```
main.py
import os
import json
from typing import Dict, List
import pandas as pd
from fastapi import FastAPI, HTTPException, Request, Response
from fastapi.responses import JSONResponse
from fastapi.staticfiles import StaticFiles

import tempfile
from io import BytesIO




BASE_DIR = os.path.dirname(os.path.abspath(__file__))
SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
TEMPLATES_PATH = os.path.join(BASE_DIR, "formula_templates.json")
SIGNAL_INDEX_PATH = os.path.join(BASE_DIR, ".signal_index.pkl")

def load_templates() -> Dict:
    if not os.path.exists(TEMPLATES_PATH):
        return {"templates": []}
    with open(TEMPLATES_PATH, "r", encoding="utf-8") as f:
        return json.load(f)

def load_settings() -> Dict:
    with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
        return json.load(f)

def load_signals_from_folder(folder: str) -> List[Dict]:
    # folder может быть относительным
    folder_abs = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))
    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")

    signals_map = {}  # Tagname -> Description (последний wins)
    for name in os.listdir(folder_abs):
        if not name.lower().endswith(".csv"):
            continue
        path = os.path.join(folder_abs, name)
        try:
            df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
            df = df.dropna(subset=['Tagname'])
            for _, row in df.iterrows():
                tag = str(row['Tagname']).strip()
                desc = "" if pd.isna(row['Description']) else
str(row['Description']).strip()
                if tag:
                    signals_map[tag] = desc
        except Exception as e:
            # пропускаем "плохие" csv, но можно логировать
            print(f"[WARN] failed to read {path}: {e}")

    # в список
    out = [{'Tagname': k, 'Description': v} for k, v in signals_map.items()]
    out.sort(key=lambda x: x['Tagname'])
    return out




app = FastAPI()

```

```
# Кэш сигналов в памяти
STATE = {
    "settings": None,
    "signals": None,
    "signal_index": None
}


def build_signal_index(folder: str) -> Dict[str, List[str]]:
    """
    При запуске: проходим по всем CSV файлам и создаем индекс
    signal_name -> list of files where it's present
    """
    folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
        os.path.join(BASE_DIR, folder)
    )

    if not os.path.isdir(folder_abs):
        raise FileNotFoundError(f"Signal data folder not found: {folder_abs}")

    signal_index = {}

    print(f"[INFO] Building signal index from {folder_abs}...")

    for filename in os.listdir(folder_abs):
        if not filename.lower().endswith(".csv"):
            continue

        filepath = os.path.join(folder_abs, filename)

        try:
            # Читаем только первую строку (заголовок)
            df_header = pd.read_csv(
                filepath,
                nrows=0,  # Читаем только заголовок
                encoding="ISO-8859-2",
                sep=";"
            )

            columns = df_header.columns.tolist()

            # Удаляем служебные столбцы из индекса
            signal_columns = [c for c in columns if c not in ["DATE", "TIME",
"datetime"]]

            for signal_name in signal_columns:
                if signal_name not in signal_index:
                    signal_index[signal_name] = []
                signal_index[signal_name].append(filepath)

            print(f"  ✓ {filename}: {len(signal_columns)} signals")

        except Exception as e:
            print(f"  ✗ Failed to index {filename}: {e}")
            continue

    print(f"[OK] Total unique signals indexed: {len(signal_index)}")

    # Сохраняем индекс в pickle для быстрого восстановления
    try:
        with open(SIGNAL_INDEX_PATH, "wb") as f:
            pickle.dump(signal_index, f)
        print(f"[OK] Signal index cached to {SIGNAL_INDEX_PATH}")
    except Exception as e:
        print(f"[WARN] Failed to cache signal index: {e}")
```

```
128
129     return signal_index
130
131
132 def load_signal_index(folder: str) -> Dict[str, List[str]]:
133     """
134     Загружает индекс либо из кэша, либо перестраивает его
135     """
136     if os.path.exists(SIGNAL_INDEX_PATH):
137         try:
138             with open(SIGNAL_INDEX_PATH, "rb") as f:
139                 index = pickle.load(f)
140             print(f"[OK] Signal index loaded from cache")
141             return index
142         except Exception as e:
143             print(f"[WARN] Failed to load cached index: {e}")
144
145     # Если кэша нет, перестраиваем
146     return build_signal_index(folder)
147
148 def load_signal_data_optimized(signal_names: List[str], folder: str) -> Dict[str,
    pd.DataFrame]:
149     """
150     Загружает только нужные сигналы из только нужных файлов
151     Returns: {signal_name -> DataFrame}
152     """
153     folder_abs = folder if os.path.isabs(folder) else os.path.normpath(
154         os.path.join(BASE_DIR, folder)
155     )
156
157     signal_index = STATE.get("signal_index", {})
158     if not signal_index:
159         raise RuntimeError("Signal index not initialized")
160
161     signal_names_set = set(signal_names)
162     found_signals = {}
163     files_to_load = set()
164
165     # Определяем, какие файлы нужно загружать
166     for signal_name in signal_names_set:
167         if signal_name in signal_index:
168             files_to_load.update(signal_index[signal_name])
169
170     print(f"[INFO] Loading {len(signal_names_set)} signals from {len(files_to_load)}
    files")
171
172     # Загружаем данные из файлов
173     for filepath in files_to_load:
174         try:
175             df = pd.read_csv(
176                 filepath,
177                 encoding="ISO-8859-2",
178                 sep=";"
179             )
180
181             # Обработка даты/времени
182             df["TIME"] = df["TIME"].str.replace(",", ".", regex=False)
183             df["TIME"] = df["TIME"].str.split(".").str[0]
184             combined = df["DATE"] + " " + df["TIME"]
185             df["datetime"] = pd.to_datetime(
186                 combined, format="%d.%m.%Y %H:%M:%S", errors="coerce"
187             )
188             df = df.dropna(subset=["datetime"])
189             df = df.drop(['DATE', 'TIME'], axis=1)
190
```

```
            # Сортируем по datetime
            df = df.sort_values("datetime")

            # Извлекаем только нужные сигналы
            available_columns = set(df.columns) & signal_names_set
            for signal_name in available_columns:
                if signal_name not in found_signals:
                    # Сохраняем datetime и значение сигнала
                    found_signals[signal_name] = df[["datetime", signal_name]].copy()
                    found_signals[signal_name].columns = ["datetime", "value"]

        except Exception as e:
            print(f"[WARN] Failed to read {filepath}: {e}")
            continue

    return found_signals




@app.on_event("startup")
def startup():
    settings = load_settings()
    STATE["settings"] = settings
    folder = settings.get("signalDataFolder")
    if not folder:
        raise RuntimeError("settings.json: signalDataFolder is required")
    STATE["signals"] = load_signals_from_folder(folder)
    STATE["templates"] = load_templates()
    STATE["signal_index"] = load_signal_index(settings.get("signalArchiveFolder"))

    print(f"[OK] loaded signals: {len(STATE['signals'])}")
    print(f"[OK] signal index has {len(STATE['signal_index'])} unique signals")
    print(f"[OK] loaded templates: {len(STATE['templates'].get('templates', []))}")

@app.get("/api/settings")
def api_settings():
    return STATE["settings"]

@app.get("/api/signals")
def api_signals(q: str = "", limit: int = 50):
    """
    q — маска со * (например *МАА*СР*)
    """
    signals = STATE["signals"] or []
    if not q:
        return {"items": signals[:limit], "total": len(signals)}

    # маска * -> regex
    import re
    escaped = re.escape(q).replace(r"\*", ".*")
    rx = re.compile("^" + escaped + "$", re.IGNORECASE)

    items = [s for s in signals if rx.match(s["Tagname"])]
    return {"items": items[:max(1, min(limit, 500))], "total": len(items)}

# Helper: возвращает абсолютный путь к файлу проекта
def get_project_path(filename: str):
    folder = STATE["settings"].get("projectDataFolder")
    if not folder:
        raise RuntimeError("projectDataFolder not configured")

    # Нормализуем путь к папке проектов
```

```
256      project_dir = folder if os.path.isabs(folder) else
     os.path.normpath(os.path.join(BASE_DIR, folder))
257
258      # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
259      if '..' in filename or '/' in filename or '\\' in filename:
260          raise HTTPException(status_code=400, detail="Invalid filename")
261
262      path = os.path.join(project_dir, filename)
263      # Проверяем, что итоговый путь лежит внутри разрешенной директории
264      if not path.startswith(project_dir):
265           raise HTTPException(status_code=400, detail="Path traversal attempt")
266
267      return path
268
269  @app.post("/api/project/save")
270  async def save_project(request: Request):
271      try:
272          data = await request.json()
273          filename = data.get("filename")
274          content = data.get("content")
275
276          if not filename or not content:
277              raise HTTPException(status_code=400, detail="Filename and content are
     required")
278
279          path = get_project_path(filename)
280
281          # Сохраняем как JSON
282          with open(path, "w", encoding="utf-8") as f:
283              json.dump(content, f, indent=2)
284
285          return {"status": "ok", "message": f"Project saved to {filename}"}
286
287      except HTTPException as e:
288          raise e
289      except Exception as e:
290          print(f"Error saving project: {e}")
291          raise HTTPException(status_code=500, detail="Internal server error during
     save")
292
293  @app.get("/api/project/load/{filename}")
294  def load_project(filename: str):
295      try:
296          path = get_project_path(filename)
297
298          if not os.path.exists(path):
299              raise HTTPException(status_code=404, detail="Project not found")
300
301          with open(path, "r", encoding="utf-8") as f:
302              content = json.load(f)
303
304          return content
305
306      except HTTPException as e:
307          raise e
308      except Exception as e:
309          print(f"Error loading project: {e}")
310          raise HTTPException(status_code=500, detail="Internal server error during
     load")
311
312  @app.get("/api/formula-templates")
313  def api_formula_templates():
314      return STATE.get("templates") or {"templates": []}
315
316  @app.get("/api/project/list")
```

```
317 def list_projects():
318     folder = STATE["settings"].get("projectDataFolder")
319     if not folder:
320         raise HTTPException(status_code=500, detail="Project folder not configured")
321 
322     project_dir = folder if os.path.isabs(folder) else
    os.path.normpath(os.path.join(BASE_DIR, folder))
323     os.makedirs(project_dir, exist_ok=True)
324 
325     projects = []
326     for fname in sorted(os.listdir(project_dir)):
327         if not fname.endswith(".json"):
328             continue
329         path = os.path.join(project_dir, fname)
330         try:
331             with open(path, "r", encoding="utf-8") as f:
332                 payload = json.load(f)
333         except Exception:
334             continue
335         project_meta = payload.get("project", {})
336         projects.append({
337             "filename": fname,
338             "code": project_meta.get("code") or project_meta.get("tagname") or "",
339             "description": project_meta.get("description") or "",
340             "type": project_meta.get("type") or ""
341         })
342     return {"projects": projects}
343 
344 @app.post("/api/signal-data")
345 async def api_signal_data(request: Request):
346     """
347     POST с JSON телом:
348     {
349         "signal_names": ["SIGNAL1", "SIGNAL2", ...],
350         "format": "parquet"  # или "json"
351     }
352 
353     Returns: Parquet файл с данными или JSON
354     """
355     try:
356         data = await request.json()
357         signal_names = data.get("signal_names", [])
358         output_format = data.get("format", "parquet")  # По умолчанию Parquet
359 
360         if not signal_names:
361             raise HTTPException(status_code=400, detail="signal_names is required")
362 
363         folder = STATE["settings"].get("signalDataFolder")
364         if not folder:
365             raise HTTPException(status_code=500, detail="signalDataFolder not
    configured")
366 
367         # Загружаем данные сигналов
368         signals_data = load_signal_data_optimized(signal_names, folder)
369 
370         # Подготавливаем ответ
371         response = {
372             "found": list(signals_data.keys()),
373             "not_found": [s for s in signal_names if s not in signals_data],
374             "format": output_format
375         }
376 
377         if not signals_data:
378             raise HTTPException(status_code=404, detail="No signals found")
379 
```

```
380         # Экспортируем данные в зависимости от формата
381         if output_format == "parquet":
382             return await _export_parquet(signals_data, response)
383         else:
384             return await _export_json(signals_data, response)
385 
386     except HTTPException as e:
387         raise e
388     except Exception as e:
389         print(f"Error in api_signal_data: {e}")
390         raise HTTPException(status_code=500, detail=str(e))
391 
392 async def _export_parquet(signals_data: Dict[str, pd.DataFrame], meta: Dict):
393     """
394     Экспортирует данные в Parquet (НАМНОГО меньше чем JSON!)
395     """
396     from fastapi.responses import FileResponse
397 
398     try:
399         # Создаем временный файл
400         with tempfile.NamedTemporaryFile(suffix=".parquet", delete=False) as tmp:
401             tmp_path = tmp.name
402 
403         # Каждый сигнал сохраняем как отдельную таблицу в Parquet
404         # Используем структуру: datetime, signal_name, value
405         rows = []
406         for signal_name, df in signals_data.items():
407             df_copy = df.copy()
408             df_copy["signal_name"] = signal_name
409             rows.append(df_copy)
410 
411         combined = pd.concat(rows, ignore_index=True)
412         combined.to_parquet(tmp_path, compression='snappy', index=False)
413 
414         file_size = os.path.getsize(tmp_path)
415         print(f"[OK] Exported {len(signals_data)} signals to Parquet: {file_size /
    1024 / 1024:.2f} MB")
416 
417         return FileResponse(
418             tmp_path,
419             media_type="application/octet-stream",
420             filename="signal_data.parquet",
421             headers={"X-Signal-Meta": json.dumps(meta)}
422         )
423 
424     except Exception as e:
425         print(f"[ERROR] Parquet export failed: {e}")
426         raise
427 
428 
429 async def _export_json(signals_data: Dict[str, pd.DataFrame], meta: Dict):
430     """
431     Экспортирует данные в JSON (медленнее и больше, но совместимее)
432     """
433     from fastapi.responses import JSONResponse
434 
435     try:
436         # Формируем JSON с каждым сигналом отдельно
437         data_dict = {}
438         for signal_name, df in signals_data.items():
439             df_copy = df.copy()
440             df_copy["datetime"] = df_copy["datetime"].astype(str)
441             data_dict[signal_name] = df_copy.to_dict(orient="records")
442 
443         response_data = {
```

```
            **meta,
            "data": data_dict
        }

        return JSONResponse(response_data)

    except Exception as e:
        print(f"[ERROR] JSON export failed: {e}")
        raise


# Раздаём фронтенд
WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")

visualizer_app.py
import streamlit as st
import pandas as pd
import numpy as np
import holoviews as hv
from holoviews import opts
import datashader as ds
from bokeh.models import HoverTool
from datetime import datetime
import requests
import json

# Включаем расширения
hv.extension('bokeh')

st.set_page_config(page_title="Signal Visualizer", layout="wide")

st.title("📊 Signal Data Visualizer")

# Получаем параметры из query string
query_params = st.query_params
signal_codes = query_params.get("signals", [])
if isinstance(signal_codes, str):
    signal_codes = [signal_codes]

api_url = query_params.get("api_url", "http://localhost:8000")
additional_data = query_params.get("data", "")

# Инициализируем состояние приложения
if "signals_data" not in st.session_state:
    st.session_state.signals_data = None
if "selected_signals" not in st.session_state:
    st.session_state.selected_signals = set()
if "plot_areas" not in st.session_state:
    st.session_state.plot_areas = []

# Загружаем данные сигналов
if signal_codes and st.session_state.signals_data is None:
    with st.spinner("Loading signal data..."):
        try:
            response = requests.post(
                f"{api_url}/api/signal-data",
                json={"signal_names": signal_codes}
            )
            response.raise_for_status()
            result = response.json()

            # Парсим данные
            if result.get("data"):
                df_data = pd.read_json(result["data"], orient="split")
```

```
509                 df_data["datetime"] = pd.to_datetime(df_data["datetime"])
510                 df_data = df_data.set_index("datetime")
511                 st.session_state.signals_data = df_data
512 
513                 st.success(f"✅ Loaded {len(result['found'].get('signals', []))}
    signals")
514                 if result.get("not_found"):
515                     st.warning(f"⚠️ Not found: {', '.join(result['not_found'])}")
516 
517         except Exception as e:
518             st.error(f"❌ Error loading data: {e}")
519 
520 # Боковая панель для выбора сигналов
521 with st.sidebar:
522     st.header("Signals Selection")
523 
524     if st.session_state.signals_data is not None:
525         available_signals = st.session_state.signals_data.columns.tolist()
526 
527         # Чекбоксы для выбора сигналов
528         for signal in available_signals:
529             is_selected = st.checkbox(
530                 signal,
531                 value=signal in st.session_state.selected_signals,
532                 key=f"signal_{signal}"
533             )
534             if is_selected:
535                 st.session_state.selected_signals.add(signal)
536             else:
537                 st.session_state.selected_signals.discard(signal)
538 
539     st.divider()
540 
541     # Управление областями визуализации
542     st.subheader("Plot Areas")
543     col1, col2 = st.columns(2)
544     with col1:
545         if st.button("➕ Add Plot Area"):
546             new_area = {
547                 "id": max([a.get("id", 0) for a in st.session_state.plot_areas] + [0])
    + 1,
548                 "signals": []
549             }
550             st.session_state.plot_areas.append(new_area)
551             st.rerun()
552 
553     with col2:
554         if st.button("❌ Clear All"):
555             st.session_state.plot_areas = []
556             st.session_state.selected_signals = set()
557             st.rerun()
558 
559 # Основная область
560 if st.session_state.signals_data is not None and
    len(st.session_state.selected_signals) > 0:
561 
562     # Если нет областей визуализации, создаем одну по умолчанию
563     if not st.session_state.plot_areas:
564         st.session_state.plot_areas.append({
565             "id": 1,
566             "signals": list(st.session_state.selected_signals)
567         })
568 
569     # Рисуем области визуализации
570     for i, plot_area in enumerate(st.session_state.plot_areas):
```

```
        with st.container():
            col1, col2 = st.columns([3, 1])

            with col1:
                st.subheader(f"Plot Area #{plot_area['id']}")

            with col2:
                if st.button("🗑️ Remove", key=f"remove_{i}"):
                    st.session_state.plot_areas.pop(i)
                    st.rerun()

            # Выбор сигналов для этой области
            area_signals = st.multiselect(
                "Select signals for this area:",
                options=st.session_state.selected_signals,
                default=plot_area.get("signals", []),
                key=f"area_signals_{i}"
            )
            st.session_state.plot_areas[i]["signals"] = area_signals

            # Построение графика
            if area_signals:
                try:
                    df_plot = st.session_state.signals_data[area_signals].copy()

                    # Создаем HoloViews кривые
                    curves = []
                    for signal in area_signals:
                        curve = hv.Curve(
                            (df_plot.index, df_plot[signal]),
                            label=signal
                        )
                        curves.append(curve)

                    # Объединяем кривые
                    plot = hv.Overlay(curves)

                    # Применяем опции визуализации
                    plot = plot.opts(
                        opts.Curve(
                            width=1000,
                            height=300,
                            tools=['pan', 'wheel_zoom', 'box_zoom', 'reset', 'save'],
                            xaxis='bottom',
                            shared_axes=False
                        ),
                        opts.Overlay(
                            legend_position='top_right'
                        )
                    )

                    # Отображаем с помощью st_bokeh_chart или hv.save
                    st.bokeh_chart(hv.render(plot), use_container_width=True)

                except Exception as e:
                    st.error(f"Error plotting: {e}")
            else:
                st.info("Select signals to display in this area")

            st.divider()

elif st.session_state.signals_data is None:
    st.info("📥 Awaiting signal data...")
