index.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Редактор логических схем</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div id="app">
        <div id="menu">
            <button class="menu-btn" id="btn-new">📄 Новый</button>
            <button class="menu-btn" id="btn-save">💾 Сохранить</button>
            <button class="menu-btn" id="btn-load">📂 Загрузить</button>
            <button class="menu-btn" id="btn-generate-code">🧩 Код</button>
            <button class="menu-btn" id="btn-project-settings">⚙️ Свойства проекта</button>
            <div class="menu-separator"></div>
            <div class="zoom-controls">
                <button class="menu-btn zoom-btn" id="btn-zoom-out">−</button>
                <span id="zoom-level">100%</span>
                <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
                <button class="menu-btn" id="btn-zoom-fit">⊡ Вписать</button>
                <button class="menu-btn" id="btn-zoom-reset">1:1</button>
            </div>
            <input type="file" id="file-input" accept=".json">
        </div>

        <div id="main">
            <div id="palette">
                <h3>📦 Элементы</h3>

                <div class="palette-section">
                    <div class="palette-section-title">ВХОДЫ</div>

                    <div class="palette-item" data-type="input-signal">
                        <svg viewBox="0 0 60 40">
                            <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460" stroke="#4a90d9" stroke-width="2"/>
                            <text x="12" y="24" fill="#eee" font-size="10">IN</text>
                        </svg>
                        <div class="palette-item-name">Входной сигнал</div>
                    </div>
                </div>
                <div class="palette-section">
                    <div class="palette-section-title">ВЫХОДЫ</div>

                    <div class="palette-item" data-type="output">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="6" fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
                            <text x="12" y="24" fill="#10b981" font-size="9">ВЫХОД</text>
                        </svg>
                        <div class="palette-item-name">Выход</div>
                    </div>
                </div>

                <div class="palette-section">
                    <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>

                    <div class="palette-item" data-type="and">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5"
```

```
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
 62                             <text x="22" y="25" fill="#eee" font-size="12" font-
    weight="bold">И</text>
 63                         </svg>
 64                         <div class="palette-item-name">И (AND)</div>
 65                     </div>
 66
 67                     <div class="palette-item" data-type="or">
 68                         <svg viewBox="0 0 60 40">
 69                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
 70                             <text x="12" y="25" fill="#eee" font-size="11" font-
    weight="bold">ИЛИ</text>
 71                         </svg>
 72                         <div class="palette-item-name">ИЛИ (OR)</div>
 73                     </div>
 74
 75                     <div class="palette-item" data-type="not">
 76                         <svg viewBox="0 0 60 40">
 77                             <rect x="5" y="5" width="50" height="30" rx="5"
    fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
 78                             <text x="12" y="25" fill="#eee" font-size="11" font-
    weight="bold">НЕТ</text>
 79                         </svg>
 80                         <div class="palette-item-name">НЕТ (NOT)</div>
 81                     </div>
 82                 </div>
 83
 84                 <div class="palette-section">
 85                     <div class="palette-section-title">СРАВНЕНИЕ</div>
 86
 87                     <div class="palette-item" data-type="if">
 88                         <svg viewBox="0 0 60 40">
 89                             <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460"
    stroke="#e94560" stroke-width="2"/>
 90                             <text x="14" y="24" fill="#eee" font-size="9" font-
    weight="bold">ЕСЛИ</text>
 91                         </svg>
 92                         <div class="palette-item-name">ЕСЛИ (IF)</div>
 93                     </div>
 94                 </div>
 95
 96                 <div class="palette-section">
 97                     <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>
 98
 99                     <div class="palette-item" data-type="separator">
100                         <svg viewBox="0 0 60 40">
101                             <rect x="5" y="8" width="50" height="24" rx="3"
    fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
102                             <text x="8" y="25" fill="#f59e0b" font-size="10" font-
    weight="bold">✓/✗</text>
103                         </svg>
104                         <div class="palette-item-name">Сепаратор</div>
105                     </div>
106                 </div>
107
108                 <div class="palette-section">
109                     <div class="palette-section-title">ЗНАЧЕНИЯ</div>
110
111                     <div class="palette-item" data-type="const">
112                         <svg viewBox="0 0 60 40">
113                             <rect x="10" y="8" width="40" height="24" rx="3"
    fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
114                             <text x="24" y="25" fill="#3b82f6" font-size="14" font-
    weight="bold">C</text>
```

```
                        </svg>
                        <div class="palette-item-name">Константа</div>
                    </div>

                    <div class="palette-item" data-type="formula">
                        <svg viewBox="0 0 60 40">
                            <rect x="5" y="5" width="50" height="30" rx="5"
fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
                            <text x="12" y="25" fill="#f59e0b" font-size="11" font-
weight="bold">f(x)</text>
                        </svg>
                        <div class="palette-item-name">Формула</div>
                    </div>
                </div>

                <div class="type-legend">
                    <div class="type-legend-item">
                        <div class="type-legend-dot logic"></div>
                        <span>Логический</span>
                    </div>
                    <div class="type-legend-item">
                        <div class="type-legend-dot number"></div>
                        <span>Числовой</span>
                    </div>
                </div>
            </div>

            <div id="workspace-container">
                <svg id="connections-svg"></svg>
                <div id="workspace"></div>
                  <!-- Прямоугольник для выделения элементов -->
                <div id="selection-rect"></div>

                <!-- Мини-карта -->
                <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>
```

```
178             <div class="modal-buttons">
179                 <button class="modal-btn cancel" id="project-modal-cancel">Отмена</
    button>
180                 <button class="modal-btn save" id="project-modal-save">Сохранить</
    button>
181             </div>
182         </div>
183     </div>
184 
185     <div id="code-modal-overlay" class="modal-overlay-class">
186         <div id="code-modal" class="modal-class">
187             <h3>Сгенерированный код</h3>
188             <textarea id="code-output" style="width:100%; height:300px;"></textarea>
189             <div class="modal-buttons">
190                 <button class="modal-btn cancel" id="code-modal-close">Закрыть</
    button>
191             </div>
192         </div>
193     </div>
194 
195     <div id="context-menu">
196         <div class="context-item" id="ctx-properties">⚙️ Свойства</div>
197         <div class="context-item" id="ctx-delete">🗑️ Удалить</div>
198     </div>
199 
200    <!-- Модули JavaScript -->
201     <!-- Модули JavaScript -->
202     <script src="js/config.js"></script>
203     <script src="js/state.js"></script>
204     <script src="js/utils.js"></script>
205     <script src="js/viewport.js"></script>
206     <script src="js/elements.js"></script>
207     <script src="js/connections.js"></script>
208     <script src="js/outputs.js"></script>  <!-- ← Этот файл опционален теперь -->
209     <script src="js/modal.js"></script>
210     <script src="js/project.js"></script>
211     <script src="js/codegen_graph.js"></script>
212     <script src="js/codegen_optimizer.js"></script>
213     <script src="js/codegen.js"></script>
214     <script src="js/settings.js"></script>
215 
216     <script src="js/app.js"></script>
217 
218     <div id="modal-project-list" class="modal hidden">
219         <div class="modal__content modal__content--wide">
220             <h2 class="modal__title">Выбор проекта</h2>
221 
222             <div class="project-list__toolbar">
223             <input id="project-search" type="text" placeholder="Фильтр по имени или
    описанию…" />
224             <button id="project-refresh" class="btn btn-secondary">Обновить</button>
225             </div>
226 
227             <div class="project-list__table-container">
228             <table class="project-list__table">
229                 <thead>
230                 <tr>
231                     <th>Файл</th>
232                     <th>Tagname</th>
233                     <th>Description</th>
234                     <th>Тип</th>
235                 </tr>
236                 </thead>
237                 <tbody id="project-list-body">
238                 <tr><td colspan="4" class="project-list__empty">Загрузка…</td></tr>
```

```
                </tbody>
            </table>
            </div>

            <div class="modal__actions">
                <button id="project-cancel" class="btn btn-secondary">Отмена</button>
                <button id="project-load" class="btn btn-primary" disabled>Загрузить</
button>
            </div>
        </div>
    </div>

</body>
</html>

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;
    transition: all 0.3s;
    border: 2px solid transparent;
    user-select: none;
}

.palette-item:hover {
    border-color: #e94560;
    transform: scale(1.02);
}

.palette-item:active {
    cursor: grabbing;
}

.palette-item svg {
    width: 50px;
    height: 32px;
    margin-bottom: 2px;
    pointer-events: none;
}

.palette-item-name {
    font-size: 10px;
    color: #aaa;
    pointer-events: none;
}

.type-legend {
    margin-top: 15px;
    padding-top: 10px;
    border-top: 1px solid #333;
    font-size: 10px;
}

.type-legend-item {
    display: flex;
    align-items: center;
    gap: 8px;
    margin-bottom: 5px;
}

.type-legend-dot {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    border: 2px solid #fff;
}
.type-legend-dot.logic { background: #a855f7; }
.type-legend-dot.number { background: #3b82f6; }

/* ====== РАБОЧАЯ ОБЛАСТЬ ====== */
#workspace-container {
    flex: 1;
    position: relative;
    overflow: hidden;
    background-color: #0a0a1a;
    background-image:
        linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
```

```
        linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);
    background-size: 25px 25px;
}

#workspace {
    position: absolute;
    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;
}

/* ====== ПОРТЫ ====== */
.ports-left, .ports-right {
    display: flex;
    flex-direction: column;
    justify-content: space-around;
    gap: 10px;
    height: 100%;
}

.port {
    width: 14px;
    height: 14px;
    border-radius: 50%;
    border: 2px solid #fff;
    cursor: crosshair;
    transition: all 0.2s;
    position: relative;
    flex-shrink: 0;
}

.port:hover { transform: scale(1.3); }
.port.input { margin-left: -8px; }
.port.output { margin-right: -8px; }
.port.connected { background: #4ade80; }

/* Типы портов */
.port.logic-port { background: #a855f7; border-color: #e9d5ff; }
.port.logic-port:hover { background: #c084fc; }
.port.logic-port.connected { background: #7c3aed; }

.port.number-port { background: #3b82f6; border-color: #bfdbfe; }
.port.number-port:hover { background: #60a5fa; }
.port.number-port.connected { background: #2563eb; }

.port.any-port { background: #6b7280; border-color: #d1d5db; }
.port.any-port:hover { background: #9ca3af; }
.port.any-port.connected { background: #4b5563; }

.port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !important; }
.port.output.no-port { background: #f87171 !important; border-color: #fecaca !important; }

.port.incompatible { opacity: 0.3; cursor: not-allowed; }
.port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }

/* ====== RESIZE HANDLES ====== */
.resize-handle {
    position: absolute;
    width: 12px;
    height: 12px;
    background: #e94560;
    border: 1px solid #fff;
    border-radius: 3px;
    z-index: 20;
    opacity: 0;
    transition: opacity 0.2s;
```

```
}
.element.selected .resize-handle { opacity: 0.8; }
.resize-handle:hover { opacity: 1; }
.resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
.resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor: ew-resize; }
.resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%); cursor: ns-resize; }


/* ====== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ====== */
.element.input-signal {
    background: transparent;
    border: none;
}

.element.input-signal .element-header {
    display: none; /* У трапеции нет заголовка */
}

.element.input-signal .element-body {
    padding: 0;
    background: #0f3460;
    border: 2px solid #4a90d9;
    clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding-left: 15px;
    padding-right: 25px;
}

.element.input-signal .element-symbol {
    text-align: left;
    color: #eee;
}

.element.input-signal.selected .element-body {
    border-color: #e94560;
}

/* ====== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ====== */
.element.output {
    background: rgba(16, 185, 129, 0.1);
    border: 2px dashed #10b981;
}

.element.output .element-header {
    display: none; /* У выхода нет заголовка */
}

.element.output .element-body {
    padding-left: 20px;
}

.element.output .element-symbol {
    color: #10b981;
    font-size: 14px;
}

.element.output.selected {
    border-color: #e94560;
    border-style: dashed;
}

```

```
624
625  /* Formula condition port */
626  /* Универсальный стиль для технического порта (сверху) */
627  .element.has-condition-port {
628      margin-top: 30px; /* Даем место порту над элементом */
629  }
630
631  .condition-port-wrapper {
632      position: absolute;
633      top: -28px;
634      left: 50%;
635      transform: translateX(-50%);
636      display: flex;
637      flex-direction: column;
638      align-items: center;
639      gap: 4px;
640      pointer-events: none;
641      z-index: 21;
642  }
643
644  .condition-port-label {
645      font-size: 10px;
646      color: #f59e0b;
647      font-weight: 600;
648      white-space: nowrap;
649  }
650
651  .port.condition-port {
652      pointer-events: auto;
653      width: 16px;
654      height: 16px;
655      border-radius: 50%;
656      border: 2px solid #f59e0b;
657      background: #fff7ed;
658      margin: 0; /* Сбрасываем лишние отступы */
659  }
660  .element.formula .condition-port:hover { background: #fde68a; }
661
662
663  /* ====== СОЕДИНЕНИЯ ====== */
664  .connection {
665      fill: none !important;  /* ← добавляем !important */
666      stroke: #4a90d9;
667      stroke-width: 2.5;
668  }
669  .connection:hover {
670      stroke: #e94560;
671      stroke-width: 4;
672  }
673
674  .connection.logic-conn { stroke: #a855f7; }
675  .connection.numeric-conn { stroke: #3b82f6; }
676  .connection.any-conn { stroke: #6b7280; }
677  .connection.true-conn { stroke: #4ade80; }
678  .connection.false-conn { stroke: #f87171; }
679
680  .connection.yes-conn { stroke: #4ade80; }
681  .connection.no-conn { stroke: #f87171; }
682
683  .temp-connection {
684      fill: none !important;  /* ← добавляем !important */
685      stroke: #e94560;
686      stroke-width: 2;
687      stroke-dasharray: 5, 5;
688  }
```

```
.temp-connection.invalid { stroke: #ef4444; }

/* ====== ПРОЧЕЕ ====== */
.drag-preview {
    position: fixed;
    pointer-events: none;
    opacity: 0.8;
    z-index: 1000;
    background: #0f3460;
    border: 2px solid #e94560;
    border-radius: 8px;
    padding: 10px 15px;
    color: #fff;
    font-size: 12px;
}

#minimap {
    position: absolute;
    bottom: 20px;
    right: 20px;
    width: 200px;
    height: 150px;
    background: #16213e;
    border: 2px solid #0f3460;
    border-radius: 8px;
    overflow: hidden;
    z-index: 50;
}

#minimap-canvas { width: 100%; height: 100%; }
#minimap-viewport {
    position: absolute;
    border: 2px solid #e94560;
    background: rgba(233, 69, 96, 0.2);
    pointer-events: none;
}

#viewport-info {
    position: absolute;
    bottom: 20px;
    left: 20px;
    background: rgba(22, 33, 62, 0.9);
    padding: 8px 12px;
    border-radius: 5px;
    font-size: 11px;
    color: #888;
    z-index: 50;
    display: flex;
    gap: 15px;
}
#selection-info { color: #e94560; }

#modal-overlay, .modal-overlay-class {
    display: none;
    position: fixed;
    top: 0; left: 0;
    width: 100%; height: 100%;
    background: rgba(0, 0, 0, 0.7);
    z-index: 1000;
    justify-content: center;
    align-items: center;
}

#modal, .modal-class {
    background: #16213e;
```

```
754       border-radius: 10px;
755       padding: 20px;
756       min-width: 400px;
757       max-width: 600px;
758       max-height: 80vh;
759       overflow-y: auto;
760       border: 2px solid #0f3460;
761   }
762
763   #modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
764   .modal-row { margin-bottom: 15px; }
765   .modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
766   .modal-row input, .modal-row select, .modal-row textarea {
767       width: 100%;
768       padding: 10px;
769       background: #0f3460;
770       border: 1px solid #4a90d9;
771       border-radius: 5px;
772       color: #eee;
773       font-size: 14px;
774   }
775   .modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline:
      none; border-color: #e94560; }
776   .modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
777   .signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-
      radius: 5px; padding: 5px; margin-top: 5px; }
778   .signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size:
      12px; }
779   .signal-item:hover { background: #4a90d9; }
780   .modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top:
      20px; }
781   .modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer;
      font-size: 14px; transition: background 0.3s; }
782   .modal-btn.save { background: #4ade80; color: #000; }
783   .modal-btn.save:hover { background: #22c55e; }
784   .modal-btn.cancel { background: #6b7280; color: #fff; }
785   .modal-btn.cancel:hover { background: #4b5563; }
786
787   #context-menu {
788       display: none;
789       position: fixed;
790       background: #16213e;
791       border: 1px solid #0f3460;
792       border-radius: 5px;
793       padding: 5px 0;
794       z-index: 1001;
795       min-width: 150px;
796       box-shadow: 0 5px 20px rgba(0,0,0,0.3);
797   }
798   .context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition:
      background 0.2s; }
799   .context-item:hover { background: #0f3460; }
800
801   #file-input { display: none; }
802
803   .project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
804   .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; }
805   .project-type-btn:hover { border-color: #e94560; }
806   .project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
807   .project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
808   .project-type-btn .type-name { font-weight: bold; }
809   .project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }
810
```

```
.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;
}
.port.false-port.connected {
    background: #dc2626 !important;
}

/* Сепаратор стиль */
.element.separator {
    background: #0f3460;
    border: 2px solid #f59e0b;
}

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

/* === Выделение рамкой === */
#selection-rect {
  position: absolute;
  border: 1px dashed #e94560;
  background: rgba(233, 69, 96, 0.1);
  pointer-events: none;
  display: none;
  z-index: 200;
}

/* === Кастомный элемент “Группа” === */
.element.group {
  background: rgba(107, 114, 128, 0.15);
  border: 2px dashed #6b7280;
  border-radius: 8px;
  pointer-events: none; /* не мешает клику по внутренним элементам */
  position: absolute;
}

