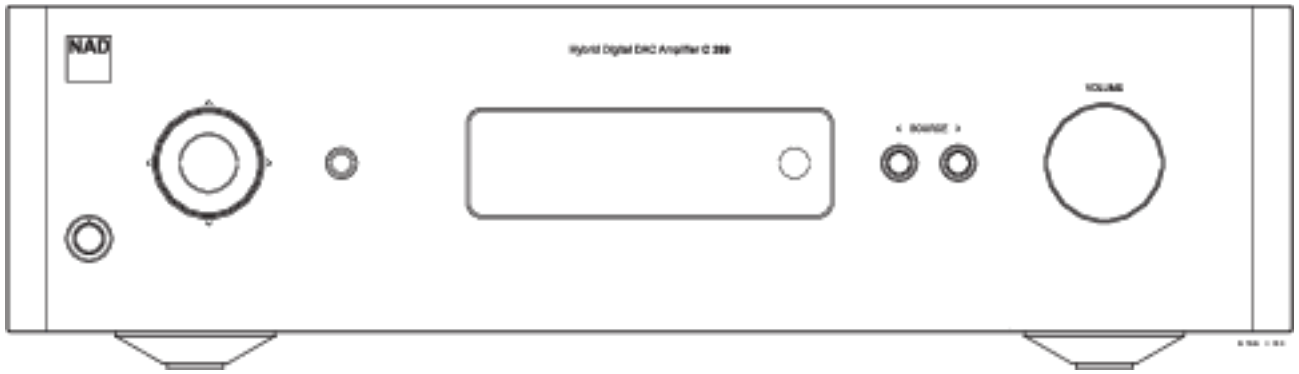
ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

Данный интегральный усилитель разработан компанией NAD Electronics (Канада) для профессионального и коммерческого применения. Усилитель не предназначен для использования в жилых помещениях в качестве бытового аудиооборудования. Для достижения оптимальных результатов в профессиональной среде рекомендуется использование с модулем MDC2 BluOS-D, обеспечивающим коррекцию акустики помещения Dirac Live и интеграцию с профессиональными системами управления.

Интегральный усилитель NAD C379



Класс: гибридный цифровой усилитель/ЦАП для студийного мониторинга и пост-продакшн

ЦЕЛЕВОЕ НАЗНАЧЕНИЕ (ПРОФЕССИОНАЛЬНОЕ НЕБЫТОВОЕ ПРИМЕНЕНИЕ)

Интегральный усилитель NAD C379 **предназначен исключительно для профессионального небытового использования**.

Основные профессиональные сценарии применения:

Сфера применения	Тип задач
Студии звукозаписи (малые и средние)	Контроль записи (tracking), сведение (mixing), мониторинг через пассивные мониторы
Пост-продакшн студии	Контроль микса для кинопоказа, телевидения, стриминговых платформ
Монтажные комнаты (edit suites)	Предварительное сведение и оценка совместимости
Мастеринговые студии	Контроль качества финального микса
Радиостанции и подкастинговые студии	Речевой мониторинг, контроль эфира
Образовательные учреждения	Лаборатории аудиотехники, классы звукорежиссуры
Профессиональные демонстрационные зоны (showrooms)	Демонстрация контента в форматах высокой четкости

Категорически не предназначено для: бытового использования в жилых помещениях.

Ключевые особенности для профессионального применения:

1. **Программируемость через Python API:** Полная поддержка BluOS API для автоматизации и интеграции в профессиональные системы управления (при установке модуля MDC2 BluOS-D) .
2. **Dirac Live Room Correction:** Встроенная система коррекции акустики помещения (до 5 сохраняемых профилей), критически важная для профессионального контроля в неидеальных акустических условиях .
3. **Премиальный ЦАП ESS Sabre ES9028:** 32-бит/384 кГц конвертер с высоким динамическим диапазоном и сверхнизкими искажениями для точного контроля записи .
4. **Технология HybridDigital UcD:** Усиление класса D с почти неизмеримыми искажениями в слышимом диапазоне, обеспечивающее чистый сигнал для профессионального мониторинга .
5. **Профессиональные системные интеграции:** Поддержка Crestron, Control4, RTI, Lutron, URC .

1. ВВЕДЕНИЕ И ОБЩИЕ СВЕДЕНИЯ

Настоящая инструкция предназначена для **квалифицированного персонала**, использующего интегральный усилитель NAD C379 в профессиональных целях. Усилитель построен на фирменной технологии HybridDigital с выходными каскадами UcD (фирменная версия класса D) и оснащен двумя слотами расширения MDC2 для установки опциональных модулей, включая MDC2 BluOS-D, обеспечивающий стриминг высокого разрешения и коррекцию акустики помещения Dirac Live .

Основные технические характеристики:

Параметр	**Значение**
Выходная мощность (непрерывная)	2 x 80 Вт (4/8 Ом)
Выходная мощность (динамическая)	2 x 120 Вт (4 Ом)
Мощность в мостовом режиме (с C268)	до 2 x 300 Вт
THD (коэффициент гармоник)	<0,03% (20 Гц - 20 кГц)
Соотношение сигнал/шум	>95 дБ (IHF, A-взвешенный)
Коэффициент демпфирования	>300
Разделение каналов	>90 дБ (1 кГц)
Частотный диапазон	20 Гц - 20 кГц (±0.3 дБ)
ЦАП	ESS Sabre ES9028 (32 бит/384 кГц)
Максимальный выходной ток	>26 A (1 Ом, 1 мс)

Подключения и разъемы :

Входы:

Тип	Количество	Назначение
HDMI eARC	1	Подключение к источникам (ТВ, проекторы)
Коаксиальный S/PDIF (RCA)	2	Цифровой вход
Оптический Toslink	2	Цифровой вход
Аналоговый RCA (Line)	3	Линейный вход
Phono MM (RCA)	1	Вход для проигрывателя винила
USB Type-A	1	Сервисный порт / накопители
Bluetooth	1	aptX HD (двунаправленный)