else:
    st.info("👈 Select signals from the sidebar to visualize")
```

```
636
637 # Информационная панель внизу
638 if st.session_state.signals_data is not None:
639     with st.expander("ℹ️ Data Info"):
640         col1, col2, col3 = st.columns(3)
641         with col1:
642             st.metric("Total Signals", len(st.session_state.signals_data.columns))
643         with col2:
644             st.metric("Data Points", len(st.session_state.signals_data))
645         with col3:
646             time_range = st.session_state.signals_data.index.max() -
    st.session_state.signals_data.index.min()
647             st.metric("Time Range", str(time_range).split('.')[0])
648
649
650 index.html
651
652 <!DOCTYPE html>
653 <html lang="ru">
654 <head>
655     <meta charset="UTF-8">
656     <meta name="viewport" content="width=device-width, initial-scale=1.0">
657     <title>Редактор логических схем</title>
658     <link rel="stylesheet" href="css/styles.css">
659 </head>
660 <body>
661     <div id="app">
662         <div id="menu">
663             <button class="menu-btn" id="btn-new">📄 Новый</button>
664             <button class="menu-btn" id="btn-save">💾 Сохранить</button>
665             <button class="menu-btn" id="btn-load">📂 Загрузить</button>
666             <button class="menu-btn" id="btn-generate-code">🧩 Код</button>
667             <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</
    button>
668             <div class="menu-separator"></div>
669             <div class="zoom-controls">
670                 <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
671                 <span id="zoom-level">100%</span>
672                 <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
673                 <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
674                 <button class="menu-btn" id="btn-zoom-reset">1:1</button>
675             </div>
676             <input type="file" id="file-input" accept=".json">
677         </div>
678
679         <div id="main">
680             <div id="palette">
681                 <h3>📦 Элементы</h3>
682                 <div class="palette-section">
683                     <div class="palette-section-title">ВИЗУАЛЬНОЕ</div>
684
685                     <div class="palette-item" data-type="group">
686                         <svg viewBox="0 0 60 40">
687                         <rect x="6" y="8" width="48" height="24" rx="4"
688                                 fill="none" stroke="#6b7280" stroke-width="2" stroke-
    dasharray="4,2"/>
689                         <text x="14" y="25" fill="#6b7280" font-size="10" font-
    weight="bold">GROUP</text>
690                         </svg>
691                         <div class="palette-item-name">Группа</div>
692                     </div>
693                     </div>
694
695                 <div class="palette-section">
696                     <div class="palette-section-title">ВХОДЫ</div>
```

```
697
698                     <div class="palette-item" data-type="input-signal">
699                         <svg viewBox="0 0 60 40">
700                             <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460"
    stroke="#4a90d9" stroke-width="2"/>
701                             <text x="12" y="24" fill="#eee" font-size="10">IN</text>
702                         </svg>
703                         <div class="palette-item-name">Входной сигнал</div>
704                     </div>
705                 </div>
706                 <div class="palette-section">
707                     <div class="palette-section-title">ВЫХОДЫ</div>
708
709                     <div class="palette-item" data-type="output">
710                         <svg viewBox="0 0 60 40">
711                             <rect x="5" y="5" width="50" height="30" rx="6"
    fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
712                             <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</
    text>
713                         </svg>
714                         <div class="palette-item-name">Выход</div>
715                     </div>
716                 </div>
717
718                 <div class="palette-section">
719                     <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>
720
721                     <div class="palette-item" data-type="and">
722                         <svg viewBox="0 0 60 40">
723                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
724                             <text x="22" y="25" fill="#eee" font-size="12" font-
    weight="bold">И</text>
725                         </svg>
726                         <div class="palette-item-name">И (AND)</div>
727                     </div>
728
729                     <div class="palette-item" data-type="or">
730                         <svg viewBox="0 0 60 40">
731                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
732                             <text x="12" y="25" fill="#eee" font-size="11" font-
    weight="bold">ИЛИ</text>
733                         </svg>
734                         <div class="palette-item-name">ИЛИ (OR)</div>
735                     </div>
736
737                     <div class="palette-item" data-type="not">
738                         <svg viewBox="0 0 60 40">
739                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
740                             <text x="12" y="25" fill="#eee" font-size="11" font-
    weight="bold">НЕТ</text>
741                         </svg>
742                         <div class="palette-item-name">НЕТ (NOT)</div>
743                     </div>
744                 </div>
745
746                 <div class="palette-section">
747                     <div class="palette-section-title">СРАВНЕНИЕ</div>
748
749                     <div class="palette-item" data-type="if">
750                         <svg viewBox="0 0 60 40">
751                             <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460"
    stroke="#e94560" stroke-width="2"/>
```

```
752                             <text x="14" y="24" fill="#eee" font-size="9" font-
    weight="bold">ЕСЛИ</text>
753                         </svg>
754                         <div class="palette-item-name">ЕСЛИ (IF)</div>
755                     </div>
756                 </div>
757 
758                 <div class="palette-section">
759                     <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>
760 
761                     <div class="palette-item" data-type="separator">
762                         <svg viewBox="0 0 60 40">
763                             <rect x="5" y="8" width="50" height="24" rx="3"
    fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
764                             <text x="8" y="25" fill="#f59e0b" font-size="10" font-
    weight="bold">✓/✗</text>
765                         </svg>
766                         <div class="palette-item-name">Сепаратор</div>
767                     </div>
768                 </div>
769 
770                 <div class="palette-section">
771                     <div class="palette-section-title">ЗНАЧЕНИЯ</div>
772 
773                     <div class="palette-item" data-type="const">
774                         <svg viewBox="0 0 60 40">
775                             <rect x="10" y="8" width="40" height="24" rx="3"
    fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
776                             <text x="24" y="25" fill="#3b82f6" font-size="14" font-
    weight="bold">C</text>
777                         </svg>
778                         <div class="palette-item-name">Константа</div>
779                     </div>
780 
781                     <div class="palette-item" data-type="formula">
782                         <svg viewBox="0 0 60 40">
783                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
784                             <text x="12" y="25" fill="#f59e0b" font-size="11" font-
    weight="bold">f(x)</text>
785                         </svg>
786                         <div class="palette-item-name">Формула</div>
787                     </div>
788                 </div>
789 
790                 <div class="type-legend">
791                     <div class="type-legend-item">
792                         <div class="type-legend-dot logic"></div>
793                         <span>Логический</span>
794                     </div>
795                     <div class="type-legend-item">
796                         <div class="type-legend-dot number"></div>
797                         <span>Числовой</span>
798                     </div>
799                 </div>
800             </div>
801 
802             <div id="workspace-container">
803                 <svg id="connections-svg"></svg>
804                 <div id="workspace"></div>
805                   <!-- Прямоугольник для выделения элементов -->
806                 <div id="selection-rect"></div>
807 
808                 <!-- Мини-карта -->
809                 <div id="minimap">
```

```
                    <div id="minimap-viewport"></div>
                    <canvas id="minimap-canvas"></canvas>
                </div>

                <!-- Координаты и информация -->
                <div id="viewport-info">
                    <span id="cursor-pos">X: 0, Y: 0</span>
                    <span id="selection-info"></span>
                </div>
            </div>
        </div>
    </div>

    <!-- Модальные окна -->
    <div id="modal-overlay">
        <div id="modal">
            <h3 id="modal-title">Свойства элемента</h3>
            <div id="modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="modal-cancel">Отмена</button>
                <button class="modal-btn save" id="modal-save">Сохранить</button>
            </div>
        </div>
    </div>

    <!-- Модальное окно свойств проекта -->
    <div id="project-modal-overlay" class="modal-overlay-class">
        <div id="project-modal" class="modal-class">
            <h3>Свойства проекта</h3>
            <div id="project-modal-content"></div>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="project-modal-cancel">Отмена</
button>
                <button class="modal-btn save" id="project-modal-save">Сохранить</
button>
            </div>
        </div>
    </div>

    <div id="code-modal-overlay" class="modal-overlay-class">
        <div id="code-modal" class="modal-class">
            <h3>Сгенерированный код</h3>
            <textarea id="code-output" style="width:100%; height:300px;"></textarea>
            <div class="modal-buttons">
                <button class="modal-btn cancel" id="code-modal-close">Закрыть</
button>
            </div>
        </div>
    </div>

    <div id="context-menu">
        <div class="context-item" id="ctx-properties">⚙️ Свойства</div>
        <div class="context-item" id="ctx-delete">🗑️ Удалить</div>
    </div>

   <!-- Модули JavaScript -->
    <!-- Модули JavaScript -->
    <script src="js/config.js"></script>
    <script src="js/state.js"></script>
    <script src="js/utils.js"></script>
    <script src="js/viewport.js"></script>
    <script src="js/elements.js"></script>
    <script src="js/connections.js"></script>
    <script src="js/outputs.js"></script>  <!-- ← Этот файл опционален теперь -->
    <script src="js/modal.js"></script>
```

```
872      <script src="js/project.js"></script>
873      <script src="js/codegen_graph.js"></script>
874      <script src="js/codegen_optimizer.js"></script>
875      <script src="js/codegen.js"></script>
876      <script src="js/settings.js"></script>
877
878      <script src="js/app.js"></script>
879
880      <div id="modal-project-list" class="modal hidden">
881          <div class="modal__content modal__content--wide">
882              <h2 class="modal__title">Выбор проекта</h2>
883
884              <div class="project-list__toolbar">
885              <input id="project-search" type="text" placeholder="Фильтр по имени или
     описанию…" />
886              <button id="project-refresh" class="btn btn-secondary">Обновить</button>
887              </div>
888
889              <div class="project-list__table-container">
890              <table class="project-list__table">
891                  <thead>
892                  <tr>
893                      <th>Файл</th>
894                      <th>Tagname</th>
895                      <th>Description</th>
896                      <th>Тип</th>
897                  </tr>
898                  </thead>
899                  <tbody id="project-list-body">
900                  <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
901                  </tbody>
902              </table>
903              </div>
904
905              <div class="modal__actions">
906                  <button id="project-cancel" class="btn btn-secondary">Отмена</button>
907                  <button id="project-load" class="btn btn-primary" disabled>Загрузить</
     button>
908              </div>
909          </div>
910      </div>
911
912  </body>
913  </html>
914
915  app.js
916
917  /**
918   * Главный модуль приложения
919   */
920
921  const App = {
922      /**
923       * Инициализация приложения
924       */
925      init() {
926          Settings.init().catch(console.error);
927          //Settings.init().then(() => {
928          //    // если хочешь — можно обновить UI (например, статус “Сигналы
     загружены”)
929          //    console.log('Settings loaded, signals:', Settings.signals.length);
930          //    }).catch(err => console.error(err));
931          //console.log('signals loaded:', Settings.signals.slice(0, 5));
932          this.setupPaletteDragDrop();
933          this.setupGlobalMouseHandlers();
```

```
        this.setupContextMenu();
        this.setupWorkspaceClick();
        this.setupOutputCounter();
        this.setupMultiSelection();

        // Инициализация модулей
        Viewport.init();
        Modal.init();
        Project.init();

        // Первоначальное определение выходов (только если модуль загружен)
        if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
            Outputs.updateOutputStatus();
        }

        console.log('Logic Scheme Editor initialized');
        document.getElementById('btn-generate-code').addEventListener('click', () => {
            const code = CodeGen.generate();
            document.getElementById('code-output').value = code;
            document.getElementById('code-modal-overlay').style.display = 'flex';
        });

        document.getElementById('code-modal-close').addEventListener('click', () => {
            document.getElementById('code-modal-overlay').style.display = 'none';
        });
    },

    /**
     * Отмена состояния drag из палитры (helper)
     */
    cancelPaletteDrag() {
        if (AppState.dragPreview) {
            try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
            AppState.dragPreview = null;
        }
        AppState.isDraggingFromPalette = false;
        AppState.dragType = null;
    },

    /**
     * Настройка счётчика выходов в меню
     */
    setupOutputCounter() {
        // Не создавать повторно, если уже есть
        if (document.getElementById('btn-outputs')) return;

        const menu = document.getElementById('menu');

        // Создаём кнопку с счётчиком выходов
        const outputBtn = document.createElement('button');
        outputBtn.className = 'menu-btn output-btn';
        outputBtn.id = 'btn-outputs';
        outputBtn.innerHTML = `
            📤 Выходы
            <span id="output-counter" class="output-counter">0</span>
        `;

        // Вставляем после кнопки свойств проекта
        const projectBtn = document.getElementById('btn-project-settings');
        if (projectBtn) {
            projectBtn.after(outputBtn);
        } else {
            menu.appendChild(outputBtn);
        }

```

```
 999          outputBtn.addEventListener('click', () => {
1000              Modal.showProjectPropertiesModal();
1001          });
1002      },
1003
1004      /**
1005       * Настройка drag & drop из палитры
1006       */
1007      setupPaletteDragDrop() {
1008          document.querySelectorAll('.palette-item').forEach(item => {
1009              item.addEventListener('mousedown', (e) => {
1010                  // Только левая кнопка мыши должна запускать drag из палитры
1011                  if (e.button !== 0) return;
1012                  e.preventDefault();
1013
1014                  AppState.isDraggingFromPalette = true;
1015                  AppState.dragType = item.dataset.type;
1016
1017                  AppState.dragPreview = document.createElement('div');
1018                  AppState.dragPreview.className = 'drag-preview';
1019                  AppState.dragPreview.textContent =
      ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
1020                  AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1021                  AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1022                  document.body.appendChild(AppState.dragPreview);
1023              });
1024          });
1025      },
1026
1027      /**
1028       * Глобальные обработчики мыши
1029       */
1030  /**
1031   * Глобальные обработчики мыши
1032   */
1033  setupGlobalMouseHandlers() {
1034      document.addEventListener('mousemove', (e) => {
1035          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
1036              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1037              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1038          }
1039          if (AppState.resizing) {
1040              Elements.handleResize(e);
1041              return;
1042          }
1043          if (AppState.draggingElement) {
1044              Elements.handleDrag(e);
1045          }
1046          if (AppState.tempLine && AppState.connectingFrom) {
1047              Connections.drawTempConnection(e);
1048          }
1049      });
1050
1051      document.addEventListener('mouseup', (e) => {
1052          if (AppState.resizing) {
1053              AppState.resizing = null;
1054              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1055          }
1056
1057          if (AppState.isDraggingFromPalette) {
1058              try {
1059                  if (AppState.dragPreview) {
1060                      AppState.dragPreview.remove();
1061                      AppState.dragPreview = null;
1062                  }
```

```
1063
1064                 const container = document.getElementById('workspace-container');
1065                 const rect = container.getBoundingClientRect();
1066
1067                 if (e.clientX >= rect.left && e.clientX <= rect.right &&
1068                     e.clientY >= rect.top && e.clientY <= rect.bottom) {
1069
1070                     const canvasPos = screenToCanvas(e.clientX, e.clientY);
1071                     const config = ELEMENT_TYPES[AppState.dragType];
1072                     if (config) {
1073                         const defaultWidth = config.minWidth || 120;
1074                         const defaultHeight = config.minHeight || 60;
1075
1076                         // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
     обработать
1077                         const newElement = Elements.addElement(
1078                             AppState.dragType,
1079                             canvasPos.x - defaultWidth / 2,
1080                             canvasPos.y - defaultHeight / 2
1081                         );
1082
1083                         if (newElement && typeof Outputs !== 'undefined') {
1084                             Outputs.updateOutputStatus();
1085                         }
1086                     } else {
1087                         console.error('Неизвестный тип элемента при drop:',
     AppState.dragType);
1088                     }
1089                 }
1090             } finally {
1091                 App.cancelPaletteDrag();
1092             }
1093         }
1094
1095         if (AppState.draggingElement) {
1096             AppState.draggingElement = null;
1097         }
1098
1099         Connections.clearConnectionState();
1100     });
1101
1102     document.addEventListener('keydown', (e) => {
1103         if (e.key === 'Delete' && AppState.selectedElement) {
1104             Elements.deleteElement(AppState.selectedElement);
1105             if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1106         }
1107         if (e.key === 'Escape') {
1108             Elements.deselectAll();
1109             Connections.clearConnectionState();
1110             if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
1111         }
1112     });
1113 },
1114
1115     /**
1116      * Настройка контекстного меню
1117      */
1118     setupContextMenu() {
1119         document.addEventListener('click', (e) => {
1120             const menu = document.getElementById('context-menu');
1121             if (!menu.contains(e.target)) {
1122                 menu.style.display = 'none';
1123             }
1124         });
1125
```

```
1126          document.getElementById('ctx-properties').addEventListener('click', () => {
1127              const elemId = document.getElementById('context-menu').dataset.elementId;
1128              document.getElementById('context-menu').style.display = 'none';
1129              const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
1130              if (config?.hasProperties) {
1131                  Modal.showPropertiesModal(elemId);
1132              }
1133          });
1134
1135          document.getElementById('ctx-delete').addEventListener('click', () => {
1136              const elemId = document.getElementById('context-menu').dataset.elementId;
1137              document.getElementById('context-menu').style.display = 'none';
1138              Elements.deleteElement(elemId);
1139              // Обновляем выходы только если модуль загружен
1140              if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1141                  Outputs.updateOutputStatus();
1142              }
1143          });
1144      },
1145
1146      /**
1147       * Клик по рабочей области
1148       */
1149      setupWorkspaceClick() {
1150          const workspace = document.getElementById('workspace');
1151
1152          workspace.addEventListener('click', (e) => {
1153              if (e.target === workspace) {
1154                  Elements.deselectAll();
1155              }
1156          });
1157      },
1158      /**
1159   * --- Выделение рамкой и множественное перемещение ---
1160   */
1161      setupMultiSelection() {
1162          const container = document.getElementById('workspace-container');
1163          const rectEl = document.getElementById('selection-rect');
1164
1165          container.addEventListener('mousedown', (e) => {
1166              if (e.button !== 0) return;
1167              if (e.target !== document.getElementById('workspace')) return;
1168
1169              const pos = screenToCanvas(e.clientX, e.clientY);
1170              AppState.multiSelecting = true;
1171              AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
      pos.y, w: 0, h: 0 };
1172
1173              rectEl.style.left = e.clientX + 'px';
1174              rectEl.style.top = e.clientY + 'px';
1175              rectEl.style.width = '0px';
1176              rectEl.style.height = '0px';
1177              rectEl.style.display = 'block';
1178          });
1179
1180          document.addEventListener('mousemove', (e) => {
1181              if (!AppState.multiSelecting) return;
1182
1183              const pos = screenToCanvas(e.clientX, e.clientY);
1184              const sx = AppState.selectionRect.startX;
1185              const sy = AppState.selectionRect.startY;
1186              const x = Math.min(sx, pos.x);
1187              const y = Math.min(sy, pos.y);
1188              const w = Math.abs(pos.x - sx);
1189              const h = Math.abs(pos.y - sy);
```

```
1190
1191              rectEl.style.left = x * AppState.viewport.zoom + AppState.viewport.panX +
      'px';
1192              rectEl.style.top = y * AppState.viewport.zoom + AppState.viewport.panY +
      'px';
1193              rectEl.style.width = w * AppState.viewport.zoom + 'px';
1194              rectEl.style.height = h * AppState.viewport.zoom + 'px';
1195
1196              const selected = [];
1197              for (const [id, elData] of Object.entries(AppState.elements)) {
1198              if (!elData || elData.type === 'output-frame') continue;
1199              if (
1200                  elData.x >= x && elData.x + elData.width <= x + w &&
1201                  elData.y >= y && elData.y + elData.height <= y + h
1202              ) selected.push(id);
1203              }
1204
1205              AppState.selectedElements = selected;
1206              document.querySelectorAll('.element').forEach(el =>
1207              el.classList.toggle('selected', selected.includes(el.id))
1208              );
1209          });
1210
1211          document.addEventListener('mouseup', () => {
1212              if (AppState.multiSelecting) {
1213              AppState.multiSelecting = false;
1214              rectEl.style.display = 'none';
1215              }
1216          });
1217      }
1218  };
1219
1220  // Запуск приложения при загрузке страницы
1221  document.addEventListener('DOMContentLoaded', () => {
1222      App.init();
1223  });
1224
1225  codegen_graph.js
1226  // js/codegen_graph.js
1227
1228
1229  const CodeGenGraph = {
1230          /**
1231       * Собрать все условия вверх по цепочке cond-портов (до корня).
1232       * Возвращает null или объединённое через AND условие.
1233       */
1234  /**
1235   * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
1236   */
1237      collectAllCond(graph) {
1238          if (!graph) return null;
1239
1240          let c = null;
1241          const elem = graph.elem;
1242
1243          // 1. Собираем условия через cond-порт (как было)
1244          if (graph.condInput) {
1245              const condConn = graph.condInput.conn;
1246              const fromGraph = graph.condInput.fromGraph;
1247              const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
1248              c = oneCond;
1249
1250              // Рекурсивно идём вверх по cond-цепочке
1251              const upCond = this.collectAllCond(fromGraph);
1252              if (upCond) {
```

```
                c = c ? Optimizer.And(c, upCond) : upCond;
            }
        }

        // 2. НОВОЕ: если это separator — учитываем контекст его входа
        if (elem.type === 'separator' && graph.inputs.length > 0) {
            const inputGraph = graph.inputs[0].fromGraph;
            const inputContext = this.collectAllCond(inputGraph);
            if (inputContext) {
                c = c ? Optimizer.And(c, inputContext) : inputContext;
            }
        }

        return c;
    },
    buildDependencyGraph(elementId) {
        const graph = {
            nodeId: elementId,
            elem: AppState.elements[elementId],
            inputs: [],
            condInput: null,
        };

        if (!graph.elem) return null;

        const inConns = AppState.connections
        .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
        .sort((a, b) => {
            const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
            const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
            return ai - bi;
        });

        inConns.forEach(conn => {
        graph.inputs.push({
            conn,
            fromGraph: this.buildDependencyGraph(conn.fromElement)
        });
        });

        const condConn = AppState.connections.find(c =>
            c.toElement === elementId && c.toPort === 'cond-0'
        );
        if (condConn) {
            graph.condInput = {
                conn: condConn,
                fromGraph: this.buildDependencyGraph(condConn.fromElement)
            };
        }

        return graph;
    },

    /**
     * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
     */
    evalLogic(graph) {
        if (!graph) return Optimizer.TrueCond;
        const elem = graph.elem;

        switch (elem.type) {
            case 'if': {
                const left = graph.inputs[0]?.fromGraph;
                const right = graph.inputs[1]?.fromGraph;

```

```
                const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
                const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);

                const op = elem.props.operator || '=';
                return this.buildIfLogic(leftVal, op, rightVal);
            }

            case 'and': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.And(result, inLogic) : inLogic;
                }
                return result || Optimizer.TrueCond;
            }

            case 'or': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.Or(result, inLogic) : inLogic;
                }
                return result || Optimizer.FalseCond;
            }

            case 'not': {
                const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
                return Optimizer.Not(inLogic);
            }

            case 'separator': {
                return this.evalLogic(graph.inputs[0]?.fromGraph);
            }

            default:
                return Optimizer.TrueCond;
        }
    },

    /**
     * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
     */
    evalValue(graph) {
        if (!graph) return Optimizer.Const(0);
        const elem = graph.elem;

        switch (elem.type) {
            case 'input-signal':
                return Optimizer.Var(elem.props.name || graph.nodeId);

            case 'const':
                return Optimizer.Const(Number(elem.props.value) || 0);

            case 'formula': {
                const expr = this.buildFormulaExpr(elem);
                return Optimizer.Var(expr);
            }

            case 'separator':
                return this.evalValue(graph.inputs[0]?.fromGraph);

            default:
                return Optimizer.Const(0);
        }
    },
```

```
1383
1384      // js/codegen_graph.js
1385
1386      /**
1387       * Рекурсивно собрать полный контекст условий для элемента
1388       * через всю цепочку cond-портов вверх
1389       */
1390    // В codegen_graph.js, в evalFullContext добавь:
1391
1392      evalFullContext(graph) {
1393          if (!graph) return null;
1394
1395          let context = null;
1396          const elem = graph.elem;
1397
1398          console.log(`evalFullContext для ${elem.id} (${elem.type})`);
1399
1400          // 1. Если сам элемент имеет cond-порт — собираем его условие
1401          if (graph.condInput) {
1402              const condConn = graph.condInput.conn;
1403              console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
      {condConn.fromPort}`);
1404
1405              const condLogic = this.evalConditionFromPort(
1406                  graph.condInput.fromGraph,
1407                  condConn.fromPort
1408              );
1409              console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
1410              context = condLogic;
1411
1412              // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
1413              const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
1414              if (upstreamContext) {
1415                  console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
1416                  context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
1417              }
1418          } else {
1419              console.log(`  → нет cond-0`);
1420          }
1421
1422          console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
1423          return context;
1424      },
1425
1426      /**
1427       * Получить УСЛОВИЕ для cond-порта элемента
1428       * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
1429       */
1430      evalConditionFromPort(graph, fromPort) {
1431          if (!graph) return null;
1432          const elem = graph.elem;
1433
1434          // Если это сепаратор — вычисляем его вход и применяем ветвление
1435          if (elem.type === 'separator') {
1436              const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
1437
1438              if (fromPort === 'out-0') {
1439                  return inputLogic;
1440              } else if (fromPort === 'out-1') {
1441                  return Optimizer.Not(inputLogic);
1442              }
1443          }
```

```
1444
1445          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
1446          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
1447              return this.evalLogic(graph);
1448          }
1449
1450          return null;
1451      },
1452
1453      /**
1454       * Главная функция: получить {cond, expr} для элемента
1455       */
1456      evalGraphValue(graph) {
1457
1458          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
1459
1460          const elem = graph.elem;
1461          //let cond = null;
1462
1463          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
1464          let cond = this.collectAllCond(graph);
1465
1466          let expr = null;
1467
1468          switch (elem.type) {
1469              case 'input-signal':
1470                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
1471                  break;
1472
1473              case 'const':
1474                  expr = Optimizer.Const(Number(elem.props.value) || 0);
1475                  break;
1476
1477              case 'formula': {
1478                  // Для формулы также собираем условия от всех входных элементов
1479                  const inputConds = graph.inputs.map(inp => {
1480                      const inResult = this.evalGraphValue(inp.fromGraph);
1481                      return inResult.cond;
1482                  }).filter(c => c);
1483
1484                  // Объединяем cond-порт с условиями от входов
1485                  for (const inCond of inputConds) {
1486                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
1487                  }
1488
1489                  expr = Optimizer.Var(this.buildFormulaExpr(elem));
1490                  break;
1491              }
1492
1493              case 'separator':
1494                  // Сепаратор — просто пробрасываем значение дальше
1495                  return this.evalGraphValue(graph.inputs[0]?.fromGraph);
1496
1497              // Логические элементы не должны здесь быть
1498              case 'and':
1499              case 'or':
1500              case 'not':
1501              case 'if':
1502              default:
1503                  expr = Optimizer.Const(0);
1504          }
1505
1506          return { cond, expr };
1507      },
```

```
1508
1509      buildIfLogic(leftVal, op, rightVal) {
1510          const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
1511          const rightName = rightVal.type === 'var' ? rightVal.name :
      String(rightVal.n);
1512
1513          const leftZero = leftVal.type === 'const' && leftVal.n === 0;
1514          const rightZero = rightVal.type === 'const' && rightVal.n === 0;
1515
1516          switch (op) {
1517              case '=':
1518                  if (rightZero) return Optimizer.Eq0(leftName);
1519                  if (leftZero) return Optimizer.Eq0(rightName);
1520                  return Optimizer.Cmp(leftName, '=', rightName);
1521              case '!=':
1522                  if (rightZero) return Optimizer.Ne0(leftName);
1523                  if (leftZero) return Optimizer.Ne0(rightName);
1524                  return Optimizer.Cmp(leftName, '!=', rightName);
1525              case '>':
1526              case '<':
1527              case '>=':
1528              case '<=':
1529                  return Optimizer.Cmp(leftName, op, rightName);
1530              default:
1531                  return Optimizer.TrueCond;
1532          }
1533      },
1534
1535
1536      buildFormulaExpr(elem) {
1537          let result = elem.props.expression || '0';
1538
1539          // 1) Сначала раскрываем шаблоны (h и др.)
1540          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
1541              ? Settings.getTemplatesMap()
1542              : null;
1543          result = expandFormulaTemplates(result, map);
1544
1545          // 2) Потом раскрываем ссылки на формулы
1546          const formulaRefs = result.match(/formula[_-]\d+/g) || [];
1547          for (const ref of formulaRefs) {
1548              const refElem = AppState.elements[ref];
1549              if (refElem && refElem.type === 'formula') {
1550                  const refExpr = this.buildFormulaExpr(refElem);
1551                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
1552              }
1553          }
1554
1555          return result;
1556      }
1557  };
1558
1559  window.CodeGenGraph = CodeGenGraph;
1560
1561  codegen_graph.js
1562
1563  // js/codegen_graph.js
1564
1565
1566  const CodeGenGraph = {
1567          /**
1568       * Собрать все условия вверх по цепочке cond-портов (до корня).
1569       * Возвращает null или объединённое через AND условие.
1570       */
1571  /**
```

```
1572   * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
1573   */
1574      collectAllCond(graph) {
1575          if (!graph) return null;
1576
1577          let c = null;
1578          const elem = graph.elem;
1579
1580          // 1. Собираем условия через cond-порт (как было)
1581          if (graph.condInput) {
1582              const condConn = graph.condInput.conn;
1583              const fromGraph = graph.condInput.fromGraph;
1584              const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
1585              c = oneCond;
1586
1587              // Рекурсивно идём вверх по cond-цепочке
1588              const upCond = this.collectAllCond(fromGraph);
1589              if (upCond) {
1590                  c = c ? Optimizer.And(c, upCond) : upCond;
1591              }
1592          }
1593
1594          // 2. НОВОЕ: если это separator — учитываем контекст его входа
1595          if (elem.type === 'separator' && graph.inputs.length > 0) {
1596              const inputGraph = graph.inputs[0].fromGraph;
1597              const inputContext = this.collectAllCond(inputGraph);
1598              if (inputContext) {
1599                  c = c ? Optimizer.And(c, inputContext) : inputContext;
1600              }
1601          }
1602
1603          return c;
1604      },
1605      buildDependencyGraph(elementId) {
1606          const graph = {
1607              nodeId: elementId,
1608              elem: AppState.elements[elementId],
1609              inputs: [],
1610              condInput: null,
1611          };
1612
1613          if (!graph.elem) return null;
1614
1615          const inConns = AppState.connections
1616          .filter(c => c.toElement === elementId && c.toPort.startsWith('in-'))
1617          .sort((a, b) => {
1618              const ai = parseInt(a.toPort.split('-')[1] || '0', 10);
1619              const bi = parseInt(b.toPort.split('-')[1] || '0', 10);
1620              return ai - bi;
1621          });
1622
1623          inConns.forEach(conn => {
1624          graph.inputs.push({
1625              conn,
1626              fromGraph: this.buildDependencyGraph(conn.fromElement)
1627          });
1628          });
1629
1630          const condConn = AppState.connections.find(c =>
1631              c.toElement === elementId && c.toPort === 'cond-0'
1632          );
1633          if (condConn) {
1634              graph.condInput = {
1635                  conn: condConn,
1636                  fromGraph: this.buildDependencyGraph(condConn.fromElement)
```

```
            };
        }

        return graph;
    },

    /**
     * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
     */
    evalLogic(graph) {
        if (!graph) return Optimizer.TrueCond;
        const elem = graph.elem;

        switch (elem.type) {
            case 'if': {
                const left = graph.inputs[0]?.fromGraph;
                const right = graph.inputs[1]?.fromGraph;

                const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
                const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);

                const op = elem.props.operator || '=';
                return this.buildIfLogic(leftVal, op, rightVal);
            }

            case 'and': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.And(result, inLogic) : inLogic;
                }
                return result || Optimizer.TrueCond;
            }

            case 'or': {
                let result = null;
                for (const inp of graph.inputs) {
                    const inLogic = this.evalLogic(inp.fromGraph);
                    result = result ? Optimizer.Or(result, inLogic) : inLogic;
                }
                return result || Optimizer.FalseCond;
            }

            case 'not': {
                const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
                return Optimizer.Not(inLogic);
            }

            case 'separator': {
                return this.evalLogic(graph.inputs[0]?.fromGraph);
            }

            default:
                return Optimizer.TrueCond;
        }
    },

    /**
     * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
     */
    evalValue(graph) {
        if (!graph) return Optimizer.Const(0);
        const elem = graph.elem;

        switch (elem.type) {
```

```
1702              case 'input-signal':
1703                  return Optimizer.Var(elem.props.name || graph.nodeId);
1704
1705              case 'const':
1706                  return Optimizer.Const(Number(elem.props.value) || 0);
1707
1708              case 'formula': {
1709                  const expr = this.buildFormulaExpr(elem);
1710                  return Optimizer.Var(expr);
1711              }
1712
1713              case 'separator':
1714                  return this.evalValue(graph.inputs[0]?.fromGraph);
1715
1716              default:
1717                  return Optimizer.Const(0);
1718          }
1719      },
1720
1721      // js/codegen_graph.js
1722
1723      /**
1724       * Рекурсивно собрать полный контекст условий для элемента
1725       * через всю цепочку cond-портов вверх
1726       */
1727   // В codegen_graph.js, в evalFullContext добавь:
1728
1729      evalFullContext(graph) {
1730          if (!graph) return null;
1731
1732          let context = null;
1733          const elem = graph.elem;
1734
1735          console.log(`evalFullContext для ${elem.id} (${elem.type})`);
1736
1737          // 1. Если сам элемент имеет cond-порт — собираем его условие
1738          if (graph.condInput) {
1739              const condConn = graph.condInput.conn;
1740              console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
      {condConn.fromPort}`);
1741
1742              const condLogic = this.evalConditionFromPort(
1743                  graph.condInput.fromGraph,
1744                  condConn.fromPort
1745              );
1746              console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
1747              context = condLogic;
1748
1749              // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
1750              const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
1751              if (upstreamContext) {
1752                  console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
1753                  context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
1754              }
1755          } else {
1756              console.log(`  → нет cond-0`);
1757          }
1758
1759          console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
1760          return context;
1761      },
1762
```

```
1763      /**
1764       * Получить УСЛОВИЕ для cond-порта элемента
1765       * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
1766       */
1767      evalConditionFromPort(graph, fromPort) {
1768          if (!graph) return null;
1769          const elem = graph.elem;
1770
1771          // Если это сепаратор — вычисляем его вход и применяем ветвление
1772          if (elem.type === 'separator') {
1773              const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
1774
1775              if (fromPort === 'out-0') {
1776                  return inputLogic;
1777              } else if (fromPort === 'out-1') {
1778                  return Optimizer.Not(inputLogic);
1779              }
1780          }
1781
1782          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
1783          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
1784              return this.evalLogic(graph);
1785          }
1786
1787          return null;
1788      },
1789
1790      /**
1791       * Главная функция: получить {cond, expr} для элемента
1792       */
1793      evalGraphValue(graph) {
1794
1795          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
1796
1797          const elem = graph.elem;
1798          //let cond = null;
1799
1800          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
1801          let cond = this.collectAllCond(graph);
1802
1803          let expr = null;
1804
1805          switch (elem.type) {
1806              case 'input-signal':
1807                  expr = Optimizer.Var(elem.props.name || graph.nodeId);
1808                  break;
1809
1810              case 'const':
1811                  expr = Optimizer.Const(Number(elem.props.value) || 0);
1812                  break;
1813
1814              case 'formula': {
1815                  // Для формулы также собираем условия от всех входных элементов
1816                  const inputConds = graph.inputs.map(inp => {
1817                      const inResult = this.evalGraphValue(inp.fromGraph);
1818                      return inResult.cond;
1819                  }).filter(c => c);
1820
1821                  // Объединяем cond-порт с условиями от входов
1822                  for (const inCond of inputConds) {
1823                      cond = cond ? Optimizer.And(cond, inCond) : inCond;
1824                  }
1825
1826                  expr = Optimizer.Var(this.buildFormulaExpr(elem));
```

```
                break;
            }

            case 'separator':
                // Сепаратор — просто пробрасываем значение дальше
                return this.evalGraphValue(graph.inputs[0]?.fromGraph);

            // Логические элементы не должны здесь быть
            case 'and':
            case 'or':
            case 'not':
            case 'if':
            default:
                expr = Optimizer.Const(0);
        }

        return { cond, expr };
    },

    buildIfLogic(leftVal, op, rightVal) {
        const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
        const rightName = rightVal.type === 'var' ? rightVal.name :
    String(rightVal.n);

        const leftZero = leftVal.type === 'const' && leftVal.n === 0;
        const rightZero = rightVal.type === 'const' && rightVal.n === 0;

        switch (op) {
            case '=':
                if (rightZero) return Optimizer.Eq0(leftName);
                if (leftZero) return Optimizer.Eq0(rightName);
                return Optimizer.Cmp(leftName, '=', rightName);
            case '!=':
                if (rightZero) return Optimizer.Ne0(leftName);
                if (leftZero) return Optimizer.Ne0(rightName);
                return Optimizer.Cmp(leftName, '!=', rightName);
            case '>':
            case '<':
            case '>=':
            case '<=':
                return Optimizer.Cmp(leftName, op, rightName);
            default:
                return Optimizer.TrueCond;
        }
    },


    buildFormulaExpr(elem) {
        let result = elem.props.expression || '0';

        // 1) Сначала раскрываем шаблоны (h и др.)
