INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* BETIM
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Wdson Filipe Marçal

Desenvolvimento de Macrocélulas Lógicas Modulares em FPGA para Controle Automatizado de Processos

Betim
2025

WDSON FILIPE MARÇAL

Desenvolvimento de Macrocélulas Lógicas Modulares em FPGA para Controle Automatizado de Processos

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Me. Helbert Ribeiro de Sá

Betim

2025

FICHA CATALOGRÁFICA

M313d Marçal, Wdson Filipe

Desenvolvimento de macrocélulas lógicas modulares em FPGA para controle automatizado de processos / Wdson Filipe Marçal. – 2025.

105 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2025.

Orientação: Prof. Me. Helbert Ribeiro de Sá

1. Lógica digital. 2. Planta didática. 3. Semicndutores. 4. Informações digitais. 5. Engenharia de Controle e Automação. I Marçal, Wdson Filipe. II. Título.

CDU: 681.5

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

Wdson Filipe Marçal

Desenvolvimento de Macrocélulas Lógicas Modulares em FPGA para Controle Automatizado de Processos

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovado em: 25/07/2025 pela banca examinadora:

Digitally signed by Helbert Ribeiro de Sá
DN: cn=Helbert Ribeiro de Sá gn=Helbert Ribeiro de Sá c=BR Brazil l=BR Brazil o=Betim ou=IFMG e=helbert.desa@ifmg.edu.br
Reason: Confirmo assinatura
Location: IFMG
Date: 2025-07-30 20:35-03:00

Prof. Me. Helbert Ribeiro de Sá (Orientador) – IFMG Campus Betim

gov.br Documento assinado digitalmente
WELINTON LA FONTAINE LOPES
Data: 30/07/2025 15:39:28-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Welinton La Fontaine Lopes – IFMG Campus Betim

gov.br Documento assinado digitalmente
DANIEL ALMEIDA GODINHO
Data: 29/07/2025 23:30:45-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Daniel Almeida Godinho – IFMG Campus Betim

RESUMO

Este trabalho descreve o Desenvolvimento de Macrocélulas Lógicas Modulares em FPGA para Controle Automatizado de Processos. Ele apresenta a criação de macrocélulas reutilizáveis, que são blocos de hardware sintetizados e configurados na própria FPGA, e que podem ser instanciadas múltiplas vezes para uso em diferentes partes do processo, desenvolvidas por meio de diagramas gráficos (BDF) e linguagem VHDL. A proposta é aplicada a uma planta didática que simula um processo de mistura e transferência de materiais, permitindo explorar práticas de controle digital sequencial. A abordagem modular adotada visa facilitar a implementação lógica, reduzir a complexidade do desenvolvimento e promover maior acessibilidade à tecnologia FPGA por profissionais com diferentes níveis de familiaridade com linguagens de descrição de hardware. A lógica implementada foi estruturada em etapas de controle, envolvendo temporização, leitura de sensores e acionamento de atuadores. A validação prática demonstrou a funcionalidade esperada da solução proposta, com operação estável e coerente com os objetivos definidos. Os resultados obtidos confirmam a viabilidade da aplicação das macrocélulas em sistemas de automação didáticos, reforçando o potencial do uso de FPGAs em projetos educacionais e industriais.

Palavras-chave: FPGA, automação, macrocélulas, lógica digital, planta didática.

ABSTRACT

This study describes the Development of Modular Logic Macrocells in FPGA for Automated Process Control. It presents the creation of reusable macrocells, which are hardware blocks synthesized and configured within the FPGA itself, and which can be instantiated multiple times for use in different parts of the process, developed through graphical diagrams (BDF) and VHDL language. The proposal is applied to a didactic plant that simulates a material mixing and transfer process, allowing the exploration of sequential digital control practices. The adopted modular approach aims to facilitate logical implementation, reduce development complexity, and promote greater accessibility to FPGA technology for professionals with different levels of familiarity with hardware description languages. The implemented logic was structured into control stages involving timing, sensor reading, and actuator activation. Practical validation demonstrated the expected functionality of the proposed solution, with stable operation consistent with the defined objectives. The obtained results confirm the feasibility of applying macrocells in didactic automation systems, reinforcing the potential of FPGA use in educational and industrial projects.

Keywords: FPGA, automation, macrocells, digital logic, didactic plant.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
VHDL	*Very High-Speed Integrated Circuit Hardware Description Language*
BDF	*Block Diagram File*
IO	*Input/Output*
CLB	*Configurable Logic Blocks*
LUT	*Look-Up Tables*
GND	*Ground*
PPP	*Pronto Para Partida*
CLK	*Clock*
EN	*Enable*
LVTTL	*Low Voltage Transistor-Transistor Logic*
EDA	*Electronic Design Automation*
IDE	*Integrated Development Environment*
NA	*Normalmente Aberto*
NF	*Normalmente Fechado*
C	*Comum*
LED	*Light Emitting Diode*
ATX	*Advanced Technology Extended*
JTAG	*Joint Test Action Group*

SUMÁRIO

1 INTRODUÇÃO

A utilização de dispositivos baseados na arquitetura FPGA (*Field-Programmable Gate Array*) tem despertado crescente interesse na comunidade científica e tecnológica, em virtude de suas notáveis capacidades de síntese de hardware. Diferentemente dos microcontroladores convencionais, que operam por meio da execução sequencial de instruções de código, os FPGAs são projetados para sintetizar múltiplos blocos lógicos que operam de forma paralela e independente, o que representa uma significativa vantagem em termos de desempenho e flexibilidade de aplicação (PAIVA, 2019).

Nesse cenário, torna-se essencial compreender não apenas as potencialidades oferecidas pelas FPGAs, mas também os desafios inerentes à sua programação e implementação. A aplicação dessa tecnologia em sistemas de controle baseados em lógica combinacional evidencia sua versatilidade, sobretudo quando se adota o uso de macrocélulas como recurso visual que facilita a modelagem e a programação, em contraste com as abordagens convencionais baseadas em linguagens textuais de descrição de hardware.

A principal vantagem da incorporação de FPGAs reside na possibilidade de definir diversos blocos de hardware que operam simultaneamente, proporcionando um expressivo ganho de capacidade computacional nos sistemas em que são empregados (JIANG et al., 2025). Ademais, os ambientes de desenvolvimento voltados para essa arquitetura possibilitam maior eficiência no processo de projeto, promovendo economia de tempo e de recursos quando comparados aos métodos tradicionais. Esses ambientes oferecem suporte a simulações e validações em estágios iniciais, tanto em protótipos quanto na versão final do sistema, com o auxílio de ferramentas robustas de automação de projeto eletrônico (EDA – *Electronic Design Automation*).

Cabe destacar que as macrocélulas, enquanto recurso gráfico, proporcionam uma abordagem mais intuitiva e acessível à configuração de circuitos, favorecendo a compreensão dos projetos, sobretudo para usuários menos familiarizados com linguagens de descrição de hardware como VHDL ou Verilog. Contudo, mesmo com esses benefícios, a programação de FPGAs ainda pode apresentar elevada complexidade em aplicações que exigem níveis avançados de abstração e controle, o que reforça a necessidade de domínio técnico e metodológico por parte dos

desenvolvedores. Essas macrocélulas, que são blocos de hardware sintetizados e configurados na própria FPGA, operam de forma paralela, proporcionando uma abstração visual que simplifica a complexidade da programação e otimiza a reconfigurabilidade do sistema.

1.1 Justificativa

A exploração da tecnologia FPGA (*Field-Programmable Gate Array*) nas diversas áreas da engenharia configura-se como um campo de pesquisa promissor, especialmente em razão da característica reconfigurável desses dispositivos, o que lhes confere elevada versatilidade em uma ampla gama de aplicações. Tradicionalmente, os FPGAs têm sido amplamente empregados em sistemas embarcados, no processamento de sinais digitais, na automação industrial, entre

requisitos críticos. Com o avanço tecnológico e o desenvolvimento de dispositivos cada vez mais potentes, novas possibilidades têm emergido para a aplicação dessa tecnologia em contextos ainda mais variados e complexos.

Com o intuito de evidenciar como o desenvolvimento pode ser compreendido de forma mais acessível, este trabalho demonstra a aplicação de macrocélulas como recurso para simplificar a programação de FPGAs, sem restringir-se apenas a especialistas. A programação tradicional desses dispositivos exige conhecimentos avançados em linguagens de descrição de hardware, como o VHDL (Very High-Speed Integrated Circuit Hardware Description Language), o que frequentemente representa uma barreira técnica significativa. Nesse sentido, a adoção de abordagens visuais baseadas em macrocélulas pode democratizar o uso dos FPGAs, permitindo que profissionais sem domínio em linguagens de baixo nível também possam projetar e implementar soluções de hardware de forma eficaz.

As macrocélulas atuam como blocos funcionais encapsulados, com entradas, saídas e lógicas previamente definidas, possibilitando a montagem de circuitos por meio da interligação desses blocos, sem a necessidade de escrever códigos complexos em HDL (*Hardware Description Language*). Tal abordagem reduz a complexidade do processo de desenvolvimento, contribuindo para uma curva de aprendizado menos íngreme e incentivando a adoção da tecnologia por profissionais de diferentes áreas do conhecimento.

Dessa forma, a simplificação da programação por meio de macrocélulas não apenas amplia o acesso à tecnologia FPGA, como também potencializa a criação de soluções personalizadas em contextos específicos. O presente estudo tem como objetivo explorar essa abordagem por meio da implementação em uma planta didática, a qual será utilizada como plataforma experimental para testar e validar a funcionalidade das macrocélulas e demonstrar a viabilidade da técnica proposta.

1.2 Definição do Problema

De que maneira as macrocélulas em dispositivos FPGA podem ser empregadas de forma eficaz na simplificação do processo de programação de circuitos digitais, aplicados em uma planta didática?

1.3 Objetivos

1.3.1 Objetivo geral

Este trabalho tem como objetivo investigar a utilização de abordagens visuais baseadas em macrocélulas como estratégia para tornar a programação de dispositivos FPGA mais acessível e eficiente para engenheiros e demais profissionais da área tecnológica. O foco central da pesquisa está na elaboração de estruturas de controle lógico booleano voltadas para o desenvolvimento de um circuito de controle aplicado a uma planta didática.

1.3.2 Objetivos Específicos

1. Definir os parâmetros delimitadores da planta didática;
2. Desenvolver o projeto elétrico e montagem física da planta didática.
3. Estruturar blocos de macrocélulas de forma a desmembrar os circuitos da planta didática em partes menores e definidas;
4. Testar e validar o uso das macrocélulas do FPGA em conjunto com a placa de I/O (*Input/Output*) no controle da planta didática.

1.4 Organização do Trabalho

O Capítulo 1 apresenta uma introdução ao tema abordado, explicitando os objetivos gerais e específicos da pesquisa, bem como a justificativa que fundamenta a realização do estudo. O Capítulo 2 reúne a fundamentação teórica indispensável para a compreensão dos conceitos envolvidos, oferecendo embasamento técnico e científico para o desenvolvimento do projeto. O Capítulo 3 descreve a metodologia empregada, detalhando as etapas seguidas desde a concepção da planta didática até a modelagem das macrocélulas utilizadas no sistema.

No Capítulo 4, é exposto o desenvolvimento prático do sistema proposto, abrangendo a estruturação da lógica digital e a montagem do hardware necessário para sua implementação. O Capítulo 5 é dedicado à validação do sistema, com a apresentação dos testes funcionais realizados e dos ajustes estruturais efetuados com base nos resultados obtidos. O Capítulo 6 contempla as considerações finais, nas quais são discutidos os principais achados da pesquisa, bem como a análise dos resultados à luz dos objetivos inicialmente estabelecidos.

2 FUNDAMENTAÇÃO TEÓRICA

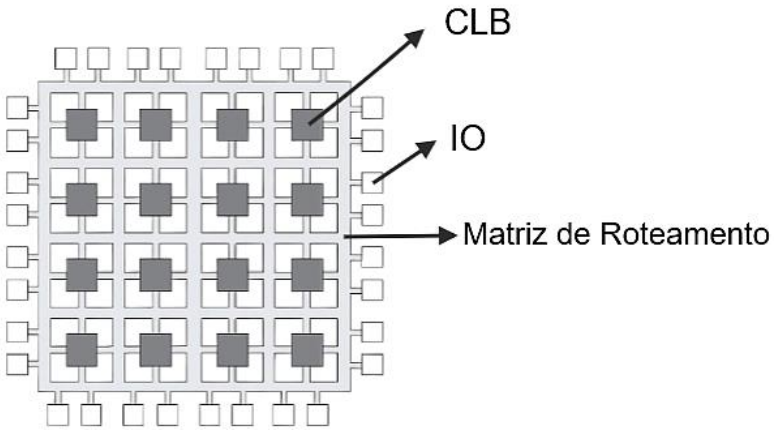
2.1 Arquitetura FPGA

A compreensão da arquitetura interna dos dispositivos FPGA é essencial para a adequada exploração de seu potencial como plataforma reconfigurável. Essa arquitetura é formada por blocos modulares que permitem a implementação de circuitos digitais personalizados, adequando-se a distintas necessidades e aplicações. Na sequência, apresentam-se os principais componentes dessa estrutura, com ênfase nas funções desempenhadas pelos blocos lógicos, pelas interfaces de entrada e saída, bem como pela matriz de roteamento programável.

2.1.1 Estrutura Interna e Componentes

O Field-Programmable Gate Array (FPGA) é um circuito integrado programável usado para implementar sistemas digitais arbitrários. A estrutura básica dos FPGAs consiste em dois elementos: os blocos lógicos configuráveis *(Configurable Logic Blocks – CLBs*) e os blocos de entrada e saída (*Input/Output – IOs*), conforme ilustrado na Figura 1.

Figura 1 – Estrutura interna típica de um FPGA.



Fonte: Adaptado de WEBER, 2016.

Os CLBs são compostos por tabelas de consulta (*Look-Up Tables – LUTs*) e flip-flops, que permitem implementar em paralelo e em série, circuitos de lógica combinacional e sequencial (WEBER, 2016). Cabe às LUTs realizar as funções de

lógica booleana e aos flip-flops, o armazenamento dos estados dos sinais. A comunicação entre CLBs e os IOs é feita por meio de uma matriz de roteamento programável, que consiste em uma rede de interconexões que estabelece malhas de caminhos lógicos entre as tabelas de consulta, os flip-flops e entre as entradas e saídas.

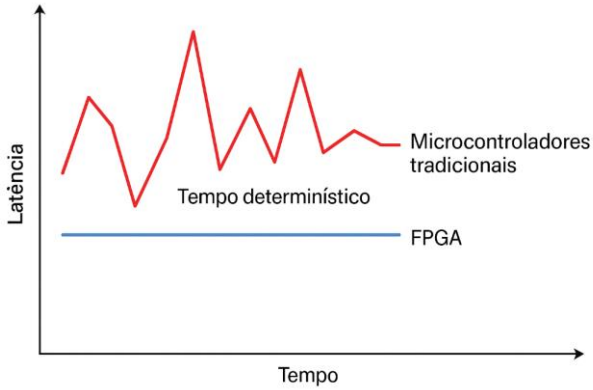
Os IOs por sua vez, estabelecem a interface entre a FPGA e o meio externo, podendo ser configurados de acordo com diferentes padrões elétricos predeterminados, esta versatilidade assegura a compatibilidade com uma gama de dispositivos periféricos.

A arquitetura modular dos FPGAs oferece grande flexibilidade no desenvolvimento de aplicações específicas, possibilitando atualizações na lógica sem a necessidade de modificações físicas no hardware.

2.1.2 Funcionamento Paralelo e Tempo Determinístico

Enquanto microcontroladores tradicionais operam por meio da execução sequencial de instruções, os dispositivos FPGA funcionam com base em lógica concorrente e paralela, permitindo que múltiplos blocos lógicos atuem de forma independente e simultânea. Essa característica possibilita a realização de diversas tarefas ao mesmo tempo, sem a necessidade de interrupções no fluxo de execução, o que contribui significativamente para a redução da latência do sistema. Tal redução é especialmente relevante em aplicações que exigem controle em tempo real, como sistemas embarcados e automação industrial (PAIVA, 2019). A Figura 2 apresenta uma representação esquemática comparativa entre a latência típica de microcontroladores convencionais e a de sistemas baseados em FPGA.

Figura 2 – Representação da latência: Microcontrolador vs FPGA.



Fonte: Elaborado pelo Autor, 2025.

O paralelismo inerente à arquitetura FPGA também contribui para uma elevada previsibilidade temporal, característica conhecida como tempo determinístico. Essa propriedade refere-se à capacidade do sistema de responder de forma precisa e consistente dentro dos intervalos de tempo previamente estabelecidos. Em ambientes industriais, essa previsibilidade é essencial para aplicações que exigem alto grau de sincronismo, como no caso de máquinas CNC e robôs industriais. A baixa latência e a precisão no controle dos sinais de entrada e saída promovem repetibilidade, confiabilidade e segurança operacional, especialmente em sistemas de automação síncrona e de alta velocidade, como aqueles utilizados em manipulação robótica, processos de corte automatizado e controle de componentes mecânicos (NADIR et al., 2016; BOLTON, 2009).

Adicionalmente, nas modernas linhas de produção, o controle determinístico de eventos torna-se indispensável para a prevenção de falhas intermitentes, a otimização do desempenho operacional e o cumprimento de rigorosos padrões de segurança funcional, como os estabelecidos pela norma IEC 61508, frequentemente exigidos em ambientes industriais críticos de alta produtividade.

2.1.3 Considerações sobre Aplicabilidade

A escolha pela utilização de uma FPGA neste projeto justifica-se por sua capacidade de executar múltiplas tarefas simultaneamente, com elevada velocidade, confiabilidade e flexibilidade. Essa tecnologia possibilita a reconfiguração lógica do hardware, o que representa uma vantagem significativa tanto em contextos didáticos

quant em ambientes industriais dinâmicos e sujeitos a constantes mudanças (CRESPO, 2013).

Entre os benefícios adicionais, destaca-se a possibilidade de construção de blocos funcionais reutilizáveis — as chamadas macrocélulas — que encapsulam funções específicas, como eliminação de ruído (*debounce*), controle de motores e lógica de temporização. Esses blocos permitem a criação de bibliotecas modulares que podem ser aplicadas a diferentes projetos, promovendo a padronização, a redução do tempo de desenvolvimento e a simplificação dos processos de manutenção.

No contexto deste trabalho, a FPGA não se limita a substituir controladores tradicionais, mas apresenta-se como uma alternativa flexível e adaptável para o controle de plantas automatizadas compostas por múltiplas etapas, tais como esteiras transportadoras, tanques de mistura e válvulas de transferência. Essa aplicação evidencia o potencial da tecnologia tanto como ferramenta de ensino quanto como plataforma para prototipagem de soluções industriais com relevância prática.

2.2 Linguagem HDL

As linguagens de descrição de hardware, conhecidas como HDL (*Hardware Description Language*), são fundamentais no desenvolvimento de projetos com dispositivos FPGA, uma vez que possibilitam a especificação do comportamento e da estrutura do hardware em diferentes níveis de abstração. As duas principais linguagens utilizadas nesse contexto são o VHDL (*VHSIC Hardware Description Language*) e o Verilog HDL. Ambas permitem a modelagem de circuitos digitais de forma estruturada, clara e eficiente (D’AMORE, 2012).

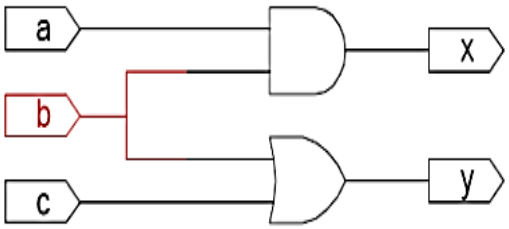
Neste projeto, a linguagem adotada é o VHDL, cuja sintaxe é fortemente tipada e compartilha diversas características com linguagens de programação de alto nível. O VHDL é particularmente recomendado em projetos que demandam elevado controle estrutural e organizacional, bem como em situações que envolvem sistemas complexos e com múltiplos níveis de hierarquia (D’AMORE, 2012).

É importante destacar que, ao projetar um circuito utilizando HDL, o projetista não está desenvolvendo um programa no sentido tradicional, mas sim descrevendo, de forma textual, o comportamento funcional do hardware desejado. Essa descrição é posteriormente sintetizada e convertida em um arquivo binário de configuração, o

qual será gravado na FPGA, determinando seu funcionamento. A partir dessa abordagem, torna-se possível projetar desde circuitos básicos, como portas lógicas, até sistemas completos de controle de processos industriais.

Na Figura 3, apresenta-se uma ilustração comparativa entre um circuito lógico, composto por portas AND e OR, e sua respectiva descrição em VHDL. À esquerda está o diagrama lógico do circuito, enquanto à direita encontra-se o código VHDL correspondente.

Figura 3 –Descrição em VHDL das portas lógicas AND e OR.




```
1  ENTITY portas IS
2    PORT (a, b, c : IN  BIT;
3          x, y    : OUT BIT);
4  END portas;
5
6  ARCHITECTURE teste OF portas IS
7  BEGIN
8    x <= a AND b;
9    y <= c OR  b;
10 END teste;
```

Fonte: Codá, 2020.

2.3 Representação Lógica Gráfica

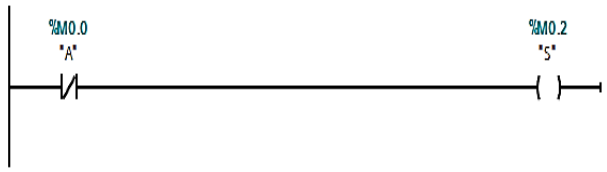
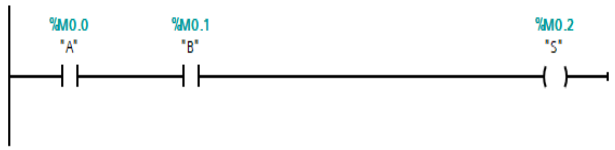
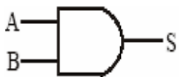
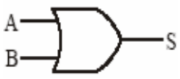
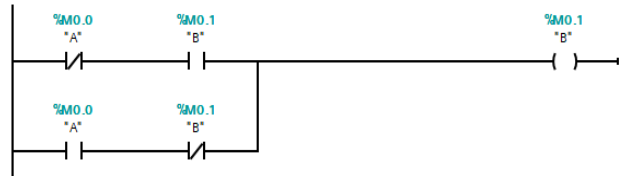
No contexto da automação industrial, a programação em Controladores Lógicos Programáveis (CLPs) utilizando a linguagem Ladder estabelece uma correspondência direta com os fundamentos da lógica booleana. Essa linguagem gráfica é amplamente utilizada por sua intuitividade e facilidade de interpretação, especialmente em ambientes industriais onde decisões precisam ser tomadas com base em condições lógicas.

A lógica Ladder se baseia em elementos gráficos, como contatos normalmente abertos (NO), normalmente fechados (NF) e bobinas, que refletem operações lógicas como AND, OR, NOT e XOR. A interpretação visual desses elementos torna a construção e o entendimento das estruturas de controle mais acessíveis, mesmo para profissionais com pouca familiaridade com linguagens de programação textual. Segundo Tocci, Widmer e Moss (2007), o domínio da lógica booleana é essencial para a síntese de circuitos digitais, pois fornece uma estrutura simbólica clara e sistemática

para representar e implementar funções lógicas. Na prática, um circuito com dois contatos em série representa uma operação AND; contatos em paralelo representam uma operação OR; e o uso de contatos NF permite a implementação da lógica NOT. Essa representação gráfica facilita tanto a implementação quanto a depuração de circuitos lógicos industriais.

A Tabela 1 apresenta a correspondência direta entre as representações gráficas na linguagem Ladder e os símbolos tradicionais das portas lógicas digitais, demonstrando que as mesmas estruturas podem ser aplicadas tanto no ambiente de programação de CLPs quanto na lógica digital implementada em hardware.

Tabela 1 – Correspondência entre lógica digital e Ladder.

	Ladder	Porta Lógica
NOT		
AND		
OR		
XOR		

Fonte: Elaborado pelo Autor, 2025.

Diferentemente de blocos de função em Controladores Lógicos Programáveis, que operam em cima de uma abstração de memória e são executados sequencialmente por um processador, as macrocélulas em FPGAs correspondem a circuitos de hardware físicos dedicados. A reutilização, neste caso, significa a capacidade de instanciar múltiplas cópias desse hardware na FPGA, cada um

operando em paralelismo como um hardware individual, o que garante um tempo de resposta determinístico e otimiza o uso dos recursos lógicos do dispositivo.

A equivalência entre a lógica booleana e a construção de macrocélulas digitais reutilizáveis, fundamentadas em portas lógicas, é essencial para o desenvolvimento deste trabalho, cujo objetivo é o controle de processos industriais por meio de dispositivos FPGA. Ao estabelecer a correlação entre os conceitos empregados em Controladores Lógicos Programáveis (CLPs) e os blocos digitais implementados por meio de linguagens de descrição de hardware, busca-se evidenciar que os mesmos princípios de controle utilizados na linguagem Ladder podem ser reproduzidos em sistemas digitais dedicados, proporcionando maior flexibilidade e desempenho.

Nesse sentido, a lógica Ladder funciona como uma interface visual que conecta o ambiente industrial tradicional à lógica digital estruturada, facilitando a migração, otimização e reimplementação de estruturas de controle em circuitos digitais baseados em FPGA. A clareza decorrente dessa representação gráfica favorece não apenas as fases de projeto, mas também os processos de validação, manutenção e documentação dos sistemas desenvolvidos.

Dentro desse contexto, a abordagem prática adotada neste trabalho consiste na utilização de macrocélulas digitais — blocos lógicos funcionais com comportamento encapsulado e bem definido, que podem ser reutilizados em diferentes partes do sistema sem a necessidade de reescrever sua lógica interna. Exemplos típicos de macrocélulas incluem temporizadores, contadores, registradores e blocos de debounce. A adoção dessa estratégia promove modularidade, minimiza erros decorrentes de replicações inadequadas e reduz o tempo de desenvolvimento, características particularmente relevantes para sistemas digitais implementados em FPGA. A implementação e aplicação dessas macrocélulas serão detalhadas no capítulo de desenvolvimento.

2.4 Hardware

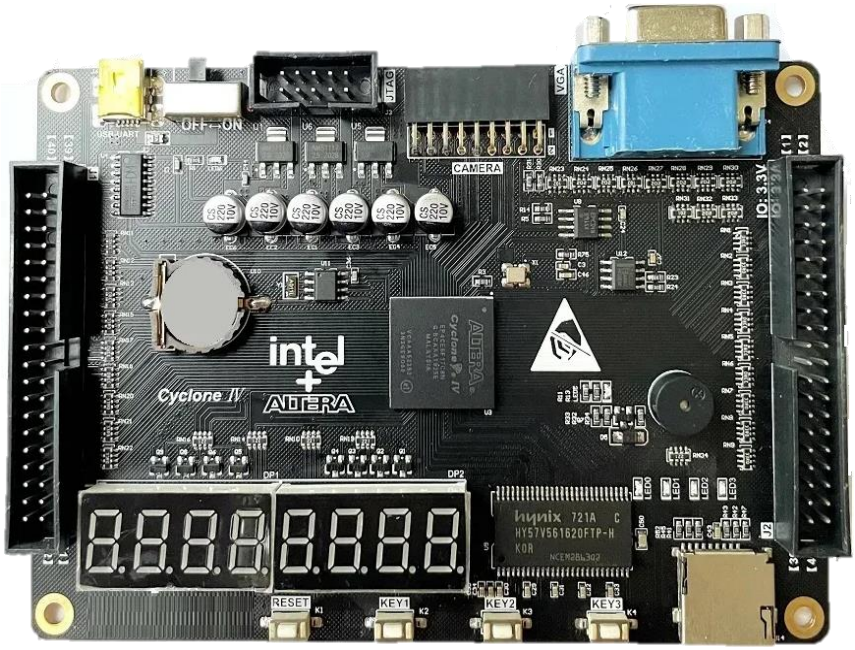
O sistema desenvolvido neste trabalho baseia-se na integração entre elementos físicos e uma plataforma digital reconfigurável. Nesta seção, são apresentados os principais componentes de hardware utilizados na implementação do projeto, com ênfase na placa FPGA, interfaces de entrada e saída, módulos de relé

e demais dispositivos auxiliares que compõem a estrutura física do controle automatizado.

2.4.1 FPGA e Placa Base

O hardware principal utilizado neste projeto consiste em uma placa de desenvolvimento equipada com uma FPGA da família Cyclone IV, modelo EP4CE6F17I7, compatível com o ambiente de desenvolvimento Quartus II 13.1, da Intel/Altera. Essa placa dispõe de uma interface prática para conexão com dispositivos externos por meio do conector IDC40, além de incorporar os principais recursos necessários ao desenvolvimento de lógica digital, tais como um clock interno operando a 50 MHz, entradas e saídas digitais acessíveis via pinos dedicados, e suporte à alimentação elétrica de 3,3 V. A Figura 4 ilustra a placa de desenvolvimento empregada no presente estudo.

Figura 4 – Placa de desenvolvimento FPGA EP4CE6F17I7.



Fonte: Elaborado pelo Autor, 2025.

A escolha desta placa de desenvolvimento justifica-se por sua ampla disponibilidade e pela boa documentação técnica. A placa apresenta conectores para gravação via cabo USB Blaster, LEDs para testes de saída, *push-buttons* para entradas lógicas e compatibilidade com barramentos de expansão. Esses elementos

facilitaram tanto o desenvolvimento quanto a validação experimental das lógicas implementadas no projeto. Para fins de referência, a documentação técnica da placa encontra-se no Anexo A.

2.4.2 USB-Blaster

Para a transferência do projeto desenvolvido no ambiente Quartus II para a FPGA, utiliza-se o conversor USB-Blaster, compatível com o padrão de configuração JTAG. Este dispositivo atua como uma interface confiável entre o software Quartus II 13.1 e a FPGA EP4CE6F17I7, possibilitando a gravação do *bitstream* gerado após o processo de síntese diretamente na placa por meio de um conector padrão de 10 pinos.

A interface JTAG (*Joint Test Action Group*) é adotada como padrão de configuração interna nas FPGAs da família Cyclone IV (INTEL CORPORATION, 2009). O emprego do USB-Blaster proporciona gravações ágeis e seguras durante o ciclo de desenvolvimento, facilitando a realização de testes funcionais iterativos e validações experimentais. Conforme especificado no manual da Intel, o cabo USB-Blaster suporta níveis de tensão de entrada e saída entre 1,5 V e 3,3 V, garantindo compatibilidade com os níveis lógicos da FPGA (INTEL CORPORATION, 2009). A Figura 5 ilustra o conversor USB-Blaster utilizado no processo de gravação da FPGA via interface JTAG.

Figura 5 – Conversor USB-Blaster.

Fonte: Elaborado pelo Autor, 2025.

2.4.3 Fonte de Alimentação

Para o fornecimento estável de energia, a fonte ATX configura-se como uma solução prática e confiável para alimentar os módulos de relés do sistema. Essa fonte disponibiliza tensões estabilizadas de 12 V, 5 V e 3,3 V, oferecendo corrente adequada para suprir com segurança todos os dispositivos envolvidos no projeto. A escolha pela fonte ATX justifica-se pela sua robustez e capacidade de fornecer energia simultaneamente a múltiplos componentes, eliminando a necessidade de fontes dedicadas para cada módulo.

Além da escolha da fonte, aspectos relacionados à distribuição e proteção da alimentação elétrica também foram considerados de forma criteriosa. Em projetos que envolvem módulos de relés, microcontroladores ou FPGAs, a utilização de uma referência de terra comum (GND) entre os diferentes subsistemas é recomendada para garantir a equipotencialização, minimizar o risco de ruídos e preservar a integridade dos sinais (SEDRA; SMITH, 2010). Adicionalmente, a inclusão de fusíveis de proteção na linha de alimentação constitui uma medida de segurança fundamental, conforme orientações da norma IEC 60664, protegendo o sistema contra sobrecorrentes e curtos-circuitos acidentais (IEC, 2007).

2.4.4 Módulo de Relés com Optoacopladores

Para o acionamento seguro de cargas externas no projeto, optou-se pela utilização do Módulo Relé 3,3V 10A com 8 canais e borne KRE, compatível com a plataforma da FPGA EP4CE6F17I7. Este módulo integra, em cada canal, um optoacoplador de isolamento entre o controle lógico e a carga de potência, permitindo a comutação de dispositivos com correntes de até 10 A sob tensões de até 250 VAC ou 30 VDC. A Figura 06 apresenta o módulo de relés utilizado no projeto.

Figura 6 – Módulo de relés com isolamento de optoacopladores.

Fonte: Elaborado pelo Autor, 2025.

O isolamento entre os circuitos lógicos, operando a um nível de 3,3 VDC, e os circuitos de potência assegura a integridade funcional do sistema, prevenindo interferências e possíveis danos decorrentes de ruídos ou surtos elétricos. Conforme enfatizado por Horowitz e Hill (2016), o isolamento entre os diferentes estágios de um circuito é essencial para evitar a propagação de ruídos e potenciais indesejados que possam comprometer o funcionamento de componentes sensíveis.

Adicionalmente, o módulo dispõe de contatos normalmente abertos (NA), normalmente fechados (NF) e comuns (C), conferindo maior flexibilidade para o acionamento de cargas. O dispositivo também inclui LEDs indicadores de status, jumper para seleção do nível de ativação (*HIGH* ou *LOW*) e diodos de *flyback* integrados em cada canal, os quais protegem contra sobretensões reversas geradas por cargas indutivas, contribuindo para a segurança e confiabilidade do sistema.

