ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ NAD C700

Данный интегральный усилитель разработан компанией NAD Electronics (Канада) для профессионального и коммерческого применения (требуется интеграция с профессиональными системами управления). Усилитель не предназначен для использования в жилых помещениях в качестве бытового аудиооборудования. интеграцию с профессиональными системами управления.

NAD C 700
BluOS Streaming Amplifier

Сетевой стриминг-усилитель NAD C 700 (включая версию V2)

Компактный гибридный усилитель/ЦАП для студийного мониторинга и пост-продакшн

ЦЕЛЕВОЕ НАЗНАЧЕНИЕ (ПРОФЕССИОНАЛЬНОЕ НЕБЫТОВОЕ ПРИМЕНЕНИЕ)

Сетевой стриминг-усилитель NAD C 700 (включая версию V2) **предназначен исключительно для профессионального небытового использования** .

Основные профессиональные сценарии применения:

Сфера применения	Тип задач
Малые студии звукозаписи (project studios)	Контроль записи (tracking), сведение (mixing), мониторинг через пассивные мониторы
Монтажные комнаты (edit suites)	Предварительное сведение и контроль совместимости
Пост-продакшн студии	Контроль микса для телевидения и стриминговых платформ
Радиостанции и подкастинговые студии	Речевой мониторинг, контроль эфира
Образовательные учреждения	Лаборатории аудиотехники, классы звукорежиссуры

Сфера применения	Тип задач
Профессиональные демонстрационные зоны	Демонстрация контента в форматах высокой четкости
Системы 4.1 Dolby Digital Surround для контроля многоканального контента	Оценка объемного звука в профессиональных целях

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.
2. **ESS Sabre ES9028 ЦАП:** 32-бит/192 кГц конвертер высокой четкости, обеспечивающий широкий динамический диапазон и низкий уровень шума для точного контроля записи .
3. **Dirac Live Ready:** Готовность к коррекции акустики помещения (лицензия приобретается отдельно) .
4. **Многоканальный контроль 4.1:** Поддержка Dolby Digital для оценки surround-миксов через беспроводные тыловые каналы BluOS и проводной сабвуфер .
5. **Профессиональные системные интеграции:** Поддержка Crestron, Control4, RTI, URC, Elan, Lutron, iPort, KNX, PUSH .
6. **Компактный форм-фактор:** Идеальное решение для монтажных комнат и передвижных студий с ограниченным пространством .

---

1. ВВЕДЕНИЕ И ОБЩИЕ СВЕДЕНИЯ

Настоящая инструкция предназначена для **квалифицированного персонала**, использующего сетевой стриминг-усилитель NAD C 700 V2 в профессиональных целях. Усилитель построен на фирменной технологии HybridDigital UcD (класс D) и оснащен улучшенным ЦАП ESS Sabre ES9028, ранее доступным только в флагманской серии Masters.

Основные технические характеристики:

Параметр	Значение
Выходная мощность (непрерывная)	2 x 80 Вт (8/4 Ом) / 100 Вт (4 Ом)
Выходная мощность (пиковая)	2 x 120 Вт
IHF динамическая мощность	100 Вт (8 Ом), 125 Вт (4 Ом)
THD+N	<0,04% (20 Гц - 20 кГц)
Соотношение сигнал/шум	>84 дБ
Коэффициент демпфирования	>90
Частотный диапазон	20 Гц - 20 кГц (±0.18 дБ)
ЦАП	ESS Sabre ES9028 (32 бит/192 кГц)
Разделение каналов	>93 дБ (1 кГц), >72 дБ (10 кГц)
Тон-компенсация	ВЧ: ±6,0 дБ (20 кГц), НЧ: ±6,0 дБ (60 Гц)
Потребляемая мощность (ожидание)	<0,5 Вт

Подключения и разъемы:

Входы:

Тип	Количество	Назначение
HDMI eARC	1	Подключение к источникам (ТВ, проекторы) с поддержкой Dolby Digital