        const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
            ? Settings.getTemplatesMap()
            : null;
        result = expandFormulaTemplates(result, map);

        // 2) Потом раскрываем ссылки на формулы
        const formulaRefs = result.match(/formula[_-]\d+/g) || [];
        for (const ref of formulaRefs) {
            const refElem = AppState.elements[ref];
            if (refElem && refElem.type === 'formula') {
                const refExpr = this.buildFormulaExpr(refElem);
                result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
            }
        }
```

```
1891
1892          return result;
1893      }
1894  };
1895
1896  window.CodeGenGraph = CodeGenGraph;
1897
1898  codegen_optimizer.js
1899
1900  // js/codegen_optimizer.js
1901
1902  let _depth = 0;
1903  const MAX_DEPTH = 200;
1904
1905  // === Конструкторы ===
1906  function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
1907  function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
1908  function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
1909  function And(a, b) {
1910      if (!a) return b;
1911      if (!b) return a;
1912      return { kind: 'cond', type: 'and', a, b };
1913  }
1914  function Or(a, b) {
1915      if (!a) return b;
1916      if (!b) return a;
1917      return { kind: 'cond', type: 'or', a, b };
1918  }
1919  function Not(x) {
1920      if (!x) return null;
1921      return { kind: 'cond', type: 'not', x };
1922  }
1923  const TrueCond = { kind: 'cond', type: 'true' };
1924  const FalseCond = { kind: 'cond', type: 'false' };
1925
1926  function Const(n) { return { kind: 'expr', type: 'const', n }; }
1927  function Var(name) { return { kind: 'expr', type: 'var', name }; }
1928  function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
1929  function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }
1930
1931  // === Утилиты ===
1932  function atomKey(c) {
1933      if (!c) return null;
1934      switch (c.type) {
1935          case 'eq0': return `eq0:${c.v}`;
1936          case 'ne0': return `ne0:${c.v}`;
1937          case 'cmp': return `cmp:${c.l}:${c.op}:${c.r}`;
1938          case 'true': return 'true';
1939          case 'false': return 'false';
1940          default: return null;
1941      }
1942  }
1943
1944  function splitAndCond(c) {
1945      if (!c || c.type !== 'and') return null;
1946      return [c.a, c.b];
1947  }
1948
1949  function findSharedAndComplement(c1, c2) {
1950      const p1 = splitAndCond(c1);
1951      const p2 = splitAndCond(c2);
1952      if (!p1 || !p2) return null;
1953
1954      const combos = [
1955          [p1[0], p1[1], p2[0], p2[1]],
```

```
            [p1[0], p1[1], p2[1], p2[0]],
            [p1[1], p1[0], p2[0], p2[1]],
            [p1[1], p1[0], p2[1], p2[0]],
        ];

        for (const [s1, x1, s2, x2] of combos) {
            if (condEq(s1, s2) && condNegationEq(x1, x2)) {
                return { shared: s1 };
            }
        }
        return null;
    }

    function negateOp(op) {
        switch (op) {
            case '=': return '!=';
            case '!=': return '=';
            case '>': return '<=';
            case '<': return '>=';
            case '>=': return '<';
            case '<=': return '>';
            default: return null;
        }
    }

    // Преобразует cmp-условие в интервал по одной переменной
    // Возвращает { varName, min, minInc, max, maxInc } или null
    function cmpToInterval(c) {
        if (!c || c.type !== 'cmp') return null;

        const lNum = parseNumberLiteral(c.l);
        const rNum = parseNumberLiteral(c.r);

        let varName, op, val;

        if (lNum == null && rNum != null) {
            // var OP const
            varName = c.l;
            op = c.op;
            val = rNum;
        } else if (lNum != null && rNum == null) {
            // const OP var  ->  var (OP') const
            varName = c.r;
            op = reverseOp(c.op);
            if (!op) return null;
            val = lNum;
        } else {
            // Либо обе стороны числа, либо обе не числа — не трогаем
            return null;
        }

        // Интересуют только упорядочивающие операторы
        switch (op) {
            case '<':
            case '<=':
            case '>':
            case '>=':
            case '=':
                break;
            default:
                return null;
        }

        let min = Number.NEGATIVE_INFINITY;
        let max = Number.POSITIVE_INFINITY;
```

```
2021      let minInc = false;
2022      let maxInc = false;
2023
2024      switch (op) {
2025          case '<':
2026              max = val; maxInc = false; break;
2027          case '<=':
2028              max = val; maxInc = true; break;
2029          case '>':
2030              min = val; minInc = false; break;
2031          case '>=':
2032              min = val; minInc = true; break;
2033          case '=':
2034              min = val; minInc = true;
2035              max = val; maxInc = true;
2036              break;
2037      }
2038
2039      return { varName, min, minInc, max, maxInc };
2040  }
2041
2042  function intervalSubset(a, b) {
2043      if (!a || !b) return false;
2044
2045      // Нижняя граница: a.min >= b.min
2046      const amin = a.min, bmin = b.min;
2047      if (amin === Number.NEGATIVE_INFINITY) {
2048          if (bmin !== Number.NEGATIVE_INFINITY) return false;
2049          // оба -∞ — ок
2050      } else if (bmin === Number.NEGATIVE_INFINITY) {
2051          // b начинается “раньше” — ок
2052      } else if (amin > bmin) {
2053          // a стартует правее b — ок
2054      } else if (amin < bmin) {
2055          // a захватывает меньшее значение — не подмножество
2056          return false;
2057      } else {
2058          // amin === bmin
2059          if (a.minInc && !b.minInc) {
2060              // a включает границу, а b — нет → в a есть точка, не входящая в b
2061              return false;
2062          }
2063      }
2064
2065      // Верхняя граница: a.max <= b.max
2066      const amax = a.max, bmax = b.max;
2067      if (amax === Number.POSITIVE_INFINITY) {
2068          if (bmax !== Number.POSITIVE_INFINITY) return false;
2069      } else if (bmax === Number.POSITIVE_INFINITY) {
2070          // b идёт дальше — ок
2071      } else if (amax < bmax) {
2072          // a заканчивается раньше — ок
2073      } else if (amax > bmax) {
2074          return false;
2075      } else {
2076          // amax === bmax
2077          if (a.maxInc && !b.maxInc) {
2078              return false;
2079          }
2080      }
2081
2082      return true;
2083  }
2084
2085  // Удаляет избыточные cmp-условия в массиве атомов
```

```
// mode: 'and' | 'or'
function removeRedundantCmpAtoms(atoms, mode) {
    if (!atoms || atoms.length < 2) return atoms;

    const keep = new Array(atoms.length).fill(true);

    for (let i = 0; i < atoms.length; i++) {
        if (!keep[i]) continue;
        const a = atoms[i];
        if (!a || a.type !== 'cmp') continue;

        for (let j = 0; j < atoms.length; j++) {
            if (i === j || !keep[j]) continue;
            const b = atoms[j];
            if (!b || b.type !== 'cmp') continue;

            const rel = cmpImplicationRelation(a, b);
            if (!rel) continue;

            if (rel === 'a_in_b') {
                if (mode === 'or') {
                    // A ⊆ B → A OR B = B  → A лишнее
                    keep[i] = false;
                    break;
                } else if (mode === 'and') {
                    // A ⊆ B → A AND B = A → B лишнее
                    keep[j] = false;
                }
            } else if (rel === 'b_in_a') {
                if (mode === 'or') {
                    // B ⊆ A → A OR B = A → B лишнее
                    keep[j] = false;
                } else if (mode === 'and') {
                    // B ⊆ A → A AND B = B → A лишнее
                    keep[i] = false;
                    break;
                }
            }
        }
    }

    return atoms.filter((_, idx) => keep[idx]);
}

// Отношение между двумя cmp-условиями через интервалы
// 'a_in_b'  — A ⊆ B
// 'b_in_a'  — B ⊆ A
// 'equal'   — одинаковые интервалы (редко используем)
// null      — не можем определить
function cmpImplicationRelation(c1, c2) {
    const i1 = cmpToInterval(c1);
    const i2 = cmpToInterval(c2);
    if (!i1 || !i2) return null;
    if (i1.varName !== i2.varName) return null;

    const aInB = intervalSubset(i1, i2);
    const bInA = intervalSubset(i2, i1);

    if (aInB && bInA) return 'equal';
    if (aInB) return 'a_in_b';
    if (bInA) return 'b_in_a';
    return null;
}

// Разворот оператора при перестановке аргументов (левый/правый)
```

```
function reverseOp(op) {
    switch (op) {
        case '<':  return '>';
        case '>':  return '<';
        case '<=': return '>=';
        case '>=': return '<=';
        case '=':
        case '!=':
            return op;
        default:
            return null;
    }
}

// Аккуратный парсер числового литерала.
// Возвращает число или null, если строка не чисто числовая.
function parseNumberLiteral(s) {
    if (typeof s !== 'string') return null;
    const trimmed = s.trim().replace(',', '.');

    // Только простые вещи: -123, 45, 3.14
    if (!/^-?\d+(\.\d+)?$/.test(trimmed)) return null;

    const n = Number(trimmed);
    return Number.isFinite(n) ? n : null;
}


function negateAtomKey(key) {
    if (!key) return null;
    if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
    if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
    if (key.startsWith('cmp:')) {
        const parts = key.slice(4).split(':');
        if (parts.length === 3) {
            const negOp = negateOp(parts[1]);
            if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
        }
    }
    return null;
}

function isNegation(a, b) {
    if (!a || !b) return false;
    if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
    if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
    if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
        return a.op === negateOp(b.op);
    }
    if (a.type === 'not' && condEq(a.x, b)) return true;
    if (b.type === 'not' && condEq(b.x, a)) return true;
    return false;
}

function isAtomCond(t) {
    return t && (t.type === 'eq0' || t.type === 'ne0' || t.type === 'cmp');
}

function pruneOrByContext(orTerm, contextAtoms) {
    const branches = flattenOr(orTerm);
    const kept = [];

    for (const br of branches) {
        let contradicts = false;

```

```
2216          for (const ctx of contextAtoms) {
2217              if (isNegation(br, ctx)) {
2218                  contradicts = true;
2219                  break;
2220              }
2221          }
2222
2223          if (!contradicts) kept.push(br);
2224      }
2225
2226      if (kept.length === 0) return FalseCond;
2227      if (kept.length === 1) return kept[0];
2228      return buildOr(kept);
2229  }
2230
2231  function condNegationEq(a, b) {
2232      if (!a || !b) return false;
2233
2234      // Простая проверка: a == NOT(b)
2235      if (condEq(a, Not(b)) || condEq(b, Not(a))) return true;
2236
2237      // Де Морган: NOT(A OR B) == (NOT A AND NOT B)
2238      // Проверяем: если a = (A OR B), то b должно быть (NOT A AND NOT B)
2239      if (a.type === 'or' && b.type === 'and') {
2240          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2241                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2242      }
2243      // Симметрично
2244      if (a.type === 'and' && b.type === 'or') {
2245          return condNegationEq(a.a, b.a) && condNegationEq(a.b, b.b) ||
2246                 condNegationEq(a.a, b.b) && condNegationEq(a.b, b.a);
2247      }
2248
2249      // Проверка атомов: (X = 0) vs (X != 0)
2250      if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
2251      if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
2252
2253      // Проверка сравнений: (X > Y) vs (X <= Y) и т.д.
2254      if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
2255          return a.op === negateOp(b.op);
2256      }
2257
2258      return false;
2259  }
2260
2261
2262
2263  function condEq(a, b) {
2264      if (a === b) return true;
2265      if (!a || !b) return false;
2266      if (a.type !== b.type) return false;
2267
2268      switch (a.type) {
2269          case 'eq0':
2270          case 'ne0':
2271              return a.v === b.v;
2272          case 'cmp':
2273              return a.l === b.l && a.op === b.op && a.r === b.r;
2274          case 'true':
2275          case 'false':
2276              return true;
2277          case 'not':
2278              return condEq(a.x, b.x);
2279          case 'and':
2280          case 'or':
```

```
            return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
                   (condEq(a.a, b.b) && condEq(a.b, b.a));
        default:
            return false;
    }
}

function flattenAnd(c) {
    if (!c) return [];
    if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
    return [c];
}

function flattenOr(c) {
    if (!c) return [];
    if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
    return [c];
}

function buildAnd(terms) {
    if (terms.length === 0) return TrueCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = And(result, terms[i]);
    }
    return result;
}

function buildOr(terms) {
    if (terms.length === 0) return FalseCond;
    let result = terms[0];
    for (let i = 1; i < terms.length; i++) {
        result = Or(result, terms[i]);
    }
    return result;
}

// Поглощение для AND: X AND (X OR Y) = X
function applyAndAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'or') continue;

        const orParts = flattenOr(ti);
        let drop = false;

        outer:
        for (const part of orParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
```

```
    }

    return terms.filter((_, idx) => keep[idx]);
}

// Поглощение для OR: X OR (X AND Y) = X
function applyOrAbsorption(terms) {
    if (!terms || terms.length < 2) return terms;

    const keep = new Array(terms.length).fill(true);

    for (let i = 0; i < terms.length; i++) {
        if (!keep[i]) continue;
        const ti = terms[i];
        if (!ti || ti.type !== 'and') continue;

        const andParts = flattenAnd(ti);
        let drop = false;

        outer:
        for (const part of andParts) {
            for (let j = 0; j < terms.length; j++) {
                if (j === i || !keep[j]) continue;
                if (condEq(part, terms[j])) {
                    drop = true;
                    break outer;
                }
            }
        }

        if (drop) {
            keep[i] = false;
        }
    }

    return terms.filter((_, idx) => keep[idx]);
}

// === Упрощение условий ===
function simplifyCond(c) {
    _depth++;
    if (_depth > MAX_DEPTH) {
        _depth--;
        return c;
    }

    try {
        return simplifyCondCore(c);
    } finally {
        _depth--;
    }
}

function simplifyCondCore(c) {
    if (!c || c.kind !== 'cond') return c;

    switch (c.type) {
        case 'true':
        case 'false':
        case 'eq0':
        case 'ne0':
        case 'cmp':
            return c;

        case 'not': {
```

```
2411              const x = simplifyCondCore(c.x);
2412              if (!x) return TrueCond;
2413              if (x.type === 'true') return FalseCond;
2414              if (x.type === 'false') return TrueCond;
2415              if (x.type === 'not') return simplifyCondCore(x.x);
2416              if (x.type === 'eq0') return Ne0(x.v);
2417              if (x.type === 'ne0') return Eq0(x.v);
2418              if (x.type === 'cmp') {
2419                  const negOp = negateOp(x.op);
2420                  if (negOp) return Cmp(x.l, negOp, x.r);
2421              }
2422              if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
2423              if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
2424              return Not(x);
2425          }
2426
2427  case 'and': {
2428      const a = simplifyCondCore(c.a);
2429      const b = simplifyCondCore(c.b);
2430
2431      if (!a) return b;
2432      if (!b) return a;
2433      if (a.type === 'false' || b.type === 'false') return FalseCond;
2434      if (a.type === 'true') return b;
2435      if (b.type === 'true') return a;
2436
2437      const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
2438
2439      // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
2440      const eq0Vars = new Map(); // var -> term
2441      const ne0Vars = new Map(); // var -> term
2442      const cmpTerms = [];
2443      const otherTerms = [];
2444
2445      for (const t of allTerms) {
2446          if (t.type === 'true') continue;
2447          if (t.type === 'false') return FalseCond;
2448
2449          if (t.type === 'eq0') {
2450              // Проверка на противоречие сразу
2451              if (ne0Vars.has(t.v)) {
2452                  console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
2453                  return FalseCond;
2454              }
2455              eq0Vars.set(t.v, t);
2456          } else if (t.type === 'ne0') {
2457              // Проверка на противоречие сразу
2458              if (eq0Vars.has(t.v)) {
2459                  console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
2460                  return FalseCond;
2461              }
2462              ne0Vars.set(t.v, t);
2463          } else if (t.type === 'cmp') {
2464              cmpTerms.push(t);
2465          } else if (t.type === 'or') {
2466              // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
2467              const orTerms = flattenOr(t);
2468              const validBranches = [];
2469
2470              for (const branch of orTerms) {
2471                  let branchValid = true;
2472
2473                  if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
2474                      console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
      {branch.v} = 0`);
```

```
2475                     branchValid = false;
2476                 } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
2477                     console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
      {branch.v} != 0`);
2478                     branchValid = false;
2479                 }
2480 
2481                 if (branchValid) {
2482                     validBranches.push(branch);
2483                 }
2484             }
2485 
2486             if (validBranches.length === 0) {
2487                 console.log(`Все ветки OR противоречат контексту → FALSE`);
2488                 return FalseCond;
2489             } else if (validBranches.length === 1) {
2490                 // Если осталась только одна ветка OR, добавляем её напрямую
2491                 const singleBranch = validBranches[0];
2492                 if (singleBranch.type === 'eq0') {
2493                     if (ne0Vars.has(singleBranch.v)) return FalseCond;
2494                     eq0Vars.set(singleBranch.v, singleBranch);
2495                 } else if (singleBranch.type === 'ne0') {
2496                     if (eq0Vars.has(singleBranch.v)) return FalseCond;
2497                     ne0Vars.set(singleBranch.v, singleBranch);
2498                 } else {
2499                     otherTerms.push(singleBranch);
2500                 }
2501             } else {
2502                 // Перестраиваем OR только с валидными ветками
2503                 otherTerms.push(buildOr(validBranches));
2504             }
2505         } else {
2506             otherTerms.push(t);
2507         }
2508     }
2509 
2510     // Собираем уникальные атомы
2511     const atomMap = new Map();
2512 
2513     for (const [v, term] of eq0Vars) {
2514         const key = atomKey(term);
2515         if (key) atomMap.set(key, term);
2516     }
2517 
2518     for (const [v, term] of ne0Vars) {
2519         const key = atomKey(term);
2520         if (key) atomMap.set(key, term);
2521     }
2522 
2523     for (const term of cmpTerms) {
2524         const key = atomKey(term);
2525         if (key) {
2526             const negKey = negateAtomKey(key);
2527             if (negKey && atomMap.has(negKey)) {
2528                 return FalseCond;
2529             }
2530             if (!atomMap.has(key)) {
2531                 atomMap.set(key, term);
2532             }
2533         }
2534     }
2535 
2536     let uniqueAtoms = Array.from(atomMap.values());
2537     uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
2538 
```

```
2539      let result = [...uniqueAtoms, ...otherTerms];
2540
2541      // Поглощение: X AND (X OR Y) = X
2542      // === НОВОЕ: выбрасываем из OR ветки, противоречащие контексту AND ===
2543      const contextAtoms = result.filter(t => isAtomCond(t));
2544      result = result.map(t => {
2545          if (t.type !== 'or') return t;
2546          return pruneOrByContext(t, contextAtoms);
2547      }).filter(t => t.type !== 'true'); // на всякий случай
2548
2549      result = applyAndAbsorption(result);
2550
2551      if (result.length === 0) return TrueCond;
2552      if (result.length === 1) return result[0];
2553
2554      return buildAnd(result);
2555  }
2556
2557          case 'or': {
2558              const a = simplifyCondCore(c.a);
2559              const b = simplifyCondCore(c.b);
2560
2561              if (!a) return b;
2562              if (!b) return a;
2563              if (a.type === 'true' || b.type === 'true') return TrueCond;
2564              if (a.type === 'false') return b;
2565              if (b.type === 'false') return a;
2566
2567              const allTerms = [...flattenOr(a), ...flattenOr(b)];
2568              const atomMap = new Map();
2569              const otherTerms = [];
2570
2571              for (const t of allTerms) {
2572                  if (t.type === 'true') return TrueCond;
2573                  if (t.type === 'false') continue;
2574
2575                  const key = atomKey(t);
2576                  if (key) {
2577                      const negKey = negateAtomKey(key);
2578                      if (negKey && atomMap.has(negKey)) {
2579                          return TrueCond;
2580                      }
2581                      if (!atomMap.has(key)) {
2582                          atomMap.set(key, t);
2583                      }
2584                  } else {
2585                      otherTerms.push(t);
2586                  }
2587              }
2588
2589              let uniqueAtoms = Array.from(atomMap.values());
2590              uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
2591
2592              let result = [...uniqueAtoms, ...otherTerms];
2593
2594              // Поглощение: X OR (X AND Y) = X
2595              result = applyOrAbsorption(result);
2596
2597              if (result.length === 0) return FalseCond;
2598              if (result.length === 1) return result[0];
2599
2600              return buildOr(result);
2601          }
2602
2603          default:
```

```
            return c;
        }
    }

    // === Сравнение выражений ===
    function exprEq(a, b) {
        if (a === b) return true;
        if (!a && !b) return true;
        if (!a || !b) return false;
        if (a.type !== b.type) return false;

        switch (a.type) {
            case 'const': return a.n === b.n;
            case 'var': return a.name === b.name;
            case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
            case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
            default: return false;
        }
    }

    // === Упрощение выражений ===
    function simplifyExpr(expr) {
        _depth++;
        if (_depth > MAX_DEPTH) {
            _depth--;
            return expr;
        }

        try {
            return simplifyExprCore(expr);
        } finally {
            _depth--;
        }
    }

    function simplifyExprCore(expr) {
        if (!expr || expr.kind !== 'expr') return expr;

        switch (expr.type) {
            case 'const':
            case 'var':
                return expr;

            case 'op': {
                const l = simplifyExprCore(expr.l);
                const r = simplifyExprCore(expr.r);

                if (expr.op === '+') {
                    if (r?.type === 'const' && r.n === 0) return l;
                    if (l?.type === 'const' && l.n === 0) return r;
                }
                if (expr.op === '*') {
                    if (l?.type === 'const' && l.n === 0) return Const(0);
                    if (r?.type === 'const' && r.n === 0) return Const(0);
                    if (l?.type === 'const' && l.n === 1) return r;
                    if (r?.type === 'const' && r.n === 1) return l;
                }
                return Op(expr.op, l, r);
            }

            case 'when': {
                const c = simplifyCond(expr.c);
                const t = simplifyExprCore(expr.t);
                const e = simplifyExprCore(expr.e);

```

```
2669              if (c?.type === 'true') return t;
2670              if (c?.type === 'false') return e;
2671              if (exprEq(t, e)) return t;
2672              // ✅ НОВОЕ: WHEN(C, T, WHEN(NOT C, X, 0)) => WHEN(C, T, X)
2673              if (e && e.type === 'when') {
2674                  const c2 = simplifyCond(e.c);
2675                  const t2 = simplifyExprCore(e.t);
2676                  const e2 = simplifyExprCore(e.e);
2677
2678                  if (e2?.type === 'const' && e2.n === 0 && condNegationEq(c, c2)) {
2679                      return When(c, t, t2);
2680                  }
2681              }
2682              // Узкое правило: WHEN(AлB, t1, WHEN(Aл¬B, t2, WHEN(¬A, t3, e3))) -> ...
      t3
2683              if (e && e.type === 'when') {
2684                  const c2 = e.c, t2 = e.t, e2 = e.e;
2685
2686                  if (e2 && e2.type === 'when') {
2687                      const c3 = e2.c, t3 = e2.t;
2688
2689                      const shared = findSharedAndComplement(c, c2);
2690                      if (shared && condNegationEq(c3, shared.shared)) {
2691                          return When(c, t, When(c2, t2, t3));
2692                      }
2693                  }
2694              }
2695
2696              return When(c, t, e);
2697          }
2698
2699          default:
2700              return expr;
2701      }
2702  }
2703
2704  // === Печать ===
2705  function printCond(c) {
2706      if (!c) return 'TRUE';
2707
2708      switch (c.type) {
2709          case 'eq0': return `${c.v} = 0`;
2710          case 'ne0': return `${c.v} != 0`;
2711          case 'cmp': return `${c.l} ${c.op} ${c.r}`;
2712          case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
2713          case 'or':  return `(${printCond(c.a)} OR ${printCond(c.b)})`;
2714          case 'not': return `NOT(${printCond(c.x)})`;
2715          case 'true': return 'TRUE';
2716          case 'false': return 'FALSE';
2717          default: return '?';
2718      }
2719  }
2720
2721  function printExpr(e) {
2722      if (!e) return '0';
2723
2724      switch (e.type) {
2725          case 'const': return String(e.n);
2726          case 'var': return e.name;
2727          case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
2728          case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, $
      {printExpr(e.e)})`;
2729          default: return '?';
2730      }
2731  }
```

```
2732
2733  window.Optimizer = {
2734      Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
2735      Const, Var, Op, When,
2736      simplifyCond, simplifyExpr,
2737      printCond, printExpr,
2738      condEq, exprEq
2739  };
2740
2741  codegen.js
2742
2743  // js/codegen.js
2744
2745  const CodeGen = {
2746      _cache: {},
2747      _branchCache: {},
2748      _resolveCache: {},
2749      _visiting: new Set(),
2750
2751      reset() {
2752          this._cache = {};
2753          this._branchCache = {};
2754          this._resolveCache = {};
2755          this._visiting = new Set();
2756      },
2757
2758      toExpr(valueStr) {
2759          const s = String(valueStr).trim();
2760          if (s === '0') return Optimizer.Const(0);
2761          const num = parseFloat(s);
2762          if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
2763          return Optimizer.Var(s);
2764      },
2765
2766      exprToName(exprAst) {
2767          if (!exprAst) return '0';
2768          if (exprAst.type === 'var') return exprAst.name;
2769          if (exprAst.type === 'const') return String(exprAst.n);
2770          return Optimizer.printExpr(exprAst);
2771      },
2772
2773      mergeCond(a, b) {
2774          if (!a && !b) return null;
2775          if (!a) return b;
2776          if (!b) return a;
2777          if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
2778          return Optimizer.And(a, b);
2779      },
2780
2781      getConn(toId, toPort) {
2782          return AppState.connections.find(c => c.toElement === toId && c.toPort ===
      toPort);
2783      },
2784
2785      getConns(toId, prefix) {
2786          return AppState.connections.filter(c => c.toElement === toId &&
      c.toPort.startsWith(prefix));
2787      },
2788
2789      buildFormulaExpr(elem) {
2790          let result = elem.props.expression || '0';
2791
2792          // 1) Сначала раскрываем шаблоны (h и др.)