.group-title {
  position: absolute;
  top: -20px;
  left: 5px;
  font-size: 11px;
  color: #ccc;
  background: #16213e;
  padding: 2px 6px;
  border-radius: 4px;
  pointer-events: auto; /* можно кликнуть для выбора */
}

.modal.hidden { display: none; }
.modal { position: fixed; inset: 0; display: flex; align-items: center; justify-content: center; background: rgba(0,0,0,0.4); z-index: 1000; }
.modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px; max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow: hidden; }
.modal__content--wide { width: 800px; }
.modal__title { margin: 0; }

.project-list__toolbar { display: flex; gap: 12px; }
.project-list__toolbar input { flex: 1; padding: 6px 10px; }
.project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd;
```

```
    border-radius: 6px; }
.project-list__table { width: 100%; border-collapse: collapse; }
.project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom:
    1px solid #eee; }
.project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
.project-list__table tbody tr:hover { background: #f0f6ff; }
.project-list__empty { text-align: center; color: #888; padding: 16px; }
.modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
.project-list__table th,
.project-list__table td {
  color: #111;          /* насыщённый чёрный текст */
  padding: 8px 12px;
  border-bottom: 1px solid #eee;
}
.modal__content--wide {
  width: 860px;
  max-height: 90vh;    /* занимает 90% экрана */
}

.project-list__table-container {
  flex: 1;
  overflow: auto;
  border: 1px solid #ddd;
  border-radius: 6px;
  max-height: 60vh;     /* много строк */
}
app.js:
/**
 * Главный модуль приложения
 */

const App = {
    /**
     * Инициализация приложения
     */
    init() {
        //Settings.init().then(() => {
        //    // если хочешь — можно обновить UI (например, статус “Сигналы
    загружены”)
        //    console.log('Settings loaded, signals:', Settings.signals.length);
        //    }).catch(err => console.error(err));
        //console.log('signals loaded:', Settings.signals.slice(0, 5));
        this.setupPaletteDragDrop();
        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);
        }

        outputBtn.addEventListener('click', () => {
            Modal.showProjectPropertiesModal();
        });
    },

    /**
     * Настройка drag & drop из палитры
     */
    setupPaletteDragDrop() {
        document.querySelectorAll('.palette-item').forEach(item => {
            item.addEventListener('mousedown', (e) => {
                // Только левая кнопка мыши должна запускать drag из палитры
                if (e.button !== 0) return;
                e.preventDefault();

                AppState.isDraggingFromPalette = true;
                AppState.dragType = item.dataset.type;

                AppState.dragPreview = document.createElement('div');
                AppState.dragPreview.className = 'drag-preview';
                AppState.dragPreview.textContent =
```

```
      ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
1049                  AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1050                  AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1051                  document.body.appendChild(AppState.dragPreview);
1052              });
1053          });
1054      },
1055
1056      /**
1057       * Глобальные обработчики мыши
1058       */
1059  /**
1060   * Глобальные обработчики мыши
1061   */
1062  setupGlobalMouseHandlers() {
1063      document.addEventListener('mousemove', (e) => {
1064          if (AppState.isDraggingFromPalette && AppState.dragPreview) {
1065              AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1066              AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1067          }
1068          if (AppState.resizing) {
1069              Elements.handleResize(e);
1070              return;
1071          }
1072          if (AppState.draggingElement) {
1073              Elements.handleDrag(e);
1074          }
1075          if (AppState.tempLine && AppState.connectingFrom) {
1076              Connections.drawTempConnection(e);
1077          }
1078      });
1079
1080      document.addEventListener('mouseup', (e) => {
1081          if (AppState.resizing) {
1082              AppState.resizing = null;
1083              if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1084          }
1085
1086          if (AppState.isDraggingFromPalette) {
1087              try {
1088                  if (AppState.dragPreview) {
1089                      AppState.dragPreview.remove();
1090                      AppState.dragPreview = null;
1091                  }
1092
1093                  const container = document.getElementById('workspace-container');
1094                  const rect = container.getBoundingClientRect();
1095
1096                  if (e.clientX >= rect.left && e.clientX <= rect.right &&
1097                      e.clientY >= rect.top && e.clientY <= rect.bottom) {
1098
1099                      const canvasPos = screenToCanvas(e.clientX, e.clientY);
1100                      const config = ELEMENT_TYPES[AppState.dragType];
1101                      if (config) {
1102                          const defaultWidth = config.minWidth || 120;
1103                          const defaultHeight = config.minHeight || 60;
1104
1105                          // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
      обработать
1106                          const newElement = Elements.addElement(
1107                              AppState.dragType,
1108                              canvasPos.x - defaultWidth / 2,
1109                              canvasPos.y - defaultHeight / 2
1110                          );
1111
```

```
                        if (newElement && typeof Outputs !== 'undefined') {
                            Outputs.updateOutputStatus();
                        }
                    } else {
                        console.error('Неизвестный тип элемента при drop:',
    AppState.dragType);
                    }
                }
            } finally {
                App.cancelPaletteDrag();
            }
        }

        if (AppState.draggingElement) {
            AppState.draggingElement = null;
        }

        Connections.clearConnectionState();
    });

    document.addEventListener('keydown', (e) => {
        if (e.key === 'Delete' && AppState.selectedElement) {
            Elements.deleteElement(AppState.selectedElement);
            if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
        }
        if (e.key === 'Escape') {
            Elements.deselectAll();
            Connections.clearConnectionState();
            if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
        }
    });
},

    /**
     * Настройка контекстного меню
     */
    setupContextMenu() {
        document.addEventListener('click', (e) => {
            const menu = document.getElementById('context-menu');
            if (!menu.contains(e.target)) {
                menu.style.display = 'none';
            }
        });

        document.getElementById('ctx-properties').addEventListener('click', () => {
            const elemId = document.getElementById('context-menu').dataset.elementId;
            document.getElementById('context-menu').style.display = 'none';
            const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
            if (config?.hasProperties) {
                Modal.showPropertiesModal(elemId);
            }
        });

        document.getElementById('ctx-delete').addEventListener('click', () => {
            const elemId = document.getElementById('context-menu').dataset.elementId;
            document.getElementById('context-menu').style.display = 'none';
            Elements.deleteElement(elemId);
            // Обновляем выходы только если модуль загружен
            if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
                Outputs.updateOutputStatus();
            }
        });
    },

    /**
```

```
1176       * Клик по рабочей области
1177       */
1178      setupWorkspaceClick() {
1179          const workspace = document.getElementById('workspace');
1180
1181          workspace.addEventListener('click', (e) => {
1182              if (e.target === workspace) {
1183                  Elements.deselectAll();
1184              }
1185          });
1186      },
1187      /**
1188   * --- Выделение рамкой и множественное перемещение ---
1189   */
1190      setupMultiSelection() {
1191          const container = document.getElementById('workspace-container');
1192          const rectEl = document.getElementById('selection-rect');
1193
1194          container.addEventListener('mousedown', (e) => {
1195              if (e.button !== 0) return;
1196              if (e.target !== document.getElementById('workspace')) return;
1197
1198              const pos = screenToCanvas(e.clientX, e.clientY);
1199              AppState.multiSelecting = true;
1200              AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
      pos.y, w: 0, h: 0 };
1201
1202              rectEl.style.left = e.clientX + 'px';
1203              rectEl.style.top = e.clientY + 'px';
1204              rectEl.style.width = '0px';
1205              rectEl.style.height = '0px';
1206              rectEl.style.display = 'block';
1207          });
1208
1209          document.addEventListener('mousemove', (e) => {
1210              if (!AppState.multiSelecting) return;
1211
1212              const pos = screenToCanvas(e.clientX, e.clientY);
1213              const sx = AppState.selectionRect.startX;
1214              const sy = AppState.selectionRect.startY;
1215              const x = Math.min(sx, pos.x);
1216              const y = Math.min(sy, pos.y);
1217              const w = Math.abs(pos.x - sx);
1218              const h = Math.abs(pos.y - sy);
1219
1220              rectEl.style.left = x * AppState.viewport.zoom + AppState.viewport.panX +
      'px';
1221              rectEl.style.top = y * AppState.viewport.zoom + AppState.viewport.panY +
      'px';
1222              rectEl.style.width = w * AppState.viewport.zoom + 'px';
1223              rectEl.style.height = h * AppState.viewport.zoom + 'px';
1224
1225              const selected = [];
1226              for (const [id, elData] of Object.entries(AppState.elements)) {
1227              if (!elData || elData.type === 'output-frame') continue;
1228              if (
1229                  elData.x >= x && elData.x + elData.width <= x + w &&
1230                  elData.y >= y && elData.y + elData.height <= y + h
1231              ) selected.push(id);
1232              }
1233
1234              AppState.selectedElements = selected;
1235              document.querySelectorAll('.element').forEach(el =>
1236              el.classList.toggle('selected', selected.includes(el.id))
1237              );
```

```
        });

        document.addEventListener('mouseup', () => {
            if (AppState.multiSelecting) {
            AppState.multiSelecting = false;
            rectEl.style.display = 'none';
            }
        });
    }
};

// Запуск приложения при загрузке страницы
document.addEventListener('DOMContentLoaded', () => {
    App.init();
});

codegen_graph.js:
// js/codegen_graph.js

const CodeGenGraph = {
        /**
     * Собрать все условия вверх по цепочке cond-портов (до корня).
     * Возвращает null или объединённое через AND условие.
     */
/**
 * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
 */
    collectAllCond(graph) {
        if (!graph) return null;

        let c = null;
        const elem = graph.elem;

        // 1. Собираем условия через cond-порт (как было)
        if (graph.condInput) {
            const condConn = graph.condInput.conn;
            const fromGraph = graph.condInput.fromGraph;
            const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
            c = oneCond;

            // Рекурсивно идём вверх по cond-цепочке
            const upCond = this.collectAllCond(fromGraph);
            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,
        };
```

```
1303
1304            if (!graph.elem) return null;
1305
1306            const inConns = AppState.connections.filter(c =>
1307                c.toElement === elementId && c.toPort.startsWith('in-')
1308            );
1309            inConns.forEach(conn => {
1310                graph.inputs.push({
1311                    conn,
1312                    fromGraph: this.buildDependencyGraph(conn.fromElement)
1313                });
1314            });
1315
1316            const condConn = AppState.connections.find(c =>
1317                c.toElement === elementId && c.toPort === 'cond-0'
1318            );
1319            if (condConn) {
1320                graph.condInput = {
1321                    conn: condConn,
1322                    fromGraph: this.buildDependencyGraph(condConn.fromElement)
1323                };
1324            }
1325
1326            return graph;
1327        },
1328
1329        /**
1330         * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
1331         */
1332        evalLogic(graph) {
1333            if (!graph) return Optimizer.TrueCond;
1334            const elem = graph.elem;
1335
1336            switch (elem.type) {
1337                case 'if': {
1338                    const left = graph.inputs[0]?.fromGraph;
1339                    const right = graph.inputs[1]?.fromGraph;
1340
1341                    const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
1342                    const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);
1343
1344                    const op = elem.props.operator || '=';
1345                    return this.buildIfLogic(leftVal, op, rightVal);
1346                }
1347
1348                case 'and': {
1349                    let result = null;
1350                    for (const inp of graph.inputs) {
1351                        const inLogic = this.evalLogic(inp.fromGraph);
1352                        result = result ? Optimizer.And(result, inLogic) : inLogic;
1353                    }
1354                    return result || Optimizer.TrueCond;
1355                }
1356
1357                case 'or': {
1358                    let result = null;
1359                    for (const inp of graph.inputs) {
1360                        const inLogic = this.evalLogic(inp.fromGraph);
1361                        result = result ? Optimizer.Or(result, inLogic) : inLogic;
1362                    }
1363                    return result || Optimizer.FalseCond;
1364                }
1365
1366                case 'not': {
1367                    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);
        }
    },

    // js/codegen_graph.js

    /**
     * Рекурсивно собрать полный контекст условий для элемента
     * через всю цепочку cond-портов вверх
     */
 // В codegen_graph.js, в evalFullContext добавь:

    evalFullContext(graph) {
        if (!graph) return null;

        let context = null;
        const elem = graph.elem;

        console.log(`evalFullContext для ${elem.id} (${elem.type})`);

        // 1. Если сам элемент имеет cond-порт — собираем его условие
        if (graph.condInput) {
            const condConn = graph.condInput.conn;
            console.log(`  → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.$
{condConn.fromPort}`);

            const condLogic = this.evalConditionFromPort(
                graph.condInput.fromGraph,
                condConn.fromPort
            );
```

```
1432              console.log(`  → условие от cond-0: ${Optimizer.printCond(condLogic)}`);
1433              context = condLogic;
1434
1435              // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
      порт
1436              const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
1437              if (upstreamContext) {
1438                  console.log(`  → upstreamContext: $
      {Optimizer.printCond(upstreamContext)}`);
1439                  context = context ? Optimizer.And(context, upstreamContext) :
      upstreamContext;
1440              }
1441          } else {
1442              console.log(`  → нет cond-0`);
1443          }
1444
1445          console.log(`  → итоговый контекст: ${Optimizer.printCond(context)}`);
1446          return context;
1447      },
1448
1449      /**
1450       * Получить УСЛОВИЕ для cond-порта элемента
1451       * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
1452       */
1453      evalConditionFromPort(graph, fromPort) {
1454          if (!graph) return null;
1455          const elem = graph.elem;
1456
1457          // Если это сепаратор — вычисляем его вход и применяем ветвление
1458          if (elem.type === 'separator') {
1459              const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
1460
1461              if (fromPort === 'out-0') {
1462                  return inputLogic;
1463              } else if (fromPort === 'out-1') {
1464                  return Optimizer.Not(inputLogic);
1465              }
1466          }
1467
1468          // Если это логический элемент (AND/OR/NOT/IF) — просто вычисляем логику
1469          if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
      elem.type === 'if') {
1470              return this.evalLogic(graph);
1471          }
1472
1473          return null;
1474      },
1475
1476      /**
1477       * Главная функция: получить {cond, expr} для элемента
1478       */
1479      evalGraphValue(graph) {
1480
1481          if (!graph) return { cond: null, expr: Optimizer.Const(0) };
1482
1483          const elem = graph.elem;
1484          //let cond = null;
1485
1486          // ← НОВОЕ: собираем полный контекст через цепочку cond-портов
1487          let cond = this.collectAllCond(graph);
1488
1489          let expr = null;
1490
1491          switch (elem.type) {
1492              case 'input-signal':
```

```
                expr = Optimizer.Var(elem.props.name || graph.nodeId);
                break;

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

            case 'formula': {
                // Для формулы также собираем условия от всех входных элементов
                const inputConds = graph.inputs.map(inp => {
                    const inResult = this.evalGraphValue(inp.fromGraph);
                    return inResult.cond;
                }).filter(c => c);

                // Объединяем cond-порт с условиями от входов
                for (const inCond of inputConds) {
                    cond = cond ? Optimizer.And(cond, inCond) : inCond;
                }

                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';
        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})`);
            }
        }

        return result;
    }
};

window.CodeGenGraph = CodeGenGraph;

codegen_optimizer.js:

// js/codegen_optimizer.js

let _depth = 0;
const MAX_DEPTH = 200;

// === Конструкторы ===
function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
function And(a, b) {
    if (!a) return b;
    if (!b) return a;
    return { kind: 'cond', type: 'and', a, b };
}
function Or(a, b) {
    if (!a) return b;
    if (!b) return a;
    return { kind: 'cond', type: 'or', a, b };
}
function Not(x) {
    if (!x) return null;
    return { kind: 'cond', type: 'not', x };
}
const TrueCond = { kind: 'cond', type: 'true' };
const FalseCond = { kind: 'cond', type: 'false' };

function Const(n) { return { kind: 'expr', type: 'const', n }; }
function Var(name) { return { kind: 'expr', type: 'var', name }; }
function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }

// === Утилиты ===
function atomKey(c) {
    if (!c) return null;
    switch (c.type) {
        case 'eq0': return `eq0:${c.v}`;
        case 'ne0': return `ne0:${c.v}`;
        case 'cmp': return `cmp:${c.l}:${c.op}:${c.r}`;
        case 'true': return 'true';
        case 'false': return 'false';
        default: 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;
    let minInc = false;
    let maxInc = false;

    switch (op) {
        case '<':
            max = val; maxInc = false; break;
        case '<=':
            max = val; maxInc = true; break;
        case '>':
            min = val; minInc = false; break;
        case '>=':
            min = val; minInc = true; break;
        case '=':
```

```
            min = val; minInc = true;
            max = val; maxInc = true;
            break;
    }

    return { varName, min, minInc, max, maxInc };
}

function intervalSubset(a, b) {
    if (!a || !b) return false;

    // Нижняя граница: a.min >= b.min
    const amin = a.min, bmin = b.min;
    if (amin === Number.NEGATIVE_INFINITY) {
        if (bmin !== Number.NEGATIVE_INFINITY) return false;
        // оба -∞ — ок
    } else if (bmin === Number.NEGATIVE_INFINITY) {
        // b начинается “раньше” — ок
    } else if (amin > bmin) {
        // a стартует правее b — ок
    } else if (amin < bmin) {
        // a захватывает меньшее значение — не подмножество
        return false;
    } else {
        // amin === bmin
        if (a.minInc && !b.minInc) {
            // a включает границу, а b — нет → в a есть точка, не входящая в b
            return false;
        }
    }