Тип	Количество	Назначение
Коаксиальный S/PDIF (RCA)	1	Цифровой вход
Оптический Toslink	1	Цифровой вход
Аналоговый RCA (Line)	1 пара	Линейный вход
USB Type-A	1	Воспроизведение с накопителей (FAT32/NTFS) и подключение калибровочного микрофона
Bluetooth	1	aptX HD (двунаправленный)

Выходы:

Тип	Количество	Назначение
Акустические терминалы	1 пара	Подключение пассивных мониторов
PRE OUT (RCA)	1	Выход на внешний усилитель
Сабвуферный выход (RCA)	1	Подключение активного сабвуфера
Наушники	6,3 мм	Выход на наушники (Bluetooth передача также доступна)

Управление и интеграция:

Тип	Назначение
RS232	Серийное управление

Тип	Назначение
IR IN (3,5 мм)	Вход для ИК-приемника
IR OUT (3,5 мм)	Выход ИК-сигнала
Trigger OUT (12 В)	Управление питанием внешних устройств
Wi-Fi	802.11ac (2,4/5 ГГц)
Ethernet	Gigabit (RJ45)

---

2. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ

Перед эксплуатацией системы ознакомьтесь со следующими требованиями:

- **Электропитание:** Используйте только прилагаемый кабель питания. Напряжение питания: соответствует региону. Защищайте кабель питания от повреждений.
- **Вентиляция:** Обеспечьте свободный доступ воздуха вокруг усилителя. Не блокируйте вентиляционные отверстия.
- **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 использует высокоэффективный источник питания. Тем не менее, обеспечьте достаточную вентиляцию при длительной работе на высокой громкости.
⚠ **ПРОФЕССИОНАЛЬНОЕ ПРИМЕЧАНИЕ:** Усилитель является сетевым устройством и требует правильно настроенной локальной сети с доступом в Интернет для обновлений ПО и использования облачных сервисов.
⚠ **ПРИМЕЧАНИЕ ПО DIRAC LIVE:** Усилитель готов к работе с Dirac Live, но для использования требуется приобретение отдельной лицензии (Limited Bandwidth включена? требуется уточнение у производителя) . Функция Dirac Live для C700 V2 может находиться в стадии разработки — проверьте актуальный статус на сайте производителя .

3. РАСПАКОВКА И ПРОВЕРКА КОМПЛЕКТАЦИИ

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

Компонент	Описание
NAD C 700 V2	Сетевой стриминг-усилитель
Кабель питания	Соответствует региону
Пульт дистанционного управления BC1 IR	С батарейками
Антенны Wi-Fi/Bluetooth	Для беспроводного подключения
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)	218 × 96 × 266 мм
Вес	4,8 кг

4.2. Размещение в профессиональной среде

- Усилитель имеет компактные размеры, что позволяет устанавливать его в ограниченных пространствах монтажных комнат и передвижных студий .
- Обеспечьте зазор не менее 5-10 см сверху для вентиляции.
- Обеспечьте достаточное пространство для подключения кабелей с задней стороны.
- 5-дюймовый цветной дисплей на передней панели отображает информацию о состоянии, обложки альбомов и настройки системы .

4.3. Акустические подключения

Усилитель оснащен одной парой акустических терминалов для подключения пассивных студийных мониторов. Используйте качественные акустические кабели соответствующего сечения.

4.4. Схема подключения для студийного мониторинга

text

```
Источник (DAW/компьютер) → Цифровой вход (Optical/Coaxial) → NAD C700 V2 → Пассивные студийные мониторы
                                  │
                                  ├→ Активный сабвуфер (SUBW OUT)
                                  ├→ Наушники (6,3 мм или Bluetooth)
                                  └→ Внешний усилитель (PRE OUT)

ТВ/Видеоисточник → HDMI eARC → NAD C700 V2 (для контроля видеоконтента)

Беспроводные тыловые каналы → BluOS совместимые динамики (для 4.1 surround) [citation:2][citation:8]

Системы управления (Crestron/Control4) → RS232/сеть → NAD C700 V2
```

5. ПРОГРАММИРОВАНИЕ НА PYTHON (BLUOS API)

5.1. Общие сведения

Усилитель NAD C 700 V2 имеет встроенный модуль BluOS и поддерживает **BluOS API** — RESTful веб-сервис, идентичный API устройств Bluesound и других NAD компонентов . API доступен через HTTP-запросы к встроенному веб-серверу.