2793          const map = (typeof Settings !== 'undefined' && Settings.getTemplatesMap)
2794              ? Settings.getTemplatesMap()
```

```
2795             : null;
2796         result = expandFormulaTemplates(result, map);
2797 
2798         // 2) Потом раскрываем ссылки на формулы
2799         const formulaRefs = result.match(/formula[_-]\d+/g) || [];
2800         for (const ref of formulaRefs) {
2801             const refElem = AppState.elements[ref];
2802             if (refElem && refElem.type === 'formula') {
2803                 const refExpr = this.buildFormulaExpr(refElem);
2804                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
2805             }
2806         }
2807 
2808         return result;
2809     },
2810 
2811     // === Получить ЧИСТУЮ логику элемента ===
2812     getPureLogic(id) {
2813         const cacheKey = `logic:${id}`;
2814         if (cacheKey in this._cache) {
2815             return this._cache[cacheKey];
2816         }
2817 
2818         const elem = AppState.elements[id];
2819         if (!elem) return null;
2820 
2821         let logic = null;
2822 
2823         switch (elem.type) {
2824             case 'if': {
2825                 const leftConn = this.getConn(id, 'in-0');
2826                 const rightConn = this.getConn(id, 'in-1');
2827 
2828                 const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
     Optimizer.Const(0);
2829                 const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
     Optimizer.Const(0);
2830 
2831                 const op = (elem.props.operator || '=').trim();
2832                 const leftName = this.exprToName(leftVal);
2833                 const rightName = this.exprToName(rightVal);
2834 
2835                 const leftZero = leftVal.type === 'const' && leftVal.n === 0;
2836                 const rightZero = rightVal.type === 'const' && rightVal.n === 0;
2837 
2838                 switch (op) {
2839                     case '=':
2840                         if (rightZero) {
2841                             logic = Optimizer.Eq0(leftName);
2842                         } else if (leftZero) {
2843                             logic = Optimizer.Eq0(rightName);
2844                         } else {
2845                             logic = Optimizer.Cmp(leftName, '=', rightName);
2846                         }
2847                         break;
2848                     case '!=':
2849                         if (rightZero) {
2850                             logic = Optimizer.Ne0(leftName);
2851                         } else if (leftZero) {
2852                             logic = Optimizer.Ne0(rightName);
2853                         } else {
2854                             logic = Optimizer.Cmp(leftName, '!=', rightName);
2855                         }
2856                         break;
2857                     case '>':
```

```
2858                    case '<':
2859                    case '>=':
2860                    case '<=':
2861                        logic = Optimizer.Cmp(leftName, op, rightName);
2862                        break;
2863                    default:
2864                        logic = Optimizer.TrueCond;
2865                }
2866                break;
2867            }
2868
2869            case 'and':
2870            case 'or': {
2871                const isAnd = elem.type === 'and';
2872                const count = elem.props.inputCount || 2;
2873                let result = null;
2874
2875                for (let i = 0; i < count; i++) {
2876                    const conn = this.getConn(id, `in-${i}`);
2877                    if (!conn) continue;
2878
2879                    const val = this.getPureLogic(conn.fromElement);
2880                    if (!val) continue;
2881
2882                    if (result === null) {
2883                        result = val;
2884                    } else {
2885                        result = isAnd ? Optimizer.And(result, val) :
     Optimizer.Or(result, val);
2886                    }
2887                }
2888                logic = result || Optimizer.FalseCond;
2889                break;
2890            }
2891
2892            case 'not': {
2893                const conn = this.getConn(id, 'in-0');
2894                const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
2895                logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
2896                break;
2897            }
2898
2899            case 'separator': {
2900                const conn = this.getConn(id, 'in-0');
2901                logic = conn ? this.getPureLogic(conn.fromElement) :
     Optimizer.FalseCond;
2902                break;
2903            }
2904
2905            default:
2906                logic = null;
2907        }
2908
2909        // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
2910        if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
     elem.type === 'not') {
2911            const ctx = this.getConditionFromPort(id);
2912            if (ctx) {
2913                if (logic) {
2914                    logic = Optimizer.And(ctx, logic);
2915                } else {
2916                    logic = ctx;
2917                }
2918            }
2919        }
```

```
2920
2921          this._cache[cacheKey] = logic;
2922          return logic;
2923      },
2924
2925      // === Получить значение ===
2926      getValue(id) {
2927          const elem = AppState.elements[id];
2928          if (!elem) return Optimizer.Const(0);
2929
2930          switch (elem.type) {
2931              case 'input-signal':
2932                  // Имя сигнала или id как Var(...)
2933                  return this.toExpr(elem.props.name || id);
2934
2935              case 'const':
2936                  return Optimizer.Const(Number(elem.props.value) || 0);
2937
2938              case 'formula': {
2939                  // Используем текст формулы как выражение
2940                  const exprStr = this.buildFormulaExpr(elem) || '0';
2941                  return this.toExpr(exprStr);
2942              }
2943
2944              default:
2945                  // На всякий случай — даём символическое имя, а не 0
2946                  if (elem.props && typeof elem.props.name === 'string') {
2947                      return this.toExpr(elem.props.name);
2948                  }
2949                  return this.toExpr(id);
2950          }
2951      },
2952
2953      // === Получить ПОЛНОЕ условие для ветки сепаратора ===
2954      getBranchCondition(sepId, fromPort) {
2955          const cacheKey = `${sepId}:${fromPort}`;
2956          if (cacheKey in this._branchCache) {
2957              return this._branchCache[cacheKey];
2958          }
2959
2960          const sep = AppState.elements[sepId];
2961          if (!sep || sep.type !== 'separator') return null;
2962
2963          const inputLogic = this.getPureLogic(sepId);
2964          const sepContext = this.getConditionFromPort(sepId);
2965
2966          let branchLogic;
2967          if (fromPort === 'out-1') {
2968              branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
2969          } else {
2970              branchLogic = inputLogic || Optimizer.TrueCond;
2971          }
2972
2973          let result;
2974          if (sepContext) {
2975              result = Optimizer.And(sepContext, branchLogic);
2976          } else {
2977              result = branchLogic;
2978          }
2979
2980          this._branchCache[cacheKey] = result;
2981          return result;
2982      },
2983
2984      // === Получить условие от cond-порта ===
```

```
    getConditionFromPort(id) {
        const conn = this.getConn(id, 'cond-0');
        if (!conn) return null;

        const sourceElem = AppState.elements[conn.fromElement];
        if (!sourceElem) return null;

        if (sourceElem.type === 'separator') {
            return this.getBranchCondition(conn.fromElement, conn.fromPort);
        }

        return this.getPureLogic(conn.fromElement);
    },

    // === Основная функция разрешения ===
    resolve(id) {
        if (id in this._resolveCache) {
            return this._resolveCache[id];
        }

        if (this._visiting.has(id)) {
            return null;
        }
        this._visiting.add(id);

        const elem = AppState.elements[id];
        if (!elem) {
            this._visiting.delete(id);
            return null;
        }

        let result = null;

        try {
            switch (elem.type) {
                case 'input-signal':
                    result = {
                        isValue: true,
                        cond: null,
                        expr: this.toExpr(elem.props.name || id)
                    };
                    break;

                case 'const': {
                    const cond = this.getConditionFromPort(id);
                    result = {
                        isValue: true,
                        cond: cond,
                        expr: Optimizer.Const(Number(elem.props.value) || 0)
                    };
                    break;
                }

                case 'formula': {
                    let cond = this.getConditionFromPort(id);

                    const inConns = this.getConns(id, 'in-');
                    for (const conn of inConns) {
                        const inputNode = this.resolve(conn.fromElement);
                        if (inputNode && inputNode.cond) {
                            cond = this.mergeCond(cond, inputNode.cond);
                        }
                    }

                    const fullExpr = this.buildFormulaExpr(elem);
```

```
3050                     result = {
3051                         isValue: true,
3052                         cond: cond,
3053                         expr: Optimizer.Var(fullExpr)
3054                     };
3055                     break;
3056                 }
3057 
3058                 default:
3059                     result = null;
3060             }
3061         } finally {
3062             this._visiting.delete(id);
3063         }
3064 
3065         this._resolveCache[id] = result;
3066         return result;
3067     },
3068 
3069     generate() {
3070         console.log('=== Генерация кода (граф) ===');
3071         this.reset();
3072 
3073         try {
3074             const outputs = Object.values(AppState.elements).filter(e => e.type === 
     'output');
3075 
3076             if (outputs.length === 0) {
3077                 return '/* Нет выходов */';
3078             }
3079 
3080             const allVariants = [];
3081 
3082             for (const out of outputs) {
3083                 const conns = this.getConns(out.id, 'in-');
3084 
3085                 for (const conn of conns) {
3086                     console.log(`\n=== Обработка выхода ${out.id}, вход от $
     {conn.fromElement} ===`);
3087                     const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
3088                     const result = CodeGenGraph.evalGraphValue(graph);
3089                     console.log(`Результат: cond=${Optimizer.printCond(result.cond)}, 
     expr=${Optimizer.printExpr(result.expr)}`);
3090 
3091                     if (!result || !result.expr) continue;
3092 
3093                     const cond = result.cond ? Optimizer.simplifyCond(result.cond) : 
     null;
3094                     const isZero = result.expr.type === 'const' && result.expr.n === 
     0;
3095 
3096                     if (isZero && !cond) continue;
3097 
3098                     allVariants.push({
3099                         cond,
3100                         expr: result.expr,
3101                         isZero
3102                     });
3103                 }
3104             }
3105 
3106             console.log('Варианты:', allVariants.map(v => ({
3107                 cond: Optimizer.printCond(v.cond),
3108                 expr: Optimizer.printExpr(v.expr)
3109             })));
```

```
3110
3111              if (allVariants.length === 0) return '0';
3112
3113              const valueVariants = allVariants.filter(v => !v.isZero || v.cond);
3114              if (valueVariants.length === 0) return '0';
3115
3116              let result = Optimizer.Const(0);
3117
3118              for (let i = valueVariants.length - 1; i >= 0; i--) {
3119                  const v = valueVariants[i];
3120                  if (v.cond) {
3121                      result = Optimizer.When(v.cond, v.expr, result);
3122                  } else {
3123                      result = v.expr;
3124                  }
3125              }
3126
3127              const simplified = Optimizer.simplifyExpr(result);
3128              return Optimizer.printExpr(simplified);
3129
3130          } catch (err) {
3131              console.error('Ошибка:', err);
3132              return `/* Ошибка: ${err.message} */`;
3133          }
3134      }
3135  };
3136
3137  window.CodeGen = CodeGen;
3138
3139  config.js
3140
3141  /**
3142   * Конфигурация приложения
3143   */
3144
3145  // Типы сигналов
3146  const SIGNAL_TYPE = {
3147      NUMERIC: 'numeric',   // Числовой сигнал
3148      LOGIC: 'logic',       // Логический (может быть TRUE или FALSE)
3149      TRUE: 'true',         // Явно ИСТИНА
3150      FALSE: 'false',       // Явно ЛОЖЬ
3151      ANY: 'any'            // Любой тип
3152  };
3153
3154  // Типы проекта
3155  const PROJECT_TYPE = {
3156      PARAMETER: 'parameter',
3157      RULE: 'rule'
3158  };
3159
3160  // Конфигурация элементов
3161  const ELEMENT_TYPES = {
3162      'input-signal': {
3163          name: 'Вход',
3164          inputs: 0,
3165          outputs: 1,
3166          outputLabels: ['out'],
3167          outputTypes: [SIGNAL_TYPE.NUMERIC],
3168          color: '#4a90d9',
3169          hasProperties: true,
3170          defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
3171          resizable: true,
3172          minWidth: 150,
3173          minHeight: 50
3174      },
```

```
3175      'and': {
3176          name: 'И',
3177          inputs: 2,  // По умолчанию 2, но может быть изменено
3178          outputs: 1,
3179          inputLabels: ['A', 'B'],
3180          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3181          outputLabels: ['результат'],
3182          outputTypes: [SIGNAL_TYPE.LOGIC],
3183          color: '#a855f7',
3184          hasProperties: true,  // ← Теперь есть свойства (для изменения количества
      входов)
3185          resizable: true,
3186          minWidth: 120,
3187          minHeight: 80,
3188          hasConditionPort: true,
3189          conditionPortType: SIGNAL_TYPE.LOGIC,
3190          defaultProps: {
3191              inputCount: 2  // ← Новое свойство
3192          }
3193      },
3194      'or': {
3195          name: 'ИЛИ',
3196          inputs: 2,  // По умолчанию 2
3197          outputs: 1,
3198          inputLabels: ['A', 'B'],
3199          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
3200          outputLabels: ['результат'],
3201          outputTypes: [SIGNAL_TYPE.LOGIC],
3202          color: '#a855f7',
3203          hasProperties: true,  // ← Теперь есть свойства
3204          resizable: true,
3205          minWidth: 120,
3206          minHeight: 80,
3207          hasConditionPort: true,
3208          conditionPortType: SIGNAL_TYPE.LOGIC,
3209          defaultProps: {
3210              inputCount: 2  // ← Новое свойство
3211          }
3212      },
3213      'not': {
3214          name: 'НЕ',
3215          inputs: 1,
3216          outputs: 1,
3217          inputLabels: ['A'],
3218          inputTypes: [SIGNAL_TYPE.LOGIC],
3219          outputLabels: ['¬A'],
3220          outputTypes: [SIGNAL_TYPE.LOGIC],
3221          color: '#a855f7',
3222          hasProperties: true,
3223          resizable: true,
3224          minWidth: 100,
3225          minHeight: 60,
3226          hasConditionPort: true,
3227          conditionPortType: SIGNAL_TYPE.LOGIC
3228      },
3229      'if': {
3230          name: 'ЕСЛИ',
3231          inputs: 2,
3232          outputs: 1,  // ← Только один выход!
3233          inputLabels: ['A', 'B'],
3234          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
3235          outputLabels: ['результат'],  // ← Просто результат
3236          outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
3237          color: '#e94560',
3238          hasProperties: true,
```

```
3239          defaultProps: { operator: '=' },
3240          resizable: true,
3241          minWidth: 120,
3242          minHeight: 80,
3243          hasConditionPort: true,
3244          conditionPortType: SIGNAL_TYPE.LOGIC
3245      },
3246      'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
3247          name: 'Сепаратор',
3248          inputs: 1,
3249          outputs: 2,
3250          inputLabels: ['сигнал'],
3251          inputTypes: [SIGNAL_TYPE.LOGIC],
3252          outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
3253          outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
3254          color: '#f59e0b',
3255          hasProperties: true,
3256          resizable: true,
3257          minWidth: 120,
3258          minHeight: 80,
3259          hasConditionPort: true,
3260          conditionPortType: SIGNAL_TYPE.LOGIC
3261      },
3262      'const': {
3263          name: 'Константа',
3264          inputs: 0,
3265          outputs: 1,
3266          outputLabels: ['out'],
3267          outputTypes: [SIGNAL_TYPE.NUMERIC],
3268          color: '#3b82f6',
3269          hasProperties: true,
3270          defaultProps: { value: 0 },
3271          resizable: true,
3272          minWidth: 120,
3273          minHeight: 60,
3274          hasConditionPort: true,
3275          conditionPortType: SIGNAL_TYPE.LOGIC
3276      },
3277      'formula': {
3278          name: 'Формула',
3279          inputs: 2,
3280          outputs: 1,
3281          inputLabels: ['in₁', 'in₂'],
3282          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
3283          outputLabels: ['результат'],
3284          outputTypes: [SIGNAL_TYPE.NUMERIC],
3285          color: '#f59e0b',
3286          hasProperties: true,
3287          resizable: true,
3288          minWidth: 140,
3289          minHeight: 80,
3290          defaultProps: {
3291              expression: '',
3292              inputCount: 2
3293          },
3294          hasConditionPort: true,
3295          conditionPortType: SIGNAL_TYPE.LOGIC
3296      },
3297      'output': {
3298          name: 'Выход',
3299          inputs: 1,
3300          outputs: 0,
3301          inputLabels: ['сигнал'],
3302          inputTypes: [SIGNAL_TYPE.ANY],
3303          color: '#10b981',
```

```
        hasProperties: true,
        defaultProps: { label: 'Выход', outputGroup: '' },
        resizable: true,
        minWidth: 150,
        minHeight: 60,
    },  // ← важно, если предыдущий элемент не заканчивается запятой
    'group': {
        name: 'Группа',
        inputs: 0,
        outputs: 0,
        color: '#6b7280',
        resizable: true,
        minWidth: 200,
        minHeight: 120,
        hasProperties: true,
        defaultProps: { title: 'Группа' }
    }
};

const VIEWPORT_CONFIG = {
    minZoom: 0.1,
    maxZoom: 3,
    zoomStep: 0.1,
    panSpeed: 1,
    canvasWidth: 5000,
    canvasHeight: 5000
};

const MINIMAP_CONFIG = {
    width: 200,
    height: 150,
    padding: 10
};

connections.js

/**
 * Модуль работы с соединениями
 */

const Connections = {
    /**
     * Настройка обработчиков порта
     */
    setupPortHandlers(port) {
        port.addEventListener('mousedown', (e) => {
            e.stopPropagation();

            if (port.classList.contains('output')) {
                const elemId = port.dataset.element;
                const portName = port.dataset.port;
                const signalType = getOutputPortType(elemId, portName);

                AppState.connectingFrom = {
                    element: elemId,
                    port: portName
                };
                AppState.connectingFromType = signalType;

                this.highlightCompatiblePorts(signalType);

                const svg = document.getElementById('connections-svg');
                const startPos = this._getPortCanvasCenter(port);

                AppState.tempLine = document.createElementNS('http://www.w3.org/2000/
```

```
      svg', 'path');
3369                  AppState.tempLine.setAttribute('class', 'temp-connection');
3370                  AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L $
      {startPos.x} ${startPos.y}`);
3371                  svg.appendChild(AppState.tempLine);
3372              }
3373          });
3374
3375          port.addEventListener('mouseup', (e) => {
3376              e.stopPropagation();
3377              e.preventDefault();
3378
3379              if (AppState.connectingFrom && port.classList.contains('input')) {
3380                  const toElement = port.dataset.element;
3381                  const toPortName = port.dataset.port;
3382                  const inputType = getInputPortType(toElement, toPortName);
3383
3384                  if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
3385                      this.clearConnectionState();
3386                      return;
3387                  }
3388
3389                  if (AppState.connectingFrom.element !== toElement) {
3390                      const targetElem = AppState.elements[toElement];
3391                      const allowMultipleInputs = targetElem?.type === 'output';
3392
3393                      const exists = AppState.connections.some(c =>
3394                          c.toElement === toElement && c.toPort === toPortName
3395                      );
3396
3397                      if (!exists || allowMultipleInputs) {
3398                          AppState.connections.push({
3399                              fromElement: AppState.connectingFrom.element,
3400                              fromPort: AppState.connectingFrom.port,
3401                              toElement,
3402                              toPort: toPortName,
3403                              signalType: AppState.connectingFromType
3404                          });
3405
3406                          port.classList.add('connected');
3407                          this.drawConnections();
3408                          this.clearConnectionState();
3409                          return;
3410                      }
3411                  }
3412              }
3413
3414              this.clearConnectionState();
3415          });
3416
3417          port.addEventListener('mouseenter', () => {
3418              if (AppState.connectingFrom && port.classList.contains('input')) {
3419                  const toPortName = port.dataset.port;
3420                  const inputType = getInputPortType(port.dataset.element, toPortName);
3421
3422                  if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
3423                      if (AppState.tempLine) {
3424                          AppState.tempLine.classList.add('invalid');
3425                      }
3426                  }
3427              }
3428          });
3429
3430          port.addEventListener('mouseleave', () => {
3431              if (AppState.tempLine) {
```

```
3432                 AppState.tempLine.classList.remove('invalid');
3433             }
3434         });
3435     },
3436
3437     /**
3438      * Подсветка совместимых портов
3439      */
3440     highlightCompatiblePorts(signalType) {
3441         document.querySelectorAll('.port.input').forEach(port => {
3442             const inputType = getInputPortType(port.dataset.element,
      port.dataset.port);
3443
3444             if (areTypesCompatible(signalType, inputType)) {
3445                 port.classList.add('compatible-highlight');
3446             } else {
3447                 port.classList.add('incompatible');
3448             }
3449         });
3450     },
3451
3452     /**
3453      * Очистка состояния соединения
3454      */
3455     clearConnectionState() {
3456         if (AppState.tempLine) {
3457             AppState.tempLine.remove();
3458             AppState.tempLine = null;
3459         }
3460         AppState.connectingFrom = null;
3461         AppState.connectingFromType = null;
3462
3463         document.querySelectorAll('.port').forEach(port => {
3464             port.classList.remove('compatible-highlight', 'incompatible');
3465         });
3466     },
3467
3468     /**
3469      * Отрисовка временной линии соединения
3470      */
3471     drawTempConnection(e) {
3472     if (!AppState.tempLine || !AppState.connectingFrom) return;
3473
3474     const fromElem = document.getElementById(AppState.connectingFrom.element);
3475     if (!fromElem) return;
3476
3477     const fromPort = fromElem.querySelector(`[data-port="$
      {AppState.connectingFrom.port}"]`);
3478     if (!fromPort) return;
3479
3480     const startPos = this._getPortCanvasCenter(fromPort);
3481     const endPos = screenToCanvas(e.clientX, e.clientY);
3482
3483     const horizontalDist = Math.abs(endPos.x - startPos.x);
3484     const controlDist = Math.max(horizontalDist * 0.4, 50);
3485
3486     // Тянем всегда от выхода (вектор 1, 0)
3487     const cx1 = startPos.x + controlDist;
3488     const cy1 = startPos.y;
3489
3490     // Вторая точка контроля для плавности за курсором
3491     const cx2 = endPos.x - controlDist;
3492     const cy2 = endPos.y;
3493
3494     AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} $
```

```
      {cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
3495      AppState.tempLine.setAttribute('fill', 'none');
3496  },
3497
3498      /**
3499       * Отрисовка всех соединений
3500       */
3501      drawConnections() {
3502      const svg = document.getElementById('connections-svg');
3503
3504      // 1. Очистка старых линий
3505      svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
3506
3507      // 2. Сброс визуального состояния портов
3508      document.querySelectorAll('.port.connected').forEach(port => {
3509          port.classList.remove('connected');
3510      });
3511
3512      // 3. Перебор всех соединений из AppState
3513      AppState.connections.forEach(conn => {
3514          const fromElem = document.getElementById(conn.fromElement);
3515          const toElem = document.getElementById(conn.toElement);
3516
3517          if (!fromElem || !toElem) return;
3518
3519          const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
3520          const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
3521
3522          if (!fromPort || !toPort) return;
3523
3524          fromPort.classList.add('connected');
3525          toPort.classList.add('connected');
3526
3527          const startPos = this._getPortCanvasCenter(fromPort);
3528          const endPos = this._getPortCanvasCenter(toPort);
3529
3530          if (!startPos || !endPos) return;
3531
3532          // Расстояние для изгиба кривой
3533          const horizontalDist = Math.abs(endPos.x - startPos.x);
3534          const verticalDist = Math.abs(endPos.y - startPos.y);
3535          const controlDist = Math.max(horizontalDist * 0.4, 50);
3536
3537          // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
3538          let d;
3539          let cx1 = startPos.x;
3540          let cy1 = startPos.y;
3541          let cx2 = endPos.x;
3542          let cy2 = endPos.y;
3543
3544          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
3545          cx1 = startPos.x + controlDist;
3546          cy1 = startPos.y;
3547
3548          // ВХОД (Target):
3549          if (conn.toPort === 'cond-0') {
3550              // Технический порт: Касательная (0, 1) в декартовой (вверх)
3551              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
3552              cx2 = endPos.x;
3553              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
3554          } else {
3555              // Обычный вход: Касательная (-1, 0) -> Слева направо
3556              cx2 = endPos.x - controlDist;
3557              cy2 = endPos.y;
3558          }
```

```
3559
3560          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
      ${endPos.y}`;
3561
3562          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
3563          path.setAttribute('d', d);
3564          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
3565
3566          // --- ЛОГИКА ЦВЕТА (Классы) ---
3567          let cssClass = 'connection';
3568          const type = conn.signalType;
3569
3570          // Приоритет новым типам сигналов
3571          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
3572          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
3573          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
3574          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
3575          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
3576
3577          path.setAttribute('class', cssClass);
3578
3579          // Обработчики событий
3580          path.style.pointerEvents = 'stroke';
3581          path.style.cursor = 'pointer';
3582          path.addEventListener('click', () => this.handleConnectionClick(conn));
3583
3584          svg.appendChild(path);
3585      });
3586
3587      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
3588          Outputs.updateOutputStatus();
3589      }
3590      Viewport.updateMinimap();
3591  },
3592      /**
3593       * Обработка клика по соединению (удаление)
3594       */
3595      handleConnectionClick(conn) {
3596          if (confirm('Удалить соединение?')) {
3597              AppState.connections = AppState.connections.filter(c =>
3598                  !(c.fromElement === conn.fromElement &&
3599                    c.fromPort === conn.fromPort &&
3600                    c.toElement === conn.toElement &&
3601                    c.toPort === conn.toPort)
3602              );
3603
3604              this.drawConnections();
3605          }
3606      },
3607
3608      /**
3609       * Получение центра порта в координатах Canvas
3610       */
3611      _getPortCanvasCenter(portEl) {
3612          if (!portEl) return null;
3613
3614          const rect = portEl.getBoundingClientRect();
3615          return screenToCanvas(
3616              rect.left + rect.width / 2,
3617              rect.top + rect.height / 2
3618          );
3619      }
3620  };
3621
3622  elements.js
```

```
3623
3624  /**
3625   * Модуль работы с элементами схемы
3626   */
3627
3628  const Elements = {
3629      /**
3630       * Генерация HTML для элемента
3631       */
3632          createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
3633              const config = ELEMENT_TYPES[elemType];
3634              if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
3635
3636              const safe = (value, fallback = '') => (value === null || value ===
      undefined) ? fallback : String(value);
3637              const w = width ?? config.minWidth ?? 120;
3638              const h = height ?? config.minHeight ?? 60;
3639
3640              const getPortClass = (signalType, direction) => {
3641                  const base = direction === 'output' ? 'port output' : 'port input';
3642                  if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
3643                  if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
3644                  return `${base} any-port`;
3645              };
3646
3647              // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
      innerHTML
3648              // Она тут остается, но ее результат не встраивается в HTML-строку
      напрямую, кроме формулы
3649              const buildConditionPortHTML = () => {
3650                  return `
3651                      <div class="condition-port-wrapper">
3652                          <div class="condition-port-label">условие</div>
3653                          <div class="port input condition-port"
3654                              data-port="cond-0"
3655                              data-element="${elemId}"
3656                              data-signal-type="${SIGNAL_TYPE.LOGIC}"
3657                              title="Техническое условие">
3658                          </div>
3659                      </div>`;
3660              };
3661
3662
3663              const buildInputPorts = (count, types = [], labels = []) => {
3664                  let html = '';
3665                  for (let i = 0; i < count; i++) {
3666                      const type = types[i] ?? types[types.length - 1] ??