Выходы:

Тип	Количество	Назначение
Акустические терминалы	2 пары (A, B)	Подключение пассивных мониторов
Pre Out (RCA)	1	Выход на внешний усилитель
Субвуферный выход (RCA)	2	Подключение активных сабвуферов
Наушники	1 x 6,3 мм	Выход на наушники

Управление и интеграция:

Тип	Назначение
RS232	Профессиональное управление
IR IN (3,5 мм)	Вход для ИК-приемника
Trigger OUT (12 В)	Управление внешними устройствами

---

2. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ

Перед эксплуатацией системы ознакомьтесь со следующими требованиями:

- **Электропитание:** Используйте только прилагаемый кабель питания. Напряжение питания: 230 В . Защищайте кабель питания от повреждений.
- **Вентиляция:** Обеспечьте свободный доступ воздуха вокруг усилителя. Не блокируйте вентиляционные отверстия. Рекомендуемый зазор: не менее 10 см сверху и по бокам.
- **Температура:** Эксплуатируйте при температуре от 0 до 40°C.
- **Влажность:** Не допускайте попадания жидкости на корпус. Не эксплуатируйте вблизи воды.
- **Перегрузка:** Усилитель оснащен защитой от перегрузки. Признаки искажений — проверьте уровень источника.
- **Чистка:** Используйте только сухую мягкую ткань. Не применяйте абразивные средства.

⚠ **ПРОФЕССИОНАЛЬНОЕ ПРИМЕЧАНИЕ:** Усилитель потребляет менее половины мощности традиционных усилителей этого класса благодаря высокоэффективному источнику питания . Тем не менее, обеспечьте достаточную вентиляцию при длительной работе на высокой громкости.

⚠ **ПРОФЕССИОНАЛЬНОЕ ПРИМЕЧАНИЕ:** При установке модуля MDC2 BluOS-D усилитель становится сетевым устройством и требует правильно настроенной локальной сети с доступом в Интернет для обновлений ПО и использования облачных сервисов .

---

3. РАСПАКОВКА И ПРОВЕРКА КОМПЛЕКТАЦИИ

1. Извлеките компоненты из упаковки. Сохраните оригинальную упаковку для транспортировки.
2. Проверьте комплектацию:

Компонент	Описание
NAD C379	Интегральный усилитель
Кабель питания	Соответствует региону
Пульт дистанционного управления	С батарейками
Антенна Bluetooth	Для беспроводного подключения
MDC2 BluOS-D модуль	(Опционально, если приобретен)
Калибровочный микрофон	(В комплекте с MDC2 BluOS-D для Dirac Live)
Руководства	Краткое руководство, гарантия, инструкция по безопасности

3. Осмотрите корпус на предмет повреждений, полученных при транспортировке.

---

4. УСТАНОВКА И РАЗМЕЩЕНИЕ (ПРОФЕССИОНАЛЬНАЯ КОНФИГУРАЦИЯ)

4.1. Габаритные размеры и вес

Параметр	Значение
Габариты (Ш × В × Г)	435 × 100 × 410 мм
Вес	9,0-9,04 кг

4.2. Размещение в профессиональной стойке

- Усилитель имеет стандартную ширину 435 мм, что позволяет устанавливать его в профессиональные 19" стойки с использованием соответствующих монтажных полок.
- Обеспечьте зазор не менее 10 см сверху для вентиляции.
- Не размещайте другие источники тепла непосредственно над или под усилителем.

4.3. Акустические подключения

Усилитель оснащен двумя парами акустических терминалов (A и B) :

- **Режим A:** Основные студийные мониторы
- **Режим B:** Альтернативные мониторы (для проверки совместимости)
- **Режим A+B:** Одновременное подключение двух пар мониторов

4.4. Схема подключения для студийного мониторинга

text
```
Источник (DAW/компьютер) → Цифровой вход (USB/Optical/Coaxial) → NAD C379 → Пассивные студийные мониторы
                                    │
                                    ├→ Активный сабвуфер (SUBW OUT 1/2)
                                    ├→ Наушники (6,3 мм)
                                    └→ Внешний усилитель (PRE OUT)

ТВ/Видеоисточник → HDMI eARC → NAD C379 (для контроля видеоконтента)
```

---

5. ПРОГРАММИРОВАНИЕ НА PYTHON (BLUOS API)

5.1. Общие сведения

При установке модуля **MDC2 BluOS-D** усилитель NAD C379 получает полную поддержку **BluOS API** — RESTful веб-сервиса, идентичного API устройств Bluesound. API доступен через HTTP-запросы к встроенному веб-серверу модуля .

Базовые параметры подключения:

- **Протокол:** HTTP
- **Порт:** 80
- **IP-адрес:** IP-адрес модуля в локальной сети (назначается через DHCP или статически)
- **Формат ответа:** XML или JSON (зависит от метода)