    // Верхняя граница: a.max <= b.max
    const amax = a.max, bmax = b.max;
    if (amax === Number.POSITIVE_INFINITY) {
        if (bmax !== Number.POSITIVE_INFINITY) return false;
    } else if (bmax === Number.POSITIVE_INFINITY) {
        // b идёт дальше — ок
    } else if (amax < bmax) {
        // a заканчивается раньше — ок
    } else if (amax > bmax) {
        return false;
    } else {
        // amax === bmax
        if (a.maxInc && !b.maxInc) {
            return false;
        }
    }

    return true;
}

// 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 condEq(a, b) {
    if (a === b) return true;
    if (!a || !b) return false;
    if (a.type !== b.type) return false;

    switch (a.type) {
        case 'eq0':
        case 'ne0':
            return a.v === b.v;
        case 'cmp':
            return a.l === b.l && a.op === b.op && a.r === b.r;
        case 'true':
        case 'false':
            return true;
        case 'not':
            return condEq(a.x, b.x);
        case 'and':
        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': {
            const x = simplifyCondCore(c.x);
            if (!x) return TrueCond;
            if (x.type === 'true') return FalseCond;
            if (x.type === 'false') return TrueCond;
            if (x.type === 'not') return simplifyCondCore(x.x);
            if (x.type === 'eq0') return Ne0(x.v);
```

```
            if (x.type === 'ne0') return Eq0(x.v);
            if (x.type === 'cmp') {
                const negOp = negateOp(x.op);
                if (negOp) return Cmp(x.l, negOp, x.r);
            }
            if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
            if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
            return Not(x);
        }

case 'and': {
    const a = simplifyCondCore(c.a);
    const b = simplifyCondCore(c.b);

    if (!a) return b;
    if (!b) return a;
    if (a.type === 'false' || b.type === 'false') return FalseCond;
    if (a.type === 'true') return b;
    if (b.type === 'true') return a;

    const allTerms = [...flattenAnd(a), ...flattenAnd(b)];

    // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
    const eq0Vars = new Map(); // var -> term
    const ne0Vars = new Map(); // var -> term
    const cmpTerms = [];
    const otherTerms = [];

    for (const t of allTerms) {
        if (t.type === 'true') continue;
        if (t.type === 'false') return FalseCond;

        if (t.type === 'eq0') {
            // Проверка на противоречие сразу
            if (ne0Vars.has(t.v)) {
                console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
                return FalseCond;
            }
            eq0Vars.set(t.v, t);
        } else if (t.type === 'ne0') {
            // Проверка на противоречие сразу
            if (eq0Vars.has(t.v)) {
                console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
                return FalseCond;
            }
            ne0Vars.set(t.v, t);
        } else if (t.type === 'cmp') {
            cmpTerms.push(t);
        } else if (t.type === 'or') {
            // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
            const orTerms = flattenOr(t);
            const validBranches = [];

            for (const branch of orTerms) {
                let branchValid = true;

                if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
                    console.log(`OR ветка ${branch.v} != 0 противоречит контексту $
{branch.v} = 0`);
                    branchValid = false;
                } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
                    console.log(`OR ветка ${branch.v} = 0 противоречит контексту $
{branch.v} != 0`);
                    branchValid = false;
                }
```

```
2075
2076                if (branchValid) {
2077                    validBranches.push(branch);
2078                }
2079            }
2080
2081            if (validBranches.length === 0) {
2082                console.log(`Все ветки OR противоречат контексту → FALSE`);
2083                return FalseCond;
2084            } else if (validBranches.length === 1) {
2085                // Если осталась только одна ветка OR, добавляем её напрямую
2086                const singleBranch = validBranches[0];
2087                if (singleBranch.type === 'eq0') {
2088                    if (ne0Vars.has(singleBranch.v)) return FalseCond;
2089                    eq0Vars.set(singleBranch.v, singleBranch);
2090                } else if (singleBranch.type === 'ne0') {
2091                    if (eq0Vars.has(singleBranch.v)) return FalseCond;
2092                    ne0Vars.set(singleBranch.v, singleBranch);
2093                } else {
2094                    otherTerms.push(singleBranch);
2095                }
2096            } else {
2097                // Перестраиваем OR только с валидными ветками
2098                otherTerms.push(buildOr(validBranches));
2099            }
2100        } else {
2101            otherTerms.push(t);
2102        }
2103    }
2104
2105    // Собираем уникальные атомы
2106    const atomMap = new Map();
2107
2108    for (const [v, term] of eq0Vars) {
2109        const key = atomKey(term);
2110        if (key) atomMap.set(key, term);
2111    }
2112
2113    for (const [v, term] of ne0Vars) {
2114        const key = atomKey(term);
2115        if (key) atomMap.set(key, term);
2116    }
2117
2118    for (const term of cmpTerms) {
2119        const key = atomKey(term);
2120        if (key) {
2121            const negKey = negateAtomKey(key);
2122            if (negKey && atomMap.has(negKey)) {
2123                return FalseCond;
2124            }
2125            if (!atomMap.has(key)) {
2126                atomMap.set(key, term);
2127            }
2128        }
2129    }
2130
2131    let uniqueAtoms = Array.from(atomMap.values());
2132    uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
2133
2134    let result = [...uniqueAtoms, ...otherTerms];
2135
2136    // Поглощение: X AND (X OR Y) = X
2137    result = applyAndAbsorption(result);
2138
2139    if (result.length === 0) return TrueCond;
```

```
    if (result.length === 1) return result[0];

    return buildAnd(result);
}

        case 'or': {
            const a = simplifyCondCore(c.a);
            const b = simplifyCondCore(c.b);

            if (!a) return b;
            if (!b) return a;
            if (a.type === 'true' || b.type === 'true') return TrueCond;
            if (a.type === 'false') return b;
            if (b.type === 'false') return a;

            const allTerms = [...flattenOr(a), ...flattenOr(b)];
            const atomMap = new Map();
            const otherTerms = [];

            for (const t of allTerms) {
                if (t.type === 'true') return TrueCond;
                if (t.type === 'false') continue;

                const key = atomKey(t);
                if (key) {
                    const negKey = negateAtomKey(key);
                    if (negKey && atomMap.has(negKey)) {
                        return TrueCond;
                    }
                    if (!atomMap.has(key)) {
                        atomMap.set(key, t);
                    }
                } else {
                    otherTerms.push(t);
                }
            }

            let uniqueAtoms = Array.from(atomMap.values());
            uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');

            let result = [...uniqueAtoms, ...otherTerms];

            // Поглощение: X OR (X AND Y) = X
            result = applyOrAbsorption(result);

            if (result.length === 0) return FalseCond;
            if (result.length === 1) return result[0];

            return buildOr(result);
        }

        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);

            if (c?.type === 'true') return t;
            if (c?.type === 'false') return e;
            if (exprEq(t, e)) return t;

            return When(c, t, e);
        }

        default:
            return expr;
    }
}

// === Печать ===
```

```
function printCond(c) {
    if (!c) return 'TRUE';

    switch (c.type) {
        case 'eq0': return `${c.v} = 0`;
        case 'ne0': return `${c.v} != 0`;
        case 'cmp': return `${c.l} ${c.op} ${c.r}`;
        case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
        case 'or':  return `(${printCond(c.a)} OR ${printCond(c.b)})`;
        case 'not': return `NOT(${printCond(c.x)})`;
        case 'true': return 'TRUE';
        case 'false': return 'FALSE';
        default: return '?';
    }
}

function printExpr(e) {
    if (!e) return '0';

    switch (e.type) {
        case 'const': return String(e.n);
        case 'var': return e.name;
        case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
        case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, ${printExpr(e.e)})`;
        default: return '?';
    }
}

window.Optimizer = {
    Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
    Const, Var, Op, When,
    simplifyCond, simplifyExpr,
    printCond, printExpr,
    condEq, exprEq
};

codegen.js:
// js/codegen.js

const CodeGen = {
    _cache: {},
    _branchCache: {},
    _resolveCache: {},
    _visiting: new Set(),

    reset() {
        this._cache = {};
        this._branchCache = {};
        this._resolveCache = {};
        this._visiting = new Set();
    },

    toExpr(valueStr) {
        const s = String(valueStr).trim();
        if (s === '0') return Optimizer.Const(0);
        const num = parseFloat(s);
        if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
        return Optimizer.Var(s);
    },

    exprToName(exprAst) {
        if (!exprAst) return '0';
        if (exprAst.type === 'var') return exprAst.name;
        if (exprAst.type === 'const') return String(exprAst.n);
```

```
2334          return Optimizer.printExpr(exprAst);
2335      },
2336
2337      mergeCond(a, b) {
2338          if (!a && !b) return null;
2339          if (!a) return b;
2340          if (!b) return a;
2341          if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
2342          return Optimizer.And(a, b);
2343      },
2344
2345      getConn(toId, toPort) {
2346          return AppState.connections.find(c => c.toElement === toId && c.toPort ===
      toPort);
2347      },
2348
2349      getConns(toId, prefix) {
2350          return AppState.connections.filter(c => c.toElement === toId &&
      c.toPort.startsWith(prefix));
2351      },
2352
2353      buildFormulaExpr(elem) {
2354          const expression = elem.props.expression || '0';
2355          let result = expression;
2356          const formulaRefs = result.match(/formula-\d+/g) || [];
2357
2358          for (const ref of formulaRefs) {
2359              const refElem = AppState.elements[ref];
2360              if (refElem && refElem.type === 'formula') {
2361                  const refExpr = this.buildFormulaExpr(refElem);
2362                  result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
2363              }
2364          }
2365
2366          return result;
2367      },
2368
2369      // === Получить ЧИСТУЮ логику элемента ===
2370      getPureLogic(id) {
2371          const cacheKey = `logic:${id}`;
2372          if (cacheKey in this._cache) {
2373              return this._cache[cacheKey];
2374          }
2375
2376          const elem = AppState.elements[id];
2377          if (!elem) return null;
2378
2379          let logic = null;
2380
2381          switch (elem.type) {
2382              case 'if': {
2383                  const leftConn = this.getConn(id, 'in-0');
2384                  const rightConn = this.getConn(id, 'in-1');
2385
2386                  const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
      Optimizer.Const(0);
2387                  const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
      Optimizer.Const(0);
2388
2389                  const op = (elem.props.operator || '=').trim();
2390                  const leftName = this.exprToName(leftVal);
2391                  const rightName = this.exprToName(rightVal);
2392
2393                  const leftZero = leftVal.type === 'const' && leftVal.n === 0;
2394                  const rightZero = rightVal.type === 'const' && rightVal.n === 0;
```

```
2395
2396                switch (op) {
2397                    case '=':
2398                        if (rightZero) {
2399                            logic = Optimizer.Eq0(leftName);
2400                        } else if (leftZero) {
2401                            logic = Optimizer.Eq0(rightName);
2402                        } else {
2403                            logic = Optimizer.Cmp(leftName, '=', rightName);
2404                        }
2405                        break;
2406                    case '!=':
2407                        if (rightZero) {
2408                            logic = Optimizer.Ne0(leftName);
2409                        } else if (leftZero) {
2410                            logic = Optimizer.Ne0(rightName);
2411                        } else {
2412                            logic = Optimizer.Cmp(leftName, '!=', rightName);
2413                        }
2414                        break;
2415                    case '>':
2416                    case '<':
2417                    case '>=':
2418                    case '<=':
2419                        logic = Optimizer.Cmp(leftName, op, rightName);
2420                        break;
2421                    default:
2422                        logic = Optimizer.TrueCond;
2423                }
2424                break;
2425            }
2426
2427            case 'and':
2428            case 'or': {
2429                const isAnd = elem.type === 'and';
2430                const count = elem.props.inputCount || 2;
2431                let result = null;
2432
2433                for (let i = 0; i < count; i++) {
2434                    const conn = this.getConn(id, `in-${i}`);
2435                    if (!conn) continue;
2436
2437                    const val = this.getPureLogic(conn.fromElement);
2438                    if (!val) continue;
2439
2440                    if (result === null) {
2441                        result = val;
2442                    } else {
2443                        result = isAnd ? Optimizer.And(result, val) :
      Optimizer.Or(result, val);
2444                    }
2445                }
2446                logic = result || Optimizer.FalseCond;
2447                break;
2448            }
2449
2450            case 'not': {
2451                const conn = this.getConn(id, 'in-0');
2452                const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
2453                logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
2454                break;
2455            }
2456
2457            case 'separator': {
2458                const conn = this.getConn(id, 'in-0');
```

```
                logic = conn ? this.getPureLogic(conn.fromElement) :
    Optimizer.FalseCond;
                break;
            }

            default:
                logic = null;
        }

        // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
        if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
    elem.type === 'not') {
            const ctx = this.getConditionFromPort(id);
            if (ctx) {
                if (logic) {
                    logic = Optimizer.And(ctx, logic);
                } else {
                    logic = ctx;
                }
            }
        }

        this._cache[cacheKey] = logic;
        return logic;
    },

    // === Получить значение ===
    getValue(id) {
        const elem = AppState.elements[id];
        if (!elem) return Optimizer.Const(0);

        switch (elem.type) {
            case 'input-signal':
                // Имя сигнала или id как Var(...)
                return this.toExpr(elem.props.name || id);

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

            case 'formula': {
                // Используем текст формулы как выражение
                const exprStr = this.buildFormulaExpr(elem) || '0';
                return this.toExpr(exprStr);
            }

            default:
                // На всякий случай — даём символическое имя, а не 0
                if (elem.props && typeof elem.props.name === 'string') {
                    return this.toExpr(elem.props.name);
                }
                return this.toExpr(id);
        }
    },

    // === Получить ПОЛНОЕ условие для ветки сепаратора ===
    getBranchCondition(sepId, fromPort) {
        const cacheKey = `${sepId}:${fromPort}`;
        if (cacheKey in this._branchCache) {
            return this._branchCache[cacheKey];
        }

        const sep = AppState.elements[sepId];
        if (!sep || sep.type !== 'separator') return null;

        const inputLogic = this.getPureLogic(sepId);
```

```
2522          const sepContext = this.getConditionFromPort(sepId);
2523
2524          let branchLogic;
2525          if (fromPort === 'out-1') {
2526              branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
2527          } else {
2528              branchLogic = inputLogic || Optimizer.TrueCond;
2529          }
2530
2531          let result;
2532          if (sepContext) {
2533              result = Optimizer.And(sepContext, branchLogic);
2534          } else {
2535              result = branchLogic;
2536          }
2537
2538          this._branchCache[cacheKey] = result;
2539          return result;
2540      },
2541
2542      // === Получить условие от cond-порта ===
2543      getConditionFromPort(id) {
2544          const conn = this.getConn(id, 'cond-0');
2545          if (!conn) return null;
2546
2547          const sourceElem = AppState.elements[conn.fromElement];
2548          if (!sourceElem) return null;
2549
2550          if (sourceElem.type === 'separator') {
2551              return this.getBranchCondition(conn.fromElement, conn.fromPort);
2552          }
2553
2554          return this.getPureLogic(conn.fromElement);
2555      },
2556
2557      // === Основная функция разрешения ===
2558      resolve(id) {
2559          if (id in this._resolveCache) {
2560              return this._resolveCache[id];
2561          }
2562
2563          if (this._visiting.has(id)) {
2564              return null;
2565          }
2566          this._visiting.add(id);
2567
2568          const elem = AppState.elements[id];
2569          if (!elem) {
2570              this._visiting.delete(id);
2571              return null;
2572          }
2573
2574          let result = null;
2575
2576          try {
2577              switch (elem.type) {
2578                  case 'input-signal':
2579                      result = {
2580                          isValue: true,
2581                          cond: null,
2582                          expr: this.toExpr(elem.props.name || id)
2583                      };
2584                      break;
2585
2586                  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);
                    result = {
                        isValue: true,
                        cond: cond,
                        expr: Optimizer.Var(fullExpr)
                    };
                    break;
                }

                default:
                    result = null;
            }
        } finally {
            this._visiting.delete(id);
        }

        this._resolveCache[id] = result;
        return result;
    },

    generate() {
        console.log('=== Генерация кода (граф) ===');
        this.reset();

        try {
            const outputs = Object.values(AppState.elements).filter(e => e.type ===
    'output');

            if (outputs.length === 0) {
                return '/* Нет выходов */';
            }

            const allVariants = [];

            for (const out of outputs) {
                const conns = this.getConns(out.id, 'in-');

                for (const conn of conns) {
                    console.log(`\n=== Обработка выхода ${out.id}, вход от $
    {conn.fromElement} ===`);
                    const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
                    const result = CodeGenGraph.evalGraphValue(graph);
                    console.log(`Результат: cond=${Optimizer.printCond(result.cond)},
    expr=${Optimizer.printExpr(result.expr)}`);

```

```
                    if (!result || !result.expr) continue;

                    const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
    null;
                    const isZero = result.expr.type === 'const' && result.expr.n ===
    0;

                    if (isZero && !cond) continue;

                    allVariants.push({
                        cond,
                        expr: result.expr,
                        isZero
                    });
                }
            }

            console.log('Варианты:', allVariants.map(v => ({
                cond: Optimizer.printCond(v.cond),
                expr: Optimizer.printExpr(v.expr)
            })));

            if (allVariants.length === 0) return '0';

            const valueVariants = allVariants.filter(v => !v.isZero);
            if (valueVariants.length === 0) return '0';

            let result = Optimizer.Const(0);

            for (let i = valueVariants.length - 1; i >= 0; i--) {
                const v = valueVariants[i];
                if (v.cond) {
                    result = Optimizer.When(v.cond, v.expr, result);
                } else {
                    result = v.expr;
                }
            }

            const simplified = Optimizer.simplifyExpr(result);
            return Optimizer.printExpr(simplified);

        } catch (err) {
            console.error('Ошибка:', err);
            return `/* Ошибка: ${err.message} */`;
        }
    }
};

window.CodeGen = CodeGen;

config.js:
/**
 * Конфигурация приложения
 */

// Типы сигналов
const SIGNAL_TYPE = {
    NUMERIC: 'numeric',   // Числовой сигнал
    LOGIC: 'logic',       // Логический (может быть TRUE или FALSE)
    TRUE: 'true',         // Явно ИСТИНА
    FALSE: 'false',       // Явно ЛОЖЬ
    ANY: 'any'            // Любой тип
};