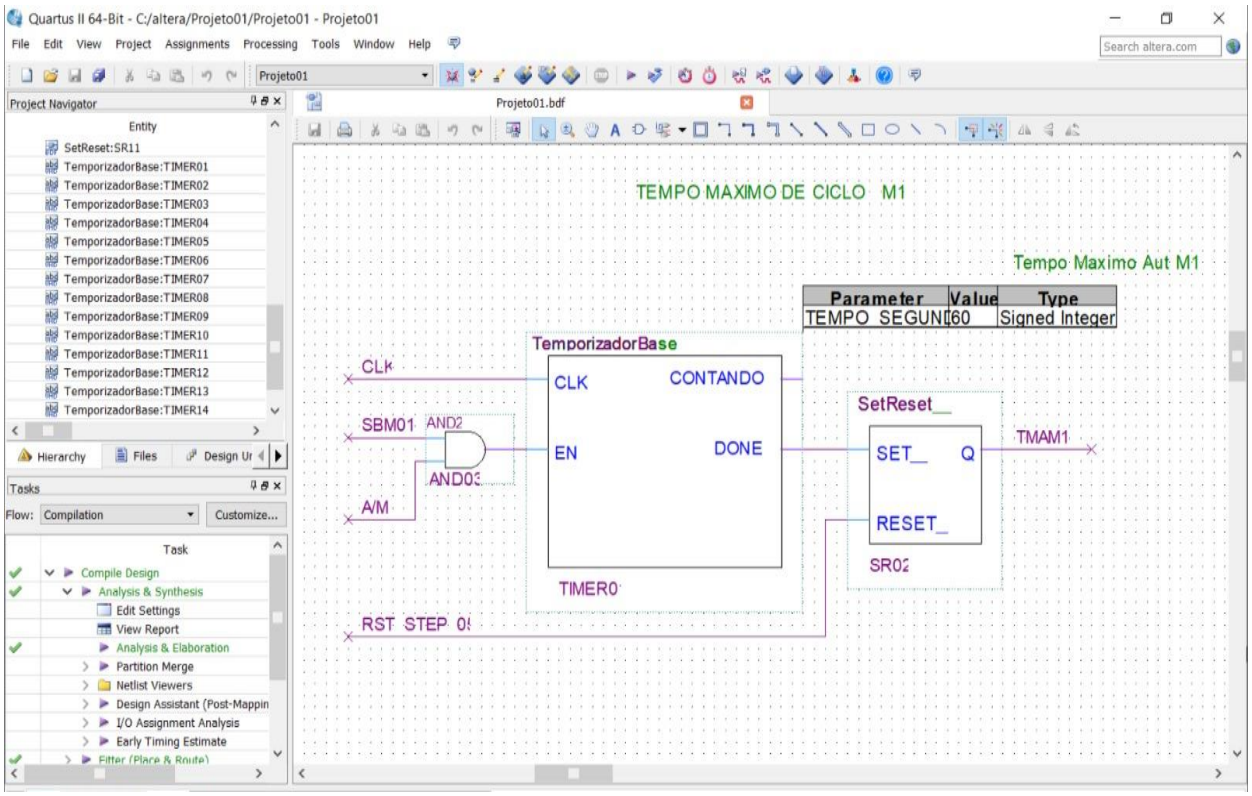
2.5 Ferramentas de Desenvolvimento

O desenvolvimento de sistemas digitais em FPGA requer um ambiente dedicado para projeto, simulação e síntese lógica. Uma das ferramentas mais usadas pela academia e pela indústria é o Quartus II(Intel/Altera), adotado neste trabalho na versão 13.1 para modelar, construir e testar os circuitos implementados.

O Quartus II é uma IDE (*Integrated Development Environment*), disponibilizada pela Altera (atualmente Intel), compatível com o hardware utilizado neste projeto (EP4CE6F17I7, família Cyclone IV). Ele fornece o suporte tanto para linguagens HDLs

quanto ao desenvolvimento gráfico e esquemático por meio de diagramas de blocos (INTEL, 2013), como exemplificado na Figura 7.

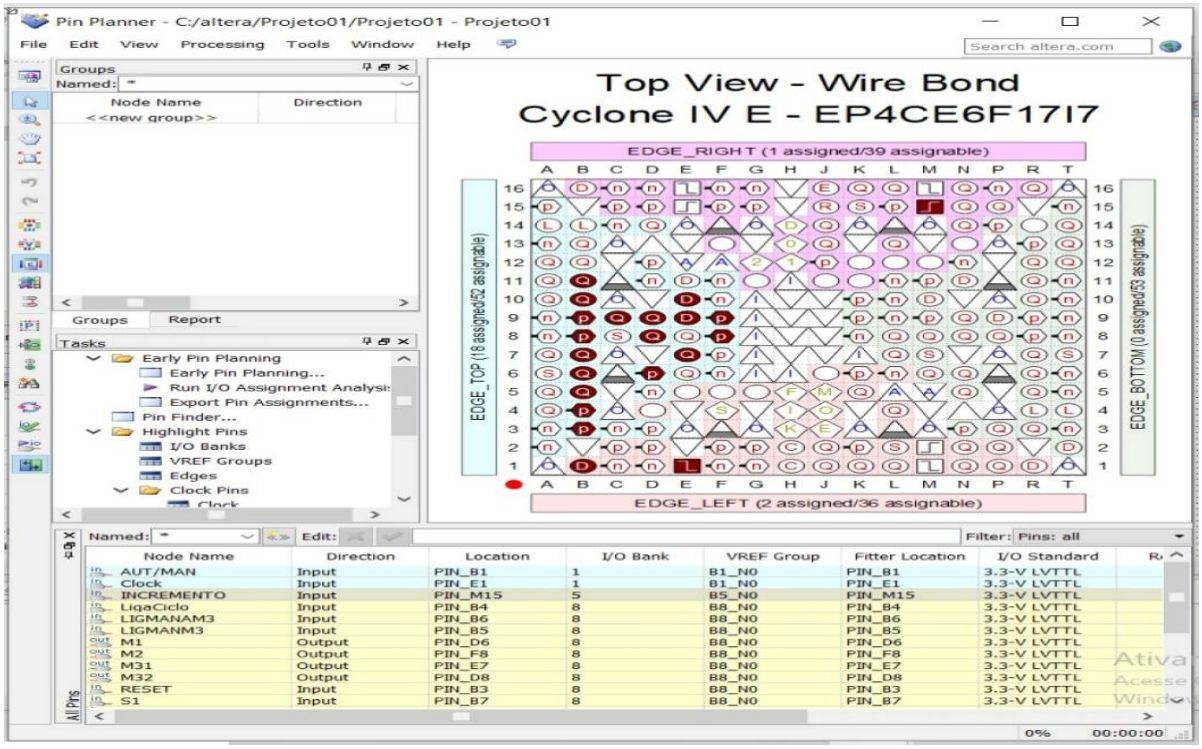
Figura 7 – Interface do editor Quartus II com o diagrama de blocos (.bdf).



Fonte: Elaborado pelo Autor, 2025.

Além do desenvolvimento do código do projeto, o Quartus II incorpora outras ferramentas, como simulação funcional, atribuição de pinos (*Pin Planner*), análise temporal (*TimeQuest*) e geração de arquivos de software (*bitstream*) personalizados para o FPGA, conforme as características descritas no manual da família Cyclone IV (INTEL, 2015). Dentre essas ferramentas, destaca-se o *Pin Planner*, que foi utilizado para realizar a associação entre os sinais lógicos desenvolvidos no projeto e os pinos físicos da FPGA. A Figura 8 ilustra a atribuição física dos sinais.

Figura 8 – Interface Pin Planner no Quartus II, com a atribuição dos pinos físicos da FPGA.



Fonte: Elaborado pelo Autor, 2025.

No contexto educacional e prototípico deste trabalho, a interface gráfica do ambiente Quartus II possibilitou a criação e modificação ágil das estruturas lógicas, além de facilitar o mapeamento das entradas e saídas físicas aos componentes da planta didática. Essa integração entre o software de desenvolvimento e o hardware real revelou-se essencial para assegurar a coerência entre a lógica projetada e sua aplicação prática, permitindo a realização de testes controlados e ajustes rápidos durante o processo de desenvolvimento.

2.6 Descrição da Planta Didática

A planta didática proposta neste projeto tem como finalidade simular, de forma simplificada, um processo industrial automatizado de mistura e transferência de materiais. Controlada integralmente por uma lógica digital implementada em FPGA, a planta permite explorar conceitos fundamentais da automação industrial, como leitura de sensores, controle sequencial, temporização, intertravamentos e acionamento de atuadores.

O sistema é composto por dois módulos de alimentação, representados pelas esteiras M1 e M2, responsáveis pelo transporte de materiais até o tanque de mistura. O volume inserido no tanque é monitorado por sensores de nível S1 (nível máximo) e S2 (nível médio), enquanto o motor M3 promove a homogeneização do conteúdo durante um período controlado. Após a mistura, o material é direcionado ao reservatório de destino por meio da válvula de descarga V1, cujo estado é monitorado pelos sensores de posição S3 (válvula aberta) e S4 (válvula fechada). O sensor S5, posicionado no reservatório final, detecta se há material presente, indicando o fim do processo.

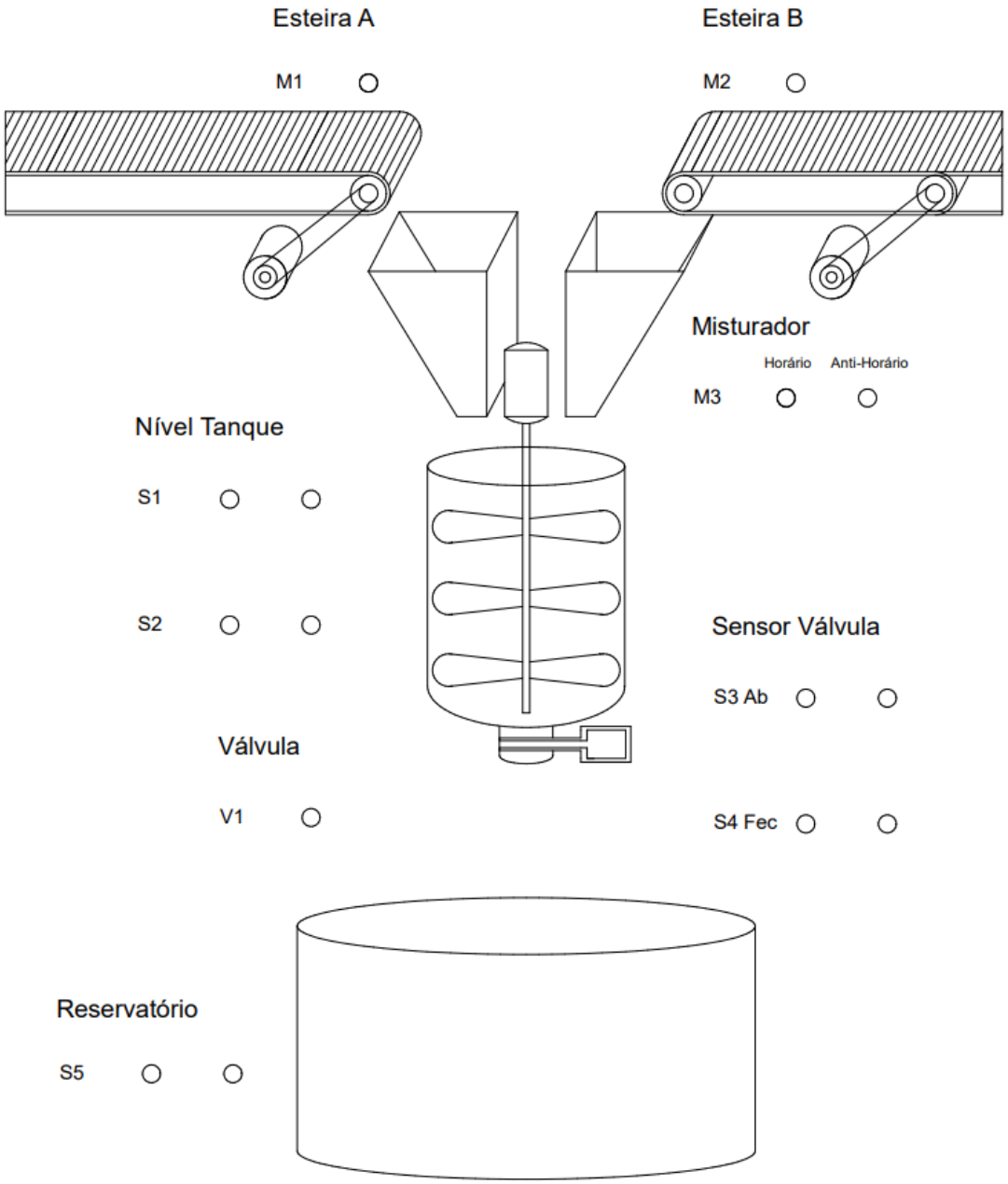
Além dos dispositivos físicos principais, a planta conta com entradas digitais para controle operacional, como a seleção entre modo automático e manual, o acionamento do ciclo automático e a reinicialização do sistema por meio de um botão RESET.

O processo automatizado segue uma lógica sequencial estruturada. Inicialmente, o sistema analisa as condições necessárias para o início do ciclo, verificando se o reservatório de destino está vazio (S5 = 0), se a válvula encontra-se totalmente fechada (S4 = 1) e se o tanque está desabastecido (S1 e S2 = 0). Uma vez atendidas essas condições e estando o sistema em modo automático com o ciclo ativado, a esteira M1 é acionada para alimentar o tanque com o material A até que o sensor de nível médio (S2) indique que o nível 1 foi atingido. Neste ponto, M1 é desligada e M2 é ativada, iniciando a alimentação do material B até que o sensor S1 (nível máximo) detecte que o tanque foi completamente preenchido. A esteira M2 é então desligada e o sistema aguarda um tempo de estabilização.

Em seguida, o motor M3 é ativado, realizando a mistura do conteúdo do tanque, primeiro no sentido horário e, após um intervalo, no sentido anti-horário. Após a mistura, o sistema verifica novamente se o reservatório de destino está liberado (S5 = 0). Estando a condição atendida, a válvula V1 é aberta e o sistema monitora o escoamento do conteúdo do tanque, observando os sensores S1 e S2 até que ambos retornem ao estado lógico 0, indicando que o tanque está vazio. Um tempo de segurança é então contado antes do fechamento da válvula. O sistema finaliza o ciclo e retorna à verificação das condições iniciais, reiniciando automaticamente o processo. O detalhamento dos passos do ciclo de automação e sua implementação lógica será abordado no Capítulo 4.1.1, que descreve a estrutura lógica do sistema em etapas sequenciais.

A Figura 9 apresenta a representação esquemática da planta didática, ilustrando a disposição funcional dos elementos físicos e suas respectivas conexões com o sistema de controle. O diagrama tem como objetivo facilitar a compreensão da estrutura geral e das interações lógicas que ocorrem ao longo do ciclo automatizado.

Figura 9 – Planta Misturador.



Fonte: Elaborado pelo Autor, 2025.

3 METODOLOGIA

O presente trabalho caracteriza-se, quanto à sua natureza, como aplicada, pois objetiva empregar conhecimentos teóricos e técnicos na construção de uma solução prática para o controle de processos industriais. Sua finalidade está centrada no desenvolvimento e validação de uma lógica de controle digital, utilizando dispositivos FPGA em uma planta didática simulada.

Em relação à forma de abordagem, trata-se de uma pesquisa quantitativa, visto que se fundamenta na análise de variáveis digitais discretas e na verificação objetiva de funcionamento por meio de testes físicos e mensuráveis.

Quanto aos objetivos, adota-se uma abordagem descritiva e experimental. O caráter descritivo está presente na documentação detalhada das etapas de desenvolvimento, da estrutura da planta e da lógica implementada. Já o caráter experimental manifesta-se na construção de protótipos, na aplicação de testes em bancada e na avaliação do comportamento real da lógica programada em um sistema físico, permitindo a observação direta dos resultados.

Do ponto de vista dos procedimentos técnicos, a pesquisa configura-se como um estudo de caso, pois concentra-se na análise e resolução de um problema específico em um contexto delimitado, com variáveis operacionais previamente definidas (LAKATOS; MARCONI, 2017).

Para viabilizar a aplicação prática da abordagem metodológica adotada, as etapas metodológicas foram estruturadas de forma sequencial, contemplando desde a definição da planta didática até os procedimentos de validação funcional do sistema. Essa organização visa garantir o encadeamento lógico das fases do projeto e permitir a replicabilidade dos procedimentos empregados. A seguir, são descritas as etapas que compõem o desenvolvimento do trabalho, evidenciando os critérios técnicos e operacionais adotados ao longo do processo de implementação do controle digital baseado em FPGA.

3.1 Definição da Planta Didática

Nesta etapa, pretende-se definir a planta didática a ser simulada, realizando o levantamento dos requisitos operacionais do sistema e a identificação das variáveis de entrada e saída. Será conduzida uma análise funcional do processo automatizado,

contemplando as etapas de alimentação, mistura e escoamento de materiais, com a especificação dos critérios de transição entre os estados operacionais.

3.2 Elaboração do Desenho Elétrico e Montagem da Estrutura Física

Esta etapa contemplará a elaboração do diagrama elétrico do sistema para documentar as interligações entre os componentes e orientar a montagem física da estrutura sobre uma bancada. Serão incluídas as conexões entre os dispositivos de entrada e saída, resistores de proteção, relés optoacoplados e a fonte de alimentação, buscando otimizar a visualização do funcionamento da lógica e facilitar a identificação de eventuais falhas.

3.3 Desenvolvimento das Macrocélulas Lógicas

Com o intuito de atender às exigências do processo automatizado, serão desenvolvidas macrocélulas lógicas reutilizáveis no ambiente Quartus II, utilizando arquivos BDF e entidades VHDL. Esses blocos modulares terão entradas e saídas padronizadas para facilitar a integração e o reuso em diferentes segmentos da lógica de controle. A validação individual de cada macrocélula será realizada por meio de testes físicos na planta didática, verificando suas respostas de entrada e saída para assegurar o funcionamento isolado correto antes da integração.

3.4 Implementação da Lógica de Controle

A lógica de controle do processo será implementada por meio da interligação de portas lógicas combinacionais e das macrocélulas desenvolvidas, organizadas em blocos funcionais interdependentes. O objetivo é representar o avanço entre as etapas (*steps*) do ciclo automático, controlando a transição de estados com base em sinais de sensores e temporizadores, inspirando-se na linguagem Ladder para transposição à lógica digital na FPGA.

3.5 Testes e Validação Funcional

Após a implementação da lógica de controle, serão realizados testes funcionais para verificar o desempenho do sistema sob diferentes cenários operacionais. Cada etapa do ciclo automatizado será validada individualmente, monitorando tempos de resposta, atuação dos dispositivos e intertravamentos. Serão efetuados ajustes finos nos blocos lógicos e parâmetros (temporização, reinicialização) para garantir a estabilidade e confiabilidade do funcionamento global da planta. A simulação de falhas em sinais de entrada também será contemplada para avaliar a robustez da lógica.

4 DESENVOLVIMENTO

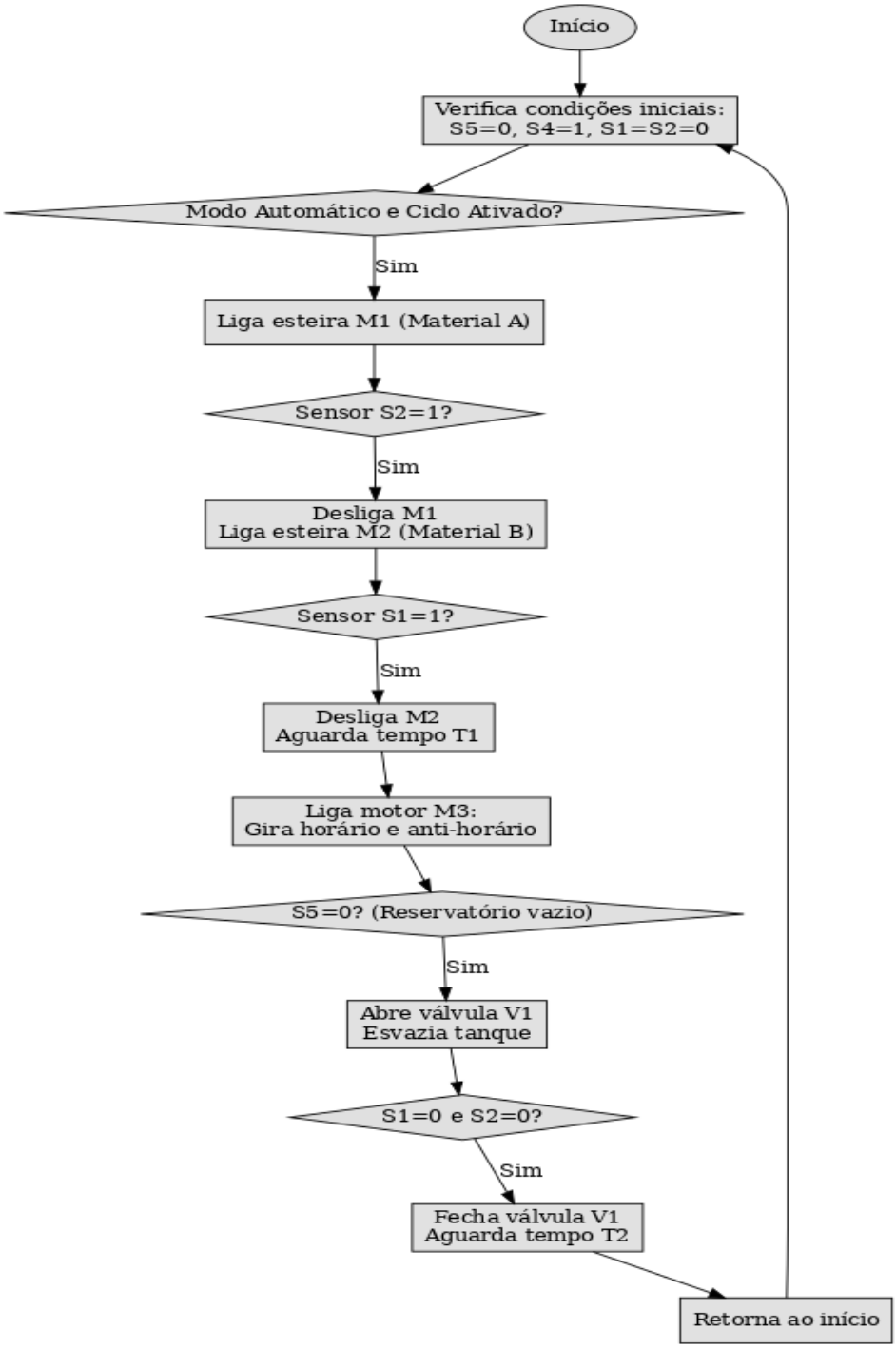
Esta seção descreve o desenvolvimento prático do projeto de automação baseado em FPGA, com ênfase na estrutura física, lógica e funcional da solução proposta. Foram implementadas estratégias de controle digital por meio de lógica combinacional, organizadas em macrocélulas reutilizáveis. Destaca-se, nesse processo, a atenção dedicada à organização dos sinais, à proteção elétrica dos circuitos e à padronização dos blocos lógicos. A seguir, são apresentadas as etapas de construção e validação do sistema automatizado.

4.1 Definição da Planta Didática

4.1.1 Estrutura Lógica do Ciclo de Automação

Este item descreve a sequência operacional da planta didática, conforme implementada na lógica digital programada na FPGA. As etapas foram definidas com base no comportamento esperado do processo de mistura e transferência de materiais, utilizando entradas digitais provenientes de sensores e botões, e acionando as saídas de acordo com condições logicamente estabelecidas. O controle do processo segue uma lógica estruturada, composta por dez etapas principais, organizadas de forma sequencial e funcionalmente coerente com a dinâmica da aplicação, conforme detalhado no fluxograma apresentado na Figura 10.

Figura 10 - Fluxograma do Processo Automatizado.



Fonte: Elaborado pelo Autor, 2025.

A seguir, é apresentada a descrição detalhada de cada etapa do processo, acompanhada de seus respectivos acionamentos e condições de transição.

Condições Iniciais. O ciclo automático somente pode ser iniciado se o sistema estiver em modo automático (ALT_MANUAL = 1), com o botão de ciclo ativado (LIGA_CICLO = 1) e o sensor S5 indicando que o reservatório de destino está vazio (S5 = 0). Além disso, a válvula deve estar completamente fechada (S4 = 1) e o motor de mistura desligado.

Abastecimento com Material A (M1). A esteira M1 é acionada, iniciando o abastecimento do tanque com o material A. Esse processo continua até que o sensor de nível médio (S2) seja ativado. Esta etapa é controlada pela Macrocélula PartidaDiretaV1 associada à esteira M1, que aciona a esteira até a ativação do sensor S2, cujo sinal é filtrado pela Macrocélula Debounce15ms.

Pausa Temporizada. Ao detectar S2 = 1, a esteira M1 é desligada e inicia-se um temporizador. Esse intervalo simula o tempo necessário para estabilização do material no tanque. A temporização desta pausa é gerenciada por uma instância da Macrocélula TemporizadorBase, que sinaliza o fim do intervalo.

Abastecimento com Material B (M2). Encerrado o tempo de espera, a esteira M2 é acionada, promovendo o carregamento do material B no tanque até que o sensor de nível máximo (S1) seja ativado. A Macrocélula PartidaDiretaV1 referente à esteira M2 é ativada, e o processo continua enquanto o sensor S1 não detectar o nível máximo, sendo seu sinal também estabilizado pela Macrocélula Debounce15ms.

Pausa Temporizada. A esteira M2 é desligada e um novo temporizador é iniciado para garantir o repouso do conteúdo no tanque antes da agitação. Uma segunda instância da Macrocélula TemporizadorBase é utilizada para controlar o tempo desta pausa.

Mistura dos Materiais (M3). Ao término da pausa, o motor de agitação (M3) é acionado no sentido horário (M3.1) e depois no sentido anti-horário (M3.2), realizando a mistura dos materiais por um período pré-definido, determinado por temporizador. A Macrocélula PartidaReversão é responsável pelo acionamento sequencial dos sentidos horário e anti-horário do motor M3, com a duração de cada sentido controlada por instâncias da Macrocélula TemporizadorBase.

Verificação do Reservatório de Destino. Ao fim da mistura, verifica-se novamente se o reservatório de destino permanece vazio (S5 = 0), condição necessária para autorizar a transferência do produto. O sinal do sensor S5 é lido, e sua estabilidade é assegurada pela Macrocélula Debounce15ms antes de validar a condição.

Abertura da Válvula (V1). Com a condição satisfeita, a válvula de descarga V1 é acionada. A lógica monitora os sensores S3 (válvula totalmente aberta) e S4 (válvula totalmente fechada) para garantir o estado correto da válvula durante a operação. Esta operação é controlada pela Macrocélula ControleValvula, que gerencia o acionamento de V1 e monitora os sinais de posição S3 e S4, ambos filtrados pela Macrocélula Debounce15ms.

Transferência Temporizada. Após confirmação de abertura total da válvula (S3 = 1), inicia-se a transferência do material por um tempo determinado. Esse tempo simula a vazão do material até o completo esvaziamento do tanque. Uma instância da Macrocélula TemporizadorBase é configurada para controlar o tempo desta transferência, garantindo a vazão adequada do material.

Fechamento da Válvula e Reinício. Encerrado o tempo de transferência, a válvula é desligada e aguarda-se a confirmação de fechamento total (S4 = 1). Com isso, o sistema retorna à etapa Condições Iniciais, reiniciando o ciclo de operação. A Macrocélula ControleValvula é comandada para fechar V1, aguardando a validação do sensor S4 (filtrado por Debounce15ms). O retorno à etapa Condições Iniciais é orquestrado pela lógica top-level, que gerencia os estados dos SetReset dos passos.

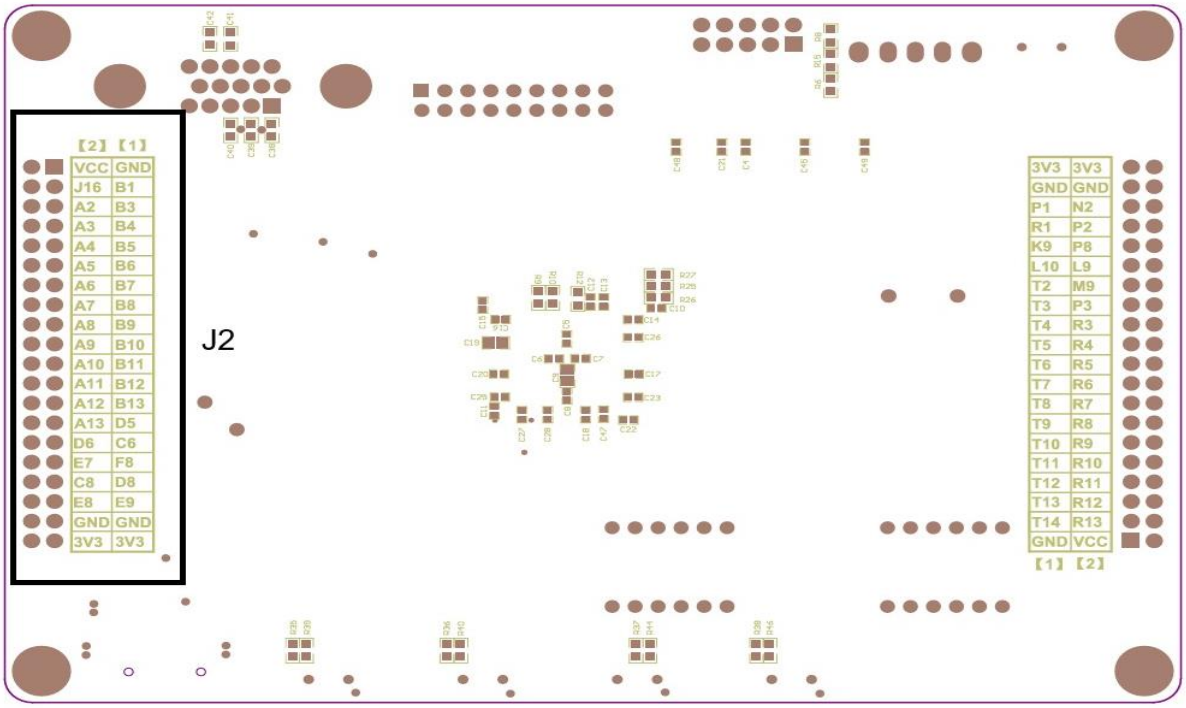
4.1.2 Mapeamento dos Sinais

O sistema é constituído por dois módulos de alimentação (esteiras M1 e M2), um tanque de mistura, um motor de agitação com reversão de sentido (M3), uma válvula de descarga (V1), sensores de nível no tanque (S1 e S2), sensores de posição da válvula (S3 e S4) e um sensor no reservatório de destino (S5). O controle dessas unidades é realizado por uma FPGA da família Cyclone IV, programada para

interpretar os sinais provenientes das entradas e acionar as saídas conforme a lógica sequencial previamente definida.

A definição da pinagem física foi realizada com base na análise do diagrama elétrico da placa fornecido pelo fabricante, apresentado no Anexo A, o qual detalha a disposição dos pinos de entrada e saída no conector J2 (IDC40). A Figura 11 ilustra a topologia dos terminais disponíveis na placa. A partir dessa disposição, foram selecionados os pinos mais adequados para as funções de entrada e saída, considerando critérios como acessibilidade física e facilidade de agrupamento lógico dos sinais.

