Mestrado em Engenharia Eletrotécnica

Um Controlador Programável Baseado em CoDeSys – Estágio na Bresimar Automação

Relatório de Estágio apresentado para a obtenção do grau de Mestre em Engenharia Eletrotécnica – Área de Especialização em Automação e Comunicações em Sistemas Industriais

Autor
Rui Filipe Pedrosa Silva

Orientador
Doutor Fernando José Pimentel Lopes
Professor do Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia de Coimbra

Supervisor
Engenheiro Ricardo José de Almeida Carvalho
Bresimar Automação S.A.

Coimbra, abril, 2017

AGRADECIMENTOS

Tudo o que fazemos é fruto do nosso esforço, mas não só. Acredito que tudo o que fazemos tem uma grande influência de todas as pessoas que nos rodeiam e com as quais nos relacionamos, e nos ajudam a ser a pessoa e o indivíduo que a sociedade vê.

Gostava por isso de expressar aqui os meus especiais agradecimentos:

À minha querida esposa, pela ajuda, apoio, paciência e as suas sempre presentes palavras de incentivo.

Aos meus pais e irmã, pela confiança demonstrada e por todo o esforço que fizeram para me proporcionarem o percurso académico do qual pude usufruir.

Ao meu orientador, Doutor Fernando José Pimentel Lopes, pelo apoio e insistência demonstrada e ainda pela orientação na realização do estágio e durante o processo de escrita do relatório.

O meu especial agradecimento ao Grupo Bresimar, por todas as condições que me foram proporcionadas, em especial ao meu Supervisor, Engenheiro Ricardo Carvalho, e a toda a equipa da TekOn, pelo apoio e encorajamento sempre disponível.

Seria também impossível não mencionar todos os meus amigos que fui fazendo durante todo este percurso, antes e durante o meu percurso académico.

A todos, o meu muito obrigado

RESUMO

Nos equipamentos de processo e controlo industrial, os controladores programáveis foram até à presente data um fator preponderante, e estima-se que assim continuem nos próximos anos. No entanto, as necessidades de evolução e de adaptação às mais recentes tendências tecnológicas neste sector, têm contribuído para que um mercado tradicionalmente mais conservador como este, tenha nos últimos anos tido uma cada vez maior evolução tecnológica. São tendências e conceitos como o da Indústria 4.0 que vêm trazer uma mudança significativa nas tradicionais linhas de produção e prometem uma cada vez maior customização, mudanças de *layout* constantes e um contacto cada vez maior com o consumidor final.

Este trabalho desenvolvido na Bresimar Automação S.A., teve como principal objetivo permitir estudar as plataformas e tecnologias existentes no mercado e que vão no sentido de permitir aos seus utilizadores uma cada vez maior padronização nas ferramentas de programação. Uma dessas ferramentas é comercializada pela empresa 3S-Smart, e chama-se CodeSys.

Assim, este estágio teve como objetivo específico estudar a solução CodeSys e os seus requisitos de implementação. Neste relatório percorrem-se detalhadamente todas as etapas necessárias para o desenvolvimento deste tipo de solução, desde a pesquisa, implementação e teste, passando pelo desenho de *hardware* (HW) de um protótipo, e ainda pela identificação dos passos necessários para obter a certificação CE.

Palavras-chave: Sistemas Industriais, IEC61131-3, CodeSys, Controladores Programáveis

ABSTRACT

In industrial process and control equipment, Programmable Controllers or PLCs, have, until now, played a preponderant role, and it is estimated that this will continue in the years to come.

However, the need for evolution and adaptation to the latest technological trends in this area, has contributed to the fact that, a traditionally more conservative market such as this one has had, in the last years, a very strong increase in technological evolution. Trends and concepts such as Industry 4.0, bring significant changes to traditional production lines, promising ever-greater customization, constant layout changes and increasing contact with the end consumer.

This work developed at Bresimar Automação S.A., had as its main objective to study the platforms and technologies existing in the market, which allow its users an increasing standardization in the programming tools. One of these tools is marketed by the company 3S-Smart, and is called CodeSys.

This internship aimed specifically to study the CodeSys solution and its implementation requirements.

This report goes in great detail through all the necessary steps for the development of this type of solution, from the research, implementation and testing, to the hardware design of a complete prototype, including the identification of the necessary steps to obtain the CE Marking.

Keywords: Industrial Systems, IEC61131-3, CodeSys, Programmable Logic Controller

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIMBOLOGIA

ampere (símbolo: **A) –** É a unidade de intensidade de corrente elétrica do SI

bit **(**símbolo: **b) –** É a simplificação para um dígito binário (unidade de informação)

byte **(**símbolo: **B) –** É a codificação padronizada que foi definida como sendo 8 bit

hertz (símbolo: **Hz) –** É a unidade de medida derivada do SI para frequência

kHz – Múltiplo do SI para Hz e representa 10^3 Hz

metro (símbolo: **m) –** É a unidade medida de comprimento do SI

miliampere (símbolo: **mA) –** Múltiplo do SI para Ampere e representa 10^{-3} A

milímetro (símbolo: **mm) –** Múltiplo do SI para metro e representa 10^{-3} m

milissegundo (símbolo: **ms) –** Múltiplo do SI para segundos e representa 10^{-3} s

segundo (símbolo: **s) –** É a unidade de medida do tempo de SI

volt (símbolo: **V) –** É a unidade de tensão elétrica do SI

ABREVIATURAS

ASA – Aplicações e Sistemas de Automação

BGA – do Inglês, *Ball Grid Array*

BOM – do Inglês, *Bill of Materials*

CEM – Compatibilidade EletroMagnética

CPU – do Inglês, *Central Processing Unit*

DIP – do Inglês, *Dual In-line Package*

EDA – do Inglês, *Electronic Design Automation*

EMC – do Inglês, *Electromagnetic Compatibility*

EMI – do Inglês, *Electromagnetic Interference*

EUT – do Inglês, *Equipment Under Test*

F-RAM – do Inglês, *Ferroelectric* RAM

HMI – do Inglês, *Human Machine Interface*

HW – do Inglês, *Hardware*

I2C – do Inglês, *Inter-Integrated Circuit*

IC – do Inglês, *Integrated Circuit*

IDE – do Inglês, *Integrated Development Environment*

IEC – do Inglês, *International Electrotechnical Commission*

IO – do Inglês, *Inputs and Outputs*

IoT – do Inglês, *Internet of Things*

OS – do Inglês, *Operating System*

PCB – do Inglês, *Printed Circuit Board*

PLC – do Inglês, *Programmable Logic Controller*

POU – do Inglês, *Program Organization Units*

RAM – do Inglês, *Random Access Memory*

RT – do Inglês, *Run Time*

RTS – do Inglês, *Run Time System*

SPI – do Inglês, *Serial Peripheral Interface*

THT – do Inglês, *Through Hole Technology*

TSP – do Inglês, *Target Support Package*

USB – do Inglês, *Universal Serial Bus*

1 - INTRODUÇÃO

Este capítulo pretende apresentar uma introdução ao presente relatório de estágio na empresa Bresimar Automação S. A., expondo um resumo sobre a empresa, os objetivos do trabalho e a organização deste documento.

1.1 - Sobre a Bresimar Automação S.A.

A Bresimar Automação S.A. encontra-se sediada em Aveiro e conta também com uma delegação em Lisboa localizada no TagusPark.

Figura 1 - Localização Bresimar [1]

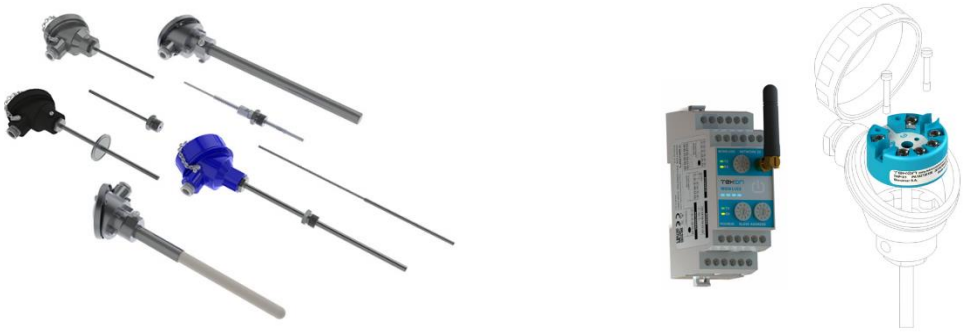
A empresa foi fundada em 1982, por Carlos Breda e restantes sócios, e teve como início de atividade a comercialização de material elétrico para a indústria, construção civil e naval. Desde então o foco da sua principal atividade passou a ser a automação industrial e em 1985 iniciou o processo de representação de grandes marcas do sector para o mercado nacional. Algumas dessas marcas podem se observadas na Figura 2.

Em 1991 a Bresimar passou a ter apenas um acionista (Carlos Breda) e após essa data passou ainda por datas também elas bastantes marcantes, que se passam a destacar. A abertura de uma filial em Lisboa onde conta desde então com uma equipa de comerciais para que possa chegar de forma mais eficiente e prestar também um suporte de maior proximidade aos seus clientes situados mais a sul do país. Desde 2001 que a Bresimar tem vindo a aumentar as suas áreas de negócio, criando para isso duas marcas registadas, a AsaTek e a TekOn. A AsaTek foi criada com o objetivo de fornecer soluções globais de automatização e controlo

industrial e desta forma a Bresimar complementa a sua atividade com o desenvolvimento à medida, adaptado às exigências dos clientes e simultaneamente envolvendo-se na criação de condições para suporte e assistência técnica aos sistemas e soluções que comercializa. Por outro lado, a TekOn é a marca de fabrico próprio da Bresimar, e que se especializou no fabrico próprio de sondas de temperatura adaptadas às necessidades do cliente, e também de transmissores para processo de medição de temperatura com e sem fios, completamente desenvolvidos e fabricados pela Bresimar. O fabrico próprio permite proporcionar uma solução completa aos clientes e que vai desde o elemento sensor até ao equipamento que proporciona a sua leitura e envio dessa informação para o processo industrial.

Figura 2 - Exemplo de marcas representadas pela Bresimar [1]

Foi também com a marca TekOn que a Bresimar em 2013 deu início ao seu processo de internacionalização, começando desde então a estabelecer parcerias no mercado internacional com vista à representação da marca em vários países, contando atualmente com uma rede de distribuição presente em 20 países.



a) Sondas TekOn - Fonte Bresimar

b) Tekon, transmissor *Wireless* - Fonte Bresimar

Figura 3 – Exemplo de equipamentos produzidos pela TekOn [13]

Em 2013 a Bresimar acaba por adquirir o capital maioritário da empresa Selmatron Lda., e desta forma alarga uma vez mais as suas áreas de negócio e de atuação no mercado da automação industrial. As áreas de negócio no grupo Bresimar ficam distribuídas da forma como a Figura 4 pretende demonstrar e resumir. Na Figura 5 podemos consultar a história do percurso da empresa de uma forma cronológica e resumida.

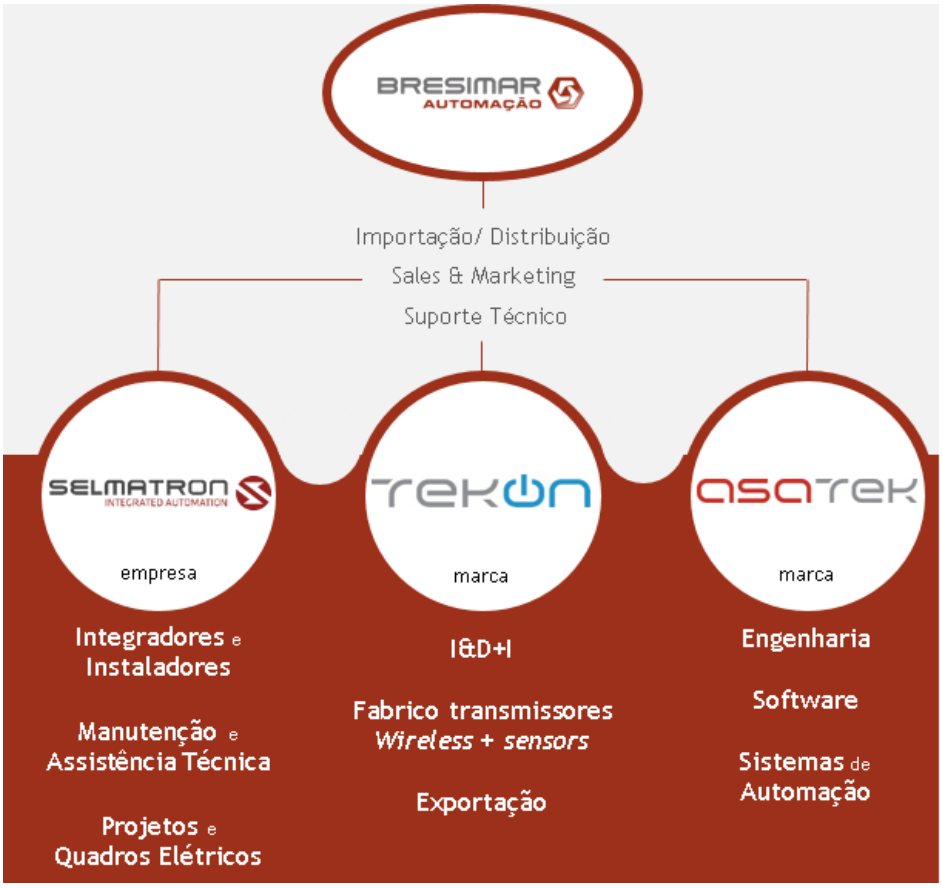


Figura 4 - Áreas de negócio Grupo Bresimar [1]

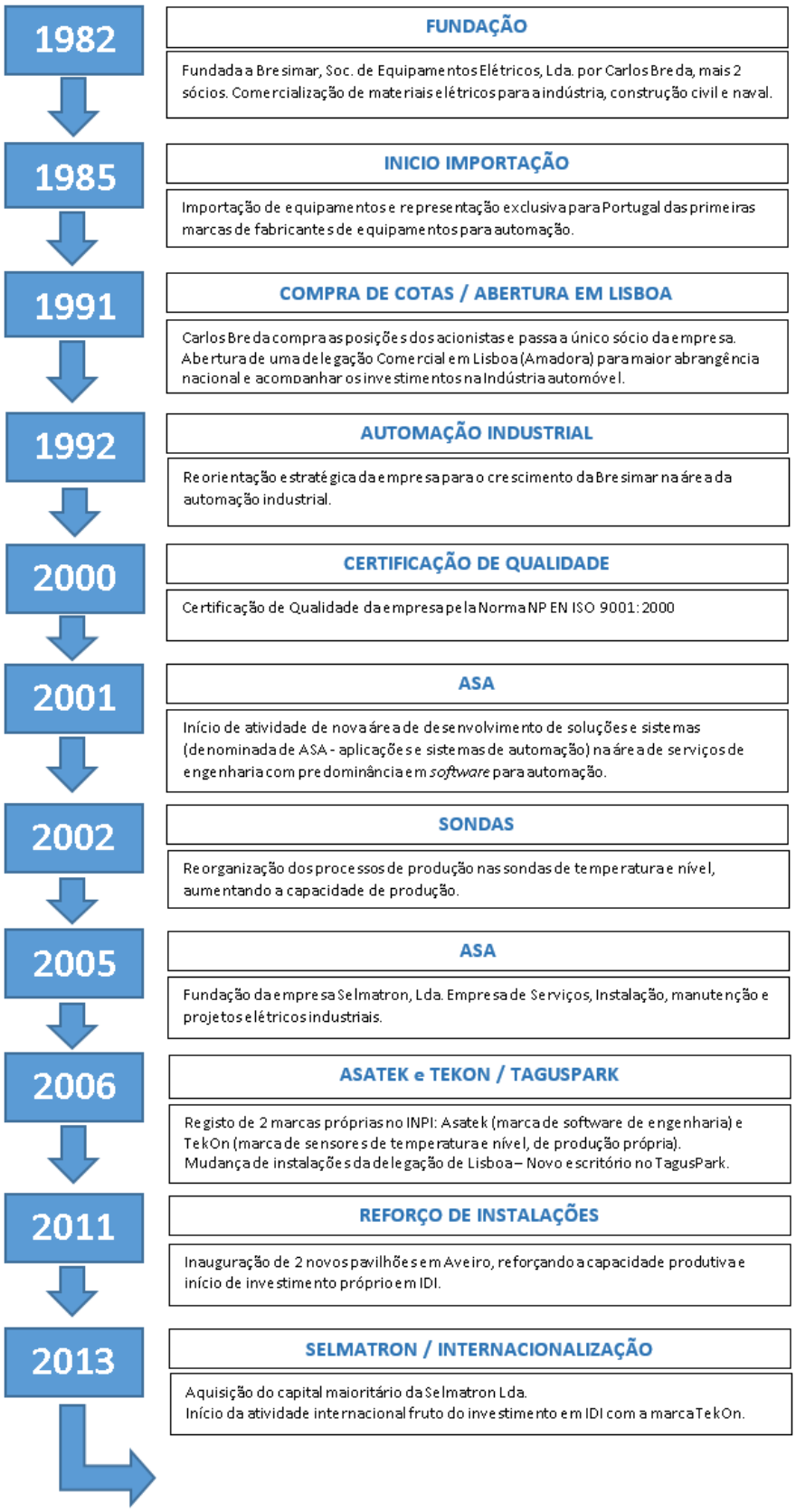


Figura 5 - Formação e história Bresimar [1]

Ao longo dos últimos anos a Bresimar tem recebido ainda alguns prémios na área da gestão dos recursos humanos e de boas práticas de gestão empresarial. Sendo de destacar o mais recente, que ao tempo da redação deste relatório de estágio, foi atribuído pela revista EXAME, onde foi estudado um grande leque de empresas representativas do tecido empresarial nacional, nessa edição são classificadas as "MELHORES EMPRESAS PARA TRABALHAR" edição de 2016, tendo a Bresimar atingido este ano o sétimo lugar, sendo que no ano anterior tinha ficado em vigésimo terceiro. Foi ainda nesta edição distinguida com o primeiro lugar na satisfação dos seus colaboradores na categoria de pequenas empresas e também conseguiu atingir o primeiro lugar na classificação para a geração "*Baby Boomers*" [3].

Figura 6 - Prémios recebidos pela Bresimar [1]

1.2 - Enquadramento do estágio

Nos dias de hoje observa-se cada vez mais uma necessidade crescente do mercado na área dos controladores programáveis, nomeadamente controladores com um nível de complexidade e de custo mais reduzidos. Observamos atualmente que os principais e maiores fabricantes colocam no seu portefólio de produtos, soluções que pretendem abranger e atingir franjas de mercado onde os seus autómatos mais complexos e também com um custo maior não conseguem obter taxas de penetração. Tomemos por exemplo integradores e fabricantes de máquinas, onde, tradicionalmente esse tipo de equipamento necessita de um número muito elevado de entradas e de saídas, e por esse motivo as soluções mais complexas impedem que este sector de mercado as possa adotar. Para fazer face a essa necessidade, a solução atual passa pela aquisição de controladores dedicados e que não permitem qualquer

tipo de adaptação a outras funções. São por exemplo placas eletrónicas desenhadas à medida daquela máquina em particular e que por isso não permitem que as mesmas possam ser programadas e adaptadas a outras necessidades. E ainda assim as que permitem adaptação são programadas em linguagens que não são comuns no mundo da automação industrial, e que por isso não permitem que a equipa técnica as possa trabalhar. São por isso soluções de controlo mais dedicadas pois o custo de autómatos tradicionais é ainda um pouco elevado para alguns sectores de mercado.

Outra das características a que a Bresimar tem estado e à qual continua muito atenta é o facto de que cada vez mais observamos uma maior tendência na uniformização e standardização entre as grandes empresas produtoras de autómatos ou *Programmable Logic Controllers* (PLC), nas suas ferramentas de programação dos seus controladores. Preocupando-se assim em oferecer uma ferramenta de programação que não necessite de uma curva de aprendizagem muito longa, o que por seu lado vai permitir um *time to market* cada vez menor, quer em termos de execução de projeto de desenvolvimento quer na manutenção, pois estes são fatores cada vez mais valorizados e com maior peso, no momento da decisão entre optar por um sistema ou outro seu concorrente.

Resultante dessa necessidade por parte do mercado surgiu então a primeira normalização em linguagens de programação para o sector industrial não proprietária de qualquer marca, e que foi estabelecida pelo IEC (*International Electrotechnical Commission*), com a designação de IEC 61131 (define um conjunto de normas com vista à padronização da programação dos controladores programáveis), mais concretamente a Parte 3 dessa norma que é referente às linguagens de programação. Caminhando nesse sentido os fabricantes de Controladores Programáveis têm procurado e baseado grande parte dos seus produtos num ambiente de desenvolvimento que cumpra com essa norma, cada um com o seu ambiente de trabalho ou *Integrated Development Environm*ent (IDE) e cada um à sua maneira. Desta forma, quando um cliente ou um técnico trabalha com uma determinada marca de autómatos apresenta sempre alguma resistência à mudança, não que provavelmente não esteja disposto a aprender com uma outra qualquer marca, mas porque os tempos de execução de projeto não permitem o luxo de nessa fase se poder utilizar tempo para apreender e ganhar conhecimento em novas ferramentas.

Fruto da sua vigilância tecnológica, na Bresimar é já conhecida uma ferramenta e um produto existente no mercado, que promete ultrapassar algumas dessas limitações expostas acima, nomeadamente a da falta de uniformização nas ferramentas de desenvolvimento e das linguagens de programação dos controladores. A Bresimar comercializa até uma gama de produtos de uma das suas representadas para o mercado nacional, que é o caso do fabricante Alemão Beckhoff, que integra em alguns dos seus produtos alguma dessa tecnologia. A ferramenta em causa tem o nome de CodeSys, e foi desenvolvida pela empresa 3S-Smart Software Solutions.

1.3 - Objetivos

Pretendeu-se com este estágio conhecer e explorar a ferramenta de desenvolvimento criada pela 3S-Smart Software Solutions e avaliar as mais-valias da sua utilização. Foram também objetivos estudar a possibilidade técnica e viabilidade financeira do desenvolvimento de um controlador programável de baixa complexidade e de custo mais reduzido, tendo como linguagem de programação a norma IEC 61131-3, e baseado em CoDeSys.

- Estudar a solução CodeSys;
- Desenhar do ponto de vista de criação de produto os passos necessários até à sua implementação;
- Explorar a viabilidade técnica da integração do sistema CodeSys da 3S-Smart Software Solutions.

1.4 - Organização do documento

Este documento encontra-se dividido e organizado em sete capítulos principais e respetivos anexos, que são os seguintes:

- **Capitulo 1** – Introdução
  Neste capítulo pretende-se apresentar uma introdução ao presente relatório de estágio, o seu enquadramento, e os objetivos a atingir.

- **Capitulo 2** – Características do CodeSys
  Neste capítulo é apresentado o produto CoDeSys e as suas principais características, assim como as suas exigências de recursos. É ainda apresentada uma pesquisa de soluções concorrentes com características semelhantes.

- **Capitulo 3** – Seleção de uma base tecnológica
  Este capítulo apresenta e analisa uma solução semelhante à apresentada pelo CodeSys. Apresenta também uma análise do mercado sobre soluções para potenciar o desenvolvimento de novos produtos utilizando o CoDeSys como ferramenta de programação.

- **Capitulo 4** – Fase de testes
  Neste capítulo é realizado e documentado, passo por passo, todo o processo necessário para a implementação de uma base de testes, que contempla a preparação do *hardware*, do *software*, e de como os utilizar.

- **Capitulo 5** – Arquitetura e desenho detalhado
  Neste capítulo é apresentada uma proposta de uma base de desenvolvimento de *hardware* que permite estudar as necessidades e requisitos de implementação de um produto baseado em CoDeSys.

- **Capitulo 6** – Certificação
  Este capítulo é dedicado à apresentação das principais exigências na marcação CE de um produto com características semelhantes ao desenvolvido no Capítulo 5.

- **Capitulo 7** – Conclusões
  É apresentado neste capítulo um resumo da forma como decorreu o estágio na Bresimar, bem como as principais conclusões e reflexões sobre o trabalho, e ainda uma secção sobre desenvolvimentos futuros na área do trabalho desenvolvido.

- **Referências Bibliográficas**
  Podemos localizar as referências bibliográficas nesta secção.

- **Anexos**
  Nesta secção podemos encontrar todos os documentos considerados relevantes, elaborados e organizados no decorrer do estágio na Bresimar Automação.

2 - CARACTERÍSTICAS DO CODESYS

Este capítulo apresenta as características do produto da 3S-Smart, o CodeSys. Pretende também documentar uma pesquisa de mercado sobre aquelas que são consideradas as opções concorrenciais e as suas principais características, assim como identificar para cada uma delas produtos existentes que as utilizem.

2.1 - Origem do CodeSys

O CodeSys é o nome de um produto desenvolvido por uma empresa Alemã denominada de "3S-Smart Software Solutions" [17], o seu nome é um acrónimo de ***Co**ntroller **De**velopment **Sy**stem* [11] [18] e consiste num ambiente de desenvolvimento de programação que segue a norma IEC 61131, Parte 3.

A norma IEC 61131 vem estabelecer um grande número de padrões para a área dos PLC (*Programmable Logic Controllers*), e está subdividida em 9 Partes:

- Parte 1 – Capítulo introdutório
- Parte 2 – Requisitos de Equipamentos e Testes
- Parte 3 – Linguagens de programação
- Parte 4 – Orientações para o utilizador
- Parte 5 – Comunicações
- Parte 6 – Comunicação via *Fieldbus*
- Parte 7 – Programação de controlo *FUZZY*
- Parte 8 – Implementação das Linguagens
- Parte 9 – Comunicações para pequenos sensores e atuadores

Os criadores do CodeSys garantem aos seus clientes que o seu produto está em conformidade com a Parte 3 da norma em questão. Por esse motivo podemos possibilitar ao programador utilizar qualquer uma das cinco linguagens que fazem parte da norma para desenvolver a sua aplicação ou até mesmo utilizar mais do que uma linguagem no mesmo programa.

As cinco linguagens são:

- ***Structured Text* (ST)** - Semelhante ao Pascal e ao C
- ***Instruction List* (IL)** - Parecido com a linguagem *Assembly*
- ***Ladder* (LD)** - A combinação de relés e bobines
- ***Function Block Diagram* (FBD)** - Blocos de funções com entradas e saídas
- ***Sequential Function Chart* (SFC)** - Linguagem semelhante ao *Grafcet*

Vamos ter oportunidade de observar com maior detalhe este conjunto de linguagens de programação mais à frente neste documento.

Importa também salientar que a utilização do CoDeSys pelos técnicos dispensa qualquer licença e é por isso livre de qualquer custo para o utilizador.

Sabemos que atualmente o CodeSys é uma solução adotada por mais de 250 fabricantes. São exemplo marcas como a Mitsubishi, BECKHOFF, ABB, Schneider, SEW, Bosch ou a Turck [19]. De realçar que das marcas atrás referidas, a Bresimar Automação comercializa para o mercado Português, a Beckhoff a Turck e a SEW.

Todos estes fabricantes de controladores programáveis ao integrarem o CoDeSys nos seus produtos garantem logo à partida uma padronização na linguagem de programação dos mesmos, garantida pela Norma IEC 61131-3.

Tal como já foi exposto no Capítulo 1, existe na Bresimar uma área de engenharia denominada de ASATEK que fornece soluções chave na mão, na área da automação industrial e que tem um forte conhecimento adquirido ao longo de muitos anos, na utilização e implementação de soluções que têm como base o CoDeSys. Foi também este um importante fator na decisão tomada de se estudar a possibilidade de realizar um controlador baseado em CoDeSys e desta forma aproveitar todo o *know-how* já existente.

2.2 - Necessidades para execução do *software* CodeSys

Embora o *software* de desenvolvimento (IDE), integrante do produto da 3S-Smart (CoDeSys) utilizado pelos técnicos para desenvolverem as suas aplicações, seja livre de qualquer custo para o utilizador, o mesmo já não se pode dizer para com quem desenvolve, fabrica e comercializa equipamentos compatíveis com o CoDeSys. Neste caso e por cada equipamento colocado no mercado será necessário proceder ao seu licenciamento junto da 3S-Smart Software Solutions. Desta forma, o fabricante terá de suportar o custo do sistema que a empresa 3S-Smart Software Solutions denomina de *CoDeSys Run Time System* (CodeSys RTS). Esta é a forma de como a empresa detentora do CodeSys implementa o seu modelo de negócio, comercializando licenças para RTS (*Run Time System*).

Como se pode observar na Figura 7, podemos resumir o sistema como sendo constituído por 3 camadas (*Layers*). A camada de mais baixo nível com a designação de *Device Layer*, a camada de comunicação ou *Comunication Layer* e a camada de desenvolvimento ou *Development Layer*.

- ***Development Layer*** - É nesta camada que se encontra o ambiente de desenvolvimento para que o programador possa desenvolver o seu programa do autómato.

- ***Comunication Layer*** - Esta camada serve de interface de comunicação entre a camada de desenvolvimento e a do dipositivo através de uma *OPC Server*, e é responsável pela troca de variáveis entre camadas.
- ***Device Layer*** - Para que um sistema ou dispositivo possa ser programado com o CoDeSys, é necessário que este tenha já o *Runtime* System do CoDeSys implementado e adaptado para esse mesmo *hardware*.

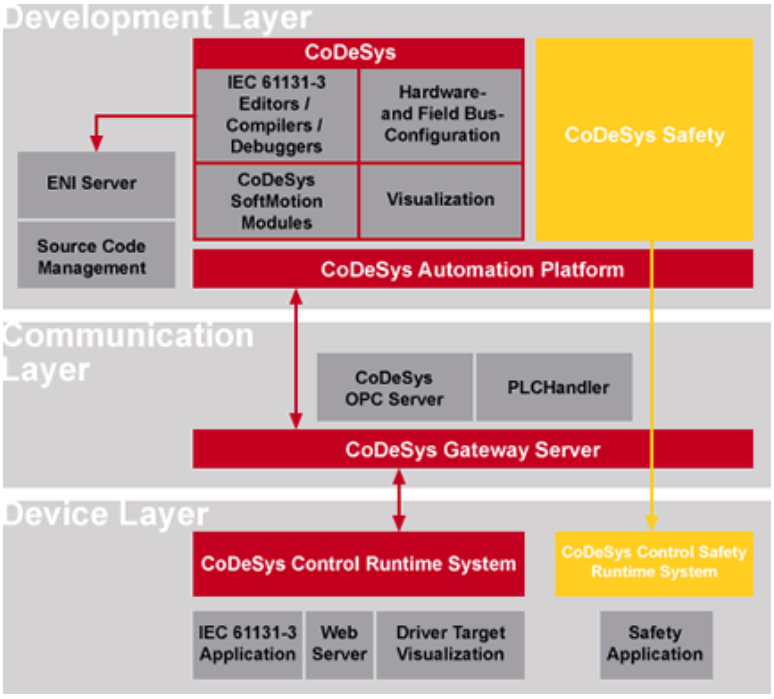


Figura 7 - Esquema Funcional CodeSys [19]

Desta forma, quando se pretende desenvolver um produto que tenha como base o CoDeSys, é necessário em primeiro lugar ter em mente que necessitamos de escolher uma plataforma de *hardware* compatível com o sistema em causa e onde vamos ter de adaptar e customizar o RTS da 3S-Smart para que o mesmo conheça as suas características, como por exemplo, o número de entradas, de saídas, os protocolos de comunicação, e também os seus endereços internos.

A 3S fornece, segundo a informação disponibilizada, um outro serviço a que dá o nome de "*RunTime Toolkit*". Este serviço permite aos seus clientes a redução dos seus tempos de desenvolvimento, a redução dos custos de projeto e um menor tempo de colocação dos seus produtos no mercado. O seu *Toolkit* inclui o seguinte [39]:

- RTS adaptado ao nosso *HardWare* (teremos que especificar todas as características do nosso dispositivo);
- O ambiente de programação (IDE devidamente parametrizado);
- Suporte/Formação e dois anos de manutenção.

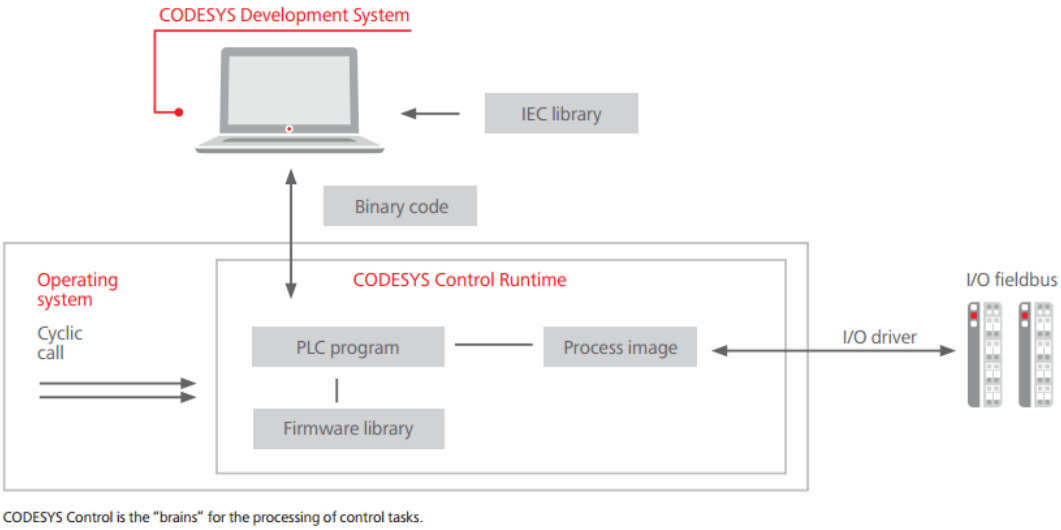


Figura 8 - Esquema de funcionamento de todo o sistema [7]

A 3S-Smart Software Solutions desenhou o produto para que o mesmo seja compatível com várias plataformas ou arquiteturas de processadores existentes no mercado, e até ao momento da escrita deste relatório todas as plataformas descritas em baixo são compatíveis com o seu RTS [40]:

- 8051
- Hitachi H8
- Infineon SAB80C167
- Intel 80186
- Motorola MC68000 (até MC86060)
- Motorola MC683xx
- Motorola ColdFire
- Motorola PowerPC
- ARM CPUs
- 80386, 80486, Pentium X
- Hitachi SH 2 / 3 / 4
- MIPS
- Infineon TriCore
- Blackfin Processors
- Nios II Core for Altera FPGAs
- Outros CPUs estão em preparação

Dependendo da plataforma de *hardware* escolhida, podemos executar o CoDeSys sobre um dos seguintes sistemas operativos:

- Win NT/2K/XP (Real Time)
- Win CE
- RT Kernel
- QNX
- OS/9
- VxWorks
- Linux
- PSOS
- Nucleus
- Sem Sistema Operativo

Tabela 1 - Tabela resumo (CPU vs SO) [19]

	CPU											
Sistema Operativo	8051	80c167	80186	TriCore	680x0 683x0	80x86	ARM	POWER PC	Renesas SH 2/3/4	CodFire	MIPS	NiosII
Sem SO	SP8	SP16	SP32E	SP32E	SP32E	SP32E	*	*	*	*	*	SP32E
Customizado	SP8	SP16	SP32E	SP32E	SP32E SP32F	SP32E SP32F	*	*	*	*	*	SP32E
WinNT/2K/XP (Real time)						SP32F						
WinCE						SP32F	SP32F	SP32F	SP32F		SP32F	
RTKernel						SP32F						
QNX						SP32F						
OS/9					SP32E	*		SP32E				
VxWorks					SP32F	SP32F	SP32F	SP32F	SP32F	SP32E SP32F		
Linux					*	SP32F	SP32F	SP32F	*	*	*	
PSOS					*	*				*	*	
Nucleus		*	*	*	*	*	*	*	SP32F	*	*	

* Possibilidade de adaptação por parte da 3S

Depois de definirmos o *Hardware* e o Sistema Operativo, resta-nos escolher qual a versão do CoDeSys que pretendemos e mais uma vez esta encontra-se limitada pelas opções feitas anteriormente. Da mais simples à mais complexa, são elas:

- CoDeSys SP 8 Bit (SP8)
- CoDeSys SP 16 Bit (SP16)
- CoDeSys SP 32 Bit embedded (SP32E)
- CoDeSys SP 32 Bit full (SP32F)

Na Tabela 1 podemos observar em maior pormenor a compatibilidade entre o CPU e o sistema operativo suportado e testado pela 3S para que seja possível executar o sistema CoDeSys.

2.3 - Alternativas ao CodeSys

Pretende-se neste subcapítulo realizar um levantamento das soluções existentes no mercado e que sejam concorrentes com o produto comercializado pela 3S Smart, descrever não só as suas principais características bem como também, e se possível, identificar fabricantes que as integrem no seu portfólio de produtos. São disso exemplo projetos como o ProConOS, Straton, SmartPLC, ISaGRAF e o LogicLab, os quais se expõem com maior detalhe de seguida.

2.3.1 - ProConOS

O ProConOS é um sistema operativo de tempo real multitarefa que conta com um RTS (*Runtime System*) PLC para execução de código segundo a norma IEC 61131-3 que é comercializado pela empresa KW-Software GmbH na Alemanha [24].

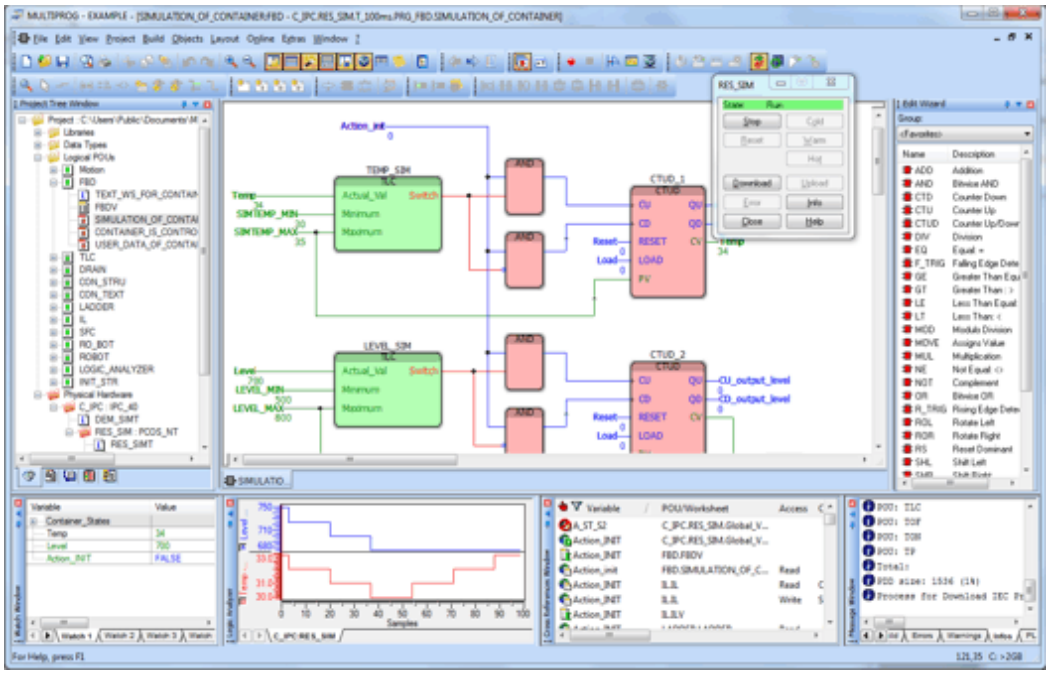


Figura 9 - KW-Software MULTIPROG IDE [23]

Para além do ProConOS a marca tem também a possibilidade do ProConOS *embedded* que pode ser incorporado em outros sistemas operativos, tais como o VxWorks, Windows CE, Intime *Soft-PLC* e *Linux*. O seu RTS está compatibilizado com plataformas que possuam os

seguintes CPU: x86, NIOS II/III, MIPS64, ARM7, ARM9, ARM11, Renesas SH2/SH3/SH4, PowerPC e300 e Cortex M3. Segundo a KW-Software é ainda possível portar ou adaptar o seu RTS para outras arquiteturas.