Базовые параметры подключения:

- **Протокол:** HTTP
- **Порт:** 80
- **IP-адрес:** IP-адрес устройства в локальной сети
- **Формат ответа:** XML (key=value) или JSON

5.2. Определение IP-адреса устройства

Перед началом программирования необходимо определить IP-адрес NAD C700 V2 в сети:

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

def discover_bluos_devices() -> List[Dict]:
    """
    Обнаружение устройств BluOS (включая NAD C700 V2)
    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 C700 V2)
    """
    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
import requests
from typing import Dict, Any, Optional

class NADC700V2Controller:
    """
    Класс для управления NAD C700 V2 через Python API
    """

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

        Args:
            ip_address: IP-адрес устройства 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, optical, coaxial, line, phono, bluetooth)
    """
    input_map = {
        'hdmi': 'hdmi_arc',
        'optical': 'optical',
        'coaxial': 'coaxial',
        'line': 'analog',
        '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})

# === Dolby Digital / Surround настройки ===

def set_surround_mode(self, enabled: bool) -> Dict[str, Any]:
    """
    Включение/выключение surround режима 4.1

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

def add_surround_speakers(self, speaker_ips: List[str]) -> Dict[str, Any]:
    """
    Добавление беспроводных тыловых динамиков для 4.1 surround

    Args:
        speaker_ips: Список IP-адресов BluOS динамиков
    """
    results = {}
    for speaker_ip in speaker_ips:
        group_url = f"http://{speaker_ip}:80/Sync?cmd=group&master={self.ip}"
        try:
            response = requests.get(group_url, timeout=5)
            results[speaker_ip] = response.status_code == 200
        except Exception as e:
            results[speaker_ip] = False
            results[f"{speaker_ip}_error"] = str(e)
    return results

# === Тон-компенсация ===

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

    Args:
        value: Значение от -6 до +6 дБ [citation:5]
    """
    value = max(-6, min(6, 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 -6 до +6 дБ [citation:5]
    """
    value = max(-6, min(6, 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})

# === Dirac Live управление ===

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:8]

    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)}

# === Предустановки ===

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}")

# === Группировка устройств (multi-room) ===

def group_devices(self, slave_ips: List[str]) -> Dict[str, Any]:
    """
    Группировка нескольких устройств BluOS (до 63 зон) [citation:5]
```

```
        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 C700 V2
controller = NADC700V2Controller("192.168.1.100")

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

# Выбор линейного входа (подключение к DAW)
controller.select_input("line")

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

# Настройка тон-компенсации (нейтральное положение)
controller.set_bass(0)     # Нейтральные НЧ
controller.set_treble(0)   # Нейтральные ВЧ

# Включение сабвуфера при необходимости
controller.set_subwoofer(True)
```

```
# 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: Профессиональный мониторинг с логированием

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

class NADC700V2ProfessionalMonitor:
    """Профессиональный мониторинг NAD C700 V2"""

    def __init__(self, ip_address: str, log_file: str = "nad_c700v2_session.csv"):
        self.controller = NADC700V2Controller(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')
        }

        # 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', '')

        # Получение статуса Dirac Live (если активен)
        dirac_status = self.controller.get_dirac_status()
        if 'active_profile' in dirac_status:
            state['dirac_profile'] = dirac_status['active_profile']

        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 C700 V2',
        '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)

                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)