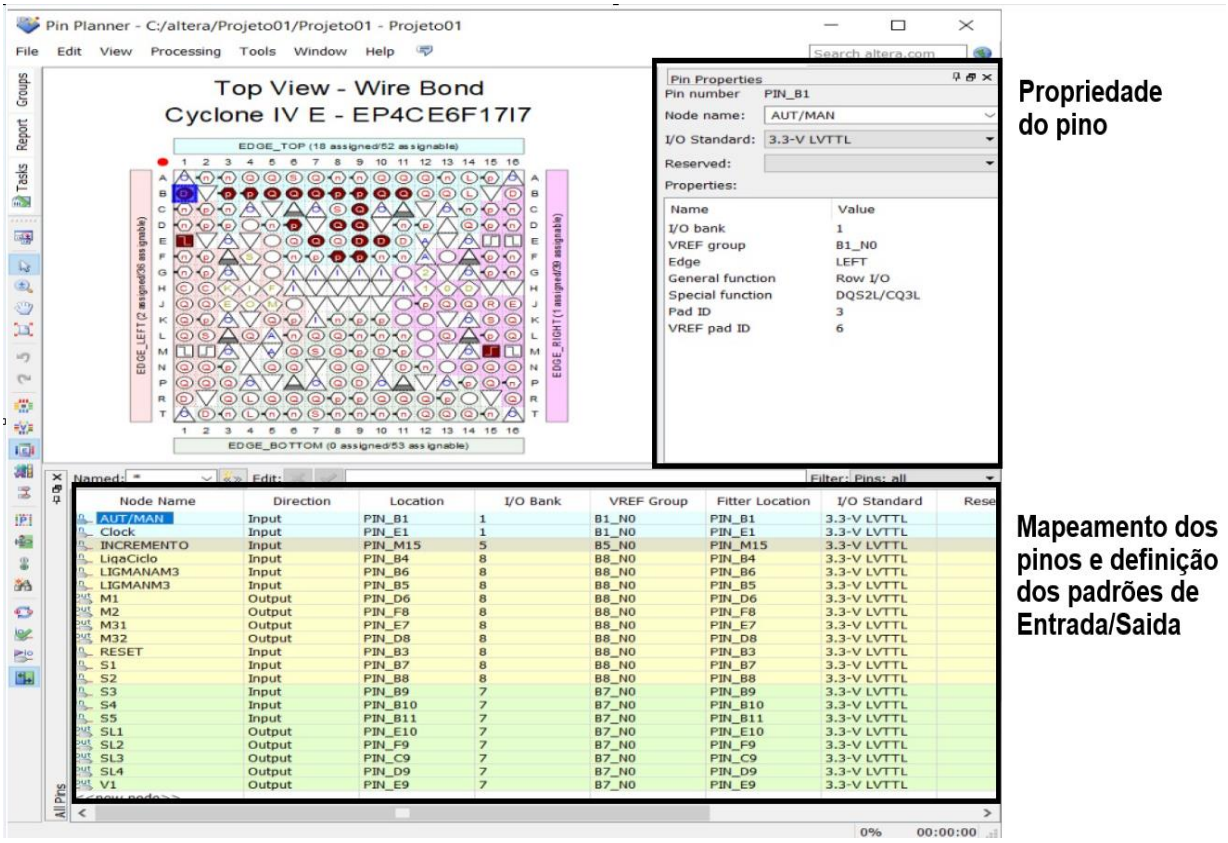
Figura 11 – Representação do conector J2 da placa FPGA EP4CE6F17I7.



Fonte: Saylinx, 2017.

Para realizar a atribuição lógica dos sinais aos pinos físicos da FPGA, foi utilizada a ferramenta Pin Planner, integrante do ambiente Quartus II. Essa ferramenta permite a visualização tanto do mapa físico do encapsulamento quanto das propriedades elétricas de cada pino, como o banco de I/O, grupo VREF, posição na matriz e funções especiais. A Figura 12 apresenta o mapeamento visual dos pinos, e aba de propriedades técnicas de um pino específico (PIN_B1), contendo as informações que orientaram a escolha do terminal.

Figura 12 – Mapeamento de pinos no Pin Planner Quartus II 13.4.



Fonte: Elaborado pelo Autor, 2025.

Durante esse processo, todos os sinais foram configurados conforme o padrão de operação 3,3 V LVTTL (*Low Voltage Transistor-Transistor Logic*), adequado para circuitos de lógica digital discreta. A escolha desse padrão técnico foi fundamentada na tensão de operação dos módulos periféricos, na compatibilidade dos sinais com os componentes de entrada e saída, bem como na estabilidade elétrica exigida pelo sistema. O padrão LVTTL oferece comutação rápida, baixo consumo de energia e boa imunidade a ruídos elétricos, características compatíveis com os requisitos operacionais da planta simulada neste trabalho.

Com a finalização do planejamento da pinagem e a definição das características elétricas, foi possível elaborar o mapeamento completo dos sinais digitais utilizados no sistema. Esse mapeamento está apresentado na Tabela 2, que lista todas as entradas e saídas, suas respectivas funções e os pinos físicos aos quais foram atribuídas na FPGA.

Tabela 2 – Mapeamento dos sinais digitais (IO) da FPGA.

Tipo	Sinal	Descrição	Pino FPGA
Entrada	AUT/MAN	Seleção de modo de operação: 1=Automático, 0 = Manual	PIN_B1
Entrada	RESET	Reinicializa o sistema e intertravamentos	PIN_B7
Entrada	LIGA_CICLO	Habilita a execução do ciclo automático	PIN_M15
Entrada	LIGA_MANUAL_M3_H	Liga motor M3 no sentido horário (modo manual)	PIN_B6
Entrada	LIGA_MANUAL_M3_AH	Liga motor M3 no sentido anti-horário (modo manual)	PIN_B5
Entrada	S1	Sensor de nível máximo no tanque	PIN_B4
Entrada	S2	Sensor de nível médio no tanque	PIN_B3
Entrada	S3	Sensor de válvula totalmente aberta	PIN_E8
Entrada	S4	Sensor de válvula totalmente fechada	PIN_E9
Entrada	S5	Sensor do reservatório de destino (vazio = 0)	PIN_E10
Saída	M1	Aciona esteira A (material A)	PIN_D6
Saída	M2	Aciona esteira B (material B)	PIN_D5
Saída	M3.1	Motor M3 no sentido horário	PIN_E7
Saída	M3.2	Motor M3 no sentido anti-horário	PIN_E6
Saída	V1	Acionamento da válvula de descarga	PIN_D4

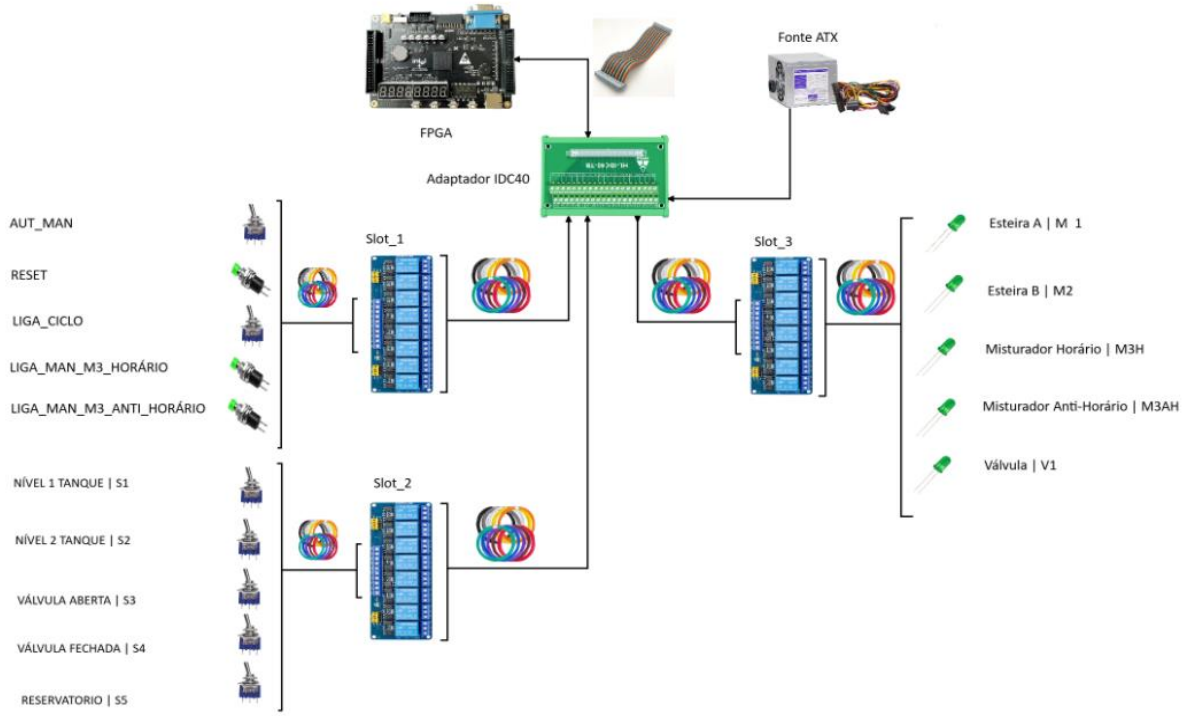
Fonte: Elaborado pelo Autor, 2025.

4.2 Elaboração do Projeto Elétrico e Justificativas Técnicas

A elaboração do projeto elétrico foi essencial para assegurar a organização, funcionalidade e segurança durante a montagem da planta didática. Para esse fim, foi desenvolvido um esboço preliminar (Figura 13) com o propósito de orientar a disposição dos componentes e mapear as interligações entre a FPGA, os dispositivos de entrada e saída, os módulos de relé e a fonte de alimentação. Embora não

represente o esquema definitivo, esse diagrama funcionou como uma base conceitual para a montagem prática, garantindo a coerência entre a lógica implementada e sua infraestrutura física.

Figura 13 – Esboço da Arquitetura Elétrica da Planta Didática.



Fonte: Elaborado pelo Autor, 2025.

A estruturação do projeto seguiu boas práticas de engenharia, orientada por critérios de confiabilidade, proteção, isolamento e padronização. Diversas decisões tomadas foram fundamentadas nos esquemas de referência fornecidos pelo fabricante da placa FPGA (conforme Anexo A), especialmente no que diz respeito ao tratamento dos sinais de entrada e saída, visando assegurar a compatibilidade e estabilidade dos circuitos. Os tópicos subsequentes detalham essas escolhas, abordando aspectos relativos à interface com os módulos de relé, alimentação elétrica, conectividade e à disposição dos componentes físicos da planta didática.

4.2.1 Isolamento dos Sinais com Módulos de Relé

A utilização de módulos de relé com optoacopladores configurou-se como uma decisão estratégica, fundamentada na necessidade de isolamento elétrico tanto dos

sinais de saída quanto dos sinais de entrada da FPGA. Considerando que os pinos de entrada e saída operam com baixos níveis de corrente e tensão, o acionamento direto não seria viável, tampouco seria possível garantir a estabilidade das entradas diante de variações externas. Os relés optoacoplados funcionam como uma interface segura, proporcionando isolamento galvânico entre os circuitos de controle e os elementos externos do sistema. Essa separação previne que ruídos, picos de tensão ou transientes provenientes das cargas e dispositivos de entrada comprometam os componentes sensíveis da FPGA, elevando, assim, a confiabilidade e a durabilidade do sistema.

Adicionalmente, os módulos de relé incorporam LEDs indicadores por canal, o que possibilita o monitoramento visual do acionamento das saídas, facilitando testes, depuração e a identificação de falhas. Na prática, os módulos foram organizados em três slots funcionais: Slot_1 destinado às entradas manuais, Slot_2 aos sensores de processo (botões) e Slot_3 às saídas dos atuadores (LEDs).

4.2.2 Entradas Digitais

A leitura confiável dos sinais digitais é imprescindível para o correto funcionamento da lógica implementada na FPGA. Neste projeto, adotou-se o padrão de lógica negativa, no qual os sinais são considerados ativos quando apresentam nível baixo (0 V). Essa convenção é amplamente utilizada devido à sua maior imunidade a ruídos transitórios, que normalmente tendem a elevar o potencial das linhas, ao invés de reduzi-lo.

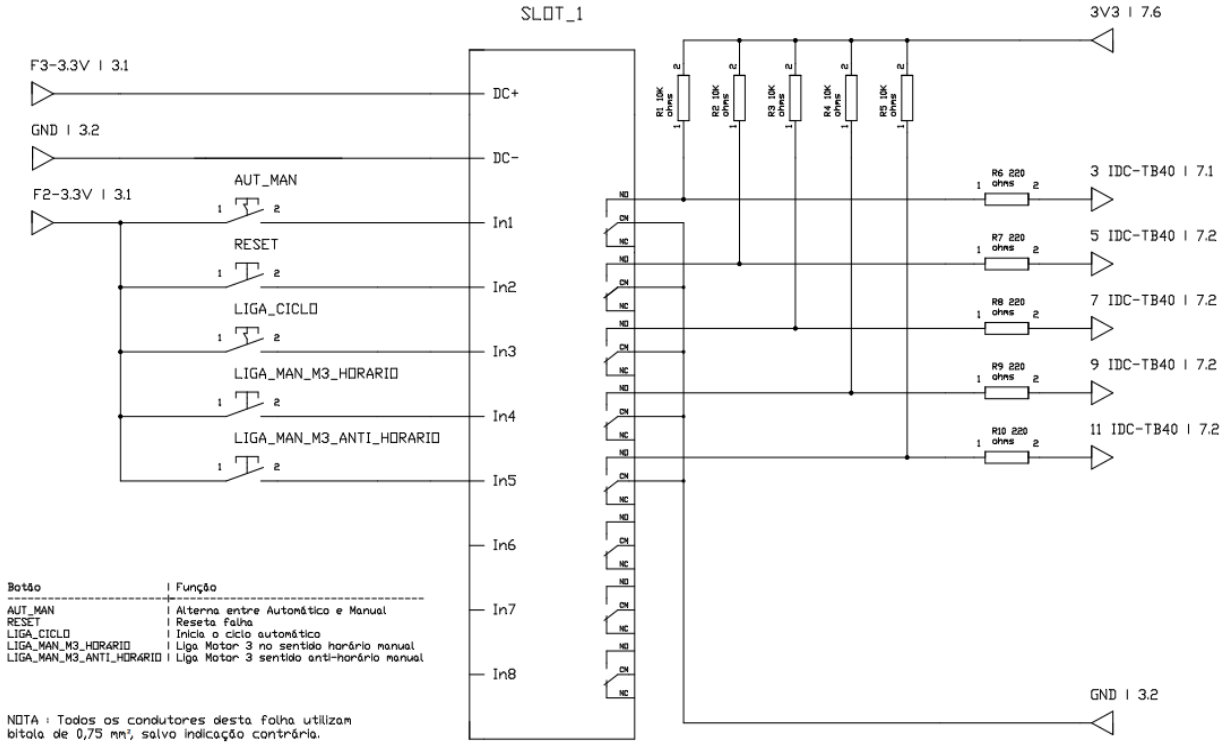
Para assegurar que os sinais mantenham um nível definido na ausência de acionamento externo, foram empregados resistores pull-down de 10 KΩ, conectados a tensão de referência 3V3 da placa. Tal valor foi selecionado por proporcionar um equilíbrio adequado entre impedância e estabilidade, prevenindo flutuações e interferências indesejadas.

Além disso, cada entrada possui um resistor de 220 Ω em série, cuja função é limitar a corrente em casos de curto-circuito e suavizar transições abruptas de tensão. Esta proteção é eficaz para mitigar ruídos gerados por dispositivos mecânicos, como botões e sensores *reed*.

A combinação do resistor pull-down com o resistor em série segue o padrão adotado nos botões onboard da placa FPGA, promovendo compatibilidade elétrica e padronização entre os circuitos internos e externos.

A Figura 14 apresenta o esquema elétrico das entradas do Slot 1, extraído do projeto completo disponível no Apêndice A. Nesse recorte, evidencia-se a aplicação da topologia de proteção com resistores pull-down e resistores em série, adotada para assegurar estabilidade e segurança na leitura dos sinais digitais conectados à FPGA.

Figura 14 – Esquema elétrico das entradas digitais do Slot 1.



Fonte: Elaborado pelo Autor, 2025.

4.2.3 Estratégia de Acionamento das Saídas Digitais

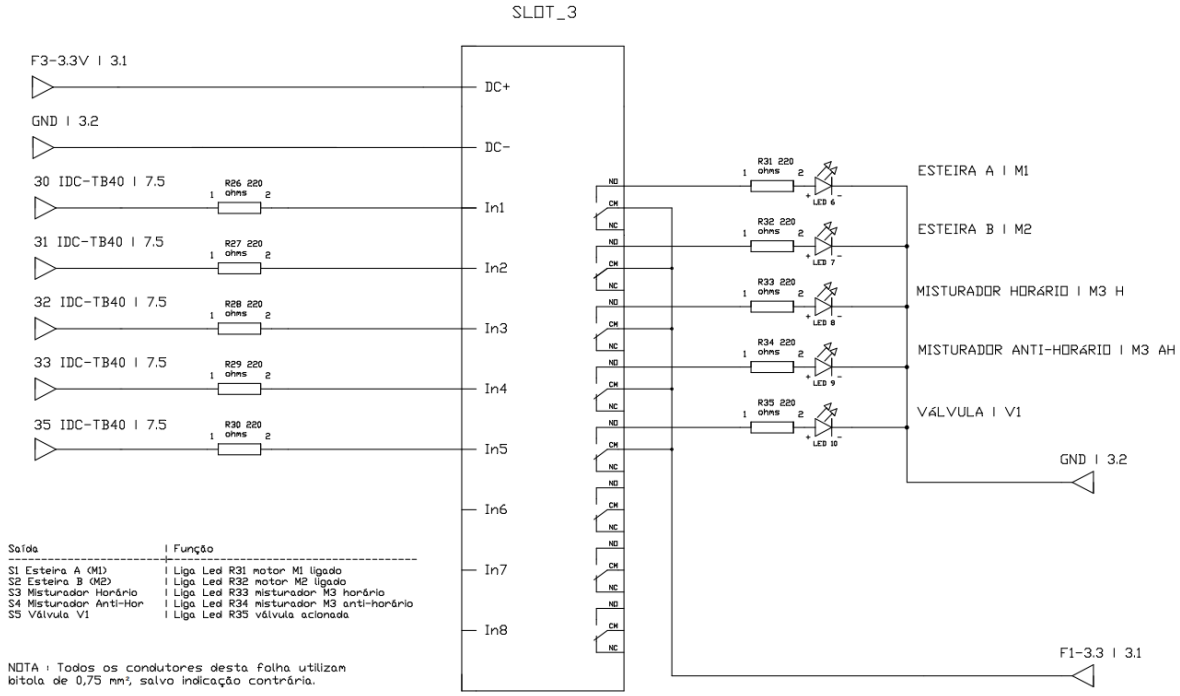
O acionamento das saídas digitais da FPGA foi estabelecido considerando critérios de compatibilidade lógica, isolamento elétrico e facilidade de verificação durante os testes. Neste projeto, as saídas operam no modo *sourcing*, ou seja, são consideradas ativas quando o sinal atinge nível lógico alto (1), conforme as especificações da placa FPGA utilizada.

Cada saída foi conectada a um canal de relé optoacoplado, possibilitando a comutação segura de cargas externas, garantindo o isolamento entre o circuito lógico

e o circuito de potência. O sinal lógico alto ativa o optoacoplador, o qual fecha o contato do relé e energiza o dispositivo correspondente.

A Figura 15 apresenta o esquema elétrico das saídas digitais conectadas ao Slot 3, evidenciando as ligações entre os pinos da FPGA, os resistores em série de 220 Ω e os relés responsáveis pelo acionamento dos motores e da válvula da planta. O diagrama é um recorte do projeto completo disponível no Apêndice A, ilustrando a estratégia de projeto adotada.

Figura 15 – Esquema elétrico das saídas digitais do Slot 3.



Fonte: Elaborado pelo Autor, 2025.

4.2.4 Alimentação e Referência Comum

A alimentação do sistema foi realizada por meio de uma fonte ATX, a qual fornece tensão estabilizada de 3,3V garantindo estabilidade operacional adequada. A escolha dessa fonte fundamentou-se em sua robustez e disponibilidade. Todas as referências de sinal e alimentação compartilham um ponto comum de aterramento (GND), aspecto essencial para evitar diferenças de potencial entre os módulos e assegurar o funcionamento correto da lógica digital implementada.

4.2.5 Distribuição dos Sinais e Conectividade

A comunicação entre a FPGA e os módulos externos foi estabelecida por meio de um adaptador IDC40, que organiza os sinais em um conector padrão de 40 vias, facilitando a distribuição ordenada e a identificação dos cabos utilizados no sistema.

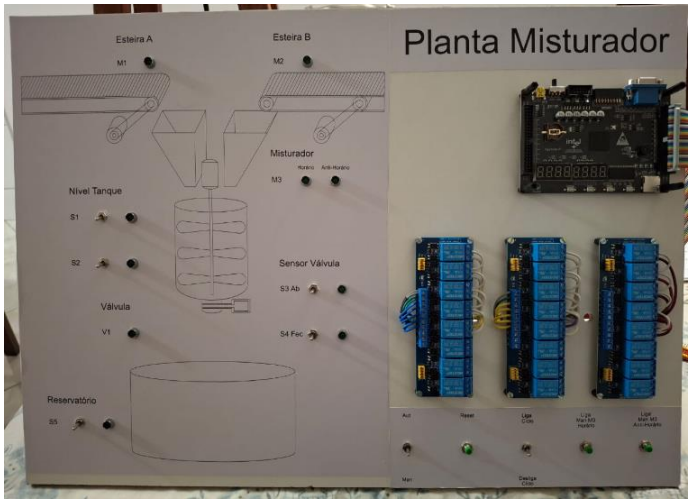
4.2.6 Considerações sobre a Bancada e Organização Física

A planta foi montada sobre uma bancada inclinada de madeira, projetada para acomodar os componentes de forma segura e organizada. Um layout esquemático foi afixado na superfície da bancada, servindo como referência para a fixação de botões, LEDs, bornes e demais elementos, assegurando o alinhamento com o projeto elétrico.

A disposição dos componentes foi planejada para simular visualmente um processo industrial, onde os botões atuam como sensores de entrada e os LEDs representam atuadores, proporcionando uma compreensão intuitiva do funcionamento da lógica implementada na FPGA.

A Figura 16 ilustra a bancada finalizada, evidenciando a fixação dos elementos e a organização dos cabos conforme o layout estabelecido. Tal montagem constituiu uma interface prática para a validação da lógica implementada, além de facilitar os testes durante o desenvolvimento do projeto.

Figura 16 – Bancada Misturador.



Fonte: Elaborado pelo Autor, 2025.

4.3 Elaboração das Macrocélulas na FPGA

Durante o desenvolvimento da lógica de controle da planta didática, adotou-se a estratégia de encapsular blocos funcionais em macrocélulas reutilizáveis, visando modularizar o projeto, aprimorar a organização estrutural e facilitar futuras manutenções.

Cada macrocélula, nesse contexto, representa um circuito de hardware dedicado, compilado e gravado na FPGA, que realiza uma função lógica específica, podendo ser instanciado múltiplas vezes de forma independente. Cada macrocélula foi elaborada a partir de arquivos do tipo BDF (*Block Diagram File*) ou de entidades em VHDL, as quais foram posteriormente encapsuladas utilizando o recurso "Create Symbol Files for Current File" disponível no ambiente Quartus II.

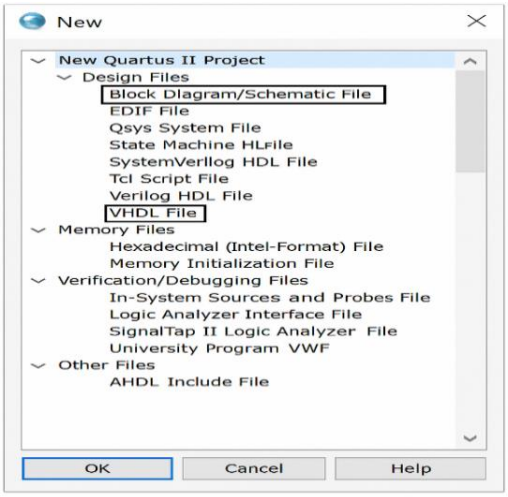
4.3.1 Criação de Macrocélulas no Quartus II 13.1

A criação de macrocélulas no ambiente Quartus II constitui uma estratégia eficiente para encapsular blocos lógicos reutilizáveis em projetos baseados em FPGA. Neste trabalho, foram empregadas duas das formas de desenvolvimento disponíveis na plataforma: *Block Diagram File* (BDF), que permite a construção da lógica a partir de portas lógicas, e VHDL, por meio de descrição textual do comportamento do hardware. A escolha entre essas abordagens foi pautada na complexidade e na finalidade específica de cada bloco funcional.

Após a elaboração da lógica, utilizou-se o recurso “*Create Symbol Files for Current File*”, que gera um símbolo representativo do bloco, convertendo-o em uma macrocélula reutilizável. Tal bloco encapsulado pode ser instanciado múltiplas vezes no projeto principal, desde que cada instância receba um nome único, prevenindo conflitos de sobreposição ou erros lógicos.

A Figura 17 apresenta a interface de criação de novos arquivos no ambiente Quartus II, onde é possível selecionar a linguagem ou a forma de desenvolvimento desejada — como BDF ou VHDL — para a construção da lógica que será posteriormente encapsulada em uma macrocélula.

Figura 17 – Opções de criação de lógicas no Quartus II 13.1.

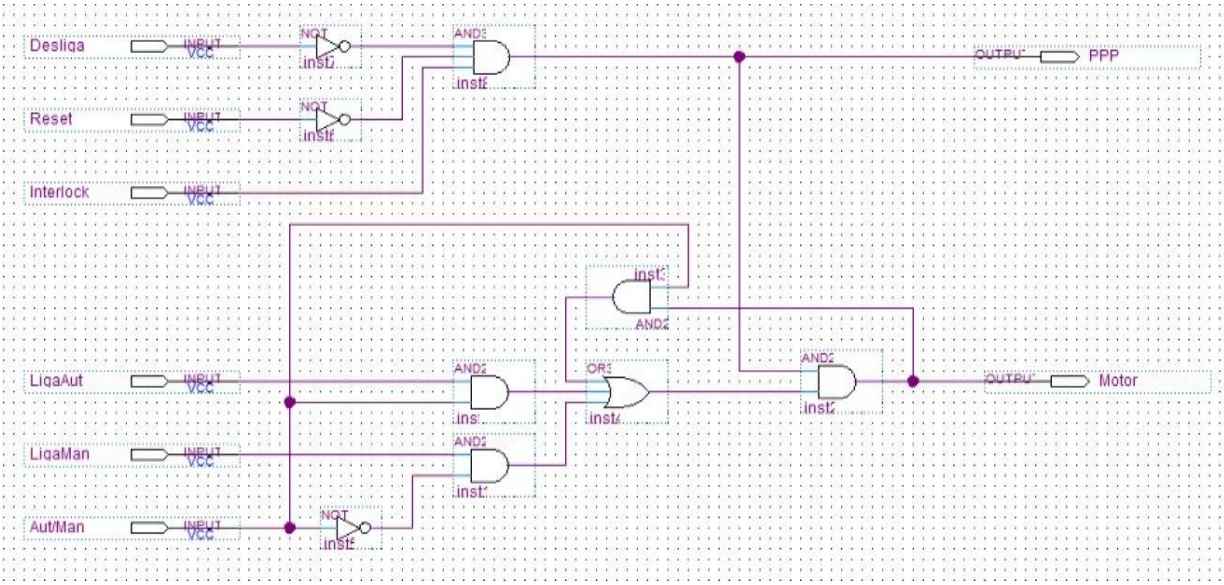


Fonte: Elaborado pelo Autor, 2025.

A Figura 18 apresenta a lógica interna que deu origem ao bloco PartidaDiretaV1, desenvolvida por meio de um *Block Diagram File* (BDF). Nessa visualização, observa-se a estrutura combinacional formada por portas lógicas básicas — NOT, AND e OR — responsável por interpretar as condições de entrada e gerar as saídas correspondentes.

Essa representação permite compreender como cada entrada do bloco — Desliga, Reset, Interlock, Aut/Man, LigaMan e LigaAut — atua no controle da saída Motor e no sinal de Pronto Para Partir (PPP), validando o comportamento lógico encapsulado na macrocélula.

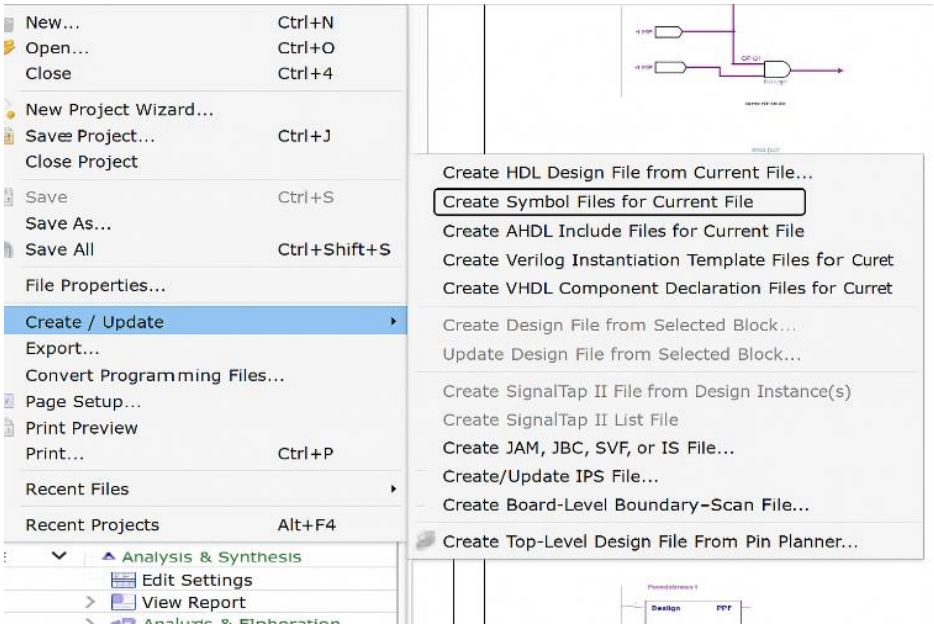
Figura 18 – Lógica interna do bloco Partida Direta.



Fonte: Elaborado pelo Autor, 2025.

Após a elaboração da lógica interna, o passo seguinte consiste na geração do símbolo gráfico da macrocélula, por meio do comando disponível na aba de ferramentas do Quartus II, conforme ilustrado na Figura 19. Esse procedimento permite encapsular o circuito em um bloco com entradas e saídas definidas, tornando sua organização, reutilização e visualização mais prática no projeto principal.

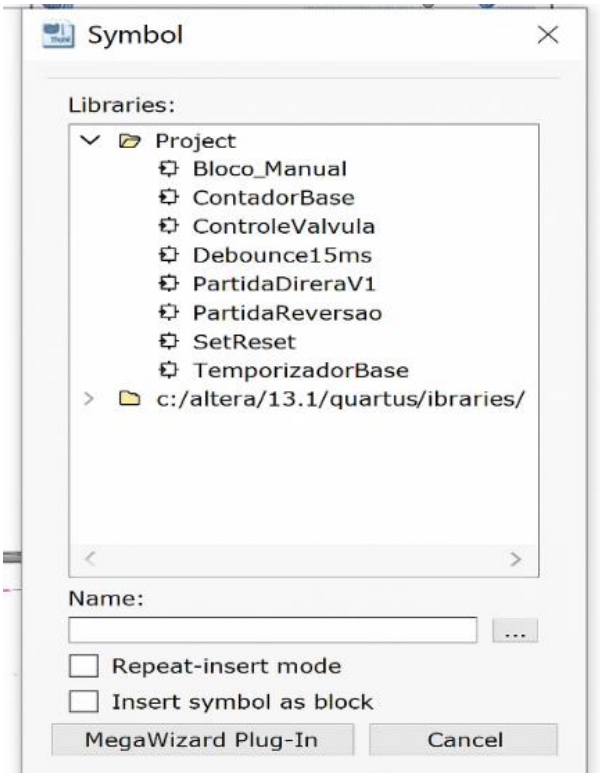
Figura 19 – Criação de arquivo de símbolo.



Fonte: Elaborado pelo Autor, 2025.

Uma vez gerada, a macrocélula é automaticamente armazenada na pasta de biblioteca denominada “Project”, situada no diretório principal do projeto, conforme ilustrado na Figura 20. Essa organização assegura que o bloco encapsulado permaneça disponível para reutilização em diversos pontos do projeto, eliminando a necessidade de recriação ou duplicação.

Figura 20 –Inserção de macrocélulas no programa principal.

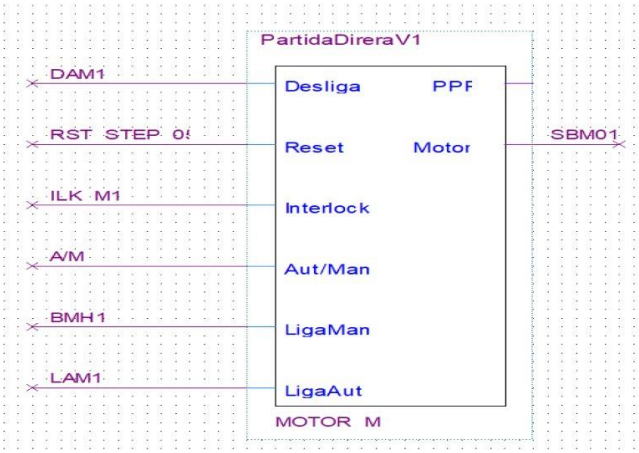


Fonte: Elaborado pelo Autor, 2025.

Essas macrocélulas, sejam construídas por meio de portas lógicas ou em VHDL, podem ser chamadas diversas vezes no programa principal, sendo tratadas como instâncias independentes. No entanto, é essencial que cada instância seja nomeada de forma única, pois a duplicação de identificadores resulta em erro de compilação.

A Figura 21 ilustra um exemplo prático de reutilização de uma macrocélula dentro do projeto principal, inserida por meio de um novo diagrama BDF. Os sinais de entrada e saída foram conectados conforme as necessidades do processo automatizado, permitindo uma abordagem modular, organizada e escalável para o desenvolvimento da lógica de controle.

Figura 21 – Chamada da macrocélula.



Fonte: Elaborado pelo Autor, 2025.

Dessa forma, a estratégia adotada neste trabalho possibilitou a construção de blocos lógicos robustos, modulares e reutilizáveis, promovendo maior clareza no desenvolvimento do projeto, padronização das conexões e facilitação das atividades de manutenção. A utilização de macrocélulas contribuiu para a organização do ambiente de programação, resultando em um esquema mais limpo e menos suscetível a erros. Ao encapsular conjuntos de instruções lógicas em blocos funcionais bem definidos, evita-se a duplicação desnecessária da lógica, reduzindo o risco de falhas decorrentes da replicação manual dos circuitos. Assim, o desenvolvedor pode concentrar-se na análise do comportamento das saídas em função das variações das entradas, sem a necessidade de revisar internamente toda a lógica do circuito a cada nova utilização. As seções subsequentes apresentam, de forma separada, os blocos desenvolvidos por meio do uso de portas lógicas e da descrição em VHDL, evidenciando a aplicação prática dessa abordagem modular.