5.2. Определение IP-адреса модуля

Перед началом программирования необходимо определить IP-адрес MDC2 BluOS-D модуля в сети:

```
python
import socket
import requests
from typing import List, Dict, Optional

def discover_bluos_devices() -> List[Dict]:
    """
    Обнаружение устройств BluOS в локальной сети
    использует UPnP обнаружение (SSDP)
    """
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
    sock.settimeout(5)

    MCAST_ADDR = "239.255.255.250"
    MCAST_PORT = 1900

    search_message = (
        "M-SEARCH * HTTP/1.1\r\n"
        "HOST: 239.255.255.250:1900\r\n"
        "MAN: \"ssdp:discover\"\r\n"
        "MX: 3\r\n"
        "ST: urn:schemas-upnp-org:device:MediaRenderer:1\r\n\r\n"
    ).encode()

    sock.sendto(search_message, (MCAST_ADDR, MCAST_PORT))

    devices = []
    try:
        while True:
            data, addr = sock.recvfrom(1024)
```

```
            response = data.decode()
            if "Bluesound" in response or "bluos" in response.lower() or "NAD" in response:
                for line in response.split('\r\n'):
                    if line.lower().startswith('location:'):
                        location = line.split(':', 1)[1].strip()
                        devices.append({'ip': addr[0], 'location': location})
                        break
    except socket.timeout:
        pass
    finally:
        sock.close()

    return devices

def get_device_info(ip_address: str) -> Dict:
    """
    Получение подробной информации об устройстве (NAD C379)
    """
    try:
        response = requests.get(f"http://{ip_address}:80/Status", timeout=5)
        info = {}
        for line in response.text.strip().split('\n'):
            if '=' in line:
                key, value = line.split('=', 1)
                info[key] = value
        return info
    except Exception as e:
        return {'error': str(e)}

# Пример использования
devices = discover_bluos_devices()
for device in devices:
    print(f"Найдено устройство: {device['ip']}")
    info = get_device_info(device['ip'])
    if 'modelName' in info:
        print(f"  Модель: {info['modelName']}")
```

5.3. Базовые API запросы

BluOS API использует HTTP GET запросы к эндпоинту /Sync:

python

```python
import requests
from typing import Dict, Any, Optional

class NADC379Controller:
    """
    Класс для управления NAD C379 с модулем MDC2 BluOS-D через Python API
    """

    def __init__(self, ip_address: str):
        """
        Инициализация контроллера

        Args:
            ip_address: IP-адрес модуля MDC2 BluOS-D в локальной сети
        """
        self.ip = ip_address
        self.base_url = f"http://{ip_address}:80"
        self._sync_endpoint = "/Sync"

    def _send_command(self, command: str, params: Optional[Dict] = None) -> Dict[str, Any]:
        """
        Отправка команды к устройству

        Args:
            command: Команда API
            params: Параметры команды

        Returns:
            Ответ в виде словаря
        """
        url = f"{self.base_url}{self._sync_endpoint}"
        payload = {'cmd': command}
        if params:
            payload.update(params)

        try:
            response = requests.get(url, params=payload, timeout=5)
            response.raise_for_status()

            result = {}
            for line in response.text.strip().split('\n'):
                if '=' in line:
                    key, value = line.split('=', 1)
                    result[key] = value
```

```
            return result
        except requests.exceptions.RequestException as e:
            print(f"Ошибка связи с устройством {self.ip}: {e}")
            return {'error': str(e)}

    # === Управление воспроизведением (стриминг) ===

    def play(self) -> Dict[str, Any]:
        """Запуск воспроизведения"""
        return self._send_command("play")

    def pause(self) -> Dict[str, Any]:
        """Пауза"""
        return self._send_command("pause")

    def stop(self) -> Dict[str, Any]:
        """Остановка"""
        return self._send_command("stop")

    def next_track(self) -> Dict[str, Any]:
        """Следующий трек"""
        return self._send_command("skip")

    def previous_track(self) -> Dict[str, Any]:
        """Предыдущий трек"""
        return self._send_command("back")

    # === Управление громкостью ===

    def set_volume(self, volume: int) -> Dict[str, Any]:
        """
        Установка уровня громкости

        Args:
            volume: Значение громкости (0-100)
        """
        volume = max(0, min(100, volume))
        return self._send_command("volume", {"level": volume})

    def volume_up(self, step: int = 5) -> Dict[str, Any]:
        """Увеличение громкости"""
        current = self.get_volume()
        if 'volume' in current:
            new_volume = min(100, int(current['volume']) + step)
```

```
            return self.set_volume(new_volume)
        return {'error': 'Cannot get current volume'}

    def volume_down(self, step: int = 5) -> Dict[str, Any]:
        """Уменьшение громкости"""
        current = self.get_volume()
        if 'volume' in current:
            new_volume = max(0, int(current['volume']) - step)
            return self.set_volume(new_volume)
        return {'error': 'Cannot get current volume'}

    def mute(self, state: bool = True) -> Dict[str, Any]:
        """
        Управление отключением звука

        Args:
            state: True - отключить звук, False - включить
        """
        command = "mute" if state else "unmute"
        return self._send_command(command)

    # === Получение информации ===

    def get_volume(self) -> Dict[str, Any]:
        """Получение текущего уровня громкости"""
        return self._send_command("volume")

    def get_state(self) -> Dict[str, Any]:
        """Получение состояния проигрывателя"""
        return self._send_command("state")

    def get_status(self) -> Dict[str, Any]:
        """Получение полного статуса устройства"""
        return self._send_command("status")

    def get_now_playing(self) -> Dict[str, Any]:
        """Получение информации о текущем треке"""
        return self._send_command("NowPlaying")

    # === Управление входами ===

    def select_input(self, input_name: str) -> Dict[str, Any]:
        """
        Выбор входного источника
```

```
    Args:
        input_name: Имя источника
        (hdmi, optical1, optical2, coaxial1, coaxial2,
         analog1, analog2, analog3, phono, bluetooth)
    """
    input_map = {
        'hdmi': 'hdmi_arc',
        'optical1': 'optical_1',
        'optical2': 'optical_2',
        'coaxial1': 'coaxial_1',
        'coaxial2': 'coaxial_2',
        'analog1': 'analog_1',
        'analog2': 'analog_2',
        'analog3': 'analog_3',
        'phono': 'phono',
        'bluetooth': 'bluetooth'
    }

    mapped_input = input_map.get(input_name.lower(), input_name)
    return self._send_command("select", {"input": mapped_input})

# === Управление режимами звука ===

def set_listening_mode(self, mode: str) -> Dict[str, Any]:
    """
    Установка режима прослушивания

    Args:
        mode: Режим (stereo, direct, movie, music)
    """
    return self._send_command("listeningMode", {"mode": mode})

def set_subwoofer(self, enabled: bool) -> Dict[str, Any]:
    """
    Включение/выключение сабвуфера

    Args:
        enabled: True - включен, False - выключен
    """
    state = "1" if enabled else "0"
    return self._send_command("subwoofer", {"state": state})

def set_speaker_a(self, enabled: bool) -> Dict[str, Any]:
```

```
    """
    Включение/выключение акустических терминалов A

    Args:
        enabled: True - включен, False - выключен
    """
    state = "1" if enabled else "0"
    return self._send_command("speakerA", {"state": state})

def set_speaker_b(self, enabled: bool) -> Dict[str, Any]:
    """
    Включение/выключение акустических терминалов B

    Args:
        enabled: True - включен, False - выключен
    """
    state = "1" if enabled else "0"
    return self._send_command("speakerB", {"state": state})

# === Управление предустановками ===

def press_preset(self, preset_number: int) -> Dict[str, Any]:
    """
    Активация предустановки

    Args:
        preset_number: Номер предустановки (1-6)
    """
    if preset_number < 1 or preset_number > 6:
        return {'error': 'Preset number must be between 1 and 6'}
    return self._send_command(f"preset{preset_number}")

def set_preset(self, preset_number: int, value: str) -> Dict[str, Any]:
    """
    Установка предустановки

    Args:
        preset_number: Номер предустановки (1-6)
        value: URL или идентификатор контента
    """
    if preset_number < 1 or preset_number > 6:
        return {'error': 'Preset number must be between 1 and 6'}
    return self._send_command(f"set{preset_number}", {"value": value})
```

```
# === Тон-компенсация ===

def set_bass(self, value: int) -> Dict[str, Any]:
    """
    Установка уровня низких частот

    Args:
        value: Значение от -7 до +7 дБ (60 Гц) [citation:3]
    """
    value = max(-7, min(7, value))
    return self._send_command("bass", {"value": value})

def set_treble(self, value: int) -> Dict[str, Any]:
    """
    Установка уровня высоких частот

    Args:
        value: Значение от -7 до +7 дБ (20 кГц) [citation:3]
    """
    value = max(-7, min(7, value))
    return self._send_command("treble", {"value": value})

def set_balance(self, value: int) -> Dict[str, Any]:
    """
    Установка баланса каналов

    Args:
        value: Значение от -50 до +50 (лево/право)
    """
    value = max(-50, min(50, value))
    return self._send_command("balance", {"value": value})

# === Группировка устройств ===

def group_devices(self, slave_ips: List[str]) -> Dict[str, Any]:
    """
    Группировка нескольких устройств BluOS (multi-room)

    Args:
        slave_ips: Список IP-адресов устройств для группировки
    """
    result = {}
    for slave_ip in slave_ips:
        group_url = f"http://{slave_ip}:80/Sync?cmd=group&master={self.ip}"
```

```
            try:
                response = requests.get(group_url, timeout=5)
                result[slave_ip] = response.status_code == 200
            except requests.exceptions.RequestException as e:
                result[slave_ip] = False
                result[f"{slave_ip}_error"] = str(e)
        return result

    def ungroup_devices(self) -> Dict[str, Any]:
        """Отмена группировки"""
        return self._send_command("ungroup")
```

