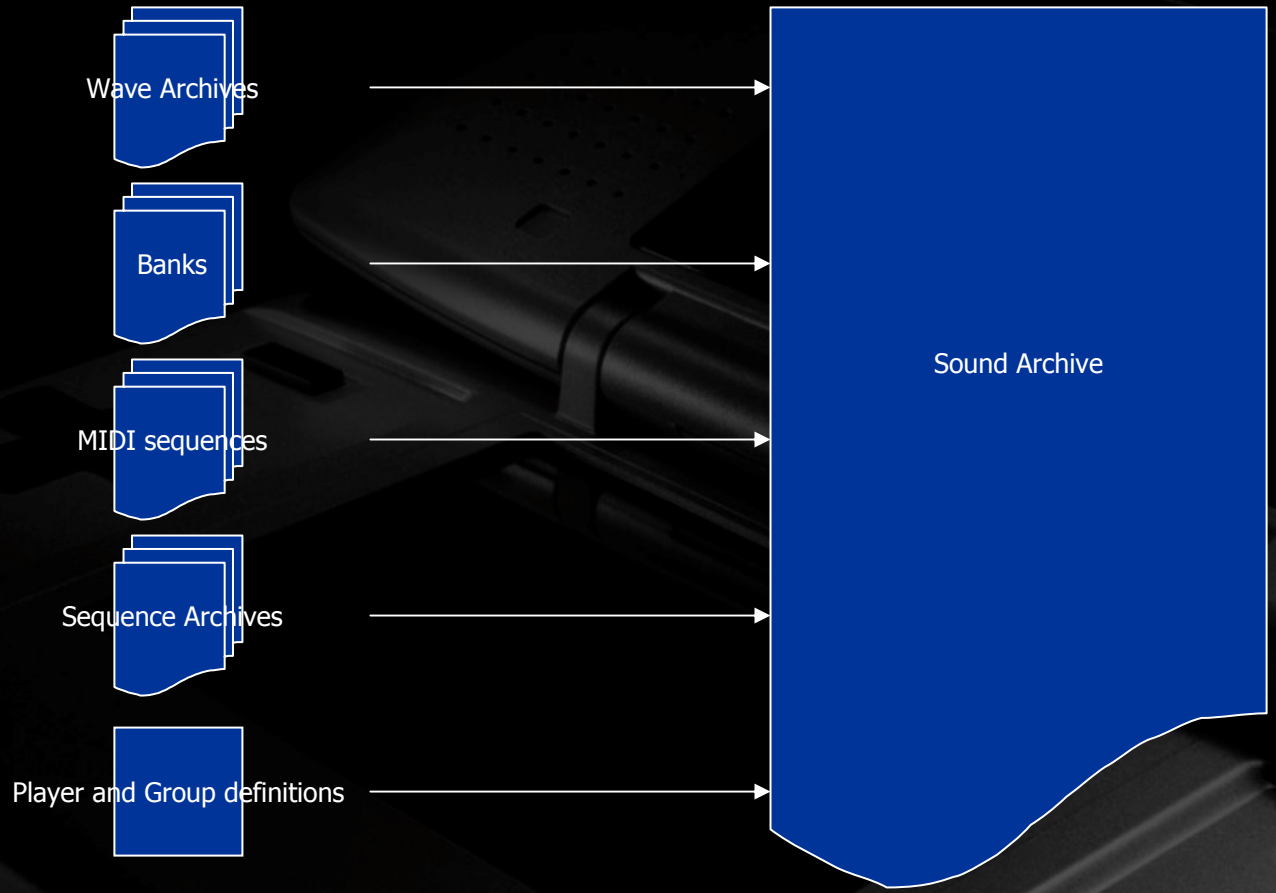
NINTENDO DS DEVELOPERS CONFERENCE 2004

Sequence Archive – Cont'

```
@SEQ_DATA
;-------------------------------------------------------------------------
;   Each DTMF tone consists of two tones as follows:
;
;               1209Hz 1336Hz 1477Hz 1633Hz
;               (ds4)  (en4)  (fs4)  (gs4)
;                 |      |      |      |
;
;   697Hz(fn3) -  1      2      3      A
;
;   770Hz(gn3) -  4      5      6      B
;
;   852Hz(gs3) -  7      8      9      C
;
;   941Hz(as3) -  *      0      #      D
;-------------------------------------------------------------------------
#include "../bank/dtmf.spdl"
#define length 268435455

dtmf_1:

    notewait_off
    prg DTMF_ROOT
    fn3 127, length
    ds4 127, length
    wait length
    fin
```