4.3.2 Blocos Desenvolvidos com Portas Lógicas (BDF)

Nesta subseção, são apresentados os blocos lógicos concebidos por meio da ferramenta Block Diagram File (BDF) do ambiente de desenvolvimento Quartus II, que possibilita a modelagem de circuitos digitais a partir da interconexão de portas lógicas elementares. Tal abordagem foi adotada em virtude de sua eficácia na representação gráfica dos sistemas digitais e da facilidade proporcionada na compreensão das

funções combinacionais e sequenciais implementadas. Ademais, a estrutura visual do BDF potencializa os processos de depuração e validação de lógicas de complexidade simples a intermediária, configurando-se como uma metodologia adequada para contextos educacionais e prototipagem ágil, como no presente caso de estudo.

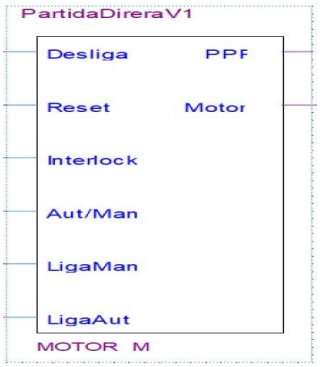
Considerando-se a análise da complexidade funcional dos módulos, verificou-se que os blocos apresentados a seguir foram implementados de forma otimizada e simplificada por meio da utilização direta de portas lógicas no BDF. A representação da lógica interna de cada bloco está disponível no Apêndice B, permitindo uma compreensão aprofundada das arquiteturas desenvolvidas.

4.3.2.1 Bloco: PartidaDiretaV1

O bloco PartidaDiretaV1 (Figura 22) foi desenvolvido para controlar a partida direta de um motor, incorporando lógica para os modos automático e manual. No modo automático, o bloco opera com um selo interno, requerendo um pulso de comando "Liga" para iniciar a operação e um pulso de "Desliga" para interrompê-la. No modo manual, o motor permanece acionado apenas enquanto o sinal de controle estiver ativo, sem a presença do selo.

Adicionalmente, o bloco dispõe de uma entrada de intertravamento (Interlock), que impede o acionamento caso o sinal correspondente não esteja em nível lógico alto. O sinal de saída PPP (Pronto Para Partida) indica quando todas as condições de habilitação foram satisfeitas, sinalizando que o sistema está apto para a operação do motor.

Figura 22 – Macrocélula Partida Direta.



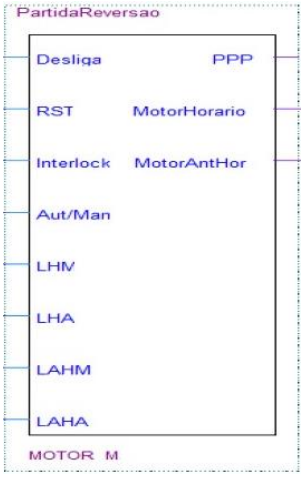
Fonte: Elaborado pelo Autor, 2025.

4.3.2.2 Bloco: PartidaReversão

O bloco PartidaReversão (Figura 23) foi desenvolvido a partir da estrutura do PartidaDiretaV1, com adaptações para permitir a reversão de rotação de motores. Assim como seu predecessor, ele suporta operação nos modos automático e manual, mantendo as características de selo interno no modo automático e acionamento direto no modo manual.

Além das funcionalidades básicas, o bloco PartidaReversão incorpora uma lógica de intertravamento entre os comandos de sentido horário e anti-horário, evitando a ativação simultânea de ambas as direções, o que assegura a proteção do motor e da carga acoplada. O bloco também utiliza o sinal de intertravamento (interlock) como condição de segurança, requerendo nível lógico alto para habilitação do sistema. O sinal PPP é empregado para indicar que todas as condições de partida segura foram atendidas.

Figura 23 – Macrocélula Partida Direta com Reversão.



Fonte: Elaborado pelo Autor, 2025.

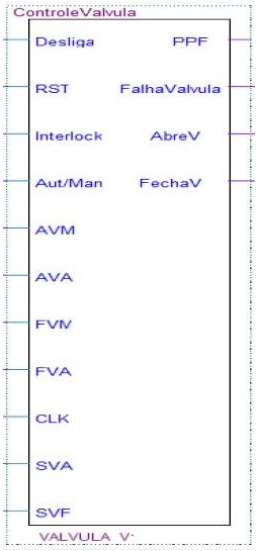
4.3.2.3 Bloco: ControleValvula

O bloco ControleValvula (Figura 24) foi desenvolvido com base na estrutura do bloco PartidaReversão, adaptando sua lógica para o controle de válvulas pneumáticas com monitoramento de atuação. Esse bloco inclui entradas de comando para avanço e recuo da válvula, além de sinais de sensores que indicam o término do movimento do cilindro.

Ao acionar o comando de avanço, inicia-se uma contagem regressiva — um tempo máximo de resposta configurado. Caso o sensor de avanço não seja ativado dentro deste intervalo, o bloco acusa falha de avanço. O mesmo ocorre no sentido oposto: ao acionar o comando de recuo, o bloco espera que o sensor de recuo seja ativado dentro do tempo máximo determinado; caso contrário, uma falha de recuo é sinalizada.

Além disso, o bloco implementa uma lógica de segurança que verifica se os sinais de avanço e recuo estão sendo ativados simultaneamente. Caso essa condição seja detectada, o sistema interpreta como uma falha crítica e bloqueia o acionamento.

Figura 24 – Macrocélula Controle Válvula.



Fonte: Elaborado pelo Autor, 2025.

4.3.2.4 Bloco: SetReset

O bloco SetReset (Figura 25) encapsula uma lógica de controle do tipo liga/desliga com priorização do sinal de reset, funcionando como um selo que mantém o estado ativado até que uma condição de desligamento seja acionada. Essa estrutura substitui a repetição manual de portas lógicas nos diagramas, simplificando o desenho e facilitando a leitura e manutenção do projeto. Sua aplicação é comum em circuitos de controle de estado, onde é necessário manter uma saída ativa até a ocorrência de uma condição específica de reinicialização.

Figura 25 – Macrocélula SetReset.

Fonte: Elaborado pelo Autor, 2025.

4.3.3 Blocos Desenvolvidos com VHDL

Nesta subseção, são apresentados os blocos lógicos desenvolvidos por meio da linguagem VHDL, uma abordagem textual amplamente utilizada no projeto de sistemas digitais complexos. Diferentemente dos blocos construídos com portas lógicas no ambiente gráfico BDF, os módulos descritos nesta seção foram implementados por meio de código descritivo, o que possibilita maior flexibilidade e controle sobre o comportamento dos sinais, especialmente em operações sequenciais e temporizadas.

A escolha pelo VHDL foi motivada pela natureza das funções, que demandam controle temporal, filtragem de sinais e parametrização de contadores. De acordo com avaliação prévia, tais blocos seriam implementados de maneira mais eficiente via VHDL do que por meio do BDF. A utilização de parâmetros genéricos (generics) permitiu a adaptação dos módulos conforme as necessidades específicas da aplicação, promovendo a reutilização e a escalabilidade do projeto. Ademais, foram adotadas práticas recomendadas e exemplos do autor D'Amore (2011), contribuindo para a robustez e a padronização da lógica desenvolvida.

As estruturas detalhadas e os códigos completos encontram-se disponíveis no Apêndice C.

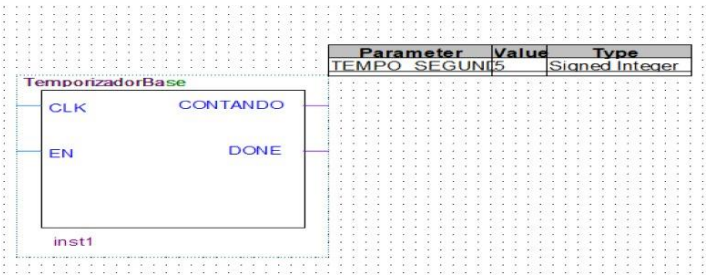
4.3.3.1 Bloco: TemporizadorBase

O bloco TemporizadorBase (Figura 26) foi desenvolvido para gerar um sinal de ativação após a contagem de um tempo predeterminado. Implementado em VHDL, este bloco recebe como parâmetro um valor genérico que define a quantidade de ciclos de clock (CLK) necessários para completar a temporização. Ele possui entrada de habilitação (EN), e fornece sinais de ativo (CONTANDO) e concluído (DONE). A

utilização de parâmetros genéricos (generics), característica da linguagem VHDL, permitiu que uma única descrição de hardware gerasse instâncias de temporizadores com tempos distintos, configurados dinamicamente, sem a necessidade de reescrever a lógica para cada aplicação, maximizando a reutilização eficiente dos recursos da FPGA.

Para alterar o tempo de contagem de uma instância específica, basta acessar as propriedades do bloco no ambiente de desenvolvimento e modificar o valor do parâmetro GENERIC ‘Value’. Essa estrutura permite sua reutilização em diferentes partes do projeto com tempos distintos, bastando ajustar o parâmetro genérico correspondente.

Figura 26 – Macrocélula TemporizadorBase.



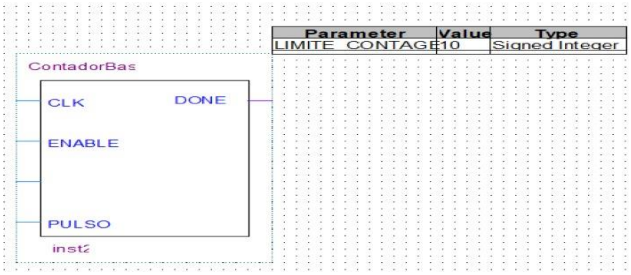
Fonte: Elaborado pelo Autor, 2025.

4.3.3.2 Bloco: ContadorBase

O bloco ContadorBase (Figura 27) realiza a contagem progressiva de pulsos (PULSO) até um valor limite parametrizado, sendo útil em processos que exigem contagem controlada de eventos ou ciclos de operação. Sua estrutura inclui as entradas de clock (CLK) e habilitação (ENABLE) e a saída para sinalização de conclusão (DONE).

Para alterar o limite de contagem de uma instância específica, basta acessar as propriedades do bloco no ambiente de desenvolvimento e modificar o valor do parâmetro GENERIC ‘Value’.

Figura 27 – Macrocélula ContadorBase.



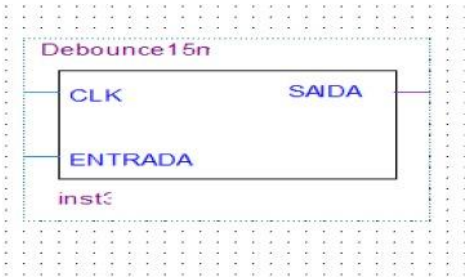
Fonte: Elaborado pelo Autor, 2025.

4.3.3.3 Bloco: Debounce15ms

O bloco Debounce15ms (Figura 28) foi desenvolvido para filtrar sinais provenientes de botões e sensores mecânicos sujeitos a interferências durante o acionamento. A necessidade de sua criação foi constatada durante os testes individuais da macrocélula do contador, onde se observou que um único pulso no botão físico resultava em múltiplos incrementos, evidenciando o ruído típico de contatos mecânicos.

A lógica implementada em VHDL monitora o sinal de entrada e o considera estável apenas após a persistência do mesmo nível lógico por um período mínimo de 15 milissegundos. Esse tempo foi determinado por meio de testes empíricos realizados durante o desenvolvimento da planta. Este bloco elimina falsas leituras e garante a confiabilidade no processamento dos comandos. O bloco apresenta entradas de clock e sinal de entrada e uma saída digital, e sua estrutura é compatível com diversas aplicações que envolvam entrada de usuário ou sensores sujeitos a ruídos.

Figura 28 – Macrocélula Debounce15ms.



Fonte: Elaborado pelo Autor, 2025.

4.4 Lógica *Top-Level* do Sistema

A lógica *top-level* do projeto foi implementada no arquivo Projeto01.bdf, desenvolvido no ambiente Quartus II 13.1. Esse arquivo representa a camada superior da estrutura lógica, na qual foram instanciadas todas as macrocélulas criadas ao longo do desenvolvimento, tanto em BDF quanto em VHDL. Sua principal função é integrar os blocos funcionais conforme a sequência operacional da planta automatizada descrita no item 4.1.1, adotando essa estrutura como padrão de funcionamento.

Durante a construção da lógica *top-level*, foram declaradas todas as entradas físicas, associadas a sensores e botões, bem como as saídas correspondentes aos atuadores. Cada sinal foi nomeado de forma intuitiva, refletindo sua função na planta, o que facilitou a leitura, a depuração e a manutenção do projeto. A interligação dos blocos foi organizada visualmente no BDF, respeitando a lógica sequencial do processo e considerando as interdependências entre os elementos de entrada, controle e saída.

O processo automatizado foi segmentado em etapas lógicas (steps), cada uma correspondendo a um estágio distinto da operação controlada. As transições entre essas etapas foram condicionadas a sinais provenientes de sensores e temporizadores, garantindo uma progressão sistemática, controlada e segura do ciclo operacional. Essa estrutura modular e hierarquizada favoreceu o desenvolvimento de uma lógica de controle clara, robusta e coerente com os requisitos funcionais da planta didática, promovendo maior facilidade na implementação, manutenção e análise do sistema.

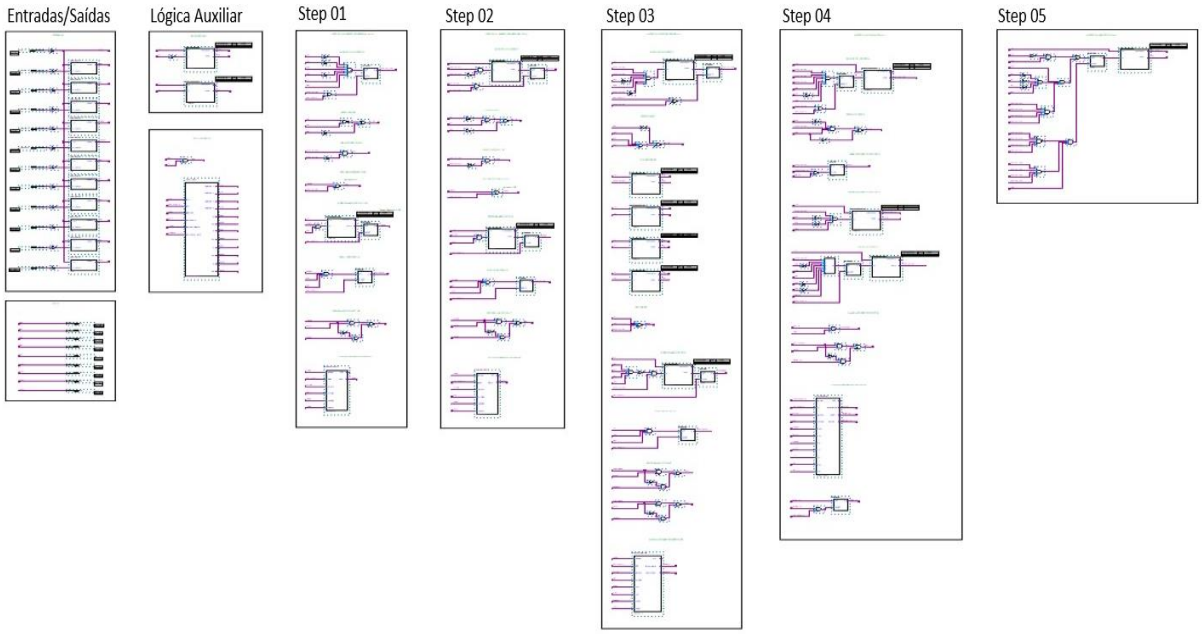
A sequência operacional foi estruturada em cinco etapas lógicas (steps) que se repetem ciclicamente:

- ***Step 1* – Condições iniciais + abastecimento do material A:** Verifica-se que o reservatório de destino está vazio, a válvula de descarga (V1) encontra-se fechada e todos os sensores indicam estado seguro. Habilita-se o ciclo automático e aciona-se a esteira A (M1), interrompendo-se o abastecimento ao atingir o sensor de nível médio (S2).
- ***Step 2* – Abastecimento do material B:** A esteira B (M2) é acionada após o nível médio, transportando o material B até o tanque. O processo se encerra quando o sensor de nível máximo (S1) é ativado.

- ***Step 3* – Mistura do lote:** O motor misturador (M3) entra em funcionamento por tempo determinado, promovendo a homogeneização do material. Ao fim do tempo, é desligado automaticamente.
- ***Step 4* – Abertura da válvula e descarga para o reservatório:** A válvula V1 é aberta e o conteúdo é transferido. O processo encerra-se quando o sensor S5 detecta reservatório cheio e tempo máximo de descarga expira.
- ***Step 5* – Reset de ciclos e selos de segurança:** Os temporizadores e selos de controle são reinicializados, garantindo que o sistema retorne às condições do *Step 1* para reinício do ciclo.

A Figura 29 apresenta uma visão esquemática geral da arquitetura top-level, evidenciando a disposição dos blocos funcionais e o layout das etapas que compõem o processo automatizado. Para uma visualização detalhada de cada step, consulte o Apêndice D. As lógicas internas dos blocos de macrocélulas utilizados nessa estrutura encontram-se descritas nos Apêndices B e C, conforme sua implementação em BDF e VHDL, respectivamente.

Figura 29 – Visão geral da lógica top-level com etapas do processo.

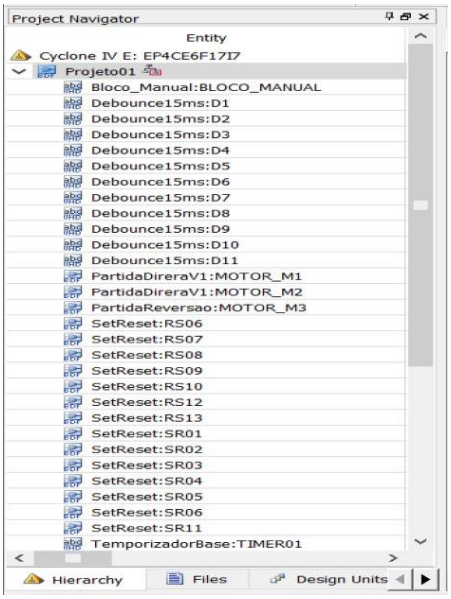


Elaborado pelo Autor, 2025.

Adicionalmente, a Figura 30 mostra os blocos lógicos efetivamente instanciados no nível top-level do projeto, conforme configurado no ambiente Quartus II. Essa visualização destaca como os componentes desenvolvidos foram organizados

e instanciados no projeto final, servindo como referência estrutural para validação e análise da lógica implementada.

Figura 30 – Blocos instanciados no nível top-level do projeto no Quartus II.



Fonte: Elaborado pelo Autor, 2025.

5 VALIDAÇÃO DO PROJETO

A validação teve como objetivo avaliar a robustez do hardware e a funcionalidade das macrocélulas desenvolvidas, tanto isoladamente quanto integradas à lógica hierárquica (*top-level*). Ademais, buscou-se assegurar a estabilidade e a confiabilidade da lógica sequencial implementada, conforme delineado na estrutura apresentada no Capítulo 4.

5.1 Validação do Hardware e Testes de Entradas/Saídas

A validação do hardware foi conduzida por meio de testes práticos no protótipo físico da planta didática, conectada ao sistema de controle implementado em FPGA. A principal estratégia adotada foi a verificação individual das entradas e saídas digitais, bem como a análise da estabilidade elétrica do sistema sob diversas condições operacionais.

Inicialmente, para o teste das entradas e saídas, foi desenvolvido um programa dedicado na FPGA, no qual cada botão físico da bancada acionava individualmente uma saída possibilitando a conferência do mapeamento lógico e da integridade dos sinais. Durante esses testes, utilizou-se um multímetro digital Minipa ET-1507B para medir os níveis de tensão nas entradas e saídas do sistema. As leituras confirmaram que os sinais estavam dentro do padrão LVTTL (próximos de 0 V e 3,3 V), e que os resistores de pull-down de 10 kΩ estavam atuando corretamente, evitando estados flutuantes nas entradas digitais. As saídas também foram avaliadas individualmente para verificar se cada relé era corretamente acionado e desligado, sem ruídos ou falhas.

Entretanto, ao acionar simultaneamente todas as saídas conectadas ao Slot 3 da placa de relés, observou-se uma queda significativa da tensão de alimentação proveniente da FPGA, que reduzia de 3,3 V para aproximadamente 2,9 V. Essa limitação da tensão fornecida pela FPGA comprometia a estabilidade do sistema, podendo gerar falhas intermitentes no acionamento de alguns relés sob plena carga.

A solução adotada consistiu na desativação da alimentação proveniente da FPGA ao Slot 3, substituindo-a pela saída de 3,3 V da fonte externa, devidamente dimensionada para fornecer corrente suficiente a todos os canais de relé simultaneamente. Para garantir a integridade elétrica e evitar diferenças de potencial

entre os circuitos, foi realizada a equipotencialização dos terminais GND da fonte externa e da placa FPGA.

Após essa modificação, os testes foram repetidos com sucesso. Todas as saídas passaram a operar de forma estável e com a tensão nominal constante (3,3 V), mesmo sob carga total. Essa solução aumentou a robustez elétrica e a confiabilidade do sistema, assegurando que o controle digital fosse executado sem sobrecargas e com segurança adequada.

5.2 Validação das Macrocélulas

A validação das macrocélulas desenvolvidas foi realizada por meio de testes físicos diretos na bancada didática, com o objetivo de assegurar o correto funcionamento de cada bloco lógico implementado na FPGA. Cada macrocélula foi isoladamente conectada a entradas e saídas reais do processo, como botões e LEDs, permitindo a simulação de sinais de controle e a observação da resposta esperada.

Todos os blocos foram avaliados em condições normais e sob diferentes combinações de sinais, contemplando as lógicas de intertravamento, acionamento automático, acionamento manual e resposta das entradas.

5.2.1 Blocos de Controle de Motores — PartidaReversão e PartidaDiretaV1

As macrocélulas de controle de motores desenvolvidas neste projeto contemplam as lógicas de partida direta e reversão de sentido, voltadas ao acionamento de cargas com diferentes formas de controle. A validação desses blocos foi conduzida considerando três condições principais: modo manual, modo automático e sinal de interlock.

Nos testes em modo manual, constatou-se que as saídas permanecem ativas apenas enquanto o botão físico correspondente mantém nível lógico alto, caracterizando controle direto e momentâneo. Já em modo automático, os blocos responderam corretamente aos comandos de ligar e desligar, priorizando sempre a condição de desligamento quando está se apresentava ativa, conforme programado na lógica do processo.

O interlock, por sua vez, atua como uma entrada de segurança externa que, quando ativada, força imediatamente a saída da macrocélula para nível lógico baixo,

independentemente dos demais comandos. Esse comportamento foi validado em bancada e se mostrou eficaz na prevenção de acionamentos indevidos, reforçando a robustez lógica das macrocélulas desenvolvidas.

Esses testes confirmaram que as macrocélulas de partida direta e reversão operaram conforme o programado, respeitando as condições estabelecidas na lógica do sistema e contribuindo para a confiabilidade geral da automação.

5.2.2 Bloco de Controle da Válvula — ControleValvula

A macrocélula responsável pelo controle da válvula (VálvulaV1) foi validada em condições de operação manual e automática. Em ambos os modos, a válvula era corretamente aberta ou fechada conforme os comandos recebidos, obedecendo à lógica de controle programada. A estrutura da macrocélula incorporou verificações de segurança, garantindo que a válvula não operasse em estados indevidos ou sem a devida confirmação de posicionamento.

As entradas digitais SVA (Sensor de Válvula Aberta) e SVF (Sensor de Válvula Fechada) foram devidamente monitoradas pela lógica da macrocélula. O bloco reagiu corretamente às mudanças dos sensores, fornecendo retroalimentação confiável ao sistema de controle. O comando de abertura ou fechamento somente era finalizado após a confirmação da posição esperada, reforçando a precisão operacional.

Adicionalmente, a lógica foi configurada para gerar falha caso os sensores não comutassem dentro do tempo estipulado, indicando possíveis travamentos ou anomalias mecânicas. Essa abordagem garante maior confiabilidade e integridade ao processo.

Outro elemento validado foi o sinal de interlock, utilizado como medida de segurança adicional. Quando ativo, o interlock forçava a saída da macrocélula para nível lógico baixo (0), impedindo qualquer operação da válvula, mesmo diante de comandos manuais ou automáticos. Esse comportamento assegura a impossibilidade de acionamento acidental durante manutenções ou em situações críticas do processo.

Esses testes confirmaram que o bloco de controle da válvula operou conforme projetado, respeitando integralmente os comandos, restrições e condições de segurança definidas pelo sistema de controle.

5.2.3 Blocos TemporizadorBase, ContadorBase e *Debounce15ms*

As macrocélulas TemporizadorBase e ContadorBase foram desenvolvidas com o objetivo de introduzir monitoramentos temporais e realizar contagens discretas dentro da lógica de controle da planta automatizada. Ambas as estruturas foram testadas individualmente em bancada, utilizando o acionamento por botões físicos e saídas conectadas a LEDs, permitindo a validação visual do comportamento esperado.

Durante os testes do TemporizadorBase, foram avaliadas as saídas de tempo ativo (ACTIVE) e tempo finalizado (DONE), comparando-se os tempos programados internamente com cronômetros digitais externos — incluindo aplicativos de smartphone e o relógio do sistema do computador. Os resultados demonstraram coerência com os tempos parametrizados, respeitando margens de erro aceitáveis para aplicações didáticas. Isso validou a exatidão do bloco como base para comandos temporizados do processo.

A macrocélula ContadorBase, por sua vez, demonstrou comportamento funcional correto no controle de contagens ascendentes baseadas em pulsos digitais. Contudo, durante os testes iniciais, foi observado um problema recorrente: a trepidação mecânica (*bouncing*) dos botões físicos gerava múltiplos pulsos para um único toque, ocasionando incrementos indevidos na contagem. Tal falha comprometia a confiabilidade da lógica de controle sequencial.

Para contornar esse problema, foi implementada a macrocélula Debounce15ms, responsável por filtrar os sinais vindos de entradas mecânicas. Essa macrocélula assegura que apenas transições de sinal com duração superior a 15 milissegundos sejam reconhecidas como válidas, eliminando os ruídos de comutação típicos de botões. Com a inserção do filtro, os testes foram repetidos e demonstraram total estabilidade: cada acionamento físico passou a ser interpretado como um único pulso válido.

A utilização conjunta das macrocélulas TemporizadorBase, ContadorBase e Debounce15ms assegurou a integridade do controle temporal e sequencial do processo, permitindo respostas coerentes ao ambiente físico da planta. A aplicação do Debounce demonstrou ser indispensável para garantir a robustez e previsibilidade do sistema, especialmente em contextos educacionais ou industriais onde entradas mecânicas estão presentes.

5.2.4 Lógica Top-Level

A lógica top-level do sistema representa a integração completa das macrocélulas desenvolvidas ao longo deste projeto, conforme descrito no Capítulo 4.4. Essa estrutura foi implementada no ambiente Quartus II utilizando arquivos BDF (Block Diagram File), conectando os blocos funcionais de controle de motores, válvula, temporizadores, contadores, debounce e lógica de estado em um único diagrama de controle. Essa lógica condensa e executa todo o ciclo operacional da planta, conforme o fluxo do processo apresentado na Figura 10.

A validação da lógica integrada foi realizada diretamente na planta física, utilizando os sinais reais de sensores (S1 a S5), entradas digitais (como seleção de modo, acionamento de ciclo e botão de reinicialização) e atuadores (relés e LEDs) conectados à FPGA. O funcionamento completo do sistema foi testado com base na sequência definida no fluxo do processo, desde as condições iniciais de operação (tanque vazio, válvula fechada e reservatório disponível) até o reabastecimento, mistura temporizada, verificação de sensores e esvaziamento automatizado do tanque.

Durante os testes, cada etapa foi acompanhada em tempo real, com verificação da troca correta de sinais entre as macrocélulas e da resposta das saídas. Os sinais internos, como habilitação, conclusão de operação, atividade em execução e reinicialização, garantiram a coordenação entre os blocos, evitando sobreposição de comandos ou falhas de sincronização. A lógica respondeu adequadamente às condições programadas, atuando com segurança mesmo em situações de falha ou inconsistência — como ausência de leitura de sensores ou ativação de intertravamento.

O sistema também foi testado quanto à alternância entre os modos manual e automático. No modo manual, foi possível acionar diretamente os motores e a válvula por meio de botões físicos, com resposta imediata. Já no modo automático, o processo seguiu o fluxo lógico estabelecido, com transições entre fases monitoradas e controladas de forma segura e eficiente.

Dessa forma, a lógica top-level demonstrou funcionamento robusto e confiável, capaz de conduzir o processo automatizado com precisão. A associação entre a lógica digital e o fluxo do processo representado na Figura 10 evidencia a fidelidade do

sistema ao projeto original, validando sua aplicação tanto em ambientes didáticos quanto em simulações práticas de controle sequencial.

6 CONCLUSÃO

O presente trabalho teve como objetivo principal o Desenvolvimento de Macrocélulas Lógicas Modulares em FPGA para Controle Automatizado de Processos, aplicado a uma planta didática de mistura e transferência de materiais. A proposta consistiu na modelagem desses blocos lógicos reutilizáveis, implementados nos formatos BDF e VHDL, integrados em nível top-level, e testados com sinais reais de entrada e saída.

Durante o desenvolvimento, buscou-se conciliar simplicidade na implementação com clareza na estruturação lógica, adotando uma abordagem modular e escalável. A utilização dessas macrocélulas possibilitou a construção de uma lógica de controle sequencial distribuída em cinco etapas (steps), cada uma com condições de transição fundamentadas em sensores e temporizadores, favorecendo tanto a compreensão do processo quanto a organização interna do sistema.

A validação prática realizada em bancada confirmou a funcionalidade, confiabilidade e robustez do sistema. Desafios práticos como a filtragem de ruídos em entradas mecânicas (através da Macrocélula Debounce15ms) e a otimização da alimentação do hardware foram solucionados e testados, assegurando a integridade dos sinais e a operação estável e determinística do controle digital, mesmo sob carga máxima. A precisão na temporização dos ciclos foi verificada, demonstrando a coerência dos resultados com os parâmetros programados.

Com base nos resultados obtidos, conclui-se que os objetivos propostos foram plenamente atingidos. O projeto demonstrou a viabilidade do emprego de FPGAs na implementação desses blocos lógicos modulares para controle de processos automatizados, validando sua aplicabilidade didática e protótipa.

Dessa forma, este estudo não apenas oferece uma solução funcional e confiável para o controle automatizado de processos, mas também evidencia o potencial das FPGAs como uma tecnologia-chave para a automação industrial, especialmente em aplicações que demandam alta velocidade, paralelismo e tempo de resposta preciso. Para trabalhos futuros, sugere-se a expansão deste projeto para o desenvolvimento de uma interface de supervisão e controle remoto usando UART.

Para consolidar o caráter prático e replicável deste estudo, e com o intuito de contribuir para a comunidade acadêmica e de engenharia, todos os arquivos de projeto, incluindo os diagramas de blocos (BDF), códigos VHDL das macrocélulas e

esquemas elétricos utilizados, estão publicamente disponíveis para consulta e replicabilidade no repositório GitHub, acessível através do link: https://github.com/WdsonMarcal/FPGA-Automacao-Macrocelulas.git.

REFERÊNCIAS

ANTONIO, V.J. **Obtenido de automatizacion em el processo de manufactura**. 2012. Disponível em: https://es.slideshare.net/jvelasquezc/automatizacin-en-el-proceso-de-manufactura. Acesso em: 17 jun. 2023.

BOLTON, W. **Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering**. 6. ed. Harlow: Pearson, 2015.

BOLTON, W. **Programmable Logic Controllers**. EUA: Newnes, 2009.

CODÁ, Luiza Maria Romeiro. **Dispositivos Lógicos Programáveis**. São Carlos: Universidade de São Paulo – Escola de Engenharia de São Carlos, Departamento de Engenharia Elétrica e de Computação, 2020. Apostila da disciplina SEL 405 – Laboratório de Introdução aos Sistemas Digitais I.

COSTA, C. **Implementação de Controlador Lógico baseado em Lógica Programável Estruturada (FPGA)**, 2014.

CRESPO, J. **Control practices using Simulink with Arduino as Low Cost Hardware**. Symposium on Advances in Control Education. 2013.

D'AMORE, R. VHDL: **Descrição e síntese de circuitos digitais**. 2ª ed. Rio de Janeiro: LTC, 2012.

HOROWITZ, Paul; HILL, Winfield. **A arte da eletrônica**. 3. ed. Cambridge: Cambridge University Press, 2016.

INTERNATIONAL ELECTROTECHNICAL COMMISSION – IEC. **IEC 60664-1: Insulation Coordination for Equipment within Low-Voltage Systems**. 2007.

INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC). IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. Genebra: IEC, 2010. Parte 1 a 7.

INTEL. **Cyclone IV Device Handbook**. Volume 1: Device Overview, Design Guidelines, and Device Datasheet. Santa Clara, CA: Intel Corporation, 2015. Disponível em: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51001.pdf. Acesso em: 10 jun. 2025.