      SIGNAL_TYPE.ANY;
3667                      html += `<div class="${getPortClass(type, 'input')}" data-
      port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Вход ${i+1}`}"></div>`;
3668                  }
3669                  return html;
3670              };
3671
3672              const buildOutputPorts = (count, types = [], labels = []) => {
3673                  let html = '';
3674                  for (let i = 0; i < count; i++) {
3675                      const type = types[i] ?? types[types.length - 1] ??
      SIGNAL_TYPE.ANY;
3676                      html += `<div class="${getPortClass(type, 'output')}" data-
      port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Выход ${i+1}`}"></div>`;
3677                  }
3678                  return html;
```

```
3679             };
3680
3681             const resizeHandles = config.resizable ? `<div class="resize-handle
     handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
     direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
     '';
3682             // hasCondClass будет добавляться в addElement
3683             // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
     '';
3684
3685             let innerHTML = '';
3686
3687             if (elemType === 'input-signal') {
3688                 const name = safe(props.name, 'Сигнал');
3689                 const type = props.signalType || SIGNAL_TYPE.NUMBER;
3690                 const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
3691                 innerHTML = `
3692                     <div class="element-header" style="background:$
     {config.color};">Источник</div>
3693                     <div class="element-body">
3694                         <div class="element-symbol">
3695                             <span class="input-signal-icon">${symbol}</span>
3696                             <span class="input-signal-name">${name}</span>
3697                         </div>
3698                         <div class="ports-right">
3699                             ${buildOutputPorts(1, [type], ['Выход'])}
3700                         </div>
3701                     </div>`;
3702             }
3703             else if (elemType === 'const') {
3704                 innerHTML = `
3705                     <div class="element-header" style="background:$
     {config.color};">Константа</div>
3706                     <div class="element-body">
3707                         <div class="element-symbol">${props.value ?? 0}</div>
3708                         <div class="ports-right">
3709                             ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
3710                         </div>
3711                     </div>`;
3712             }
3713             else if (elemType === 'separator') {
3714                 innerHTML = `
3715                     <div class="element-header" style="background:$
     {config.color};">Сепаратор</div>
3716                     <div class="element-body">
3717                         <div class="ports-left">${buildInputPorts(1,
     config.inputTypes, config.inputLabels)}</div>
3718                         <div class="element-symbol">✓/✗</div>
3719                         <div class="ports-right">
3720                             <div class="port output logic-port true-port" data-
     port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
     title="ИСТИНА"></div>
3721                             <div class="port output logic-port false-port" data-
     port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
     title="ЛОЖЬ"></div>
3722                         </div>
3723                     </div>`;
3724             }
3725             else if (elemType === 'and' || elemType === 'or') {
3726                 const gateSymbol = elemType === 'and' ? '∧' : '∨';
3727                 const inputCount = props.inputCount || config.defaultProps?.inputCount
     || 2;
3728
3729                 // Генерируем динамические входы
3730                 let inputsHTML = '';
```

```
3731                 for (let i = 0; i < inputCount; i++) {
3732                     inputsHTML += `<div class="port input logic-port" data-port="in-${i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход ${i+1}"></div>`;
3733                 }
3734 
3735                 innerHTML = `
3736                     <div class="element-header" style="background:${config.color};">${config.name}</div>
3737                     <div class="element-body">
3738                         <div class="ports-left">
3739                             ${inputsHTML}
3740                         </div>
3741                         <div class="element-symbol">${gateSymbol}</div>
3742                         <div class="ports-right">
3743                             <div class="port output logic-port" data-port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></div>
3744                         </div>
3745                     </div>`;
3746             }
3747             else if (elemType === 'if') {
3748                 const op = safe(props.operator, '=');
3749                 innerHTML = `
3750                     <div class="element-header" style="background:${config.color};">Условие</div>
3751                     <div class="element-body">
3752                         <div class="ports-left">${buildInputPorts(2, config.inputTypes, config.inputLabels)}</div>
3753                         <div class="element-symbol">${op}</div>
3754                         <div class="ports-right">
3755                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
3756                         </div>
3757                     </div>`;
3758             }
3759             else if (elemType === 'not') {
3760                 innerHTML = `
3761                     <div class="element-header" style="background:${config.color};">НЕ</div>
3762                     <div class="element-body">
3763                         <div class="ports-left">${buildInputPorts(1, [SIGNAL_TYPE.LOGIC], ['A'])}</div>
3764                         <div class="element-symbol">¬</div>
3765                         <div class="ports-right">
3766                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
3767                         </div>
3768                     </div>`;
3769             }
3770             else if (elemType === 'formula') {
3771                 const inputCount = props.inputCount || config.defaultProps?.inputCount || config.inputs || 2;
3772                 const expression = safe(props.expression);
3773                 const displayExpression = expression
3774                     ? (expression.length > 12 ? `${expression.slice(0, 12)}…` : expression)
3775                     : 'f(x)';
3776 
3777                 innerHTML = `
3778                     ${buildConditionPortHTML()}
3779                     <div class="element-header" style="background:${config.color};">Формула</div>
3780                     <div class="element-body">
3781                         <div class="ports-left">${buildInputPorts(inputCount, config.inputTypes, config.inputLabels)}</div>
3782                         <div class="element-symbol">${displayExpression}</div>
```

```
3783                         <div class="ports-right">
3784                             ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER],
      ['Результат'])}
3785                         </div>
3786                     </div>`;
3787             }
3788             else if (elemType === 'output') {
3789                 innerHTML = `
3790                     <div class="element-header" style="background:$
      {config.color};">Выход</div>
3791                     <div class="element-body">
3792                         <div class="ports-left">
3793                             ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
3794                         </div>
3795                         <div class="element-symbol">${safe(props.label, 'Выход')}</
      div>
3796                         <div class="ports-right"></div>
3797                     </div>`;
3798 
3799             }
3800             else if (elemType === 'group') {
3801                 const title = props.title || 'Группа';
3802                 innerHTML = `
3803                     <div class="group-content">
3804                     <div class="group-title">${title}</div>
3805                     </div>`;
3806             }
3807 
3808             else { // Для любых других (fallback)
3809                 innerHTML = `
3810                     <div class="element-header" style="background:${config.color};">$
      {config.name}</div>
3811                     <div class="element-body">
3812                         <div class="ports-left">${buildInputPorts(config.inputs || 0,
      config.inputTypes, config.inputLabels)}</div>
3813                         <div class="element-symbol">${config.name}</div>
3814                         <div class="ports-right">
3815                             ${buildOutputPorts(config.outputs || 0,
      config.outputTypes, config.outputLabels)}
3816                         </div>
3817                     </div>`;
3818             }
3819             const commentHtml = `<div class="element-comment">${safe(props.comment,
      '')}</div>`;
3820 
3821 
3822             const html = `
3823                 <div class="element ${elemType}" id="${elemId}"
3824                     style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;"
      data-type="${elemType}">
3825                     ${innerHTML}
3826                     ${commentHtml}
3827                     ${resizeHandles}
3828                 </div>`;
3829 
3830             return { html, width: w, height: h };
3831         },
3832 
3833     /**
3834      * Добавление элемента
3835      */
3836         addElement(elemType, x, y, props = {}, elemId = null, customWidth = null,
      customHeight = null) {
3837             const config = ELEMENT_TYPES[elemType];
3838             if (!config) {
```

```
3839                 console.error(`Неизвестный тип элемента: ${elemType}`);
3840                 return null;
3841             }
3842 
3843             if (!elemId) {
3844                 elemId = `${elemType}_${++AppState.elementCounter}`;
3845             }
3846 
3847             let width = customWidth;
3848             let height = customHeight;
3849 
3850             if (width === null || width === undefined) {
3851                 width = config.minWidth || 140;
3852             }
3853             if (height === null || height === undefined) {
3854                 height = config.minHeight || 70;
3855             }
3856 
3857             try {
3858                 const result = this.createElementHTML(elemType, elemId, x, y, props,
      width, height);
3859                 if (!result || !result.html) {
3860                     console.error('createElementHTML вернул пустой результат');
3861                     return null;
3862                 }
3863 
3864                 const workspace = document.getElementById('workspace');
3865                 const wrapper = document.createElement('div');
3866                 wrapper.innerHTML = result.html.trim();
3867                 const element = wrapper.firstElementChild;
3868                 if (!element) {
3869                     console.error('Не удалось создать DOM элемент из HTML');
3870                     return null;
3871                 }
3872 
3873                 // Добавляем класс для отступа
3874                 if (config.hasConditionPort) {
3875                     element.classList.add('has-condition-port');
3876                 }
3877 
3878                 workspace.appendChild(element);
3879 
3880                 AppState.elements[elemId] = {
3881                     id: elemId,
3882                     type: elemType,
3883                     x,
3884                     y,
3885                     width: result.width || width,
3886                     height: result.height || height,
3887                     props: { ...(config.defaultProps || {}), ...(props || {}) }
3888                 };
3889 
3890                 // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ
      ЕГО В HTML)
3891                 if (config.hasConditionPort && elemType !== 'formula') {
3892                     const condPortWrapper = document.createElement('div');
3893                     condPortWrapper.innerHTML = `
3894                         <div class="condition-port-wrapper">
3895                             <div class="condition-port-label">условие</div>
3896                             <div class="port input condition-port"
3897                                 data-port="cond-0"
3898                                 data-element="${elemId}"
3899                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"
3900                                 title="Техническое условие">
3901                             </div>
```

```
                        </div>`;
                    element.prepend(condPortWrapper.firstElementChild); // Вставляем в самое начало элемента
                }


                this.setupElementHandlers(elemId); // Передаем ID элемента

                // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
                // element.querySelectorAll('.port').forEach(port => {
                //     Connections.setupPortHandlers(port);
                // });

                Connections.drawConnections(); // Перерисовываем соединения, чтобы учесть новые порты
                Viewport.updateMinimap();
                return elemId;
            } catch (err) {
                console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
                return null;
            }
        },

    /**
     * Обновление входов логического элемента (AND, OR)
     */
    updateLogicGateInputs(elemId, inputCount) {
        const elem = document.getElementById(elemId);
        if (!elem) return;

        const portsLeft = elem.querySelector('.ports-left');
        if (!portsLeft) return;

        // Удаляем соединения к портам, которые больше не существуют
        AppState.connections = AppState.connections.filter(c => {
            if (c.toElement === elemId && c.toPort.startsWith('in-')) {
                const portNum = parseInt(c.toPort.split('-')[1], 10);
                return portNum < inputCount;
            }
            return true;
        });

        // Генерируем новые входы
        let inputsHTML = '';
        for (let i = 0; i < inputCount; i++) {
            inputsHTML += `
                <div class="port input logic-port"
                    data-port="in-${i}"
                    data-element="${elemId}"
                    data-signal-type="${SIGNAL_TYPE.LOGIC}"
                    title="Вход ${i+1}">
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

        // Переподключаем обработчики
        portsLeft.querySelectorAll('.port').forEach(port =>
            Connections.setupPortHandlers(port)
        );

        Connections.drawConnections();
    },

    /**
```

```
3965        * Удаление элемента
3966        */
3967       deleteElement(elemId) {
3968           AppState.connections = AppState.connections.filter(c =>
3969               c.fromElement !== elemId && c.toElement !== elemId
3970           );
3971
3972           const elem = document.getElementById(elemId);
3973           if (elem) elem.remove();
3974
3975           delete AppState.elements[elemId];
3976
3977           if (AppState.selectedElement === elemId) {
3978               AppState.selectedElement = null;
3979           }
3980
3981           Connections.drawConnections();
3982           Viewport.updateMinimap();
3983       },
3984
3985       /**
3986        * Выделение элемента
3987        */
3988       selectElement(elemId) {
3989           if (AppState.selectedElement) {
3990               const oldElem = document.getElementById(AppState.selectedElement);
3991               if (oldElem) oldElem.classList.remove('selected');
3992           }
3993
3994           AppState.selectedElement = elemId;
3995           const elem = document.getElementById(elemId);
3996           if (elem) elem.classList.add('selected');
3997
3998           const elemData = AppState.elements[elemId];
3999           if (elemData) {
4000               document.getElementById('selection-info').textContent =
4001                   `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
4002           }
4003       },
4004
4005       /**
4006        * Снять выделение
4007        */
4008       deselectAll() {
4009           if (AppState.selectedElement) {
4010               const elem = document.getElementById(AppState.selectedElement);
4011               if (elem) elem.classList.remove('selected');
4012               AppState.selectedElement = null;
4013           }
4014           document.getElementById('selection-info').textContent = '';
4015       },
4016
4017       /**
4018        * Настройка обработчиков элемента
4019        */
4020       setupElementHandlers(elemId) {
4021           try {
4022               const elem = document.getElementById(elemId);
4023               if (!elem) return;
4024
4025               elem.addEventListener('mousedown', (e) => {
4026                   if (e.target.classList.contains('port')) return;
4027                   if (e.target.classList.contains('resize-handle')) return;
4028
4029                   e.preventDefault();
```

```
                e.stopPropagation();

                this.selectElement(elemId);

                AppState.draggingElement = elemId;
                const canvasPos = screenToCanvas(e.clientX, e.clientY);
                const elemData = AppState.elements[elemId];
                AppState.dragOffset.x = canvasPos.x - elemData.x;
                AppState.dragOffset.y = canvasPos.y - elemData.y;
            });

            elem.addEventListener('dblclick', (e) => {
                if (e.target.classList.contains('port')) return;
                const config = ELEMENT_TYPES[AppState.elements[elemId].type];
                if (config?.hasProperties) {
                    Modal.showPropertiesModal(elemId);
                }
            });

            elem.addEventListener('contextmenu', (e) => {
                e.preventDefault();
                this.showContextMenu(e.clientX, e.clientY, elemId);
            });

            const handles = elem.querySelectorAll('.resize-handle');
            handles.forEach(handle => this.setupResizeHandlers(handle, elemId));

            const ports = elem.querySelectorAll('.port');
            ports.forEach(port => Connections.setupPortHandlers(port));

        } catch (err) {
            console.error('setupElementHandlers error for', elemId, err);
        }
    },

    /**
     * Контекстное меню
     */
    showContextMenu(x, y, elemId) {
        const menu = document.getElementById('context-menu');
        menu.style.left = `${x}px`;
        menu.style.top = `${y}px`;
        menu.style.display = 'block';
        menu.dataset.elementId = elemId;
    },

    /**
     * Настройка resize
     */
    setupResizeHandlers(handle, elemId) {
        handle.addEventListener('mousedown', (e) => {
            e.stopPropagation();
            e.preventDefault();

            const elemData = AppState.elements[elemId];

            AppState.resizing = {
                elemId: elemId,
                handle: handle.dataset.direction,
                startX: e.clientX,
                startY: e.clientY,
                startWidth: elemData.width,
                startHeight: elemData.height,
                startLeft: elemData.x,
                startTop: elemData.y
```

```
4095              };
4096          });
4097      },
4098
4099      /**
4100       * Обработка resize
4101       */
4102      handleResize(e) {
4103          if (!AppState.resizing) return;
4104
4105          const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
      startTop } = AppState.resizing;
4106          const elem = document.getElementById(elemId);
4107          const elemData = AppState.elements[elemId];
4108          const config = ELEMENT_TYPES[elemData.type];
4109
4110          const dx = (e.clientX - startX) / AppState.viewport.zoom;
4111          const dy = (e.clientY - startY) / AppState.viewport.zoom;
4112
4113          let newWidth = startWidth;
4114          let newHeight = startHeight;
4115          let newLeft = startLeft;
4116          let newTop = startTop;
4117
4118          if (handle.includes('e')) {
4119              newWidth = Math.max(config.minWidth, startWidth + dx);
4120          }
4121          if (handle.includes('w')) {
4122              newWidth = Math.max(config.minWidth, startWidth - dx);
4123              newLeft = startLeft + (startWidth - newWidth);
4124          }
4125          if (handle.includes('s')) {
4126              newHeight = Math.max(config.minHeight, startHeight + dy);
4127          }
4128          if (handle.includes('n')) {
4129              newHeight = Math.max(config.minHeight, startHeight - dy);
4130              newTop = startTop + (startHeight - newHeight);
4131          }
4132
4133          elem.style.width = `${newWidth}px`;
4134          elem.style.height = `${newHeight}px`;
4135          elem.style.left = `${newLeft}px`;
4136          elem.style.top = `${newTop}px`;
4137
4138          elemData.width = newWidth;
4139          elemData.height = newHeight;
4140          elemData.x = newLeft;
4141          elemData.y = newTop;
4142
4143          Connections.drawConnections();
4144      },
4145
4146      /**
4147       * Обработка перетаскивания элемента
4148       */
4149      handleDrag(e) {
4150          if (!AppState.draggingElement) return;
4151
4152          const canvasPos = screenToCanvas(e.clientX, e.clientY);
4153          const elemId = AppState.draggingElement;
4154          const elemData = AppState.elements[elemId];
4155          if (!elemData) return;
4156
4157          const newX = canvasPos.x - AppState.dragOffset.x;
4158          const newY = canvasPos.y - AppState.dragOffset.y;
```

```
4159          const dx = newX - elemData.x;
4160          const dy = newY - elemData.y;
4161
4162          // если выделено несколько
4163          const group = AppState.selectedElements && AppState.selectedElements.length >
      1
4164              ? AppState.selectedElements
4165              : [elemId];
4166
4167          for (const id of group) {
4168              const elData = AppState.elements[id];
4169              if (!elData) continue;
4170              elData.x += dx;
4171              elData.y += dy;
4172              const el = document.getElementById(id);
4173              if (el) {
4174              el.style.left = elData.x + 'px';
4175              el.style.top = elData.y + 'px';
4176              }
4177          }
4178
4179          Connections.drawConnections();
4180      },
4181
4182      /**
4183       * Обновление входов формулы
4184       */
4185      updateFormulaInputs(elemId, inputCount) {
4186          const elem = document.getElementById(elemId);
4187          if (!elem) return;
4188
4189          const portsLeft = elem.querySelector('.ports-left');
4190          if (!portsLeft) return;
4191
4192          AppState.connections = AppState.connections.filter(c => {
4193              if (c.toElement === elemId && c.toPort.startsWith('in-')) {
4194                  const portNum = parseInt(c.toPort.split('-')[1], 10);
4195                  return portNum < inputCount;
4196              }
4197              return true;
4198          });
4199
4200          let inputsHTML = '';
4201          for (let i = 0; i < inputCount; i++) {
4202              inputsHTML += `
4203                  <div class="port input any-port"
4204                       data-port="in-${i}"
4205                       data-element="${elemId}"
4206                       data-signal-type="${SIGNAL_TYPE.ANY}"
4207                       title="in${i} (Любой)">
4208                  </div>
4209              `;
4210          }
4211          portsLeft.innerHTML = inputsHTML;
4212
4213          portsLeft.querySelectorAll('.port').forEach(port =>
4214              Connections.setupPortHandlers(port)
4215          );
4216
4217          Connections.drawConnections();
4218      },
4219
4220      /**
4221       * Рассчитать оптимальный размер элемента на основе количества портов
4222       */
```

```
4223      calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
4224          const elem = AppState.elements[elemId];
4225          if (!elem) return null;
4226
4227          const config = ELEMENT_TYPES[elem.type];
4228          if (!config || !config.resizable) return null;
4229
4230          // Базовая высота
4231          let baseHeight = config.minHeight || 60;
4232
4233          // Каждый порт требует примерно 25-30px высоты
4234          const portSpacing = 28;
4235          const maxPorts = Math.max(inputCount, outputCount);
4236
4237          // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
4238          const additionalHeight = (maxPorts - 1) * portSpacing;
4239          const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
4240
4241          return newHeight;
4242      },
4243
4244      /**
4245       * Обновление размера элемента при изменении портов
4246       */
4247      updateElementSize(elemId) {
4248          const elem = document.getElementById(elemId);
4249          const elemData = AppState.elements[elemId];
4250
4251          if (!elem || !elemData) return;
4252
4253          const config = ELEMENT_TYPES[elemData.type];
4254          if (!config || !config.resizable) return;
4255
4256          let inputCount = 0;
4257          let outputCount = config.outputs || 1;
4258
4259          // Определяем количество входов
4260          if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
      'formula') {
4261              inputCount = elemData.props.inputCount || config.inputs || 2;
4262          } else {
4263              inputCount = config.inputs || 0;
4264          }
4265
4266          // Рассчитываем новую высоту
4267          const newHeight = this.calculateOptimalHeight(elemId, inputCount,
      outputCount);
4268
4269          if (newHeight && newHeight !== elemData.height) {
4270              elemData.height = newHeight;
4271              elem.style.height = `${newHeight}px`;
4272
4273              // Перерисовываем соединения, т.к. изменился размер элемента
4274              Connections.drawConnections();
4275              Viewport.updateMinimap();
4276          }
4277      }
4278
4279
4280  };
4281
4282  modal.js
4283
4284  /**
4285   * Модуль модальных окон
```

```
 */

const Modal = {
    /**
     * Инициализация модальных окон
     */
    init() {
        // Модальное окно свойств элемента
        document.getElementById('modal-save').addEventListener('click', () => {
            this.saveElementProperties();
        });

        document.getElementById('modal-cancel').addEventListener('click', () => {
            this.hideModal('modal-overlay');
        });

        document.getElementById('modal-overlay').addEventListener('click', (e) => {
            if (e.target.id === 'modal-overlay') {
                this.hideModal('modal-overlay');
            }
        });

        // Модальное окно свойств проекта
        document.getElementById('project-modal-save').addEventListener('click', () =>
{
            this.saveProjectProperties();
        });

        document.getElementById('project-modal-cancel').addEventListener('click', ()
=> {
            this.hideModal('project-modal-overlay');
        });

        document.getElementById('project-modal-overlay').addEventListener('click', (e)
=> {
            if (e.target.id === 'project-modal-overlay') {
                this.hideModal('project-modal-overlay');
            }
        });
    },

    /**
     * Показать модальное окно
     */
    showModal(modalId) {
        document.getElementById(modalId).style.display = 'flex';
    },

    /**
     * Скрыть модальное окно
     */
    hideModal(modalId) {
        document.getElementById(modalId).style.display = 'none';
    },

    /**
     * Показать свойства элемента
     */
    showPropertiesModal(elemId) {
        const elemData = AppState.elements[elemId];
        const elemType = elemData.type;
        const props = elemData.props;
        const config = ELEMENT_TYPES[elemType];

        const modalOverlay = document.getElementById('modal-overlay');
```

```
4348          const modalTitle = document.getElementById('modal-title');
4349          const modalContent = document.getElementById('modal-content');
4350
4351          modalTitle.textContent = `Свойства: ${config.name}`;
4352
4353          let contentHTML = '';
4354
4355   if (elemType === 'input-signal') {
4356    const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
4357
4358    contentHTML = `
4359      <div class="modal-row">
4360        <label>Название сигнала:</label>
4361        <input type="text" id="prop-name" value="${props.name || ''}"
     placeholder="Например: 10LBA..." />
4362        <small style="color:#999;">
4363          Поиск по маске через * (например: *MAA*CP*)
4364        </small>
4365        <div id="signal-filter-results"
4366             style="max-height:160px; overflow-y:auto; background:#0f3460; border-
     radius:5px; margin-top:6px; display:none;">
4367        </div>
4368      </div>
4369
4370      <div class="modal-row">
4371        <label>Описание сигнала:</label>
4372        <textarea id="prop-description" readonly>${props.description || ''}</textarea>
4373      </div>
4374
4375      <div class="modal-row">
4376        <label>Тип сигнала:</label>
4377        <select id="prop-signal-type">
4378          <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
     'selected' : ''}>Числовой</option>
4379          <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
     'selected' : ''}>Логический</option>
4380        </select>
4381      </div>
4382    `;
4383
4384    // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
     DOM.
4385    // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
4386    // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
4387  } else if (elemType === 'if') {
4388              contentHTML = `
4389                  <div class="modal-row">
4390                      <label>Оператор сравнения:</label>
4391                      <select id="prop-operator">
4392                          <option value="=" ${props.operator === '=' ? 'selected' : ''}
     >=  (равно)</option>
4393                          <option value=">" ${props.operator === '>' ? 'selected' : ''}
     >>  (больше)</option>
4394                          <option value="<" ${props.operator === '<' ? 'selected' : ''}
     ><  (меньше)</option>
4395                          <option value=">=" ${props.operator === '>=' ? 'selected' :
     ''}>= (больше или равно)</option>
4396                          <option value="<=" ${props.operator === '<=' ? 'selected' :
     ''}>= (меньше или равно)</option>
4397                          <option value="!=" ${props.operator === '!=' ? 'selected' :
     ''}>!= (не равно)</option>
4398                      </select>
4399                  </div>
4400              `;
4401          } else if (elemType === 'and' || elemType === 'or') {
```

```
4402             contentHTML = `
4403                 <div class="modal-row">
4404                     <label>Количество входов:</label>
4405                     <input type="number" id="prop-input-count" value="$
     {props.inputCount || 2}" min="2" max="10">
4406                 </div>
4407                 <div class="modal-row">
4408                     <p style="color: #aaa; font-size: 12px;">
4409                         Измените количество входных портов для этого логического
     элемента.
4410                         Лишние соединения будут автоматически удалены.
4411                     </p>
4412                 </div>
4413             `;
4414         } else if (elemType === 'const') {
4415             contentHTML = `
4416                 <div class="modal-row">
4417                     <label>Значение:</label>
4418                     <input type="number" id="prop-value" value="${props.value ?? 0}"
     step="any">
4419                 </div>
4420             `;
4421         }
4422         else if (elemType === 'group') {
4423             contentHTML = `
4424             <div class="modal-row">
4425                 <label>Название группы:</label>
4426                 <input type="text" id="prop-title" value="${props.title || 'Группа'}">
4427             </div>`;
4428         }
4429 
4430         else if (elemType === 'formula') {
4431             let signalsHTML = '';
4432             AppState.connections.forEach(conn => {
4433                 if (conn.toElement === elemId) {
4434                     const fromElem = AppState.elements[conn.fromElement];
4435                     if (fromElem) {
4436                         const signalName = fromElem.props?.name || fromElem.id;
4437                         signalsHTML += `<div class="signal-item" data-signal="$
     {signalName}">${signalName} (${conn.toPort})</div>`;
4438                     }
4439                 }
4440             });
4441 
4442             // ... (где-то выше код сбора signalsHTML) ...