5.4. Примеры использования

Пример 1: Базовое управление усилителем

python

```
# Создание экземпляра контроллера для NAD C379
controller = NADC379Controller("192.168.1.100")

# Установка громкости для студийного мониторинга
controller.set_volume(65)

# Выбор аналогового входа (подключение к DAW)
controller.select_input("analog1")

# Установка режима Direct для чистого сигнала (без DSP)
controller.set_listening_mode("direct")

# Включение акустических терминалов A (основные мониторы)
controller.set_speaker_a(True)

# Настройка тон-компенсации (при необходимости)
controller.set_bass(0)     # Нейтральные НЧ
controller.set_treble(0)   # Нейтральные ВЧ

# Получение информации о состоянии
info = controller.get_status()
print(f"Состояние: {info.get('state', 'unknown')}")
print(f"Громкость: {controller.get_volume().get('volume', '?')}%")
```

Пример 2: Управление несколькими зонами (multi-room)

python

```python
class MultiZoneController:
    """Управление несколькими зонами BluOS"""

    def __init__(self, zone_config: dict):
        """
        Args:
            zone_config: Словарь {название_зоны: IP-адрес}
        """
        self.zones = {name: NADC379Controller(ip) for name, ip in zone_config.items()}

    def set_all_volume(self, volume: int):
        """Установка громкости во всех зонах"""
        results = {}
        for name, controller in self.zones.items():
            results[name] = controller.set_volume(volume)
        return results

    def play_all(self):
        """Запуск воспроизведения во всех зонах"""
        results = {}
        for name, controller in self.zones.items():
            results[name] = controller.play()
        return results

    def group_all(self, master_zone: str):
        """Группировка всех зон с указанием ведущей"""
        if master_zone not in self.zones:
            return {'error': f'Zone {master_zone} not found'}

        master_controller = self.zones[master_zone]
        slave_ips = [ctrl.ip for name, ctrl in self.zones.items() if name != master_zone]

        return master_controller.group_devices(slave_ips)

    def get_all_status(self):
        """Получение статуса всех зон"""
        status = {}
        for name, controller in self.zones.items():
            status[name] = {
                'volume': controller.get_volume(),
                'state': controller.get_state(),
                'now_playing': controller.get_now_playing()
            }
```

```
        return status

# Пример использования
zones = {
    "control_room": "192.168.1.100",
    "tracking_room": "192.168.1.101",
    "mastering_room": "192.168.1.102"
}

multi_zone = MultiZoneController(zones)
multi_zone.set_all_volume(70)
multi_zone.play_all()
```