INTEL. **Quartus II Handbook**. Volume 1: Design and Synthesis. San Jose, CA: Intel Corporation, 2013. Disponível em: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/qts/qts_qii51007.pdf. Acesso em: 10 jun. 2025.

INTEL CORPORATION**. Intel FPGA Download Cable User Guide**. Rev. C. San Jose, CA: Intel, 2009. Disponível em: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_usb_blstr.pdf. Acesso em: 11 jun. 2025.

JIANG, Junye; ZHOU, Yaan; GONG, Yuanhao; YUAN, Haoxuan; LIU, Shuanglong. **FPGA-based Acceleration for Convolutional Neural Networks: A Comprehensive Review**. 2025.

JUNIOR, Marcos Augusto Faglioni. **Filtro stateless a 10G definido em bluespec e implementado em FPGA**. Trabalho de Conclusão de Curso (Graduação – Engenharia de Computação). Universidade Federal de São Carlos. São Carlos, 2021.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos de metodologia científica**. 8. ed. São Paulo: Atlas, 2017.

NADIR, F. E.; BSISS, M.; JBILOU, M.; AMAMI, B. **Safety Fuzzy Logic Controller with architecture implemented in FPGA**. 2016 5Th International Conference On Systems And Control (Icsc), [S.L.], p. 186-191, maio 2016. IEEE.

PAIVA, Davi Alexandre**. Projeto de controlador PI usando um dispositivo SOC FPGA e processador ARM**. Trabalho de Conclusão de Curso (Graduação – Engenharia Elétrica). Universidade Federal do Ceará. Fortaleza, 2019.

RAMALHO, L. A. **Uso de linguagem de descrição de hardware e dispositivos de alto desempenho na educação tecnológica**. Jornada de Pesquisa e Extensão. Cuiabá, 2013.

RIBEIRO, Fabricio Leonardo et al. **Avaliação técnica e de viabilidade econômica da utilização da tecnologia PLC**. Conexão Ciência e Tecnologia, v.14, n.3, 2020.

ROPELATO, Giovani. **Desenvolvimento de um controlador para sistema de coordenadas XZ descrito em VHDL e implementado em FPGA**. Dissertação (Mestrado – Mecatrônica). Instituto Federal de Santa Catarina. Florianópolis, 2019.

SEDRA, A. S.; SMITH, K. C. **Microelectronic Circuits**. 7. ed. New York: Oxford University Press, 2010.

SILVA, Hugo Vinicius. **Metodologia de projeto de automação industrial visando a conversão automática de redes de petri interpretadas em códigos implementáveis**. Dissertação (Mestrado – Engenharia Elétrica). Universidade Tecnológica Federal do Paraná. 2013.

TEIXEIRA, M. A. **Técnicas de reconfigurabilidade dos FPGAs da família APEX 20K Altera**. Dissertação (Mestrado). São Carlos, 2002.

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas Digitais: Princípios e Aplicações**. 10ª ed. São Paulo: Pearson, 2007.

WEBER, A. F. **Arquitetura FPGAs e CPLDs da ALTERA**, 2016.

ZAGHETTO, A.; PRADO, A. C.; TAVARES, A. **Trabalho sobre FPGA**. Disponível em: https://www.gta.ufrj.br/grad/01_1/pld/index.html. Acesso em: 17 jun. 2023.

ZANZOTI, Fernando Henrique Oliveira. **Desenvolvimento do protótipo de hardware de um micro PLC**. Trabalho de Conclusão de Curso (Graduação – Engenharia Elétrica). Universidade Federal de Uberlândia. Uberlândia, 2019.

APÊNDICE A - DIAGRAMA ELÉTRICO

0	1	2	3	4	5	6	7

COMPONENTES

1 - Kit de desenvolvimento FPGA Altera, Cyclone IV, EP4CE6, NIOS II, placa PCB e programador USB - 1 UNIDADE

2 - Módulo relé 3V 10A, 8 canais com borne KRE - 3 UNIDADES

3 - Placa bloco terminal IDC-40 - 1 UNIDADE

4 - Fonte ATX 12V K-MEX PX-300CNG - 1 UNIDADE

5 - Chave alavanca 2 posições, 3 pinos, MTS-102, 6A - 7 UNIDADES

6 - Chave botão PBS-110 verde, push button NA 2T - 3 UNIDADES

7 - Porta fusível de rosca para painel 5×20 - 3 UNIDADES

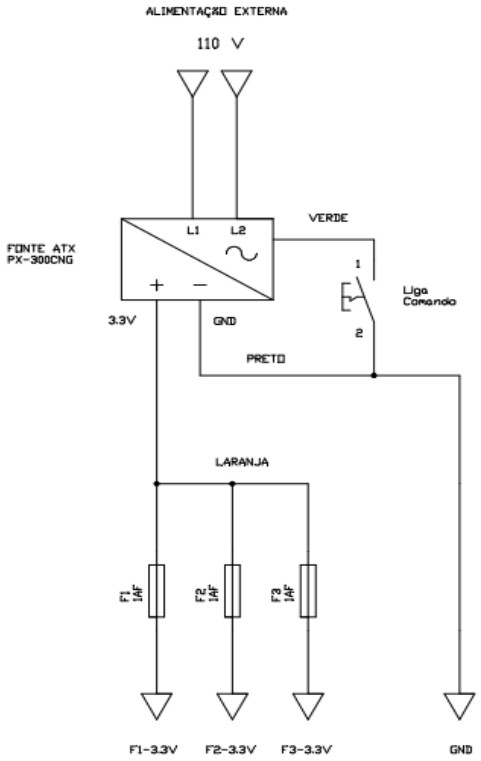
8 - Resistor 10kΩ, 5%, 1/4W, CR25 - 10 UNIDADES

9 - Resistor 220Ω, 5%, 1/4W, CR25 - 25 UNIDADES

10 - Cabo flexível 0,75 mm², cabinho colorido 18 AWG - 20 METROS

11 - LED 5 mm verde - 10 UNIDADES

	REVISION	DATE	NAME	PROJECT	PAGE
A	V1	18/04/2025	WDSON		2 DE 9
B					
C					



FONTE ATX

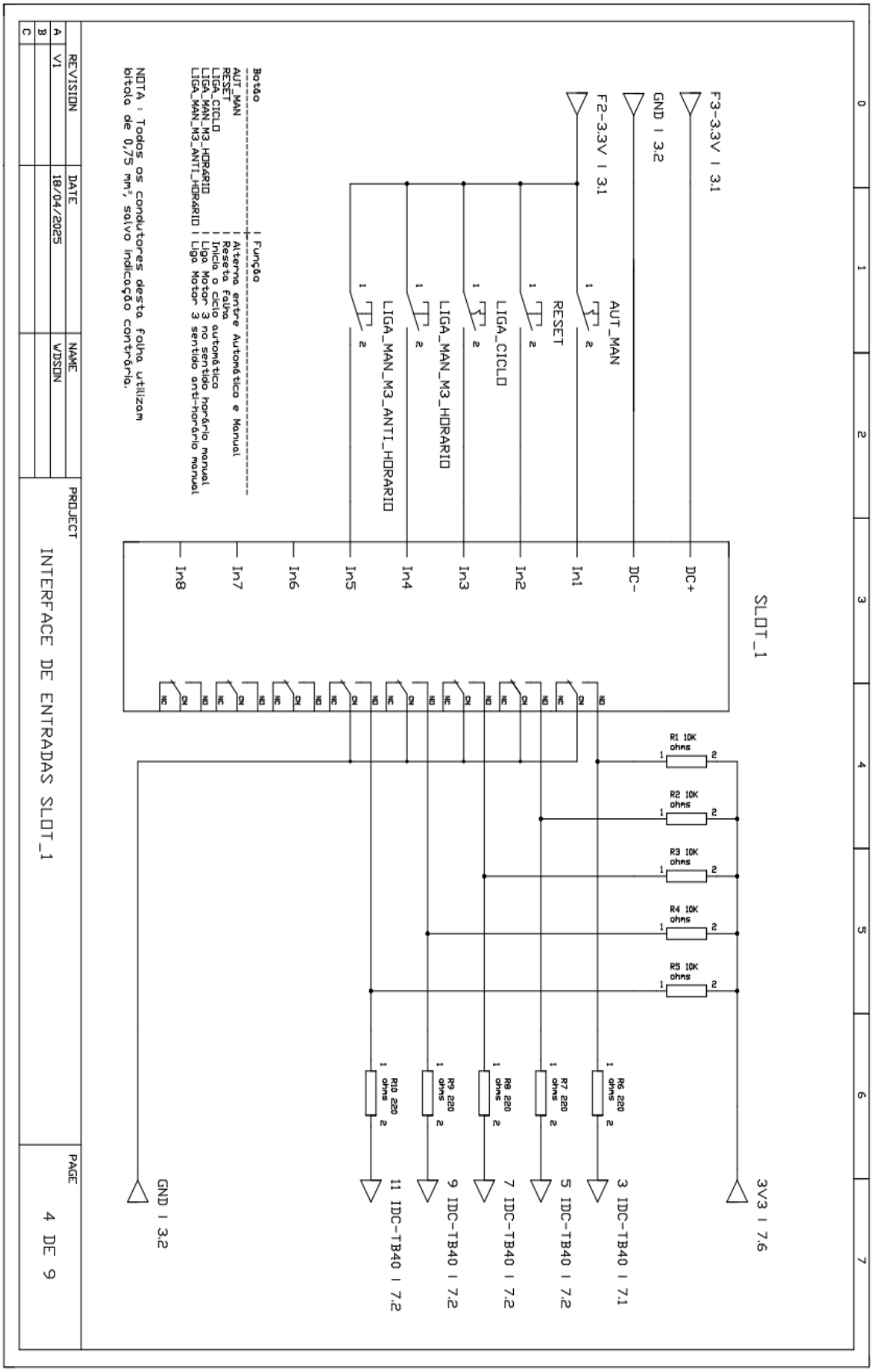
PRETO	GND
LARANJA	+3.3V
VERDE	PS_ON
AMARELO	+12V
VERMELHO	+5V
AZUL	-12V

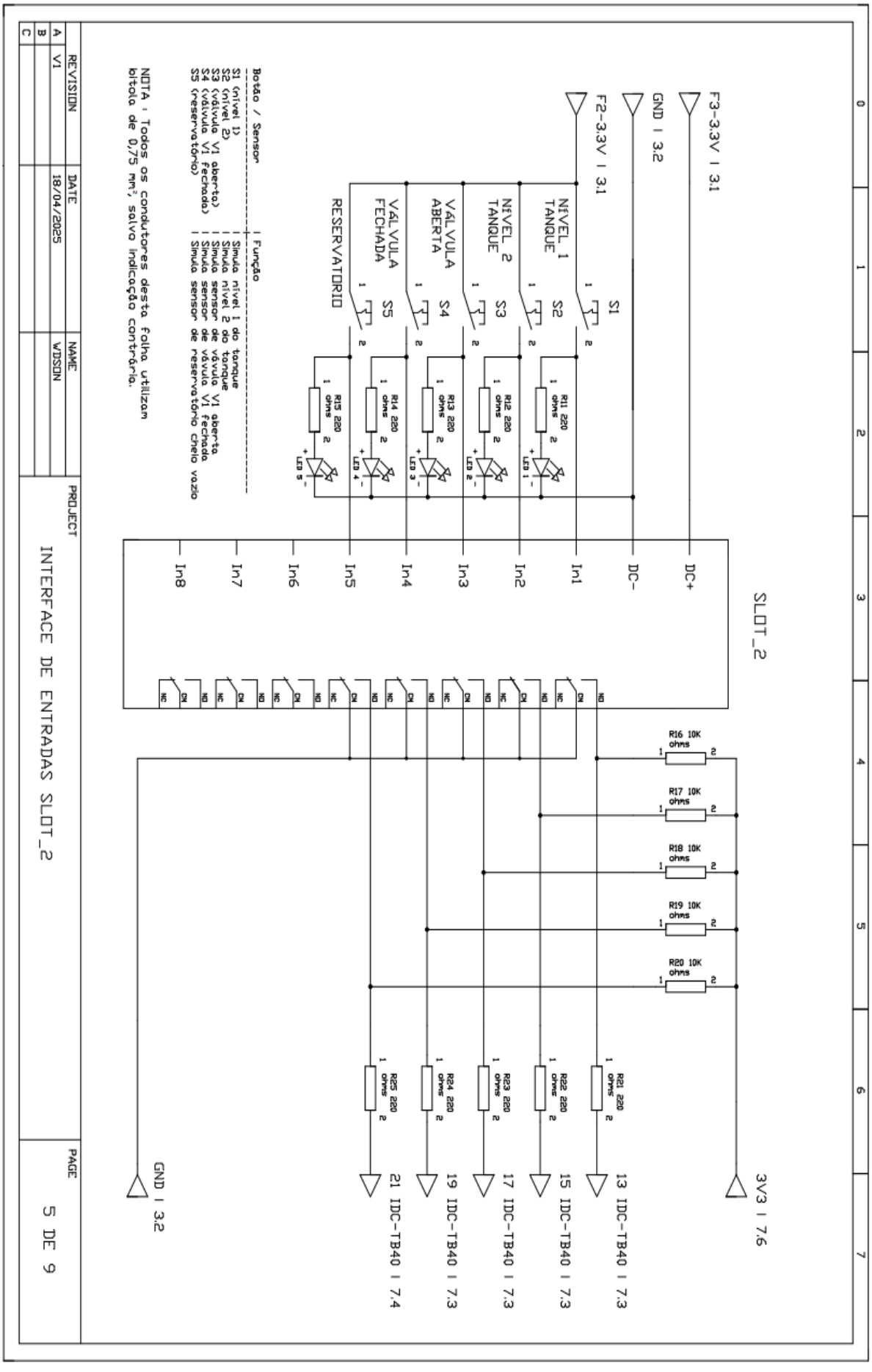
CIRCUITOS

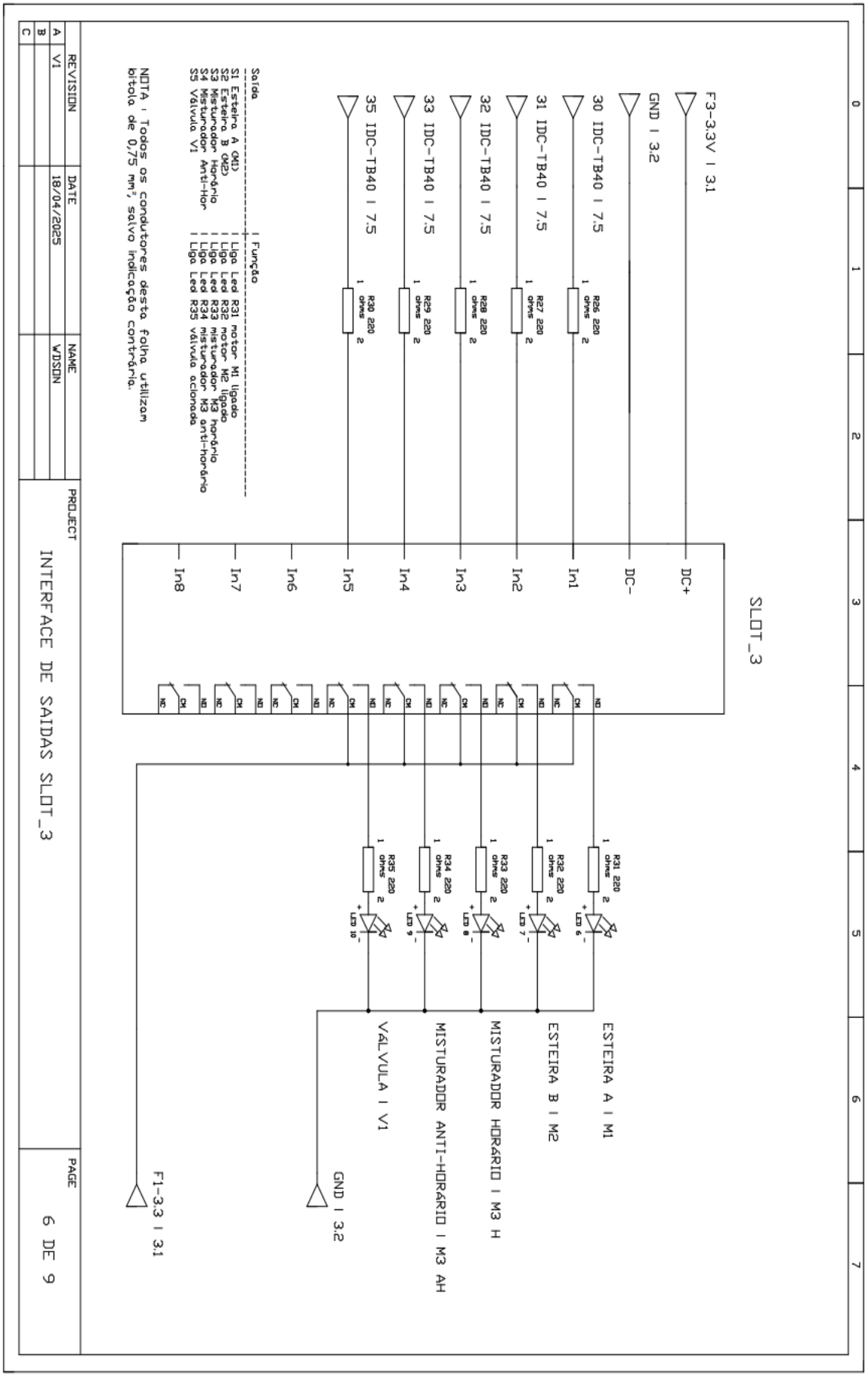
F1-3.3V	CARGA SAIDA
F2-3.3V	BOTÕES ENTRADA
F3-3.3V	ALIMENTAÇÃO INTERFACES DC+ DC-

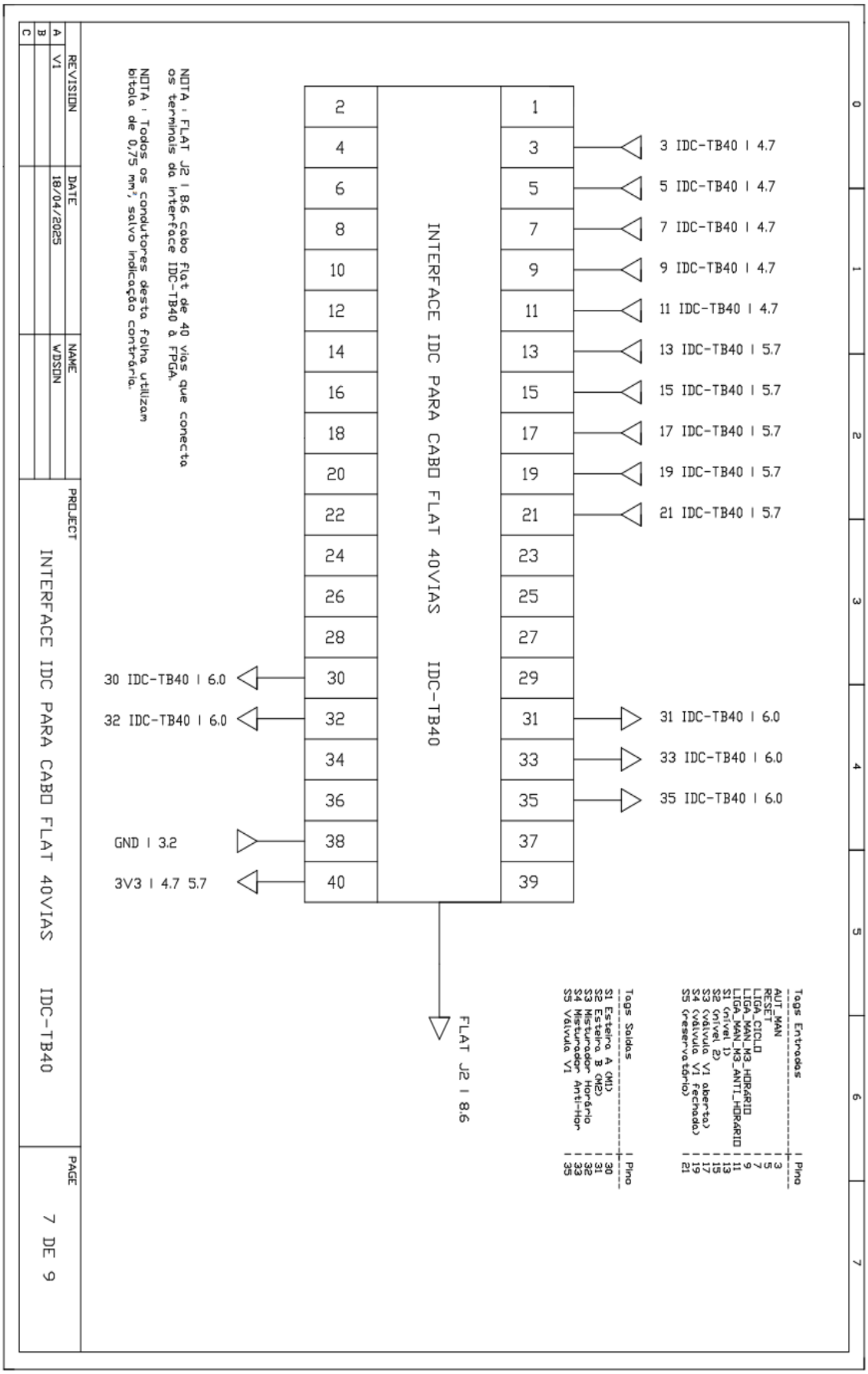
NOTA : Todos os condutores desta folha utilizam bitola de 0,75 mm², salvo indicação contrária.

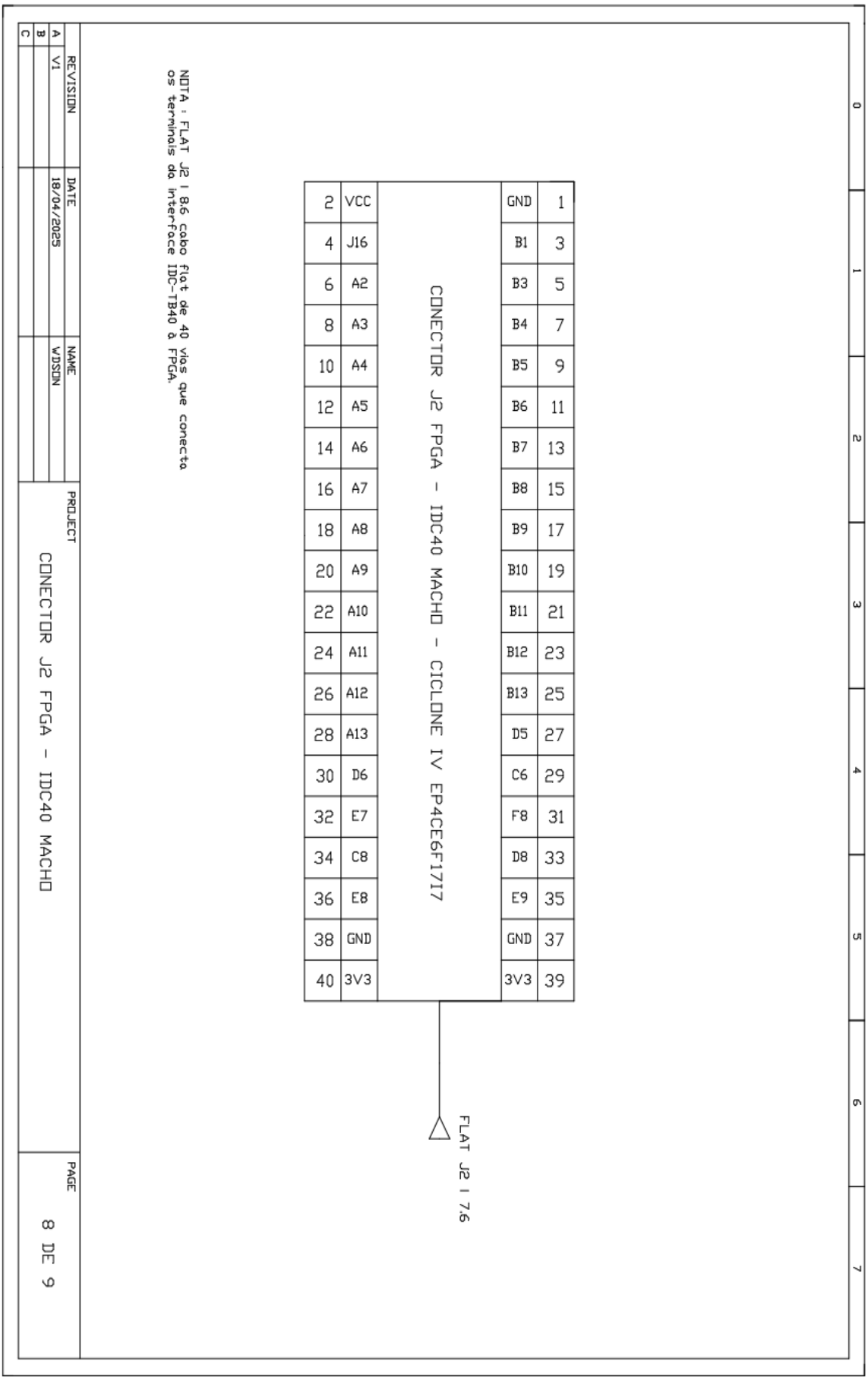
	REVISION	DATE	NAME	PROJECT	PAGE
A	V1	18/04/2025	WDSON	FONTE ATX	3 DE 9
B					
C					

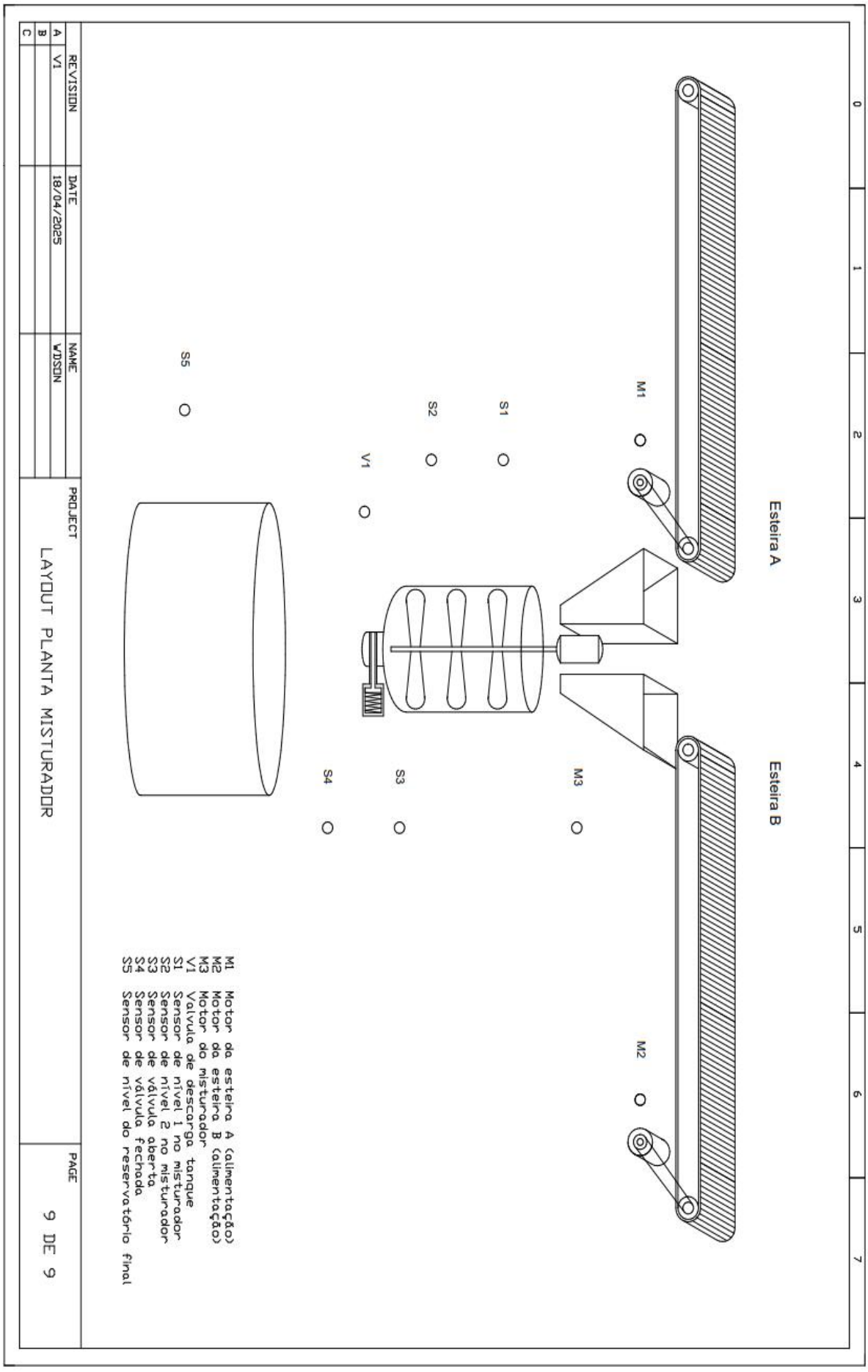

SLOT_1
F3-3.3V | 3.1
GND | 3.2
F2-3.3V | 3.1
AUT_MAN
RESET
LIGA_CICLO
LIGA_MAN_M3_HORARIO
LIGA_MAN_M3_ANTI_HORARIO
DC+
DC-
In1
In2
In3
In4
In5
In6
In7
In8
Botão | Função
AUT_MAN | Alterna entre Automático e Manual
RESET | Reseta falha
LIGA_CICLO | Inicia o ciclo automático
LIGA_MAN_M3_HORARIO | Liga Motor 3 no sentido horário manual
LIGA_MAN_M3_ANTI_HORARIO | Liga Motor 3 sentido anti-horário manual
NOTA : Todos os condutores desta folha utilizam bitola de 0,75 mm², salvo indicação contrária.
R1 10K ohms
R2 10K ohms
R3 10K ohms
R4 10K ohms
R5 10K ohms
R6 220 ohms
R7 220 ohms
R8 220 ohms
R9 220 ohms
R10 220 ohms
3V3 | 7.6
3 IDC-TB40 | 7.1
5 IDC-TB40 | 7.2
7 IDC-TB40 | 7.2
9 IDC-TB40 | 7.2
11 IDC-TB40 | 7.2
GND | 3.2
REVISION
V1
DATE
18/04/2025
NAME
WDSON
PROJECT
INTERFACE DE ENTRADAS SLOT_1
PAGE
4 DE 9


F3-3.3V | 3.1
GND | 3.2
F2-3.3V | 3.1
NÍVEL 1 TANQUE
NÍVEL 2 TANQUE
VÁLVULA ABERTA
VÁLVULA FECHADA
RESERVATÓRIO
S1
S2
S3
S4
S5
R11 220 ohms
R12 220 ohms
R13 220 ohms
R14 220 ohms
R15 220 ohms
LED 1
LED 2
LED 3
LED 4
LED 5
SLOT_2
DC+
DC-
In1
In2
In3
In4
In5
In6
In7
In8
R16 10K ohms
R17 10K ohms
R18 10K ohms
R19 10K ohms
R20 10K ohms
R21 220 ohms
R22 220 ohms
R23 220 ohms
R24 220 ohms
R25 220 ohms
3V3 | 7.6
13 IDC-TB40 | 7.3
15 IDC-TB40 | 7.3
17 IDC-TB40 | 7.3
19 IDC-TB40 | 7.3
21 IDC-TB40 | 7.4
GND | 3.2
Botão / Sensor | Função
S1 (nível 1) | Simula nível 1 do tanque
S2 (nível 2) | Simula nível 2 do tanque
S3 (válvula V1 aberta) | Simula sensor de válvula V1 aberta
S4 (válvula V1 fechada) | Simula sensor de válvula V1 fechada
S5 (reservatório) | Simula sensor de reservatório cheio vazio
NOTA : Todos os condutores desta folha utilizam bitola de 0,75 mm², salvo indicação contrária.
REVISION
DATE
NAME
A V1 18/04/2025 WISON
PROJECT
INTERFACE DE ENTRADAS SLOT_2
PAGE
5 DE 9


SLOT_3
F3-3.3V | 3.1
GND | 3.2
30 IDC-TB40 | 7.5
31 IDC-TB40 | 7.5
32 IDC-TB40 | 7.5
33 IDC-TB40 | 7.5
35 IDC-TB40 | 7.5
DC+
DC-
In1
In2
In3
In4
In5
In6
In7
In8
R26 220 ohms
R27 220 ohms
R28 220 ohms
R29 220 ohms
R30 220 ohms
R31 220 ohms
R32 220 ohms
R33 220 ohms
R34 220 ohms
R35 220 ohms
LED 6
LED 7
LED 8
LED 9
LED 10
ESTEIRA A | M1
ESTEIRA B | M2
MISTURADOR HORÁRIO | M3 H
MISTURADOR ANTI-HORÁRIO | M3 AH
VÁLVULA | V1
GND | 3.2
F1-3.3 | 3.1
Saída | Função
S1 Esteira A (M1) | Liga Led R31 motor M1 ligado
S2 Esteira B (M2) | Liga Led R32 motor M2 ligado
S3 Misturador Horário | Liga Led R33 misturador M3 horário
S4 Misturador Anti-Hor | Liga Led R34 misturador M3 anti-horário
S5 Válvula V1 | Liga Led R35 válvula acionada
NOTA | Todos os condutores desta folha utilizam bitola de 0,75 mm², salvo indicação contrária.
REVISION
A V1
B
C
DATE
18/04/2025
NAME
WDSDN
PROJECT
INTERFACE DE SAIDAS SLOT_3
PAGE
6 DE 9