4443 
4444             contentHTML = `
4445                 <div class="modal-row">
4446                     <label>Количество входов:</label>
4447                     <input type="number" id="prop-input-count" value="$
     {props.inputCount || 2}" min="1" max="10">
4448                 </div>
4449 
4450                 <!-- Верхний блок: Две колонки (Сигналы и Шаблоны) -->
4451                 <div style="display: flex; gap: 15px; margin-bottom: 15px; height:
     140px;">
4452                     <!-- Левая колонка: Сигналы -->
4453                     <div style="flex: 1; display: flex; flex-direction: column;">
4454                         <label style="margin-bottom: 5px; display:block;">Входные
     сигналы:</label>
4455                         <div class="signal-list" id="signal-list" style="flex: 1;
     overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px
     solid #4a90d9;">
4456                             ${signalsHTML || '<div style="color:#888;padding:5px;">Нет
     сигналов</div>'}
```

```
                        </div>
                    </div>

                    <!-- Правая колонка: Шаблоны -->
                    <div style="flex: 1; display: flex; flex-direction: column;">
                        <label style="margin-bottom: 5px; display:block;">Шаблоны:</label>
                        <div class="signal-list" id="template-list" style="flex: 1; overflow-y: auto; background: #0f3460; padding: 5px; border-radius: 4px; border: 1px solid #4a90d9;">
                            <div style="color:#888;padding:5px;">Загрузка…</div>
                        </div>
                    </div>
                </div>

                <!-- Нижний блок: Поле формулы (во всю ширину) -->
                <div class="modal-row">
                    <label>Выражение формулы:</label>
                    <textarea id="prop-expression"
                              style="width: 100%; min-height: 80px; font-family: monospace; font-size: 14px; line-height: 1.4;"
                              spellcheck="false">${props.expression || ''}</textarea>
                    <small style="color:#999; display:block; margin-top:4px;">
                        Двойной клик по сигналу или шаблону вставит его в позицию курсора (или заменит выделенный текст).
                    </small>
                </div>
            `;
        }
        if (!contentHTML) {
            contentHTML = `<div style="color:#aaa; font-size:12px;">Нет специальных свойств.</div>`;
            }
        contentHTML += `
            <div class="modal-row">
                <label>Комментарий:</label>
                <textarea id="prop-comment" placeholder="Комментарий к элементу...">${props.comment || ''}</textarea>
            </div>
            `;


        modalContent.innerHTML = contentHTML;
        if (elemType === 'formula') {
            const listEl = document.getElementById('template-list');
            (async () => {
                try {
                const data = await Settings.fetchFormulaTemplates();
                const items = data.templates || [];
                if (!items.length) {
                    listEl.innerHTML = '<div style="color:#888;padding:5px;">Нет шаблонов</div>';
                    return;
                }
                listEl.innerHTML = items.map(t => {
                    const sig = `${t.name}(${(t.args || []).join(', ')})`;
                    return `<div class="signal-item template-item" data-insert="${sig}">${sig}</div>`;
                }).join('');

                listEl.querySelectorAll('.template-item').forEach(div => {
                    div.addEventListener('dblclick', () => {
                        const insert = div.dataset.insert;
                        const textarea = document.getElementById('prop-expression');

```

```
4513                        // БЫЛО: textarea.value += ...;
4514                        // СТАЛО:
4515                        insertAtCursor(textarea, insert);
4516                    });
4517                });
4518                } catch (e) {
4519                console.error(e);
4520                listEl.innerHTML = '<div style="color:#888;padding:5px;">Ошибка
      загрузки</div>';
4521                }
4522            })();
4523        }
4524
4525
4526
4527        // --- post init handlers (когда DOM модалки уже существует) ---
4528        if (elemType === 'input-signal') {
4529            const input = document.getElementById('prop-name');
4530            const results = document.getElementById('signal-filter-results');
4531            const descField = document.getElementById('prop-description');
4532
4533            let timer = null;
4534
4535            const renderList = (items) => {
4536                if (!items || items.length === 0) {
4537                results.innerHTML = '<div style="color:#666;padding:6px;">Нет
      совпадений</div>';
4538                results.style.display = 'block';
4539                return;
4540                }
4541
4542                results.innerHTML = items.map(s => `
4543                <div class="signal-result-item"
4544                    style="padding:6px 8px; cursor:pointer; border-bottom:1px solid
      rgba(255,255,255,0.08);">
4545                    <div style="font-weight:600;">${s.Tagname}</div>
4546                    <div style="color:#aaa; font-size:11px;">${s.Description || ''}</
      div>
4547                </div>
4548                `).join('');
4549
4550                results.style.display = 'block';
4551
4552                results.querySelectorAll('.signal-result-item').forEach((div, i) => {
4553                div.addEventListener('click', () => {
4554                    const chosen = items[i];
4555                    input.value = chosen.Tagname;
4556                    descField.value = chosen.Description || '';
4557                    results.style.display = 'none';
4558                });
4559                });
4560            };
4561
4562            const search = async () => {
4563                const mask = (input.value || '').trim();
4564
4565                // Показываем список только если пользователь реально использует маску
4566                if (!mask.includes('*')) {
4567                results.style.display = 'none';
4568                return;
4569                }
4570
4571                results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</
      div>';
4572                results.style.display = 'block';
```

```
4573
4574                 try {
4575                 // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
4576                 const data = await Settings.fetchSignals(mask, 50);
4577                 renderList(data.items || []);
4578                 } catch (e) {
4579                 results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка
     загрузки сигналов</div>';
4580                 results.style.display = 'block';
4581                 console.error(e);
4582                 }
4583             };
4584
4585             input.addEventListener('input', () => {
4586                 clearTimeout(timer);
4587                 timer = setTimeout(search, 200); // debounce
4588             });
4589
4590             // опционально: закрывать список кликом вне
4591             document.addEventListener('mousedown', (e) => {
4592                 if (!results.contains(e.target) && e.target !== input) {
4593                 results.style.display = 'none';
4594                 }
4595             }, { once: true });
4596         }
4597         modalOverlay.dataset.elementId = elemId;
4598         this.showModal('modal-overlay');
4599
4600         // Функция для умной вставки текста в позицию курсора
4601         const insertAtCursor = (field, text) => {
4602             if (!field) return;
4603
4604             // Получаем позиции выделения
4605             const startPos = field.selectionStart;
4606             const endPos = field.selectionEnd;
4607             const currentValue = field.value;
4608
4609             // Вставляем текст: (текст до) + (новый текст) + (текст после)
4610             field.value = currentValue.substring(0, startPos) +
4611                         text +
4612                         currentValue.substring(endPos, currentValue.length);
4613
4614             // Возвращаем фокус и ставим курсор сразу после вставленного текста
4615             field.focus();
4616             const newCursorPos = startPos + text.length;
4617             field.setSelectionRange(newCursorPos, newCursorPos);
4618         };
4619
4620         // Обработчик вставки сигналов для формулы
4621         if (elemType === 'formula') {
4622             document.querySelectorAll('.signal-item').forEach(item => {
4623             item.addEventListener('dblclick', () => {
4624                 const signal = item.dataset.signal;
4625                 const textarea = document.getElementById('prop-expression');
4626
4627                 // БЫЛО: textarea.value += signal;
4628                 // СТАЛО:
4629                 insertAtCursor(textarea, signal);
4630                 });
4631             });
4632         }
4633     },
4634
4635     /**
4636      * Сохранить свойства элемента
```

```
     */
/**
 * Сохранить свойства элемента
 */
    saveElementProperties() {
        try {
            const modalOverlay = document.getElementById('modal-overlay');
            const elemId = modalOverlay.dataset.elementId;
            const elemData = AppState.elements[elemId];
            const elem = document.getElementById(elemId);
            if (!elemData) {
                alert('⚠ Элемент не найден — возможно, он был удалён или переименован.');
                console.warn(`saveElementProperties: элемент ${elemId} не найден.`);
                this.hideModal('modal-overlay');
                return;
            }

            const elemType = elemData.type;

            if (elemType === 'input-signal') {
                const name = document.getElementById('prop-name').value || 'Сигнал';
                const description = document.getElementById('prop-description').value || '';
                const signalType = document.getElementById('prop-signal-type').value;

                const oldSignalType = elemData.props.signalType;
                elemData.props.name = name;
                elemData.props.description = description;
                elemData.props.signalType = signalType;

                if (oldSignalType !== signalType) {
                    AppState.connections = AppState.connections.filter(conn => {
                    if (conn.fromElement === elemId) {
                        const toPortIndex = parseInt(conn.toPort.split('-')[1]);
                        const inputType = getInputPortType(conn.toElement, toPortIndex);
                        return areTypesCompatible(signalType, inputType);
                    }
                    return true;
                    });
                }

                const { html } = Elements.createElementHTML(
                    elemType, elemId, elemData.x, elemData.y, elemData.props, elemData.width, elemData.height
                );
                elem.outerHTML = html;

                Elements.setupElementHandlers(elemId);
                Connections.drawConnections();
            } else if (elemType === 'if') {
                const operator = document.getElementById('prop-operator').value;
                elemData.props.operator = operator;
                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = operator;

            } else if (elemType === 'const') {
                const value = parseFloat(document.getElementById('prop-value').value) || 0;
                elemData.props.value = value;
                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = String(value);

            } else if (elemType === 'formula') {
```

```
4697                 const expression = document.getElementById('prop-expression').value;
4698                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
4699 
4700                 elemData.props.expression = expression;
4701                 elemData.props.inputCount = inputCount;
4702 
4703                 const symbol = elem.querySelector('.element-symbol');
4704                 if (symbol) {
4705                     symbol.textContent = expression.length > 12 ? `$
      {expression.slice(0, 12)}…` : (expression || 'f(x)');
4706                 }
4707 
4708                 Elements.updateFormulaInputs(elemId, inputCount);
4709                 Elements.updateElementSize(elemId);  // ← Добавляем это
4710             } else if (elemType === 'and' || elemType === 'or') {
4711                 const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
4712                 elemData.props.inputCount = inputCount;
4713 
4714                 Elements.updateLogicGateInputs(elemId, inputCount);
4715                 Elements.updateElementSize(elemId);  // ← Добавляем это
4716 
4717                 const symbol = elem.querySelector('.element-symbol');
4718                 if (symbol) {
4719                     symbol.textContent = elemType === 'and' ? '∧' : '∨';
4720                 }
4721 
4722         } else if (elemType === 'output') {
4723                 const label = document.getElementById('prop-label').value || 'Выход';
4724                 const outputGroup = document.getElementById('prop-output-group').value
      || '';
4725 
4726                 elemData.props.label = label;
4727                 elemData.props.outputGroup = outputGroup;
4728 
4729                 const symbol = elem.querySelector('.element-symbol');
4730                 if (symbol) symbol.textContent = label;
4731             }
4732             else if (elemType === 'group') {
4733                 const title = document.getElementById('prop-title').value || 'Группа';
4734                 elemData.props.title = title;
4735                 const titleEl = elem.querySelector('.group-title');
4736                 if (titleEl) titleEl.textContent = title;
4737                 }
4738             const commentEl = document.getElementById('prop-comment');
4739             if (commentEl) elemData.props.comment = commentEl.value || '';
4740 
4741             this.hideModal('modal-overlay');
4742 
4743         } catch (error) {
4744             console.error('❌ Ошибка при сохранении свойств:', error);
4745             alert('Ошибка сохранения: ' + error.message);
4746         }
4747     },
4748 
4749     /**
4750      * Показать свойства проекта
4751      */
4752     showProjectPropertiesModal() {
4753         const content = document.getElementById('project-modal-content');
4754         const project = AppState.project;
4755 
4756         // Генерируем HTML для списка выходов только если модуль загружен
4757         let outputsHtml = '';
```

```
4758          if (typeof Outputs !== 'undefined' && AppState.outputs) {
4759              const logicalOutputsHtml = AppState.outputs.logical.length > 0
4760                  ? AppState.outputs.logical.map(output => `
4761                      <div class="output-item"
4762                          data-element-id="${output.elementId}"
4763                          onmouseenter="Outputs.highlightOutput('${output.elementId}',
      true)"
4764                          onmouseleave="Outputs.highlightOutput('${output.elementId}',
      false)"
4765                          onclick="Outputs.navigateToOutput('${output.elementId}');
      Modal.hideModal('project-modal-overlay');">
4766                          <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
      '❌'}</span>
4767                          <span class="output-name">${output.elementName}</span>
4768                          <span class="output-port">→ ${output.portLabel}</span>
4769                      </div>
4770                  `).join('')
4771                  : '<div class="no-outputs">Нет логических выходов</div>';
4772
4773              const numericOutputsHtml = AppState.outputs.numeric.length > 0
4774                  ? AppState.outputs.numeric.map(output => `
4775                      <div class="output-item numeric"
4776                          data-element-id="${output.elementId}"
4777                          onmouseenter="Outputs.highlightOutput('${output.elementId}',
      true)"
4778                          onmouseleave="Outputs.highlightOutput('${output.elementId}',
      false)"
4779                          onclick="Outputs.navigateToOutput('${output.elementId}');
      Modal.hideModal('project-modal-overlay');">
4780                          <span class="output-icon">🔢</span>
4781                          <span class="output-name">${output.elementName}</span>
4782                          <span class="output-port">→ значение</span>
4783                      </div>
4784                  `).join('')
4785                  : '<div class="no-outputs">Нет числовых выходов</div>';
4786
4787              outputsHtml = `
4788                  <div class="modal-row">
4789                      <label>Выходные сигналы схемы:</label>
4790                      <div class="outputs-container">
4791                          <div class="outputs-section">
4792                              <div class="outputs-section-title">
4793                                  <span class="section-icon">🔀</span>
4794                                  Логические выходы (${AppState.outputs.logical.length})
4795                              </div>
4796                              <div class="outputs-list">
4797                                  ${logicalOutputsHtml}
4798                              </div>
4799                          </div>
4800                          <div class="outputs-section">
4801                              <div class="outputs-section-title">
4802                                  <span class="section-icon">📐</span>
4803                                  Числовые выходы (${AppState.outputs.numeric.length})
4804                              </div>
4805                              <div class="outputs-list">
4806                                  ${numericOutputsHtml}
4807                              </div>
4808                          </div>
4809                      </div>
4810                      <div class="outputs-hint">
4811                          💡 Выходами автоматически становятся элементы, чьи выходные
      порты не подключены к другим элементам.
4812                          Кликните на выход, чтобы перейти к нему на схеме.
4813                      </div>
4814                  </div>
```

```
4815             `;
4816         }
4817 
4818         content.innerHTML = `
4819             <div class="modal-row">
4820                 <label>Код проекта:</label>
4821                 <input type="text" id="project-code" value="${project.code || ''}"
      placeholder="Уникальный идентификатор">
4822             </div>
4823 
4824             <div class="modal-row">
4825                 <label>Тип проекта:</label>
4826                 <div class="project-type-selector">
4827                     <div class="project-type-btn ${project.type ===
      PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
4828                         <div class="type-icon">📊</div>
4829                         <div class="type-name">Параметр</div>
4830                         <div class="type-desc">Вычисляемое значение</div>
4831                     </div>
4832                     <div class="project-type-btn ${project.type ===
      PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
4833                         <div class="type-icon">📋</div>
4834                         <div class="type-name">Правило</div>
4835                         <div class="type-desc">Логическое условие</div>
4836                     </div>
4837                 </div>
4838             </div>
4839 
4840             <div id="parameter-fields" class="conditional-fields ${project.type ===
      PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
4841                 <div class="modal-row">
4842                     <label>Размерность:</label>
4843                     <input type="text" id="project-dimension" value="$
      {project.dimension || ''}" placeholder="Например: м/с, кг, °C">
4844                 </div>
4845             </div>
4846 
4847             <div id="rule-fields" class="conditional-fields ${project.type ===
      PROJECT_TYPE.RULE ? 'visible' : ''}">
4848                 <div class="modal-row">
4849                     <label>Возможная причина:</label>
4850                     <textarea id="project-possible-cause" placeholder="Описание
      возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
4851                 </div>
4852                 <div class="modal-row">
4853                     <label>Методические указания:</label>
4854                     <textarea id="project-guidelines" placeholder="Инструкции и
      рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
4855                 </div>
4856             </div>
4857 
4858             ${outputsHtml}
4859         `;
4860 
4861         // Обработчики переключения типа
4862         content.querySelectorAll('.project-type-btn').forEach(btn => {
4863             btn.addEventListener('click', () => {
4864                 content.querySelectorAll('.project-type-btn').forEach(b =>
      b.classList.remove('active'));
4865                 btn.classList.add('active');
4866 
4867                 const type = btn.dataset.type;
4868                 document.getElementById('parameter-
      fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
4869                 document.getElementById('rule-fields').classList.toggle('visible',
```

```
    type === PROJECT_TYPE.RULE);
            });
        });

        this.showModal('project-modal-overlay');
    },

    /**
     * Сохранить свойства проекта
     */
    saveProjectProperties() {
        const activeTypeBtn = document.querySelector('.project-type-btn.active');
        const type = activeTypeBtn ? activeTypeBtn.dataset.type :
    PROJECT_TYPE.PARAMETER;

        AppState.project.code = document.getElementById('project-code').value;
        AppState.project.type = type;

        if (type === PROJECT_TYPE.PARAMETER) {
            AppState.project.dimension = document.getElementById('project-
    dimension').value;
            AppState.project.possibleCause = '';
            AppState.project.guidelines = '';
        } else {
            AppState.project.dimension = '';
            AppState.project.possibleCause = document.getElementById('project-
    possible-cause').value;
            AppState.project.guidelines = document.getElementById('project-
    guidelines').value;
        }

        this.hideModal('project-modal-overlay');
    }
};

output.js

/**
 * Модуль управления выходными сигналами
 */

const Outputs = {
    /**
     * Обновление статуса выходных элементов
     * Вызывается при каждом изменении схемы
     */
    updateOutputStatus() {
        this.clearAllOutputHighlights();
        AppState.outputs.logical = [];
        AppState.outputs.numeric = [];
        updateFrameChildren();

        // Обработка элементов-выходов
        Object.values(AppState.elements).forEach(elem => {
            if (!elem || elem.type !== 'output') return;

            // Проверяем, к чему подключен вход этого выхода
            const inputConns = AppState.connections.filter(c =>
                c.toElement === elem.id && c.toPort === 'in-0'
            );

            // Каждое соединение к выходу — это отдельный выход
            inputConns.forEach((conn, index) => {
                const fromElem = AppState.elements[conn.fromElement];
                if (!fromElem) return;
```

```
4930
4931                const outputType = conn.signalType;
4932                const outputInfo = {
4933                    id: `${elem.id}_conn_${index}`,
4934                    elementId: elem.id,
4935                    sourceElementId: conn.fromElement,
4936                    sourcePort: conn.fromPort,
4937                    portIndex: 0,
4938                    portId: 'in-0',
4939                    type: outputType,
4940                    label: elem.props?.label || 'Выход',
4941                    elementType: 'output',
4942                    elementName: elem.props?.label || 'Выход',
4943                    name: elem.props?.label || 'Выход'
4944                };
4945
4946                if (outputType === SIGNAL_TYPE.LOGIC) {
4947                    AppState.outputs.logical.push(outputInfo);
4948                } else if (outputType === SIGNAL_TYPE.NUMBER) {
4949                    AppState.outputs.numeric.push(outputInfo);
4950                }
4951
4952                // Подсветим входной порт
4953                this.highlightOutputPort(elem.id, 0, outputType);
4954            });
4955        });
4956
4957        this.updateOutputCounter();
4958    },
4959
4960    /**
4961     * Очистка всех выделений выходов
4962     */
4963    clearAllOutputHighlights() {
4964        document.querySelectorAll('.port.output-active').forEach(port => {
4965            port.classList.remove('output-active');
4966        });
4967
4968        document.querySelectorAll('.element.has-output').forEach(elem => {
4969            elem.classList.remove('has-output');
4970        });
4971
4972        document.querySelectorAll('.element.output-ambiguous').forEach(el =>
     el.classList.remove('output-ambiguous'));
4973        document.querySelectorAll('.element.output-missing').forEach(el =>
     el.classList.remove('output-missing'));
4974    },
4975
4976    /**
4977     * Выделение выходного порта
4978     */
4979    highlightOutputPort(elemId, portIndex, portType) {
4980        const elem = document.getElementById(elemId);
4981        if (!elem) return;
4982
4983        const port = elem.querySelector(`.port.output[data-port="out-${portIndex}"]`);
4984        if (port) {
4985            port.classList.add('output-active');
4986        }
4987
4988        // Добавляем класс элементу (даёт общий визуал)
4989        elem.classList.add('has-output');
4990    },
4991
4992    /**
```

```
4993        * Обновление счётчика выходов в меню
4994        */
4995       updateOutputCounter() {
4996           const counter = document.getElementById('output-counter');
4997           if (counter) {
4998               const total = AppState.outputs.logical.length +
      AppState.outputs.numeric.length;
4999               counter.textContent = total;
5000               counter.style.display = total > 0 ? 'inline-block' : 'none';
5001           }
5002       },
5003
5004       /**
5005        * Получить все выходы для сохранения в проект
5006        */
5007       getOutputsForSave() {
5008           // Сохраняем информацию о frame/inner для рамок
5009           return {
5010               logical: AppState.outputs.logical.map(o => ({
5011                   id: o.id,
5012                   elementId: o.elementId,
5013                   frameId: o.frameId || null,
5014                   innerElementId: o.innerElementId || null,
5015                   portIndex: o.portIndex ?? o.innerPortIndex ?? null,
5016                   portLabel: o.label
5017               })),
5018               numeric: AppState.outputs.numeric.map(o => ({
5019                   id: o.id,
5020                   elementId: o.elementId,
5021                   frameId: o.frameId || null,
5022                   innerElementId: o.innerElementId || null,
5023                   portIndex: o.portIndex ?? o.innerPortIndex ?? null,
5024                   portLabel: o.label
5025               }))
5026           };
5027       },
5028
5029       /**
5030        * Подсветить конкретный выход (при наведении в списке)
5031        */
5032       highlightOutput(elementId, highlight = true) {
5033           const elem = document.getElementById(elementId);
5034           if (elem) {
5035               if (highlight) {
5036                   elem.classList.add('output-highlighted');
5037               } else {
5038                   elem.classList.remove('output-highlighted');
5039               }
5040           }
5041       },
5042
5043       /**
5044        * Перейти к элементу выхода на схеме (elementId — фокусируемый элемент; для рамок
      это id рамки)
5045        */
5046       navigateToOutput(elementId) {
5047           const elemData = AppState.elements[elementId];
5048           if (!elemData) return;
5049
5050           // Центрируем viewport на элементе
5051           const container = document.getElementById('workspace-container');
5052           const rect = container.getBoundingClientRect();
5053
5054           const centerX = elemData.x + elemData.width / 2;
5055           const centerY = elemData.y + elemData.height / 2;
```

```
5056
5057          AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
5058          AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;
5059
5060          Viewport.updateTransform();
5061
5062          // Выделяем элемент
5063          Elements.selectElement(elementId);
5064
5065          // Временная подсветка
5066          this.highlightOutput(elementId, true);
5067          setTimeout(() => this.highlightOutput(elementId, false), 2000);
5068      }
5069  };
5070
5071  project.js
5072
5073  /**
5074   * Модуль управления проектом (сохранение, загрузка)
5075   */
5076
5077  // --- миграция id: '-' -> '_' с обновлением всех ссылок ---
5078  function migrateIdsDashToUnderscore() {
5079    const map = {};
5080
5081    // 1) собрать map старых id → новых
5082    Object.values(AppState.elements).forEach(el => {
5083      if (typeof el.id === 'string' && el.id.includes('-')) {
5084        map[el.id] = el.id.replace(/-/g, '_');
5085      }
5086    });