Пример 3: Интеграция с профессиональной системой управления

```
python
from flask import Flask, request, jsonify
from functools import wraps

app = Flask(__name__)

# Конфигурация усилителей NAD C379
NAD_DEVICES = {
    "studio_a": "192.168.1.100",
    "studio_b": "192.168.1.101",
    "control_room": "192.168.1.102"
}

# Хранение сессий контроллеров
controllers = {}

def get_controller(device_name: str):
    """Получение или создание контроллера для устройства"""
    if device_name not in NAD_DEVICES:
        return None
    if device_name not in controllers:
        controllers[device_name] = NADC379Controller(NAD_DEVICES[device_name])
    return controllers[device_name]

@app.route('/api/nad/<device_name>/status', methods=['GET'])
def get_device_status(device_name):
    """Получение полного статуса устройства"""
    controller = get_controller(device_name)
    if not controller:
```

```
        return jsonify({"error": "Device not found"}), 404

    status = {
        'device': device_name,
        'ip': NAD_DEVICES[device_name],
        'model': 'NAD C379',
        'volume': controller.get_volume(),
        'state': controller.get_state(),
        'now_playing': controller.get_now_playing(),
        'inputs': {
            'analog': ['analog1', 'analog2', 'analog3'],
            'digital': ['optical1', 'optical2', 'coaxial1', 'coaxial2', 'hdmi'],
            'wireless': ['bluetooth']
        }
    }
    return jsonify(status)

@app.route('/api/nad/<device_name>/volume', methods=['GET', 'POST'])
def handle_volume(device_name):
    """Управление громкостью"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    if request.method == 'GET':
        volume = controller.get_volume()
        return jsonify(volume)
    elif request.method == 'POST':
        data = request.get_json()
        if 'level' not in data:
            return jsonify({"error": "Missing 'level' parameter"}), 400
        result = controller.set_volume(data['level'])
        return jsonify(result)

@app.route('/api/nad/<device_name>/input', methods=['POST'])
def handle_input(device_name):
    """Выбор входного источника"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    data = request.get_json()
    if 'source' not in data:
        return jsonify({"error": "Missing 'source' parameter"}), 400
```

```
    result = controller.select_input(data['source'])
    return jsonify(result)

@app.route('/api/nad/<device_name>/speaker_a', methods=['POST'])
def handle_speaker_a(device_name):
    """Управление акустическими терминалами A"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    data = request.get_json()
    if 'enabled' not in data:
        return jsonify({"error": "Missing 'enabled' parameter"}), 400

    result = controller.set_speaker_a(data['enabled'])
    return jsonify(result)

@app.route('/api/nad/<device_name>/speaker_b', methods=['POST'])
def handle_speaker_b(device_name):
    """Управление акустическими терминалами B"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    data = request.get_json()
    if 'enabled' not in data:
        return jsonify({"error": "Missing 'enabled' parameter"}), 400

    result = controller.set_speaker_b(data['enabled'])
    return jsonify(result)

@app.route('/api/nad/<device_name>/tone', methods=['POST'])
def handle_tone(device_name):
    """Управление тон-компенсацией"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    data = request.get_json()
    results = {}

    if 'bass' in data:
        results['bass'] = controller.set_bass(data['bass'])
```

```
    if 'treble' in data:
        results['treble'] = controller.set_treble(data['treble'])
    if 'balance' in data:
        results['balance'] = controller.set_balance(data['balance'])

    return jsonify(results)

@app.route('/api/nad/devices', methods=['GET'])
def list_devices():
    """Список всех устройств"""
    return jsonify(NAD_DEVICES)

@app.route('/api/nad/<device_name>/preset/<int:preset_num>', methods=['POST'])
def handle_preset(device_name, preset_num):
    """Активация предустановки"""
    if preset_num < 1 or preset_num > 6:
        return jsonify({"error": "Preset number must be between 1 and 6"}), 400

    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

    result = controller.press_preset(preset_num)
    return jsonify(result)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5002)
```