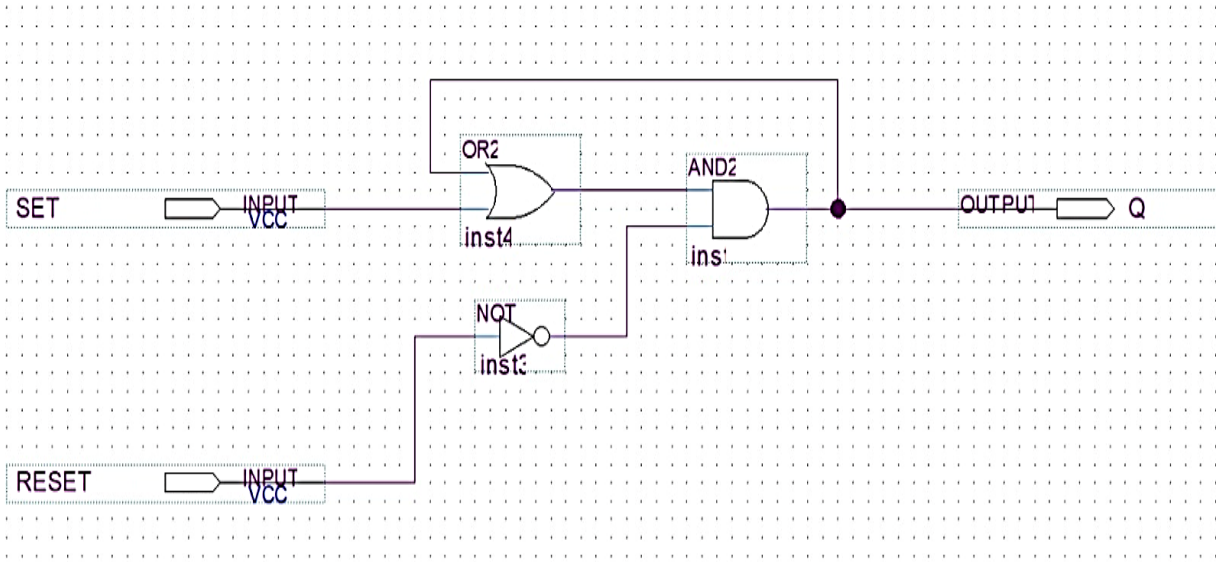

INTERFACE IDC PARA CABO FLAT 40VIAS IDC-TB40
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
3 IDC-TB40 | 4.7
5 IDC-TB40 | 4.7
7 IDC-TB40 | 4.7
9 IDC-TB40 | 4.7
11 IDC-TB40 | 4.7
13 IDC-TB40 | 5.7
15 IDC-TB40 | 5.7
17 IDC-TB40 | 5.7
19 IDC-TB40 | 5.7
21 IDC-TB40 | 5.7
31 IDC-TB40 | 6.0
33 IDC-TB40 | 6.0
35 IDC-TB40 | 6.0
30 IDC-TB40 | 6.0
32 IDC-TB40 | 6.0
GND | 3.2
3V3 | 4.7 5.7
FLAT J2 | 8.6
Tags Entradas | Pino
AUT_MAN | 3
RESET | 5
LIGA_CICLO | 7
LIGA_MAN_M3_HORARIO | 9
LIGA_MAN_M3_ANTI_HORARIO | 11
S1 (nivel 1) | 13
S2 (nivel 2) | 15
S3 (válvula V1 aberto) | 17
S4 (válvula V1 fechado) | 19
S5 (reservatório) | 21
Tags Saídas | Pino
S1 Esteira A (M1) | 30
S2 Esteira B (M2) | 31
S3 Misturador Horário | 32
S4 Misturador Anti-Hor | 33
S5 Válvula V1 | 35
NOTA : FLAT J2 | 8.6 cabo flat de 40 vias que conecta os terminais da interface IDC-TB40 à FPGA.
NOTA : Todos os condutores desta folha utilizam bitola de 0,75 mm², salvo indicação contrária.
REVISION
A V1
B
C
DATE
18/04/2025
NAME
WDSON
PROJECT
INTERFACE IDC PARA CABO FLAT 40VIAS IDC-TB40
PAGE
7 DE 9


CONECTOR J2 FPGA - IDC40 MACHO - CICLONE IV EP4CE6F17I7
GND 1
B1 3
B3 5
B4 7
B5 9
B6 11
B7 13
B8 15
B9 17
B10 19
B11 21
B12 23
B13 25
D5 27
C6 29
F8 31
D8 33
E9 35
GND 37
3V3 39
2 VCC
4 J16
6 A2
8 A3
10 A4
12 A5
14 A6
16 A7
18 A8
20 A9
22 A10
24 A11
26 A12
28 A13
30 D6
32 E7
34 C8
36 E8
38 GND
40 3V3
FLAT J2 | 7.6
NOTA : FLAT J2 | 8.6 cabo flat de 40 vias que conecta os terminais do interface IDC-TB40 à FPGA.
REVISION
A V1
B
C
DATE
18/04/2025
NAME
WISON
PROJECT
CONECTOR J2 FPGA - IDC40 MACHO
PAGE
8 DE 9

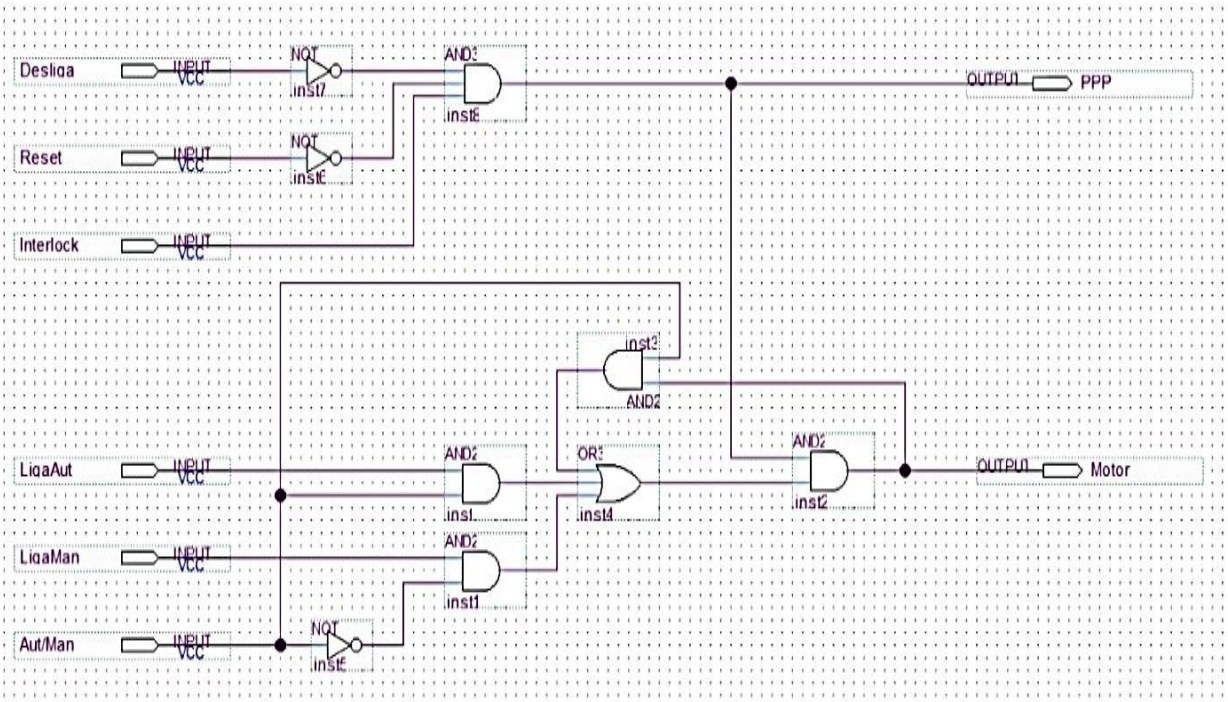

Esteira A
Esteira B
M1
M2
M3
S1
S2
S3
S4
S5
V1
M1 Motor da esteira A (alimentação)
M2 Motor da esteira B (alimentação)
M3 Motor do misturador
V1 Valvula de descarga tanque
S1 Sensor de nivel 1 no misturador
S2 Sensor de nivel 2 no misturador
S3 Sensor de válvula aberta
S4 Sensor de válvula fechada
S5 Sensor de nivel do reservatório final
REVISION
A V1
B
C
DATE
18/04/2025
NAME
WDSON
PROJECT
LAYOUT PLANTA MISTURADOR
PAGE
9 DE 9

APÊNDICE B - MACROCÉLULAS BDF

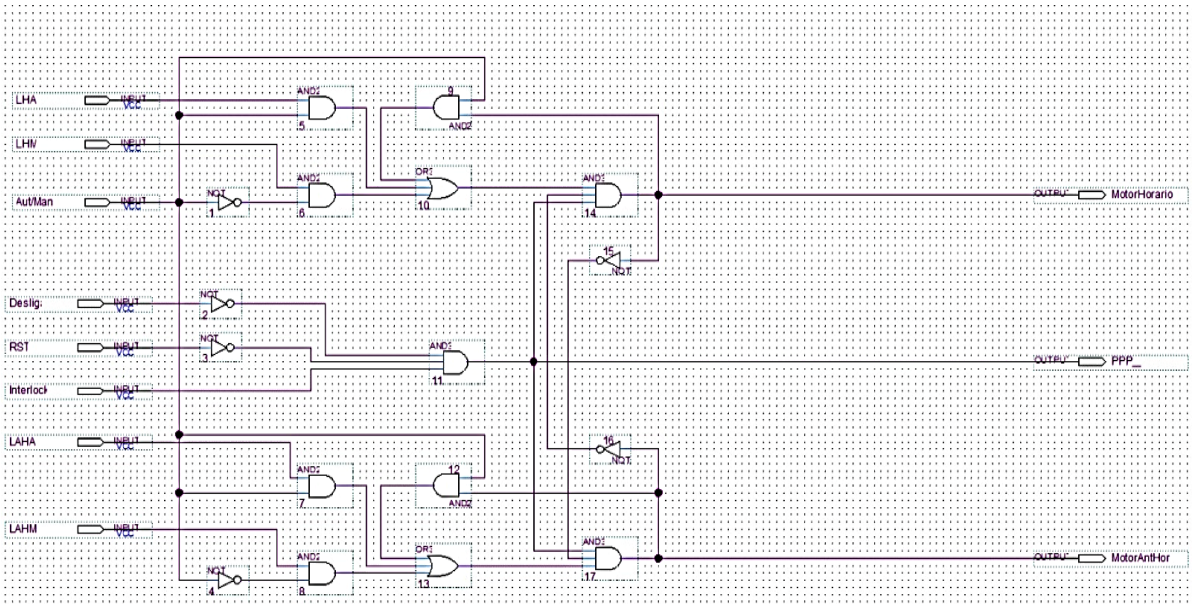
SETRESET



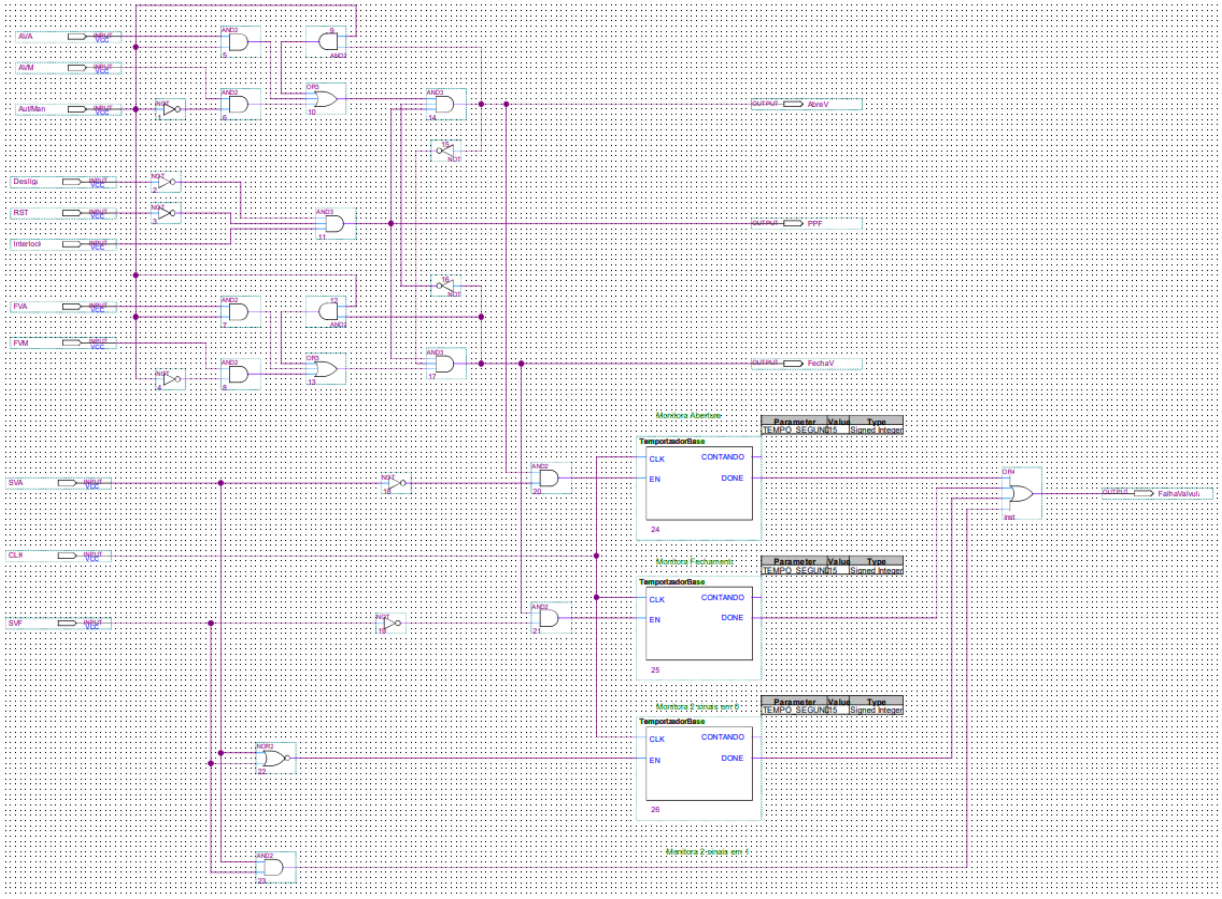
PARTIDA DIRETA



PARTIDA DIRETA COM REVERSÃO



CONTROLE VALVULA



APÊNDICE C - MACROCÉLULAS VHDL

TEMPORIZADORBASE

```
-- ================================================================================ --
-- AUTOR:          WDSON FILIPE MARÇAL                                              --
-- PROPÓSITO:      PROJETO DE TEMPORIZADOR BASE CONFIGURÁVEL VIA GENERIC-       --
--                 DESENVOLVIDO PARA O TRABALHO DE CONCLUSÃO DE CURSO (TCC)     --
--                 EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO                        --
--                                                                              --
-- INSTITUIÇÃO:    INSTITUTO FEDERAL DE MINAS GERAIS (IFMG) – CAMPUS BETIM      --
-- DATA:           ABRIL DE 2025                                                --
--                                                                              --
-- ================================================================================ --

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY TEMPORIZADORBASE IS
   GENERIC (
      TEMPO_SEGUNDOS : INTEGER := 5  -- TEMPO PADRÃO (PODE SER ALTERADO NO BFD)
   );
   PORT (
      CLK     : IN  STD_LOGIC;     -- CLOCK DO SISTEMA (50 MHZ)
      RESET   : IN  STD_LOGIC;     -- RESET SÍNCRONO (ATIVO EM '1')
      ENABLE  : IN  STD_LOGIC;     -- HABILITA A CONTAGEM (ATIVO EM '1')
      DONE    : OUT STD_LOGIC;     -- INDICA QUE O TEMPO FOI ATINGIDO
      CONTANDO  : OUT STD_LOGIC    -- FICA EM '1' ENQUANTO ESTÁ CONTANDO
   );
END TEMPORIZADORBASE;

ARCHITECTURE BEHAVIORAL OF TEMPORIZADORBASE IS

   CONSTANT CLOCK_FREQ : INTEGER := 50000000;  -- FREQUÊNCIA DO CLOCK (50 MHZ PADRÃO)
   CONSTANT LIMITE_COUNT : UNSIGNED(31 DOWNTO 0) := TO_UNSIGNED(TEMPO_SEGUNDOS * CLOCK_FREQ, 32);

   SIGNAL COUNT       : UNSIGNED(31 DOWNTO 0) := (OTHERS => '0');
   SIGNAL DONE_INT    : STD_LOGIC := '0';
   SIGNAL ACTIVE_INT  : STD_LOGIC := '0';

BEGIN

   PROCESS(CLK)
   BEGIN
      IF RISING_EDGE(CLK) THEN
         IF RESET = '1' THEN
            COUNT <= (OTHERS => '0');
            DONE_INT <= '0';
```

```
            ACTIVE_INT <= '0';
         ELSIF ENABLE = '1' THEN
            IF DONE_INT = '0' THEN
               IF COUNT < LIMITE_COUNT THEN
                  COUNT <= COUNT + 1;
                  ACTIVE_INT <= '1';
               ELSE
                  DONE_INT <= '1';
                  ACTIVE_INT <= '0';
               END IF;
            END IF;
         ELSE
            COUNT <= (OTHERS => '0');
            DONE_INT <= '0';
            ACTIVE_INT <= '0';
         END IF;
      END IF;
   END PROCESS;

   DONE   <= DONE_INT;
   CONTANDO <= ACTIVE_INT;

END BEHAVIORAL;
```

DEBOUNCE 15 MS

```
-- ================================================================================ --
-- AUTOR:          WDSON FILIPE MARÇAL                                  --
-- ARQUIVO:        DEBOUNCE15MS.VHD                                      --
-- ENTIDADE:       DEBOUNCE15MS                                        --
-- DESCRIÇÃO:      BLOCO DEBOUNCE  15 MS                                    --
-- ================================================================================ --
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY DEBOUNCE15MS IS
   PORT (
      CLK     : IN  STD_LOGIC;     -- CLOCK DE 50 MHZ
      ENTRADA : IN  STD_LOGIC;     -- ENTRADA RUIDOSA (EX: BOTÃO)
      SAIDA   : OUT STD_LOGIC      -- SAÍDA LIMPA E ESTÁVEL
   );
END DEBOUNCE15MS;

ARCHITECTURE BEHAVIORAL OF DEBOUNCE15MS IS
   CONSTANT LIMITE_CONTAGEM : INTEGER := 750000; -- 15 MS COM CLOCK DE 50 MHZ

   SIGNAL CONTADOR        : INTEGER RANGE 0 TO LIMITE_CONTAGEM := 0;
```

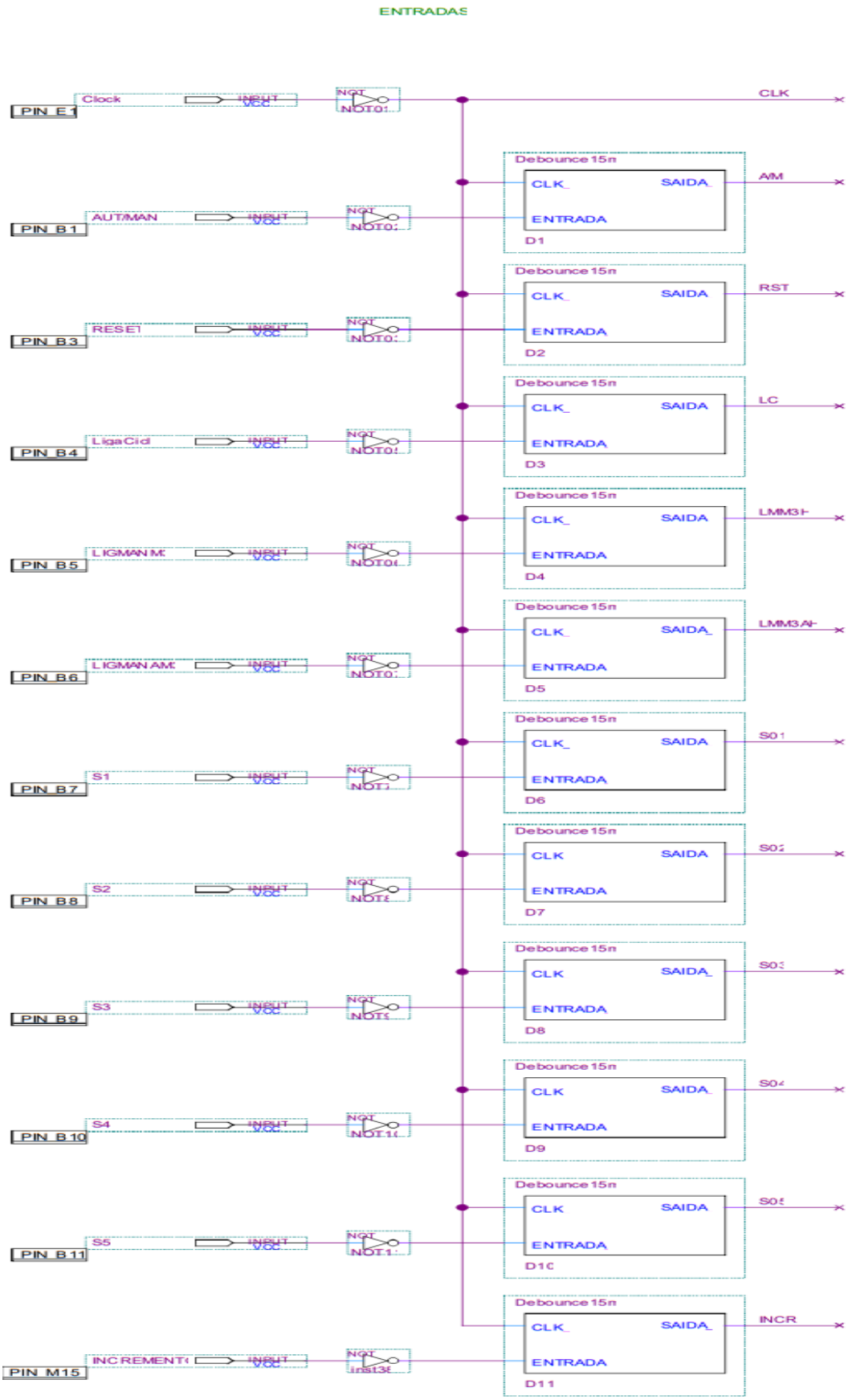
```
    SIGNAL ENTRADA_SYNC1   : STD_LOGIC := '0';
    SIGNAL ENTRADA_SYNC2   : STD_LOGIC := '0';
    SIGNAL ENTRADA_ANTIGA  : STD_LOGIC := '0';
    SIGNAL SAIDA_INT       : STD_LOGIC := '0';
BEGIN

    PROCESS(CLK)
    BEGIN
        IF RISING_EDGE(CLK) THEN
            -- SINCRONIZAÇÃO PARA EVITAR GLITCHES
            ENTRADA_SYNC1 <= ENTRADA;
            ENTRADA_SYNC2 <= ENTRADA_SYNC1;

            -- SE HOUVER MUDANÇA NA ENTRADA, REINICIA O CONTADOR
            IF ENTRADA_SYNC2 /= ENTRADA_ANTIGA THEN
                CONTADOR <= 0;
                ENTRADA_ANTIGA <= ENTRADA_SYNC2;
            ELSIF CONTADOR < LIMITE_CONTAGEM THEN
                CONTADOR <= CONTADOR + 1;
            ELSE
                -- APÓS 15 MS ESTÁVEIS, ATUALIZA A SAÍDA
                SAIDA_INT <= ENTRADA_SYNC2;
            END IF;
        END IF;
    END PROCESS;

    SAIDA <= SAIDA_INT;

END BEHAVIORAL;
```

CONTADOR

```
-- ================================================================================ --
-- AUTOR:          WDSON FILIPE MARÇAL                                              --
-- PROPÓSITO:      PROJETO DE CONTADOR CRESCENTE COM VALOR LIMITE CONFIGURÁVEL       --
--                 VIA GENERIC PARA USO EM TCC DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO --
--                                                                                  --
-- INSTITUIÇÃO:    INSTITUTO FEDERAL DE MINAS GERAIS (IFMG) – CAMPUS BETIM           --
-- DATA:           ABRIL DE 2025                                                    --
-- ================================================================================ --
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY CONTADORBASE IS
    GENERIC (
        LIMITE_CONTAGEM : INTEGER := 10  -- NÚMERO DE PULSOS A CONTAR
    );
```

```
    PORT (
        CLK     : IN  STD_LOGIC;     -- CLOCK DO SISTEMA
        RESET   : IN  STD_LOGIC;     -- RESET SÍNCRONO
        ENABLE  : IN  STD_LOGIC;     -- HABILITA O CONTADOR
        PULSO   : IN  STD_LOGIC;     -- PULSO A SER CONTADO
        DONE    : OUT STD_LOGIC      -- FICA '1' QUANDO ATINGE O LIMITE
    );
END CONTADORBASE;

ARCHITECTURE BEHAVIORAL OF CONTADORBASE IS
    SIGNAL CONTADOR       : UNSIGNED(31 DOWNTO 0) := (OTHERS => '0');
    SIGNAL DONE_INT       : STD_LOGIC := '0';
    SIGNAL PULSO_ANTIGO   : STD_LOGIC := '0';  -- USADO PARA DETECTAR BORDA DE SUBIDA
BEGIN

    PROCESS(CLK)
    BEGIN
        IF RISING_EDGE(CLK) THEN
            IF RESET = '1' THEN
                CONTADOR     <= (OTHERS => '0');
                DONE_INT     <= '0';
                PULSO_ANTIGO <= '0';

            ELSIF ENABLE = '1' THEN
                -- DETECÇÃO DE BORDA DE SUBIDA NO SINAL PULSO
                IF PULSO = '1' AND PULSO_ANTIGO = '0' THEN
                    IF DONE_INT = '0' THEN
                        CONTADOR <= CONTADOR + 1;
                        IF CONTADOR = TO_UNSIGNED(LIMITE_CONTAGEM - 1, 32) THEN
                            DONE_INT <= '1';
                        END IF;
                    END IF;
                END IF;

                -- ATUALIZA O ESTADO ANTERIOR DO PULSO
                PULSO_ANTIGO <= PULSO;

            ELSE
                CONTADOR     <= (OTHERS => '0');
                DONE_INT     <= '0';
                PULSO_ANTIGO <= '0';
            END IF;
        END IF;
    END PROCESS;

    DONE <= DONE_INT;

END BEHAVIORAL
```

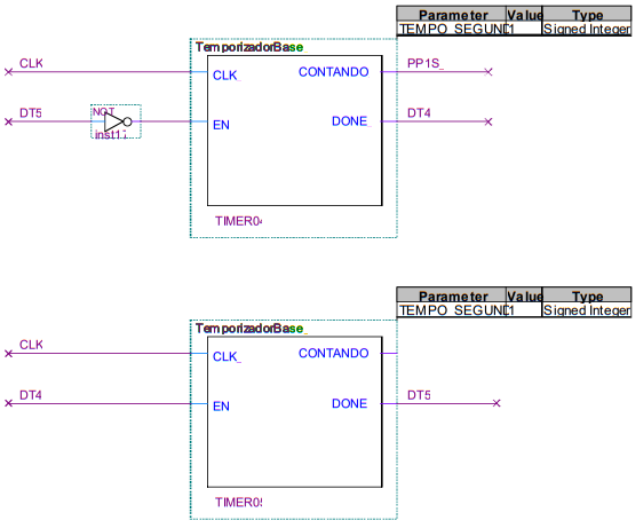
APÊNDICE D - TOP-LEVEL DO PROJETO (PROJETO01.BDF)



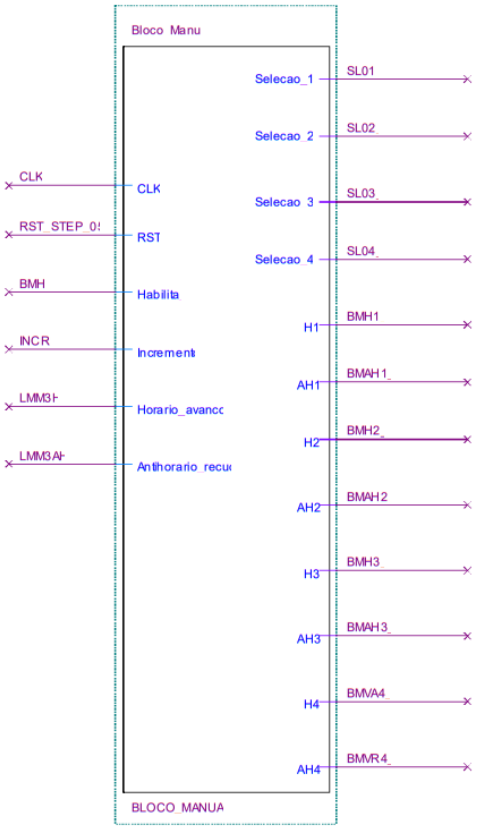
SAIDAS

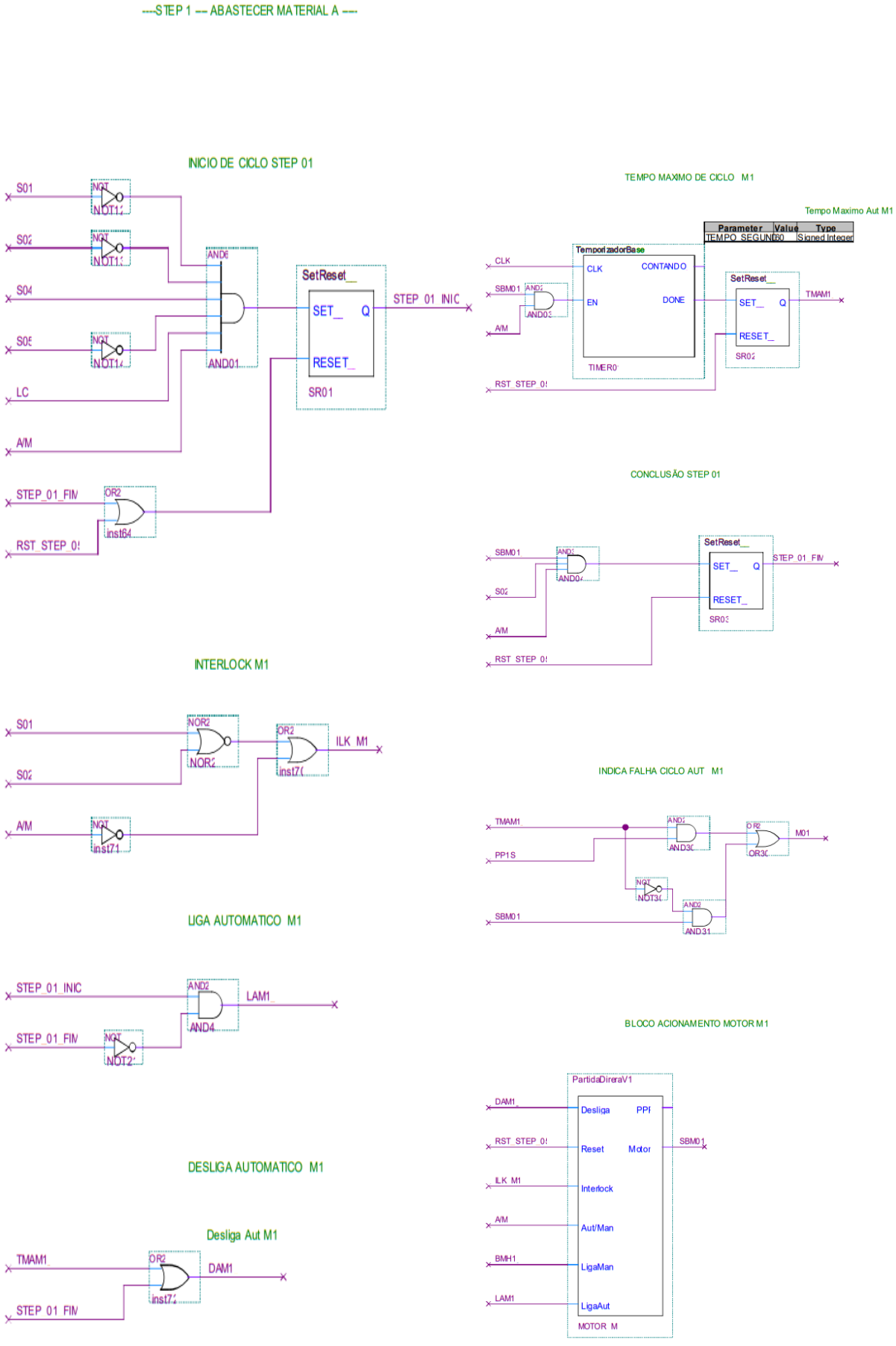

M01
OUTPUT
M1
PIN_D6
M02
OUTPUT
M2
PIN_F8
M031
OUTPUT
M31
PIN_E7
M032
OUTPUT
M32
PIN_D8
V01
OUTPUT
V1
PIN_E9
SL01
OUTPUT
SL1
PIN_E1
SL02
OUTPUT
SL2
PIN_F9
SL03
OUTPUT
SL3
PIN_C9
SL04
OUTPUT
SL4
PIN_D9