                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 = NADC700V2ProfessionalMonitor("192.168.1.100")
# monitor.monitor_session(duration_minutes=30, interval_seconds=5)
# report = monitor.generate_session_report()
# print(json.dumps(report, indent=2))
```

Пример 3: Интеграция с профессиональной системой управления

python

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

app = Flask(__name__)

# Конфигурация усилителей NAD C700 V2
NAD_DEVICES = {
    "edit_suite_a": "192.168.1.100",
    "edit_suite_b": "192.168.1.101",
    "qc_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] = NADC700V2Controller(NAD_DEVICES[device_name])
    return controllers[device_name]

@app.route('/api/nadc700/<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 C700 V2',
        'volume': controller.get_volume(),
        'state': controller.get_state(),
        'now_playing': controller.get_now_playing(),
        'dirac': controller.get_dirac_status(),
        'inputs': {
            'analog': ['line', 'phono'],
            'digital': ['optical', 'coaxial', 'hdmi'],
            'wireless': ['bluetooth']
        }
    }
    return jsonify(status)
```

```
@app.route('/api/nadc700/<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/nadc700/<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/nadc700/<device_name>/dirac/profile', methods=['POST'])
def handle_dirac_profile(device_name):
    """Выбор профиля Dirac Live"""
    controller = get_controller(device_name)
    if not controller:
        return jsonify({"error": "Device not found"}), 404

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

    result = controller.select_dirac_profile(data['profile'])
```

```
    return jsonify(result)

@app.route('/api/nadc700/<device_name>/surround', methods=['POST'])
def handle_surround(device_name):
    """Управление surround режимом 4.1"""
    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_surround_mode(data['enabled'])
    return jsonify(result)

@app.route('/api/nadc700/devices', methods=['GET'])
def list_devices():
    """Список всех устройств"""
    return jsonify(NAD_DEVICES)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5004)
```

Пример 4: Управление 4.1 surround системой для пост-продакшн

python

```
class SurroundPostProductionController:
    """Управление системой 4.1 Dolby Digital для пост-продакшн"""

    def __init__(self, nad_ip: str, rear_speaker_ips: List[str], subwoofer_enabled: bool = True):
        """
        Args:
            nad_ip: IP-адрес NAD C700 V2
            rear_speaker_ips: Список IP-адресов тыловых BluOS динамиков
            subwoofer_enabled: Включен ли сабвуфер
        """
        self.controller = NADC700V2Controller(nad_ip)
        self.rear_speakers = rear_speaker_ips
        self.subwoofer_enabled = subwoofer_enabled

    def enable_surround_mode(self):
        """Включение режима 4.1 surround для контроля многоканального контента"""
        print("=== ВКЛЮЧЕНИЕ 4.1 SURROUND РЕЖИМА ===")
```

```
        # Включение surround режима
        self.controller.set_surround_mode(True)

        # Добавление тыловых динамиков
        if self.rear_speakers:
            result = self.controller.add_surround_speakers(self.rear_speakers)
            print(f"Тыловые динамики подключены: {result}")

        # Включение сабвуфера при необходимости
        if self.subwoofer_enabled:
            self.controller.set_subwoofer(True)

        # Выбор HDMI входа для Dolby Digital контента
        self.controller.select_input("hdmi")

        # Установка Movie режима для киноконтента
        self.controller.set_listening_mode("movie")

        print("4.1 surround режим активирован для контроля многоканального микса")

    def disable_surround_mode(self):
        """Отключение surround режима"""
        self.controller.set_surround_mode(False)
        self.controller.ungroup_devices()
        print("Surround режим отключен, возврат к стерео")

    def check_surround_status(self) -> Dict:
        """Проверка статуса surround системы"""
        status = {
            'surround_enabled': self.controller.get_status().get('surround', '0'),
            'rear_speakers': self.rear_speakers,
            'subwoofer_enabled': self.subwoofer_enabled,
            'active_dirac_profile': self.controller.get_dirac_status().get('active_profile', 'none')
        }
        return status

# Пример использования для пост-продакшн студии
surround_system = SurroundPostProductionController(
    nad_ip="192.168.1.100",
    rear_speaker_ips=["192.168.1.101", "192.168.1.102"],
    subwoofer_enabled=True
)
```

```
# Включение surround режима для проверки многоканального микса
surround_system.enable_surround_mode()
```