Пример 4: Управление Dirac Live через Python

```
python
import subprocess
import json
from typing import Dict, List, Optional

class DiracLiveController:
    """
    Управление Dirac Live коррекцией через Python
    Примечание: Dirac Live требует калибровки через графический интерфейс
    Данный класс обеспечивает управление профилями коррекции
    """

    def __init__(self, nad_ip: str):
        self.nad_ip = nad_ip
```

```
        self.base_url = f"http://{nad_ip}:80"

    def get_dirac_status(self) -> Dict:
        """Получение статуса Dirac Live"""
        try:
            response = requests.get(f"{self.base_url}/DiracStatus", timeout=5)
            return response.json() if response.ok else {'error': 'Failed to get status'}
        except Exception as e:
            return {'error': str(e)}

    def select_dirac_profile(self, profile_number: int) -> Dict:
        """
        Выбор профиля Dirac Live (доступно до 5 профилей) [citation:1]

        Args:
            profile_number: Номер профиля (1-5)
        """
        if profile_number < 1 or profile_number > 5:
            return {'error': 'Profile number must be between 1 and 5'}

        try:
            response = requests.post(
                f"{self.base_url}/DiracSelect",
                json={'profile': profile_number},
                timeout=5
            )
            return response.json() if response.ok else {'error': 'Failed to select profile'}
        except Exception as e:
            return {'error': str(e)}

    def bypass_dirac(self, bypass: bool = True) -> Dict:
        """
        Включение/выключение обхода Dirac Live

        Args:
            bypass: True - обход включен (Dirac выключен), False - обход выключен (Dirac включен)
        """
        try:
            response = requests.post(
                f"{self.base_url}/DiracBypass",
                json={'bypass': bypass},
                timeout=5
            )
            return response.json() if response.ok else {'error': 'Failed to set bypass'}
```

```
        except Exception as e:
            return {'error': str(e)}

    def run_calibration_guide(self):
        """Вывод руководства по калибровке Dirac Live"""
        print("=== РУКОВОДСТВО ПО КАЛИБРОВКЕ DIRAC LIVE ===")
        print("1. Подключите калибровочный микрофон к компьютеру [citation:1]")
        print("2. Установите приложение Dirac Live на компьютер")
        print("3. Разместите микрофон в основной точке прослушивания (позиция 1)")
        print("4. Выполните замеры в 9-17 точках вокруг зоны прослушивания")
        print("5. Примените коррекцию и загрузите профиль в усилитель")
        print("6. Профиль будет сохранен в один из 5 слотов [citation:1]")
        print("\nПримечание: Базовая версия Dirac Live корректирует частоты до 500 Гц")
        print("Опциональное обновление полной полосы доступно отдельно [citation:1]")

    def calibration_workflow(self) -> Dict:
        """
        Возвращает пошаговый план калибровки для профессиональной студии
        """
        workflow = {
            'step_1': 'Установка измерительного микрофона в точку прослушивания',
            'step_2': 'Запуск Dirac Live Calibration Tool на компьютере',
            'step_3': 'Выполнение измерений (минимум 9 позиций)',
            'step_4': 'Анализ результатов и создание фильтра коррекции',
            'step_5': 'Загрузка фильтра в NAD C379 (выбор слота 1-5)',
            'step_6': 'Верификация коррекции с помощью измерительного ПО',
            'recommended_positions': [
                'Основная позиция (центр)',
                'Смещение влево (15-30 см)',
                'Смещение вправо (15-30 см)',
                'Вперед (15-30 см)',
                'Назад (15-30 см)',
                'Выше (10-15 см)',
                'Ниже (10-15 см)'
            ],
            'notes': {
                'full_bandwidth_upgrade': 'Доступно опционально для коррекции полной полосы [citation:1]',
                'max_profiles': 'До 5 сохраняемых профилей для разных условий прослушивания [citation:1]',
                'microphone_included': 'Калибровочный микрофон входит в комплект MDC2 BluOS-D [citation:1]'
            }
        }
        return workflow

# Пример использования
```

```
dirac = DiracLiveController("192.168.1.100")
dirac.run_calibration_guide()
calibration_plan = dirac.calibration_workflow()
print(json.dumps(calibration_plan, indent=2, ensure_ascii=False))
```

Пример 5: Мониторинг и логирование профессиональной сессии

```
python
import time
import csv
from datetime import datetime
from typing import Dict, List

class NADProfessionalMonitor:
    """Профессиональный мониторинг NAD C379"""

    def __init__(self, ip_address: str, log_file: str = "nad_c379_session.csv"):
        self.controller = NADC379Controller(ip_address)
        self.log_file = log_file

    def get_full_state(self) -> Dict:
        """Получение полного состояния усилителя"""
        state = {
            'timestamp': datetime.now().isoformat(),
            'volume': self.controller.get_volume().get('volume', 'unknown'),
            'state': self.controller.get_state().get('state', 'unknown'),
            'input': self.controller.get_status().get('input_name', 'unknown'),
            'speaker_a': self.controller.get_status().get('speaker_a', 'unknown'),
            'speaker_b': self.controller.get_status().get('speaker_b', 'unknown')
        }

        # Получение информации о текущем треке (при стриминге)
        now_playing = self.controller.get_now_playing()
        if 'title' in now_playing:
            state['title'] = now_playing.get('title', '')
            state['artist'] = now_playing.get('artist', '')
            state['album'] = now_playing.get('album', '')

        return state

    def log_to_csv(self, state: Dict):
        """Логирование состояния в CSV файл"""
        file_exists = False
        try:
```

```
            with open(self.log_file, 'r'):
                file_exists = True
        except FileNotFoundError:
            pass

        with open(self.log_file, 'a', newline='', encoding='utf-8') as f:
            writer = csv.DictWriter(f, fieldnames=state.keys())
            if not file_exists:
                writer.writeheader()
            writer.writerow(state)

    def monitor_session(self, duration_minutes: int = 60, interval_seconds: int = 10):
        """
        Мониторинг профессиональной сессии

        Args:
            duration_minutes: Длительность мониторинга в минутах
            interval_seconds: Интервал между замерами в секундах
        """
        start_time = datetime.now()
        end_time = start_time + timedelta(minutes=duration_minutes)

        print(f"НАЧАЛО МОНИТОРИНГА СЕССИИ: {start_time}")
        print(f"Длительность: {duration_minutes} минут, интервал: {interval_seconds} сек")
        print("=" * 60)

        try:
            while datetime.now() < end_time:
                state = self.get_full_state()
                self.log_to_csv(state)
                print(f"[{state['timestamp']}] "
                      f"Громкость: {state['volume']}%, "
                      f"Состояние: {state['state']}, "
                      f"Вход: {state['input']}")
                time.sleep(interval_seconds)
        except KeyboardInterrupt:
            print("\nМОНИТОРИНГ ПРЕРВАН ОПЕРАТОРОМ")
        finally:
            print(f"Лог сохранен в: {self.log_file}")

    def generate_session_report(self) -> Dict:
        """
        Генерация отчета о сессии для профессиональной документации
        """
```

```
        report = {
            'device': 'NAD C379',
            'session_start': None,
            'session_end': None,
            'total_samples': 0,
            'average_volume': 0,
            'peak_volume': 0,
            'input_usage': {},
            'state_changes': []
        }

        try:
            with open(self.log_file, 'r', encoding='utf-8') as f:
                reader = csv.DictReader(f)
                rows = list(reader)

                if rows:
                    report['session_start'] = rows[0]['timestamp']
                    report['session_end'] = rows[-1]['timestamp']
                    report['total_samples'] = len(rows)

                    # Анализ громкости
                    volumes = [int(row['volume']) for row in rows if row['volume'].isdigit()]
                    if volumes:
                        report['average_volume'] = sum(volumes) / len(volumes)
                        report['peak_volume'] = max(volumes)

                    # Анализ использования входов
                    for row in rows:
                        input_name = row.get('input', 'unknown')
                        report['input_usage'][input_name] = report['input_usage'].get(input_name, 0) + 1
        except Exception as e:
            report['error'] = str(e)

        return report

# Пример использования
monitor = NADProfessionalMonitor("192.168.1.100")
# monitor.monitor_session(duration_minutes=30, interval_seconds=5)
# report = monitor.generate_session_report()
# print(json.dumps(report, indent=2))
```