Finalizada a adaptação do RTS da ProConOS para o *hardware* desejado, esses equipamentos ficam desde logo habilitados a serem programados com o IDE também fornecido pela KW-Software, neste caso o MULTIPROG (Figura 9).

Existem três versões do seu programa. Uma versão Express, que é a versão mais básica e com um menor número de funcionalidades, e que é por isso recomendada para sistemas de menor complexidade e para programadores menos experientes. Existe também a versão PRO+ e a versão *Suite*. Estas duas últimas encontram-se dotadas de um maior número de ferramentas. Para além destas três versões, está também disponível o MULTIPROG 10 que é um *framework* para automação permite a programação com as linguagens IEC 61131-3 combinada com as potencialidades de ferramentas da *framework .NET*.

Figura 10 - KinCon-8045 [43]

Figura 11 - ADAM-5510EKW [27]

Podemos encontrar no mercado alguns produtos com a tecnologia desenvolvida pela empresa KW-Software, são exemplo disso o caso da ICP DAS, com o seu modelo KinCon-8045 [23] (Figura 10), e da Advantech com o seu modelo ADAM-5510EKW [27] (Figura 11), que que têm como base o RTS ProConOS.

2.3.2 - COPALP

A solução apresentada nesta mesma área pela empresa COPALP tem o nome de Straton. O Straton é um RTS PLC compatível com a norma IEC 61131-3, que pode ser instalado com ou sem sistema operativo e é compatível com uma vasta gama de diferentes SO.

Segundo a COPALP o Straton está disponível para o Windows CE, Windows XP Embedded, Linux, VxWorks, ThreadX, VRTX, DOS, Nucleus, Xenomai, ucLinux, Intime e RTX.

Juntamente com o seu RTS, a COPALP tem também editor de programas (Figura 12) nas linguagens de programação da norma IEC 61131-3.

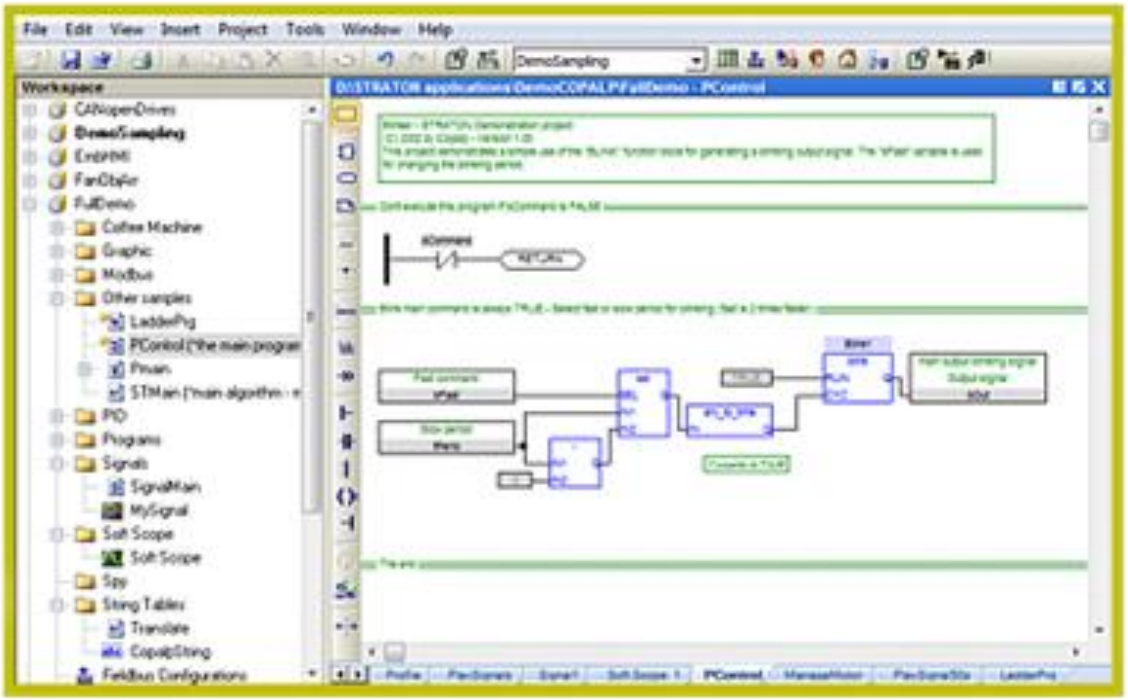


Figura 12 - COPALP IDE [28]

Tal como no caso anterior, é também possível encontrar no mercado, e com alguma facilidade, produtos que fazem referência ao facto de terem adotado a solução da COPALP e do seu RTS. Para exemplificar isso mesmo vamos observar apenas os produtos das marcas WAGO, Brodersen e Siemens (Figura 13, Figura 14, Figura 15 e Figura 16 respetivamente).

Figura 13 - WAGO Series 758 [12]

Figura 14 - WAGO Series 750-865 [12]

Figura 15 - Brodersen RTU32 [31]

Figura 16 - Siemens S7 mEC EC31 [47]

2.3.3 - Infoteam

A empresa Infoteam Software AG tem também um RTS PLC, semelhante aos apresentados anteriormente, diverge contudo no modelo de negócio para a comercialização do seu produto, pois o custo da licença é anual e independente do número de unidades vendidas. A Infoteam disponibiliza também vários tipos de *workshops* para acelerar o processo de desenvolvimento inicial, e para facilitar a adaptação do seu RTS à plataforma do seu cliente. Tem como ambiente de desenvolvimento o OpenPCS também da Infoteam, e o seu aspeto gráfico pode ser visto na Figura 17. Como referência de utilização do RTS da Infoteam e do seu IDE, podemos encontrar um fabricante com sede na Rússia. A marca tem a designação de EleSy (Figura 18), e após consulta da sua página na Internet, podemos confirmar que os modelos de controladores que fabrica têm como base o produto da Infoteam [48].

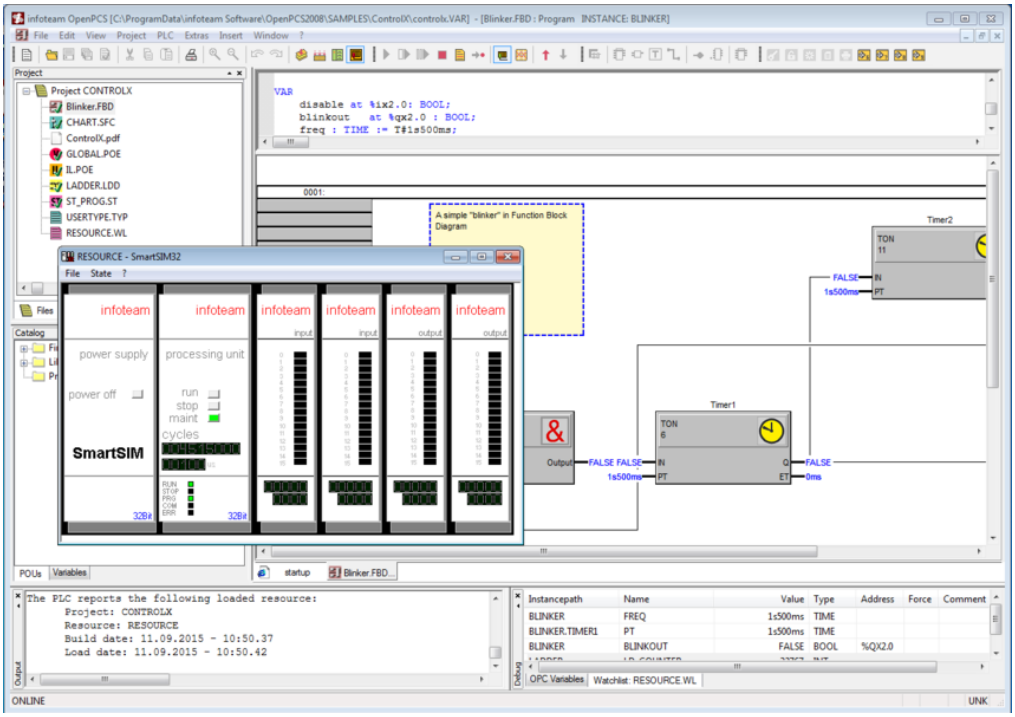


Figura 17 - IDE Open PCS [42]

Um segundo exemplo de utilização leva-nos até um fabricante Italiano de produtos para automação com o nome AsconTecnologic (Figura 19) que também utiliza como base nos seus controladores o seu RTS e IDE [16].

Figura 18 - EleSy - TC 506 C400 C [48]

Figura 19 - Ascon sigmaPAC CU-02 [16]

2.3.4 - ISaGRAF

Foi analisada mais uma solução que também, como as anteriores até aqui apresentadas, é constituída por um IDE, a que a ISaGRAF dá o nome de *Application Workbench* e por um RTS. O IDE pode ser observado na Figura 20. Este ambiente de desenvolvimento contempla também a programação segundo a norma internacional IEC 61131-3.

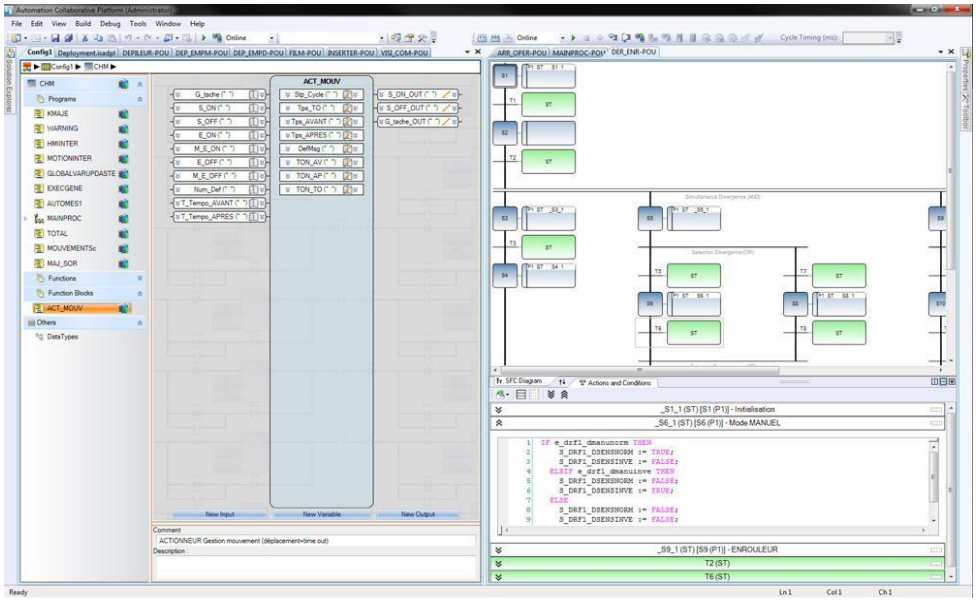


Figura 20 - ISaGRAF IDE [34]

Segundo a ISaGRAF, o seu RTS foi já adaptado e compatibilizado para uma grande variedade de sistemas operativos tais como NT, RTX, CE, LINUX, VxWORKS, QNX, OS9, ThreadX ou o PSOS. Tal como as soluções encontradas e descritas até aqui, não é difícil encontrar marcas que utilizam o seu RTS nem o seu IDE. Exemplo disso é a marca Sixnet com a sua gama de controladores SixTRAK (Figura 21), e que recentemente foi adquirida pela empresa Redlion, ou mais duas marcas com sede em Itália. A Seneca, (Figura 22) e a Arteco [15] que também adotaram o ISaGRAF como seu *Runtime System* e ambiente de desenvolvimento (Figura 23 e Figura 24).

Figura 21 - Sixnet - SixTRAK [29]

Figura 22 - SENECA Z-TWS-3 [25]

Figura 23 - ARTECO SU-PLC [15]

Figura 24 - ARTECO SU310-PLC [15]

2.3.5 - LogicLab

Da procura no mercado de mais soluções concorrentes com o CoDeSys, surgiu mais uma empresa a oferecer um produto com características semelhantes, a AXEL [44]. A AXEL é uma empresa Italiana que iniciou a sua atividade em 1998 e desde essa altura que se tem dedicado à oferta de soluções para o mercado industrial. A AXEL comercializa um IDE compatível com a norma IEC 61131-3 que desde 2001 é denominado por LogicLab.

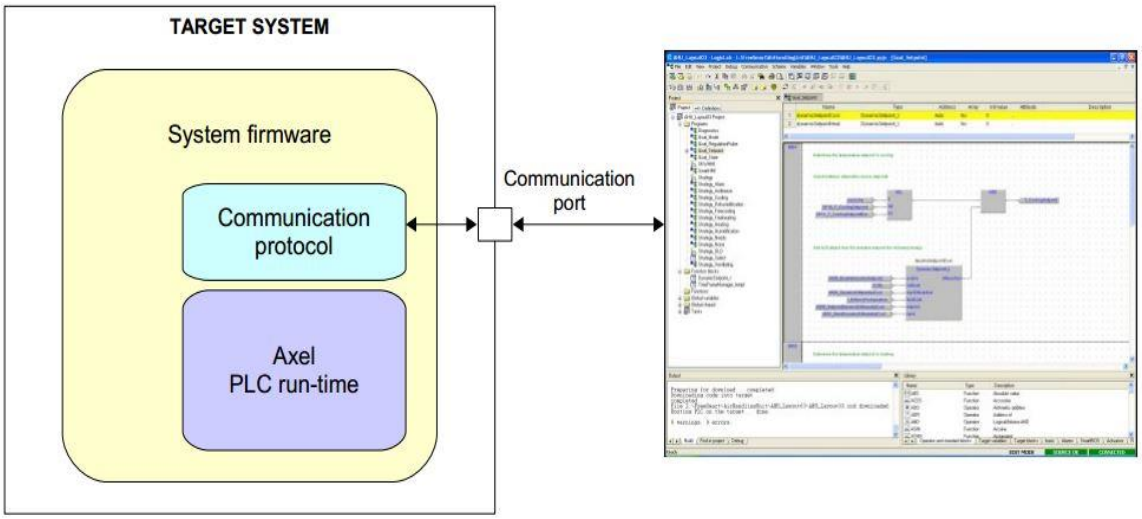


Figura 25 - AXEL RT e LOGICLAB IDE [44]

O IDE LogicLab (Figura 25) suporta as cinco linguagens presentes na norma, e segundo a AXEL consegue gerar o código máquina do processador de uma forma muito eficiente. O LogicLab consegue distanciar-se dos seus concorrentes por ser o único concorrente do CoDeSys aqui estudado capaz de funcionar com processadores de 8 bits da família de microprocessadores AVR da Atmel.

O LogicLab consegue gerar segundo a Axel, o código para as seguintes arquiteturas de microprocessadores:

- Intel X86 e compatíveis (Pentium, VIA, Geode, Atom etc.)
- ARM7, ARM9, ARM11
- ARM Cortex M3/M4
- Texas TMS320x
- Família Infineon C16x (XC16x, XE16x, C16x)
- Família Infineon TriCore
- Freescale ColdFire
- Fujitsu 16FX/LX
- Mitsubishi M16C
- Atmel AVR
- Intel i960

O seu IDE (Figura 26) é de utilização gratuita e sem qualquer limitação na sua utilização. No entanto, para que possamos integrar o RT em nossos produtos é necessário o pagamento de uma licença por cada unidade vendida.

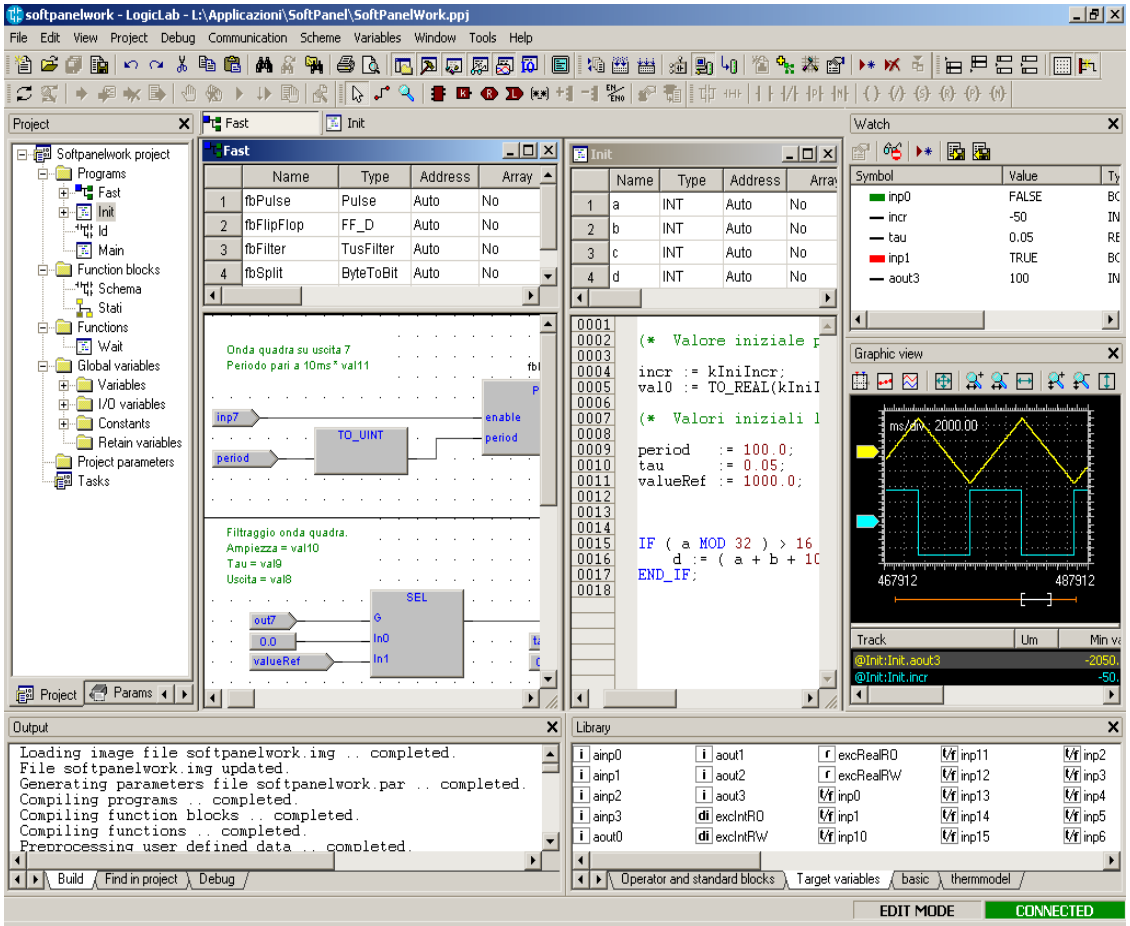


Figura 26 - LogicLab IDE [44]

A AXEL disponibiliza ainda um módulo (AXC25) pronto a integrar um qualquer produto de seus clientes, e que já conta com um RTS PLC e com um RTS HMI (*Human Machine Interface*) devidamente adaptado e integrado no módulo (Figura 27). Este produto visa acelerar o processo de desenvolvimento de novos produtos, ficando a cargo dos clientes que adotem esta solução apenas a necessidade de eletricamente adaptarem as suas ligações ao processo onde pretendem instalar o seu produto. O AXC25 pode ser programado através da ferramenta PageLab também da AXEL. As suas principais características são as seguintes:

- **Processador** - i.MX257, 400 Mhz
- **Memoria** - 64 MB SDRAM / 128 MB NAND Flash
- **Comunicações** - Ethernet/USB/Serial/I2C/SPI/CAN
- **HMI** - Controlador LCD 800x600/Teclado 4X4/Controlador ecrã táctil
- **Redes de campo**
  - Modbus RTU
  - Modbus TCP/IP
  - CAN Open
  - EtherCat Master

Figura 27 - Módulo AXC25 da AXEL [44]

Figura 28 - SlimLine CPU - Elsist [21]

Figura 29 - OPD EXP [26]

Também podemos encontrar no mercado produtos baseados na solução da da Axel, como são o caso do CPU da empresa Elsist (Figura 28**)** e do OPD EXP da TDE-MACNO (Figura 29).

2.4 - Resumo de alternativas ao CodeSys

Após a realização da análise ao estado da arte para ambientes de desenvolvimento integrados (IDE) e de *Run Time System* compatíveis com a norma IEC 61131-3, podemos obviamente concluir que não existe apenas o produto e a solução proposta pela 3S (CoDeSys). Temos por isso possibilidade e flexibilidade na escolha de um sistema completo. A 3S-Smart tem assim alternativas e uma forte concorrência, e que mediante as características e os requisitos do produto a desenvolver, ou da linha de produtos, bem como da relação entre o tempo de desenvolvimento versus o custo necessário ao seu desenvolvimento, é possível que uma solução possa ter vantagens sobre a outra. Não foi possível, no entanto realizar uma análise mais exaustiva às soluções oferecidas pela K-Software, COPALP, ISaGRAF, AXEL ou pela Infoteam, quer pelas licenças envolvidas na utilização do seu *software*, quer pela inexistência de equipamentos das marcas atrás mencionadas, na vasta gama de produtos comercializados pela Bresimar.

Na Tabela 2, temos a oportunidade de analisar um quadro resumo com as principais características de cada solução aqui descrita.

Tabela 2 - Quadro Resumo com alternativas concorrentes ao CodeSys

Produto	Fabricante	IEC 61131	Sistema Operativo	Arquitetura
ProConOS	KW-Software GmbH	SIM	• ProConOS embedded CLR • ProConOS embedded CLR Single • ProConOS embedded CLR VxWorks • ProConOS embedded CLR Windows CE • ProConOS embedded CLR Intime Soft-PLC • ProConOS embedded CLR Linux	• X86 • NIOS II/III • MIPS64 • ARM7/ARM9/ARM11 • SH2/SH3/SH4 • PowerPC e300 • Cortex M3 • Outros tipos de CPU a pedido
STRATON	COPALP	SIM	• QNX • Linux (uClinux, RTAI, Xenomai, RTLinux • Debian, Fedora, SuSE) • VxWorks • Windows (WinCE, XP, Vista and 7) • ThreadX • VRTX • DOS • Nucleus • Intime • RTX	-----

Tabela 2 - Quadro Resumo com alternativas concorrentes ao CodeSys (cont.)

Produto	Fabricante	IEC 61131	Sistema Operativo	Arquitetura
SmartPLC	infoteam	SIM	• Linux • CE • XP • OSX • RTX • CMX • pxPros • VRTX • Kadak AMX • µCOS • VxWorks • Nucleus • eCOS • EmbOS • TI BIOS • RMOS	• Intel 486, 586, Pentium • ARM7, ARM9, ARM11 • Xscale • AMD • PowerPC • Motorola 68k • Coldfire • TMS320 • TI • SH2, SH4,H8 • CortexM3 • DSP • GEODE • Inf. 167 • Dual- and multi-core
LogicLab	AXEL	SIM	• Windows • Vários RTOS compatíveis • Outros a pedido	• Intel X86 e compatíveis (Pentium, VIA, Geode, Atom etc.) • ARM7, ARM9, ARM11 • ARM Cortex M3/M4 • Texas TMS320x • Família Infineon C16x (XC16x, XE16x, C16x) • Família Infineon TriCore • Freescale ColdFire • Fujitsu 16FX/LX • Mitsubishi M16C • Atmel AVR • Intel i960
CoDeSys	3S-Smart Software Solutions	SIM	• Win NT/2K/XP • (Real Time) • Win CE • RT Kernel • QNX • OS/9 • VxWorks • Linux • PSOS • Nucleus • Sem Sistema Operativo	• 8051 • Hitachi H8 • Infineon SAB80C167 • Intel 80186 • Motorola MC68000 (até MC86060) • Motorola MC683xx • Motorola ColdFire • Motorola PowerPC • ARM CPUs • 80386, 80486, Pentium X • Hitachi SH 2 / 3 / 4 • MIPS • Infineon TriCore • Blackfin Processors • Nios II Core for Altera FPGAs
ISaGRAF	ISaGRAF	SIM	• NT • RTX • CE • LINUX • VxWORKS • QNX • OS9 • ThreadX • PSOS	

3 – SELECÇÃO DE UMA BASE TECNOLÓGICA

Neste capítulo vamos abordar uma solução baseada em CodeSys e explorar o seu funcionamento e interação com o utilizador. Pretende-se também pesquisar se o mercado tem já disponíveis soluções ou módulos que possibilitem a implementação do produto da 3S-Smart de uma forma mais expedita e se possível que essa solução seja possível de adotar para futuros produtos a desenvolver pela equipa da TekOn na Bresimar.

3.1 - *Twincat Software System*

Com o objetivo de ganhar um maior conhecimento sobre o IDE CoDeSys, o caminho apontado foi o de explorar um dos fabricantes que o adotaram nos seus produtos. A sugestão foi a de explorar o Twincat do fabricante Alemão Beckhoff. O Twincat, embora visualmente e à primeira vista não seja idêntico ao IDE do CodeSys, foi construído tendo como base o CoDeSys. No entanto, fruto da sua grande variedade de produtos e soluções, este fabricante optou por customizar o seu IDE de acordo com as suas necessidades. Para isso foi necessário realizar e implementar uma aplicação de um sistema de controlo real.

a) BECKHOFF EK1100 [30]

b) BECKHOFF Twincat [30]

Figura 30 – Solução BECKHOFF [30]

O *hardware* que vamos usar para esse efeito é constituído por um módulo de I/O's remoto, modelo EK1100, e de cartas de I/O que vão ficar ligadas ao módulo EK1100. O módulo EK1100 que pode ser visto na Figura 30, é um *slave* da rede de comunicações EtherCat (protocolo de rede desenvolvido também pela Beckhoff), que não tem funcionalidades de PLC local, ou seja, podemos dizer que não tem inteligência local. Assim sendo, este módulo serve de interface entre o *master* da rede *EtherCat* e os I/O das cartas que agregarmos ao

EK1100. Como *master* de rede vamos usar o PC, pois juntamente com o IDE proporcionado pelo Twincat é possível utilizarmos o PC como *Soft PLC*.

O TwinCat pode ser descarregado gratuitamente da página do fabricante, bastando para isso realizar um pequeno registo. O *Master EtherCAT* é assim programado através da ferramenta TwinCat que é o IDE de programação para todos os autómatos da marca BECKHOFF.

Após a instalação do *software* ficamos com o ícone na barra do *Windows* igual e de cor azul, o que significa que o sistema foi bem instado e que o PLC se encontra desligado.

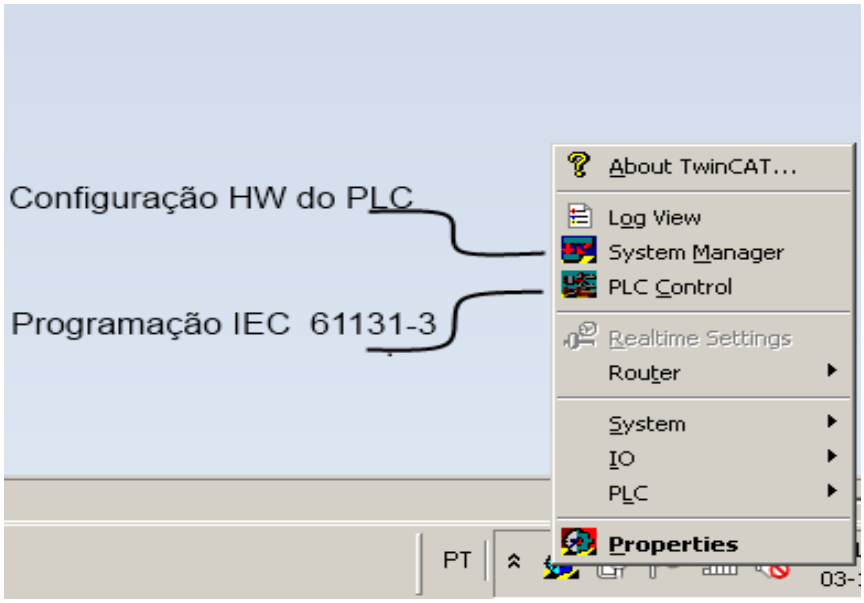


Figura 31 - TwinCat menu

Ao selecionarmos o ícone com o botão direito do rato no menu do programa principal do programa é apresentada a Figura 31, e desta forma podemos facilmente optar por qual dos módulos pretendemos usar, existem dois blocos principais. São eles:

- *PLC Control* - O *PLC Control* (Figura 32) é o IDE propriamente dito de programação da aplicação que vai estar a ser executada pelo PLC.
- *System Manager* - É no *System Manager* (Figura 33) que realizamos a configuração do *hardware* e a ligação entre as variáveis do programa e os I/O do autómato.

É no *PLC Control* onde vamos escrever o código da nossa aplicação que vai ser executado pelo PLC. Tal como vimos anteriormente, esta componente do TwinCat tem como base o CodeSys e por isso as linguagens de programação são as que a norma IEC 61131-3 descreve. Podemos observar isso na Figura 32 onde ao iniciarmos um novo programa (POU), podemos escolher entre uma das seis linguagens disponíveis, e ainda definir se o que vamos escrever será um programa, uma função, ou uma função-bloco.

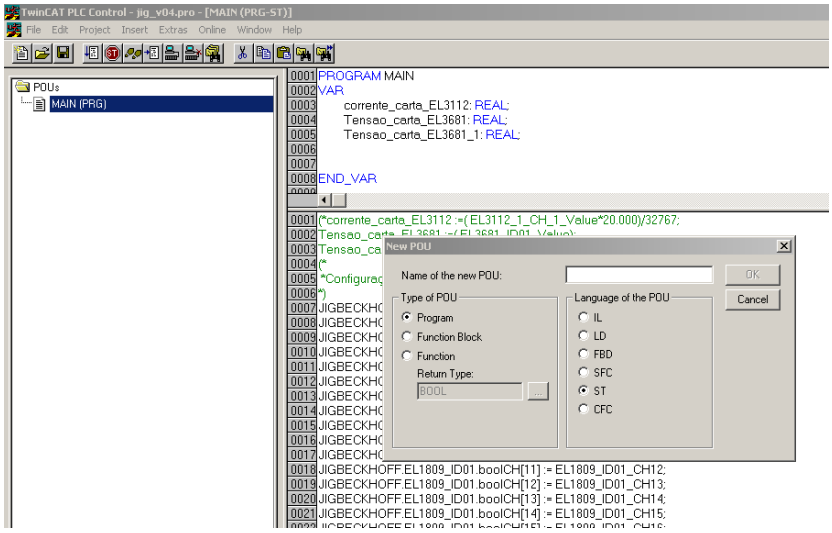


Figura 32 - TwinCat *PLC Control*

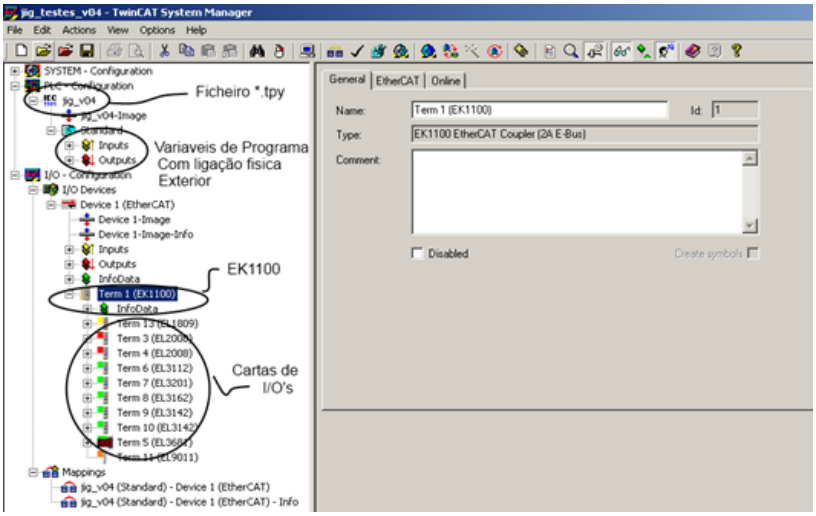


Figura 33 - TwinCat *System Manager*

Terminada a escrita do nosso programa e após a compilação com sucesso do mesmo, é gerado um ficheiro que vai ser necessário abrir no *TwinCat System Manager*. Esse ficheiro tem a extensão “tpy” e nele está contida informação essencial para o próximo passo, pois trás com ele a informação das variáveis de programa, às quais vamos no próximo passo atribuir uma variável física e presente no *hardware* das cartas de I/O, e que vai interagir com o mundo exterior.

Na Figura 33 podemos ainda observar o local onde vamos abrir o ficheiro gerado pelo *PLC Control* (extensão “tpy”) e que contém as variáveis de entrada e de saída que definimos na altura da criação do nosso programa. É nesta altura que as vamos fazer relacionar com as nossas cartas de I/O por forma a que a mudança de estado de uma variável de entrada numa das cartas digitais por exemplo faça alterar a variável dentro do nosso programa.

É possível também aqui observar toda a nossa rede de entradas e saídas do nosso *Master PLC EtherCAT*, e neste caso verificamos que apenas temos um escravo na rede, que é o nosso módulo EK1100.

Este módulo EK1100 está dotado de cartas de I/O dedicadas para uma aplicação específica, e neste caso este conjunto de entradas e de saídas foram selecionadas e montadas no nosso laboratório (Laboratório TekOn).

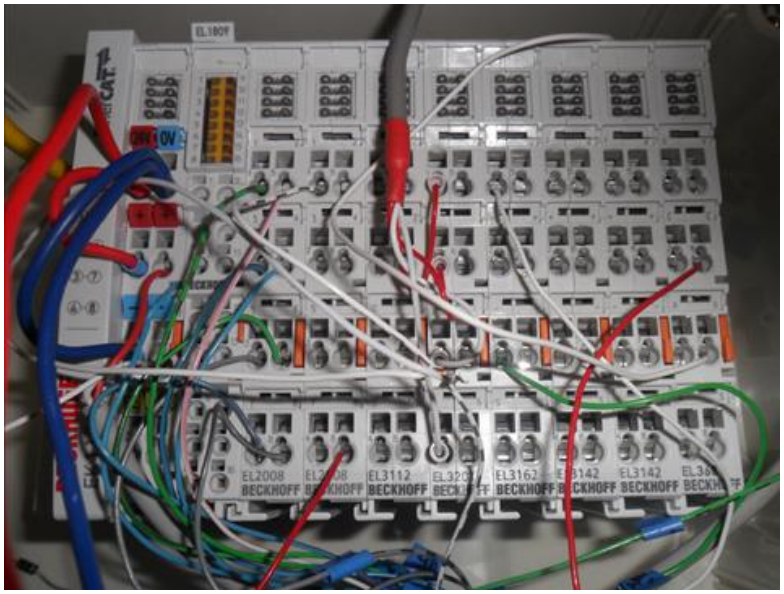


Figura 34 - PLC com Modulo EK1100 e respetivas cartas

Essa montagem (Figura 34) tinha como principal objetivo o conhecimento mais aprofundado do TwinCAT, e simultaneamente a realização de tarefas de *Data Logger*. Por este último motivo as suas características em relação às suas entradas e saídas, foram selecionadas tendo em mente esse objetivo. Este PLC está dotado de características especiais, tendo em conta as características dos produtos desenvolvidos pela Bresimar, e pretende-se que seja uma ferramenta para implementar testes de longa duração dos equipamentos em desenvolvimento. A arquitetura modular de criação do PLC desenvolvida pela Beckhoff, permite ao cliente adicionar apenas o conjunto de variáveis de que necessita, o que garante uma flexibilidade total por parte do utilizador. Tendo em conta as necessidades, o PLC foi selecionado para ser constituído pelo seguinte conjunto de cartas de expansão:

- **EL1809** - 16 Entradas Digitais
- **EL2008** - 8 Saídas Digitais
- **EL2008** - 8 Saídas Digitais
- **EL3112** - 2 Entradas Analógicas de 0 mA a 20 mA de 16 *bits* Diferencial
- **EL3201** - 1 Entrada Analógica de Medição de RTD PT100
- **EL3162** - 2 Entradas Analógicas de 0 V a 10 V de 16 *bits*
- **EL3142** - 2 Entradas Analógicas de 0 mA a 20 mA de 16 *bits*

- **EL3142** - 2 Entradas Analógicas de 0 mA a 20 mA de 16 *bits*
- **EL3681** - Multímetro Digital de 18 *bits*

O módulo completo com as respetivas cartas pode ser observado na Figura 34. A foto foi tirada ainda numa fase de testes ao programa do PLC.

O trabalho e esforço desenvolvidos até esta fase permitiu dar os primeiros passos para uma melhor compreensão de um sistema de controlo baseado numa tecnologia em muito semelhante ao CodeSys e desta forma também ganhar uma maior sensibilidade para que os próximos passos de contacto direto com a sua implementação possam decorrer de uma forma mais expedita e proveitosa.

3.2 - Pesquisa de módulos com CodeSys integrado

Será necessário selecionar do mercado um fabricante de módulos que já integrem o RTS do CodeSys. Este será sempre um passo importante para o desenvolvimento de novos produtos. Assim, é possível testar de uma forma mais rápida recorrendo às ferramentas normalmente disponibilizados pelos fabricantes, bem como à sua documentação, e caso o resultado seja positivo, então nessa fase posterior desenvolver o produto recorrendo ao desenho do próprio *hardware*, que normalmente permite otimizar a produção, o custo e as restrições mecânicas.

Assim sendo e fruto desta pesquisa, foi possível identificar no mercado um conjunto significativo de componentes e de módulos que cumprem com o nosso objetivo. O mesmo pode ser observoda na Tabela 3.

Tabela 3- Tabela Resumo com módulos com CoDeSys

Fabricante	Referencia	CPU	RAM	Programa	Conectividade	Fooprint	Imagem
BECK	SC123-IEC-LF	SC186EX/96 MHz	8 MB	2 MB	34 GPIOs 2 x CAN 2.0 USB 1.1 host and device 4x TTL USART 2 x Ethernet	BGA 177	
BECK	SC143-IEC-LF	SC186EX/96 MHz	8 MB	8 MB	34 GPIOs 2 x CAN 2.0 USB 1.1 host and device 4x TTL USART 2 x Ethernet	BGA 177	

Tabela 4- Tabela Resumo com módulos com CoDeSys (cont.)

Fabricante	Referencia	CPU	RAM	Programa	Conectividade	Fooprint	Imagem
BECK	SC23-IEC	SC186EX/96 MHz	8 MB	2 MB	17 GPIOs 2 x CAN 2.0 USB 1.1 host and device 3x TTL/RS232 1 x Ethernet	DIL 32	
BECK	SC24-IEC	SC186EX/96 MHz	8 MB	8 MB	17 GPIOs 2 x CAN 2.0 USB 1.1 host and device 3x TTL/RS232 1 x Ethernet	DIL 32	
BECK	SC243-IEC3	MPC5200/400 MHz	64 MB	32 MB	41 GPIOs 6x TTL/RS232 1x10/100BaseT USB1.1 2 x CAN2.0b	PCB Module 60 x 56 mm	
Exorint	SCM11-C	79RV3041	8 MB	1 MB	Ethernet		
Exorint	SCM-05-C	79R3041/24 MHz	8 MB	4 MB	CAN		
telestar	ZERO CPU	ARM9/624MHz	64 MB	1GB	RS485 1x ethernet 10/100	PCB Module 90mm x 50mm	
frenzel berg	EASY215	C164CI 16bit	32 KByte	16 KByte	20 GPIOs 1 CAN	DIL-40 socket 52 x 20 mm	
frenzel berg	EASY217	C164CI 16bit	16 KByte	48 KByte	20 GPIOs 1 CAN	DIL-40 socket 52 x 20 mm	
frenzel berg	EASY219	C164CI 16bit	16 KByte	48 KByte	20 GPIOs 1 CAN	DIL-40 socket 52 x 20 mm	

Tabela 5- Tabela Resumo com módulos com CoDeSys (cont.)