Пример 5: Dirac Live калибровка для профессиональной студии

```
python
class DiracLiveCalibrationC700:
    """Управление Dirac Live калибровкой для NAD C700 V2"""

    def __init__(self, nad_ip: str):
        self.controller = NADC700V2Controller(nad_ip)

    def check_dirac_availability(self) -> Dict:
        """
        Проверка доступности Dirac Live
        Примечание: Функция может находиться в стадии разработки для C700 V2 [citation:4]
        """
        try:
            status = self.controller.get_dirac_status()
            return {
                'available': 'error' not in status,
                'status': status,
                'note': 'Для использования Dirac Live требуется приобретение лицензии [citation:2][citation:5]'
            }
        except Exception as e:
            return {
                'available': False,
                'error': str(e),
                'note': 'Dirac Live для C700 V2 может находиться в стадии разработки [citation:4]'
            }

    def run_calibration_guide(self):
        """Вывод руководства по калибровке Dirac Live"""
        print("=== РУКОВОДСТВО ПО КАЛИБРОВКЕ DIRAC LIVE ===")
        print("1. Приобретите лицензию Dirac Live для C700 V2 [citation:2][citation:5]")
        print("2. Подключите калибровочный микрофон к USB порту C700 V2 или компьютеру")
        print("3. Установите приложение Dirac Live на компьютер")
        print("4. Разместите микрофон в основной точке прослушивания")
        print("5. Выполните измерения (рекомендуется минимум 9 позиций)")
        print("6. Создайте фильтр коррекции")
        print("7. Загрузите профиль в один из 5 слотов C700 V2 [citation:8]")
        print("\nПРИМЕЧАНИЕ: Уточните статус поддержки Dirac Live для C700 V2")
        print("на официальном сайте NAD перед началом калибровки [citation:4]")
```

```
    def calibration_workflow(self) -> Dict:
        """
        Возвращает пошаговый план калибровки для профессиональной студии
        """
        workflow = {
            'device': 'NAD C700 V2',
            'dirac_version': 'Dirac Live Ready (лицензия требуется) [citation:5]',
            'license_required': True,
            'license_cost': '~$99 для Limited Bandwidth [citation:2]',
            'microphone_required': True,
            'max_profiles': 5,
            'step_1': 'Приобретение лицензии Dirac Live',
            'step_2': 'Подключение калибровочного микрофона',
            'step_3': 'Установка и запуск Dirac Live приложения',
            'step_4': 'Размещение микрофона в основной точке прослушивания',
            'step_5': 'Выполнение измерений в 9-17 позициях',
            'step_6': 'Создание и загрузка фильтра коррекции в слот 1-5',
            'step_7': 'Верификация коррекции с помощью измерительного ПО',
            'notes': {
                'full_bandwidth_upgrade': 'Доступна опционально для коррекции полной полосы [citation:2]',
                'development_status': 'Проверьте актуальный статус на сайте производителя [citation:4]'
            }
        }
        return workflow

# Пример использования
dirac = DiracLiveCalibrationC700("192.168.1.100")
availability = dirac.check_dirac_availability()
print(json.dumps(availability, indent=2, ensure_ascii=False))
dirac.run_calibration_guide()
```

5.5. Справочник команд BluOS API для NAD C700 V2

Команда	Описание	Параметры
volume	Управление громкостью	level (0-100)
mute / unmute	Отключение/включение звука	-