5087
5088    if (!Object.keys(map).length) return;
5089
5090    // 2) DOM id + data-element
5091    Object.entries(map).forEach(([oldId, newId]) => {
5092      const dom = document.getElementById(oldId);
5093      if (dom) dom.id = newId;
5094
5095      if (dom) {
5096        dom.querySelectorAll('[data-element]').forEach(p => {
5097          if (p.dataset.element === oldId) p.dataset.element = newId;
5098        });
5099      }
5100    });
5101
5102    // 3) AppState.elements ключи
5103    Object.entries(map).forEach(([oldId, newId]) => {
5104      const el = AppState.elements[oldId];
5105      if (!el) return;
5106      el.id = newId;
5107      AppState.elements[newId] = el;
5108      delete AppState.elements[oldId];
5109    });
5110
5111    // 4) connections
5112    AppState.connections.forEach(c => {
5113      if (map[c.fromElement]) c.fromElement = map[c.fromElement];
5114      if (map[c.toElement]) c.toElement = map[c.toElement];
5115    });
5116
5117    // 5) формулы
5118    const escapeRegex = s => s.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
5119    Object.values(AppState.elements).forEach(el => {
5120      if (el.type === 'formula' && el.props?.expression) {
```

```
5121        let expr = el.props.expression;
5122        Object.entries(map).forEach(([oldId, newId]) => {
5123          const re = new RegExp(`(^|[^A-Za-z0-9_])${escapeRegex(oldId)}(?![A-Za-
      z0-9_])`, 'g');
5124          expr = expr.replace(re, (m, p1) => `${p1}${newId}`);
5125        });
5126        el.props.expression = expr;
5127      }
5128    });
5129
5130    // 6) selected + modal
5131    if (map[AppState.selectedElement]) AppState.selectedElement =
      map[AppState.selectedElement];
5132    const modal = document.getElementById('modal-overlay');
5133    if (modal && map[modal.dataset.elementId]) modal.dataset.elementId =
      map[modal.dataset.elementId];
5134  }
5135
5136  const Project = {
5137      /**
5138       * Инициализация
5139       */
5140          /**
5141       * Инициализация
5142       */
5143  init() {
5144      document.getElementById('btn-new').addEventListener('click', () =>
      this.newProject());
5145      document.getElementById('btn-save').addEventListener('click', () =>
      this.saveProject());
5146      document.getElementById('btn-load').addEventListener('click', () =>
      this.openProjectListModal());
5147      document.getElementById('btn-project-settings').addEventListener('click', () => {
5148          Modal.showProjectPropertiesModal();
5149      });
5150
5151      // Работа с модалкой выбора проекта
5152      this.projectList = [];
5153      this.filteredProjectList = [];
5154      this.selectedProjectFilename = null;
5155
5156      document.getElementById('project-cancel').addEventListener('click', () =>
      this.closeProjectListModal());
5157      document.getElementById('project-refresh').addEventListener('click', () =>
      this.refreshProjectList());
5158
5159      document.getElementById('project-load').addEventListener('click', () => {
5160          if (this.selectedProjectFilename) {
5161              this.loadProjectFromList(this.selectedProjectFilename);
5162          }
5163      });
5164
5165      document.getElementById('project-search').addEventListener('input', (event) => {
5166          this.filterProjectList(event.target.value);
5167      });
5168  },
5169
5170      /**
5171       * Новый проект
5172       */
5173      newProject() {
5174          if (Object.keys(AppState.elements).length > 0) {
5175              if (!confirm('Создать новый проект? Несохранённые изменения будут
      потеряны.')) {
5176                  return;
```

```
5177             }
5178         }
5179
5180         document.getElementById('workspace').innerHTML = '';
5181         document.getElementById('connections-svg').innerHTML = '';
5182
5183         resetState();
5184         Viewport.updateTransform();
5185     },
5186
5187         /**
5188      * Запрос имени файла и загрузка с сервера
5189      */
5190     async loadProjectPrompt() {
5191         const filename = window.prompt(
5192             "Введите имя файла проекта для загрузки (с сервера). Пример:
      scheme_logic.json",
5193             AppState.project.code ? `${AppState.project.code}_$
      {AppState.project.type}.json` : "scheme_type.json"
5194         );
5195
5196         if (!filename) return; // Отмена
5197
5198         try {
5199             // Используем обертку из Settings.js для запроса к /api/project/load
5200             const data = await Settings.loadProject(filename);
5201
5202             // Если загрузка успешна, вызываем основную функцию обработки данных
5203             this._processLoadedData(data);
5204             alert(`Проект "${filename}" успешно загружен с сервера.`);
5205
5206         } catch (error) {
5207             console.error('Ошибка загрузки проекта:', error);
5208             alert(`Ошибка загрузки проекта: ${error.message}`);
5209         }
5210     },
5211
5212     /**
5213      * Сохранение проекта
5214      */
5215         async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
5216         // 1. Проверяем свойства проекта
5217         if (!AppState.project.code) {
5218             Modal.showProjectPropertiesModal();
5219             alert('Пожалуйста, укажите код проекта перед сохранением.');
5220             return;
5221         }
5222
5223         // Обновляем размеры рамок перед сохранением
5224         updateFrameChildren();
5225         // ✅ нормализуем, даже если проект был открыт до фикса
5226         migrateIdsDashToUnderscore();
5227
5228         // ✅ подчистим связи прямо перед сохранением
5229         const exists = (id) => !!AppState.elements[id];
5230         AppState.connections = (AppState.connections || [])
5231           .map(c => ({
5232             ...c,
5233             fromElement: exists(c.fromElement) ? c.fromElement :
      c.fromElement.replace(/-/g, '_'),
5234             toElement: exists(c.toElement) ? c.toElement : c.toElement.replace(/-/g,
      '_')
5235           }))
5236           .filter(c => exists(c.fromElement) && exists(c.toElement))
5237           .filter((c, idx, arr) => {
```

```
5238             const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5239             return arr.findIndex(x =>
5240               `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5241             ) === idx;
5242           });
5243         // ✅ 1. Генерируем код заранее
5244         let generatedCode = '';
5245         if (typeof CodeGen !== 'undefined' && typeof CodeGen.generate === 'function')
     {
5246             try {
5247                 generatedCode = CodeGen.generate() || '';
5248             } catch (err) {
5249                 console.error('Code generation failed:', err);
5250             }
5251         }
5252
5253         // 2. Сборка объекта проекта
5254         const project = {
5255             version: '1.0',
5256             project: AppState.project,
5257             elements: AppState.elements,
5258             connections: AppState.connections,
5259             counter: AppState.elementCounter,
5260             viewport: {
5261                 zoom: AppState.viewport.zoom,
5262                 panX: AppState.viewport.panX,
5263                 panY: AppState.viewport.panY
5264             },
5265             code: generatedCode
5266         };
5267
5268         const filename = `${AppState.project.code || 'scheme'}_$
     {AppState.project.type}.json`;
5269
5270         // 3. Сохранение на сервер
5271         try {
5272             await Settings.saveProject(filename, project);
5273             alert(`Проект успешно сохранен на сервере как: ${filename}`);
5274         } catch (error) {
5275             console.error('Ошибка сохранения проекта:', error);
5276             alert(`Ошибка сохранения проекта: ${error.message}`);
5277         }
5278     },
5279
5280     async showProjectList() {
5281         try {
5282             const result = await Settings.listProjects(); // нужно реализовать в
     settings.js
5283             const list = result.projects || [];
5284
5285             if (list.length === 0) {
5286                 alert('Проекты в папке не найдены.');
5287                 return;
5288             }
5289
5290             const choice = window.prompt(
5291                 'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code ||
     p.filename} — ${p.description}`).join('\n') +
5292                 '\n\nВведите номер проекта для загрузки:',
5293                 '1'
5294             );
5295             const index = parseInt(choice, 10) - 1;
5296             if (isNaN(index) || !list[index]) return;
5297
5298             await this.loadProjectByFilename(list[index].filename);
```

```
5299          } catch (error) {
5300              console.error(error);
5301              alert('Не удалось получить список проектов: ' + error.message);
5302          }
5303      },
5304
5305      async loadProjectByFilename(filename) {
5306          try {
5307              const data = await Settings.loadProject(filename);
5308              this._processLoadedData(data);
5309              alert(`Проект "${filename}" загружен.`);
5310          } catch (error) {
5311              console.error(error);
5312              alert('Ошибка загрузки проекта: ' + error.message);
5313          }
5314      },
5315
5316      openProjectListModal() {
5317    const modal = document.getElementById('modal-project-list');
5318    modal.classList.remove('hidden');
5319    document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
5320    this.refreshProjectList();
5321  },
5322
5323  closeProjectListModal() {
5324    const modal = document.getElementById('modal-project-list');
5325    modal.classList.add('hidden');
5326    document.body.classList.remove('modal-open');
5327  },
5328
5329  async refreshProjectList() {
5330    const tbody = document.getElementById('project-list-body');
5331    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
5332    try {
5333      const result = await Settings.listProjects();
5334      this.projectList = result.projects || [];
5335      this.filteredProjectList = [...this.projectList];
5336      this.renderProjectList();
5337    } catch (err) {
5338      console.error(err);
5339      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
5340    }
5341  },
5342
5343  renderProjectList() {
5344    const tbody = document.getElementById('project-list-body');
5345    const loadBtn = document.getElementById('project-load');
5346    loadBtn.disabled = true;
5347    this.selectedProjectFilename = null;
5348
5349    if (!this.filteredProjectList.length) {
5350      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
5351      return;
5352    }
5353
5354    tbody.innerHTML = '';
5355    this.filteredProjectList.forEach((item) => {
5356      const tr = document.createElement('tr');
5357      tr.innerHTML = `
5358        <td>${item.filename}</td>
5359        <td>${item.code || ''}</td>
```

```
        <td>${item.description || ''}</td>
        <td>${item.type || ''}</td>
      `;
      tr.addEventListener('click', () => {
        this.highlightRow(tr);
        this.selectedProjectFilename = item.filename;
        loadBtn.disabled = false;
      });
      tr.addEventListener('dblclick', () => {
        this.highlightRow(tr);
        this.selectedProjectFilename = item.filename;
        loadBtn.disabled = false;
        this.loadProjectFromList(item.filename);
      });
      tbody.appendChild(tr);
    });
  },

  highlightRow(row) {
    const tbody = row.parentElement;
    [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
    row.classList.add('selected');
  },


  // Фильтр по поисковой строке
  filterProjectList(query) {
    const q = (query || '').trim().toLowerCase();
    if (!q) {
      this.filteredProjectList = [...this.projectList];
    } else {
      this.filteredProjectList = this.projectList.filter((item) => {
        return [
          item.filename,
          item.code,
          item.description,
          item.type
        ].some((field) => (field || '').toLowerCase().includes(q));
      });
    }
    this.renderProjectList();
  },

  async loadProjectFromList(filename) {
    try {
      const data = await Settings.loadProject(filename);
      this._processLoadedData(data);
      this.closeProjectListModal();
      alert(`Проект "${filename}" успешно загружен.`);
    } catch (error) {
      console.error(error);
      alert('Ошибка загрузки проекта: ' + error.message);
    }
  },









      /**
       * Загрузка проекта
       */
   _processLoadedData(data) {
    try {
```

```
5425      document.getElementById('workspace').innerHTML = '';
5426      document.getElementById('connections-svg').innerHTML = '';
5427      resetState();
5428
5429      if (data.project) {
5430        AppState.project = { ...AppState.project, ...data.project };
5431      }
5432
5433      AppState.elementCounter = data.counter || 0;
5434
5435      if (data.viewport) {
5436        AppState.viewport.zoom = data.viewport.zoom || 1;
5437        AppState.viewport.panX = data.viewport.panX || 0;
5438        AppState.viewport.panY = data.viewport.panY || 0;
5439      }
5440
5441      const elements = data.elements || {};
5442      Object.values(elements)
5443        .filter(e => e.type === 'output-frame')
5444        .forEach(elemData => {
5445          Elements.addElement(
5446            elemData.type,
5447            elemData.x,
5448            elemData.y,
5449            elemData.props,
5450            elemData.id,
5451            elemData.width,
5452            elemData.height
5453          );
5454        });
5455
5456      Object.values(elements)
5457        .filter(e => e.type !== 'output-frame')
5458        .forEach(elemData => {
5459          Elements.addElement(
5460            elemData.type,
5461            elemData.x,
5462            elemData.y,
5463            elemData.props,
5464            elemData.id,
5465            elemData.width,
5466            elemData.height
5467          );
5468        });
5469
5470      AppState.connections = data.connections || [];
5471
5472      // ✅ ВСТАВЬ ЭТОТ БЛОК СРАЗУ ЗДЕСЬ (до вычисления счётчика)
5473      //Object.values(AppState.elements).forEach(e => {
5474      //  if (typeof e.id === 'string') {
5475      //    e.id = e.id.replace(/-/g, '_');
5476      //  }
5477      //  if (e.props?.name) {
5478      //    e.props.name = e.props.name.replace(/-/g, '_');
5479      //  }
5480      //});
5481      // ✅ конец добавленной секции
5482      // ✅ Миграция id: '-' -> '_'
5483      migrateIdsDashToUnderscore();
5484      // ✅ очистка соединений: удалить битые и дубликаты
5485      const exists = (id) => !!AppState.elements[id];
5486
5487      AppState.connections = (AppState.connections || [])
5488        // оставить только те, где оба конца реально существуют
5489        .filter(c => exists(c.fromElement) && exists(c.toElement))
```

```
5490        // убрать дубликаты
5491        .filter((c, idx, arr) => {
5492          const key = `${c.fromElement}|${c.fromPort}|${c.toElement}|${c.toPort}`;
5493          return arr.findIndex(x =>
5494            `${x.fromElement}|${x.fromPort}|${x.toElement}|${x.toPort}` === key
5495          ) === idx;
5496        });
5497
5498
5499      // корректно восстанавливаем счётчик
5500      const counterFromFile = Number(data.counter);
5501      AppState.elementCounter = Number.isFinite(counterFromFile) ? counterFromFile : 0;
5502
5503      const maxIdSuffix = Object.values(AppState.elements).reduce((max, el) => {
5504          if (!el?.id) return max;
5505          const match = String(el.id).match(/_(\d+)$/);   // теперь хвост по
      подчёркиванию
5506          const num = match ? parseInt(match[1], 10) : NaN;
5507          return Number.isFinite(num) ? Math.max(max, num) : max;
5508      }, 0);
5509
5510      AppState.elementCounter = Math.max(AppState.elementCounter, maxIdSuffix);
5511
5512      Viewport.updateTransform();
5513      Connections.drawConnections();
5514      updateFrameChildren();
5515
5516    } catch (e) {
5517      alert('Ошибка обработки данных проекта: ' + e.message);
5518      console.error(e);
5519    }
5520  }
5521  };
5522
5523  settings.js
5524
5525  const Settings = {
5526    config: null,
5527    templates: null,
5528
5529    async init() {
5530      // тянем настройки (не обязательно, но полезно)
5531      try {
5532        const r = await fetch('/api/settings');
5533        if (r.ok) this.config = await r.json();
5534      } catch (e) {
5535        console.warn('Settings load failed:', e);
5536      }
5537      try {
5538        const t = await this.fetchFormulaTemplates();
5539        this.templates = t.templates || [];
5540      } catch (e) {
5541        this.templates = [];
5542      }
5543    },
5544
5545    getTemplatesMap() {
5546      const map = {};
5547      (this.templates || []).forEach(t => { if (t?.name) map[t.name] = t; });
5548      return map;
5549    },
5550
5551    async fetchSignals(mask, limit = 50) {
5552      const url = `/api/signals?q=${encodeURIComponent(mask || '')}&limit=$
      {encodeURIComponent(limit)}`;
```

```
    const r = await fetch(url);
    if (!r.ok) throw new Error('Failed to fetch signals');
    return await r.json(); // {items, total}
  },
  // ... в объекте Settings

  async saveProject(filename, projectData) {
    if (!filename.endsWith('.json')) {
      filename += '.json';
    }
    const r = await fetch('/api/project/save', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        filename: filename,
        content: projectData
      })
    });
    if (!r.ok) throw new Error('Failed to save project');
    return r.json();
  },

  async listProjects() {
    const r = await fetch('/api/project/list');
    if (!r.ok) throw new Error('Failed to list projects');
    return r.json();
  },

  async fetchFormulaTemplates() {
      const r = await fetch('/api/formula-templates');
      if (!r.ok) throw new Error('Failed to fetch formula templates');
      return await r.json(); // {templates:[...]}
  },

  async loadProject(filename) {
    if (!filename.endsWith('.json')) {
      filename += '.json';
    }
    const r = await fetch(`/api/project/load/${encodeURIComponent(filename)}`);
    if (!r.ok) {
        if (r.status === 404) {
             throw new Error(`Project "${filename}" not found (404)`);
        }
        throw new Error('Failed to load project');
    }
    return r.json();
  }

// ...
};

state.js

/**
 * Глобальное состояние приложения
 */

const AppState = {
    // Элементы схемы
    elements: {},
    connections: [],
    elementCounter: 0,

    // Выделение
    selectedElement: null,
```

```
5618
5619      // Перетаскивание
5620      draggingElement: null,
5621      dragOffset: { x: 0, y: 0 },
5622      isDraggingFromPalette: false,
5623      dragPreview: null,
5624      dragType: null,
5625
5626      // Соединения
5627      connectingFrom: null,
5628      connectingFromType: null,
5629      tempLine: null,
5630
5631      // Resize
5632      resizing: null,
5633
5634      // Viewport (масштабирование и перемещение)
5635      viewport: {
5636          zoom: 1,
5637          panX: 0,
5638          panY: 0,
5639          isPanning: false,
5640          lastMouseX: 0,
5641          lastMouseY: 0
5642      },
5643
5644      // Свойства проекта
5645      project: {
5646          code: '',
5647          type: PROJECT_TYPE.PARAMETER,
5648          // Для параметра
5649          dimension: '',
5650          // Для правила
5651          possibleCause: '',
5652          guidelines: ''
5653      },
5654
5655      // Выходные сигналы (автоматически определяются)
5656      outputs: {
5657          logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
5658          numeric: []     // Числовые выходы (формулы)
5659      }
5660  };
5661
5662  /**
5663   * Сброс состояния
5664   */
5665  function resetState() {
5666      AppState.elements = {};
5667      AppState.connections = [];
5668      AppState.elementCounter = 0;
5669      AppState.selectedElement = null;
5670      AppState.draggingElement = null;
5671      AppState.connectingFrom = null;
5672      AppState.tempLine = null;
5673      AppState.resizing = null;
5674
5675      AppState.viewport = {
5676          zoom: 1,
5677          panX: 0,
5678          panY: 0,
5679          isPanning: false,
5680          lastMouseX: 0,
5681          lastMouseY: 0
5682      };
```

```
5683
5684      AppState.project = {
5685          code: '',
5686          type: PROJECT_TYPE.PARAMETER,
5687          dimension: '',
5688          possibleCause: '',
5689          guidelines: ''
5690      };
5691
5692      AppState.outputs = {
5693          logical: [],
5694          numeric: []
5695      };
5696  }
5697
5698  utils.js
5699
5700  /**
5701   * Вспомогательные функции
5702   */
5703
5704  /**
5705   * Генерация уникального ID
5706   */
5707  function generateId() {
5708      AppState.elementCounter++;
5709      return `elem_${AppState.elementCounter}`;
5710  }
5711
5712  function getInputPortType(elementId, portIdentifier) {
5713      const element = AppState.elements[elementId];
5714      if (!element) return SIGNAL_TYPE.ANY;
5715
5716      const config = ELEMENT_TYPES[element.type];
5717      if (!config) return SIGNAL_TYPE.ANY;
5718
5719      let portIndex = portIdentifier;
5720
5721      // Обработка технического порта условия
5722      if (typeof portIdentifier === 'string') {
5723          if (portIdentifier === 'cond-0' && config.hasConditionPort) {
5724              return config.conditionPortType || SIGNAL_TYPE.LOGIC;
5725          }
5726
5727          if (portIdentifier.startsWith('in-')) {
5728              portIndex = parseInt(portIdentifier.split('-')[1], 10);
5729          }
5730      }
5731
5732      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5733          portIndex = 0;
5734      }
5735
5736      // Динамические входы для AND/OR берут тип из конфига
5737      if ((element.type === 'and' || element.type === 'or')) {
5738          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
5739      }
5740
5741      if (element.type === 'formula') {
5742          return SIGNAL_TYPE.ANY;
5743      }
5744
5745      const types = config.inputTypes || [];
5746      if (types.length === 0) return SIGNAL_TYPE.ANY;
```

```
5747
5748      if (portIndex < types.length) {
5749          return types[portIndex] || SIGNAL_TYPE.ANY;
5750      }
5751
5752      return types[types.length - 1] || SIGNAL_TYPE.ANY;
5753  }
5754
5755  function getOutputPortType(elementId, portIdentifier) {
5756      const element = AppState.elements[elementId];
5757      if (!element) return SIGNAL_TYPE.ANY;
5758
5759      const config = ELEMENT_TYPES[element.type];
5760      if (!config) return SIGNAL_TYPE.ANY;
5761
5762      let portIndex = portIdentifier;
5763
5764      if (typeof portIdentifier === 'string') {
5765          if (portIdentifier.startsWith('out-')) {
5766              portIndex = parseInt(portIdentifier.split('-')[1], 10);
5767          }
5768      }
5769
5770      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5771          portIndex = 0;
5772      }
5773
5774      const types = config.outputTypes || [];
5775      if (types.length === 0) return SIGNAL_TYPE.ANY;
5776
5777      if (portIndex < types.length) {
5778          return types[portIndex] || SIGNAL_TYPE.ANY;
5779      }
5780
5781      return types[types.length - 1] || SIGNAL_TYPE.ANY;
5782  }
5783  /**
5784   * Проверка совместимости типов сигналов
5785   *
5786   * Новая логика:
5787   * - ANY совместим со всем
5788   * - TRUE совместим с LOGIC, TRUE, ANY
5789   * - FALSE совместим с LOGIC, FALSE, ANY
5790   * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
5791   * - NUMERIC совместим с NUMERIC, ANY
5792   */
5793  function areTypesCompatible(outputType, inputType) {
5794      // Если один из типов ANY - совместимы
5795      if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
5796          return true;
5797      }
5798
5799      // Если типы одинаковые - совместимы
5800      if (outputType === inputType) {
5801          return true;
5802      }
5803
5804      // TRUE/FALSE совместимы с LOGIC
5805      if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
5806          inputType === SIGNAL_TYPE.LOGIC) {
5807          return true;
5808      }
5809
5810      // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
5811      if (outputType === SIGNAL_TYPE.LOGIC &&
```

```
5812          (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
5813          return true;
5814      }
5815
5816      return false;
5817  }
5818
5819  /**
5820   * Проверка, находится ли элемент внутри рамки
5821   */
5822  function isInsideFrame(elemId, frameId) {
5823      const elem = AppState.elements[elemId];
5824      const frame = AppState.elements[frameId];
5825
5826      if (!elem || !frame || frame.type !== 'output-frame') return false;
5827
5828      const elemCenterX = elem.x + elem.width / 2;
5829      const elemCenterY = elem.y + elem.height / 2;
5830
5831      return elemCenterX > frame.x &&
5832             elemCenterX < frame.x + frame.width &&
5833             elemCenterY > frame.y &&
5834             elemCenterY < frame.y + frame.height;
5835  }
5836
5837  /**
5838   * Обновить принадлежность элементов к рамкам
5839   */
5840  function updateFrameChildren() {
5841      // Сначала очистим children у рамок и parentFrame у всех элементов
5842      Object.values(AppState.elements).forEach(elem => {
5843          if (elem.type === 'output-frame') {
5844              elem.children = [];
5845          } else {
5846              // удаляем parentFrame по умолчанию (пересчитаем ниже)
5847              if (elem.parentFrame) delete elem.parentFrame;
5848          }
5849      });
5850
5851      // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
     попадает
5852      Object.values(AppState.elements).forEach(elem => {
5853          if (!elem || elem.type === 'output-frame') return;
5854
5855          Object.values(AppState.elements).forEach(frame => {
5856              if (!frame || frame.type !== 'output-frame') return;
5857
5858              if (isInsideFrame(elem.id, frame.id)) {
5859                  // добавляем в массив детей рамки
5860                  frame.children.push(elem.id);
5861                  // отмечаем у элемента родительскую рамку
5862                  if (AppState.elements[elem.id]) {
5863                      AppState.elements[elem.id].parentFrame = frame.id;
5864                  }
5865              }
5866          });
5867      });
5868  }
5869
5870  /**
5871   * Преобразование координат экрана в координаты холста
5872   */
5873  function screenToCanvas(screenX, screenY) {
5874      const container = document.getElementById('workspace-container');
5875      const rect = container.getBoundingClientRect();
```

```
5876
5877      const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
5878      const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;
5879
5880      return { x, y };
5881  }
5882
5883  /**
5884   * Преобразование координат холста в координаты экрана
5885   */
5886  function canvasToScreen(canvasX, canvasY) {
5887      const container = document.getElementById('workspace-container');
5888      const rect = container.getBoundingClientRect();
5889
5890      const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
5891      const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;
5892
5893      return { x, y };
5894  }
5895
5896  /**
5897   * Проверка, является ли порт выходным (не подключен к другим элементам)
5898   */
5899  function isOutputPort(elemId, portIndex) {
5900      const portKey = `out-${portIndex}`;
5901
5902      // Проверяем, есть ли соединения от этого порта
5903      const hasConnection = AppState.connections.some(conn =>
5904          conn.fromElement === elemId && conn.fromPort === portKey
5905      );
5906
5907      return !hasConnection;
5908  }
5909
5910  /**
5911   * Получить информацию о выходном порте
5912   */
5913  function getOutputPortInfo(elemId, portIndex) {
5914      const elem = AppState.elements[elemId];
5915      if (!elem) return null;
5916
5917      const config = ELEMENT_TYPES[elem.type];
5918      if (!config) return null;
5919
5920      return {
5921          elementId: elemId,
5922          elementType: elem.type,
5923          elementName: config.name,
5924          portIndex: portIndex,
5925          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
5926          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
5927          // Дополнительная информация для идентификации
5928          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
5929      };
5930  }
5931
5932  function splitArgsTopLevel(argStr) {
5933    const out = [];
5934    let cur = '';
5935    let depth = 0;
5936    for (let i = 0; i < argStr.length; i++) {
5937      const ch = argStr[i];
5938      if (ch === '(') depth++;
5939      if (ch === ')') depth--;
```

```
5940      if (ch === ',' && depth === 0) {
5941        out.push(cur.trim());
5942        cur = '';
5943      } else {
5944        cur += ch;
5945      }
5946    }
5947    if (cur.trim()) out.push(cur.trim());
5948    return out;
5949  }
5950
5951  function expandFormulaTemplates(expr, templatesMap) {
5952    if (!expr) return expr;
5953    if (!templatesMap) return expr;
5954
5955    // несколько проходов на случай вложенных шаблонов
5956    for (let pass = 0; pass < 10; pass++) {
5957      let changed = false;
5958
5959      expr = expr.replace(/([A-Za-z_]\w*)\s*\(([^()]|\([^()]*\))*\)/g, (match, name) =>
     {
5960        const tpl = templatesMap[name];
5961        if (!tpl) return match;
5962
5963        // вытащим аргументы вручную: name(....)