Fabricante	Referencia	CPU	RAM	Programa	Conectividade	Fooprint	Imagem
frenzel berg	EASY235	ST10F269 16Bit	156 KByte	256 KByte	48 GPIOs 1 CAN 4 Serial port	PCB Module 63 x 45 mm	
Hilscher	NPLC-M100-DP	NETX 100 ARM 926 200MHz	8 MB	4 MB	84 GPIOs UART PROFIBUS SPI I2C Bus	PCB Module 80 x 50 mm	

Após todo o trabalho de pesquisa e de identificação destes módulos, o passo seguinte foi o contacto com os fabricantes, para conseguir obter mais informação, sobre algumas características técnicas e também sobre o custo destas soluções. Os preços negociados entre a Bresimar os respetivos fabricantes, por uma questão de confidencialidade, não foram considerados relevantes e por esse motivo acabaram por não constar neste relatório de estágio. Na grande maioria dos casos os fabricantes reencaminharam-nos para os seus distribuidores para Portugal ou para a Península Ibérica. De realçar que o fabricante Exorint apenas disponibiliza os módulos SCM11-C e SCM-05-C para que possam integrar as suas consolas gráficas e desta forma dotar esses equipamentos de capacidades avançadas de PLC.

Foram avaliadas varias características para cada módulo e fabricante com vista a escolher o melhor deles a adotar para testes e para que possa integrar, se for caso disso, os produtos da Bresimar. Este foi um passo muito importante e que tomou um tempo significativo no decorrer do estágio, pois uma escolha errada ou menos acertada nesta fase poderia por em causa o desenvolvimento futuro de todo o projeto.

3.3 - Seleção de uma base tecnológica

Após avaliar características como, capacidade de memória, frequência de trabalho, quantidades de I/O disponíveis, flexibilidade para outras aplicações, comunicação, suporte por parte do fabricante, disponibilidade de placas de desenvolvimento, tamanho (*fooprint*) que ocupa em PCB, e por último mas não menos importante, o custo associado a cada um deles, chegámos a um consenso na escolha do fabricante e o módulo em causa. A escolha depois da análise de todos estes pontos, recaíu sobre o módulo SC143-IEC-LF do fabricante BECK.

De referir que após contacto com os fabricantes alguma informação nomeadamente a cotação foi aqui omitida por uma questão de sigilo e de ética empresarial.

Desta forma foi possível selecionar um *kit* de desenvolvimento baseado nesse módulo da empresa BECK. O modelo então selecionado foi o EK61 (Figura 35) para que possa servir de base inicial ao desenvolvimento da nossa aplicação. Este conjunto inclui um conjunto de *software* e de *hardware* básico necessário. O kit EK61 inclui uma placa de circuito impresso com o SC143-IEC-LF incluído (EB60), o *Runtime* da 3S (CoDeSys) e a sua licença, juntamente com o RTOS da BECK também com a sua licença incluída e ainda um compilador da Paradigm C/C++ BECK IPC Edition.

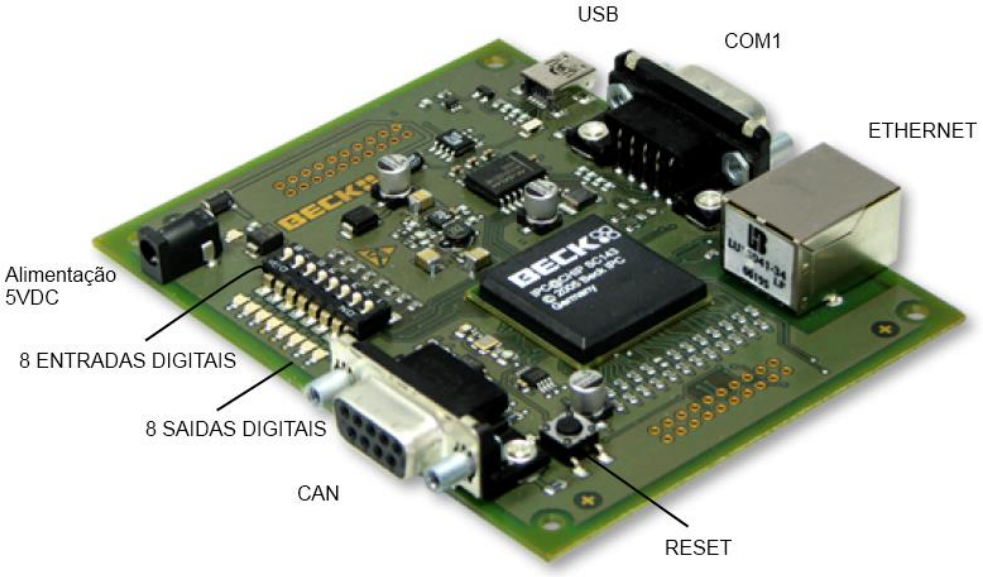


Figura 35 - Placa de desenvolvimento EB60 presente no *kit* EK61

4 - FASE DE TESTES

Neste capítulo vamos ter oportunidade de detalhar todos os passos necessários para implementar o sistema completo, e que vão desde a preparação do ambiente de trabalho, passando pela criação do RTS necessário e devidamente adaptado ao *hardware* da placa de desenvolvimento da BECK até ao teste de uma aplicação. Recorrendo para isso ao IDE da 3S-Smart e à criação de um programa para testar a correta implementação do sistema. Todos estes passos são realizados e devidamente documentados recorrendo à placa de desenvolvimento selecionada no Capítulo 3, a EK61.

4.1 - Parametrização do ambiente de trabalho

Após a aquisição da placa de desenvolvimento por parte da Bresimar, foi possível iniciar o estudo e trabalho de aprendizagem para colocar todo o sistema em funcionamento. A opção por esta plataforma foi reforçada pelo facto de satisfazer todas as condições iniciais para realizar e implementar este estudo. Ela permite também por outro lado uma maior concentração no esforço da implementação do *software* pois partimos de uma base de *hardware* que nos dá garantias por parte do seu fabricante de tudo estar testado e funcional.

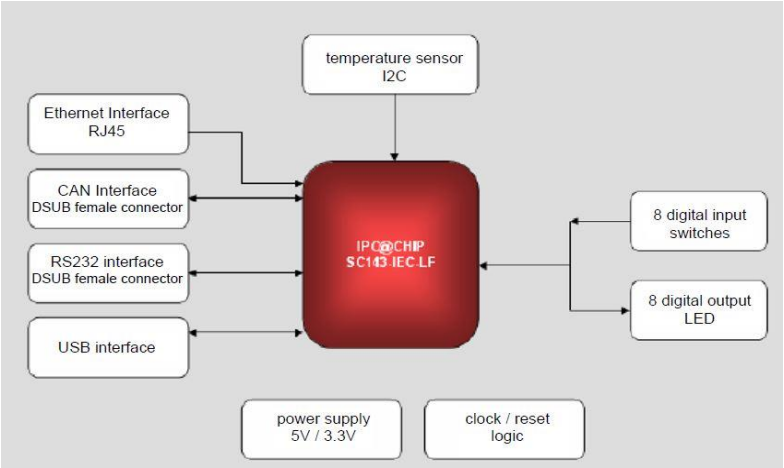


Figura 36 - Placa de desenvolvimento EB60 da BECK [46]

Por forma a ciar uma plataforma de desenvolvimento estável, duradoura e independente dos sistemas operativos atuais, foi tomada a decisão de que todo o desenvolvimento a partir deste ponto decorresse utilizando um sistema operativo virtualizado, recorrendo a uma máquina (computador) virtual com sistema operativo *Windows XP*.

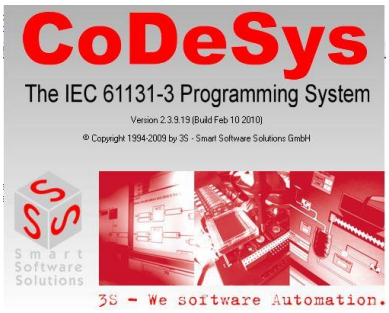
A placa EB60 (Figura 36) pode ser alimentada por um transformador de 5V DC ou diretamente através da ficha mini-USB presente na placa. Após a sua alimentação o passo

seguinte passa por ligar a placa ao computador para que possamos parametrizar o sistema. Devemos ter em conta nesta fase que a placa possui já um sistema operativo de tempo real proprietário (RTOS) da BECK e um RT (*Run Time* CoDeSys da 3S) tendo por isso já capacidades de comunicações avançadas e que nos permitem inclusivamente escolher uma de três formas distintas de comunicação com o computador, USB, *Serial* ou *Ethernet*.

Para facilitar o desenvolvimento de aplicações para a sua família de controladores a BECK desenvolveu um conjunto de ferramentas que vêm incluídas já com a sua placa de desenvolvimento EK61. Vamos discutir nos próximos passos em detalhe esse conjunto de ferramentas, as quais podemos consultar na Figura 37, bem como a versão de *software* usada neste trabalho.

a)CHIPTOOL V6.1.3.6

b)CoDeSys@CHIP SDK V1.32.0



c)CoDeSys V2.3.9.19

d)Paradigm BECK Compiler Bulld 12/2010

Figura 37 - Ferramentas de desenvolvimento fornecidas pela BECK

Cada uma destas ferramentas vai ser necessária em diferentes estágios do nosso desenvolvimento. O CHIPTOOL é uma ferramenta que nos permite estabelecer a comunicação com o processador (SC143-IEC-LF) e desta forma podermos realizar por exemplo atualizações ao sistema operativo, aceder ao sistema de ficheiros e realizar operações como por exemplo a de modificação, colocação ou remoção de ficheiros. Por exemplo permite que possamos colocar os ficheiros necessários para parametrizar o sistema segundo as nossas necessidades.

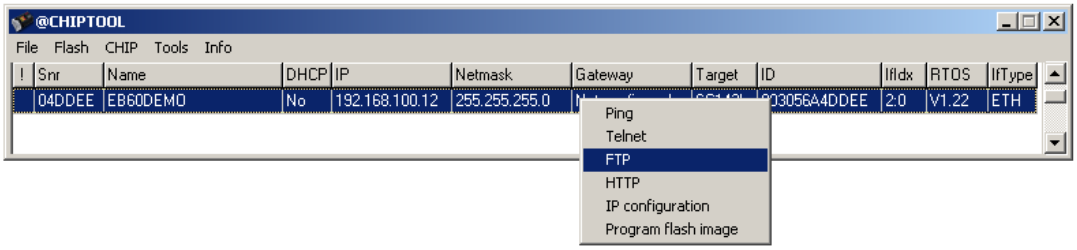


Figura 38 - Imagem da ferramenta CHIPTOOL - BECK

Como foi já referido anteriormente, a comunicação com a placa EB60 pode ser realizada através de três formas, USB, *Serial*, ou por *Ethernet*. Por ser o tipo de transferência de dados mais rápida para transferência de ficheiros pelo protocolo FTP vamos optar pela ligação *Ethernet*. A comunicação *Ethernet* é baseada no protocolo TCP/IP e é por isso importante realçar nesta fase que caso não tenhamos nenhum servidor de DHCP instalado, vamos ter de definir um IP fixo da mesma gama de endereços configurada no processador que se encontra na placa de desenvolvimento, que por defeito vem configurado para se ligar a uma rede que lhe atribua o IP. Assim sendo, vamos ter de definir um IP fixo quer na plataforma de desenvolvimento, quer na placa de rede do computador. Para o processador da BECK fazer isso através do menu "*IP configuration*" que se pode observar na Figura 39, vamos configurar com o seguinte endereço IP "192.168.100.12" e preencher os restantes campos da forma que se pode ver na ilustração da mesma figura.

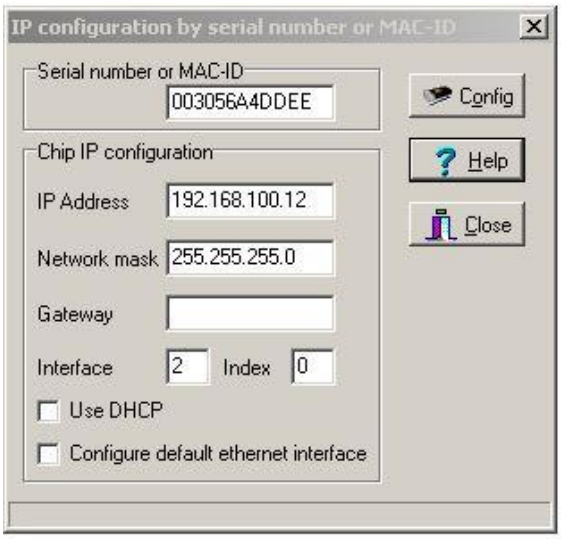


Figura 39 - Configuração IP da placa EB60

Para garantirmos sucesso na comunicação necessitamos agora de configurar o IP da placa de rede da máquina virtual para a mesma gama, ficando definido para o seguinte IP fixo "192.168.100.20" e colocamos no campo *Netmask* o valor "255.255.255.0".

Ao efetuarmos a ligação através de um cabo *Ethernet* da placa de rede do nosso computador à plataforma de desenvolvimento, esta aplicação vai pesquisar toda a rede ciclicamente tentando encontrar um processador da família IPC@CHIP da BECK, e em caso de sucesso vai disponibilizar essa informação para o utilizador como podemos observar na Figura 38. Mais à frente neste documento, assim que seja necessário, vamos detalhar o acesso FTP, pois nesta fase de preparação para testar a ligação basta realizarmos uma simples execução do comando "*ping*" (ping 192.168.100.12) e que pode ser executado diretamente da aplicação CHIPTOOL.

O CoDeSys CHIP SDK Platform Builder é a próxima aplicação de que vamos necessitar, ela é também uma importante ferramenta da BECK. É esta ferramenta que vai possibilitar criar os ficheiros e pastas com a estrutura básica contendo o código fonte que vai permitir criar o TSP (*Target Support Package*) e o RTS (*Run Time System*).

- O TSP é um conjunto de ficheiros que vai descrever o *hardware* e *software* do nosso produto, e que vai posteriormente ser instalado no computador para que ao criarmos aplicações com o CoDeSys, este conheça o dispositivo para o qual o código que se vai gerar seja o adequado e que contenha o mapeamento correto dos seus I/O.

- O RTS é um programa que é executado no processador SC143-IEC-LF e que é responsável por receber e interpretar a aplicação IEC proveniente do IDE do CoDeSys e realizar a abstração necessária com o nosso *hardware* e todos os dispositivos de entrada e saída do PLC.

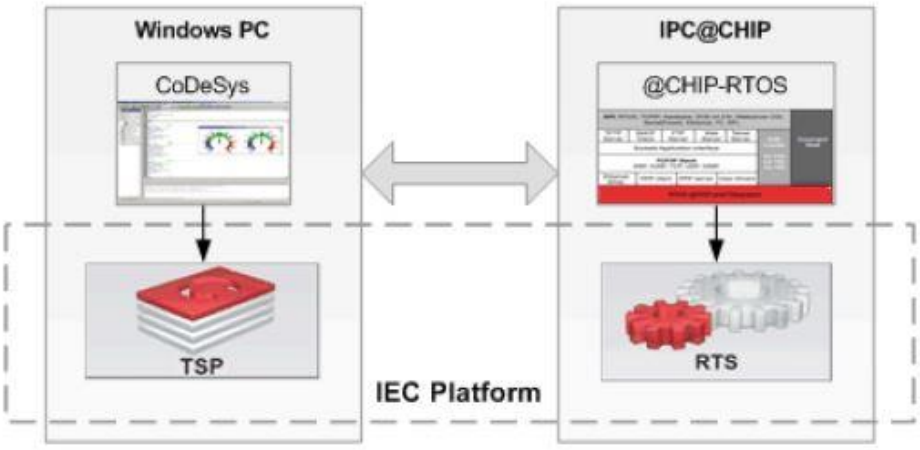


Figura 40 - Plataforma IEC [46]

É graças ao uso desta ferramenta gráfica que vai ser possível adaptar as necessidades do produto que pretendemos desenvolver e possibilitar que nele sejam executadas aplicações compatíveis com a norma IEC 61131-3. Nos passos seguintes é exposto passo por passo como configurar o *SDK Platform Builder* de acordo com os parâmetros necessários para cumprir com os objetivos deste trabalho.

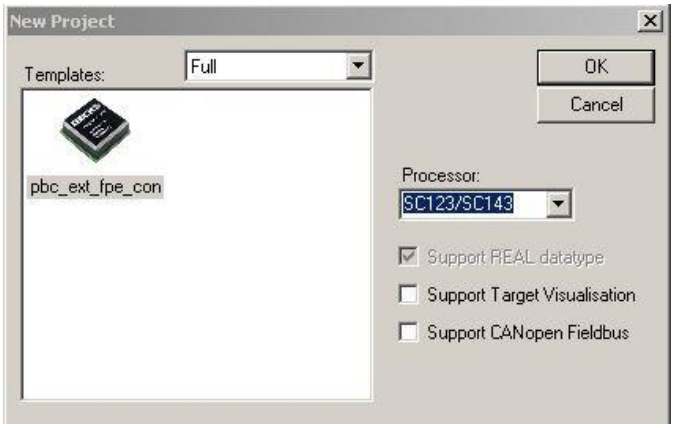


Figura 41 - *SDK Platform Builder*

Para começar a usar este SDK da BECK, o primeiro passo consiste emselecionarmos o controlador com o qual vamos construir o nosso sistema, no nosso caso é o IPC@CHIP® SC143-IEC. Podemos observar esse passo na Figura 41.

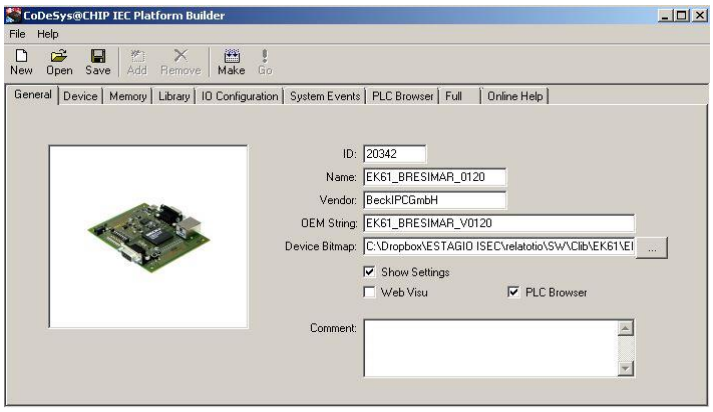


Figura 42 - *SDK Platform Builder - General*

Na Figura 42 podemos ver a janela principal da aplicação. É aí que vamos introduzir os dados que vão identificar a nossa plataforma/produto e que, como vamos ter oportunidade de ver mais à frente neste documento, vão aparecer no IDE do CoDeSys quando selecionarmos a plataforma para a qual vamos escrever o programa.

- **ID** - Neste campo vamos colocar a identificação que deve ser única na criação da plataforma. A BECK disponibiliza a gama de ID de 20300 até 20399 o que dá 100 possíveis entradas, que foi criada a pensar na fase de desenvolvimento. Existe também outra gama de 1000 entradas (25300-26299) disponibilizada pela BECK, no entanto, caso seja pretendido, este ID pode ser pedido à 3S.

- ***Name*** - Os dados aqui colocados vão ser usados como identificação para seleção futura do nosso dispositivo e deve por isso também ser única dentro dos produtos por nós já gerados, pois só desta forma é possível evitarmos erros na seleção da plataforma para a qual vamos gerar o nosso código.

- ***Vendor*** - É um conjunto de caracteres (*string*) onde normalmente é colocado o nome da marca que o desenvolve e serve apenas como indicação para o SDK *Platform Builder*.

- ***OEM String*** -. É uma *string* que vai aparecer no RTS gerado pelo SDK e a qual podemos por exemplo usar para colocar informação tal como a marca, o produto e até a versão do mesmo.

- ***Device Bitmap*** - Introduzimos aqui o caminho para uma fotografia a qual permita visualmente identificar o nosso produto. A imagem aqui selecionada deverá ter como dimensões 125 por 125 pixel para uma ótima visualização. Esta imagem também vai ser inserida no TSP e desta forma estar acessível no IDE do CoDeSys.

- ***Web Visu*** - Se for nosso propósito criar um *webserver* no nosso produto e disponibilizar o estado das nossas variáveis do programa IEC no exterior em um qualquer *Browser* que suporte essas funções, teremos de ativar esta opção, no nosso caso vamos deixar desativada pois não necessitamos dela.

- ***PLC Brower*** - Permite ativar ou desativar a possibilidade de o *PLC Browser* aparecer no IDE CoDeSys no separador recursos (*Resources*).

- ***Comment*** - É um campo onde podemos colocar mais alguma informação que se considere importante. Esta informação vai ficar apenas disponível para o *SDK Platform Builder* e não vai aparecer nem no RTS nem no TSP.

No separador com o nome de "*Device*" (Figura 43), as opções assinaladas vão ter repercussão tanto na criação do RTS como do TSP. É aqui que vamos configurar por exemplo a forma de comunicação entre a nossa placa de desenvolvimento e o IDE CoDeSys. Apenas os itens ativos poderão ser configurados, uma vez que os restantes já foram selecionados anteriormente.

- ***CoDeSys Comms*** - Aqui podemos definir a forma de comunicação por nós pretendida, para que o nosso IDE CoDeSys comunique com o RTS no dispositivo. Podemos ativar a comunicação por TCP/IP ou RS232 ou até mesmo ambas. No nosso caso vamos optar por ativar a comunicação TCP/IP.

- ***Working Path*** - Este é o caminho onde vão ficar localizados os ficheiros de parametrização do RTS, em conjunto com o campo "*Working Drive*".

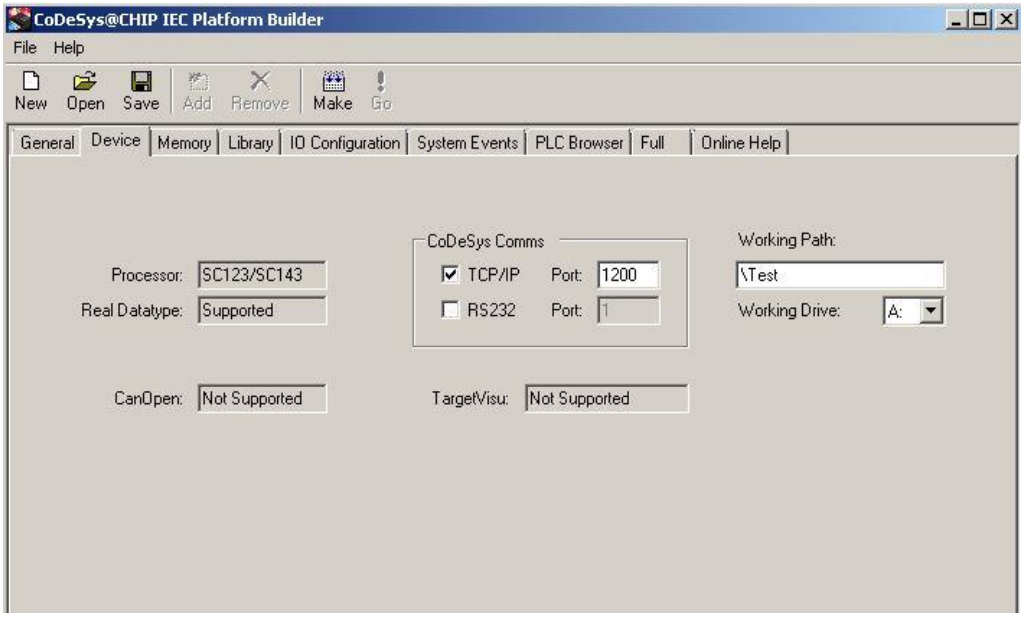


Figura 43 - *SDK Platform Builder - Device*

- ***Working Drive*** - É a drive onde vamos colocar os ficheiros de parametrização do nosso RTS. Caso necessitemos de colocar mais dados ou programas que ocupem mais espaço do que aquele que está já disponível no módulo, será necessário usar um espaço externo, como por exemplo um suporte do tipo "*Compact Flash*" (CF), e nesse caso selecionar a drive B:. No nosso caso isso não é necessário e podemos simplesmente usar a memória interna e selecionar a drive A:.

O próximo separador é o da memória ("*Memory*", Figura 44), aqui podemos reservar a memória de que vamos necessitar para as diferentes áreas da nossa aplicação IEC, estes valores vão ser usados para criar tanto o RTS como o TSP, de referir que os valores aqui inseridos são em *bytes*. Vamos analisar em maior detalhe alguns dos dados mais importantes.

- ***Size of Code Area*** - Este é o espaço de memória que vamos reservar para o código que vamos desenvolver na aplicação IEC, ou seja, todo o código que vai ser escrito e posteriormente compilado pelo CoDeSys vai ter de ocupar um espaço de 40000 *bytes* (valor em hexadecimal), o que dá um valor de 256 KBytes.

- ***Size of Global Area*** - Área de memória reservada para as variáveis globais e locais de um programa (PRG). É também neste espaço onde vão ficar alojadas as funções bloco (FB) necessárias à aplicação IEC, para o nosso caso vamos reservar um espaço de C800, o que dá um total de 50 KBytes.

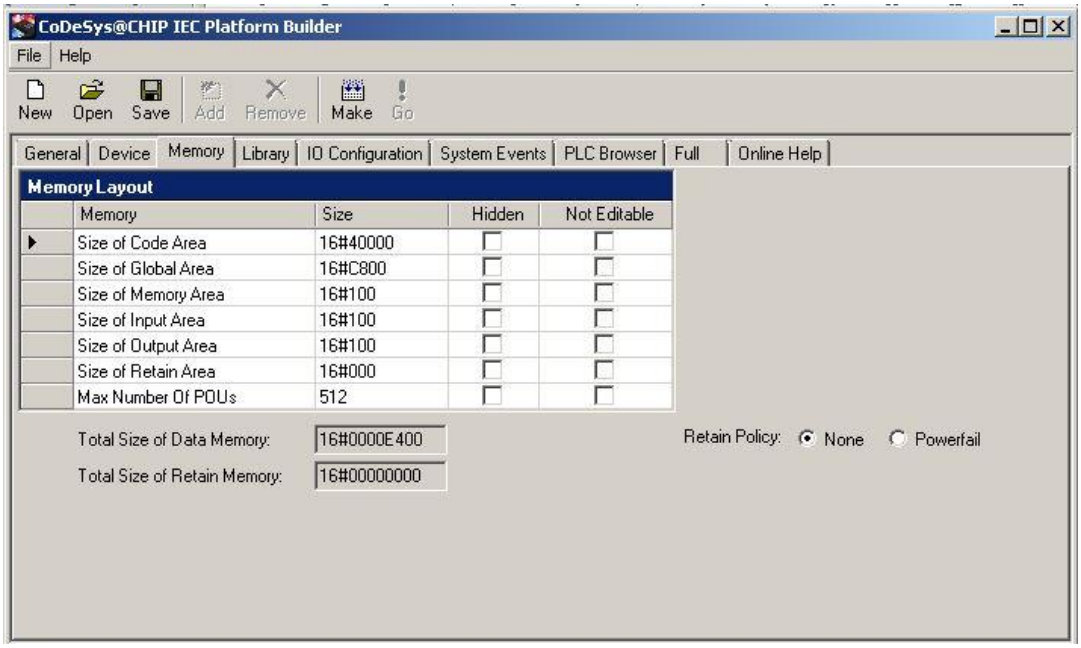


Figura 44 - *SDK Platform Builder - Memory*

- ***Size of Input Area*** - Este vai ser o espaço necessário reservar para mapear todas as variáveis de entrada na nossa aplicação, para o nosso caso vamos reservar um espaço de 100, o que dá um total de 256 *bytes*.

- ***Size of Output Area*** - Semelhante ao anterior mas desta vez reservado às variáveis de saída, para o nosso caso vamos reservar um espaço de 100, o que dá um total de 256 *bytes*.

- ***Size of Retain Area*** - Em algumas aplicações é necessário ter variáveis retentivas, que são aquelas que mantêm o seu valor mesmo em caso de perda de energia. Caso seja o pretendido podemos aqui definir o tamanho da área de memória reservada para esse efeito.

- ***Max number of POUs*** - Cada POU (*Program Organizations Units*) necessita de 12 *bytes* de espaço de memória necessária para a sua declaração, esse espaço é aqui reservado ao especificarmos o número máximo de POUs que a nossa aplicação IEC pode conter.

- ***Total Size of Data Memory*** - Este campo mostra o somatório do espaço de memória por nós reservado nos campos acima com exceção da área de código (*Size of Global Area + Size of Input Area + Size of Output Area*).

- ***Total Size of Retain Memory*** - No nosso caso não foi reservado nenhum espaço de memória para memória não volátil ou retentiva. Caso o tivéssemos feito o seu valor total apareceria neste campo. De realçar que para a versão FULL RTS que estamos a usar este valor não pode ser superior a 10000 *bytes* (em hexadecimal).

Após definirmos e reservamos a memória disponível para o nosso RTS, vamos para o separador seguinte, que é o "*Library*" (Biblioteca) (Figura 45) e nele podemos remover ou inserir bibliotecas por nós desenvolvidas e que vão ser inseridas no TSP e que poderão ser utilizadas aquando da compilação do código no CoDeSys.

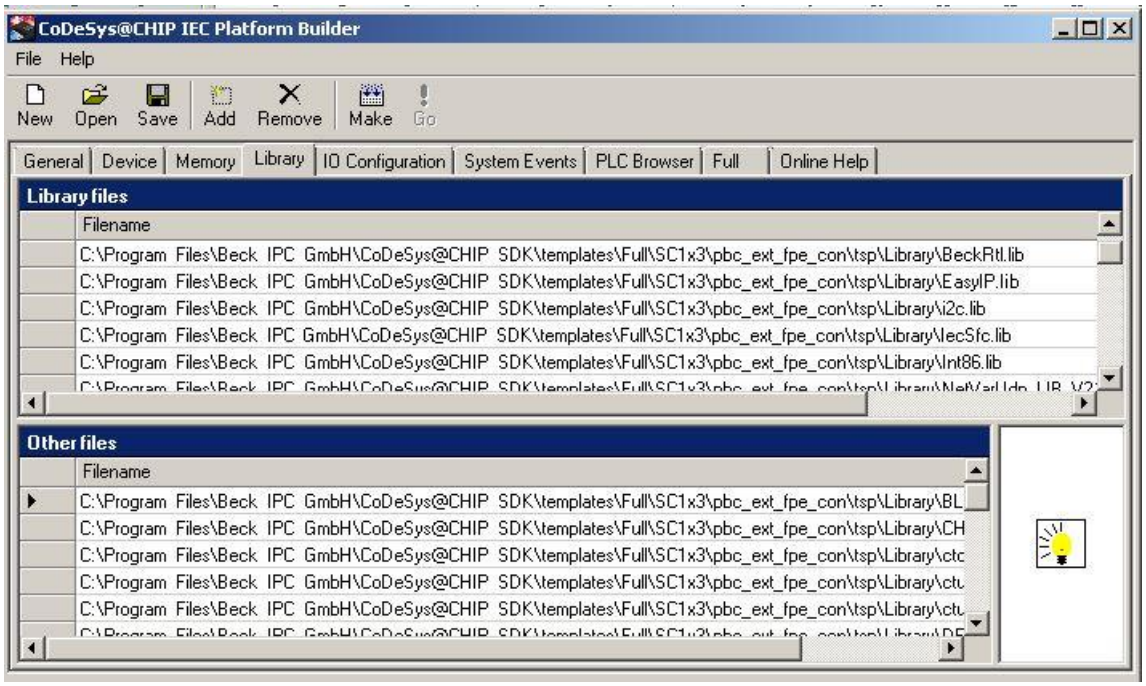


Figura 45 - *SDK Platform Builder - Library*

Este separador tem duas tabelas, na primeira indicamos o caminho da biblioteca no formato *.LIB e na segunda tabela indicamos o caminho de uma imagem no formato *.BMP para a qual vamos associar a um determinado POU nas bibliotecas inseridas na tabela anteriormente descrita.

Na primeira tabela, para além de indicarmos qual a biblioteca que pretendemos inserir, temos também a possibilidade de, para cada biblioteca, selecionar ou ativar se queremos que essa biblioteca esteja disponível por defeito assim que selecionarmos um novo projeto no CodeSys. Caso não ativemos essa opção, e pretendamos utilizar essa mesma biblioteca, temos de no CoDeSys a ativar para esse projeto manualmente. Existe ainda outra opção que consiste na ativação do campo "*Hidden*" e que em caso de seleção esconde a utilização dessa biblioteca no separador "*Resources*" no IDE CoDeSys.

O próximo separador é onde vamos configurar as nossas variáveis IEC que vão ter ligação com o mundo exterior, ou seja, é aqui que vamos definir as nossas entradas digitais ou analógicas, e as nossas saídas digitais ou analógicas, tal como podemos ver na Figura 46.

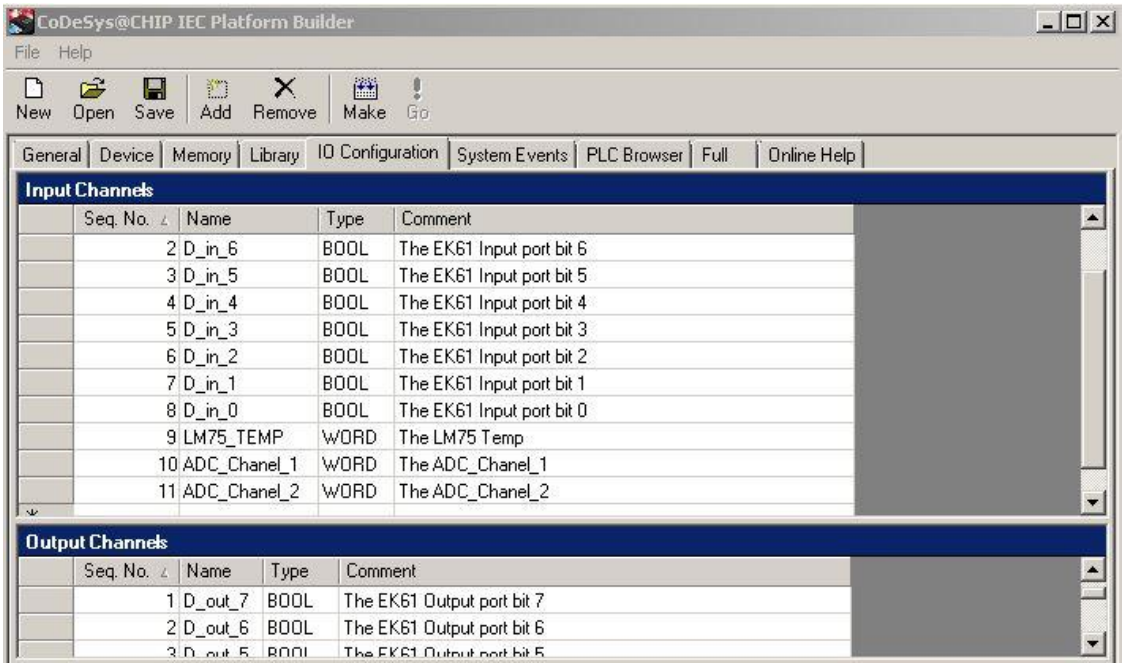


Figura 46 - *SDK Platform Builder - IO Configuration*

Aqui temos duas tabelas para preencher, uma com variáveis de entrada e outra com as variáveis de saída. Podemos atribuir nomes simbólicos a cada uma das variáveis, nomes esses que quando instalarmos o TSP vão estar disponíveis para de uma forma mais amigável podermos identificar as variáveis aqui definidas no IDE do CoDeSys.

É no campo com o nome de "*Name*" que vamos atribuir o nome à nossa variável, e esse nome tem de seguir algumas regras. Tem de ser único para não criar ambiguidades e segundo a norma IEC 61131-3, este é um campo onde não são tidas em conta diferenças entre caracteres maiúsculos e minúsculos. Pode ser constituído por letras, números e ou *underscores* ("_"). É também necessário ter em conta que o primeiro caracter não pode ser um dígito.

Para o campo "*Type*" vamos selecionar o tipo de variável que pretendemos e o seu tamanho em *bits*. Podemos ter vários tipos de variáveis, que segundo a norma podem ser representadas da seguinte forma:

- **Bit String** - representam valores ON ou OFF
  - **BOOL** - É uma variável de um bit apenas e que pode assumir o valor lógico de 1 ou 0
  - **BYTE** - É uma sucessão de 8 *bits*.

  - **WORD** - É uma sucessão de 16 *bits*
  - **DWORD** - É uma sucessão de 32 *bits*
- **Inteiros**
  - **SINT** - 8 *bits* (sem sinal)
  - **INT** - 16 *bits* (com sinal)
  - **DINT** - 32 *bits* (com sinal)
  - **USINT** - 8 *bits* (sem sinal)
  - **UINT** - 16 *bits* (sem sinal)
  - **UDINT**. 32 *bits* (sem sinal)
- **Real**
  - **REAL** - 32 *bits*
  - **LREAL** - 64 *bits*

No campo "*Comment*" vamos introduzir uma pequena descrição da nossa variável. Essa descrição aqui inserida vai também aparecer no CoDeSys sob a forma de comentário, e permite que o programador do sistema mais facilmente consiga distinguir as variáveis.

Neste separador vamos por isso inserir os nossos requisitos detalhados no capítulo anterior, em termos de variáveis de entrada e de saída. Assim sendo vamos configurar o SDK da forma que é visível na Figura 46 e que se passa a detalhar:

- 8 Entradas Digitais - Cada variável de entrada é representada por um *bit* e desta forma, para cada entrada digital, a variável é do tipo BOOL e vai ser inserida na tabela "*Input Channels*".

- 8 Saídas Digitais - Cada variável de saída é representada por um *bit* e desta forma, para cada saída digital, a variável é do tipo BOOL e vai ser inserida na tabela "*Output Channels*".

- 2 Entradas Analógicas - Para as entradas analógicas, tanto para aquela que representa o sinal de entrada de 0-20 mA, como para aquela de 0-10 V, vamos usar um circuito que vai converter o valor de analógico para digital com uma resolução de 12 *bits*. Assim sendo vamos atribuir a estas duas entradas uma variável de 16 *bits*, neste caso uma WORD.

O próximo separador do SDK tem o nome de "*System Events*" e podemos utilizar esse campo para criar eventos que por exemplo durante a execução da aplicação IEC podem executar determinados POUs. Esses eventos vão estar disponíveis no TSP e poderão ser consultados no CoDeSys no separador "*Resources*". Tendo em conta os requisitos este é um campo que não vai ser usado.

O próximo separador que vai ser necessário configurar é o de "*Full*" (Figura 47).