Команда	Описание	Параметры
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, coaxial, analog, phono, bluetooth)
listeningMode	Режим прослушивания	mode (stereo, direct, movie, music)
subwoofer	Сабвуфер	state (0/1)
surround	Режим 4.1 surround	state (0/1)
bass	Низкие частоты	value (-6 to +6 дБ)
treble	Высокие частоты	value (-6 to +6 дБ)
balance	Баланс	value (-50 to +50)
preset1 - preset6	Предустановки (1-6)	-
group	Группировка устройств	master (IP мастер-устройства)
ungroup	Отмена группировки	-
DiracStatus	Статус Dirac Live	-

Команда	Описание	Параметры
DiracSelect	Выбор профиля Dirac	profile (1-5)
DiracBypass	Обход Dirac	bypass (true/false)

---

6. ОРГАНЫ УПРАВЛЕНИЯ

6.1. Передняя панель

Элемент	Функция
5" HD Color Display	Высококачественный цветной дисплей для отображения обложек альбомов, прогресса трека и настроек системы
Volume Control	Регулятор громкости (интегрирован с навигацией)
Navigation Buttons	Навигация по меню дисплея
Play/Pause	Управление воспроизведением
Standby/Power	Включение/перевод в режим ожидания
Headphone Output	6,3 мм выход для наушников

Дисплей позволяет настраивать параметры усилителя через меню на передней панели, без необходимости использования мобильного приложения .

6.2. Пульт дистанционного управления BC1 IR

Кнопка	Функция
ON/OFF	Включение/режим ожидания
SOURCE	Переключение источников
VOL +/–	Регулировка громкости
MUTE	Отключение звука
PLAY/PAUSE	Управление воспроизведением
SKIP	Следующий/предыдущий трек
DIMMER	Регулировка яркости дисплея

---

7. ПОДКЛЮЧЕНИЯ И ИНТЕРФЕЙСЫ

7.1. Аналоговые входы

Вход	Тип	Чувствительность	Импеданс
Line IN	RCA стерео	550 мВ	56 кОм
Phono MM	RCA стерео	—	—

Примечание: Вход Phono MM оснащен точной RIAA эквалайзерной коррекцией для проигрывателей винила .

7.2. Цифровые входы

Вход	Тип	Макс. частота	Поддержка
Коаксиальный	S/PDIF RCA	192 кГц / 24 бит	—
Оптический	Toslink	192 кГц / 24 бит	—
HDMI eARC	HDMI	до 192 кГц	Dolby Digital, управление громкостью с пульта источника
Bluetooth	5.0	aptX HD	Двунаправленный (прием и передача)

Важное примечание по HDMI eARC: Порт HDMI eARC поддерживает Dolby Digital и позволяет создавать 4.1 surround систему с беспроводными тыловыми динамиками BluOS . Убедитесь, что аудио настройки источника установлены в соответствующий формат.

7.3. Выходы

Выход	Тип	Назначение
Акустические	1 пара клемм	Пассивные студийные мониторы
PRE OUT	RCA	Выход на внешний усилитель
SUBW OUT	RCA	Активный сабвуфер
Headphone	6,3 мм	Выход на наушники

7.4. Управляющие интерфейсы

Интерфейс	Назначение
RS232	Серийное управление (AMX, Crestron)

Интерфейс	Назначение
IR IN (3,5 мм)	Подключение внешнего ИК-приемника
IR OUT (3,5 мм)	Выход ИК-сигнала
TRIGGER OUT (12 В)	Выход триггера для управления питанием внешних устройств
USB Type-A	Порт для USB-накопителей (FAT32/NTFS) и подключения калибровочного микрофона
Ethernet (RJ45)	Gigabit проводное сетевое подключение
Wi-Fi	802.11ac (2,4/5 ГГц) беспроводное подключение

---

8. НАСТРОЙКИ ЗВУКА

8.1. Тон-компенсация

Параметр	Диапазон	Центральная частота
Bass	±6,0 дБ	60 Гц
Treble	±6,0 дБ	20 кГц

8.2. Режимы прослушивания

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

8.3. 4.1 Surround режим (Dolby Digital)