// Типы проекта
```

```
2712  const PROJECT_TYPE = {
2713      PARAMETER: 'parameter',
2714      RULE: 'rule'
2715  };
2716
2717  // Конфигурация элементов
2718  const ELEMENT_TYPES = {
2719      'input-signal': {
2720          name: 'Вход',
2721          inputs: 0,
2722          outputs: 1,
2723          outputLabels: ['out'],
2724          outputTypes: [SIGNAL_TYPE.NUMERIC],
2725          color: '#4a90d9',
2726          hasProperties: true,
2727          defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
2728          resizable: true,
2729          minWidth: 150,
2730          minHeight: 50
2731      },
2732      'and': {
2733          name: 'И',
2734          inputs: 2,  // По умолчанию 2, но может быть изменено
2735          outputs: 1,
2736          inputLabels: ['A', 'B'],
2737          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
2738          outputLabels: ['результат'],
2739          outputTypes: [SIGNAL_TYPE.LOGIC],
2740          color: '#a855f7',
2741          hasProperties: true,  // ← Теперь есть свойства (для изменения количества
      входов)
2742          resizable: true,
2743          minWidth: 120,
2744          minHeight: 80,
2745          hasConditionPort: true,
2746          conditionPortType: SIGNAL_TYPE.LOGIC,
2747          defaultProps: {
2748              inputCount: 2  // ← Новое свойство
2749          }
2750      },
2751      'or': {
2752          name: 'ИЛИ',
2753          inputs: 2,  // По умолчанию 2
2754          outputs: 1,
2755          inputLabels: ['A', 'B'],
2756          inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
2757          outputLabels: ['результат'],
2758          outputTypes: [SIGNAL_TYPE.LOGIC],
2759          color: '#a855f7',
2760          hasProperties: true,  // ← Теперь есть свойства
2761          resizable: true,
2762          minWidth: 120,
2763          minHeight: 80,
2764          hasConditionPort: true,
2765          conditionPortType: SIGNAL_TYPE.LOGIC,
2766          defaultProps: {
2767              inputCount: 2  // ← Новое свойство
2768          }
2769      },
2770      'not': {
2771          name: 'НЕ',
2772          inputs: 1,
2773          outputs: 1,
2774          inputLabels: ['A'],
2775          inputTypes: [SIGNAL_TYPE.LOGIC],
```

```
2776          outputLabels: ['¬A'],
2777          outputTypes: [SIGNAL_TYPE.LOGIC],
2778          color: '#a855f7',
2779          hasProperties: false,
2780          resizable: true,
2781          minWidth: 100,
2782          minHeight: 60,
2783          hasConditionPort: true,
2784          conditionPortType: SIGNAL_TYPE.LOGIC
2785      },
2786      'if': {
2787          name: 'ЕСЛИ',
2788          inputs: 2,
2789          outputs: 1,  // ← Только один выход!
2790          inputLabels: ['A', 'B'],
2791          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
2792          outputLabels: ['результат'],  // ← Просто результат
2793          outputTypes: [SIGNAL_TYPE.LOGIC],  // ← Выход типа LOGIC
2794          color: '#e94560',
2795          hasProperties: true,
2796          defaultProps: { operator: '=' },
2797          resizable: true,
2798          minWidth: 120,
2799          minHeight: 80,
2800          hasConditionPort: true,
2801          conditionPortType: SIGNAL_TYPE.LOGIC
2802      },
2803      'separator': {  // ← НОВЫЙ ЭЛЕМЕНТ
2804          name: 'Сепаратор',
2805          inputs: 1,
2806          outputs: 2,
2807          inputLabels: ['сигнал'],
2808          inputTypes: [SIGNAL_TYPE.LOGIC],
2809          outputLabels: ['ИСТИНА', 'ЛОЖЬ'],
2810          outputTypes: [SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE],  // ← TRUE и FALSE
2811          color: '#f59e0b',
2812          hasProperties: false,
2813          resizable: true,
2814          minWidth: 120,
2815          minHeight: 80,
2816          hasConditionPort: true,
2817          conditionPortType: SIGNAL_TYPE.LOGIC
2818      },
2819      'const': {
2820          name: 'Константа',
2821          inputs: 0,
2822          outputs: 1,
2823          outputLabels: ['out'],
2824          outputTypes: [SIGNAL_TYPE.NUMERIC],
2825          color: '#3b82f6',
2826          hasProperties: true,
2827          defaultProps: { value: 0 },
2828          resizable: true,
2829          minWidth: 120,
2830          minHeight: 60,
2831          hasConditionPort: true,
2832          conditionPortType: SIGNAL_TYPE.LOGIC
2833      },
2834      'formula': {
2835          name: 'Формула',
2836          inputs: 2,
2837          outputs: 1,
2838          inputLabels: ['in₁', 'in₂'],
2839          inputTypes: [SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY],
2840          outputLabels: ['результат'],
```

```
        outputTypes: [SIGNAL_TYPE.NUMERIC],
        color: '#f59e0b',
        hasProperties: true,
        resizable: true,
        minWidth: 140,
        minHeight: 80,
        defaultProps: {
            expression: '',
            inputCount: 2
        },
        hasConditionPort: true,
        conditionPortType: SIGNAL_TYPE.LOGIC
    },
    'output': {
        name: 'Выход',
        inputs: 1,
        outputs: 0,
        inputLabels: ['сигнал'],
        inputTypes: [SIGNAL_TYPE.ANY],
        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');
                AppState.tempLine.setAttribute('class', 'temp-connection');
                AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L $
    {startPos.x} ${startPos.y}`);
                svg.appendChild(AppState.tempLine);
            }
        });

        port.addEventListener('mouseup', (e) => {
            e.stopPropagation();
            e.preventDefault();

            if (AppState.connectingFrom && port.classList.contains('input')) {
                const toElement = port.dataset.element;
                const toPortName = port.dataset.port;
                const inputType = getInputPortType(toElement, toPortName);

                if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
                    this.clearConnectionState();
                    return;
                }

                if (AppState.connectingFrom.element !== toElement) {
                    const targetElem = AppState.elements[toElement];
                    const allowMultipleInputs = targetElem?.type === 'output';

                    const exists = AppState.connections.some(c =>
                        c.toElement === toElement && c.toPort === toPortName
                    );

                    if (!exists || allowMultipleInputs) {
                        AppState.connections.push({
                            fromElement: AppState.connectingFrom.element,
                            fromPort: AppState.connectingFrom.port,
                            toElement,
                            toPort: toPortName,
                            signalType: AppState.connectingFromType
                        });

                        port.classList.add('connected');
                        this.drawConnections();
                        this.clearConnectionState();
                        return;
                    }
                }
            }
```

```
2969
2970                this.clearConnectionState();
2971            });
2972
2973            port.addEventListener('mouseenter', () => {
2974                if (AppState.connectingFrom && port.classList.contains('input')) {
2975                    const toPortName = port.dataset.port;
2976                    const inputType = getInputPortType(port.dataset.element, toPortName);
2977
2978                    if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
2979                        if (AppState.tempLine) {
2980                            AppState.tempLine.classList.add('invalid');
2981                        }
2982                    }
2983                }
2984            });
2985
2986            port.addEventListener('mouseleave', () => {
2987                if (AppState.tempLine) {
2988                    AppState.tempLine.classList.remove('invalid');
2989                }
2990            });
2991        },
2992
2993        /**
2994         * Подсветка совместимых портов
2995         */
2996        highlightCompatiblePorts(signalType) {
2997            document.querySelectorAll('.port.input').forEach(port => {
2998                const inputType = getInputPortType(port.dataset.element,
      port.dataset.port);
2999
3000                if (areTypesCompatible(signalType, inputType)) {
3001                    port.classList.add('compatible-highlight');
3002                } else {
3003                    port.classList.add('incompatible');
3004                }
3005            });
3006        },
3007
3008        /**
3009         * Очистка состояния соединения
3010         */
3011        clearConnectionState() {
3012            if (AppState.tempLine) {
3013                AppState.tempLine.remove();
3014                AppState.tempLine = null;
3015            }
3016            AppState.connectingFrom = null;
3017            AppState.connectingFromType = null;
3018
3019            document.querySelectorAll('.port').forEach(port => {
3020                port.classList.remove('compatible-highlight', 'incompatible');
3021            });
3022        },
3023
3024        /**
3025         * Отрисовка временной линии соединения
3026         */
3027        drawTempConnection(e) {
3028        if (!AppState.tempLine || !AppState.connectingFrom) return;
3029
3030        const fromElem = document.getElementById(AppState.connectingFrom.element);
3031        if (!fromElem) return;
3032
```

```
3033        const fromPort = fromElem.querySelector(`[data-port="$
      {AppState.connectingFrom.port}"]`);
3034        if (!fromPort) return;
3035
3036        const startPos = this._getPortCanvasCenter(fromPort);
3037        const endPos = screenToCanvas(e.clientX, e.clientY);
3038
3039        const horizontalDist = Math.abs(endPos.x - startPos.x);
3040        const controlDist = Math.max(horizontalDist * 0.4, 50);
3041
3042        // Тянем всегда от выхода (вектор 1, 0)
3043        const cx1 = startPos.x + controlDist;
3044        const cy1 = startPos.y;
3045
3046        // Вторая точка контроля для плавности за курсором
3047        const cx2 = endPos.x - controlDist;
3048        const cy2 = endPos.y;
3049
3050        AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} $
      {cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);
3051        AppState.tempLine.setAttribute('fill', 'none');
3052    },
3053
3054        /**
3055         * Отрисовка всех соединений
3056         */
3057        drawConnections() {
3058        const svg = document.getElementById('connections-svg');
3059
3060        // 1. Очистка старых линий
3061        svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());
3062
3063        // 2. Сброс визуального состояния портов
3064        document.querySelectorAll('.port.connected').forEach(port => {
3065            port.classList.remove('connected');
3066        });
3067
3068        // 3. Перебор всех соединений из AppState
3069        AppState.connections.forEach(conn => {
3070            const fromElem = document.getElementById(conn.fromElement);
3071            const toElem = document.getElementById(conn.toElement);
3072
3073            if (!fromElem || !toElem) return;
3074
3075            const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);
3076            const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);
3077
3078            if (!fromPort || !toPort) return;
3079
3080            fromPort.classList.add('connected');
3081            toPort.classList.add('connected');
3082
3083            const startPos = this._getPortCanvasCenter(fromPort);
3084            const endPos = this._getPortCanvasCenter(toPort);
3085
3086            if (!startPos || !endPos) return;
3087
3088            // Расстояние для изгиба кривой
3089            const horizontalDist = Math.abs(endPos.x - startPos.x);
3090            const verticalDist = Math.abs(endPos.y - startPos.y);
3091            const controlDist = Math.max(horizontalDist * 0.4, 50);
3092
3093            // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---
3094            let d;
3095            let cx1 = startPos.x;
```

```
3096          let cy1 = startPos.y;
3097          let cx2 = endPos.x;
3098          let cy2 = endPos.y;
3099
3100          // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
3101          cx1 = startPos.x + controlDist;
3102          cy1 = startPos.y;
3103
3104          // ВХОД (Target):
3105          if (conn.toPort === 'cond-0') {
3106              // Технический порт: Касательная (0, 1) в декартовой (вверх)
3107              // В экранных координатах Y инвертирован, поэтому отнимаем от Y
3108              cx2 = endPos.x;
3109              cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
3110          } else {
3111              // Обычный вход: Касательная (-1, 0) -> Слева направо
3112              cx2 = endPos.x - controlDist;
3113              cy2 = endPos.y;
3114          }
3115
3116          d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
      ${endPos.y}`;
3117
3118          const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
3119          path.setAttribute('d', d);
3120          path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
3121
3122          // --- ЛОГИКА ЦВЕТА (Классы) ---
3123          let cssClass = 'connection';
3124          const type = conn.signalType;
3125
3126          // Приоритет новым типам сигналов
3127          if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
3128          else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
3129          else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
3130          else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
3131          else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
3132
3133          path.setAttribute('class', cssClass);
3134
3135          // Обработчики событий
3136          path.style.pointerEvents = 'stroke';
3137          path.style.cursor = 'pointer';
3138          path.addEventListener('click', () => this.handleConnectionClick(conn));
3139
3140          svg.appendChild(path);
3141      });
3142
3143      if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
3144          Outputs.updateOutputStatus();
3145      }
3146      Viewport.updateMinimap();
3147  },
3148      /**
3149       * Обработка клика по соединению (удаление)
3150       */
3151      handleConnectionClick(conn) {
3152          if (confirm('Удалить соединение?')) {
3153              AppState.connections = AppState.connections.filter(c =>
3154                  !(c.fromElement === conn.fromElement &&
3155                    c.fromPort === conn.fromPort &&
3156                    c.toElement === conn.toElement &&
3157                    c.toPort === conn.toPort)
3158              );
3159
```

```
            this.drawConnections();
        }
    },

    /**
     * Получение центра порта в координатах Canvas
     */
    _getPortCanvasCenter(portEl) {
        if (!portEl) return null;

        const rect = portEl.getBoundingClientRect();
        return screenToCanvas(
            rect.left + rect.width / 2,
            rect.top + rect.height / 2
        );
    }
};

elements.js:
/**
 * Модуль работы с элементами схемы
 */

const Elements = {
    /**
     * Генерация HTML для элемента
     */
        createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
            const config = ELEMENT_TYPES[elemType];
            if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);

            const safe = (value, fallback = '') => (value === null || value ===
undefined) ? fallback : String(value);
            const w = width ?? config.minWidth ?? 120;
            const h = height ?? config.minHeight ?? 60;

            const getPortClass = (signalType, direction) => {
                const base = direction === 'output' ? 'port output' : 'port input';
                if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
                if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
                return `${base} any-port`;
            };

            // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
innerHTML
            // Она тут остается, но ее результат не встраивается в HTML-строку
напрямую, кроме формулы
            const buildConditionPortHTML = () => {
                return `
                    <div class="condition-port-wrapper">
                        <div class="condition-port-label">условие</div>
                        <div class="port input condition-port"
                            data-port="cond-0"
                            data-element="${elemId}"
                            data-signal-type="${SIGNAL_TYPE.LOGIC}"
                            title="Техническое условие">
                        </div>
                    </div>`;
            };


            const buildInputPorts = (count, types = [], labels = []) => {
                let html = '';
                for (let i = 0; i < count; i++) {
                    const type = types[i] ?? types[types.length - 1] ??
```

```
      SIGNAL_TYPE.ANY;
3222                      html += `<div class="${getPortClass(type, 'input')}" data-
      port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Вход ${i+1}`}"></div>`;
3223                  }
3224                  return html;
3225              };
3226
3227              const buildOutputPorts = (count, types = [], labels = []) => {
3228                  let html = '';
3229                  for (let i = 0; i < count; i++) {
3230                      const type = types[i] ?? types[types.length - 1] ??
      SIGNAL_TYPE.ANY;
3231                      html += `<div class="${getPortClass(type, 'output')}" data-
      port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
      || `Выход ${i+1}`}"></div>`;
3232                  }
3233                  return html;
3234              };
3235
3236              const resizeHandles = config.resizable ? `<div class="resize-handle
      handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
      direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
      '';
3237              // hasCondClass будет добавляться в addElement
3238              // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
      '';
3239
3240              let innerHTML = '';
3241
3242              if (elemType === 'input-signal') {
3243                  const name = safe(props.name, 'Сигнал');
3244                  const type = props.signalType || SIGNAL_TYPE.NUMBER;
3245                  const symbol = type === SIGNAL_TYPE.LOGIC ? '🔀' : '🔢';
3246                  innerHTML = `
3247                      <div class="element-header" style="background:$
      {config.color};">Источник</div>
3248                      <div class="element-body">
3249                          <div class="element-symbol">
3250                              <span class="input-signal-icon">${symbol}</span>
3251                              <span class="input-signal-name">${name}</span>
3252                          </div>
3253                          <div class="ports-right">
3254                              ${buildOutputPorts(1, [type], ['Выход'])}
3255                          </div>
3256                      </div>`;
3257              }
3258              else if (elemType === 'const') {
3259                  innerHTML = `
3260                      <div class="element-header" style="background:$
      {config.color};">Константа</div>
3261                      <div class="element-body">
3262                          <div class="element-symbol">${props.value ?? 0}</div>
3263                          <div class="ports-right">
3264                              ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
3265                          </div>
3266                      </div>`;
3267              }
3268              else if (elemType === 'separator') {
3269                  innerHTML = `
3270                      <div class="element-header" style="background:$
      {config.color};">Сепаратор</div>
3271                      <div class="element-body">
3272                          <div class="ports-left">${buildInputPorts(1,
      config.inputTypes, config.inputLabels)}</div>
```