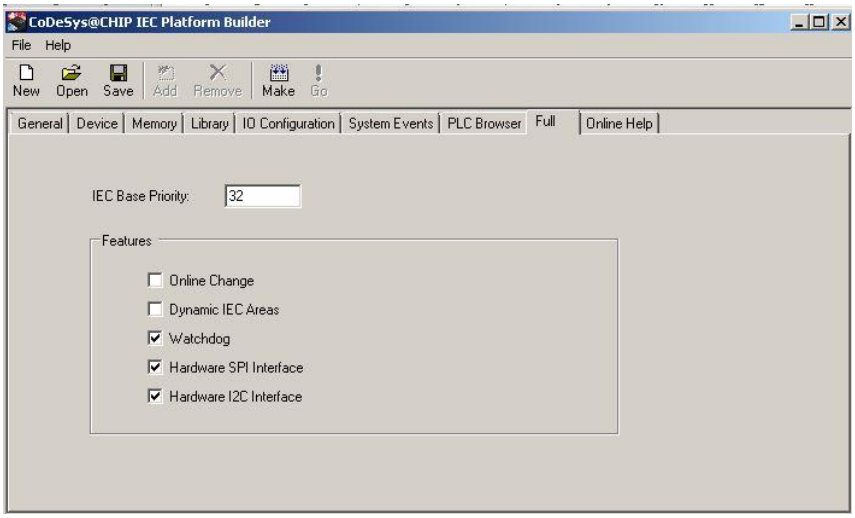


Figura 47 - *SDK Platfotm Builder – Full*

Aqui podemos alterar e ativar os seguintes campos:

- ***IEC Base Priority*** - Neste campo vamos inserir a base de prioridades pretendidas, que no CoDeSys vai de 0 a 31, perfazendo um total de 32. De notar que uma tarefa com prioridade 0 é a mais prioritária e a menos prioritária é a tarefa com prioridade 31. Esta gama de prioridades pode ser utilizada no CoDeSys no separador "*Tasks Configuration*" para atribuir diferentes prioridades a cada tarefa e programa IEC.
- ***Online Change*** - Se pretendemos ativar esta funcionalidade vamos conseguir realizar alterações nas variáveis em tempo real na aplicação. No entanto, esta opção implica uma maior necessidade de memória.
- ***Dynamic IEC Areas*** - Ativando esta opção estamos a dar a possibilidade do RTS realizar a alocação de memória de código de forma dinâmica, caso contrário ela é feita de forma estática. De salientar que na versão atual, a BECK não recomenda a ativação desta opção porque em alguns casos pode levar a um erro na reserva dinâmica de memória.
- ***Watchdog*** - Permite que possamos atribuir diferentes tempos de *watchdog* por cada tarefa no CodeSys.
- ***Hardware SPI Interface*** - O SPI é um protocolo de comunicação síncrono que está disponível no processador IPC@CHIP® SC143-IEC, ao selecionarmos esta opção vamos ativar o canal SPI por *hardware*. Caso não o façamos podemos, como alternativa, usar uma comunicação SPI mas desta forma será por *software*, o que a torna mais lenta.
- ***Hardware I2C Interface*** - Semelhante à opção anterior, mas desta forma relativa ao protocolo de comunicação I2C, que também, tal como o SPI, pode ser implementado por *software* caso esta opção não seja ativada, mas também com as mesmas limitações do SPI por *software*.

Uma vez que a grande maioria dos periféricos utiliza estes dois protocolos de comunicação de proximidade, vamos deixar estes canais ativos e assim tirar todas as vantagens da sua utilização. A configuração utilizada é a que se pode observar na Figura 47, ativando o *watchdog*, o canal SPI, o canal I2C e atribuindo a prioridade 32 à aplicação IEC.

O último separador é o de "*Help*". Os dados aqui inseridos vão fazer parte do TSP e vão estar disponíveis após a instalação do TSP no CoDeSys na opção "*Target Help*". Neste separador podemos inserir ficheiros de ajuda que deverá conter informação útil para o programador desenvolver a aplicação. Esta opção não vai ser usada e vamos deixar desativada.

Neste momento temos todos os dados necessários inseridos na plataforma desenvolvida pela BECK (*SDK Platform Builder*), para gerar os ficheiros necessários à criação do TSP e do RTS. O passo seguinte é a criação das pastas e ficheiros. Para isso é necessário selecionar a opção "*RUN*" no SDK.

Após a execução do comando *RUN* e caso o SDK não tenha reportado nenhum erro, ele vai criar uma pasta com o mesmo nome com que gravámos o ficheiro de configuração, no nosso caso o nome da pasta ficou "EK61_BRESIMAR", pois foi este nome que foi escolhido no momento da gravação (Figura 48).

Figura 48 - Pastas criadas pelo SDK

Figura 49 - Pastas RTS e TSP

Na Figura 48 podemos observar o ficheiro XML que foi criado quando salvamos a configuração, a imagem que foi selecionada para identificação do produto e que foi colocada no SDK, e a pasta gerada onde se encontra o TSP e o RTS (EK61_BRESIMAR).

Na Figura 49 podemos confirmar a localização das pastas geradas pelo SDK que se encontram dentro da pasta "EK61 BRESIMAR":

- RTS - Run Time System
- TSP - Target Support Package

4.2 - Instalação do TSP (*Target Support Package*)

Com este ponto pretende-se descrever em detalhe a instalação do TSP criado anteriormente e todos os passos necessários para concluir a instalação. Vamos para isso observar a pasta TSP criada pelo SDK e os ficheiros nela contidos.

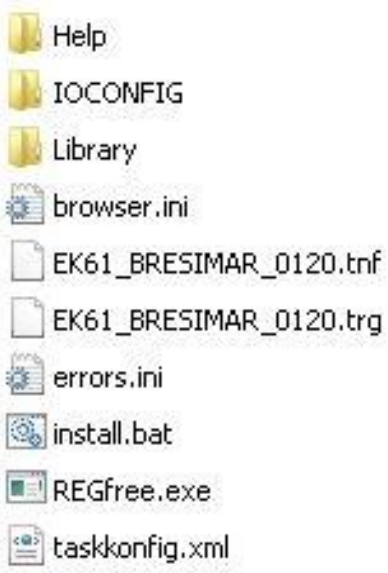


Figura 50 - Estrutura de ficheiros do TSP

Na pasta TSP está um conjunto de ficheiros criados pelo SDK e neles encontra-se reunida toda a informação necessária para descrever o nosso "*hardware*". Aqui podemos ainda efetuar algumas alterações. No entanto, temos de ter especial atenção pois existe uma grande dependência entre o RTS e o TSP, e caso decidamos editar manualmente algum destes ficheiros, temos que também realizar a equivalente alteração nos ficheiros do RTS. Aqui podemos destacar a pasta *IOCONFIG* (Figura 50) que contém todas as variáveis por nós criadas no SDK, e que permitem realizar a ponte e estabelecer a relação entre as variáveis do programa com os I/O físicos presentes no nosso "*hardware*".

Para a instalação do TSP existem duas formas distintas de a realizarmos. Podemos recorrer a uma ferramenta criada pela companhia 3S criadora do CoDeSys, que se chama "*InstallTarget*" (Figura 51), e que é instalada automaticamente assim que instalamos o CoDeSys no nosso computador. Para além de esta ferramenta possibilitar a instalação do TSP possibilita também verificar outros TSP que já se encontrem instalados.

O segundo método de instalação do TSP passa por usar uma ferramenta desenvolvida pela BECK. De notar que a BECK recomenda inclusivamente a utilização deste método pois segundo a documentação associada, poderão ocorrer falhas na cópia das pastas para o sistema operativo se usarmos a ferramenta da 3S-Smart.

Desta forma o método selecionado para a instalação foi o segundo método, e a ferramenta da 3S foi usada para avaliar se o TSP tinha sido instalado com sucesso ou insucesso.

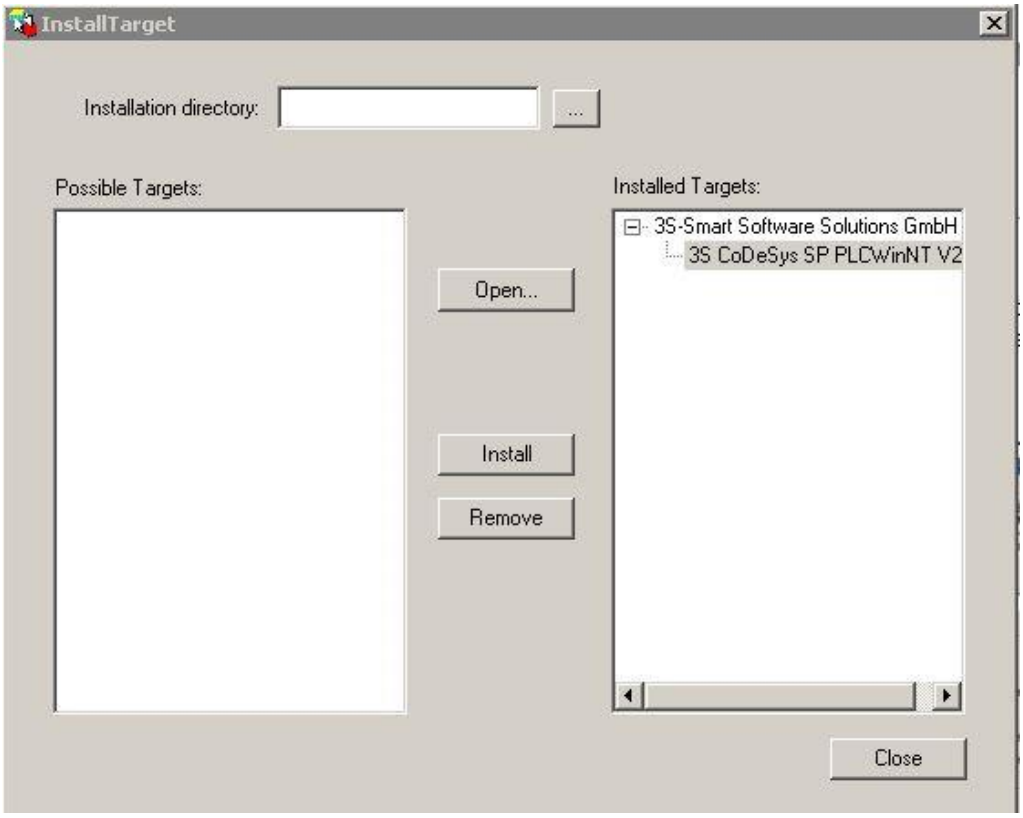


Figura 51 - Ferramenta da 3S - *Install Target da 3S*

- **Passo 1** - Como a instalação do TSP necessita de recorrer à linha de comando do *Windows*, vamos iniciar a linha de comandos em "Iniciar > cmd, colocar a localização da pasta em que temos a pasta com o TSP, como se pode observar na Figura 52.

Figura 52 - Linha de comandos

- **Passo 2** - Executar o ficheiro *install.bat* que se encontra na raiz da pasta TSP. O resultado deverá ser o que se observa na Figura 53. Aí podemos observar que a ferramenta da BECK para instalação do TSP realiza uma série de verificações antes

de simplesmente instalar os ficheiros. É importante neste passo garantir que não temos já um *target* instalado com o mesmo nome, pois caso isso aconteça a instalação vai dar erro. A forma de ultrapassar esse erro é usarmos a ferramenta da CoDeSys (*InstallTarget*) para remoção desse *target* ou então executarmos o ficheiro *install.bat* com a opção F (*install* F) e nesse caso vamos forçar a instalação deste *target* mesmo que já tenhamos um com o mesmo nome.

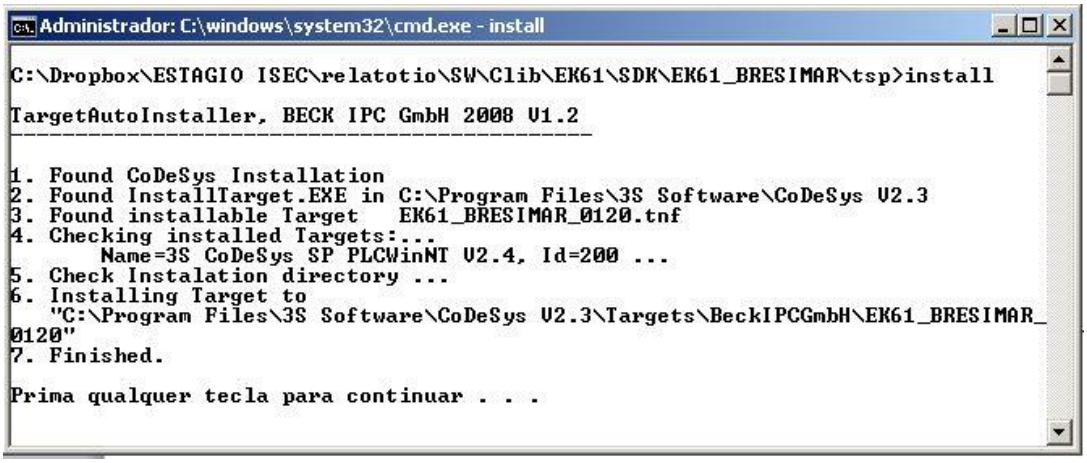


Figura 53 - Linha de comandos – *install.bat*

- **Passo 3** - Verificar se o TSP se encontra corretamente instalado. Para isso vamos executar a aplicação *InstallTarget* (da 3S-Smart) e verificar se adicionamos corretamente o nosso TSP ao CoDeSys no Passo 2. Tal como podemos observar na Figura 54, o TSP foi corretamente adicionado ao CoDeSys, onde o podemos observar com o nome "EK61_BRESIMAR_0120". Vai ser este o nome que daqui para a frente vamos usar no CoDeSys para escrever os programas para a placa de desenvolvimento.

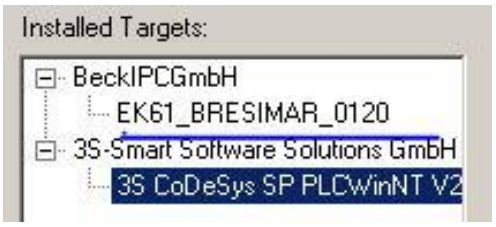


Figura 54 - *InstallTarget* EK61

4.3 - Criação do RTS (*Run Time System*)

Agora que realizámos com sucesso a instalação do TSP no ambiente de desenvolvimento CoDeSys necessitamos de seguida de preparar o RTS (*Run Time System*). Este ponto do documento tem como objetivo descrever todos os passos necessários para a criação do ficheiro RTS e da sua instalação na placa de desenvolvimento.

Vamos começar por analisar os ficheiros gerados pelo SDK dentro da pasta RTS (Figura 55). Dentro da pasta RTS, encontramos cinco pastas e três ficheiros, sendo que o *myrts.pdl* é um ficheiro de projeto do compilador *Paradigm C*, e que ao ser executado carrega já as configurações do projeto, incluído o ficheiro *myrts.c*.

Figura 55 - Estrutura de ficheiros do RTS

De todos os ficheiros que aqui se encontram apenas vamos necessitar nesta fase de efetuar alterações a um deles. Para adaptar o RTS segundo os nossos requisitos, o ficheiro que necessita de ser adaptado é o *myrts.c*.

Lib - Contém as bibliotecas C necessárias durante o processo de compilação, e onde podemos também adicionar as nossas.

Intermediate - Durante o processo de compilação vão ser gerados ficheiros temporários e é nestas pastas que vão ser colocados.

Include - Contém os ficheiros com extensão *.h (*header file*).

bin - É nesta pasta onde vai ser criado o ficheiro executável "myrts.exe".

Nesta fase vamos necessitar de indicar quais os endereços das nossas variáveis. Já sabemos que as mesmas vão aparecer no IDE como ficaram definidas na Figura 46 (D _in_0, D_in_1, D_in_2, etc.), mas agora necessitamos de as fazer corresponder aos I/O da nossa placa de desenvolvimento. Para isso e para efeitos de testes, vamos configurar oito entradas digitais, oito saídas digitais e vamos também já testar nesta fase as comunicações I2C, necessitando para isso de selecionar um periférico com essas características.

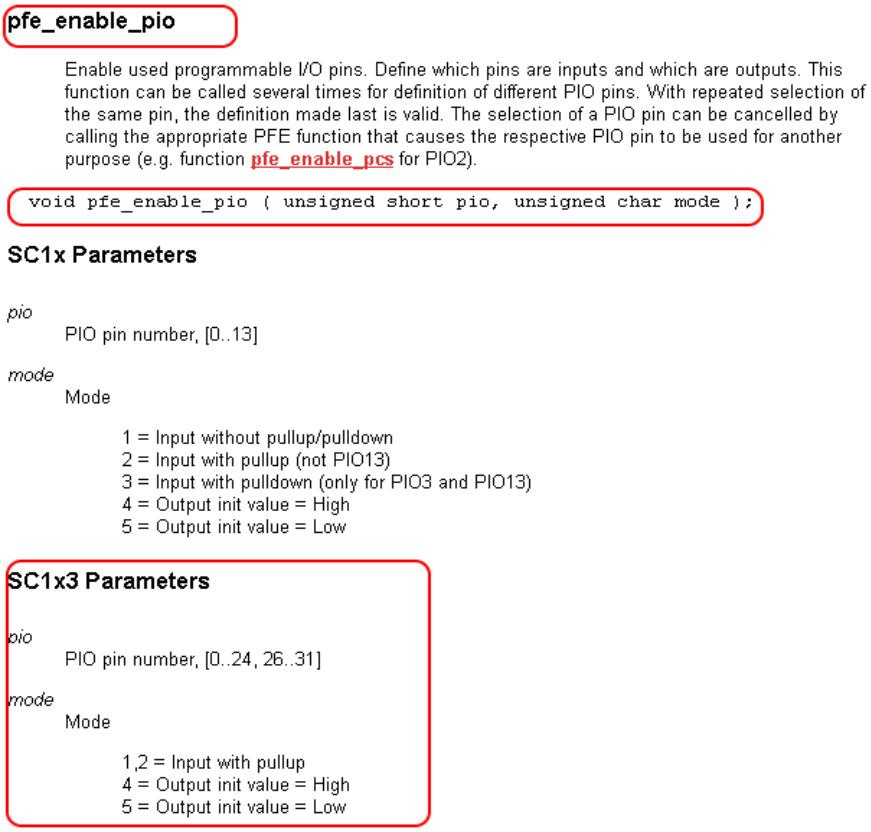
Tabela 6 - Mapeamento SC143-IEC

Nome da variável	Endereço interno no SC143-IEC	I/O
D_in_0	PIO pin 0	Entrada Digital
D_in_1	PIO pin 1	Entrada Digital
D_in_2	PIO pin 2	Entrada Digital
D_in_3	PIO pin 3	Entrada Digital
D_in_4	PIO pin 4	Entrada Digital
D_in_5	PIO pin 5	Entrada Digital
D_in_6	PIO pin 6	Entrada Digital
D_in_7	PIO pin 7	Entrada Digital
D_out_1	PIO pin 16	Saída Digital
D_out_2	PIO pin 17	Saída Digital
D_out_3	PIO pin 18	Saída Digital
D_out_4	PIO pin 19	Saída Digital
D_out_5	PIO pin 20	Saída Digital
D_out_6	PIO pin 21	Saída Digital
D_out_7	PIO pin 22	Saída Digital
	PIO pin 31	I2C CLK
	PIO pin 13	I2C DTA

A documentação com toda a informação referente ao controlador da BECK SC143-IEC, pode ser encontrada na sua página de Internet. Desta forma teremos de iniciar o nosso processador com os dados presentes na Tabela 6, e para isso teremos de editar no ficheiro *myrts.c* e a função *RHIIOInit*. Esta função é apenas chamada no momento em que a aplicação é iniciada. Nesta fase é também necessário conhecer a biblioteca que a BECK disponibiliza para que possamos criar o nosso código. Essa biblioteca tem o nome de "*@CHIP-RTOS C Library*" e também se encontra disponível na página do fabricante. Nesta fase vamos usar apenas duas das suas funções. A primeira é a de inicialização dos I/O e a última de inicialização do barramento I2C. A primeira então a ser utilizada será a *pfe_enable_pio*, e segundo a documentação (Figura 56) a mesma será utilizada da seguinte forma que podemos observar na Figura 57.

Para a utilização do barramento I2C e para realizar a sua inicialização basta chamar a função de *I2C_init()*, porque estamos a usar os I/O dedicados para esta função e desta forma não temos de realizar configurações adicionais. Agora que temos todo o nosso *hardware* inicializado, necessitamos de implementar as funções de leitura e de escrita dos nossos I/O. Para isso vamos utilizar as funções de *RHIIOReadInput* e de *RHIIOWriteOutput.* Estas são funções que são chamadas de uma forma cíclica, e desta forma garantimos que por cada ciclo de execução do nosso programa as nossas variáveis de entrada são lidas, e que da mesma forma as variáveis de saída também são atualizadas.

Para esta fase de testes as implementações até aqui descritas no ficheiro *myrts.c*, serão suficientes para que possamos compilar o projeto e passar à fase seguinte. Após a compilação com sucesso, é criado um ficheiro executável com o nome de *myrts.exe*.



pfe_enable_pio

Enable used programmable I/O pins. Define which pins are inputs and which are outputs. This function can be called several times for definition of different PIO pins. With repeated selection of the same pin, the definition made last is valid. The selection of a PIO pin can be cancelled by calling the appropriate PFE function that causes the respective PIO pin to be used for another purpose (e.g. function **pfe_enable_pcs** for PIO2).

```
void pfe_enable_pio ( unsigned short pio, unsigned char mode );
```

SC1x Parameters

pio
PIO pin number, [0..13]

mode
Mode

1 = Input without pullup/pulldown
2 = Input with pullup (not PIO13)
3 = Input with pulldown (only for PIO3 and PIO13)
4 = Output init value = High
5 = Output init value = Low

SC1x3 Parameters

pio
PIO pin number, [0..24, 26..31]

mode
Mode

1,2 = Input with pullup
4 = Output init value = High
5 = Output init value = Low

Figura 56 - Protótipo da função *pfe_enable_pio*

Este ficheiro contém todas as funções por nós escritas. Para que o mesmo seja executado pelo nosso processador necessitamos novamente de aceder ao programa que permite estabelecer a ligação com a placa de desenvolvimento (CHIPTOOL, já descrito acima), da mesma forma que aparece na Figura 38. Selecionamos o acesso por FTP e preenchemos os campos *User* e *Password*, caso não tenhamos alterado nada ainda na placa de desenvolvimento, devemos preencher os dois campos com *ftp*. Vamos agora copiar o ficheiro criado no passo anterior (*myrts.exe*) para a raiz do drive A, tal como mostra a Figura 58. Antes de terminarmos o acesso por FTP ao sistema, teremos ainda de efetuar uma edição ao ficheiro *autoexec.bat*, onde vamos dar informação para que o sistema operativo presente na placa de desenvolvimento execute a nossa aplicação, basta para isso acrescentarmos uma linha com o nome do nosso ficheiro, neste caso *myrts.exe*.

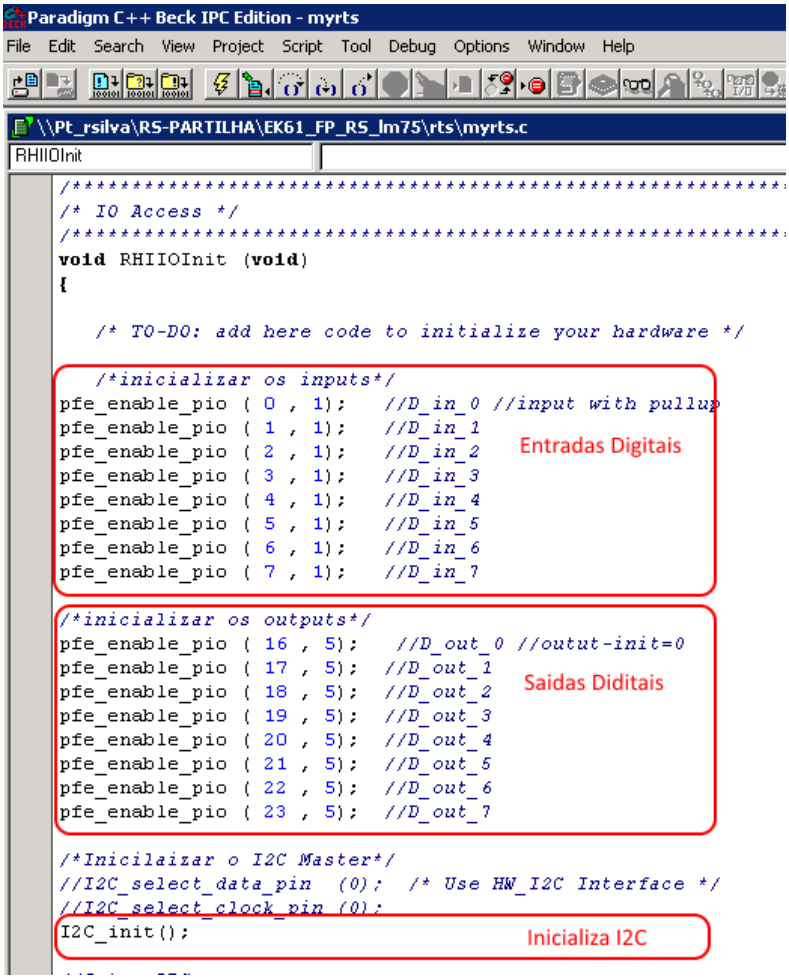


Figura 57 - Edição da função RHIOInit

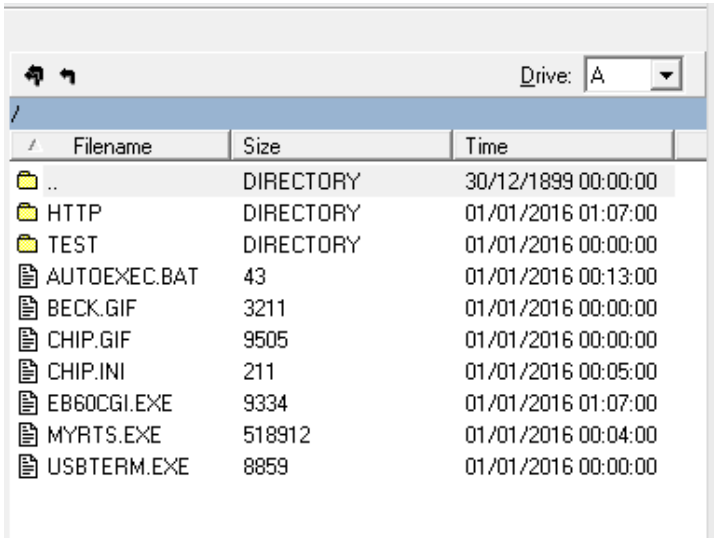


Figura 58 - Acesso por FTP

4.4 - Escrita de um programa em CoDeSys

Para implementar um programa com o CodeSys, foi necessário realizar as ligações à placa de desenvolvimento, tal como tivemos oportunidade de descrever na Tabela 6.

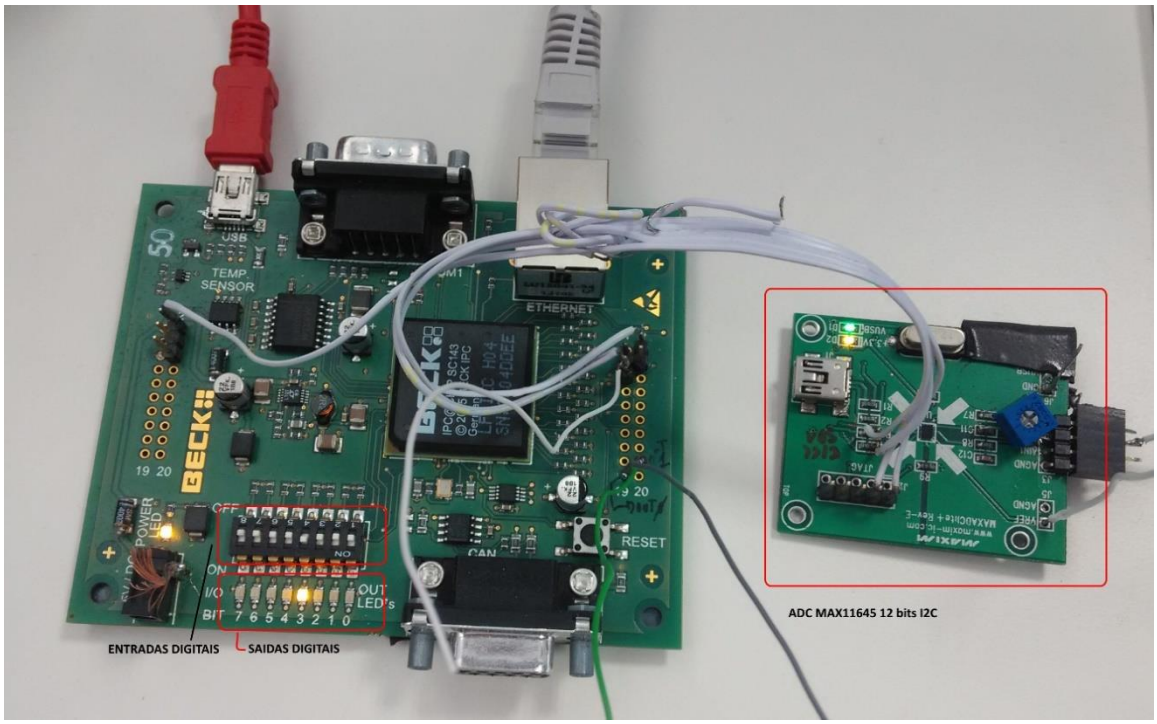


Figura 59 - Placa de desenvolvimento com ADC MAX11645

Para as entradas e saídas digitais a placa de desenvolvimento já tem os componentes necessários para realizar o teste. Para simular as entradas digitais usamos um comutador de oito pontos, e para simular as saídas, vamos utilizar os oito LED que a placa já tem incorporados. A escolha do mapeamento descrito na Tabela 6 já foi pensado de forma a ser possível a utilização destes componentes. Para testar as comunicações série em I2C, vamos usar um conversor analógico/digital do fabricante MAXIM, modelo MAX11645. O resultado final pode ser observado na Figura 59.

Será necessário agora iniciarmos o programa CoDeSys, selecionando *File*, seguindo-se *New*, e logo após somos questionados para qual TSP desejamos escrever a nossa aplicação. Como tivemos já oportunidade de descrever anteriormente todo esse processo de instalação, neste caso, e como o TSP já se encontra devidamente instalado, vamos apenas selecioná-lo, como demonstra a Figura 60.

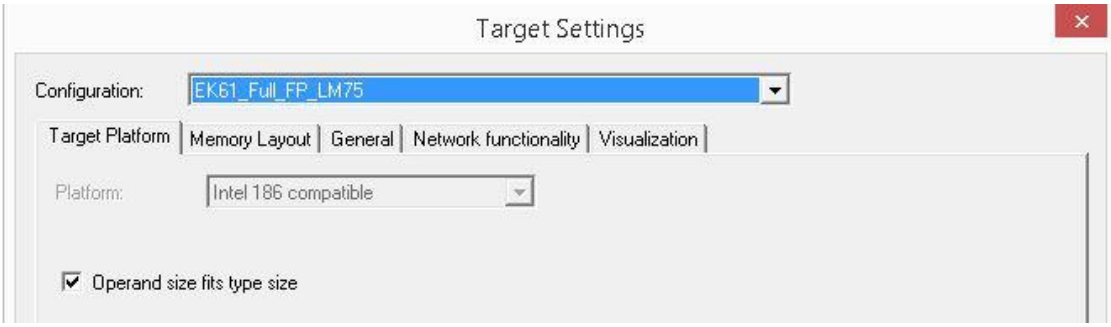


Figura 60 - Seleção do TSP para a realização de um novo programa

Como o objetivo deste ponto é testar todo o processo, vamos escrever um programa muito simples, que vai permitir ler as entradas digitais e analógicas da nossa placa e mostrar o seu valor. O programa vai copiar o valor das entradas digitais para as saídas digitais, vai ser escrito em linguagem estruturada (*Structured Text*) [3] e pode ser observado na imagem da Figura 61.

Na Figura 62 podemos observar já o estado das variáveis após a compilação com sucesso do programa e o respetivo *download* do programa para o PLC (Placa de Desenvolvimento). Aí podemos observar que o valor das variáveis da saída se encontram iguais ao valor das entradas digitais, mesmo após a alteração da entrada digital D_in_03, tal como seria de esperar. Podemos também aí consultar o valor das entradas analógicas.

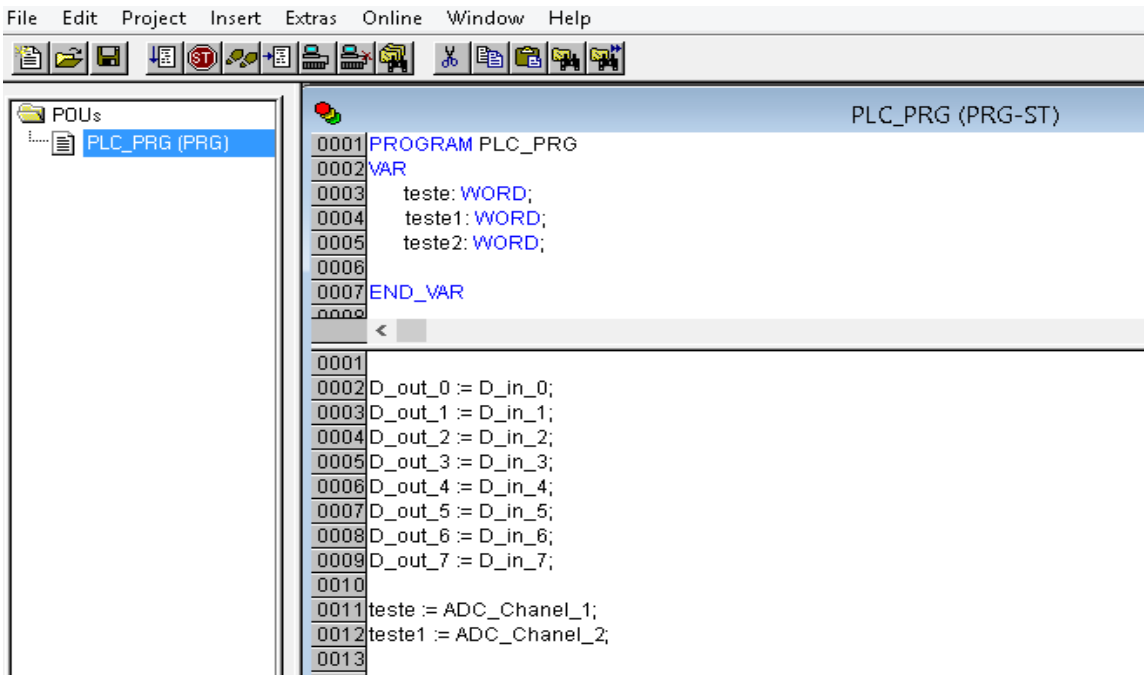


Figura 61 - Programa de teste em linguagem estruturada

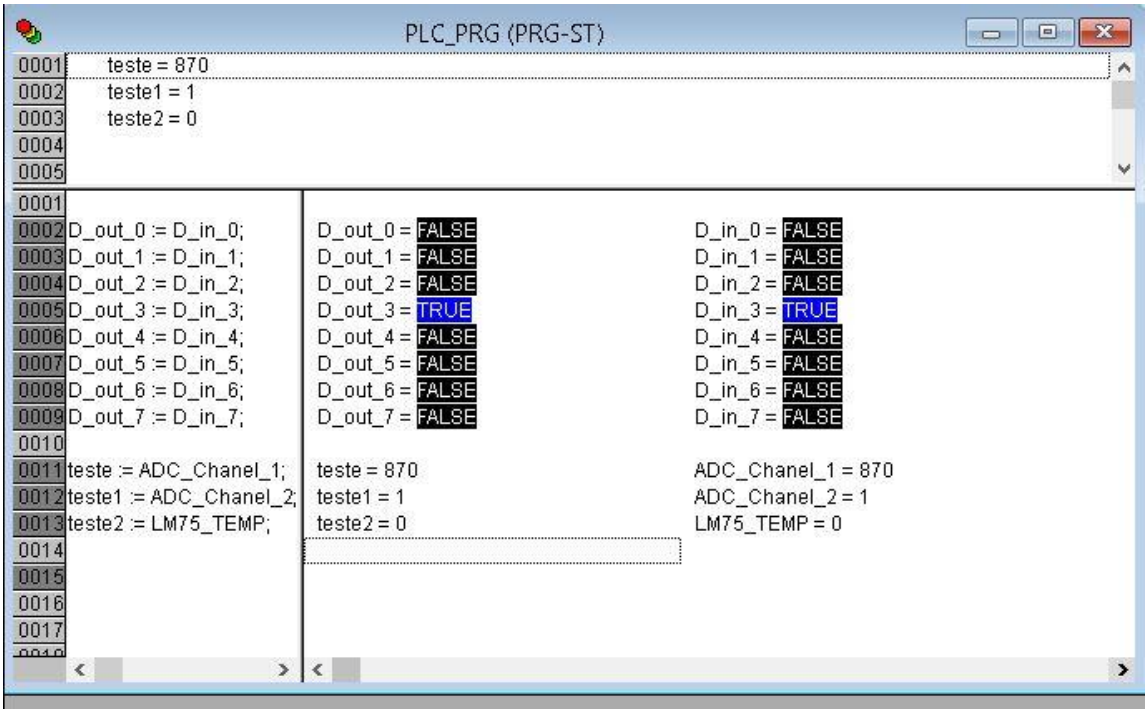


Figura 62 - Estado das variáveis em execução

Para testar se o programa está a aceder com sucesso ao barramento de comunicações I2C, e a realizar as leituras dos dados da forma correta, foi possível com o recurso ao osciloscópio digital, confirmar o estado do barramento.

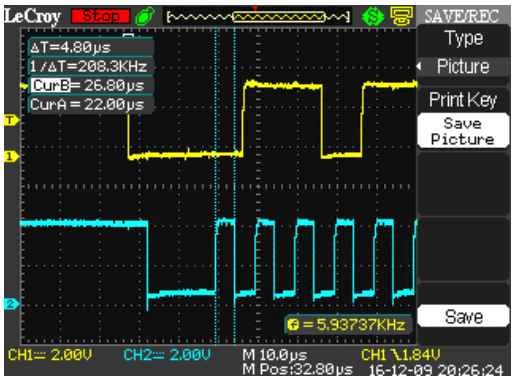


Figura 63 - Análise do barramento I2C

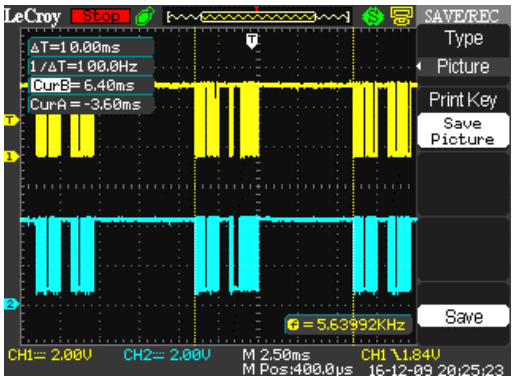


Figura 64 - Analise do tempo de ciclo de programa

Pela análise da Figura 63, podemos confirmar que o *master* da comunicação (processador BECK), se encontra corretamente a iniciar a comunicação e a realizar o acesso das linhas de dados (PIO13 - canal 1 do osciloscópio – amarelo) e que a linha de relógio também se encontra a funcionar de acordo com o esperado (PIO 31 - canal 2 do osciloscópio – azul). Aqui podemos ainda validar se a frequência de relógio se encontra correta. Conforme

podemos observar na referida imagem, a frequência medida é de 100 kHz, que é precisamente o valor que deverá dar para um *clock* de sistema de 96 MHz, como é o caso.

Na Figura 64 podemos também verificar a frequência com que os pedidos de novas leituras se encontram distribuídos no tempo. Neste caso o *Master* I2C, realiza um pedido de novas leituras ao ADC a cada 10 ms (milissegundos). Na Figura 63 podemos ver isso ocorrer três vezes. Este tempo é o tempo de ciclo do programa IEC, definido no PLC.

5 - ARQUITETURA E DESENHO DETALHADO

Após o Capítulo 4 ter sido inteiramente dedicado à descrição detalhada do procedimento necessário para implementar um PLC recorrendo ao RTS da 3S-Smart e da placa de desenvolvimento da BECK, onde foi possível testar e estudar todo o procedimento, será agora a vez de neste capítulo, dedicarmos a nossa atenção à possibilidade de transformarmos o conhecimento adquirido até aqui, e podermos estudar a forma de implementação desta plataforma num produto.

5.1 - Requisitos de implementação

Após chegar a este ponto no decorrer do estagio, e concluída toda a analise até aqui efetuada, estão reunidas as condições necessárias para definir o que pretendemos implementar, com vista a chegarmos a uma solução o mais equilibrada possível, tendo em conta as tecnologias existentes, e com as quais podemos contar para o objetivo que pretendemos atingir. Pretende-se assim uma solução que não tenha um tempo de desenvolvimento demasiadamente elevado e que, por conseguinte, tenha um *time to market* reduzido. Para este controlador não necessitamos de um número elevado de entradas ou de saídas, pois o que é pretendido é apenas criar uma base de conhecimento para avaliar as potencialidades que um controlador deste tipo poderá ter num desenvolvimento de um futuro produto. Tendo por base estes conceitos, passamos de seguida a apresentar os requisitos técnicos básicos para a realização de um produto com base tecnológica baseada em CodeSys.