Усилитель C700 V2 поддерживает создание **4.1 Dolby Digital Surround** системы:

- **Фронтальные каналы:** Подключенные пассивные динамики
- **Тыловые каналы:** Беспроводные BluOS динамики (до 2 шт.)
- **Сабвуфер:** Проводной сабвуфер (SUBW OUT)

Применение в профессиональной среде:

- Контроль surround-миксов для телевидения и стриминговых платформ
- Оценка звуковых эффектов и диалогов в многоканальном формате
- Пост-продакшн Dolby Digital контента

8.4. Dirac Live Room Correction

Особенности:

- **Требуется отдельная лицензия**
- Сохранение до 5 профилей для разных позиций/условий
- Коррекция акустики помещения с использованием калибровочного микрофона (приобретается отдельно)
- Limited Bandwidth версия может быть включена в некоторые комплектации (требуется уточнение)

Важное примечание: По информации от производителя, функция Dirac Live для C700 V2 может находиться в стадии разработки. Проверьте актуальный статус на официальном сайте NAD перед приобретением лицензии .

Процедура калибровки:

1. Приобретите лицензию Dirac Live
2. Подключите калибровочный микрофон к USB порту C700 V2 или компьютеру
3. Установите приложение Dirac Live
4. Разместите микрофон в основной точке прослушивания
5. Выполните измерения (рекомендуется минимум 9 позиций)
6. Создайте фильтр коррекции
7. Загрузите профиль в один из 5 слотов C700 V2

---

9. BLUOS СТРИМИНГ И МУЛЬТИ-РУМ

9.1. BluOS возможности

- **Мульти-рум стриминг:** Поддержка до 63 дополнительных зон
- **Совместимые устройства:** Bluesound, NAD, Roksan, Cyrus
- **Hi-Res Audio:** до 24 бит/192 кГц PCM
- **MQA:** Полное декодирование и рендеринг
- **Беспроводной surround:** Создание 4.1 системы с BluOS динамиками

9.2. Поддерживаемые сервисы и форматы

Категория	Поддержка
Стриминговые сервисы	Amazon Music HD, Deezer, Qobuz, TIDAL, Spotify
Интернет-радио	TuneIn Radio, iHeartRadio, Calm Radio, Radio Paradise

Категория	Поддержка
Интеграции	AirPlay 2, Spotify Connect, TIDAL Connect, Roon Ready
Голосовое управление	Amazon Alexa, Apple Siri, Google Assistant
Форматы	FLAC, ALAC, WAV, AIFF, MP3, AAC, OGG (до 24/192)

9.3. BluOS App

- **Платформы:** iOS, Android, Windows, macOS
- **Функции:** Управление воспроизведением, выбор источников, настройка multi-room, просмотр библиотеки
- **Бесплатная загрузка:** Google Play и Apple App Store

---

10. ПРОФЕССИОНАЛЬНЫЕ ФУНКЦИИ

10.1. Профессиональные системные интеграции

Сертифицированная поддержка систем управления :

- **Crestron** (через RS232 или сеть)
- **Control4**
- **RTI**
- **URC**
- **Elan**
- **Lutron**
- **iPort**
- **KNX**
- **PUSH**

10.2. HDMI eARC с Dolby Digital

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

10.3. Двунаправленный Bluetooth aptX HD

- **Прием:** Стриминг с мобильных устройств на усилитель
- **Передача:** Отправка звука с усилителя на Bluetooth-наушники

10.4. 5-дюймовый HD дисплей

- Отображение обложек альбомов
- Прогресс воспроизведения
- Настройки системы
- Меню настройки параметров

---

11. BLUOS APP (УПРАВЛЕНИЕ)

- **Поддерживаемые платформы:** iOS, Android, Windows, macOS
- **Управление многокомнатной системой (до 63 зон)**
- **Доступ к музыкальным сервисам:** Amazon Music HD, Deezer, Qobuz, TIDAL, 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 громкость не на минимуме.
Искажения звука	Снизьте громкость. Проверьте настройки тон-компенсации. Убедитесь, что нагрузка соответствует спецификациям.