```
3273                         <div class="element-symbol">✓/✗</div>
3274                         <div class="ports-right">
3275                             <div class="port output logic-port true-port" data-
      port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
      title="ИСТИНА"></div>
3276                             <div class="port output logic-port false-port" data-
      port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
      title="ЛОЖЬ"></div>
3277                         </div>
3278                     </div>`;
3279             }
3280             else if (elemType === 'and' || elemType === 'or') {
3281                 const gateSymbol = elemType === 'and' ? '∧' : '∨';
3282                 const inputCount = props.inputCount || config.defaultProps?.inputCount
      || 2;
3283 
3284                 // Генерируем динамические входы
3285                 let inputsHTML = '';
3286                 for (let i = 0; i < inputCount; i++) {
3287                     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>`;
3288                 }
3289 
3290                 innerHTML = `
3291                     <div class="element-header" style="background:${config.color};">$
      {config.name}</div>
3292                     <div class="element-body">
3293                         <div class="ports-left">
3294                             ${inputsHTML}
3295                         </div>
3296                         <div class="element-symbol">${gateSymbol}</div>
3297                         <div class="ports-right">
3298                             <div class="port output logic-port" data-port="out-0"
      data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></
      div>
3299                         </div>
3300                     </div>`;
3301             }
3302             else if (elemType === 'if') {
3303                 const op = safe(props.operator, '=');
3304                 innerHTML = `
3305                     <div class="element-header" style="background:$
      {config.color};">Условие</div>
3306                     <div class="element-body">
3307                         <div class="ports-left">${buildInputPorts(2,
      config.inputTypes, config.inputLabels)}</div>
3308                         <div class="element-symbol">${op}</div>
3309                         <div class="ports-right">
3310                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
3311                         </div>
3312                     </div>`;
3313             }
3314             else if (elemType === 'not') {
3315                 innerHTML = `
3316                     <div class="element-header" style="background:$
      {config.color};">НЕ</div>
3317                     <div class="element-body">
3318                         <div class="ports-left">${buildInputPorts(1,
      [SIGNAL_TYPE.LOGIC], ['A'])}</div>
3319                         <div class="element-symbol">¬</div>
3320                         <div class="ports-right">
3321                             ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
3322                         </div>
3323                     </div>`;
```

```
3324             }
3325             else if (elemType === 'formula') {
3326                 const inputCount = props.inputCount || config.defaultProps?.inputCount
     || config.inputs || 2;
3327                 const expression = safe(props.expression);
3328                 const displayExpression = expression
3329                     ? (expression.length > 12 ? `${expression.slice(0, 12)}…` :
     expression)
3330                     : 'f(x)';
3331 
3332                 innerHTML = `
3333                     ${buildConditionPortHTML()}
3334                     <div class="element-header" style="background:$
     {config.color};">Формула</div>
3335                     <div class="element-body">
3336                         <div class="ports-left">${buildInputPorts(inputCount,
     config.inputTypes, config.inputLabels)}</div>
3337                         <div class="element-symbol">${displayExpression}</div>
3338                         <div class="ports-right">
3339                             ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER],
     ['Результат'])}
3340                         </div>
3341                     </div>`;
3342             }
3343             else if (elemType === 'output') {
3344                 innerHTML = `
3345                     <div class="element-header" style="background:$
     {config.color};">Выход</div>
3346                     <div class="element-body">
3347                         <div class="ports-left">
3348                             ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
3349                         </div>
3350                         <div class="element-symbol">${safe(props.label, 'Выход')}</
     div>
3351                         <div class="ports-right"></div>
3352                     </div>`;
3353 
3354             }
3355             else if (elemType === 'group') {
3356                 const title = props.title || 'Группа';
3357                 innerHTML = `
3358                     <div class="group-content">
3359                     <div class="group-title">${title}</div>
3360                     </div>`;
3361             }
3362 
3363             else { // Для любых других (fallback)
3364                 innerHTML = `
3365                     <div class="element-header" style="background:${config.color};">$
     {config.name}</div>
3366                     <div class="element-body">
3367                         <div class="ports-left">${buildInputPorts(config.inputs || 0,
     config.inputTypes, config.inputLabels)}</div>
3368                         <div class="element-symbol">${config.name}</div>
3369                         <div class="ports-right">
3370                             ${buildOutputPorts(config.outputs || 0,
     config.outputTypes, config.outputLabels)}
3371                         </div>
3372                     </div>`;
3373             }
3374 
3375 
3376             const html = `
3377                 <div class="element ${elemType}" id="${elemId}"
3378                     style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;"
```

```
      data-type="${elemType}">
3379                      ${innerHTML}
3380                      ${resizeHandles}
3381                  </div>`;
3382
3383              return { html, width: w, height: h };
3384          },
3385
3386      /**
3387       * Добавление элемента
3388       */
3389          addElement(elemType, x, y, props = {}, elemId = null, customWidth = null,
      customHeight = null) {
3390              const config = ELEMENT_TYPES[elemType];
3391              if (!config) {
3392                  console.error(`Неизвестный тип элемента: ${elemType}`);
3393                  return null;
3394              }
3395
3396              if (!elemId) {
3397                  elemId = `${elemType}-${++AppState.elementCounter}`;
3398              }
3399
3400              let width = customWidth;
3401              let height = customHeight;
3402
3403              if (width === null || width === undefined) {
3404                  width = config.minWidth || 140;
3405              }
3406              if (height === null || height === undefined) {
3407                  height = config.minHeight || 70;
3408              }
3409
3410              try {
3411                  const result = this.createElementHTML(elemType, elemId, x, y, props,
      width, height);
3412                  if (!result || !result.html) {
3413                      console.error('createElementHTML вернул пустой результат');
3414                      return null;
3415                  }
3416
3417                  const workspace = document.getElementById('workspace');
3418                  const wrapper = document.createElement('div');
3419                  wrapper.innerHTML = result.html.trim();
3420                  const element = wrapper.firstElementChild;
3421                  if (!element) {
3422                      console.error('Не удалось создать DOM элемент из HTML');
3423                      return null;
3424                  }
3425
3426                  // Добавляем класс для отступа
3427                  if (config.hasConditionPort) {
3428                      element.classList.add('has-condition-port');
3429                  }
3430
3431                  workspace.appendChild(element);
3432
3433                  AppState.elements[elemId] = {
3434                      id: elemId,
3435                      type: elemType,
3436                      x,
3437                      y,
3438                      width: result.width || width,
3439                      height: result.height || height,
3440                      props: { ...(config.defaultProps || {}), ...(props || {}) }
```

```
                };

                // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ ЕГО В HTML)
                if (config.hasConditionPort && elemType !== 'formula') {
                    const condPortWrapper = document.createElement('div');
                    condPortWrapper.innerHTML = `
                        <div class="condition-port-wrapper">
                            <div class="condition-port-label">условие</div>
                            <div class="port input condition-port"
                                data-port="cond-0"
                                data-element="${elemId}"
                                data-signal-type="${SIGNAL_TYPE.LOGIC}"
                                title="Техническое условие">
                            </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();
    },

    /**
     * Удаление элемента
     */
    deleteElement(elemId) {
        AppState.connections = AppState.connections.filter(c =>
            c.fromElement !== elemId && c.toElement !== elemId
        );

        const elem = document.getElementById(elemId);
        if (elem) elem.remove();

        delete AppState.elements[elemId];

        if (AppState.selectedElement === elemId) {
            AppState.selectedElement = null;
        }

        Connections.drawConnections();
        Viewport.updateMinimap();
    },

    /**
     * Выделение элемента
     */
    selectElement(elemId) {
        if (AppState.selectedElement) {
            const oldElem = document.getElementById(AppState.selectedElement);
            if (oldElem) oldElem.classList.remove('selected');
        }

        AppState.selectedElement = elemId;
        const elem = document.getElementById(elemId);
        if (elem) elem.classList.add('selected');

        const elemData = AppState.elements[elemId];
        if (elemData) {
            document.getElementById('selection-info').textContent =
                `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
        }
    },

    /**
     * Снять выделение
     */
    deselectAll() {
        if (AppState.selectedElement) {
            const elem = document.getElementById(AppState.selectedElement);
            if (elem) elem.classList.remove('selected');
            AppState.selectedElement = null;
        }
        document.getElementById('selection-info').textContent = '';
```

```
    },

    /**
     * Настройка обработчиков элемента
     */
    setupElementHandlers(elemId) {
        try {
            const elem = document.getElementById(elemId);
            if (!elem) return;

            elem.addEventListener('mousedown', (e) => {
                if (e.target.classList.contains('port')) return;
                if (e.target.classList.contains('resize-handle')) return;

                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) {
```

```
3633          handle.addEventListener('mousedown', (e) => {
3634              e.stopPropagation();
3635              e.preventDefault();
3636
3637              const elemData = AppState.elements[elemId];
3638
3639              AppState.resizing = {
3640                  elemId: elemId,
3641                  handle: handle.dataset.direction,
3642                  startX: e.clientX,
3643                  startY: e.clientY,
3644                  startWidth: elemData.width,
3645                  startHeight: elemData.height,
3646                  startLeft: elemData.x,
3647                  startTop: elemData.y
3648              };
3649          });
3650      },
3651
3652      /**
3653       * Обработка resize
3654       */
3655      handleResize(e) {
3656          if (!AppState.resizing) return;
3657
3658          const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
      startTop } = AppState.resizing;
3659          const elem = document.getElementById(elemId);
3660          const elemData = AppState.elements[elemId];
3661          const config = ELEMENT_TYPES[elemData.type];
3662
3663          const dx = (e.clientX - startX) / AppState.viewport.zoom;
3664          const dy = (e.clientY - startY) / AppState.viewport.zoom;
3665
3666          let newWidth = startWidth;
3667          let newHeight = startHeight;
3668          let newLeft = startLeft;
3669          let newTop = startTop;
3670
3671          if (handle.includes('e')) {
3672              newWidth = Math.max(config.minWidth, startWidth + dx);
3673          }
3674          if (handle.includes('w')) {
3675              newWidth = Math.max(config.minWidth, startWidth - dx);
3676              newLeft = startLeft + (startWidth - newWidth);
3677          }
3678          if (handle.includes('s')) {
3679              newHeight = Math.max(config.minHeight, startHeight + dy);
3680          }
3681          if (handle.includes('n')) {
3682              newHeight = Math.max(config.minHeight, startHeight - dy);
3683              newTop = startTop + (startHeight - newHeight);
3684          }
3685
3686          elem.style.width = `${newWidth}px`;
3687          elem.style.height = `${newHeight}px`;
3688          elem.style.left = `${newLeft}px`;
3689          elem.style.top = `${newTop}px`;
3690
3691          elemData.width = newWidth;
3692          elemData.height = newHeight;
3693          elemData.x = newLeft;
3694          elemData.y = newTop;
3695
3696          Connections.drawConnections();
```

```
3697      },
3698
3699      /**
3700       * Обработка перетаскивания элемента
3701       */
3702      handleDrag(e) {
3703          if (!AppState.draggingElement) return;
3704
3705          const canvasPos = screenToCanvas(e.clientX, e.clientY);
3706          const elemId = AppState.draggingElement;
3707          const elemData = AppState.elements[elemId];
3708          if (!elemData) return;
3709
3710          const newX = canvasPos.x - AppState.dragOffset.x;
3711          const newY = canvasPos.y - AppState.dragOffset.y;
3712          const dx = newX - elemData.x;
3713          const dy = newY - elemData.y;
3714
3715          // если выделено несколько
3716          const group = AppState.selectedElements && AppState.selectedElements.length >
     1
3717              ? AppState.selectedElements
3718              : [elemId];
3719
3720          for (const id of group) {
3721              const elData = AppState.elements[id];
3722              if (!elData) continue;
3723              elData.x += dx;
3724              elData.y += dy;
3725              const el = document.getElementById(id);
3726              if (el) {
3727              el.style.left = elData.x + 'px';
3728              el.style.top = elData.y + 'px';
3729              }
3730          }
3731
3732          Connections.drawConnections();
3733      },
3734
3735      /**
3736       * Обновление входов формулы
3737       */
3738      updateFormulaInputs(elemId, inputCount) {
3739          const elem = document.getElementById(elemId);
3740          if (!elem) return;
3741
3742          const portsLeft = elem.querySelector('.ports-left');
3743          if (!portsLeft) return;
3744
3745          AppState.connections = AppState.connections.filter(c => {
3746              if (c.toElement === elemId && c.toPort.startsWith('in-')) {
3747                  const portNum = parseInt(c.toPort.split('-')[1], 10);
3748                  return portNum < inputCount;
3749              }
3750              return true;
3751          });
3752
3753          let inputsHTML = '';
3754          for (let i = 0; i < inputCount; i++) {
3755              inputsHTML += `
3756                  <div class="port input any-port"
3757                       data-port="in-${i}"
3758                       data-element="${elemId}"
3759                       data-signal-type="${SIGNAL_TYPE.ANY}"
3760                       title="in${i} (Любой)">
```

```
                </div>
            `;
        }
        portsLeft.innerHTML = inputsHTML;

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

        Connections.drawConnections();
    },

    /**
     * Рассчитать оптимальный размер элемента на основе количества портов
     */
    calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
        const elem = AppState.elements[elemId];
        if (!elem) return null;

        const config = ELEMENT_TYPES[elem.type];
        if (!config || !config.resizable) return null;

        // Базовая высота
        let baseHeight = config.minHeight || 60;

        // Каждый порт требует примерно 25-30px высоты
        const portSpacing = 28;
        const maxPorts = Math.max(inputCount, outputCount);

        // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
        const additionalHeight = (maxPorts - 1) * portSpacing;
        const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);

        return newHeight;
    },

    /**
     * Обновление размера элемента при изменении портов
     */
    updateElementSize(elemId) {
        const elem = document.getElementById(elemId);
        const elemData = AppState.elements[elemId];

        if (!elem || !elemData) return;

        const config = ELEMENT_TYPES[elemData.type];
        if (!config || !config.resizable) return;

        let inputCount = 0;
        let outputCount = config.outputs || 1;

        // Определяем количество входов
        if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
    'formula') {
            inputCount = elemData.props.inputCount || config.inputs || 2;
        } else {
            inputCount = config.inputs || 0;
        }

        // Рассчитываем новую высоту
        const newHeight = this.calculateOptimalHeight(elemId, inputCount,
    outputCount);

        if (newHeight && newHeight !== elemData.height) {
            elemData.height = newHeight;
```

```
            elem.style.height = `${newHeight}px`;

            // Перерисовываем соединения, т.к. изменился размер элемента
            Connections.drawConnections();
            Viewport.updateMinimap();
        }
    }


};

modal.js:
/**
 * 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
     */
```

```
3886      hideModal(modalId) {
3887          document.getElementById(modalId).style.display = 'none';
3888      },
3889
3890      /**
3891       * Показать свойства элемента
3892       */
3893      showPropertiesModal(elemId) {
3894          const elemData = AppState.elements[elemId];
3895          const elemType = elemData.type;
3896          const props = elemData.props;
3897          const config = ELEMENT_TYPES[elemType];
3898
3899          const modalOverlay = document.getElementById('modal-overlay');
3900          const modalTitle = document.getElementById('modal-title');
3901          const modalContent = document.getElementById('modal-content');
3902
3903          modalTitle.textContent = `Свойства: ${config.name}`;
3904
3905          let contentHTML = '';
3906
3907   if (elemType === 'input-signal') {
3908    const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
3909
3910    contentHTML = `
3911      <div class="modal-row">
3912        <label>Название сигнала:</label>
3913        <input type="text" id="prop-name" value="${props.name || ''}"
      placeholder="Например: 10LBA..." />
3914        <small style="color:#999;">
3915          Поиск по маске через * (например: *MAA*CP*)
3916        </small>
3917        <div id="signal-filter-results"
3918             style="max-height:160px; overflow-y:auto; background:#0f3460; border-
      radius:5px; margin-top:6px; display:none;">
3919        </div>
3920      </div>
3921
3922      <div class="modal-row">
3923        <label>Описание сигнала:</label>
3924        <textarea id="prop-description" readonly>${props.description || ''}</textarea>
3925      </div>
3926
3927      <div class="modal-row">
3928        <label>Тип сигнала:</label>
3929        <select id="prop-signal-type">
3930          <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
      'selected' : ''}>Числовой</option>
3931          <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
      'selected' : ''}>Логический</option>
3932        </select>
3933      </div>
3934    `;
3935
3936    // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
      DOM.
3937    // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
3938    // (Смотри пункт 2 — небольшая вставка в конце showPropertiesModal)
3939  } else if (elemType === 'if') {
3940              contentHTML = `
3941                  <div class="modal-row">
3942                      <label>Оператор сравнения:</label>
3943                      <select id="prop-operator">
3944                          <option value="=" ${props.operator === '=' ? 'selected' : ''}
      >=  (равно)</option>
```

```
                        <option value=">" ${props.operator === '>' ? 'selected' : ''}
    >>  (больше)</option>
                        <option value="<" ${props.operator === '<' ? 'selected' : ''}
    ><  (меньше)</option>
                        <option value=">=" ${props.operator === '>=' ? 'selected' :
    ''}>= (больше или равно)</option>
                        <option value="<=" ${props.operator === '<=' ? 'selected' :
    ''}>= (меньше или равно)</option>
                        <option value="!=" ${props.operator === '!=' ? 'selected' :
    ''}>!= (не равно)</option>
                    </select>
                </div>
            `;
        } else if (elemType === 'and' || elemType === 'or') {
            contentHTML = `
                <div class="modal-row">
                    <label>Количество входов:</label>
                    <input type="number" id="prop-input-count" value="$
    {props.inputCount || 2}" min="2" max="10">
                </div>
                <div class="modal-row">
                    <p style="color: #aaa; font-size: 12px;">
                        Измените количество входных портов для этого логического
    элемента.
                        Лишние соединения будут автоматически удалены.
                    </p>
                </div>
            `;
        } else if (elemType === 'const') {
            contentHTML = `
                <div class="modal-row">
                    <label>Значение:</label>
                    <input type="number" id="prop-value" value="${props.value ?? 0}"
    step="any">
                </div>
            `;
        }
        else if (elemType === 'group') {
            contentHTML = `
            <div class="modal-row">
                <label>Название группы:</label>
                <input type="text" id="prop-title" value="${props.title || 'Группа'}">
            </div>`;
        }

        else if (elemType === 'formula') {
            let signalsHTML = '';
            AppState.connections.forEach(conn => {
                if (conn.toElement === elemId) {
                    const fromElem = AppState.elements[conn.fromElement];
                    if (fromElem) {
                        const signalName = fromElem.props?.name || fromElem.id;
                        signalsHTML += `<div class="signal-item" data-signal="$
    {signalName}">${signalName} (${conn.toPort})</div>`;
                    }
                }
            });

            contentHTML = `
                <div class="modal-row">
                    <label>Количество входов:</label>
                    <input type="number" id="prop-input-count" value="$
    {props.inputCount || 2}" min="1" max="10">
                </div>
                <div class="modal-row">
```