5.2 - Requisitos técnicos

Tendo em consideração o ambiente de funcionamento habitual deste tipo de produtos, a sua instalação-tipo e as necessidades mais básicas a que um PLC deverá responder, foi definido um conjunto de requisitos a que o protótipo a desenvolver deverá obedecer.

5.2.1 - Tensão de Alimentação

A grande maioria de locais onde encontramos produtos industriais disponibilizam uma tensão de alimentação de 24 V DC. No entanto, para aqueles onde assim não seja, o controlador tem de funcionar de 12 V a 24 V em corrente contínua. Desta forma cobrimos a grande maioria de situações e garantimos também uma maior compatibilidade com outro tipo de áreas onde também são aplicados PLC. A alimentação deverá também ter proteção contra polaridade inversa e também proteção em caso de sobretensão.

- Tensão de funcionamento de 12 V a 24 V DC
- Proteção para polaridade inversa
- Proteção em caso de sobretensão

5.2.2 - Entradas Digitais

Pretende-se que o controlador possa receber 8 entradas digitais. No entanto, estas entradas, por forma a satisfazerem a mesma condição do ponto anterior, têm de ser compatíveis com a tensão de alimentação, ou seja, compatíveis com um valor de entrada de 12 V a 24 V em corrente contínua.

- 8 Entradas digitais
- Entradas compatíveis com uma tensão de 12 a 24 V DC

5.2.3 - Saídas Digitais

O número de saídas digitais deverá ser de 8, todas as saídas deverão ter a tensão nominal igual à tensão de alimentação, ou seja, de 12V a 24V em corrente contínua. Cada saída deverá estar protegida contra curto circuitos e a sua corrente máxima limitada, no entanto deverá ter capacidade de fornecimento máximo de 200 mA sem qualquer limitação.

- 8 Saídas digitais
- Tensão de saída igual à tensão de entrada
- Proteção contra curto-circuitos
- Corrente mínima de saída > 200 mA

5.2.4 - Comunicações

Embora o objetivo não seja o estudo e análise deste tipo de comunicação, pretende-se deixar uma base de contínuo desenvolvimento para o futuro, e desta forma permitir evoluir o protótipo. Estas comunicações foram selecionadas porque são aquelas que maior utilização têm no meio industrial. O protótipo deverá possibilitar as seguintes camadas físicas de comunicação.

- Ethernet
- RS-232
- RS-485
- CAN

5.2.5 - Mecânica

Todo o *hardware* deve estar dentro de uma caixa plástica. Desta forma garante-se uma proteção dos circuitos internos do controlador que deverá ser no mínimo de IP20 [38]. Esta caixa deverá ser escolhida de acordo com a instalação típica em ambiente industrial. Deverá por isso possibilitar fixação em calha DIN de 35 mm [37], deverá também possibilitar um meio de fixação alternativo, para locais onde não exista sistema de fixação em calha DIN de 35 mm, e que poderá ser através de uma fixação mural.

- Caixa plástica
- Índice de proteção IP20 ou superior
- Fixação em calha DIN de 35 mm
- Fixação mural

5.3 - Arquitetura

Chegando a esta fase estão reunidos os dados necessários para que possamos proceder ao desenho detalhado daquela que vai ser a arquitetura do protótipo do controlador. Este é um passo essencial pois permite ter uma visão global dos principais diagramas de blocos de todo o sistema. Assim sendo, o passo seguinte é analisar todos os requisitos técnicos e desenhar todo o sistema.

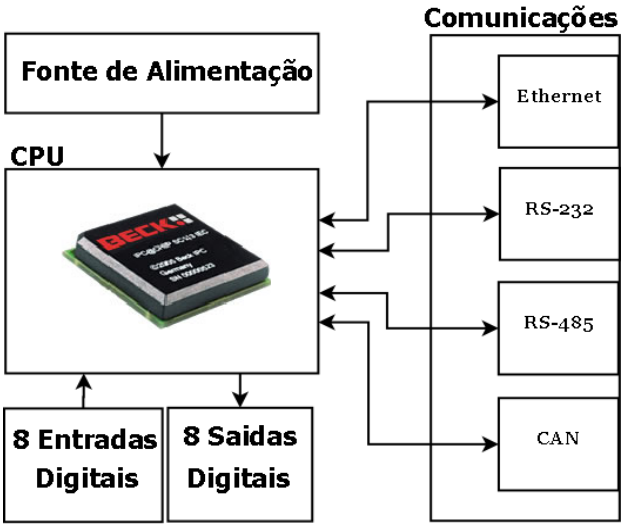


Figura 65 - Diagrama geral de blocos do sistema

Após uma atenta análise de todos os requisitos foi possível contruir o diagrama geral do sistema, bloco por bloco e verificar a dependências entre cada um deles. Esse trabalho poderá ser observado na Figura 65.

O desenho e explicação detalhada de cada um dos blocos representado podem ser consultados nos subcapítulos seguintes deste documento.

5.4 - Desenho detalhado

Para se proceder ao desenho detalhado vamos começar por aquele que foi o último requisito a ser definido, porque depois de definirmos a mecânica do protótipo a restante seleção de componentes vai estar dependente dessa escolha, daí ser a primeira etapa, e por onde normalmente se começa.

Assim sendo, e após análise de algumas caixas de plástico, o modelo escolhido foi o EN-DRE-14-11 que se encontra em Allendale Electronics Ltd (http://www.cases-and-enclosures.co.uk). Esta é uma caixa em ABS e tem a particularidade de possibilitar tanto a fixação em calha DIN como a fixação mural, indo ao encontro dos requisitos mecânicos definidos na Secção 5.2.5. Os pormenores da caixa podem ser observados na Figura 66 e na Figura 67. O desenho da caixa permite-nos também desde logo acomodar ligadores onde vamos colocar as entradas e as saídas que o controlador vai incorporar. Quanto ao nível de proteção IP20 (*Ingress Protection Rating*), que significa que a caixa tem de ser resistente à entrada de sólidos com dimensões iguais ou superiores a 12.5 mm [38], e não resistente a qualquer contacto com água. Como a caixa não possui qualquer orifício com dimensões superiores a 12,5 mm, será por isso de considerar viável a certificação futura do protótipo com o nível de proteção IP20.

Figura 66 - Caixa EN-DRE-14-11

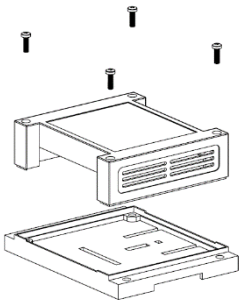


Figura 67 - EN-DRE-14-11 detalhe

Estando definido e escolhido o modelo de caixa a utilizar, qualquer componente que de seguida se vá selecionar terá de ter em conta as dimensões e características que a caixa possibilita. O passo seguinte foi definir a localização dos ligadores, ou seja, das interfaces com o mundo físico exterior ao protótipo, sendo que as necessidades de ligação são as seguintes:

- **Tensão de Alimentação**: 3 contactos (POWER +, GND, SHIELD)
- **Entradas Digitais:** Ligador com 8 contactos
- **Saídas Digitais:** Ligador com 8 contactos

- **Ethernet:** Ficha RJ45
- **RS-232:** Ligador com 5 contactos (TxD, RxD, RTS, CTS, GND)
- **RS-485:** Ligador com 3 contactos (A, B, GND)

Decidiu-se então colocar todos os conectores de comunicações no topo da caixa e na sua base as entradas e saídas digitais, bem como a alimentação do controlador, tal como se pretende demonstrar na Figura 68.

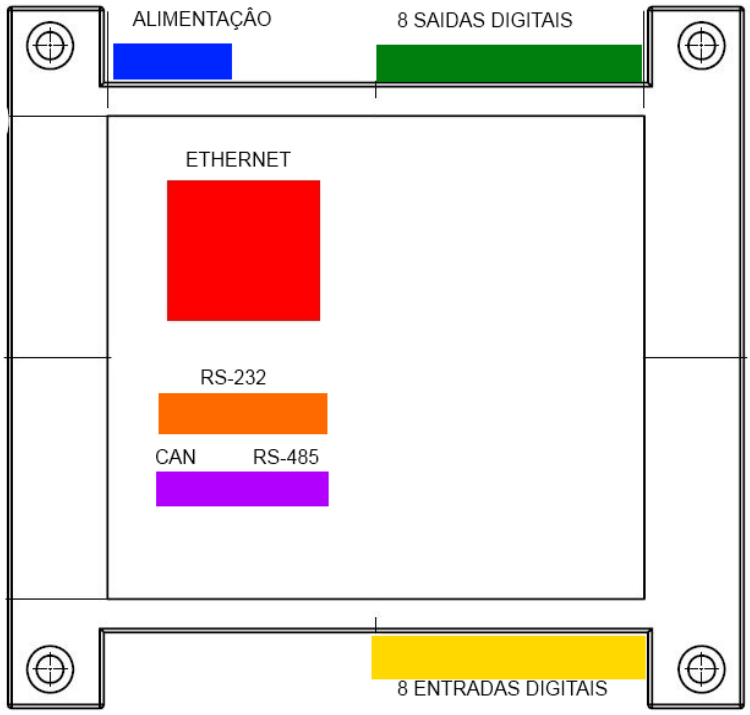


Figura 68 - Localização e definição das ligações na caixa

Esta solução vai obrigar à realização de dois circuitos, ou seja, duas placas de circuito impresso (PCB), um inferior e outro superior. O PCB inferior vai ter a fonte de alimentação e as interfaces com as entradas e as saídas digitais, ao passo que o PCB superior vai ter os circuitos elétricos responsáveis pelas comunicações (CAN, RS-232, RS-485 e *Ethernet*), e vai também ter o Processador da BECK, o SC143-IEC-LF. Os dois PCB vão estar interligados através um *Flexible Flat Cable* (FFC), e suportados mecanicamente por parafusos e por componentes mecânicos designados de *spacers.*

Por forma a garantir o cumprimento na íntegra dos restantes requisitos já analisados no Subcapítulo 5.2 deste documento, o desenho de esquemático foi baseado nos seguintes pressupostos:

A fonte PT5101A foi a escolhida para realizar a tarefa de regulação primária, pois permite regular a tensão para o valor de 5 V e fornecer até um máximo de 1.5 A caso isso seja

necessário. Esta fonte permite manter a regulação da tensão de 5 V na sua saída mesmo com variações de tensão na entrada da mesma. Estas variações poderão ir de 9 V a 36 V. O processador, dependendo da sua utilização, pode atingir no máximo 400 mA de consumo, o que somado ao consumo da restante eletrónica não ultrapassará os 600 mA, visto que as saídas digitais vão estar ligadas à tensão de entrada e o seu consumo não entra para este cálculo.

As entradas e as saídas são realizadas através de IC dedicados para este tipo de aplicações. No caso das entradas o IC escolhido foi o CLT3-4BT6 da STMicroelectronics. Este IC permite a ligação de quatro entradas e garantimos com esta escolha um nível de proteção elevado no que diz respeito à imunidade eletromagnética. Para o circuito de saída e por forma a cumprir com os requisitos, a escolha foi feita no IC VNQ860, também da STMicroelectronics, e que apresenta uma gama de funcionamento de 5.5 V a 36 V, inclui proteção para curto-circuito e tem a possibilidade de fornecer ao circuito de saída até 250mA de corrente por saída.

No que diz respeito às comunicações, foram necessários circuitos dedicados que implementem a camada física do protocolo em causa. No caso do RS-232 a escolha recaiu no IC MAX3232, para o RS-485 e CAN a escolha foi a dos modelos SN65HVD11 e o SN65HVD232 respetivamente. Para o *Ethernet* foi necessário apenas a seleção de uma ficha do tipo modular RJ45, e dos transformadores de isolamento, pois o SC143-IEC já possui o driver interno que realiza a adaptação para a camada física. A escolha recaiu numa ficha já com os transformadores incluídos.

5.5 - Esquema elétrico

Para a realização do desenho do esquema elétrico e do PCB, foi usado um *software* de EDA (*Electronic Design Automation*), já usado pela Bresimar no desenvolvimento dos seus produtos. O programa utilizado para este efeito é o Altium Designer. Devido ao facto de necessitarmos de desenhar dois circuitos separados, foi necessário criar dois projetos distintos, aos quais demos os nomes de “ISEC-CoDeSys-BECK-CPU” (Figura 70) ao projeto onde vamos desenvolver o PCB superior e que vai incluir o processador e todas as comunicações digitais do protótipo, e “ISEC-CoDeSys-BECK”, ao projeto onde vamos desenvolver a fonte de alimentação, as entradas e saídas digitais. Os PCB que resultarem destes projetos formarão a base do protótipo.

É importante referir que no final do desenho da arquitetura foi proposto dotar o protótipo de blocos adicionais que embora não fossem um objetivo do estágio, vão potencializar o desenvolvimento futuro do protótipo.

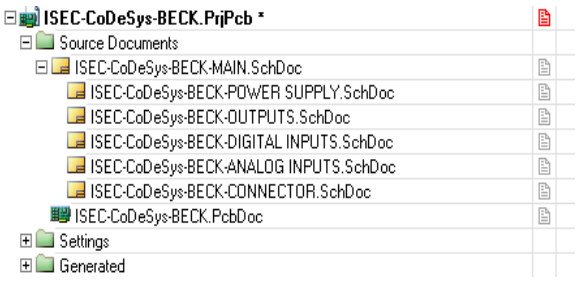


Figura 69 - Projeto ISEC-CoDeSys-BECK

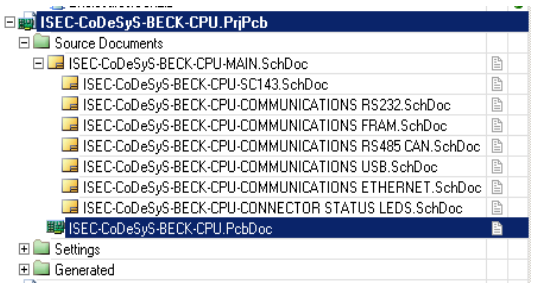


Figura 70 - ISEC-CoDeSys-BECK-CPU

São o caso da adição de um módulo de aquisição analógica com uma entrada em tensão de 0-10 V e outra de corrente de 0-20 mA, isto para o PCB inferior. Para o PCB superior adicionaram-se o módulo USB e um módulo que permite dotar o controlador de memória retentiva, neste caso selecionou-se uma memória do tipo F-RAM de 64 Kbit. Esta memória vai permitir ao utilizador/programador ter mais opções para garantir a integridade dos seus dados e variáveis críticas.

O passo seguinte foi definir a organização de cada projeto no *software* de desenho Altium, desta forma chegou-se ao resultado que pode ser consultado na Figura 71 e na Figura 72.

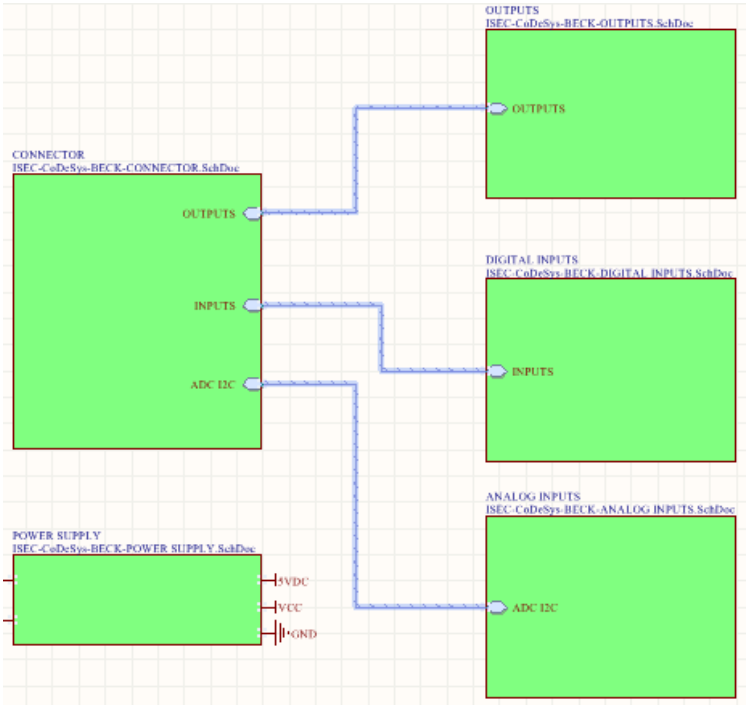


Figura 71 - Diagrama de blocos PCB inferior

O detalhe dos esquemáticos completos, bem como a lista de componentes ou *Bill of Materials* (BOM) com a localização dos componentes, podem ser consultados nos Anexos 1, 2, 3, 4, 5 e 6.

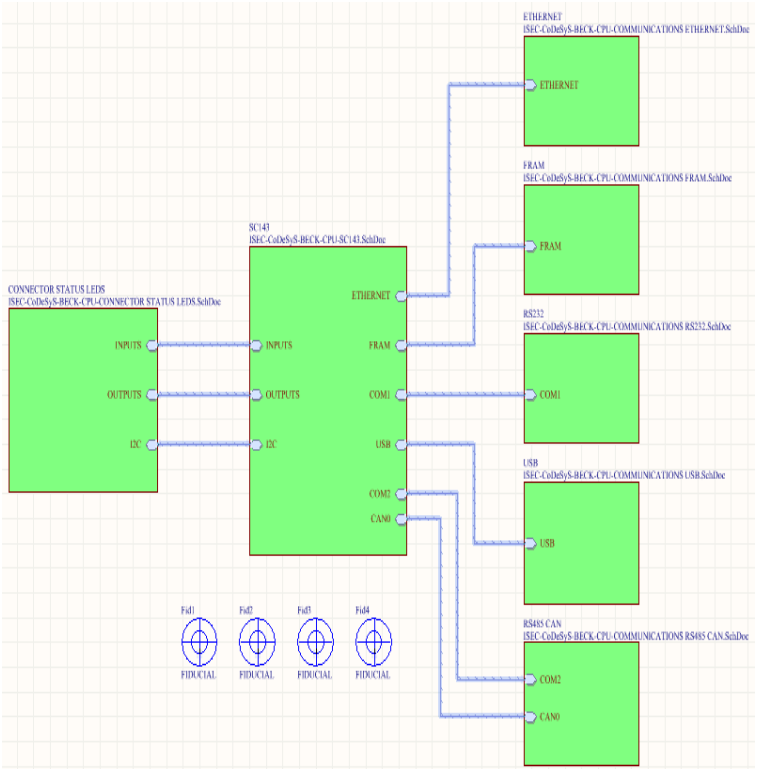


Figura 72 - Diagrama de blocos PCB superior

5.6 - Desenho da placa de circuito impresso

Os circuitos integrados que vão ser usados no PCB inferior e o espaçamento entre pinos possibilita-nos usar uma tecnologia de dupla camada, e desta forma reduzirmos o custo do PCB. No entanto, o PCB superior vai ter um componente com um formato BGA com 177 pinos numa área de 25 mm por 25 mm, e desta forma torna-se necessário aumentar o número de camadas (*Layers*) para possibilitar a extração de todas as ligações que este componente permite. Embora alguns dos pinos não vão ser usados, vamos deixá-los todos disponíveis na camada inferior, pois numa fase de desenvolvimento como esta pode ser necessário aceder a mais algum dos pinos que nesta fase não tenha sido contemplado. Evita-se dessa forma o processo de redesenho do PCB, com todos os custos que isso representa, quer ao nível da compra do PCB, quer ao nível do tempo de atraso para o projeto. Neste tipo de projeto o redesenho implica sempre um atraso significativo porque será necessário o tempo de desenho da nova versão, o tempo de espera para que o fabricante possa executar novos PCB, e ainda o tempo necessário para a soldadura dos novos protótipos.

Este tipo de componentes possui também linhas de dados com uma frequência de *clock* e tempos de subida e de descida extremamente elevados, o que implica um cuidado redobrado, na eliminação sempre que possível de caminhos de retorno longos da corrente para o plano de referência. Assim sendo, este PCB vai ser constituído por seis camadas, o que vai permitir

uma melhor passagem das pistas em cobre, e possibilita também uma maior imunidade a fenómenos de interferência eletromagnética ou *ElectroMagnetic Interference* (EMI) [5].

Em resumo, a construção dos PCB vai obedecer ao que se pode observar nas ilustrações seguintes (Figura 73 e Figura 74). Relativamente ao PCB de duas camadas as opções não são muitas para a sua configuração, mas no PCB superior que vai ser constituído por seis camadas de cobre, aí o número de configurações possíveis já é bem maior. No entanto, e tendo em conta a experiencia da equipa da Bresimar no desenho destas soluções, a opção adotada foi a presente na Figura 74 onde se pode observar uma grande preocupação em que entre camadas de sinal possa estar sempre no mínimo uma de plano de massa ou de VCC, diminuindo assim o comprimento que as correntes transitórias têm de percorrer e contribuindo assim para minimizar os problemas normalmente observados nos ensaios de emissão, a que o produto está legalmente obrigado a ser submetido.

Figura 73 - Composição do PCB inferior [32]

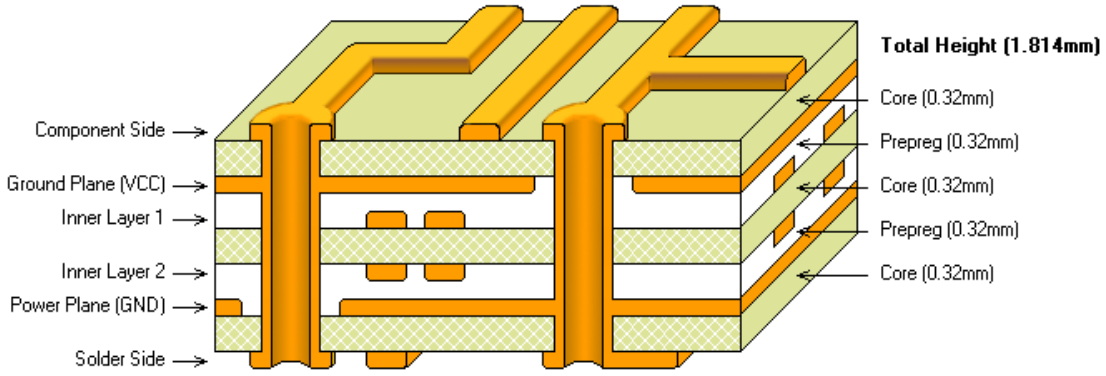


Figura 74 - Composição do PCB superior [32]

Na Bresimar, devido à área de desenvolvimento de produto TekOn, existe já uma extensa biblioteca de componentes devidamente validados e prontos a serem usados. Esta biblioteca é compatível para o *software* Altium Designer e por esse motivo sempre que necessário vamos usar componentes já existentes. No entanto, durante o desenho do esquema elétrico e do PCB, foi possível observar que existem muitos outros componentes, que nunca tinham

sido usados anteriormente, e que por esse motivo não existiam ainda nessa biblioteca. Para que os mesmos pudessem ser usados neste projeto, foi necessário proceder ao seu desenho, pois só assim se conseguiria interligar um desenho de esquemático para o seu equivalente físico e que vai aparecer no desenho final do PCB.

Foi esse o passo seguinte, o da realização do desenho desses componentes. A cada componente é necessário fazer corresponder dois ficheiros, um que descreve o esquema elétrico, onde consta o número de pinos e suas funções, e outro que descreve a localização e tamanho dos pontos de soldadura dos seus pinos [6]. Para além disso o *software* Altium permite ainda a realização de componentes em 3 dimensões, o que se veio a provar ser de uma grande mais-valia, pois possibilita ao desenhador realizar uma primeira validação da interferência entre os diversos componentes numa fase muito preliminar e antes mesmo de mandar executar os PCB e de soldar os componentes, permitindo poupar alguns passos intermédios de validação mecânica. A título de exemplo podemos observar o desenho do componente VNQ860 (Figura 75, Figura 76 e Figura 77).

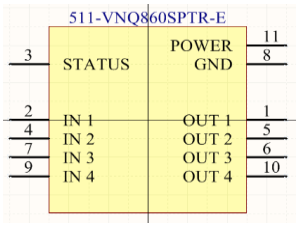


Figura 75 - Componente Esquema Elétrico

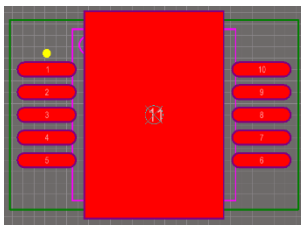


Figura 76 - Componente PCB Footprint

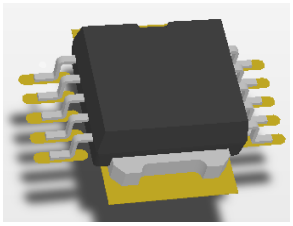


Figura 77 - Componente PCB Footprint 3D

Embora o componente seja apenas um, como podemos ver nas imagens acima, o mesmo aparece descrito em três ficheiros diferentes. O ficheiro presente na Figura 75 vai estar presente em todas as folhas de desenho de esquema elétrico onde seja necessário utilizar o componente em causa., quando o utilizador transforma o seu esquema elétrico em PCB, o *software* Altium vai passar a utilizar apenas o ficheiro da Figura 76, sendo que nesse ficheiro vão estar representadas as ligações elétricas implementadas no desenho do esquema. O ficheiro com o corpo a três dimensões presente na Figura 77, é o que vai permitir ao utilizador realizar a primeira validação mecânica em conjunto com todos os restantes componentes presentes no produto.

Após o posicionamento dos componentes nos PCB e depois de concluído o seu desenho, o resultado final pode ser verificado nas imagens das Figuras 78 a 81. O resultado camada a camada com todo o detalhe pode ser consultado no Anexo 3 e no Anexo 6.

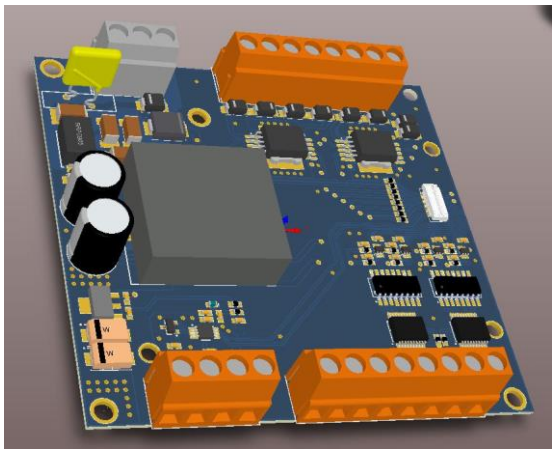


Figura 78 – Imagem 3D do PCB inferior

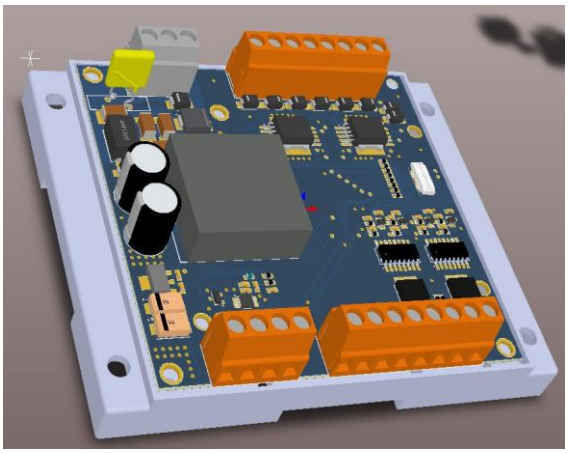


Figura 79 – Imagem 3D do PCB inferior mais base da caixa

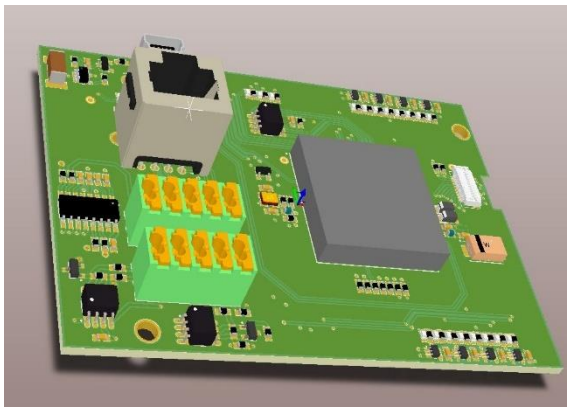


Figura 80 – Imagem 3D do PCB superior

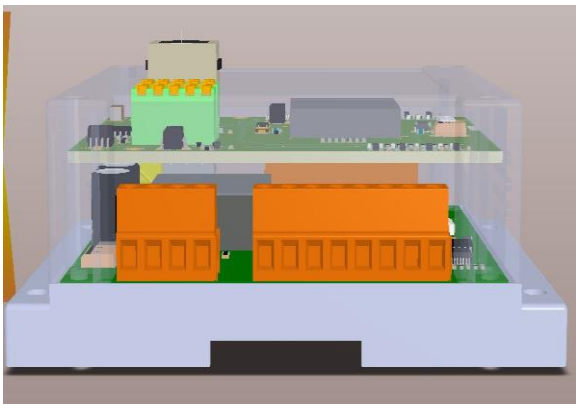


Figura 81 – Imagem 3D do PCB inferior e Superior mais caixa

Nesta fase após a conclusão do desenho de todo o *hardware* do controlador, foram também criados os ficheiros necessários para dar início à produção do protótipo, quer das placas de circuito impresso, quer das telas que são normalmente usadas para aplicação de pasta de soldar, antes de os PCB chegarem às linhas para aplicação automática de componentes. Normalmente nesta fase é também necessário proceder ao desenho de um PCB em painel, que permite otimizar o tempo de montagem dos componentes eletrónicos nas linhas de montagem e que por norma contém um número mais elevado de circuitos por cada PCB[7]. Este passo é normalmente realizado apenas após a validação em completo do protótipo.

6 - CERTIFICAÇÃO

Neste capítulo vamos abordar o tema da certificação de produto e as respetivas exigências legais, nacionais e europeias, tais como a necessidade de marcação CE e o que necessitamos atingir para que tal possa acontecer. Relativamente a esta matéria é comum surgirem sempre algumas questões muito pertinentes, como por exemplo o que é e para que serve a marcação CE.

6.1 - Marcação CE

Todos nós já tivermos oportunidade de observar junto de muitos dos produtos que adquirimos o símbolo que se encontra na imagem da Figura 82, o da marcação CE. Este símbolo que tem sempre que seguir as regras de grafismo previamente estabelecidas pela legislação, é constituído pelas inicias "CE" que são a abreviatura da designação Francesa *Conformité Européene* e que significa Conformidade Europeia.

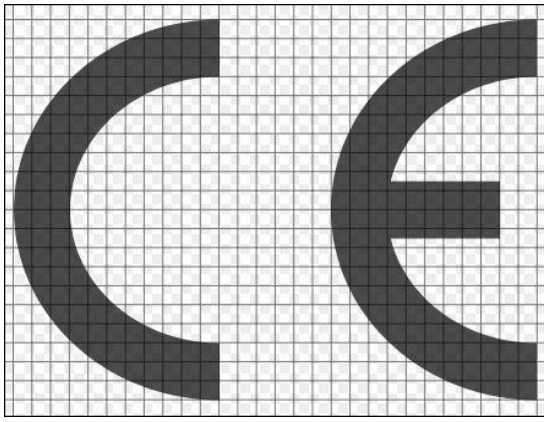


Figura 82 - Marcação CE com grade de construção [8]

Este símbolo permite a um qualquer consumidor saber se um determinado produto que pretende adquirir se encontra em conformidade com os requisitos estabelecidos em diretivas comunitárias.

Assim sendo, quando um determinado fabricante coloca a marcação CE nos seus produtos, está a dar evidências de que os seus produtos estão em conformidade com as Normas Europeias e que desta forma poderão ser comercializados e também poderão ter direito a livre circulação pelo Espaço Económico Europeu (EEE).

A legislação exige ao fabricante, neste caso à Bresimar Automação S.A., que afixe nos seus produtos a marcação CE, sendo que desta forma o fabricante é legalmente o responsável por

assegurar se um determinado produto por si produzido ou colocado no mercado está conforme as especificações técnicas designadas para o respetivo processo de conceção e de fabrico, de acordo com as disposições da Diretiva que se lhe aplica.

É de extrema importância para uma equipa de desenvolvimento de produto, que numa fase muito embrionária do projeto consiga já saber quais vão ser os requisitos legais e normativos, para que dessa forma eles possam fazer parte dos requisitos do produto e serem tidos em conta durante a fase de desenvolvimento. Desta forma é possível minimizar o risco de um produto, depois de chegar à fase de protótipo, chumbar na fase dos testes de conformidade legal. Quanto mais tarde no projeto for identificada a obrigatoriedade legal a que o mesmo vai estar sujeito, maior vai ser o risco associado a uma não conformidade do produto, com todas as consequências que dai poderão advir, como um esforço maior da equipa de desenvolvimento, o derrapar do plano de projeto e também os custos adicionais de uma nova etapa de ensaios a desenvolver no organismo notificado para o efeito.

Para se assegurar que todos os ensaios são realizados de acordo com as normas vigentes, a União Europeia possui uma base de dados que permite esclarecer algumas dúvidas relativamente a este assunto. Essa base de dados encontra-se no seguinte endereço eletrónico:

- *http://ec.europa.eu/growth/tools-databases/nando/*

É então essencial obter essa informação, o que nem sempre é fácil e por vezes obriga ao pedido de apoio por parte de organismos especializados nessa matéria. Nos próximos subcapítulos vai ser abordada uma das diretivas, pois tendo em conta a experiencia do grupo de trabalho TekOn da Bresimar, essa diretiva é uma das que mais tem sido objeto de análise, tendo em conta o tipo de produtos realizados pela Bresimar. A diretiva em causa é a Directiva 2004/108/CE [8], relativa à aproximação das legislações dos Estados-Membros respeitantes à compatibilidade eletromagnética, também conhecida por diretiva CEM (Compatibilidade EletroMagnética). Trata-se de uma diretiva de marcação CE e o seu incumprimento pode acarretar pesadas sanções e obrigar à retirada dos produtos do mercado.

6.1 - Diretiva CEM

A lista de normas harmonizadas no âmbito da diretiva CEM, é publicada no Jornal Oficial da União Europeia [2] e, garantindo a conformidade do nosso produto face às normas aí expostas e que lhe são aplicáveis, confere a presunção da conformidade desse equipamento com os requisitos essenciais da diretiva. Tendo em conta o nosso tipo de equipamento, olhando para as suas funções e ambiente de funcionamento, teremos então agora de o enquadrar numa das normas aí referidas [8]. Não vamos aqui expor todas as tabelas relevantes, pois esse não é o objetivo desta apresentação, no entanto, numa dessas tabelas,

encontramos um ponto que refere o seguinte texto, “Equipamento elétrico de medição, de comando e de laboratório – EN 61326-1”, que é precisamente aquilo que o nosso equipamento desenhado no Capítulo 5 pretende ser. Importa também neste ponto referir que estes são os requisitos legais para que o mesmo seja colocado no mercado, mas existe ainda um conjunto mais alargado de normas e de certificações a que um produto deste género se pode sujeitar. São exemplos as normas e certificações para ambientes qualificados como perigosos, ambientes marinhos, entre outros. No entanto, o agente responsável pela colocação no mercado europeu não está obrigado legalmente à sua conformidade.

Chegando a este passo, foi possível já enquadrar o nosso produto num conjunto de normas a que ele vai ter de obter conformidade. Neste caso a EN 61326 [2].

Esta norma tem como principal objetivo o de garantir que as emissões do nosso equipamento (EUT - *Equipment Under Test)*, estão dentro dos limites estabelecidos nessa mesma norma, e ao mesmo tempo garantir igualmente que o EUT possa suportar perturbações bem acima das estabelecidas para emissão. Desta forma, e se este objetivo for atingido para todos os equipamentos, podemos afirmar que todos os equipamentos podem operar dentro de um determinado ambiente eletromagnético sem que os mesmos causem perturbações entre si (Figura 83).

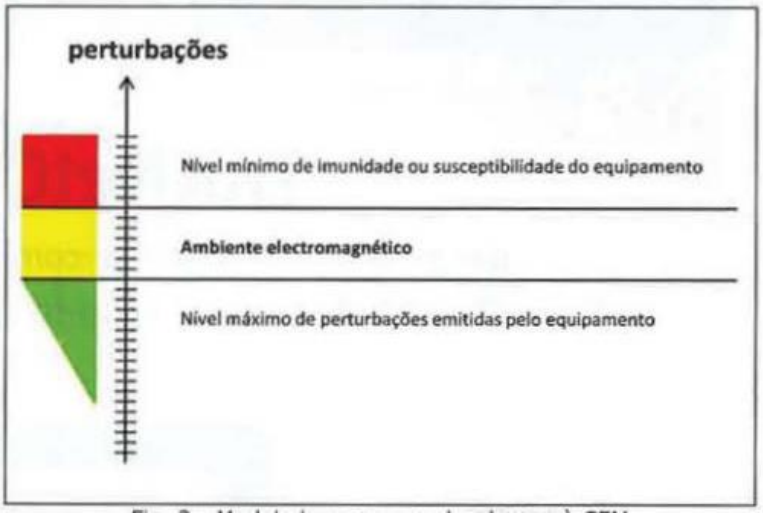


Figura 83 – Ambiente eletromagnético ideal [49]

Teremos por isso de olhar para todos os equipamentos segunda esta norma, como um conjunto de entradas e de saídas das perturbações, ou seja, de imunidade e de emissão, e para os quais é necessário avaliar se os níveis estabelecidos são respeitados.

É ainda necessário que o laboratório que vai realizar os ensaios CEM, para além do enquadramento do nosso equipamento, da descrição de todas as entradas e saídas, e

respetivos comprimentos máximo dos cabos que vão realizar a interligação, tenha também informação sobre o tipo de tensão e o seu valor nominal necessário para alimentar o EUT. É também necessário indicar qual o ambiente em que o mesmo vai ser tipicamente instalado, pois deste dependem os níveis de teste a que o equipamento vai estar sujeito. Só com todos estes detalhes conhecidos é possível à entidade responsável pela condução dos ensaios CEM, determinar os níveis máximos de emissão e os níveis mínimos que o EUT tem de suportar para os ensaios de imunidade.

6.1 - Enquadramento do produto

Para que possamos saber os índices de severidade e os limites de todos os ensaios aplicáveis, será necessário enviar para o laboratório de ensaio informação adicional do nosso equipamento ou EUT.

Na Figura 84 podemos observar algumas das características sobre o equipamento que vai ser alvo de ensaios, que o laboratório selecionado para realizar os ensaios vai solicitar.