Sound Archive



Wave Archive

```
; Wave archive, swls files are written automatically by bankconv
@WAVEARC
@PATH "wave_archive"

;    WAVE_PIANO      : AUTO, "piano.swls"
     WAVE_DTMF       : AUTO, "dtmf.swls"
```

Wave Archive Entry Label

Automatically generated data, this is the only option available here

Automatically generated script file

Bank

```
; sound banks, each has instruments assigned to program numbers
@BANK
@PATH "sound_bank"

;    BANK_PIANO      : TEXT, "piano.instrument_bank",     WAVE_PIANO
     BANK_DTMF       : TEXT, "dtmf.instrument_bank",      WAVE_DTMF
```

Bank Entry Label

Text Data

Bank Definitions

Wave Archive for this Bank

Sequences

```
; sequences, MIDI files to be converted to sequence files, typically for music
@SEQ
@PATH "midi"

;    SEQ_MOZART: SMF, "mz_330_1.mid", BANK_PIANO, 127, 64, 64, 0
```

Sequence Label
Data Type
MIDI File
Bank
Volume
Sound Priority
Player Priority
Player Number

```
; sequence archives, hand written sequences, typically for sound effects
@SEQARC
@PATH "sequences"

    SEQ_DTMF: TEXT, "dtmf.sequence_archive"
```

Sequence Archive Label
Sequence Archive File

Sound Archive

Player and Group

```
; players, define resources to allow sequence players
@PLAYER
    PLAYER_DTMF : 1, 8000
;   PLAYER_PIANO: 1, 8000
```