```
                    <label>Входные сигналы (двойной клик для вставки):</label>
                    <div class="signal-list" id="signal-list">
                        ${signalsHTML || '<div style="color:#888;padding:5px;">Нет подключённых сигналов</div>'}
                    </div>
                </div>
                <div class="modal-row">
                    <label>Выражение:</label>
                    <textarea id="prop-expression">${props.expression || ''}</textarea>
                </div>
            `;
        } else if (elemType === 'output') {
            contentHTML = `
                <div class="modal-row">
                    <label>Название выхода:</label>
                    <input type="text" id="prop-label" value="${props.label || 'Выход'}">
                </div>
                <div class="modal-row">
                    <label>Группировка (опционально):</label>
                    <input type="text" id="prop-output-group" value="${props.outputGroup || ''}" placeholder="для логической группировки выходов">
                </div>
            `;
        }


        modalContent.innerHTML = contentHTML;
        // --- post init handlers (когда DOM модалки уже существует) ---
        if (elemType === 'input-signal') {
        const input = document.getElementById('prop-name');
        const results = document.getElementById('signal-filter-results');
        const descField = document.getElementById('prop-description');

        let timer = null;

        const renderList = (items) => {
            if (!items || items.length === 0) {
            results.innerHTML = '<div style="color:#666;padding:6px;">Нет совпадений</div>';
            results.style.display = 'block';
            return;
            }

            results.innerHTML = items.map(s => `
            <div class="signal-result-item"
                style="padding:6px 8px; cursor:pointer; border-bottom:1px solid rgba(255,255,255,0.08);">
                <div style="font-weight:600;">${s.Tagname}</div>
                <div style="color:#aaa; font-size:11px;">${s.Description || ''}</div>
            </div>
            `).join('');

            results.style.display = 'block';

            results.querySelectorAll('.signal-result-item').forEach((div, i) => {
            div.addEventListener('click', () => {
                const chosen = items[i];
                input.value = chosen.Tagname;
                descField.value = chosen.Description || '';
                results.style.display = 'none';
            });
            });
        };
```

```
4059
4060          const search = async () => {
4061              const mask = (input.value || '').trim();
4062
4063              // Показываем список только если пользователь реально использует маску
4064              if (!mask.includes('*')) {
4065              results.style.display = 'none';
4066              return;
4067              }
4068
4069              results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</div>';
4070              results.style.display = 'block';
4071
4072              try {
4073              // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
4074              const data = await Settings.fetchSignals(mask, 50);
4075              renderList(data.items || []);
4076              } catch (e) {
4077              results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка загрузки
      сигналов</div>';
4078              results.style.display = 'block';
4079              console.error(e);
4080              }
4081          };
4082
4083          input.addEventListener('input', () => {
4084              clearTimeout(timer);
4085              timer = setTimeout(search, 200); // debounce
4086          });
4087
4088          // опционально: закрывать список кликом вне
4089          document.addEventListener('mousedown', (e) => {
4090              if (!results.contains(e.target) && e.target !== input) {
4091              results.style.display = 'none';
4092              }
4093          }, { once: true });
4094          }
4095          modalOverlay.dataset.elementId = elemId;
4096          this.showModal('modal-overlay');
4097
4098          // Обработчик вставки сигналов для формулы
4099          if (elemType === 'formula') {
4100              document.querySelectorAll('.signal-item').forEach(item => {
4101                  item.addEventListener('dblclick', () => {
4102                      const signal = item.dataset.signal;
4103                      const textarea = document.getElementById('prop-expression');
4104                      textarea.value += signal;
4105                      textarea.focus();
4106                  });
4107              });
4108          }
4109      },
4110
4111      /**
4112       * Сохранить свойства элемента
4113       */
4114  /**
4115   * Сохранить свойства элемента
4116   */
4117      saveElementProperties() {
4118          try {
4119              const modalOverlay = document.getElementById('modal-overlay');
4120              const elemId = modalOverlay.dataset.elementId;
4121              const elemData = AppState.elements[elemId];
4122              const elemType = elemData.type;
```

```
4123              const elem = document.getElementById(elemId);
4124
4125              if (elemType === 'input-signal') {
4126                  const name = document.getElementById('prop-name').value || 'Сигнал';
4127                  const description = document.getElementById('prop-description').value
      || '';
4128                  const signalType = document.getElementById('prop-signal-type').value;
4129
4130                  const oldSignalType = elemData.props.signalType;
4131                  elemData.props.name = name;
4132                  elemData.props.description = description;
4133                  elemData.props.signalType = signalType;
4134
4135                  if (oldSignalType !== signalType) {
4136                      AppState.connections = AppState.connections.filter(conn => {
4137                      if (conn.fromElement === elemId) {
4138                          const toPortIndex = parseInt(conn.toPort.split('-')[1]);
4139                          const inputType = getInputPortType(conn.toElement,
      toPortIndex);
4140                          return areTypesCompatible(signalType, inputType);
4141                      }
4142                      return true;
4143                      });
4144                  }
4145
4146                  const { html } = Elements.createElementHTML(
4147                      elemType, elemId, elemData.x, elemData.y, elemData.props,
      elemData.width, elemData.height
4148                  );
4149                  elem.outerHTML = html;
4150
4151                  Elements.setupElementHandlers(elemId);
4152                  Connections.drawConnections();
4153              } else if (elemType === 'if') {
4154                  const operator = document.getElementById('prop-operator').value;
4155                  elemData.props.operator = operator;
4156                  const symbol = elem.querySelector('.element-symbol');
4157                  if (symbol) symbol.textContent = operator;
4158
4159              } else if (elemType === 'const') {
4160                  const value = parseFloat(document.getElementById('prop-value').value)
      || 0;
4161                  elemData.props.value = value;
4162                  const symbol = elem.querySelector('.element-symbol');
4163                  if (symbol) symbol.textContent = String(value);
4164
4165              } else if (elemType === 'formula') {
4166                  const expression = document.getElementById('prop-expression').value;
4167                  const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
4168
4169                  elemData.props.expression = expression;
4170                  elemData.props.inputCount = inputCount;
4171
4172                  const symbol = elem.querySelector('.element-symbol');
4173                  if (symbol) {
4174                      symbol.textContent = expression.length > 12 ? `$
      {expression.slice(0, 12)}…` : (expression || 'f(x)');
4175                  }
4176
4177                  Elements.updateFormulaInputs(elemId, inputCount);
4178                  Elements.updateElementSize(elemId);  // ← Добавляем это
4179              } else if (elemType === 'and' || elemType === 'or') {
4180                  const inputCount = parseInt(document.getElementById('prop-input-
      count').value) || 2;
```

```
                elemData.props.inputCount = inputCount;

                Elements.updateLogicGateInputs(elemId, inputCount);
                Elements.updateElementSize(elemId);  // ← Добавляем это

                const symbol = elem.querySelector('.element-symbol');
                if (symbol) {
                    symbol.textContent = elemType === 'and' ? '∧' : '∨';
                }

        } else if (elemType === 'output') {
                const label = document.getElementById('prop-label').value || 'Выход';
                const outputGroup = document.getElementById('prop-output-group').value
    || '';

                elemData.props.label = label;
                elemData.props.outputGroup = outputGroup;

                const symbol = elem.querySelector('.element-symbol');
                if (symbol) symbol.textContent = label;
            }
            else if (elemType === 'group') {
                const title = document.getElementById('prop-title').value || 'Группа';
                elemData.props.title = title;
                const titleEl = elem.querySelector('.group-title');
                if (titleEl) titleEl.textContent = title;
                }

            this.hideModal('modal-overlay');

        } catch (error) {
            console.error('❌ Ошибка при сохранении свойств:', error);
            alert('Ошибка сохранения: ' + error.message);
        }
    },

    /**
     * Показать свойства проекта
     */
    showProjectPropertiesModal() {
        const content = document.getElementById('project-modal-content');
        const project = AppState.project;

        // Генерируем HTML для списка выходов только если модуль загружен
        let outputsHtml = '';
        if (typeof Outputs !== 'undefined' && AppState.outputs) {
            const logicalOutputsHtml = AppState.outputs.logical.length > 0
                ? AppState.outputs.logical.map(output => `
                    <div class="output-item"
                        data-element-id="${output.elementId}"
                        onmouseenter="Outputs.highlightOutput('${output.elementId}',
    true)"
                        onmouseleave="Outputs.highlightOutput('${output.elementId}',
    false)"
                        onclick="Outputs.navigateToOutput('${output.elementId}');
    Modal.hideModal('project-modal-overlay');">
                        <span class="output-icon">${output.portLabel === 'Да' ? '✅' :
    '❌'}</span>
                        <span class="output-name">${output.elementName}</span>
                        <span class="output-port">→ ${output.portLabel}</span>
                    </div>
                `).join('')
                : '<div class="no-outputs">Нет логических выходов</div>';

            const numericOutputsHtml = AppState.outputs.numeric.length > 0
```

```
                ? AppState.outputs.numeric.map(output => `
                    <div class="output-item numeric"
                        data-element-id="${output.elementId}"
                        onmouseenter="Outputs.highlightOutput('${output.elementId}', true)"
                        onmouseleave="Outputs.highlightOutput('${output.elementId}', false)"
                        onclick="Outputs.navigateToOutput('${output.elementId}'); Modal.hideModal('project-modal-overlay');">
                        <span class="output-icon">🔢</span>
                        <span class="output-name">${output.elementName}</span>
                        <span class="output-port">→ значение</span>
                    </div>
                `).join('')
                : '<div class="no-outputs">Нет числовых выходов</div>';

            outputsHtml = `
                <div class="modal-row">
                    <label>Выходные сигналы схемы:</label>
                    <div class="outputs-container">
                        <div class="outputs-section">
                            <div class="outputs-section-title">
                                <span class="section-icon">🔀</span>
                                Логические выходы (${AppState.outputs.logical.length})
                            </div>
                            <div class="outputs-list">
                                ${logicalOutputsHtml}
                            </div>
                        </div>
                        <div class="outputs-section">
                            <div class="outputs-section-title">
                                <span class="section-icon">📐</span>
                                Числовые выходы (${AppState.outputs.numeric.length})
                            </div>
                            <div class="outputs-list">
                                ${numericOutputsHtml}
                            </div>
                        </div>
                    </div>
                    <div class="outputs-hint">
                        💡 Выходами автоматически становятся элементы, чьи выходные порты не подключены к другим элементам.
                        Кликните на выход, чтобы перейти к нему на схеме.
                    </div>
                </div>
            `;
        }

        content.innerHTML = `
            <div class="modal-row">
                <label>Код проекта:</label>
                <input type="text" id="project-code" value="${project.code || ''}" placeholder="Уникальный идентификатор">
            </div>

            <div class="modal-row">
                <label>Тип проекта:</label>
                <div class="project-type-selector">
                    <div class="project-type-btn ${project.type === PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">
                        <div class="type-icon">📊</div>
                        <div class="type-name">Параметр</div>
                        <div class="type-desc">Вычисляемое значение</div>
                    </div>
                    <div class="project-type-btn ${project.type ===
```

```
      PROJECT_TYPE.RULE ? 'active' : ''}" data-type="${PROJECT_TYPE.RULE}">
4300                          <div class="type-icon">📋</div>
4301                          <div class="type-name">Правило</div>
4302                          <div class="type-desc">Логическое условие</div>
4303                      </div>
4304                  </div>
4305              </div>
4306
4307              <div id="parameter-fields" class="conditional-fields ${project.type ===
      PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
4308                  <div class="modal-row">
4309                      <label>Размерность:</label>
4310                      <input type="text" id="project-dimension" value="$
      {project.dimension || ''}" placeholder="Например: м/с, кг, °C">
4311                  </div>
4312              </div>
4313
4314              <div id="rule-fields" class="conditional-fields ${project.type ===
      PROJECT_TYPE.RULE ? 'visible' : ''}">
4315                  <div class="modal-row">
4316                      <label>Возможная причина:</label>
4317                      <textarea id="project-possible-cause" placeholder="Описание
      возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
4318                  </div>
4319                  <div class="modal-row">
4320                      <label>Методические указания:</label>
4321                      <textarea id="project-guidelines" placeholder="Инструкции и
      рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
4322                  </div>
4323              </div>
4324
4325              ${outputsHtml}
4326          `;
4327
4328          // Обработчики переключения типа
4329          content.querySelectorAll('.project-type-btn').forEach(btn => {
4330              btn.addEventListener('click', () => {
4331                  content.querySelectorAll('.project-type-btn').forEach(b =>
      b.classList.remove('active'));
4332                  btn.classList.add('active');
4333
4334                  const type = btn.dataset.type;
4335                  document.getElementById('parameter-
      fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
4336                  document.getElementById('rule-fields').classList.toggle('visible',
      type === PROJECT_TYPE.RULE);
4337              });
4338          });
4339
4340          this.showModal('project-modal-overlay');
4341      },
4342
4343      /**
4344       * Сохранить свойства проекта
4345       */
4346      saveProjectProperties() {
4347          const activeTypeBtn = document.querySelector('.project-type-btn.active');
4348          const type = activeTypeBtn ? activeTypeBtn.dataset.type :
      PROJECT_TYPE.PARAMETER;
4349
4350          AppState.project.code = document.getElementById('project-code').value;
4351          AppState.project.type = type;
4352
4353          if (type === PROJECT_TYPE.PARAMETER) {
4354              AppState.project.dimension = document.getElementById('project-
```

```
      dimension').value;
4355              AppState.project.possibleCause = '';
4356              AppState.project.guidelines = '';
4357          } else {
4358              AppState.project.dimension = '';
4359              AppState.project.possibleCause = document.getElementById('project-
      possible-cause').value;
4360              AppState.project.guidelines = document.getElementById('project-
      guidelines').value;
4361          }
4362
4363          this.hideModal('project-modal-overlay');
4364      }
4365  };
4366
4367  output.js:
4368  /**
4369   * Модуль управления выходными сигналами
4370   */
4371
4372  const Outputs = {
4373      /**
4374       * Обновление статуса выходных элементов
4375       * Вызывается при каждом изменении схемы
4376       */
4377      updateOutputStatus() {
4378          this.clearAllOutputHighlights();
4379          AppState.outputs.logical = [];
4380          AppState.outputs.numeric = [];
4381          updateFrameChildren();
4382
4383          // Обработка элементов-выходов
4384          Object.values(AppState.elements).forEach(elem => {
4385              if (!elem || elem.type !== 'output') return;
4386
4387              // Проверяем, к чему подключен вход этого выхода
4388              const inputConns = AppState.connections.filter(c =>
4389                  c.toElement === elem.id && c.toPort === 'in-0'
4390              );
4391
4392              // Каждое соединение к выходу — это отдельный выход
4393              inputConns.forEach((conn, index) => {
4394                  const fromElem = AppState.elements[conn.fromElement];
4395                  if (!fromElem) return;
4396
4397                  const outputType = conn.signalType;
4398                  const outputInfo = {
4399                      id: `${elem.id}_conn_${index}`,
4400                      elementId: elem.id,
4401                      sourceElementId: conn.fromElement,
4402                      sourcePort: conn.fromPort,
4403                      portIndex: 0,
4404                      portId: 'in-0',
4405                      type: outputType,
4406                      label: elem.props?.label || 'Выход',
4407                      elementType: 'output',
4408                      elementName: elem.props?.label || 'Выход',
4409                      name: elem.props?.label || 'Выход'
4410                  };
4411
4412                  if (outputType === SIGNAL_TYPE.LOGIC) {
4413                      AppState.outputs.logical.push(outputInfo);
4414                  } else if (outputType === SIGNAL_TYPE.NUMBER) {
4415                      AppState.outputs.numeric.push(outputInfo);
4416                  }
```

```
4417
4418                 // Подсветим входной порт
4419                 this.highlightOutputPort(elem.id, 0, outputType);
4420             });
4421         });
4422
4423         this.updateOutputCounter();
4424     },
4425
4426     /**
4427      * Очистка всех выделений выходов
4428      */
4429     clearAllOutputHighlights() {
4430         document.querySelectorAll('.port.output-active').forEach(port => {
4431             port.classList.remove('output-active');
4432         });
4433
4434         document.querySelectorAll('.element.has-output').forEach(elem => {
4435             elem.classList.remove('has-output');
4436         });
4437
4438         document.querySelectorAll('.element.output-ambiguous').forEach(el =>
      el.classList.remove('output-ambiguous'));
4439         document.querySelectorAll('.element.output-missing').forEach(el =>
      el.classList.remove('output-missing'));
4440     },
4441
4442     /**
4443      * Выделение выходного порта
4444      */
4445     highlightOutputPort(elemId, portIndex, portType) {
4446         const elem = document.getElementById(elemId);
4447         if (!elem) return;
4448
4449         const port = elem.querySelector(`.port.output[data-port="out-${portIndex}"]`);
4450         if (port) {
4451             port.classList.add('output-active');
4452         }
4453
4454         // Добавляем класс элементу (даёт общий визуал)
4455         elem.classList.add('has-output');
4456     },
4457
4458     /**
4459      * Обновление счётчика выходов в меню
4460      */
4461     updateOutputCounter() {
4462         const counter = document.getElementById('output-counter');
4463         if (counter) {
4464             const total = AppState.outputs.logical.length +
      AppState.outputs.numeric.length;
4465             counter.textContent = total;
4466             counter.style.display = total > 0 ? 'inline-block' : 'none';
4467         }
4468     },
4469
4470     /**
4471      * Получить все выходы для сохранения в проект
4472      */
4473     getOutputsForSave() {
4474         // Сохраняем информацию о frame/inner для рамок
4475         return {
4476             logical: AppState.outputs.logical.map(o => ({
4477                 id: o.id,
4478                 elementId: o.elementId,
```