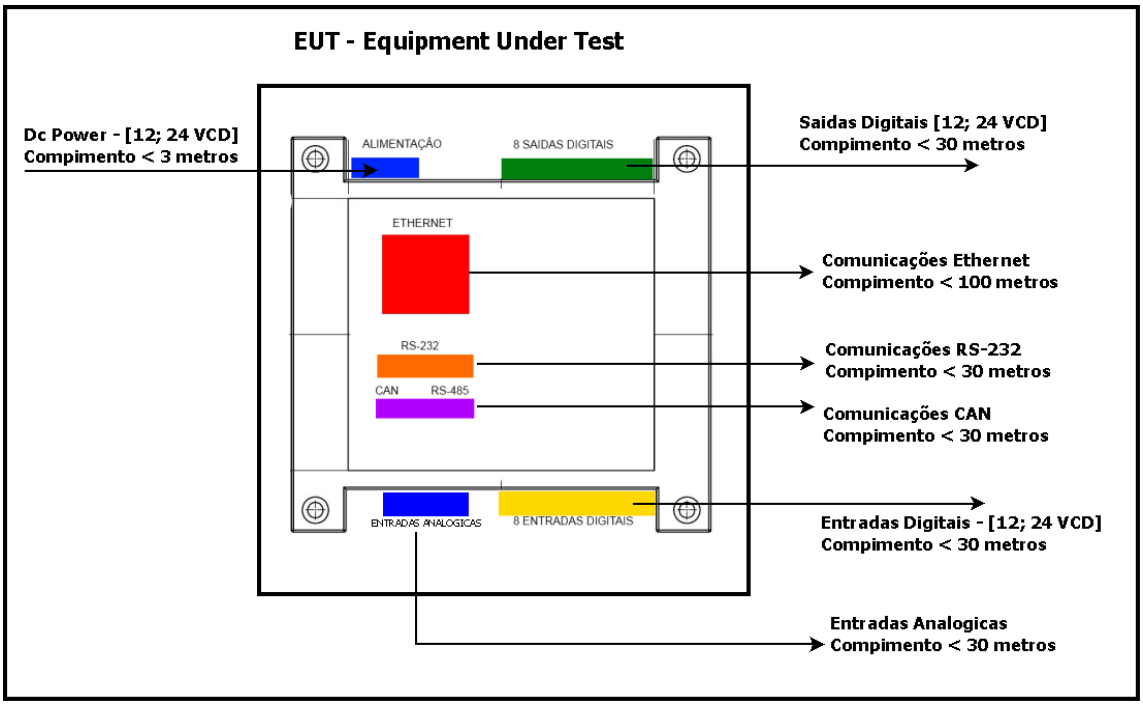


Figura 84 – Enquadramento do EUT para os ensaios CEM

Falta, no entanto, ainda referir que o ambiente para o qual pretendemos que o nosso equipamento seja ensaiado é o industrial, ou seja, segundo a EN 61326 para equipamento que vai ser usado num ambiente eletromagnético industrial (Tabela 2 da norma).

Os ensaios normalizados exigidos pela norma podem ser então classificados em dois grandes grupos, imunidade e emissão. Tendo em conta a descrição do nosso equipamento e o ambiente onde se pretende que seja operado, o mesmo vai ter de ser submetido aos seguintes ensaios:

CEM – Emissão

O equipamento deverá ser testado em pleno funcionamento e dentro de um ambiente devidamente controlado, normalmente uma câmara anecoica ou semi-anecoica, e não deverá emitir radiação superior aos limites estabelecidos pelo CISPR 11 [49].

CEM – Imunidade

Para realizar os ensaios de imunidade existe um número maior de ensaios que serão necessários realizar. Estes ensaios e os seus limites poderão ser diferentes para tipos diferentes de entradas, saídas ou até mesmo do invólucro do equipamento ou EUT, e por isso vão ser descritos na tabela que se segue (Tabela 7).

Tabela 7 - Quadro resumo dos ensaios de imunidade [2]

Local do ensaio (Port)	**Fenómeno**	**Ensaio**	**Valores de ensaio**	**Critério de aceitação**
Invólucro	Descargas Electroestáticas (ESD)	IEC 61000-4-2	4KV descarga de contacto 8KV descarga no ar	B B
Invólucro	Campo eletromagnético	IEC 61000-4-3	10V/m (80MHz até 1GHz) 3V/m (1.4GHz até 2GHz) 1V/m (3.0GHz até 2.7GHz)	A A A
Invólucro	Campo Magnético	IEC 61000-4-8	30A/m (50Hz, 60Hz)	A
DC *Power*	*Busrt*	IEC 61000-4-4	2KV (5/50nS, 5kHz)	B
DC *Power*	*Surge*	IEC 61000-4-5	1KV (*Line to Line*) 2KV (*Line to Ground*)	B
DC *Power*	*Conducted* RF	IEC 61000-4-6	3V (150 kHz to 80MHz	A
Entradas digitais Analógicas Saídas Digitais RS-232 CAN	*Busrt*	IEC 61000-4-4	1KV (5/50nS, 5kHz)	B
Entradas digitais Analógicas Saídas Digitais RS-232 CAN	*Conducted* RF	IEC 61000-4-6	3V (150kHz to 80MHz)	A

Tabela 8 - Quadro resumo dos ensaios de imunidade [2] (cont.)

Local do ensaio (Port)	**Fenómeno**	**Ensaio**	**Valores de ensaio**	**Critério de aceitação**
Ethernet	*Busrt*	IEC 61000-4-4	1KV (5/50nS, 5kHz)	B
Ethernet	*Surge*	IEC 61000-4-5	1KV (*Line to Ground*)	B
Ethernet	*Conducted* RF	IEC 61000-4-6	3V (150kHz to 80MHz	A

Relativamente ao critério de aceitação, temos ainda de referir que quando é exigido ao EUT um critério, B, significa que após a perturbação provocada pelo ensaio, o equipamento em testes não deverá apresentar qualquer anomalia e funcionar perfeitamente logo após o fim da perturbação, mesmo que durante o ensaio se observe perda de funcionalidade, no entanto se isto ocorrer, o EUT deverá recuperar o seu correto funcionamento sem qualquer intervenção. Pelo contrário o critério A, obriga a que o equipamento mantenha o seu pleno funcionamento, sem perda de funcionalidade, mesmo durante o tempo que em a perturbação estiver a ocorrer. Caso exista perda de funcionalidade, a mesma deverá ser especificada pelo fabricante, assim como o seu valor.

Cada ensaio normalizado expresso na Tabela 7 deverá agora ser consultado para avaliação e descrição da forma de teste, e das características do sinal que vai ser considerado para simular a perturbação. Caso o equipamento alvo de todos estes testes (EUT), obtenha resultados positivos em todos os ensaios, podemos assim assumir como responsáveis pela introdução do produto no mercado nacional e europeu, que nos responsabilizamos perante o mercado que o nosso produto cumpre com todos os requisitos legais exigidos pela diretiva CEM.

7 - CONCLUSÕES

O principal contributo deste trabalho consistiu em criar *know-how* e preparar uma plataforma de desenvolvimento e um protótipo versátil, para que de futuro seja possível a empresa estar preparada para oportunidades de negócio em que seja necessária a implementação de um sistema de controlo, bastante dedicado ou "customizado" para uma determinada aplicação e baseado em CodeSys. Este trabalho passou pela pesquisa de soluções semelhantes e concorrentes no mercado, pela implementação de um sistema para testes numa placa de desenvolvimento onde foi possível observar o real funcionamento de uma solução deste tipo e ainda pelo projeto detalhado de um protótipo com um conjunto de interfaces consideradas básicas para este tipo de ambientes. Adicionalmente, foi analisado o percurso futuro necessário em termos de normalização e ensaios de conformidade, com vista à obtenção da marcação CE.

7.1 - Reflexão sobre o trabalho realizado

Este trabalho permitiu um ganho muito importante em conhecimento na área dos sistemas de controlo industriais. A pesquisa e projeto desenvolvidos neste estágio criaram a oportunidade de contacto com um tipo de ferramentas que até aqui não tinha ainda sido possível ao autor conhecer no decorrer do seu percurso académico e profissional.

No desenvolvimento deste trabalho todos os pontos relatados foram muito relevantes. No entanto destacam-se alguns em particular. Em primeiro lugar destaca-se a pesquisa que foi necessário realizar para reunir toda a informação que permitiu a redação deste relatório de estágio. Foi possível com esta pesquisa ter um contacto muito grande com toda esta realidade e perceber que é uma área muito ativa e onde existe muita concorrência. Em segundo lugar salienta-se toda a fase de testes, onde foi possível a realização e implementação do núcleo da solução. Foi aí também possível perceber a real complexidade que se encontra por de trás da implementação de um sistema deste tipo. Por último, o desenho de *hardware*, não poderia deixar de se referir, pois desde que o autor integrou a equipa de colaboradores do Grupo Bresimar, que tem sido uma das suas principais responsabilidades dentro da unidade de negócios TekOn. Dentro desta área, este trabalho permitiu também crescer enquanto profissional. O desenho do PCB de uma solução como a aqui apresentada, exigiu a realização de um estudo muito detalhado sobre a complexidade relacionada com o desenho do esquema elétrico e também sobre o PCB.

7.2 - Conclusões gerais

Relativamente aos objetivos a que este trabalho se propunha, pode-se considerar que todos eles foram alcançados. Em particular foi possível estudar detalhadamente a solução CodeSys. Foram também devidamente estudados, implementados e documentados todos os passos necessários para a colocação em funcionamento de todo o sistema, composto pela criação do RTS, TSP e da escrita de um programa de testes que permitiu testar a implementação de todos os passos anteriores. Foi ainda cumprido o último ponto dos objetivos deste trabalho, e que se propunha a estudar a viabilidade técnica da integração do produto da 3S-Smart, o CodeSys em soluções que possam ser desenvolvidas pela Bresimar. Ou seja, a partir deste momento a equipa de desenvolvimento da Bresimar tem na sua posse o conhecimento técnico que lhe permitirá o desenvolvimento de uma solução como aquela que aqui foi proposta, de forma a poder dar resposta em tempo útil a oportunidades de negócio futuras nesta área.

7.3 - Desenvolvimentos futuros

Terminada esta fase de tomada de conhecimento de uma nova plataforma, surgiram durante este estágio algumas ideias para implementações futuras, e que a serem implementadas poderiam melhorar o protótipo, pois poderiam alargar o âmbito da sua aplicação. Estas ideias são fruto de uma revolução que se está a sentir na área industrial, e que já se refere como podendo ser a quarta revolução industrial (*Industry 4.0*). Algumas dessas ideias foram as seguintes:

- Adicionar novos protocolos e tecnologias de comunicação sem fios, como por exemplo o Sigfox ou ainda a tecnologia LoRa;
- Implementar no sistema um servidor de páginas *Web*, para permitir uma fácil parametrização do sistema por parte do utilizador;
- Adicionar a capacidade de o PLC comunicar com algumas das mais conhecidas plataformas de IoT, como por exemplo a plataforma Watson da IBM ou a Azure da *Microsoft*;
- Implementar um HMI, através de uma interface gráfica, por exemplo um LCD.

Por fim espera-se que este trabalho aqui apresentado em conjunto com as novas ideias para desenvolvimentos futuros, possam permitir e motivar novos trabalhos nesta área, quer seja na Bresimar, no ISEC, ou noutra instituição de ensino.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BRESIMAR, (2016). [Online]. Available: http://www.bresimar.pt/. [Acedido em 01 12 2016].

[2] EN61326-1, Electrical equipment for measurement, control and laboratory use. EMC requirements. General requirements, CENELEC, (2013).

[3] J. Karl-Heinz e M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems, Berlin : Springer-Verlag, (2010).

[4] REVISTA-EXAME-392, "100 MELHORES EMPRESAS PARA TRABALHAR," *REVISTA EXAME 392*, p. 38, DEZEMBRO (2016).

[5] E. B. Joffe, High-Speed PCB Design for EMC and Signal Integrity, Sweden: CEI-Europe, (2016).

[6] IPC-2221A, Generic Standard on Printed Circuit Design, IPC -Association Connecting Electronics Industries, (2003).

[7] IPC-D-325A, Documentation Requirements for Printed Board, assemblies and Support Drawings, IPC -Association Connecting Electronics Industries, (1995).

[8] DIRECTIVA-2004/108/CE, relativa à aproximação das legislações dos Estados-Membros respeitantes à compatibilidade electromagnética, Parlamento Europeu, (2004).

[9] D. R. S. Mroczkowski, Trilogy of Connectors. Basic Principles And Connector Design Explanations, WÜRTH ELEKTRONIK, 2012.

[1 0] T. Brander, Trilogy of Magnetics - Design Guide For EMI Filter Design, SMPS & RF Circuits, 4 edição, WÜRTH ELEKTRONIK, 2009.

[1 1] R. C. Harwell e K. L. Sparks, EC 61131-3, CoDeSys standardize control logic programming, Control Engineering;Jan2011, Vol. 58 Issue 1, p20, (2011).

[1 2] Wago, "Modular WAGO-I/O-SYSTEM, IP 20 (750/753 Series)," (2016). [Online]. Available: http://global.wago.com/en/products/product-catalog/components-automation/modular-io-system-series-750-753/overview/index.jsp.

[1 3] TekonElectronics, "WGW410 – Wireless Modbus Gateway with Analog Outputs," (2016). [Online]. Available:

http://www.tekonelectronics.com/Cache/binImagens/WGW410_Wireless_Modbus_Gateway_com_8xAO_29-8795.pdf. [Acedido em 20 04 2017].

[1 4] ARTECO, “SU30-PLC,” [Online]. Available: http://www.arteco-global.com/en/industrial-automation/products/axis-controller-plc/axis-controller-su310/. [Acedido em 01 12 2016].

[1 5] ARTECO, “SU-PLC-USER-MANUAL,” (2016). [Online]. Available: http://www.arteco-global.com/wp-content/uploads/2008/05/SU-PLC-USER-MANUAL-1.pdf. [Acedido em 01 12 2016].

[1 6] Ascontecnologic, “SigmaPAC - CU-02-MU-UK User Manual,” (2016). [Online]. Available: http://www.ascontecnologic.com/images/PRODOTTI/AutomazioneIndustriale/Sistemi-programmabili/PDF-SigmaPAC/CU-02-MU-UK.pdf. [Acedido em 01 12 2016].

[1 7] 3S-Smart, 2016. [Online]. Available: http://www.3s-software.com/. [Acedido em 01 12 2016].

[1 8] 3S-Smart, “CODESYS Manual,” (2015). [Online]. Available: https://www.codesys.com/support-training/self-help/codesys-manual.html. [Acedido em 01 12 2016].

[1 9] 3S-Smart, “Device Directory - Devices programmable with CODESYS,” (2015). [Online]. Available: http://devices.codesys.com/device-directory.html. [Acedido em 01 12 2016].

[2 0] EEN, (2016). [Online]. Available: http://www.enterpriseeuropenetwork.pt/. [Acedido em 01 12 2016].

[2 1] Elsist, (2016). [Online]. Available: http://www.elsist.it/. [Acedido em 01 12 2016].

[2 2] ProConOS-Advantech, (2015). [Online]. Available: http://www.advantech.com/products/ADAM-5510EKW/mod_2DB0BD05-7A9C-48E2-B442-82D7002CA125.aspx. [Acedido em 01 12 2016].

[2 3] ProConOS-Spectra, (2015). [Online]. Available: http://www.spectra.de/newsletter/files/CH_17112006/Datenblatt-KinCon-Serie.pdf. [Acedido em 01 12 2015].

[2 4] ProConOS, “MultiProg and ProConOS Datasheet,” (2016). [Online]. Available: https://www.amplicon.com/data/multiprog_proconos_e.pdf. [Acedido em 01 12 2016].

[2 5] SENECA, (2016). [Online]. Available: http://www.seneca.it/. [Acedido em 01 12 2016].

[2 6] TDE-MACNO, (2016). [Online]. Available: http://www.tdemacno.com/. [Acedido em 01 12 2016].

[2 7] Advantech, (2016). [Online]. Available: http://www.advantech.com/products/1-368qnc/adam-5510ekw-tp/mod_3ed8a868-df45-4465-9342-de6ef3adc1dc. [Acedido em 01 12 2016].

[2 8] Copalp, (2016). [Online]. Available: http://www.copalp.com/fr/developpement/. [Acedido em 01 12 2016].

[2 9] sixnet, (2015). [Online]. Available: http://www.sixnet.com/. [Acedido em 01 12 2015].

[3 0] Beckhoff, (2016). [Online]. Available: https://www.beckhoff.com/. [Acedido em 01 12 2016].

[3 1] Brodersen, (2016). [Online]. Available: http://brodersen.com/products/rtu/rtu32/. [Acedido em 01 12 2016].

[3 2] Altium, (2016). [Online]. Available: http://www.altium.com/. [Acedido em 01 12 2016].

[3 3] Texas-Instruments, “PCB Design Guidelines,” (2016). [Online]. Available: http://www.ti.com/lit/an/szza009/szza009.pdf. [Acedido em 01 12 2016].

[3 4] ISaGRAF, “ISaGRAF Getting Started,” (2016). [Online]. Available: http://www.isagraf.com/get/isagraf5/gettingstarted.pdf. [Acedido em 01 12 2016].

[3 5] Texas-Instruments, “Electromagnetic Emission From Logic Circuits,” (2016). [Online]. Available: http://www.ti.com/lit/an/szza007/szza007.pdf. [Acedido em 01 12 2016].

[3 6] Texas-Instruments, “Printed Circuit Board Layout for Improved Electromagnetic Compatibility,” (2016). [Online]. Available: http://hsi.web.cern.ch/hsi/s-link/devices/g-ldc/sdya011.pdf. [Acedido em 01 12 2016].

[3 7] wikipedia, “DIN rail,” (2016). [Online]. Available: http://en.wikipedia.org/wiki/DIN_rail. [Acedido em 01 12 2016].

[3 8] wikipedia, “International Protection Marking,” (2016). [Online]. Available: http://en.wikipedia.org/wiki/IP_Code. [Acedido em 01 12 2016].

[3 9] 3S-Smart, “Licensing,” (2015). [Online]. Available: https://www.codesys.com/the-system/licensing.html. [Acedido em 01 12 2016].

[4 0] 3S-Smart, “Lista de Fabricantes,” (2015). [Online]. Available: http://www.3s-software.com/index.shtml?en_SP_CPUs. [Acedido em 01 12 2015].

[4 1] B. A. S.A., “SONDAS DE TEMPERATURA E DE NÍVEL,” (2016). [Online]. Available: http://www.bresimar.pt/pt/tekon-electronics/sondas-de-temperatura-e-de-nivel. [Acedido em 01 12 2016].

[4 2] Infoteam-OPENPCS, “Data sheet OpenPCS V.7,” (2016). [Online]. Available: https://www.infoteam.de/en/industry/building-automation/industrial-control-systems-in-accordance-with-iec-61131/sps-software-openpcs-v7/. [Acedido em 01 12 2016].

[4 3] ICP-DAS, “KinCon-8045/8345/8745 page,” (2016). [Online]. Available: http://www.icpdas.com/products/PAC/kincon/indusoft_kincon.htm. [Acedido em 01 12 2016].

[4 4] AXEL, “LogicLab Product Page,” (2016). [Online]. Available: http://www.axelsw.it/index.php?option=com_content&view=article&id=163&Itemid=1897&lang=en. [Acedido em 01 12 2016].

[4 5] BECK, “IPC@CHIP® SC123/SC143 Getting started v1.5,” (2016). [Online]. Available: https://www.beck-ipc.com/download.php?file=95. [Acedido em 01 12 2016].

[4 6] BECK, “IPC@CHIP® RTOS API Documentation,” (2016). [Online]. Available: https://www.beck-ipc.com/api_files/scxxx/index.htm?cache=1481718957. [Acedido em 01 12 2016].

[4 7] Siemens, “SIMATIC S7 Modular Embedded Controlle Operating Instructions,” (2016). [Online]. Available: https://cache.industry.siemens.com/dl/files/353/46360353/att_67956/v1/s7_mec_operation_manual_en-US_en-US.pdf. [Acedido em 01 12 2016].

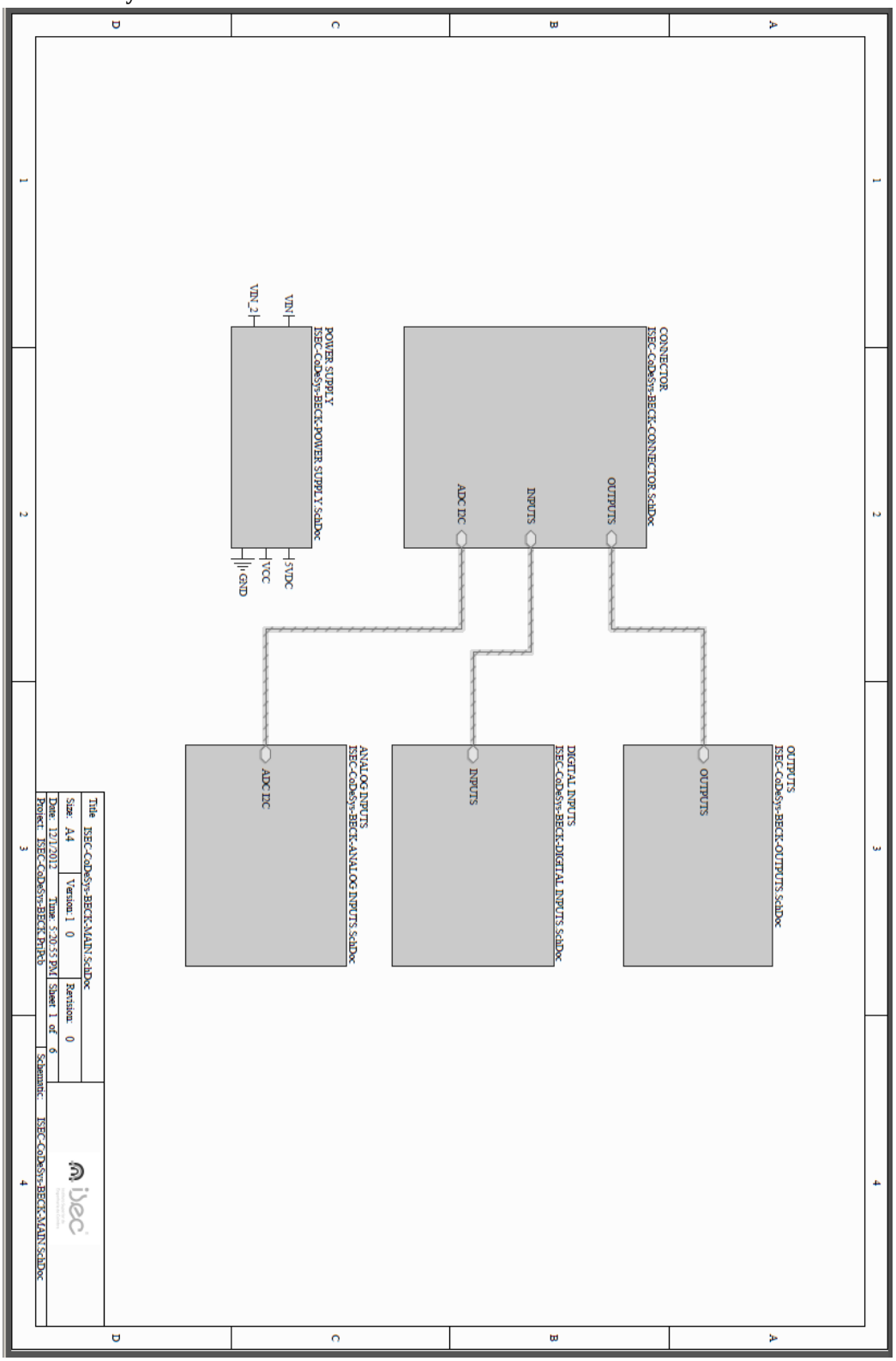
[4 8] Elesy, “CPU Modules - TC 506 C400 C Manual,” (2016). [Online]. Available: http://elesy.com/media/15601/tc%20506-507.pdf. [Acedido em 01 12 2016].

[4 9] R. O. Instalador, “A Compatibilidade Electromagnética de equipamentos electrónicos,” *Revista O Instalador,* vol. 204, 2013.

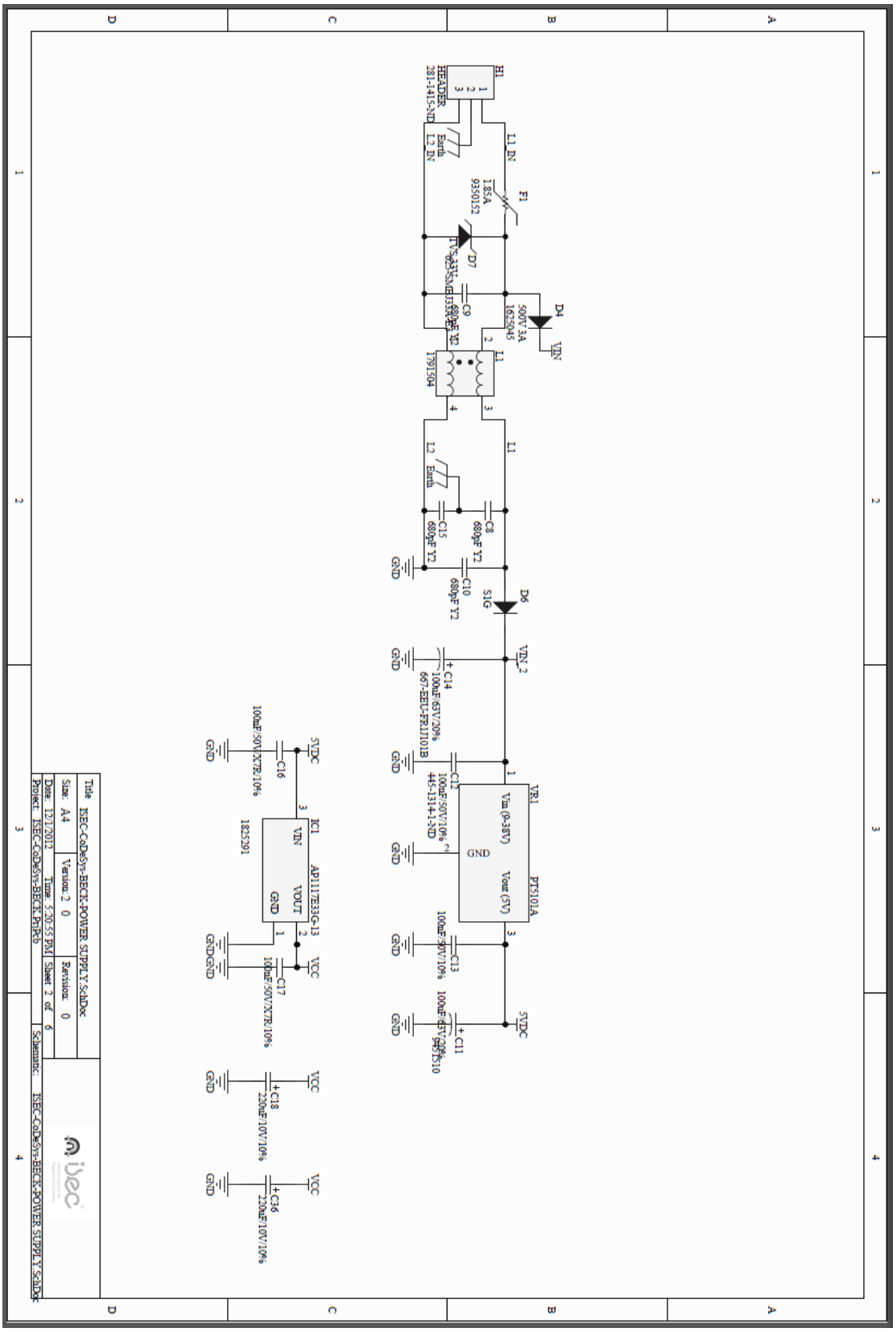
ANEXOS

ANEXO A - DESENHO DO ESQUEMA ELÉTRICO DO PCB INFERIOR

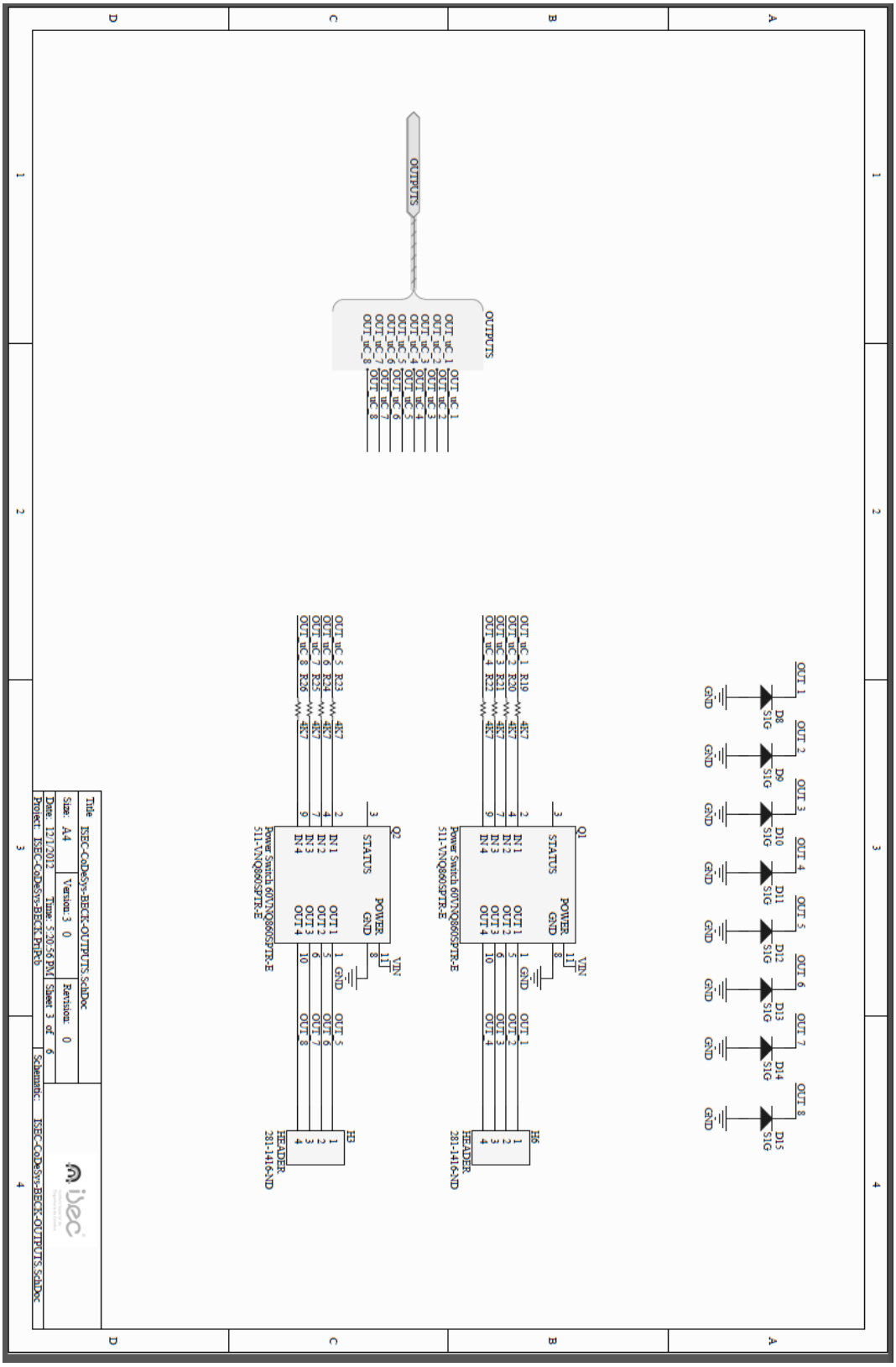
A.1 - ISEC-CoDeSys-BECK-MAIN.SchDoc



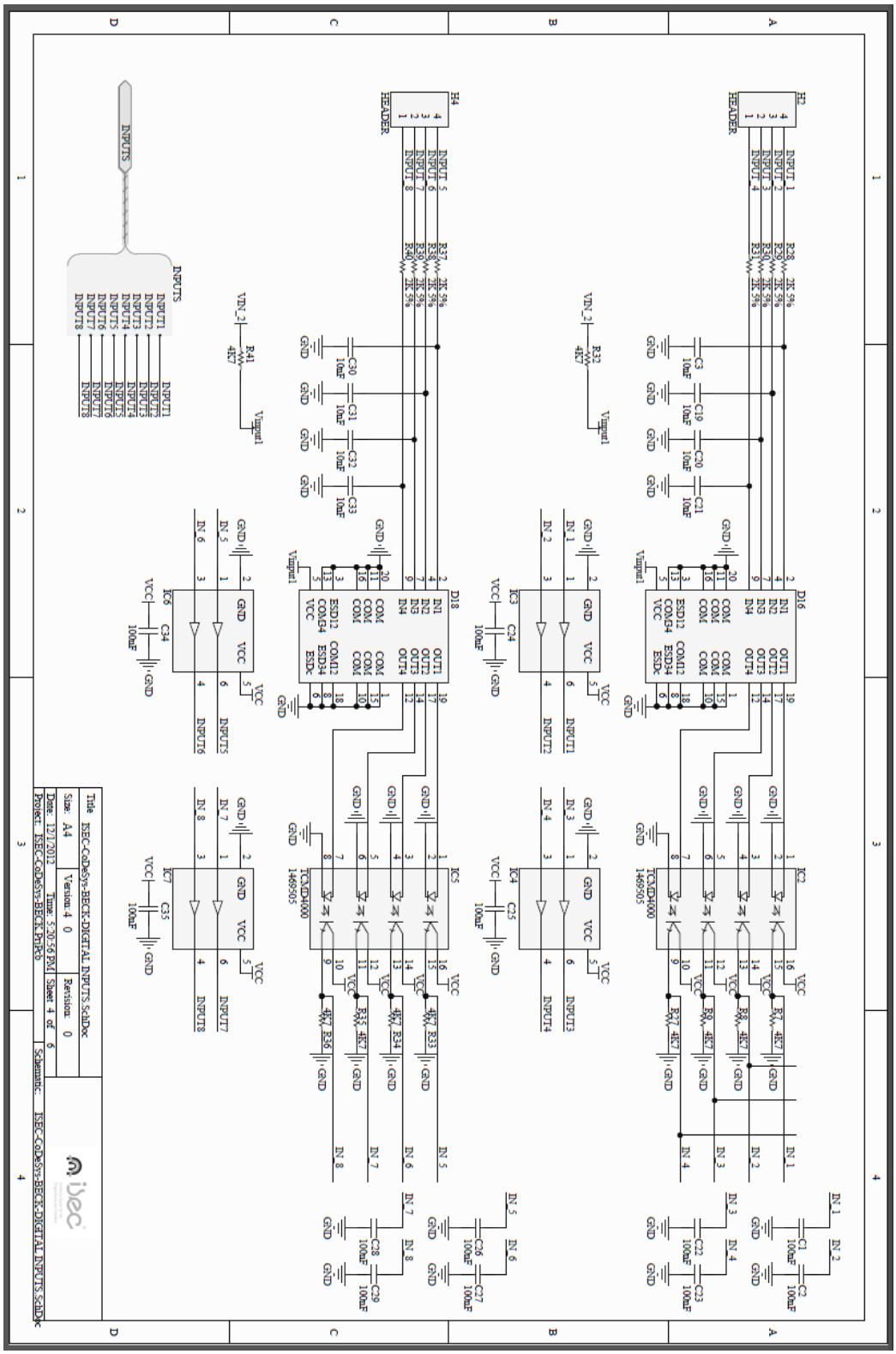
A.2 - ISEC-CoDeSys-BECK-POWER SUPPLY.SchDoc



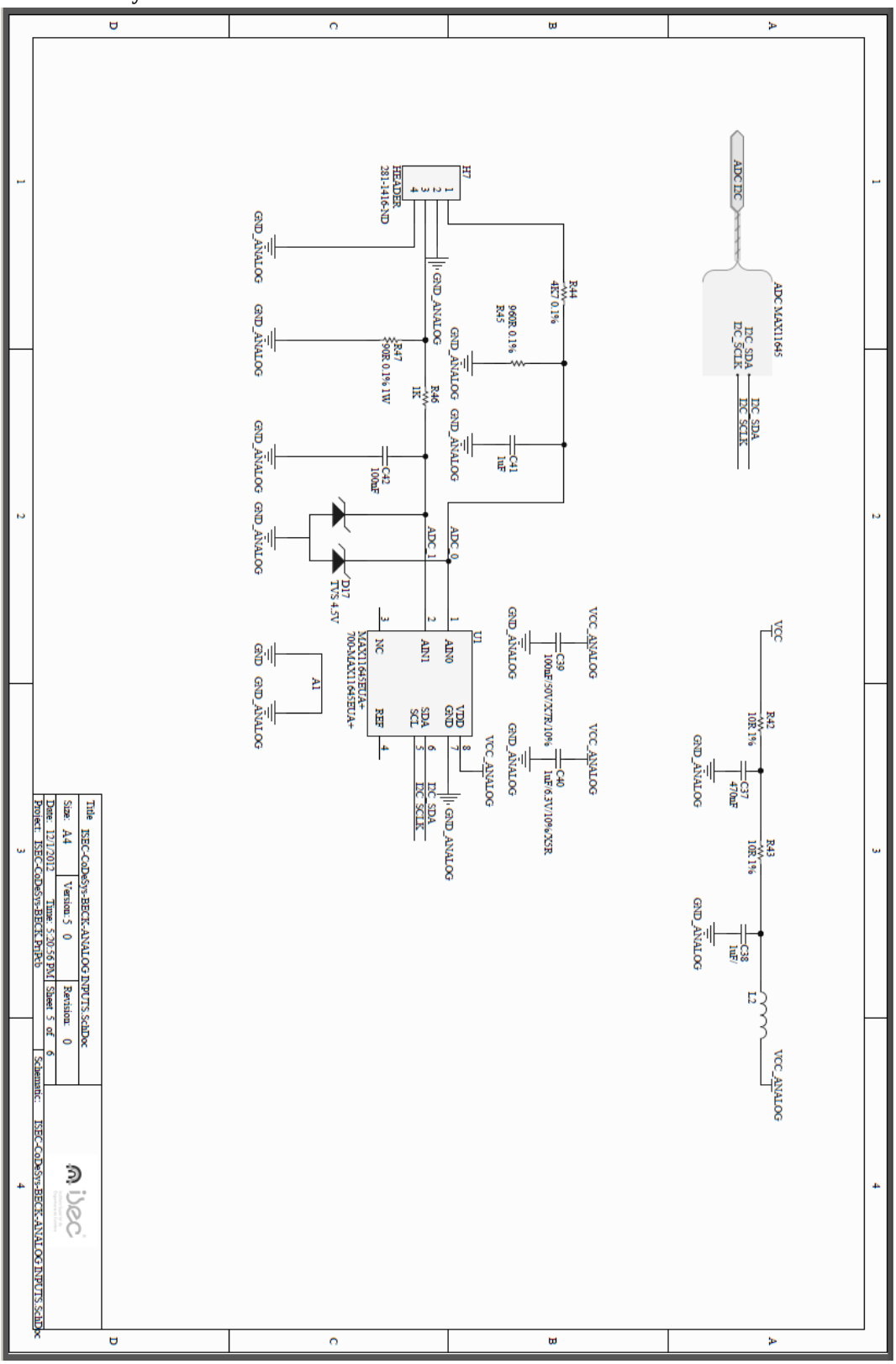
A.3 - ISEC-CoDeSys-BECK-OUTPUTS.SchDoc



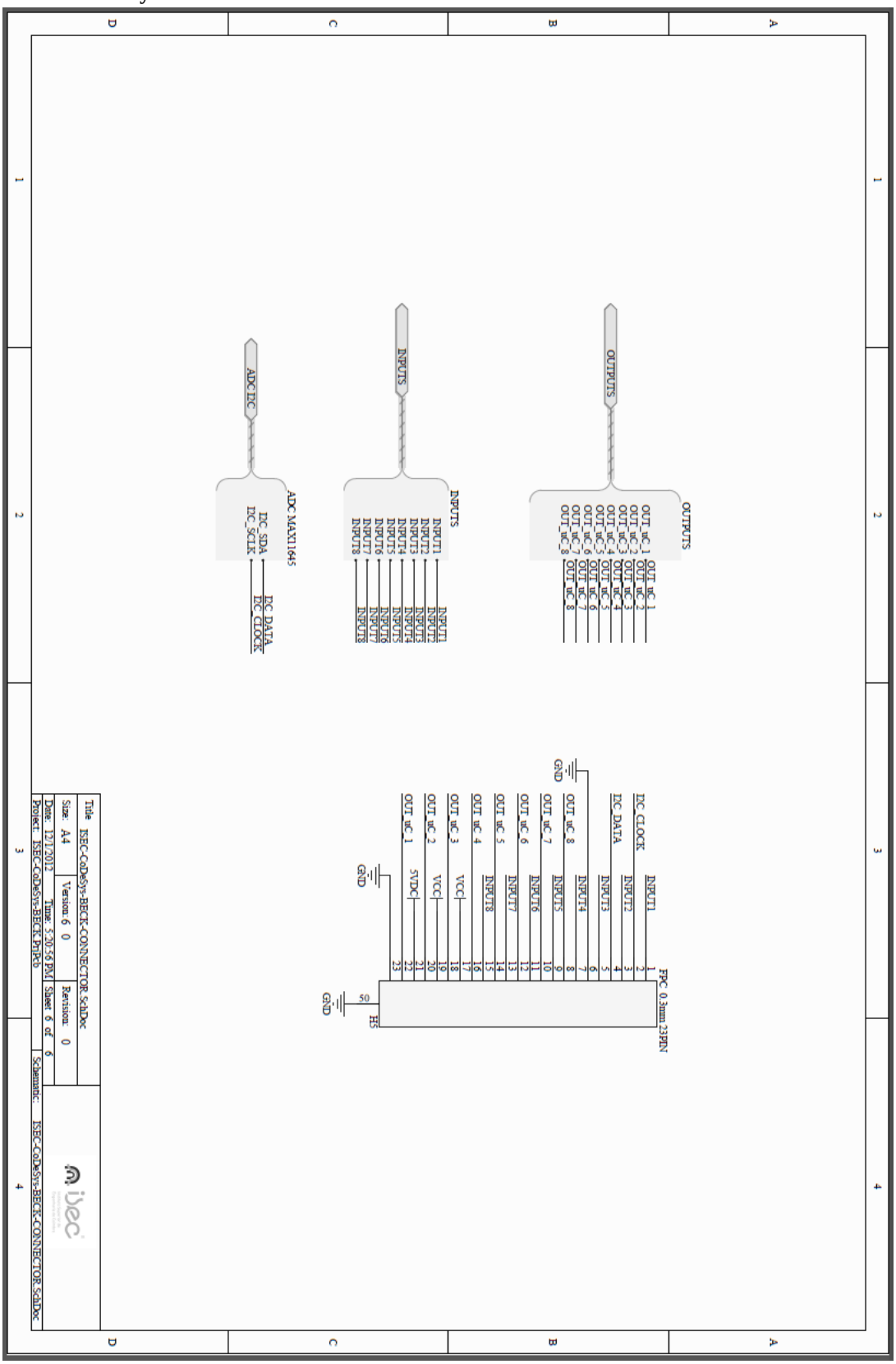
A.4 - ISEC-CoDeSys-BECK-DIGITAL INPUTS.SchDoc