Player Label

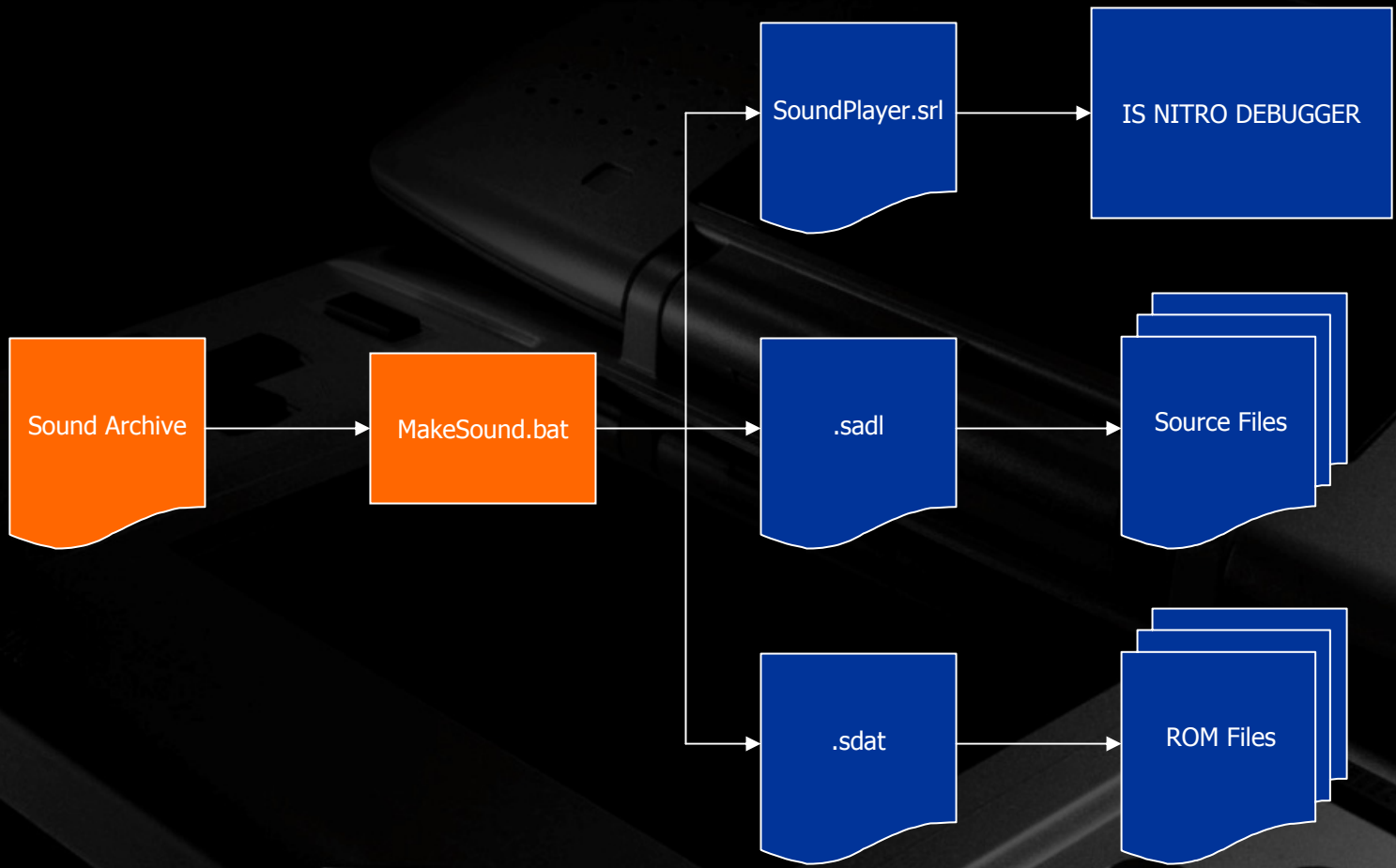
Maximum Sequences

Heap Size

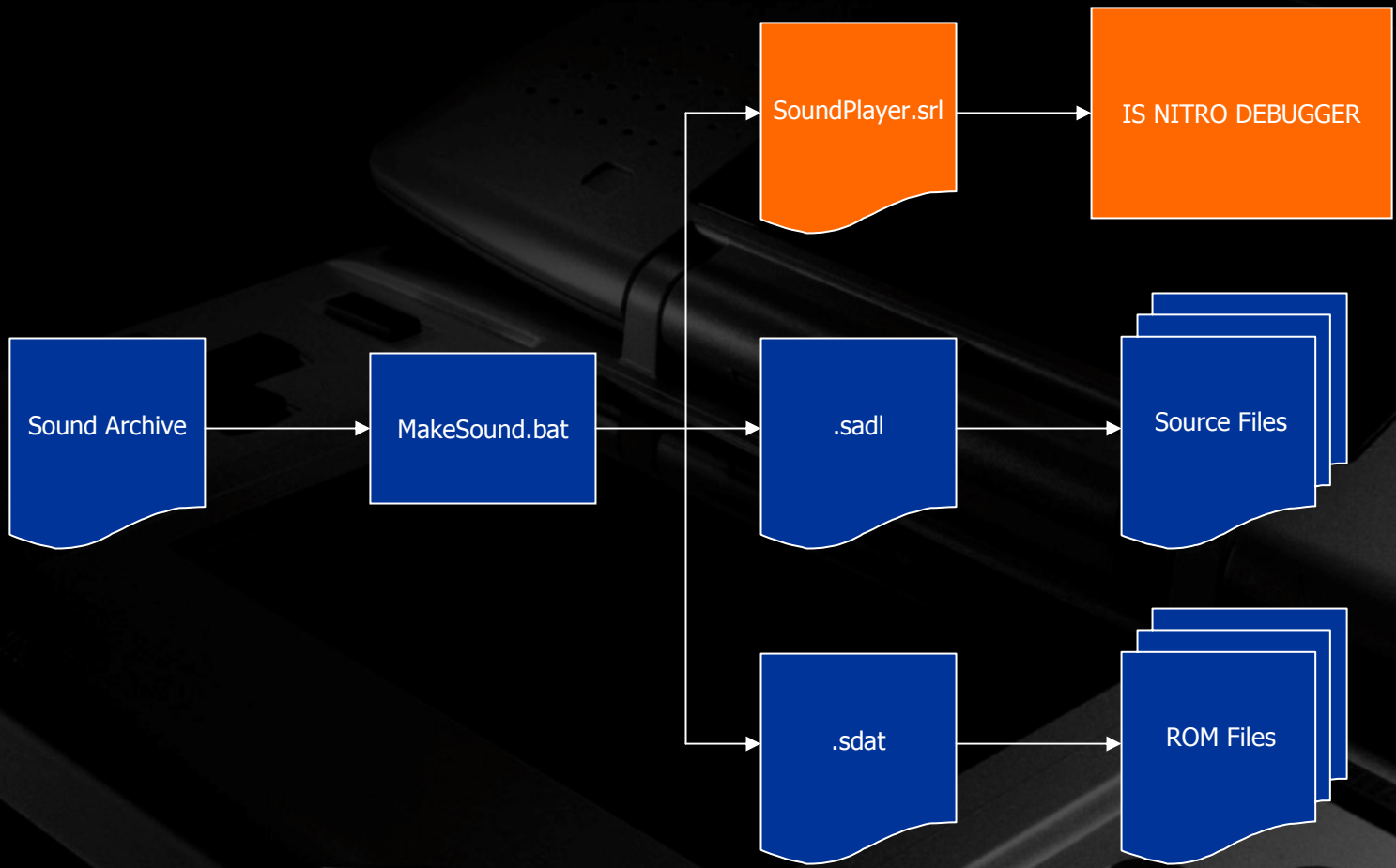
Sound Archive

```
; groups, for runtime sound heap management
@GROUP
    GROUP_STATIC = 0:
        BANK_DTMF
;       BANK_PIANO
; GROUP_LEVEL_1_SFX:
        ; BANK_LEVEL_1_SFX
        ; BANK_LEVEL_1_BGM
```