PISCA PISCA 1S



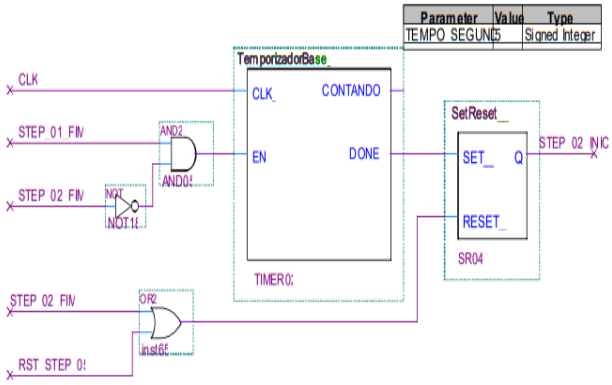
SELECAO MANUA



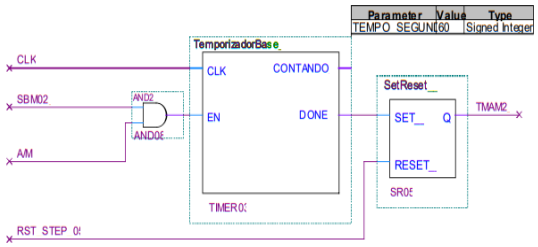

---STEP 1 --- ABASTECER MATERIAL A ---
INICIO DE CICLO STEP 01
S01
S02
S04
S05
LC
A/M
STEP_01_FIN
RST_STEP_01
SetReset
SET_
Q
RESET_
SR01
STEP_01_INIC
TEMPO MAXIMO DE CICLO M1
Tempo Maximo Aut M1
Parameter
Value
Type
TEMPO_SEGUNDO
Signed Integer
TemporizadorBase
CLK
CONTANDO
EN
DONE
TIMER0
SBM01
TMAM1
CONCLUSÃO STEP 01
INTERLOCK M1
ILK_M1
LIGA AUTOMATICO M1
LAM1
DESLIGA AUTOMATICO M1
Desliga Aut M1
DAM1
INDICA FALHA CICLO AUT M1
PP1S
M01
BLOCO ACIONAMENTO MOTOR M1
PartidaDireraV1
Desliga
PPI
Reset
Motor
Interlock
Aut/Man
LigaMan
LigaAut
MOTOR_M
BMH1

---STEP 2 ---- ABASTECER MATERIAL B ----

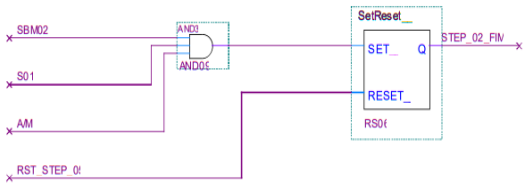
INICIO DE CICLO STEP 2



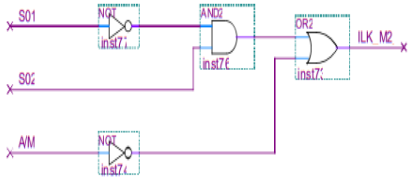
TEMPO MAXIMO DE CICLO



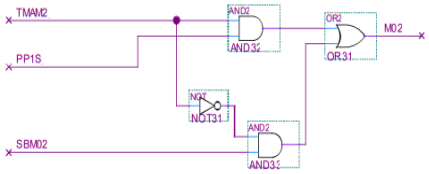
CONCLUSÃO STEP 02



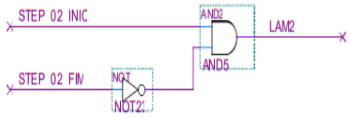
INTERLOCK M2



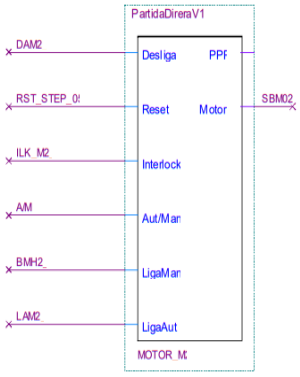
INDICA FALHA CICLO AUT



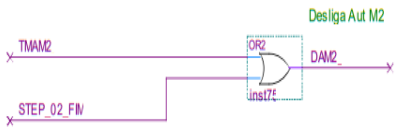
LIGA AUTOMATICO M2

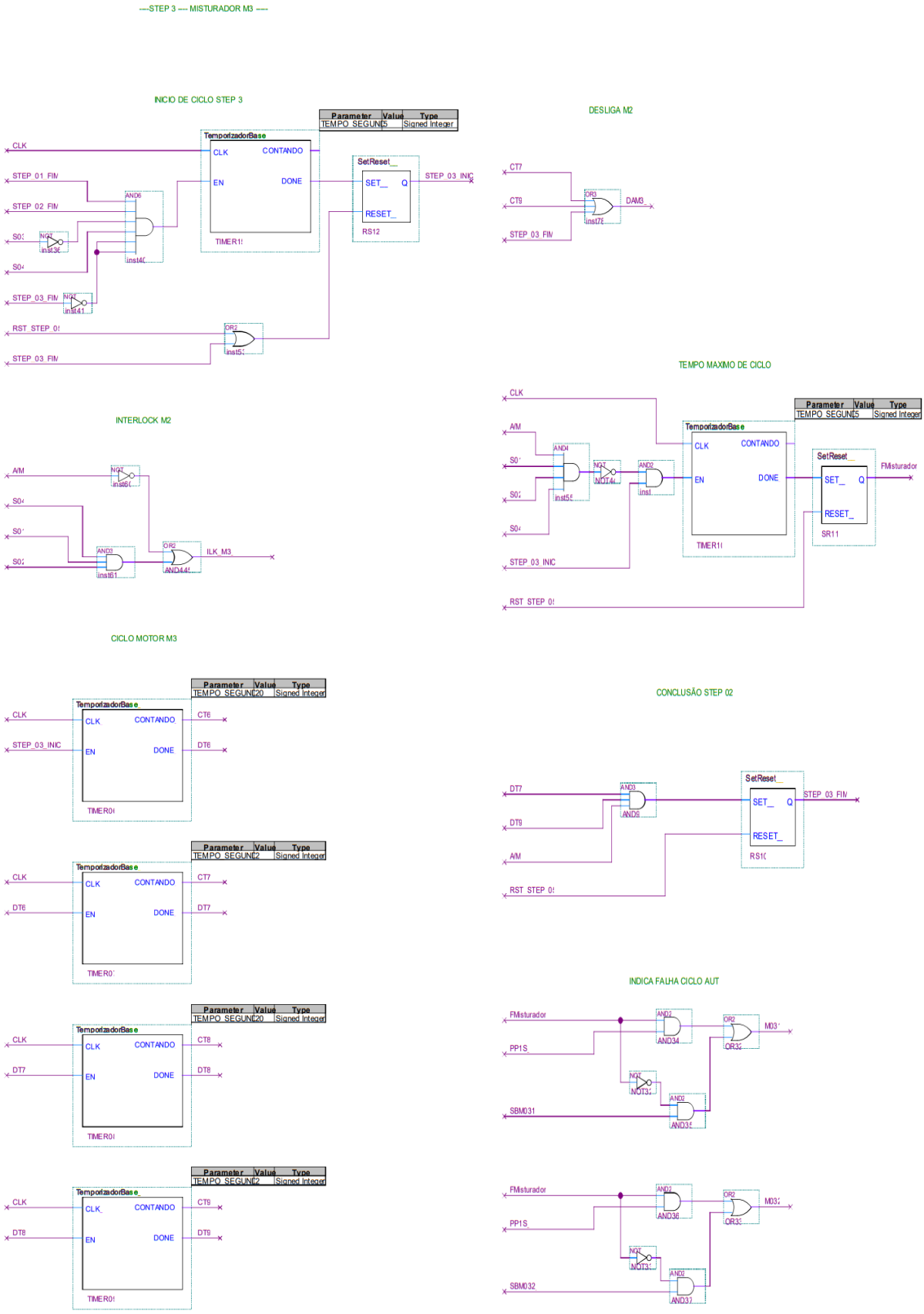


BLOCO ACIONAMENTO MOTOR M2

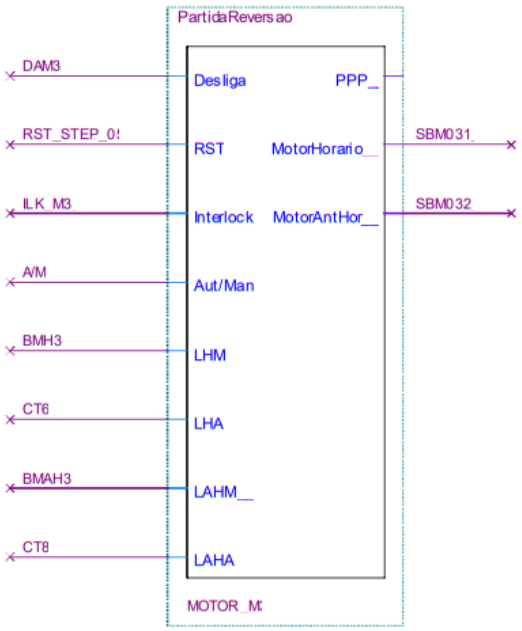


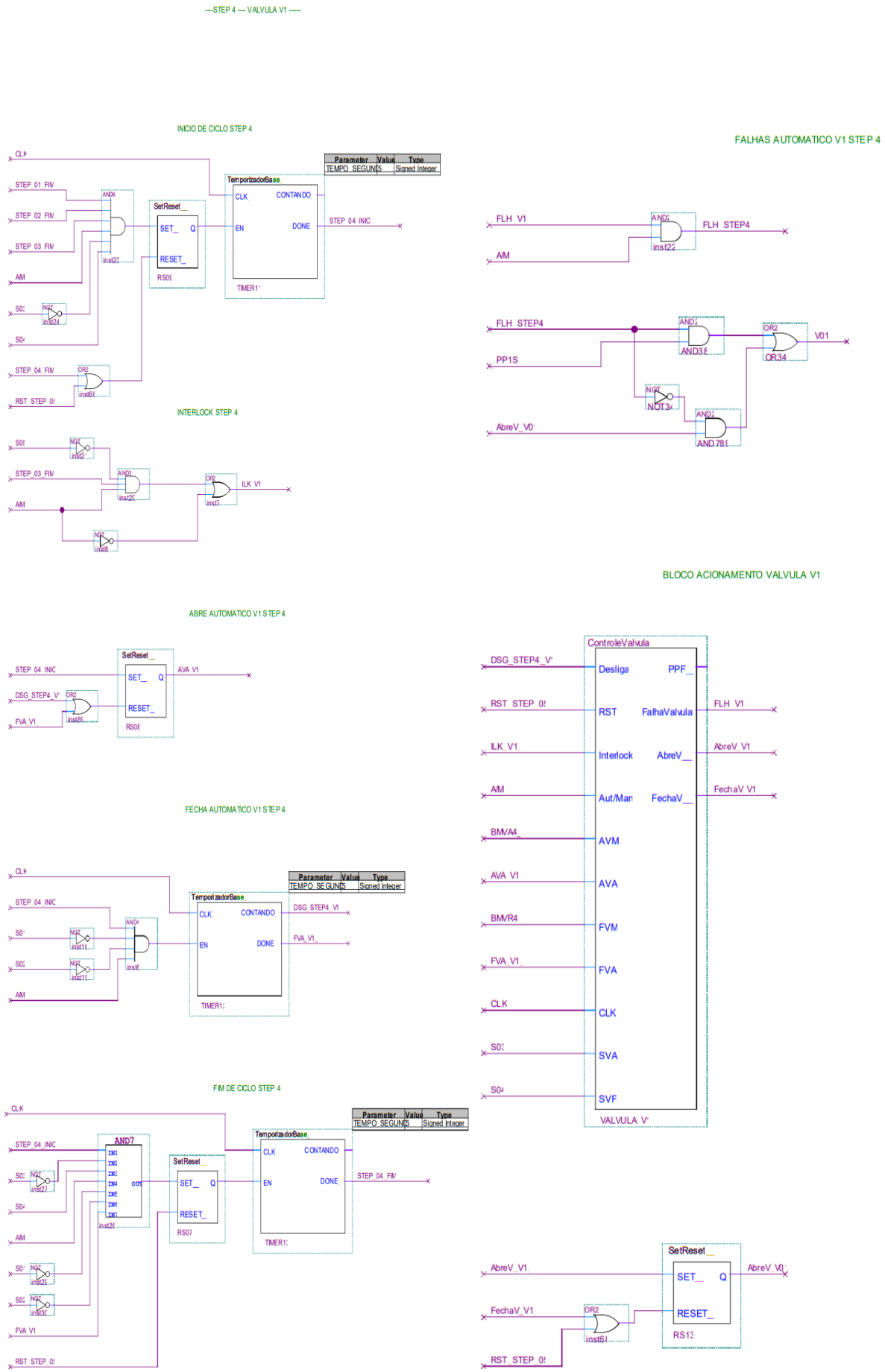
DESLIGA AUTOMATICO M2

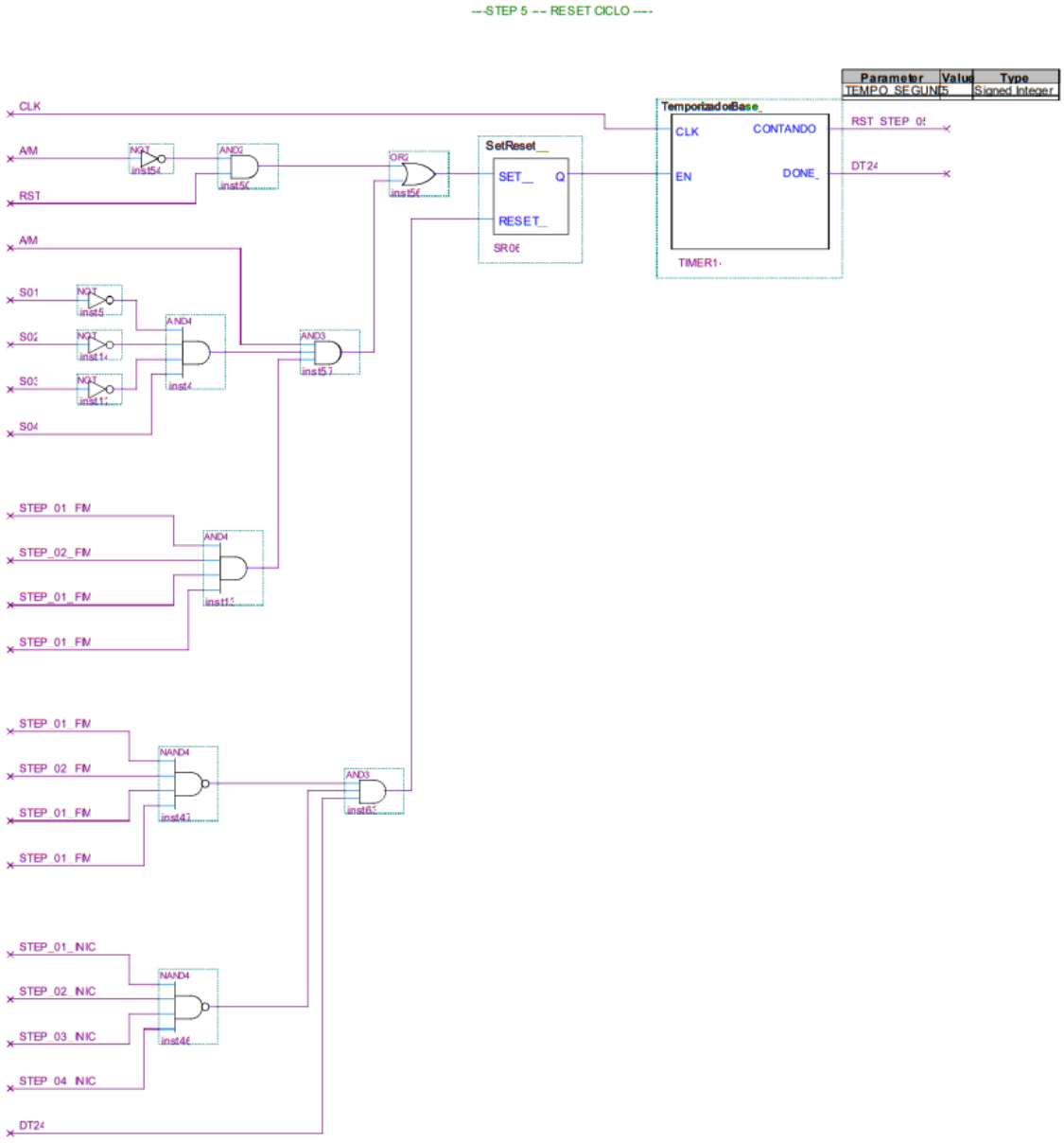



---STEP 3 --- MISTURADOR M3 ---
INICIO DE CICLO STEP 3
Parameter Value Type
TEMPO_SEGUNDOS 5 Signed Integer
TemporizadorBase
CLK
CONTANDO
EN
DONE
TIMER1
SetReset_
SET_
Q
RESET_
RS12
STEP_03_INIC
CLK
STEP_01_FIN
STEP_02_FIN
S0
S0
S0
STEP_03_FIN
RST_STEP_0
STEP_03_FIN
AND6
NOT
OR2
DESLIGA M2
CT7
CT9
STEP_03_FIN
OR3
DAM3
INTERLOCK M2
A/M
S0
S0
S0
AND3
OR2
ILK_M3
TEMPO MAXIMO DE CICLO
CLK
A/M
S0
S0
S0
STEP_03_INIC
RST_STEP_0
AND4
NOT
AND2
SR11
FMisturador
CICLO MOTOR M3
TEMPO_SEGUNDOS 20
TEMPO_SEGUNDOS 2
CT6
DT6
CT7
DT7
CT8
DT8
CT9
DT9
TIMER0
CONCLUSÃO STEP 02
DT7
DT9
A/M
RST_STEP_0
AND8
RS10
STEP_03_FIN
INDICA FALHA CICLO AUT
FMisturador
PP1S_
SBM031
SBM032
AND2
OR2
NOT
M03
M032

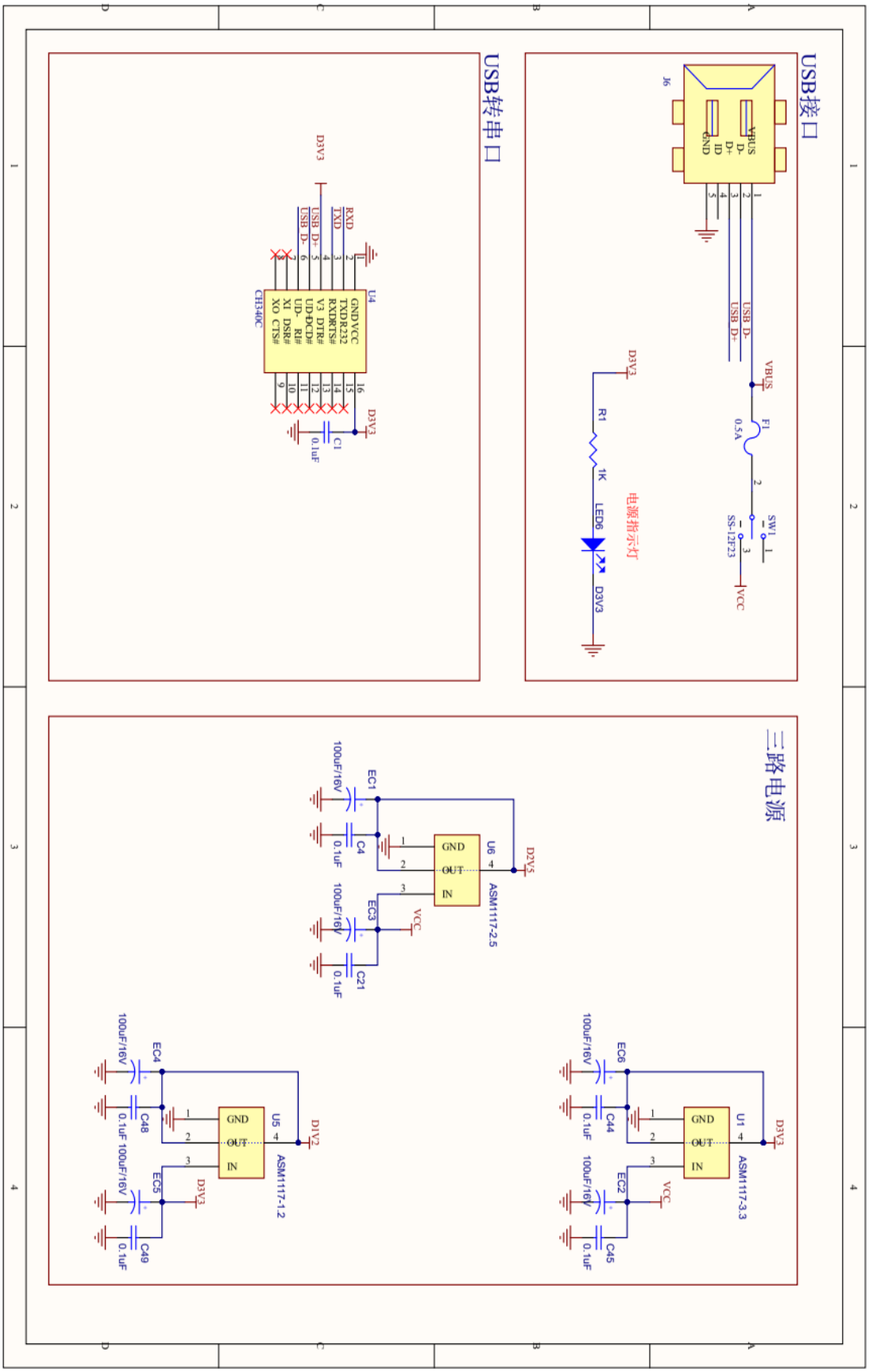
BLOCO ACIONAMENTO MOTOR M3

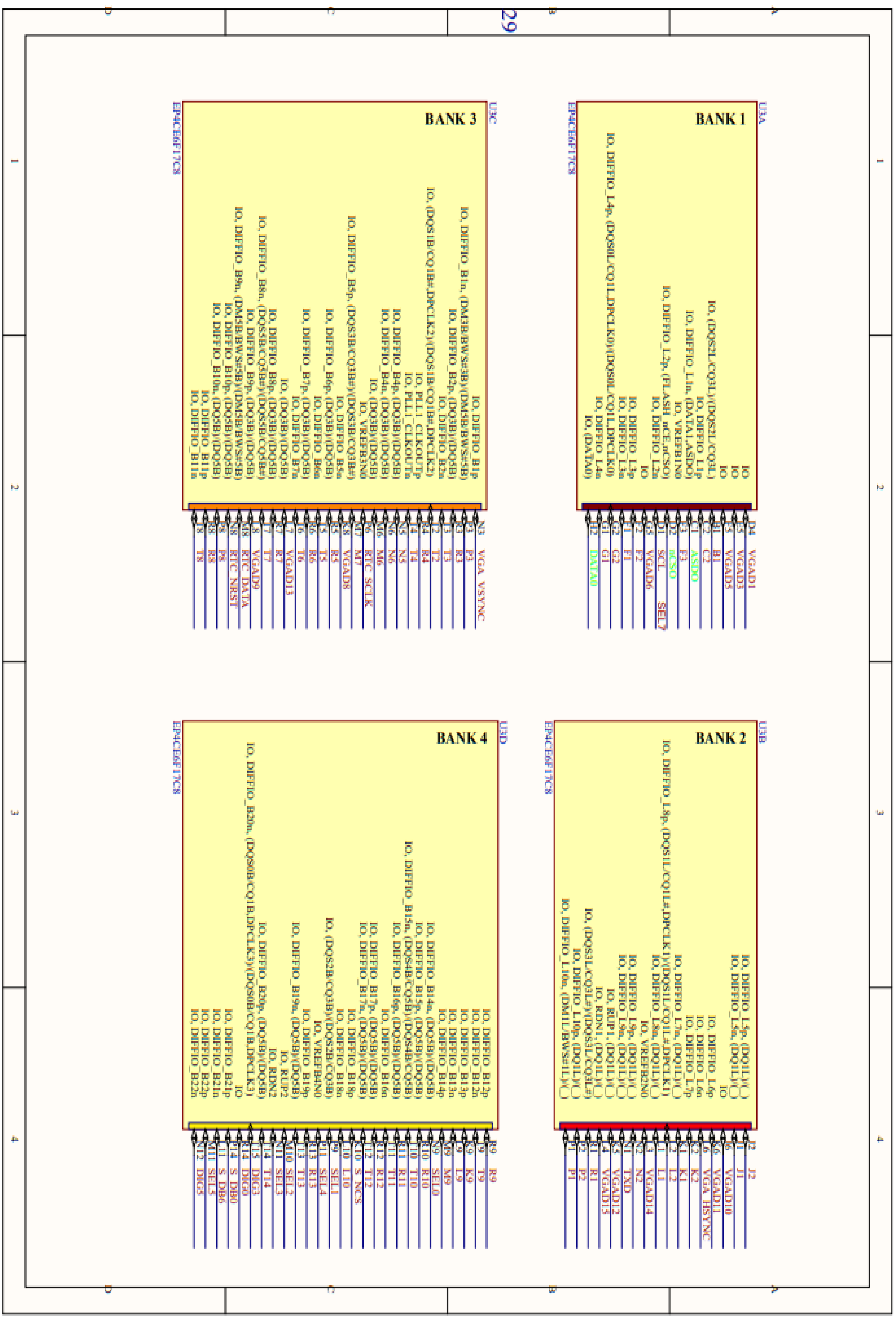


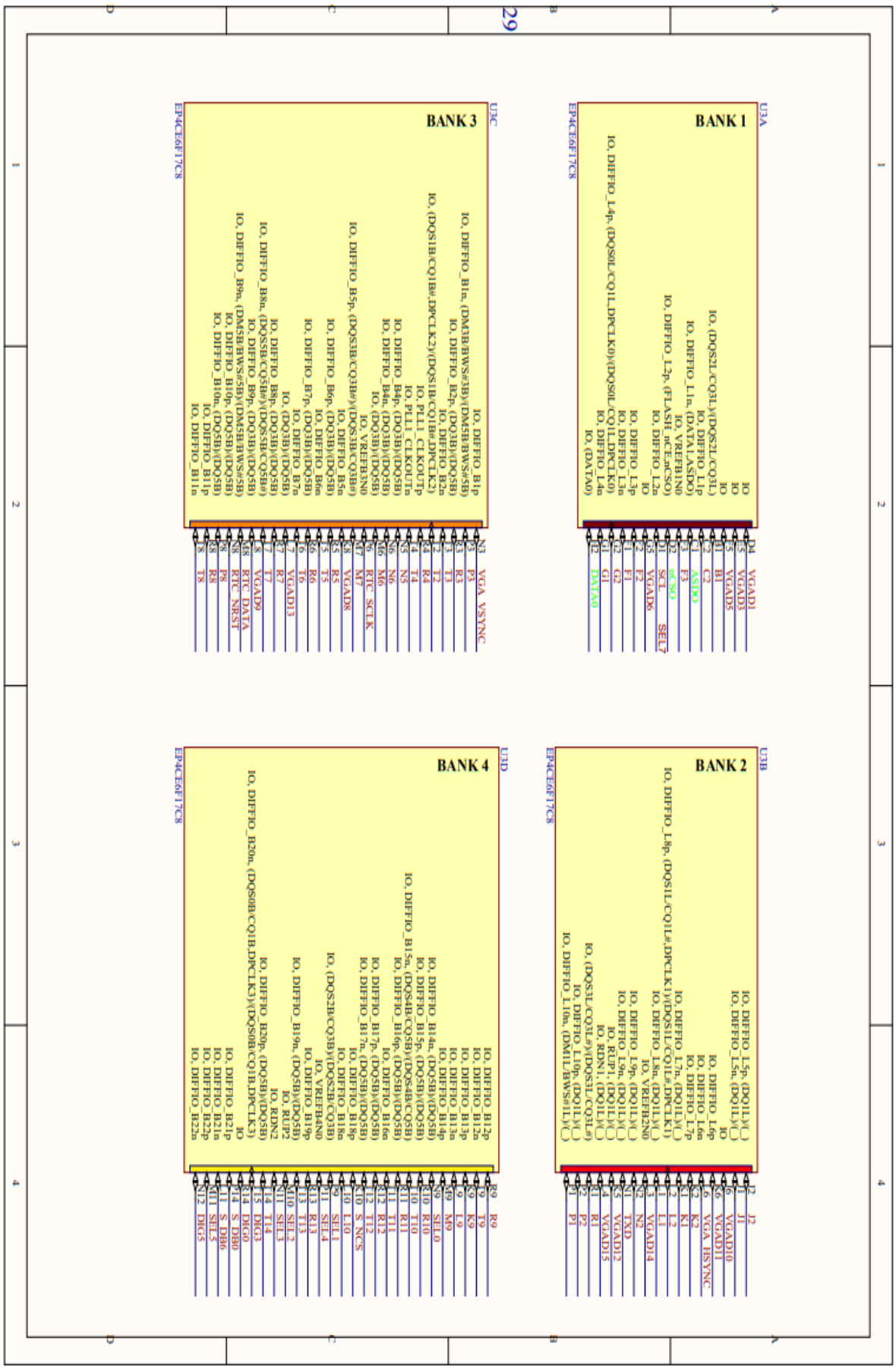

---STEP 4 --- VALVULA V1 -----
INICIO DE CICLO STEP 4
CLK
STEP_01_FIN
STEP_02_FIN
STEP_03_FIN
AM
S0:
S04
STEP_04_FIN
RST_STEP_0!
AND6
inst22
SetReset_
SET_
Q
RESET_
RS0!
TemporizadorBase
CLK
EN
CONTANDO
DONE
TIMER1'
Parameter
Value
Type
TEMPO_SEGUNDS
Signed Integer
STEP_04_INIC
INTERLOCK STEP 4
S0!
STEP_03_FIN
AM
NOT
inst12'
AND3
inst12C
OR2
inst7
ILK_V1
inst8
FALHAS AUTOMATICO V1 STEP 4
FLH_V1
AM
AND2
inst22
FLH_STEP4
PP1S
AND3.8
OR3.4
V01
NOT3.4
AbreV_V0'
AND2.78
BLOCO ACIONAMENTO VALVULA V1
ABRE AUTOMATICO V1 STEP 4
STEP_04_INIC
DSG_STEP4_V'
FVA_V1_
inst16!
AVA_V1
RS0!
FECHA AUTOMATICO V1 STEP 4
DSG_STEP4_V1
FVA_V1_
inst11!
inst11'
inst1!
TIMER1;
ControleValvula
DSG_STEP4_V'
RST_STEP_0!
ILK_V1
AM
BM/A4_
AVA_V1
BM/R4
FVA_V1
CLK
S0;
S04
Desliga
RST
Interlock
Aut/Man
AVM
AVA
FVM
FVA
CLK
SVA
SVF
PPF_
FalhaValvula
AbreV_
FechaV_
FLH_V1
AbreV_V1
FechaV_V1
VALVULA_V'
FIM DE CICLO STEP 4
AND7
inst26
inst27
inst28
inst30
FVA_V1
RS07
TIMER1!
STEP_04_FIM
SetReset_
AbreV_V1
FechaV_V1
RST_STEP_0!
OR2
inst61
RS13
AbreV_V0