A.5 - ISEC-CoDeSys-BECK-ANALOG INPUTS.SchDoc



A.6 - ISEC-CoDeSys-BECK-CONNECTOR.SchDoc



ANEXO B - LISTA DE COMPONENTES -PCB INFERIOR (*BOM- BILL OF MATERIALS*)

BOM - INTERNAL ASSEMBLY

Source Data From: **Print Date:**

ISEC-CoDeSys-BECK.PrjPcb 01-Dec-12

Project:

ISEC-CoDeSys-BECK.PrjPcb **Report Date:**

Variant: 12/1/2012

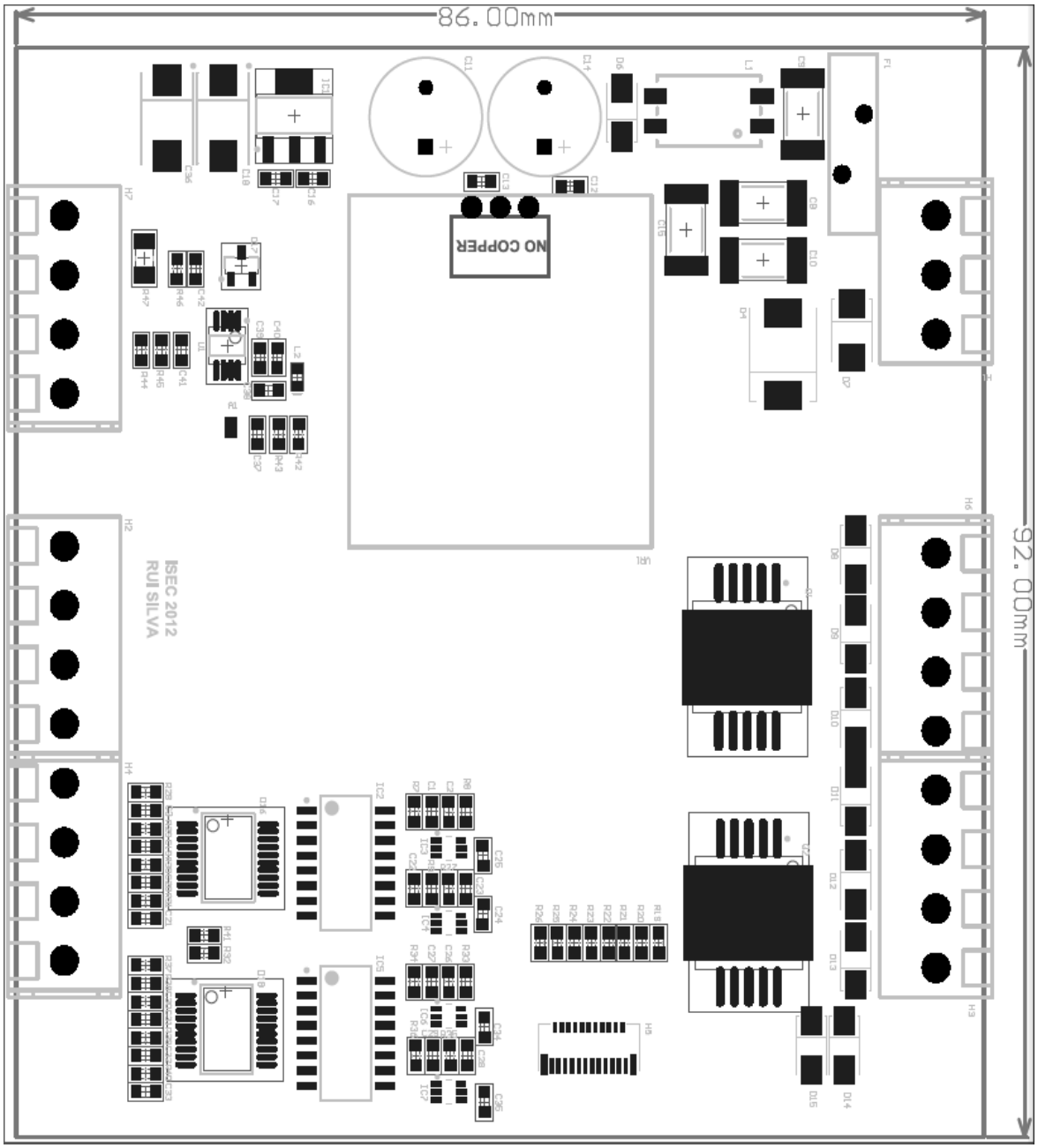
None

Instituto Superior de Engenharia de Coimbra

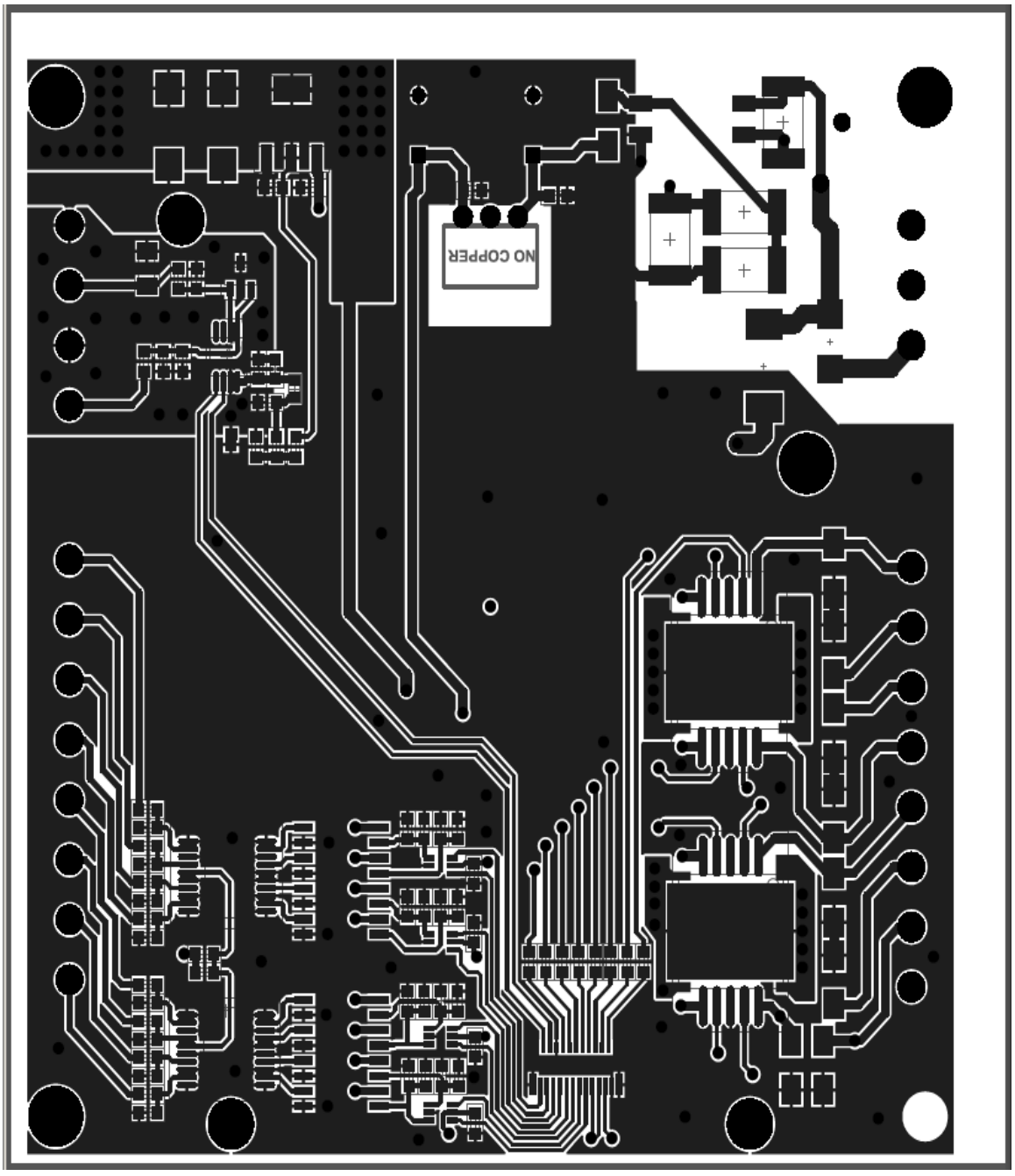
LibRef	Designator	Quantity	Manufacturer	Manufacturer Part Number	PCB Footprint
CAP CERAMIC 0603 100nF 50V 10% X7R	C1, C2, C22, C23, C24, C25, C26, C27, C28, C29, C34, C35, C42, C12, C13, C16, C17, C39	18	TDK	C1608X7R1H104KT	0603
CAP CERAMIC 0603 10nF 50V 10% X7R	C3, C19, C20, C21, C30, C31, C32, C33	8	MULTICOMP	MCCA000237	0603
CAP CERAMIC 2211 680pF 250VAC 10% Y2	C8, C9, C10, C15	4	JOHANSON DIELECTRICS	502R30W681KV3E	2211
CAP ELECTRO 10x13x5 100uF 62V 20%	C11, C14	2	Panasonic	667-EEU-FR1J101B	10x13x5
CAP TANTALUM 7343 220uF 10V 10% TAJD227K010RNJ	C18, C36	2	AVX	TAJD227K010RNJ	2617-7343 CASE D
CAP CERAMIC 0603 470nF 16V 10%	C37	1	TDK	C1608X5R1C474K	0603
CAP CERAMIC 0603 1uF 6.3V 10% X5R	C38, C40, C41	3	TDK	C1608X5R0J105K	0603
Diode 500V 3A DO-214A BES3H	D4	1	MULTICOMP	ES3H	SMC/DO-214AB
Diodo 400V 1A DO-214AC S1GA-E3	D6, D8, D9, D10, D11, D12, D13, D14, D15	9	Vishay	S1G-E3/61T	SMA
TVS 33V 600W SMB SMBJ33A-E3/5B	D7	1	Vishay	SMBJ33A-E3/5B	SMB
INPUT DIGITAL TSSOP-20 SMB CLT3-4BT6-TR	D16, D18	2	STMicroelectronics	CLT3-4BT6-TR	SMB
TVS 4.5V 40W SOT23 MMBZ6V8AL DUAL	D17	1	ON Semiconductor	MMBZ6V8AL	SOT23-3
PolySwitch RADIAL 1.85A 72V	F1	1	BOURNS	MF-RX185/72-0	RADIAL
CONNECTOR 3 5.00 LM	H1	1	Weidmuller	1715260000	3 5.00 LM
CONNECTOR 4 5.00 LM	H2, H3, H4, H6, H7	5	Weidmuller	9993300000	4 5.00 LM
FFC 501912-2390	H5	1	Molex	501912-2390	23P FPC
LDO 3.3V 1A SOT223 AP1117E33G-13	IC1	1	Diodes	AP1117E33G-13	SOT-223
Optocoupler TCMD4000	IC2, IC5	2	Vishay	TCMD4000	SOP-16
Schmitt Trigger Buffer 2x SC-70 NC7WZ17P6X	IC3, IC4, IC6, IC7	4	Fairchild	NC7WZ17P6X	SC-70
COMMON MODE CHOKE 10uH 1.6A SRF0905-100Y	L1	1	Bourns	SRF0905-100Y	0905
FERRITE CHIP BEAD 500mA 60R 0603 MMZ1608Y600B	L2	1	TDK	MMZ1608Y600B	0603
POWER SWITCH QUAD 36V 250mA PowerSO-10 VNQ860SPTR-E	Q1, Q2	2	STMicroelectronics	VNQ860SPTR-E	PowerSO-10
Resistor 0603 4K7 5% 200ppm	R7, R8, R9, R19, R20, R21, R22, R23, R24, R25, R26, R27, R32, R33, R34, R35, R36, R41, R44, R45	20	PANASONIC	ERJ3GEYJ472V	0603
Resistor 0603 2K 5% 200ppm	R28, R29, R30, R31, R37, R38, R39, R40	8	PANASONIC	ERJ3GEYJ202V	0603
Resistor 0603 10R 1%	R42, R43	2	VISHAY	CRCW060310R0FKEA	0603
Resistor 0603 1K 1% 100ppm	R46, R47	2	VISHAY	CRCW06031K00FKEA	0603
ADC_MAX11645EUA+	U1	1	Maxim	MAX11645EUA+	uMax-8 TSOP-8
PT5101A	VR1	1	Texas Instruments	PT5101A	

ANEXO C - DESENHO DO PCB INFERIOR

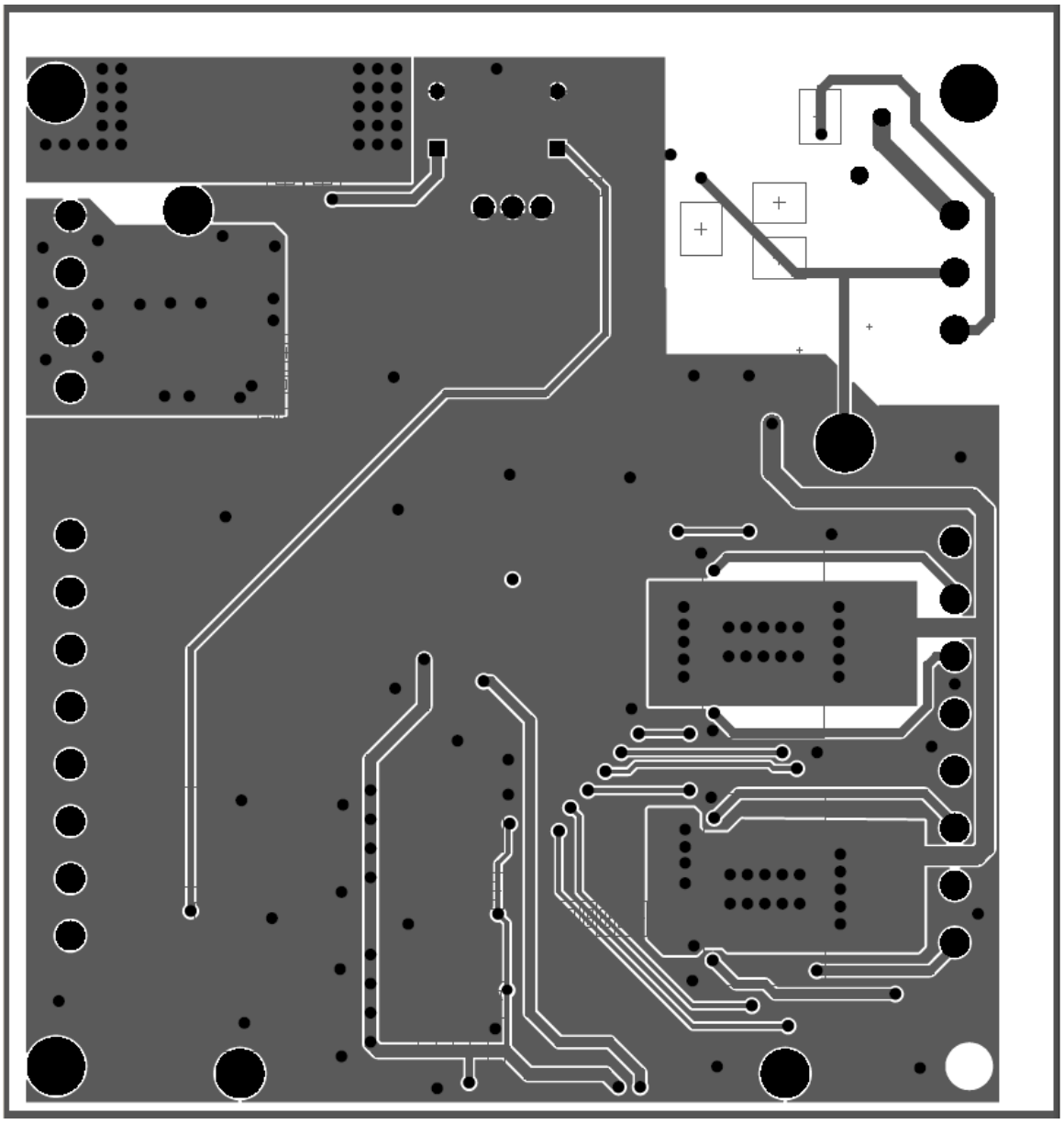
C.1 - Posicionamento dos Componentes e dimensões do PCB Inferior



C.2 - *Top Layer* (camada dos componentes) do PCB Inferior



C.3 - *Bottom Layer* (camada da soldadura THT) do PCB Inferior



C.4 - Relatório de erros referente ao PBC Inferior

Design Rules Verification Report

Filename : C:\Dropbox\ESTAGIO ISEC\relatotio\HW\ISEC-CoDeSys-BECK.PcbDoc

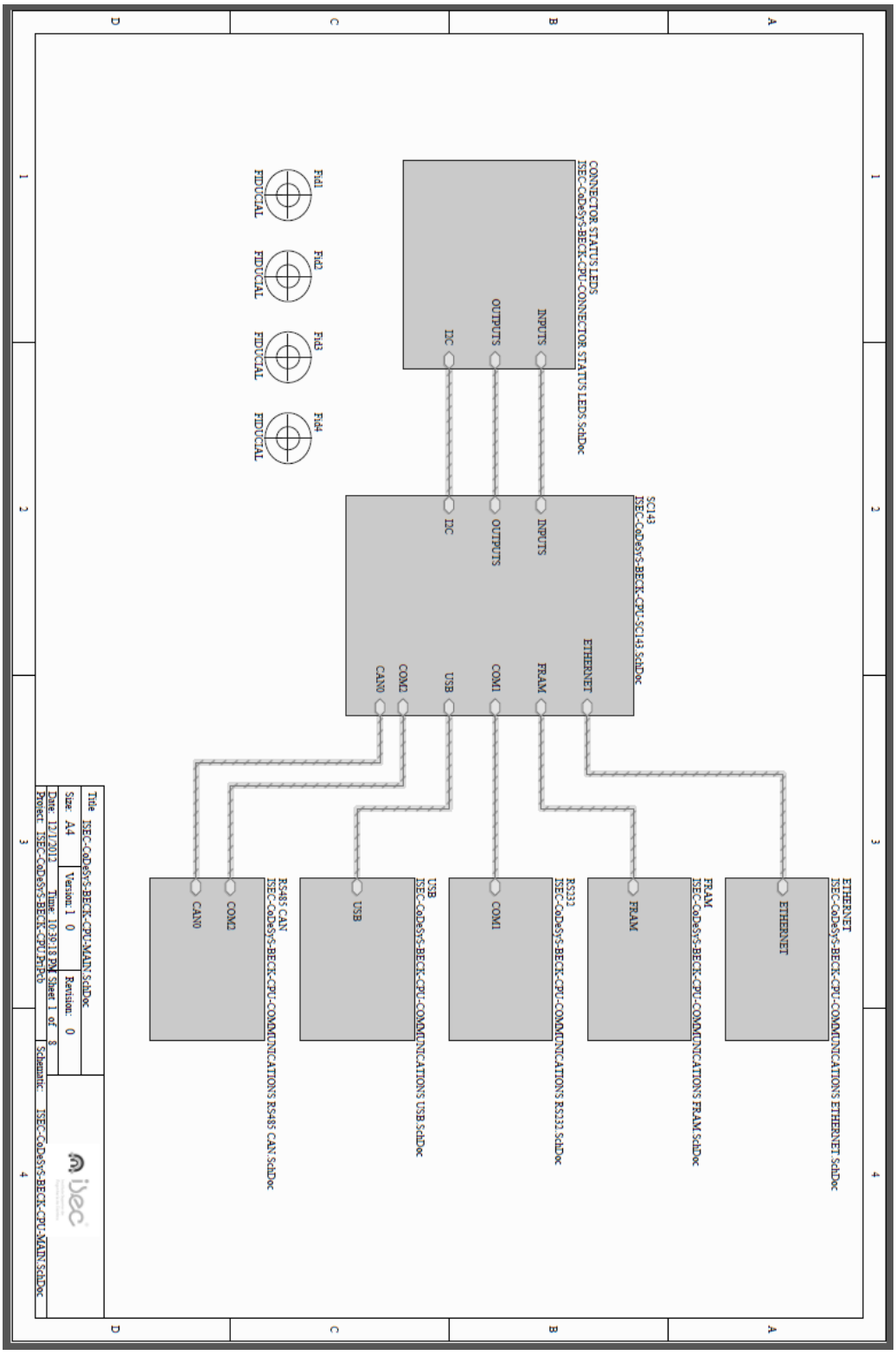
Warnings 0
Rule Violations 0

Warnings	
Total	0

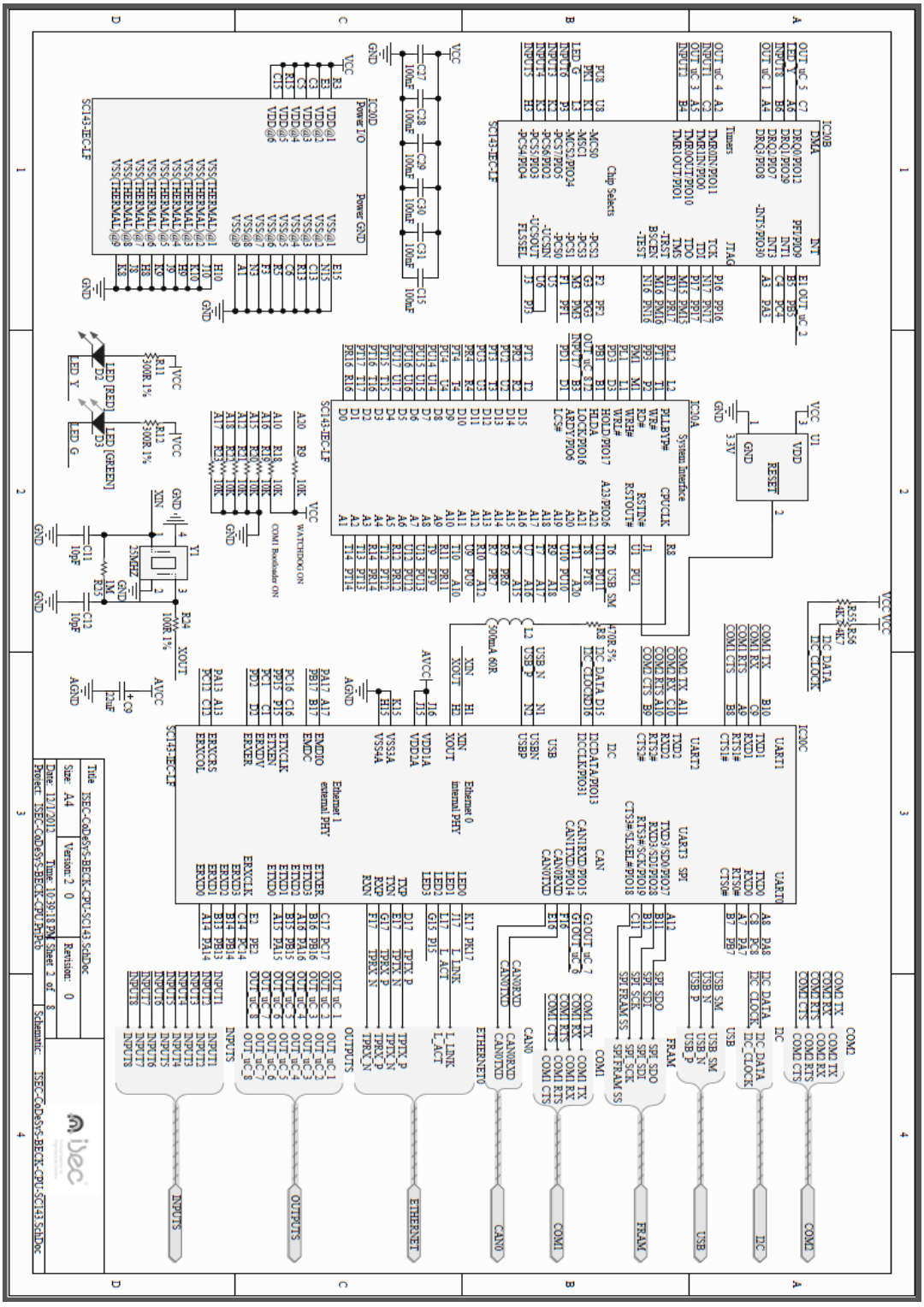
Rule Violations	
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Clearance Constraint (Gap=0.2mm) (All),(All)	0
Power Plane Connect Rule(Relief Connect)(Expansion=0.508mm) (Conductor Width=0.254mm) (Air Gap=0.254mm)	0
Width Constraint (Min=0.254mm) (Max=2mm) (Preferred=0.254mm) (All)	0
Height Constraint (Min=0mm) (Max=25.4mm) (Prefered=12.7mm) (All)	0
Hole Size Constraint (Min=0.025mm) (Max=5mm) (All)	0
Hole To Hole Clearance (Gap=0.254mm) (All),(All)	0
Net Antennae (Tolerance=0mm) (All)	0
Total	0

ANEXO D - DESENHO DO ESQUEMA ELÉTRICO DO PCB SUPERIOR

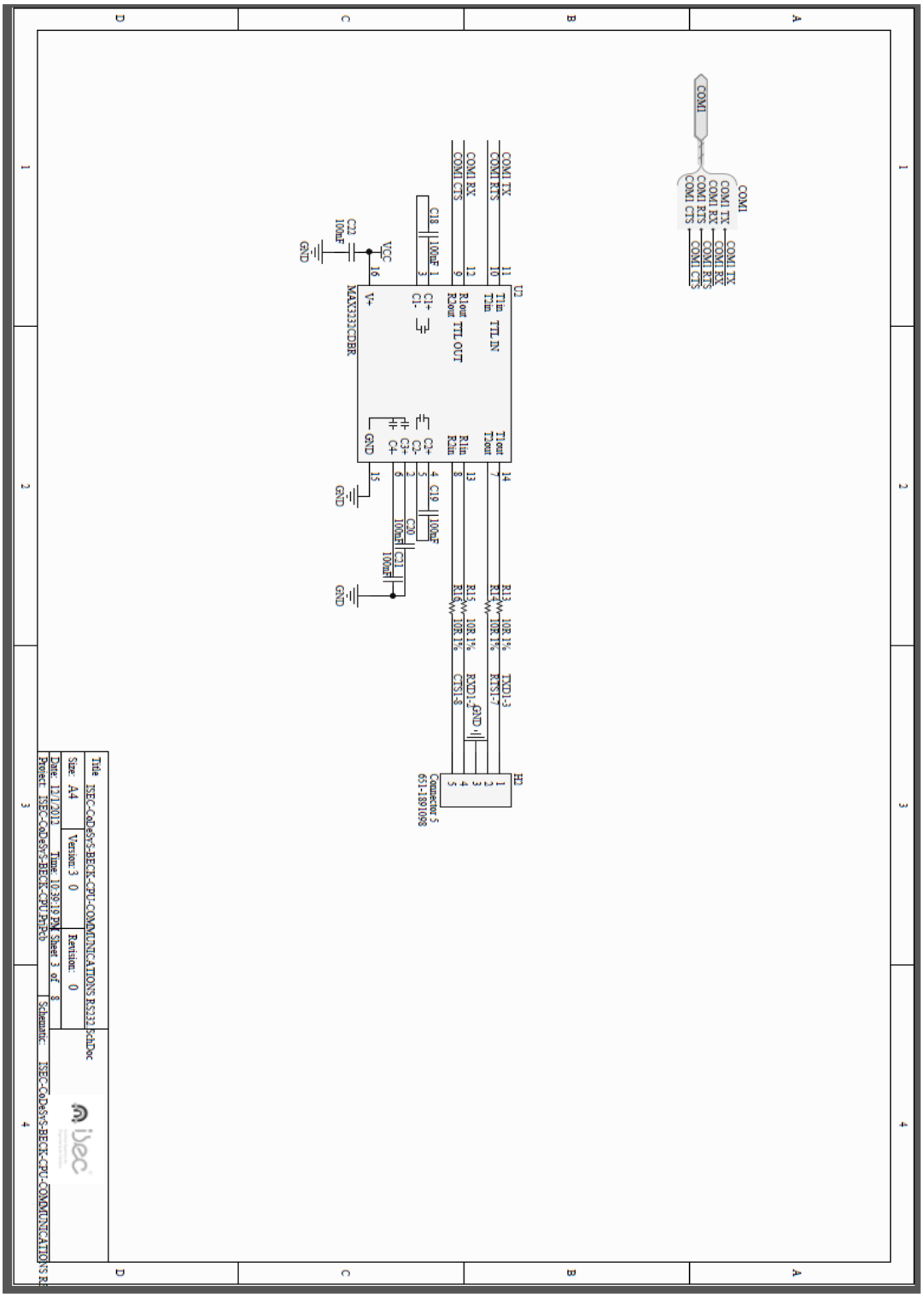
D.1 - ISEC-CoDeSyS-BECK-CPU-MAIN.SchDoc



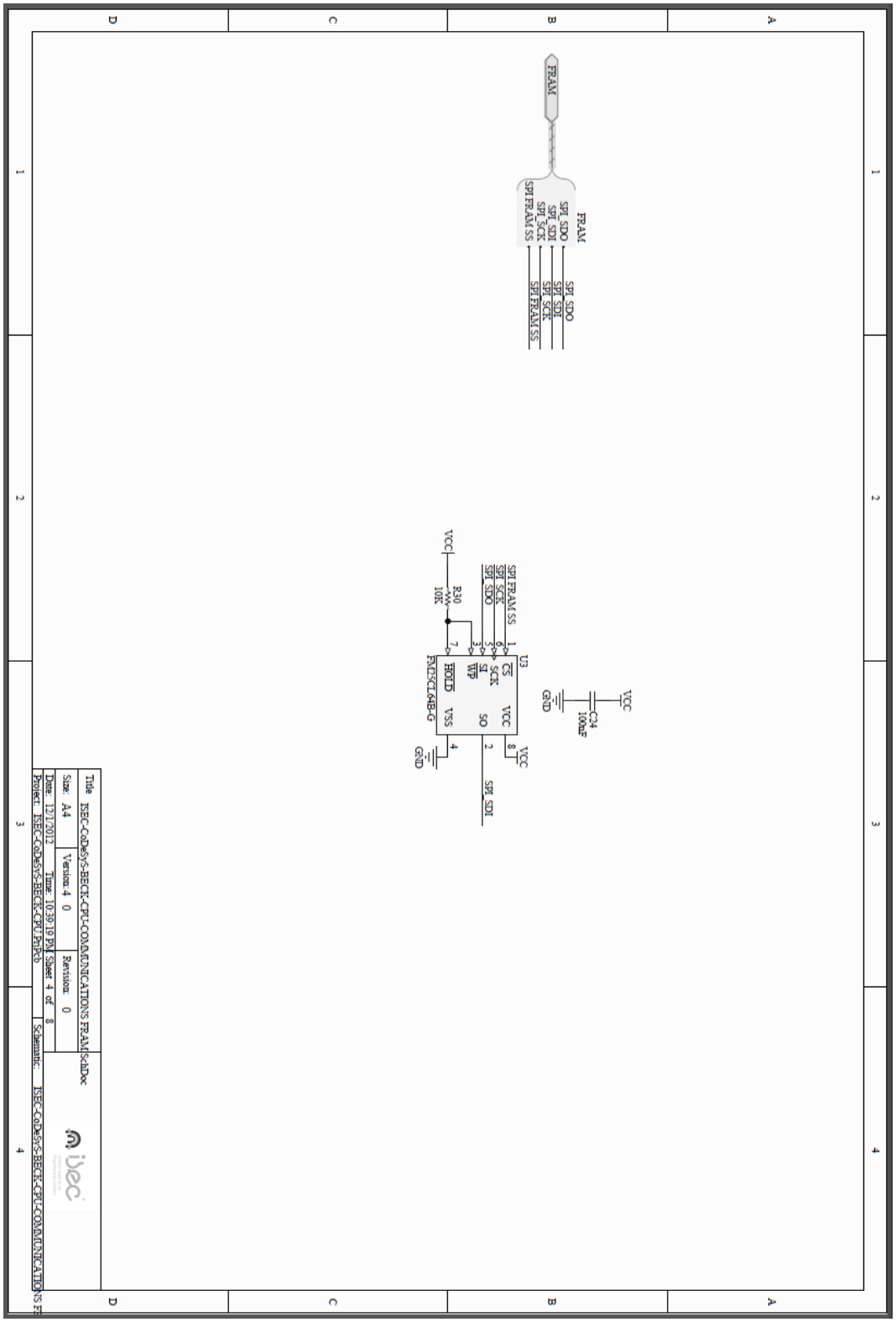
D.2 - ISEC-CoDeSyS-BECK-CPU-SC143.SchDoc



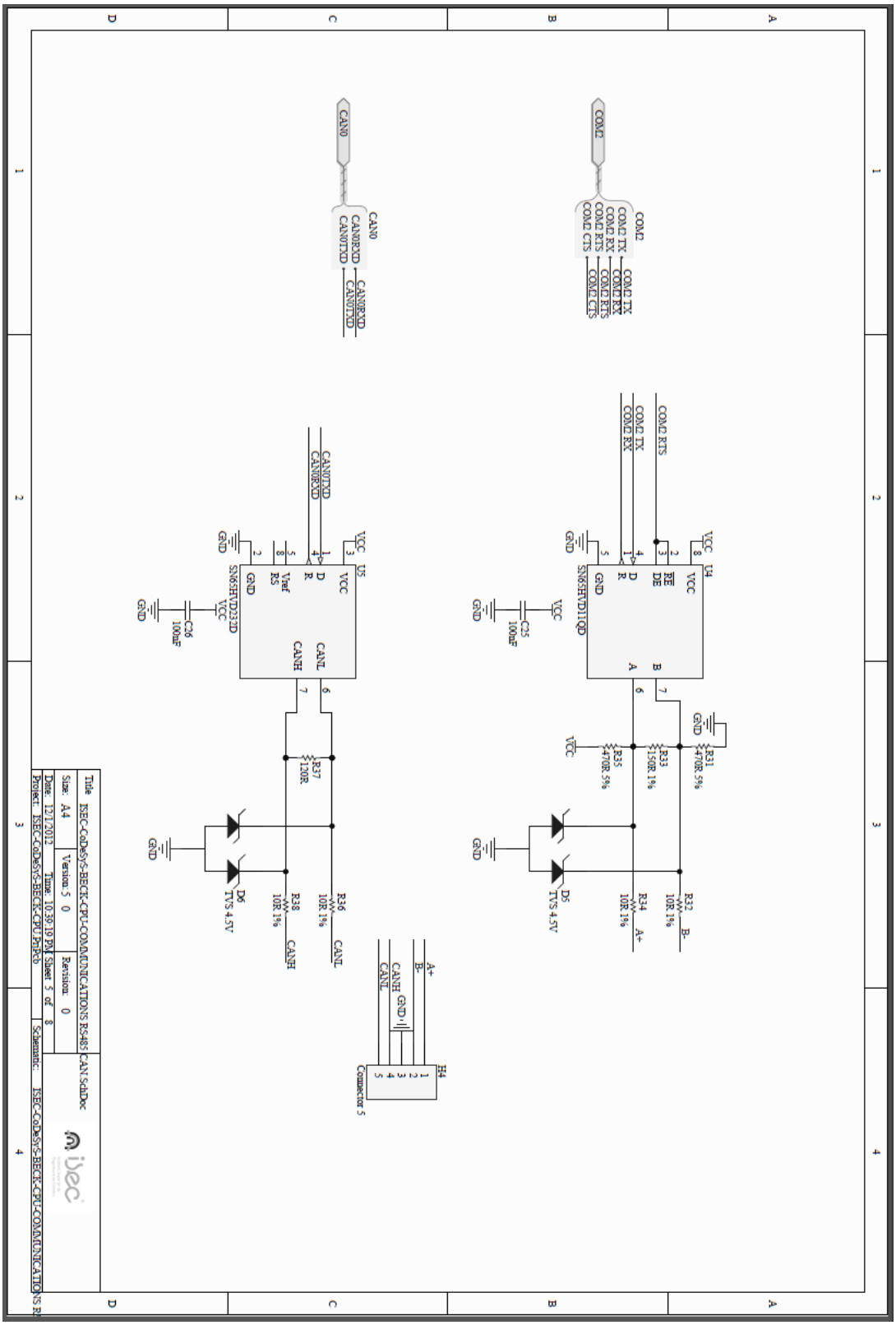
D.3 - ISEC-CoDeSyS-BECK-CPU-COMMUNICATIONS RS232.SchDoc