```
                frameId: o.frameId || null,
                innerElementId: o.innerElementId || null,
                portIndex: o.portIndex ?? o.innerPortIndex ?? null,
                portLabel: o.label
            })),
            numeric: AppState.outputs.numeric.map(o => ({
                id: o.id,
                elementId: o.elementId,
                frameId: o.frameId || null,
                innerElementId: o.innerElementId || null,
                portIndex: o.portIndex ?? o.innerPortIndex ?? null,
                portLabel: o.label
            }))
        };
    },

    /**
     * Подсветить конкретный выход (при наведении в списке)
     */
    highlightOutput(elementId, highlight = true) {
        const elem = document.getElementById(elementId);
        if (elem) {
            if (highlight) {
                elem.classList.add('output-highlighted');
            } else {
                elem.classList.remove('output-highlighted');
            }
        }
    },

    /**
     * Перейти к элементу выхода на схеме (elementId — фокусируемый элемент; для рамок это id рамки)
     */
    navigateToOutput(elementId) {
        const elemData = AppState.elements[elementId];
        if (!elemData) return;

        // Центрируем viewport на элементе
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const centerX = elemData.x + elemData.width / 2;
        const centerY = elemData.y + elemData.height / 2;

        AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
        AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;

        Viewport.updateTransform();

        // Выделяем элемент
        Elements.selectElement(elementId);

        // Временная подсветка
        this.highlightOutput(elementId, true);
        setTimeout(() => this.highlightOutput(elementId, false), 2000);
    }
};

project.js:
/**
 * Модуль управления проектом (сохранение, загрузка)
 */

const Project = {
```

```
4543      /**
4544       * Инициализация
4545       */
4546          /**
4547       * Инициализация
4548       */
4549  init() {
4550      document.getElementById('btn-new').addEventListener('click', () =>
      this.newProject());
4551      document.getElementById('btn-save').addEventListener('click', () =>
      this.saveProject());
4552      document.getElementById('btn-load').addEventListener('click', () =>
      this.openProjectListModal());
4553      document.getElementById('btn-project-settings').addEventListener('click', () => {
4554          Modal.showProjectPropertiesModal();
4555      });
4556
4557      // Работа с модалкой выбора проекта
4558      this.projectList = [];
4559      this.filteredProjectList = [];
4560      this.selectedProjectFilename = null;
4561
4562      document.getElementById('project-cancel').addEventListener('click', () =>
      this.closeProjectListModal());
4563      document.getElementById('project-refresh').addEventListener('click', () =>
      this.refreshProjectList());
4564
4565      document.getElementById('project-load').addEventListener('click', () => {
4566          if (this.selectedProjectFilename) {
4567              this.loadProjectFromList(this.selectedProjectFilename);
4568          }
4569      });
4570
4571      document.getElementById('project-search').addEventListener('input', (event) => {
4572          this.filterProjectList(event.target.value);
4573      });
4574  },
4575
4576      /**
4577       * Новый проект
4578       */
4579      newProject() {
4580          if (Object.keys(AppState.elements).length > 0) {
4581              if (!confirm('Создать новый проект? Несохранённые изменения будут
      потеряны.')) {
4582                  return;
4583              }
4584          }
4585
4586          document.getElementById('workspace').innerHTML = '';
4587          document.getElementById('connections-svg').innerHTML = '';
4588
4589          resetState();
4590          Viewport.updateTransform();
4591      },
4592
4593          /**
4594       * Запрос имени файла и загрузка с сервера
4595       */
4596      async loadProjectPrompt() {
4597          const filename = window.prompt(
4598              "Введите имя файла проекта для загрузки (с сервера). Пример:
      scheme_logic.json",
4599              AppState.project.code ? `${AppState.project.code}_$
      {AppState.project.type}.json` : "scheme_type.json"
```

```
        );

        if (!filename) return; // Отмена

        try {
            // Используем обертку из Settings.js для запроса к /api/project/load
            const data = await Settings.loadProject(filename);

            // Если загрузка успешна, вызываем основную функцию обработки данных
            this._processLoadedData(data);
            alert(`Проект "${filename}" успешно загружен с сервера.`);

        } catch (error) {
            console.error('Ошибка загрузки проекта:', error);
            alert(`Ошибка загрузки проекта: ${error.message}`);
        }
    },

    /**
     * Сохранение проекта
     */
        async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
        // 1. Проверяем свойства проекта
        if (!AppState.project.code) {
            Modal.showProjectPropertiesModal();
            alert('Пожалуйста, укажите код проекта перед сохранением.');
            return;
        }

        // Обновляем размеры рамок перед сохранением
        updateFrameChildren();

        // 2. Сборка объекта проекта
        const project = {
            version: '1.0',
            project: AppState.project,
            elements: AppState.elements,
            connections: AppState.connections,
            counter: AppState.elementCounter,
            viewport: {
                zoom: AppState.viewport.zoom,
                panX: AppState.viewport.panX,
                panY: AppState.viewport.panY
            }
        };

        const filename = `${AppState.project.code || 'scheme'}_$
    {AppState.project.type}.json`;

        // 3. Сохранение на сервер
        try {
            await Settings.saveProject(filename, project);
            alert(`Проект успешно сохранен на сервере как: ${filename}`);
        } catch (error) {
            console.error('Ошибка сохранения проекта:', error);
            alert(`Ошибка сохранения проекта: ${error.message}`);
        }
    },

    async showProjectList() {
        try {
            const result = await Settings.listProjects(); // нужно реализовать в
    settings.js
            const list = result.projects || [];

```

```
4663              if (list.length === 0) {
4664                  alert('Проекты в папке не найдены.');
4665                  return;
4666              }
4667
4668              const choice = window.prompt(
4669                  'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code ||
      p.filename} — ${p.description}`).join('\n') +
4670                  '\n\nВведите номер проекта для загрузки:',
4671                  '1'
4672              );
4673              const index = parseInt(choice, 10) - 1;
4674              if (isNaN(index) || !list[index]) return;
4675
4676              await this.loadProjectByFilename(list[index].filename);
4677          } catch (error) {
4678              console.error(error);
4679              alert('Не удалось получить список проектов: ' + error.message);
4680          }
4681      },
4682
4683      async loadProjectByFilename(filename) {
4684          try {
4685              const data = await Settings.loadProject(filename);
4686              this._processLoadedData(data);
4687              alert(`Проект "${filename}" загружен.`);
4688          } catch (error) {
4689              console.error(error);
4690              alert('Ошибка загрузки проекта: ' + error.message);
4691          }
4692      },
4693
4694      openProjectListModal() {
4695    const modal = document.getElementById('modal-project-list');
4696    modal.classList.remove('hidden');
4697    document.body.classList.add('modal-open'); // если есть такой класс для блокировки
      скролла
4698    this.refreshProjectList();
4699  },
4700
4701  closeProjectListModal() {
4702    const modal = document.getElementById('modal-project-list');
4703    modal.classList.add('hidden');
4704    document.body.classList.remove('modal-open');
4705  },
4706
4707  async refreshProjectList() {
4708    const tbody = document.getElementById('project-list-body');
4709    tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка…</td></
      tr>`;
4710    try {
4711      const result = await Settings.listProjects();
4712      this.projectList = result.projects || [];
4713      this.filteredProjectList = [...this.projectList];
4714      this.renderProjectList();
4715    } catch (err) {
4716      console.error(err);
4717      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $
      {err.message}</td></tr>`;
4718    }
4719  },
4720
4721  renderProjectList() {
4722    const tbody = document.getElementById('project-list-body');
4723    const loadBtn = document.getElementById('project-load');
```

```
4724    loadBtn.disabled = true;
4725    this.selectedProjectFilename = null;
4726
4727    if (!this.filteredProjectList.length) {
4728      tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
      найдено</td></tr>`;
4729      return;
4730    }
4731
4732    tbody.innerHTML = '';
4733    this.filteredProjectList.forEach((item) => {
4734      const tr = document.createElement('tr');
4735      tr.innerHTML = `
4736        <td>${item.filename}</td>
4737        <td>${item.code || ''}</td>
4738        <td>${item.description || ''}</td>
4739        <td>${item.type || ''}</td>
4740      `;
4741      tr.addEventListener('click', () => {
4742        this.highlightRow(tr);
4743        this.selectedProjectFilename = item.filename;
4744        loadBtn.disabled = false;
4745      });
4746      tr.addEventListener('dblclick', () => {
4747        this.highlightRow(tr);
4748        this.selectedProjectFilename = item.filename;
4749        loadBtn.disabled = false;
4750        this.loadProjectFromList(item.filename);
4751      });
4752      tbody.appendChild(tr);
4753    });
4754  },
4755
4756  highlightRow(row) {
4757    const tbody = row.parentElement;
4758    [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
4759    row.classList.add('selected');
4760  },
4761
4762
4763  // Фильтр по поисковой строке
4764  filterProjectList(query) {
4765    const q = (query || '').trim().toLowerCase();
4766    if (!q) {
4767      this.filteredProjectList = [...this.projectList];
4768    } else {
4769      this.filteredProjectList = this.projectList.filter((item) => {
4770        return [
4771          item.filename,
4772          item.code,
4773          item.description,
4774          item.type
4775        ].some((field) => (field || '').toLowerCase().includes(q));
4776      });
4777    }
4778    this.renderProjectList();
4779  },
4780
4781  async loadProjectFromList(filename) {
4782    try {
4783      const data = await Settings.loadProject(filename);
4784      this._processLoadedData(data);
4785      this.closeProjectListModal();
4786      alert(`Проект "${filename}" успешно загружен.`);
4787    } catch (error) {
```

```
      console.error(error);
      alert('Ошибка загрузки проекта: ' + error.message);
    }
  },




      /**
       * Загрузка проекта
       */
      _processLoadedData(data) {
    try {
      document.getElementById('workspace').innerHTML = '';
      document.getElementById('connections-svg').innerHTML = '';
      resetState();

      if (data.project) {
        AppState.project = { ...AppState.project, ...data.project };
      }

      AppState.elementCounter = data.counter || 0;

      if (data.viewport) {
        AppState.viewport.zoom = data.viewport.zoom || 1;
        AppState.viewport.panX = data.viewport.panX || 0;
        AppState.viewport.panY = data.viewport.panY || 0;
      }

      const elements = data.elements || {};
      Object.values(elements)
        .filter(e => e.type === 'output-frame')
        .forEach(elemData => {
          Elements.addElement(
            elemData.type,
            elemData.x,
            elemData.y,
            elemData.props,
            elemData.id,
            elemData.width,
            elemData.height
          );
        });

      Object.values(elements)
        .filter(e => e.type !== 'output-frame')
        .forEach(elemData => {
          Elements.addElement(
            elemData.type,
            elemData.x,
            elemData.y,
            elemData.props,
            elemData.id,
            elemData.width,
            elemData.height
          );
        });

      AppState.connections = data.connections || [];
      AppState.elementCounter = Math.max(
        AppState.elementCounter,
        ...Object.values(elements).map(e => e.id || 0)
      );

      Viewport.updateTransform();
```

```
4853      Connections.drawConnections();
4854      updateFrameChildren();
4855    } catch (e) {
4856      alert('Ошибка обработки данных проекта: ' + e.message);
4857      console.error(e);
4858    }
4859  }
4860  };
4861
4862  settings.js:
4863  const Settings = {
4864    config: null,
4865
4866    async init() {
4867      // тянем настройки (не обязательно, но полезно)
4868      try {
4869        const r = await fetch('/api/settings');
4870        if (r.ok) this.config = await r.json();
4871      } catch (e) {
4872        console.warn('Settings load failed:', e);
4873      }
4874    },
4875
4876    async fetchSignals(mask, limit = 50) {
4877      const url = `/api/signals?q=${encodeURIComponent(mask || '')}&limit=$
      {encodeURIComponent(limit)}`;
4878      const r = await fetch(url);
4879      if (!r.ok) throw new Error('Failed to fetch signals');
4880      return await r.json(); // {items, total}
4881    },
4882    // ... в объекте Settings
4883
4884    async saveProject(filename, projectData) {
4885      if (!filename.endsWith('.json')) {
4886        filename += '.json';
4887      }
4888      const r = await fetch('/api/project/save', {
4889        method: 'POST',
4890        headers: { 'Content-Type': 'application/json' },
4891        body: JSON.stringify({
4892          filename: filename,
4893          content: projectData
4894        })
4895      });
4896      if (!r.ok) throw new Error('Failed to save project');
4897      return r.json();
4898    },
4899
4900    async listProjects() {
4901      const r = await fetch('/api/project/list');
4902      if (!r.ok) throw new Error('Failed to list projects');
4903      return r.json();
4904    },
4905
4906    async loadProject(filename) {
4907      if (!filename.endsWith('.json')) {
4908        filename += '.json';
4909      }
4910      const r = await fetch(`/api/project/load/${encodeURIComponent(filename)}`);
4911      if (!r.ok) {
4912          if (r.status === 404) {
4913               throw new Error(`Project "${filename}" not found (404)`);
4914          }
4915          throw new Error('Failed to load project');
4916      }
```

```
      return r.json();
    }

// ...
};

state.js:
/**
 * Глобальное состояние приложения
 */

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

    // Выделение
    selectedElement: null,

    // Перетаскивание
    draggingElement: null,
    dragOffset: { x: 0, y: 0 },
    isDraggingFromPalette: false,
    dragPreview: null,
    dragType: null,

    // Соединения
    connectingFrom: null,
    connectingFromType: null,
    tempLine: null,

    // Resize
    resizing: null,

    // Viewport (масштабирование и перемещение)
    viewport: {
        zoom: 1,
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    },

    // Свойства проекта
    project: {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        // Для параметра
        dimension: '',
        // Для правила
        possibleCause: '',
        guidelines: ''
    },

    // Выходные сигналы (автоматически определяются)
    outputs: {
        logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
        numeric: []     // Числовые выходы (формулы)
    }
};

/**
 * Сброс состояния
```

```
 */
function resetState() {
    AppState.elements = {};
    AppState.connections = [];
    AppState.elementCounter = 0;
    AppState.selectedElement = null;
    AppState.draggingElement = null;
    AppState.connectingFrom = null;
    AppState.tempLine = null;
    AppState.resizing = null;

    AppState.viewport = {
        zoom: 1,
        panX: 0,
        panY: 0,
        isPanning: false,
        lastMouseX: 0,
        lastMouseY: 0
    };

    AppState.project = {
        code: '',
        type: PROJECT_TYPE.PARAMETER,
        dimension: '',
        possibleCause: '',
        guidelines: ''
    };

    AppState.outputs = {
        logical: [],
        numeric: []
    };
}

utils.js:
/**
 * Вспомогательные функции
 */

/**
 * Генерация уникального ID
 */
function generateId() {
    AppState.elementCounter++;
    return `elem_${AppState.elementCounter}`;
}

function getInputPortType(elementId, portIdentifier) {
    const element = AppState.elements[elementId];
    if (!element) return SIGNAL_TYPE.ANY;

    const config = ELEMENT_TYPES[element.type];
    if (!config) return SIGNAL_TYPE.ANY;

    let portIndex = portIdentifier;

    // Обработка технического порта условия
    if (typeof portIdentifier === 'string') {
        if (portIdentifier === 'cond-0' && config.hasConditionPort) {
            return config.conditionPortType || SIGNAL_TYPE.LOGIC;
        }

        if (portIdentifier.startsWith('in-')) {
            portIndex = parseInt(portIdentifier.split('-')[1], 10);
        }
```

```
5047      }
5048
5049      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5050          portIndex = 0;
5051      }
5052
5053      // Динамические входы для AND/OR берут тип из конфига
5054      if ((element.type === 'and' || element.type === 'or')) {
5055          return SIGNAL_TYPE.LOGIC;  // Логические элементы всегда ожидают LOGIC на
      входе
5056      }
5057
5058      if (element.type === 'formula') {
5059          return SIGNAL_TYPE.ANY;
5060      }
5061
5062      const types = config.inputTypes || [];
5063      if (types.length === 0) return SIGNAL_TYPE.ANY;
5064
5065      if (portIndex < types.length) {
5066          return types[portIndex] || SIGNAL_TYPE.ANY;
5067      }
5068
5069      return types[types.length - 1] || SIGNAL_TYPE.ANY;
5070  }
5071
5072  function getOutputPortType(elementId, portIdentifier) {
5073      const element = AppState.elements[elementId];
5074      if (!element) return SIGNAL_TYPE.ANY;
5075
5076      const config = ELEMENT_TYPES[element.type];
5077      if (!config) return SIGNAL_TYPE.ANY;
5078
5079      let portIndex = portIdentifier;
5080
5081      if (typeof portIdentifier === 'string') {
5082          if (portIdentifier.startsWith('out-')) {
5083              portIndex = parseInt(portIdentifier.split('-')[1], 10);
5084          }
5085      }
5086
5087      if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5088          portIndex = 0;
5089      }
5090
5091      const types = config.outputTypes || [];
5092      if (types.length === 0) return SIGNAL_TYPE.ANY;
5093
5094      if (portIndex < types.length) {
5095          return types[portIndex] || SIGNAL_TYPE.ANY;
5096      }
5097
5098      return types[types.length - 1] || SIGNAL_TYPE.ANY;
5099  }
5100  /**
5101   * Проверка совместимости типов сигналов
5102   *
5103   * Новая логика:
5104   * - ANY совместим со всем
5105   * - TRUE совместим с LOGIC, TRUE, ANY
5106   * - FALSE совместим с LOGIC, FALSE, ANY
5107   * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
5108   * - NUMERIC совместим с NUMERIC, ANY
5109   */
5110  function areTypesCompatible(outputType, inputType) {
```