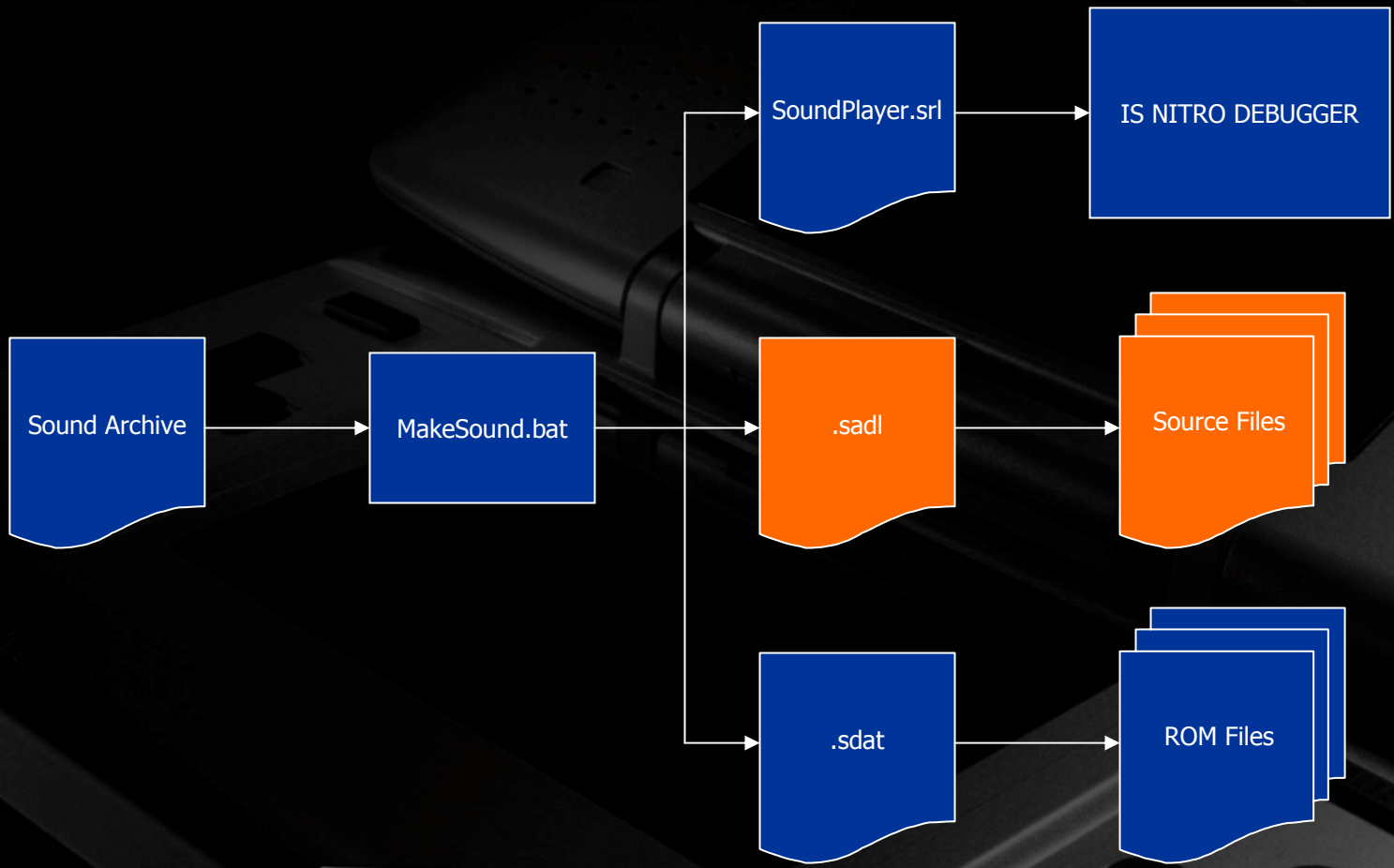
The Hand-off


SoundPlayer.srl
IS NITRO DEBUGGER
Sound Archive
MakeSound.bat
.sadl
Source Files
.sdat
ROM Files