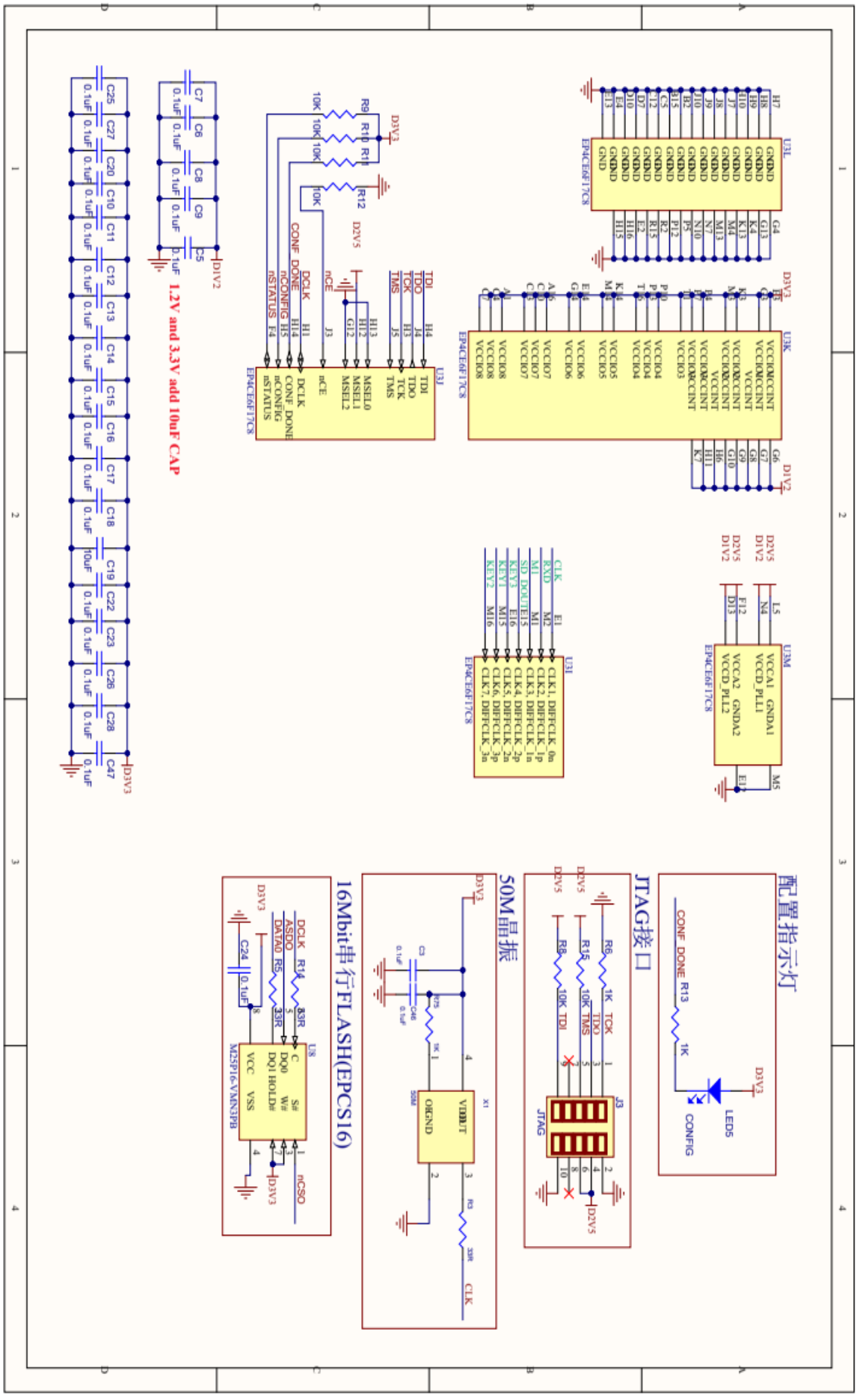

--STEP 5 -- RESET CICLO ----
Parameter
Value
Type
TEMPO_SEGUNDOS
5
Signed Integer
CLK
AIM
RST
S01
S02
S03
S04
STEP_01_FIM
STEP_02_FIM
STEP_01_FIM
STEP_01_FIM
STEP_01_FIM
STEP_02_FIM
STEP_01_FIM
STEP_01_FIM
STEP_01_INIC
STEP_02_INIC
STEP_03_INIC
STEP_04_INIC
DT24
SetReset__
SET__
Q
RESET_
SR0(
TemporizadorBase_
CLK
EN
CONTANDO
DONE_
TIMER1
RST_STEP_0(
DT24

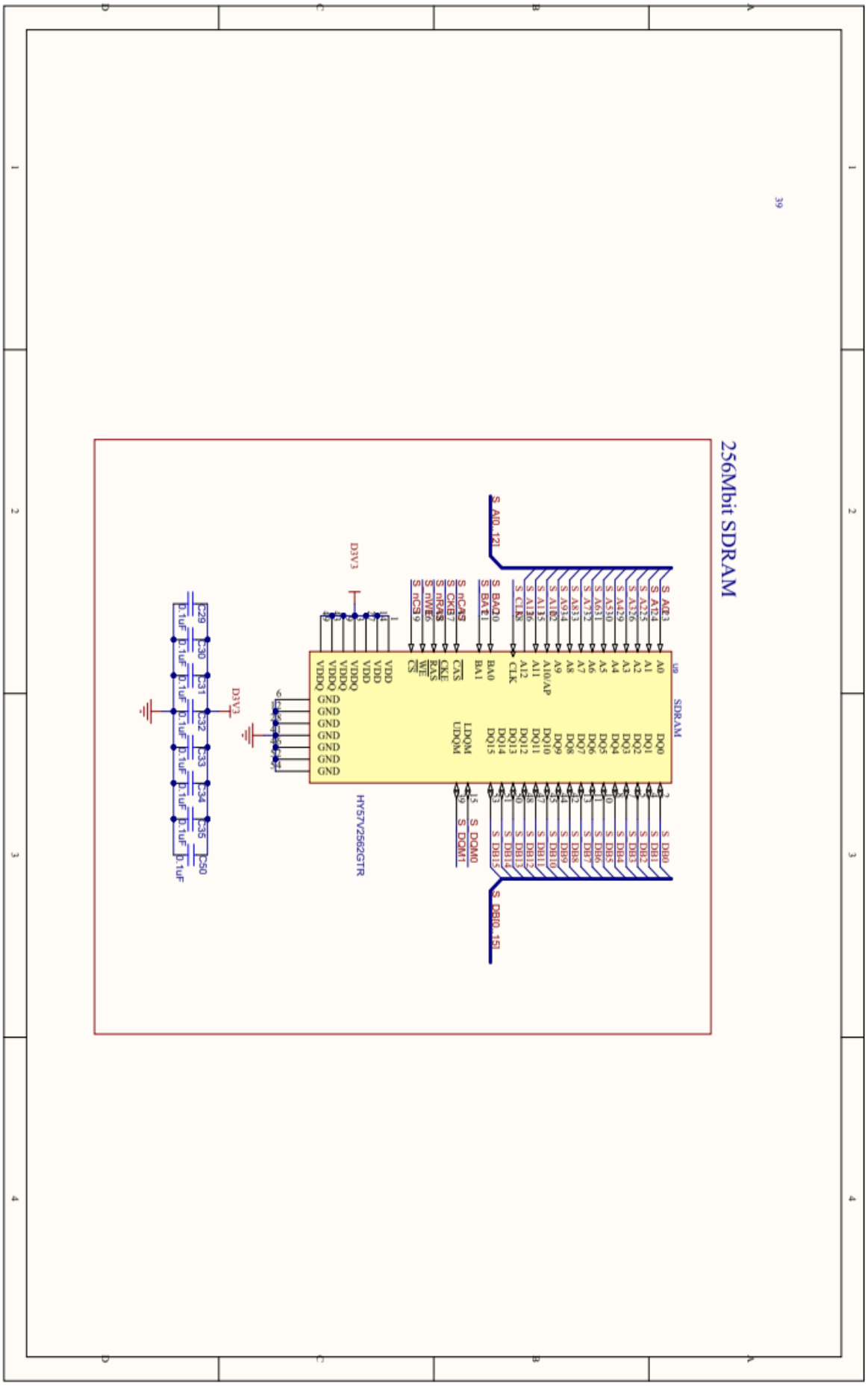
ANEXO A - ESQUEMA DE REFERÊNCIA DA PLACA FPGA

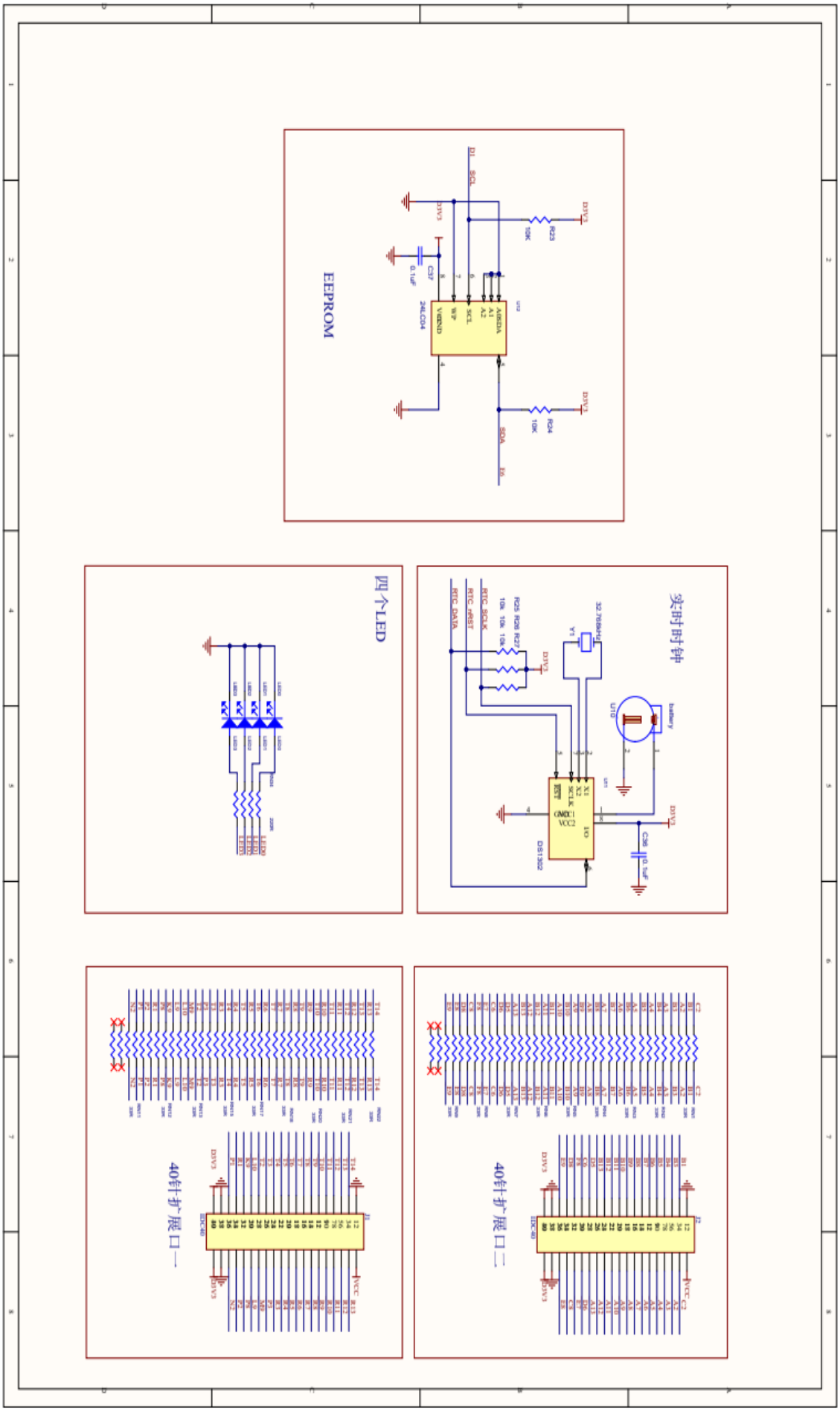


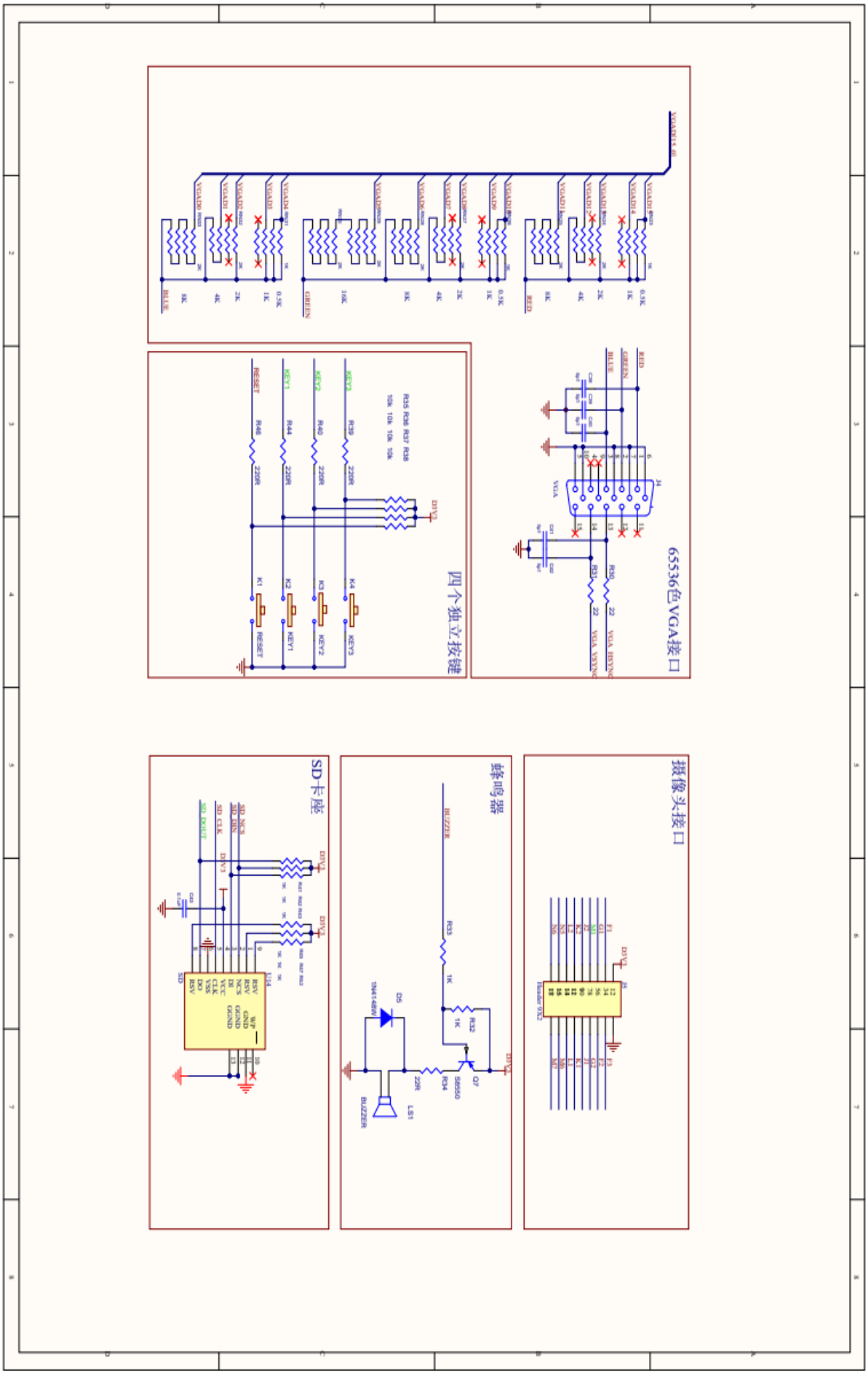

BANK 1
BANK 2
BANK 3
BANK 4

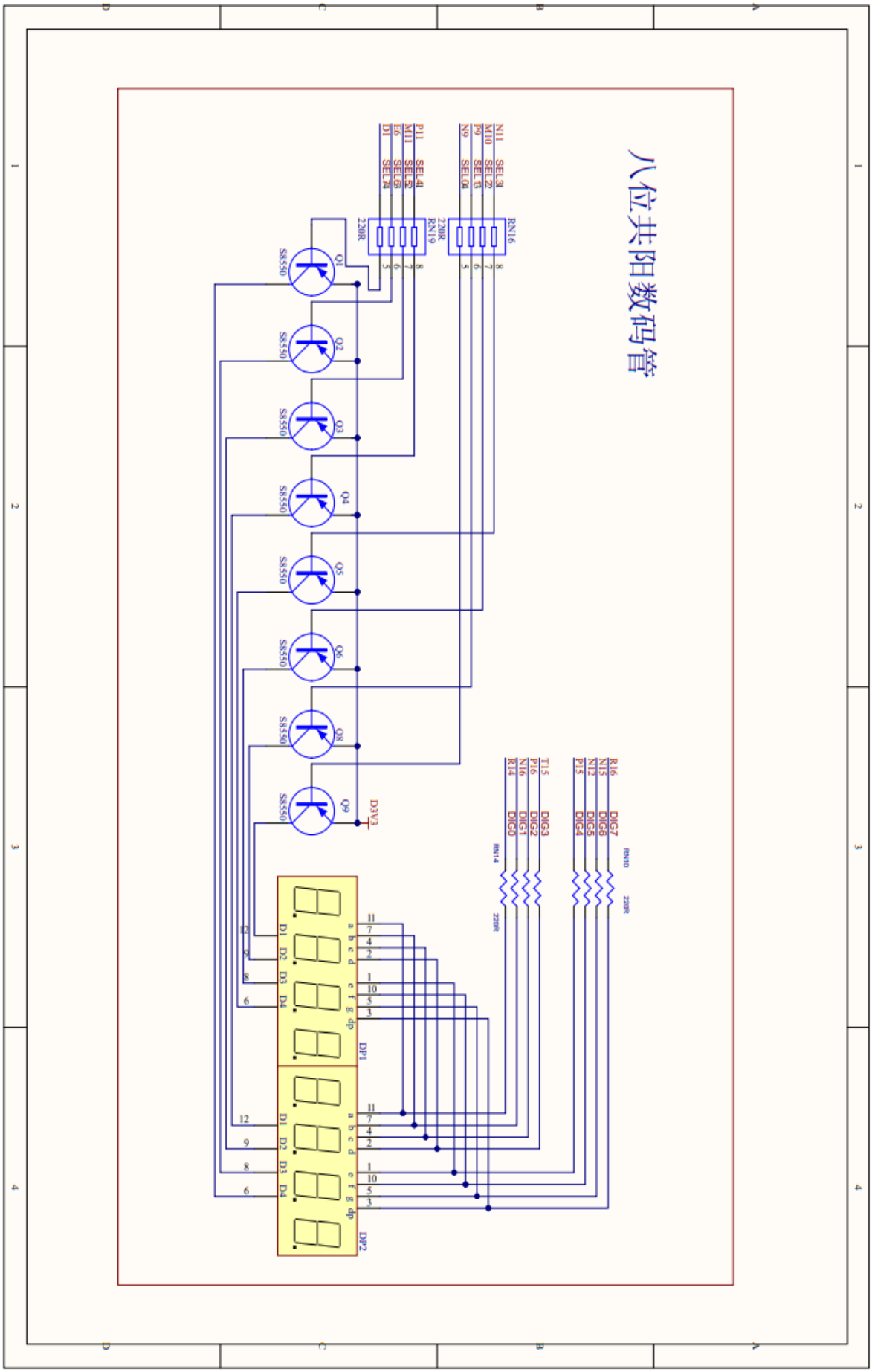

BANK 1
BANK 2
BANK 3
BANK 4
U3A
U3B
U3C
U3D
EP4CE6F17C8
IO, DIFFIO_L1n, (DATA1/ASDO)
IO, DIFFIO_L1p
IO, DIFFIO_L2n
IO, DIFFIO_L2p, (FLASH_nCE/nCSO)
IO, DIFFIO_L3n
IO, DIFFIO_L3p
IO, DIFFIO_L4n
IO, DIFFIO_L4p, (DQS0L/CQ1L,DPCLK0)
IO, DIFFIO_L5n, (DQ1L/nCEO)
IO, DIFFIO_L5p, (DQ1L/CQ1L#)
IO, DIFFIO_L6n
IO, DIFFIO_L6p
IO, DIFFIO_L7n, (DQ1L/CQ1L#,DPCLK1)
IO, DIFFIO_L7p
IO, DIFFIO_L8n, (DQ1L/CQ1L#)
IO, DIFFIO_L8p, (DQS1L/CQ1L#,DPCLK1)
IO, DIFFIO_L9n, (DQ1L/CQ1L#)
IO, DIFFIO_L9p, (DQ1L/CQ1L#)
IO, DIFFIO_L10n, (DM1L/BWS#1L/CQ1L#)
IO, DIFFIO_L10p, (DQ1L/CQ1L#)
IO, RUP1, (DQ1L/CQ1L#)
IO, RDN1, (DQ1L/CQ1L#)
IO, (DQS3L/CQ3L#/DQS3L/CQ3L#)
IO, VREFB2N0
IO, DIFFIO_B1p
IO, DIFFIO_B1n, (DM3B/BWS#3B)(DM5B/BWS#5B)
IO, DIFFIO_B2p, (DQ3B)(DQ5B)
IO, DIFFIO_B2n
IO, (DQS1B/CQ1B#,DPCLK2)(DQS3B/CQ3B#,DPCLK2)
IO, PLL1_CLKOUTp
IO, PLL1_CLKOUTn
IO, DIFFIO_B4p, (DQ3B)(DQ5B)
IO, DIFFIO_B4n, (DQ3B)(DQ5B)
IO, (DQ3B)(DQ5B)
IO, VREFB3N0
IO, DIFFIO_B5p, (DQS3B/CQ3B#)(DQS3B/CQ3B#)
IO, DIFFIO_B5n
IO, DIFFIO_B6p, (DQ3B)(DQ5B)
IO, DIFFIO_B6n
IO, DIFFIO_B7p, (DQ3B)(DQ5B)
IO, DIFFIO_B7n
IO, (DQ3B)(DQ5B)
IO, DIFFIO_B8p, (DQ3B)(DQ5B)
IO, DIFFIO_B8n, (DQS3B/CQ3B#)(DQS5B/CQ5B#)
IO, DIFFIO_B9p, (DQ3B)(DQ5B)
IO, DIFFIO_B9n, (DM5B/BWS#5B)(DM5B/BWS#5B)
IO, DIFFIO_B10p, (DQ5B)(DQ5B)
IO, DIFFIO_B10n, (DQ5B)(DQ5B)
IO, DIFFIO_B11p
IO, DIFFIO_B11n
IO, DIFFIO_B12p
IO, DIFFIO_B12n
IO, DIFFIO_B13p
IO, DIFFIO_B13n
IO, DIFFIO_B14p
IO, DIFFIO_B14n, (DQ5B)(DQ5B)
IO, DIFFIO_B15p, (DQ5B)(DQ5B)
IO, DIFFIO_B15n, (DQS4B/CQ5B)(DQS4B/CQ5B)
IO, DIFFIO_B16p, (DQ5B)(DQ5B)
IO, DIFFIO_B16n
IO, DIFFIO_B17p, (DQ5B)(DQ5B)
IO, DIFFIO_B17n, (DQ5B)(DQ5B)
IO, DIFFIO_B18p
IO, DIFFIO_B18n
IO, (DQS2B/CQ3B)(DQS2B/CQ3B)
IO, VREFB4N0
IO, DIFFIO_B19p
IO, DIFFIO_B19n, (DQ5B)(DQ5B)
IO, RUP2
IO, RDN2
IO, DIFFIO_B20p, (DQ5B)(DQ5B)
IO, DIFFIO_B20n, (DQS0B/CQ1B,DPCLK3)(DQS0B/CQ1B,DPCLK3)
IO
IO, DIFFIO_B21p
IO, DIFFIO_B21n
IO, DIFFIO_B22p
IO, DIFFIO_B22n
VGAD1
VGAD3
VGAD5
VGAD7
ASDO
nCSO
SCL
SEL7
VGAD6
DATA0
VGA_VSYNC
RTC_SCLK
VGAD8
VGAD9
RTC_DATA
RTC_NRST
VGAD13
VGAD4
VGAD2
VGAD0
VGA_HSYNC
VGAD11
VGAD10
VGAD14
VGAD12
VGAD15
SEL0
SEL1
SEL4
SEL2
SEL3
SEL5
DIG5
DIG4
DIG3
DIG2
DIG1
DIG0
S_DIG0
S_DIG1
SEL6
TXD
S_NCS
R9
T9
K9
L9
M9
T10
R10
L10
R11
T11
R12
T12
R13
T13
T14
R14
P14
P13
P12
P11
J2
J1
K1
K2
L1
L2
M1
N2
P2
P1
29

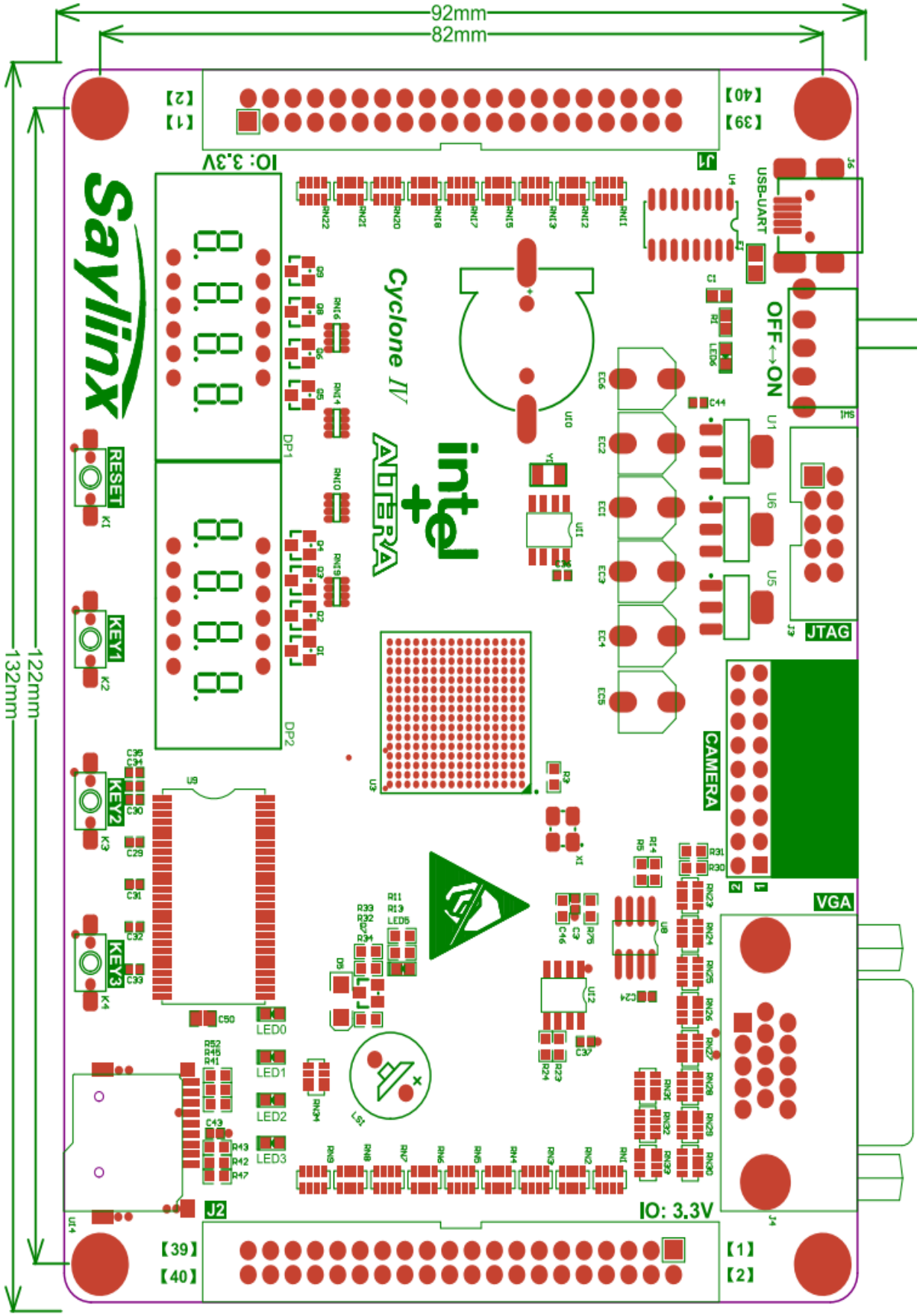

配置指示灯
CONF_DONE
R13
1K
D3V3
LED5
CONFIG
JTAG接口
R6
1K
TCK
R15
10K
TMS
R8
10K
TDI
TDO
J3
JTAG
D2V5
50M晶振
D3V3
C3
0.1uF
C46
0.1uF
R25
1K
50M
X1
VDD
OUT
GND
R3
33R
CLK
16Mbit串行FLASH(EPCS16)
DCLK
R14
33R
ASDO
R5
33R
DATA0
C24
0.1uF
U8
C
S#
DQ0
W#
DQ1
HOLD#
VCC
VSS
M25P16-VMN3PB
nCSO
D3V3
1.2V and 3.3V add 10uF CAP
D1V2
D3V3
C7
C6
C8
C9
C5
0.1uF
R9
R10
R11
R12
10K
D2V5
nCE
CONF_DONE
nCONFIG
nSTATUS
DCLK
U3J
TDI
TDO
TCK
TMS
MSEL0
MSEL1
MSEL2
EP4CE6F17C8
U3K
U3L
U3M
U3I
CLK
RXD
M1
SD_DOUT
KEY3
KEY1
KEY2
CLK1, DIFFCLK_0n
CLK2, DIFFCLK_1p
CLK3, DIFFCLK_1n
CLK4, DIFFCLK_2p
CLK5, DIFFCLK_2n
CLK6, DIFFCLK_3p
CLK7, DIFFCLK_3n
C25
C27
C20
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C22
C23
C26
C28
C47
10uF

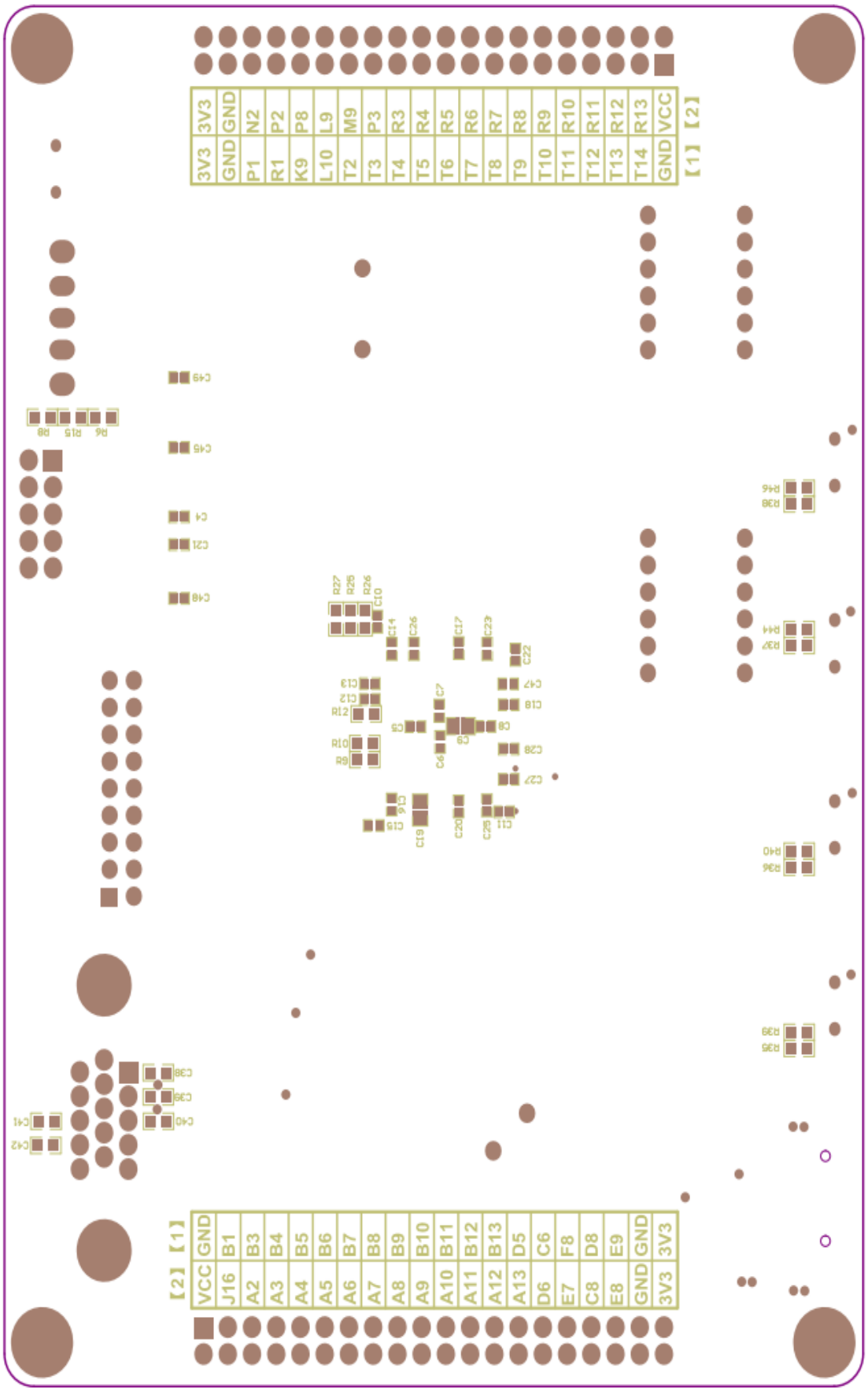

256Mbit SDRAM
SDRAM
HY57V2562GTR
D3V3
S_A[0..12]
S_DB[0..15]
S_DQM0
S_DQM1
S_CLKS
S_CKE
S_nCAS
S_nRAS
S_nWE
S_nCS
S_BA0
S_BA1
C29
C30
C31
C32
C33
C34
C35
C50
0.1uF


EEPROM
四个LED
实时时钟
40针扩展口二
40针扩展口一


65536色VGA接口
四个独立按键
蜂鸣器
SD卡座
摄像头接口


八位共阳数码管
SEL1
SEL2
SEL3
SEL0
SEL4
SEL5
SEL6
SEL7
220R
RN16
RN19
Q1
Q2
Q3
Q4
Q5
Q6
Q8
Q9
S8550
D3V3
DIG0
DIG1
DIG2
DIG3
DIG4
DIG5
DIG6
DIG7
RN14
RN10
DP1
DP2
a b c d e f g dp
D1 D2 D3 D4


92mm
82mm
132mm
122mm
Saylinx
Cyclone IV
intel
ALTERA
IO: 3.3V
IO: 3.3V
USB-UART
OFF↔ON
JTAG
CAMERA
VGA
RESET
KEY1
KEY2
KEY3
LED0
LED1
LED2
LED3
J1
J2
J3
J4
J5
J6
DP1
DP2
【1】
【2】
【39】
【40】


【1】 【2】
3V3 GND N2 P2 P8 L9 M9 P3 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 VCC
3V3 GND P1 R1 K9 L10 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13 T14 GND
【2】 【1】
VCC J16 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 D6 E7 C8 E8 GND 3V3
GND B1 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 D5 C6 F8 D8 E9 GND 3V3