```
5111      // Если один из типов ANY - совместимы
5112      if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
5113          return true;
5114      }
5115
5116      // Если типы одинаковые - совместимы
5117      if (outputType === inputType) {
5118          return true;
5119      }
5120
5121      // TRUE/FALSE совместимы с LOGIC
5122      if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
5123          inputType === SIGNAL_TYPE.LOGIC) {
5124          return true;
5125      }
5126
5127      // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
5128      if (outputType === SIGNAL_TYPE.LOGIC &&
5129          (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
5130          return true;
5131      }
5132
5133      return false;
5134  }
5135
5136  /**
5137   * Проверка, находится ли элемент внутри рамки
5138   */
5139  function isInsideFrame(elemId, frameId) {
5140      const elem = AppState.elements[elemId];
5141      const frame = AppState.elements[frameId];
5142
5143      if (!elem || !frame || frame.type !== 'output-frame') return false;
5144
5145      const elemCenterX = elem.x + elem.width / 2;
5146      const elemCenterY = elem.y + elem.height / 2;
5147
5148      return elemCenterX > frame.x &&
5149             elemCenterX < frame.x + frame.width &&
5150             elemCenterY > frame.y &&
5151             elemCenterY < frame.y + frame.height;
5152  }
5153
5154  /**
5155   * Обновить принадлежность элементов к рамкам
5156   */
5157  function updateFrameChildren() {
5158      // Сначала очистим children у рамок и parentFrame у всех элементов
5159      Object.values(AppState.elements).forEach(elem => {
5160          if (elem.type === 'output-frame') {
5161              elem.children = [];
5162          } else {
5163              // удаляем parentFrame по умолчанию (пересчитаем ниже)
5164              if (elem.parentFrame) delete elem.parentFrame;
5165          }
5166      });
5167
5168      // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
      попадает
5169      Object.values(AppState.elements).forEach(elem => {
5170          if (!elem || elem.type === 'output-frame') return;
5171
5172          Object.values(AppState.elements).forEach(frame => {
5173              if (!frame || frame.type !== 'output-frame') return;
5174
```

```
            if (isInsideFrame(elem.id, frame.id)) {
                // добавляем в массив детей рамки
                frame.children.push(elem.id);
                // отмечаем у элемента родительскую рамку
                if (AppState.elements[elem.id]) {
                    AppState.elements[elem.id].parentFrame = frame.id;
                }
            }
        });
    });
}

/**
 * Преобразование координат экрана в координаты холста
 */
function screenToCanvas(screenX, screenY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
    const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;

    return { x, y };
}

/**
 * Преобразование координат холста в координаты экрана
 */
function canvasToScreen(canvasX, canvasY) {
    const container = document.getElementById('workspace-container');
    const rect = container.getBoundingClientRect();

    const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
    const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;

    return { x, y };
}

/**
 * Проверка, является ли порт выходным (не подключен к другим элементам)
 */
function isOutputPort(elemId, portIndex) {
    const portKey = `out-${portIndex}`;

    // Проверяем, есть ли соединения от этого порта
    const hasConnection = AppState.connections.some(conn =>
        conn.fromElement === elemId && conn.fromPort === portKey
    );

    return !hasConnection;
}

/**
 * Получить информацию о выходном порте
 */
function getOutputPortInfo(elemId, portIndex) {
    const elem = AppState.elements[elemId];
    if (!elem) return null;

    const config = ELEMENT_TYPES[elem.type];
    if (!config) return null;

    return {
        elementId: elemId,
        elementType: elem.type,
```

```
5240          elementName: config.name,
5241          portIndex: portIndex,
5242          portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
5243          portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
5244          // Дополнительная информация для идентификации
5245          displayName: `${config.name} → ${config.outputLabels?.[portIndex] || `out$
      {portIndex}`}`
5246      };
5247  }
5248
5249  viewport.js:
5250  /**
5251   * Модуль управления viewport (масштабирование и перемещение)
5252   */
5253
5254  const Viewport = {
5255      /**
5256       * Инициализация viewport
5257       */
5258      init() {
5259          this.setupZoomControls();
5260          this.setupPanning();
5261          this.setupMouseWheel();
5262          this.setupMinimap();
5263          this.setupCursorPosition();
5264          this.updateTransform();
5265          const container = document.getElementById('workspace-container');
5266          const rect = container.getBoundingClientRect();
5267          AppState.viewport.panX = 100; // немного отступить от левого края
5268          AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
      AppState.viewport.zoom;
5269          this.updateTransform();
5270      },
5271
5272      /**
5273       * Настройка кнопок масштабирования
5274       */
5275      setupZoomControls() {
5276          document.getElementById('btn-zoom-in').addEventListener('click', () => {
5277              this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
5278          });
5279
5280          document.getElementById('btn-zoom-out').addEventListener('click', () => {
5281              this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
5282          });
5283
5284          document.getElementById('btn-zoom-reset').addEventListener('click', () => {
5285              this.setZoom(1);
5286              this.setPan(0, 0);
5287          });
5288
5289          document.getElementById('btn-zoom-fit').addEventListener('click', () => {
5290              this.fitToContent();
5291          });
5292      },
5293
5294      /**
5295       * Настройка перемещения (pan)
5296       */
5297      setupPanning() {
5298          const container = document.getElementById('workspace-container');
5299
5300          container.addEventListener('mousedown', (e) => {
5301              // Средняя кнопка мыши или пробел + левая кнопка
5302              if (e.button === 1 || (e.button === 0 && e.target === container)) {
```

```
                e.preventDefault();
                AppState.viewport.isPanning = true;
                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
                container.style.cursor = 'grabbing';
            }
        });

        document.addEventListener('mousemove', (e) => {
            if (AppState.viewport.isPanning) {
                const dx = e.clientX - AppState.viewport.lastMouseX;
                const dy = e.clientY - AppState.viewport.lastMouseY;

                this.setPan(
                    AppState.viewport.panX + dx,
                    AppState.viewport.panY + dy
                );

                AppState.viewport.lastMouseX = e.clientX;
                AppState.viewport.lastMouseY = e.clientY;
            }
        });

        document.addEventListener('mouseup', (e) => {
            if (AppState.viewport.isPanning) {
                AppState.viewport.isPanning = false;
                document.getElementById('workspace-container').style.cursor = '';
            }
        });

        // Клавиша пробел для режима перемещения
        document.addEventListener('keydown', (e) => {
            if (e.code === 'Space' && !e.repeat) {
                document.getElementById('workspace-container').style.cursor = 'grab';
            }
        });

        document.addEventListener('keyup', (e) => {
            if (e.code === 'Space') {
                document.getElementById('workspace-container').style.cursor = '';
            }
        });
    },

    /**
     * Настройка масштабирования колесом мыши
     */
    setupMouseWheel() {
        const container = document.getElementById('workspace-container');

        container.addEventListener('wheel', (e) => {
            e.preventDefault();

            const rect = container.getBoundingClientRect();
            const mouseX = e.clientX - rect.left;
            const mouseY = e.clientY - rect.top;

            // Позиция мыши на холсте до масштабирования
            const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
    AppState.viewport.zoom;
            const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
    AppState.viewport.zoom;

            // Новый масштаб
            const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
```

```
      VIEWPORT_CONFIG.zoomStep;
5366              const newZoom = Math.max(
5367                  VIEWPORT_CONFIG.minZoom,
5368                  Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
5369              );
5370
5371              // Корректируем pan, чтобы точка под курсором осталась на месте
5372              const newPanX = mouseX - canvasPosBeforeX * newZoom;
5373              const newPanY = mouseY - canvasPosBeforeY * newZoom;
5374
5375              AppState.viewport.zoom = newZoom;
5376              AppState.viewport.panX = newPanX;
5377              AppState.viewport.panY = newPanY;
5378
5379              this.updateTransform();
5380          }, { passive: false });
5381      },
5382
5383      /**
5384       * Установить масштаб
5385       */
5386      setZoom(zoom) {
5387          const container = document.getElementById('workspace-container');
5388          const rect = container.getBoundingClientRect();
5389
5390          // Центр экрана
5391          const centerX = rect.width / 2;
5392          const centerY = rect.height / 2;
5393
5394          // Позиция центра на холсте
5395          const canvasCenterX = (centerX - AppState.viewport.panX) /
      AppState.viewport.zoom;
5396          const canvasCenterY = (centerY - AppState.viewport.panY) /
      AppState.viewport.zoom;
5397
5398          // Новый масштаб
5399          const newZoom = Math.max(
5400              VIEWPORT_CONFIG.minZoom,
5401              Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
5402          );
5403
5404          // Корректируем pan
5405          AppState.viewport.panX = centerX - canvasCenterX * newZoom;
5406          AppState.viewport.panY = centerY - canvasCenterY * newZoom;
5407          AppState.viewport.zoom = newZoom;
5408
5409          this.updateTransform();
5410      },
5411
5412      /**
5413       * Установить смещение
5414       */
5415      setPan(x, y) {
5416          AppState.viewport.panX = x;
5417          AppState.viewport.panY = y;
5418          this.updateTransform();
5419      },
5420
5421      /**
5422       * Вписать содержимое в экран
5423       */
5424      fitToContent() {
5425          const elements = Object.values(AppState.elements);
5426          if (elements.length === 0) {
5427              this.setZoom(1);
```

```
            this.setPan(0, 0);
            return;
        }

        // Находим границы содержимого
        let minX = Infinity, minY = Infinity;
        let maxX = -Infinity, maxY = -Infinity;

        elements.forEach(elem => {
            minX = Math.min(minX, elem.x);
            minY = Math.min(minY, elem.y);
            maxX = Math.max(maxX, elem.x + elem.width);
            maxY = Math.max(maxY, elem.y + elem.height);
        });

        const contentWidth = maxX - minX;
        const contentHeight = maxY - minY;

        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const padding = 50;
        const availableWidth = rect.width - padding * 2;
        const availableHeight = rect.height - padding * 2;

        const zoomX = availableWidth / contentWidth;
        const zoomY = availableHeight / contentHeight;
        const newZoom = Math.min(zoomX, zoomY, 1);

        AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
        AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
    (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
        AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
    (availableHeight - contentHeight * AppState.viewport.zoom) / 2;

        this.updateTransform();
    },

    /**
     * Обновить трансформацию
     */
    updateTransform() {
        const workspace = document.getElementById('workspace');
        const svg = document.getElementById('connections-svg');

        const transform = `translate(${AppState.viewport.panX}px, $
    {AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;

        workspace.style.transform = transform;
        svg.style.transform = transform;

        // Обновляем отображение масштаба
        document.getElementById('zoom-level').textContent = `$
    {Math.round(AppState.viewport.zoom * 100)}%`;

        // Обновляем мини-карту
        this.updateMinimap();
    },

    /**
     * Настройка мини-карты
     */
    setupMinimap() {
        const minimap = document.getElementById('minimap');
        const canvas = document.getElementById('minimap-canvas');
```

```

        canvas.width = MINIMAP_CONFIG.width;
        canvas.height = MINIMAP_CONFIG.height;

        // Клик по мини-карте для перемещения
        minimap.addEventListener('click', (e) => {
            const rect = minimap.getBoundingClientRect();
            const x = e.clientX - rect.left;
            const y = e.clientY - rect.top;

            this.navigateToMinimapPosition(x, y);
        });
    },

    /**
     * Обновить мини-карту
     */
    updateMinimap() {
        const canvas = document.getElementById('minimap-canvas');
        const ctx = canvas.getContext('2d');
        const viewportEl = document.getElementById('minimap-viewport');

        // Очищаем
        ctx.fillStyle = '#0a0a1a';
        ctx.fillRect(0, 0, canvas.width, canvas.height);

        // Масштаб мини-карты
        const scale = Math.min(
            canvas.width / VIEWPORT_CONFIG.canvasWidth,
            canvas.height / VIEWPORT_CONFIG.canvasHeight
        );

        // Рисуем элементы
        Object.values(AppState.elements).forEach(elem => {
            const x = elem.x * scale;
            const y = elem.y * scale;
            const w = Math.max(elem.width * scale, 2);
            const h = Math.max(elem.height * scale, 2);

            ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
            ctx.fillRect(x, y, w, h);
        });

        // Рисуем viewport
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();

        const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
        const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
        const vpW = (rect.width / AppState.viewport.zoom) * scale;
        const vpH = (rect.height / AppState.viewport.zoom) * scale;

        viewportEl.style.left = `${vpX}px`;
        viewportEl.style.top = `${vpY}px`;
        viewportEl.style.width = `${vpW}px`;
        viewportEl.style.height = `${vpH}px`;
    },

    /**
     * Перейти к позиции на мини-карте
     */
    navigateToMinimapPosition(minimapX, minimapY) {
        const canvas = document.getElementById('minimap-canvas');
        const container = document.getElementById('workspace-container');
        const rect = container.getBoundingClientRect();
```

```
5554
5555          const scale = Math.min(
5556              canvas.width / VIEWPORT_CONFIG.canvasWidth,
5557              canvas.height / VIEWPORT_CONFIG.canvasHeight
5558          );
5559
5560          const canvasX = minimapX / scale;
5561          const canvasY = minimapY / scale;
5562
5563          // Центрируем viewport на этой точке
5564          AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
5565          AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
5566
5567          this.updateTransform();
5568      },
5569
5570      /**
5571       * Отслеживание позиции курсора
5572       */
5573      setupCursorPosition() {
5574          const container = document.getElementById('workspace-container');
5575
5576          container.addEventListener('mousemove', (e) => {
5577              const pos = screenToCanvas(e.clientX, e.clientY);
5578              document.getElementById('cursor-pos').textContent =
5579                  `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)}`;
5580          });
5581      }
5582  };
5583
5584  main.py:
5585  import os
5586  import json
5587  from typing import Dict, List
5588  import pandas as pd
5589  from fastapi import FastAPI, HTTPException, Request, Response
5590  from fastapi.responses import JSONResponse
5591  from fastapi.staticfiles import StaticFiles
5592
5593  BASE_DIR = os.path.dirname(os.path.abspath(__file__))
5594  SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
5595
5596  def load_settings() -> Dict:
5597      with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
5598          return json.load(f)
5599
5600  def load_signals_from_folder(folder: str) -> List[Dict]:
5601      # folder может быть относительным
5602      folder_abs = folder if os.path.isabs(folder) else
      os.path.normpath(os.path.join(BASE_DIR, folder))
5603      if not os.path.isdir(folder_abs):
5604          raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")
5605
5606      signals_map = {}  # Tagname -> Description (последний wins)
5607      for name in os.listdir(folder_abs):
5608          if not name.lower().endswith(".csv"):
5609              continue
5610          path = os.path.join(folder_abs, name)
5611          try:
5612              df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
5613              df = df.dropna(subset=['Tagname'])
5614              for _, row in df.iterrows():
5615                  tag = str(row['Tagname']).strip()
5616                  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
}

@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)
    print(f"[OK] loaded signals: {len(STATE['signals'])}")

@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
    project_dir = folder if os.path.isabs(folder) else
os.path.normpath(os.path.join(BASE_DIR, folder))

    # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
    if '..' in filename or '/' in filename or '\\' in filename:
        raise HTTPException(status_code=400, detail="Invalid filename")
```

```
5681
5682      path = os.path.join(project_dir, filename)
5683      # Проверяем, что итоговый путь лежит внутри разрешенной директории
5684      if not path.startswith(project_dir):
5685           raise HTTPException(status_code=400, detail="Path traversal attempt")
5686
5687      return path
5688
5689  @app.post("/api/project/save")
5690  async def save_project(request: Request):
5691      try:
5692          data = await request.json()
5693          filename = data.get("filename")
5694          content = data.get("content")
5695
5696          if not filename or not content:
5697              raise HTTPException(status_code=400, detail="Filename and content are
      required")
5698
5699          path = get_project_path(filename)
5700
5701          # Сохраняем как JSON
5702          with open(path, "w", encoding="utf-8") as f:
5703              json.dump(content, f, indent=2)
5704
5705          return {"status": "ok", "message": f"Project saved to {filename}"}
5706
5707      except HTTPException as e:
5708          raise e
5709      except Exception as e:
5710          print(f"Error saving project: {e}")
5711          raise HTTPException(status_code=500, detail="Internal server error during
      save")
5712
5713  @app.get("/api/project/load/{filename}")
5714  def load_project(filename: str):
5715      try:
5716          path = get_project_path(filename)
5717
5718          if not os.path.exists(path):
5719              raise HTTPException(status_code=404, detail="Project not found")
5720
5721          with open(path, "r", encoding="utf-8") as f:
5722              content = json.load(f)
5723
5724          return content
5725
5726      except HTTPException as e:
5727          raise e
5728      except Exception as e:
5729          print(f"Error loading project: {e}")
5730          raise HTTPException(status_code=500, detail="Internal server error during
      load")
5731
5732  @app.get("/api/project/list")
5733  def list_projects():
5734      folder = STATE["settings"].get("projectDataFolder")
5735      if not folder:
5736          raise HTTPException(status_code=500, detail="Project folder not configured")
5737
5738      project_dir = folder if os.path.isabs(folder) else
      os.path.normpath(os.path.join(BASE_DIR, folder))
5739      os.makedirs(project_dir, exist_ok=True)
5740
5741      projects = []
```

```
    for fname in sorted(os.listdir(project_dir)):
        if not fname.endswith(".json"):
            continue
        path = os.path.join(project_dir, fname)
        try:
            with open(path, "r", encoding="utf-8") as f:
                payload = json.load(f)
        except Exception:
            continue
        project_meta = payload.get("project", {})
        projects.append({
            "filename": fname,
            "code": project_meta.get("code") or project_meta.get("tagname") or "",
            "description": project_meta.get("description") or "",
            "type": project_meta.get("type") or ""
        })
    return {"projects": projects}


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