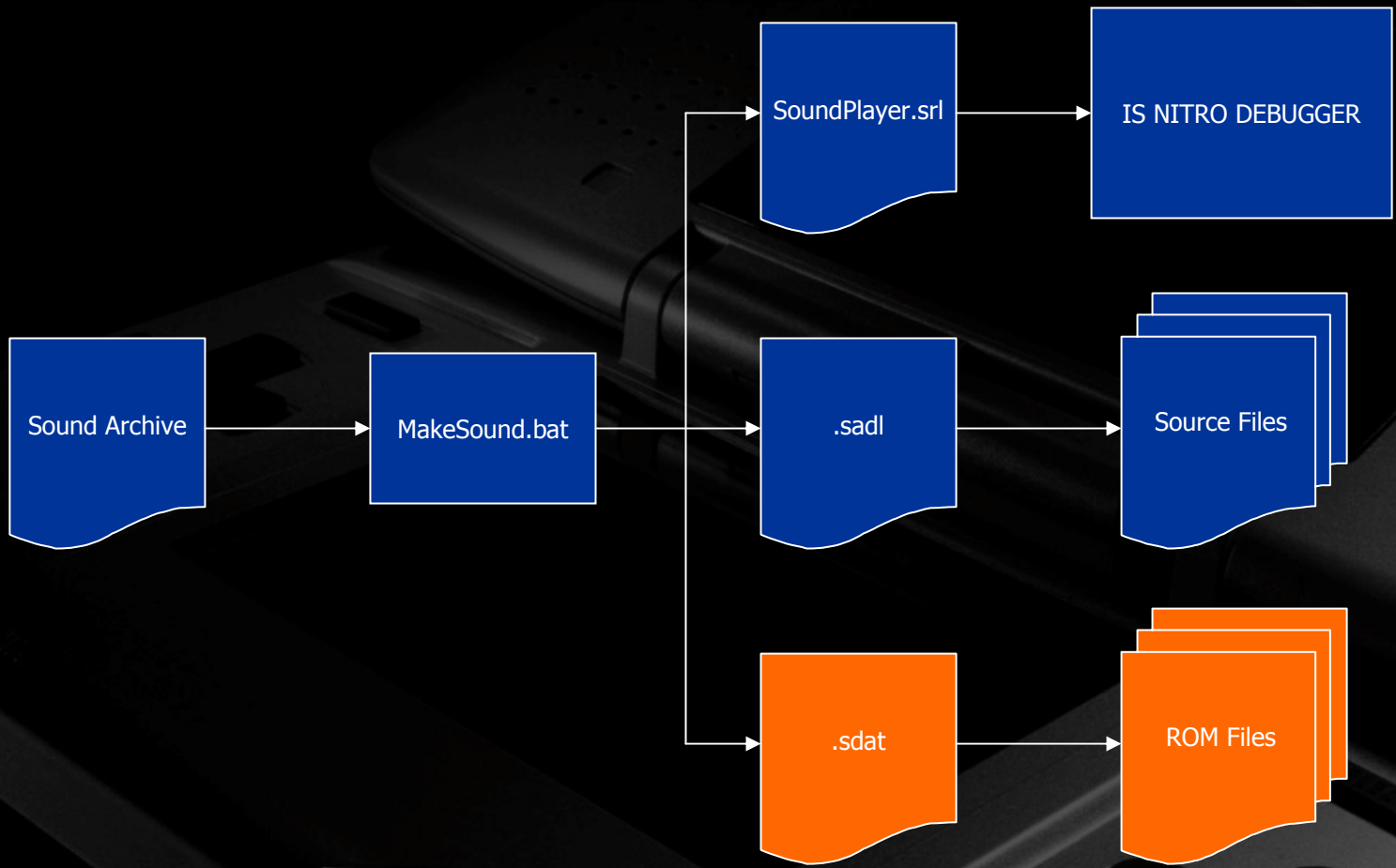
The Hand-off


SoundPlayer.srl
IS NITRO DEBUGGER
Sound Archive
MakeSound.bat
.sadl
Source Files
.sdat
ROM Files

The Hand-off

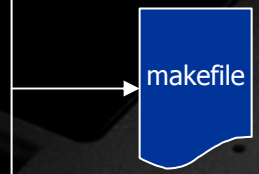


The Hand-off



Building a program – Makefile

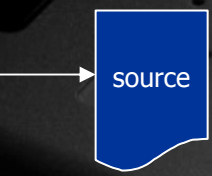
```
SUBDIRS             =

#---------------------------------------------------------------------------

SRCS                = main.c draw.c

TARGET_NEF          = main.nef
TARGET_BIN          = main.srl

MAKEROM_ROMROOT     = c:/nitrotestsounds
MAKEROM_ROMFILES    = sounds.sdat

Include             $(NITROSYSTEM_ROOT)/build/buildtools/commondefs

#---------------------------------------------------------------------------

do-build:           $(TARGETS)

Include             $(NITROSYSTEM_ROOT)/build/buildtools/modulerules

#---------------------------- End of Makefile ------------------------------
```



Building a program – Initialization and Framework

```
// initialize audio system
NNS_SndInit();

// create a sound heap
heap = NNS_SndHeapCreate(&sndHeap, sizeof(sndHeap));

// initialize the sound data to use the heap
NNS_SndArcInit(&arc, "/sounds.sdat", heap, FALSE);

// setup the player to use the heap
NNS_SndArcPlayerSetup(heap);

// load sequence archive and sound bank to heap
NNS_SndArcLoadSeqArc(SEQ_DTMF, heap);
NNS_SndArcLoadBank(BANK_DTMF, heap);

// initialize sound handles
NNS_SndHandleInit(&seHandleDTMF);
```

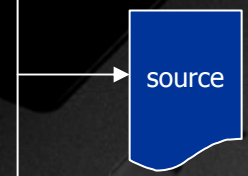
```
while (1)
{
    SVC_WaitVBlankIntr();    // Wait V-Blank interrupt
    CheckTPInput();          // check user input
    NNS_SndMain();           // run sound
}
```