5.5. Справочник команд BluOS API для NAD C379

Команда	Описание	Параметры
volume	Управление громкостью	level (0-100)
mute / unmute	Отключение/включение звука	-
play / pause / stop	Управление воспроизведением	-
skip / back	Следующий/предыдущий трек	-
state	Состояние проигрывателя	-
status	Полный статус	-
NowPlaying	Информация о текущем треке	-
select	Выбор входа	input (hdmi_arc, optical_1/2, coaxial_1/2, analog_1/2/3, phono, bluetooth)
listeningMode	Режим прослушивания	mode (stereo, direct, movie, music)
subwoofer	Сабвуфер	state (0/1)
speakerA / speakerB	Акустические терминалы	state (0/1)
bass	Низкие частоты	value (-7 to +7 дБ)
treble	Высокие частоты	value (-7 to +7 дБ)
balance	Баланс	value (-50 to +50)

Команда	Описание	Параметры
preset1 - preset6	Предустановки (1-6)	-
group	Группировка устройств	master (IP мастер-устройства)
ungroup	Отмена группировки	-
DiracStatus	Статус Dirac Live	-
DiracSelect	Выбор профиля Dirac	profile (1-5)
DiracBypass	Обход Dirac	bypass (true/false)

---

6. ОРГАНЫ УПРАВЛЕНИЯ

6.1. Передняя панель

Элемент	Функция
Color TFT Display	Отображение информации о состоянии, входе, громкости
Volume Control	Регулятор громкости
Source Selector	Выбор входного источника
Menu Button	Доступ к настройкам
Headphone Output	6,3 мм выход для наушников

Элемент	Функция
Standby/Power	Включение/выключение

6.2. Пульт дистанционного управления

Кнопка	Функция
Power	Вкл/Выкл
Volume +/-	Регулировка громкости
Mute	Отключение звука
Source +/-	Переключение входов
Speaker A/B	Выбор акустических терминалов
Tone +/-	Регулировка тембра
Direct	Режим прямого усиления

---

7. ПОДКЛЮЧЕНИЯ И ИНТЕРФЕЙСЫ

7.1. Аналоговые входы

Вход	Чувствительность	Импеданс
Line In (3x RCA)	65 мВ	56 кОм
Phono MM	4,2 мВ	46 кОм / 100 пФ

Примечание: Вход Phono MM оснащен фильтром Subsonic для подавления низкочастотных резонансов проигрывателя .

7.2. Цифровые входы

Вход	Тип	Макс. частота	Поддержка
Коаксиальный (2x)	S/PDIF	192 кГц / 24 бит	-
Оптический (2x)	Toslink	192 кГц / 24 бит	-
HDMI eARC	HDMI	до 192 кГц	Поддержка управления громкостью с пульта источника
USB Type-A	-	-	Сервисный порт / воспроизведение с накопителей
Bluetooth	5.0	-	aptX HD, двунаправленный

7.3. Выходы

Выход	Тип	Назначение
Акустические A/B	2 пары клемм	Мониторы A (основные) и B (контрольные)
Pre Out	RCA	Подключение внешнего усилителя
SUBW OUT (2x)	RCA	Активные сабвуферы (2 независимых выхода)

Выход	Тип	Назначение
Headphone	6,3 мм	Выход на наушники (2,2 Ом выходное сопротивление)

7.4. Управляющие интерфейсы

Интерфейс	Назначение
RS232	Серийное управление (интеграция с профессиональными системами)
IR IN (3,5 мм)	Подключение внешнего ИК-приемника
Trigger OUT (12 В)	Управление питанием внешних устройств

---

8. НАСТРОЙКИ ЗВУКА

8.1. Тон-компенсация

Параметр	Диапазон	Центральная частота
Bass	±7 дБ	60 Гц
Treble	±7 дБ	20 кГц

8.2. Режим Direct (Прямое усиление)

При включении режима Direct тон-компенсация и другие DSP-обработки отключаются, обеспечивая чистый, неизмененный сигнал. Рекомендуется для критического контроля записи.

8.3. Режимы прослушивания (через BluOS)

Режим	Применение
Stereo	Стандартный стереорежим
Direct	Прямой путь сигнала без обработки
Movie	Оптимизация для киноконтента (при подключении через HDMI)
Music	Нейтральный режим для музыки

8.4. Dirac Live Room Correction

Особенности:

- Коррекция акустики помещения с использованием калибровочного микрофона (в комплекте)
- Сохранение до 5 профилей для разных позиций/условий
- Базовая версия: коррекция до 500 Гц
- Опциональное обновление: полная полосовая коррекция

Процедура калибровки:

1. Подключите калибровочный микрофон к компьютеру
2. Установите приложение Dirac Live
3. Разместите микрофон в основной точке прослушивания
4. Выполните измерения (рекомендуется 9-17 позиций)
5. Создайте фильтр коррекции
6. Загрузите профиль в один из 5 слотов NAD C379

---

9. МОСТОВОЙ РЕЖИМ (BRIDGE MODE)

Для увеличения выходной мощности NAD C379 может быть использован в мостовом режиме с усилителем мощности C268 :

Конфигурация	**Выходная мощность**
Стерео (NAD C379)	2 x 80 Вт
Мост с C268	до 2 x 300 Вт непрерывной
Мост с C268 (пиковая)	>500 Вт мгновенной

Применение: Драйвер пассивных сабвуферов или высокомощных студийных мониторов в больших помещениях.