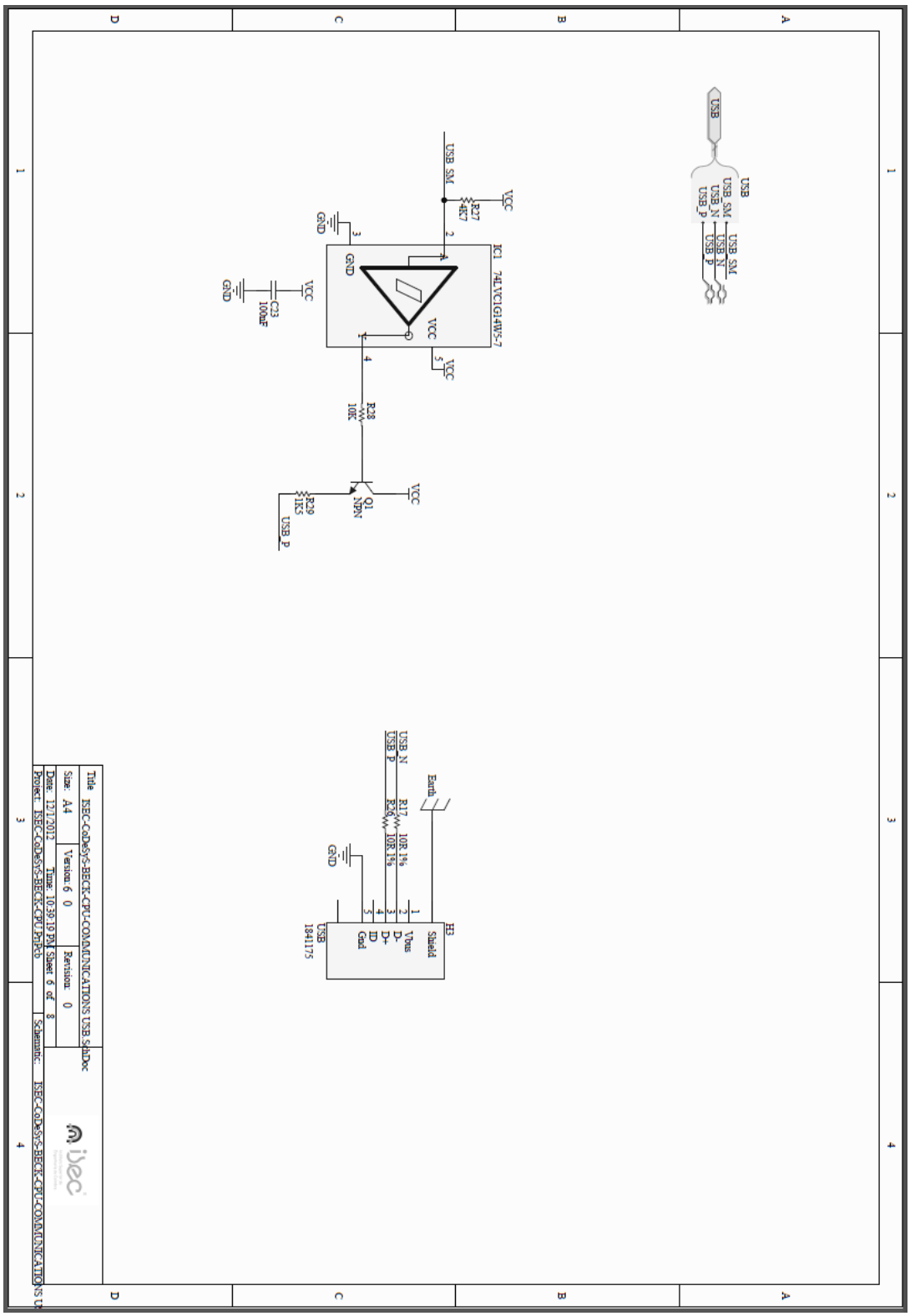
D.4 - ISEC-CoDeSyS-BECK-CPU-COMMUNICATIONS FRAM.SchDoc



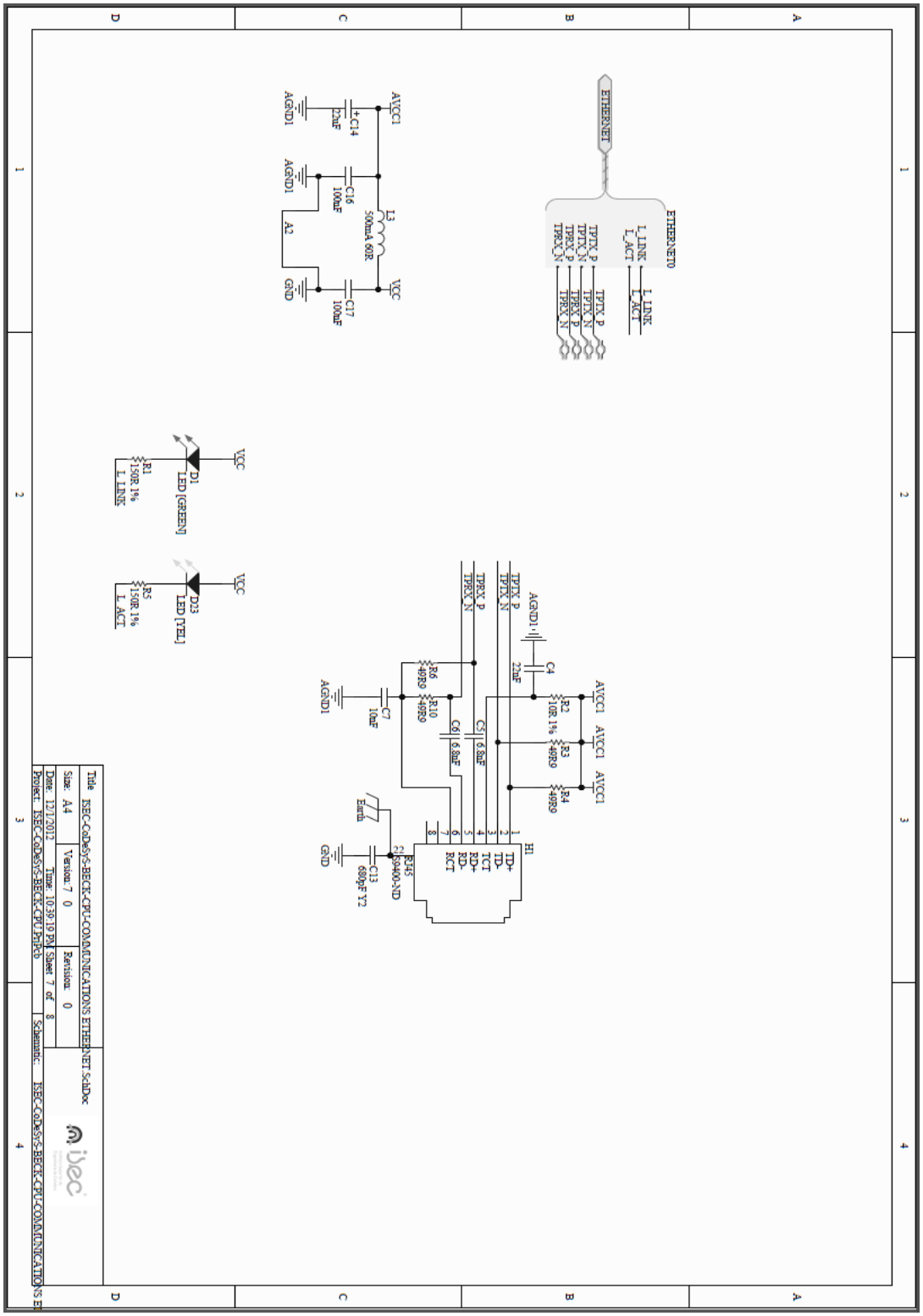
D.5 - ISEC-CoDeSyS-BECK-CPU-COMMUNICATIONS RS485 CAN.SchDoc



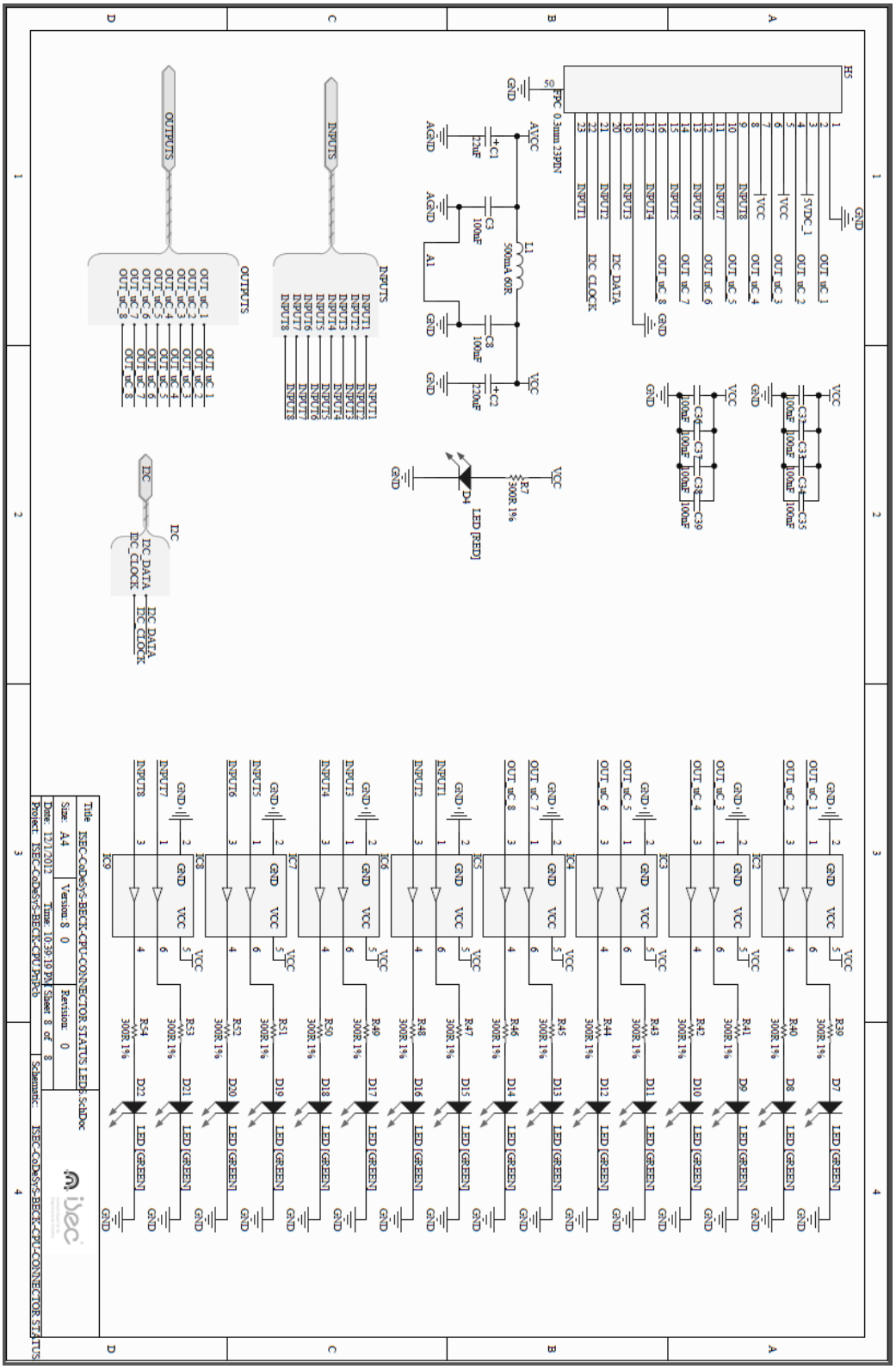
D.6 - ISEC-CoDeSyS-BECK-CPU-COMMUNICATIONS USB.SchDoc



D.7 - ISEC-CoDeSyS-BECK-CPU-COMMUNICATIONS ETHERNET.SchDoc



D.8 - ISEC-CoDeSyS-BECK-CPU-*CONNECTOR STATUS LEDS*.SchDoc



ANEXO E - LISTA DE COMPONENTES - PCB SUPERIOR (*BOM - BILL OF MATERIALS*)

BOM - INTERNAL ASSEMBLY

Source Data From:
ISEC-CoDeSyS-BECK-CPU.PrjPcb

Print Date:
01-Dec-12

Project:
ISEC-CoDeSyS-BECK-CPU.PrjPcb

Report Date:
12/1/2012

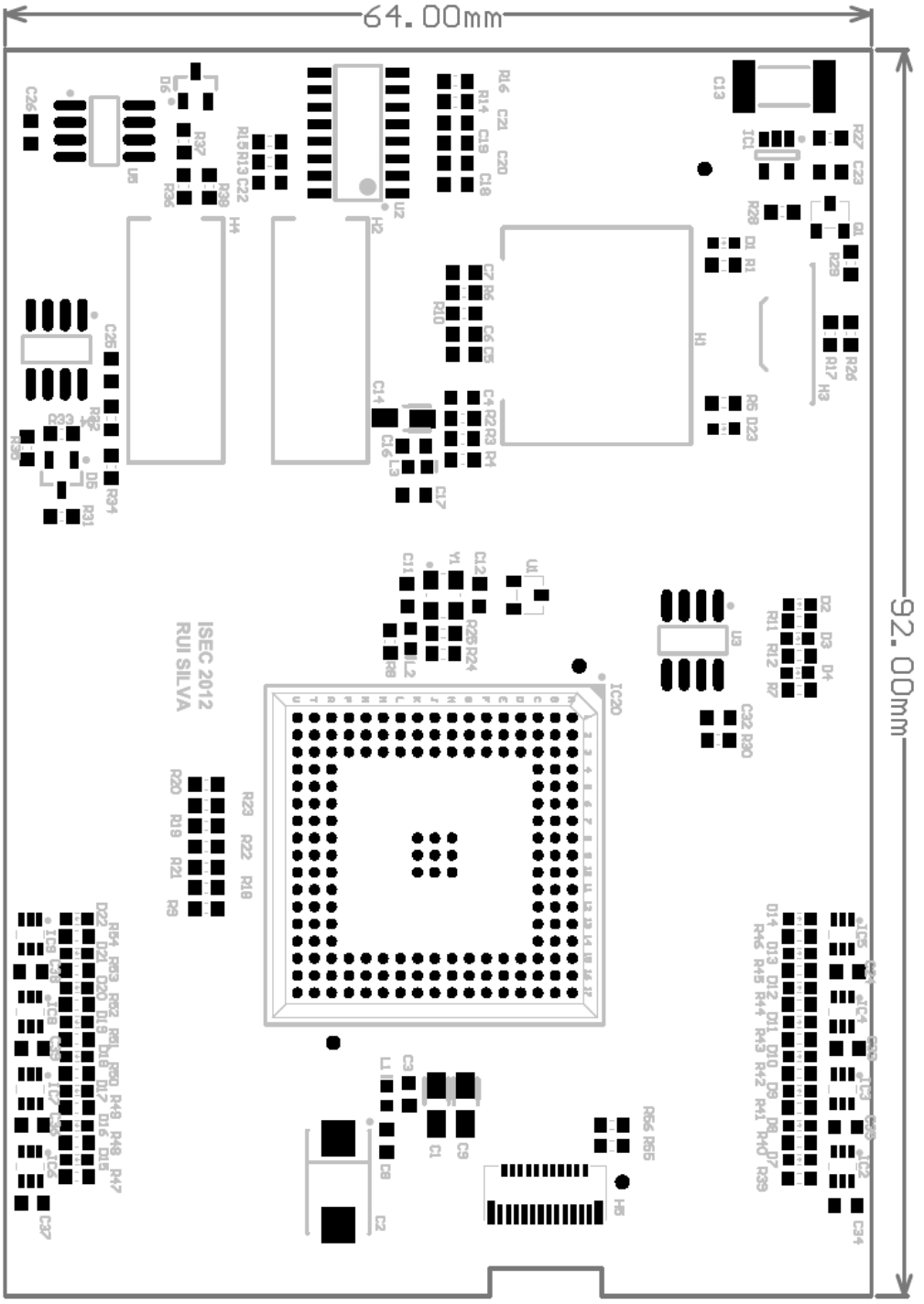
Variant:
None

Instituto Superior de Engenharia de Coimbra

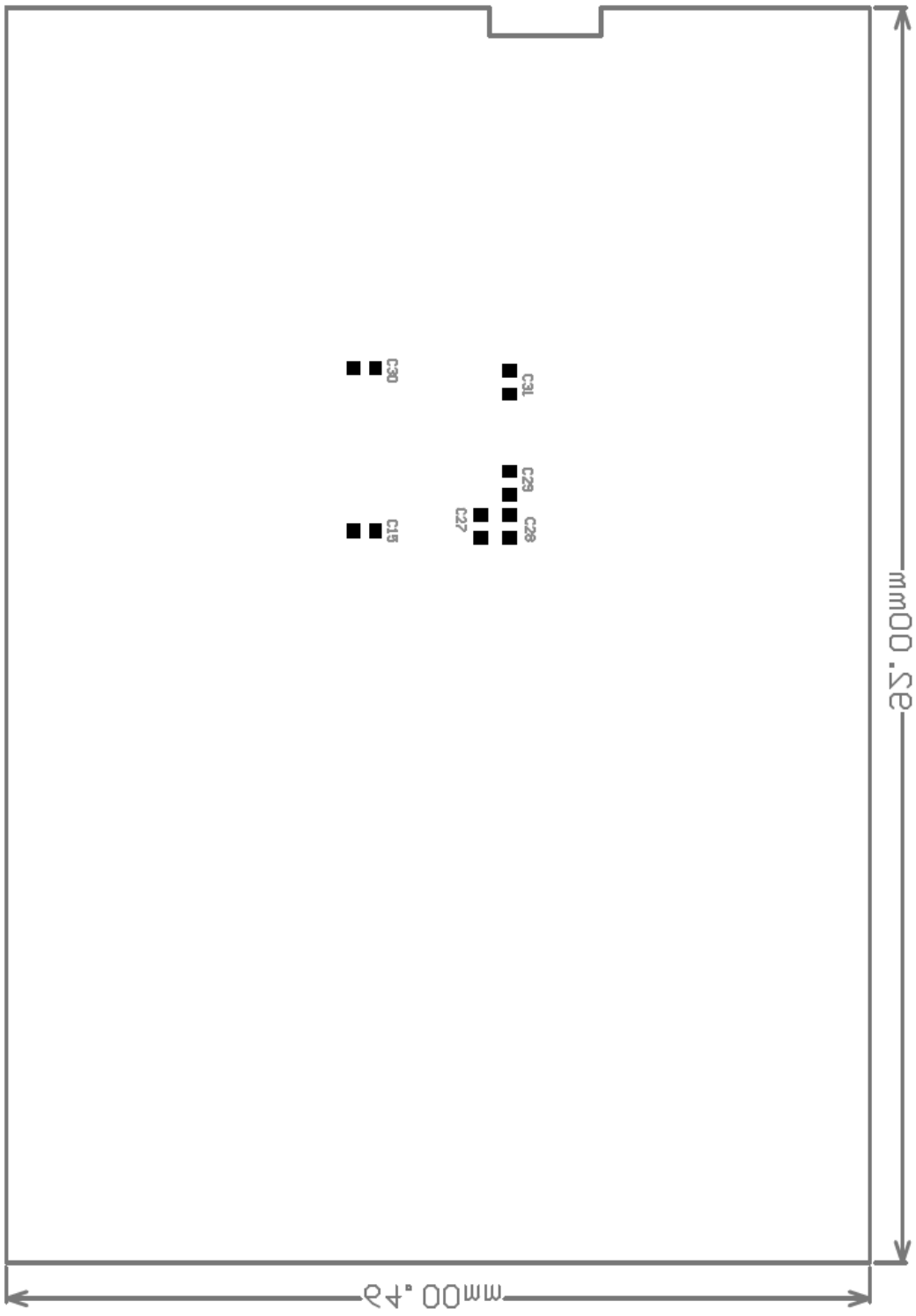
LibRef	Designator	Quantity	Manufacturer	Manufacturer Part Number	PCB Footprint
CAP TANTALUM 3216 22uF 10V 10%	C1, C9, C14	3	VISHAY	293D226X9010A2TE3	1206-3216
CAP TANTALUM 7343 220uF 10V 10%	C2	1	AVX	TAJD227K010RNJ	2617-7343
CAP CERAMIC 0603 100nF 50V 10% X7R	C3, C8, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C28, C29, C30, C31, C32, C33, C34, C35, C36, C37, C38, C39	27	TDK	C1608X7R1H104KT	0603
CAP CERAMIC 0603 10nF 50V 10% X7R	C4	1	MULTICOMP	MCCA000237	0603
CAP CERAMIC 0603 68nF 50V 10% X7R	C5, C6	2	TDK	C1608X7R1H683K	0603
CAP CERAMIC 0603 10nF 50V 10% X7R	C7	1	MULTICOMP	MCCA000237	0603
CAP CERAMIC 0603 10pF 50V 5% COP NPO	C11, C12	2	TDK	C1608C0G1H100D	0603
CAP CERAMIC 2211 680pF 250VAC 10% Y2	C13	1	JOHANSON DIELECTRICS	502R30W681KV3E	2211
LED 0603 GREEN	D1, D3, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22	18	Kingbright	APTD1608ZGC	0603
LED 0603 RED	D2, D4	2	Dialight	598-8010-107F	0603
LED 0603 YEL	D23	1	Dialight	598-8040-107F	0603
TVS 4.5V 40W SOT23 MMBZ6V8AL DUAL	D5, D6	2	ON Semiconductor	MMBZ6V8AL	SOT23-3
CONNECTOR MODULAR JACKS	H1	1	Sullins Connector	20021321-00006C4LF	RJ45
CONNECTOR 3.5MM 5 180 1891098	H2, H4	2	Phoenix Contact	1891098	5 3.5mm
CONNECTOR USB MINI B VERT 651005136421	H3	1	WURTH ELEKTRONIK	651005136421	MINI USB B
FFC 501912-2390	H5	1	Molex	501912-2390	23P FPC
Schmitt Trigger Inverter	IC1	1	Diodes Inc	74LVC1G14W5-7	SOT-23-5
Schmitt Trigger Buffer	IC2, IC3, IC4, IC5, IC6, IC7, IC8, IC9	8	Fairchild	NC7WZ17P6X	SC-70
SC143-IEC-LF	IC20	1	Beck-ipc	SC143-IEC-LF	BGA-177
FERRITE CHIP BEAD	L1, L2, L3	3	TDK	MMZ1608Y600B	0603
Transistor Bipolar NPN BC817	Q1	1	NXP	BC817	SOT23-3
Resistor 0603 150R 1% 100ppm	R1, R5, R33	3	VISHAY	CRCW0603150RFKEA	0603
Resistor 0603 10R 1% 100ppm	R2, R13, R14, R15, R16, R17, R26, R32, R34, R36, R38	11	VISHAY	CRCW060310R0FKEA	0603
Resistor 0603 1K 1% 100ppm	R3, R4, R6, R10	4	VISHAY	CRCW06031K00FKEA	0603
Resistor 0603 300R 1% 100ppm	R7, R11, R12, R39, R40, R41, R42, R43, R44, R45, R46, R47, R48, R49, R50, R51, R52, R53, R54	19	PANASONIC	ERJ3EKF3000V	0603
Resistor 0603 470R 5% 200ppm	R8, R31, R35	3	PANASONIC	ERJ3GEYJ471V	0603
Resistor 0603 10K 5% 200ppm	R9, R18, R19, R20, R21, R22, R23, R28, R29, R30	10	VISHAY	CRCW060310K0JNEA	0603
Resistor 0603 100R 1% 200ppm	R24, R37	2	MULTICOMP	MC 0.063W 0603 1%	0603
Resistor 0603 1M 1% 100ppm	R25	1	WELWYN	ASC0603-1M0FT5	0603
Resistor 0603 4K7 5% 200ppm	R27, R55, R56	3	PANASONIC	ERJ3GEYJ472V	0603
RESET MONITOR -3.3V	U1	1	MICROCHIP	TCM809TVNB713	SOT-23B
RS-232 TRANSCEIVER	U2	1	TEXAS INSTRUMENTS	MAX3232CDBR	SSOP-16
25AA1024-I/SM	U3	1	RAMTRON	FM25CL64B-G	SOIC-8
Transceiver RS485	U4	1	TEXAS INSTRUMENTS	SN65HVD11QD	SOIC-8
Transceiver CAN	U5	1	TEXAS INSTRUMENTS	SN65HVD232D	SOIC-8
CRYSTAL 25MHz	Y1	1	ABRACON	ABM8G-25.000MHZ-B4Y-T	3.2 mmx2.5 mm

ANEXO F - DESENHO DO PCB SUPERIOR

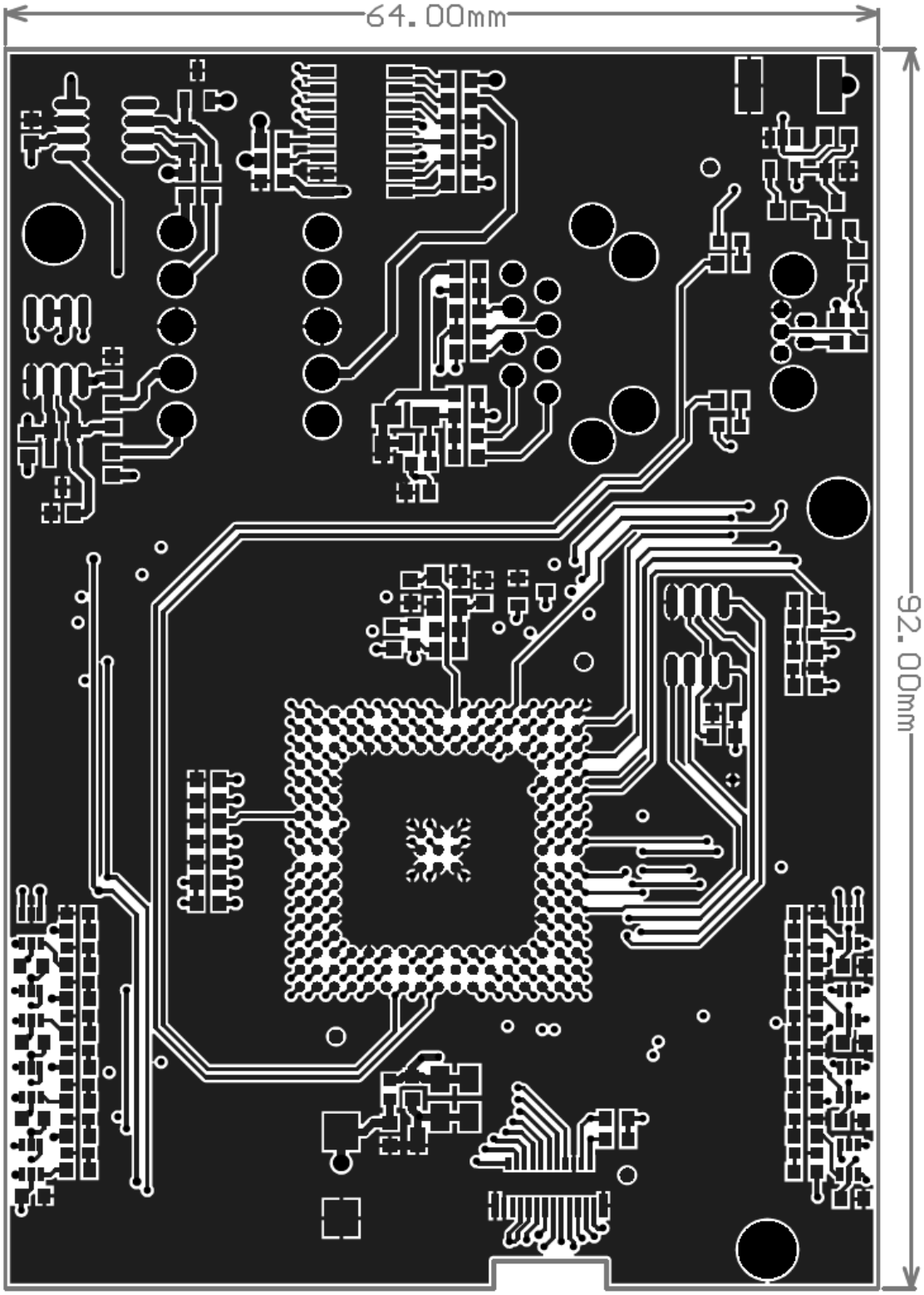
F.1 - Posicionamento dos Componentes e dimensões do PCB Superior (*Top Layer*)



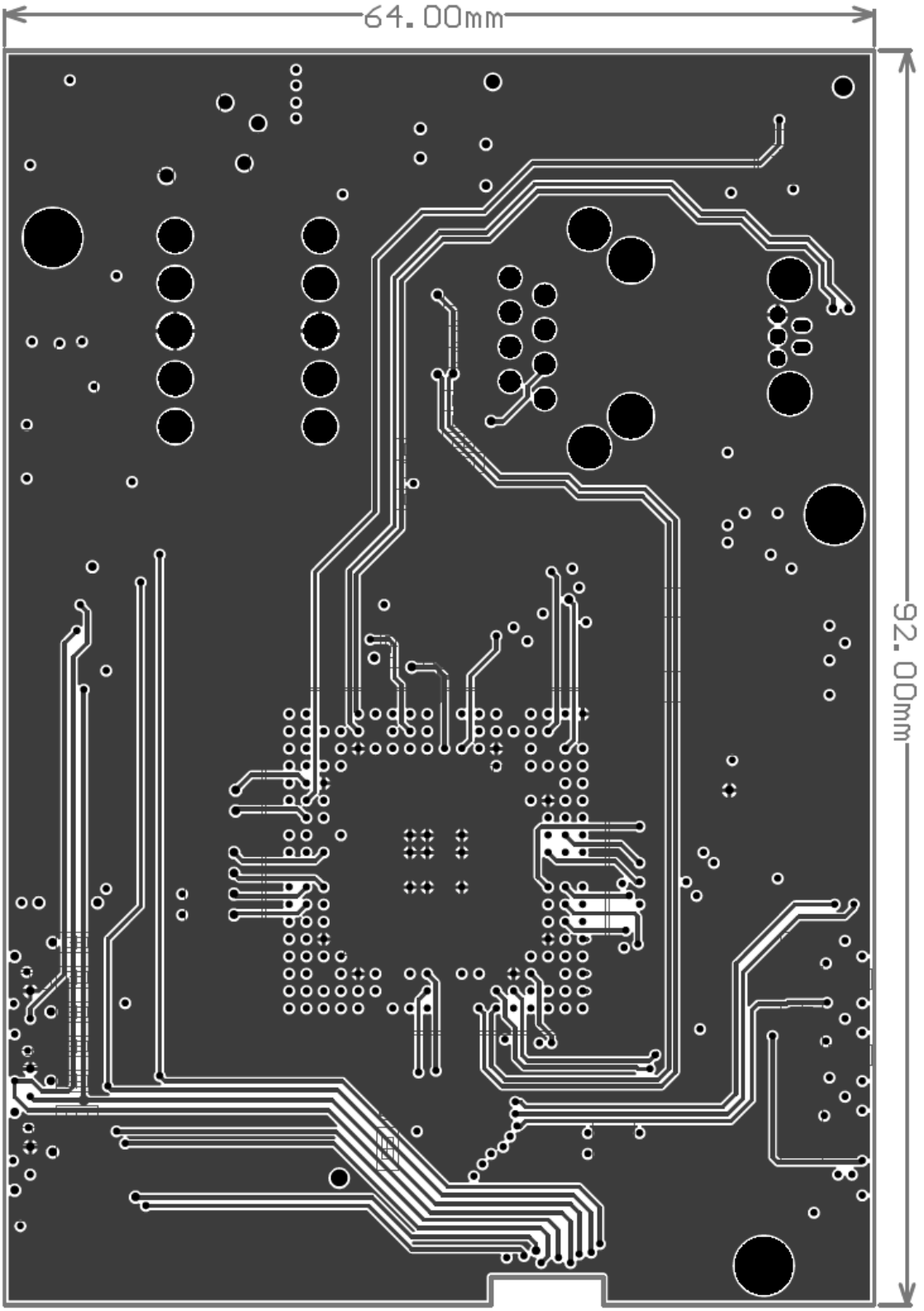
F.2 - Posicionamento dos Componentes e dimensões do PCB Superior (*Bottom Layer*)



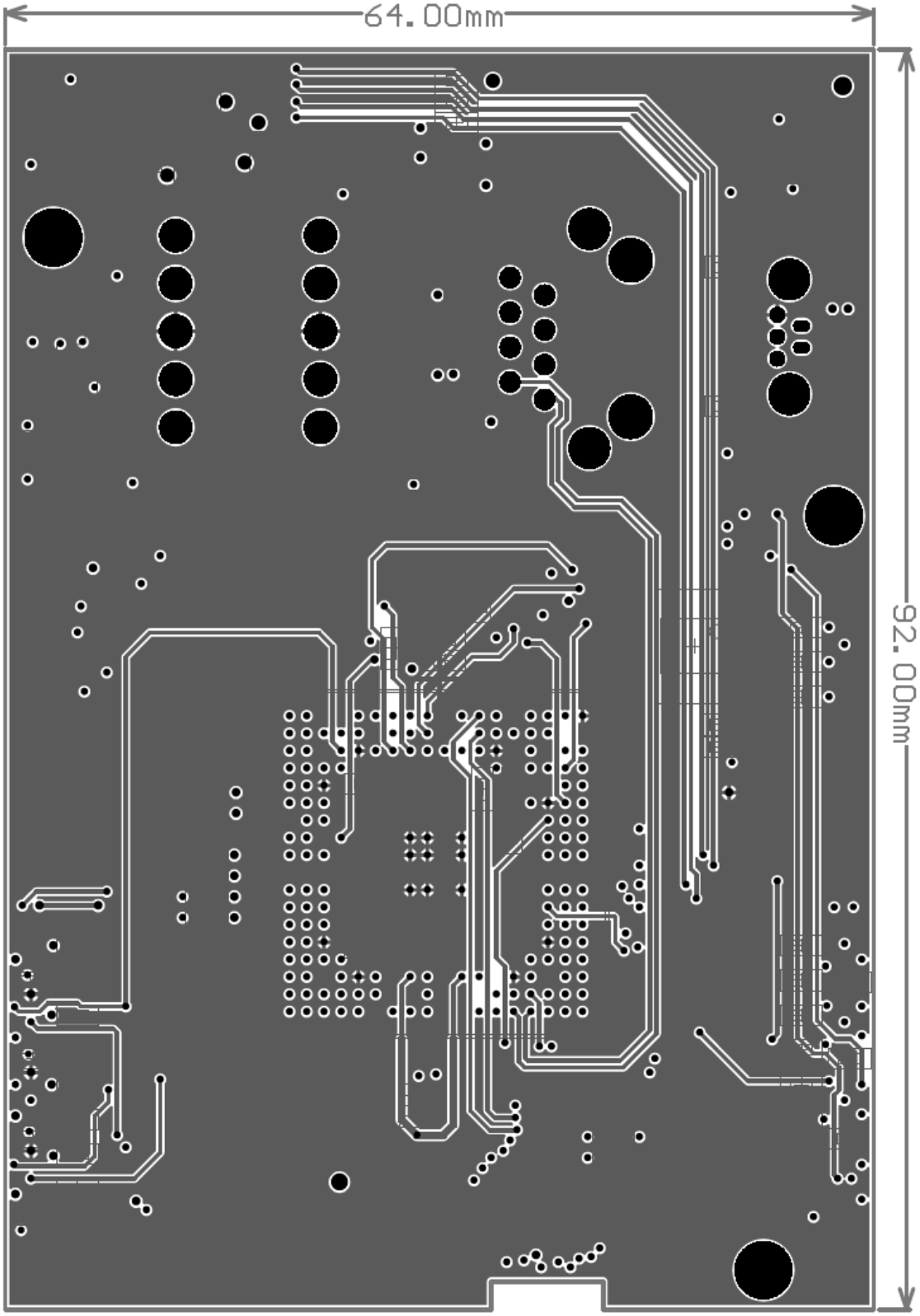
F.3 - *Top Layer* (camada dos componentes) do PCB Superior



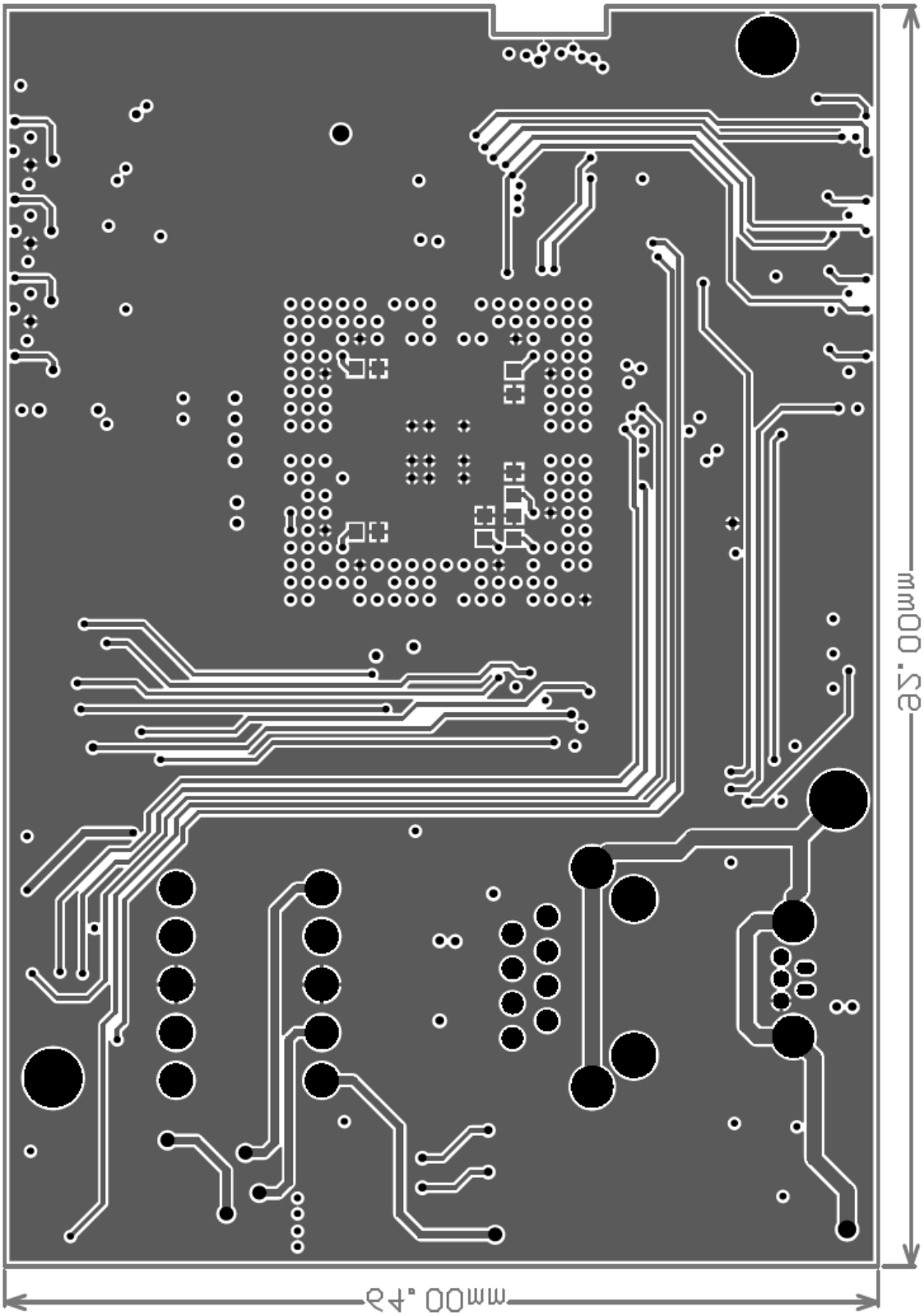
F.4 - *Internal Layer* 1 (camada interna 1) do PCB Superior



F.5 - *Internal Layer* 2 (camada interna 2) do PCB Superior



F.6 - *Bottom Layer* (Lado da soldadura THT) do PCB Superior



F.7 - Relatório de erros referente ao PBC Superior

Design Rules Verification Report

Filename : C:\Dropbox\ESTAGIO ISEC\relatotio\HW\ISEC-CoDeSyS-BECK-CPU\ISEC-Co
DeSyS-BECK-CPU.PcbDoc

Warnings 0
Rule Violations 0

Warnings	
Total	0

Rule Violations	
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Clearance Constraint (Gap=0.2mm) (All),(All)	0
Power Plane Connect Rule(Direct Connect)(Expansion=0.3mm) (Conductor Width=0.254mm) (Air Gap=0.254mm)	0
Width Constraint (Min=0.15mm) (Max=2mm) (Preferred=0.254mm) (All)	0
Height Constraint (Min=0mm) (Max=25.4mm) (Prefered=12.7mm) (Disabled)(All)	0
Hole Size Constraint (Min=0.025mm) (Max=6mm) (All)	0
Hole To Hole Clearance (Gap=0.254mm) (All),(All)	0
Net Antennae (Tolerance=0mm) (All)	0
Total	0