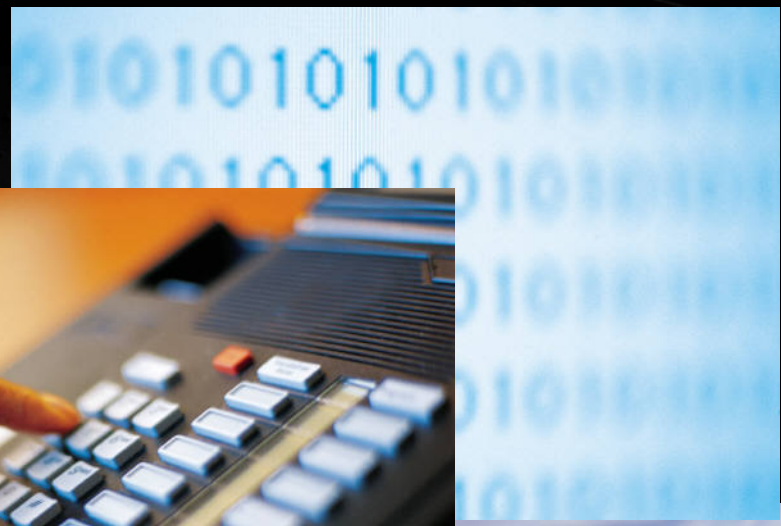
Building a program – Using Sequences

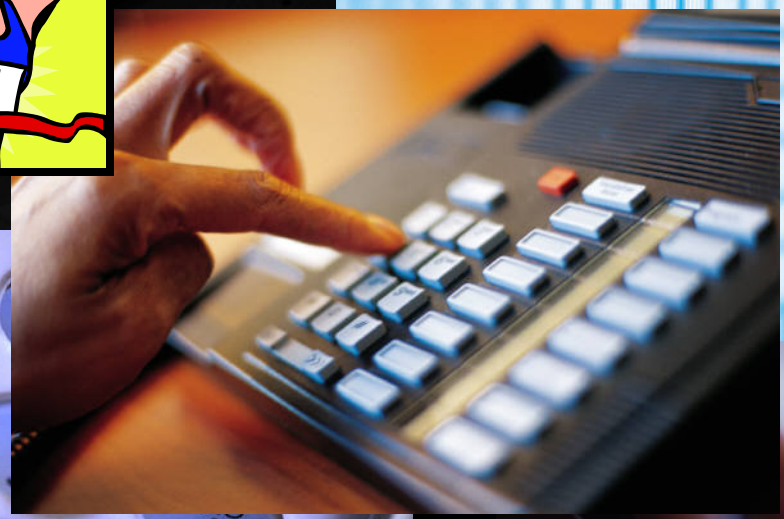
```
// 1 button is pushed
NNS_SndArcPlayerStartSeqArc(
                            &seHandleDTMF, // sequence handle
                            SEQ_DTMF,      // sequence archive
                            SEQ_DTMF_1     // sequence
                            );
```

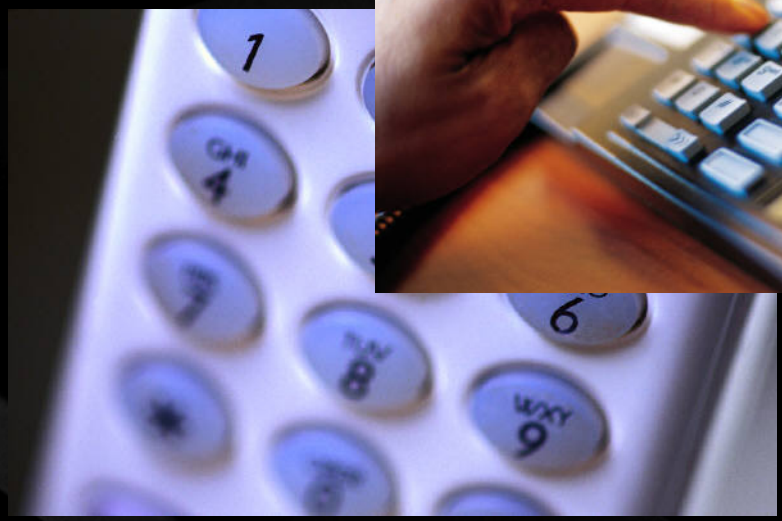
```
// button is released
if(NNS_SndHandleIsValid(&seHandleDTMF))
    NNS_SndPlayerStopSeq(&seHandleDTMF, 0);
```



Run it!







NINTENDO DS
DEVELOPERS CONFERENCE 2004

SUM It Up!

Dual **S**peakers!

Powerful 16 Voice Sound!

Built-in Microphone!

Nitro SDK!

NITRO-System!

NITRO-Composer!

\+

Your Creative Genius!

=WHOA

NINTENDO DS
DEVELOPERS CONFERENCE 2004