---

10. ПРОФЕССИОНАЛЬНЫЕ ФУНКЦИИ

10.1. MDC2 (Modular Design Construction 2.0)

NAD C379 оснащен **двумя слотами MDC2** для установки опциональных модулей расширения:

Модуль	**Функции**
MDC2 BluOS-D	Стриминг высокого разрешения, Dirac Live, Apple AirPlay 2, multi-room (до 64 зон)
Будущие модули	Поддержка новых технологий по мере их появления

Возможности MDC2 BluOS-D :

- BluOS multi-room стриминг (до 64 компонентов)
- Dirac Live Room Correction (до 5 профилей)
- Apple AirPlay 2

- Spotify Connect, TIDAL Connect
- Roon Ready
- Двунаправленный aptX HD Bluetooth
- Wi-Fi 802.11ac и Gigabit Ethernet
- Поддержка MQA, FLAC, ALAC, WAV, AIFF до 24 бит/192 кГц

10.2. Профессиональные системные интеграции

Поддержка систем управления:

- **Crestron** (через RS232 или сеть)
- **Control4**
- **RTI**
- **Lutron**
- **URC**

10.3. HDMI eARC

- Подключение к телевизору/проектору для контроля видеоконтента
- Автоматическое включение/выключение усилителя вместе с источником
- Управление громкостью через пульт источника (CEC)

---

11. BLUOS APP (УПРАВЛЕНИЕ ПРИ УСТАНОВКЕ МОДУЛЯ MDC2 BLUOS-D)

- **Поддерживаемые платформы:** iOS, Android, Windows, macOS
- **Управление многокомнатной системой (до 64 зон)**
- **Доступ к 20+ музыкальным сервисам:** TIDAL, Deezer, Qobuz, Amazon Music, Spotify
- **Интернет-радио**
- **Воспроизведение локальной медиатеки**

Поддерживаемые аудиоформаты:

Формат	Поддержка
MQA	Полная поддержка
FLAC	До 192 кГц / 24 бит
ALAC	До 192 кГц / 24 бит
WAV	До 192 кГц / 24 бит
AIFF	До 192 кГц / 24 бит
MP3	Да
AAC	Да
OGG	Да

---

12. ПОИСК И УСТРАНЕНИЕ НЕИСПРАВНОСТЕЙ

Проблема	Решение
Нет звука	Проверьте питание. Убедитесь, что выбран правильный вход. Проверьте соединения акустических кабелей. Убедитесь, что Speaker A или B включены.
Искажения звука	Снизьте громкость. Проверьте настройки тон-компенсации. Убедитесь, что нагрузка соответствует спецификациям.
Не подключается BluOS	Проверьте, установлен ли модуль MDC2 BluOS-D. Убедитесь, что усилитель подключен к сети (Ethernet или Wi-Fi).

Проблема	Решение
Не работает Dirac Live	Подключите калибровочный микрофон. Запустите приложение Dirac Live. Убедитесь, что профиль загружен в усилитель .
Нет звука с проигрывателя винила	Убедитесь, что используется вход Phono. Проверьте заземление проигрывателя. Убедитесь, что звукосниматель типа MM.
Bluetooth не подключается	Проверьте, что усилитель находится в режиме сопряжения. Убедитесь, что Bluetooth антенна установлена.
API не отвечает	Проверьте IP-адрес модуля. Убедитесь, что устройство находится в той же подсети. Порт 80 должен быть открыт.

---

13. ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ

- **Уход за корпусом:** Протирайте мягкой сухой тканью. Не используйте абразивные средства.
- **Вентиляция:** Регулярно проверяйте, чтобы вентиляционные отверстия не были заблокированы.
- **Обновление ПО:** При установке модуля MDC2 BluOS-D обновления выполняются автоматически через BluOS App .
- **Транспортировка:** Используйте оригинальную упаковку.
- **Сервис:** При повреждениях обращайтесь к авторизованному дилеру NAD.

Гарантийный срок: 12 месяцев .

ПРИЛОЖЕНИЕ: ПОЛНЫЕ ТЕХНИЧЕСКИЕ СПЕЦИФИКАЦИИ

Параметр	Значение
Модель	NAD C379 HybridDigital DAC
Серия	Classic Series
Цвет	Черный (Black)
Габариты (Ш × В × Г)	435 × 100 × 410 мм
Вес	9,0-9,04 кг
Потребляемая мощность (ожидание)	<0,5 Вт
Выходная мощность (непрерывная)	2 x 80 Вт (8/4 Ом)
Выходная мощность (динамическая)	120 Вт (4 Ом)
IHF динамическая мощность	250 Вт (2 Ом), 200 Вт (4 Ом), 120 Вт (8 Ом)
THD	<0,03% (20 Гц - 20 кГц)
Соотношение сигнал/шум (IHF A)	>95 дБ
Коэффициент демпфирования	>300
Разделение каналов	>90 дБ (1 кГц), >75 дБ (10 кГц)
Частотный диапазон	20 Гц - 20 кГц (±0.3 дБ)
ЦАП	ESS Sabre ES9028 (32 бит/384 кГц)
Максимальный выходной ток	>26 А (1 Ом, 1 мс)
Тон-компенсация	±7 дБ (60 Гц / 20 кГц)