Проблема	Решение
Не подключается BluOS	Убедитесь, что устройство подключено к сети (Ethernet или Wi-Fi). Проверьте настройки сети. Перезапустите роутер.
Не работает Dirac Live	Приобретите лицензию Dirac Live . Подключите калибровочный микрофон. Проверьте статус поддержки на сайте производителя .
API не отвечает	Проверьте IP-адрес устройства. Убедитесь, что устройство находится в той же подсети. Порт 80 должен быть открыт.
Нет surround звука	Убедитесь, что surround режим включен. Проверьте подключение тыловых BluOS динамиков. Убедитесь, что контент имеет Dolby Digital аудиодорожку .
HDMI eARC не работает	Убедитесь, что аудио настройки источника установлены в Dolby Digital или PCM. Проверьте, что используется кабель HDMI с поддержкой eARC.

---

13. СБРОС К ЗАВОДСКИМ НАСТРОЙКАМ

13.1. Сброс через переднюю панель

1. Нажмите и удерживайте кнопку **SOURCE** на передней панели.
2. Дождитесь, пока дисплей покажет опции сброса.
3. Используйте кнопки навигации для выбора опции:
    - **Factory Reset MCU?** — восстанавливает только настройки управления
    - **Factory Reset BluOS?** — восстанавливает настройки BluOS
4. Нажмите **ENTER** для подтверждения.

13.2. Принудительный сброс

1. Выключите питание усилителя.
2. Нажмите и удерживайте кнопку **Standby** на передней панели.
3. Включите питание, продолжая удерживать кнопку.
4. Дождитесь появления индикации сброса на дисплее.
5. Отпустите кнопку.

Внимание: Сброс удаляет все пользовательские настройки (сеть, предустановки, настройки звука).

---

14. ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ

- **Уход за корпусом:** Протирайте мягкой сухой тканью. Не используйте абразивные средства.
- **Уход за дисплеем:** Используйте микрофибру для очистки стеклянной панели.
- **Вентиляция:** Регулярно проверяйте, чтобы вентиляционные отверстия не были заблокированы.
- **Обновление ПО:** Выполняется автоматически через BluOS App при подключении к Интернету.
- **Транспортировка:** Используйте оригинальную упаковку.
- **Сервис:** При повреждениях обращайтесь к авторизованному дилеру NAD.

Гарантийный срок: 12 месяцев.

ПРИЛОЖЕНИЕ: ПОЛНЫЕ ТЕХНИЧЕСКИЕ СПЕЦИФИКАЦИИ

Параметр	Значение
Модель	NAD C 700 V2 BluOS Streaming Amplifier
Серия	Classic Series

Параметр	Значение
Цвет	Черный (Black)
Габариты (Ш × В × Г)	218 × 96 × 266 мм
Вес	4,8 кг
Потребляемая мощность (рабочая)	—
Потребляемая мощность (ожидание)	<0,5 Вт
Выходная мощность (непрерывная)	2 x 80 Вт (8/4 Ом) / 100 Вт (4 Ом)
Выходная мощность (пиковая)	2 x 120 Вт
IHF динамическая мощность	100 Вт (8 Ом), 125 Вт (4 Ом)
THD+N	<0,04% (20 Гц - 20 кГц)
Соотношение сигнал/шум	>84 дБ
Коэффициент демпфирования	>90
Разделение каналов	>93 дБ (1 кГц), >72 дБ (10 кГц)
Частотный диапазон	20 Гц - 20 кГц (±0.18 дБ)
ЦАП	ESS Sabre ES9028 (32 бит/192 кГц)
Тон-компенсация (Bass/Treble)	±6,0 дБ (60 Гц / 20 кГц)
Артикул	C700 (также V2/V3)