5964        const open = match.indexOf('(');
5965        const close = match.lastIndexOf(')');
5966        const inside = match.slice(open + 1, close);
5967
5968        const args = splitArgsTopLevel(inside);
5969        const formal = tpl.args || [];
5970        let body = String(tpl.body || '0');
5971
5972        // если количество аргументов не совпало — не трогаем (лучше так, чем сломать)
5973        if (args.length !== formal.length) return match;
5974
5975        formal.forEach((f, i) => {
5976          const re = new RegExp(`\\b${f}\\b`, 'g');
5977          body = body.replace(re, `(${args[i]})`);
5978        });
5979
5980        changed = true;
5981        return `(${body})`;
5982      });
5983
5984      if (!changed) break;
5985    }
5986
5987    return expr;
5988  }
5989
5990  viewport.js
5991
5992  /**
5993   * Модуль управления viewport (масштабирование и перемещение)
5994   */
5995
5996  const Viewport = {
5997      /**
5998       * Инициализация viewport
5999       */
6000      init() {
6001          this.setupZoomControls();
6002          this.setupPanning();
6003          this.setupMouseWheel();
```

```
6004          this.setupMinimap();
6005          this.setupCursorPosition();
6006          this.updateTransform();
6007          const container = document.getElementById('workspace-container');
6008          const rect = container.getBoundingClientRect();
6009          AppState.viewport.panX = 100; // немного отступить от левого края
6010          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
      AppState.viewport.zoom;
6011          this.updateTransform();
6012      },
6013
6014      /**
6015       * Настройка кнопок масштабирования
6016       */
6017      setupZoomControls() {
6018          document.getElementById('btn-zoom-in').addEventListener('click', () => {
6019              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
6020          });
6021
6022          document.getElementById('btn-zoom-out').addEventListener('click', () => {
6023              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
6024          });
6025
6026          document.getElementById('btn-zoom-reset').addEventListener('click', () => {
6027              this.setZoom(1);
6028              this.setPan(0, 0);
6029          });
6030
6031          document.getElementById('btn-zoom-fit').addEventListener('click', () => {
6032              this.fitToContent();
6033          });
6034      },
6035
6036      /**
6037       * Настройка перемещения (pan)
6038       */
6039      setupPanning() {
6040          const container = document.getElementById('workspace-container');
6041
6042          container.addEventListener('mousedown', (e) => {
6043              // Средняя кнопка мыши или пробел + левая кнопка
6044              if (e.button === 1 || (e.button === 0 && e.target === container)) {
6045                  e.preventDefault();
6046                  AppState.viewport.isPanning = true;
6047                  AppState.viewport.lastMouseX = e.clientX;
6048                  AppState.viewport.lastMouseY = e.clientY;
6049                  container.style.cursor = 'grabbing';
6050              }
6051          });
6052
6053          document.addEventListener('mousemove', (e) => {
6054              if (AppState.viewport.isPanning) {
6055                  const dx = e.clientX - AppState.viewport.lastMouseX;
6056                  const dy = e.clientY - AppState.viewport.lastMouseY;
6057
6058                  this.setPan(
6059                      AppState.viewport.panX + dx,
6060                      AppState.viewport.panY + dy
6061                  );
6062
6063                  AppState.viewport.lastMouseX = e.clientX;
6064                  AppState.viewport.lastMouseY = e.clientY;
6065              }
6066          });
6067
```

```
6068          document.addEventListener('mouseup', (e) => {
6069              if (AppState.viewport.isPanning) {
6070                  AppState.viewport.isPanning = false;
6071                  document.getElementById('workspace-container').style.cursor = '';
6072              }
6073          });
6074
6075          // Клавиша пробел для режима перемещения
6076          document.addEventListener('keydown', (e) => {
6077              if (e.code === 'Space' && !e.repeat) {
6078                  document.getElementById('workspace-container').style.cursor = 'grab';
6079              }
6080          });
6081
6082          document.addEventListener('keyup', (e) => {
6083              if (e.code === 'Space') {
6084                  document.getElementById('workspace-container').style.cursor = '';
6085              }
6086          });
6087      },
6088
6089      /**
6090       * Настройка масштабирования колесом мыши
6091       */
6092      setupMouseWheel() {
6093          const container = document.getElementById('workspace-container');
6094
6095          container.addEventListener('wheel', (e) => {
6096              e.preventDefault();
6097
6098              const rect = container.getBoundingClientRect();
6099              const mouseX = e.clientX - rect.left;
6100              const mouseY = e.clientY - rect.top;
6101
6102              // Позиция мыши на холсте до масштабирования
6103              const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6104              const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6105
6106              // Новый масштаб
6107              const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
      VIEWPORT_CONFIG.zoomStep;
6108              const newZoom = Math.max(
6109                  VIEWPORT_CONFIG.minZoom,
6110                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
6111              );
6112
6113              // Корректируем pan, чтобы точка под курсором осталась на месте
6114              const newPanX = mouseX - canvasPosBeforeX * newZoom;
6115              const newPanY = mouseY - canvasPosBeforeY * newZoom;
6116
6117              AppState.viewport.zoom = newZoom;
6118              AppState.viewport.panX = newPanX;
6119              AppState.viewport.panY = newPanY;
6120
6121              this.updateTransform();
6122          }, { passive: false });
6123      },
6124
6125      /**
6126       * Установить масштаб
6127       */
6128      setZoom(zoom) {
6129          const container = document.getElementById('workspace-container');
```

```
6130          const rect = container.getBoundingClientRect();
6131
6132          // Центр экрана
6133          const centerX = rect.width / 2;
6134          const centerY = rect.height / 2;
6135
6136          // Позиция центра на холсте
6137          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
6138          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
6139
6140          // Новый масштаб
6141          const newZoom = Math.max(
6142              VIEWPORT_CONFIG.minZoom,
6143              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
6144          );
6145
6146          // Корректируем pan
6147          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
6148          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
6149          AppState.viewport.zoom = newZoom;
6150
6151          this.updateTransform();
6152      },
6153
6154      /**
6155       * Установить смещение
6156       */
6157      setPan(x, y) {
6158          AppState.viewport.panX = x;
6159          AppState.viewport.panY = y;
6160          this.updateTransform();
6161      },
6162
6163      /**
6164       * Вписать содержимое в экран
6165       */
6166      fitToContent() {
6167          const elements = Object.values(AppState.elements);
6168          if (elements.length === 0) {
6169              this.setZoom(1);
6170              this.setPan(0, 0);
6171              return;
6172          }
6173
6174          // Находим границы содержимого
6175          let minX = Infinity, minY = Infinity;
6176          let maxX = -Infinity, maxY = -Infinity;
6177
6178          elements.forEach(elem => {
6179              minX = Math.min(minX, elem.x);
6180              minY = Math.min(minY, elem.y);
6181              maxX = Math.max(maxX, elem.x + elem.width);
6182              maxY = Math.max(maxY, elem.y + elem.height);
6183          });
6184
6185          const contentWidth = maxX - minX;
6186          const contentHeight = maxY - minY;
6187
6188          const container = document.getElementById('workspace-container');
6189          const rect = container.getBoundingClientRect();
6190
6191          const padding = 50;
6192          const availableWidth = rect.width - padding * 2;
```

```
6193          const availableHeight = rect.height - padding * 2;
6194
6195          const zoomX = availableWidth / contentWidth;
6196          const zoomY = availableHeight / contentHeight;
6197          const newZoom = Math.min(zoomX, zoomY, 1);
6198
6199          AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
6200          AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
      (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
6201          AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
      (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
6202
6203          this.updateTransform();
6204      },
6205
6206      /**
6207       * Обновить трансформацию
6208       */
6209      updateTransform() {
6210          const workspace = document.getElementById('workspace');
6211          const svg = document.getElementById('connections-svg');
6212
6213          const transform = `translate(${AppState.viewport.panX}px, $
      {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
6214
6215          workspace.style.transform = transform;
6216          svg.style.transform = transform;
6217
6218          // Обновляем отображение масштаба
6219          document.getElementById('zoom-level').textContent = `$
      {Math.round(AppState.viewport.zoom * 100)}%`;
6220
6221          // Обновляем мини-карту
6222          this.updateMinimap();
6223      },
6224
6225      /**
6226       * Настройка мини-карты
6227       */
6228      setupMinimap() {
6229          const minimap = document.getElementById('minimap');
6230          const canvas = document.getElementById('minimap-canvas');
6231
6232          canvas.width = MINIMAP_CONFIG.width;
6233          canvas.height = MINIMAP_CONFIG.height;
6234
6235          // Клик по мини-карте для перемещения
6236          minimap.addEventListener('click', (e) => {
6237              const rect = minimap.getBoundingClientRect();
6238              const x = e.clientX - rect.left;
6239              const y = e.clientY - rect.top;
6240
6241              this.navigateToMinimapPosition(x, y);
6242          });
6243      },
6244
6245      /**
6246       * Обновить мини-карту
6247       */
6248      updateMinimap() {
6249          const canvas = document.getElementById('minimap-canvas');
6250          const ctx = canvas.getContext('2d');
6251          const viewportEl = document.getElementById('minimap-viewport');
6252
6253          // Очищаем
```

```
6254          ctx.fillStyle = '#0a0a1a';
6255          ctx.fillRect(0, 0, canvas.width, canvas.height);
6256
6257          // Масштаб мини-карты
6258          const scale = Math.min(
6259              canvas.width / VIEWPORT_CONFIG.canvasWidth,
6260              canvas.height / VIEWPORT_CONFIG.canvasHeight
6261          );
6262
6263          // Рисуем элементы
6264          Object.values(AppState.elements).forEach(elem => {
6265              const x = elem.x * scale;
6266              const y = elem.y * scale;
6267              const w = Math.max(elem.width * scale, 2);
6268              const h = Math.max(elem.height * scale, 2);
6269
6270              ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
6271              ctx.fillRect(x, y, w, h);
6272          });
6273
6274          // Рисуем viewport
6275          const container = document.getElementById('workspace-container');
6276          const rect = container.getBoundingClientRect();
6277
6278          const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
6279          const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
6280          const vpW = (rect.width / AppState.viewport.zoom) * scale;
6281          const vpH = (rect.height / AppState.viewport.zoom) * scale;
6282
6283          viewportEl.style.left = `${vpX}px`;
6284          viewportEl.style.top = `${vpY}px`;
6285          viewportEl.style.width = `${vpW}px`;
6286          viewportEl.style.height = `${vpH}px`;
6287      },
6288
6289      /**
6290       * Перейти к позиции на мини-карте
6291       */
6292      navigateToMinimapPosition(minimapX, minimapY) {
6293          const canvas = document.getElementById('minimap-canvas');
6294          const container = document.getElementById('workspace-container');
6295          const rect = container.getBoundingClientRect();
6296
6297          const scale = Math.min(
6298              canvas.width / VIEWPORT_CONFIG.canvasWidth,
6299              canvas.height / VIEWPORT_CONFIG.canvasHeight
6300          );
6301
6302          const canvasX = minimapX / scale;
6303          const canvasY = minimapY / scale;
6304
6305          // Центрируем viewport на этой точке
6306          AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
6307          AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
6308
6309          this.updateTransform();
6310      },
6311
6312      /**
6313       * Отслеживание позиции курсора
6314       */
6315      setupCursorPosition() {
6316          const container = document.getElementById('workspace-container');
6317
6318          container.addEventListener('mousemove', (e) => {
```

```
            const pos = screenToCanvas(e.clientX, e.clientY);
            document.getElementById('cursor-pos').textContent =
                `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
        });
    }
};

styles.css

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: #1a1a2e;
    color: #eee;
    overflow: hidden;
}

#app {
    display: flex;
    flex-direction: column;
    height: 100vh;
}

/* ====== МЕНЮ ====== */
#menu {
    background: #16213e;
    padding: 10px 20px;
    display: flex;
    gap: 10px;
    align-items: center;
    border-bottom: 2px solid #0f3460;
    z-index: 100;
    flex-wrap: wrap;
}

.menu-btn {
    background: #0f3460;
    color: #eee;
    border: none;
    padding: 8px 16px;
    border-radius: 5px;
    cursor: pointer;
    transition: background 0.3s;
    font-size: 13px;
}

.menu-btn:hover {
    background: #e94560;
}

.menu-separator {
    width: 1px;
    height: 30px;
    background: #0f3460;
    margin: 0 10px;
}

.zoom-controls {
    display: flex;
    align-items: center;
```

```
    gap: 8px;
    background: #0a0a1a;
    padding: 5px 10px;
    border-radius: 5px;
}

.zoom-btn {
    width: 30px;
    height: 30px;
    padding: 0;
    font-size: 18px;
    font-weight: bold;
}

#zoom-level {
    min-width: 50px;
    text-align: center;
    font-size: 12px;
    color: #aaa;
}

/* ====== ОСНОВНАЯ ОБЛАСТЬ ====== */
#main {
    display: flex;
    flex: 1;
    overflow: hidden;
}

/* ====== ПАЛИТРА ====== */
#palette {
    width: 200px;
    background: #16213e;
    padding: 15px;
    border-right: 2px solid #0f3460;
    overflow-y: auto;
    z-index: 10;
    flex-shrink: 0;
}

#palette h3 {
    margin-bottom: 15px;
    color: #e94560;
    text-align: center;
    font-size: 14px;
}

.palette-section {
    margin-bottom: 15px;
}

.palette-section-title {
    font-size: 11px;
    color: #888;
    margin-bottom: 8px;
    padding-bottom: 3px;
    border-bottom: 1px solid #333;
}

.palette-item {
    background: #0f3460;
    padding: 8px;
    margin-bottom: 6px;
    border-radius: 8px;
    cursor: grab;
    text-align: center;
```

```
6449      transition: all 0.3s;
6450      border: 2px solid transparent;
6451      user-select: none;
6452  }
6453
6454  .palette-item:hover {
6455      border-color: #e94560;
6456      transform: scale(1.02);
6457  }
6458
6459  .palette-item:active {
6460      cursor: grabbing;
6461  }
6462
6463  .palette-item svg {
6464      width: 50px;
6465      height: 32px;
6466      margin-bottom: 2px;
6467      pointer-events: none;
6468  }
6469
6470  .palette-item-name {
6471      font-size: 10px;
6472      color: #aaa;
6473      pointer-events: none;
6474  }
6475
6476  .type-legend {
6477      margin-top: 15px;
6478      padding-top: 10px;
6479      border-top: 1px solid #333;
6480      font-size: 10px;
6481  }
6482
6483  .type-legend-item {
6484      display: flex;
6485      align-items: center;
6486      gap: 8px;
6487      margin-bottom: 5px;
6488  }
6489
6490  .type-legend-dot {
6491      width: 12px;
6492      height: 12px;
6493      border-radius: 50%;
6494      border: 2px solid #fff;
6495  }
6496  .type-legend-dot.logic { background: #a855f7; }
6497  .type-legend-dot.number { background: #3b82f6; }
6498
6499  /* ====== РАБОЧАЯ ОБЛАСТЬ ====== */
6500  #workspace-container {
6501      flex: 1;
6502      position: relative;
6503      overflow: hidden;
6504      background-color: #0a0a1a;
6505      background-image:
6506          linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
6507          linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);
6508      background-size: 25px 25px;
6509  }
6510
6511  #workspace {
6512      position: absolute;
6513      transform-origin: 0 0;
```

```
    width: 5000px;
    height: 5000px;
}

#connections-svg {
    position: absolute;
    transform-origin: 0 0;
    pointer-events: none;
    z-index: 5;
    width: 5000px;
    height: 5000px;
}

#connections-svg path {
    pointer-events: stroke;
}

/* ====== ЭЛЕМЕНТЫ ====== */
.element {
    position: absolute;
    background: #0f3460;
    border: 2px solid #4a90d9;
    border-radius: 8px;
    cursor: move;
    user-select: none;
    z-index: 10;
    display: flex;
    flex-direction: column;
}

.element.selected {
    border-color: #e94560;
    box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
}

.element-header {
    background: #4a90d9;
    padding: 5px 10px;
    border-radius: 5px 5px 0 0;
    font-size: 11px;
    font-weight: bold;
    text-align: center;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
}

.element-body {
    padding: 10px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex: 1;
    gap: 8px;
}

.element-symbol {
    font-size: 16px;
    font-weight: bold;
    flex: 1;
    text-align: center;
    padding: 0 5px;
    word-break: break-all;
    color: #eee;
}
```

```
6579
6580  /* ====== ПОРТЫ ====== */
6581  .ports-left, .ports-right {
6582      display: flex;
6583      flex-direction: column;
6584      justify-content: space-around;
6585      gap: 10px;
6586      height: 100%;
6587  }
6588
6589  .port {
6590      width: 14px;
6591      height: 14px;
6592      border-radius: 50%;
6593      border: 2px solid #fff;
6594      cursor: crosshair;
6595      transition: all 0.2s;
6596      position: relative;
6597      flex-shrink: 0;
6598  }
6599
6600  .port:hover { transform: scale(1.3); }
6601  .port.input { margin-left: -8px; }
6602  .port.output { margin-right: -8px; }
6603  .port.connected { background: #4ade80; }
6604
6605  /* Типы портов */
6606  .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
6607  .port.logic-port:hover { background: #c084fc; }
6608  .port.logic-port.connected { background: #7c3aed; }
6609
6610  .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
6611  .port.number-port:hover { background: #60a5fa; }
6612  .port.number-port.connected { background: #2563eb; }
6613
6614  .port.any-port { background: #6b7280; border-color: #d1d5db; }
6615  .port.any-port:hover { background: #9ca3af; }
6616  .port.any-port.connected { background: #4b5563; }
6617
6618  .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !
      important; }
6619  .port.output.no-port { background: #f87171 !important; border-color: #fecaca !
      important; }
6620
6621  .port.incompatible { opacity: 0.3; cursor: not-allowed; }
6622  .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
6623
6624  /* ====== RESIZE HANDLES ====== */
6625  .resize-handle {
6626      position: absolute;
6627      width: 12px;
6628      height: 12px;
6629      background: #e94560;
6630      border: 1px solid #fff;
6631      border-radius: 3px;
6632      z-index: 20;
6633      opacity: 0;
6634      transition: opacity 0.2s;
6635  }
6636  .element.selected .resize-handle { opacity: 0.8; }
6637  .resize-handle:hover { opacity: 1; }
6638  .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
6639  .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor:
      ew-resize; }
6640  .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%);
```

```
      cursor: ns-resize; }
6641
6642
6643  /* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
6644  .element.input-signal {
6645      background: transparent;
6646      border: none;
6647  }
6648
6649  .element.input-signal .element-header {
6650      display: none; /* У трапеции нет заголовка */
6651  }
6652
6653  .element.input-signal .element-body {
6654      padding: 0;
6655      background: #0f3460;
6656      border: 2px solid #4a90d9;
6657      clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
6658      display: flex;
6659      justify-content: space-between;
6660      align-items: center;
6661      padding-left: 15px;
6662      padding-right: 25px;
6663  }
6664
6665  .element.input-signal .element-symbol {
6666      text-align: left;
6667      color: #eee;
6668  }
6669
6670  .element.input-signal.selected .element-body {
6671      border-color: #e94560;
6672  }
6673
6674  /* ====== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ====== */
6675  .element.output {
6676      background: rgba(16, 185, 129, 0.1);
6677      border: 2px dashed #10b981;
6678  }
6679
6680  .element.output .element-header {
6681      display: none; /* У выхода нет заголовка */
6682  }
6683
6684  .element.output .element-body {
6685      padding-left: 20px;
6686  }
6687
6688  .element.output .element-symbol {
6689      color: #10b981;
6690      font-size: 14px;
6691  }
6692
6693  .element.output.selected {
6694      border-color: #e94560;
6695      border-style: dashed;
6696  }
6697
6698
6699  /* Formula condition port */
6700  /* Универсальный стиль для технического порта (сверху) */
6701  .element.has-condition-port {
6702      margin-top: 30px; /* Даем место порту над элементом */
6703  }
6704
```

```
.condition-port-wrapper {
    position: absolute;
    top: -28px;
    left: 50%;
    transform: translateX(-50%);
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 4px;
    pointer-events: none;
    z-index: 21;
}

.condition-port-label {
    font-size: 10px;
    color: #f59e0b;
    font-weight: 600;
    white-space: nowrap;
}

.port.condition-port {
    pointer-events: auto;
    width: 16px;
    height: 16px;
    border-radius: 50%;
    border: 2px solid #f59e0b;
    background: #fff7ed;
    margin: 0; /* Сбрасываем лишние отступы */
}
.element.formula .condition-port:hover { background: #fde68a; }


/* ====== СОЕДИНЕНИЯ ====== */
.connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #4a90d9;
    stroke-width: 2.5;
}
.connection:hover {
    stroke: #e94560;
    stroke-width: 4;
}

.connection.logic-conn { stroke: #a855f7; }
.connection.numeric-conn { stroke: #3b82f6; }
.connection.any-conn { stroke: #6b7280; }
.connection.true-conn { stroke: #4ade80; }
.connection.false-conn { stroke: #f87171; }

.connection.yes-conn { stroke: #4ade80; }
.connection.no-conn { stroke: #f87171; }

.temp-connection {
    fill: none !important;  /* ← добавляем !important */
    stroke: #e94560;
    stroke-width: 2;
    stroke-dasharray: 5, 5;
}
.temp-connection.invalid { stroke: #ef4444; }

/* ====== ПРОЧЕЕ ====== */
.drag-preview {
    position: fixed;
    pointer-events: none;
    opacity: 0.8;
```

```
6770      z-index: 1000;
6771      background: #0f3460;
6772      border: 2px solid #e94560;
6773      border-radius: 8px;
6774      padding: 10px 15px;
6775      color: #fff;
6776      font-size: 12px;
6777  }
6778
6779  #minimap {
6780      position: absolute;
6781      bottom: 20px;
6782      right: 20px;
6783      width: 200px;
6784      height: 150px;
6785      background: #16213e;
6786      border: 2px solid #0f3460;
6787      border-radius: 8px;
6788      overflow: hidden;
6789      z-index: 50;
6790  }
6791
6792  #minimap-canvas { width: 100%; height: 100%; }
6793  #minimap-viewport {
6794      position: absolute;
6795      border: 2px solid #e94560;
6796      background: rgba(233, 69, 96, 0.2);
6797      pointer-events: none;
6798  }
6799
6800  #viewport-info {
6801      position: absolute;
6802      bottom: 20px;
6803      left: 20px;
6804      background: rgba(22, 33, 62, 0.9);
6805      padding: 8px 12px;
6806      border-radius: 5px;
6807      font-size: 11px;
6808      color: #888;
6809      z-index: 50;
6810      display: flex;
6811      gap: 15px;
6812  }
6813  #selection-info { color: #e94560; }
6814
6815  #modal-overlay, .modal-overlay-class {
6816      display: none;
6817      position: fixed;
6818      top: 0; left: 0;
6819      width: 100%; height: 100%;
6820      background: rgba(0, 0, 0, 0.7);
6821      z-index: 1000;
6822      justify-content: center;
6823      align-items: center;
6824  }
6825
6826  #modal, .modal-class {
6827      background: #16213e;
6828      border-radius: 10px;
6829      padding: 20px;
6830      min-width: 400px;
6831      max-width: 600px;
6832      max-height: 80vh;
6833      overflow-y: auto;
6834      border: 2px solid #0f3460;
```

```
}

#modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
.modal-row { margin-bottom: 15px; }
.modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
.modal-row input, .modal-row select, .modal-row textarea {
    width: 100%;
    padding: 10px;
    background: #0f3460;
    border: 1px solid #4a90d9;
    border-radius: 5px;
    color: #eee;
    font-size: 14px;
}
.modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline: none; border-color: #e94560; }
.modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
.signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-radius: 5px; padding: 5px; margin-top: 5px; }
.signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size: 12px; }
.signal-item:hover { background: #4a90d9; }
.modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
.modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer; font-size: 14px; transition: background 0.3s; }
.modal-btn.save { background: #4ade80; color: #000; }
.modal-btn.save:hover { background: #22c55e; }
.modal-btn.cancel { background: #6b7280; color: #fff; }
.modal-btn.cancel:hover { background: #4b5563; }

#context-menu {
    display: none;
    position: fixed;
    background: #16213e;
    border: 1px solid #0f3460;
    border-radius: 5px;
    padding: 5px 0;
    z-index: 1001;
    min-width: 150px;
    box-shadow: 0 5px 20px rgba(0,0,0,0.3);
}
.context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition: background 0.2s; }
.context-item:hover { background: #0f3460; }

#file-input { display: none; }

.project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
.project-type-btn { flex: 1; padding: 15px; background: #0f3460; border: 2px solid #4a90d9; border-radius: 8px; color: #eee; cursor: pointer; text-align: center; transition: all 0.3s; }
.project-type-btn:hover { border-color: #e94560; }
.project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
.project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
.project-type-btn .type-name { font-weight: bold; }
.project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }

.conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-radius: 8px; margin-top: 10px; }
.conditional-fields.visible { display: block; }

::-webkit-scrollbar { width: 8px; height: 8px; }
::-webkit-scrollbar-track { background: #0a0a1a; }
::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
```

```
::-webkit-scrollbar-thumb:hover { background: #e94560; }

/* Стили для выходов */
.output-btn { position: relative; }
.output-counter { display: inline-block; background: #e94560; color: white; font-size: 11px; font-weight: bold; padding: 2px 6px; border-radius: 10px; margin-left: 5px; min-width: 18px; text-align: center; }
.output-counter:empty, .output-counter[style*="display: none"] { display: none; }
.element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
.element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important; border-color: #fbbf24 !important; }
.port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation: pulse-output 1.5s infinite; }
@keyframes pulse-output {
    0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
    50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
}

.outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-height: 250px; overflow-y: auto; }
.outputs-section { margin-bottom: 15px; }
.outputs-section:last-child { margin-bottom: 0; }
.outputs-section-title { color: #10b981; font-weight: bold; font-size: 13px; margin-bottom: 10px; padding-bottom: 5px; border-bottom: 1px solid #333; display: flex; align-items: center; gap: 8px; }
.outputs-section-title .section-icon { font-size: 16px; }
.outputs-list { display: flex; flex-direction: column; gap: 5px; }
.output-item { display: flex; align-items: center; gap: 10px; padding: 8px 12px; background: rgba(16, 185, 129, 0.1); border: 1px solid rgba(16, 185, 129, 0.3); border-radius: 5px; cursor: pointer; transition: all 0.2s; }
.output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981; transform: translateX(5px); }
.output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59, 130, 246, 0.3); }
.output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color: #3b82f6; }
.output-icon { font-size: 14px; }
.output-name { font-weight: bold; color: #eee; }
.output-port { color: #888; font-size: 12px; margin-left: auto; }
.no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
.outputs-hint { margin-top: 10px; padding: 10px; background: rgba(59, 130, 246, 0.1); border-radius: 5px; font-size: 12px; color: #888; line-height: 1.4; }
.element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-color: rgba(240,80,80,0.8) !important; }
.element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-color: rgba(250,200,30,0.8) !important; }
/* TRUE/FALSE порты (для сепаратора) */
.port.true-port {
    background: #4ade80 !important;
    border-color: #bbf7d0 !important;
}
.port.true-port:hover {
    background: #22c55e !important;
}
.port.true-port.connected {
    background: #16a34a !important;
}

.port.false-port {
    background: #f87171 !important;
    border-color: #fecaca !important;
}
.port.false-port:hover {
    background: #ef4444 !important;
}
```

```
6941  .port.false-port.connected {
6942      background: #dc2626 !important;
6943  }
6944
6945  /* Сепаратор стиль */
6946  .element.separator {
6947      background: #0f3460;
6948      border: 2px solid #f59e0b;
6949  }
6950
6951  .element.separator.selected {
6952      border-color: #e94560;
6953      box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);
6954  }
6955
6956  /* === Выделение рамкой === */
6957  #selection-rect {
6958    position: absolute;
6959    border: 1px dashed #e94560;
6960    background: rgba(233, 69, 96, 0.1);
6961    pointer-events: none;
6962    display: none;
6963    z-index: 200;
6964  }
6965
6966  /* === Кастомный элемент “Группа” === */
6967  .element.group {
6968    background: rgba(107, 114, 128, 0.12);
6969    border: 2px dashed #6b7280;
6970    border-radius: 8px;
6971    position: absolute;
6972    z-index: 1;              /* ниже обычных элементов (у них z-index: 10) */
6973  }
6974
6975  .element.group .group-title {
6976    pointer-events: auto;
6977  }
6978
6979  .group-title {
6980    position: absolute;
6981    top: -20px;
6982    left: 5px;
6983    font-size: 11px;
6984    color: #ccc;
6985    background: #16213e;
6986    padding: 2px 6px;
6987    border-radius: 4px;
6988    pointer-events: auto; /* можно кликнуть для выбора */
6989  }
6990
6991  .modal.hidden { display: none; }
6992  .modal { position: fixed; inset: 0; display: flex; align-items: center; justify-
      content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
6993  .modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px;
      max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow:
      hidden; }
6994  .modal__content--wide { width: 800px; }
6995  .modal__title { margin: 0; }
6996
6997  .project-list__toolbar { display: flex; gap: 12px; }
6998  .project-list__toolbar input { flex: 1; padding: 6px 10px; }
6999  .project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd;
      border-radius: 6px; }
7000  .project-list__table { width: 100%; border-collapse: collapse; }
7001  .project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom:
```

```
      1px solid #eee; }
7002  .project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
7003  .project-list__table tbody tr:hover { background: #f0f6ff; }
7004  .project-list__empty { text-align: center; color: #888; padding: 16px; }
7005  .modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
7006  .project-list__table th,
7007  .project-list__table td {
7008    color: #111;          /* насыщённый чёрный текст */
7009    padding: 8px 12px;
7010    border-bottom: 1px solid #eee;
7011  }
7012  .modal__content--wide {
7013    width: 860px;
7014    max-height: 90vh;    /* занимает 90% экрана */
7015  }
7016
7017  .project-list__table-container {
7018    flex: 1;
7019    overflow: auto;
7020    border: 1px solid #ddd;
7021    border-radius: 6px;
7022    max-height: 60vh;     /* много строк */
7023  }
7024
7025  .element-comment {
7026    padding: 6px 10px 10px;
7027    font-size: 11px;
7028    color: #cbd5e1;
7029    opacity: 0.9;
7030    border-top: 1px solid rgba(255,255,255,0.08);
7031    white-space: pre-wrap;
7032    word-break: break-word;
7033  }
7034
7035  .element-comment:empty { display: none; }
```